

Adaptação de Mesa XYZ para uma Fresadora CNC

PAULO RICARDO SOARES BARBOSA

Setembro de 2025

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Adaptação de Mesa XYZ para uma Fresadora CNC

Paulo Ricardo Soares Barbosa

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

Setembro, 2025

*Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de
Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em
Engenharia Electrotécnica e de Computadores, Área de Especialização em
Automação e Sistemas.*

Candidato: Paulo Ricardo Soares Barbosa, N.º 1201646,
1201646@isep.ipp.pt

Orientação Científica: Manuel Silva, mss@isep.ipp.pt

Empresa: Gislotica - Mechanical Solutions

Orientador: Humberto Ramos, humberto.ramos@gislotica.pt

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

Setembro, 2025

Agradecimentos

A realização deste trabalho constitui o culminar de um longo percurso académico, no qual intervieram vários apoios, incentivos e contributos, a todos os quais deixo um profundo agradecimento. Correndo o risco de omitir, de forma injusta, algumas dessas contribuições, ficam expressos os seguintes agradecimentos:

Ao orientador científico, Professor Manuel Silva, pela disponibilidade demonstrada desde o primeiro momento e pelos conselhos transmitidos, fundamentais para o desenvolvimento e concretização deste trabalho.

Ao orientador do estágio, Eng.º Humberto Ramos, pelo acompanhamento constante e pela orientação eficaz em todas as etapas do trabalho efetuado.

À empresa Gislótica, liderada pelo Eng.º Rui Fazenda, pela abertura ao diálogo que possibilitou a idealização e realização deste trabalho. A todo o corpo do Departamento de Automação e Eletrotécnica, pelo acolhimento e apoio prestado ao longo do trabalho. Um agradecimento adicional ao Pedro Nunes e ao Manuel Castro, pelo acompanhamento e contributos para a idealização do projeto, e ao Danilo Pinto, engenheiro responsável pela componente mecânica, pelo acompanhamento e dedicação demonstrados. Não esquecer os trabalhadores da fábrica, que acompanharam e auxiliaram no desenvolvimento deste trabalho.

A todo o corpo docente do ISEP que me acompanharam no percurso académico, e em particular aos professores do mestrado, pelo conhecimento transmitido e pela inspiração académica.

Aos colegas de curso, em especial aos que partilharam trabalhos académicos e apoio ao longo destes cinco anos. Um agradecimento particular ao João Moura, Tomás Valente, Gonçalo Pereira, Samuel Costa, Bruno Lima e ao Xavier Barros pela amizade e companheirismo.

A todos os Escuteiros com quem cruzei o caminho que percorri, em especial ao agrupamento 1328 - Foz do Sousa, ao qual pertenço desde a infância, pela motivação constante. Agradeço também a outros elementos de outros agrupamentos, quer ainda ativos ou que já saíram do ativo, com quem colaborei em projetos de núcleo e diversas atividades, pela amizade e incentivo.

Aos embaixadores da MOOVIT em Portugal, em especial ao Raúl, Pedro, Daniel e André, pela amizade e apoio, permitindo o meu foco para este trabalho.

Por fim, mas de forma mais significativa, à minha família, em especial aos meus pais e à minha irmã, pelo apoio incondicional ao longo dos estudos e pela presença

constante, mesmo passando pelo momento mais difícil e desafiante.

*"... if the right spirit is there, it can
knock, the "im" out of the word
'impossible'."*

Robert Baden-Powell

Resumo

Este trabalho apresenta o desenvolvimento de uma solução industrial para uma fresadora *Computer Numerical Control* (CNC), baseada em um antigo trabalho que foi abandonado, com o objetivo de permitir que a máquina efetue furações e recortes em materiais do tipo chapa ou semelhantes, de forma a reduzir o tempo de fabrico, assim como a melhoria da precisão de determinadas peças.

O projeto aborda a integração de *hardware* e *software*, incluindo o uso de motores *stepper*, sensores de fim de curso, *spindle* e sinais luminosos, assim como o estudo de todas as ligações elétricas, protocolos de comunicação e respetiva segurança. Foi ainda estudado o uso do TwinCAT como elemento de lógica da CNC, bem como a exploração do *software* dos drivers dos motores de eixo.

A solução desenvolvida procura alternativas de fresadoras CNC, aproveitando a tecnologia existente em *stock* na empresa beneficiária do trabalho, de forma a que o custo de implementação seja o mais baixo possível. Devido às dimensões da mesa, optou-se pela solução do TwinCAT, que oferece um pacote de funções dedicado ao uso de CNC, onde foram exploradas e melhoradas as soluções fornecidas. No entanto, a implementação apresenta alguns desafios, tais como a complexidade na exploração do pacote de funções e algumas falhas encontradas durante o desenvolvimento. Foi ainda explorado *software Computer-Aided Design* (CAD)/*Computer-Aided manufacturing* (CAM) para validar a execução de peças tridimensionais para um código G, de forma a comprovar o funcionamento destes códigos da máquina.

Para efetuar a validação do sistema, foram realizados diversos testes para comprovar o funcionamento na máquina, permitindo testar os sensores e atuadores, bem como a comunicação entre os protocolos de comunicação e os requisitos de segurança elétrica e mecânica.

Este projeto representa uma solução industrial viável, destinada ao uso interno na empresa, possuindo bases sólidas para melhorias futuras e incorporação de funcionalidades extras.

Palavras-Chave: Fresadora CNC, Mesa XYZ, TwinCAT, Automação Industrial, IPC, CAD/CAM, *Software* de *drivers*, Integração de *hardware* e *software*.

Abstract

This work presents the development of an industrial solution for a *Computer Numerical Control* (CNC) milling machine, based on an abandoned previous project, with the aim of enabling the machine to perform drilling and cutting operations on materials such as sheet metal or similar, in order to reduce manufacturing time and improve the precision of certain parts.

The project addresses the integration of hardware and software, including the use of stepper motors, limit switches, spindle, and indicator lights, as well as the study of all electrical connections, communication protocols, and respective safety measures. The use of TwinCAT as the logic element for the CNC was also studied, along with the exploration of the software for the axis motor drivers.

The developed solution seeks alternatives for CNC milling machines, leveraging existing technology in stock at the beneficiary company, so that the implementation cost is kept as low as possible. Due to the dimensions of the table, the TwinCAT solution was chosen, which offers a dedicated function package for CNC use, where the provided solutions were explored and improved. However, the implementation presents some challenges, such as the complexity of exploring the function package and some issues encountered during development. Software for *Computer-Aided Design* (CAD)/*Computer-Aided manufacturing* (CAM) was also explored to validate the execution of three-dimensional parts for G-code, in order to verify the functionality of these machine codes.

To validate the system, various tests were conducted to confirm the machine's operation, allowing for the testing of sensors and actuators, as well as the communication between communication protocols and the requirements for electrical and mechanical safety.

This project represents a viable industrial solution, intended for internal use within the company, providing a solid foundation for future improvements and the incorporation of additional functionalities.

Keywords: CNC Milling Machine, XYZ Table, TwinCAT, Industrial Automation, IPC, CAD/CAM, Driver Software, Hardware and Software Integration.

Índice

Lista de Figuras	xiii
Lista de Tabelas	xix
Listagens	xxi
Lista de Algoritmos	xxiii
Lista de Acrónimos	xxv
Lista de Símbolos	xxix
1 Introdução	1
1.1 Contextualização	1
1.2 Definição do Problema	2
1.2.1 Objetivos	2
1.2.2 Resultados Esperados	3
1.3 Plano de Trabalho	3
1.4 Organização da Dissertação	3
2 Fundamentação Teórica	7
2.1 <i>Computer Numerical Control</i> (CNC)	7
2.1.1 Contexto Histórico das CNC	8
2.1.2 Fundamentos da Tecnologia NC	9
Componentes Básicos	10
Sistema de Coordenadas	10
Sistema de Controlo de Movimentos	11
Sistema de Coordenadas para Movimentação	13
2.1.3 Fundamentos da Tecnologia CNC	14
Unidade de Controlo	14
<i>Software de CNC</i>	16
2.1.4 Conceitos de Maquinação	16
<i>Feed Rate</i> e Velocidade do <i>Spindle</i>	16
Sistema de Controlo Elétrico e Mecânico	18
Precisão	20

2.1.5	Definições da Máquina	21
	<i>Homing</i>	21
	Jogging	22
	<i>Manual Data Input</i> (MDI)	22
2.1.6	<i>Computer-aided Design/Computer-aided manufacturing</i> . . .	22
2.1.7	Outros Tipos de Aplicações de CNC	23
2.1.8	Vantagens e Limitações da Tecnologia NC	23
2.2	Código G	25
2.2.1	Contexto Histórico sobre Código G	25
2.2.2	Estrutura de um Programa de Instruções	25
2.2.3	Explicação de Códigos Relevantes	26
	Código G	26
	Código M	29
2.3	Hardware	29
2.3.1	<i>Industrial PC</i> (IPC)	29
	Tipos de Instalação de <i>Industrial PC</i>	30
	<i>Industrial PC</i> ou <i>Programmable Logic Controller</i>	30
2.3.2	Motores	31
	Motores de Passo	32
	Motores Servomotores	34
	Comparação entre Motores para a Aplicação	34
2.3.3	<i>Spindle</i>	35
	Tipos de <i>Spindles</i>	36
	Controlo de <i>Spindles</i>	37
	Sistemas de Retenção da Ferramenta	38
	Análise Técnica	39
2.4	Protocolos de Comunicação e Sinais Relevantes	39
2.4.1	<i>Ethernet for Control Automation Technology</i> (EtherCAT) . .	40
	Princípio de Funcionamento	40
2.4.2	CANopen	42
	Funcionamento do CANopen	42
	<i>Process Data Object</i> (PDO)	42
	<i>Service Data Object</i> (SDO)	43
	Conexões	44
2.4.3	Sinais Digitais para Controlo de Posição	44
	Sinal A/B	45
	Sinal CW/CCW	45
	Sinal CLK/DIR	46
2.4.4	Protocolos do Controlador Beckhoff	46
	<i>Automation Device Specification</i> (ADS)	46

Conceito de Dispositivos de <i>Automation Device Specification</i> (ADS)	47
<i>Automation Message Specification</i> (AMS)	48
Arquitetura e Funcionamento do <i>Automation Message Specification</i> (AMS)	49
Endereço AMSNetID	49
Portas <i>Automation Message Specification</i> (AMS)	50
2.5 Conclusão da Fundamentação Teórica	50
3 Trabalhos Relacionados	53
3.1 Trabalhos com Controladores de Baixo Custo	53
3.1.1 Impressora de PCB com o uso de GRBL	54
3.1.2 <i>Computer Numerical Control</i> Controlada por LinuxCNC	55
3.2 Trabalho com <i>Industrial PC</i>	58
3.2.1 Controlo do TwinCAT numa máquina CNC com CANopen sobre EtherCAT	59
3.2.2 Sistema CNC para Funções Inteligentes com base em TwinCAT e .NET <i>Framework</i>	61
3.3 Conclusão dos Trabalhos Relacionados	63
4 Arquitetura do Sistema	67
4.1 Arquitetura Encontrada	67
4.2 Arquitetura Proposta	70
4.3 Hardware	72
4.3.1 Alimentação	72
Chave Seleccionadora de Corrente	72
Interruptor Diferencial Residual	73
Disjuntor Magnetotérmico	73
Fonte de Alimentação	74
Disjuntores Eletrónicos	75
4.3.2 Segurança	75
Relé de Segurança	75
Botão de Emergência	77
4.3.3 Controlo	78
Controlador	78
Acoplador EK1100	79
Terminal de Entradas EL1008	80
Terminal de Saídas EL2008	81
Terminal de Saída por <i>Encoder</i> Incremental EL2522	81
Terminal de Saída Analógica EL4002	83
4.3.4 Motores do Eixo	83

	Motores de Passo	83
	<i>Drivers</i> dos Motores	84
	Caixa Redutora dos motores	85
4.3.5	Sensores e Atuadores	85
	Fins de Curso na Mesa	86
	Coluna Luminosa de Sinalização	86
4.3.6	<i>Spindle</i>	87
	Motor <i>Spindle</i>	87
	Inversor de Frequência	87
4.3.7	Relés	87
4.3.8	Lista de Componentes	88
4.4	Software	88
4.4.1	<i>Festo Configuration Tool</i>	89
4.4.2	TwinCAT	89
4.4.3	Pacote de Funções TF5200 (CNC)	91
	Funcionamento Interno	91
4.4.4	Autodesk Inventor: Funcionalidades CAD e Expansão CAM .	92
4.4.5	Fluxogramas e Máquina de Estados	94
	Controlo dos Motores	94
	Controlo da Coluna Luminosa de Sinalização	95
	Sistema de Limites	95
4.5	Conclusão da Arquitetura Proposta	97
5	Implementação Prática	99
5.1	Configurações Referente ao <i>Hardware</i>	100
5.1.1	Implementação de Controlo	100
	Implementação dos Terminais EL1008 e EL2008	100
	Implementação do Terminal EL2522	101
5.1.2	Implementação de Segurança	102
5.1.3	Resultado Final do <i>Hardware</i>	102
5.2	<i>Drivers</i> dos Motores	102
5.2.1	Configurações Gerais dos <i>Drivers</i>	103
	Configuração Geral	103
	Configuração da Aplicação	105
	Configuração dos Motores e Eixo	105
	Configuração de Controlo	106
	Configuração do <i>Trace Data</i>	106
5.2.2	Conexões do <i>Driver</i>	107
	<i>Input/Output Interface</i>	108
	<i>Encoder</i> e Motor	108

	STO interface	108
	Alimentação do <i>Driver</i>	109
	<i>Master/Slave Interface</i>	109
5.3	Configurações em Programa <i>Point-to-Point</i> (PTP) no TwinCAT . .	109
5.3.1	Configuração dos Terminais	109
	Criação da interface I/O do TwinCAT	110
	Configuração do Terminal EL1008	111
	Configuração do Terminal EL2008	112
	Configuração do Terminal EL2522	112
5.3.2	Configuração dos Eixos	113
5.3.3	Código Efetuado para PTP	114
5.4	Configurações em Programa <i>Computer Numerical Control</i> no TwinCAT	114
5.4.1	Configuração do Eixo	115
	Calibração e Ganho do Eixo	115
	Dinâmica do Eixo	116
	Definições de <i>Lag</i> e de Controlo	116
	Configurações de <i>Homing</i>	117
	Outras Configurações de Eixo	118
5.4.2	Configuração de Alarmes	119
5.4.3	Configuração do <i>Spindle</i>	120
5.4.4	Configurações Diversas	121
5.4.5	Consola HMI	122
5.4.6	Alterações do Código Base	122
	Controlo de <i>Input/Output</i>	125
	Controlo do PLC	129
	Obtenção de Dados do Interpretador	130
	Adaptação de Funções Base	130
5.5	<i>Software CAD/CAM - Inventor</i>	131
5.6	Conclusão da Implementação Prática	133
6	Testes e Resultados	135
6.1	Teste em Ambiente Simulado	135
6.2	Testes e Resultados dos Motores de Eixo e <i>Drivers</i>	137
6.2.1	Motor Separado da Mesa	137
	Ambiente <i>Festo Configuration Tool</i> (FCT)	138
	Controlo pelo TwinCAT	138
6.2.2	Motor Acoplado à Mesa	139
	Calibração em PTP	140
	Teste de Consistência - Curta Distância	140
	Teste de Consistência - Longa Distância	143

6.3	Testes e Resultados usando o Código da CNC	145
6.3.1	Calibração em CNC	145
6.3.2	Ganhos	148
6.3.3	Testes Básico do Processo Automático	149
6.3.4	<i>Homing</i>	149
6.3.5	Interrupção de Emergência	151
6.3.6	Teste dos Limites Físicos	152
6.3.7	Testes de Códigos G Complexos	153
	Código Proveniente de um Site	153
	Código Proveniente da Ferramenta CAD/CAM	156
6.4	Testes por Realizar	162
6.4.1	Teste do <i>Spindle</i> Separado do Conjunto	162
6.4.2	Teste no Eixo Z	162
6.4.3	Implementação do Conjunto Final	163
6.4.4	Teste numa Peça	163
6.5	Conclusão dos Testes e Resultados	163
7	Conclusões	165
7.1	Trabalho Futuro	166
	Referências	168
	Anexo A Código G - Padrões	177
A.1	Código G	177
A.2	Código M	179
	Anexo B Esquema Elétrico	181
B.1	Ligações de Energia	182
B.2	Ligações do Relé de Emergência, Botões e Luzes	183
B.3	Ligações de Controlo	184
B.4	Ligações dos <i>Drivers</i>	185
B.5	Ligações dos Motores de Eixo	186
B.6	Ligações do <i>Spindle</i>	187
	Anexo C Códigos Efetuados	189
C.1	Código PTP	189
C.1.1	Variáveis	189
C.1.2	Código	190
C.2	Variáveis Globais do CNC	192
C.2.1	Global_HMI	192
C.2.2	GVL_Input_Output	193
C.3	PRG_Write_INPUT	194

C.3.1	Variáveis	194
C.3.2	Código	195
C.4	PRG_Light_Tower	195
C.4.1	Variáveis	195
C.4.2	Código	196
C.5	PRG_Control_Light_Mode	197
C.5.1	Variáveis	197
C.5.2	Código	198
C.6	FB_Limit_Switch	200
C.6.1	Variáveis	200
C.6.2	Código	201
C.6.3	Write_Switch	202
C.6.4	Jog_Condition	204
C.7	PRG_ControlFunction	205
C.7.1	Variáveis	205
C.7.2	Código	205
C.8	PRG_LoadDataFromPLC	208
C.8.1	Variáveis	208
C.8.2	Código	208
C.8.3	Recolha_Eixo	209
C.9	FB_HLI3_CNCChannel_Upgrade	210
C.9.1	Variáveis	210
C.9.2	Código Modificado	212
C.9.3	Função Copy_Parameter	218
Anexo D Código G - TwinCAT		219
D.1	Código G	219
D.2	Código M	223

Lista de Figuras

2.1	Primeira máquina NC desenvolvida pelo MIT [6]	9
2.2	Componentes básicos de um sistema NC [7]	10
2.3	Sistema de coordenadas típicas em NC com peças planas [7]	11
2.4	Controlo ponto a ponto [7]	12
2.5	Controlo contínuo [7]	12
2.6	Posicionamento absoluto vs Posicionamento incremental [7]	14
2.7	Arquitetura da <i>Machine Control Unit</i> (MCU) de uma <i>Computer Numerical Control</i> (CNC) [7]	15
2.8	Operações mecânicas comuns [7]	17
2.9	Sistema de movimento linear	19
2.10	Tipos de controlo de movimento	20
2.11	Definição de resolução de controlo, exatidão e repetibilidade [7]	20
2.12	Eixo linear com os dois <i>switches</i> limitadores e o <i>home switch</i> [9]	21
2.13	Movimento de <i>Homing Move-Until-Trigger Scheme</i> [9]	22
2.14	Aplicações de máquinas NC	24
2.15	Exemplo de um programa NC [2]	26
2.16	Exemplo de uma linha de código [15]	27
2.17	Movimentações lineares	27
2.18	Interpolação circular [15]	28
2.19	Exemplos de <i>Industrial PC</i> (IPC)	31
2.20	Motores de passo [22]	32
2.21	Controlo em malha aberta de um motor de passo [22]	33
2.22	Resposta típica de um motor de passo a um conjunto de impulsos [22]	33
2.23	Controlo em malha fechada de um servomotor [22]	34
2.24	Exemplo de um <i>spindle</i> [23]	36
2.25	Exemplo de tipos de <i>spindles</i>	36
2.26	Exemplo de diferentes tipos de retenção da ferramenta	39
2.27	Funcionamento do EtherCAT [27]	41
2.28	Simplificação de arquiteturas com o uso de EtherCAT	41
2.29	Mensagem <i>Process Data Object</i> (PDO) da comunicação CANopen [28]	43
2.30	Mensagem <i>Service Data Object</i> (SDO) da comunicação CANopen [28]	44
2.31	Tipos de conectores e <i>pinout</i> usados no CANopen	45
2.32	Sinal <i>A/B Phase</i> (A/B) [31]	45

2.33	Sinal <i>Clockwise/Counterclockwise</i> (CW/CCW) [32]	46
2.34	Sinal <i>Clock/Direction</i> (CLK/DIR) [32]	46
2.35	Conceito de dispositivos por ADS [19]	48
2.36	Interligação entre dispositivos por ADS [33]	48
2.37	Endereço AMSNetId [33]	49
3.1	Arquitetura da mini-impressora de PCB [37]	54
3.2	Esquemático da mini-impressora de PCB [37]	55
3.3	Impressora de PCB [37]	56
3.4	Modelo CAD da mini CNC [38]	57
3.5	Arquitetura do projeto da mini-CNC [38]	58
3.6	Mini-CNC [38]	59
3.7	CNC proposta para o trabalho do TwinCAT CoE [39]	60
3.8	Arquitetura de controlo e comunicação do trabalho do TwinCAT CoE [39]	60
3.9	Testes efetuados para a execução do projeto do TwinCAT CoE	61
3.10	Arquitetura de sistema aberto das CNC [40]	62
3.11	Troca de dados <i>Numerical Control</i> (NC) entre os lados <i>Human-Machine Interface</i> (HMI) e TwinCAT [40]	63
3.12	Estrutura do <i>hardware</i> no sistema <i>Numerical Control Kernel</i> (NCK) do trabalho TwinCAT .NET [40]	64
3.13	Estrutura do <i>hardware</i> no sistema periférico do trabalho TwinCAT .NET [40]	64
3.14	Teste prático do trabalho TwinCAT .NET [40]	65
4.1	Mesa XY	68
4.2	Estado do quadro antes da adaptação	69
4.3	Arquitetura da proposta anterior [41]	69
4.4	<i>Layout</i> da mesa para a CNC	70
4.5	Arquitetura de controlo proposta para a CNC	71
4.6	Chave seccionadora de corrente Siemens Sentron	72
4.7	Interruptor diferencial residual	73
4.8	Disjuntor magnetotérmico [47]	74
4.9	Fonte de alimentação Siemens SITOP PSU100S [42]	74
4.10	Disjuntor eletrónico MICO	76
4.11	Relé de segurança Pilz PNOZ XV2	76
4.12	Modos que serão usados do relé de segurança PILZ	77
4.13	Botão de emergência	78
4.14	Níveis de plataforma da Beckhoff [19]	79
4.15	IPC CX9020 [19]	80
4.16	Acoplador EK1100 [19]	80

4.17	Terminal EL1008 [19]	81
4.18	Terminal EL2008 [19]	81
4.19	Terminal EL2522 [19]	82
4.20	Modos de saída do EL2522	82
4.21	Terminal EL4002 [19]	83
4.22	Motor <i>stepper</i> EMMS-ST-87-S-SEB-G2 [63]	84
4.23	<i>Driver</i> CMMS-ST-C8-7-G2 [63]	84
4.24	Caixa de engrenagens EMGA-80-P-G5-SST-87 [63]	85
4.25	Fim de curso 3SE5 423-0CD20-1EB1 [42]	86
4.26	Coluna luminosa sinalizadora de máquinas industriais [68]	86
4.27	Motor <i>spindle</i> [70]	88
4.28	Inversor de frequência YL620 [70]	88
4.29	Relé HF41F e respetivo suporte DIN [72]	89
4.30	Página principal do <i>Festo Configuration Tool</i> (FCT)	91
4.31	Execução do CNC e PLC no controlador [19]	92
4.32	Funcionamento do HLI [19]	93
4.33	Autodesk Inventor com extensão <i>Computer-Aided manufacturing</i> (CAM) [73]	94
4.34	Máquina de estado dos motores	95
4.35	Fluxograma da coluna luminosa de sinalização	96
4.36	Fluxograma da coluna luminosa de sinalização - Funções	97
4.37	Fluxograma dos limites	98
5.1	Esquemático dos terminais EL1008 [19]	101
5.2	<i>Pinout</i> dos terminais EL2522 [19]	102
5.3	Implementação do <i>hardware</i>	103
5.4	Parâmetros da configuração	104
5.5	Parâmetro do eixo	104
5.6	Parâmetros de aplicação	105
5.7	Parâmetro do <i>closed loop</i>	106
5.8	Parâmetros de <i>control interface</i>	107
5.9	Parâmetros de <i>trace</i>	107
5.10	Conexão para o motor	109
5.11	Conexão do <i>Master/Slave</i> [63]	110
5.12	Criação de um <i>device</i>	111
5.13	<i>Device</i> configurado	111
5.14	Canais de <i>Input</i>	112
5.15	Configuração dos eixos referente ao canal no terminal EL2522	113
5.16	Realização do <i>homing</i>	117
5.17	Configuração dos sinais <i>homing</i>	118

5.18	Edição de alarmes	120
5.19	Menu principal do HMI	123
5.20	Funções extras do HMI	124
5.21	Fluxograma da implementação do controlo dos fim de cursos	128
5.22	Máquina de estados implementada do controlo do PLC	129
5.23	Algumas variáveis existentes no lado do interpretador	130
5.24	Alguns movimentos existentes da barra CAM [73]	132
5.25	Simulação do código fornecido por um sistema CAM	132
6.1	Resultado do código G efetuado para o teste do código em ambiente simulado	136
6.2	Consola HMI em fase de desenvolvimento	137
6.3	Motor separado da mesa	138
6.4	Barra de operação do FCT	138
6.5	Resultado do movimento via FCT	139
6.6	Teste de consistência do eixo X - curta distância	142
6.7	Teste de consistência do eixo Y - curta distância	143
6.8	Teste de consistência do eixo X - longa distância	145
6.9	Teste de consistência do eixo Y - longa distância	146
6.10	Teste com diferentes ganhos ao eixo Y	148
6.11	Teste do primeiro código G para teste de movimentos	150
6.12	Erros comuns encontrados do decorrer de testes	151
6.13	Correção dos erros encontrados do decorrer dos testes	151
6.14	Teste de <i>homing</i> efetuado no eixo Y	152
6.15	Teste da interrupção de emergência	152
6.16	Teste dos limites do eixo	153
6.17	Texto efetuado no CNC-Apps vista no GCodeViewer	154
6.18	Teste efetuado ao código G "Paulo"	155
6.19	Testes efetuados ao código G "Gislotica" registados no papel	156
6.20	Teste efetuado ao código G "Gislotica" registados ao <i>scope</i>	157
6.21	Quadro HMI numa Vitec	158
6.22	Teste efetuado no código G elaborada por ferramenta CAD/CAM registados no GCodeViewer	159
6.23	Teste efetuado ao <i>spindle</i> em simulação	159
6.24	Resultados obtidos no código G elaborada por ferramenta CAD/CAM	160
6.25	Erros encontrados resultantes de uma possível falha mecânica	161
6.26	Quadro Vitec desenhado após a correção da falha mecânica e várias passagens	162
B.1	Circuito de ligações de energia	182
B.2	Circuito de ligações do relé de emergência, botões e luzes	183

B.3	Circuito de ligações do controlo	184
B.4	Circuito de ligações dos <i>drivers</i>	185
B.5	Circuito de ligações dos motores do eixo	186
B.6	Circuito de ligações do <i>spindle</i>	187

Lista de Tabelas

1.1	Diagrama de Gantt da Dissertação	5
2.1	Velocidade de corte e avanço por dente para diferentes materiais (Fresagem com ferramenta de metal duro)	18
2.2	Endereços do código G [15]	27
4.1	Tabela de quantidades dos componentes existentes no projeto anterior	70
4.2	Modos operacionais ilustrados pelo semáforo	87
4.3	Tabela de quantidades e respectivos componentes a utilizar	90
5.1	Entradas dos EL1008 e saídas do EL2008	101
5.2	<i>Input</i> da função PRG_Light_Tower	125
5.3	Variáveis de estado e modo na máquina e respectivos valores	126
6.1	Dados obtidos da calibração com motor desacoplado	140
6.2	Dados obtidos da calibração com motor acoplado	141
6.3	Distâncias testadas para o teste de consistência em distância curta .	141
6.4	Dados obtidos do teste de consistência - percurso curto	144
6.5	Dados obtidos do teste de consistência - percurso longo	147
6.6	Calibração dos eixos no programa em CNC	147
6.7	Tempos para a execução do teste ao código G "Paulo"	154
6.8	Tempos para a execução do teste no código G "Gislotica"	156
A.1	Código - G Comuns	177
A.2	Código - M Comuns	179
D.1	Comparação da conformidade do código G DIN/ISO com o lido da TwinCAT [19]	219
D.2	Comparação da conformidade do código M DIN/ISO com o lido da TwinCAT [19]	223

Listagens

C.1	Declaração das variáveis no código PTP	189
C.2	Função principal do código PTP	190
C.3	Declaração das variáveis globais Global_HMI	192
C.4	Declaração das variáveis globais GVL_Input_Output	193
C.5	Declaração de variáveis da função PRG_Write_INPUT	195
C.6	Função PRG_Write_INPUT	195
C.7	Declaração de variáveis da função PRG_Light_Tower	196
C.8	Função PRG_Light_Tower	196
C.9	Declaração de variáveis da função PRG_Control_Light_Mode . . .	197
C.10	Função PRG_Control_Light_Mode	198
C.11	Declaração de variáveis da função FB_Limit_Switch	200
C.12	Função FB_Limit_Switch	201
C.13	Função PRG_Write_INPUT	202
C.14	Função Jog_Condition	204
C.15	Declaração de variáveis da função PRG_ControlFunction	205
C.16	Função PRG_ControlFunction	205
C.17	Declaração de variáveis da função PRG_LoadDataFromPLC	208
C.18	Função PRG_LoadDataFromPLC	208
C.19	Função Recolha_Eixo	209
C.20	Declaração de variáveis da função FB_HLI3_CNCChannel_Upgrade	210
C.21	Declaração de variáveis da função FB_HLI3_CNCChannel_Upgrade	213
C.22	Função Copy_Parameter	218

Lista de Algoritmos

5.1	Declaração de variáveis físicas de entrada	111
5.2	Declaração de variáveis físicas de saída	112
5.3	Exemplo de uma função de ação para ativação de alarmes	119

Lista de Acrónimos

3D	<i>Three Dimensional</i>
A/B	<i>A/B Phase</i>
ADS	<i>Automation Device Specification</i>
AMS	<i>Automation Message Specification</i>
ARM	<i>Advanced RISC Machine</i>
BT	<i>Belt Taper</i>
CAD	<i>Computer-Aided Design</i>
CAM	<i>Computer-Aided manufacturing</i>
CAN	<i>Controller Area Network</i>
CD/DVD	<i>Compact Disc/Digital Versatile Disc</i>
CiA	<i>Certified Internal Auditor</i>
CLK/DIR	<i>Clock/Direction</i>
CMM	<i>Coordinate Measuring Machine</i>
CNC	<i>Computer Numerical Control</i>
CoE	<i>CANopen over EtherCAT</i>
CPU	<i>Central Processing Unit</i>
CW/CCW	<i>Clockwise/Counterclockwise</i>
DIN	<i>German Institute for Standardization</i>
DIY	<i>Do It Yourself</i>
EIA	<i>Electronic Industries Alliance</i>
EMCY	<i>Emergency Object</i>
FCT	<i>Festo Configuration Tool</i>

GRBL	<i>G-code Relay Broadcaster Library</i>
GUI	<i>Graphical User Interface</i>
HAL	<i>Hardware Abstraction Layer</i>
HLI	<i>High-Level Interface</i>
HMI	<i>Human-Machine Interface</i>
HSK	<i>Hohlschaftkegel (Hollow Shank Taper)</i>
I/O	<i>Input/Output</i>
IEC	<i>International Electrotechnical Commission</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
IP	<i>Transmission Control Protocol/Internet Protocol</i>
IP	<i>User Datagram Protocol/Internet Protocol</i>
IPC	<i>Industrial PC</i>
ISO	<i>International Organization for Standardization</i>
LED	<i>Light Emitting Diode</i>
MCU	<i>Machine Control Unit</i>
MDI	<i>Manual Data Input</i>
MICO	<i>Murrelektronik Intelligent Current Optimizer</i>
MIT	Massachusetts Institute of Technology
MQTT	<i>Message Queuing Telemetry Transport</i>
NC	<i>Numerical Control</i>
NCK	<i>Numerical Control Kernal</i>
NF	Normalmente Fechado
NMT	<i>Network Management Objects</i>
NO	Normalmente Aberto
PCB	<i>Printed Circuit Board</i>

PDO	<i>Process Data Object</i>
PLC	<i>Programmable Logic Controller</i>
PROFINET	<i>Process Field Network</i>
PTP	<i>Point to Point</i>
PWM	<i>Pulse-Width Modulation</i>
RAM	<i>Random Access Memory</i>
ROM	<i>Read-Only Memory</i>
RPM	<i>Revolutions per Minute</i>
SDO	<i>Service Data Object</i>
SK	<i>Schaftkegel (Belt Taper)</i>
ST	<i>Structured Text</i>
STO	<i>Safe Torque Off</i>
SYNC	<i>Synchronization Object</i>
TF	<i>TwinCAT Functions</i>
TwinCAT	<i>The Windows Control and Automation Technology</i>
VFD	<i>Variable-Frequency Drive</i>

Lista de Símbolos

Símbolo	Descrição	Unidades
D	Denominador	
d	Distância Pretendida	mm
D_f	Diâmetro da Ferramenta	mm
D_p	Diâmetro Primitivo	mm
D_v	Denominador de Velocidade	°/min
$d_{estimada}$	Distância Estimada	mm
d_{obtida}	Distância Obtida	mm
F	<i>Feed Rate</i>	mm/min
f_z	Avanço por Dente	mm/dente
N	Numerador	
N_v	Numerador de Velocidade	incrementos
P_p	Perímetro Primitivo	mm
R	Velocidade de Rotação do <i>Spindle</i>	RPM
V_c	Velocidade de Corte	m/min
x_c	Valor resultante entre Numerador e Denominador quando o eixo está calibrado	inc/0,1 um
x_{nc}	Valor resultante entre Numerador e Denominador quando o eixo não está calibrado	inc/0,1 um
y_{eixo}	Inverso do valor entre Numerador e Denominador do respectivo eixo	0,1 um/inc
z	Número de Dentes da Ferramenta	

Capítulo 1

Introdução

Neste capítulo propõe-se uma breve introdução ao tema desta dissertação, definindo os principais objetivos e os requisitos necessários, seguido de uma apresentação da estrutura do relatório e do seu conteúdo.

1.1 Contextualização

Cada vez mais, encontra-se no nosso quotidiano máquinas que usufruem de mecanismos que operam nos eixos do referencial cartesiano XYZ, implementadas em diferentes áreas, como o corte, furação e fresagem de materiais, manipulação de objetos, adição de material (impressão 3D), entre outras funções.

A fabricação industrial continua em constante evolução, tendo sido revolucionada nas últimas décadas, devido ao surgimento de novas tecnologias nas máquinas, principalmente com o aparecimento das *Computer Numerical Control* (CNC) e os seus aperfeiçoamentos, com foco principal na fresagem e no torneamento, que utilizam o controlo cartesiano [1]. Muitas vezes, o objetivo não é a automação total, mas a busca por uma solução tecnicamente apropriada, rentável e económica, para além das vantagens que uma CNC oferece [2].

Segundo um estudo de mercado realizado pela Precedence [3], prevê-se que, em 2034, este mercado atinja aproximadamente 162,7 biliões de euros, em comparação com os 95 biliões de euros registados em 2023. Este aumento está relacionado com a automação das linhas de produção, a necessidade de precisão nos processos de produção e a necessidade de produtos personalizados.

Relativamente à empresa que irá beneficiar dos resultados desta dissertação, a Gislotica é uma empresa que projeta e constrói soluções mecânicas, com um grande foco na indústria de pneus. A máquina resultante desta dissertação visa automatizar o processo de furação de chapas finas para quadros ou semelhantes, com os benefícios de uma CNC.

1.2 Definição do Problema

O problema proposto para esta dissertação consiste em adaptar uma mesa XYZ, desenvolvida a partir de um projeto interno da empresa, que já foi tema de um projeto de licenciatura, mas para funções diferentes, tendo sido somente realizada em simulação.

O estado atual da mesa é que nunca funcionou com os motores nos eixos X e Y montados da mesa. Relativamente ao último trabalho na estrutura em questão, alguns componentes serão substituídos, como o controlador e a ferramenta, além da adição do motor no eixo Z, e será introduzido código específico para a CNC, que será abordado em profundidade no Capítulo 4.

A mesa em questão deve ser capaz de se deslocar entre pontos interpolados após a leitura do código G, além de controlar a ferramenta, dispõe de uma HMI para o utilizador. É importante mencionar que a geração do código G será feita por meio de uma estrutura *Computer-Aided Design* (CAD)/CAM, que será introduzida na máquina, mas sem ter grande foco nesta dissertação.

1.2.1 Objetivos

Para alcançar o desafio proposto nesta dissertação, é necessário cumprir os seguintes objetivos:

- introdução e pesquisa de código para implementar uma CNC, que interprete código G e execute as instruções correspondentes;
- interpretar o quadro elétrico existente na mesa XYZ e fazer a adaptação para novos componentes e retiradas de outros;
- controlo dos movimentos de cada eixo, manualmente;
- controlo da ferramenta a ser manuseado, principalmente as rotações da mesma;
- controlo da mesa a partir de uma HMI;
- criação de um manual de utilização da máquina.

1.2.2 Resultados Esperados

Relativamente aos resultados esperados desta dissertação, pretende-se obter os seguintes resultados:

- execução da furação/fresagem de uma chapa ou outro tipo de material, a partir de um código G fornecido pelo utilizador;
- HMI didático e de fácil uso, para a máquina ser intuitiva.

1.3 Plano de Trabalho

O plano de trabalho para o desenvolvimento desta dissertação inclui várias fases importantes. Os elementos principais do plano de trabalho estão apresentados na Tabela 1.1.

1.4 Organização da Dissertação

A organização do relatório segue uma estrutura lógica que facilita a compreensão dos conteúdos abordados. Inicialmente, o Capítulo 1 apresenta a introdução, contextualiza o projeto a ser desenvolvido na dissertação, descreve o problema em estudo, define os objetivos específicos e os resultados esperados, expõe a calendarização e apresenta a estrutura da dissertação.

A seguir, o Capítulo 2 é dedicado à fundamentação teórica. Neste capítulo, serão abordados de forma teórica vários conteúdos que foram necessários estudar para a realização da dissertação. Uma parte significativa será dedicada à abordagem do que é uma CNC, os métodos de funcionamento e fundamentos da tecnologia das mesmas e das suas antecessoras NC, além de conceitos necessários para trabalhar com a CNC, vantagens e desvantagens do seu uso, entre outros conceitos relevantes. Numa fase mais avançada, será abordado o código G, a sua estrutura e alguns conceitos relevantes. Posteriormente, será discutido o *hardware* em estudo, a nível de *Industrial PC* (IPC), motores de movimento de eixos, o *spindle* e outros componentes de *hardware* que sejam relevantes. Para finalizar este capítulo, serão estudados protocolos que possam ser relevantes para este trabalho, quer para implementação, quer como opções possíveis, como o *Automation Device Specification* (ADS), *ethercat*, *openCAN* ou os sinais digitais.

Relativamente ao Capítulo 3, serão abordados de uma forma breve vários projetos semelhantes, tanto a nível doméstico, com o uso de microcontroladores como o Arduino, ESP32 ou outros equiparados, como em implementações industriais, com o uso de IPC ou outras soluções.

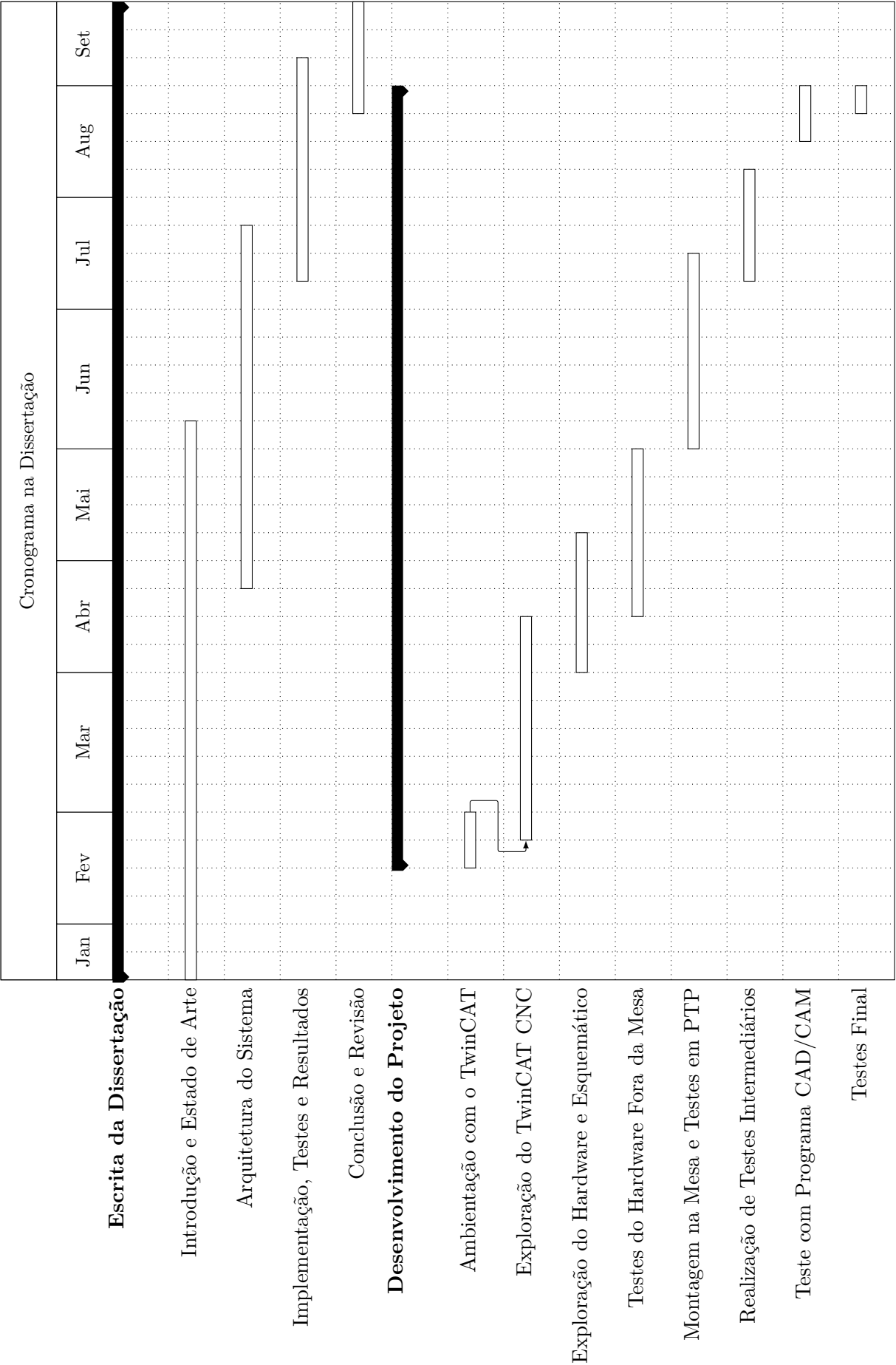
No Capítulo 4, será apresentada a arquitetura proposta para este trabalho. Inicialmente, será descrito o estado em que se encontrava o projeto no início do trabalho, onde serão referidos os pontos a serem adaptados. Após esta explicação, será apresentada uma proposta da arquitetura final, acompanhada de uma explicação detalhada, incluindo alguns aspectos sobre a escolha do *hardware* e uma descrição do *software* utilizado, destacando as suas funcionalidades e a sua integração com o sistema.

No Capítulo 5, será abordada a implementação prática do projeto, detalhando todos os passos dados para a realização da proposta apresentada no capítulo anterior. Aqui, serão descritos os processos de montagem e configuração do sistema, bem como a integração dos diferentes componentes.

No Capítulo 6, serão discutidos os testes realizados e os resultados obtidos. Serão mencionados os diferentes tipos de testes efetuados para avaliar a implementação. A análise dos resultados permitirá verificar se o trabalho está conforme o idealizado e se a execução atende às expectativas.

Por fim, no Capítulo 7, será feita uma conclusão do projeto, onde será avaliado o trabalho realizado, destacando os principais resultados alcançados e as contribuições da dissertação para a área em questão. Serão mencionadas algumas limitações detetadas ao longo do desenvolvimento e discutidas possíveis melhorias que possam ser implementadas em futuras versões do projeto. Este capítulo servirá também para refletir sobre as aprendizagens adquiridas durante o processo e as implicações práticas do trabalho realizado.

Tabela 1.1: Diagrama de Gantt da Dissertação



Capítulo 2

Fundamentação Teórica

Este capítulo procura explorar o conteúdo teórico de diversos temas associados a esta dissertação. Numa primeira abordagem, será discutido o conceito de CNC, a história por detrás destas máquinas, o seu funcionamento, a interpolação de movimentos e outros parâmetros relevantes, com um foco especial nas CNC fresadoras. Em seguida, será abordado o código G, a sua utilização e alguns dos códigos mais relevantes. Numa fase mais avançada, serão analisadas algumas tecnologias, tanto a nível de *hardware* como de comunicação, que podem ser aplicadas na dissertação, incluindo uma comparação entre diferentes tecnologias.

2.1 *Computer Numerical Control* (CNC)

O *Computer Numerical Control* (CNC) é uma tecnologia que permite a automação de máquinas-ferramenta por meio de programas baseados em números e códigos, designados por *G-code* ou código G. Antes das máquinas CNC, existiam as máquinas *Numerical Control* (NC), que utilizavam fitas perfuradas ou outras tecnologias para armazenar comandos, permitindo a produção automatizada, mas com menor flexibilidade.

Com o avanço dos computadores, as CNC surgiram como uma evolução das NC, trazendo maior precisão, repetibilidade e facilidade na programação, revolucionando a manufatura industrial. As CNC conseguem executar uma ampla gama de operações, desde o corte e furação até a fresagem e gravação, tornando-se essenciais em diversos setores, como a indústria automóvel, aeroespacial e de moldes.

A principal vantagem das CNC é a sua capacidade de realizar operações complexas com alta precisão e consistência, reduzindo o erro humano e aumentando a eficiência da produção. Além disso, a programação das CNC permite a fácil replicação de peças, o que é fundamental para a produção em massa [4].

2.1.1 Contexto Histórico das CNC

O princípio da produção em massa, desenvolvido por Eli Whitney [4], transferiu muitas operações e funções originalmente realizadas por artesãos qualificados para as máquinas-ferramenta. Com as melhorias e, conseqüentemente, maior precisão dessas ferramentas na época, elas foram rapidamente adotadas pela indústria para a produção em massa.

Este desenvolvimento ocorreu na segunda metade do século XIX, quando foram criadas máquinas-ferramenta para operações básicas como corte de metal, torneamento, perfuração, fresagem e retificação [4].

A fabricação industrial, até meados do século XX, começou a enfrentar dificuldades na produção de máquinas e respectivas peças. Com as ferramentas disponíveis até então, não era possível criar peças complexas e de diferentes formas com elevada precisão, resultando numa taxa de produção baixa, custos elevados e altas taxas de rejeição. Os fabricantes não conseguiam cumprir os prazos de entrega devido ao aumento da procura [5].

A Força Aérea Americana identificou que, em projetos e formas de peças complexas para aeronaves, surgiam problemas para os fabricantes, que não conseguiam acompanhar os cronogramas de produção planejados. Em resposta a essa situação, John Parsons começou a experimentar a ideia de construir uma máquina que gerasse uma "curva de passagem" utilizando dados numéricos para controlar o movimento da ferramenta numa máquina [4, 6].

Em 1952, o Laboratório de Servomecanismos do Massachusetts Institute of Technology (MIT), que Parsons subcontratou após aprovação da Força Aérea para o desenvolvimento da tecnologia NC, conseguiu demonstrar com sucesso um fresamento vertical numa máquina, conforme ilustrado na Figura 2.1. Esta máquina fabricava peças por meio de movimentos simultâneos dos eixos de corte. Com este desenvolvimento, quase todos os fabricantes de máquinas-ferramenta começaram a produzir máquinas NC [4, 6].

Esta NC era operada por meio de um computador com um leitor de fita. As instruções a serem executadas estão perfuradas numa fita, permitindo à máquina realizar as tarefas especificadas. Com esta inovação, foi possível produzir peças típicas e com formas complexas de forma mais rápida, o que resultou numa melhoria na taxa de produção e maior precisão [4, 5].

Por volta da década de 1970, a introdução de controlos baseados em *software* permitiu substituir o controlo de programas por *hardware*, ou seja, fitas (perfuradas

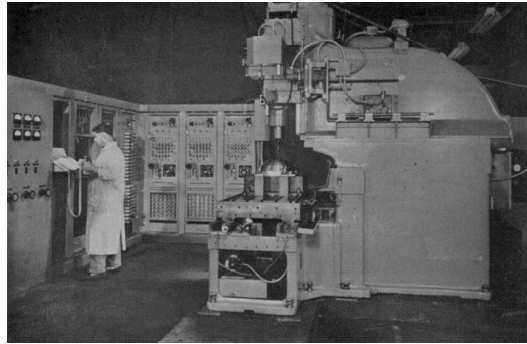


Figura 2.1: Primeira máquina NC desenvolvida pelo MIT [6]

ou magnéticas) foram substituídas por lógica de computador complexa, que possuía maior capacidade e podia ser programada para diversas funções simultaneamente. A esta nova máquina foi dado o nome de *Computer Numerical Control (CNC)*. Com esta evolução, possibilitou-se fazer revisão, modificar ou atualizar programas ou partes de programas na CNC a qualquer momento através de um computador, algo que a tecnologia antecessora não permitia. Estas máquinas tornaram-se mais fáceis de usar, pois incluíam telas de menu interativo, gráficos avançados e facilidade de programação.

Desde então, as CNC tornaram-se o método de produção dominante na maioria das indústrias de fabricação, incluindo automóvel, aviação, defesa, petróleo e gás, medicina, indústria eletrônica e indústria óptica, entre outras. As CNC evoluem em consonância com o avanço das tecnologias e outras questões, como a proteção ambiental e a eficiência energética. Os fabricantes de máquinas-ferramenta e controladores têm introduzido e pensado em formas de otimizar esses processos para atender aos requisitos de eficiência energética [4, 5].

Na Feira Industrial de *Hannover* de 2011, foi apresentado o conceito da Indústria 4.0. Esta revolução representa um aumento na personalização, mas, ao mesmo tempo, na produção de bens em massa. Para a indústria de máquinas-ferramenta, isso significa que, desde a idealização do produto até à sua fabricação, o processo passa por uma implementação mais consistente de uma cadeia de processos contínua, com a utilização da Internet e de ferramentas CAD/CAM nas quais as CNC desempenham um papel fundamental [2].

2.1.2 Fundamentos da Tecnologia NC

Antes de se abordar a tecnologia CNC, é fundamental entender as bases do sistema NC. Como o estudo se concentra numa CNC fresadora, a discussão será direcionada para esses equipamentos.

Componentes Básicos

Um sistema NC é composto por três componentes básicos: o programa de instruções, MCU e o equipamento de processamento da peça. A relação entre os três componentes está ilustrada na Figura 2.2.

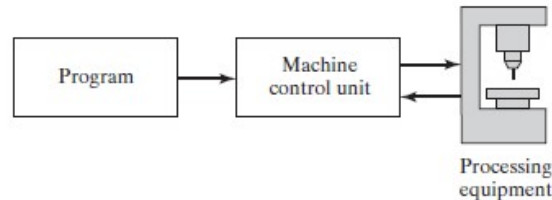


Figura 2.2: Componentes básicos de um sistema NC [7]

O programa de instruções é o conjunto de comandos que detalha o passo a passo das ações a serem aplicadas no processamento da peça. Nestas aplicações, os comandos individuais referem-se às posições de uma ferramenta relativamente à mesa de trabalho onde se encontra a peça a ser trabalhada. Instruções adicionais, como a velocidade da ferramenta, a taxa de avanço ou *feed rate*, a seleção da ferramenta e outras funções, também são incluídas. Este tema será explorado com mais detalhe na Secção 2.2.

A MCU é um microcontrolador e *hardware* de controlo que armazena o programa de instruções e executa as operações, convertendo cada comando em ações mecânicas do equipamento de processamento. O *hardware* incluído abrange componentes que interagem com os atuadores e recebem os elementos de *feedback* de controlo. O *software* que reside na MCU inclui o sistema de controlo, algoritmos de cálculo e um sistema de tradução que converte um programa NC num formato utilizável.

O terceiro componente básico de um sistema NC é o equipamento de processamento, onde se realiza o trabalho produtivo. É neste equipamento que se executam os passos enviados pelo programa, transformando a peça inicial na peça final. A operação destes componentes é dirigida pela MCU e conduzida pelas instruções contidas no programa. Este componente inclui a mesa de trabalho, motores e os controlos necessários para acioná-los [7].

Sistema de Coordenadas

Para programar um sistema NC, é necessário definir um sistema de eixos padrão através do qual a posição da cabeça da ferramenta é determinada. Existem dois sistemas de eixos utilizados em NC, ambos baseados no sistema cartesiano: um para peças planas e outro para peças de revolução. Para este estudo, focar-nos-emos somente no primeiro sistema cartesiano.

O sistema de eixos para peças planas consiste em três eixos lineares (x , y e z) num sistema cartesiano, além de três eixos de rotação (a , b e c), conforme demonstrado na

Figura 2.3. Geralmente, os eixos x e y são utilizados para posicionar ou trabalhar a ferramenta sobre a peça, enquanto o eixo z controla a posição da ferramenta relativamente à peça.

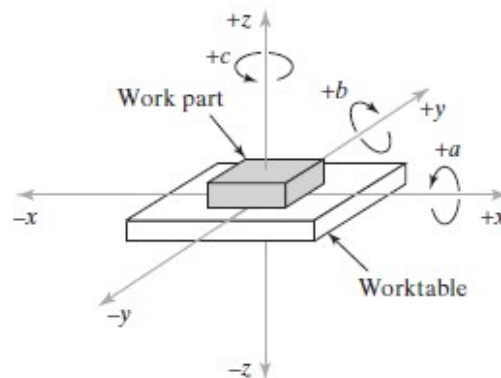


Figura 2.3: Sistema de coordenadas típicas em NC com peças planas [7]

A decisão sobre onde deve ser localizada a origem do sistema de eixos de coordenadas é da responsabilidade do programador. Esta escolha baseia-se na conveniência da programação. A origem pode estar situada num canto da peça ou no centro de uma peça simétrica. Este ponto zero é comunicado ao operador da máquina-ferramenta. No início do trabalho, o operador deve deslocar a ferramenta manualmente até um ponto-alvo na mesa de trabalho, onde a ferramenta pode ser posicionada de forma fácil e precisa. O ponto de destino é previamente referenciado pelo eixo de coordenadas definido pelo programador. Alternativamente, a origem pode ser estabelecida pelo próprio controlador através de vários mecanismos [7].

Sistema de Controlo de Movimentos

Durante os processos de NC, é necessária a execução constante de movimentos da ferramenta. Alguns movimentos são utilizados em zonas que não interferem diretamente na peça de trabalho, como a posição para uma perfuração, enquanto outros são executados na peça, como a fresagem. Neste último, pode ser necessária uma trajetória em linha reta ou um trajeto curvilíneo. Para efetuar estes movimentos, é utilizado um sistema de controlo de movimento. Alguns sistemas de controlo para movimentos são os seguintes [2, 7]:

- Controlo ponto a ponto: estes sistemas, também designados por sistemas de posicionamento, efetuam o movimento da mesa de trabalho e/ou da ferramenta para uma posição programada, sem considerar o caminho percorrido para chegar à localização pretendida. Este tipo de deslocações é utilizado em trabalhos cujas deslocações são efetuadas em espaço livre e livre de contacto com a peça,

ou em perfuração, onde se posiciona a ferramenta no local da operação, por exemplo furação. Após a conclusão do processo, a ferramenta avança para uma nova posição para realizar um novo processo, conforme visualizado na Figura 2.4.

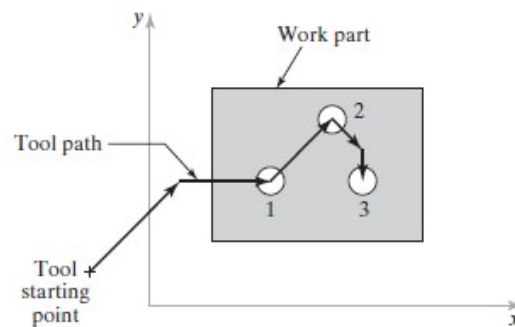


Figura 2.4: Controle ponto a ponto [7]

- Controle contínuo: este sistema de controle permite o controle contínuo de dois ou mais eixos, possibilitando controlar a trajetória da ferramenta enquanto esta executa um trabalho. Isso permite a criação de superfícies angulares, curvas bidimensionais ou contornos tridimensionais na peça de trabalho, como visualizado na Figura 2.5. Este modo é utilizado em operações de fresagem ou torneamento.

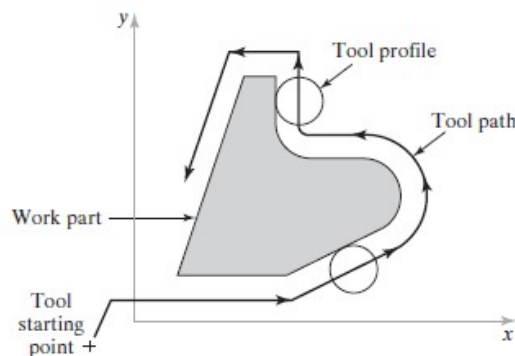


Figura 2.5: Controle contínuo [7]

Para que um controle contínuo funcione, são necessários diferentes métodos de interpolação. Um sistema NC deve criar trajetórias com formatos de retas, arcos e outras formas não lineares. A grande maioria das formas é definida matematicamente por fórmulas geométricas ou, em casos específicos, por aproximação.

Para gerar uma linha reta ou uma linha curva, sem este método, o programador teria de colocar vários segmentos de reta para efetuar o contorno final, tornando um programa complexo e sujeito a erros. Por isso, foram desenvolvidas rotinas de

interpolação que calculam os pontos intermédios a serem seguidos pela ferramenta, seguindo um determinado caminho definido matematicamente.

Existem vários métodos de interpolação para auxiliar na geração de uma trajetória, nomeadamente [7]:

- Interpolação Linear: este método é utilizado para efetuar uma trajetória linear em dois ou três eixos. O programador especifica o ponto inicial e o ponto final da linha reta, bem como a velocidade de avanço (*feed*) a utilizar. O interpolador calcula o *feed rate* para cada um dos eixos para atingir a taxa de avanço especificada;
- Interpolação Circular: este método permite a programação de um arco em círculo, especificando os seguintes parâmetros: coordenadas do ponto inicial, coordenadas do ponto final, centro ou raio do arco e a direção da fresagem. O caminho da ferramenta é efetuado numa série de pequenos segmentos de linha reta, calculados pelo módulo de interpolação, onde a ferramenta é direcionada para se deslocar ao longo destes segmentos para efetuar a trajetória circular. Existe uma limitação desta funcionalidade, pois só pode ser efetuada num plano;
- Interpolação Helicoidal: este método combina o esquema de interpolação circular para dois eixos com o movimento linear de um terceiro eixo. Isto permite a definição de uma trajetória helicoidal no espaço tridimensional;
- Interpolação Parabólica e Cúbica: estas rotinas fornecem aproximações de curvas livres utilizando equações de ordem superior. Estas interpolações requerem uma potência computacional considerável.

Com estes procedimentos, permite-se ao programador introduzir a trajetória com poucos parâmetros de entrada. O módulo de interpolação é executado pela MCU, que realiza os cálculos e dirige a ferramenta ao longo do caminho pretendido. Os interpoladores lineares e circulares são frequentes, enquanto a interpolação helicoidal é uma opção comum. As restantes interpolações não são muito frequentes, mas são utilizadas em ambientes que requerem contornos complexos [7].

Sistema de Coordenadas para Movimentação

Um aspeto que requer atenção no controlo de movimentos é a questão de saber se os pontos são definidos relativamente à origem do sistema de coordenadas (posicionamento absoluto) ou relativamente à localização anterior da ferramenta (posicionamento incremental).

No posicionamento absoluto, as posições da ferramenta são sempre definidas relativamente à origem do sistema de eixos. Por outro lado, no posicionamento

incremental, a posição seguinte da ferramenta é definida relativamente à posição atual.

Na Figura 2.6 são observados dois pontos: um inicial e um final. Para movimentar do ponto inicial para o final, na forma absoluta, são introduzidas as coordenadas desse ponto (40,50). Na forma incremental, é introduzido o valor incremental, ou seja, a diferença entre os pontos, (20,30) para alcançar o ponto final [7].

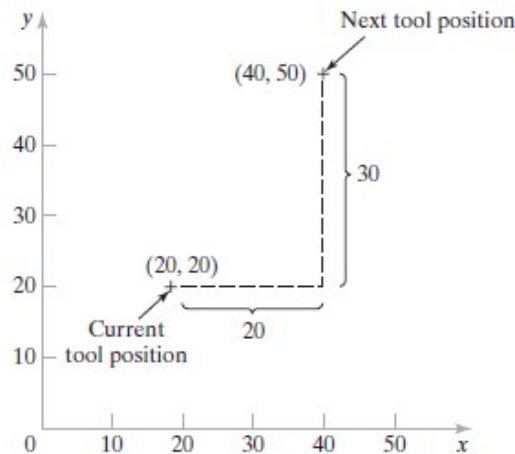


Figura 2.6: Posicionamento absoluto vs Posicionamento incremental [7]

2.1.3 Fundamentos da Tecnologia CNC

Desde a introdução da primeira NC, registaram-se avanços significativos na tecnologia que permitiram a redução do tamanho físico e do custo da unidade central de controlo, enquanto a capacidade computacional aumentou consideravelmente. Com estes avanços, foram introduzidos computadores nas NC, reformulando a arquitetura.

A evolução da NC para a CNC trouxe melhorias significativas, incluindo uma maior capacidade de armazenamento e edição de programas diretamente na máquina, o uso de ciclos fixos e sub-rotinas programáveis, controlo adaptativo para otimização do desempenho, interpolações avançadas, facilidades de posicionamento para configuração, cálculos de aceleração e desaceleração que suavizam os movimentos, bem como a integração com sistemas de comunicação e diagnósticos automatizados para manutenção e prevenção de falhas [7].

Unidade de Controlo

A *Machine Control Unit* (MCU) foi o que possibilitou a evolução para a CNC. A configuração geral da MCU num sistema CNC é ilustrada na Figura 2.7, sendo constituída por: unidade central de processamento, memória, interface *Input/Output*

(I/O), controlo dos eixos e da velocidade da ferramenta, e controlo de sequência para outras funções.

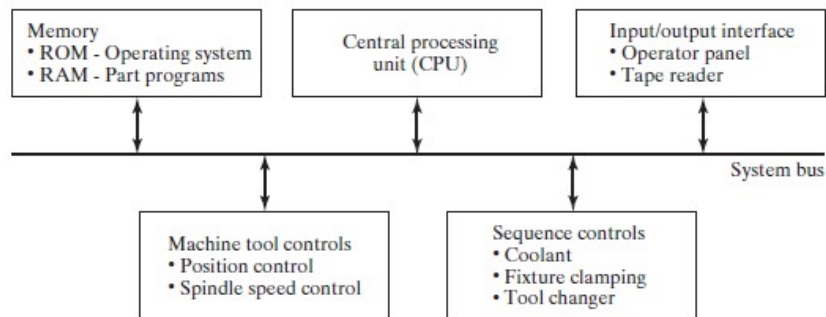


Figura 2.7: Arquitetura da MCU de uma CNC [7]

A unidade central de processamento (*Central Processing Unit (CPU)*) é o cérebro da MCU, gerindo os restantes componentes ligados à MCU com base no *software* existente nas memórias. É neste componente que são lidos os comandos e dados da memória, gerando os sinais para ativar outros componentes da MCU, onde são efetuados os cálculos exigidos, além de contar com uma memória para armazenamento temporário, muito semelhante aos computadores atuais.

Relativamente às memórias, estas são destinadas ao armazenamento do *software* da CNC. Elas incluem memórias *Random Access Memory (RAM)* e *Read-Only Memory (ROM)*. Na ROM, são guardados o *software* do sistema operativo e os programas da interface, enquanto a RAM armazena os programas de instruções.

A interface I/O permite a comunicação entre vários componentes do sistema CNC, outros sistemas informáticos e o operador da máquina, transmitindo e recebendo dados e sinais de e para dispositivos externos ou dentro da MCU. O painel de controlo do operador é a interface comum deste dispositivo, por ser aqui que o operador controla a máquina, introduzindo o programa, modo de funcionamento, velocidades e avanços, além de outras funções, e onde recebe o estado atual da máquina.

Os controlos dos eixos e da velocidade do *spindle* são componentes de *hardware* que controlam a posição e a velocidade (*feed rate*) de cada eixo da máquina, bem como a velocidade de rotação do *spindle*. Os sinais de controlo gerados pela MCU devem ser convertidos para uma forma e nível de potência adequados aos sistemas de controlo de posição específicos utilizados para acionar os eixos da máquina. Os sistemas de posicionamento podem ser classificados como circuito aberto ou circuito fechado, cada um com as suas características relativamente ao *hardware* utilizado, que será explorado na Subsecção 2.1.4. Para o *spindle*, os componentes para o controlo de velocidade consistem geralmente num circuito de controlo de acionamento

e numa interface de sensor de *feedback*. A grande maioria dos motores vem com *drivers* que controlam o respetivo motor. A comunicação entre a MCU e os *drivers* é efetuada por protocolos *standart* de comunicação, sendo os mais utilizados o *Process Field Network* (PROFINET), SERCOS III e EtherCAT, entre outros.

Por fim, existe ainda um controlo de sequência para outras funções da máquina, geralmente para funções auxiliares. Existem vários exemplos, como o controlo de fluido de corte, fixação de dispositivos, avisos de emergência e comunicações para carga e descarga de um robô [2, 7].

Software de CNC

Os sistemas CNC podem utilizar três tipos de *software*: (1) *software* do sistema operativo, (2) *software* de interface com a máquina e (3) *software* de aplicação.

O *software* do sistema operativo, instalado pelo fabricante na memória ROM da MCU, interpreta os programas de peças NC e gera os sinais de controlo correspondentes. Este *software* inclui um editor para entrada e gestão de arquivos, um programa de controlo que descodifica instruções e gera sinais de coordenadas para os eixos da máquina, e um programa executivo que gere a execução do *software* CNC e as operações de entrada e saída, incluindo rotinas de diagnóstico.

O *software* de interface com a máquina possibilita a comunicação entre a CPU e a máquina, controlando as funções auxiliares do CNC. Muitas vezes, essas funções são implementadas por meio de um *Programmable Logic Controller* (PLC), e o *software* é escrito em diagrama *ladder* ou outro formato semelhante.

Por fim, o *software* de aplicação consiste nos programas NC desenvolvidos para operações de maquinação ou outras aplicações na fábrica do utilizador [7].

2.1.4 Conceitos de Maquinação

Nesta subsecção, serão abordados alguns conceitos importantes para o controlo de uma CNC. Focar-se-á no *feed rate* e na velocidade do *spindle*. Em seguida, serão discutidos os diferentes sistemas de controlo, tanto elétricos quanto mecânicos, assim como as vantagens e desvantagens de cada um. Por fim, serão abordados alguns tópicos relativos à precisão.

Feed Rate e Velocidade do Spindle

A maquinação de peças por máquinas-ferramenta é um dos processos industriais mais utilizados, com destaque para as operações de torneamento e fresagem. Este tipo de processo permite a remoção controlada de material excedente, de forma a produzir uma peça com as dimensões e formas desejadas, a partir dos movimentos programados da ferramenta ou da peça.

As operações mais comuns incluem torneamento, perfuração, fresagem e retificação, como ilustrado na Figura 2.8.

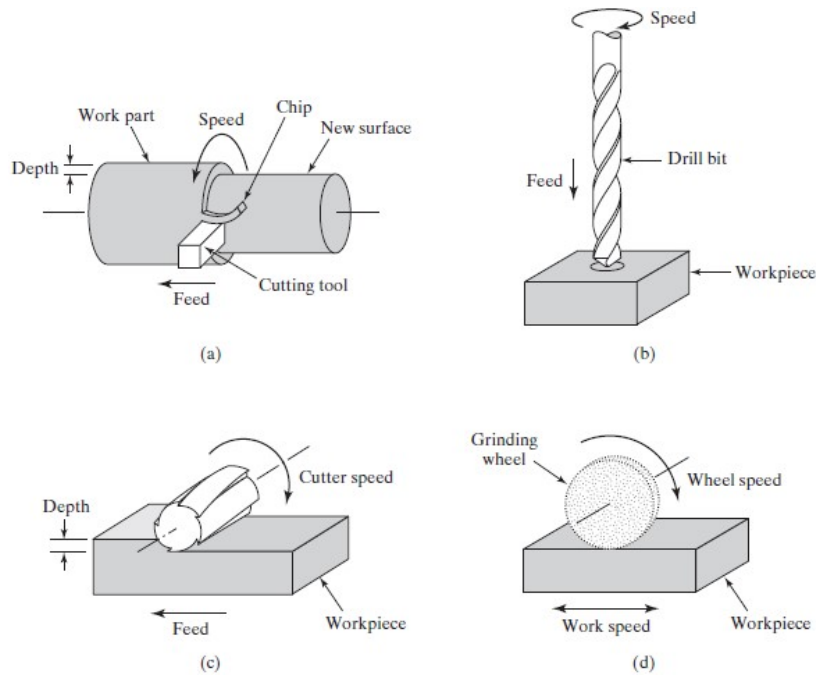


Figura 2.8: Operações mecânicas comuns [7]: a) Torneamento b) Furação c) Fresamento d) Retificação (Desbastamento ou Polimento)

Cada operação possui características próprias, incluindo a velocidade de rotação do *spindle*, o *feed rate*, a profundidade de corte e o tipo de ferramenta.

O *feed rate* representa a velocidade de avanço da ferramenta relativamente à peça. Este valor pode ser definido diretamente nos eixos ou calculado com base em parâmetros como o avanço por dente, o número de dentes da ferramenta e a rotação do *spindle*. A velocidade do *spindle* (N) refere-se à rotação da ferramenta (ou da peça, no caso de tornos), expressa em *Revolutions per Minute* (RPM) [7].

A definição correta destes parâmetros depende de diversos fatores, nomeadamente:

- tipo de material da peça a ser trabalhada;
- material e geometria da ferramenta;
- tipo de operação (desbaste ou acabamento);
- capacidade da máquina.

Para o cálculo da rotação do (*spindle*) e do avanço (*feed rate*), utilizam-se as seguintes expressões:

$$N = \frac{v_c \times 1000}{\pi \times D_f} \quad (2.1)$$

$$F = f_z \times z \times R \quad (2.2)$$

Onde R é a rotação do *spindle* em *Revolutions per Minute* (RPM), v_c é a velocidade de corte em m/min, D_f é o diâmetro da ferramenta em mm, F é o avanço por minuto ou *feed rate* em mm/min, f_z é o avanço por dente em (mm/dente) e z é o número de dentes da ferramenta.

Com base nas tabelas do *Machinery's Handbook* e nas recomendações de fabricantes de ferramentas, é possível encontrar valores típicos para diferentes materiais, conforme apresentado na Tabela 2.1 [7, 8].

Tabela 2.1: Velocidade de corte e avanço por dente para diferentes materiais (Fresagem com ferramenta de metal duro)

Material	v_c (mm/min)	f_z (mm/dente)	Notas
Alumínio (6061)	300 – 800	0,10 – 0,25	Alta velocidade permitida, fácil maquinagem
Aço carbono (1020)	120 – 180	0,06 – 0,12	Maquinagem comum, cuidado com rebarbas
Aço temperado (HRC 45+)	80 – 120	0,03 – 0,08	Exige ferramenta resistente, baixo avanço
Aço inoxidável (304/316)	80 – 150	0,04 – 0,10	Tendência a encruamento, usar refrigeração
Ferro fundido cinzento	150 – 250	0,08 – 0,16	Boa rigidez, maquinando geralmente a seco
Latão	200 – 400	0,10 – 0,20	Excelente maquinabilidade
Titânio (Ti6Al4V)	60 – 100	0,03 – 0,08	Alta dureza e baixa condutividade térmica

Sistema de Controlo Elétrico e Mecânico

Um sistema NC converte os valores dos eixos de coordenadas do programa em posições relativas do programa de maquinação, dependendo do movimento da ferramenta ou peça.

Este movimento relativamente ao eixo respetivo pode ser efetuado por diferentes sistemas mecânicos: um sistema de parafuso ou uma transmissão por correias, conforme demonstrado na Figura 2.9. O sistema de parafuso é amplamente utilizado na conversão do movimento rotativo em movimento linear, utilizando um parafuso com uma rosca fixa que movimenta a carga, ou por meio de um sistema de esferas. Por outro lado, a transmissão por correia é utilizada em aplicações de controlo de movimentos de baixa inércia e baixa carga. Este sistema consiste numa carga ligada a uma correia e a duas polias, convertendo assim o movimento linear consoante a direção do motor [7, 9].

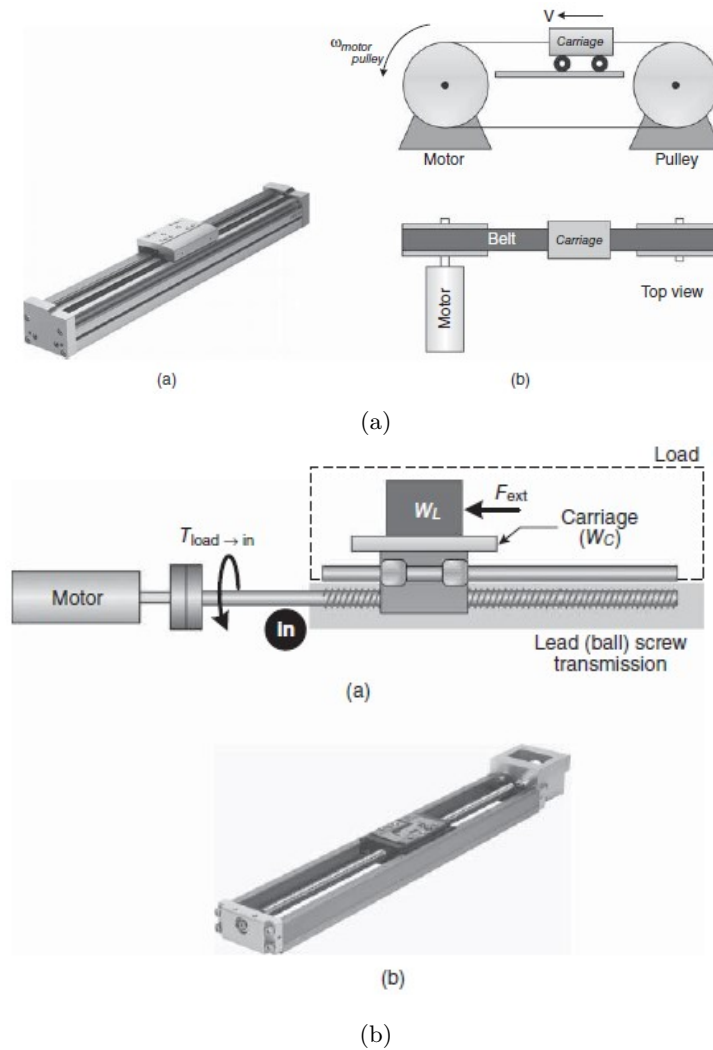


Figura 2.9: Sistema de movimento linear [9]: a) Sistema de transmissão por correia b) Sistema de parafuso

Dependendo do sistema mecânico a ser utilizado, existem dois tipos de sistema de controlo elétrico nos sistemas de posicionamento: malha aberta e malha fechada, representados na Figura 2.10.

Um sistema em malha aberta opera sem verificar se a posição real alcançada no movimento corresponde à posição desejada, utilizando apenas um motor de passo a passo. Em contraste, o sistema de malha fechada utiliza medições de *feedback* para confirmar que a posição final é a desejada, empregando um servomotor, *encoders* e outros dispositivos de controlo, como demonstrado na Figura 2.10. Os sistemas de malha aberta são mais económicos do que os de malha fechada e adequados quando a força do movimento é baixa. Por outro lado, os sistemas de malha fechada são especificados para aplicações que requerem movimento contínuo, onde as forças são significativas [7].

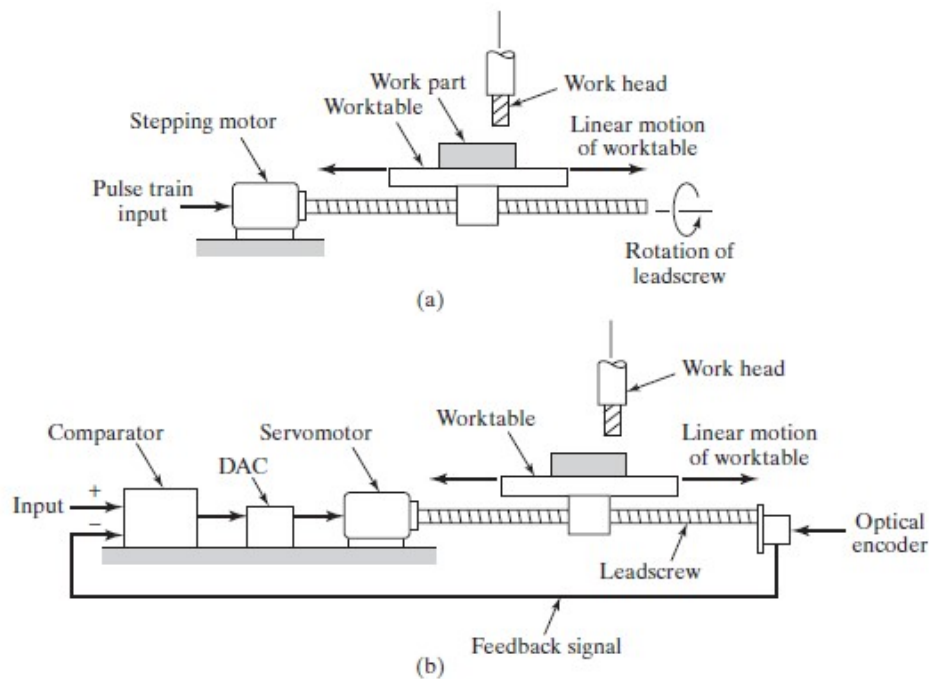


Figura 2.10: Tipos de controle de movimento [7]: a) Malha aberta
b) Malha fechada

Precisão

Para maquinar ou processar uma peça de trabalho com exatidão, um sistema de posicionamento deve possuir um elevado grau de precisão. Para medir essa precisão, existem três métricas que podem ser utilizadas: a resolução de controle, a exatidão e a repetibilidade, conforme ilustrado na Figura 2.11.

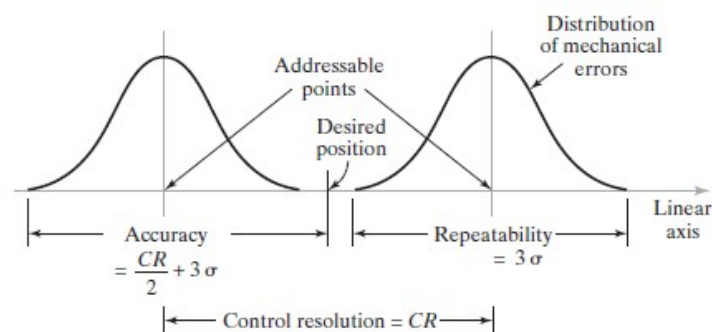


Figura 2.11: Definição de resolução de controle, exatidão e repetibilidade [7]

A resolução de controle é definida pela menor distância que separa dois pontos endereçáveis adjacentes no movimento do eixo. Esta resolução está limitada por fatores eletromecânicos e pelo número de bits utilizados no registo do controlador para definir posições.

A exatidão (*accuracy*) é definida pela diferença entre a posição desejada e a real, considerando o pior cenário. Como a posição final só pode permanecer em posições endereçadas, isso provoca um erro.

Por fim, a repetibilidade refere-se à capacidade do sistema de retornar ao mesmo ponto repetidamente, mesmo após vários ciclos [7].

2.1.5 Definições da Máquina

Quando se começa a trabalhar diretamente com uma CNC, nota-se a existência de várias funções disponíveis que o operador pode executar. Como é evidente, a CNC deve ser capaz de processar programas originados por *softwares*, como o CAD/CAM, ou programas elaborados por um programador, que o interpretador da CNC executa. No entanto, existem ainda outras funções cruciais, algumas das quais são comuns no movimento de eixos, como o *Homing* e o *Jogging*. Também será abordado o *Manual Data Input* (MDI), uma funcionalidade das CNC [7, 9].

Homing

A grande maioria das máquinas com movimentação de eixos contém diversas rotinas incorporadas para a procura da localização. O objetivo é encontrar uma posição de referência, designada como posição inicial. É necessário realizar este processo sempre que a máquina é ligada ou quando passa por processos de manutenção, uma vez que a posição da máquina é desconhecida em todos os eixos.

Uma das ligações mais comuns para eixos utiliza um sensor que determina a posição inicial (*Home*), e dois sensores de limite, como demonstra a Figura 2.12.

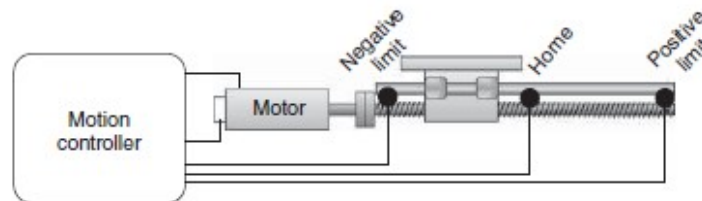


Figura 2.12: Eixo linear com os dois *switches* limitadores e o *home switch* [9]

Um dos métodos utilizados chama-se *move-until-trigger* e consiste num pré-acionamento e um pós-acionamento. No pré-acionamento, o eixo move-se na direção do sensor *Home* até que o sensor seja acionado. A partir desse momento, o eixo desacelera até parar, pois encontra-se no lado oposto do sensor. Após parar, inicia-se o processo pós-disparo, que consiste em mover-se no sentido inverso, com velocidade reduzida, até que o sensor *Home* abra, registrando assim a posição *Home* do eixo, como ilustrado na Figura 2.13 [9].

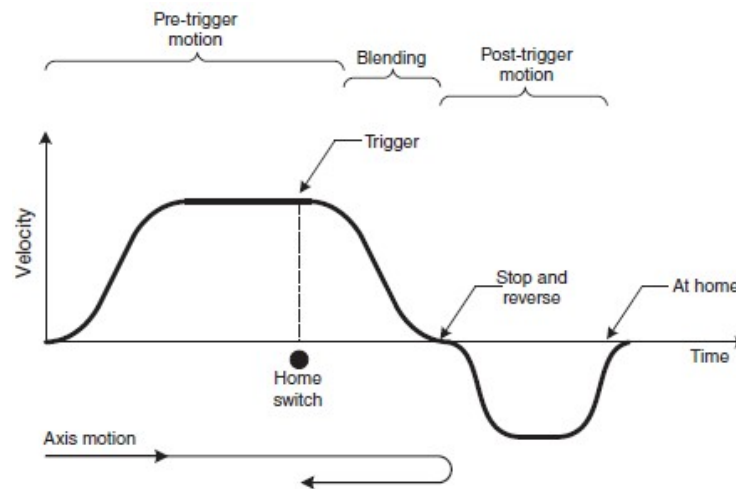


Figura 2.13: Movimento de *Homing Move-Until-Trigger Scheme* [9]

Jogging

O *jogging* consiste em movimentar os eixos, individualmente, manualmente. Os parâmetros de configuração podem ser ajustados para definir a velocidade e a aceleração do movimento. Os comandos são emitidos pelo operador e consistem em mover o eixo na direção desejada, conforme a especificação do próprio [9].

Manual Data Input (MDI)

Apesar da automatização na construção do programa de instruções, é necessário confirmar se este procedimento será executado conforme o esperado. Para isso, foi criada uma ferramenta que permite ao operador introduzir manualmente a programação da máquina, designada como *Manual Data Input* (MDI).

Esta ferramenta é concebida para a introdução de pequenos trabalhos, permitindo a programação destas máquinas.

Muitos sistemas permitem visualizar o programa que está a ser executado e alertam para possíveis erros de programação, como a visualização de como a peça será tratada [7].

2.1.6 Computer-aided Design/Computer-aided manufacturing

Um sistema CAD/CAM é um sistema de computação gráfica interativa equipado com *software* para realizar tarefas específicas de conceção e fabrico, integrando as funções de conceção e controlo.

Com a introdução deste sistema, foi possível automatizar a transição entre a conceção do produto e algumas fases do fabrico, facilitando a integração entre estas etapas.

O conceito principal é pegar nas especificações do projeto do produto, tal como se encontram na base de dados CAD, e convertê-las automaticamente num plano de processo para fabricar o produto, geralmente num programa de instruções NC que pode ser introduzido diretamente na CNC [7].

2.1.7 Outros Tipos de Aplicações de CNC

O princípio de funcionamento do NC não é exclusivo do torneamento e fresagem, mas é aplicado em diversas áreas, como [7]:

- Prototipagem rápida e manufatura aditiva: processos que adicionam material em camadas para construir peças, como estereolitografia, sinterização a *laser* seletiva e modelagem por fusão e deposição. Um exemplo são as impressoras 3D (Figura 2.14a) e, mais recentemente, em construção de casas;
- Corte por jato de água e jato abrasivo: máquinas que cortam materiais metálicos e não metálicos com um fluxo de água de alta pressão, podendo incluir partículas abrasivas para cortes mais difíceis;
- Máquinas de posicionamento de componentes: equipamentos que posicionam componentes em placas de circuito impresso, seja para montagem por inserção ou montagem superficial;
- *Coordinate Measuring Machine* (CMM): equipamentos de inspeção automatizada que verificam dimensões de peças por meio de uma sonda manipulada em três eixos (Figura 2.14b);
- Roteadoras de madeira e cortadores de granito: máquinas NC semelhantes a fresadoras, mas voltadas para materiais não metálicos;
- Máquinas de aplicação de fita para compósitos poliméricos: aplicam camadas de fita sobre moldes contornados para formar painéis multicamadas;
- Máquinas de enrolamento de filamentos para compósitos poliméricos: enrolam filamentos impregnados com polímero em moldes cilíndricos rotativos para criar peças estruturais.

2.1.8 Vantagens e Limitações da Tecnologia NC

A tecnologia de NC oferece uma série de benefícios em relação aos métodos manuais de produção, especialmente em aplicações onde há repetitividade, complexidade geométrica e necessidade de alta precisão. Quando bem aplicada, a tecnologia NC resulta em ganhos de produtividade, qualidade e flexibilidade. No entanto, a sua

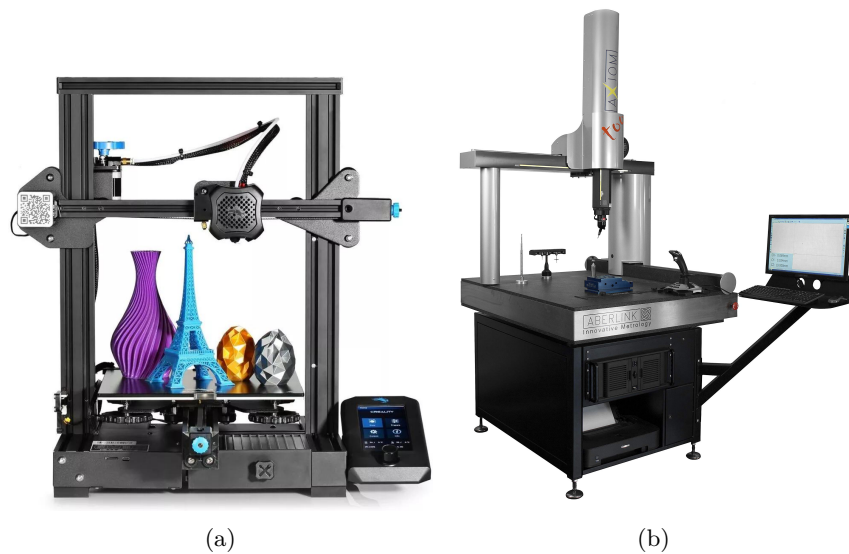


Figura 2.14: Aplicações de máquinas NC: a) impressora 3D [10] b) CMM [11]

implementação também envolve custos e exigências técnicas que precisam ser considerados. A seguir, apresentam-se de forma resumida as principais vantagens e limitações da utilização de sistemas NC.

Entre as principais vantagens da tecnologia NC estão a redução do tempo improdutivo, maior precisão e repetibilidade, menor taxa de refugo e menor necessidade de inspeção. A tecnologia permite a produção de geometrias mais complexas, facilita alterações de engenharia por meio da modificação do programa e exige dispositivos de fixação mais simples. Além disso, contribui para a redução dos estoques, encurta os prazos de fabricação e requer menos espaço físico na planta. A operação das máquinas NC requer outras qualificações para o operador, uma vez que o ciclo de fabrico ocorre de forma automatizada e controlada por programa.

Por outro lado, as desvantagens incluem o elevado custo de investimento inicial, devido à presença de sistemas eletrônicos e mecânicos mais sofisticados. A manutenção é mais exigente e cara, por envolver componentes eletrônicos que requerem técnicos especializados. Outro ponto importante é a necessidade de programação das peças, representando uma etapa adicional em comparação aos métodos convencionais. Além disso, para aproveitar economicamente os benefícios da tecnologia NC, muitas empresas precisam aumentar a utilização das máquinas, o que pode levar à adoção de múltiplos turnos de trabalho e, conseqüentemente, à ampliação do suporte técnico e administrativo [7].

2.2 Código G

O código G, ou código numérico, é uma função utilizada na linguagem de programação das CNC que contém informações sobre a posição de uma ferramenta para executar um comando, acionando assim as máquinas [12].

2.2.1 Contexto Histórico sobre Código G

Com a primeira implementação de NC desenvolvida pelo MIT, foi criada a primeira linguagem de NC, que não era padronizada. Durante algumas décadas, surgiram diferentes implementações. O meio de entrada do código passava por fitas perfuradas nas primeiras implementações, evoluindo para outras tecnologias, como fita magnética, disquetes e, mais recentemente, memória da máquina. A primeira versão padronizada surgiu em 1963, pela *Electronic Industries Alliance* (EIA), com o RS-274, nos EUA. Após isso, surgiram versões atualizadas para acompanhar as evoluções e novos métodos. Outros países utilizam padrões diferentes, como o *German Institute for Standardization* (DIN) 66025 (Alemanha) ou a *International Organization for Standardization* (ISO) 6983 [7, 13, 14].

2.2.2 Estrutura de um Programa de Instruções

Um programa de instruções é uma lista de instruções codificadas que descreve como será fabricada a peça. Estas instruções são chamadas de dados e incluem informações tecnológicas e geométricas, para executar as funções e movimentos necessários para o fabrico das peças.

O programa de instruções encontra-se organizado em linhas de dados separadas, onde cada linha descreve um conjunto particular de operações. Esta divisão é designada de blocos. Os blocos contêm o comando respetivo para a execução da operação desejada. A linguagem de programação reconhecida pela CNC é composta pelos grupos de códigos G e M. Um exemplo de um programa de peças aparece na Figura 2.15.

Cada bloco ou linha de código contém endereços (letras) descritas na Tabela 2.2, que aparecem nesta ordem: N, G, X, Y, Z, F, M, S, T. Podem ainda existir mais letras conforme os códigos, como o R, que indica o tamanho do arco num movimento circular. Esta ordem deve ser mantida em todos os blocos do programa, embora não seja necessário que todos apareçam, como demonstrado na Figura 2.16 [7, 15].

As funções preparatórias, ou códigos G, são utilizadas para determinar a geometria dos movimentos da ferramenta e o estado de funcionamento do controlador da máquina, como realizar movimentos ou operações definidas.

```
N1000
N5 G90 F600 S3600 T08 M06
N10 M3
N15 G00 X-20 Y-20 M8
N20 G00 Z-5
N25 G41
N30 G01 X0 Y0
N35 G01 X0 Y30
N40 G02 X14.374 Y33 I7.5 J0
N45 G03 X18.965 Y30 I4.581 J2
N50 G01 X33.860
N55 G01 X44.5 Y40
N60 G01 X50
N65 G02 X65 Y25 I7.5 J-7.5
N70 G03 X49 Y5.932 I9.226 J-23.989
N75 G01 X0 Y0
N80 G40
N85 G00 X-25 Y-25
N90 G0 Z100 M9
N95 M5
N100 M30
```

Figura 2.15: Exemplo de um programa NC [2]

As funções diversas, ou código M, são utilizadas para comandar sinais de ligar/desligar a máquina e o *spindle*. Dependendo do fabricante, códigos com numeração mais alta podem variar, enquanto os de ordem inferior são constantes entre fabricantes [15].

2.2.3 Explicação de Códigos Relevantes

O código G é muito abrangente, podendo ser utilizado tanto em tornos como em fresadoras. Muitas instruções são comuns a ambas as máquinas, enquanto outras são específicas para cada uma. Nesta subsecção, será abordado, de forma breve, os códigos mais relevantes e a respetiva explicação. Os restantes códigos encontram-se descritos no Anexo A.

Código G

Como já mencionado nesta secção, as funções G são utilizadas em momentos em que são efetuados movimentos ou algo que afeta os movimentos entre eixos.

Tabela 2.2: Endereços do código G [15]

Endereço	Função
N	Número do Bloco
G	Código G (Função Preparatória)
X	Distância absoluta/incremental percorrido do eixo X
Y	Distância absoluta/incremental percorrido do eixo Y
Z	Distância absoluta/incremental percorrido do eixo Z
F	<i>Feed Rate</i>
M	Código M (Função Diversa)
S	Velocidade do <i>Spindle</i>
T	Gestão de Ferramentas

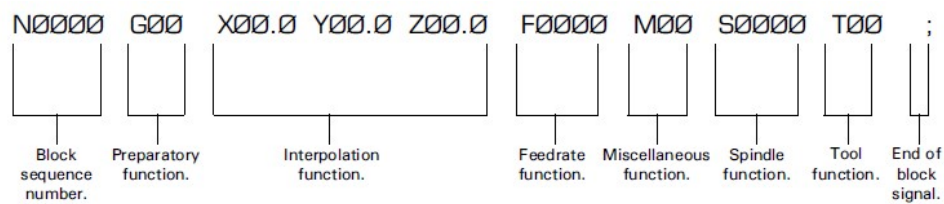


Figura 2.16: Exemplo de uma linha de código [15]

As funções de movimento encontram-se nos códigos G00, G01, G02 e G03. Estes códigos são responsáveis por deslocar o objeto/ *spindle* para os valores mencionados.

As funções G00 e G01 têm semelhanças ao serem chamadas, exigindo apenas as coordenadas. A grande diferença é que o G00 (ou chamada de posicionamento rápido/ transversal) executa um movimento sem corte e com um *feed rate* maior que o normal, podendo não efetuar um movimento em linha reta do ponto inicial para o final, perfeito para movimentos de posicionamento. Por outro lado, o G01 (ou interpolação linear) executa um movimento de corte em linha reta do ponto inicial ao ponto final, por meio de interpolação, com o *feed rate* estimado pelo programa. Estes movimentos estão representados na Figura 2.17 [7, 15].

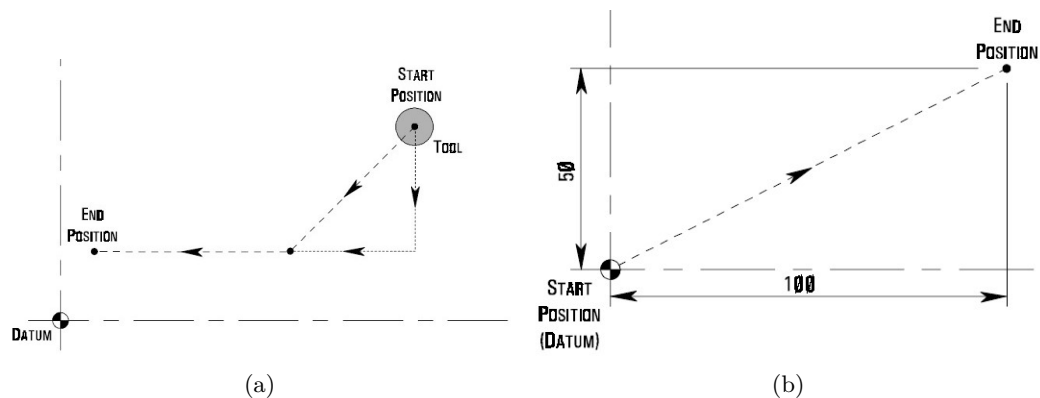


Figura 2.17: Movimentações lineares: a) Posicionamento rápido b) Interpolação linear [15]

As funções G02 e G03 são chamadas de interpolação circular. Estas funções são indicadas para corte, com o *feed rate* definido pela máquina, onde o G02 realiza um movimento no sentido dos ponteiros do relógio, enquanto o G03 o efetua no sentido contrário. Para chamar estas funções, existem vários modos, mas o mais comum é referir o ponto final e o raio do arco. Também é possível mencionar um ponto central e marcar o raio. Um exemplo da aplicação destas funções encontra-se na Figura 2.18 [7, 15].

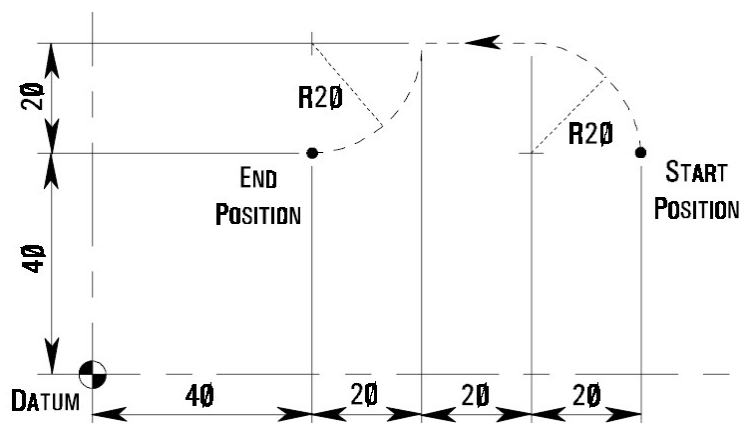


Figura 2.18: Interpolação circular [15]

Outra função utilizada é o G04, chamada de *dwell* ou tempo de espera, que é usada principalmente quando é necessário aguardar um determinado tempo antes de alterar as velocidades do *spindle* ou realizar outras operações. Para chamar esta função, basta inserir o G04 seguido de um X ou P com o valor, onde o valor de X corresponde a segundos e o P a milissegundos.

Um dos elementos principais refere-se às unidades que a máquina deve utilizar. Para isso, o primeiro bloco de um programa de código deve conter um G20 ou G21. Este código refere-se à unidade que o controlador irá utilizar, se em polegadas (G20) ou em milímetros (G21) [7, 15].

Outra função importante a ser referenciada no início, ou a meio quando necessário, é a definição do referencial que está a ser utilizada. Ao efetuar movimentos, pode-se optar por coordenadas absolutas ou incrementais. Quando se utilizam coordenadas absolutas, usa-se o G90 antes do conjunto de blocos que utilizam estas coordenadas, enquanto, para coordenadas incrementais, insere-se um G91 no conjunto de blocos, jogando com os sinais introduzidos dos valores para efetuar o movimento relativamente à última posição.

Por fim, e com semelhanças à função anterior, é necessário referenciar a métrica utilizada para o *feed rate*, utilizando o G94, sendo o *feed rate* definido em milímetros por minuto, ou o G95, em que o *feed rate* é definido em milímetros por rotação, sendo que estes últimos são usados em conjunto com o G21 [7, 15].

Código M

As funções M são utilizadas para controlar os estados das máquinas, como o envio e/ou recepção de sinais, a paragem da máquina e a ativação de atuadores, como o fluido de corte para o processamento de metais densos.

Dois dos códigos-base são o M00 e o M02. O M00 efetua um *Stop* no programa, avançando somente após sinal do operador, enquanto o M02 efetua um *Reset* ao programa, indicando o final do mesmo e retornando ao estado inicial.

Os códigos M03, M04 e M05 referem-se à movimentação do *spindle*. Para iniciar a movimentação, pode-se utilizar o M03 ou o M04, sendo que o M03 inicia o movimento no sentido dos ponteiros do relógio, enquanto o M04 o faz no sentido anti-horário. O M05, por sua vez, é responsável pela paragem da rotação do *spindle*. Para iniciar ou parar a rotação, é recomendado fazê-lo fora da peça a ser trabalhada e aguardar um *delay* até que a velocidade desejada seja alcançada.

Outro código utilizado em máquinas com múltiplas ferramentas é o M06. Este código é empregue quando se deseja efetuar uma alteração da ferramenta, acompanhada pela numeração (posição) da ferramenta a ser trocada.

Uma alternativa ao M02 é o M30, que, além de realizar o *Reset*, desliga outros atuadores, como o M05, entre outros que possam estar ativos [7, 15].

2.3 Hardware

Esta secção visa a exploração teórica dos equipamentos que podem vir a ser utilizados no projeto, numa forma abreviada. Numa primeira fase, será explorado o que é um IPC e quais são as suas diferenças em relação ao convencional PLC, como também outras características. De seguida, e de forma breve, serão apresentados os motores de passo e servomotores, quais as suas diferenças, que podem ser cruciais para a escolha de motores. De seguida, será mencionado o *spindle*, que características existem e qual a melhor escolha. Por fim, serão mencionados equipamentos de proteção referentes à queda de tensão ou princípios de curto-circuito.

2.3.1 Industrial PC (IPC)

Uma definição aceitável de *Industrial PC* (IPC) é a de computadores projetados e construídos com características e componentes que os ajudam a suportar o ambiente de uma fábrica. A arquitetura de um IPC é semelhante à de um computador comercial, geralmente com mais interfaces de ligação e suporte a protocolos industriais, com o uso de sistemas operativos modernos como os computadores comerciais.

As vantagens destes equipamentos relativamente aos computadores convencionais baseiam-se em características que os IPC suportam, como a capacidade de

operar em temperaturas mais elevadas, estrutura metálica, maior resistência mecânica e a vibrações, isolamento contra interferências magnéticas e proteção contra pó e humidade. Por isso, estes equipamentos são frequentemente utilizados em áreas expostas ao ar livre, em ambientes com altas temperaturas e humidade, ou em locais com elevadas vibrações [16, 17, 18].

Tipos de Instalação de *Industrial PC*

Entre os vários fabricantes, como a Beckhoff e a Siemens, existem diversas alternativas de soluções e funcionalidades, de forma a adaptar a melhor funcionalidade, ambiente e aplicação às necessidades do cliente [16].

- PC embebido (Figura 2.19a): são IPC que tendem a ser compactos e projetados para serem montados em armários (calha DIN geralmente). Estes IPC oferecem facilidade de integração com entradas e saídas ou outros terminais num armário de controlo;
- PC de painel (Figura 2.19b): um IPC híbrido, ou, no dia a dia, um PC *one-in-all*, que integra o painel de controlo (HMI) e um IPC convencional. É utilizado em ambientes onde o espaço e o custo são limitados, sendo ideal para situações em que a visualização é necessária para o manuseamento da respetiva máquina;
- *Rack PC* (Figura 2.19c): IPC maiores do que os embebidos e que geralmente não são montados em armários de controlo de máquinas. São utilizados em aplicações de servidores industriais, onde oferecem uma camada adicional de segurança de dados.

Além disso, são utilizados outros equipamentos, como *tablets* ou monitores, que servem meramente para enviar comandos ou receber informações do IPC em uso. É possível converter computadores convencionais para funcionar como IPC, numa solução de testes ou até final.

Industrial PC* ou *Programmable Logic Controller

Quem está no mundo da automação industrial sabe perfeitamente a importância dos PLC. Estes são "computadores industriais" utilizados para controlar máquinas e processos, sendo capazes de monitorar entradas e controlar saídas.

Mas qual é a diferença entre um IPC e um PLC, uma vez que ambos permitem executar funções semelhantes? A principal diferença reside no sistema operativo: um PLC é projetado para operar em tempo real, focando-se em tarefas de controlo, enquanto um IPC não só executa as funções de um PLC convencional, mas também é capaz de correr diversos aplicativos que não estão disponíveis num PLC [17, 18].

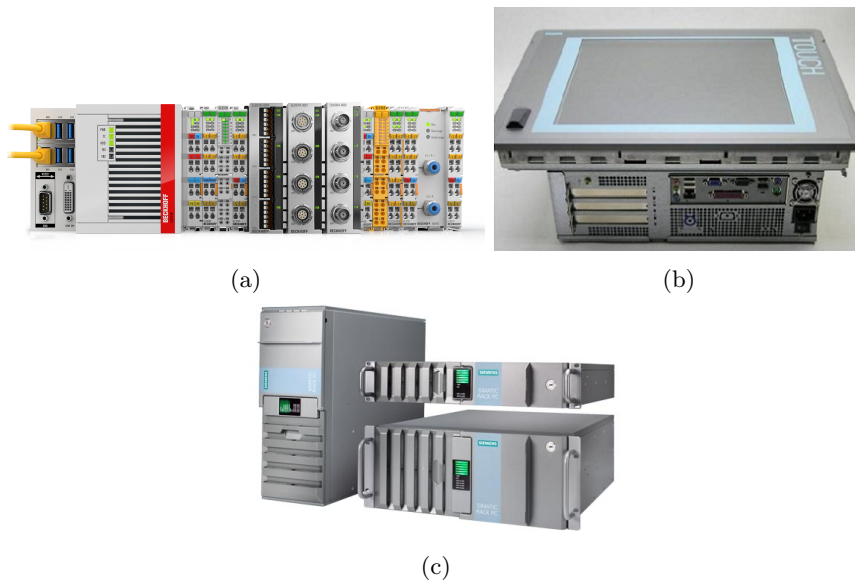


Figura 2.19: Exemplos de IPC: a) PC embebido [19] b) PC de painel [20] c) *Rack* PC [21]

Outra diferença importante é a capacidade de expansão. Os PLC possuem múltiplas portas de entrada e saída e permitem o uso de protocolos como CANbus, Modbus, entre outros. Por outro lado, os IPC oferecem vários *slots* de expansão, sendo mais versáteis e capazes de suportar um maior número de entradas e saídas, incluindo portas que não são permitidas nos PLC. Além disso, os IPC podem ser expandidos com placas sem fio, módulos 5G e outras tecnologias.

Assim, os PLC são adequados para tarefas de automação menores e específicas, além de serem mais económicos em comparação com os IPC.

Por sua vez, os IPC são mais indicados para aplicações complexas que exigem alta capacidade de computação e armazenamento, devido à sua flexibilidade, capacidade de processamento e conectividade. No entanto, uma das suas limitações é o custo inicial e a complexidade de implementação [17, 18].

2.3.2 Motores

Os motores elétricos desempenham um papel fundamental na automação industrial, na robótica e em inúmeras aplicações do quotidiano. A sua principal função é converter energia elétrica em energia mecânica, permitindo a movimentação de cargas, atuadores e sistemas de precisão com elevada eficiência e controlo.

O princípio de funcionamento baseia-se na interação entre campos magnéticos e correntes elétricas, que produzem força e, consequentemente, movimento. Embora existam diversos tipos de motores, cada um com características específicas, a escolha do motor adequado depende de fatores como precisão de controlo, binário necessário, velocidade de operação, custo e complexidade do sistema.

Duas das soluções mais recorrentes em sistemas de controlo de movimento são os motores de passo (*stepper motors*) e os servomotores. Ambos são amplamente utilizados, mas apresentam diferenças significativas no modo como operam, no tipo de controlo que requerem e nas aplicações para as quais são mais adequados.

Segundo Hughes [22], a compreensão do funcionamento dos motores e dos seus respetivos sistemas de controlo permite não só uma utilização mais eficiente, mas também uma integração mais robusta nos sistemas eletromecânicos modernos. O autor salienta ainda que, apesar da crescente sofisticação dos sistemas de controlo, os princípios fundamentais dos motores elétricos mantêm-se praticamente inalterados desde a sua concepção.

Nos próximos tópicos, serão analisados os motores de passo e os servomotores, destacando o seu princípio de funcionamento, vantagens e limitações, e comparando os seus desempenhos em diferentes contextos de aplicação.

Motores de Passo

Os motores passo-a-passo, ou motores *stepper*, são amplamente conhecidos pela facilidade de controlo através de computadores, microcontroladores e PLC, sendo frequentemente aplicados em máquinas que exigem controlo de posição, como ilustrado na Figura 2.20. A característica distintiva destes motores é que o seu veio de saída roda em uma série de intervalos angulares conhecidos; a cada impulso recebido, o veio avança um determinado passo. Após um número definido de impulsos, é possível determinar que o veio terá rodado um ângulo específico, o que torna o motor adequado para o controlo em malha aberta [22].

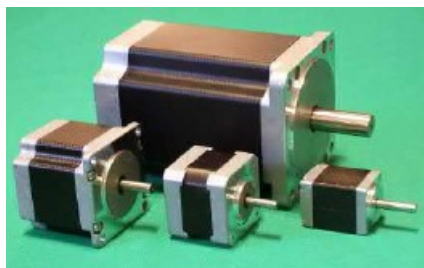


Figura 2.20: Motores de passo [22]

Quando os passos são dados a elevadas frequências, o motor de passo comporta-se como um motor convencional, correspondendo ao número de passos efetuados. A maioria dos motores de passo apresenta características semelhantes às dos motores convencionais, com binário e potência equivalentes.

O controlo destes motores é realizado, geralmente, em malha aberta, utilizando um sistema de montagem semelhante ao da Figura 2.21. O acionamento inclui circuitos eletrónicos de comutação. A saída corresponde à posição angular do motor, enquanto a entrada consiste em dois sinais digitais de baixa potência: um para os

impulsos e outro para a direção. O sistema é digital devido ao controlo da posição e do ângulo rodado após um número de impulsos, e, como mencionado, é em malha aberta, pois não é necessário obter *feedback* da saída [22].

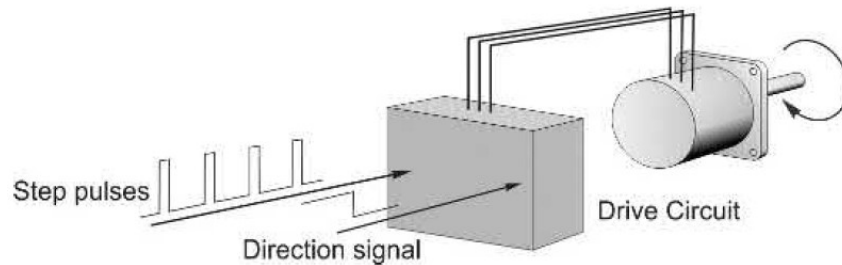


Figura 2.21: Controlo em malha aberta de um motor de passo [22]

Os impulsos de passo podem ser gerados por um controlador digital ou microprocessadores. A Figura 2.22 ilustra a sequência de impulsos e a sua atuação no veio. Como se pode observar, a cada impulso fornecido, ocorre um movimento na saída. A velocidade média do veio depende da frequência com que os impulsos são gerados: quanto maior a frequência, menor será o tempo necessário para efetuar os passos. É notada uma oscilação após o movimento, provocada pela rapidez do movimento, embora essa oscilação não seja perceptível a olho nu devido à rapidez do sistema.

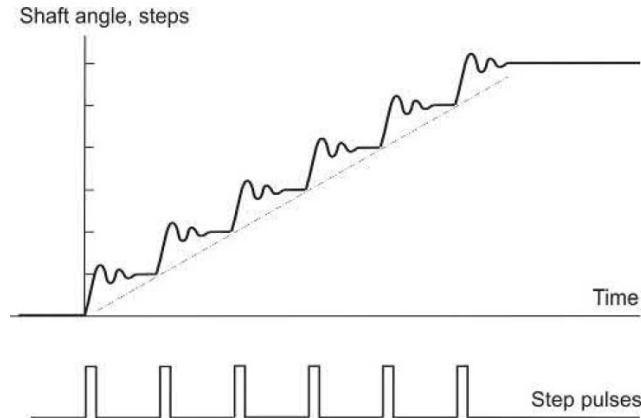


Figura 2.22: Resposta típica de um motor de passo a um conjunto de impulsos [22]

Os motores de passo mais comuns são do tipo híbrido, combinando as vantagens dos motores de relutância variável e dos ímanes permanentes. Embora geralmente funcionem em malha aberta, em algumas aplicações críticas é utilizado um sensor de posição para implementar malha fechada, aumentando a precisão e evitando perdas de passos. Uma limitação inerente a este tipo de motor é a redução significativa do binário a altas velocidades, tornando-o mais adequado para movimentos lentos e precisos [22].

Motores Servomotores

Os servomotores são motores elétricos utilizados em aplicações que requerem controlo rigoroso de posição, velocidade ou binário. Ao contrário dos motores de passo, os servomotores funcionam tipicamente em malha fechada, utilizando sensores de realimentação, como *encoders* ou *resolvers*, para garantir que o movimento executado corresponde com precisão ao comando recebido [22].

A arquitetura básica de um sistema servo inclui um motor (normalmente de corrente contínua sem escovas ou síncrono de ímanes permanentes), um sensor de posição e um controlador, conforme visualizado na Figura 2.23. Este controlador compara continuamente a posição atual com a posição de referência, ajustando a corrente no motor, por meio de um controlador, ao erro detetado. Esta abordagem assegura um movimento suave, preciso e com alta capacidade de resposta, mesmo na presença de variações de carga.

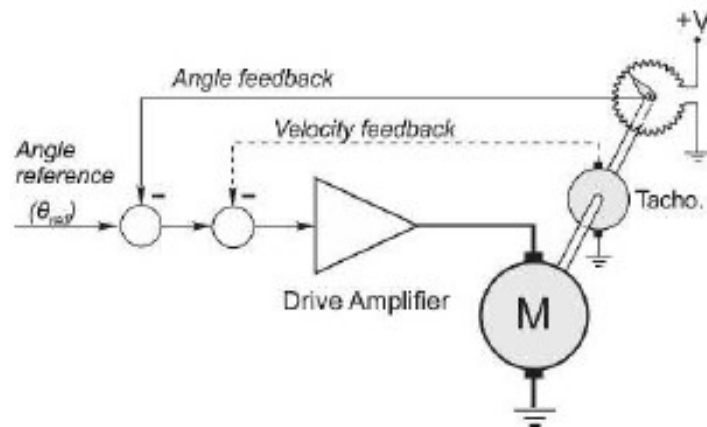


Figura 2.23: Controlo em malha fechada de um servomotor [22]

O uso de servomotores é comum em aplicações industriais que exigem desempenho dinâmico elevado, como robótica, sistemas CNC e braços articulados. Estes motores são também conhecidos pela sua capacidade de fornecer binário elevado a diferentes velocidades, bem como pela eficiência energética e pelo controlo contínuo de movimento [22].

Comparação entre Motores para a Aplicação

Para comparar ambos os motores, e tendo em conta a aplicação em sistemas de CNC, a escolha dos atuadores elétricos responsáveis pelo movimento dos eixos é um dos fatores críticos que influenciam o desempenho global da máquina. As opções mais comuns para movimentar os eixos são os motores de passo e os servomotores, ambos com características distintas em termos de controlo, precisão e resposta dinâmica.

Os motores de passo são amplamente utilizados em equipamentos CNC de pequena escala e menor custo, como fresadoras de bancada, impressoras 3D e máquinas de gravação. A sua principal vantagem reside na simplicidade de controlo: funcionam em malha aberta e não requerem sensores de posição, permitindo um controlo digital direto por meio de impulsos. Este tipo de motor é ideal para movimentos repetitivos e previsíveis, com cargas constantes, proporcionando uma boa precisão posicional sem necessidade de *feedback*. No entanto, os motores de passo perdem eficácia a altas velocidades, apresentando uma redução significativa de binário e o risco de perda de passos caso ocorra uma variação súbita na carga, um cenário comum em operações de corte ou maquinagem.

Por outro lado, os servomotores são a solução predominante em máquinas CNC industriais, onde se exige elevada precisão, velocidade e fiabilidade. Operando em malha fechada e recorrendo a sensores de posição (normalmente *encoders*), os servos monitorizam continuamente o movimento do veio e corrigem automaticamente qualquer desvio da trajetória programada. Isto traduz-se numa resposta dinâmica superior e numa maior capacidade para lidar com cargas variáveis, sem comprometer o controlo de posição ou velocidade.

Assim, num contexto de aplicação em CNC, os motores de passo podem ser uma escolha adequada para prototipagem, máquinas de baixo custo ou onde os requisitos dinâmicos não sejam críticos. Já os servomotores são indispensáveis em ambientes industriais, onde o desempenho, robustez e controlo em tempo real são fatores determinantes [22].

2.3.3 *Spindle*

O *spindle* (Figura 2.24) é um dos elementos mais críticos numa máquina CNC, sendo responsável pela velocidade de rotação da ferramenta de corte e pela transmissão de potência necessária ao processo de maquinagem. A sua função central no sistema determina, em grande medida, a qualidade do acabamento superficial, a precisão dimensional, a velocidade de corte e a capacidade de remoção de material.

Num sistema CNC moderno, o *spindle* não é somente um componente mecânico rotativo, mas sim uma unidade integrada que combina um motor de alto desempenho, sistemas de controlo de velocidade e binário, sensores de posição e interfaces mecânicas para a retenção da ferramenta. O *design* do *spindle* deve atender simultaneamente a critérios de rigidez, estabilidade térmica, rotação equilibrada e compatibilidade com sistemas automáticos de troca de ferramentas.

Dependendo da aplicação — seja corte pesado a baixas rotações ou maquinagem de alta velocidade — os requisitos técnicos do *spindle* variam significativamente. Assim, a seleção adequada do tipo de motor, das características de controlo e do método de fixação da ferramenta é fundamental para garantir o desempenho ótimo da máquina CNC no seu contexto operacional.

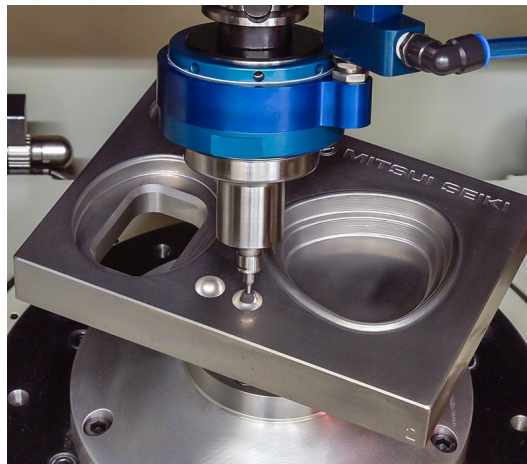


Figura 2.24: Exemplo de um *spindle* [23]

Para escolher um *spindle*, é necessário considerar características como a potência nominal e o binário fornecido pelo motor, a velocidade máxima e mínima de rotação, o tipo e tamanho da ferramenta a utilizar, o tipo de refrigeração e o peso [2, 24, 25].

Tipos de *Spindles*

Existem comercialmente dois tipos de *spindles*: os *spindles* com motor integrado e os *spindles* acionados por correias, como demonstrado na Figura 2.25.

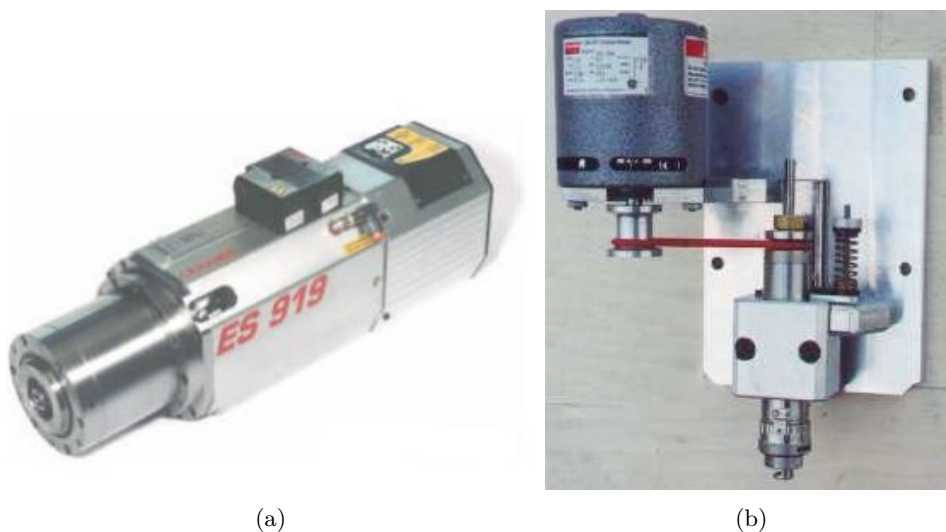


Figura 2.25: Exemplo de tipos de *spindles* [25]: a) *Spindle* com motor integrado b) *Spindle* movido a correias

Os *spindles* acionados por correias são constituídos por um eixo apoiado num sistema de rolamentos, que é introduzido dentro de um corpo onde se situa o sistema de fixação da ferramenta. A potência e a rotação são fornecidas ao *spindle* por um

motor externo, transmitidas por correias. Estes *spindles* estão limitados a uma velocidade máxima de rotação entre 12 e 15 mil rotações por minuto [24, 25].

As principais vantagens destes *spindles* são:

- custo reduzido, uma vez que necessitam de menos componentes e têm um custo inferior em comparação com os *spindles* de motor integrado;
- maior variedade de características, como potência, binário e velocidades de rotação máxima, devido a estas dependerem do motor externo;
- maior potência e binário, já que estas características provêm do motor externo, permitindo a utilização de motores de grandes dimensões que oferecem maior potência e binário, em comparação com os de motor integrado, que não têm essa capacidade.

Nos *spindles* com motor integrado, o motor está integrado na estrutura do *spindle* e acoplado ao eixo. É importante notar que as características do motor estão diretamente associadas ao motor utilizado, e a substituição por outro com características diferentes é difícil. Estes equipamentos necessitam do arrefecimento do motor, utilizando ar (ventoinha ou ar comprimido) ou água.

As principais vantagens deste tipo de *spindle* relativamente ao tipo anterior são:

- velocidades máximas mais elevadas, uma vez que não utilizam correias, permitindo aproveitar ao máximo o motor, pois este está acoplado diretamente ao eixo do *spindle*;
- menores esforços nos rolamentos, já que não suportam a tensão das correias. Os rolamentos dos *spindles* com motor integrado são poupados às forças adicionais, reduzindo a necessidade de manutenções frequentes.

Para garantir o correto funcionamento e a longevidade dos *spindles*, existem alguns parâmetros de segurança que são obrigatórios, como a temperatura interna do *spindle* (especialmente nos modelos com motor integrado) e o bloqueio/desbloqueio da ferramenta. Durante as operações de manipulação, manutenção e reparação do *spindle*, é indispensável que o *spindle* esteja parado e, preferencialmente, desligado da corrente elétrica [24, 25].

Controlo de *Spindles*

O controlo de *spindles* está intrinsecamente associado à utilização de *Variable-Frequency Drive* (VFD), ou variadores de frequência, que permitem a regulação da velocidade de rotação do motor através do ajuste da frequência da corrente elétrica, geralmente por meio de um sinal *Pulse-Width Modulation* (PWM). Esta abordagem tornou-se amplamente adotada pela sua eficiência, simplicidade e compatibilidade

com motores de indução trifásicos, sendo aplicável tanto em *spindles* acionados por correia como em versões com motor integrado.

Além dos VFD, podem ser utilizados outros sistemas de controlo, consoante os requisitos da aplicação, nomeadamente motores síncronos com ímanes permanentes, que oferecem maior eficiência e menor dissipação térmica, e configurações com realimentação por sensores (*encoders* magnéticos ou ópticos), que permitem controlo em malha fechada. Estas soluções são particularmente relevantes em contextos de elevada precisão, onde é necessário garantir estabilidade de velocidade, posicionamento ou sincronismo [24].

Sistemas de Retenção da Ferramenta

O sistema de retenção da ferramenta constitui um elemento funcional essencial em qualquer aplicação que recorra a *spindles* rotativos, desde equipamentos de maquinaria a sistemas automatizados ou robóticos. A sua função principal é garantir que a ferramenta permanece firmemente acoplada ao eixo rotativo durante o funcionamento, resistindo a forças axiais e radiais, vibrações e variações térmicas.

De uma forma geral, a retenção da ferramenta pode ser efetuada manualmente ou por meio de sistemas automáticos. Nos sistemas manuais, o operador introduz e fixa a ferramenta através de porcas ou parafusos de aperto, sendo uma solução comum em equipamentos de menor complexidade ou utilização intermitente. Estes sistemas, apesar de simples, requerem paragens e intervenção humana, limitando a sua aplicabilidade em processos contínuos ou automatizados.

Por outro lado, os sistemas automáticos recorrem a mecanismos internos de fixação (*drawbar*), frequentemente assistidos por atuadores pneumáticos ou hidráulicos. Estes sistemas conseguem aplicar forças de aperto controladas e repetíveis, com integração possível em ambientes controlados eletronicamente, permitindo a comutação rápida e segura de ferramentas. Em ambos os casos, o controlo do sistema pode envolver sensores de posição, deteção de presença da ferramenta ou monitorização da força de aperto, os quais são elementos de interesse no domínio da eletrónica e da automação industrial.

A escolha do tipo de interface entre a ferramenta e o *spindle* (como *Hohlschaftkegel* (*Hollow Shank Taper*) (HSK), *Schaftkegel* (*Belt Taper*) (SK), *Belt Taper* (BT), entre outros, visualizados na Figura 2.26a) influencia diretamente a precisão, o equilíbrio dinâmico e a facilidade de controlo do sistema. Interfaces de alta rigidez, como o HSK (Figura 2.26b), permitem, por exemplo, melhores resultados em aplicações de elevada velocidade, ao assegurar contacto simultâneo entre flange (base do cone) e cone, reduzindo os efeitos de dilatação térmica e melhorando a estabilidade dimensional [24].

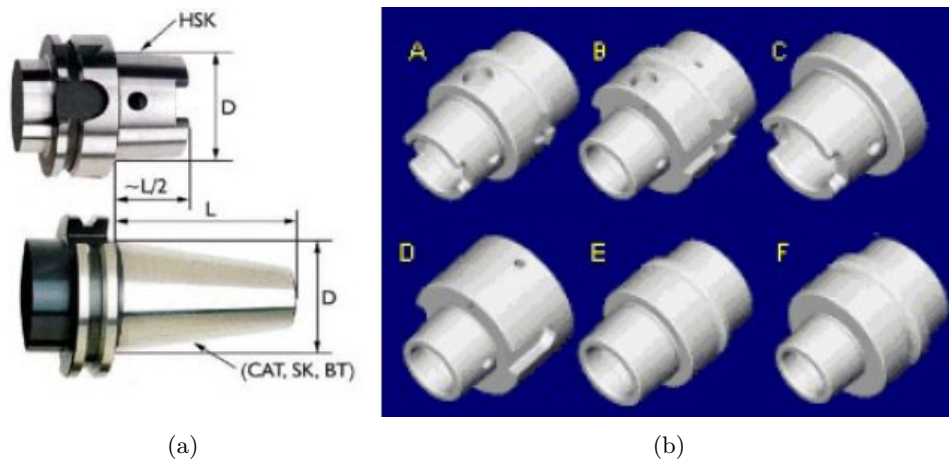


Figura 2.26: Exemplo de diferentes tipos de retenção da ferramenta [24]: a) Diferença entre o HSK e os modelos CAT b) Diferentes tipos de HSK

Análise Técnica

Ao abordar este componente, foram apresentados os principais aspetos associados ao *spindle*, que é o componente central no desempenho de uma máquina CNC. A escolha do *spindle* adequado implica mais do que selecionar somente a potência ou a velocidade; exige uma análise cuidadosa de diversos fatores, em função do tipo de maquinação e do ambiente de trabalho.

Entre as características mais relevantes estão o tipo de fixação da ferramenta (manual ou automática), a compatibilidade com os sistemas de troca rápida (como HSK ou BT, no caso de troca automática), a precisão e a repetibilidade da fixação e a integração com o sistema de controlo da máquina.

Adicionalmente, aspetos como o método de controlo, o sistema de refrigeração e a qualidade da ligação entre o *spindle* e o sistema de retenção da ferramenta influenciam diretamente a fiabilidade e produtividade do conjunto.

A seleção do *spindle* deve, assim, equilibrar desempenho, robustez, facilidade de manutenção e adaptabilidade ao tipo de operação pretendida, garantindo que este responda com eficácia às exigências de um ambiente CNC moderno.

2.4 Protocolos de Comunicação e Sinais Relevantes

Cada vez mais, o contexto industrial moderno necessita de uma comunicação eficaz entre equipamentos, assumindo um papel fundamental na garantia da coordenação, fiabilidade e desempenho dos sistemas automatizados. A crescente complexidade dos processos industriais exige soluções de comunicação que sejam flexíveis e escaláveis.

Nesta secção, serão abordados diferentes tipos de protocolos de comunicação utilizados na transmissão de dados e sinais entre dispositivos, com especial enfoque na

comunicação direta entre equipamentos de controlo e *drivers* de atuadores. Serão analisados protocolos como o EtherCAT e o CANopen, amplamente utilizados em aplicações de controlo em tempo real devido à sua elevada eficiência e sincronização. Serão igualmente considerados sinais digitais clássicos, como A/B, CW/CCW e CLK/DIR, frequentemente utilizados em sistemas de controlo de movimento, por representarem métodos simples e fiáveis de transmissão de comandos e posições.

Por fim, serão explorados outros protocolos que estarão presentes nesta dissertação devido ao controlador utilizado, como o protocolo *Automation Device Specification* (ADS) e o *Automation Message Specification* (AMS), desenvolvido pela Beckhoff para o ambiente TwinCAT, destacando-se pela sua capacidade de integração transparente entre diferentes níveis de *hardware* e *software* em arquiteturas de controlo industrial.

2.4.1 *Ethernet for Control Automation Technology* (EtherCAT)

O *Ethernet for Control Automation Technology* (EtherCAT) é uma tecnologia *Ethernet* industrial em tempo real, desenvolvida inicialmente pela Beckhoff. O desenvolvimento passou, em 2003, para a *EtherCAT Technology Group*, uma das maiores organizações industriais de *Ethernet* e *fieldbus* do mundo. O protocolo EtherCAT, descrito na norma *International Electrotechnical Commission* (IEC) 61158, é adequado para requisitos em tempo real, tanto rígidos como flexíveis, em tecnologia de automação, testes e medições, além de outras aplicações [26].

Princípio de Funcionamento

O EtherCAT funciona como um sistema *master/slave* e opera com uma arquitetura *on the fly*. Basicamente, o *master* envia um telegrama contendo vários datagramas que serão distribuídos por diversos *slaves* existentes na rede. Ao passar pelo primeiro *slave*, o sistema transmite os datagramas associados a esse *slave* e recolhe os dados que serão enviados de volta para o *master*. Este processo é realizado por todos os *slaves*, que estão dispostos em série, respeitando o atraso inerente à transmissão de dados. Ao chegar ao último *slave*, o sistema deteta que a porta seguinte está aberta e transmite o telegrama de volta para o *master*, utilizando o recurso *full duplex* da tecnologia *Ethernet*. É importante ressaltar que os datagramas contidos no telegrama foram recolhidos pelos *slaves*. Na Figura 2.27, é apresentada uma representação gráfica do funcionamento, onde um telegrama é comparado a um comboio a efetuar um trajeto circular (rede), em que, em cada estação (*slave*), ocorre uma troca de passageiros (*datagramas*). Os passageiros que saem são representados em vermelho, enquanto os que entram no comboio são representados em azul, sendo que os passageiros que entram saem somente no final (*master*) [26, 27].

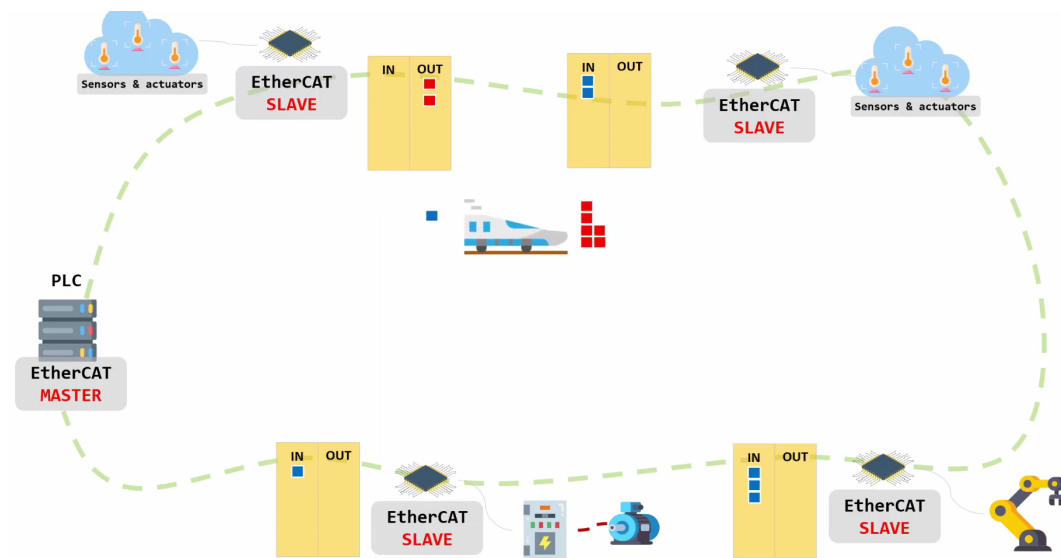


Figura 2.27: Funcionamento do EtherCAT [27]

Esta tecnologia permite simplificar circuitos complexos, como ilustrado na Figura 2.28, uma vez que, com uma única linha de barramento (*bus*), é possível interligar todos os *slaves* necessários para a aplicação, em vez de puxar uma linha para cada componente. É ainda possível interligar *gateways* para conectar outras redes que sejam necessárias, como CANopen, Modbus, entre outras.

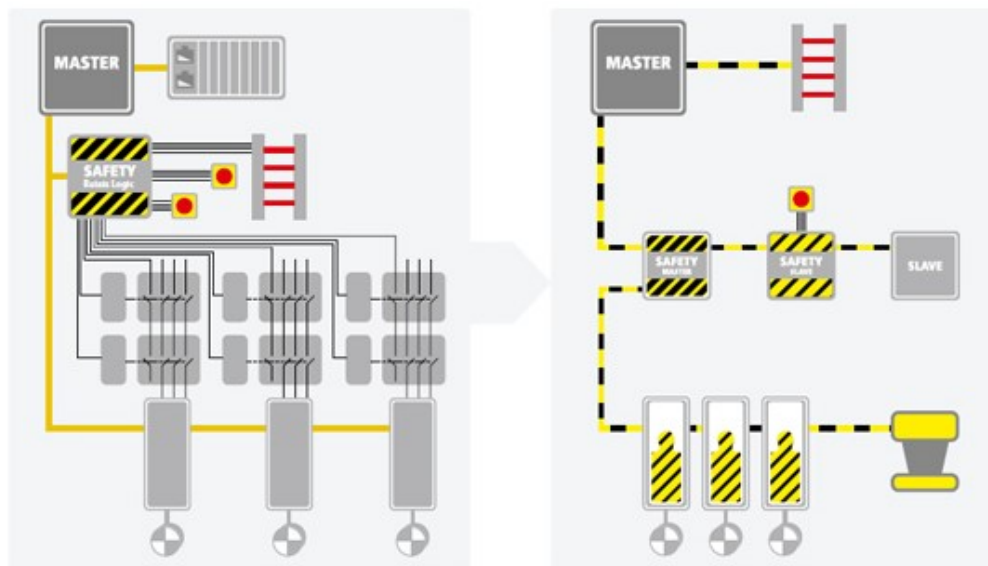


Figura 2.28: Simplificação de arquiteturas com o uso de EtherCAT: no circuito da esquerda, vê-se um exemplo utilizando controle por relés, enquanto à direita o controle é efetuado por EtherCAT [26]

Este protocolo apresenta baixa latência e alta eficiência, suporta diferentes combinações de rede, elimina a necessidade de *hardware* como *switches* e é altamente

flexível e escalável, permitindo a introdução de centenas de dispositivos associados. As suas ligações físicas podem estender-se por barramentos (como as cartas da Beckhoff) e/ou cabo Ethernet CAT6, ou outros cabos designados para tal [26].

2.4.2 CANopen

CANopen é um protocolo de comunicação de alto nível e especificação de perfil de dispositivo baseado no protocolo *Controller Area Network* (CAN). Enquanto o CAN foi desenvolvido para aplicações de redes embarcadas, como redes de automóveis, o CANopen fornece uma estrutura de programação da rede, descrição de dispositivos, definições de interfaces e perfis de aplicação. O uso deste protocolo permite padronizar a comunicação entre dispositivos e aplicações de diferentes fábricas, sendo muito utilizado nas áreas de automação e movimento de motores [28, 29].

Funcionamento do CANopen

O protocolo CANopen organiza a comunicação entre dispositivos numa rede CAN através de uma estrutura baseada em objetos, onde cada nó contém um *Object Dictionary* com os parâmetros e dados operacionais.

A comunicação é realizada por diferentes tipos de mensagens, cada uma com uma função específica:

- *Process Data Object* (PDO): troca rápida de dados em tempo real;
- *Service Data Object* (SDO): leitura/escrita de dados no *Object Dictionary*;
- *Emergency Object* (EMCY): notificação imediata de falhas críticas;
- *Synchronization Object* (SYNC): sinal de sincronização da rede;
- *Network Management Objects* (NMT): controlo dos estados dos nós (*start*, *stop*, *reset*);
- *Heartbeat / Node Guarding*: monitorização da presença e estado dos nós.

Esta arquitetura modular permite alta flexibilidade, facilitando a integração de dispositivos de diferentes fabricantes e a configuração dinâmica da rede [28].

Process Data Object (PDO)

Os *Process Data Object* (PDO) são utilizados para comunicação rápida e periódica entre dispositivos, sendo ideais para o envio de valores de sensores ou comandos de controlo de forma eficiente. Cada PDO transporta até 8 bytes e está diretamente ligado a variáveis definidas no *Object Dictionary*.

As principais características desta forma de comunicação incluem a comunicação em tempo real, baixo *overhead*, mapeamento flexível de variáveis e transmissão síncrona ou assíncrona.

Os PDO são o mecanismo base de troca de dados contínuos em aplicações como controlo de motores ou sistemas de monitorização rápida. Para efetuar esta comunicação, o *slave* recolhe as informações dos sensores e, em seguida, envia para o seu *master* a respetiva informação, podendo ainda receber sinais de controlo do *master*, como demonstrado na Figura 2.29 [28].

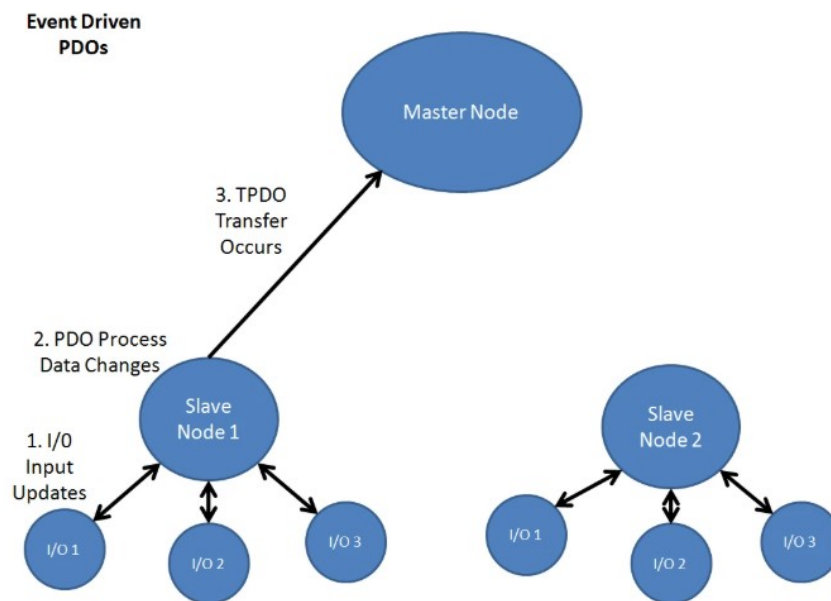


Figura 2.29: Mensagem PDO da comunicação CANopen [28]

Service Data Object (SDO)

Os *Service Data Object* (SDO) são utilizados para acesso completo ao *Object Dictionary*, permitindo a configuração de parâmetros de dispositivos e a leitura de estados internos. A comunicação ocorre entre um cliente e um servidor e suporta transferências segmentadas.

As principais características deste tipo de comunicação incluem o acesso a qualquer variável por índice/sub-índice, suporte a mensagens maiores que 8 bytes (segmentação) e é utilizado principalmente na configuração e diagnóstico.

Enquanto os PDO são usados durante a operação normal, os SDO são essenciais na fase de arranque e manutenção. Quando o *master* deseja comunicar com o *slave*, envia a mensagem para todos os *slaves* da rede, onde o respetivo *slave* irá enviar uma mensagem de resposta confirmando a receção, como demonstrado na Figura 2.30 [28].

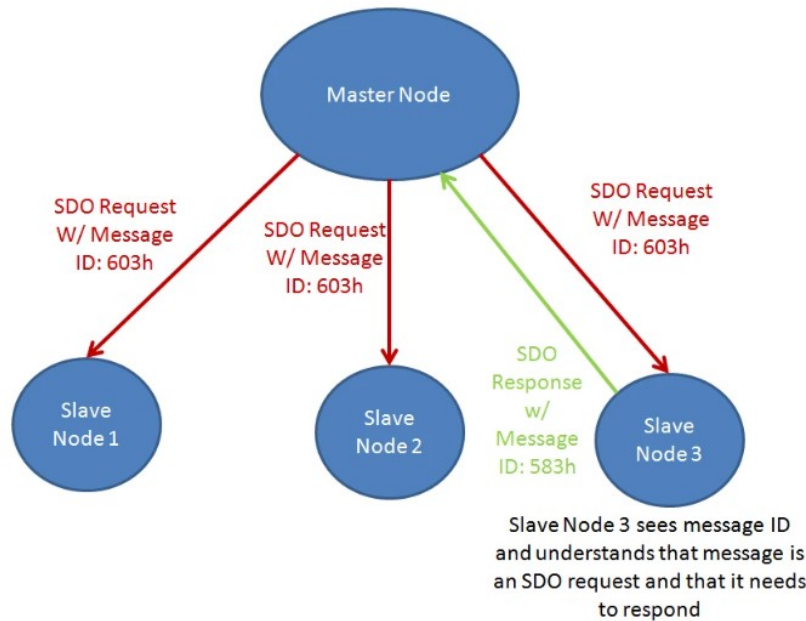


Figura 2.30: Mensagem SDO da comunicação CANopen [28]

Conexões

O padrão formal do CANopen não especifica conectores físicos. Porém, o *Certified Internal Auditor* (CiA) 303-1 e 106 oferecem recomendações quanto ao *pinout*, utilizando os seguintes [29]:

- conector DB-9: um dos conectores mais utilizados e recomendados, com o *pinout* na Figura 2.31a. É bastante utilizado devido à facilidade da integração na rede CAN por meio de dispositivos externos;
- conector M12: um conector comum, com o *pinout* mostrado na Figura 2.31b. Estes conectores são usados em ambientes mais agressivos, pois oferecem melhor resistência;
- conector de fio aberto: em alguns casos, a ligação pode ser efetuada por meio de fios abertos, com o *pinout* comum mostrado na Figura 2.31c.

2.4.3 Sinais Digitais para Controle de Posição

Uma forma popular de controlar o movimento de motores é através da comunicação entre o controlador e o atuador por meio de sinais digitais padronizados. Após a configuração do controlador e do atuador, os motores conseguem rodar em ambas as direções (horário e anti-horário), além de permitir o controle da velocidade.

Entre os métodos utilizados, destacam-se três tipos de sinais: sinais em quadratura (A/B), sinais de sentido de rotação (CW/CCW) e sinais de passo e direção

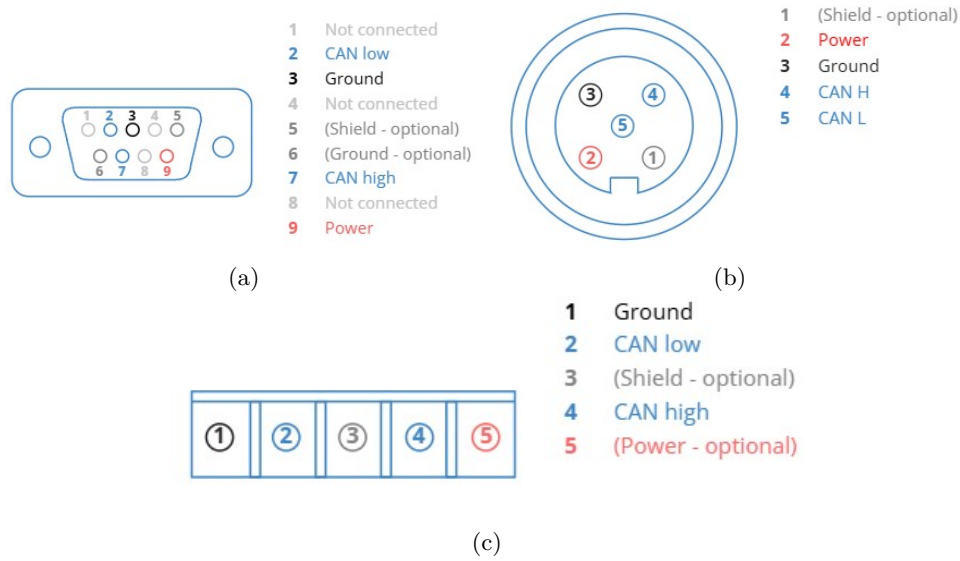


Figura 2.31: Tipos de conectores e *pinout* usados no CANopen [29]:
a) DB9 b) M12 c) fio aberto

(CLK/DIR). Para utilizar esses sinais, é necessário entender o funcionamento de cada um para movimentar o motor [30].

Sinal A/B

Os sinais A/B são utilizados em *encoders* incrementais para fornecer informações sobre a posição e a direção de rotação de um eixo. Estes sinais estão desfasados em 90° , permitindo a detecção da direção de rotação com base na ordem dos pulsos, como demonstrado na Figura 2.32. Este método é comum em aplicações que requerem alta resolução e precisão no posicionamento [30].

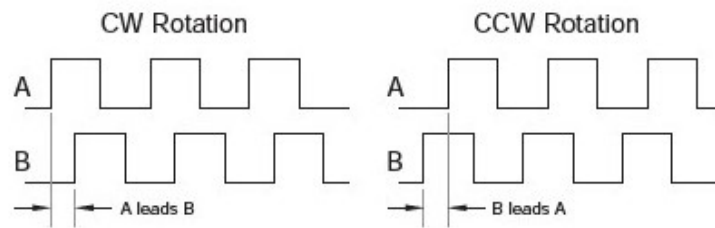


Figura 2.32: Sinal A/B [31]

Sinal CW/CCW

Os sinais CW/CCW são utilizados para indicar a direção de rotação de um motor. Este método utiliza dois sinais independentes: um para o sentido horário (CW) e outro para o anti-horário (CCW), como representado na Figura 2.33. É comum

ser implementado em sistemas onde a direção de movimento precisa ser claramente definida [30].

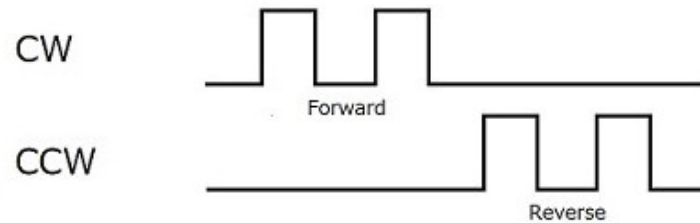


Figura 2.33: Sinal CW/CCW [32]

Sinal CLK/DIR

Os sinais CLK/DIR são usados para controlar motores passo-a-passo. Neste método, um sinal de *clock* determina o número de passos que o motor deve dar e a que velocidade, enquanto um sinal de direção define o sentido desses passos, como demonstrado na Figura 2.34 [30].

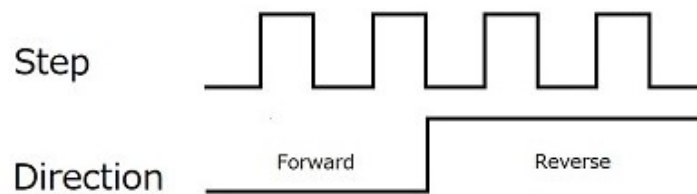


Figura 2.34: Sinal CLK/DIR [32]

2.4.4 Protocolos do Controlador Beckhoff

Cada controlador utiliza vários protocolos internos para comunicar com todos os níveis, tanto em *software* como em *hardware*, nas arquiteturas de controlo industrial existentes. A Beckhoff não é exceção. Para isso, a empresa criou protocolos que operam em segundo plano nos seus *softwares*, passando despercebidos para quem está a programar um controlador da empresa, como o protocolo ADS e o AMS.

Automation Device Specification (ADS)

O protocolo *Automation Device Specification* é um dos principais protocolos de comunicação do TwinCAT. A sua função principal é permitir a troca de dados e o controlo dos sistemas do TwinCAT. Este protocolo é independente do meio e pode comunicar via conexões série ou em rede, como o EtherCAT. O protocolo permite [19]:

- acesso à imagem do processo;
- troca de dados consistente;
- acesso a tarefas de entrada/saída;
- detecção de alterações de estado;
- leitura da informação do símbolo PLC;
- acesso por nome de variável;
- comandos de soma;
- acesso síncrono e assíncrono;
- mensagens cíclicas e baseadas em eventos.

Conceito de Dispositivos de *Automation Device Specification* (ADS)

A arquitetura do sistema TwinCAT permite que os módulos individuais do *software* (por exemplo, TwinCAT PLC, HMI, ...) sejam tratados como dispositivos independentes. Para cada tarefa, existe um módulo de *software* (“Servidor” ou “Cliente”). Os servidores do sistema são os dispositivos de trabalho em execução sob a forma de *software*, cujo comportamento operacional é exatamente igual ao de um dispositivo de *hardware*. Por esta razão, pode-se falar de dispositivos virtuais implementados no *software*. Os “clientes” são programas que solicitam os serviços dos servidores, como uma visualização ou um dispositivo de programação sob a forma de um programa. Assim, o TwinCAT pode crescer, uma vez que podem ser sempre adicionados novos servidores e clientes para tarefas como controladores de árvores de *comes*, osciloscópios, controladores PID, entre outros. A Figura 2.35 mostra o conceito de dispositivo TwinCAT, baseado no ADS.

As mensagens entre estes objetos são trocadas por meio de uma interface ADS consistente, através do “*router* de mensagens”. Este gera e distribui todas as mensagens no sistema através das ligações TCP/IP.

Os roteadores de mensagens TwinCAT existem em todos os dispositivos TwinCAT. Isso permite que todos os programas clientes e servidores TwinCAT troquem comandos e dados, enviem mensagens, transfiram informações de estados, entre outros [19].

No entanto, estes sistemas não se limitam somente ao PLC. O ADS consegue abranger todas as tecnologias existentes, sendo bastante flexível com o uso de tecnologias da Beckhoff. É possível usar diferentes PLC a comunicar entre si na mesma rede ADS, assim como comunicar diretamente com o desenvolvedor, obter e passar dados provenientes de um controlador, como o Raspberry, ou de dispositivos *Internet of Things* (IoT), como visualizado na Figura 2.36. Existem ainda outras

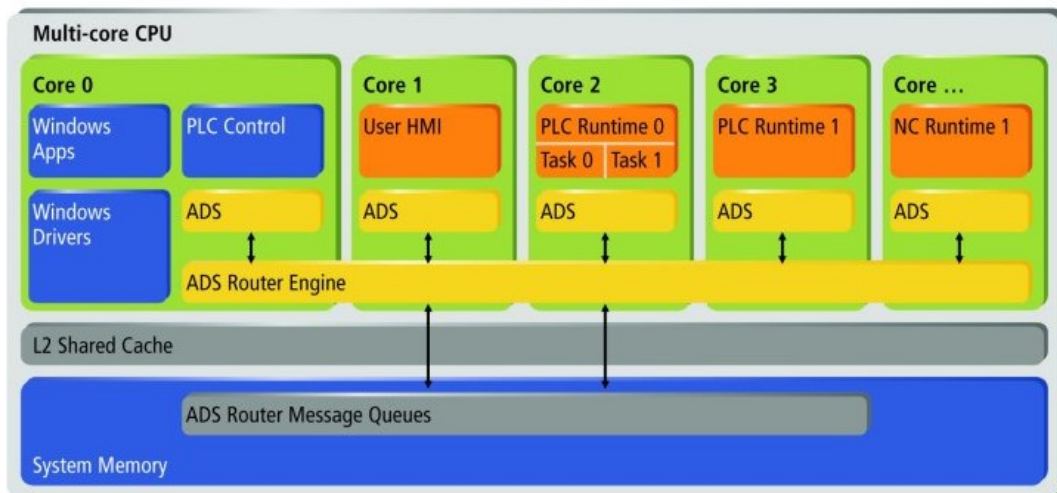


Figura 2.35: Conceito de dispositivos por ADS [19]

aplicações, como o usufruto do *Message Queuing Telemetry Transport* (MQTT), Java, PowerShell, C++ e ligação direta a sites .NET [19, 33].

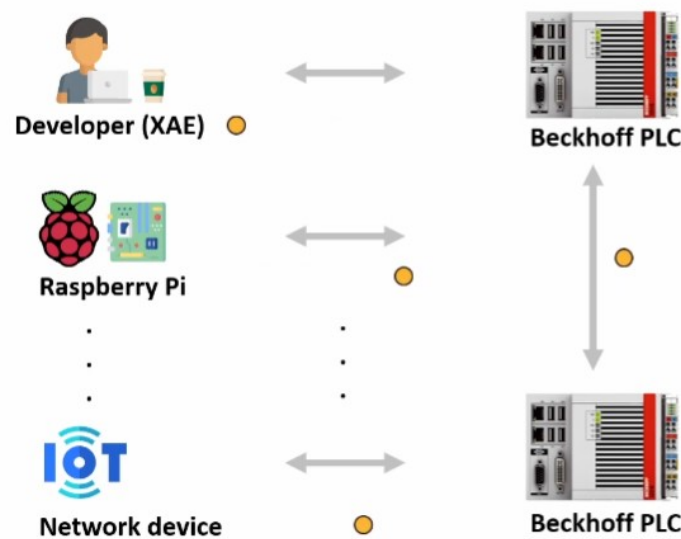


Figura 2.36: Interligação entre dispositivos por ADS [33]

Automation Message Specification (AMS)

O *Automation Message Specification* (AMS) é a camada de transporte subjacente ao protocolo ADS, ambos desenvolvidos pela Beckhoff para o ecossistema TwinCAT. Enquanto o ADS define a estrutura das mensagens e os serviços disponíveis para a comunicação com módulos TwinCAT (como o PLC, NC ou IO), o AMS é responsável

por assegurar que essas mensagens sejam corretamente entregues entre os diversos participantes do sistema [19].

Arquitetura e Funcionamento do *Automation Message Specification* (AMS)

O AMS funciona como um sistema de troca de mensagens entre dispositivos e aplicações, podendo operar em comunicações locais (dentro do mesmo sistema) ou distribuídas (entre dispositivos em rede). Cada participante na rede AMS é identificado por um AMSNetID, uma espécie de endereço lógico ou *Internet Protocol* (IP) que representa um dispositivo ou módulo TwinCAT no sistema global.

A estrutura de uma comunicação AMS é composta por:

- AMSNetID de origem e destino: identificadores dos participantes;
- portas AMS – permitem direcionar a comunicação para o módulo específico (por exemplo, PLC, IO, entre outros);
- mensagem ADS: o conteúdo útil, ou seja, a instrução ou os dados a transmitir.

Este sistema de endereçamento torna o AMS altamente escalável, permitindo que múltiplos sistemas TwinCAT e aplicações clientes comuniquem entre si de forma estruturada, independentemente da topologia de rede ou do meio físico utilizado (*Transmission Control Protocol/Internet Protocol* (IP), *User Datagram Protocol/Internet Protocol* (IP), entre outros) [19, 33].

Endereço AMSNetID

O AMSNetID é geralmente baseado no endereço IP do dispositivo, mas com maior flexibilidade. Por exemplo, um Net ID válido pode ser 10.42.128.22.1.1, onde os últimos dois dígitos permitem distinguir instâncias ou contextos diferentes dentro do mesmo dispositivo, como visualizado na Figura 2.37 [19, 33].



Figura 2.37: Endereço AMSNetId [33]

Portas *Automation Message Specification* (AMS)

As portas AMS são utilizadas para direcionar as mensagens ADS para os módulos específicos dentro de um sistema TwinCAT. Cada serviço ou componente escuta numa porta definida, permitindo comunicações simultâneas e organizadas entre diferentes partes do sistema.

Alguns exemplos de portas AMS comuns incluem:

- 851 – TwinCAT PLC *Runtime* 1;
- 852 – TwinCAT PLC *Runtime* 2;
- 100 – TwinCAT NC;
- 500 – TwinCAT *System Manager*;
- 501 – TwinCAT CNC;
- 300 – TwinCAT IO;
- 10000 e acima – Outras aplicações na qual o utilizador pode personalizar.

Esta estrutura facilita a escalabilidade e a integração de vários serviços no mesmo dispositivo ou entre dispositivos distintos numa rede industrial [19].

2.5 Conclusão da Fundamentação Teórica

A fundamentação teórica apresentada neste capítulo permitiu conhecer e compreender diferentes tecnologias, métodos e configurações associadas ao universo das máquinas-ferramentas das CNC.

Numa primeira abordagem, foi analisada a evolução das tecnologias de NC e CNC, bem como os sistemas de coordenadas, os tipos de movimentos e respetivas interpolações. Foram ainda descritas as principais operações mecânicas e os sistemas de controlo, tanto mecânicos como elétricos, complementados pela análise de ferramentas CAD/CAM e pela discussão das vantagens e desvantagens inerentes ao uso de máquinas CNC.

Seguidamente, abordou-se o código G, incluindo o seu enquadramento histórico, a sua estrutura e a normalização aplicada, destacando ainda os principais códigos utilizados, entre os quais se encontram os códigos M.

Posteriormente, foram explorados os componentes de *hardware* mais relevantes, com destaque para os IPC, os diferentes tipos de motores e respetivo funcionamento, comparando as suas vantagens e limitações nas aplicações CNC, bem como os diferentes tipos de *spindle*.

Por fim, foram analisados protocolos de comunicação industrial, como o EtherCAT e o OpenCAN, assim como os protocolos específicos do TwinCAT, designadamente o ADS e o AMS.

Com este capítulo, permitiu consolidar conhecimentos teóricos de forma a desenvolver os capítulos seguintes.

Capítulo 3

Trabalhos Relacionados

Neste capítulo, exploram-se vários projetos de CNC já existentes, que promovem a exploração prática do desenvolvimento de CNC a partir de diversos controladores.

Na primeira secção, serão analisados dois casos de estudo, realizados com controladores mais acessíveis, visando explorar o seu funcionamento e arquitetura, como métodos de análise.

Na segunda secção, será examinado um caso que utiliza um IPC, semelhante ao que será desenvolvido nesta dissertação, estudando assim a sua forma de funcionamento e arquitetura, que será próxima ao trabalho a desenvolver.

Por fim, será apresentada uma visão geral dos três casos de estudo, tanto a nível do esforço financeiro, como das aplicações, facilidade de uso e ideias a implementar.

3.1 Trabalhos com Controladores de Baixo Custo

Existem muitos trabalhos realizados para CNC onde o proprietário constrói a sua máquina utilizando recursos mais económicos, quotidianos e em dimensões reduzidas. Para auxiliar o desenvolvimento destas máquinas, existem várias bibliotecas de códigos e programas que facilitam o controlo das mesmas, funcionando como interpretadores, como o *G-code Relay Broadcaster Library* (GRBL) [34], LinuxCNC [35], Mach3/4 [36], entre outros. Neste caso, serão focados em dois trabalhos: uma mini-impressora de *Printed Circuit Board* (PCB), utilizando GRBL, e uma mini-CNC com controlo efetuado por LinuxCNC.

3.1.1 Impressora de PCB com o uso de GRBL

A *G-code Relay Broadcaster Library* (GRBL) é uma biblioteca que fornece uma alternativa de alto desempenho e baixo custo para o controlo de movimento em aplicações de fresadoras CNC. A implementação deste código é preferencialmente feita em Arduino, mas já existem aplicações que utilizam ESP32 e outros controladores comuns no mercado.

Este código é escrito em C e otimizado, executando várias funções do código G, exceto funções específicas, como a troca de ferramenta [34].

Um exemplo representativo da aplicação do GRBL, em contextos educativos e de prototipagem, é o de um grupo de investigadores da Indonésia [37] que desenvolveram uma impressora de circuito impresso baseada em máquinas CNC e no código GRBL.

A arquitetura apresentada pelo trabalho evidencia três níveis: um pelo computador, outro pelo Arduino e, por fim, os atuadores, como demonstrado na Figura 3.1. O computador efetua a comunicação e o controlo, contendo o GRBL Gru, que funciona como controlador, monitoriza e envia ficheiros de código G. O Arduino atua como o cérebro do sistema, controlando os motores de passo e interrompendo a operação quando acionado o botão de emergência ou o fim de curso. Por fim, os atuadores foram projetados para controlar três eixos, sendo estes controlados pelo Arduino através da placa *CNC Shield*.

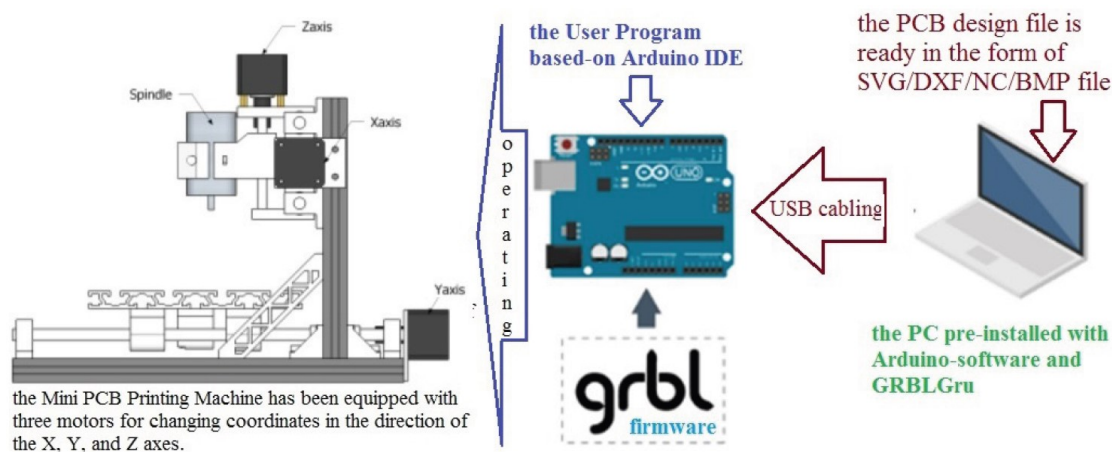


Figura 3.1: Arquitetura da mini-impressora de PCB [37]

Os componentes utilizados, além do Arduino e da *CNC Shield* para Arduino, incluem três motores de passo, um botão de emergência, dois fins de curso para cada eixo e um motor DC como *spindle*, sem controlo de velocidade, conforme visualizado na Figura 3.2.

O resultado final foi a construção funcional da máquina de 20 x 20 cm, conforme visualizado na Figura 3.3. A calibração é realizada movendo os eixos 10 mm de cada vez e registrando com um lápis a posição marcada pela ponta, em todas as direções,

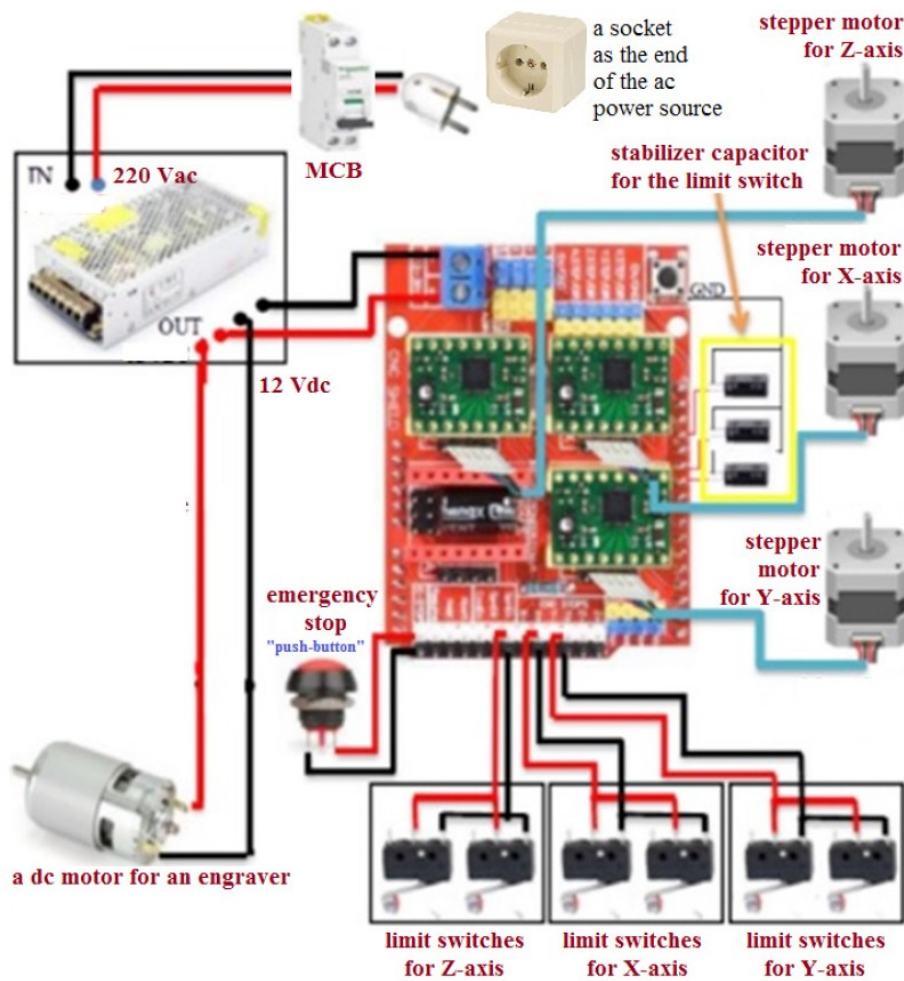


Figura 3.2: Esquemático da mini-impressora de PCB [37]

ajustando os parâmetros (step/mm) até que o movimento seja o esperado. Após a calibração, realizam-se testes de precisão, que consistem em movimentar os eixos de 5 a 10 mm em todas as direções, executando cada movimento 10 vezes. Após isso, verificou-se que o valor de precisão é aproximadamente de 0,6 mm, com um erro de 1,1% [37].

Ao analisar esta arquitetura detalhadamente, nota-se que possui algumas limitações, como o número de eixos, a pouca robustez relativamente ao controlo em tempo real e a baixa flexibilidade para expansão de I/O. No entanto, é uma opção bastante viável para quem deseja uma máquina CNC económica e adaptável às necessidades do utilizador.

3.1.2 *Computer Numerical Control* Controlada por LinuxCNC

O LinuxCNC é um sistema operativo baseado em Linux, de controlo de máquinas CNC de código aberto, que transforma um computador convencional numa plataforma de controlo em tempo real, permitindo a operação em fresadoras, tornos,

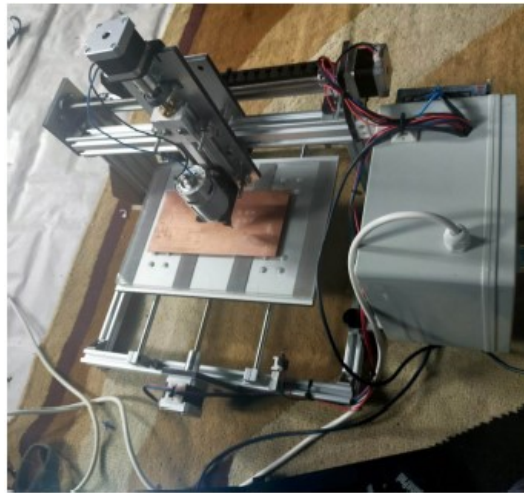


Figura 3.3: Impressora de PCB [37]

impressoras 3D, entre outras operações. Desenvolvido para oferecer flexibilidade, escalabilidade e alto desempenho, o LinuxCNC suporta diversas arquiteturas de *hardware* e interfaces, sendo utilizado tanto em ambientes acadêmicos como industriais. Consegue suportar até 9 eixos, fornecendo diversos HMI à escolha, interpretador de código G, sistema de planeamento do movimento e operação de eletrónica de baixo nível.

A operação de controlo pode ser efetuada em malha fechada com servos ou em malha aberta com motores de passo. Este sistema também pode operar em sistemas de movimento não cartesiano, trabalhando com hexápodes (plataforma Stewart) ou sistemas com juntas rotativas como os robôs PUMA ou SCARA [35].

Em 2019, estudantes sérvios [38] apresentaram uma mini-CNC de três eixos controlada por LinuxCNC, destinada a criar um protótipo educacional, fabricada com componentes facilmente disponíveis e de baixo custo, podendo ser uma alternativa viável para soluções industriais. A visão destes estudantes passa por explicar e indicar como construir estas máquinas educativas para introduzir o ensino de máquinas CNC nas escolas e faculdades, por serem equipamentos mais acessíveis.

O *hardware* base deste trabalho utilizou *drivers* de *Compact Disc/Digital Versatile Disc* (CD/DVD) para criar os eixos, interligados a uma Raspberry, que fará a interface dos motores, sendo a ferramenta uma caneta. O espaço de trabalho da mesa final foi de 35 x 35 x 35 mm, conforme visualizado na Figura 3.4 [38].

A arquitetura do LinuxCNC, demonstrada na Figura 3.5, é composta por quatro módulos principais:

- EMCOT: responsável pela interpolação em tempo real dos movimentos programados, com base em modelos cinemáticos diretos e inversos. Controla diretamente os motores dos eixos;

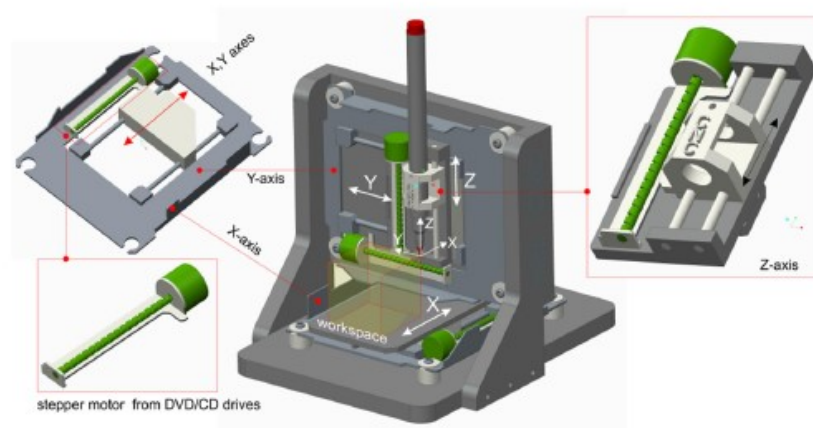


Figura 3.4: Modelo CAD da mini CNC [38]

- EMCIO: gere as operações periféricas, como comandos do *spindle*, sistema de refrigeração e sinais auxiliares, sem interferir no controlo direto dos motores;
- EMCTASK: interpreta os comandos *G-code* e coordena os módulos EMCOT e EMCIO. Atua como elo entre processos em tempo real e não real, utilizando *buffers* de memória partilhada;
- *Graphical User Interface* (GUI): interface de utilizador para controlo, visualização da trajetória e execução de código. Existem várias opções de interface como a GUI Axis, TkLinuxCNC, MiniGUI, Gmoccapy, entre outros.

A arquitetura ainda inclui o *Hardware Abstraction Layer* (HAL), que permite mapear sinais entre o *hardware* e o *software*, simplificando a integração de novos componentes e modelos cinemáticos.

Foi realizado um projeto em simulação virtual, que demonstra como configurar a parte virtual e o *hardware*. Embora não tenham sido demonstrados os testes efetuados, foi desenhada a imagem da águia sérvia como forma de teste. Com a apresentação deste trabalho, visualizado na Figura 3.6, é mostrada uma possível proposta para ser introduzida nas faculdades e em ambientes de ensino, ou em empresas que necessitam de mini-máquinas CNC [38].

Com uma noção sobre o LinuxCNC apresentada por este trabalho e por vários outros existentes, pode-se considerar este *software* como uma solução mais complexa do que o GRBL, mas facilmente versátil para a introdução de formatos diferentes de máquinas, constituindo uma forte alternativa para quem procura um sistema de controlo robusto e adaptável.

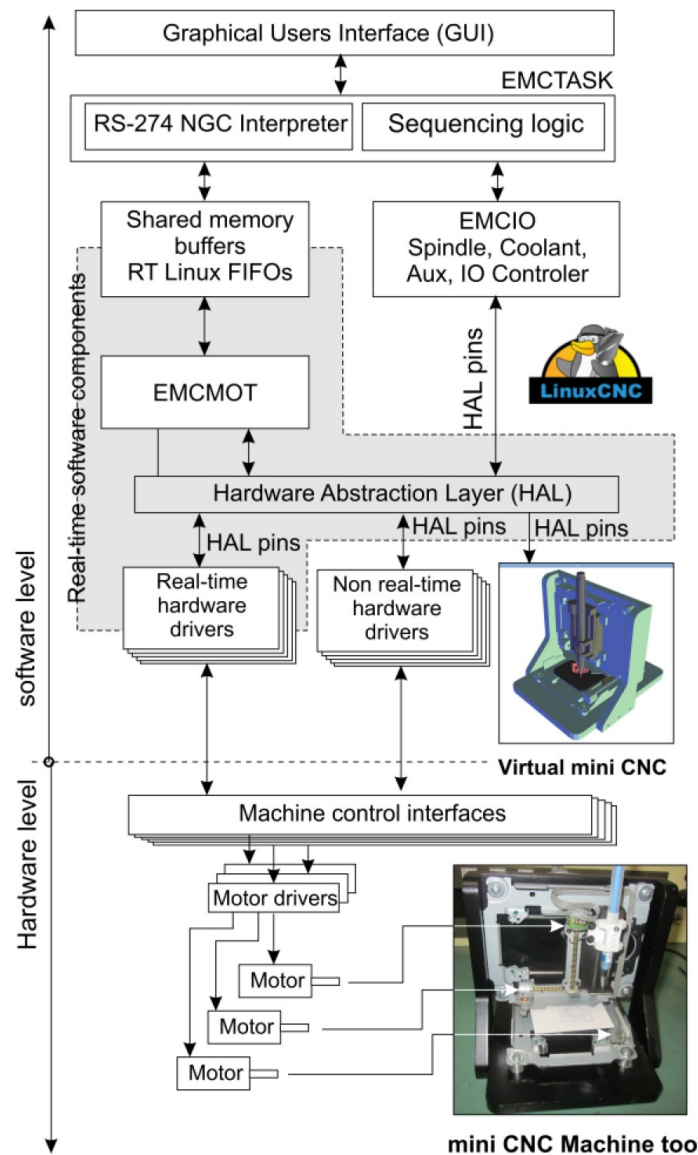


Figura 3.5: Arquitetura do projeto da mini-CNC [38]

3.2 Trabalho com *Industrial PC*

Para além dos projetos vistos anteriormente, existem trabalhos que utilizam IPC, aplicados na produção industrial, devido ao número de peças a processar ou à dimensão da peça. Vários fabricantes de IPC desenvolveram soluções para adaptar um IPC a uma CNC. Para tal, serão direccionados estes dois trabalhos para o funcionamento do CNC do TwinCAT. No primeiro caso será mencionado um trabalho onde a comunicação entre controlador e *drivers* utiliza o protocolo *CANopen over EtherCAT* (CoE), enquanto o segundo caso explora uma CNC que utiliza funções inteligentes do TwinCAT e .NET *Framework*.

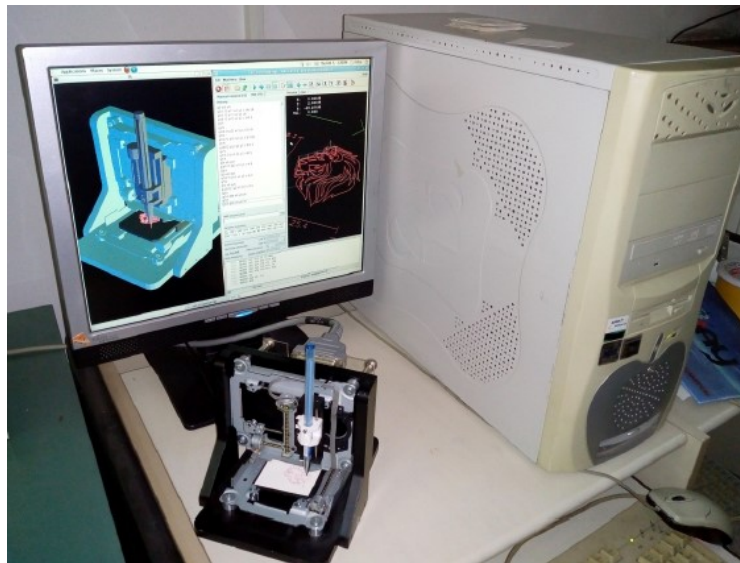


Figura 3.6: Mini-CNC [38]

3.2.1 Controlo do TwinCAT numa máquina CNC com CANopen sobre EtherCAT

O seguinte trabalho que será estudado envolve uma máquina CNC controlada por TwinCAT sobre CoE [39]. O objetivo destes estudantes da Eslováquia foi controlar máquinas CNC antigas usando TwinCAT, em que os barramentos dessas máquinas operam exclusivamente com o protocolo CANopen. Por essa razão, foi necessário o uso de dois protocolos de comunicação e verificar se a solução de controlo de eixos sobre CoE é viável.

A ideia inicial partia de uma mesa de três eixos, com a dimensão de 1250 x 650 x 53 mm, não automatizada, contendo *drivers* e servomotores da Lenze, redutores e correias de transmissão para cada eixo, conforme visualizado na Figura 3.7. Foram apresentadas várias hipóteses, sendo a mais simples conectar todos os dispositivos ao EtherCAT. No entanto, devido à limitação dos *drivers*, que não possuem suporte para EtherCAT, foi necessário procurar uma solução de comunicação [39].

A solução adotada foi o uso do protocolo de comunicação *CANopen over EtherCAT* (CoE). Este protocolo permite a compatibilidade da comunicação CANopen com EtherCAT. Em resumo, o EtherCAT assume a camada de aplicação da comunicação CANopen e fornece a interface e perfil de aplicação para dispositivos CANopen.

Para a execução da tarefa, foi utilizado um computador convertido para PC industrial, conectado ao módulo de comunicação EtherCAT, que, por sua vez, se liga aos *drivers* via CANopen, como demonstrado na Figura 3.8. Não foram mencionadas as bibliotecas utilizadas para a execução referida.

De forma a testar a solução obtida, foram efetuadas duas trajetórias: a primeira

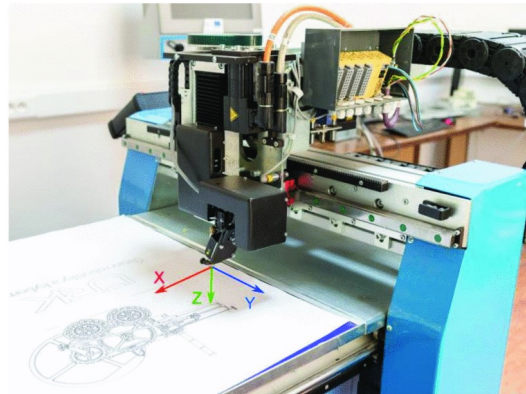


Figura 3.7: CNC proposta para o trabalho do TwinCAT CoE [39]

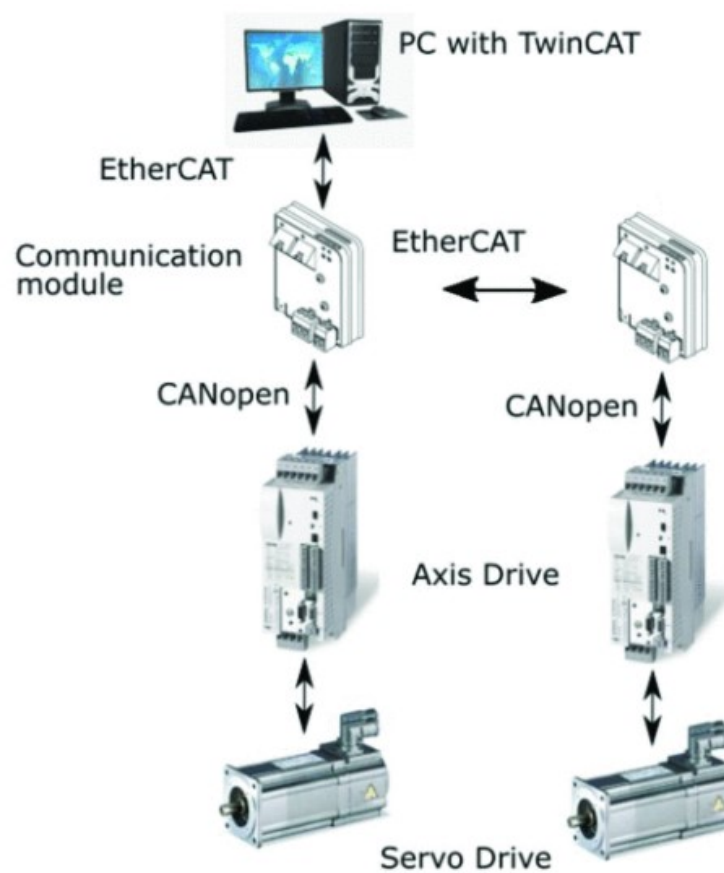
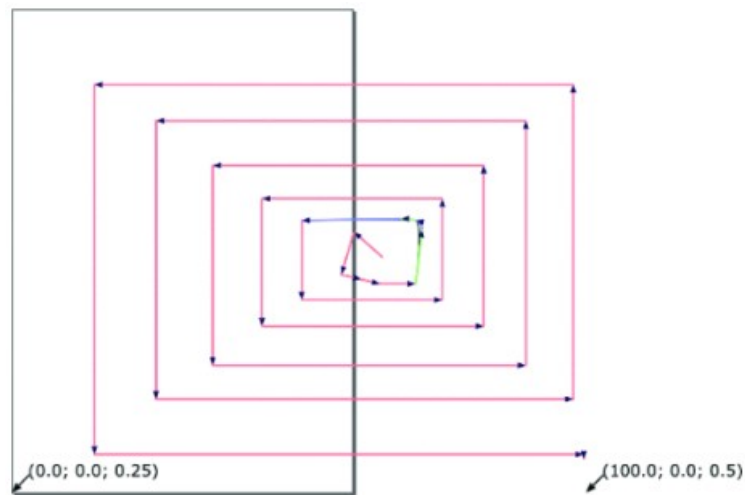
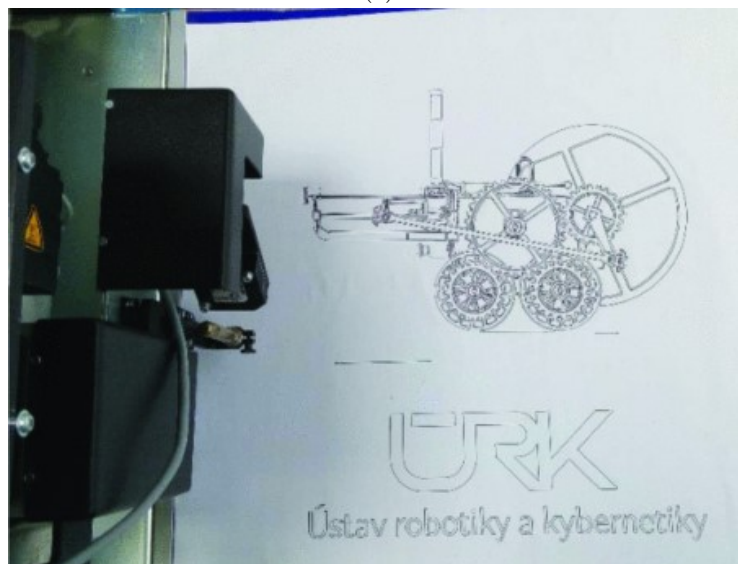


Figura 3.8: Arquitetura de controlo e comunicação do trabalho do TwinCAT CoE [39]

constitui um conjunto de linhas retas, com o objetivo de analisar a posição, a velocidade e o erro de seguimento, enquanto a segunda foi realizada com uma imagem em CAD para verificar o comportamento, conforme demonstradas na Figura 3.9. Foi verificado um desvio mínimo de posição de 0,08 mm, mantendo a velocidade de corte desejada [39].



(a)



(b)

Figura 3.9: Testes efetuados para a execução do projeto do TwinCAT CoE [39]: a) Teste 1 - verificação de posições b) Teste 2 - análise de desenhos complexos

3.2.2 Sistema CNC para Funções Inteligentes com base em TwinCAT e .NET Framework

Um grupo de estudantes chineses [40] propôs um sistema CNC baseado em computador, que incorpora várias funções inteligentes utilizando TwinCAT e .NET. O sistema NCK apresenta funcionalidades em tempo real e, com a aplicação numa HMI, foi construída uma CNC de 5 eixos com interpolação, monitorização da vibração do fuso e controlo do consumo.

O .NET Framework é um componente interno do Windows que suporta a criação e operação de aplicações. Este fornece um ambiente de programação orientado a

objetos consistente, independentemente do estado de execução do código objeto e da sua distribuição pela *Internet*.

O artigo menciona três subsistemas que permitem a execução de um sistema CNC aberto, capaz de realizar tarefas em tempo real e não real na mesma CPU, conforme ilustrado na Figura 3.10:

- sistema HMI, que integra múltiplas funções, como a interpretação de códigos NC, processamento de dados para avaliação do estado dos componentes e transferência de informações da máquina-ferramenta;
- sistema NCK que é um sistema em tempo real aberto e escalável baseado na plataforma TwinCAT, fornecendo interpoladores para curvas, linhas, círculos e outros;
- sistema de percepção periférica, que recolhe as informações não captadas pela NC e pelo sistema de servomotores, como a vibração do eixo, consumo de energia e sinais sonoros na maquinação.

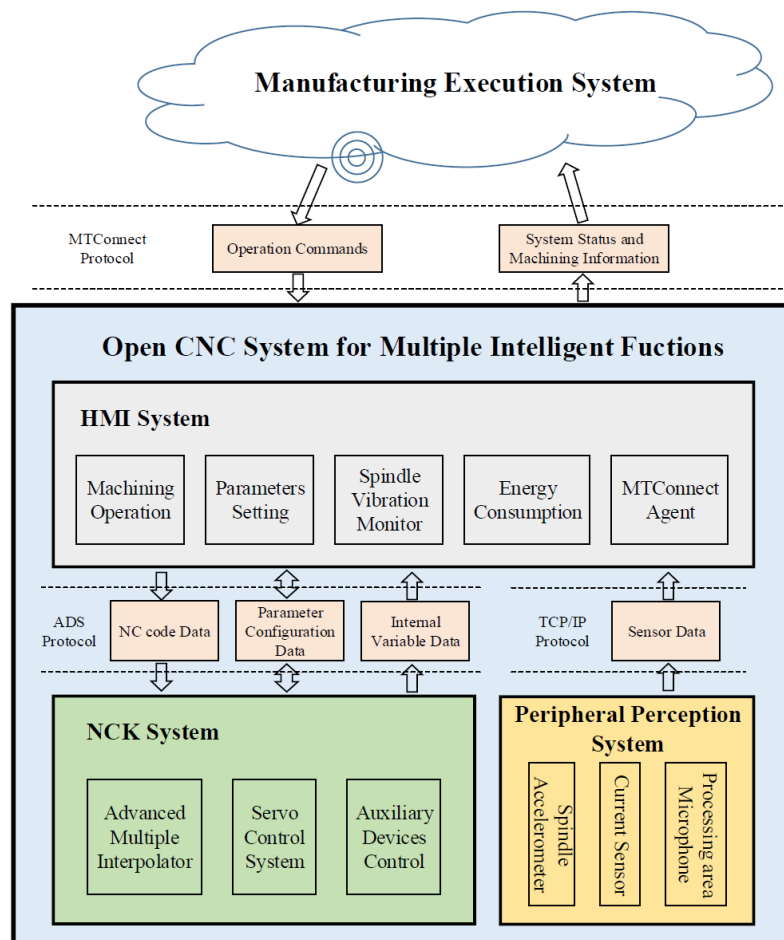


Figura 3.10: Arquitetura de sistema aberto das CNC [40]

O NURBS, um interpolador utilizado para a maquinação de alto desempenho, apresenta uma limitação que impede a execução em tempo real do planeamento de velocidades de avanço. Para contornar este problema, o planeamento da velocidade de avanço foi implementado no sistema .NET HMI. Para tal, foram definidos dois *buffers* de anel de dados NC, um no lado NC e outro no lado HMI, permitindo assim a troca de dados, como visualizado na Figura 3.11.

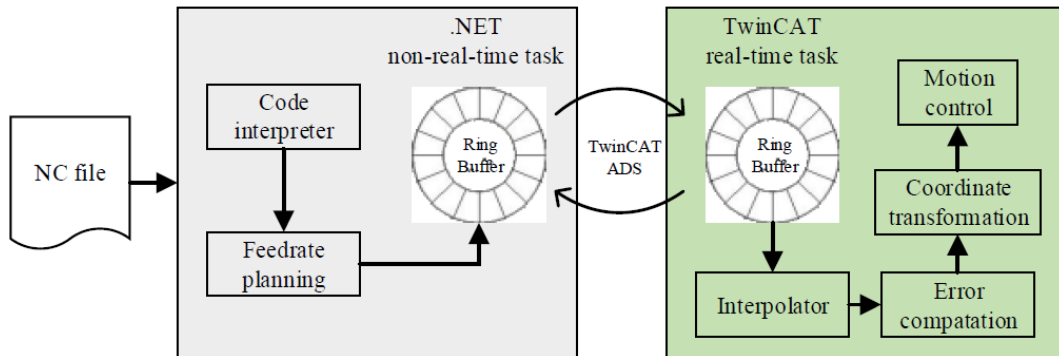


Figura 3.11: Troca de dados NC entre os lados HMI e TwinCAT [40]

A estrutura do sistema é dividida em duas partes: o lado do sistema NCK e o sistema de percepção periférica. No lado do sistema NCK encontra-se o *hardware* físico, onde foi implementado um IPC da *Beckhoff*, que comunica via EtherCAT com o sistema, incluindo os terminais de entrada e saída de dados e os controladores AX5000, que controlam os servomotores AM 8000, todos da *Beckhoff*, conforme ilustrado na Figura 3.12.

O lado do sistema de percepção periférica é dividido em duas partes, conforme visualizado na Figura 3.13. A primeira parte inclui um transformador de tensão e um chip de medição de energia trifásica ATT7022B, que permite medir as correntes e tensões de entrada, assim como a potência total da máquina-ferramenta. Estes dados são enviados via Wi-Fi através de um STM32. A segunda parte utiliza um microfone de matriz e um acelerómetro tridimensional para captar sinais de som e vibração, que são enviados para o sistema HMI.

Para validar o sistema, o objetivo foi a maquinação de uma peça com cavidade, paredes finas e inclinadas, conforme visualizado na Figura 3.14. Este teste teve como finalidade avaliar a precisão, a estabilidade e a capacidade de reprodução de geometrias complexas pela CNC [40].

3.3 Conclusão dos Trabalhos Relacionados

Ao analisar os trabalhos apresentados, pode-se confirmar a grande diversidade de máquinas CNC com diferentes modos de controlo.

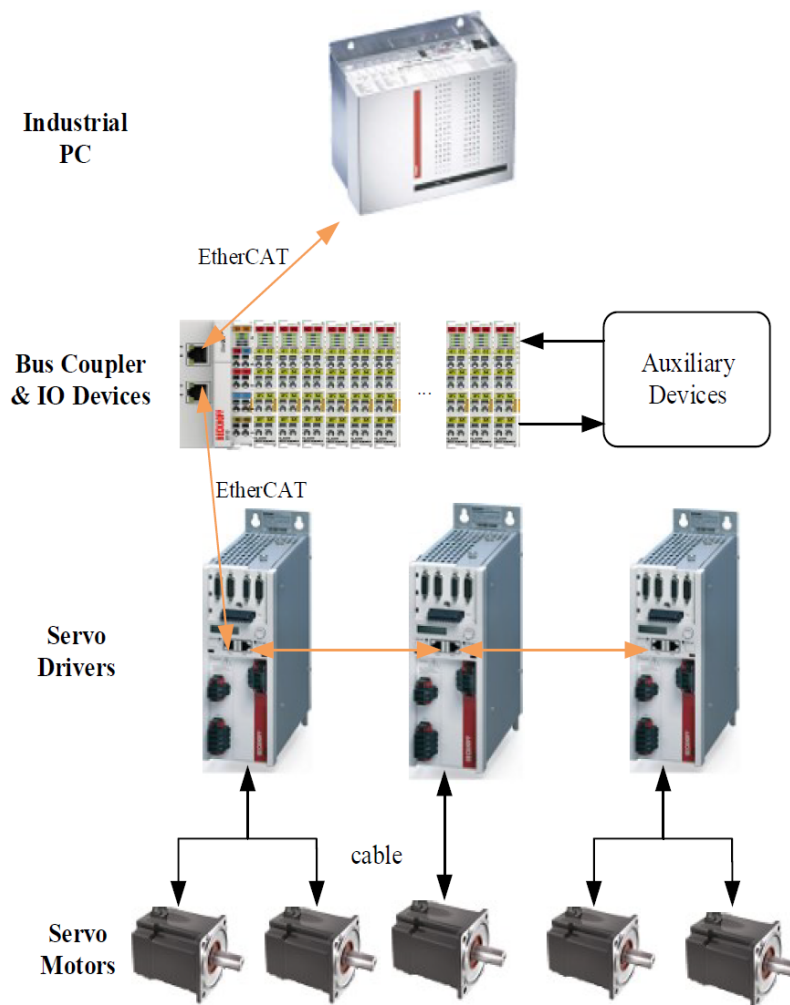


Figura 3.12: Estrutura do *hardware* no sistema NCK do trabalho TwinCAT .NET [40]

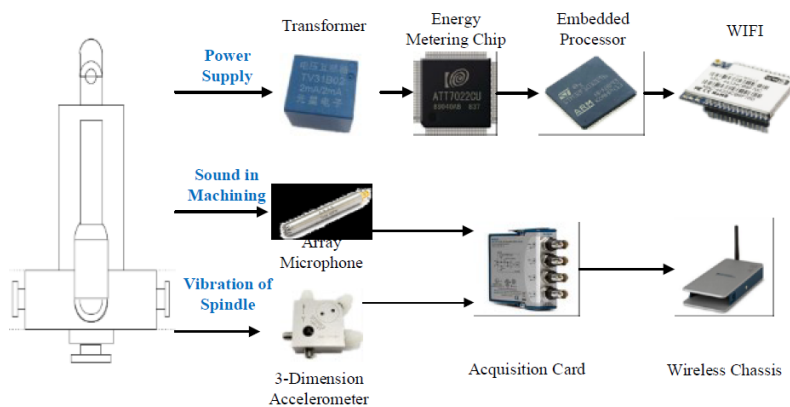


Figura 3.13: Estrutura do *hardware* no sistema periférico do trabalho TwinCAT .NET [40]

Para quem procura uma solução económica, fácil de montar e com precisão média, o GRBL destaca-se, sendo aplicável a projetos *Do It Yourself* (DIY).

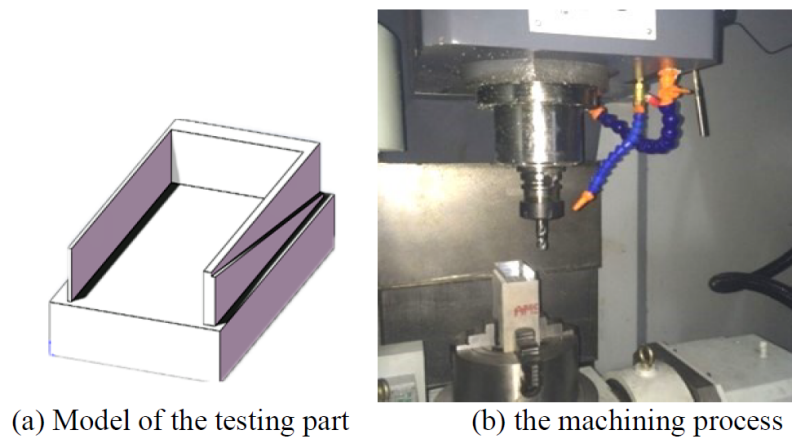


Figura 3.14: Teste prático do trabalho TwinCAT .NET [40]

Para uma solução mais robusta, mas ainda assim económica, e com boa precisão, o LinuxCNC é uma opção interessante para a introdução de máquinas para pequenas indústrias ou para o ensino.

Para quem busca uma opção mais industrial, os sistemas com IPC, apesar de terem um custo mais elevado, oferecem uma precisão elevada e tornam-se viáveis, como contém uma facilidade de adicionar novas funcionalidades.

Destaca-se o projeto da mini-CNC como uma excelente hipótese para um projeto final de licenciatura, tanto para os cursos de Engenharia Eletrotécnica como para os de Engenharia Mecânica, com o intuito para o ensino de CNC, possibilitando que o ensino de *G-code* seja transferido para fora da simulação antes de ser aplicado em máquinas CNC industriais.

Este capítulo contribuiu assim, para a consolidação essenciais à definição da arquitetura do projeto desenvolvido nesta dissertação, fornecendo também referências úteis para a realização de testes e validações na fase de implementação.

Capítulo 4

Arquitetura do Sistema

Neste capítulo, serão abordados dois assuntos: o estado da máquina antes de ser indicada para este projeto e a arquitetura proposta para a dissertação.

Numa primeira fase, será mencionado o estado no qual a máquina se encontrava, qual era a funcionalidade a que estava destinada e o estado em que foi deixada.

Após a análise da máquina em questão, será proposta uma nova arquitetura, onde, numa primeira instância, será apresentado um esquema de *hardware* a ser montado, os equipamentos a serem utilizados e uma breve explicação da sua funcionalidade. Por fim, será discutido o *software*, a razão do uso de cada um e um breve resumo do seu funcionamento.

Além disso, serão inseridos alguns fluxogramas de forma a tornar mais gráfica a lógica introduzida para o funcionamento da máquina.

4.1 Arquitetura Encontrada

A mesa que será alvo do trabalho desta dissertação foi projetada pelo departamento de mecânica da Gislotica, e o trabalho elétrico realizado antes desta dissertação foi para a execução de um projeto final de licenciatura, que visava medir o perímetro de um talão, elemento de arame de elevada resistência que contém a função manter o pneu preso ao aro da roda e garante a estanquicidade lateral e direcional do mesmo, por meio de uma nuvem de pontos, o qual não foi completado fisicamente [41].

Em termos mecânicos, as dimensões da mesa são 1,6 x 2,5 x 1,6 m, sendo que o espaço útil para trabalho ronda os 1 x 2 x 0,4 m. A mesa foi projetada para utilizar

um *laser* como ferramenta, dispensando o eixo Z, como verificado na Figura 4.1. A mesa foi construída com base em sistemas de transmissão por correias.

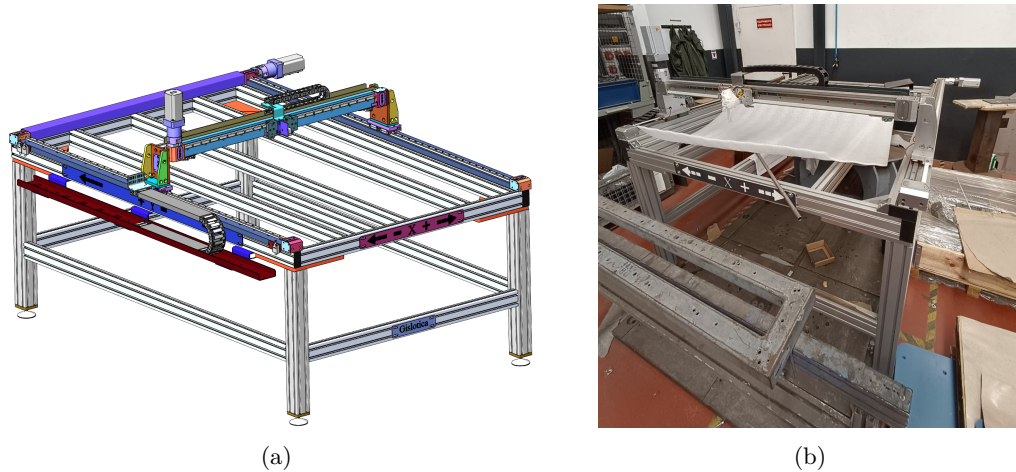


Figura 4.1: Mesa XY: a) esboço da mesa [41] b) estado da mesa antes da adaptação - o motor no eixo X encontra-se no lado esquerdo montado da estrutura móvel, enquanto o motor Y encontra-se no lado direito, no fundo da mesa. Os fins de curso encontram-se nos limites, onde no canto esquerdo na frente da mesa encontram-se, em cada eixo, os dois fins de cursos correspondentes ao limite e *home*

Em termos elétricos, na mesa, e conforme verificado na Figura 4.1, encontravam-se apenas dois motores de passo (eixo X e Y) e seis fins de curso, sendo que três correspondiam a um dos eixos (dois como limites e um como *Home*), sem nenhuma ligação para interligar ao quadro. À parte, existe um quadro que contém o sistema de controlo para esta mesa, o qual integra o circuito de alimentação, segurança, controlo e os *drivers* dos motores, conforme verificado na Figura 4.2. É importante mencionar que, ao realizar uma revisão do esquema elétrico até então, detetou-se a falta de alguns componentes, provavelmente utilizados em outros projetos na empresa.

A arquitetura apresentada pelo projeto anterior a esta dissertação, conforme visualizado na Figura 4.3, evidencia como foi delineado o sistema. O controlo era efetuado por meio de um PLC e comunicava com os *drivers* via CANbus, realizando o controlo dos *drivers* do motor, onde os mesmos atuavam. Após se encontrar na posição, o sistema efetuava o registo pretendido pelo *laser* [41].

Na Tabela 4.1, encontram-se todos os componentes encontrados no conjunto quadro e mesa, assim como as respetivas quantidades.

No relatório do respetivo aluno, são mencionados alguns atrasos, o que não permitiu a integração do *hardware* do trabalho final e a realização de testes por meio de simulação, com os testes realizados no *hardware* existente à parte [41].



Figura 4.2: Estado do quadro antes da adaptação

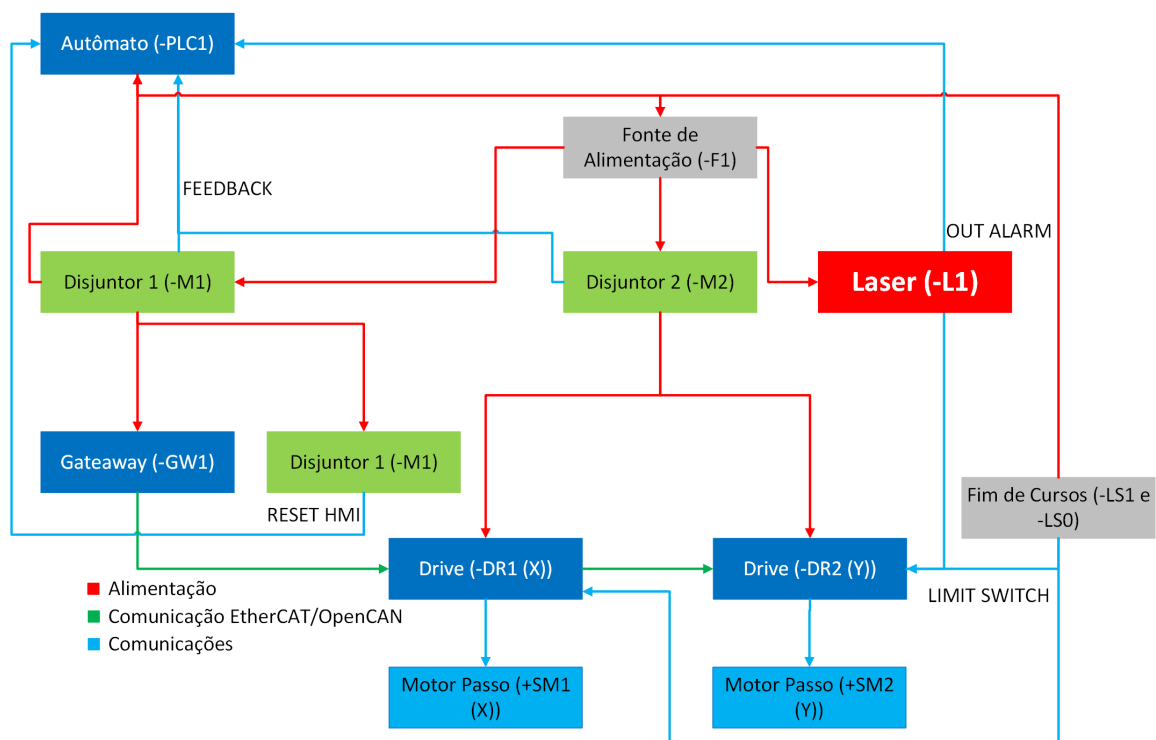


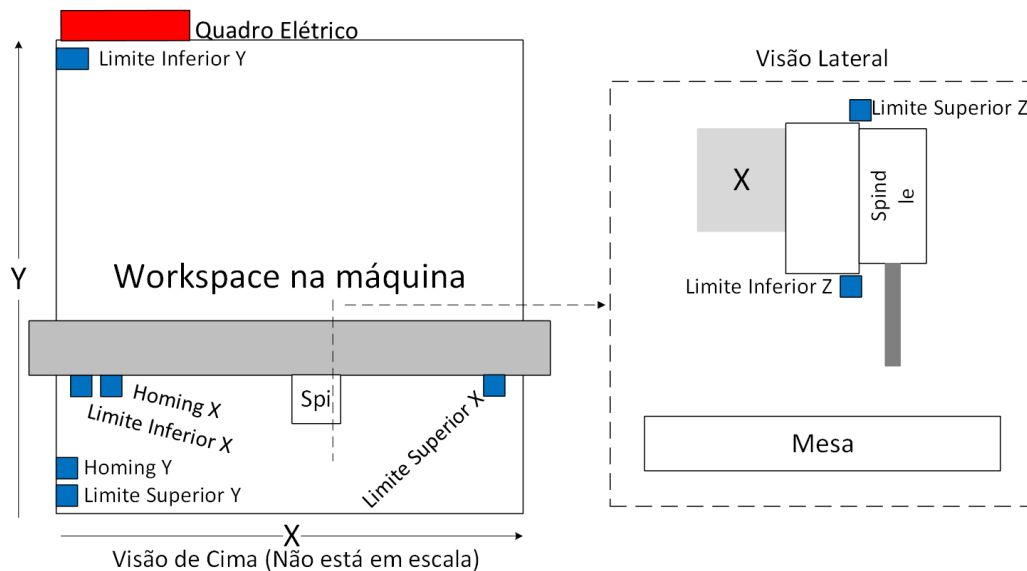
Figura 4.3: Arquitetura da proposta anterior [41]

Tabela 4.1: Tabela de quantidades dos componentes existentes no projeto anterior

Quantidade	Produto
1	Siemens Sentron 3LD2003-1TL53
1	Siemens Differential Switch
1	Circuit Breaker (Allen Bradley 2P, 2 A, Type C Curve, Cutting Power 10 kA)
1	Mico 4.6
1	Botão Emergência 2 Canais mais dois botões
1	Relé de Emergência PILZ PNOZ XV2
1	X-gateway CANopen EtherNet/IP
1	PLC Allen Bradley CHT 1769-L18ER-BB1B
2	Drivers Motor <i>Stepper</i> Festo CMMS-ST-C8-7-G2
2	Motor <i>Stepper</i> Festo EMMS-ST-87-S-SEB-G2
1	Laser Keyence-IL-065
6	Fim de Curso Siemens 3SE5-423-0CD20-1EB1

4.2 Arquitetura Proposta

Após uma análise detalhada do circuito e do trabalho realizado no projeto anterior, e ao examinar atentamente as especificações necessárias para executar uma CNC, chegou-se a um *layout* da mesa e a uma arquitetura de controlo proposta, demonstradas nas Figuras 4.4 e 4.5, respetivamente.

Figura 4.4: *Layout* da mesa para a CNC

O *layout* da mesa segue o padrão estabelecido pelo projeto anterior, de modo a minimizar as alterações à mesa. O movimento do eixo X é realizado pelo motor acoplado à estrutura que movimenta, enquanto o motor Y está acoplado à parte física da mesa, proporcionando o movimento do mecanismo. Os movimentos são

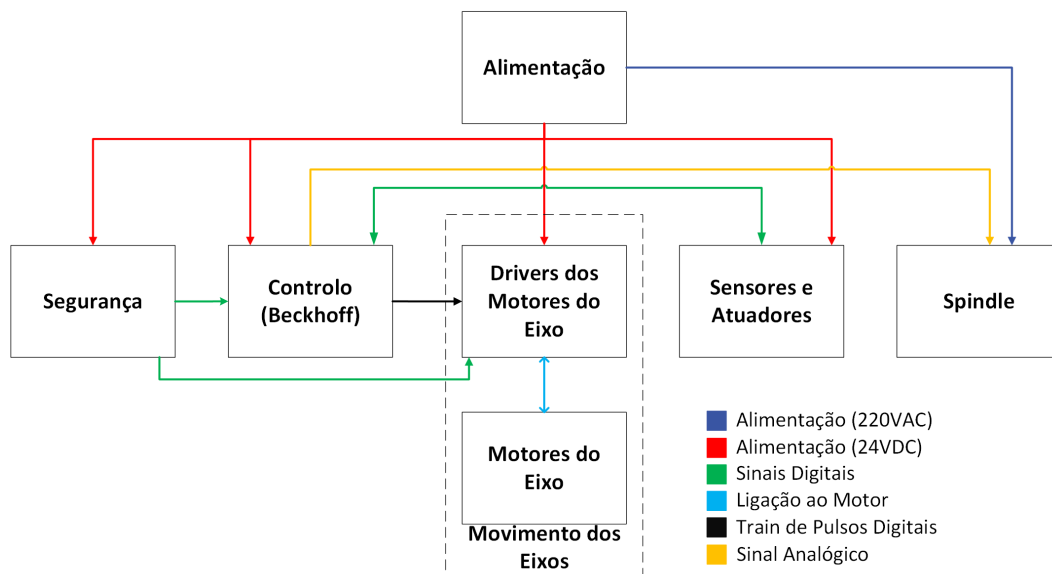


Figura 4.5: Arquitetura de controle proposta para a CNC

realizados por meio de uma caixa redutora interligada a uma polia dentada, que aciona a correia e, conseqüentemente, o patim. Acoplado ao cabeçote, há outro motor que fará o movimento do mecanismo responsável por levantar e descer a ferramenta.

Ao analisar a arquitetura proposta, pode-se dividi-la em seis grupos-base: Alimentação, Segurança, Controle, Movimento dos Eixos, *Spindle* e Sensores e Atuadores.

Além da reestruturação da arquitetura fornecida na Seção 4.1, existem grupos que mantiveram os seus respectivos componentes, enquanto outros sofreram alterações ou foram introduzidos, como o *spindle*.

O grupo da alimentação compreende todos os componentes que fornecem energia, evitam sobrecargas e curto-circuitos, e dividem e limitam a corrente a ser fornecida para cada grupo.

O grupo de segurança representa todos os equipamentos relacionados com a segurança do sistema, mais concretamente quando acionado o botão de emergência.

O grupo de controle visa a exploração de todos os grupos e o seu controle, funcionando como o cérebro da CNC e monitorizando todos os sinais que recebe.

O grupo dos motores do eixo está dividido em duas partes: os *drivers* e os motores. Os *drivers* encontram-se na caixa de controle, enquanto os motores estão montados na mesa. Este grupo explora a comunicação com o controle para realizar os movimentos pretendidos.

O grupo de sensores e atuadores refere-se a todos os sensores e atuadores que serão integrados no projeto para proporcionar um controle mais eficaz e adequado.

Por fim, o grupo do *spindle* controla a ferramenta da CNC, com foco principal na velocidade.

Estes grupos serão explorados na Secção 4.3, dedicada ao *hardware*, enquanto o controlo e *softwares* utilizados serão mencionados na Secção 4.4.

4.3 Hardware

Nesta secção, serão explorados os componentes de *hardware* utilizados, detalhando os vários grupos e os respetivos componentes que os integram. No final, será apresentada uma tabela com todos os componentes. No Anexo B serão introduzidos os esquemas elétricos propostos para esta dissertação.

4.3.1 Alimentação

O grupo de alimentação consiste em toda a parte necessária para alimentar e proteger o circuito. Relativamente à arquitetura anterior, foram reintroduzidas a fonte de alimentação e um interruptor eletrónico, mencionados na arquitetura anterior, mas retirados para outros testes. Assim, este grupo é composto por uma chave seccionadora de corrente, um interruptor diferencial residual, um disjuntor magnetotérmico, uma fonte de alimentação e dois disjuntores eletrónicos, um de quatro canais e outro de dois canais.

Chave Seccionadora de Corrente

Para desligar a corrente sem ter de abrir o quadro durante a manutenção, utiliza-se uma chave seccionadora de corrente. Reaproveitado do circuito anterior, foi utilizado o modelo Sentron 3LD2003-1TL53 (Figura 4.6) da Siemens [42, 43], que cumpre a respetiva função.

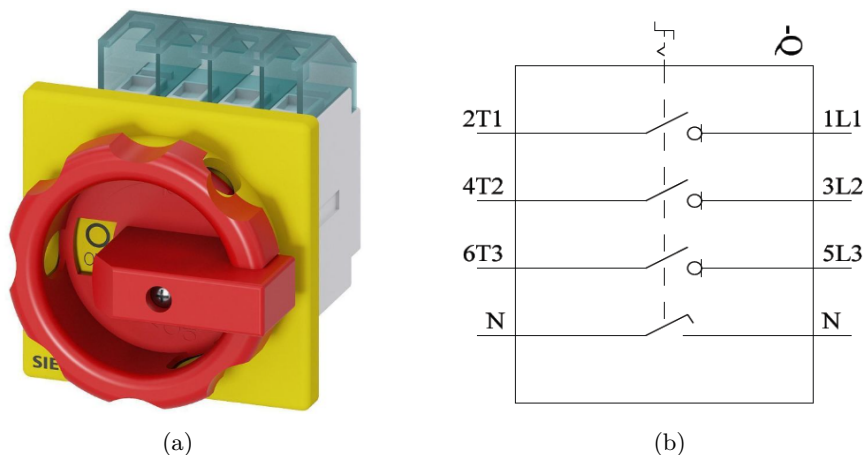


Figura 4.6: Chave seccionadora de corrente Siemens Sentron [42, 43]:
a) Componente b) Esquema elétrico

Interruptor Diferencial Residual

O interruptor diferencial residual utilizado é o Siemens 5SM33116 (Figura 4.7), um dispositivo de proteção que deteta fugas de corrente para a terra, protegendo assim as pessoas contra choques elétricos.

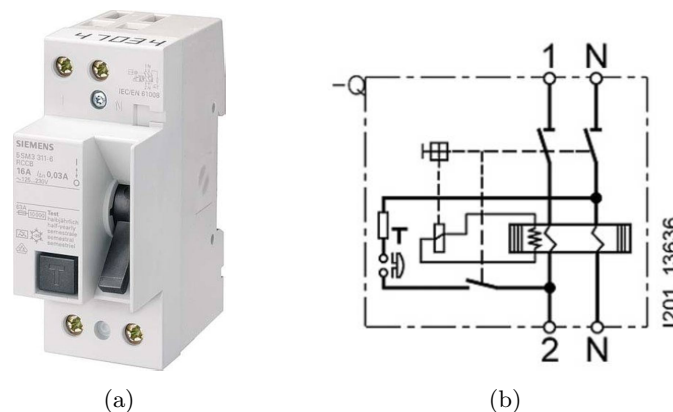


Figura 4.7: Interruptor diferencial residual [42, 44]: a) Componente
b) Esquema elétrico

Este dispositivo atua sempre que deteta uma corrente diferencial superior a um determinado limiar, geralmente fixado em 30 mA para proteção de pessoas, desligando automaticamente o circuito. A sua principal função é garantir a segurança dos utilizadores finais, intervindo em situações de isolamento comprometido, contactos indiretos ou falhas de isolamento em equipamentos elétricos.

Este tipo de equipamento não oferece proteção contra sobrecargas ou curtos-circuitos, sendo, por isso, imprescindível a sua associação a um disjuntor magnetotérmico.

A utilização deste interruptor está consoante as exigências das normas de instalações elétricas, nomeadamente a norma IEC 60364 (normas de instalações elétricas de baixa tensão) e 61439 (projeto e construção de quadros elétricos industriais), que recomenda a proteção diferencial como medida essencial de segurança [45, 46, 42, 44].

Disjuntor Magnetotérmico

O disjuntor Allen-Bradley 1492-D2 (Figura 4.8) é um dispositivo magnetotérmico de dois polos (2P), com corrente nominal de 2 A e curva de disparo tipo C. Este equipamento é projetado para a proteção de circuitos elétricos contra sobrecargas e curtos-circuitos, atuando automaticamente em caso de corrente superior ao valor nominal permitido.

A curva C indica que o disjuntor permite picos de corrente típicos de cargas indutivas (como motores ou relés), sem disparar prematuramente, mas ainda assim



Figura 4.8: Disjuntor magnetotérmico [47]

oferece uma proteção eficaz contra curtos-circuitos, com uma capacidade de interrupção de 10 kA. Este valor torna-o adequado para aplicações industriais ou em quadros elétricos com níveis de curto-circuito significativos.

Em conjunto com o interruptor diferencial residual, o disjuntor Allen-Bradley 1492-D2 assegura uma proteção completa do circuito: enquanto o disjuntor protege os condutores e os equipamentos contra sobrecorrentes, o diferencial protege os utilizadores contra correntes de fuga perigosas. Esta combinação é essencial para garantir a conformidade normativa e a segurança operacional da instalação elétrica [45, 46, 47, 48].

Fonte de Alimentação

A fonte de alimentação utilizada é a SITOP PSU100S 6EP1334-3BA00 (Figura 4.9), um dispositivo modular destinado a aplicações industriais, fornecendo uma tensão de saída contínua e estabilizada de 24 V DC com corrente nominal de 10 A. Está desenhada para operação em sistemas monofásicos ou bifásicos com uma ampla gama de tensões de entrada.



Figura 4.9: Fonte de alimentação Siemens SITOP PSU100S [42]

Trata-se de uma fonte com regulação ajustável da tensão de saída (entre 24 V e 28,8 V), eficiência típica de 87%, e capacidade para operações em paralelo com até dois módulos para aumento de potência. Oferece elevada fiabilidade em condições industriais, com resistência a curto-circuito, proteção contra sobrecarga e capacidade de operação contínua, mesmo com picos de corrente de até 30 A por períodos curtos.

Possui proteção interna contra sobretensões (até 1300 V pico), além de conformidade com normas de segurança, tornando-a adequada para ambientes exigentes. A montagem é efetuada sobre calha DIN padrão (EN 60715), facilitando a integração em quadros elétricos industriais modulares [42, 49].

Disjuntores Eletrónicos

Os disjuntores eletrónicos, mais conhecidos como *Murrelektronik Intelligent Current Optimizer* (MICO) são módulos de monitorização e distribuição de corrente para circuitos DC, especialmente para sistemas industriais de 24 V.

Existem diversos modelos, mas o funcionamento é o mesmo. O primeiro número corresponde ao número de canais disponíveis no respetivo MICO, enquanto o número seguinte corresponde à corrente máxima associada.

Estes componentes permitem distribuir a corrente de forma inteligente, possibilitando que cada canal tenha um máximo de corrente configurável, detetando sobrecargas e curto-circuitos, com diagnóstico visual e comportamento seletivo, ou seja, se um canal entrar em curto-circuito, somente aquele canal é desligado [50].

Para este trabalho, serão utilizados dois MICO (Figura 4.10), um de quatro canais e outro de dois canais, com um máximo de corrente de 6 A por canal, contabilizando seis canais disponíveis. Para alimentar os restantes grupos, a distribuição será feita da seguinte forma [51, 52]:

- segurança e controlo;
- alimentação dos *drivers*;
- 3x alimentação dos motores de eixo (um canal para cada).

4.3.2 Segurança

O grupo de segurança refere-se ao mecanismo que permite parar instantaneamente os motores. Os componentes deste grupo mantêm-se relativamente ao projeto anterior, sendo eles um relé de segurança e um botão de emergência.

Relé de Segurança

O relé de segurança PNOZ XV2 (Figura 4.11), fabricado pela Pilz, é um dispositivo essencial em aplicações industriais onde é necessário garantir a interrupção segura

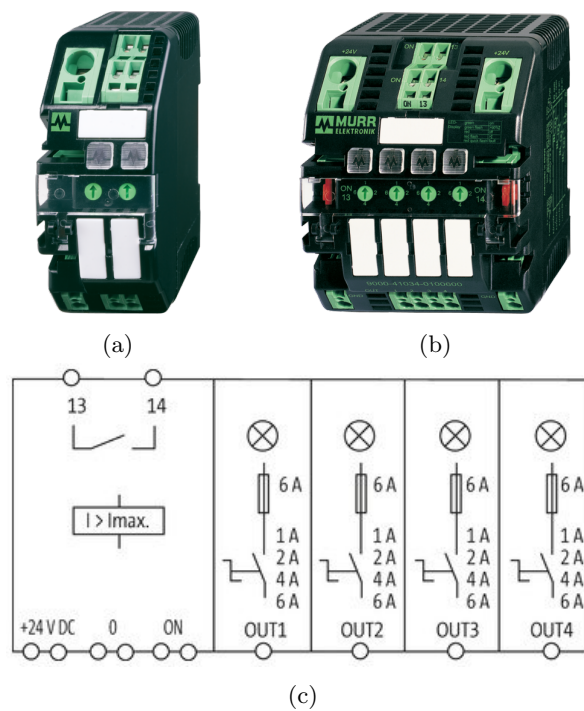


Figura 4.10: Disjuntor eletrônico MICO [50, 52]: a) MICO 2.6 b) MICO 4.6 c) Esquemático MICO 4.6

de circuitos em situações de emergência. Este relé é utilizado, por exemplo, com botões de paragem de emergência ou interruptores de portas de segurança, oferecendo uma solução redundante e auto-monitorizada para aplicações críticas de segurança funcional.

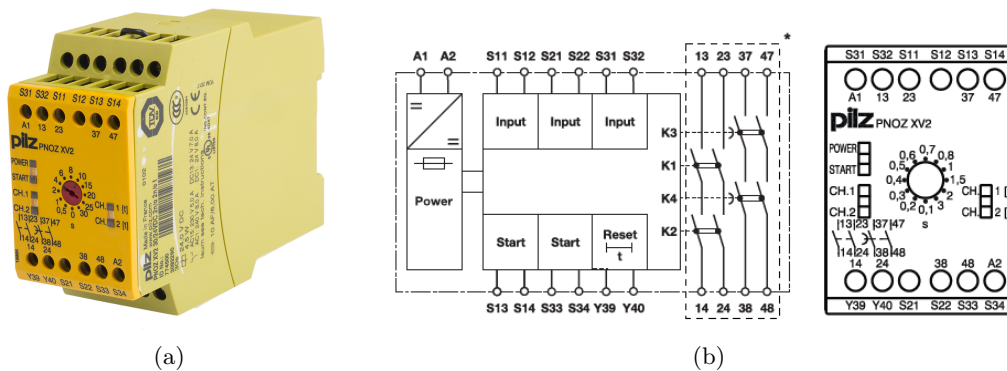


Figura 4.11: Relé de segurança Pilz PNOZ XV2 [53, 54]: a) Componente b) Esquema elétrico

A presença de relés de segurança como o PNOZ XV2 é fundamental para cumprir os requisitos das diretivas europeias de segurança em máquinas e respetivas normas.

O PNOZ XV2 distingue-se por:

- contém quatro contactos de segurança (dois imediatos e dois com atraso na desenergização);
- suportar modos de operação em canal simples ou duplo com detecção de curto-circuitos;
- permitir ajuste de tempo de atraso nos contactos temporizados, útil em sequências controladas de desligamento;
- ter monitorização automática dos contactos, assegurando que estes abrem e fecham corretamente em cada ciclo;
- fornecer sinalização por *Light Emitting Diode* (LED) do estado de alimentação, início de ciclo e estado de cada canal.

Este tipo de relé contribui para aumentar a segurança do operador e do equipamento, garantir a conformidade com normas internacionais de segurança, reduzir falhas não detetadas em sistemas de paragem de emergência e permitir uma resposta controlada e segura a falhas ou intervenções manuais no sistema.

Para o uso deste relé, serão utilizados os seguintes modos, com os esquemáticos de cada modo na Figura 4.12 [53, 55, 54]:

- E-STOP por dois canais;
- *start* após um impulso de sinal;
- sem *reset*.

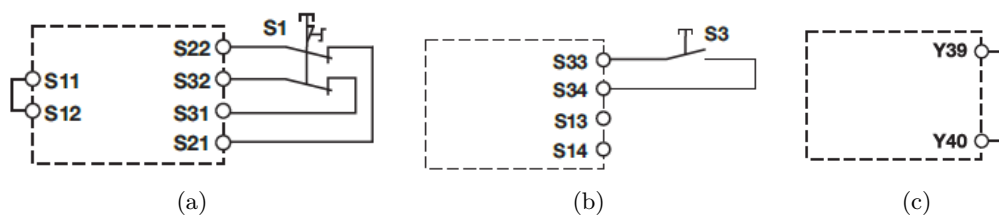


Figura 4.12: Modos que serão usados do relé de segurança PILZ [53, 54]: a) E-STOP com dois canais b) Start a partir de um impulso c) Sem *reset*

A razão para a utilização destes modos relaciona-se com o controlo exercido na aplicação, onde, ao detetar um *stop*, o movimento é imediatamente parado, sem que haja necessidade de um *reset*, e o início é dado por um sinal digital.

Botão de Emergência

Para implementar o botão de emergência, foi utilizada uma caixa de controlo que possui um botão de emergência comum, conforme mostrando na Figura 4.13, com dois canais, de forma a conectar-se ao relé de segurança.



Figura 4.13: Botão de emergência

A utilização do interruptor de emergência é projetada para uma zona de fácil acesso e próxima ao utilizador, permitindo a interrupção do sistema o mais rapidamente possível.

Além disso, ao longo da máquina, deveriam existir vários botões de emergência, segundo a norma ISO 13850 [56], que menciona a altura ideal a que o botão deve estar colocado face ao chão de fábrica e a necessidade de fácil acesso ao operador. Por ser uma máquina longa, deveria haver vários interruptores em diferentes acessos. No entanto, experimentalmente, será utilizado apenas um.

4.3.3 Controlo

O grupo do controlo é um dos que sofreu alterações em relação ao antigo projeto. Neste caso, foi retirado o PLC existente e introduzido um IPC. A razão para esta mudança deve-se à existência de um módulo para o TwinCAT, que será explorado na Secção 4.4, além da facilidade de integração de mais entradas e saídas, conforme as cartas necessárias para o projeto. Aqui serão abordados o IPC a utilizar e o método a ser utilizado, assim como as cartas empregues no projeto.

Controlador

A escolha do controlador recaiu sobre um computador com uma placa de rede capaz de comunicar via EtherCAT, permitindo o trabalho a nível da plataforma P90.

Os computadores da *Beckhoff* estão classificados em diferentes níveis de plataforma, que vão do P10 (básico) até ao P90 (habitual para computadores comuns), conforme visualizado na Figura 4.14. Os níveis são determinados pela configuração e pelas especificações técnicas do computador (incluindo o processador). Dependendo da aplicação a ser implementada, esta requer um nível mínimo de utilização [19].

No entanto, quanto maior for o nível da plataforma, mais elevado será o custo do licenciamento. Se o resultado final fosse destinado a uma produção em massa ou se o projeto fosse para venda, a solução necessária seria adquirir um IPC com uma licença mínima com o nível P60, conforme mencionado pela Beckhoff sobre o pacote de *software* a utilizar [19].

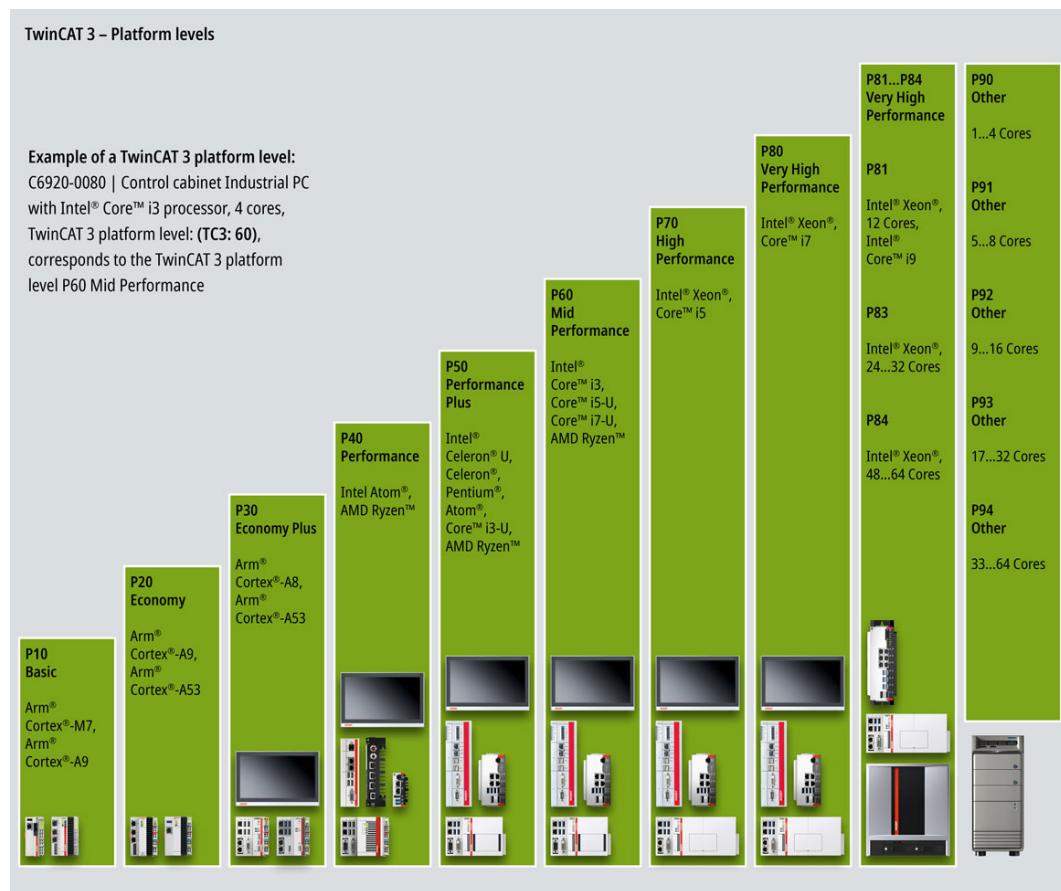


Figura 4.14: Níveis de plataforma da Beckhoff [19]

Contudo, dado que se trata de um projeto destinado a testes de códigos da Beckhoff para trabalhos acadêmicos futuros, bem como para testes internos de soluções mecânicas antes da aprovação da solução final ou para a utilização da máquina internamente, e considerando a utilização de licenças *trial* que são renováveis, optou-se por adaptar um computador que possua uma placa de rede capaz de comunicar via EtherCAT, uma vez que os computadores de desenvolvimento suportam todas as funcionalidades disponibilizadas pela Beckhoff.

Inicialmente, para testes básicos, como movimentos de eixos sem interpolação ou testes de *hardware*, foi utilizado o IPC CX9020, conforme visualizado na Figura 4.15. Este IPC da Beckhoff permite testar o funcionamento dos primeiros requisitos, permitindo avaliar a tecnologia existente [57]. No entanto, este IPC não possui a arquitetura necessária para executar o *software* pretendido.

Acoplador EK1100

O EK1100 (Figura 4.16) é um acoplador EtherCAT que fornece o elo entre o protocolo EtherCAT e os terminais EtherCAT. Este equipamento converte telegramas transmitidos pelo protocolo EtherCAT em sinais do barramento, iniciando assim



Figura 4.15: IPC CX9020 [19]

uma estação. Uma estação consiste num acoplador e em diversos tipos de terminais introduzidos pelo utilizador, com um limite de até 65534 canais EtherCAT, onde cada terminal contém vários canais.



Figura 4.16: Acoplador EK1100 [19]

Este equipamento contém duas portas RJ45, onde uma recebe do controlador e a segunda remete para outros acopladores existentes na rede, diversificando a tipologia da rede, seja em estrela ou em linha. A alimentação da estação é efetuada pelo acoplador diretamente com os 24 V e não necessita de alimentação externa para os terminais, desde que não ultrapasse os 2 A de corrente. Se for necessária uma corrente maior, existem cartas terminais de alimentação de energia que podem ser agrupadas [19, 58].

Terminal de Entradas EL1008

Para receber todos os sinais de entrada, como o MICO, *drivers* e fins de curso, serão utilizados os terminais EL1008 (Figura 4.17). Este terminal é introduzido em conjunto num acoplador, num IPC ou em outros terminais, montados sobre uma calha DIN.



Figura 4.17: Terminal EL1008 [19]

Este terminal permite obter 8 entradas digitais, que recebem os dados e os enviam eletricamente para a unidade de automação. Contém ainda LED que demonstram o estado atual das saídas [19, 59].

Terminal de Saídas EL2008

Relativamente às saídas de sinais, serão utilizados terminais de saída EL2008 (Figura 4.18). A sua instalação é idêntica à do EL1008, e a sua estrutura é semelhante, uma característica comum dos diversos terminais da Beckhoff.

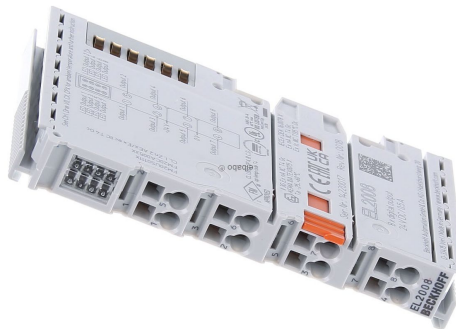


Figura 4.18: Terminal EL2008 [19]

Este terminal contém 8 saídas digitais controladas pelo dispositivo de automação, permitindo a ligação de diferentes tipos de carga (resistiva, indutiva, entre outras), com uma corrente de saída máxima de 0,5 A por canal, além de saídas à prova de curto-circuito e proteção contra inversão de polaridade [19, 60].

Terminal de Saída por *Encoder* Incremental EL2522

Com aspeto físico e montagem semelhantes aos restantes terminais, o terminal EL2522 (Figura 4.19) é um terminal de saída por *encoder* incremental ou *trem* de pulsos, emitindo um sinal modulado em frequência de dois canais com quatro saídas (A/B e respetivos negados) ou um canal de seis saídas (A/B/C e respetivos negados).

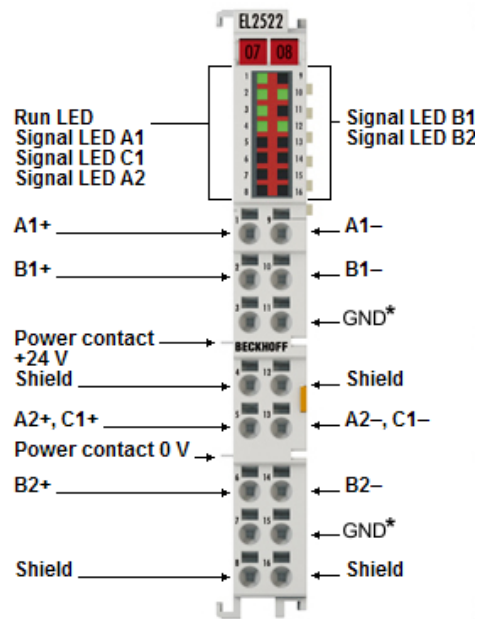


Figura 4.19: Terminal EL2522 [19]

Esta carta é utilizada para o controlo de *drivers* de motor ou outros recetores de sinal, utilizando a frequência, o número de pulsos e os formatos das ondas que permitem o controlo da trajetória [19, 61]. Podem ser escolhidos três métodos de saída: *frequency modulation*, *pulse direction setting* e *incremental encoder*, conforme visualizado na Figura 4.20, respetivamente.

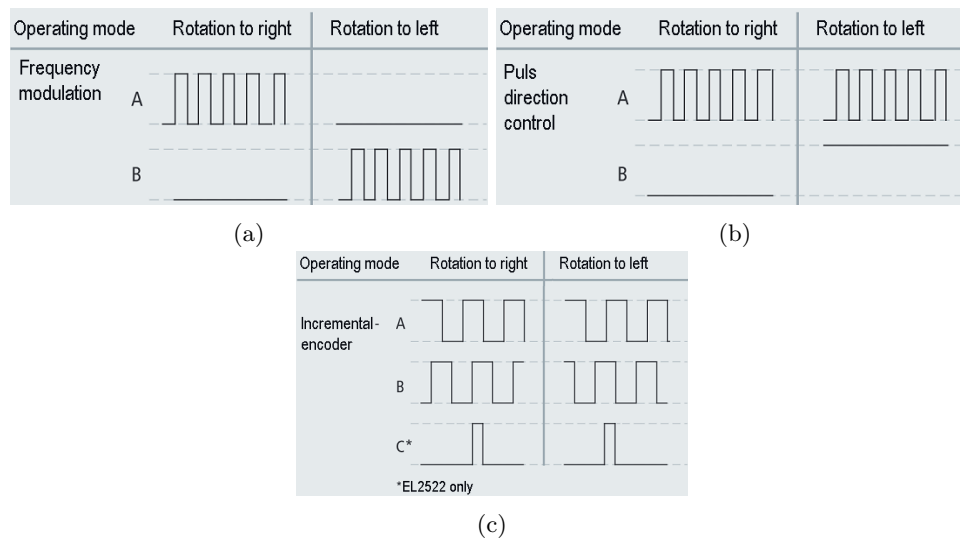


Figura 4.20: Modos de saída do EL2522 [19]: a) *frequency modulation*
 b) *pulse direction setting* c) *incremental encoder*

De acrescentar que a escolha recaiu sobre os sinais de pulso em vez de usar o CANopen, devido a problemas detetados anteriormente ao trabalhar com os *drivers* da FESTO com a tecnologia CANopen.

Terminal de Saída Analógica EL4002

Para o controlo do *spindle*, é necessária uma saída analógica com uma tensão entre 0 a 10 V. O terminal EL4002 (Figura 4.21) é o dispositivo mais adequado para esta função.



Figura 4.21: Terminal EL4002 [19]

Este terminal é alimentado por uma tensão de 24 V e a tensão de saída chega até aos 10 V, sendo capaz de suportar dois canais de saída [19, 62].

4.3.4 Motores do Eixo

No grupo dos motores do eixo, encontram-se os *drivers* e os respetivos motores, incluindo as caixas de engrenagens ou *gearbox*. Dentro deste grupo, mantiveram-se os componentes do projeto anterior, adicionando um novo para efetuar o movimento no eixo Z.

Motores de Passo

Os motores de passo EMMS-ST-87-S-SEB-G2 da FESTO (Figura 4.22) foram integrados nos eixos X e Y do sistema, sendo alimentados a 24 V DC. Estes motores, baseados em tecnologia híbrida de duas fases, incluem um codificador incremental e um travão de retenção, e com um ângulo de passo de $1,8^\circ \pm 5\%$, e um binário de retenção de até 2,5 Nm. Com grau de proteção IP54 e construção robusta, são

particularmente adequados para ambientes industriais exigentes e aplicações de movimento interpolado [63, 64].

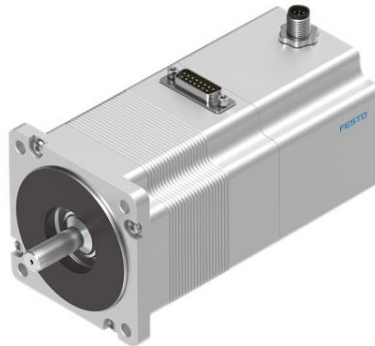


Figura 4.22: Motor *stepper* EMMS-ST-87-S-SEB-G2 [63]

Drivers dos Motores

O CMMS-ST-C8-7-G2 (Figura 4.23) é um controlador de motor (*driver*) da Festo, concebido para o controlo de motores de passo bifásicos das séries EMMS-ST e MTR-ST. Este equipamento permite o controlo em malha fechada de binário (corrente), velocidade e posição, possibilitando ainda o posicionamento por meio de registos de posicionamento armazenados.



Figura 4.23: *Driver* CMMS-ST-C8-7-G2 [63]

Este *driver* destina-se a ambientes industriais e foi desenvolvido para montagem em armários de controlo. Suporta a função de segurança *Safe Torque Off* (STO), oferecendo proteção contra movimentos perigosos durante operações críticas.

O controlador comunica por interfaces como CANopen, DeviceNet, PROFIBUS e RS232/RS485, permitindo uma integração flexível em diferentes arquiteturas de

automação. Inclui também entradas e saídas digitais configuráveis que possibilitam a seleção e ativação de diferentes modos de operação, incluindo posicionamento, *jog/teach*, sequência de registos e sincronização [63, 65].

A escolha do *driver* CMMS-ST-C8-7-G2 e dos respetivos motores de passo prendeu-se essencialmente com a disponibilidade dos componentes na empresa, o que permitiu evitar custos adicionais na implementação da solução. Embora a utilização de servomotores pudesse representar uma opção tecnicamente mais avançada e eficaz, especialmente em termos de precisão, resposta dinâmica e controlo de força, optou-se por maximizar os recursos existentes, assegurando simultaneamente a viabilidade e funcionalidade do sistema desenvolvido no âmbito deste projeto.

Caixa Redutora dos motores

Está acoplada ao motor uma caixa redutora EMGA-80-P-G5-SST-87 da Festo, visualizada na Figura 4.24, cuja principal função é a redução da velocidade angular e o aumento do binário transmitido ao eixo de saída. Com uma relação de transmissão de 5:1, elevada rigidez torcional (6 Nm/arcmin) e uma folga torcional reduzida (0,12°), esta unidade assegura uma eficiência de 98 %, característica essencial para o controlo suave e fiável do acionamento dos motores de uma CNC [63, 66].

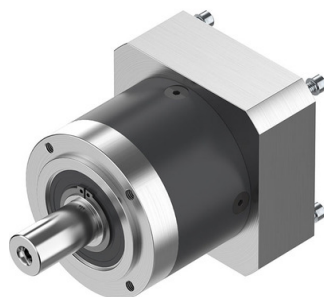


Figura 4.24: Caixa de engrenagens EMGA-80-P-G5-SST-87 [63]

Este componente é importante ser referenciado devido à taxa de transmissão na caixa redutora (5:1), para que se possa configurar os *drivers* corretamente, assim como outras características do eixo.

4.3.5 Sensores e Atuadores

No grupo dos sensores e atuadores, encontram-se todos os componentes relacionados com atuadores e sensores, que permitem acrescentar funcionalidades ao funcionamento da máquina. Relativamente à arquitetura anterior, somente se mantiveram os fins de curso já instalados na mesa, adicionando um sistema de luzes para identificar o estado atual da máquina.

Fins de Curso na Mesa

Foram utilizados vários interruptores de fim de curso 3SE5 423-0CD20-1EB1 da Siemens, conforme se mostra na Figura 4.25, com atuador de êmbolo com roldana e contactos de ação rápida (1 Normalmente Aberto (NO) + 1 Normalmente Fechado (NF)). Estes dispositivos compactos e robustos, com grau de proteção IP67 e ligação por conector M12, garantem deteção fiável das posições extremas da mesa CNC, mesmo em ambientes industriais exigentes [42, 67].



Figura 4.25: Fim de curso 3SE5 423-0CD20-1EB1 [42]

Coluna Luminosa de Sinalização

Para permitir uma melhor perceção do estado em que a máquina se encontra, e ao analisar protocolos referentes a estas atividades, como o IEC 60204-1 e ISO 13850, definiu-se que serão usadas três luzes em estilo semáforo nesta solução, com vermelho, amarelo e verde como as cores de cada luz, conforme visto na Figura 4.26 [56, 45].



Figura 4.26: Coluna luminosa sinalizadora de máquinas industriais [68]

Para estas cores, e alterando ligeiramente o que é exigido pelos protocolos mencionados, existirão 6 modos, apresentados na Tabela 4.2. A grande alteração relativamente às normas foi a execução normal, geralmente indicada pelo verde; devido

ao fato de que esta máquina pode realizar movimentos não estimados pelo operador, o amarelo intermitente é o mais indicado para alertar sobre esses movimentos. Relativamente ao verde, este indica os modos em que a máquina não está em funcionamento ou está sob supervisão de um operador.

Tabela 4.2: Modos operacionais ilustrados pelo semáforo

Cor/Estado	Definição
Verde permanente	Operação automática sem correr - <i>stand-by</i>
Verde intermitente	Modo manual - intervenção de um operador
Amarelo permanente	<i>Drivers</i> desligados
Amarelo intermitente	Operação automática e <i>homing</i> da CNC
Vermelho permanente	Botão de emergência pressionado
Vermelho intermitente	Erro ou falha

4.3.6 *Spindle*

No grupo do *spindle*, aborda-se a ferramenta crucial para a maquinação de uma peça. Este grupo inclui um motor *spindle* equipado com uma ferramenta na sua extremidade, além de um VFD para controlar a velocidade da ferramenta e relés para efetuar o controlo em contacto seco.

Motor *Spindle*

O motor *spindle* selecionado para esta aplicação é um motor da HLTNC [69], visualizado na Figura 4.27, com uma potência de 1,5 kW, frequência máxima de 400 Hz e velocidade máxima de 24 mil RPM, contendo refrigeração a ar. Este motor é indicado para trabalhos em madeira, metal leve e alumínio [70].

Inversor de Frequência

Para controlar a velocidade do *spindle*, é necessário introduzir um VFD. Para esse efeito, foi escolhido o YL620-A *single-phase* 220 V 50/60 Hz, conforme visualizado na Figura 4.28.

Este VFD é capaz de suportar a potência do *spindle* selecionado e fornecer uma corrente máxima de 7 A [70].

4.3.7 Relés

Devido à necessidade de controlo e à impossibilidade de ligar *inputs* diretamente ao VFD, é apenas possível realizar a ligação com contactos secos e tensões diferentes dos 24V; assim, foi necessário o uso de relés.

Para tal, foi considerado o uso do relé HF41F, conforme demonstrado na Figura 4.29, por serem estreitos e permitirem a instalação em calhas DIN. A zona da bobina

Figura 4.27: Motor *spindle* [70]

Figura 4.28: Inversor de frequência YL620 [70]

será controlada pela saída do controlador, enquanto as saídas do relé serão ligadas ao respetivo controlo [71].

4.3.8 Lista de Componentes

Na Tabela 4.3 estão listados todos os componentes utilizados neste projeto.

4.4 Software

Esta secção visa explorar os *softwares* que permitirão a configuração dos motores e o controlo da máquina, mais especificamente o FCT, dedicado ao controlo dos motores, ao TwinCAT e o pacote de *software* TF5200, dedicado ao uso de CNC. Será ainda explorado o Inventor e a extensão CAM, para a criação de códigos G.

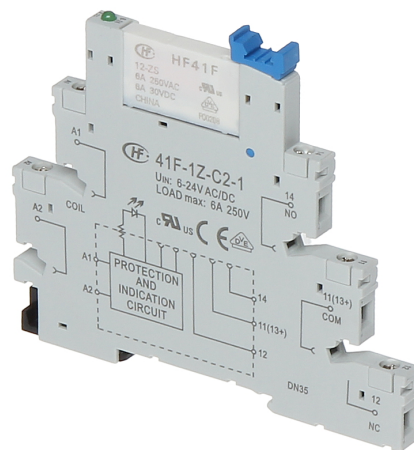


Figura 4.29: Relé HF41F e respetivo suporte DIN [72]

No final, será apresentado um fluxograma para o controlo do estado dos motores e outros componentes relacionados.

4.4.1 *Festo Configuration Tool*

O *Festo Configuration Tool* (FCT) (Figura 4.30) é uma ferramenta de *software* desenvolvida pela Festo, concebida para a parametrização, comissionamento e manutenção de dispositivos eletromecânicos, tais como acionamentos, controladores e motores. Trata-se de uma plataforma modular que combina um *framework* central com múltiplos *plugins* dedicados a cada tipo de equipamento, permitindo uma grande flexibilidade e escalabilidade na sua utilização [63].

O FCT possibilita a configuração de sistemas em modo *offline*, permitindo a preparação de projetos sem necessidade de ligação ao equipamento físico, e em modo *online*, para comissionamento e monitorização em tempo real.

A estrutura do *software* permite a gestão de múltiplos componentes num único projeto, possibilitando o reaproveitamento de configurações, como aceleração, velocidade e posições finais, definição de movimentos, entre outros [63].

4.4.2 *TwinCAT*

O *The Windows Control and Automation Technology* (TwinCAT) é uma plataforma de desenvolvimento da Beckhoff destinada ao controlo industrial e automação. Funciona como um sistema em tempo real que converte um computador baseado em Windows num controlador de tempo real, sistema de controlo de movimento e plataforma de visualização. A sua versatilidade, modularidade e capacidade de integração fazem dele uma ferramenta essencial para aplicações industriais avançadas, especialmente em ambientes orientados para a Indústria 4.0 [19].

Tabela 4.3: Tabela de quantidades e respectivos componentes a utilizar

Quantidade	Equipamento
1	Siemens sentron 3LD2003-1TL53
1	Siemens differential <i>switch</i>
1	<i>Circuit breaker</i> (Allen Bradley 2P, 2 A, Type C <i>Curve</i> , <i>Cutting Power</i> 10 kA)
1	Siemens SITOP PSU100S 6EP1334-3BA00
1	Mico 4.6
1	Mico 2.6
1	Botão de emergência 2 canais
1	Relé de emergência PILZ PNOZ XV2
3	<i>Drivers</i> Motor de passo Festo CMMS-ST-C8-7-G2
3	Motor de passo Festo EMMS-ST-87-S-SEB-G2
3	Caixa redutora Festo EMGA-80-P-G5-SST-87
9	Fim de curso Siemens 3SE5-423-0CD20-1EB1
1	Coluna luminosa de sinalização industrial de 3 cores
1	Acoplador EK1100
3	Terminal de entrada EL1008
1	Terminal de saída EL2008
2	Terminal de saída EL2522
1	Terminal de saída EL4002
1	<i>Spindle</i> 1.5 kW 220 V 24 k RPM
1	Inversor VFD YL620
2	Relés HF41F

A arquitetura do TwinCAT assenta num modelo modular e escalável, constituído por dois componentes principais. O primeiro é o TwinCAT *Engineering* (XAE), o ambiente de desenvolvimento integrado ao *Visual Studio*, que permite programar em diferentes linguagens. O segundo é o TwinCAT *Runtime* (XAR), responsável pela execução determinística do código em tempo real, comunicação com *hardware* e sincronização de processos.

Com a introdução de módulos complementares, designados por TwinCAT *Functions* (TF), é possível alargar a funcionalidade do sistema conforme os requisitos de cada aplicação, podendo utilizar diversos protocolos como Modbus, PROFINET, entre outros, ou outras funções, como HMI ou controlo de movimentos [19].

No seu conjunto, o TwinCAT constitui uma solução de engenharia extremamente versátil e poderosa. A separação clara entre o ambiente de desenvolvimento e o motor de execução, aliada à possibilidade de adicionar funcionalidades específicas, contribui para uma maior flexibilidade na criação e manutenção de sistemas. Esta modularidade também reduz os tempos de engenharia, melhora a reutilização de componentes e facilita a manutenção ao longo do ciclo de vida dos equipamentos. Além disso, ao suportar nativamente os princípios da Indústria 4.0, como conectividade, modularização, virtualização e análise de dados, o TwinCAT é uma das plataformas mais completas e preparadas para os desafios modernos.

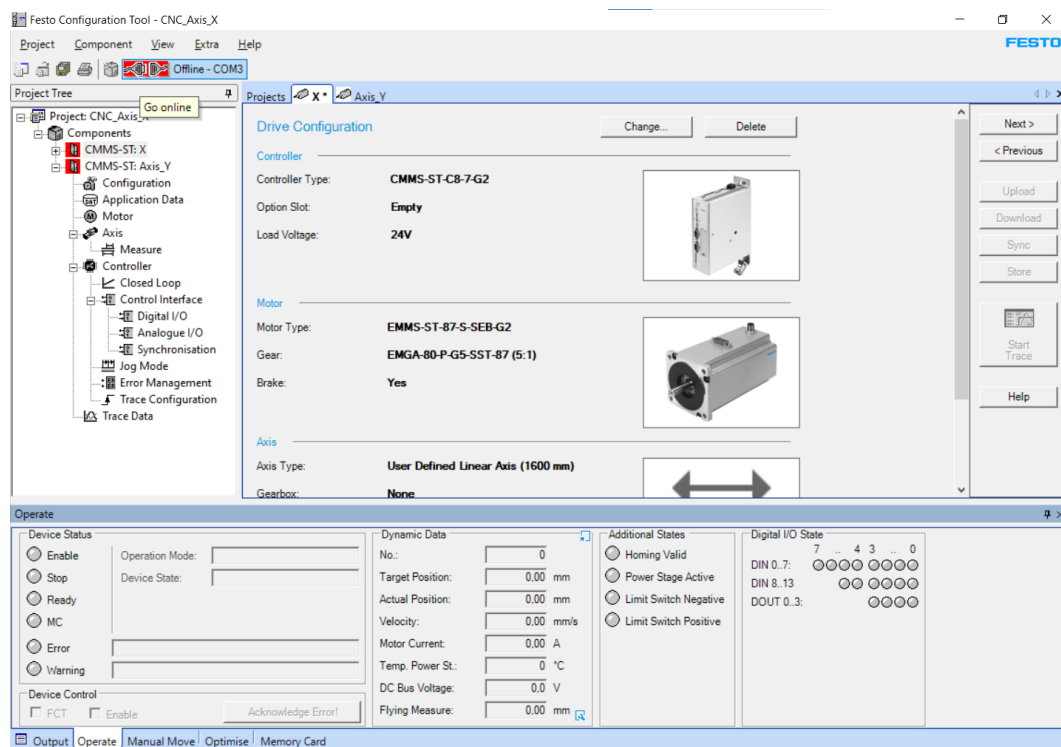


Figura 4.30: Página principal do FCT

4.4.3 Pacote de Funções TF5200 (CNC)

O pacote de funções TF5200 - CNC é um módulo funcional que permite implementar CNC de elevado desempenho em sistemas baseados na plataforma TwinCAT. Este pacote transforma um IPC num controlador CNC em tempo real, suportando até 8 eixos interpolados por canal CNC, múltiplos canais CNC, e integração total com *drivers* de motores, módulos I/O e redes como o EtherCAT. O TF5200 permite a execução de programas em código G com suporte à norma DIN 66025.

Uma das limitações a considerar ao escolher o controlador é a tecnologia a adotar, uma vez que processadores mais leves, como os *Advanced RISC Machine* (ARM), não oferecem suporte para este pacote. Assim, a escolha deve recair sobre uma CPU com arquitetura x86, que proporciona a arquitetura adequada, e conforme o nível de plataforma Beckhoff [19].

Funcionamento Interno

Este pacote constitui o núcleo da funcionalidade CNC no sistema TwinCAT. Internamente, o TF5200 implementa uma máquina NC em tempo real, que executa programas de interpolação, controlo de movimentos e descodificação baseada em instruções. O seu funcionamento baseia-se numa arquitetura modular, permitindo a definição de múltiplos canais CNC, cada um com os seus próprios conjuntos de eixos, ciclos tecnológicos e configurações.

Para permitir a integração com a lógica de controlo do PLC, que opera simultaneamente e no mesmo ambiente que o CNC, como ilustrado na Figura 4.31, o TF5200 disponibiliza uma interface chamada *High-Level Interface* (HLI), que serve de ponte entre os dois ambientes.

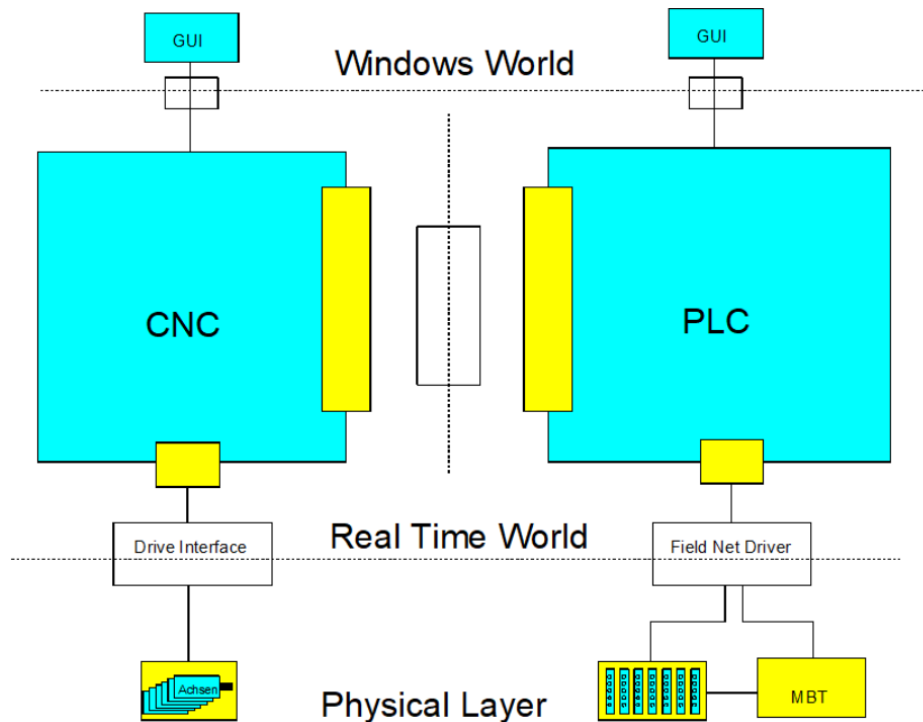


Figura 4.31: Execução do CNC e PLC no controlador [19]

Resumidamente, o HLI é uma memória partilhada que permite tanto ao CNC como ao PLC acederem para implementar a troca de dados entre os dois.

A estrutura do HLI inclui canais, eixos e dados da plataforma, como demonstrado na Figura 4.32. Como se pode observar, os canais são partilhados pelo HLI, enquanto o lado do CNC controla os eixos e *spindles* [19]. É ainda possível transmitir dados para o HLI através do PLC.

4.4.4 Autodesk Inventor: Funcionalidades CAD e Expansão CAM

O Autodesk Inventor é um *software* de modelação tridimensional assistida por computador (CAD) amplamente utilizado na concepção e desenvolvimento de produtos mecânicos. Este programa permite a criação de modelos paramétricos e sólidos, assim como a produção de desenhos técnicos, simulações de funcionamento e análises estruturais, promovendo uma abordagem integrada ao desenvolvimento do produto.

A integração do Inventor com a extensão CAM (Figura 4.33) expande significativamente as suas capacidades, possibilitando a geração de percursos de maquinaria diretamente a partir dos modelos tridimensionais. Através do módulo Inventor

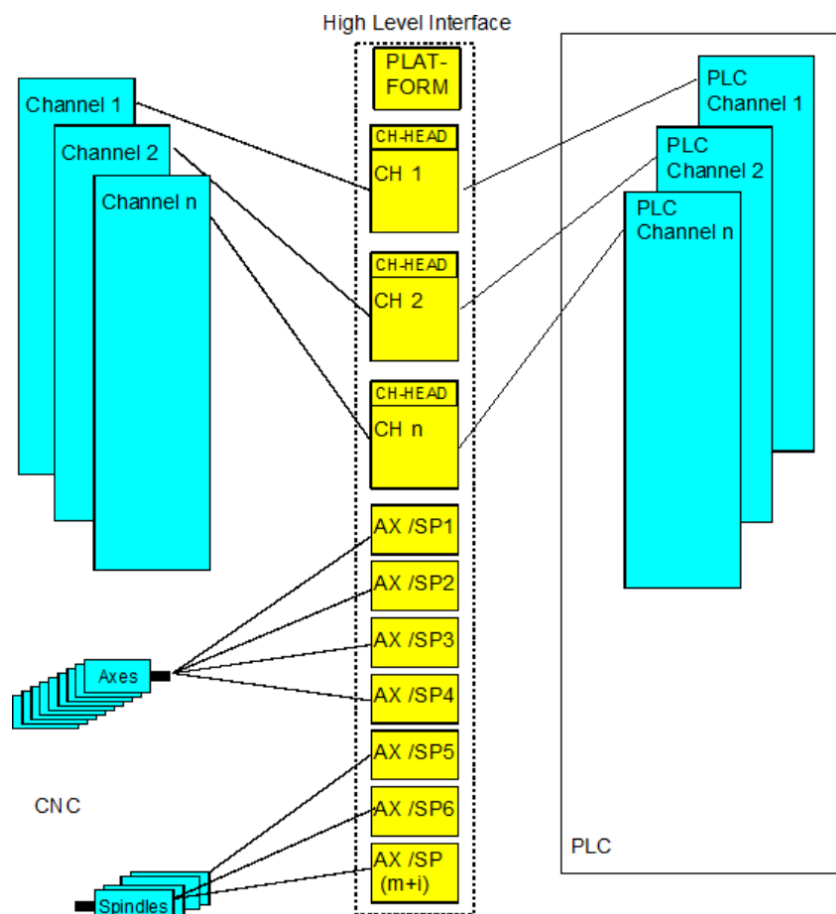


Figura 4.32: Funcionamento do HLI [19]

CAM, é possível definir operações de maquinação, simular processos de fabrico e exportar códigos G adaptados a máquinas CNC. Esta funcionalidade assegura uma maior coerência entre o projeto e a produção, reduzindo erros de fabrico e otimizando os tempos de preparação e execução.

A utilização combinada das ferramentas CAD e CAM no Inventor permite uma transição eficiente do modelo digital para a fabricação física, favorecendo um fluxo de trabalho contínuo entre as fases de conceção, verificação e produção [73].

A seleção do Inventor deveu-se, em grande parte, à disponibilidade de uma licença educacional gratuita, o que o torna particularmente acessível em contextos académicos. No entanto, para a Gislótica, recomenda-se a utilização do mesmo *software* de modelação adotado internamente pela empresa, neste caso o SolidWorks, o qual também dispõe de uma interface CAM integrada, permitindo a continuidade do fluxo digital desde a modelação até à fabricação.

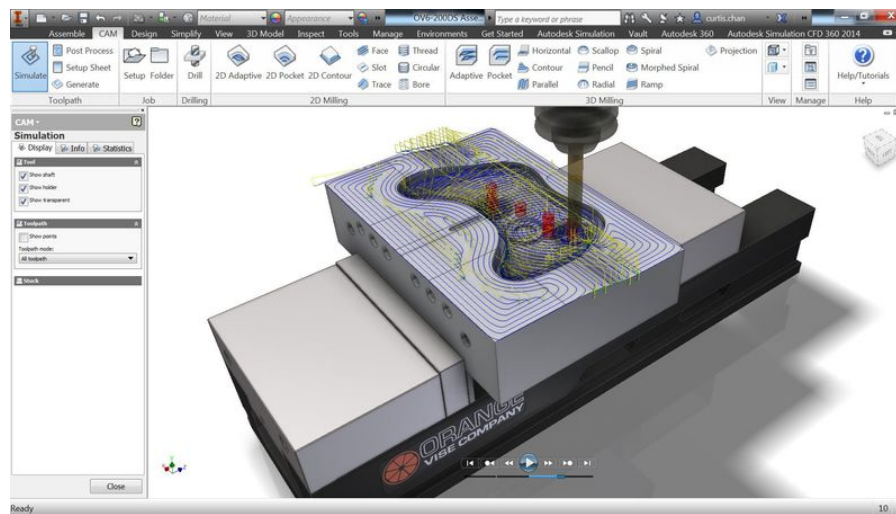


Figura 4.33: Autodesk Inventor com extensão CAM [73]

4.4.5 Fluxogramas e Máquina de Estados

Para otimizar o funcionamento da máquina, foram necessários alguns ajustes a um código base fornecido pela Bresimar [74], fornecedora da Beckhoff em Portugal. Este código contém as funcionalidades mínimas para que uma CNC funcione, onde permite a execução de um código G, controlo dos motores no modo manual e execução de linhas de código introduzidas pelo operador.

Porém, é necessário algum controlo extra, onde se explorou a HMI e o que suporta, mas que não contém lógica. Sem contar com a necessidade de correção da referência na qual continha erro, é necessário implementar lógica que permita um controlo acertado dos motores, proteção no eixo para evitar embates com os respetivos limites e alguma grafia visual para os operadores.

Para isso, foram realizadas três modificações-base: controlo dos motores, sistema da coluna luminosa e sistema de limites.

Controlo dos Motores

Para um controlo mais eficiente dos motores, foi implementada uma lógica que permite que os motores estejam ligados e disponíveis para movimento ou desligados.

Conforme ilustrado na Figura 4.34, este controlo inicia-se com uma paragem de emergência, quer inicial ou provocada pelo botão de emergência, que desabilita todos os motores. Após ligar o relé de segurança, o sistema entra num modo de espera, onde os motores permanecem desabilitados até receber um sinal de *enable* e os MICO estejam habilitados.

Quando recebe um *enable* e as condições estão conforme o esperado, passa por uma etapa temporária que permite ativar os motores e aguarda um tempo para que

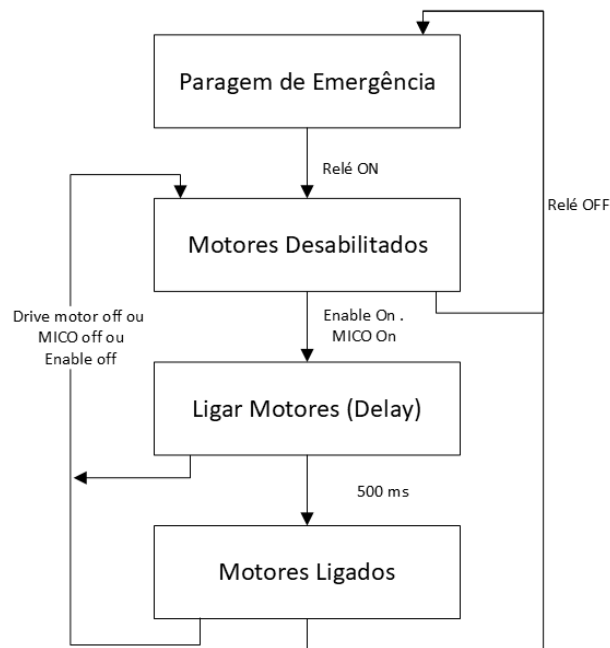


Figura 4.34: Máquina de estado dos motores

os *drivers* habilitem os motores. Após esse tempo, o sistema analisa se os motores estão disponíveis e permite que os processos de movimento sejam efetuados.

Se os motores estiverem ligados e ocorrer uma falha num dos *drivers*, ou se um MICO falhar ou o *enable* for desligado, os motores serão desabilitados.

Controlo da Coluna Luminosa de Sinalização

Para permitir que a posição de controlo seja visível para todos os operadores, foi implementado um sistema de coluna luminosa de sinalização, que está descrito na Subsubsecção 4.3.5.

O controlo das luzes está interligado com os vários processos nos quais a máquina se encontra. Na Figura 4.35a representa as etapas onde se analisa se ocorreu alguma emergência ou foi detetado algum erro, introduzindo a respetiva luz. Na Figura 4.35b representa o estado da máquina quando está no processo de ligar os motores, correspondendo às etapas do controlo de motores. Na Figura 4.36 representa os vários modos na máquina e respetivos estados, quando os motores estão ligados, adaptando-se e verificando os respetivos estados, onde analisa algumas questões como se foi efetuado o *homing* ou se o processo foi acabado.

Sistema de Limites

Devido à inexistência de interligar os fins de curso diretamente aos eixos no programa-base, foi necessária a implementação de uma lógica para parar os movimentos dos motores e analisar o seu estado, de forma a não danificar a máquina.

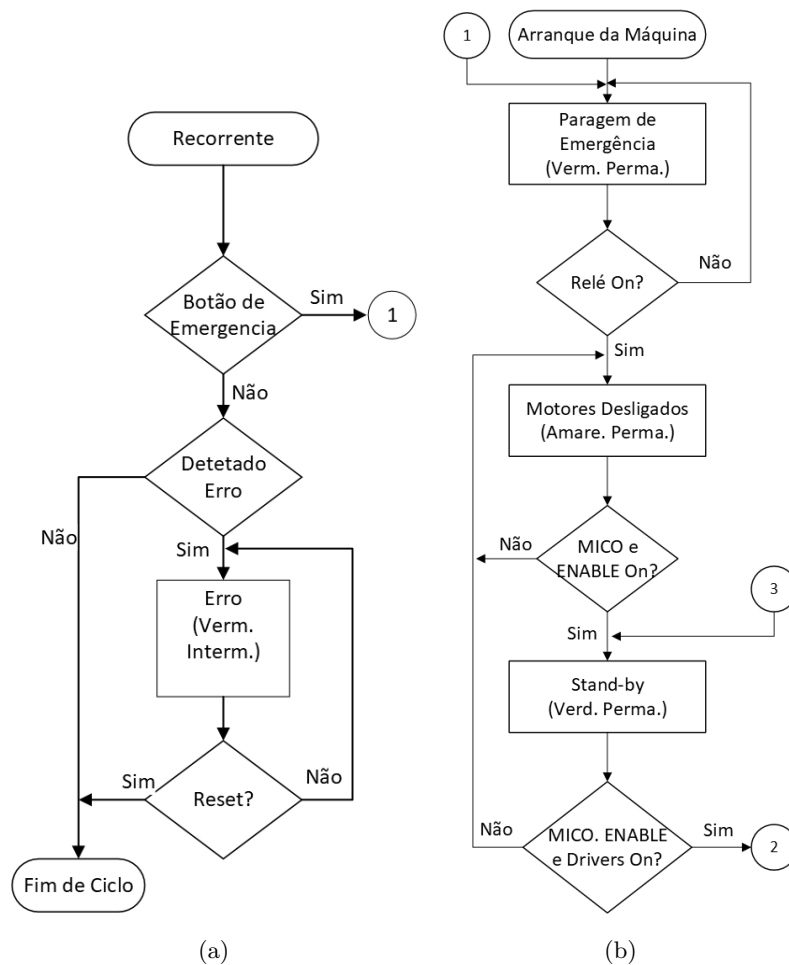


Figura 4.35: Fluxograma da coluna luminosa de sinalização: a) emergência e erro b) ligar motores

Com os fins de curso instalados na máquina, quando um destes detetar um sinal, a execução do movimento é interrompida por um *stop* e de seguida efetua um *reset* para evitar a continuidade do movimento, caso haja alguma anomalia no código ou erro do operador. Para isso, foi elaborado o fluxograma ilustrado na Figura 4.37.

Após parar o movimento, o sistema regista qual foi o sensor que bloqueou, para que o movimento na direção do limite não seja efetuado. O sistema entrará em modo manual forçado, onde não é permitido alterar o modo até que o fim de curso atuado seja desativado. Para isso, entrará em *jog*, e se o movimento for na direção do limite, o código bloqueará, permitindo somente o movimento no sentido inverso no respetivo eixo. Quando todos os sensores que foram atuados estiverem desligados, ou seja, nenhum dos eixos está num limite, é desativado o modo manual forçado, permitindo alterar o modo.

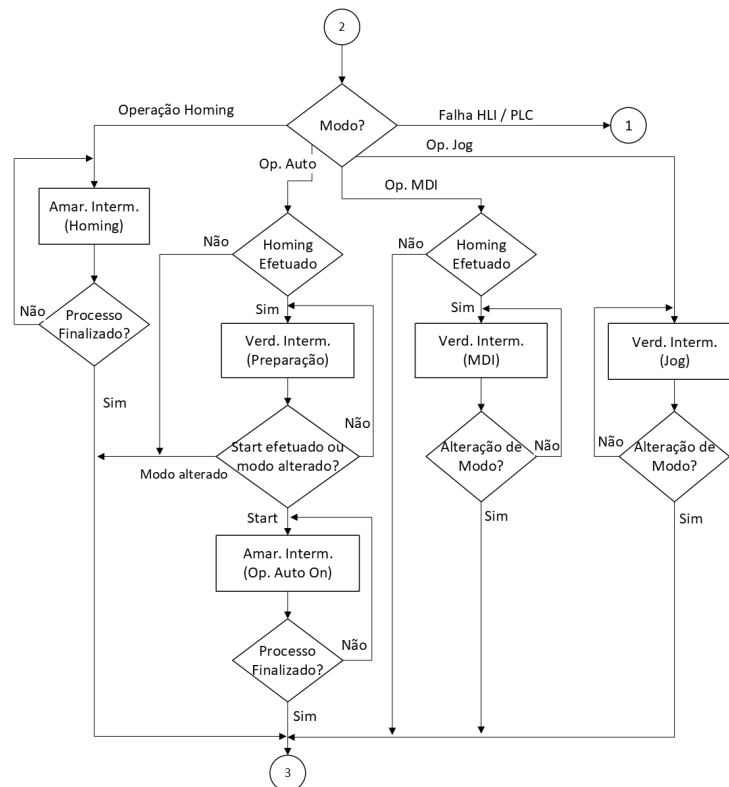


Figura 4.36: Fluxograma da coluna luminosa de sinalização - Funções

4.5 Conclusão da Arquitetura Proposta

Este capítulo permitiu identificar e analisar a arquitetura mais adequada para o desenvolvimento do projeto, aproveitando, sempre que possível, os equipamentos disponíveis.

Numa primeira fase, foi avaliado o estado inicial do projeto, através da análise da mesa, dos componentes existentes e da arquitetura previamente implementada. A partir desta avaliação, foi possível reformular e propor uma nova arquitetura e respetivo esquema da CNC, fundamentada nos conhecimentos teóricos adquiridos e nos objetivos definidos. Neste projeto, foram selecionados os componentes de *hardware* a utilizar, sendo apresentadas as respetivas funcionalidades e a justificação para a sua escolha.

Posteriormente, foi analisado o *software* necessário ao funcionamento da solução proposta, incluindo o *software* para a configuração de *drivers*, a utilização do TwinCAT e do pacote de funções CNC, bem como a integração de ferramentas CAD/CAM. Foram ainda apresentados fluxogramas que representam o controlo de estados e dos atuadores a ser implementado.

Assim, este capítulo resultou na formulação de uma proposta arquitetónica final, que constitui a base essencial para a fase de implementação prática e para a realização dos testes de validação do sistema.

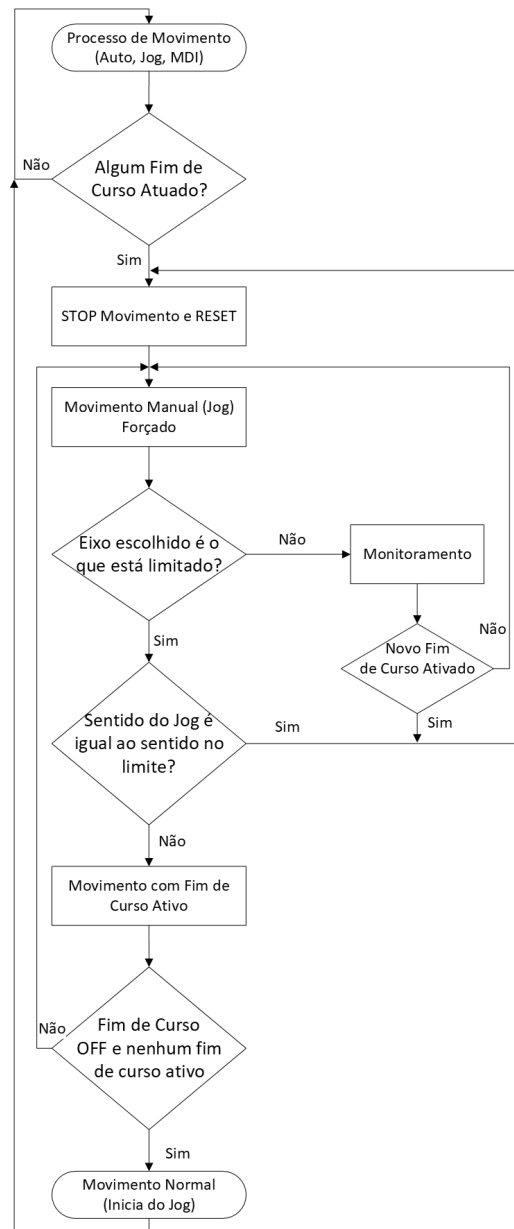


Figura 4.37: Fluxograma dos limites

Capítulo 5

Implementação Prática

O capítulo da implementação prática terá como objetivo abordar todos os temas necessários para que a CNC possa realizar o trabalho conforme os objetivos propostos inicialmente.

Na secção 5.1, serão abordadas todas as configurações do *hardware*, iniciando pelos componentes de controlo e segurança, seguidos dos restantes componentes. Serão explicadas as razões das ligações presentes no esquemático apresentado no Anexo B. Na secção 5.2, estarão os *drivers* dos motores e as configurações que foram necessárias efetuar.

Na secção 5.3, será revista a configuração de um programa simples de movimentos *Point to Point* (PTP), realizado para efetuar testes básicos e compreender o funcionamento fundamental do TwinCAT em relação ao movimento de eixos. Serão exploradas, de forma resumida, as configurações de entrada e saída dos terminais, os parâmetros que requerem atenção ao configurar eixos, bem como algum código base utilizado para estes testes.

Na secção 5.4, será analisada toda a parte do código-base e o seu funcionamento em CNC, com especial atenção aos parâmetros cruciais a considerar, às mudanças efetuadas na HMI e a outras funções, incluindo o detalhamento do código produzido, o qual se encontra no Anexo C.

Na secção 5.5, será abordado o *software* utilizado para o usufruto CAD/CAM, as funcionalidades que apresenta, o seu funcionamento e as considerações a ter em conta ao finalizar um código G.

É importante referir que os testes realizados e descritos no Capítulo 6 têm como base as informações obtidas nesta implementação. Assim, muitos testes podem não conter os parâmetros ou códigos atualizados, havendo ajustes ao longo do tempo que possibilitaram a obtenção dos resultados apresentados neste capítulo.

5.1 Configurações Referente ao *Hardware*

O *hardware* é sempre a parte crucial para o funcionamento de uma máquina. Como visto no Capítulo 4, na secção dedicada ao *hardware*, foram explicadas, de forma resumida, algumas escolhas, mas componentes como os *drivers* ou algumas entradas e saídas da parte de controlo, não foram explorados com tanto detalhe.

Assim, nesta secção será abordado o *hardware* em geral, sendo que o esquema elétrico se encontra no Anexo B. Na secção seguinte, serão explorados os *drivers* dos motores e as suas especificações.

Nesta parte, serão inicialmente discutidas as ligações do controlo, enquanto as configurações dos terminais ficarão reservadas para a Secção 5.2. Os restantes componentes serão revistos de forma geral, uma vez que foram explicados com o detalhe necessário no capítulo antecessor.

Antes de detalhar os componentes, relembra-se que o funcionamento desta CNC se baseia num controlo em malha aberta, ou seja, não existe *feedback* do movimento final, exceto os fins de curso.

5.1.1 Implementação de Controlo

Relativamente ao controlo, a montagem foi efetuada conforme o esperado. No entanto, é necessário fornecer mais alguns detalhes sobre os terminais utilizados.

Implementação dos Terminais EL1008 e EL2008

Os terminais EL1008 e EL2008, que são terminais de entrada e saída, apresentam a característica de possuírem o mesmo *pinout*.

No entanto, é necessário ter cuidado ao realizar a introdução das ligações. Existem duas numerações: uma corresponde à ordem dos pinos, numerada verticalmente, e a segunda refere-se à ordem do *pinout*, ou seja, às entradas e saídas associadas a cada pino, dispostas horizontalmente e descendo, conforme demonstrado na Figura 5.1.

As ligações recebidas serão organizadas conforme representado na Tabela 5.1.

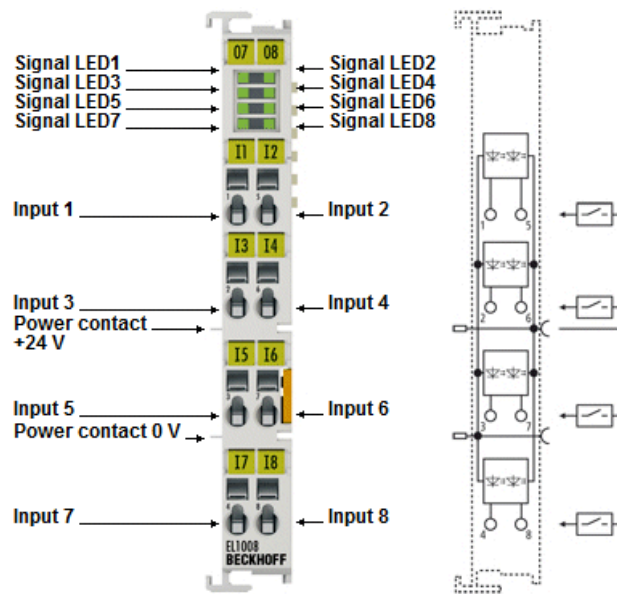


Figura 5.1: Esquemático dos terminais EL1008 [19]

Tabela 5.1: Entradas dos EL1008 e saídas do EL2008

EL1008				EL2008	
Inputs	1°	2°	3°	Outputs	1°
<i>Input 1</i>	Relé	Sensor <i>Home Y</i>	<i>Drive Z Apto</i>	<i>Output 1</i>	
<i>Input 2</i>	MICO 4	Sensor Neg. <i>Y</i>	<i>Drive Z Erro</i>	<i>Output 2</i>	<i>Drive X Enable</i>
<i>Input 3</i>	MICO 2	Sensor Neg. <i>X</i>		<i>Output 3</i>	<i>Drive Y Enable</i>
<i>Input 4</i>	<i>Drive X Apto</i>	Sensor <i>Home X</i>		<i>Output 4</i>	<i>Drive Z Enable</i>
<i>Input 5</i>	<i>Drive Y Apto</i>	Sensor Pos. <i>X</i>		<i>Output 5</i>	Luz Verde
<i>Input 6</i>	<i>Drive X Erro</i>	Sensor Neg. <i>Z</i>		<i>Output 6</i>	Luz Amarela
<i>Input 7</i>	<i>Drive Y Erro</i>	Sensor <i>Home Z</i>		<i>Output 7</i>	Luz Vermelha
<i>Input 8</i>	Sensor Pos. <i>Y</i>	Sensor Pos. <i>Z</i>		<i>Output 8</i>	

Implementação do Terminal EL2522

O terminal EL2522 possui a particularidade de transmitir dois sinais diferentes, designados por A e B. Para tal, foi introduzido um cabo de nove pinos com terminal RS232, contendo quatro fios correspondentes aos sinais A e B, bem como aos respectivos negativos, conforme demonstrado na Figura 5.2. Não foi necessário ligar o fio *ground*, uma vez que tanto a estação (conjunto do EK1100) como o *driver* dos motores partilham o mesmo *ground*. O *shield* não foi conectado, pois não se identificaram problemas de interferência eletromagnética. Para este projeto, as ondas a serem enviadas são de *frequency modulation*, conhecidas pelos sinais CW/CCW.

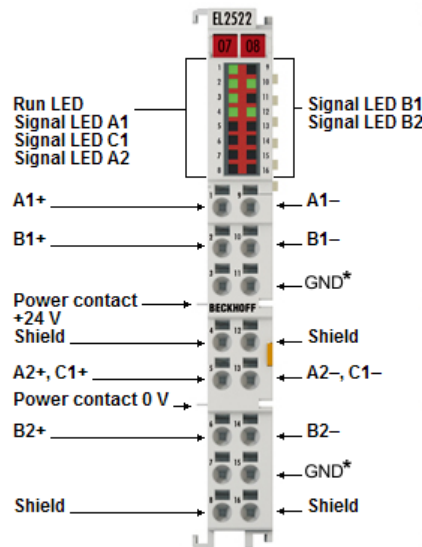


Figura 5.2: *Pinout* dos terminais EL2522 [19]

5.1.2 Implementação de Segurança

No que diz respeito ao sistema de segurança, este encontra-se em conformidade com o que foi demonstrado no capítulo anterior. É importante ressaltar uma breve explicação sobre o circuito de *start*. Este pode ser ativado pelo sistema de controle, mas optou-se para que fosse ativado por um botão externo, de modo a que todo o controle de segurança permaneça fora do circuito de controle.

5.1.3 Resultado Final do *Hardware*

Relativamente aos resultados finais, a implementação dos restantes grupos, como os sensores, os atuadores e a alimentação, foi realizada conforme o projeto existente. O *spindle*, devido a um atraso na implementação, não foi instalado, assim como o terceiro eixo. A implementação final é demonstrada na Figura 5.3.

A utilização de três sensores de fim de curso em cada eixo possibilita o processo de *homing*, sendo que outros dois sensores são utilizados como limites, empregando contatos NO para efetuar os sinais.

5.2 *Drivers* dos Motores

Os *drivers* dos motores constituem uma peça-chave para a realização dos movimentos dos motores. Com o auxílio do FCT, que corresponde à programação do *driver*, torna-se uma ferramenta crucial para este trabalho, permitindo compreender como cada configuração afeta o desempenho do motor.

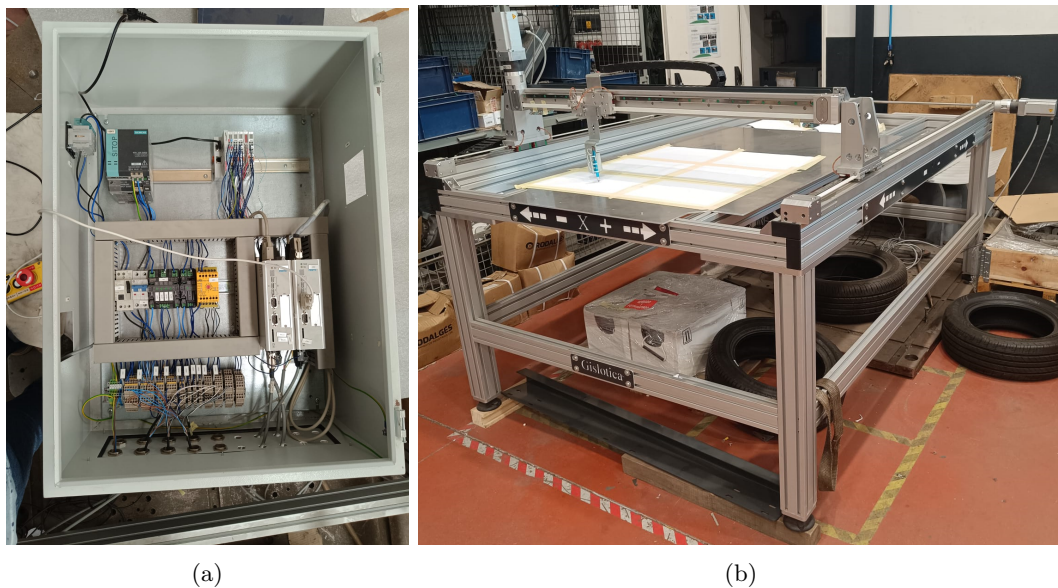


Figura 5.3: Implementação do *hardware*: a) Quadro elétrico b) Mesa

Esta secção será dividida nas seguintes subsecções: Configurações Gerais, *Input/Output* e *Trace*. Estes parâmetros são fundamentais para a avaliação de diversos fatores.

5.2.1 Configurações Gerais dos *Drivers*

Ao criar um eixo no FCT, são apresentados vários parâmetros, que incluem: configuração, aplicação, motor, eixos, controlo e *trace*. Os dois últimos serão abordados nas subsecções seguintes.

Configuração Geral

Este parâmetro tem como objetivo recolher os dados principais do controlador, motor e eixo, conforme demonstrado na Figura 5.4.

No controlador, são introduzidos o tipo de controlador, a existência de uma *slot* extra e a tensão de carga, uma vez que o motor pode ser alimentado por 48 V ou 24 V. No motor, é especificado o modelo a ser utilizado, a caixa redutora a ser implementada e a sua relação de transmissão e a presença de um travão. Relativamente ao eixo, é indicado o tipo de eixo e se este contém caixa redutora.

Como o controlador e o motor são da Festo, eles incluem os valores correspondentes, que serão calculados para o respetivo controlo. O eixo, que foi desenvolvido na empresa, exigiu o cálculo do *feed constant* para atender aos requisitos demonstrados na Figura 5.5.

Para tal, foi necessário consultar o catálogo da peça [75], onde se procurou a polia introduzida, uma polia para correias dentadas HTD 5M, e determinar o diâmetro

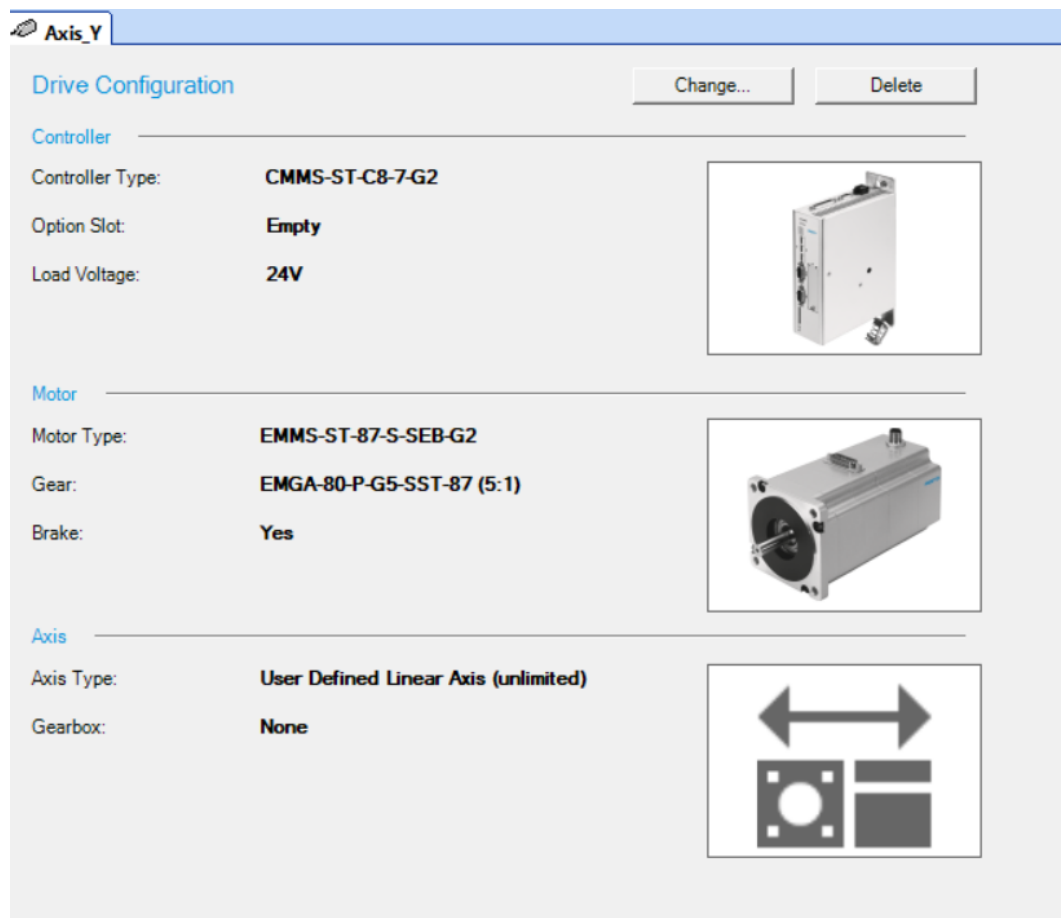


Figura 5.4: Parâmetros da configuração

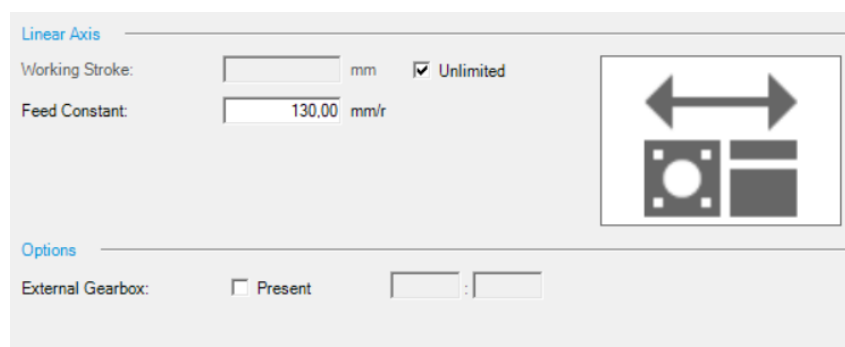


Figura 5.5: Parâmetro do eixo

primitivo, que é o diâmetro que passa pelo centro dos pinos da corrente quando esta está envolvida na engrenagem. Este valor é de 41,38 mm.

Com este valor, pode-se calcular o perímetro primitivo. Quando é efetuada uma rotação, o valor rodado é igual ao perímetro primitivo, ou seja, o valor do *feed constant* é igual ao perímetro primitivo, calculado pela seguinte fórmula:

$$P_p = \pi \cdot D_p \quad (5.1)$$

Onde P_p representa o perímetro primitivo, expresso em mm, e D_p representa o diâmetro primitivo, também em mm. O *feed constant* indica a distância que o eixo percorre numa revolução (sem caixa redutora), que corresponde ao perímetro primitivo. Após os cálculos, conclui-se que o *feed constant* é aproximadamente 130 mm/rev.

Foi definido que o eixo era infinito, para que, durante a fase de testes, os motores não parassem abruptamente ao indicarem que estão no limite do eixo, quando, na verdade, não se encontram nessa situação.

Configuração da Aplicação

Relativamente à configuração da Aplicação, existem três abas: *operation mode*, *environment* e *message*. As duas primeiras são as mais relevantes.

No que diz respeito ao *operation mode*, é possível configurar o modo de comunicação entre o *driver* e o controlador. Dependendo do modo selecionado, é possível existirem outros tipos de controlo baseado na tecnologia escolhida. Como a comunicação será efetuada via sinais de impulso, o modo mais indicado é o *Synchronisation*, conforme visualizado na Figura 5.6a.

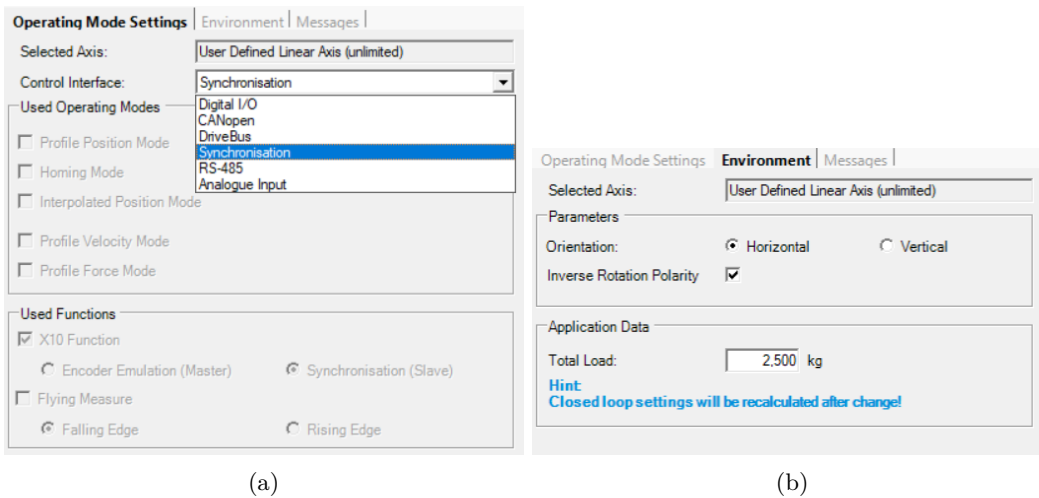


Figura 5.6: Parâmetros de aplicação: a) *operation mode* b) *environment*

Quanto ao *environment*, como demonstrado na Figura 5.6b, permite configurar a orientação do eixo, verificar se a polaridade é invertida e estimar um peso aproximado da carga sobre o eixo.

Configuração dos Motores e Eixo

Nos parâmetros do motor, estão referenciadas todas as questões relacionadas com o funcionamento. Muitos parâmetros são previamente calculados pelo *software* com a introdução das características dos motores, como a corrente máxima e os seus

derivados, ressonância e o controle de paragem. Não foram efetuadas alterações, mantendo as características padrão.

No que diz respeito ao parâmetro do eixo, este contém informações referentes aos fins de curso e medidas do eixo, caso seja necessário que o *driver* faça esta gestão. Como o controlo da posição dos motores é realizado pelo controlador, a única alteração efetuada aqui foi o tipo de fim de curso, introduzindo NO, para que o *driver* não bloqueie a movimentação.

Configuração de Controlo

Os parâmetros do controlo estão divididos em várias partes: *closed loop*, *control interface*, *jog mode*, *error management* e *trace configuration*. Os dois primeiros grupos são os mais importantes, enquanto o *jog mode* é útil quando a operação manual é realizada diretamente pelo *driver*. O *error management* é um sistema de controlo de diversos avisos, permitindo ignorar alguns ou dar mais importância a outros. O *trace* será abordado na Subsubsecção de *trace data*.

O *closed loop*, visualizado na Figura 5.7, permite efetuar o controlo de ganhos, quando o *driver* possui o controlo do movimento.

Closed Loop Settings for unsupported configuration (manually changed)	
Current Control Gain: 1.25 Time Constant: 1.13 ms	
Velocity Control Gain: 0.67 Time Constant: 14.21 ms Actual Velocity Filter: 2.00 ms	
Position Control Gain: 0.28 Max. Correction Velocity: 216.67 mm/s Dead Range: 0.00 mm	
Application Data Total Load: 2.500 kg Inertia Ratio:	
Soft — Middle — Hard Calculate	
Enable Editing	

Figura 5.7: Parâmetro do *closed loop*

No que se refere ao *control interface*, são apresentados os parâmetros que controam toda a interface, manipulando as entradas digitais, analógicas e de sincronismo.

É importante ressaltar as entradas digitais e de sincronismo, conforme ilustrado na Figura 5.8. Nas entradas digitais, é possível observar o que cada *input* e *output* representam, enquanto as de sincronismo representam a forma de controlo, recebendo ondas CW/CCW, semelhantes às que os terminais EL2522 fornecerão.

Configuração do *Trace Data*

O FCT inclui uma aba que permite efetuar um *trace*. Embora não seja possível realizar um *trace* em tempo real, permite visualizar a captação dos sinais após o

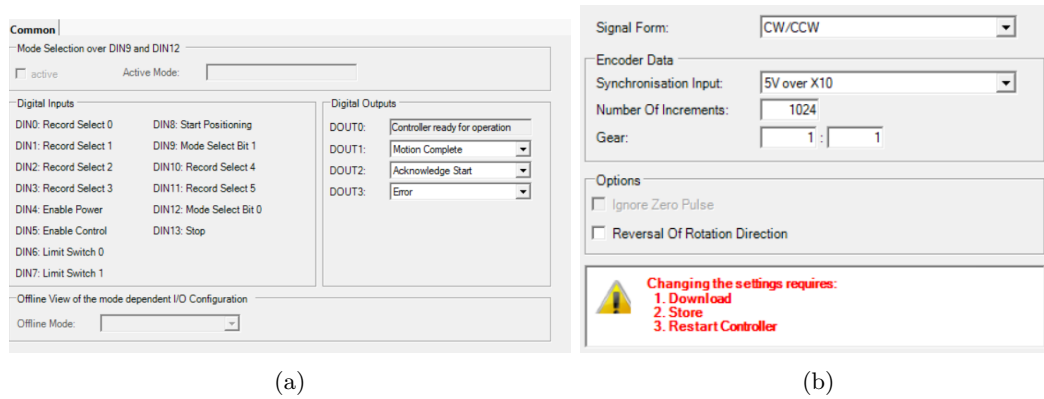
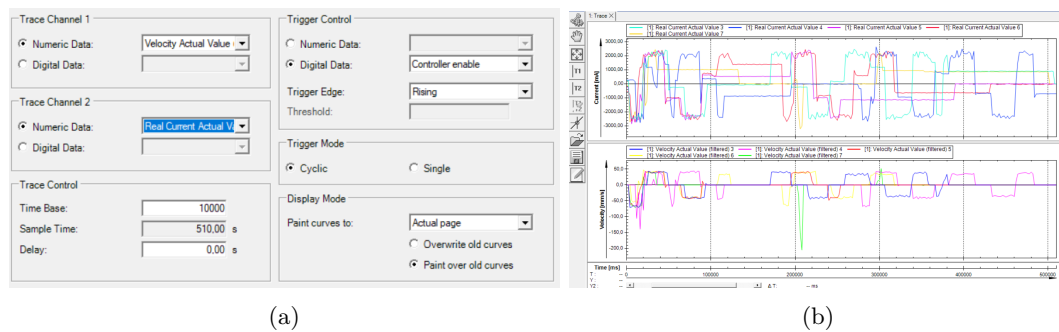


Figura 5.8: Parâmetros de *control interface*: a) entradas digitais b) sincronismo

tempo de análise programado no controlador, conforme ilustrado na Figura 5.9a. Nesta aba, podem ser captados sinais numéricos como: velocidade, corrente atual e reativa, posição do rotor e tensão. Também é possível captar sinais digitais, como os *switches*, erros, movimento completado, *homing* completo e *enable* do motor.



- X2 - *encoder*;
- X3 - STO interface;
- X5 - RS232/RS485, permitindo a comunicação entre computador e *driver*;
- X6 - motor;
- X9 - *power supply*;
- X10 - *master/slave* para receber os sinais de controlo.

Input/Output Interface

Esta porta, com um conector DB25, apresenta *inputs* e *outputs* que diferem conforme o modo em que o *driver* é configurado. Como o modo a usar é o sincronismo, é necessário respeitar a tabela indicada pelo *datasheet* [63]. Entre todos os pinos, vale a pena destacar os seguintes:

- DIN 4 (pino 21) - *output stage enable (high active)*;
- DIN 5 (pino 9) - *enable* do controlador (*high active*);
- DIN 13 (pino 15) - *stop (low active)*;
- DOUT 0 (pino 24) - controlador disponível (*high active*);
- DOUT 3 (pino 13) - erro do controlador (*low active*).

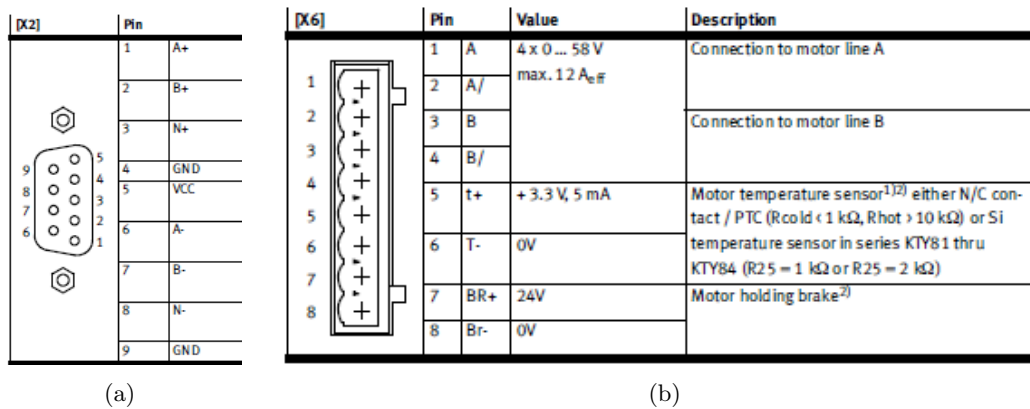
Enquanto as entradas 4 e 13 estão ligadas diretamente ao relé e à massa, os restantes pinos mencionados estão conectados ao controlador.

Encoder e Motor

As ligações para o motor são efetuadas pelos terminais do *encoder* (X2) e do motor (X6). Enquanto o *encoder* possui um conector DB9 de entrada para o *driver*, o motor é conectado por cabos soltos, conforme visualizado na Figura 5.10, respetivamente.

STO interface

O *Safe Torque Off* (STO) é uma função de segurança que o controlador possui. Sem a conexão do circuito base, o *driver* torna-se inoperacional. Para isso, é necessário ligar o relé ao pino 2 da porta X3, permitindo o controlo do *driver*. Quando o relé é desativado, o *driver* entra no modo *safe*, desligando qualquer controlo que estava a ser realizado naquele momento.

Figura 5.10: Conexão para o motor [63]: a) *Encoder* b) *Motor*

Alimentação do *Driver*

A alimentação, como o próprio nome indica, é referente à porta X9, contém os conectores para receber energia para o *driver*, bem como a alimentação para o motor (pino ZK), em referência à massa.

Master/Slave Interface

Com uma porta DB9, esta interface permite receber as informações do *train* de pulsos. Para tal, foram ligados os pinos A, B e os negativos correspondentes, conforme visualizado na Figura 5.11. Não é necessário ligar a massa e VCC, uma vez que os níveis de tensão são iguais, pois partilham a mesma massa.

5.3 Configurações em Programa *Point-to-Point* (PTP) no TwinCAT

Para inicializar e ambientar-se com o TwinCAT, foi necessário desenvolver um pequeno programa que permitisse realizar testes com os eixos e outros pormenores. Antes de executar o código com os eixos, foi imprescindível configurar os terminais e, em seguida, o eixo, de modo a garantir que o código funcionasse corretamente.

Esta Secção foi dividida em três: configuração de terminais, configuração do eixo e código. Em cada tópico, será abordado o conteúdo correspondente.

5.3.1 Configuração dos Terminais

Para permitir a utilização dos terminais introduzidos no TwinCAT, é necessário seguir alguns passos para que a conexão entre os sinais obtidos e o controlo seja a mais adequada. Assim, será explicada, passo a passo, a maneira correta de introduzir *input/output*. É importante mencionar que, no programa com o código CNC, os procedimentos serão semelhantes.

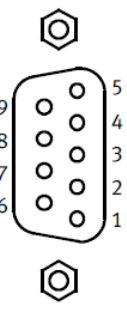
[X10]	Pin	Designation	Value	Description
	1	A CLK CW	5 V DC Ri = 120 Ω max. 150 kHz	- Tracking signal A - Pulse CLK - Pulses clockwise CW - Positive polarity in accordance with RS422
	2	B DIR CCW	5 V Ri = 120 Ω max. 150 kHz	- Tracking signal B - Direction DIR - Pulses counterclockwise CCW - Positive polarity in accordance with RS422
	3	N	5 V Ri = 120 Ω max. 150 kHz	- Incremental encoder zero pulse N - Positive polarity in accordance with RS422
	4	GND ¹⁾	–	Reference GND for incremental encoder
	5	VCC	+5 V ± 5 %, 100 mA	Auxiliary supply, max. load 100 mA, short-circuit proof
	6	#A #CLK #CW	5 V Ri = 120 Ω max. 150 kHz	- Tracking signal A - Pulse CLK - Pulses clockwise CW - Negative polarity in accordance with RS422
	7	#B #DIR #CCW	5 V Ri = 120 Ω max. 150 kHz	- Tracking signal B - Direction DIR - Pulses counterclockwise CCW - Negative polarity in accordance with RS422
	8	#N	5 V Ri = 120 Ω max. 150 kHz	- Zero pulse N - Negative polarity in accordance with RS422
	9	GND ¹⁾	–	Screening for the connecting cable

Figura 5.11: Conexão do *Master/Slave* [63]

Criação da interface I/O do TwinCAT

A criação da interface para as entradas e saídas é crucial na abordagem com o *hardware*.

Para isso, na parte de *device*, que se encontra em I/O no *Solution Explorer* (aba da esquerda do ambiente principal do TwinCAT), ao clicar com o botão direito, surge uma aba semelhante à da Figura 5.12. A partir daí, cria-se um novo item, escolhendo *EtherCAT Master*, pois é uma estação principal sem ramificações.

Após este passo, e repetindo o processo da mesma forma, mas agora em cima do *device* criado, cria-se um novo item com o acoplador utilizado, neste caso, o EK1100. Para adicionar os restantes terminais, de forma semelhante, em cima do terminal EK1100, juntam-se os restantes terminais, resultando em algo semelhante à Figura 5.13.

Existe uma outra forma de realizar esta configuração: quando o sistema está fisicamente conectado ao *hardware*, pode ser efetuado um *scan* para obter as informações sobre o acoplador e os respetivos terminais ligados.

onde, na parte do EL1008, surgem 8 canais, conforme visualizado na Figura 5.14. Estes 8 canais correspondem aos *inputs* existentes no terminal físico. Em cada canal, ao abrir as definições, encontra-se um *input* e a respetiva página. Aqui, são apresentados os parâmetros da respetiva entrada do terminal. Ao clicar em *Linked to...*, aparecerão todas as variáveis de entrada disponíveis para associação, permitindo escolher a mais adequada para a respetiva entrada.

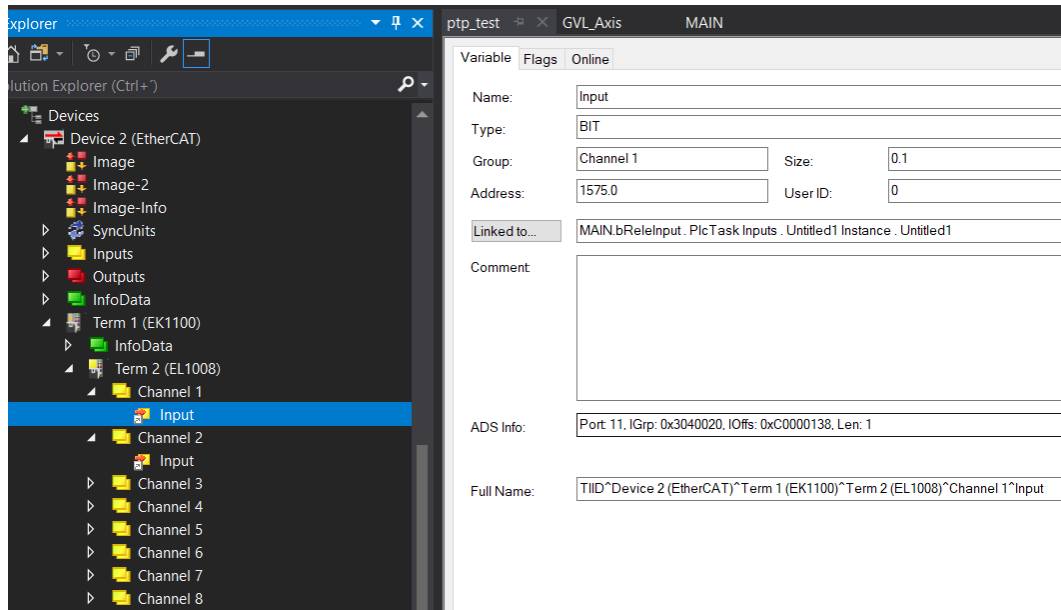


Figura 5.14: Canais de *Input*

Configuração do Terminal EL2008

De forma semelhante ao terminal EL1008, o terminal de saída EL2008 segue os mesmos procedimentos, mas com uma pequena diferença.

No momento de declarar variáveis, na mesma zona, as variáveis devem ser declaradas como saídas físicas, conforme demonstrado no Algoritmo 5.2.

- 1 bName_Variable AT %Q* : BOOL; ▷ Uma Variável de Saída
- 2 bName_Array_Variable AT %Q* : ARRAY[1..n] OF BOOL; ▷ Um *Array* de n Variáveis de Saída

Algoritmo 5.2: Declaração de variáveis físicas de saída

No que diz respeito à configuração em I/O, os passos são exatamente iguais aos do EL1008, com a diferença de que os canais aparecem em vermelho.

Configuração do Terminal EL2522

Relativamente ao terminal EL2522, este apresenta algumas diferenças. A declaração deste é realizada diretamente a partir do eixo.

O primeiro passo consiste em abrir as definições do EL2522 e, em *Process Data*, escolher o modo para dois canais com posição contínua.

Após isso, deve-se acessar o eixo, que se encontra na aba dos *Motion*, e em *Settings*, clicar em *Link to I/O*, conectando com o canal respectivo, conforme visualizado na Figura 5.15 [19].

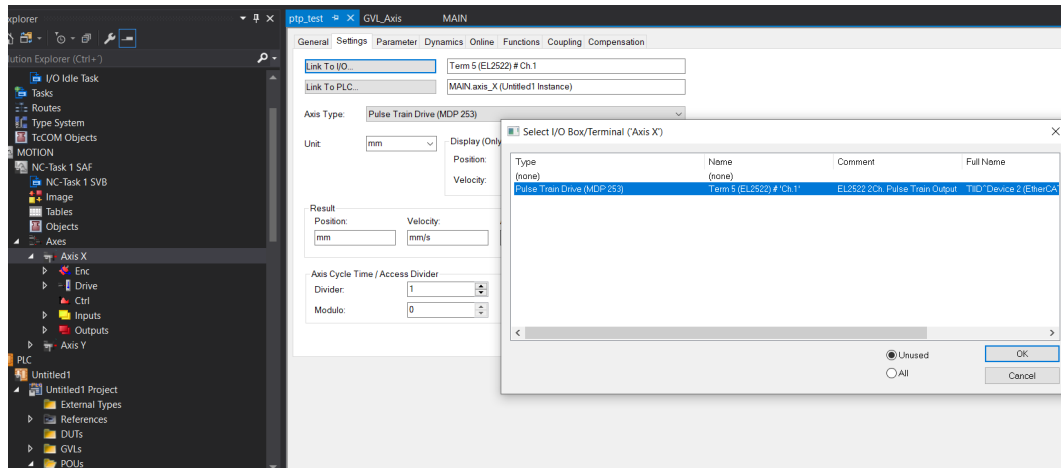


Figura 5.15: Configuração dos eixos referente ao canal no terminal EL2522

5.3.2 Configuração dos Eixos

Relativamente à configuração dos eixos, existe uma diferença significativa entre os eixos que utilizam PTP e os que utilizam CNC. Assim, a única configuração que teve relevância nesta fase foi a das calibrações.

Para efetuar a calibração, o parâmetro base utilizado encontra-se no *encoder* do respetivo eixo, e existem dois parâmetros designados como *Scaling Factor Numerator/Denominator*.

Estes dois parâmetros estão interligados com os pulsos enviados, permitindo uma calibração próxima da realidade.

No caso desta aplicação, o *numerator* manteve-se em 0,05, enquanto o *denominator* era o valor que seria modificado.

Para realizar uma calibração em PTP, devem ser efetuados várias trajetórias de diferentes distâncias, com o *denominator* igual a 1 (D), e registou-se o valor obtido pela distância percorrida quando o eixo não estava calibrado (d_{obtida}). Este valor será comparado com a distância pretendida (d), e, utilizando a seguinte equação, é possível calcular o denominador de calibração, designado de D_c . Após a alteração do denominador, é realizado um novo teste para confirmar se a calibração foi efetuada com sucesso.

$$D_c = \frac{D * d}{d_{obtida}} \quad (5.2)$$

5.3.3 Código Efetuado para PTP

A realização de um código exemplo em PTP teve como objetivo testar os motores, a tecnologia a ser utilizada e o ensaio dos motores, permitindo observar o comportamento de cada um.

Para controlar os motores, foi necessário estudar os blocos de funções referentes aos movimentos dos motores existentes no TwinCAT, os quais estão discretamente introduzidos nos códigos CNC. Para este estudo, foram analisados com atenção dois blocos de funções: o *MC_Power* e o *MC_MoveAbsolute* [19].

O *MC_Power* tem como função a ativação via *software* de um eixo. Sem este bloco de funções, não é possível controlar os eixos, uma vez que o *software* considera que os motores estão desligados. É comum ligar este bloco de função ao *enable* dos motores.

Por outro lado, o *MC_MoveAbsolute* tem como objetivo percorrer a distância mencionada, controlando diretamente o eixo referenciado. Este bloco foi utilizado para observar o comportamento do motor em relação às respectivas distâncias [19].

O código desenvolvido utilizou a manipulação de sinais e valores, de forma a proporcionar um controlo mais preciso de acordo com as necessidades. O código teve como base a ativação do bloco de funções *power* e, em seguida, a possibilidade de realizar movimentos. Este código também serviu para outros testes, como a introdução de *hardware*. O código utilizado estará disponível no Anexo C, onde as variáveis encontram-se descritas no Algoritmo C.1, enquanto a função está no Algoritmo C.2.

5.4 Configurações em Programa *Computer Numerical Control* no TwinCAT

Após um estudo inicial do código-base previamente fornecido e um período de ambientação com códigos PTP, foi realizado um estudo aprofundado sobre as melhorias do código-base, visando maximizar o aproveitamento do uso deste *software*.

Para compreender como foi efetuado o estudo das configurações e do código, este foi dividido em várias categorias: configuração do eixo, configuração de alarmes, configuração do *spindle*, configurações diversas, consola HMI e código modificado.

Exceto as duas últimas categorias, as restantes são parâmetros incluídos na parametrização da CNC do TwinCAT, sendo que a única necessidade foi o estudo dos respetivos parâmetros e das suas funções. Relativamente ao HMI, foram efetuadas algumas alterações básicas para obter uma consola mais limpa e intuitiva. Quanto ao

código modificado, serão apresentadas algumas funções implementadas ou alteradas, bem como as suas principais funcionalidades.

5.4.1 Configuração do Eixo

Para permitir que o eixo percorra a distância desejada, é necessário ajustar parâmetros fundamentais que o pacote CNC no TwinCAT proporciona.

Este processo foi dividido em diversas partes: calibração e ganho, dinâmica, posição de *lag* e de controlo, *homing* e outras configurações.

Calibração e Ganho do Eixo

O sistema de calibração e ganho é essencial para que o eixo corresponda às exigências. Sem uma calibração adequada, quando é solicitado que o eixo percorra uma determinada distância, ele pode percorrer um valor não estipulado. Assim, é necessário realizar uma calibração, utilizando os valores fornecidos pelo *driver* como referência para a distância percorrida.

O ganho, por sua vez, consiste num parâmetro de fator proporcional, cujo valor é ajustado conforme os testes realizados. Este não deve ser um valor muito baixo, pois isso aumentaria o erro de seguimento, nem um valor muito alto, uma vez que os sistemas CNC não requerem *overshoot*, o que poderia afetar o resultado final.

Os parâmetros-chave para as configurações são os seguintes [19]:

- P-AXIS-099 - fator proporcional k_v para o controlo posicional;
- P-AXIS-233 - resolução da distância de medição (denominador) (mm);
- P-AXIS-234 - resolução da distância de medição (numerador) (Incrementos).

A calibração é efetuada de forma diferente em comparação com o PTP. Enquanto no programa anterior se altera apenas o numerador, no caso do CNC, o que se altera é o denominador, uma vez que o numerador só aceita números reais.

Para realizar a calibração, utiliza-se a seguinte equação, onde x corresponde ao valor resultante entre o numerador e o denominador dos parâmetros 233 e 234, d corresponde à distância pretendida e d_{obtida} corresponde à distância percorrida com o eixo não calibrado. Os valores com índice nc correspondem aos valores antes da calibração, enquanto os mencionados com c correspondem ao valor calibrado.

$$x_{nc} = \frac{N_{nc}}{D_{nc}} \quad (5.3)$$

$$x_c = \frac{d \cdot x_{nc}}{d_{obtida}} \quad (5.4)$$

$$x_c = \frac{N_c}{D_c} \quad (5.5)$$

Para efetuar a calibração, o primeiro passo consiste em calcular o valor de x_{nc} usando a Equação 5.3. Após descobrir o respetivo valor, efetuar uma trajetória pretendida, registando o valor percorrido efetivamente. Após isso, é possível calcular os valores para a calibração usando a Equação 5.4, introduzindo a distância obtida e a distância pretendida. Ao descobrir o valor de x_c , usou-se a equação 5.5 e ajustou-se o numerador e o denominador, permitindo aproximar os valores calculados na Equação 5.4. Após introduzir os novos numeradores e denominadores dos parâmetros, efetuou-se um novo teste para verificar se o eixo ficou calibrado.

Dinâmica do Eixo

A dinâmica do eixo é uma das partes cruciais do sistema. Sem a dinâmica, não é possível escolher as acelerações e desacelerações dos eixos, velocidades máximas, entre outros parâmetros.

Existem vários parâmetros referentes à dinâmica, principalmente velocidades máximas em determinados contextos, como o *homing*, *rapid mode* ou manual. Os destaques vão para os seguintes parâmetros [19]:

- P-AXIS-003 - desaceleração em paragem de emergência (m/s^2);
- P-AXIS-008 - aceleração máxima permitida (m/s^2);
- P-AXIS-212 - velocidade máxima permitida (m/min).

Definições de *Lag* e de Controlo

Um dos parâmetros que necessitou de ajuste foi o erro de seguimento, ou nos parâmetros como *lag*, e do controlo. O erro de seguimento refere-se ao atraso existente entre o valor real no eixo e o valor de referência do eixo. Como o sistema não recebe informações do *encoder* via *drivers*, ele simula que o sistema está a operar conforme esperado.

Para isso, foi necessário visualizar alguns parâmetros, como [19]:

- P-AXIS-108 - diferença máxima de posição permitida. Este parâmetro visualiza a diferença entre o que é ordenado e o ideal; se a diferença for superior ao valor deste parâmetro, a operação é interrompida;
- P-AXIS-168/9 - máximo e mínimo para o valor de *lag*. Se ultrapassados, é criado um erro;
- P-AXIS-172 - tipo de monitorização (1 - monitorização *standart*);
- P-AXIS-176 - supressão do valor de *lag*. Quando está a "1", suprime a monitorização e a saída de um erro.

Adicionalmente, existe um parâmetro designado de P-AXIS-236, que consiste numa janela de posição para paragem exata, definindo o intervalo desejado para a paragem. O TwinCAT utiliza códigos específicos para este controlo, sendo eles o código G60 (linha) ou os códigos G360/359 (bloco).

Configurações de *Homing*

Relativamente ao *homing*, existem vários parâmetros a configurar para adequar a aplicação desejada.

Antes de verificar os parâmetros, é necessário compreender como se pretende realizar o *homing*. Para este trabalho, o *homing* é realizado com o uso de uma *cam*, um fim de curso usado para efetuar o *homing*, sem referência a um pulso zero, conforme demonstrado na Figura 5.16a.

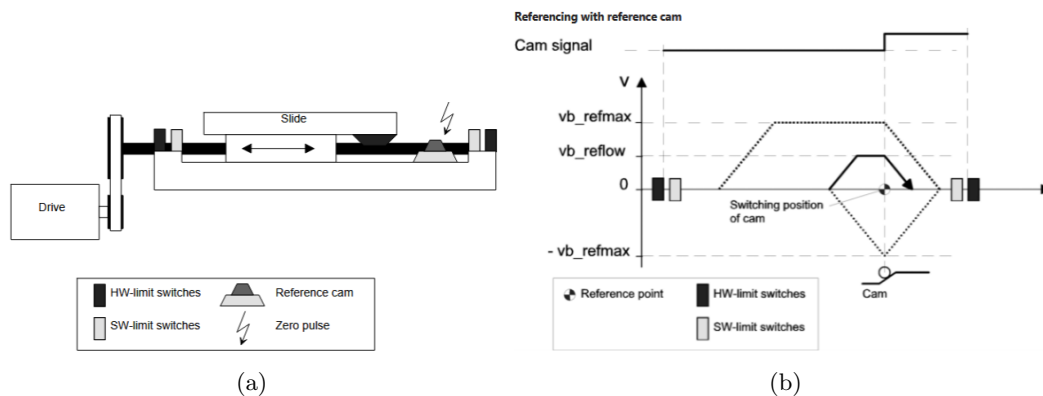


Figura 5.16: Realização do *homing* [19]: a) *Hardware* b) *Movimentação do eixo*

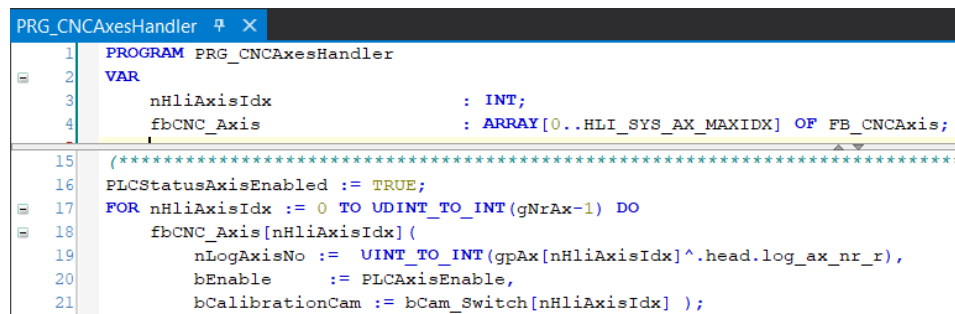
Na Figura 5.16b é visualizada a trajetória que o eixo irá percorrer. Inicialmente, o eixo move-se na direção onde se encontra a *cam*, até receber um sinal do dispositivo. Após receber o sinal, o movimento é desacelerado e para, permitindo efetuar um movimento inverso de baixa velocidade até sair da *cam*. Após sair da *cam*, o movimento é invertido novamente, realizando um movimento lento até tocar na *cam*, registrando nesse momento a referência.

Após analisar a forma como o *homing* será efetuado, pode-se proceder à escolha dos parâmetros, considerando os seguintes [19]:

- P-AXIS-064 - movimento rápido/ lento a partir da *cam*. Foi escolhido o valor *default* ('1'), pois faz a alteração de velocidade rápida para lenta da atuação do sinal da *cam*;
- P-AXIS-084 - *homing* efetuado somente com uma *cam*, sem o pulso zero. O valor deste parâmetro é '1', já que indica as condições que são pretendidas;

- P-AXIS-156 - *homíng* sem *cam*. Como é utilizado uma *cam*, o valor deste parâmetro é '0';
- P-AXIS-157 - *homíng* sem reversão. Como é efetuada a reversão, o valor do parâmetro é '0';
- P-AXIS-158 - direção para efetuar o *homíng*. Foi escolhido o movimento no sentido negativo, ou seja, '0';
- P-AXIS-218 - velocidade lenta para detecção exata do sinal de referência;
- P-AXIS-219 - velocidade rápida para detecção do sinal de referência;
- P-AXIS-329 - uso dos limites físicos para *homíng*. Neste caso, o valor deste parâmetro é '0'.

Para concluir a configuração de *homíng*, é necessário introduzir os sinais da *cam* diretamente no código. Para isso, os sinais obtidos pelas *cam* dos eixos foram introduzidos num *array*. Em "PRG_CNCAxesHandler", foi necessário introduzir na variável *bCalibrationCam* o *array* dos sinais, como verificado na Figura 5.17.



```

PRG_CNCAxesHandler
1  PROGRAM PRG_CNCAxesHandler
2  VAR
3      nHliAxisIdx          : INT;
4      fbCNC_Axis           : ARRAY[0..HLI_SYS_AX_MAXIDX] OF FB_CNCAxis;
5
6  (*****
7  15  PLCStatusAxisEnabled := TRUE;
8  16  FOR nHliAxisIdx := 0 TO UDINT_TO_INT(gNrAx-1) DO
9  17      fbCNC_Axis[nHliAxisIdx] (
10 18          nLogAxisNo := UINT_TO_INT(gpAx[nHliAxisIdx]^head.log_ax_nr_r),
11 19          bEnable     := PLCAxisEnable,
12 20          bCalibrationCam := bCam_Switch[nHliAxisIdx] );
13 21

```

Figura 5.17: Configuração dos sinais *homíng*

Outras Configurações de Eixo

Para além dos parâmetros mencionados neste subcapítulo, existem outros que são importantes, como os limites de *software* ou o modo de controlo. Existem vários parâmetros, sendo que alguns são exclusivamente utilizados para determinadas tecnologias, como o CANopen.

O modo de operação é um dos parâmetros cruciais. Consoante o modo escolhido, a máquina CNC irá comportar-se de acordo com o parâmetro escolhido. Para este trabalho, foi escolhido o CNC_POSITION_CONTROL, uma vez que é o controlador que gere as respetivas posições.

Outros parâmetros que podem ser relevantes são os *software limit switch* (P-AXIS-177/8), que definem o limite inferior e superior, respetivamente. Quando o eixo ultrapassar estes limites, o movimento será interrompido e será ativado um

aviso. É comum utilizar estes limites antes dos limites físicos, que são declarados no PLC, onde está introduzida a respetiva lógica [19].

5.4.2 Configuração de Alarmes

O conceito de alarmes é uma forma de alertar o operador da máquina sobre erros, informações ou avisos. O pacote TF5200 já inclui um sistema de alarmes que informa o operador sobre falhas ou avisos que ocorrem na máquina, além de uma extensa documentação sobre os respetivos erros [19].

É possível criar alarmes personalizados. Para isso, na aba *System*, existe uma definição designada de *Type System*, onde os alarmes estão agrupados. Nesse espaço, há uma subseção chamada *Alarms*, onde aparecem todos os grupos de alarmes ativos, permitindo a criação de novos grupos.

Para criar um alarme específico, seleciona-se o grupo desejado para edição, aparecendo uma aba semelhante à da Figura 5.18a. Nesta nova aba, há uma secção de eventos, e ao seleccioná-la, aparecem todos os alarmes existentes do grupo, como visualizado na Figura 5.18b.

Cada alarme contém um nome, um ID, um texto e um grau de severidade, que são configurados ao criar ou editar o alarme, conforme ilustrado na Figura 5.18c. O utilizador pode configurar o nome, ID e texto, assim como os comentários a serem exibidos, além da severidade, que pode variar entre informativo, aviso, erro ou crítico.

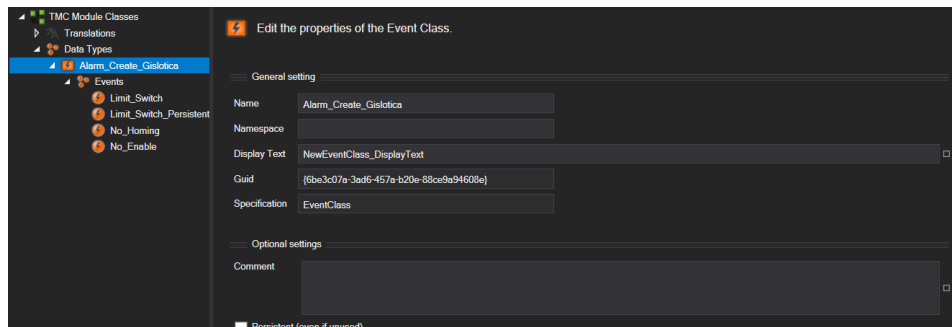
Após criar os alarmes, é necessário declará-los no código. Para isso, foram criadas várias funções de ação, que são chamadas no decorrer do código quando é necessário ativar ou desativar um alarme. Estas funções, como demonstrado no Algoritmo 5.3, foram criadas para que, quando a variável estiver em condições de ativar o alarme, seja emitido um sinal para o alarme em questão, referenciado na variável de eventos da função *fb_Alarm*. Quando o sinal é desligado, ou seja, quando o alarme termina, é emitido um sinal *cleared*, introduzido um valor falso no *raise* para remover o alarme do ecrã.

```

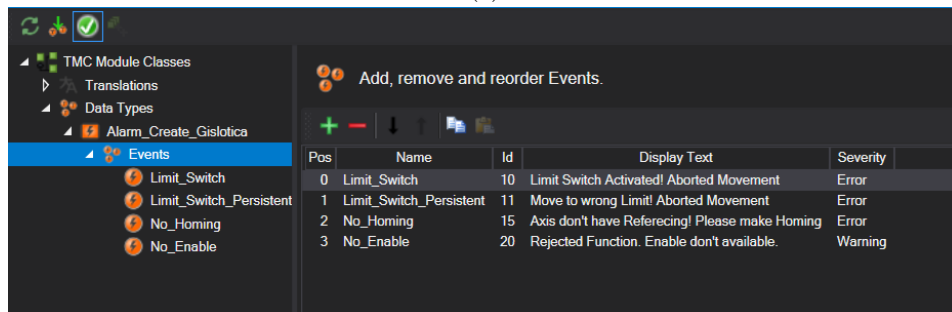
1 IF NOT stCmdModesStates.stModes.Homing THEN
2     fb_Alarm(Event := TC_EVENTS.Alarm_Create_Gislotica.No_Homing,
3             Raise := TRUE);
4 ELSE
5     fb_Alarm(Event := TC_EVENTS.Alarm_Create_Gislotica.No_Homing,
6             Raise := FALSE);
7 END_IF

```

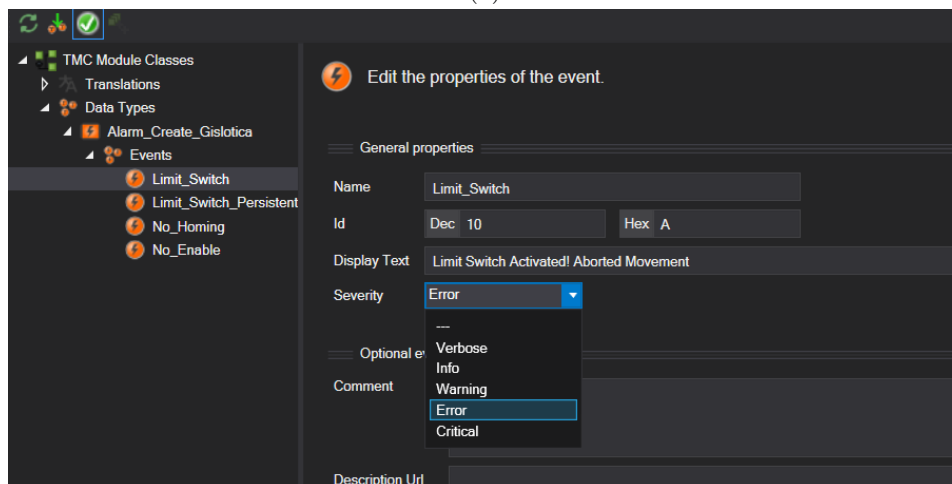
Algoritmo 5.3: Exemplo de uma função de ação para ativação de alarmes



(a)



(b)



(c)

Figura 5.18: Edição de alarmes: a) Menu b) Alarmes existentes c) Edição de alarme

5.4.3 Configuração do *Spindle*

O *spindle* é um dos componentes importantes para o funcionamento adequado da máquina CNC desenvolvida. Mesmo que não seja aplicado na implementação, foi necessário configurá-lo para ter valores simulados. Para isso, foram ajustadas algumas configurações do canal que contém todos os eixos.

No canal dos eixos, encontram-se diversos parâmetros necessários para a configuração, sendo os seguintes:

- P-CHAN-036 - número do eixo lógico. Este parâmetro é semelhante ao P-CHAN-051;
- P-CHAN-051 - número do eixo onde está o *spindle* principal. O valor a ser introduzido é '4', uma vez que os valores de '1' a '3' correspondem aos eixos;
- P-CHAN-069 - controlo do *spindle* via PLC. Como o controlo do *spindle* está do lado da CNC, este parâmetro é definido como '0';
- P-CHAN-082 - número de *spindles* configuráveis. Neste caso, apenas um está disponível.

Para disponibilizar velocidades maiores, consoante o *spindle* utilizado, é necessário configurar parâmetros do eixo, como [19]:

- P-AXIS-205 - normalização da velocidade (denominador);
- P-AXIS-206 - normalização da velocidade (numerador);
- P-AXIS-320 - modo de operação do eixo. Neste caso, foi escolhido o parâmetro OPEN_POSITION_LOOP_NO_FEEDBACK.

Para a escolha dos parâmetros em P-AXIS-205 (D_v) e P-AXIS-206 (N_v), é necessário considerar os parâmetros do *spindle*. É preciso determinar quantos incrementos são necessários para realizar uma determinada rotação. Usando as informações do exemplo fornecido pela Beckhoff no documento de configuração do *spindle* [19], para uma rotação de 3000 RPM (R), são lidos 32 767 incrementos, sendo este valor introduzido no numerador (N_v). Contudo, é necessário converter as rotações para $^\circ/min$ e depois para o formato interno, utilizando a seguinte equação:

$$D_v = R.360(^\circ/min).1000 = 1080000000 \quad (5.6)$$

5.4.4 Configurações Diversas

Com o prolongar dos testes e devido a algumas falhas identificadas, foi necessário explorar o funcionamento do TwinCAT. Durante esse processo, foram encontradas as *tasks*. O TwinCAT fornece, em cada implementação, seja em sistema, *motion* ou PLC, diferentes *tasks* configuráveis. Assim, na *PlcTask* do sistema e na *CNC-Task GEO* visualizada no *Motion*, existe um parâmetro designado por *cycle ticks*.

Antes da alteração dos valores, o parâmetro *cycle ticks* estava definido como 10ms. No entanto, em alguns testes, foi verificado um *overshoot*, o que levou ao estudo de vários parâmetros para mitigar a falha.

Como resultado, foi introduzido o *cycle ticks* mínimo permitido, ou seja, 2 ms. Após a realização de novos testes, constatou-se que o *overshoot* desapareceu, uma

vez que o tempo de cálculo da posição atual e da posição de referência foi reduzido, permitindo um cálculo mais preciso e com menor erro, além de um aumento na velocidade.

5.4.5 Consola HMI

O pacote TF5200 fornece uma consola base, utilizada para programas de teste, conforme demonstrado na Figura 5.19a. No entanto, a modificação do código-base é complexa, uma vez que a aplicação possui um código fechado, exceto a interface principal dos botões, que está descrita num ficheiro '.xml'.

Apesar disso, foi possível alterar botões e imagens, tornando a consola mais intuitiva para a sua utilização, como ilustrado na Figura 5.19b.

Na barra inferior, à esquerda, aparecem os cinco modos possíveis: automático, manual, MDI, *Homing* e *standby*. Cada função possui o seu painel, permitindo um controlo mais adequado para o respetivo modo. Os botões *start* e *stop* são indicativos para operações como o automático e MDI, iniciando ou parando as operações conforme o nome sugere.

É importante destacar o botão *Enable*, localizado na barra da direita na parte inferior, que ativa os eixos. Quando os eixos estão com o *enable* ativo, aparece um pequeno quadrado correspondente ao respetivo eixo. Durante o processo de *homing*, o retângulo do eixo muda de cor, passando de vermelho (indicando um eixo não referenciado) para verde.

A barra superior indica o estado da máquina, bem como os avisos ou erros existentes.

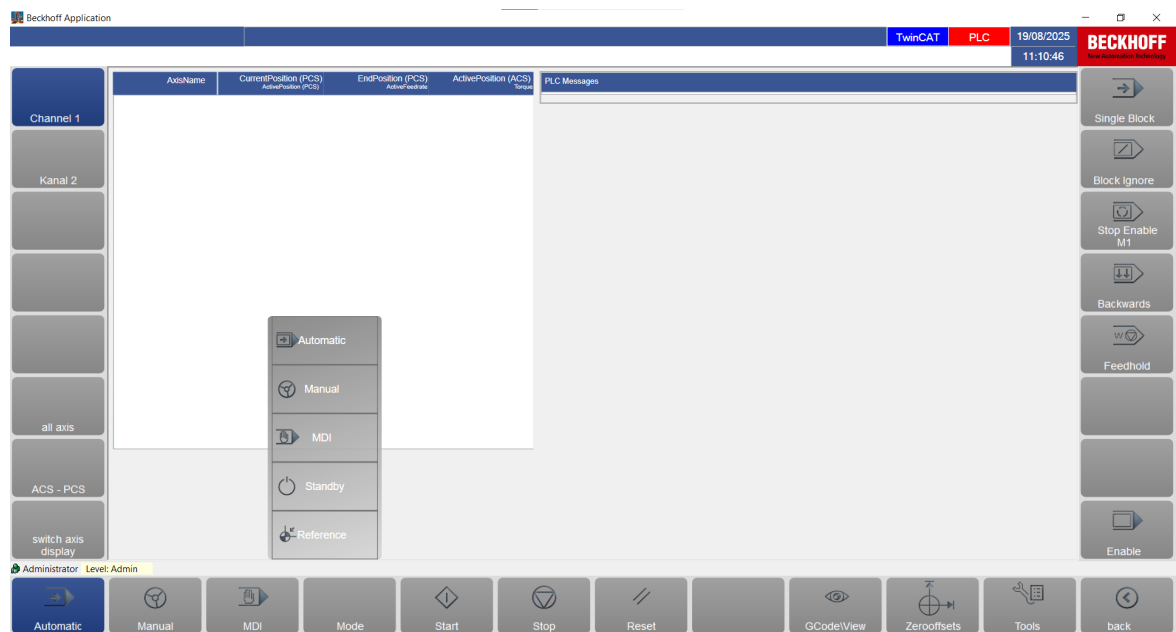
Além das funções principais, existem três funções adicionais: *GCodeViewer*, *ZeroOffset* e *Tools*. O *GCodeViewer* exibe o que o código G selecionado no modo automático irá executar, mostrando os movimentos a serem realizados. Para alterar os ângulos, é necessário pressionar os dois botões do rato simultaneamente e inserir o plano desejado.

O *ZeroOffset* é uma interface que permite alterar os *offsets* de forma prática, enquanto os *tools* fornecem informações sobre as características das ferramentas. Todas as funções mencionadas estão ilustradas na Figura 5.20.

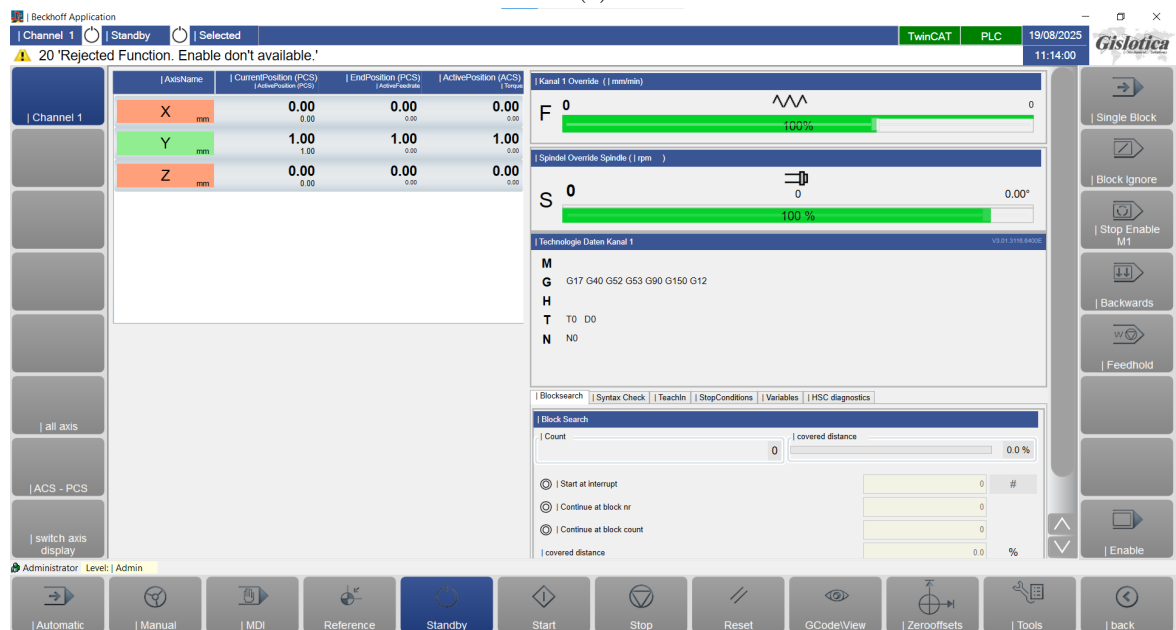
5.4.6 Alterações do Código Base

O código fornecido já executava as funções básicas, permitindo o controlo dos eixos a partir de um código G e a verificação de posições. No entanto, foram realizadas algumas alterações e correções para melhorar o desempenho do sistema, tais como:

- correção do erro na execução do *homing*;
- melhoria na lógica de controlo dos eixos, incluindo a verificação de todos os componentes;



(a)

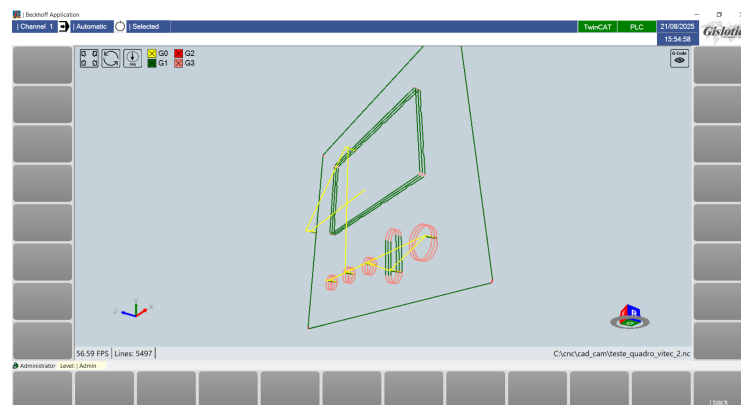


(b)

Figura 5.19: Menu principal do HMI: a) Antes da modificação b) Após modificação

- ajustes na lógica dos limites de fim de curso;
- implementação de lógica dos semáforos para leitura;
- obtenção de dados do interpretador.

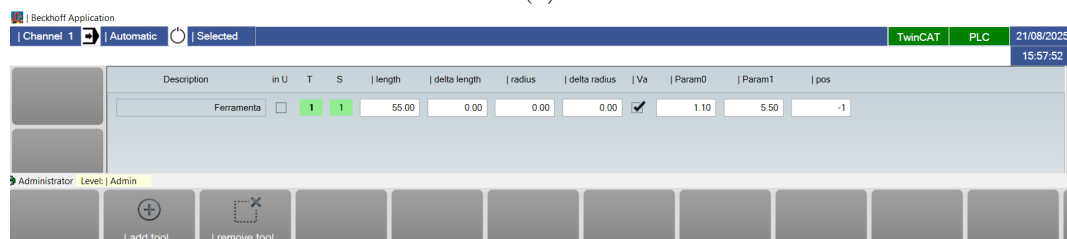
Para uma melhor organização desta subsecção, esta será dividida em vários tópicos, que incluem: controlo de *input/ output*, controlo do PLC, obtenção de dados



(a)



(b)



(c)

Figura 5.20: Funções extras do HMI: a) *GCodeViewer* b) *ZeroOffset*
c) *Tools*

do interpolador e adaptação de funções-base.

O desenvolvimento do código baseia-se na linguagem *Structured Text* (ST), que é uma linguagem de alto nível baseada em texto, permitindo o controlo de PLC. Serão introduzidos os códigos e as declarações de variáveis correspondentes a todos os

códigos mencionados no Anexo C, onde na primeira secção se encontram as variáveis globais, descritas nos Algoritmos C.3 e C.4, enquanto as secções seguintes referem a declaração de variáveis e respetivo código.

Controlo de *Input/Output*

Uma das partes fundamentais que não existia no código base é o controlo baseado em *inputs* e o controlo de *outputs*. Por isso, foi necessário desenvolver e implementar lógica em vários ramos: leitura e estruturação de variáveis e processamento de dados, onde se inclui a lógica do semáforo e dos fins de curso. Existe ainda o controlo dos eixos, mas este será abordado numa outra subsecção.

Leitura de *Input* Referente à leitura e estruturação de dados, foi implementada uma lógica simples, uma vez que a leitura proveniente de variáveis globais não funciona, pois os valores não apresentam atualizações do seu estado, por não serem chamados.

Para isso, foi criada uma função, "PRG_Write_INPUT", que é sempre chamada a partir da *main*, de forma a obter os dados atualizados. Esta função capta os sinais que estão previamente ligados aos *inputs* e os introduz em variáveis globais, permitindo que as funções que necessitam dos respetivos sinais possam aceder a eles, como organizar algumas variáveis, como o *array* de *cam*. Foi ainda adicionada a chamada da função "PRG_Control_Light_Mode", que será mencionada ainda nesta subsecção. As variáveis declaradas na função "PRG_Write_INPUT" encontram-se no Algoritmo C.5, enquanto a sua função está descrita no Algoritmo C.6

Controlo da Coluna Luminosa de Sinalização Para o processamento de dados referentes à coluna luminosa de sinalização, foi criada uma função chamada "PRG_Light_Tower", que é chamada na *main* para atualizar o processo de piscar ou quando ocorre alguma alteração de estado, conforme mencionado na Subsecção 4.3.5.

Quando é efetuada alguma alteração da sinalização, são introduzidas como variáveis de entrada uma numeração referente à cor e uma variável booleana que indica se a respetiva cor da coluna luminosa vai piscar ou não, conforme demonstrado na Tabela 5.2.

Tabela 5.2: *Input* da função PRG_Light_Tower

Cor	Número	Piscar	Bool
Vermelho	0	Não	FALSE
Amarelo	1	Sim	TRUE
Verde	2		

Para o processo de piscar, é utilizada uma função extra designada "FB_HVAC-Blink", que se encontra na biblioteca TC2_HVAC. Esta função tem como objetivo transmitir uma saída de acordo com os *timers* internos, que são ajustados pelo utilizador enquanto a função está ativa. Para este caso, é necessário um sinal de 1 Hz, por isso, os parâmetros TimeOn e TimeOff são iguais a meio segundo.

Foi ainda introduzida uma pequena máquina de estados, que determina qual será a saída. Com essas informações, as variáveis declaradas são as que se encontram no Algoritmo C.7.

Relativamente ao código, este é dividido em três momentos: o primeiro é para obter o sinal do *timer*; o segundo é um sistema de caso onde se obtém o estado que será escolhido; e a terceira fase é o controlo de *output*, onde as variáveis estão mencionadas no Algoritmo C.7, e a função respetiva demonstrada no Algoritmo C.8.

Controlo do Coluna Luminosa de Sinalização por Modos Apesar de existir código preparado para controlar a coluna luminosa de sinalização, quando o *enable* estiver desligado, é necessário controlar o estado da coluna luminosa e alguns avisos quando o *enable* está ativo.

A primeira versão deste código estava contida na função "FB_HLI3_CNC-Channel_Upgrade", que continha variáveis próprias e de fácil uso. No entanto, a incapacidade de controlar o estado do semáforo a partir do estado atual do modo, ou seja, se estava *ready*, *active* ou em outro estado, levou à reformulação desta versão.

Assim, foi necessário pesquisar novas variáveis, encontrando a seguinte variável: "gpCh[]:channel_mc_control.mode_and_state.command_w.". Com esta variável, existem dois estados importantes: "to_state" e "to_mode". Estes estados contêm valores numéricos que correspondem aos estados, cujos valores estão mencionados na Tabela 5.3.

Tabela 5.3: Variáveis de estado e modo na máquina e respetivos valores

gpCh[]:channel_mc_control.mode_and_state.command_w.			
to_mode	Mode	to_state	State
1	<i>Stand-by</i>	2	<i>Selected</i>
2	<i>Automatic</i>	3	<i>Ready</i>
3	MDI	4	<i>Active</i>
4	<i>Manual</i>	5	<i>Hold</i>
		6	<i>Error</i>
		15	<i>Reset</i>

A lógica do código passa por visualizar o estado atual, conforme mencionado na Subsecção 4.3.5, onde se verifica o estado e se determina qual cor da coluna luminosa é o mais indicado.

Além disso, foram adicionados dois alarmes informativos: um deles é ativado quando o estado é *hold*, ou seja, quando o *stop* é ativado. O segundo alarme é relativo à terminação da maquinação de uma peça em estado automático. Para isso, foi implementada uma máquina de estados que verifica se o trabalho começou ou, após terminar a maquinação, alerta o utilizador durante dez segundos.

As variáveis desta função encontram-se no Algoritmo C.9, enquanto o código da função está descrito no Algoritmo C.10.

Controlo com Fins de Curso Relativamente ao controlo com fins de curso, este foi dividido em duas partes: a primeira consiste num controlo constante para detetar qualquer sinal proveniente dos fins de curso em questão e a sua atuação nos mecanismos para parar o respetivo eixo; a segunda fase é dedicada ao destravamento do eixo e à deteção de novos sinais limites.

Para isso, foi criada uma máquina de estados com três estados: o primeiro funciona normalmente, detetando qualquer limite; o segundo atua como intermédio do terceiro, permitindo que os sinais sejam do tipo *trigger* e possibilitando a escrita da mensagem de erro; o terceiro é responsável pela receção de dados, evitando que o eixo se mova na direção do limite ou detetando um novo limite, uma vez que o utilizador pode mover outros eixos e atingir o limite. Na Figura 5.21 encontra-se um fluxograma que resume a execução deste controlo.

No primeiro estado, designado de "NORMAL", existe uma função IF que verifica todos os sensores. Se for detetado um 'TRUE' num dos sensores, este é ativado, passando para o próximo estado, e ativa a variável global *bLimitStop*, que, por sua vez, está ligada a uma variável que provoca o *stop* imediato da máquina. Ao sair do último estado, esta função irá limpar o erro.

No segundo estado, denominado "LIMIT_DETECTED_TRIGGER_STOP", é implementado um compasso de espera e, após isso, é efetuado um *reset* ao sistema, de forma a que o utilizador não continue a executar um código G que provavelmente contém algum erro. É aqui que se determina qual foi o fim de curso ativado, chamando uma função designada de "Write_Switch", para além de escrever o erro na consola.

A função "Write_Switch", chamada nos segundo e terceiro estados, conta quantos *switch* estão ativos e regista qual dos limites está ativo. A razão para registar o número de *switches* é para fazer uma comparação com o valor anterior; se este número aumentar, os movimentos na máquina são parados novamente e é registado um *delay*. O estado da máquina muda, passando para o modo manual.

No terceiro estado, denominado "IN_LIMITED", é efetuada uma nova leitura da função "Write_Switch" para detetar se foi acionado um novo limite. Se este último não for detetado, verifica-se se o eixo está em movimento e se o último valor

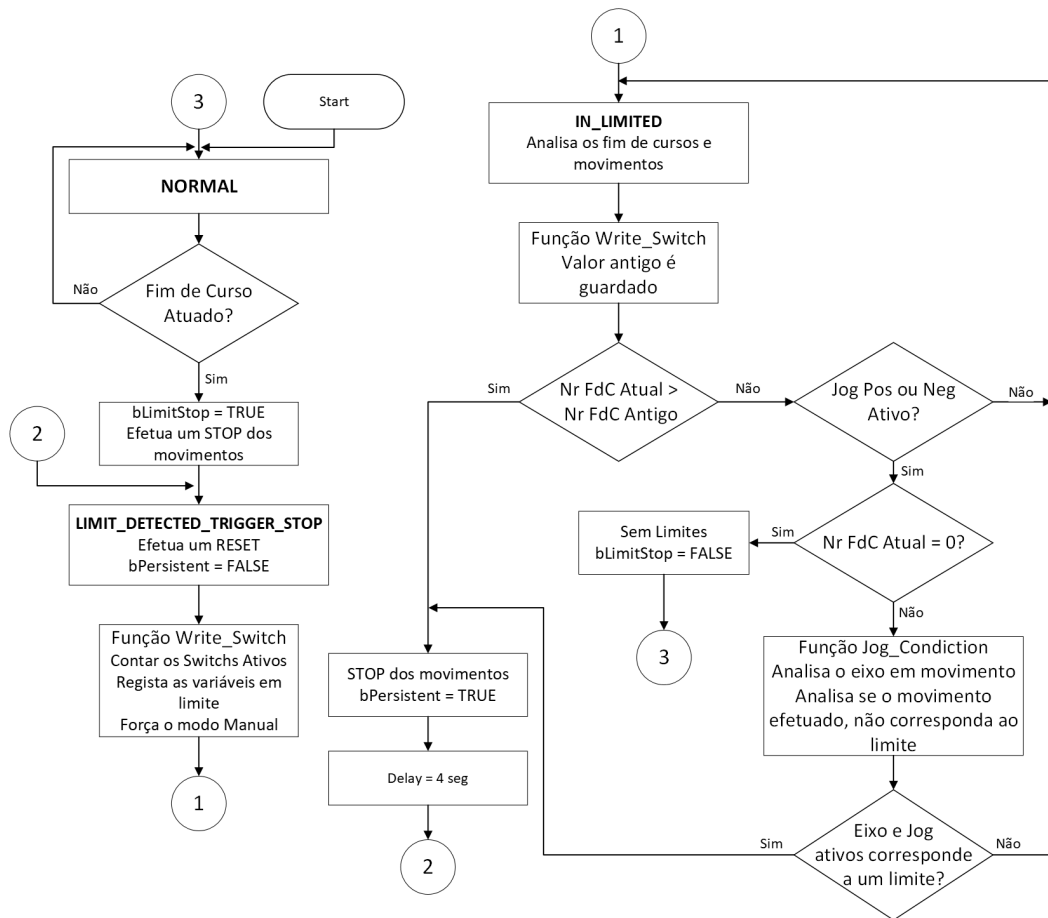


Figura 5.21: Fluxograma da implementação do controlo dos fim de cursos

fornecido pelo "Write_Switch" é igual a '0'; se for o caso, passa-se para o estado "NORMAL". Caso contrário, transita-se para a função "Jog_Condiction".

A função "Jog_Condiction" opera com base nas variáveis de *jog*. Ela associa qual é o eixo que o utilizador selecionou e verifica qual o sentido em que o eixo está a mover-se. Se o eixo se mover na direção não pretendida, ou seja, em direção ao limite ativo, o movimento é cortado imediatamente.

Como exemplo, suponha-se que o limite inferior do eixo X foi acionado durante um movimento. Ao efetuar o *jog* no eixo X na direção negativa, o código irá interromper o movimento, uma vez que o movimento desejado é na direção positiva. Após um período de tempo, é possível realizar um novo movimento; ao mover para a direção positiva, o sensor será desligado, liberando assim o eixo.

A declaração de variáveis da função principal encontra-se no Algoritmo C.11, o código da função principal está no Algoritmo C.12, a função "Write_Switch" é apresentada no Algoritmo C.13, enquanto a função "Jog_Condiction" se encontra no Algoritmo C.14.

Controlo do PLC

Para o Controlo do PLC, o objetivo era monitorar o estado da máquina em relação aos estados atuais das entradas ou ao estado atual da máquina. Para isso, foi elaborada uma máquina de estados com quatro estados: "EMERGENCY", "OFF_WAIT_ENABLE", "ON_TRIGGER_MOTOR" e "ON_MOTOR". Na Figura 5.22 está representada uma máquina de estados relativamente a este controlo.

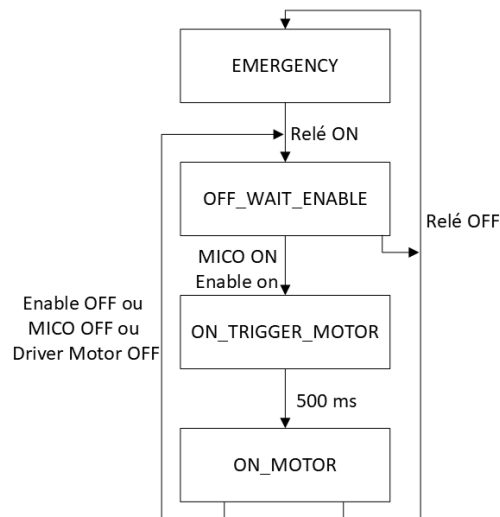


Figura 5.22: Máquina de estados implementada do controlo do PLC

No primeiro estado, que é acionado quando o botão de emergência é acionado, independentemente do estado em que a máquina se encontre, todos os estados são desligados, incluindo o *enable* e os motores. Além disso, a luz vermelha fixa na coluna luminosa de sinalização é ativada. A transição para o próximo estado ocorre apenas quando for ligado o relé.

O segundo estado é um estado de espera, que permite aguardar que todos os componentes externos, como os MICO, estejam prontos e possibilita a ativação do *enable*. Quando o *enable* do HMI é selecionado, a máquina passa para o próximo estado.

O terceiro estado é um estado de *trigger*, que permite que os motores sejam acionados e tenham tempo para serem ativados. Estes dois últimos estados estão registados como amarelo fixo na coluna luminosa de sinalização.

O quarto estado verifica se todos os motores, MICO e relé estão ativos. Se algum destes componentes for desativado, com exceção do relé, que provoca a transição para o estado de emergência, o estado da máquina será enviado para o estado "OFF_WAIT_ENABLE", assim como quando o *enable* é desligado. Este estado não contém nenhuma luz associada, pois depende do modo atual em que a máquina se encontra, seja no automático, manual, entre outros.

A declaração de variáveis para esta função encontra-se no Algoritmo C.15, enquanto o código da função é apresentado no Algoritmo C.16.

Obtenção de Dados do Interpretador

Durante o estudo, verificou-se que muitas variáveis existentes estavam bastante escondidas em estruturas ou códigos fechados, ou não apareciam no lado do PLC, mas eram visualizadas no lado do interpretador, conforme ilustrado na Figura 5.23.

Description	Port	IndexGroup	IndexOffset	Hex	Type	Size	Value	Unit
ac_1_type_r	553	20201	1		UNS16	2	1	
ac_1_active_position_acs_r	553	20201	2		SGN32	4	0,0000	mm or °
ac_1_current_position_acs_r	553	20201	3		SGN32	4	0,0000	mm or °
ac_1_end_position_acs_r	553	20201	4		SGN32	4	0,0000	mm or °
ac_1_active_feedrate_r	553	20201	5		REAL64	8	0	mm/min
ac_1_current_feedrate_r	553	20201	6		REAL64	8	0	mm/min
ac_1_direction_r	553	20201	7		SGN08	1	0	
ac_1_mode_r	553	20201	8		UNS32	4	0	
ac_1_axis_name_r	553	20201	9		STRING	16	Axis X	
ac_1_in_position_r	553	20201	A		BOOL	1	1	
ac_1_homing_done_r	553	20201	B		BOOL	1	0	
ac_1_state_r	553	20201	C		UNS16	2	5	
ac_1_cmd_feedrate_override_w	553	20201	D		UNS16	2		
ac_1_feedhold_w	553	20201	E		BOOL	1		
ac_1_channel_nr_r	553	20201	F		UNS16	2	1	
ac_1_position_leg_r	553	20201	10		SGN32	4	0,0000	mm or °
ac_1_jpo_input_disable_w	553	20201	11		BOOL	1		
ac_1_drive_type_r	553	20201	12		UNS16	2	6	
ac_1_active_m_functions_r	553	20201	13		NONE	52		
ac_1_cmd_override_valid_w	553	20201	14		BOOL	1		
ac_1_disable_pos_leg_w	553	20201	15		BOOL	1		
ac_1_disable_pos_leg_r	553	20201	16		BOOL	1	1	
ac_1_active_h_functions_r	553	20201	17		NONE	52		
ac_1_interpr_achs_mds_file_w	553	20201	18		STRING	256		
ac_1_interpr_achs_kw_file_w	553	20201	19		STRING	256		
ac_1_customer_val_r	553	20201	1A		NONE	4		
ac_1_active_m_funct_count_r	553	20201	1B		UNS32	4	0	
ac_1_active_m_funct_number_r	553	20201	1C		NONE	48		

Figura 5.23: Algumas variáveis existentes no lado do interpretador

Contudo, existe uma função contida na biblioteca TC2_System chamada ADS-READ. Esta função efetua um pedido de leitura de dados ADS para solicitar dados a um determinado endereço ADS. Para obter estes dados, apenas se necessita da porta, *IndexGroup*, *IdxOffset*, tamanho e endereço de destino. Estes dados são todos visualizados nos respetivos dados do lado do interpretador.

Esta função foi utilizada para a extração de algumas informações, como os valores obtidos da posição atual ou a velocidade de saída. A declaração de variáveis da função encontra-se no Algoritmo C.17, enquanto o código da função está no Algoritmo C.18.

Adaptação de Funções Base

Muitas das funções utilizadas pelo código base estão em bibliotecas fechadas, com maior foco nas bibliotecas TC2_CncHli e TC3_CNC, que não permitem aos utilizadores alterar as respetivas funções. No código fonte, existem algumas partes que possibilitam alterações, como já foi visto na configuração do *homing*.

Contudo, surgiu um problema relacionado ao *homing*, uma vez que existe um erro no código que é chamado para efetuar o *homing*, encontrado na função "FB_HLI_

CNCChannel”, que está fechada para alterações. Para corrigir este erro, foi necessário copiar a função contida na biblioteca e criar uma nova função, mantendo o mesmo conteúdo da função base.

Verificou-se que é neste bloco de código que se encontra a lógica que permite a transição entre os vários modos, como automático ou manual, possibilitando a introdução de lógica adicional.

Assim, neste novo código foram introduzidas as seguintes funcionalidades:

- declaração de variáveis para *motion*, como a ativação de emergência, paragem e *reset*;
- chamada da função dos limites e controlo dos *stop* e *reset*;
- controlo das entradas dos modos, verificando se o *enable* está ativo ou se o *homing* foi efetuado (estes últimos para os modos automático e MDI);
- forçar modos, quando as variáveis não permitem os modos que o utilizador pretende, para o *standby* ou manual, sendo este último ativado quando os limites estão a atuar;
- alertas informativos, como o *enable* não selecionado ou a falta de execução do *homing*;
- permissão para efetuar *homing*, mesmo que já tenha sido realizado anteriormente;
- alteração da função "HLI3_ChannelModeReference" para "HLI3_Channel-ModeAuto" e os seus ajustes, devido a um erro na função *Reference*.

As variáveis da função principal encontram-se no Algoritmo C.20, enquanto a função é apresentada no Algoritmo C.21, e uma outra função que transfere algumas variáveis para variáveis globais, designada de "Copy_Parameter", visualizada no Algoritmo C.22.

5.5 *Software CAD/CAM - Inventor*

O *software* CAD/ CAM utilizado para o estudo desta implementação foi o Inventor, conforme já explicado na Secção 4. Este *software* permite desenhar uma peça em CAD, e a partir dessa peça, e com a extensão CAM, configurar todos os passos necessários para que máquinas fresadoras, tornos ou impressoras *Three Dimensional* (3D) possam efetuar a peça, realizando uma simulação.

Consoante a peça existente, é possível realizar furos, faces e contornos, tanto em movimentos bidimensionais como tridimensionais, entre outros movimentos, que estão apresentados na barra de pesquisa, como ilustrado na Figura 5.24. É importante

referir que todos os movimentos podem ser visualizados e que informações adicionais podem ser obtidas ao passar o rato sobre o movimento ou consultando o suporte da Autodesk [73].

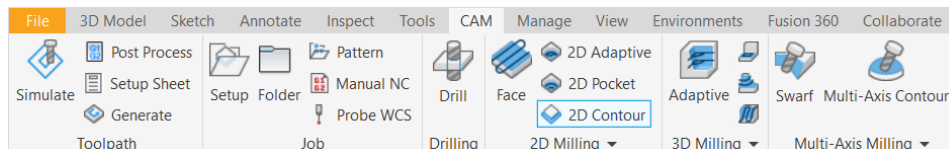


Figura 5.24: Alguns movimentos existentes da barra CAM [73]

Após selecionar o movimento, é necessário configurar vários parâmetros, como a ferramenta, velocidade de corte, velocidade do *spindle*, geometria a ser aplicada, os níveis de camadas (aproximação, corte e última camada), os passos, entre outras configurações.

Uma das vantagens de utilizar estes *softwares* é a visualização da peça a ser trabalhada, através da ferramenta "Simulate". Com esta ferramenta, é possível observar todos os movimentos que são efetuados, permitindo verificar se há alguma falha, como demonstrado na Figura 5.25.

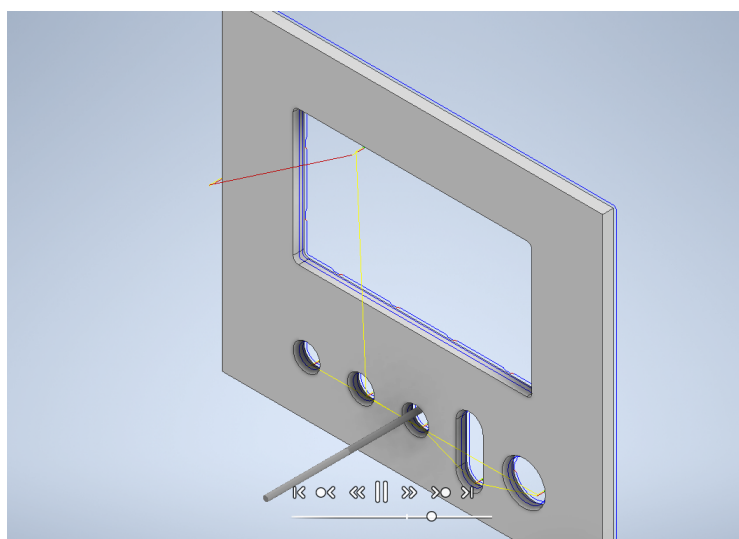


Figura 5.25: Simulação do código fornecido por um sistema CAM

Após a terminação do processo, é necessário efetuar o "Post Processor" e criar um ficheiro .nc. Contudo, dependendo da marca das CNC, existem diferenças no código G, e o TwinCAT não consta na lista do Inventor. A recomendação é realizar uma verificação do código antes de ser utilizado, utilizando o *GCodeViewer* e realizando um teste sem ferramenta, permitindo retirar partes do código e adaptá-lo, como a introdução do *zero offset*. O TwinCAT apresenta algumas diferenças no código G em relação aos protocolos DIN e ISO, que se encontram detalhadas no Anexo D.

5.6 Conclusão da Implementação Prática

Com base na arquitetura definida e nos conteúdos anteriores, este capítulo descreveu a implementação prática do sistema, evidenciando como o trabalho evoluiu da concepção à realização.

Numa fase inicial, foram especificadas as características necessárias para a montagem do *hardware*, incluindo a explicação das principais ligações e a utilização do *software* FCT para o controlo dos *drivers*, destacando-se as funcionalidades relevantes tanto ao nível do *hardware* como do *software*.

De seguida, foi abordada a utilização de um código PTP, que constituiu uma primeira aproximação ao ambiente TwinCAT. Neste contexto, foram exploradas as configurações de entradas e saídas, bem como parâmetros essenciais para a realização de testes iniciais, permitindo compreender algumas das funções associadas à movimentação dos eixos.

Posteriormente, analisou-se a implementação do programa principal em CNC, no qual foram exploradas as configurações importantes ao funcionamento global do sistema, nomeadamente o controlo dos eixos, os alarmes efetuados, o *spindle* e o estado da HMI. Foram ainda documentadas as modificações introduzidas no código-base e sistematizadas as funções desenvolvidas, de modo a garantir a obtenção do resultado final esperado.

Por fim, foi efetuada uma breve exploração da ferramenta CAD/CAM escolhida, destacando as suas funcionalidades, vantagens e limitações na aplicação em estudo.

Em síntese, este capítulo consolidou a implementação da tecnologia proposta, detalhando as etapas práticas necessárias ao desenvolvimento do projeto e preparando a base para a fase de testes e respetivos resultados.

Capítulo 6

Testes e Resultados

Neste capítulo serão abordados todos os testes realizados e os respetivos resultados obtidos ao longo do desenvolvimento desta Tese de Mestrado.

Numa primeira fase, serão discutidos os testes em ambiente simulado, bem como as primeiras análises do código-fonte, identificando os principais benefícios e falhas existentes.

Numa segunda fase, serão explorados os testes e resultados obtidos nos motores, tanto de forma separada como montados na mesa, com o objetivo de compreender o seu comportamento e configuração. Nesta fase, incluem-se os testes realizados com o código PTP, conforme mencionado na Secção 5.2.

Na terceira fase, serão analisados os testes realizados com o código CNC e os respetivos resultados obtidos. Os testes são efetuados ao longo das melhorias do código, permitindo avaliar a sua evolução.

Na fase final, serão discutidos os testes ainda por realizar, abordando brevemente a metodologia a ser utilizada e os objetivos a serem alcançados.

6.1 Teste em Ambiente Simulado

Os testes realizados em ambiente simulado tiveram como objetivo testar os códigos PTP e CNC, de forma a efetuar a eventual deteção de falhas e a compreender o funcionamento dos códigos, sem depender de *hardware*.

No caso do código PTP, os testes foram realizados de forma rápida, permitindo observar o movimento dos eixos, de forma simulada, permitindo confirmar a segurança da movimentação dos eixos.

Relativamente ao código CNC, o processo iniciou-se com um estudo detalhado do código, seguido pela elaboração de um código G, com o intuito de analisar o comportamento do código em modo automático. O resultado deste teste pode ser visualizado na Figura 6.1.

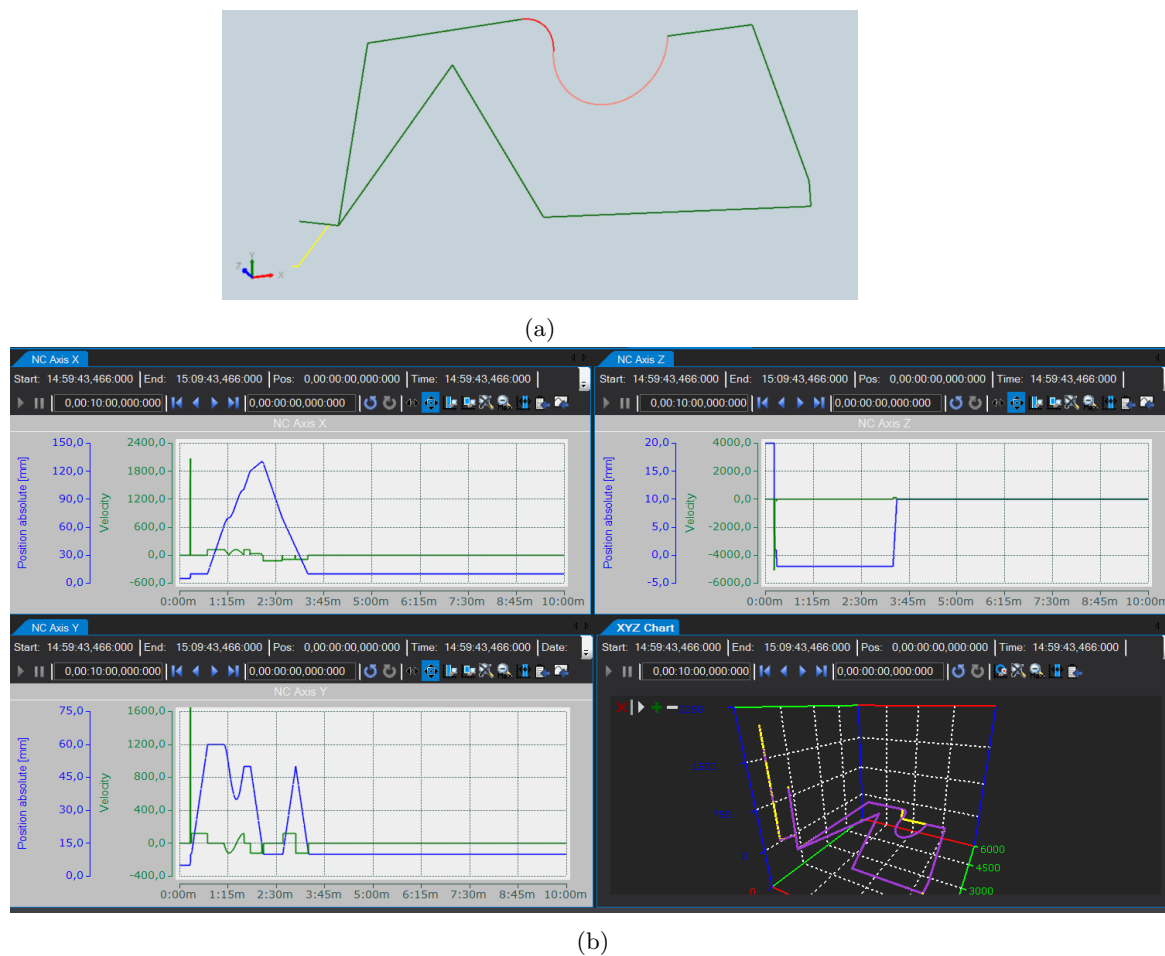


Figura 6.1: Resultado do código G efetuado para o teste do código em ambiente simulado: a) Vista do GCodeViewer b) Valores obtidos no *scope*

Com o código G a funcionar, foi possível analisar a forma de operação, a trajetória realizada e as melhorias a serem implementadas no código, como a introdução de segurança com as paragens de emergência e STOP, ou correção de erros existentes no código como o *homing*.

Num processo mais avançado, o desenvolvimento na máquina foi realizado em simulação, de modo a salvaguardar o *hardware*. Este procedimento permitiu o desenvolvimento da HMI, bem como a criação de códigos e ajustes às configurações,

visando alcançar um projeto conforme o esperado e possibilitar a realização dos testes na máquina com maior segurança. A Figura 6.2 representa um dos resultados obtidos durante o desenvolvimento da HMI, contribuindo para a melhoria do funcionamento da máquina. Como os testes durante o desenvolvimento envolvem o *hardware*, serão apresentados nas seguintes secções.

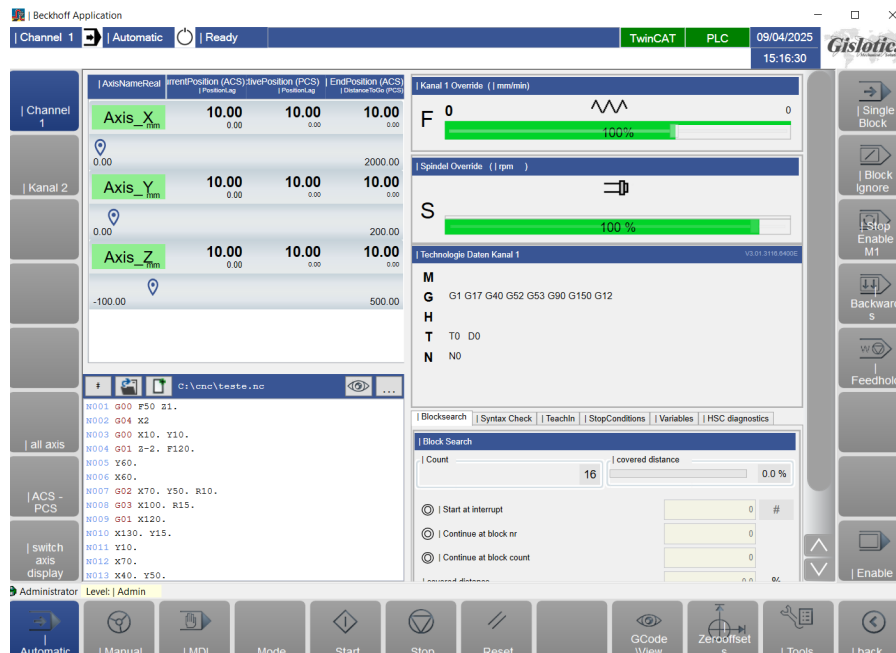


Figura 6.2: Consola HMI em fase de desenvolvimento

6.2 Testes e Resultados dos Motores de Eixo e *Drivers*

Os motores do eixo são a principal forma de movimentação da ferramenta, sendo que cada um deve atuar conforme o desejado. Os testes aos motores foram efetuados em dois momentos: um com o motor separado da mesa e outro acoplado à mesa.

6.2.1 Motor Separado da Mesa

Os testes realizados com o motor separado da mesa pretenderam verificar os seguintes pontos:

- ligação do *hardware* conforme o previsto, assim como o seu funcionamento;
- observação de parâmetros, tanto no FCT como no TwinCAT, em código PTP, e alteração de configurações, conforme exposto no Capítulo 5;
- primeiro impacto sobre o controlo dos motores.

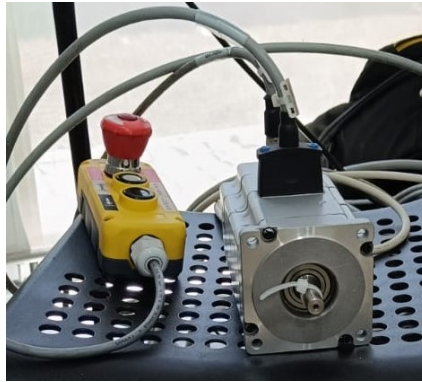


Figura 6.3: Motor separado da mesa

Para realizar todos os pontos mencionados, foi montado um motor ao lado do quadro, conforme visualizado na Figura 6.3.

Foram realizado dois testes. O primeiro teste foi realizado somente com o *software* da Festo, enquanto o segundo foi efetuado com um programa PTP simples do TwinCAT, utilizando o FCT somente para análise de movimento.

Ambiente *Festo Configuration Tool* (FCT)

O ambiente proporcionado pelo FCT permite obter parâmetros em tempo real e visualizar o estado das entradas e saídas, conforme demonstrado na Figura 6.4. Além disso, possibilita o controle do motor via *jog*.

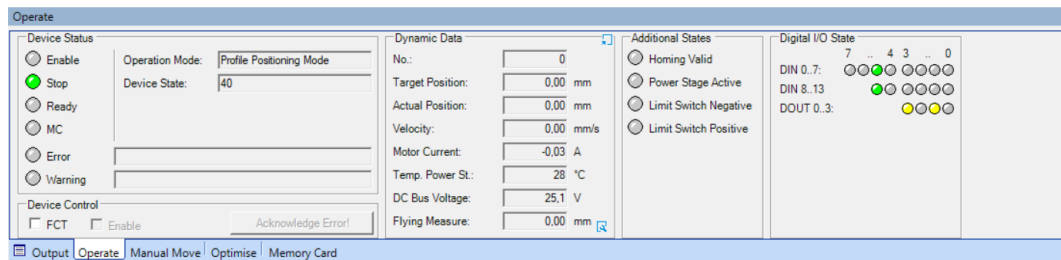


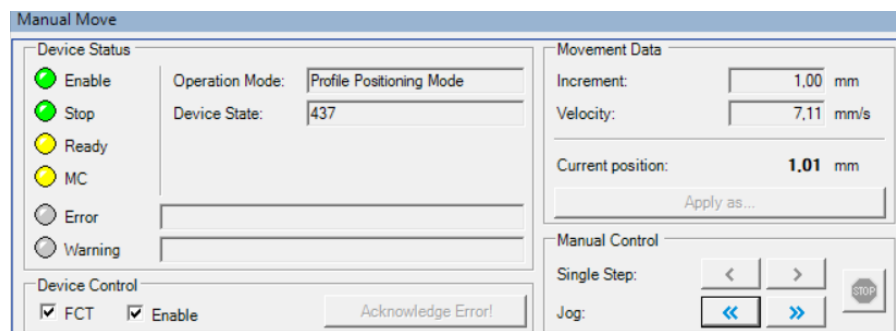
Figura 6.4: Barra de operação do FCT

Para controlar o motor via FCT, é necessário que na entrada X1 dos *drivers* do motor, os pinos 4, 5 e 13 estejam ativos. Basta clicar nos quadrados existentes no *Device Control* para habilitar o controle do motor, permitindo movimentar os motores pelas setas disponíveis na aba "Manual Move".

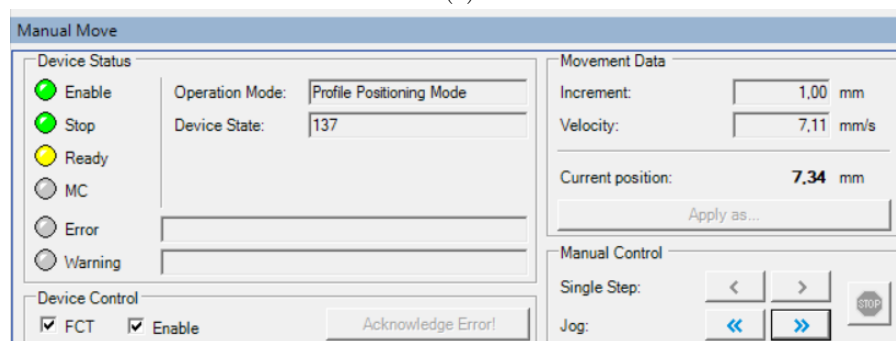
Este teste é considerado concluído quando o eixo do motor roda fisicamente e o FCT registrar o respetivo movimento, como se confirma na Figura 6.5.

Controlo pelo TwinCAT

Concluído o teste efetuado pelo FCT, foi a vez de testar o código PTP, com o objetivo de avaliar a comunicação entre o controlador, com foco principal no terminal EL2522,



(a)



(b)

Figura 6.5: Resultado do movimento via FCT: a) Antes do movimento b) Após um movimento

e o *driver*, verificando a viabilidade da arquitetura.

Após a realização de alguns testes, obteve-se movimento dos motores, confirmando assim que a arquitetura é viável, com o motor a realizar um movimento.

Como mencionado no Capítulo 5, foi necessária uma calibração para garantir que os valores obtidos no *driver* fossem equivalentes.

A calibração com o uso do código PTP foi efetuada da seguinte forma: foram realizados vinte movimentos até 100 mm, onde os dez primeiros foram efetuados a uma velocidade de 5 mm/s e os últimos dez a uma velocidade de 20 mm/s. Este procedimento permitiu analisar a repetibilidade, calcular o erro e realizar a respetiva correção. Após a correção, os testes foram repetidos para verificar se a calibração foi realizada corretamente. Os dados obtidos estão registados na Tabela 6.1.

Ao analisar os valores obtidos, e com o uso da Equação 5.2, verificou-se que o valor de correção é 1,0234, concluindo-se que, com o motor desacoplado, os valores obtidos indicam uma repetibilidade excelente.

6.2.2 Motor Acoplado à Mesa

Após a conclusão do último teste, o motor de teste foi acoplado à mesa para realizar os próximos testes. É importante notar que os eixos nunca haviam sido testados, sendo necessário testar o comportamento dos eixos, como a existência de erro de

Tabela 6.1: Dados obtidos da calibração com motor desacoplado

Teste Calibração Motor Separado Distância = 100 mm			
Sem Calibração		Com Calibração	
5 mm/s	20 mm/s	5 mm/s	20 mm/s
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95
97,66	97,66	99,95	99,95

seguimento do eixo entre o *driver* e o motor ou se era detetado alguma falha mecânica.

Foram realizados dois testes: um para efetuar a calibração em PTP e outro para avaliar o comportamento dos eixos a diferentes velocidades, designado como teste de consistência.

Calibração em PTP

A calibração é uma etapa importante, conforme comprovado quando o motor estava desacoplado. Para realizar o teste de calibração dos eixos, foram efetuados dois movimentos para cada eixo, quando os eixos não estavam calibrados, com uma distância a percorrer de 1000 mm para o eixo Y e de 500 mm para o eixo X. Após a obtenção dos valores de calibração, foram realizadas vinte repetições de movimento, dez a 5 mm/s e outras dez a 25 mm/s, para comprovar se a calibração foi efetuada com sucesso e se não houve falhas.

Os resultados obtidos estão apresentados na Tabela 6.2. É importante mencionar que estes valores foram medidos com o uso de fita métrica e lápis, o que pode introduzir algum erro nas medições.

Ao aplicar a Equação 5.2, obteve-se que, no eixo X, o denominador para a calibração seria de 0,792, enquanto no eixo Y o denominador seria igual a 0,789.

Teste de Consistência - Curta Distância

Após a calibração dos dois eixos, foram realizados os testes de consistência. O objetivo destes testes era compreender o comportamento dos eixos face a diferentes velocidades, analisar o erro de posição fornecido pelos *drivers* e verificar a existência de ressonância mecânica ao movimentar os eixos.

Tabela 6.2: Dados obtidos da calibração com motor acoplado

Teste Calibração Motor Acoplado			
Eixo X D = 500mm		Eixo Y D = 1000mm	
5 mm/s	25 mm/s	5 mm/s	25 mm/s
Não Calibrado			
633	631	1266	–
633	630	1271	–
Calibrado			
1000	1000	500	501
1000	1000	500	500
1000	1000	500	499
1000	1000	500	499
1000	1000	499	499
1000	1000	499	499
1000	1000	498	498
1000	1000	498	499
1000	1000	497	498
1000	1000	497	499

Para uma análise mais detalhada, foram efetuados dois testes: um de curta distância e outro de longa distância, sendo este último abordado na próxima subsecção.

Relativamente ao teste de curta distância, este foi realizado cinco vezes para o eixo Y e quatro vezes para o eixo X, com cada teste percorrendo uma distância de 400 mm, incluindo duas idas e voltas. O local do teste variou a cada execução, permitindo a análise de possíveis falhas mecânicas ao longo do eixo. Os locais do teste estão descritos na Tabela 6.3.

Tabela 6.3: Distâncias testadas para o teste de consistência em distância curta

Teste Consistência - Distância Curta em Teste		
Número do Teste	Eixo X (mm)	Eixo Y (mm)
1	0 - 400	0 - 400
2	400 - 800	400 - 800
3	600 - 1000	800 - 1200
4	200 - 600	1200 - 1600
5		1600 - 2000

As velocidades analisadas foram variadas para permitir a observação do comportamento dos eixos a diferentes velocidades. Foram testadas as seguintes velocidades: 50, 100, 150, 170, 200, 230, 240, 250 e 300 mm/s. Graficamente, serão apresentados os resultados para as velocidades de 50, 150, 200 e 250 mm/s, enquanto para as restantes velocidades mostrarão apenas os valores obtidos.

A obtenção de dados foi realizada através do FCT, e foi utilizado um programa em Python para calcular a velocidade média nos sentidos positivo e negativo, bem como os picos máximos, tanto no sentido positivo como no negativo.

Os gráficos das velocidades para o eixo X encontram-se na Figura 6.6, enquanto os gráficos referentes ao eixo Y estão na Figura 6.7. Na Tabela 6.4 são apresentados todos os resultados das velocidades médias em cada teste, juntamente com os valores de pico dos erros detetados.

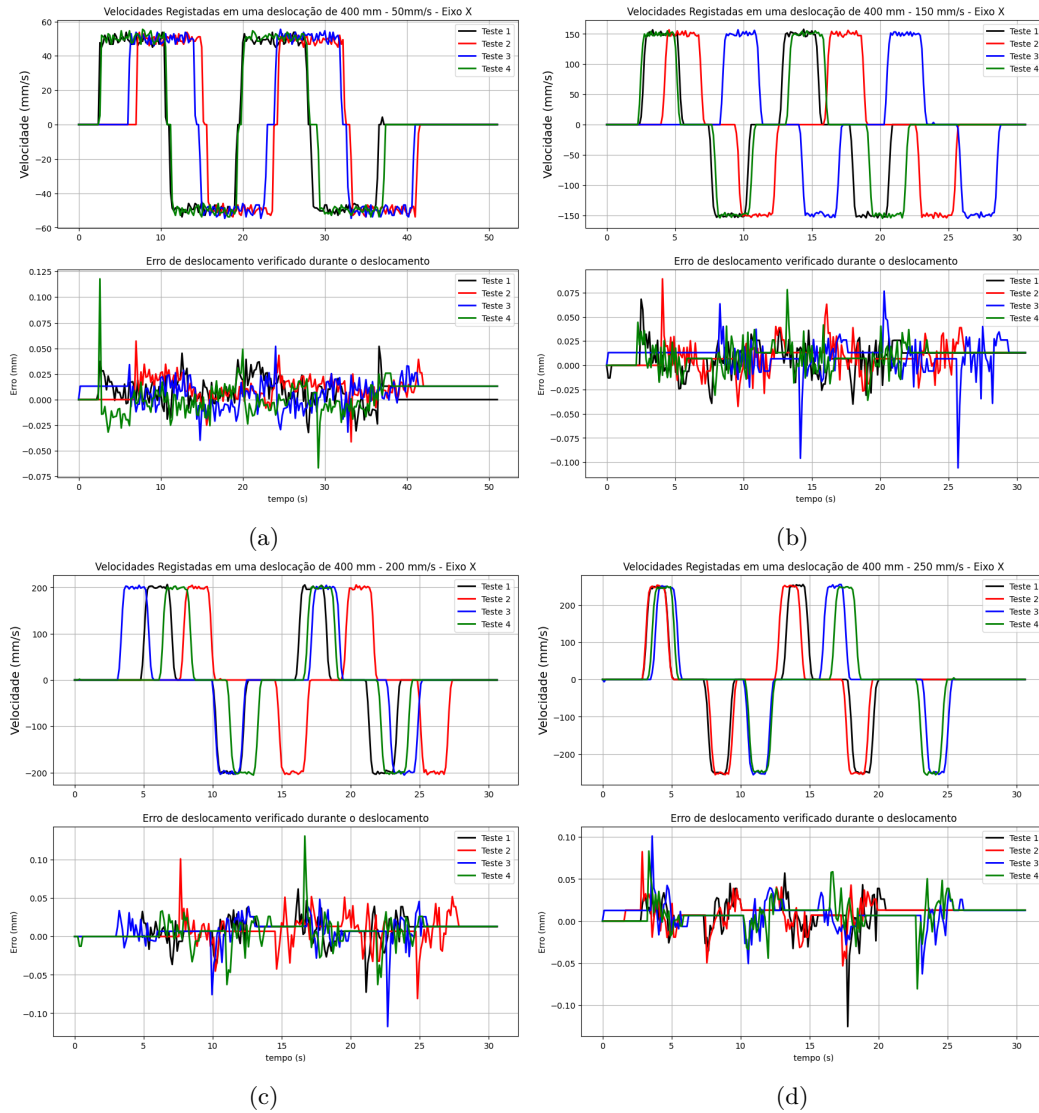


Figura 6.6: Teste de consistência do eixo X - curta distância: a) 50 mm/s b) 150 mm/s c) 200 mm/s d) 250 mm/s

A análise dos resultados obtidos neste teste revela que o eixo X apresenta uma consistência de valores, com variações pequenas na velocidade durante o movimento e picos de erros de seguimento que ocorrem geralmente no início do movimento, o que é considerado normal. Por outro lado, no eixo Y foram observadas falhas

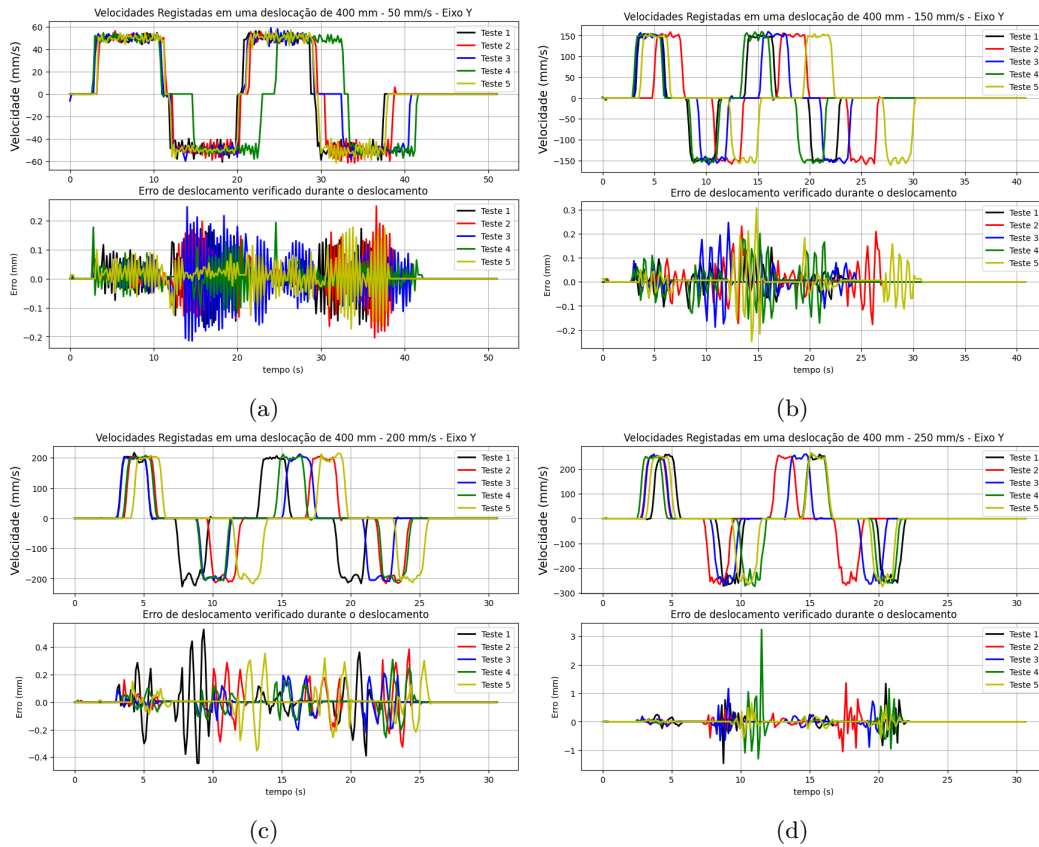


Figura 6.7: Teste de consistência do eixo Y - curta distância: a) 50 mm/s b) 150 mm/s c) 200 mm/s d) 250 mm/s

que podem ser atribuídas a diversos fatores, como elasticidade do eixo, montagem inadequada ou até falhas no *driver*, resultando em instabilidade, uma vez que os valores de erro são significativamente maiores em comparação com o eixo X. Os picos de erro de seguimento do eixo Y ocorrem em momentos finais ou a meio do movimento, indicando uma maior instabilidade na velocidade, que será analisada no teste seguinte.

Teste de Consistência - Longa Distância

Para complementar o teste de consistência de curta distância e obter mais dados para uma conclusão final, foi realizado um segundo teste de consistência, onde o eixo percorreu a máxima distância permitida.

O teste de longa distância consistiu em cinco testes iguais para cada velocidade e para cada eixo, em que cada teste o eixo realizava uma viagem de ida e volta até a distância máxima, ou seja, para o eixo X percorreu-se a distância de 1000 mm e para o eixo Y 2000 mm. As velocidades utilizadas foram as mesmas dos testes de curta distância.

Tabela 6.4: Dados obtidos do teste de consistência - percurso curto

Teste Consistência - Distância Curta - Resultados Finais					
	Vel. Usada (mm/s)	Vel. Média Dir. Pos (mm/s)	Vel. Média Dir. Neg (mm/s)	Erro Máximo Dir. Pos. (mm)	Erro Máximo Dir. Neg. (mm)
Eixo X	50	49,7846	49,5073	0,117828	-0,0666504
	100	98,5336	98,007	0,0841064	-0,15036
	150	148,981	148,943	0,0892639	-0,105927
	170	167,899	167,583	0,133301	-0,0793457
	200	195,537	196,849	0,13092	-0,117432
	230	228,031	228,632	0,120605	-0,0658569
	240	238,127	237,427	0,156708	-0,0888672
	250	243,503	245,185	0,101166	-0,125366
	300	283,535	283,274	0,0825195	-0,141235
Eixo Y	50	49,8572	49,9723	0,250732	-0,215027
	100	98,397	98,5357	0,394745	-0,312225
	150	148,99	148,59	0,308258	-0,247559
	170	167,154	168,757	0,458221	-0,467346
	200	196,608	196,42	0,528839	-0,444733
	230	227,837	229,127	0,630402	-0,499481
	240	237,686	238,922	1,2251	-0,975555
	250	245,263	244,857	3,24246	-1,45956
	300	281,667	257,485	12,9099	-12,8282

Os resultados obtidos do teste de longa distância estão apresentados na Tabela 6.5, enquanto os gráficos relativos ao comportamento do eixo X são mostrados na Figura 6.8 e os relativos ao eixo Y na Figura 6.9.

A análise dos resultados obtidos neste teste revela que o eixo X apresenta pouca variação ao atingir a velocidade pretendida, com um erro a rondar os 10 mm/s do valor indicado, e o erro de seguimento máximo a rondar 0,1 mm, ocorrendo principalmente no início do movimento, conforme observado no teste de curta distância. Por outro lado, no eixo Y, foi identificado um erro de ressonância que se manifesta no final de qualquer movimento, sendo mais pronunciado em movimentos no sentido negativo, possivelmente devido a alguma folga na correia. Em velocidades mais altas, é notável uma sequência de picos de erros de seguimento crescentes, conforme evidenciado nos gráficos da Figura 6.9, resultando em intervalos de velocidade irregulares. Na velocidade de 300 mm/s, o movimento no sentido negativo gerou um erro do *driver* do motor em vários testes, travando o motor, impossibilitando a utilização dessas velocidades.

Além dos dados mencionados, foi verificado que, em todos os testes realizados, o ponto de paragem coincidiu com o ponto inicial, indicando uma boa precisão na execução dos movimentos.

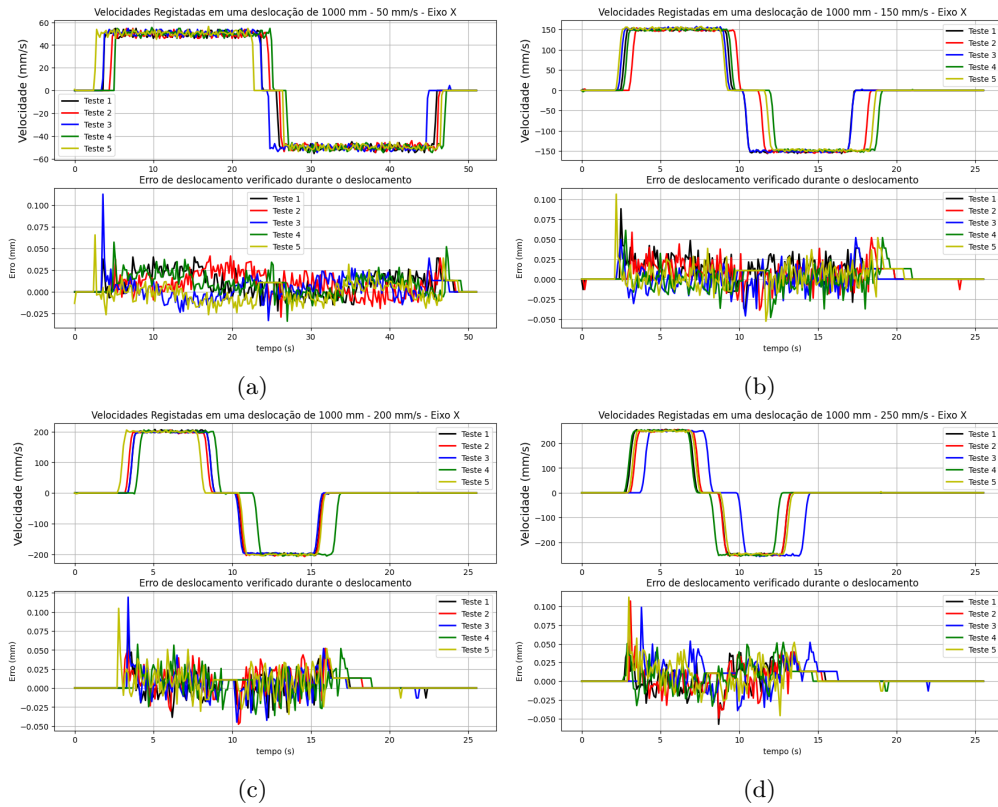


Figura 6.8: Teste de consistência do eixo X - longa distância: a) 50 mm/s b) 150 mm/s c) 200 mm/s d) 250 mm/s

6.3 Testes e Resultados usando o Código da CNC

Após a realização dos testes do *hardware* em PTP, de forma a garantir que os motores operam conforme o esperado, foram realizados testes com o código CNC.

Para isso, foram efetuados vários testes para assegurar que a solução final estava conforme o funcionamento na CNC. Os testes efetuados foram: calibração, ganhos, teste básico do processo automático, *homing*, interrupção de emergência, teste dos limites físicos e teste de códigos G complexos.

6.3.1 Calibração em CNC

Antes de se iniciar os restantes testes, foi necessário efetuar uma calibração, segundo as equações mencionadas na Subsubsecção 5.4.1.

Para esta calibração, foi executada uma única viagem, registrando o valor obtido pela distância, segundo o *driver*, onde no eixo Y foi realizada uma deslocação de 1000 mm e no eixo X outra deslocação de 500 mm. Após obter a distância percorrida, foi calculado o valor final entre o numerador e denominador ideal representado de x , onde no eixo X está representado na Equação 6.1 e no eixo Y na Equação 6.2, permitindo assim a manipulação dos numeradores e denominadores, uma vez que só

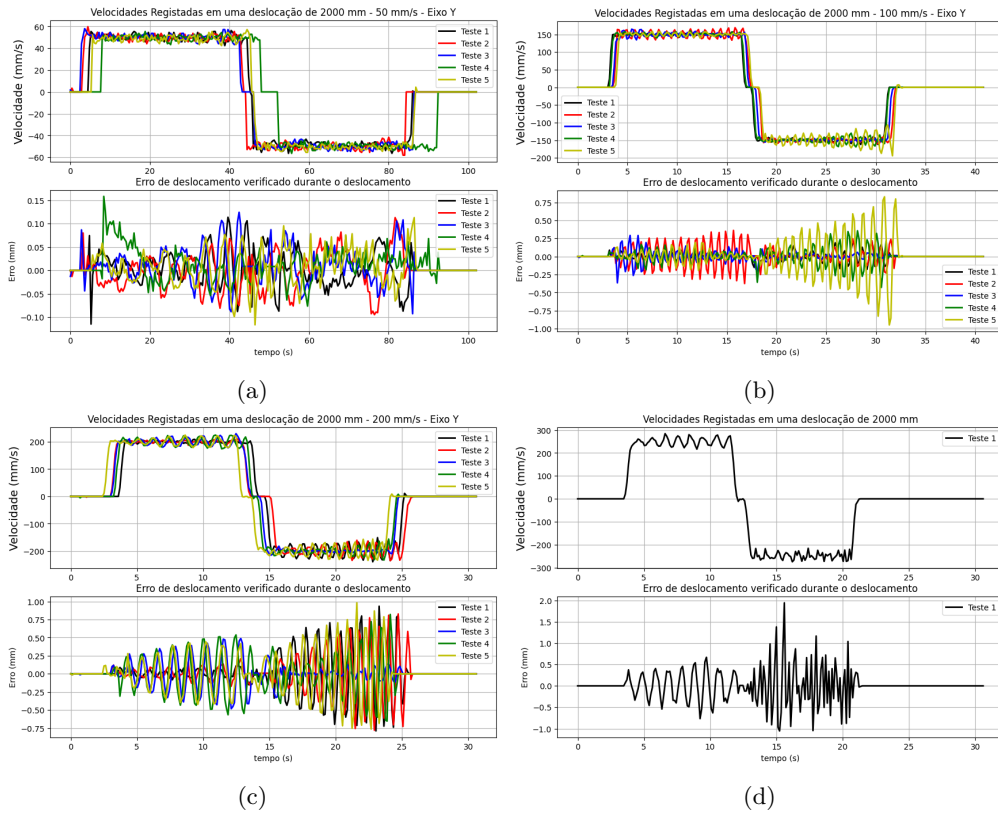


Figura 6.9: Teste de consistência do eixo Y - longa distância: a) 50 mm/s b) 150 mm/s c) 200 mm/s d) 250 mm/s

permite a introdução de números com quatro casas decimais, como está representado na Tabela 6.6 os respectivos valores, de forma a encontrar o melhor denominador e numerador. Na tabela contém ainda os resultados de uma estimativa da distância $d_{estimada_tentativa}$ de todas as variantes de numeradores e denominadores, usando a Equação 6.3, como a distância obtida ao usar o numerador e denominador em destaque.

$$y_X = \frac{1}{x_X} = \frac{1000 \cdot \frac{1}{0,0064}}{991,82} = 157,5387(0,1\mu m/inc)$$

$$x_X = \frac{N}{D} = 0,00634763(inc/0,1\mu m) \quad (6.1)$$

$$y_Y = \frac{1}{x_Y} = \frac{500 \cdot \frac{1}{0,0063}}{503,78} = 157,53916(0,1\mu m/inc)$$

$$x_Y = \frac{N}{D} = 0,00634765(inc/0,1\mu m) \quad (6.2)$$

$$d_{estimada} = \frac{d_{pretendida} \cdot x_{introduzido}}{x_{calculado}} \quad (6.3)$$

Ao analisar os dados obtidos, é possível comprovar que a calibração foi efetuada

Tabela 6.5: Dados obtidos do teste de consistência - percurso longo

Teste de consistência - distância longa - resultados finais					
	Vel. Usada (mm/s)	Vel. Média Dir. Pos (mm/s)	Vel. Média Dir. Neg (mm/s)	Erro Máximo Dir. Pos. (mm)	Erro Máximo Dir. Neg. (mm)
Eixo X - (1000 mm)	50	50,0396	50,0277	0,112274	-0,0337219
	100	99,6618	99,2498	0,0694275	-0,0539551
	150	150,213	149,187	0,106323	-0,0527649
	170	168,997	169,001	0,107117	-0,0539551
	200	198,523	198,614	0,119415	-0,0476074
	230	228,758	229,062	0,111877	-0,0551453
	240	238,898	239,012	0,105927	-0,055542
	250	248,624	248,133	0,112274	-0,0575256
	300	292,576	292,603	0,0737915	-0,0706177
Eixo Y - (2000 mm)	50	49,8472	49,825	0,158691	-0,116638
	100	99,6979	99,6355	0,652618	-0,700623
	150	149,678	149,397	0,827576	-0,947784
	170	168,991	168,841	0,455841	-0,443939
	200	199,503	199,251	0,981903	-0,78949
	230	229,319	229,59	0,649445	-0,529236
	240	239,864	239,435	0,708954	-0,612946
	250	248,867	249,519	1,94357	-1,0549
	300	296,353	250,539 (Erro)	12,3597	-9,99677

Tabela 6.6: Calibração dos eixos no programa em CNC

Calibração dos Eixos				
	N'_{eixo} (inc)	D'_{eixo} (0,1 μm)	$y'_{eixo}(inc/0,1\mu m)$	$d_{estimada_tentativa}$ (mm)
Eixo X	1	0,0063	158,7302	503,7801
	10	0,0635	157,4803	499,8131
	100	0,6348	157,5299	499,9705
	1000	6,3476	157,5399	500,0022
	10000	63,4763	157,5391	499,9997
	Pretendido		157,5392	500
	Distância Obtida (mm)			500,01
Eixo Y	1	0,0063	158,7302	1007,5632
	10	0,0635	157,4803	999,629
	100	0,6348	157,5299	999,94
	1000	6,3477	157,5374	999,9917
	10000	63,4765	157,5386	999,9994
	Pretendido		157,5387	1000
	Distância Obtida (mm)			1000

com sucesso, uma vez que a distância percorrida corresponde à distância pretendida. Este resultado indica que os parâmetros de calibração estão adequados para o funcionamento da CNC, permitindo uma operação precisa e eficiente.

6.3.2 Ganhos

Os ganhos dos eixos são fundamentais para ajustar os valores obtidos, aproximando-os dos valores pretendidos. Ganhos elevados podem resultar em *overshoot*, enquanto ganhos reduzidos podem levar a erros de seguimento mais significativos, o que pode comprometer o funcionamento da máquina.

Para determinar um ganho adequado, foi necessário analisar o comportamento dos dois eixos, assegurando que não ocorressem erros de seguimento. Os ganhos dos dois eixos devem ser iguais; caso contrário, ocorre um desvio entre a trajetória planeada e a efetuada, já que um dos eixos anda mais do que o outro devido ao ganho.

Para tal, foi realizado um teste em ambos os eixos, conforme ilustrado na Figura 6.10, que mostra a velocidade em função de vários ganhos (k) para o eixo Y, considerado o mais instável, a uma velocidade de 1000 mm/min. É importante notar que o teste foi efetuado antes da correção do tempo de ciclo, mencionada na Subsecção 5.4.4.

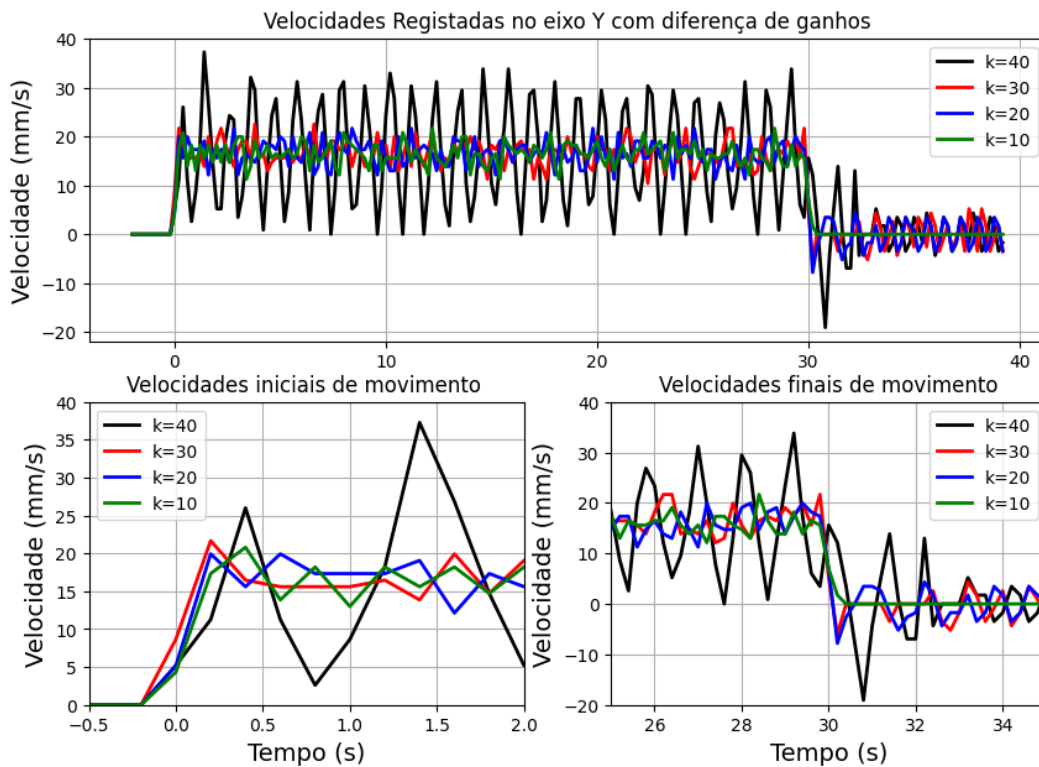


Figura 6.10: Teste com diferentes ganhos ao eixo Y

Como evidenciado no gráfico, observa-se uma instabilidade com o ganho igual a 40. Com ganhos de 30 e 20, notam-se pequenas instabilidades, sendo visíveis pequenos *overshoot* no final do movimento, resultando em pequenos movimentos indesejados quando o eixo já deveria estar parado. Com o ganho ajustado para 10,

o eixo comportou-se conforme o esperado, sem apresentar *overshoot* da velocidade e permanecendo parado no momento indicado.

Após a realização de vários testes, foi necessária a alteração deste ganho para responder aos resultados gerados. O valor final do ganho foi estabelecido em 50, uma vez que, após a correção do tempo de ciclo, ocorreram diversos erros de seguimento devido ao aumento da diferença entre o valor de referência e o valor atual.

6.3.3 Testes Básico do Processo Automático

Após a realização da calibração e dos ganhos, foi realizada a experimentação dos movimentos interpolados a partir de um código G.

O objetivo deste teste foi verificar o comportamento da máquina em relação a um programa simples, permitindo testar a trajetória ao desenhar num papel e registrar os dados do controlador, numa ferramenta chamada de *scope*. Este processo possibilitou o ajuste de parâmetros e a verificação da ocorrência de erros ao longo de dezenas de testes efetuados com o mesmo programa, assim como a identificação de melhorias a serem implementadas.

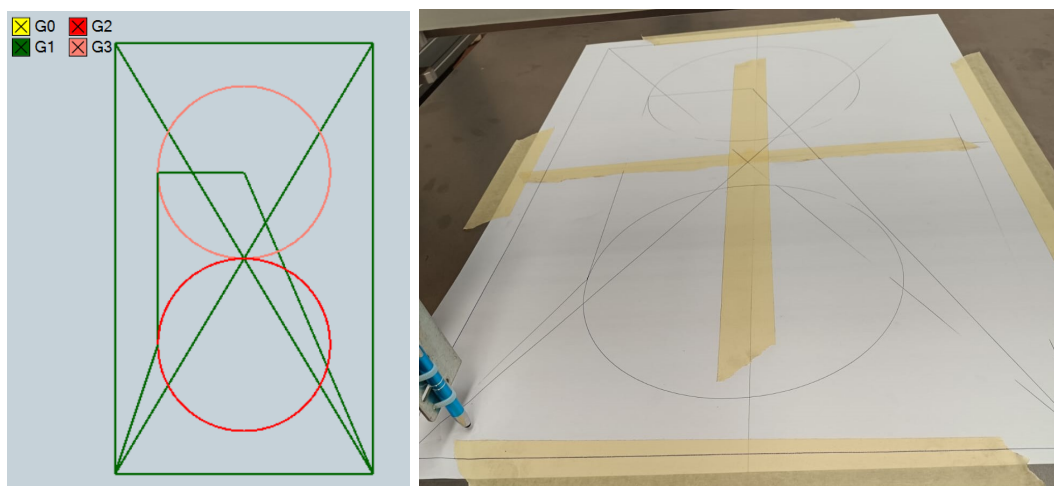
Para tal, foi elaborado um código G, cuja visualização do traçado no GCodeViewer é apresentada na Figura 6.11a. O objetivo consistia em analisar os traçados em linhas horizontais, verticais e diagonais, bem como os movimentos circulares e o respetivo comportamento. A Figura 6.11b mostra o traçado efetuado na mesa, enquanto as Figuras 6.11c e 6.11d ilustram as várias fases do diagnóstico do desenho pretendido. O intuito passa por eliminar *overshoots* no traçado e corrigir erros existentes, ajustando o *delay* ou o ganho. A Figura 6.12 apresenta alguns dos erros mencionados.

Como evidenciado nos gráficos anteriores, a intenção é que o traçado realizado, sinalizado pela linha vermelha, corresponda ao traçado pretendido, demarcado em azul. Na Figura 6.11c, pode-se observar um ligeiro desfasamento entre estas linhas, provocado por um desalinhamento dos ganhos.

Os erros comuns a serem considerados incluem os cantos e os pontos onde a direção é alterada bruscamente. Uma das formas de correção foi a utilização do G360, permitindo que o ponto de referência passasse pelos respetivos pontos, como é visualizado na Figura 6.13. Com a correção do tempo de base, realizada posteriormente, os resultados deste teste corresponderam às expectativas.

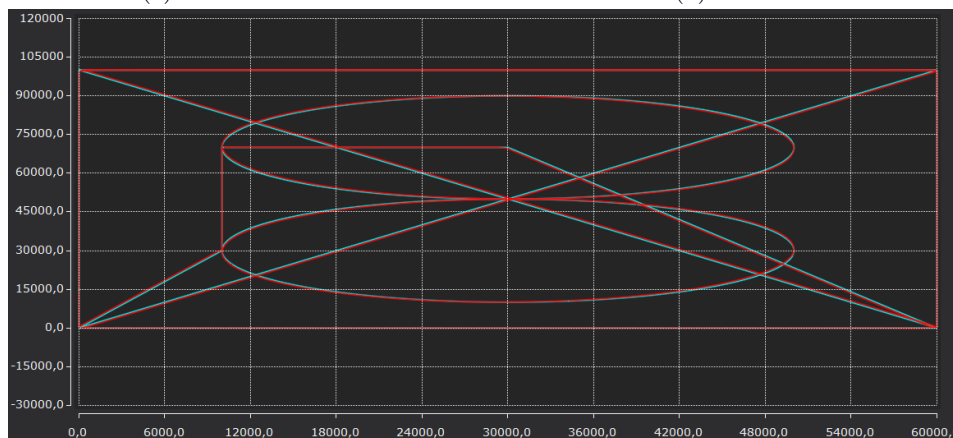
6.3.4 Homing

Após a confirmação de que o principal objetivo do teste anterior estava a funcionar, foi estudada a implementação do *homing*, de modo a garantir que os eixos estejam sempre calibrados.

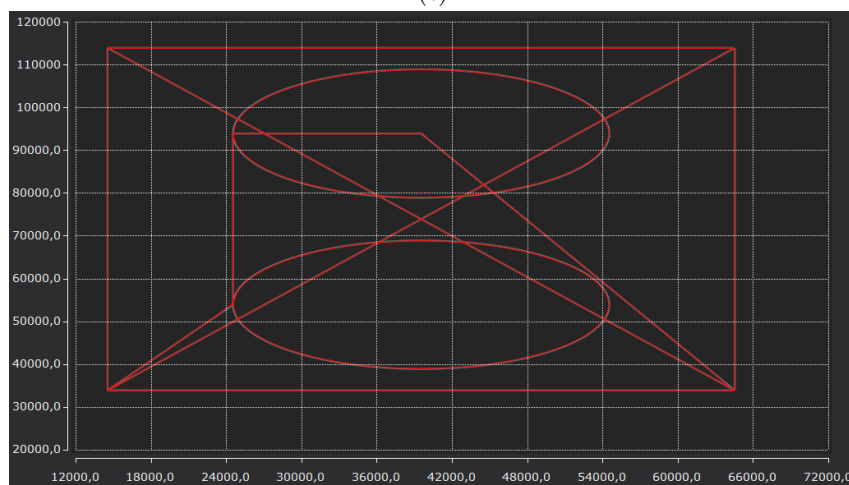


(a)

(b)



(c)



(d)

Figura 6.11: Teste do primeiro código G para teste de movimentos, onde o eixo horizontal corresponde ao eixo X e o eixo vertical ao eixo Y (Escala a $10^{-2}mm$): a) GCodeViewer b) Desenho da folha c) Teste intermédio d) Teste final

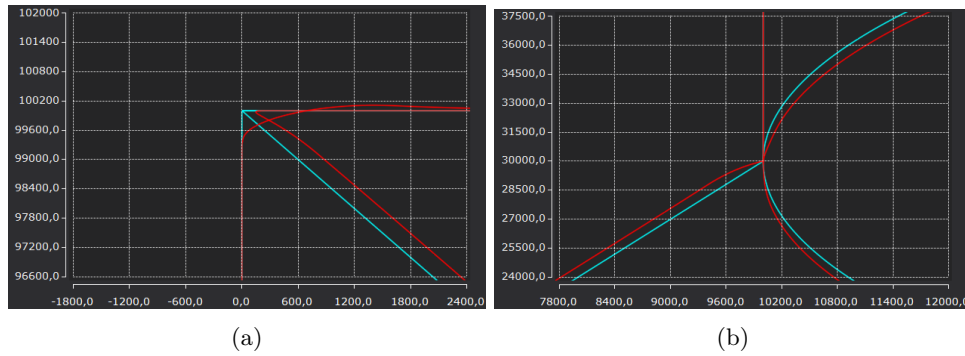


Figura 6.12: Erros comuns encontrados do decorrer de testes, onde o eixo horizontal corresponde ao eixo X e o eixo vertical ao eixo Y (Escala a $10^{-2}mm$): a) cantos b) troca repentina de direção



Figura 6.13: Correção dos erros encontrados do decorrer dos testes, onde o eixo horizontal corresponde ao eixo X e o eixo vertical ao eixo Y (Escala a $10^{-2}mm$): a) cantos b) troca repentina de direção

Após vários testes, foi possível realizar o *homing*, conforme o desejado, como demonstrado na Figura 6.14. Esta figura ilustra o movimento do *homing*, onde o sinal digital apresentado corresponde ao sinal da *cam* ligado ao respectivo eixo.

6.3.5 Interrupção de Emergência

Um dos elementos fundamentais a ser testado foi a interrupção de emergência. Sem a realização deste teste, não seria possível determinar a reação da máquina quando uma interrupção de emergência fosse acionada.

O teste foi efetuado durante uma operação automática, com ambos os eixos em funcionamento. Ao acionar o botão de emergência, verificou-se que ambos os eixos pararam de imediato. Na Figura 6.15, é possível observar que o sistema executou o processo com sucesso, onde a linha verde indica a paragem de emergência acionada, enquanto as linhas vermelhas e azul correspondem ao movimento dos eixos.

Para desbloquear os eixos, é necessário desativar o botão de emergência e clicar no botão para ligar o relé. Além disso, na HMI, é necessário realizar um *reset* e ligar novamente o *enable*.

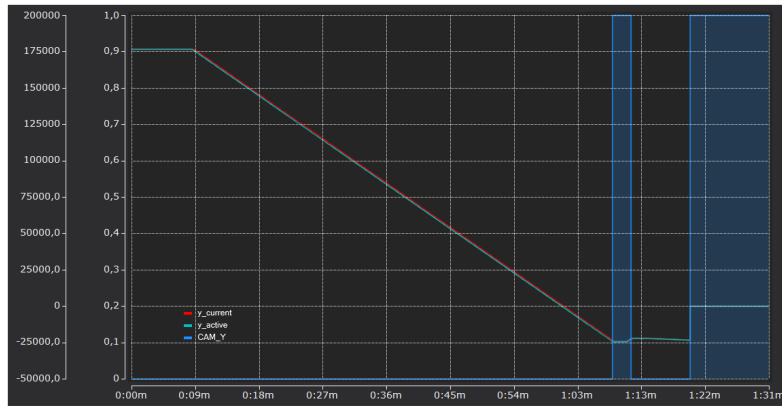


Figura 6.14: Teste de *homing* efetuado no eixo Y, onde o eixo horizontal corresponde ao tempo (em minutos) e o eixo vertical ao eixo Y (com escala a $10^{-2}mm$) e ao sinal digital da *cam*

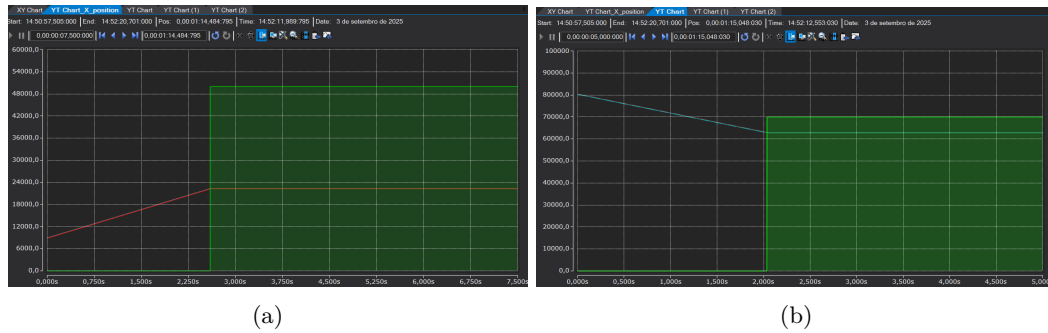
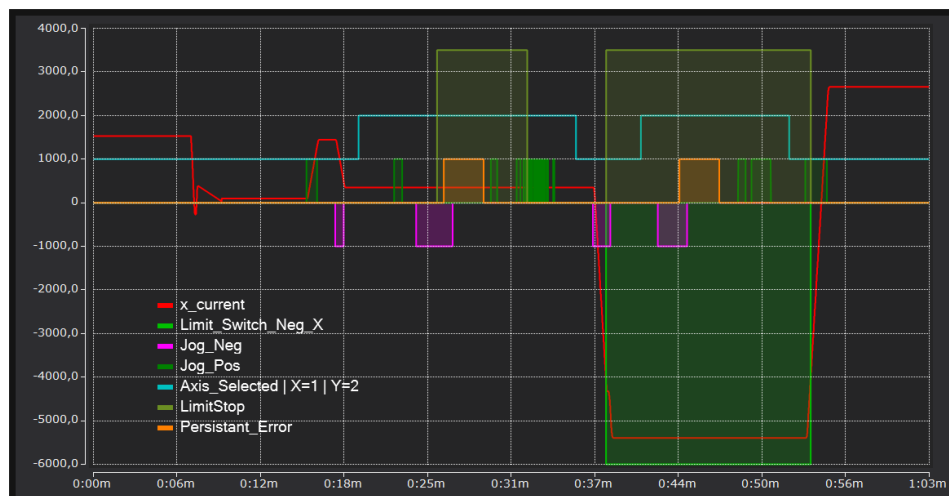


Figura 6.15: Teste da interrupção de emergência, onde o eixo horizontal corresponde ao tempo e o eixo vertical corresponde ao eixo do gráfico (Escala a $10^{-2}mm$): a) eixo X b) eixo Y

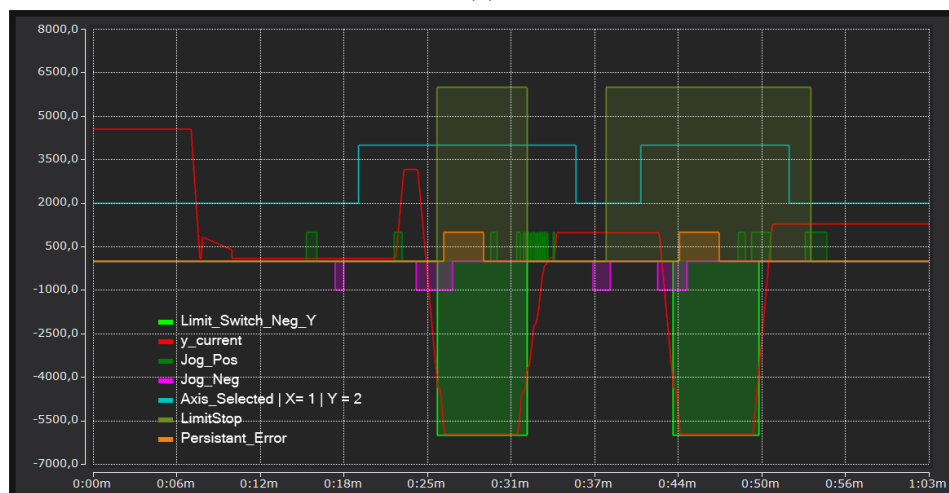
6.3.6 Teste dos Limites Físicos

Outro elemento importante de segurança são os fins de curso. Sem estes dispositivos, podem ocorrer colisões entre o eixo e a estrutura limitadora, o que pode provocar danos físicos à estrutura e/ou às correias. Com este teste, foi possível avaliar a lógica desenvolvida e mencionada na Subsecção 5.4.6.

Para realizar o teste, foi necessário verificar todos os fins de curso individualmente, de modo a confirmar o comportamento de cada sensor, bem como a detecção de um segundo sensor durante outro movimento, testando assim se o sistema bloqueia corretamente. A Figura 6.16 representa uma parte do teste, onde se observa que ambos os eixos atuam nos respectivos limites negativos da mesa. Além disso, quando um segundo sensor é detetado, o sistema entra em paragem e implementa um erro persistente.



(a)



(b)

Figura 6.16: Teste dos limites do eixo, onde o eixo horizontal corresponde ao tempo (em minutos) e o eixo vertical corresponde aos respectivos eixos em vermelho (Escala a $10^{-2}mm$), enquanto os restantes sinais correspondem aos sinais: a) Eixo X b) Eixo Y

6.3.7 Testes de Códigos G Complexos

Com todos os testes realizados com sucesso, foi possível testar códigos G elaborados por diversas ferramentas, como as provenientes de um site ou da ferramenta CAD/CAM.

Para isso, foram realizados testes em três códigos: dois elaborados por meio de um site e outro proveniente de CAD/CAM.

Código Proveniente de um Site

Existem vários *sites* e programas que conseguem adaptar figuras ou textos em código G, de acordo com as especificações fornecidas. Estes são geralmente elaborados para

o uso de imagens bidimensionais, sendo um formato fácil de processar.

Para este trabalho, utilizou-se o site CNC-Apps.com [76], onde se introduz o texto desejado, a fonte, o tamanho e outras variáveis, gerando um código G. Foram realizados testes com dois textos: "Paulo" e "Gislótica", conforme visualizado na Figura 6.17.

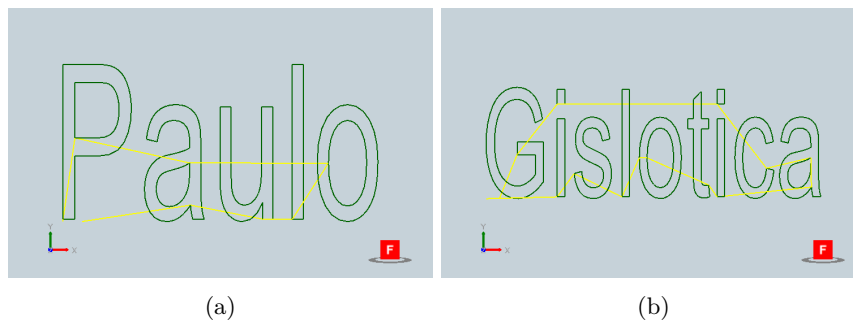


Figura 6.17: Texto efetuado no CNC-Apps vista no GCodeViewer:
a) Paulo b) Gislótica

Conforme visualizado na Figura 6.18a, é possível analisar os vários testes efetuados ao texto "Paulo", onde foi incluído o estudo do G360 como uma solução para evitar os *overshoot* e outros erros detetados, conforme evidenciado nos testes.

Na Figura 6.18b, é possível visualizar um conjunto de testes realizados com o G360, onde se altera o valor do respetivo limite. Na Tabela 6.7 encontram-se os tempos utilizados para a execução do código, e nas Figuras 6.18c e 6.18d são apresentados os dados obtidos do controlador em dois dos testes realizados.

Tabela 6.7: Tempos para a execução do teste ao código G "Paulo"

Tempos de Execução - "Paulo"		
Velocidade (mm/min)	Precisão da Janela (mm)	Tempo Necessário (min:seg)
1200	0,01	8:07
1200	0,1	4:45
1200	0,5	3:54
1200	1	3:34

Relativamente ao texto "Gislótica", o mesmo foi realizado conforme os testes anteriores. No entanto, foi nesta fase que se descobriu o tempo de base de cálculo, o que permitiu reduzir o tempo de execução, o que provocou o não acontecimento de *overshoot*, mesmo sem o G360, como demonstrado na Tabela 6.8. Na Figura 6.19, são analisados os vários testes realizados em papel, tanto sem como com o uso do G360, assim como a correção do tempo base. Na Figura 6.20, é possível observar os dados registados pelo controlador.

Com a realização deste teste, foi possível avaliar movimentos rápidos constantes e em diferentes direções, verificando a existência de erros no traçado planeado, como foi detetado nos primeiros ensaios de cada palavra, em que ocorreu *overshoot* ou



Figura 6.18: Teste efetuado ao código G "Paulo", onde o eixo horizontal corresponde ao eixo X e o eixo vertical ao eixo Y (Escala a $10^{-2}mm$): a) testes efetuados sem o G360 b) testes efetuados com o G360 c) teste com o G360 com a janela de 0,1 mm d) teste com o G360 com a janela de 1 mm

encurtamento de cantos, conforme observado no teste inicial do código G. Com o estudo do G360, foi possível executar o código com mais precisão, embora com muitas paragens entre as linhas de código G, o que aumentava o tempo de execução dos programas. A solução para este problema passaria por aumentar o valor da janela

Tabela 6.8: Tempos para a execução do teste no código G "Gislótica"

Tempos de Execução - "Gislótica"			
Tempo de Base (ms)	Velocidade (mm/min)	Precisão da Janela (mm)	Tempo Necessário (min:seg)
10	1200	0,1	7:55
10	1200	0,5	6:36
10	1200	1	6:20
2	1200	—	1:40

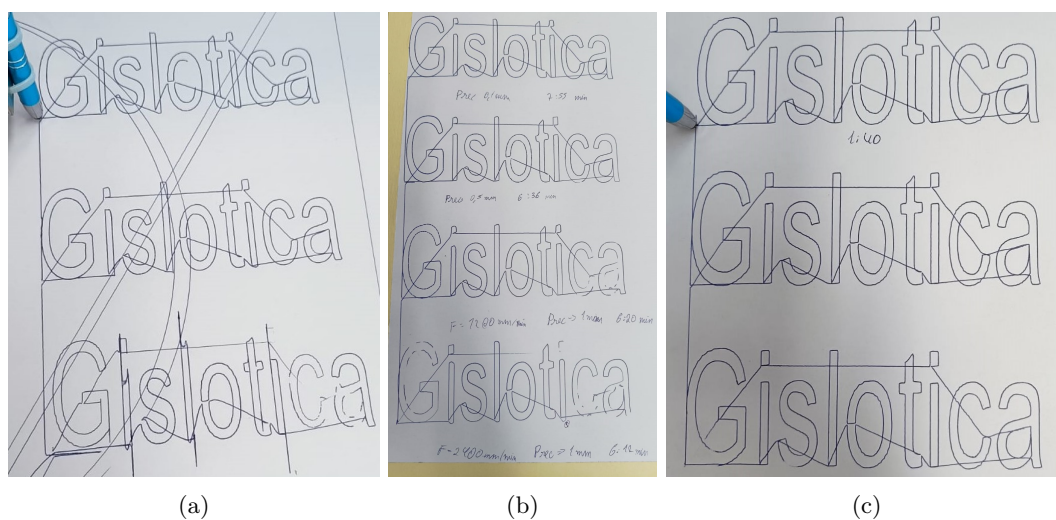


Figura 6.19: Teste efetuado ao código G "Gislótica" registados no papel: a) teste com o G360 e intervalo de 0,1 mm b) teste com o G360 e intervalo de 1 mm c) teste efetuado após a correção do tempo base

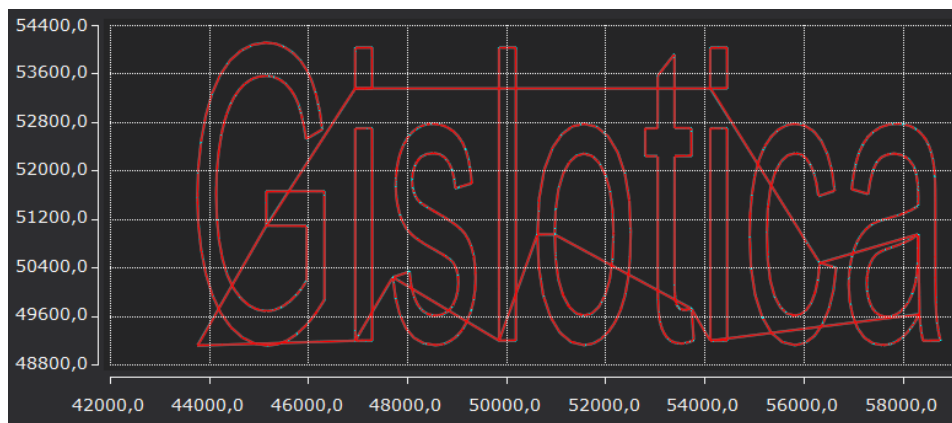
de precisão, o que poderia comprometer a precisão necessária ao trabalho. Com a redução do tempo base de execução, foi possível reduzir o tempo de execução, além de corrigir vários erros existentes e detetados ao longo do processo.

Código Proveniente da Ferramenta CAD/CAM

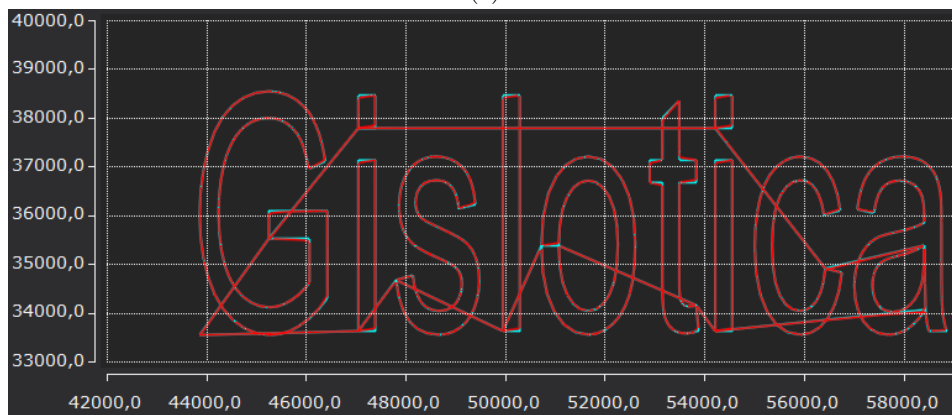
Com a conclusão dos testes com o código fornecido pelo *site*, foi possível realizar um estudo para executar um teste com um código proveniente de uma ferramenta CAD/CAM. Este teste visa comprovar a execução de uma etapa dos resultados esperados, permitindo (em teoria) a operação sobre uma peça.

O teste analisa um caso de estudo proveniente de uma peça fabricada pela empresa, a partir de um desenho CAD, configurando o *software* CAM e introduzindo-o na máquina.

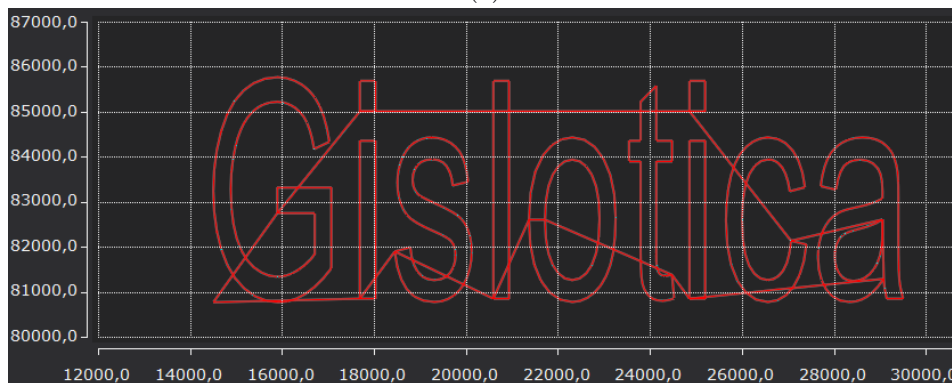
O caso de estudo analisado foi um quadro com uma HMI e botões físicos, numa máquina de verificação de pneus, designada Vitec, conforme ilustrado na Figura 6.21. O processo de fabrico desta peça, realizado manualmente com auxílio de ferramentas



(a)



(b)



(c)

Figura 6.20: Teste efetuado no código G "Gislótica" registados no *scope*, onde o eixo horizontal corresponde ao eixo X e o eixo vertical ao eixo Y (Escala a $10^{-2}mm$): a) testes efetuados sem o G360 b) testes efetuados com o G360 c) teste efetuado após a correção do tempo base

de corte, requer uma operação delicada de medidas, para que a interface gráfica e os botões existentes fiquem alinhados no quadro e de fácil acesso ao operador da máquina. Após efetuar as medidas, a operação é realizada com uma furação, seguida da execução dos retoques finais.



Figura 6.21: Quadro HMI numa Vitec

Com o recurso à fresadora aqui apresentada, e após o alinhamento da peça, pretende-se iniciar o processo, permitindo que a máquina efetue as furações, eliminando o tempo perdido com as marcações.

Para isso, no Inventor, foi criada uma representação gráfica do quadro numa chapa de 10 mm, sem estar na escala real da peça. As furações foram introduzidas no *software* CAM, onde, na parte onde se localiza a HMI e os botões são realizados em três passagens, onde o terceiro eixo descera, virtualmente. Na última passagem, pequenos recortes serão deixados, para que a parte que será extraída não se desloque, podendo causar alguma interferência com a ferramenta. Foi introduzida uma passagem da parte externa do quadro para registar o contorno. A ferramenta utilizada continha um diâmetro de dez milímetros, o que, nas passagens de corte resulta num desvio de 5 milímetros.

O código gerado passou por alguns ajustes, uma vez que os códigos G comuns não coincidiam com o código utilizado para o TwinCAT, como já mencionado na Subsecção 5.5. O código G gerado foi verificado no GCodeViewer, conforme demonstrado na Figura 6.22.

Durante os testes deste código, foi ainda possível avaliar o funcionamento do *spindle* em ambiente simulado, conforme visualizado na Figura 6.23, assim como a visualização dos resultados obtidos em papel e dos registos do TwinCAT, permitindo observar todos os eixos, como demonstrado na Figura 6.24.

Com a realização deste teste, verificou-se que o resultado foi o esperado, permitindo que a máquina prosseguisse para os testes seguintes. No entanto, foi detetada uma falha mecânica, conforme visualizado na Figura 6.24a, onde ilustra uma fase inicial do erro. Nos círculos, observam-se pequenas falhas, como desvios, ou uma diferença das posições finais entre os vários testes sem alterar o programa, mas na HMI correspondia ao mesmo valor em todos os testes, indicando que todos os pontos

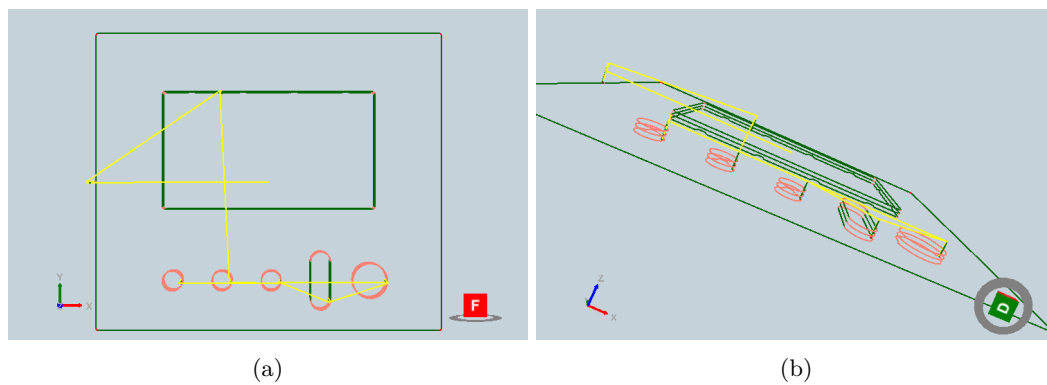
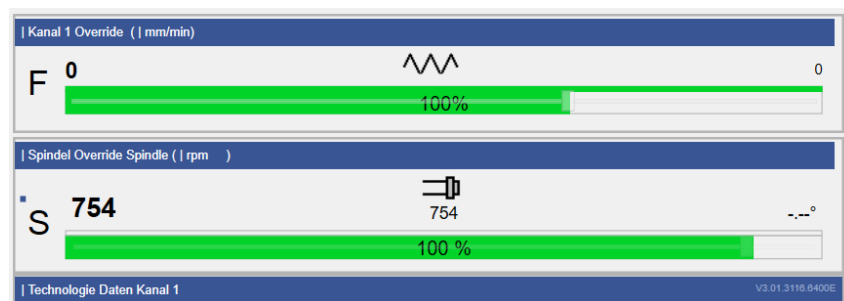
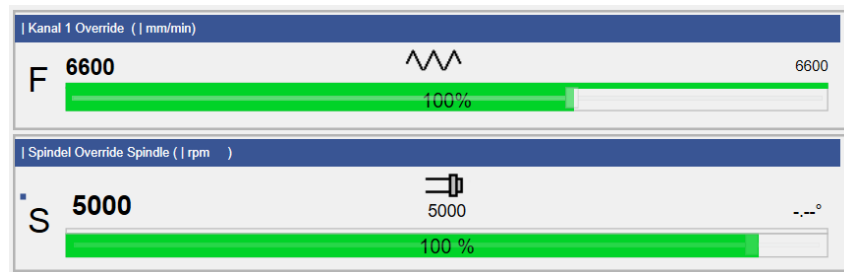


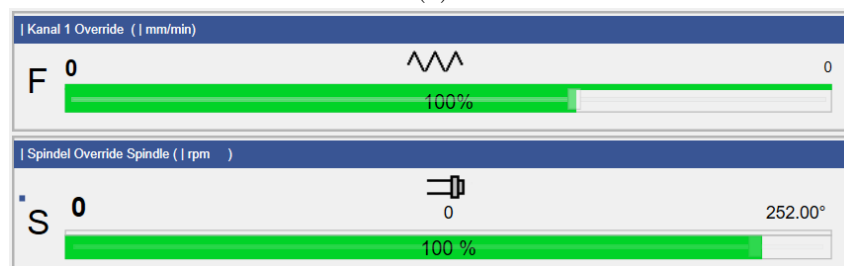
Figura 6.22: Teste efetuado no código G elaborada por ferramenta CAD/CAM registados no GCodeViewer: a) visão frontal b) visão lateral



(a)



(b)



(c)

Figura 6.23: Teste efetuado ao *spindle* em simulação: a) Instância inicial b) Instância intermédia c) Instância final

finais acabaram praticamente no mesmo sítio, o que não foi visualizado.

Em testes mais avançados, conforme visualizado na Figura 6.25, foi detetada

a falha mecânica. A Figura 6.25a mostra o último círculo efetuado em diversos testes semelhantes, marcados com um "T", que não se alinhavam corretamente. Embora os valores do TwinCAT e do *driver* do motor correspondessem, as falhas foram evidenciadas no lado esquerdo dos círculos. Na Figura 6.25b, observam-se duas falhas que culminaram no veredicto final. A primeira refere-se a um corte maior, onde se verifica um desvio de cerca de 20 mm em relação ao ponto inicial, representando um desvio considerável do eixo Y. A segunda falha é visível sobre os círculos pequenos, onde o movimento passou para uma linha reta controlada do movimento X, enquanto o eixo Y parou de funcionar, o que não era suposto acontecer.

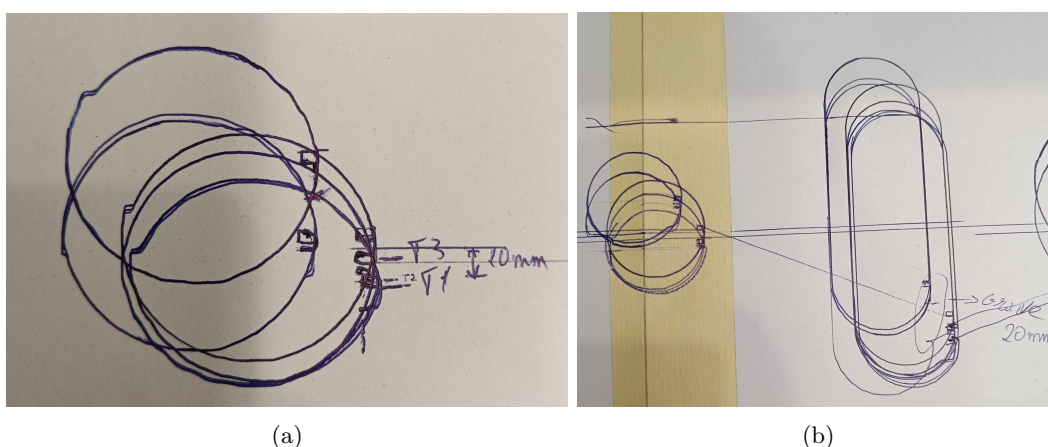


Figura 6.25: Erros encontrados resultantes de uma possível falha mecânica: a) ponto final do programa com desvio b) deslizamento das posições

Com base no que foi relatado e visualizado nas figuras anteriores, pode-se concluir que o erro deriva de um escorregamento na caixa redutora. Quando o motor foi montado, os componentes que constituem o funcionamento do eixo, neste caso motor e a caixa redutora, apresentavam uma massa gordurosa para proteger os componentes quando não usados. Antes de serem acoplados, é necessário realizar a limpeza desta massa para o eixo corresponder às especificações, o que não foi adequadamente realizado.

Na fase inicial, este deslizamento não era notável, uma vez que existe atrito suficiente. Contudo, à medida que os testes são realizados, surgiu um ligeiro desvio. Com a repetição dos testes, não se deu a devida importância a este desvio, pois era efetuado somente um desenho único e não se repetia, resultando num erro crescente ao longo dos testes. Relativamente aos picos de erro detetados no teste individual de cada motor, estes podem ser provocados por esta falha.

Após a reparação do problema mecânico e a realização de novos testes, obteve-se o resultado ilustrado na Figura 6.26, onde o desvio entre os diferentes testes foi de aproximadamente 2 mm, provenientes da ferramenta.

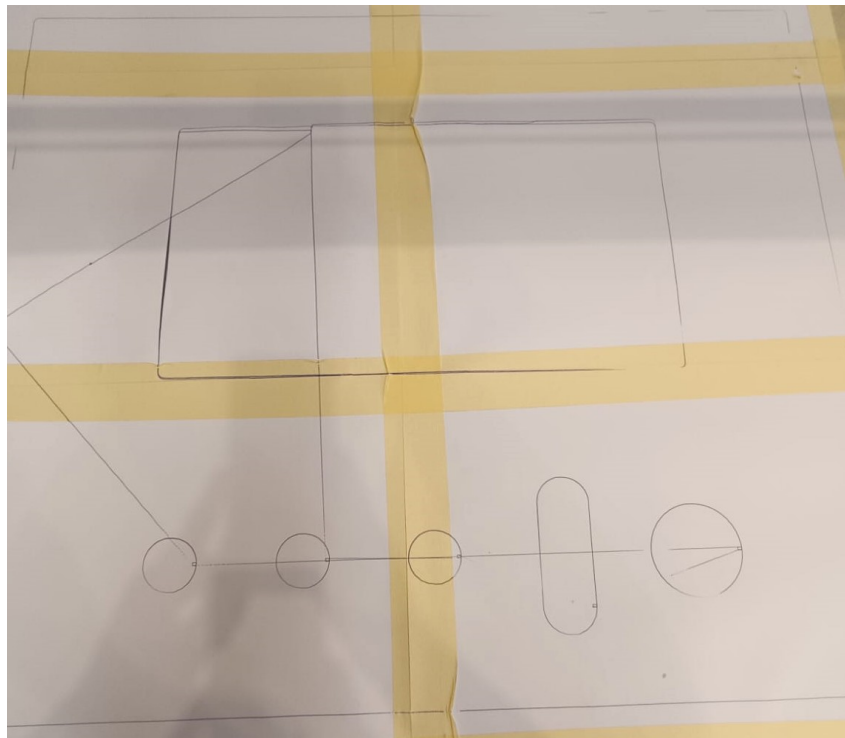


Figura 6.26: Quadro Vitec desenhado após a correção da falha mecânica e várias passagens

6.4 Testes por Realizar

Devido a vários imprevistos, não foi possível concluir o trabalho a tempo; no entanto, é necessário referir os testes projetados para finalizar o projeto e os seus propósitos.

Foram planeados vários testes, incluindo: teste do *spindle* separado do conjunto, teste no eixo Z, implementação do conjunto restante no projeto final e teste final a uma peça.

6.4.1 Teste do *Spindle* Separado do Conjunto

O teste com o *spindle* separado do conjunto permitiria a verificação do circuito do *spindle*, possibilitando testar se a implementação do *hardware* foi realizada conforme o proposto. O controlo será efetuado a partir do programa-base da CNC, permitindo o controlo do motor. Poderá ser necessário ajustar o VFD, bem como parâmetros não mencionados do TwinCAT.

6.4.2 Teste no Eixo Z

O teste no eixo Z é semelhante ao que foi realizado nos eixos X e Y, sendo necessário efetuar uma calibração e realizar testes básicos para observar o comportamento do respetivo eixo.

6.4.3 Implementação do Conjunto Final

Após a realização dos testes separados ao *spindle* e ao eixo Z, proceder-se-á aos testes do conjunto final, incluindo:

- análise da repetibilidade, permitindo verificar se o conjunto apresenta uma boa repetibilidade e se existe alguma falha não detetada;
- teste de códigos G que implementem o movimento dos três eixos e do conjunto do *spindle*, realizando testes sem a implementação de nenhuma peça.

6.4.4 Teste numa Peça

Após a verificação de que o conjunto está operacional, será realizado o teste final a uma ou várias peças, escolhidas pela empresa e processadas por um *software* CAD/CAM para obter um programa NC, a ser implementado na máquina.

Recomenda-se que os primeiros testes de cada programa sejam realizados sem a peça, permitindo identificar possíveis erros e colocar o zero *offset* adequado para a peça. Após a conclusão da análise, procede-se à operação da peça, verificando se a operação foi realizada com sucesso e repetindo o processo várias vezes para analisar a existência de erros.

6.5 Conclusão dos Testes e Resultados

Este capítulo teve como objetivo apresentar os resultados obtidos durante os diversos testes planeados (e realizados) para este projeto, comprovando a evolução do trabalho e as modificações efetuadas, conforme mencionado na implementação.

Numa fase inicial, foi realizada uma breve abordagem sobre os testes simulados, destacando a sua importância, a ambientação ao ambiente TwinCAT e a identificação dos pontos a melhorar ao longo do tempo.

Em seguida, foram testados os motores dos eixos e respetivos *drivers*, analisando as ligações, o funcionamento do FCT e outros parâmetros. Foi também feita uma observação dos resultados obtidos para verificar a existência de erros mecânicos ou elétricos, assim como a precisão. Suspeitou-se de um possível problema mecânico no eixo Y, que foi detetado, descrito e corrigido mais a frente.

Posteriormente, foi realizado um conjunto de testes ao código da CNC, evidenciando a sua evolução em relação ao projeto inicial. Foram mencionadas as várias fases, como a calibração, ganhos, *homing*, paragem de emergência e de limites, assim como os diversos testes realizados com diferentes códigos G. Analisaram-se os dados obtidos, como o seguimento da posição e a verificação da realização dos respetivos testes. Com esta etapa, foi confirmado o funcionamento da máquina.

Por fim, foram referidos os testes que ficaram por realizar, assim como os motivos que impediram a sua execução. Em cada caso, foi apresentada a razão do teste e as expectativas de dados que se pretendia obter.

Em resumo, este capítulo documentou o trabalho desenvolvido, apresentando os respetivos resultados e realizando uma análise crítica dos mesmos.

Capítulo 7

Conclusões

Após a realização deste projeto, conclui-se que os objetivos inicialmente propostos foram alcançados, embora com limitações no funcionamento do terceiro eixo e do *spindle*. Os resultados obtidos demonstraram o movimento dos dois eixos, utilizando uma caneta como ferramenta, considerando estes testes como etapas intermédias antes da implementação final.

Em termos de avaliação do projeto final, face aos equipamentos disponíveis em *stock* (sendo que apenas os terminais da Beckhoff e o *spindle* foram adquiridos), a solução encontrada revela-se viável e apresenta características interessantes em termos de execução de programas.

Embora o estado da máquina não esteja totalmente finalizado, acredita-se que, após a conclusão e finalização, esta poderá contribuir significativamente para o processo de furação e fresagem de peças em chapa ou materiais semelhantes. A implementação da lógica introduzida do *software* do TwinCAT está conforme o planeado, mas é necessário realizar vários testes, incluindo a injeção de paragens, como o *stop* e emergência, e outros procedimentos fora do comum como os *reset*, para verificar a existência de eventuais *bugs* visuais ou críticos que não tenham sido detetados até à finalização do trabalho. A solução encontrada é comparável a outras soluções disponíveis.

As principais dificuldades enfrentadas durante a realização deste trabalho incluem a introdução ao TwinCAT, uma vez que existem diversas configurações que este permite, e a análise do código-base fornecido da CNC, que apresentava pouca flexibilidade para alterações e uma documentação extensa, grande parte da qual se

revelou inútil para este trabalho, além de muitas variáveis estarem em alemão, a língua nativa da Beckhoff. Embora houvesse muita informação sobre parâmetros para a CNC, muitos deles não estavam visíveis.

No que diz respeito às dificuldades relativas ao *hardware*, foram enfrentadas algumas, destacando-se a detecção de uma avaria no *driver* do motor Y, que apresentava um estado diferente em relação ao outro *driver*, mas funcional. Após a ativação do *enable*, a posição correspondente era reiniciada, o que levou à remoção dos *software limits*. Adicionalmente, as falhas mecânicas do eixo Y provocavam frequentemente *overshoot* da posição ou falhas de posições, mas foram todas resolvidas. Em termos de ligações, houve uma complicação nas conexões entre o *driver* e o EL2522, uma vez que os cabos DB9 normais contêm o *shield* ligado a um dos pinos, o que correspondia, do lado do *driver*, ao pino VCC, resultando num erro no controlador. Para resolver esta questão, foi necessário criar um cabo específico para a ligação.

7.1 Trabalho Futuro

Considerando a solução encontrada, é possível planejar uma evolução do projeto.

Num primeiro ponto, propõe-se a implementação do terceiro eixo e do *spindle*, podendo ser utilizados os motores mencionados no Capítulo 4 ou outros de características semelhantes, e realizar os testes finais referidos no final do Capítulo 6.

Outro ponto de melhoria consiste na substituição do mecanismo mecânico de transmissão do movimento, passando de transmissão por correias para uma transmissão em parafuso, uma vez que a elasticidade da correia pode provocar erros dependendo do uso ou das forças aplicadas. Sugere-se também a substituição dos motores dos eixos, passando dos motores de passo para servomotores, uma vez que os servomotores são mais adequados para movimentos lineares em comparação com os motores de passo, além de serem amplamente utilizados em CNC.

Uma nova melhoria proposta é a introdução do *autotuning*. Todas as plataformas de automação dispõem de uma ferramenta chamada *autotuning*, que permite otimizar o funcionamento dos sistemas. Através da análise do motor, esta ferramenta determina o melhor método de controlo, incorporando vários métodos heurísticos, como o de Ziegler-Nichols, e adapta o melhor para o motor em questão. Esta implementação foi considerada, mas os componentes existentes não a permitem. Para tal, é necessário introduzir algum dos seguintes *hardware*, disponibilizados pela Beckhoff [19], para possibilitar a realização do *autotuning*:

- motores servos da Beckhoff AX5xxx, AX8xxx, AMP8xx e AMI8xxx;
- motores de passo da Beckhoff ASI8xxx.
- terminais de comunicação EtherCAT para servomotores da Beckhoff EL72xx, EJ72xx e EP72xx;

- terminais de comunicação EtherCAT para motores da Beckhoff BLDC EL7411 e EJ7411;
- terminais de comunicação para os motores de passo da Beckhoff EL7037, EJ7037, EP7037, EL7047, EJ7047, EP7047 e EL7062;
- *drivers* da Beckhoff ELM72xx;
- fontes de alimentação para os motores da Beckhoff MD6xxx e MD8xxx;
- sensores *encoder* da Beckhoff MO72xx.

Para finalizar, é possível explorar algumas aplicações adicionais que poderiam melhorar a solução, sendo três propostas para este efeito. A primeira consiste na implementação de um mecanismo de troca de ferramentas automático. A segunda proposta envolve a criação de um mecanismo de calibração automática da ferramenta, utilizando um sensor de pressão na ferramenta ou um botão localizado num ponto específico da mesa. A última proposta sugere um mecanismo que deteta automaticamente a posição da peça, permitindo à máquina realizar a operação consoante o respetivo *offset*.

Referências

- [1] E. H. Taípe, Carlos Andres Junior Sahuanay e Cayo, “Projeto e prototipagem de uma máquina cnc de 4 eixos para fabricação de pequenos acessórios,” in *2024 10ª Conferência Internacional sobre Engenharia Mecatrônica e Robótica (ICMRE)*, pp. 5–9, 2024. Acesso em: 22 jan. 2025. [Citado na página 1]
- [2] H. A. R. Hans B. Kief and K. Schwarz, *The CNC Handbook Digital Manufacturing and Automation from CNC to Industry 4.0*. Industrial Press, Inc., 2021. [Acesso em: 17 mar. 2025]. [Citado nas páginas xiii, 1, 9, 11, 16, 26 e 36]
- [3] “Computer Numerical Control (CNC) Machine Market Size, Report 2034.” <https://www.precedenceresearch.com/computer-numerical-control-machine-market>, 2025. [Acesso em: 22 jan. 2025]. [Citado na página 1]
- [4] S. F. Krar, A. Gill, and P. Smid, *Computer numerical control simplified*. Industrial Press Inc., 2001. [Acesso em: 03 fev. 2025]. [Citado nas páginas 8 e 9]
- [5] P. Kumar Negi, M. Ram, and O. P. Yadav, *1 CNC Machine and Its Importance*, pp. 1–16. IEEE, 2018. [Acesso em: 03 fev. 2025]. [Citado nas páginas 8 e 9]
- [6] S. Guler, *Finite Element Analysis of a Vertical Machining Center with Aluminum Profile Framing*. PhD thesis, İskenderun Technical University, 06 2013. [Acesso em: 03 fev. 2025]. [Citado nas páginas xiii, 8 e 9]
- [7] M. P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing*. USA: Prentice Hall PTR, 2nd ed., 2000. [Acesso em: 16 mar. 2025]. [Citado nas páginas xiii, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 177 e 179]
- [8] E. Oberg, F. Jones, C. McCauley, H. Ryffel, and R. Heald, *Machinery’s Handbook*. Industrial Press, Incorporated, 2004. [Acesso em: 20 fev. 2025]. [Citado na página 18]
- [9] H. Gurocak, *Industrial Motion Control: motor selection, drives, controller tuning, applications*. John Wiley & Sons, 2015. [Acesso em: 2 mar. 2025]. [Citado nas páginas xiii, 18, 19, 21 e 22]

- [10] “Programação por trás da impressão 3d - pet sistemas de informação.” <https://www.ufsm.br/pet/sistemas-de-informacao/2023/10/15/programacao-por-tras-da-impressao-3d>, 2023. [Acesso em: 13 mar. 2025]. [Citado na página 24]
- [11] “Quality - dannik-bg.” <https://dannik-hydraulics.tech/en/quality/>, 2025. [Acesso em: 13 mar. 2025]. [Citado na página 24]
- [12] H. H. Agrisa, “An overviwe of proses cnc machining,” *Journal of Mechanical Science and Engineering*, vol. 6, no. 2, pp. 19–23, 2019. [Acesso em: 5 mar. 2025]. [Citado na página 25]
- [13] R. Harik and T. Wuest, *Introduction to advanced manufacturing*. SAE International, 2019. [Acesso em: 5 mar. 2025]. [Citado na página 25]
- [14] X. Xu and A. Y. C. Nee, *Advanced design and manufacturing based on STEP*. Springer Science & Business Media, 2009. [Acesso em: 5 mar. 2025]. [Citado na página 25]
- [15] Denford Ltd., *G and M Programming for CNC Milling Machines*. Denford Ltd., Brighouse, West Yorkshire, England, 1997. Disponível em: https://www.iqytechnicalcollege.com/G_and_M_programming_for_mills_manual.pdf [Acesso em: 16 mar. 2025]. [Citado nas páginas xiii, xix, 25, 26, 27, 28, 29, 177 e 179]
- [16] A. Armenta, “What is an industrial pc?.” <https://control.com/>, 2021. [Acesso em: 17 mar. 2025]. [Citado na página 30]
- [17] CTSolutions, “Plc vs industrial pc: What’s the difference?.” <https://www.candtsolution.com/>, 2024. [Acesso em: 17 mar. 2025]. [Citado nas páginas 30 e 31]
- [18] PremioInc, “The differences between industrial pcs and plcs.” <https://premioinc.com/>, 2022. [Acesso em: 17 mar. 2025]. [Citado nas páginas 30 e 31]
- [19] Beckhoff, “Beckhoff information system - documentation data.” <https://infosys.beckhoff.com/>, 2025. [Acesso em: 16 set. 2025]. [Citado nas páginas xiv, xv, xix, 31, 46, 47, 48, 49, 50, 78, 79, 80, 81, 82, 83, 89, 90, 91, 92, 93, 101, 102, 113, 114, 115, 116, 117, 119, 121, 166, 219 e 223]
- [20] “Buy siemens plc system .. - aotewell automation.” <https://www.aotewell.com/>, 2025. [Acesso em: 16 mar. 2025]. [Citado na página 31]
- [21] “Proyetos y servicios - prosistec.” https://prosistec.cl/Proyectos_y_Servicios/, 2025. [Acesso em: 16 mar. 2025]. [Citado na página 31]

-
- [22] A. Hughes and B. Drury, *Electric motors and drives: fundamentals, types and applications*. Newnes, 2019. [Acesso em: 25 mar. 2025]. [Citado nas páginas xiii, 32, 33, 34 e 35]
- [23] “Série 660 - airturbinetools.com.” <https://www.airturbinetools.com/fr/products/spindles/spindle-series/660-series/>, 2025. [Acesso em: 18 mar. 2025]. [Citado nas páginas xiii e 36]
- [24] DYNOMAX, “The book of spindles - part 1,” 2006. [Acesso em: 31 mar. 2025]. [Citado nas páginas 36, 37, 38 e 39]
- [25] F. A. F. de Castro, “Conceção e instalação de um controlador de um spindle para integração em célula robótica,” Master’s thesis, Universidade do Porto, 2012. [Acesso em: 31 mar. 2025]. [Citado nas páginas 36 e 37]
- [26] E. T. Group, “Ethercat - the ethernet fieldbus - website.” <https://www.ethercat.org/en/technology.html>, 2025. [Acesso em: 27 abr. 2025]. [Citado nas páginas 40, 41 e 42]
- [27] J. Sagatowski, “Plc programming using twincat 3 - io - youtube video.” <https://www.youtube.com/@JakobSagatowski>. [Acesso em: 27 abr. 2025]. [Citado nas páginas xiii, 40 e 41]
- [28] NI, “The basics of canopen.” <https://www.ni.com/en.html>, 2025. [Acesso em: 29 abr. 2025]. [Citado nas páginas xiii, 42, 43 e 44]
- [29] M. Falch, “Canopen explained - a simple intro.” <https://www.csselectronics.com/>, 2025. [Acesso em: 29 abr. 2025]. [Citado nas páginas 42, 44 e 45]
- [30] Y. Dote, *Servo motor and motion control using digital signal processors*. USA: Prentice-Hall, Inc., 1990. [Acesso em: 28 abr. 2025]. [Citado nas páginas 45 e 46]
- [31] “Incremental encoder signals 101.” <https://www.haydonkerkpittman.de/learningzone/whitepapers/incremental-encoder-signals>, 2025. [Acesso em: 27 abr. 2025]. [Citado nas páginas xiii e 45]
- [32] “1 - pulse (step/dir) mode and 2-pulse (cw7ccw) mode for stepping motor - knowledge ni.” <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YHvvCAG&l=en-US>, 2020. [Acesso em: 27 abr. 2025]. [Citado nas páginas xiv e 46]
- [33] J. Sagatowski, “Plc programming using twincat 3 - ads - youtube video.” <https://www.youtube.com/@JakobSagatowski>. [Acesso em: 27 abr. 2025]. [Citado nas páginas xiv, 48 e 49]

- [34] “Grbl - github.” <https://github.com/grbl/grbl>, 2023. [Acesso em: 3 mai. 2025]. [Citado nas páginas 53 e 54]
- [35] “Linuxcnc - website.” <https://linuxcnc.org/>, 2025. [Acesso em: 4 mai. 2025]. [Citado nas páginas 53 e 56]
- [36] “Mach cnc software home of mach3 / mach4.” <https://www.machsupport.com/>, 2025. [Acesso em: 3 mai. 2025]. [Citado na página 53]
- [37] N. Wicaksono and A. Goeritno, “Designing an arduino board-based electronic device driven by grbl gru to operate the mini pcb printing machine,” *Jurnal Rekayasa Elektrika*, vol. 18, 09 2022. [Acesso em: 3 mai. 2025]. [Citado nas páginas xiv, 54, 55 e 56]
- [38] S. Zivanovic, Z. Dimic, V. Nikola, and S. Mitrovic, “Configuring of 3 axis mini cnc machine tool with control system based on linuxcnc,” in *International Conference on Accomplishments in Mechanical and Industrial Engineering*, 05 2019. [Acesso em: 4 mai. 2025]. [Citado nas páginas xiv, 56, 57, 58 e 59]
- [39] M. Ivan, M. Ernek, and E. Miklovičová, “Twincat control of cnc machine over coe,” in *2019 22nd International Conference on Process Control (PC19)*, pp. 91–96, 2019. [Acesso em: 7 mai. 2025]. [Citado nas páginas xiv, 59, 60 e 61]
- [40] P. Sun, Q. Liu, J. Ding, and S. Pi, “Open cnc system design for multiple intelligent functions based on twincat and .net framework,” in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 910–915, 2017. [Acesso em: 13 set. 2025]. [Citado nas páginas xiv, 61, 62, 63, 64 e 65]
- [41] R. Oliveira, “Medidor do perímetro de um talão por nuvem de pontos,” 2024. [Acesso em: 12 mai. 2025]. [Citado nas páginas xiv, 67, 68 e 69]
- [42] “Siemens website.” <https://www.siemens.com/global/en.html>, 2025. [Acesso em: 6 mai. 2025]. [Citado nas páginas xiv, xv, 72, 73, 74, 75 e 86]
- [43] “Sentron 3ld2003-1tl53.” <https://docs.rs-online.com/0ea9/A700000006969978.pdf>, 2023. [Acesso em: 6 mai. 2025]. [Citado na página 72]
- [44] “Sentron 5sm3311-6.” <https://datasheet.octopart.com/5SM33116-Siemens-datasheet-45428713.pdf>, 2015. [Acesso em: 6 mai. 2025]. [Citado na página 73]
- [45] “Iec homepage.” <https://webstore.iec.ch/en/>, 2025. [Acesso em: 26 jun. 2025]. [Citado nas páginas 73, 74 e 86]

-
- [46] GroupSumi, “As diferenças existentes entre os magnetotérmicos e os diferenci-ais.” <https://groupsumi.pt/blog/>, 2024. [Acesso em: 12 mai. 2025]. [Citado nas páginas 73 e 74]
- [47] “Allen-bradley homepage.” <https://www.rockwellautomation.com/en-us/products/hardware/allen-bradley.html>, 2025. [Acesso em: 6 mai. 2025]. [Citado nas páginas xiv e 74]
- [48] “1492-d dc circuit breaker.” <https://docs.rs-online.com/3d0c/0900766b81543fef.pdf>, 2016. [Acesso em: 6 mai. 2025]. [Citado na página 74]
- [49] “Sitop 6ep1334-3ba00.” <https://docs.rs-online.com/67c6/0900766b8131a3ba.pdf>, 2014. [Acesso em: 6 mai. 2025]. [Citado na página 75]
- [50] “Murr elektronik homepage.” <https://www.murrelektronik.pt/pt/>, 2025. [Acesso em: 6 mai. 2025]. [Citado nas páginas 75 e 76]
- [51] “Mico 2 channels.” <https://docs.rs-online.com/d115/0900766b81548d79.pdf>, 2016. [Acesso em: 6 mai. 2025]. [Citado na página 75]
- [52] “Mico 4 channels.” <https://docs.rs-online.com/e762/0900766b81548d75.pdf>, 2016. [Acesso em: 6 mai. 2025]. [Citado nas páginas 75 e 76]
- [53] “Pilz homepage.” <https://www.pilz.com/pt-PT/products/relay-modules/safety-relays-protection-relays>, 2025. [Acesso em: 6 mai. 2025]. [Citado nas páginas 76 e 77]
- [54] “Pnoz xv2.” <https://docs.rs-online.com/7382/0900766b8002f024.pdf>, 2011. [Acesso em: 6 mai. 2025]. [Citado nas páginas 76 e 77]
- [55] ABB, “O que é um relé de segurança e para que serve?.” <https://loja.br.abb.com/blog/>, 2025. [Acesso em: 14 mai. 2025]. [Citado na página 77]
- [56] “Iso - international organization for standardization.” <https://www.iso.org/home.html>, 2025. [Acesso em: 27 mai. 2025]. [Citado nas páginas 78 e 86]
- [57] “Cx9020 | embedded-pc.” https://infosys.beckhoff.com/english.php?content=../content/1033/cx9020_hw/3535296395.html&id=. [Acesso em: 6 mai. 2025]. [Citado na página 79]
- [58] “Ek110x-00xx, ek15xx | ethercat bus coupler.” https://infosys.beckhoff.com/english.php?content=../content/1033/ek110x_ek15xx/7636174091.html&id=, 2025. [Acesso em: 6 mai. 2025]. [Citado na página 80]

- [59] “El10xx, el11xx | digital input terminals.” https://infosys.beckhoff.com/english.php?content=../content/1033/el10xx_el11xx/1623205643.html&id=, 2025. [Acesso em: 6 mai. 2025]. [Citado na página 81]
- [60] “El20xx, el2124 | digital output terminals.” https://infosys.beckhoff.com/english.php?content=../content/1033/el20xx_el2124/1636470795.html&id=, 2025. [Acesso em: 6 mai. 2025]. [Citado na página 81]
- [61] “El2522.” <https://infosys.beckhoff.com/english.php?content=../content/1033/el252x/1857493643.html&id=>, 2025. [Acesso em: 6 mai. 2025]. [Citado na página 82]
- [62] “El4002.” <https://infosys.beckhoff.com/english.php?content=../content/1033/el40xx/12759408779.html&id=>, 2025. [Acesso em: 5 set. 2025]. [Citado na página 83]
- [63] “Festo - site.” <https://www.festo.com/pt/pt/>, note = [Acesso em: 5 mai. 2025], 2025. [Citado nas páginas xv, 84, 85, 89, 107, 108, 109 e 110]
- [64] “Stepper motors emms-st.” <https://docs.rs-online.com/f93e/A700000008518644.pdf>, 2012. [Acesso em: 6 mai. 2025]. [Citado na página 84]
- [65] “Cmms-st-c8-7-g2.” <https://www.manualslib.com/manual/1528228/Festo-Cmms-St-C8-7-G2.html>, 2012. [Acesso em: 6 mai. 2025]. [Citado na página 85]
- [66] “Unidade da engrenagem emga.” https://www.festo.com/media/catalog/208016_documentation.pdf, 2023. [Acesso em: 6 mai. 2025]. [Citado na página 85]
- [67] “3se5423-0cd20-1eb1.” <https://mall.industry.siemens.com/mall/en/en/Catalog/product?mlfb=3SE5423-0CD20-1EB1>, 2025. [Acesso em: 6 mai. 2025]. [Citado na página 86]
- [68] “Torre de sinalização industrial - cablematic.” <https://cablematic.com/pt/produtos/torre-de-sinalizacao-industrial-semaforo-com-luz-led-vermelha-laranja-verde>, 2020. [Acesso em: 03 jun. 2025]. [Citado nas páginas xv e 86]
- [69] “Hltnc - online shopping for stepper motors, linear guides, bearings, cnc.” <https://hlmotion.com/>, 2025. [Acesso em: 5 set. 2025]. [Citado na página 87]
- [70] “1.5kw 220v / 110v er11 24000rpm 400hz eixo refrigerado a ar com inversor yl620 vfd.” <https://hlmotion.com/collections/spindle-vfd/products/>

- hltn-c-sales-air-cooling-spindle-motor-kit-1-5kw-1500w-square-spindle-24000rpm-vfd-2025. [Acesso em: 5 set. 2025]. [Citado nas páginas xv, 87 e 88]
- [71] “Hf41f.” <https://docs.rs-online.com/dced/0900766b81663ec1.pdf>, 2017. [Acesso em: 5 set. 2025]. [Citado na página 88]
- [72] “41f-1z-c1-1 hongfa europe gmbh - rs components.” <https://pt.rs-online.com/web/p/zocalos-para-reles/9232613?gb=s>, 2025. [Acesso em: 03 jun. 2025]. [Citado nas páginas xv e 89]
- [73] “Autodesk | 3d design, engineering & construction software.” <https://www.autodesk.com/>, 2025. [Acesso em: 22 ago. 2025]. [Citado nas páginas xv, xvi, 93, 94 e 132]
- [74] “Bresimar automação, s.a. | soluções de automação industrial.” <https://bresimar.pt/>, 2025. [Acesso em: 16 mar. 2025]. [Citado na página 94]
- [75] “Norelem - peças e componentes padrão para engenharia mecânica.” <https://norelem.es/pt/>, 2025. [Acesso em: 6 jun. 2025]. [Citado na página 103]
- [76] “Text to g-code (linear path) [cnc-apps.com].” https://cnc-apps.com/en/app/texttogcode_line, 2025. [Acesso em: 20 jul. 2025]. [Citado na página 154]

Anexo A

Código G - Padrões

Neste anexo serão mencionados os códigos G e M e qual a sua função, de forma teórica. De referir ainda que, dependendo de máquina para máquina, os códigos podem ser diferentes, tal como alguns só servem para torneamento ou fresagem. Em geral, códigos mais frequentes costumam ser iguais entre fabricantes, podendo acrescentar alguns códigos para outras configurações, como introdução de sinais, ativar algum mecanismo extra, entre outros. Aconselha-se assim que ao usar uma máquina CNC, analisar com atenção as características do código G.

A.1 Código G

Na Tabela A.1 encontram-se os códigos G mais comuns em CNC [7, 15].

Tabela A.1: Código - G Comuns

<i>G-word</i>	Função
G00	Movimento ponto-a-ponto (avanço rápido) entre o ponto anterior e o ponto final definido no bloco atual. O bloco deve incluir as coordenadas x-y-z da posição final.
G01	Movimento por interpolação linear. O bloco deve incluir as coordenadas x-y-z da posição final. A <i>feed rate</i> também deve ser especificada.

G02	Interpolação circular no sentido horário. O bloco deve incluir o raio do arco ou o centro do arco; as coordenadas da posição final também devem ser especificadas.
G03	Interpolação circular no sentido anti-horário. O bloco deve incluir o raio do arco ou o centro do arco; as coordenadas da posição final também devem ser especificadas.
G04	Pausa por um tempo especificado.
G10	Entrada de dados de compensação da ferramenta, seguido por um código P e um código R.
G17	Seleção do plano x-y em fresagem.
G18	Seleção do plano x-z em fresagem.
G19	Seleção do plano y-z em fresagem.
G20	Entrada de valores especificados em polegadas.
G21	Entrada de valores especificados em milímetros.
G28	Retorno ao ponto de referência.
G32	Corte de rosca em torneamento.
G40	Cancelamento da compensação de raio da ferramenta (raio do nariz em torneamento).
G41	Compensação da ferramenta à esquerda da superfície da peça. O raio da ferramenta (ou do nariz em torneamento) deve ser especificado no bloco.
G42	Compensação da ferramenta à direita da superfície da peça. O raio da ferramenta (ou do nariz em torneamento) deve ser especificado no bloco.
G50	Especificar a localização da origem do sistema de coordenadas relativamente à posição inicial da ferramenta de corte. Usado em alguns tornos. Máquinas de fresagem e perfuração utilizam G92.
G90	Programação em coordenadas absolutas.
G91	Programação em coordenadas incrementais.
G92	Especificar a localização da origem do sistema de coordenadas relativamente à posição inicial da ferramenta de corte. Usado em fresagem e perfuração e em alguns tornos. Outros tornos utilizam G50.
G94	Especificar o avanço por minuto em fresagem e perfuração.
G95	Especificar o avanço por rotação em fresagem e perfuração.
G98	Especificar o avanço por minuto em torneamento.
G99	Especificar o avanço por rotação em torneamento.

A.2 Código M

Na Tabela A.2 encontram-se os códigos M mais comuns em CNC [7, 15].

Tabela A.2: Código - M Comuns

<i>M- Word</i>	Função
M00	Paragem do programa; usada no meio do programa. O operador deve reiniciar a máquina.
M01	Paragem opcional do programa; ativa somente quando o botão de paragem opcional no painel de controlo estiver pressionado.
M02	Fim do programa. Paragem da máquina.
M03	Iniciar o <i>spindle</i> no sentido horário para fresagem (avançar para torneamento).
M04	Iniciar o <i>spindle</i> no sentido anti-horário para fresagem (retroceder para torneamento).
M05	Parar o <i>spindle</i> .
M06	Executar a troca de ferramenta, manual ou automaticamente. Se manual, o operador deve reiniciar a máquina. Não inclui a seleção da ferramenta, feita pelo código T em automático ou pelo operador em manual.
M07	Ativar fluido de corte em modo <i>flood</i> (inundação).
M08	Ativar fluido de corte em modo <i>mist</i> (neblina).
M09	Desligar o fluido de corte.
M10	Fixação automática do dispositivo de aperto, guias da máquina, etc.
M11	Desbloqueio automático.
M13	Iniciar o <i>spindle</i> no sentido horário para fresagem (avançar para torneamento) e ligar o fluido de corte.
M14	Iniciar o <i>spindle</i> no sentido anti-horário para fresagem (retroceder para torneamento) e ligar o fluido de corte.
M17	Desligar o <i>spindle</i> e o fluido de corte.
M19	Parar o <i>spindle</i> numa posição orientada.
M30	Fim do programa. Paragem da máquina. Rebobinar a fita (em máquinas controladas por fita).

Anexo B

Esquema Elétrico

Neste anexo são apresentados todos os esquemas elétricos efetuados para este trabalho. Os esquemas estão divididos por várias páginas, sendo que cada um está organizado por grupos. De referir que o grupo dos *drivers* e motores, pelo circuito ser semelhante, só foi introduzido o primeiro motor.

B.1 Ligações de Energia

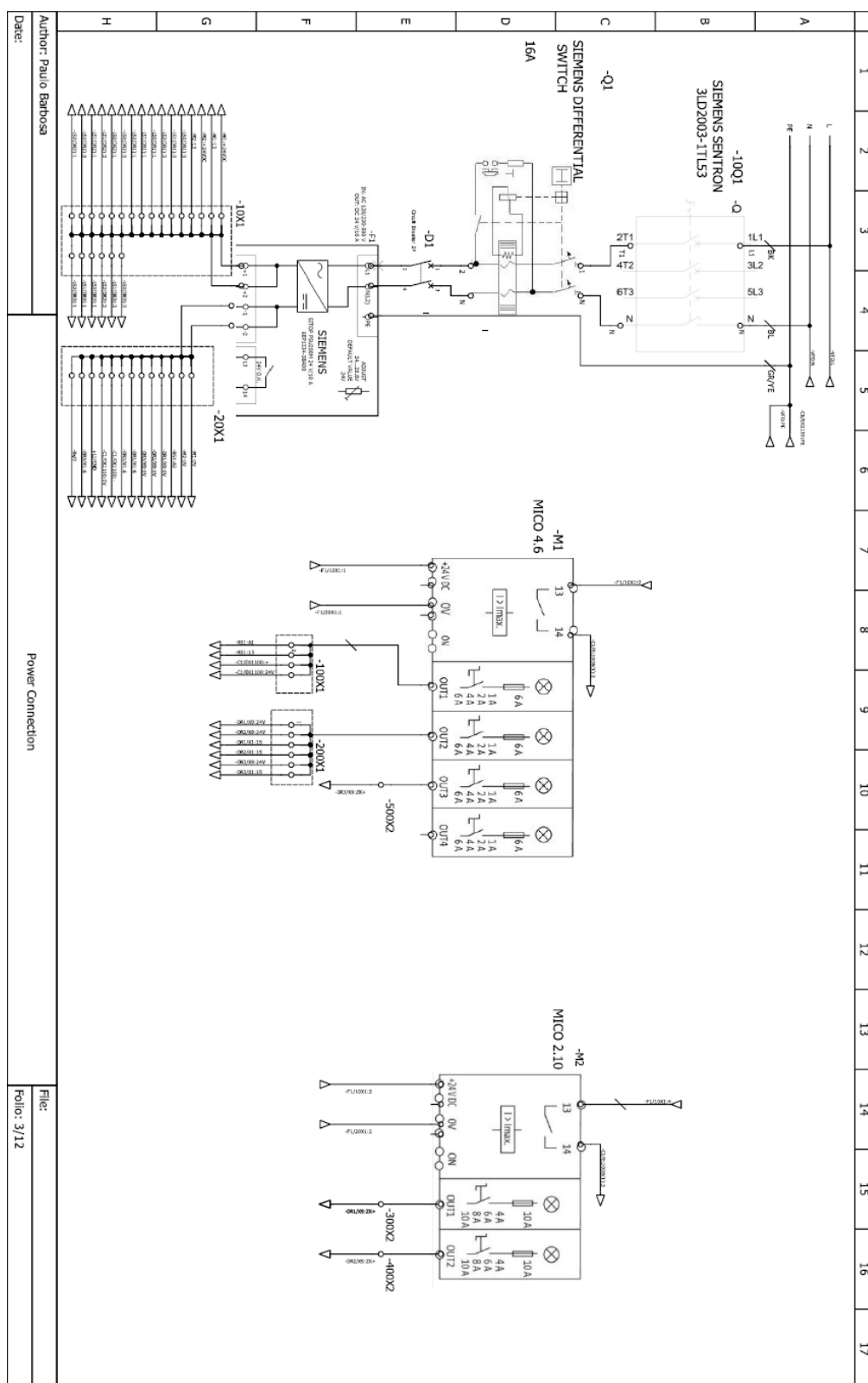


Figura B.1: Circuito de ligações de energia

B.2 Ligações do Relé de Emergência, Botões e Luzes

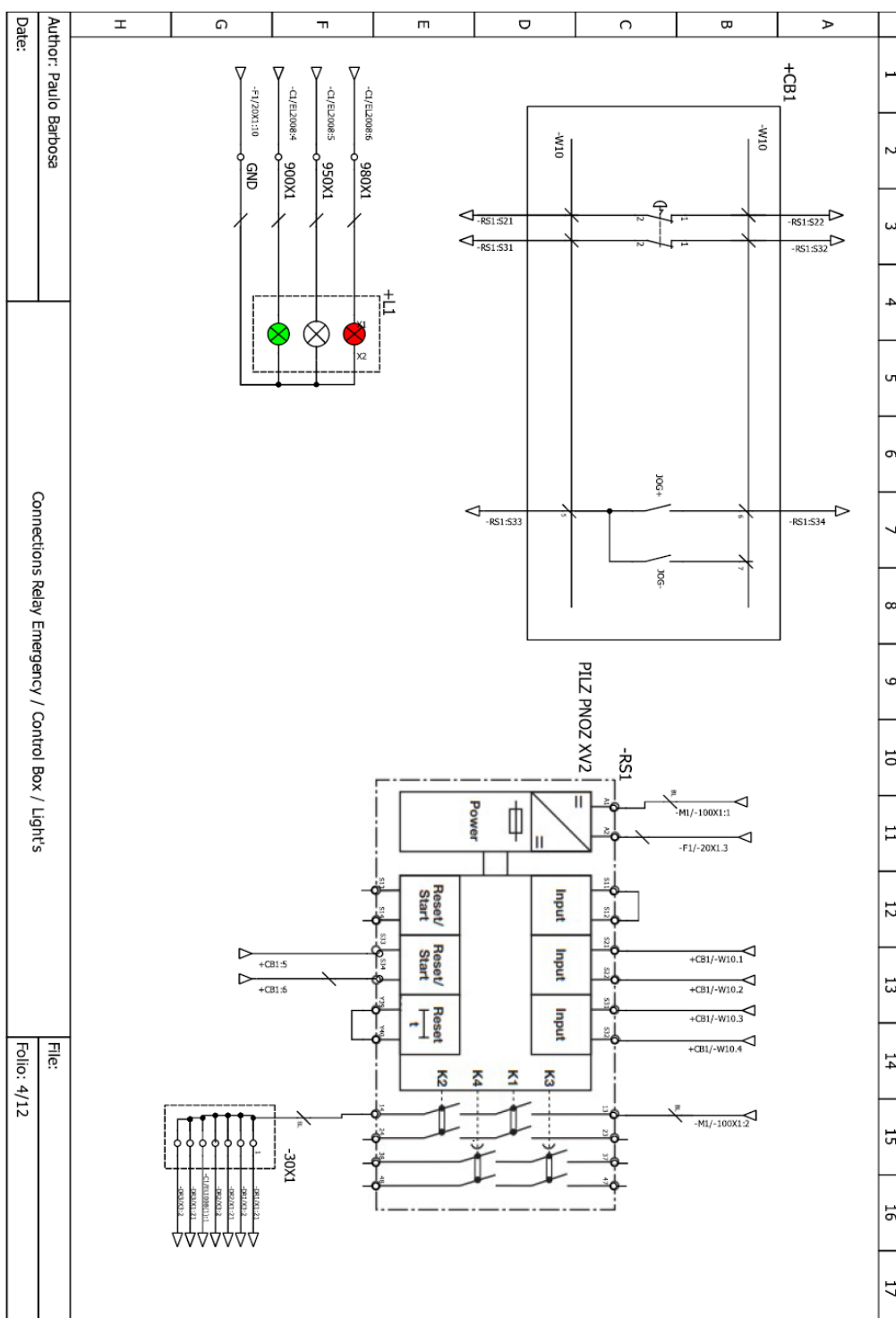


Figura B.2: Circuito de ligações do relé de emergência, botões e luzes

B.3 Ligações de Controlo

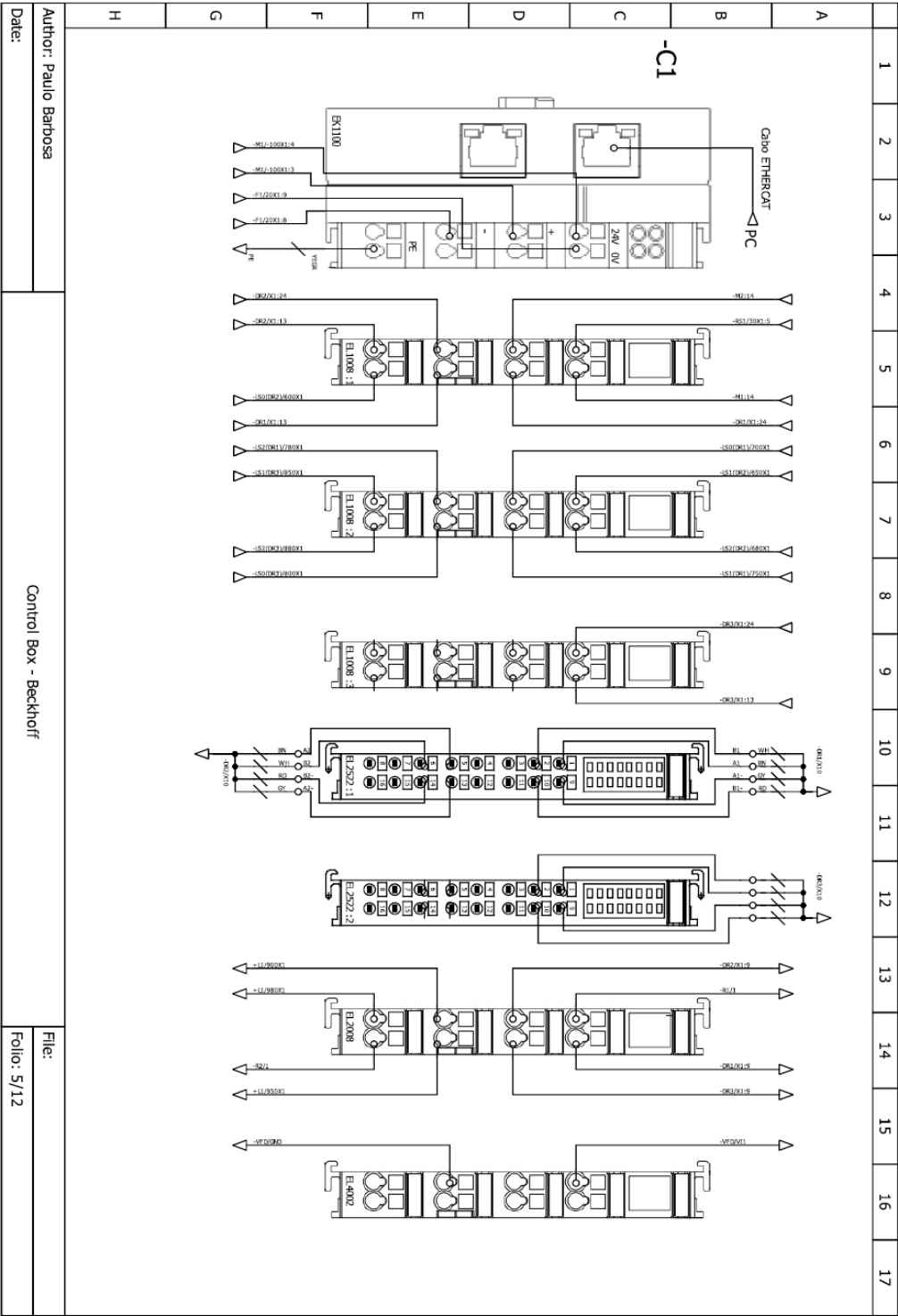


Figura B.3: Circuito de ligações do controlo

B.4 Ligações dos *Drivers*

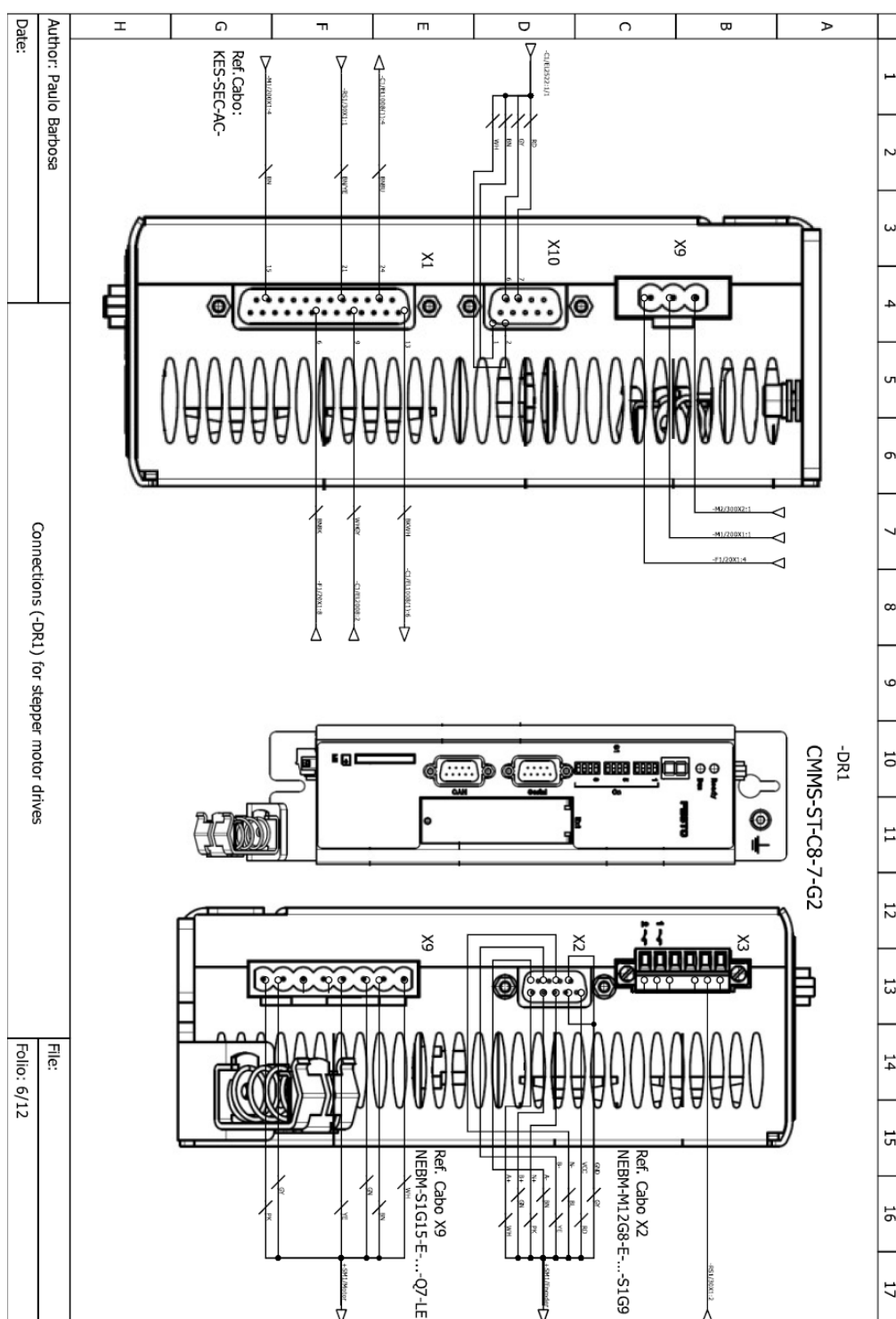


Figura B.4: Circuito de ligações dos *drivers*

B.5 Ligações dos Motores de Eixo

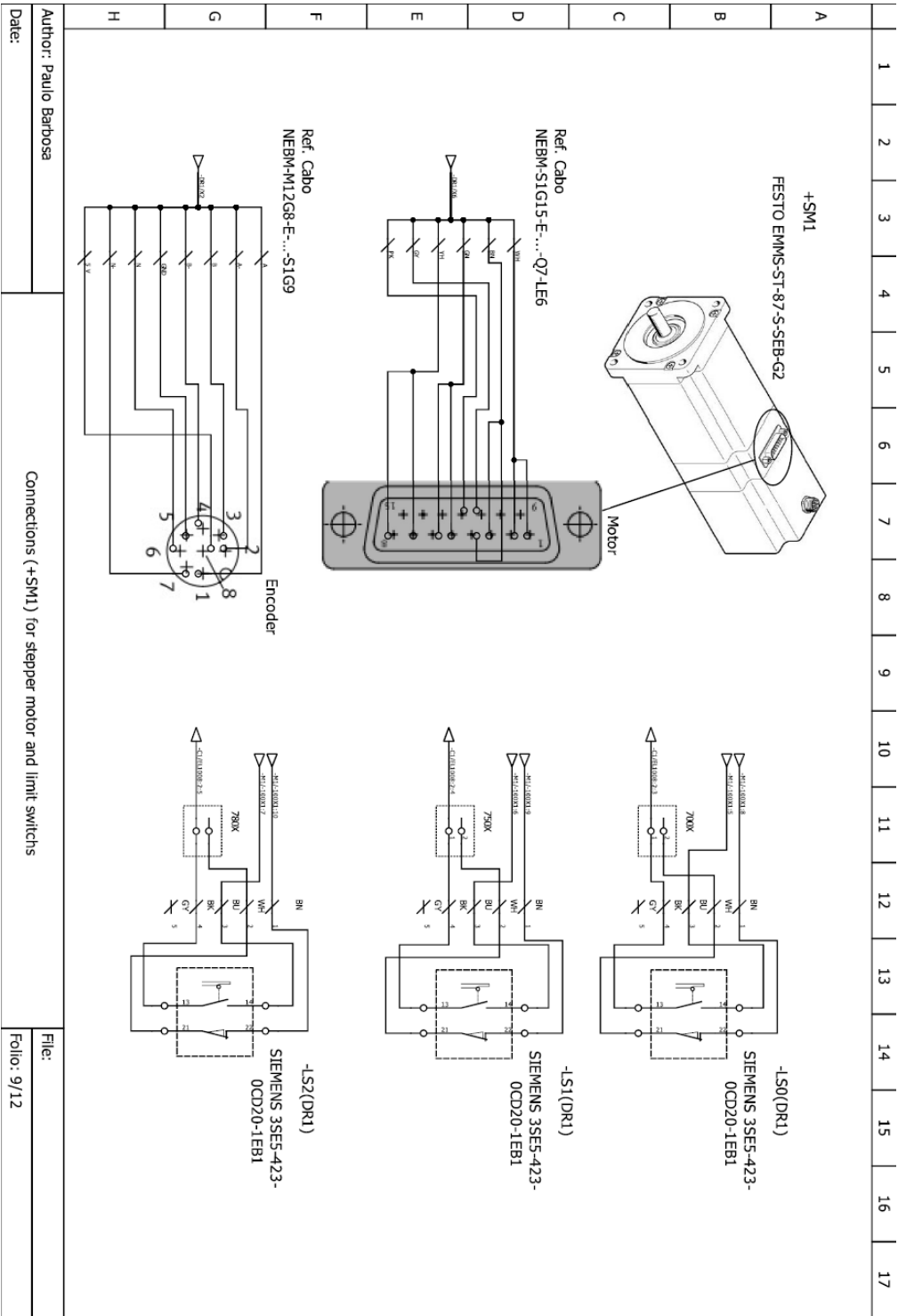


Figura B.5: Circuito de ligações dos motores do eixo

Anexo C

Códigos Efetuados

Neste anexo, serão introduzidos todos os códigos mencionados no Capítulo 5, na subseção referente a códigos introduzidos ou alterados.

C.1 Código PTP

C.1.1 Variáveis

```
1 PROGRAM MAIN
2 VAR
3     axis_X: AXIS_REF ;
4     axis_Y: AXIS_REF ;
5     fbAxisXPower: MC_Power;
6     fbAxisYPower: MC_Power;
7     fbAxisXMoveAbsolute: MC_MoveAbsolute;
8     fbAxisYMoveAbsolute: MC_MoveAbsolute;
9     bEnable: BOOL := FALSE;
10    fOverride: LREAL := 100;
11    bMoveX: BOOL := FALSE;
12    bMoveY: BOOL := FALSE;
13    bPulseRele: BOOL := FALSE; //Ativar este para ligar o Rele
14    bRele: BOOL := FALSE;
15    fTargetPositionX: LREAL := 10; // 1030 mm max
16    fTargetPositionY: LREAL := 200; // 2030 mm max
17    fTargetVelocity: LREAL := 100;
```

```

18
19     bBoolSemaforo_Blink: BOOL;
20
21 END_VAR
22 VAR_INPUT
23     bReleInput AT %I* : BOOL;
24     bMICOInput AT %I* : ARRAY[1..2] OF BOOL;
25     bContDriveAptoInput AT %I* : ARRAY[1..2] OF BOOL;
26     bContDriveErroInput AT %I* : ARRAY[1..2] OF BOOL;
27     bFimdeCursoInput AT %I* : ARRAY[1..6] OF BOOL;
28 END_VAR
29 VAR_OUTPUT
30     bReleStart AT %Q* : BOOL;
31     bControlador AT %Q* : ARRAY[1..2] OF BOOL;
32
33     bSemaforo AT %Q* : ARRAY[1..3] OF BOOL; //1-Verde 2- Amarelo 3 -
        Vermelho
34 END_VAR

```

Listagem C.1: Declaração das variáveis no código PTP

C.1.2 Código

```

1  axis_X.ReadStatus();
2  axis_Y.ReadStatus();
3
4  //Rele
5  IF bPulseRele AND NOT bRele THEN //Quando recebe impulso e rele
        desligado
6      bRele := TRUE;
7      bReleStart := TRUE;
8      bPulseRele := FALSE;
9  ELSIF bRele AND NOT bReleInput THEN //Quando o Rele desliga-se por
        paragem de emergencia
10     bRele := FALSE;
11     bReleStart := FALSE;
12     bPulseRele := FALSE;
13     bEnable := FALSE;
14 ELSE
15     bPulseRele := FALSE;
16     bReleStart := FALSE;
17 END_IF
18
19 //Teste enable
20 IF bReleInput AND bMICOInput[1] AND bMICOInput[2] THEN //Testa se
        o Rele e os MICO estao disponiveis
21

```

```
22  IF bEnable THEN                                //Quando estiver ENABLE
23      bControlador[1] := TRUE;
24      bControlador[2] := TRUE;
25
26      fbAxisXPower(
27          Axis:= axis_X,
28          Enable:= bEnable,
29          Enable_Positive:= bEnable,
30          Enable_Negative:= bEnable,
31          Override:= fOverride );
32
33      fbAxisYPower(
34          Axis:= axis_Y,
35          Enable:= bEnable,
36          Enable_Positive:= bEnable,
37          Enable_Negative:= bEnable,
38          Override:= fOverride
39      );
40  ELSE //Se desligar enable
41      bControlador[1] := FALSE;
42      bControlador[2] := FALSE;
43      bEnable := FALSE;
44
45      fbAxisXPower(
46          Axis:= axis_X,
47          Enable:= FALSE);
48
49      fbAxisYPower(
50          Axis:= axis_Y,
51          Enable:= FALSE);
52  END_IF
53  ELSE
54      bControlador[1] := FALSE;
55      bControlador[2] := FALSE;
56      bEnable := FALSE;
57
58      fbAxisXPower(
59          Axis:= axis_X,
60          Enable:= FALSE);
61
62      fbAxisYPower(
63          Axis:= axis_Y,
64          Enable:= FALSE);
65  END_IF
66
67  IF bEnable THEN
68      IF bContDriveAptoInput[1] AND bMoveX THEN
69          fbAxisXMoveAbsolute(
70              Axis:= axis_X,
```



```

71         Execute:= bMoveX,
72         Position:= fTargetPositionX,
73         Velocity:= fTargetVelocity);
74
75     bMoveX := FALSE;
76 ELSE
77     bMoveX := FALSE;
78     fbAxisXMoveAbsolute(
79         Axis:= axis_X,
80         Execute:= FALSE);
81 END_IF
82
83 IF bContDriveAptoInput[2] AND bMoveY THEN
84     fbAxisYMoveAbsolute(
85         Axis:= axis_Y,
86         Execute:= bMoveY,
87         Position:= fTargetPositionY,
88         Velocity:= fTargetVelocity);
89
90     bMoveY := FALSE;
91 ELSE
92     bMoveY := FALSE;
93     fbAxisYMoveAbsolute(
94         Axis:= axis_Y,
95         Execute:= FALSE);
96 END_IF
97 END_IF

```

Listagem C.2: Função principal do código PTP

C.2 Variáveis Globais do CNC

C.2.1 Global_HMI

```

1  VAR_GLOBAL
2      // HMI -> PLC
3      PLCChannelSignals: ST_ChannelSignals;
4      // HMI -> PLC
5      PLCSelectedChannel: INT := 0;
6      // HMI -> PLC
7      PLCFeedOverride: UINT := 100;
8      // HMI -> PLC
9      PLCSpindleOverride: UINT := 100;
10     // HMI -> PLC
11     PLCAxisEnable: BOOL;
12     // PLC -> HMI

```

```

13     PLCStatusAxisEnabled : BOOL;
14     // HMI -> PLC
15     PLCReset: BOOL;
16     PLCMachineMode: ARRAY[0..MAX_USED_CHANNEL_IDX] OF
        ST_MachineMode;
17     // PLC -> HMI : angewahltes File
18     PLCToHmiFile: STRING(255);
19     // Variablen fur das Module "InformPlc"
20     bToggle: BOOL;
21     sActualForm: STRING(255);
22     nUserLevel: INT;
23     sUserName: STRING(255);
24     // steigende Flanke: Applikation beenden
25     bExitApp: BOOL;
26
27     userActivity : BOOL;
28
29     // @1:MachineData
30     MachineData : ST_MachineData;
31
32     // spindle manual mode
33     nSpindleRpm: DINT;
34     bSpindleStart: BOOL;
35     bSpindleStop: BOOL;
36
37     ManualFnc : ARRAY[0..3] OF FB_ManualFncFull;
38
39     //Variavel Extra - Verificar se foi feito o Homing
40     PLCAxisHoming_Done: BOOL;
41     PLC_Homing_Simulation : BOOL := FALSE; //Quando em simulacao,
        forcar
42
43     PLC_Force_Homing: BOOL := FALSE;
44     bEmergencyStop: BOOL;
45
46     // Teste botoes
47     PLC_StartFunction: BOOL;
48     PLC_StopFunction: BOOL;
49     PLC_ResetFunction: BOOL;
50 END_VAR

```

Listagem C.3: Declaração das variáveis globais Global_HMI

C.2.2 GVL_Input_Output

```

1  //{attribute 'qualified_only'}
2  VAR_GLOBAL

```

```

3  //INPUT's
4  bReleStatus      : BOOL;
5  bMICOStatus      : ARRAY[1..2] OF BOOL; //Mico[1] => MICO 4
        available // Mico[2] => Mico 2 available
6
7  bDriveMotorAxisStatus : ARRAY[1..3] OF BOOL; //Eixo Z simulado
        por enquanto
8  bDriveMotorAxisError  : ARRAY[1..3] OF BOOL; //High quando se
        encontra sem erros
9
10 bLimitSwitch_AxisX    : ARRAY[1..3] OF BOOL;
11 bLimitSwitch_AxisY    : ARRAY[1..3] OF BOOL;
12 bLimitSwitch_AxisZ    : ARRAY[1..3] OF BOOL;
13
14 bCam_Switch          : ARRAY[0..2] OF BOOL;
15
16 //Output's
17 bReleStart           AT %Q* : BOOL;
18 bDriveMotorAxisEnable AT %Q* : ARRAY[1..3] OF BOOL; //Eixo Z
        simulado por enquanto
19
20 bLightTower          AT %Q* : ARRAY[1..3] OF BOOL; //[1] Green /
        [2] Yellow / [3] Red
21
22 //Outras variaveis
23 /// Botao de Emergencia (Fisico)
24 bEmergencyStop_Activate: BOOL;
25 //Enable de tudo OK (O outro Enable so ativa este)
26 bMotorAvailable : BOOL;
27
28 bLimitStop           : BOOL; //ON quando esta do limite
29 bStop_Limit_SW       : BOOL;
30 bReset_Limit_SW      : BOOL;
31
32 //Axis and direction limit
33 nAxis_Jog_Selected   : UINT;
34 bJog_Pos             : BOOL;
35 bJog_Neg             : BOOL;
36 State_Mode           : ST_CNCCmdModesStates;
37 END_VAR

```

Listagem C.4: Declaração das variáveis globais GVL_Input_Output

C.3 PRG_Write_INPUT

C.3.1 Variáveis

```

1 PROGRAM PRG_Write_INPUT
2 VAR_INPUT
3     iReleStatus      AT %I* : BOOL;
4     iMICOSStatus     AT %I* : ARRAY[1..2] OF BOOL; //Mico[1] =>
        MICO 4 available // Mico[2] => Mico 2 available
5
6     iDriveMotorAxisStatus AT %I* : ARRAY[1..3] OF BOOL; //Eixo Z
        simulado por enquanto
7     iDriveMotorAxisError AT %I* : ARRAY[1..3] OF BOOL; //High
        quando se encontra sem erros
8
9     iLimitSwitch_AxisX   AT %I* : ARRAY[1..3] OF BOOL;    //Swich
        trocados NEG = 1 POS = 3
10    iLimitSwitch_AxisY   AT %I* : ARRAY[1..3] OF BOOL; // NEG = 3
        POS = 1
11    iLimitSwitch_AxisZ   AT %I* : ARRAY[1..3] OF BOOL := [FALSE,
        TRUE,FALSE];
12 END_VAR

```

Listagem C.5: Declaração de variáveis da função
PRG_Write_INPUT

C.3.2 Código

```

1 bReleStatus := iReleStatus;
2 bMICOSStatus := iMICOSStatus;
3 bDriveMotorAxisStatus := iDriveMotorAxisStatus;
4 bDriveMotorAxisError := iDriveMotorAxisError;
5 bLimitSwitch_AxisX := iLimitSwitch_AxisX;
6 bLimitSwitch_AxisY := iLimitSwitch_AxisY;
7 //bLimitSwitch_AxisZ := iLimitSwitch_AxisZ;    //Simulacao
8
9 bCam_Switch[0] := iLimitSwitch_AxisX[2];
10 bCam_Switch[1] := iLimitSwitch_AxisY[2];
11 bCam_Switch[2] := iLimitSwitch_AxisZ[2];
12
13 PRG_Control_Light_Mode();

```

Listagem C.6: Função PRG_Write_INPUT

C.4 PRG_Light_Tower

C.4.1 Variáveis

```

1 PROGRAM PRG_Light_Tower
2 VAR
3   eTypeLight : (RED_FIX, RED_BLINK, YELLOW_FIX, YELLOW_BLINK,
4                 GREEN_FIX, GREEN_BLINK);
5   bBlink_light : BOOL := FALSE;
6   fb_Blink : FB_HVACBlink := (bEnable := TRUE,
7                                tTimeOff:= T#500MS,
8                                tTimeOn := T#500MS);
9 END_VAR
10
11 VAR_INPUT
12   Colour : INT; //0 to red, 1 to yellow and 2 to green
13   Blink : BOOL; //FALSE for FIX light and TRUE for blink light
14 END_VAR

```

Listagem C.7: Declaração de variáveis da função PRG_Light_Tower

C.4.2 Código

```

1 fb_Blink(bBlink => bBlink_light);
2
3 //Estado
4 IF Blink THEN //Pisca
5   CASE Colour OF
6     0:
7       eTypeLight := RED_BLINK;
8     1:
9       eTypeLight := YELLOW_BLINK;
10    2:
11      eTypeLight := GREEN_BLINK;
12    END_CASE;
13 ELSE //Fixo
14   CASE Colour OF
15     0:
16       eTypeLight := RED_FIX;
17     1:
18       eTypeLight := YELLOW_FIX;
19     2:
20       eTypeLight := GREEN_FIX;
21    END_CASE;
22 END_IF
23 //Output
24 CASE eTypeLight OF
25   RED_FIX: // Emergency Stop Activated // Start Inicialization
26     bLightTower[1] := FALSE;
27     bLightTower[2] := FALSE;

```

```

28         bLightTower[3] := TRUE; //Vermelho ligado
29     RED_BLINK: // Error
30         bLightTower[1] := FALSE; //luzes nao necessarias
           desligados
31         bLightTower[2] := FALSE;
32         bLightTower[3] := bBlink_light;
33     YELLOW_FIX: // Enable or Controller don't available
34         bLightTower[1] := FALSE;
35         bLightTower[2] := TRUE;
36         bLightTower[3] := FALSE;
37     YELLOW_BLINK: // Operation, Warning for movement's
38         bLightTower[1] := FALSE;
39         bLightTower[3] := FALSE;
40         bLightTower[2] := bBlink_light;
41     GREEN_FIX: //Available
42         bLightTower[1] := TRUE;
43         bLightTower[2] := FALSE;
44         bLightTower[3] := FALSE;
45     GREEN_BLINK: //Operate in command
46         bLightTower[2] := FALSE;
47         bLightTower[3] := FALSE;
48         bLightTower[1] := bBlink_light;
49 END_CASE;

```

Listagem C.8: Função PRG_Light_Tower

C.5 PRG_Control_Light_Mode

C.5.1 Variáveis

```

1 PROGRAM PRG_Control_Light_Mode
2 VAR
3     bState_Finish_Auto : (NOT_ACTIVE, AUTO_JOB, FINISH, FINISH_OFF);
4     fbDelay_Message_Output : TON := (PT := T#10S);
5
6     fb_alarm : FB_Alarm;
7     fb_alarm_finish : FB_Alarm;
8
9     bRun_Code: BOOL := FALSE;
10    bFirstMessage : BOOL := FALSE;
11 END_VAR

```

Listagem C.9: Declaração de variáveis da função PRG_Control_Light_Mode

C.5.2 Código

```

1 IF bMotorAvailable THEN
2   IF gpCh[0]^channel_mc_control.mode_and_state.command_w.to_mode
      = 1 OR
3     (gpCh[0]^channel_mc_control.mode_and_state.command_w.
      to_mode = 2 AND
4     bProgram_End ) THEN
5     //Case stand-by / Auto (Standby)
6     PRG_Light_Tower(Colour := 2 ,
7       Blink      := FALSE);
8   END_IF
9
10  // Case Auto (start) / Homing
11  IF (gpCh[0]^channel_mc_control.mode_and_state.command_w.to_mode
      = 2 AND
12    ( gpCh[0]^channel_mc_control.mode_and_state.command_w.
      to_state = 4 OR
13    gpCh[0]^channel_mc_control.mode_and_state.command_w.
      to_state = 5 ) AND
14    NOT bLimitStop AND NOT bProgram_End) THEN
15    PRG_Light_Tower(Colour := 1 ,
16      Blink      := TRUE);
17  END_IF
18
19  // Case MDI / Jog /
20  IF gpCh[0]^channel_mc_control.mode_and_state.command_w.to_mode
      = 3 OR
21    gpCh[0]^channel_mc_control.mode_and_state.command_w.to_mode
      = 4 THEN
22    PRG_Light_Tower(Colour := 2 ,
23      Blink      := TRUE);
24  END_IF
25
26  //Error
27  IF gpCh[0]^channel_mc_control.mode_and_state.command_w.to_state
      = 6 THEN
28    PRG_Light_Tower(Colour := 0 ,
29      Blink      := TRUE);
30    PLCAxisEnable := FALSE;
31  END_IF
32
33  IF bLimitStop THEN
34    PRG_Light_Tower(Colour := 0,
35      Blink := TRUE);
36  END_IF
37
38  ////Alarms

```

```
39  // Finish g-code      //Maquina de Estados !!
40  IF NOT gpCh[0]^channel_mc_control.mode_and_state.command_w.
    to_mode = 2 THEN //Se nao estiver em Auto
41      bState_Finish_Auto := NOT_ACTIVE;
42  END_IF
43
44  CASE bState_Finish_Auto OF
45      NOT_ACTIVE:
46          //Se estiver em auto e iniciar (4) e se nao estiver codigo
            pronto
47          IF gpCh[0]^channel_mc_control.mode_and_state.command_w.
            to_mode = 2 AND
48              gpCh[0]^channel_mc_control.mode_and_state.command_w.
                to_state = 4 AND
49              NOT bProgram_End THEN
50
51              bState_Finish_Auto := AUTO_JOB;
52              bRun_Code := TRUE;
53          ELSE
54              bState_Finish_Auto := NOT_ACTIVE;
55              bRun_Code := FALSE;
56          END_IF
57
58      AUTO_JOB:
59          //Se registrar que acabou o codigo e manteve da posicao 4 ou
            2
60          IF gpCh[0]^channel_mc_control.mode_and_state.command_w.
            to_mode = 2 AND bProgram_End AND
61              (gpCh[0]^channel_mc_control.mode_and_state.command_w.
                to_state >= 2) AND
62              (gpCh[0]^channel_mc_control.mode_and_state.command_w.
                to_state >= 4) THEN
63              bState_Finish_Auto := FINISH;
64
65          //Se houve reset
66          ELSIF NOT gpCh[0]^channel_mc_control.mode_and_state.
            command_w.to_mode = 2 OR
67              gpCh[0]^channel_mc_control.mode_and_state.command_w.
                to_state = 15 THEN
68              bState_Finish_Auto := NOT_ACTIVE;
69              bRun_Code := FALSE;
70          ELSE
71              bState_Finish_Auto := AUTO_JOB;
72          END_IF
73
74      FINISH:
75          fb_Alarm_finish(Event := TC_EVENTS.Alarm_Create_Gislotica.
            Complete_Program,
76              Raise := TRUE);
```



```

77     fbDelay_Message_Output (IN := TRUE);
78     IF fbDelay_Message_Output.Q THEN
79         bState_Finish_Auto := FINISH_OFF;
80         bRun_Code := FALSE;
81     END_IF
82
83     FINISH_OFF:
84         fb_Alarm_finish(Event := TC_EVENTS.Alarm_Create_Gislotica.
            Complete_Program,
85             Raise := FALSE);
86         fbDelay_Message_Output (IN := FALSE);
87         bState_Finish_Auto := NOT_ACTIVE;
88     END_CASE
89
90     // Stop activated
91     IF gpCh[0]^channel_mc_control.mode_and_state.command_w.to_state
        = 5 THEN
92         IF NOT bFirstMessage THEN
93             fb_Alarm(Event := TC_EVENTS.Alarm_Create_Gislotica.
                Stop_Action,
94                 Raise := TRUE);
95             bFirstMessage := TRUE;
96         END_IF
97
98     ELSE
99         fb_Alarm(Event := TC_EVENTS.Alarm_Create_Gislotica.Stop_Action
            ,
100             Raise := FALSE);
101         bFirstMessage := FALSE;
102     END_IF
103 END_IF

```

Listagem C.10: Função PRG_Control_Light_Mode

C.6 FB_Limit_Switch

C.6.1 Variáveis

```

1  FUNCTION_BLOCK FB_Limit_Switch
2  VAR
3      nState_Limit : (NORMAL, LIMIT_DETECTED_TRIGGER_STOP, IN_LIMITED)
4      ;
5
6      fb_Timer_delay : TON := (PT := T#500MS);
7      fb_Timer_motor_off : TON := (PT := T#3S);
8      bfisrt_entry : BOOL;

```

```

8
9     bPersistant_Error : BOOL;
10
11     fb_alarm : FB_Alarm;
12     fb_alarm_1 : FB_Alarm;
13
14     sString_Alarm : STRING := 'TESTE';
15
16     //Register switch parameter
17     bSW_Neg_X      : BOOL;
18     bSW_Pos_X      : BOOL;
19     bSW_Neg_Y      : BOOL;
20     bSW_Pos_Y      : BOOL;
21     bSW_Neg_Z      : BOOL;
22     bSW_Pos_Z      : BOOL;
23     nNumber_Limit  : INT := 0;
24     nNumber_Limit_Last : INT := 0;
25 END_VAR

```

Listagem C.11: Declaração de variáveis da função *FB_Limit_Switch*

C.6.2 Código

```

1
2 CASE nState_Limit OF
3     NORMAL:
4         //Normal, atento aos fim de cursos, ativando quando o deteta
5         IF (bLimitSwitch_AxisX[1] OR bLimitSwitch_AxisX[3] OR
6             bLimitSwitch_AxisY[1] OR bLimitSwitch_AxisY[3] OR
7             bLimitSwitch_AxisZ[1] OR bLimitSwitch_AxisZ[3]) AND
8             (NOT State_Mode.stModes.Standby AND bMotorAvailable)
9             THEN
10                //Se algum detetar, se estiver em standby nao efetua
11                nState_Limit := LIMIT_DETECTED_TRIGGER_STOP;
12                bLimitStop := TRUE;
13                bStop_Limit_SW := TRUE;
14                bfisrt_entry := TRUE;
15            ELSE
16                nState_Limit := NORMAL;
17                bLimitStop := FALSE;
18                bStop_Limit_SW := FALSE;
19                bReset_Limit_SW := FALSE;
20                Write_Consol_Error(); //Clear error
21            END_IF
22
23 LIMIT_DETECTED_TRIGGER_STOP:
24     //STOP, regista qual foi o sacana, e faz delay de 1 segundo.

```

```

24      // Ativa flag de erro (esta so desativa apos estar fora do fim
        de curso
25      bStop_Limit_SW := FALSE;
26      fb_Timer_delay(IN := TRUE);
27      IF fb_Timer_delay.Q THEN
28          nState_Limit := IN_LIMITED;
29          bLimitStop := TRUE;
30          IF bfisrt_entry THEN
31              bReset_Limit_SW := TRUE;
32              bfisrt_entry := FALSE;
33              bPersistant_Error := FALSE;
34          END_IF
35          //one time
36          Write_Switch();           //Leitura
37          Write_Consol_Error();    //Error Msg
38          fb_Timer_delay(IN := FALSE);
39      END_IF
40
41      IN_LIMITED:
42          bReset_Limit_SW := FALSE;
43          //Nova Leitura
44          Write_Switch();
45
46          //Se aumentar os switchs selecionados, retorna um stop
47          IF nNumber_Limit > nNumber_Limit_Last THEN
48              nState_Limit := LIMIT_DETECTED_TRIGGER_STOP;
49              bLimitStop := TRUE;
50              bStop_Limit_SW := TRUE;
51
52          ELSE
53              IF bJog_Neg OR bJog_Pos THEN
54                  IF nNumber_Limit = 0 THEN //Free
55                      nState_Limit := NORMAL;
56                      bLimitStop := FALSE;
57                  END_IF
58              END_IF
59              Jog_Condiction(); //Analisa mesmo com jog ou nao,
                necessita para destravar o persistent
60      END_IF
61  END_CASE

```

Listagem C.12: Função FB_Limit_Switch

C.6.3 Write_Switch

```

1  nNumber_Limit_Last := nNumber_Limit; //Put last valor
2  nNumber_Limit := 0; //Clear

```

```
3
4  //A contagem e so para evitar um novo eixo parado, se isso
   acontecer, acontece um novo STOP
5  IF bLimitSwitch_AxisX[1] THEN
6      nNumber_Limit := nNumber_Limit + 1;
7      bSW_Neg_X := TRUE;
8  ELSE
9      bSW_Neg_X := FALSE;
10 END_IF
11
12 IF bLimitSwitch_AxisX[3] THEN
13     nNumber_Limit := nNumber_Limit + 1;
14     bSW_Pos_X := TRUE;
15 ELSE
16     bSW_Pos_X := FALSE;
17 END_IF
18
19 //Eixo Y
20 IF bLimitSwitch_AxisY[1] THEN
21     nNumber_Limit := nNumber_Limit + 1;
22     bSW_Pos_Y := TRUE;
23 ELSE
24     bSW_Pos_Y := FALSE;
25 END_IF
26
27 IF bLimitSwitch_AxisY[3] THEN
28     nNumber_Limit := nNumber_Limit + 1;
29     bSW_Neg_Y := TRUE;
30 ELSE
31     bSW_Neg_Y := FALSE;
32 END_IF
33
34 //Eixo Z
35
36 IF bLimitSwitch_AxisZ[1] THEN
37     nNumber_Limit := nNumber_Limit + 1;
38     bSW_Neg_Z := TRUE;
39 ELSE
40     bSW_Neg_Z := FALSE;
41 END_IF
42
43 IF bLimitSwitch_AxisZ[3] THEN
44     nNumber_Limit := nNumber_Limit + 1;
45     bSW_Pos_Z := TRUE;
46 ELSE
47     bSW_Pos_Z := FALSE;
48 END_IF
```

C.6.4 Jog_Condition

```

1  //EIXO X  S1 = POS  S3 = NEG
2  IF nAxis_Jog_Selected = 1 THEN
3      //Jog +
4      IF bJog_Pos THEN
5          IF bSW_Pos_X THEN
6              //STOP
7              bPersistant_Error := TRUE
8          END_IF
9      END_IF
10
11     //Jog -
12     IF bJog_Neg THEN
13         IF bSW_Neg_X THEN
14             //STOP
15             bPersistant_Error := TRUE;
16         END_IF
17     END_IF
18 END_IF
19
20 //EIXO Y  S1 = NEG  S3 = POS
21 IF nAxis_Jog_Selected = 2 THEN
22     //Jog +
23     IF bJog_Pos THEN
24         IF bSW_Pos_Y THEN
25             //STOP
26             bPersistant_Error := TRUE;
27         END_IF
28     END_IF
29     //Jog -
30     IF bJog_Neg THEN
31         IF bSW_Neg_Y THEN
32             //STOP
33             bPersistant_Error := TRUE;
34         END_IF
35     END_IF
36 END_IF
37
38 IF nAxis_Jog_Selected = 3 THEN
39     //Jog +
40     IF bJog_Pos THEN
41         IF bSW_Pos_Z THEN
42             //STOP
43             bPersistant_Error := TRUE;
44         END_IF
45     END_IF
46     //Jog -

```

```

47     IF bJog_Neg THEN
48         IF bSW_Neg_Z THEN
49             //STOP
50             bPersistant_Error := TRUE;
51         END_IF
52     END_IF
53 END_IF
54
55 //Erro persistente
56 IF bPersistant_Error THEN
57     bLimitStop := TRUE;
58     bStop_Limit_SW := TRUE;
59     fb_Timer_motor_off(IN:=TRUE);
60     IF fb_Timer_motor_off.Q THEN
61         nState_Limit := LIMIT_DETECTED_TRIGGER_STOP;
62         bStop_Limit_SW := FALSE;
63         bReset_Limit_SW := TRUE;
64         bPersistant_Error := FALSE;
65         fb_Timer_motor_off(IN:=FALSE);
66     END_IF
67 END_IF

```

Listagem C.14: Função Jog_Condition

C.7 PRG_ControlFunction

C.7.1 Variáveis

```

1 PROGRAM PRG_ControlFunction
2 VAR
3     nState_Motors : (EMERGENCY, OFF_WAIT_ENABLE, ON_TRIGGER_MOTOR,
4                     ON_MOTOR);
5     nState_Enable_Stop : (ON, OFF_TRIG, OFF);
6     bAvailableEnable : BOOL := FALSE;
7     delay : TON := (PT := T#500MS);
8 END_VAR

```

Listagem C.15: Declaração de variáveis da função PRG_ControlFunction

C.7.2 Código

```

1 CASE nState_Motors OF

```

```

2      EMERGENCY: //Enable ON
3          //Motors OFF
4          bDriveMotorAxisEnable[1] := FALSE;
5          bDriveMotorAxisEnable[2] := FALSE;
6          bDriveMotorAxisEnable[3] := FALSE;
7          bMotorAvailable := FALSE;
8          PLCAxisEnable := FALSE;
9
10         //Light
11         PRG_Light_Tower(Colour := 0 ,
12                         Blink   := FALSE);
13         //Change parameter
14         IF bReleStatus THEN
15             nState_Motors := OFF_WAIT_ENABLE;
16         ELSE
17             nState_Motors := EMERGENCY;
18         END_IF
19
20     OFF_WAIT_ENABLE: //Enable OFF_Trig
21         //Motors OFF
22         bDriveMotorAxisEnable[1] := FALSE;
23         bDriveMotorAxisEnable[2] := FALSE;
24         bDriveMotorAxisEnable[3] := FALSE;
25
26         //parameters
27         bEmergencyStop_Activate := FALSE;
28         bAvailableEnable := TRUE;
29         bMotorAvailable := FALSE;
30
31         //Light
32         PRG_Light_Tower(Colour := 1 ,
33                         Blink   := FALSE);
34         //CASE EMERGENCY
35         IF NOT bReleStatus THEN //Se Rele esta OFF
36             bEmergencyStop_Activate := TRUE;
37             PLCAxisEnable := FALSE;
38             bAvailableEnable := FALSE;
39             PLCAxisEnable := FALSE;
40             nState_Motors := EMERGENCY;
41         ELSE
42             IF PLCAxisEnable AND bMICOStatus[1] AND bMICOStatus[2]
43                 THEN //test if MICO available and click AxisEnable
44                 nState_Motors := ON_TRIGGER_MOTOR;
45             ELSE
46                 nState_Motors := OFF_WAIT_ENABLE;
47                 PLCAxisEnable := FALSE;
48             END_IF
49         END_IF

```

```

50     ON_TRIGGER_MOTOR:
51         //Active Enable and wait //Yellow light
52         //Motors ON
53         bDriveMotorAxisEnable[1] := TRUE;
54         bDriveMotorAxisEnable[2] := TRUE;
55         bDriveMotorAxisEnable[3] := TRUE;
56         //parameters
57         bEmergencyStop_Activate := FALSE;
58         bAvailableEnable        := TRUE;
59         bMotorAvailable         := FALSE;
60         //Light
61         PRG_Light_Tower(Colour  := 1 ,
62                        Blink    := FALSE);
63         delay(IN := TRUE);
64         //Change parameters
65         IF delay.Q THEN
66             nState_Motors := ON_MOTOR;
67             delay(IN := FALSE);
68         END_IF
69
70     ON_MOTOR:
71         //In parameters OK, run //Drive 3 comment for don't
           running now
72         //CASE EMERGENCY
73         IF NOT bReleStatus THEN //Se Rele esta OFF
74             bEmergencyStop_Activate := TRUE;
75             PLCAxisEnable            := FALSE;
76             bAvailableEnable         := FALSE;      nState_Motors :=
           EMERGENCY;
77         ELSE
78             IF bMICOStatus[1] AND bMICOStatus[2] AND PLCAxisEnable
           AND//Test independent
79             bDriveMotorAxisStatus[1] AND bDriveMotorAxisStatus
           [2] // AND bDriveMotorAxisStatus[1]
80             THEN
81
82                 bMotorAvailable        := TRUE;
83                 //Lights is running in FB_HLI_CNCChannel_Upgrade.
           Light_Mode
84             ELSE //If conditions failed
85                 nState_Motors := OFF_WAIT_ENABLE;
86                 PLCAxisEnable := FALSE;
87             END_IF
88         END_IF
89     END_CASE

```


C.8 PRG_LoadDataFromPLC

C.8.1 Variáveis

```

1 PROGRAM PRG_LoadDataFromPLC
2 VAR
3     arraytest : ARRAY[0..MAX_ARRAY] OF UDINT := [ 16#2 , 16#3,
4             16#4, 16#5, 16#6, 16#7, 16#8, 16#A, 16#B, 16#C, 16#10,
5             16#16, 16#25, 16#26, 16#27];
6
7     stAxisX : ST_Axis_COM_Task;
8     stAxisY : ST_Axis_COM_Task;
9     stAxisZ : ST_Axis_COM_Task;
10
11     arraynameX : ARRAY[0..MAX_ARRAY] OF DINT ;
12     arrayrealX : ARRAY[0..2] OF LREAL;
13     arraynameY : ARRAY[0..MAX_ARRAY] OF DINT ;
14     arrayrealY : ARRAY[0..2] OF LREAL;
15     arraynameZ : ARRAY[0..MAX_ARRAY] OF DINT ;
16     arrayrealZ : ARRAY[0..2] OF LREAL;
17
18     fb_ADSReadX : ARRAY[0..MAX_ARRAY] OF ADSREAD;
19     fb_ADSReady : ARRAY[0..MAX_ARRAY] OF ADSREAD;
20     fb_ADSReadZ : ARRAY[0..MAX_ARRAY] OF ADSREAD;
21
22     fb_ADSReadProg_End : ADSREAD;
23 END_VAR
24 VAR CONSTANT
25     MAX_ARRAY : INT := 20;
26 END_VAR

```

Listagem C.17: Declaração de variáveis da função
PRG_LoadDataFromPLC

C.8.2 Código

```

1 Recolha_X();
2 Recolha_Y();
3 Recolha_Z();
4
5 fb_ADSReadProg_End(NETID := ,
6     PORT := 551,
7     IDXGRP := 16#23301, //20201#HEX      131329
8     IDXOFFS := 16#139, //Posicao
9     LEN := 1,
10    DESTADDR := ADR(bProgram_End),

```

```

11     READ := TRUE,
12     TMOUT := T#1S);
13
14 fb_ADSReadProg_End(READ := FALSE);
15
16 //Homing data
17 IF stAxisX.ac_homing_done_r AND stAxisY.ac_homing_done_r AND
    stAxisZ.ac_homing_done_r THEN
18     PLCAxisHoming_Done := TRUE;
19 ELSE
20     PLCAxisHoming_Done := FALSE;
21 END_IF

```

Listagem C.18: Função PRG_LoadDataFromPLC

C.8.3 Recolha_Eixo

```

1 i := 0;
2 j := 0;
3 FOR pos := 0 TO 14 DO //Read COM size
4     IF pos = 3 OR pos = 4 THEN //REAL
5         fb_ADSReadx[pos] (NETID := ,
6             PORT := 553,
7             IDXGRP := 16#20201, //20201#HEX      131329
8             IDXOFFS := arraytest[pos], //Posicao
9             LEN := SIZEOF(arrayrealx[j]),
10            DESTADDR := ADR(arrayrealx[j]),
11            READ := TRUE,
12            TMOUT := T#1S);
13
14         j := j + 1;
15     ELSE
16         fb_ADSReadx[pos] (NETID := ,
17             PORT := 553,
18             IDXGRP := 16#20201, //20201#HEX      131329
19             IDXOFFS := arraytest[pos], //Posicao
20             LEN := SIZEOF(arraynamex[i]),
21             DESTADDR := ADR(arraynamex[i]),
22             READ := TRUE,
23             TMOUT := T#1S);
24
25         i := i + 1;
26     END_IF
27     fb_ADSReadx[pos](READ := FALSE);
28 END_FOR
29

```

```

30  stAxisX.ac_active_position_acs_r := DINT_TO_REAL(arraynamex[0])
    /100;
31  stAxisX.ac_current_position_acs_r:= DINT_TO_REAL(arraynamex[1])
    /100;
32  stAxisX.ac_end_position_acs_r := DINT_TO_REAL(arraynamex[2])/100;
33  stAxisX.ac_active_feedrate_r  := arrayrealx[0]*6;
34  stAxisX.ac_current_feedrate_r  := arrayrealx[1]*6;
35  stAxisX.ac_direction_r        := DINT_TO_SINT(arraynamex[3]);
36  stAxisX.ac_mode_r            := DINT_TO_UDINT(arraynamex[4]);
37  stAxisX.ac_in_position_r      := DINT_TO_BOOL(arraynamex[5]);
38  stAxisX.ac_homing_done_r      := DINT_TO_BOOL(arraynamex[6]);
39  stAxisX.ac_state_r           := DINT_TO_UINT(arraynamex[7]);
40  stAxisX.ac_position_lag_r      := DINT_TO_REAL(arraynamex[8]);
41  stAxisX.ac_disable_pos_lag_r   := DINT_TO_BOOL(arraynamex[9]);
42  stAxisX.ac_pos_limit_conf_r    := DINT_TO_REAL(arraynamex[10]);
43  stAxisX.ac_neg_limit_conf_r    := DINT_TO_REAL(arraynamex[11]);
44  stAxisX.ac_feedhold_r          := DINT_TO_BOOL(arraynamex[12]);

```

Listagem C.19: Função Recolha_Eixo

C.9 FB_HLI3_CNCChannel_Upgrade

C.9.1 Variáveis

```

1  FUNCTION_BLOCK FB_HLI3_CNCChannel_Upgrade
2  VAR_INPUT
3      /// index of channel
4      nChanIdx: INT;
5
6      bCmdAutomatic : BOOL;
7      bCmdManual   : BOOL;
8          bCmdReference : BOOL;
9          bCmdMDI      : BOOL;
10     bCmdStandby: BOOL;
11     bCmdStart  : BOOL;
12     bCmdStop   : BOOL;
13     bCmdReset  : BOOL;
14         bCmdEmergencyStop : BOOL;
15     /// must be xxx.nc
16     sProgramName: STRING(HLI_IMCM_MODE_STATE_PARAM_STRL);
17     /// the MDI string
18     sMDIString: STRING(HLI_IMCM_MODE_STATE_PARAM_STRL);
19     /// static signal
20     bBackwardMotion: BOOL;
21     /// static signal
22     bFeedHold: BOOL;

```

```
23  /// static signal
24  bReducedFeed: BOOL;
25  /// static signal
26  bSingleBlock: BOOL;
27  /// static signal
28  bPrgBlockIgnore: BOOL;
29  /// static signal
30  bM01Stop: BOOL;
31  /// static signal
32  nSyntaxCheck: DINT;
33  // Manual Mode
34  stManualModeParameter: ST_ManualModeParameter;
35  stManualModeControlElements: ST_ManualModeControlElements;
36  // M30 Acknowledgement inside Channel FB
37  bM30Acknowledgement : BOOL;
38  // Override
39  nFeedOverride : UINT;
40  nRapidFeedOverride : UINT;
41  stOptions : ST_CNC_CH_Options;
42
43  fb_alarm : FB_Alarm;
44  END_VAR
45
46  VAR_OUTPUT
47      bError : BOOL;
48      nErrorID : UDINT;
49  END_VAR
50
51  VAR_IN_OUT
52      /// please only a flank
53      bModeStandby: BOOL;
54      /// please only a flank
55      bModeAutomatic: BOOL;
56      /// please only a flank
57      bModeManual: BOOL;
58      /// please only a flank
59      bModeMDI: BOOL;
60      /// please only a flank
61      bModeReference: BOOL;
62      /// please only a flank
63      bStateSelect: BOOL;
64      /// please only a flank
65      bStateReady: BOOL;
66      /// please only a flank
67      bStateStart: BOOL;
68      /// please only a flank
69      bStateStop: BOOL;
70      /// please only a flank
71      bStateReset: BOOL;
```

```

72  /// please only a flank
73  bDoSingleBlock: BOOL;
74  // manual mode
75  stManualModeControlElementState:
      ST_ManualModeControlElementState;
76 END_VAR
77 VAR
78  sProgramName_Home : STRING;
79  bGoToActive : BOOL;
80  /// Pointer to channel control structure
81  pChannel: POINTER TO ST_CncChannel;
82  ChannelMHTFunction: HLI3_QuitChannelMHTFunction;
83  /// Current Mode of Operation Modes;
84  currentMode: UDINT;
85  /// Current State of Operation Modes;
86  currentState: UDINT;
87  idx: INT;
88  bCheckOpModeFirst: BOOL;
89  fbRTrigReset: R_TRIG;
90  bResetRequest: BOOL;
91  nStateAutoMDIStart: INT;
92  ModeManual : HLI3_ManualModeHandler;
93  stCmdModesStates : ST_CNCCmdModesStates;
94
95  fbRTrigCmdStart: R_TRIG;
96  fbRTrigCmdStop: R_TRIG;
97  fbRTrigCmdReset: R_TRIG;
98  fbRTrigCmdAutomatic : R_TRIG;
99  fbRTrigCmdManual : R_TRIG;
100  fbRTrigCmdReference : R_TRIG;
101  fbRTrigCmdMDI : R_TRIG;
102  fbRTrigCmdStandby: R_TRIG;
103  // File Offset
104  fbADSWrite : ADSWRITE;
105  fbRTrigSetFileOffset: R_TRIG;
106  nState : INT;
107
108  fbLimit_Switch : FB_Limit_Switch;
109
110  fbDelay_Desactivated_Homing : TON := (PT := T#50MS);
111  bDontRepeat : BOOL := FALSE;
112 END_VAR

```

Listagem C.20: Declaração de variáveis da função
FB_HLI3_CNCCChannel_Upgrade

C.9.2 Código Modificado

```

1
2 // Ficheiro Identico ao HLI3_CNCChannel, com a correcao do
   Reference
3
4 IF NOT HLI3_CheckChannelValid(nChannel := nChanIdx) THEN
5     RETURN;
6 END_IF
7
8 // CNC PLC Watchdog
9 IF NOT stOptions.bDisableCNCPLCWatchDog THEN
10    HLI3_SetPLCAlive(nChanIdx := nChanIdx);
11 END_IF
12
13 (* handle reset request *)
14 fbRTrigReset
15    (CLK := bStateReset);
16
17 IF fbRTrigReset.Q THEN
18    //bRun_Code := FALSE;
19    nStateAutoMDIStart := 0;
20 END_IF
21
22 // FileOffset
23 FileOffset();
24
25 fbRTrigCmdAutomatic(CLK:=bCmdAutomatic);
26 fbRTrigCmdManual(CLK:=bCmdManual);
27 fbRTrigCmdReference(CLK:=bCmdReference);
28 fbRTrigCmdMDI(CLK:=bCmdMDI);
29 fbRTrigCmdStandby(CLK:=bCmdStandby);
30
31 fbRTrigCmdStart(CLK:=bCmdStart);
32 fbRTrigCmdStop(CLK:=bCmdStop);
33 fbRTrigCmdReset(CLK:=bCmdReset);
34
35 (* ***** *)
36 (* Motion-Control *)
37 (* ***** *)
38
39 (* EmergencyStop *)
40 HLI3_ChannelSetEmergencyStop(nChanIdx:=nChanIdx, bEStop :=
   bEmergencyStop_Activate);
41
42 (* Feed Override [0.1%] *)
43 HLI3_SetChannelOverride(nChanIdx:=nChanIdx, nOverride :=
   nFeedOverride);
44
45 (* Rapid feed override [0.1%], must be activated with parameter P-
   CHAN-00181 *)

```

```

46 HLI3_SetChannelRapidOverride(nChanIdx:=nChanIdx, nOverride :=
    nRapidFeedOverride);
47
48 (* Backward Motion *)
49 HLI3_ChannelSetBackwardMotion(nChanIdx:=nChanIdx, bBackwardMotion
    :=bBackwardMotion);
50
51 (* Program Block Ignore *)
52 HLI3_ChannelSetProgramBlockIgnore(nChanIdx:=nChanIdx,
    bPrgBlockIgnore:=bPrgBlockIgnore);
53
54 (* Feed Hold *)
55 HLI3_ChannelSetFeedHold(nChanIdx:=nChanIdx, bFeedHold:=bFeedHold);
56
57 (* Single Step *)
58 HLI3_ChannelSetSingleBlock(nChanIdx:=nChanIdx, bSingleBlock:=
    bSingleBlock);
59
60 (* Reduced Feed *)
61 HLI3_ChannelSetReducedFeed(nChanIdx:=nChanIdx, bReducedFeed:=
    bReducedFeed);
62
63 (* Syntax Check *)
64 gpCh[nChanIdx]^.decoder_mc_control.execution_mode.command_w :=
    nSyntaxCheck;
65
66 (* Optional stop M01 *)
67 gpCh[nChanIdx]^.bahn_mc_control.m01_stop_enable.command_w :=
    bM01Stop;
68
69 (* ***** *)
70 (* MHT-Function *)
71 (* ***** *)
72
73 (* MHT-Function, channel *)
74 IF NOT stOptions.bDisableMFuncHandling THEN
75     ChannelMHTFunction(nChanIdx:=nChanIdx, bM30Acknowledgement :=
        bM30Acknowledgement, bDisableAutoAcknowledgeM := stOptions.
        bDisableAutoAcknowledgeM);
76 END_IF
77
78 (* ***** *)
79 (* Channel Status Info and error *)
80 (* ***** *)
81 HLI3_GetChannelStatusInfo(nChanIdx:=nChanIdx);
82 HLI3_GetChannelError(nChanIdx:=nChanIdx);
83
84 (* ***** *)
85 (* do single stepping *)

```

```

86  (* ***** *)
87  (* Achtung: hier noch anpassen, nur in Abhaengigkeit vom aktuellen
      Status den Start setzen *)
88  IF bDoSingleBlock THEN
89      bDoSingleBlock := FALSE;
90      bStateStart    := TRUE;
91  END_IF
92
93  (* ***** *)
94  (* Operation Modes *)
95  (* ***** *)
96  (* Target modes and states *)
97  pChannel := ADR(CNCSystem.Channel[nChanIdx]);
98
99  fbLimit_Switch();
100 IF bStop_Limit_SW THEN
101     bStateStop := TRUE;
102 END_IF
103 IF bReset_Limit_SW THEN
104     bStateReset := TRUE;
105 END_IF
106
107 OpModes();
108
109 //Light_Tower
110 Light_Mode();
111
112 (* OperationMode Standby *)
113 IF stCmdModesStates.stModes.Standby = TRUE OR NOT PLCAxisEnable
      THEN //Go standby it enable is off
114
115     HLI3_ChannelModeStandby(nChanIdx := nChanIdx,ProgramName:=
          sProgramName, stCmdModesStates:=stCmdModesStates);
116     bCmdStandby := FALSE; //clear botton
117
118     IF PLCAxisEnable THEN
119         Warning_Enable(); //Clear Warning
120     END_IF
121 END_IF
122
123 (* OperationMode Auto *)
124 IF stCmdModesStates.stModes.Auto = TRUE AND NOT bLimitStop THEN
125     IF bMotorAvailable AND (PLCAxisHoming_Done OR
          PLC_Homing_Simulation) THEN (* Verificar se o Eixos estao
          enable do HMI *)
126         HLI3_ChannelModeAuto(nChanIdx := nChanIdx, ProgramName:=
            sProgramName, stCmdModesStates:=stCmdModesStates );
127     ELSE
128         IF NOT bMotorAvailable THEN

```



```

129         Warning_Enable();
130     ELSE
131
132         Error_Homing_Message();
133     END_IF
134     //stCmdModesStates.stModes.Auto := FALSE;
135     //stCmdModesStates.stModes.Standby := TRUE;
136     //bCmdStandby := TRUE;
137     //bWarningEnable := FALSE;
138 END_IF
139 END_IF
140
141 (* OperationMode MDI *)
142 IF stCmdModesStates.stModes.MDI AND NOT bLimitStop THEN
143     IF bMotorAvailable AND (PLCAxisHoming_Done OR
144         PLC_Homing_Simulation) THEN (* Verificar se o Eixos estao
145         enable do HMI *)
146         (*sProgramName := PLCCncMDIString;*)
147         HLI3_ChannelModeMDI(nChanIdx := nChanIdx, pMDIString:=ADR(
148             sMDIString),stCmdModesStates:=stCmdModesStates);
149     ELSE
150         IF NOT bMotorAvailable THEN
151             Warning_Enable();
152         ELSE
153             Error_Homing_Message();
154         END_IF
155         //stCmdModesStates.stModes.MDI := FALSE;
156         //stCmdModesStates.stModes.Standby := TRUE;
157         //bCmdStandby := TRUE;
158     END_IF
159 END_IF
160
161 (* OperationMode Reference *)
162 IF stCmdModesStates.stModes.Homing AND NOT bLimitStop THEN
163     IF PLCAxisEnable THEN (* Verificar se o Eixos estao enable do
164         HMI *)
165
166         //Se os eixos estiverem referenciados
167         IF PLCAxisHoming_Done AND NOT bDontRepeat THEN
168             PLC_Force_Homing := TRUE;
169             PRG_CNCAxesHandler.Reset_Calibration();
170
171             fbDelay_Desactivated_Homing(IN:=TRUE);
172             IF fbDelay_Desactivated_Homing.Q THEN
173                 PLC_Force_Homing := FALSE;
174                 bDontRepeat := TRUE;
175                 bModeReference := TRUE; //Esta a necessitar de dois
176                 pulsos, vamos a ver se isto da
177             END_IF

```

```

173     ELSE
174         bModeReference := FALSE;
175         PLC_Force_Homing := FALSE;
176         bDontRepeat := TRUE;
177     END_IF
178
179     PRG_CNCAxesHandler.Reset_Calibration();
180
181     IF NOT PLC_Force_Homing AND bDontRepeat THEN
182         Error_Homing_Message(); //Clear error
183         sProgramName_Home := 'C:\cnc\rpfn.nc'; (* we choose the
184             default program: rpfn.nc *)
185         HLI3_ChannelModeAuto(nChanIdx := nChanIdx, ProgramName:=
186             sProgramName_Home, stCmdModesStates:=stCmdModesStates )
187         ;
188         //HLI3_ChannelModeReference(nChanIdx := nChanIdx,
189             sRefPosString:=sProgramName_Home, stCmdModesStates:=
190             stCmdModesStates);
191         PRG_Light_Tower(Colour := 2 ,
192             Blink := TRUE);
193         IF PLCAxisHoming_Done THEN
194             fbDelay_Desactivated_Homing(IN:=FALSE);
195             stCmdModesStates.stModes.Homing := FALSE;
196             stCmdModesStates.stModes.Standby := TRUE;
197             bCmdStandby := TRUE;
198
199             bDontRepeat := FALSE;
200         END_IF
201     END_IF
202
203     ELSE
204         Warning_Enable();
205         //stCmdModesStates.stModes.Homing := FALSE;
206         //stCmdModesStates.stModes.Standby := TRUE;
207         //bCmdStandby := TRUE;
208
209     END_IF
210
211     END_IF
212
213     (* OperationMode Manual *)
214     (*
215     IF stCmdModesStates.stModes.Manual THEN
216         HLI3_ChannelModeManual(nChanIdx := nChanIdx, stCmdModesStates:=
217             stCmdModesStates);
218     END_IF
219     *)

```

```

216 Copy_Parameter();
217 //Necessario para limites
218
219 IF bMotorAvailable THEN (* Verificar se o Eixos estao enable do
    HMI *)
220   ModeManual
221     (stManualModeParameter      := stManualModeParameter,
222      stManualModeControlElements := stManualModeControlElements,
223      stManualModeControlElementState :=
224        stManualModeControlElementState,
225      stCmdModesStates           := stCmdModesStates,
226      arrAxisInG201              => stManualModeState[nChanIdx].
227        arrAxisInG201,
228     );
229 ELSE
230   Warning_Enable();
231
232 END_IF
233
234 (* Reset *)
235 HLI3_ChannelReset(nChanIdx:=nChanIdx); (* channel *)
236 FOR idx := 0 TO UDINT_TO_INT(gNrAx-1) DO (* axis *)
237   HLI3_AxisReset(nChanNo:=nChanIdx+1, nHliAxisIdx:=idx);
238 END_FOR

```

Listagem C.21: Declaração de variáveis da função
FB_HLI3_CNCCChannel_Upgrade

C.9.3 Função Copy_Parameter

```

1 nAxis_Jog_Selected := stManualModeParameter.stAxisParameter.
    stControlUnit[0].nLogAxisNr; //Obriga a ser o ecrã principal
2
3 bJog_Neg := stManualModeControlElements.stControlKeys[0].bKeyNeg;
4 bJog_Pos := stManualModeControlElements.stControlKeys[0].bKeyPos;
5
6 State_Mode := stCmdModesStates;
7
8 IF bLimitStop THEN
9   stCmdModesStates.stModes.Auto := FALSE;
10  stCmdModesStates.stModes.Homing := FALSE;
11  stCmdModesStates.stModes.MDI := FALSE;
12  stCmdModesStates.stModes.Standby := FALSE;
13  stCmdModesStates.stModes.Manual := TRUE;
14 END_IF

```

Listagem C.22: Função Copy_Parameter

Anexo D

Código G - TwinCAT

Neste anexo serão mencionados os códigos G e M que são aceites pelo TwinCAT ou que alterações existem.

De referir que serão comparados os códigos regulamentados dos protocolos ISO e DIN, para os que são usados, e depois o que o interpretador do TwinCAT lê e processa, como visualizado nas Tabelas D.1 e D.2.

De referir ainda que existe um vasto número de códigos que o TwinCAT permite, que não estão nos valores dos protocolos, podendo ser vistos na documentação da Beckhoff [19].

D.1 Código G

Tabela D.1: Comparação da conformidade do código G DIN/ISO com o lido da TwinCAT [19]

DIN/ISO code	Description	TwinCAT code	Conformity check
G00	<i>Rapid traverse</i>	G00	<i>compliant</i>
G01	<i>Linear interpolation with programmed feed rate</i>	G01	<i>compliant</i>
Continuação na próxima página			

Tabela D.1 – Continuação

DIN/ISO code	Description	TwinCAT code	Conformity check
G02	<i>Clockwise circular interpolation with programmed feed rate</i>	G02	<i>compliant</i>
G03	<i>Anti-clockwise circular interpolation with programmed feed rate</i>	G03	<i>compliant</i>
G04	<i>Programmable dwell time</i>	G04	<i>compliant</i>
G05	<i>Not assigned</i>	G05	<i>Direct tangential selection/deselection of tool radius compensation</i>
G06	<i>Selecting spline interpolation</i>	G151	<i>compliant</i>
G08	<i>Acceleration at block start</i>	G08	<i>compliant</i>
G09	<i>Deceleration at block end</i>	G09	<i>compliant</i>
G10	<i>Not assigned</i>	G10	<i>Constant feed rate with tool radius compensation</i>
G11	<i>Not assigned</i>	G11	<i>Adapted feed rate with tool radius compensation</i>
G12	<i>Not assigned</i>	G12	<i>Deselect corner deceleration</i>
G13	<i>Not assigned</i>	G13	<i>Select corner deceleration</i>
G17	<i>Select working plane XY</i>	G17	<i>compliant</i>
G18	<i>Select working plane ZX</i>	G18	<i>compliant</i>
G19	<i>Select working plane YZ</i>	G19	<i>compliant</i>
G20	<i>Not assigned</i>	G20	<i>Deselect mirroring</i>
G21	<i>Not assigned</i>	G21	<i>Mirroring programmed path on the Y axis</i>
G22	<i>Not assigned</i>	G22	<i>Mirroring programmed path on the X axis</i>
G23	<i>Not assigned</i>	G23	<i>Superimposing G21 and G22</i>
G25	<i>Not assigned</i>	G25	<i>Linear transitions with TRC</i>
G26	<i>Not assigned</i>	G26	<i>Circular transitions with TRC</i>
G33	<i>Thread cutting, constant pitch</i>	G33	<i>compliant</i>
G34	<i>Thread cutting, increasing pitch</i>	–	<i>Not assigned</i>
Continuação na próxima página			

Tabela D.1 – Continuação

DIN/ISO code	Description	TwinCAT code	Conformity check
G35	<i>Thread cutting, decreasing pitch</i>	–	<i>Not assigned</i>
G40	<i>Deactivate tool radius compensation</i>	G40	<i>compliant</i>
G41	<i>Activate tool radius compensation on left of contour</i>	G41	<i>compliant</i>
G42	<i>Activate tool radius compensation on right of contour</i>	G42	<i>compliant</i>
G51	<i>Not assigned</i>	G51	<i>Selection of diameter programming</i>
G52	<i>Not assigned</i>	G52	<i>Deselection of diameter programming</i>
G53	<i>Cancel zero offset</i>	G53	<i>compliant</i>
G54	<i>Select zero offset 1</i>	G54	<i>compliant</i>
G55	<i>Select zero offset 2</i>	G55	<i>compliant</i>
G56	<i>Select zero offset 3</i>	G56	<i>compliant</i>
G57	<i>Select zero offset 4</i>	G57	<i>compliant</i>
G58	<i>Select zero offset 5</i>	G58	<i>compliant</i>
G59	<i>Select zero offset 6</i>	G59	<i>compliant</i>
G60	<i>Not assigned</i>	G60	<i>Exact stop (stop at block end, then continue motion in next block)</i>
G61	<i>Not assigned</i>	G61	<i>Select polynomial contouring</i>
G63	<i>Tapping</i>	G63	<i>compliant</i>
G70	<i>Inputs in inch (inch)</i>	G70	<i>compliant</i>
G71	<i>Inputs in metric units</i>	G71	<i>compliant</i>
G74	<i>Homing</i>	G74	<i>compliant</i>
G80	<i>End machining cycle</i>	G80 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G81	<i>Drilling, centring cycle</i>	G81 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
Continuação na próxima página			

Tabela D.1 – Continuação

DIN/ISO code	Description	TwinCAT code	Conformity check
G82	<i>Drilling, spot facing cycle</i>	G82 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G83	<i>Deep hole drilling, chip breaking cycle</i>	G83 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G84	<i>Thread tapping cycle</i>	G84 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G85	<i>Boring 1 cycle</i>	G85 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G86	<i>Boring 2 cycle</i>	G86 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G87	<i>Boring 3 cycle</i>	G87 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G88	<i>Boring 4 cycle</i>	G88 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G89	<i>Boring 5 cycle</i>	G89 or not assigned	<i>Implicit subroutine call (if name was configured)</i>
G90	<i>Absolute dimension</i>	G90	<i>compliant</i>
G91	<i>Incremental dimension</i>	G91	<i>compliant</i>
G92	<i>Reference point offset</i>	G92	<i>compliant</i>
G93	<i>Inverse-time feed rate in 1/mm</i>	G93	<i>Machining time in seconds</i>
G94	<i>Feed rate in mm/min, inch/min, degrees/min</i>	G94	<i>compliant</i>
G95	<i>Feed rate in mm/revolution, inch/revolution</i>	G95	<i>compliant</i>
G96	<i>Constant cutting speed m/min</i>	G96	<i>compliant</i>
G97	<i>Spindle speed in rpm.</i>	G97	<i>compliant</i>
Continuação na próxima página			

Tabela D.1 – Continuação

DIN/ISO code	Description	TwinCAT code	Conformity check
G98	<i>Not assigned</i>	G98	<i>Setting negative software limit switch</i>
G99	<i>Not assigned</i>	G99	<i>Setting positive software limit switch</i>

D.2 Código M

Tabela D.2: Comparação da conformidade do código M DIN/ISO com o lido da TwinCAT [19]

DIN/ISO code	Description	TwinCAT code	Conformity check
M00	<i>Programmed stop</i>	M00	<i>compliant</i>
M01	<i>Optional stop</i>	M01	<i>compliant</i>
M02	<i>Program end</i>	M02	<i>compliant</i>
M03	<i>Clockwise spindle rotation (Classes 1 - 3) or cutting on (Class 4)</i>	M03	<i>Compliant (meaning is configurable)</i>
M04	<i>Anticlockwise spindle rotation (Classes 1 - 3) or cutting off (Class 4)</i>	M04	<i>Compliant (meaning is configurable)</i>
M05	<i>Spindle stop (Classes 1 - 3) or not assigned (Class 4)</i>	M05	<i>Compliant (meaning is configurable)</i>
M06	<i>Tool change</i>	M06	<i>mplicit subroutine call (if name was configured)</i>
M10	<i>Clamp</i>	M10	
M11	<i>Release</i>	M11	
M17	<i>Not assigned</i>	M17	<i>Subroutine end</i>
M19	<i>Spindle positioning (Classes 1 - 3) or not assigned (Class 4)</i>	M19	<i>Compliant (meaning is configurable)</i>
M29	<i>Not assigned</i>	M29	<i>Subroutine end</i>
M30	<i>Program end</i>	M30	<i>compliant</i>
M40	<i>Automatic gear changes (Classes 1 - 3) or unassigned</i>	<i>M40 or not assigned</i>	<i>compliant</i>
Continuação na próxima página			

Tabela D.2 – Continuação

DIN/ISO code	Description	TwinCAT code	Conformity check
M41	<i>Gear stage 1 (Classes 1 - 3) or free</i>	<i>M41 or not assigned</i>	<i>compliant</i>
M42	<i>Gear stage 2 (Classes 1 - 3) or free</i>	<i>M42 or not assigned</i>	<i>compliant</i>
M43	<i>Gear stage 3 (Classes 1 - 3) or free</i>	<i>M43 or not assigned</i>	<i>compliant</i>
M44	<i>Gear stage 4 (Classes 1 - 3) or free</i>	<i>M44 or not assigned</i>	<i>compliant</i>
M45	<i>Gear stage 5 (Classes 1 - 3) or free</i>	<i>M45 or not assigned</i>	<i>compliant</i>
M48	<i>Effective superimposition (e.g. override)</i>	<i>G166/G167</i>	<i>Select path/spindle override 100% (blockwise)</i>
M49	<i>Ineffective superimposition</i>	<i>not assigned</i>	
M60	<i>Tool change</i>	<i>not assigned</i>	