FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Desenvolvimento de controlador para equipamento de corte e furação de vigas

João André Ramos Silva

Mestrado Integrado em Engenharia Mecânica

Orientador: Professor Germano Veiga (FEUP)

Coorientador: Professor António Pessoa de Magalhães (FEUP)

Coorientador: Eng. Pedro Malaca (SARKKIS Robotics)

18 de Julho de 2017

Desenvolvimento de controlador para equipamento de corte e furação de vigas

João André Ramos Silva

Mestrado Integrado em Engenharia Mecânica

18 de Julho de 2017

Resumo

A automatização de linhas de produção é um ramo em constante crescimento, que se está a expandir muito para além das fronteiras das indústrias que normalmente incorporam soluções mecanizadas avançadas. À medida que as fronteiras da tecnologia se esbatem na generalidade das indústrias, o sucesso da implementação depende em grande medida da capacidade de aplicar e rentabilizar a utilização dos recursos tecnológicos de automatização. Assim, a automação industrial é, nos dias de hoje, uma área com uma importância fulcral nas diversas unidades industriais difundidas pelos diferentes setores económicos.

A presente dissertação pretende ir ao encontro deste conceito, implementando estes processos de automatização na fabricação de perfis metálicos na indústria metalomecânica, focando-se mais concretamente numa aplicação de automatização do processo de corte e furação de vigas, desde a inserção do ficheiro IFC, provenientes de softwares BIM, até à geração dos comandos necessários para o fabrico do produto final.

O processo de desenvolvimento da aplicação foi composto por duas fases. Em primeiro lugar desenvolveu-se um pós-processador de um software já existente, o 'MetroID BeamCUT', para que este adquira informação deste software e a traduza para código G, de maneira a que este seja corretamente interpretado pelo controlador. Seguidamente, foi desenvolvido o controlador, baseado na Norma IEC 61131-3.

De modo a validar a solução desenvolvida, foram inseridos no software vários exemplos de vigas reais e efetuados dois tipos de testes. Fez-se, em primeiro lugar, testes à sequenciação de comandos gerados pelo pós-processador, através da análise do código G gerado em relação aos ficheiros reais previamente, usados no fabrico de perfis por corte a plasma. Para os segundos testes, foram comparados os tempos médios da implementação da solução desenvolvida com o tempo médio de implementação do controlo NC genérico aplicado, um método ainda vigente. Estes testes comprovaram que o sistema está funcional e representa uma solução válida e pretendida pelo mercado, com um tempo de implementação muito menor relativamente às opções vigentes.

Abstract

The automation of production lines is an ever-growing business that is expanding well beyond the frontiers of industries that typically incorporate advanced mechanized solutions. As the frontiers of technology blur in most industries, the success of implementations depends to a large extent on the ability to apply and monetize the use of automation technology resources. Thus, industrial automation is today an area with a central importance in the various industrial units spread by the different economic sectors.

The present dissertation intends to meet this concept, implementing these processes of automation in the manufacture of metal profiles in the metallo-mechanical industry, focusing more concretely on the automation of the process of beam cutting and drilling, from the insertion of the IFC file, coming from any BIM software, to the commands required to manufacture the final product.

The development process was separated in two phases. In the first one, a controller based on the IEC 61131-3 Standard was developed. Subsequently, a post-processor of an existing software, 'MetroID BeamCUT', was developed so that it acquires information from this software and generates G code in order for it to be correctly interpreted by the controller.

In order to validate the developed solution, several examples of real beams were inserted into the software and two types of tests were performed. The first tested the sequencing of commands generated by the post-processor through the analysis of the generated G-code relative to the actual files previously used in the manufacture of plasma cutting profiles. For the second tests, the average times of the implementation were compared with the average implementation times of generic CNC systems, a method still in use. These tests proved that the system is functional and represents a valid solution, with a shorter implementation time than the alternatives currently used.

Agradecimentos

Porque não chegaria a esta fase do meu percurso se o tivesse caminhado sozinho, professo os meus mais sinceros agradecimentos:

Aos meus pais, do fundo do coração, por me terem dado a capacidade de sonhar e a hipótese de lutar pelos meus sonhos.

Ao meu irmão, pela amizade incondicional.

Aos meus orientadores da FEUP, Professor Germano Veiga e Professor António Pessoa de Magalhães pela proposta e ajuda na escrita desta dissertação com a qual tanto aprendi.

Aos engenheiros da SARKKIS Pedro Malaca, João Silva, Pedro Tavares, José Oliveira, Daniel Marques e Luís Ruas pela ajuda pronta sempre que uma dúvida surgiu, pelos ensinamentos dados e pela boa disposição diária.

A todos os meus amigos, que direta ou indiretamente, contribuíram para que estes 5 anos passassem demasiado rápido.

À Joana, por ter feito valer a pena.

"Twenty years from now you will be more disappointed by the things you didn't do than by the ones you did do. So throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover."

Mark Twain

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

BIM	Building Information Modeling
CAD	Computer Aided Design
CAM	Computer-Aided Manufacturing
CAPP	Computer-Aided Process Planning
CAN	Controller Area Network
CFC	Continuous Function Chart
CIP	Common Industrial Protocol
CNC	Computer Numeric Control
CRC	Cyclic Redundancy Check
DLL	Dynamic-Link Library
EtherCAT	Ethernet for Control Automation Technology
EtherNet/IP	Ethernet Industrial Protocol
FBD	Function Block Diagram
FTP	File Transfer Protocol
HMI	Human Machine Interface
IFC	Industry Foundation Classes
I/O	Input-Output
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
MAC	Media Access Control
PC	Personal Computer
PLC	Programmable Logic Controllers
POU	Program Organization Unit
RTU	Remote Terminal Unit
SERCOS	SErial Real-time COmmunication System
SFC	Sequential Function Chart
ST	Structured Text
TCP/IP	Transmission Control Protocol/Internet Protocol
XML	Extensible Markup Language

Capítulo 1

Introdução

1.1 BIM e a Fabricação de Vigas na Construção Metalomecânica

Building Information Modeling (BIM), consiste numa metodologia de abordagem à gestão da informação na construção baseada num modelo virtual do edifício, que tem vindo a ganhar maior protagonismo no sector da construção devido às grandes potencialidades que oferece (1).

A generalidade das ferramentas BIM oferece um conjunto padrão de benefícios de utilização imediatos, como a produção de informação de forma mais expedita e fiável, a redução de erros e conflitos de projeto, e a otimização geral dos custos e prazos na execução de tarefas (1).

Estas ferramentas são importantes pois, como se pode ver na figura 1.1, permitem a interoperabilidade entre departamentos relacionados com a construção, desde a arquitetura e estrutura do edifício, até à canalização e parte elétrica.

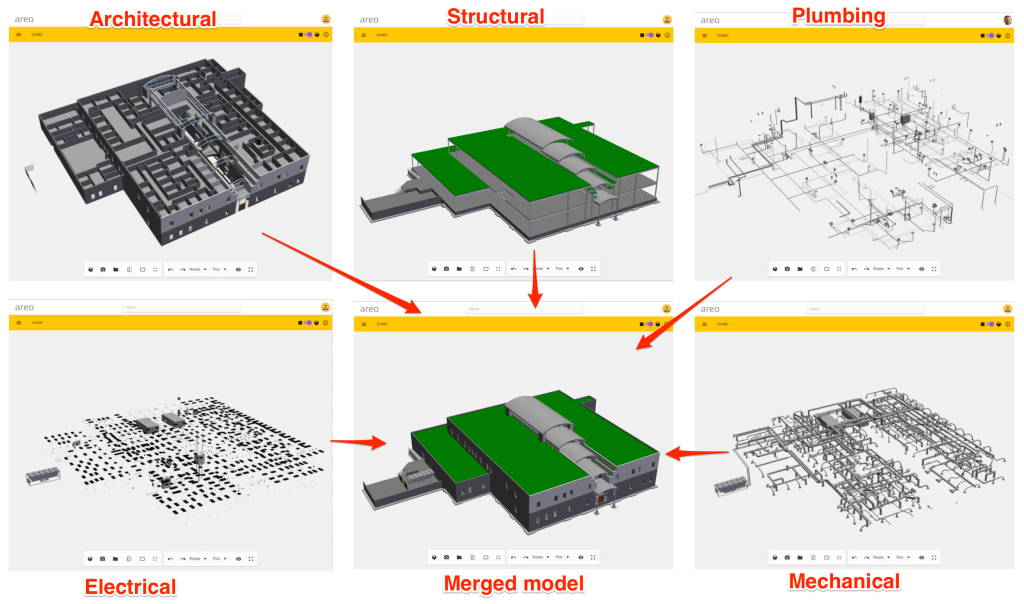

Figura 1.1: Ferramenta BIM "areo"(2)

Para que tal aconteça, é necessário que a informação seja exportada para um formato padrão neutro e *open-source*, de modo a poder ser lido por um grande espectro de softwares disponíveis para as diferentes áreas.

Com essa finalidade, e apesar de existirem outros formatos proprietários dos softwares BIM, existe o formato IFC, largamente utilizado na indústria para a partilha destes tipo de ficheiros.

O IFC, do inglês *Industry Foundation Classes*, é um modelo de dados usado para descrever dados e geometria para as diferentes especialidades e fases do ciclo de vida da construção. O IFC era já um standard *de-facto* antes de se tornar um standard ISO. Um ficheiro IFC resulta de uma exportação de uma ferramenta BIM que se pretende importar para outras aplicações (3).

Relativamente à construção metalomecânica, este formato contém já as informações relativas a todas as referências geométricas como secção, profundidade, possíveis furos ou cortes de modo a permitir a fixação desta, além de referências dimensionais como área da secção, altura ou volume e também material a utilizar (4).

Para ser possível o fabrico das vigas, é necessário haver a transformação deste formato IFC para instruções para as máquinas de corte e furação de vigas, como a representada na figura 1.2.

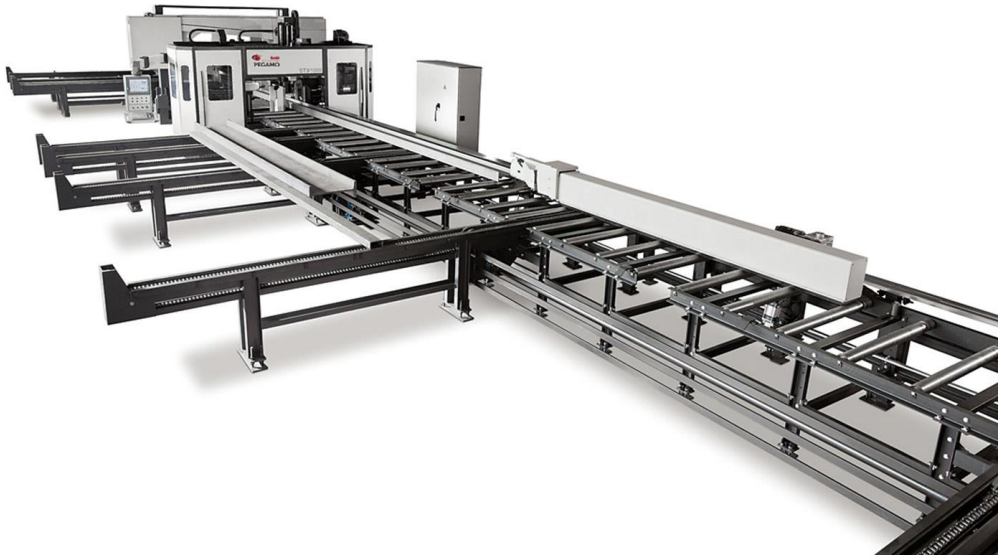

Figura 1.2: Célula de corte e furação de vigas

Devido à grande diversidade das dimensões das vigas, que na maior parte das máquinas pode ir até aos 500x1000mm em termos de altura e largura de secção, e também devido à rapidez que se quer atingir na realização das operações, as máquinas de corte e furação de vigas requerem uma flexibilidade elevada. Para tal recorre-se ao controlo numérico CNC (*Computer Numeric Control*), pois é uma ótima solução a nível de flexibilidade e rapidez de execução.

Geralmente, o controlo numérico é usado em máquinas de uma só ferramenta, como máquinas de arranque de apara e passa por várias fases, como é possível verificar na figura 1.3.

Inicialmente, é desenhada a forma desejada num software CAD (Computer Aided Design).

De seguida, é necessário recorrer a um software CAPP (Computer-Aided Process Planning) de forma a traduzir a informação do desenho CAD em processos e instruções de produzam a peça de uma forma eficiente.

Após este passo, é usado um software CAM (Computer-Aided Manufacturing) que utiliza a informação dos processos a realizar, faz o planeamento de trajetórias e traduz os movimentos para código CNC (Computer Numeric Control).

Finalmente, devido às particularidades de cada máquina e à não universalidade do código NC que corre nas máquinas ferramenta, é necessário recorrer a um pós-processador que traduza o código genérico para o código NC da máquina.

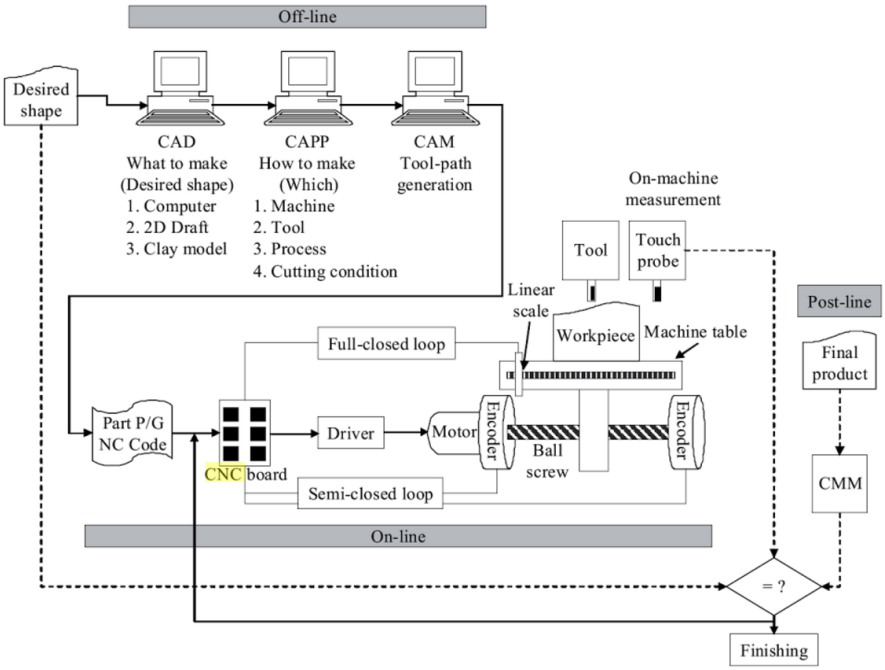

Figura 1.3: Arquitetura de ferramentas CNC de maquinagem e fluxo de operações (5)

Contudo, esta não é uma solução viável para o corte e furação de vigas, pois estas ferramentas são desadequadas ao problema em questão.

Em primeiro lugar, os softwares CAD não leem ficheiros IFC, sendo que teriam de ser adaptados para este caso. Além disso, os softwares CAM não estão pensados para operações com múltiplas ferramentas. De forma análoga, o próprio controlo CNC não possibilita o controlo de várias ferramentas em simultâneo de forma independente.

Tendo em conta todas estas problemáticas e a constante procura de soluções flexíveis e modulares no ambiente fabril, a automatização de todo este processo permitiria ir ao encontro dessa procura e reduzir custos e tempos de produção.

Assim, uma solução tendente a resolver o problema em questão passaria por um software orientado para o corte e furação de vigas, de forma a que apenas fosse requerido ao operador importar o ficheiro IFC para o sistema.

O software, juntamente com um pós-processador e um controlador, iriam ser responsáveis pela criação do código G e posterior controlo do equipamento, permitindo assim uma diminuição dos tempos de fabrico e consequente diminuição de custos.

1.2 Objetivos da Dissertação

De forma a ir ao encontro da resolução o problema, pretende-se eliminar etapas deste processo, de forma a torná-lo mais automatizado e rápido. Como se pode ver no fluxo da figura 1.4 pretende-se reduzir as fases deste processo para apenas 4 etapas que se executarão de forma conseguinte e automática.

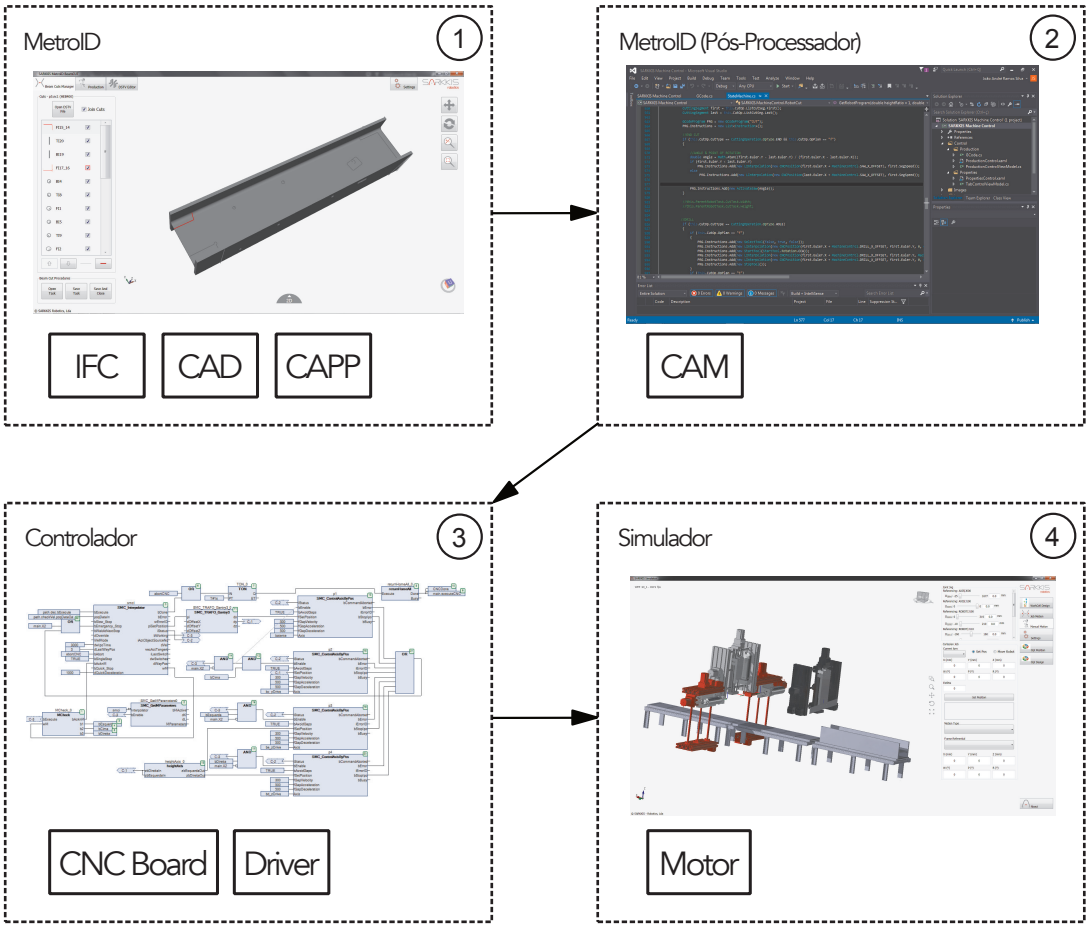

Figura 1.4: Fluxo do modelo a desenvolver

O primeiro passo trata de utilizar o software MetroID, desenvolvido pela SARKKIS Robotics, que permite a integração direta de ficheiros de formato BIM como IFC e DSTV e gera sequências de operações. Além disso, permite também a edição de ficheiros *on-site* e posterior transformação

em sequências de operações, com o principal propósito de facultar ao operador a possibilidade de fazer mudanças de última hora, sem se perder a precisão pretendida.

De seguida, pretende-se transformar essas sequências de operações em código máquina, através de um pós-processador a desenvolver.

O controlo de movimento e a sequenciação de comandos será feita no controlador, também a desenvolver.

Por fim, será criado um simulador de modo a testar o funcionamento do controlador e simular o comportamento de uma máquina industrial de corte e furação de vigas.

Em suma, pretende-se que o utilizador apenas importe o ficheiro IFC para o MetroID. O restante processo de fabrico da viga será responsabilidade do pós-processador e do controlador a desenvolver.

O principal objetivo desta dissertação passa pelo desenvolvimento de um controlador para equipamento de corte e furação de vigas.

Além disso, deseja-se criar uma solução que permita utilizar o MetroID, que apesar de ser normalmente usado para corte de vigas a plasma, será adaptado para máquinas industriais de corte e furação, para gerar vigas com cortes e furos de alta precisão. Assim, é esperado obter uma solução integrada PLC-CNC completamente compatível com a norma IEC 61131-3, capaz de controlar o equipamento multi-eixo.

Posto isto, existem uma série de etapas intermédias a cumprir, tais como:

- **Definir requisitos da instalação** — Para obter uma melhor compreensão do problema em questão é necessário ter o reconhecimento de todos os requisitos da instalação, assim como possíveis limitações que a máquina possa ter, devido às suas características ou ao layout em que está colocada;

- **Definir a arquitetura do software e componentes do sistema** — Consoante os requisitos definidos, o próximo passo passará por determinar a arquitetura do software de modo a esta responder aos problemas levantados e a cumprir todos os requisitos;

- **Especificar fluxos de informação a montante e a jusante da aplicação** — Conjuntamente com a arquitetura do software é necessária a definição dos fluxos de informação, por forma a perceber quais terão de ser os *inputs* e *outputs* do controlador;

- **Desenvolver equipamento para controlo de eixos coordenados** — Após a recolha de toda a informação prévia, será agora possível realizar a principal etapa desta dissertação, que passará pelo desenvolvimento do controlo dos eixos coordenados da máquina a partir de um softPLC;

- **Desenvolvimento do pós-processador** — É necessário desenvolver um pós-processador em C# da aplicação MetroID, para que o código G gerado seja coadjuvante com a interpretação do código G da parte do controlador.

- **Desenvolvimento da HMI para o equipamento** — Finalmente será criada uma Interface Homem Máquina, que forneça ao utilizador só uma melhor visão e controlo de todo o processo, como também permita o controlo manual dos eixos.

1.3 Estrutura da Dissertação

Em termos estruturais, a dissertação será dividida em 3 partes :

- **Definição do problema** — No capítulo 2 é retratada a definição pormenorizada do problema referente a máquinas de corte e furação de vigas, como também relativa a todos os conhecimentos necessários para a compreensão do problema em questão.

- **Soluções encontradas** — No capítulo 3 discutem-se os principais desenvolvimentos e soluções gerais relativas ao sistema, tais como componentes, arquitetura do sistema e fluxo de informação. Também se apresentam os protocolos de comunicação escolhidos para conectar os diferentes componentes e o modo como a informação é gerada e transferida entre módulos. Nos capítulos 4 e 5 discute-se em maior detalhe as soluções encontradas a nível do pós-processador e do controlador, respetivamente.

- **Solução geral e conclusões** — No capítulo 6 apresentar-se-á os testes realizados ao sistema. Além disso, discute-se os resultados obtidos a nível dos ficheiros gerados e os movimentos dos atuadores do simulador 3D. Finalmente, no capítulo 7 são expostas as conclusões a que se chegou, assim como são propostos futuros trabalhos que possam desenvolver mais este tópico.

Capítulo 2

Definição do problema

Neste capítulo irá ser abordado o problema em maior detalhe, assim como alguns conceitos e ferramentas necessários à melhor compreensão e solução desta dissertação, tais como normas e procedimentos a seguir e ferramentas de desenvolvimento.

2.1 Máquinas de Corte e Furação de Vigas

As células de corte e furação de vigas são geralmente compostas por dois componentes principais: a máquina de corte e a máquina de furação, compostas por uma serra e 3 brocas, respetivamente, sendo duas das brocas horizontais, para furar o banzo esquerdo e o banzo direito e uma vertical que fura a alma da viga.

Uma destas células está presente na NORFERSTEEL, em Vale de Cambra e trata-se de uma KALTENBACH KDXS 1015 conjunta com uma KALTENBACH KBS 1010 DG. Trata-se de uma solução integrada de corte e furação acoplada com um tapete de rolos e um batente para referenciação da viga. É controlada através de um sistema de controlo CNC de 8 eixos não convencional.

Como se pode ver pelo *layout* da célula, representado na figura 2.1, a viga passa primeiramente pela máquina de serração e só depois pela furação.

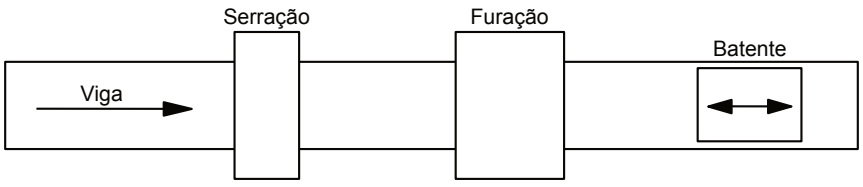

Figura 2.1: Esquema da célula

Esta organização da célula implica restrições ao nível da sequenciação de comandos, não só na parte do controlo lógico e de movimento, como também a nível do pós-processador do MetroID.

Considere-se a viga da figura 2.1, com operações de furação a serem realizados junto do corte final efetuado pelo serrote. Neste cenário, é necessário efetuar o corte antes dos furos supracitados, embora a sequência geométrica seja a contrária. Esta particularidade da máquina tem impacto no desenvolvimento do controlo lógico a desenvolver, bem como do pós-processador do MetroID.

Seguindo este *layout*, é necessário que a referenciação seja feita a jusante da viga, pois, caso contrário, perder-se-ia a referência após a serração da viga. Para tal, é utilizado um batente, como o presente na figura 2.7.

Contudo, existe um cenário alternativo em que a furação se encontra antes da serra. Neste caso, é necessário recorrer a um empurrador, auxiliado por uma pinça, ao invés do batente, como se vê na célula de corte e furação da Voortman da figura 2.2.

Figura 2.2: Célula de corte e furação da Voortman

Apesar disso, este mecanismo apresenta algumas limitações pois, como se vê no *layout* da figura 2.3, leva a que o corte tenha de ser a última tarefa a executar pois não se pode assegurar a referenciação da viga após o corte. Assim, mediante as dimensões da viga, pode ser necessário o movimento da viga em dois sentidos durante a execução das operações, o que diminui a precisão da referenciação.

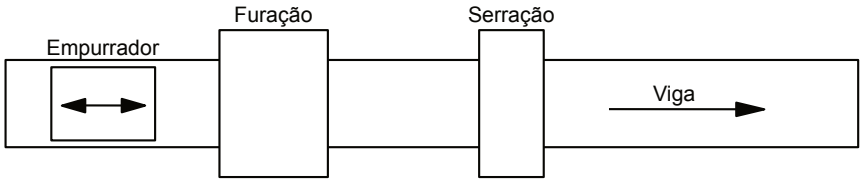

Figura 2.3: Esquema alternativo da célula

Como tal, e porque se procurou desenvolver um sistema aplicável no exemplo real presente na NORFERSTEEL, nesta dissertação será abordado o controlo da primeira solução.

A máquina de furação (KDXS 1015) contém 3 brocas, como se pode ver na figura 2.4, e tem capacidade de furar vigas cuja secção pode variar até 1000x500mm.

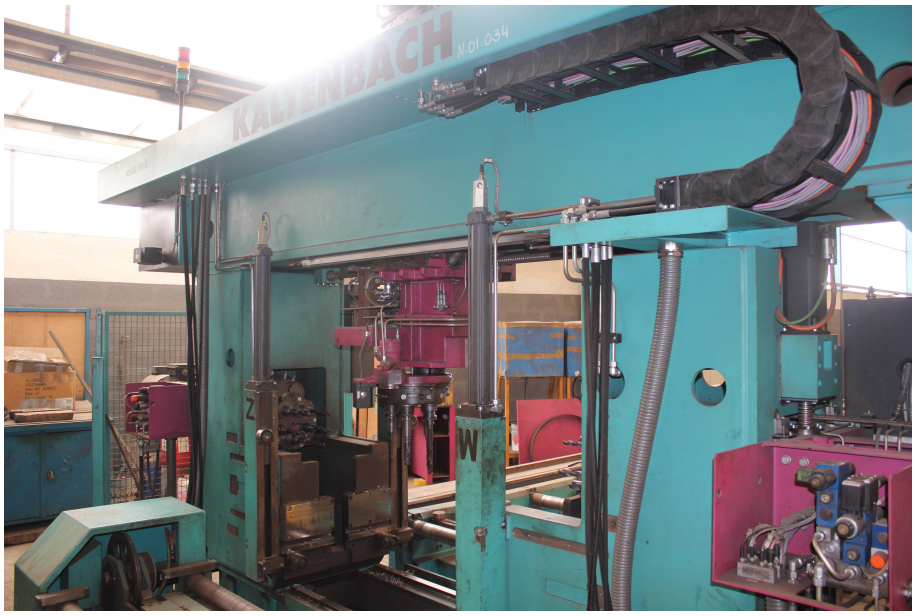

Figura 2.4: Máquina de furação

Possui também um mecanismo de aperto da viga que torna possível não só o guiamento desta ao longo do percurso, como também segurar a viga de uma forma fiável aquando da furação, permitindo assim diminuir as vibrações e prolongar o ciclo de vida do equipamento. Além disso, este mecanismo possibilita também a medição de forma inteligente e exata da largura real da viga.

Esta máquina possui as dimensões de 5370x1670x3750 mm e um peso de 6,5 toneladas. Em funcionamento, a KDXS 1015 possui uma potência de 29,5 kW, uma velocidade de avanço de 20000 mm/min e uma velocidade de rotação que varia entre 150 a 2500 rotações por minuto.

Existe também nesta máquina a possibilidade de executar furos com 5 diâmetros diferentes, pois as 3 brocas possuem um mecanismo de mudança de ferramenta, retratado na figura 2.5, que permite seleção de 5 diâmetros diferentes do furo entre os 8mm e 50 mm

Para executar a mudança de ferramenta, a máquina possui um sistema automático de troca de ferramenta, no qual esta é puxada atrás, rodada para posicionar corretamente a broca com o diâmetro escolhido e essa mesma broca é avançada ligeiramente, de forma a ser a única a alcançar a viga.

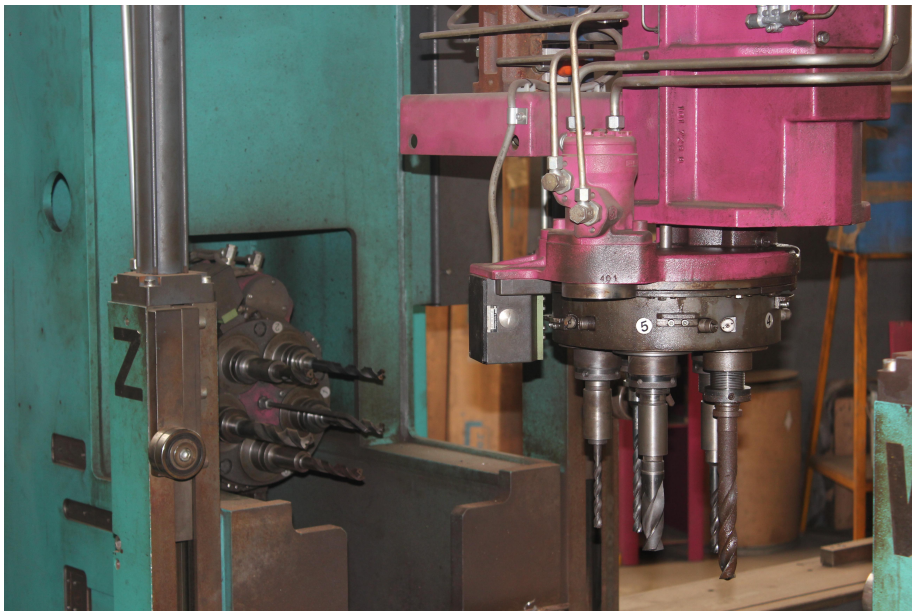

Figura 2.5: Pormenor das brocas

Relativamente à máquina de corte (KBS 1010 DG), presente na figura 2.6, permite serrar a viga num ângulo compreendido no intervalo [-45º, +60º] no plano XOY, paralelo ao tapete transportador. De forma a permitir um corte de melhor qualidade e um ciclo de vida maior da serra, esta encontra-se inclinada em 3º, permitindo uma menor área de contacto entre a serra e a viga e, consequentemente, um menor desgaste. Além disso, tem a capacidade de cortar vigas cuja secção pode variar até 1010x500mm.

Figura 2.6: Máquina de corte

A máquina de corte possui as dimensões de 3610x1160x2420 mm e um peso de 3,9 toneladas. Em funcionamento, possui uma potência de 7,5 kW, uma velocidade de avanço de 2500 mm/min e uma velocidade de corte da serra que varia entre 15 000 e 120 000 mm/min.

Por sua vez, o batente, ilustrado na figura 2.7, permite a constante referenciação da posição da viga em relação às brocas e à serra, possibilitando um corte e uma furação dimensionalmente precisa. De modo a este mecanismo funcionar com a precisão pretendida, é necessário que a viga possua mais de 1200 mm de comprimento.

Nesta célula, a posição do batente é controlada através de um servomotor com um encoder rotativo absoluto, permitindo assim o seu controlo numérico.

Figura 2.7: Batente

2.2 Controlo Numérico de Máquinas Industriais

Como referido no capítulo 1, devido à sua flexibilidade, o controlo numérico é a melhor solução e a solução vigente neste tipo de problemas. Do inglês *Numerical Control* (NC), consiste num sistema que permite uma máquina ferramenta produzir peças de forma rápida, precisa e repetitiva (5).

Um sistema de controlo CNC de um dispositivo de maquinagem consiste numa ferramenta controlável que permite o movimento da ferramenta para a modelação de um objeto.

Aquando do seu desenvolvimento, o controlo numérico de máquinas industriais tinha o propósito de maquinar peças com formas complexas de uma maneira precisa. Assim, o conceito foi primeiramente aplicado em fresagem de peças. Contudo, com o passar do tempo, a sua popularidade junto da indústria tem vindo a aumentar devido à sua enorme produtividade e ao aumento do número de máquinas (5).

Do ponto de vista funcional, um sistema CNC é composto por 3 subsistemas. Como se pode ver na figura 2.8 estes subsistemas consistem em interface homem máquina (HMI ou MMI), no controlador lógico programável (PLC) e no núcleo de controlo numérico (NCK, do inglês *Numerical Control Kernel*).

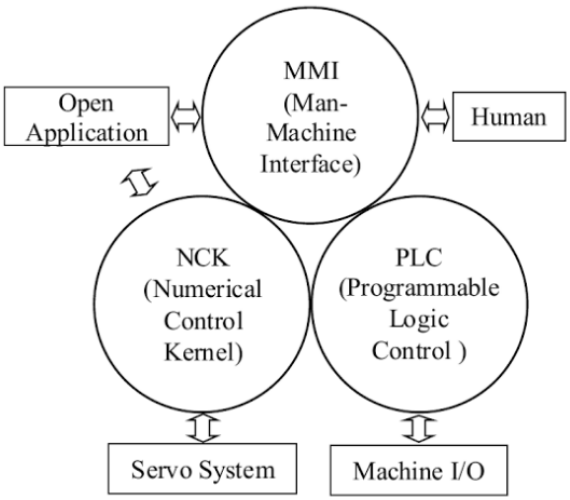

Figura 2.8: Estrutura de um sistema CNC (5)

O HMI, como o nome indica, faz a ligação entre o operador e a máquina, isto é, executa comandos de operações da máquina, mostra o seu estado e oferece a opção de editar o programa.

Por sua vez, o PLC controla o sequenciamento de operações, como mudanças de ferramenta, velocidades de rotação da ferramenta ou processamento de sinais de entrada e saída.

Finalmente, o NCK é responsável por todo o controlo de movimento e assume a interpretação do programa, a interpolação das posições, o controlo de posição e a compensação do erro com base no programa interpretado. Finalmente, controla os servomotores que levam a peça a ser maquinada.

Desse modo recebe instruções de controlo através de um sistema de mensagens e inclui um controlo de movimento através de um sistema controlo como o representado na figura 2.9. O sistema de controlo permite modificações fáceis ao sistema por pessoas com um conhecimento limitado do sistema de controlo na sua totalidade e é facilmente adaptável a avanços em novas ferramentas CNC (6).

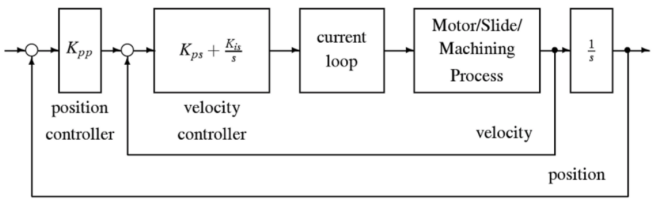

Figura 2.9: Estrutura de controlo (5)

A nível de controlo, os sistemas CNC sensores de velocidade e posição que enviam feedback para o circuito de controlo de forma a minimizar erros. O sistema consiste em 3 níveis (*loops*) de controlo. No *loop* mais exterior, efetua-se o controlo de posição, a um nível intermédio o controlo de velocidade e a um nível interno o controlo de binário.

Usualmente, o controlo de posição encontra-se no controlo numérico e os restantes na drive do servomotor. No entanto, não existe norma nem padrão sobre a localização dos *loops* de controlo e estes podem variar de acordo com as especificações do fabricante (5)

2.3 Norma IEC 61131-3

Como dito em 1.2, um dos objetivos desta dissertação é o desenvolvimento de uma solução completamente padronizada com a norma IEC 61131-3.

A rápida evolução dos controladores industriais levou ao crescimento exponencial do mercado dos PLC's, levando a que cada fabricante tivesse as suas próprias ferramentas e linguagens de programação, aumentando a complexidade dos sistemas, não só a nível de desenvolvimento, como também a nível de comunicação. A norma IEC 61131 procura solucionar este problema de modo a padronizar o desenvolvimento.

Divide-se em 9 partes e foi criada pela International Eletrotechnical Comission, uma organização criada em 1906 que se foca na preparação publicação de normas internacionais em tecnologias relacionadas com a eletricidade e eletrónica e que providencia uma plataforma para empresas e indústrias reunirem, discutirem e desenvolverem as normas internacionais necessárias (7).

A terceira parte desta norma fornece metodologias de programação de forma estruturada e modular e define os elementos básicos de programação e regras a nível semântico e sintático para as linguagens de programação mais comuns.

Isto inclui linguagens gráficas como *Ladder Diagram* e *Function Block Diagram* e linguagens textuais como *Instruction List* e *Structured Text*, assim como meios que permitem que os fabricantes expandam e adaptem estas funcionalidades nos seus controladores. Os elementos da *Sequential Function Chart* são também definidos em termos da estruturação da organização interna dos programas e funções do PLC.

Também permite o uso de mais linguagens de programação, como Visual Basic ou C++, partindo do pressuposto que são obedecidas as mesmas formas de chamadas e trocas de dados. Aborda também conceitos como as configurações necessárias para que o PLC cumpra a sua função de controlo, os recursos físicos ou virtuais exigidos, de forma a que o PLC seja capaz de executar os programas e a definição de tarefas que possibilitam o controlo da execução de programas a partir de eventos (*triggers*) ou a partir de forma periódica (ciclos). A versão corrente é a 3.0, de 2007 (8).

De acordo com a norma IEC 61131-3, as Unidades Organizacionais de Programas, também conhecidas como *Program Organizational Units* (POU) são estruturadas como mostrado na figura 2.10 e dividem-se em 3 (7):

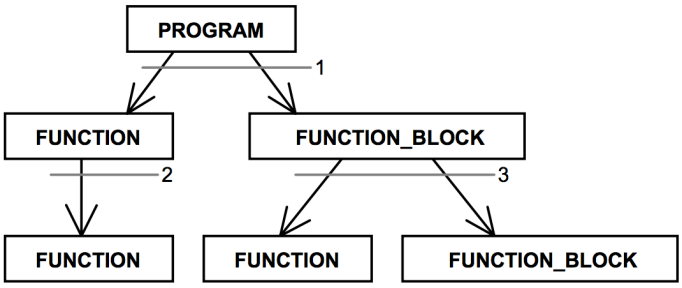

Figura 2.10: Estruturação das POUs (7)

- **Programas** — Representa o que na informática é conhecido como o *main*. Construídos a partir de sub-rotinas como Blocos Funcionais e Funções, têm acesso direto a todas as variáveis da aplicação como, por exemplo, entradas e saídas do PLC embora estas tenham de ser declaradas nesta POU ou a um nível superior (Recursos, Configuração ou Lista de Variáveis Globais) (7).
- **Blocos Funcionais** — São o principal componente da estruturação dos programas de PLCs. Podem ser chamados por programas e por outros blocos funcionais e podem eles próprios chamar funções e blocos funcionais. Ao contrário das funções, têm de ser instanciadas sempre que usadas, pois são elementos com memória. Isto significa que dois blocos funcionais iguais com as mesmas entradas podem ter saídas diferentes, dependendo do estado em que se encontram. Contadores e temporizadores são um bom exemplo ilustrativo do comportamento dos blocos funcionais (7).
- **Funções** — O conceito da função nesta norma assenta na ideia que o conjunto de instruções executadas não é ambígua, isto é, as mesmas entradas geram sempre as mesmas saídas. Como exemplos genéricos de funções tem-se funções de soma, multiplicadores ou funções trigonométricas (7).

2.3.1 Linguagens de Programação

Para a realização da dissertação estão disponíveis todas as 5 linguagens de programação definidas na norma IEC 61131-3, 2 textuais e 3 gráficas, e ainda um editor gráfico não definido nessa norma, o CFC (*Continuous Function Chart*). Estas são:

- Lista de Instruções (IL)
- Texto Estruturado (ST)
- Diagrama de Contactos (LD)
- Diagrama de Blocos (FBD)
- Controlo Sequencial de Funções (SFC)
- Controlo Continuo de Funções (CFC)

2.3.1.1 Lista de Instruções

Linguagem de baixo nível, similar à linguagem de *assembly*, mas designada para PLC's. Cada instrução começa numa nova linha e contém um operador e, dependendo do tipo de operação, um ou mais operandos separados por vírgulas. Como se pode depreender, é uma linguagem muito limitada e, por essa razão, pouco usada, sendo utilizada apenas para pequenos programas (7).

Exemplo (9):

```
LD      17
ST      lint            (* comment *)
GE      5
JMPC    next
LD      idword
EQ      istruct.sdword
STN     test
next:
```

2.3.1.2 Texto Estruturado

Linguagem de alto nível, similar a C e baseada em Pascal. Consiste numa série de instruções que podem ser executadas de forma estrutural (linha a linha), de forma condicional ("IF..THEN..ELSE/CASE") ou em loop (FOR..DO/REPEAT..UNTIL/WHILE..DO) (7).

Exemplo (9):

```
IF value < 7 THEN
     WHILE value < 8 DO
          value := value + 1;
     END_WHILE;
END_IF;
```

2.3.1.3 Diagrama de Contactos

Do inglês *Ladder Diagram* consiste num método de representar o programa graficamente, através de contactos e relés, muito usado para controlo lógico ou sequencial. É executado da esquerda para a direita em que cada relé representa um possível bit no PLC e cada contacto representa um interruptor ou um estado (7).

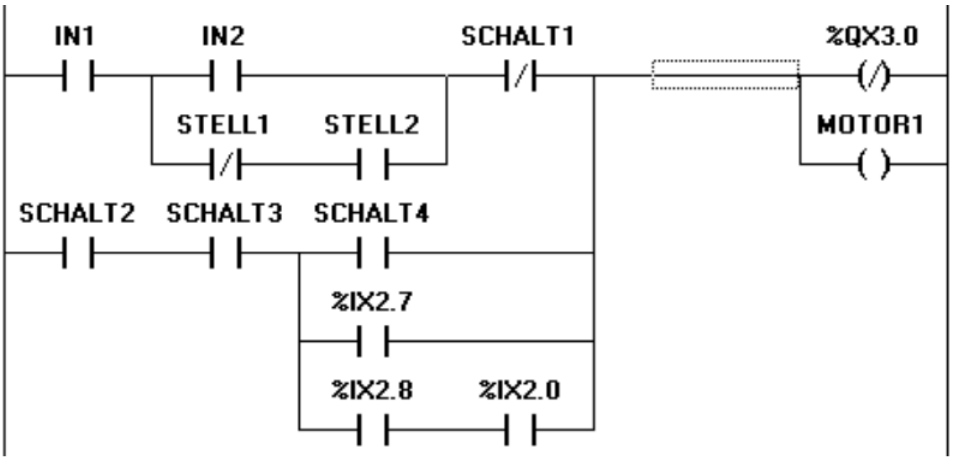

Figura 2.11: Exemplo Diagrama de Contactos (9)

2.3.1.4 Diagrama de Blocos

Function Block Diagram é uma Linguagem gráfica, fácil de interpretar, composta por blocos de funções e que descreve a relação entre o *input* e o *output*. Tal como os diagramas de contactos, é executado de forma unidirecional da esquerda para a direita, o que permite destacar o fluxo de informação a nível de entradas e saídas, destacando também o processamento dos sinais (7).

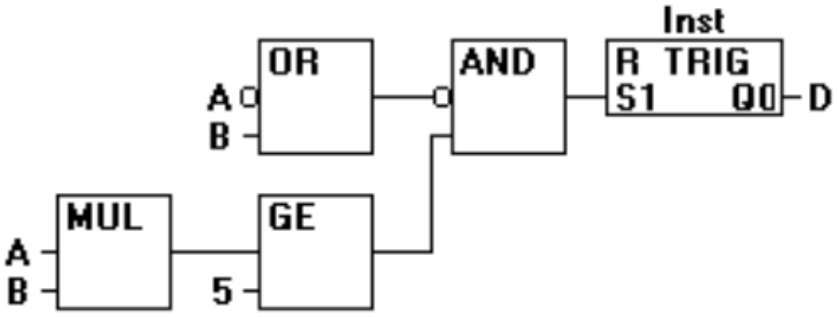

Figura 2.12: Exemplo Diagrama de Blocos (9)

2.3.1.5 Controlo Continuo de Funções

Visualmente parecido com o *Function Block Diagram*, o *Continuous Function Chart* tem a vantagem de dar liberdade em termos de colocação dos blocos sem a necessidade da execução ser da esquerda para a direita, o que, consequentemente, permite a representação de ciclos fechados e *feedback*, sem o uso de variáveis intermediárias (7).

Figura 2.13: Exemplo Controlo Sequencial de Funções(9)

2.3.1.6 Diagrama de Estados

Sequential Function Chart é também uma linguagem gráfica, mas baseada em 'Grafcet', que permite descrever um programa através de estados (onde ocorrem as ações) e transições entre estes. Representa uma máquina que, num dado momento, apenas pode estar num de vários estados possíveis. Ao longo do tempo, é possível que a máquina mude de estado quando um evento é efetuado ou quando se cumprem certas condições. A esta mudança de estado é dado o nome de transição (7).

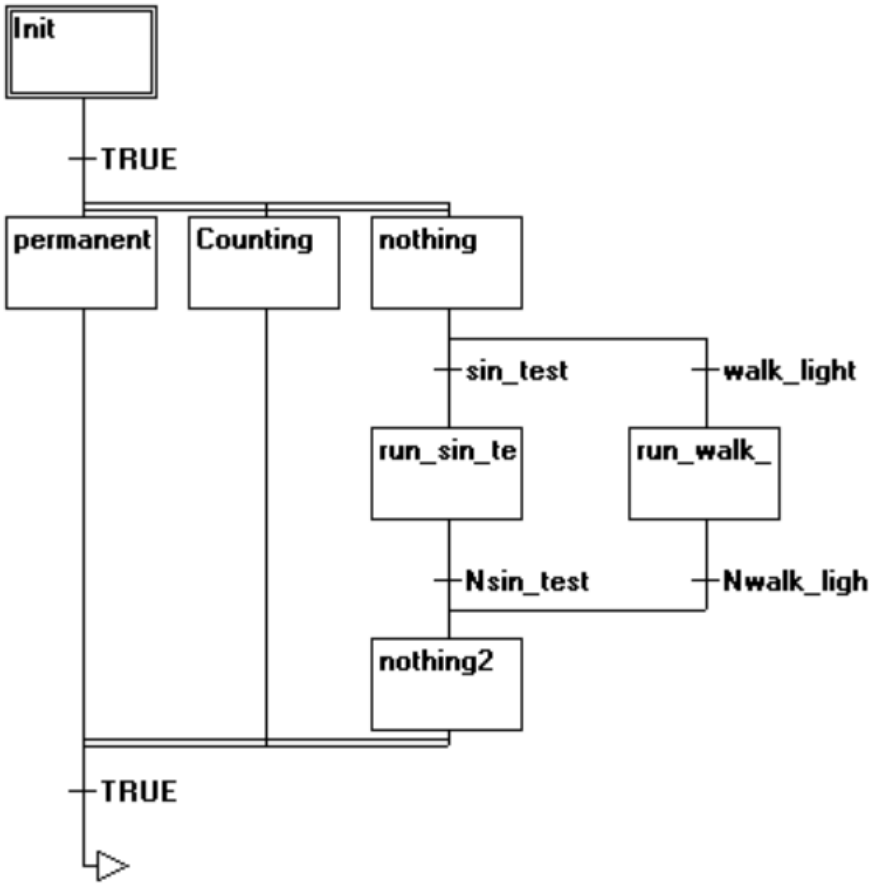

Figura 2.14: Exemplo Diagrama de Estados (9)

2.3.2 Biblioteca de Controlo de Movimento da PLCopen

De forma a ir o mais possível ao encontro de uma solução completamente normalizada, procurou-se, para além de seguir a norma IEC 61131, utilizar também a biblioteca de controlo de movimento da PLCopen.

Como referenciado em 2.5.1.6, o pacote Softmotion do Codesys é um exemplo da integração destas soluções na norma, já que é totalmente baseado na biblioteca de controlo de movimento que a PLCopen disponibiliza (10).

A PLCopen é uma organização mundial, independente de produtos ou vendedores, cuja missão consiste na resolução de tópicos relacionados com a programação de controlo, de forma a dar suporte ao uso das normas internacionais neste campo, entre as quais a norma 61131-3, mencionada em 2.3.

Foca-se em criar uma maior eficiência no desenvolvimento de software. Isto é feito através da definição de soluções e extensões baseadas nas normas, como é exemplo a biblioteca de controlo de movimento que a PLCopen dispõe, o que permite tornar o software o mais independente possível do hardware, assim como estender a reusabilidade de código e o acoplamento de ferramentas de software externas (10).

O conjunto de especificações do controlo de movimento da PLCopen divide-se em 6 partes (apesar de na versão 2.0, de 2011, se terem fundido as partes 1 e 2). Estas são (10):

- **Parte 1 e 2** — A parte 1 consiste no núcleo deste documento e especifica a biblioteca de blocos funcionais no que toca a funcionalidades relativas a eixos singulares, grupos de eixos, funções administrativas e um diagrama de estados de forma a providenciar o utilizador com um conjunto de comandos que são independentes da arquitetura do sistema subjacente. Por sua vez, a segunda parte é uma adição à primeira e não deve ser lida como um documento autónomo, visto que adiciona informação ao diagrama de estados e foca-se em transições e blocos funcionais adicionais.

- **Parte 3** — Define as diretrizes de utilização e demonstra alguns exemplos reais de utilização, de forma a que o utilizador consiga criar os próprios blocos funcionais e, consequentemente, a sua própria biblioteca dedicada a áreas especificas de aplicação.

- **Parte 4** — Foca-se na coordenação de movimento multi eixo no espaço 3D, mais especificamente no agrupamento de eixos e no seu movimento sincronizado.

- **Parte 5** — Cobre tudo o que abrange funções de *Homing*, incluindo uma definição geral do software e uma especificação de funções adicionais relacionadas.

- **Parte 6** — Procura otimizar a programação e integração de dispositivos e sistemas hidráulicos e pneumáticos ao definir blocos funcionais que aplicam o standard PLCopen e a metodologia modular.

Na figura 2.15 é possível verificar um exemplo de um bloco funcional definido na biblioteca, que permite o movimento discreto do eixo até uma posição definida.

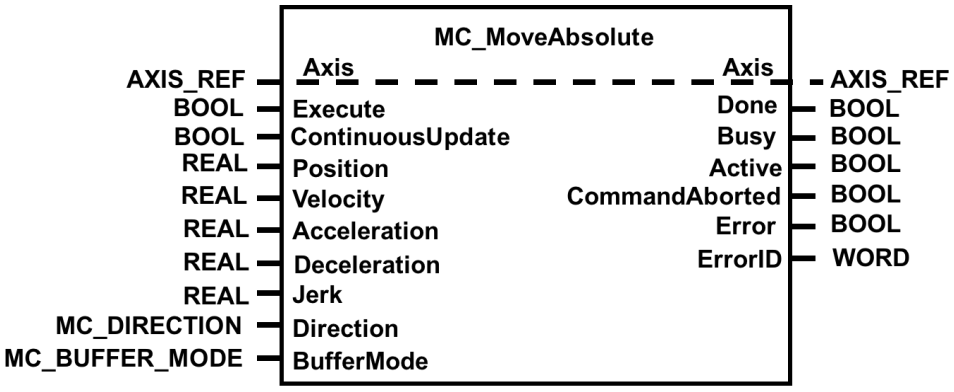

Figura 2.15: Exemplo de um Bloco Funcional de controlo de movimento

Este bloco funcional, chamado 'MoveAbsolute', tem representados à esquerda os *inputs* e os *outputs* à direita, ambos com os respetivos tipos de dados.

O que difere os blocos funcionais como o 'MoveAbsolute' presentes nesta biblioteca dos restantes é a primeira entrada 'Axis' que é igualmente saída do bloco funcional. Esta diferença prende-se com o *Data Type* 'Axis_Ref', que providencia a referência do eixo em questão. Ao usar apenas a referência do eixo e os parâmetros de entrada relevantes, pretende-se encapsular algumas tecnicidades da interface real com a drive de forma a permitir um uso modular dos blocos (10).

2.4 Requisitos da instalação

De forma a projetar os componentes a adicionar ao sistema é previamente necessário analisar que requisitos são impostos para o desenvolvimento do sistema. Assim é necessário definir:

- **Tipo de Controlador** — A primeira etapa passará por inferir a melhor opção entre um PLC convencional ou um softPLC.
- **Software de Desenvolvimento** — Após decidir o tipo de PLC, é agora necessário escolher o ambiente de desenvolvimento a utilizar podendo-se optar por softwares proprietários dos PLC's físicos ou por softwares de desenvolvimento de softPLC's. Em qualquer dos casos, é requerido que estes sejam complacentes com a norma IEC 61131.
- **Linguagens de Programação** — De forma a integrar o pós-processador no software MetroID é requerido que a programação seja feita numa linguagem abrangida pela *Framework* '.NET'.

Em relação a controladores lógicos programáveis existem várias soluções disponíveis no mercado, seja a nível de PLCs físicos como Siemens, Schneider, Omron, entre outros ou a nível de softPLCs, que consistem em pacotes de software que permitem simular o comportamento dos PLCs físicos num computador.

Como tal, tendo em conta que esta dissertação se trata de um estudo prévio para uma futura aplicação, optou-se por um softPLC, uma solução mais flexível, com um custo inferior e mais facilmente migrada para um controlador físico, caso necessário.

Em termos de programação de softPLCs existem vários ambientes de desenvolvimento integrados complacentes com a norma IEC 61131-3, tais como o Codesys da 3S-Smart Software Solutions GmbH , o ISaGRAF da Rockwell Automation ou o Unity Pro, da Schneider Eletric.

Devido ao seu software intuitivo, à facilidade de manutenção que o seu sistema de controlo de versão permite e a também oferecer uma maior liberdade ao utilizador com a sua linguagem gráfica própria *Continuous Function Chart*, o Codesys foi o software escolhido para o desenvolvimento do controlador.

Por sua vez, para o desenvolvimento do pós-processador, recorreu-se à plataforma de desenvolvimento Visual Studio e à linguagem de programação C#, desenvolvidos pela Microsoft.

Esta opção prendeu-se com o facto de toda a aplicação do MetroID ter sido desenvolvido nesta plataforma e linguagem pois, apesar de ser possível o desenvolvimento de qualquer linguagem abrangida pela *Framework* .NET, utilizar a mesma linguagem permite uma facilidade de integração inigualável.

2.5 Ferramentas de Desenvolvimento

2.5.1 Ferramenta de Desenvolvimento do Controlador

Acrónimo para *COntroller DEvelopment SYStem*, consiste numa aplicação de software que permite programar um PLC independentemente da marca e das especificações do controlador.

Contudo, a programação é apenas uma das funções presentes no Codesys pois, como se pode verificar na figura 2.16, este divide-se em seis ferramentas:

- Engenharia
- Visualização
- Segurança
- Runtime
- Redes de Campo
- Controlo de eixos e CNC

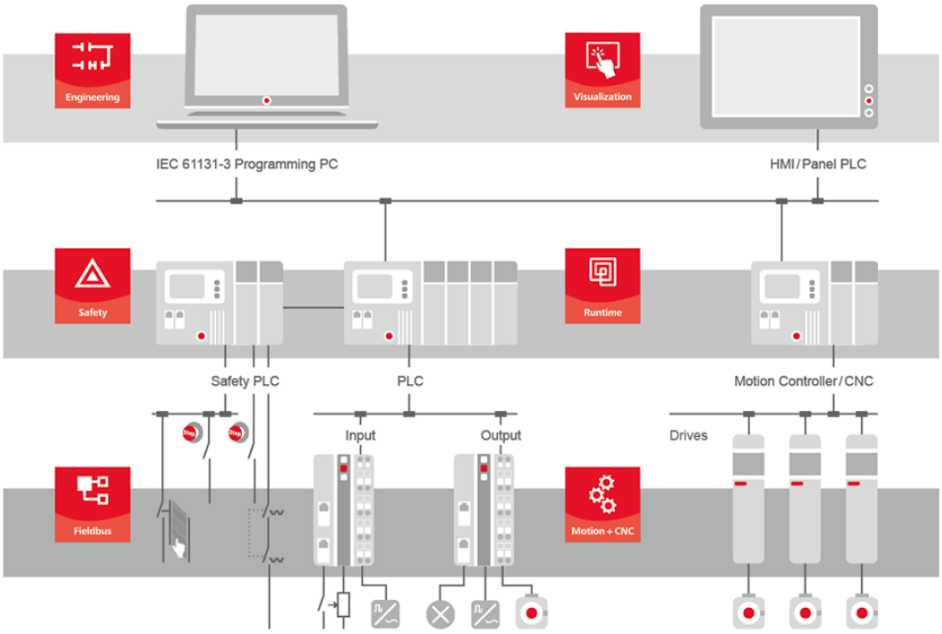

Figura 2.16: Potencialidades do Codesys (11)

2.5.1.1 Engenharia

Baseado na parte 3 da Norma IEC 61131, explicitada em 2.3, o Codesys oferece uma variedade de soluções de engenharia *user friendly* e com diferentes caraterísticas, explicitadas no subcapítulo 2.3.1, de modo a abranger uma maior panóplia de problemas que possam eventualmente surgir.

2.5.1.2 Visualização

Com a disponibilidade de um editor gráfico de HMI's é possível desenvolver ecrãs de visualização diretamente na plataforma, com base na norma IEC 61131. Desta forma, consegue-se reduzir a complexidade a nível da criação da HMI e a nível da utilização de variáveis que, não sendo necessário exportar para o PLC e importar para o HMI torna a sua atualização direta, pois a visualização é compilada e verificada em simultâneo com a aplicação.

2.5.1.3 Segurança

No que toca à segurança, a plataforma permite integrar uma grande diversidade de funcionalidades utilizando uma só ferramenta. As soluções integradas SIL 2 e SIL3 de segurança na norma IEC 61131-3 Codesys oferecem total funcionalidade requerida para soluções de automação seguras.

2.5.1.4 Runtime

De forma a programar e desenvolver um dispositivo de acordo com a norma IEC 61131, a plataforma dispõe do sistema *runtime* "Codesys Control SoftPLC", que converte qualquer computador ou sistema embebido num controlador industrial complacente com a norma IEC 61131. Além disso, o sistema *runtime* contém também funcionalidades *add-on* de forma a habilitar o controlador a comunicar com outros componentes de automação. Dessa forma, não há necessidade para o utilizador de se preocupar com o sistema *runtime* caso esteja a utilizar um controlador compatível, pois todos os equipamentos listados no Diretório de Dispositivos Codesys (12) estão equipados com este sistema e prontos a ser utilizados e programados sem qualquer configuração adicional.

2.5.1.5 Redes de Campo

Em termos de redes de campo, o Codesys dispõe de vários protocolos já implementados e integrados na plataforma, nomeadamente:

- PROFIBUS
- PROFINET
- EtherCAT
- CANopen
- J1939
- EtherNet/IP
- Sercos
- AS-i
- Modbus
- IO-Link
- BACnet
- IEC 61850

Ao disponibilizar a possibilidade de configuração de parâmetros de comunicação dentro do Codesys, consegue-se diminuir o número de erros que ocorrem em termos de compatibilidade entre software de programação e plataformas de comunicação.

2.5.1.6 Controlo de eixos e CNC

Através do pacote Softmotion, é possível um controlo de posição exato de sistemas que vão desde o mais simples, de um só eixo, até à interpolação de um sistema CNC multidimensional. Ao contrário dos sistemas clássicos de controlo de movimento, a configuração do controlo de movimento é feita de forma intuitiva através de POU's (Program Organization Units), totalmente independentes da rede de comunicação e do PLC utilizado.

Esta secção do Codesys é totalmente baseada na biblioteca de controlo de movimento da PLCopen, descrita em 2.3.2.

2.5.2 Ferramenta de Desenvolvimento do Pós-Processador

Para o desenvolvimento do pós-processador que gera o código G correto de acordo com os *inputs* inseridos no MetroID, recorreu-se à linguagem de programação orientada a objetos C# e ao IDE (*Integrated Development Environment*) Visual Studio, da Microsoft.

2.5.2.1 MetroID

O MetroID é uma plataforma modular de software de programação *offline* para equipamentos industriais robotizados e máquinas CNC que comporta uma interface gráfica 2D e 3D. Tendo sido desenvolvida centrada na tecnologia de corte a plasma, suporta todas as grandes marcas de robots, tais como FANUC, Motoman, ABB, Kuka e COMAU, assim como possui parâmetros de plasma predefinidos para marcas especializadas em equipamentos de corte a plasma como Hypertherm, ESAB ou Kjellberg. Contudo, a arquitetura do MetroID é ajustável a outras tecnologias de corte como oxycorte e laser.

Em suma, o MetroID controla e computa as funções automáticas da máquina e dispositivos associados como: geração de trajetórias de corte, posicionamento, controlo de altura e controlo do processo de corte.

Sendo uma plataforma modular, está dividida em quatro grandes componentes. Em primeiro lugar, o *DSTV Editor*, uma interface gráfica que processa ficheiros DSTV. Permite também a criação e edição de ficheiros DSTV e tem o principal propósito de dar ao operador de chão de fábrica a habilidade de editar ou criar ficheiros previamente produzidos por outros softwares como Tekla, SDS/2 ou Autodesk Revit.

De seguida, temos os outros 3 componentes com funcionamentos similares mas aplicações distintas: o TubeCUT para corte a plasma de tubos, o PlanCUT para corte a plasma de chapas metálicas e o BeamCUT para corte a plasma de vigas.

Esta dissertação debruçar-se-á sobre a integração do MetroID BeamCUT, representado na figura 2.17, com equipamentos de corte e furação de vigas.

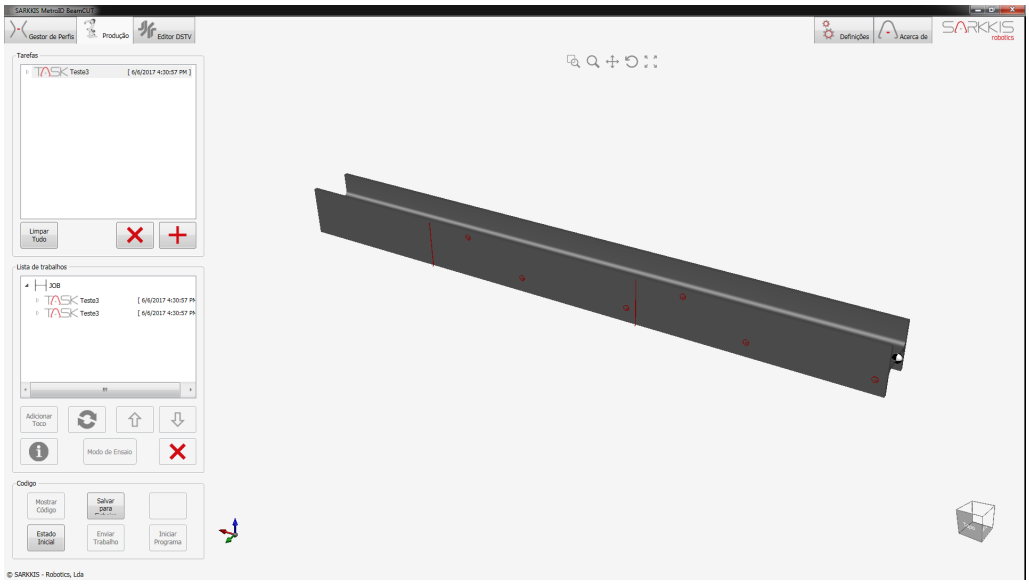

Figura 2.17: Software MetroID

O MetroID BeamCUT é o software derivado que integra o algoritmo específico de geração de trajetórias para perfis estruturais abertos (I, U, L).

Partindo de ficheiros de detalhe de aço estrutural (DSTV ou IFC) o MetroID BeamCUT gera programas para robô altamente eficientes para corte de vigas prontos a correr. O MetroID inclui já algoritmos avançados de planeamento de percursos que evitam colisões com a viga, o que significa que com apenas um clique é possível passar do software para a produção.

2.5.2.2 Pós-Processador

Como explicitado em 2.5, foi feita a opção de programar o pós-processador em C#.

O C# foi lançado em Janeiro de 2002 pela Microsoft, juntamente com a plataforma de desenvolvimento .NET. É uma linguagem de programação orientada a objetos baseada em C++ e Java e foi pensada de forma a dar resposta a problemáticas que exigiam a fusão de características de ambas. Assim, o C# consiste numa solução com um propósito mais universal e generalizado (13).

Foi criado com o propósito de providenciar uma linguagem para desenvolvimento de software baseado em componentes, mais especificamente para desenvolvimento na *framework* .NET. Nesta *framework*, componentes provenientes de várias linguagens como por exemplo C++ ou VB.NET, podem ser facilmente associados, deixando de haver necessidade de construir uma aplicação recorrendo a uma só linguagem de programação (13).

O pós-processador propriamente dito passará pelo desenvolvimento de uma *Dynamic-Link Library* (DLL) que será integrada na aplicação MetroID.

Uma DLL consiste numa biblioteca que contém código e dados e as grandes vantagens da sua utilização prendem-se maioritariamente com 3 fatores (14):

- **Uso eficiente de recursos** — Uma DLL permite a utilização da mesma biblioteca por dois programas diferentes, evitando assim a duplicação de código carregado no disco e na memória física, melhorando a performance do programa e do sistema operativo.
- **Promove arquitetura modular** — Permite um desenvolvimento mais fácil de programas modulares, como é exemplo o MetroID, no qual o mesmo programa possui diferentes aplicações consoante a aplicação desejada (BeamCUT, TubeCUT ou PlanCUT).
- **Facilidade de desenvolvimento e instalação** — Caso seja necessária a atualização de um componente, essa atualização pode apenas ser feita a uma DLL específica, sem afetar minimamente os outros componentes.

Como neste caso se pretende um componente que traduza a informação proveniente do MetroID e não a alterar intrinsecamente, fará todo o sentido desenvolver uma DLL que obtenha a informação obtida no MetroID e a traduza para código G.

2.6 Sumário

Neste capítulo foram abordadas as características principais das células de corte e furação de vigas, assim como conceitos importantes à compreensão do problema, nomeadamente o software MetroID, a norma IEC 61131-3, a biblioteca de controlo de movimento da PLCopen, a definição de DLL e as ferramentas de desenvolvimento Codesys e Visual Studio.

Capítulo 3

Perspetivas de Solução

Neste capítulo, pretende-se fazer um projeto generalizado do sistema, englobando os componentes necessários, a arquitetura do sistema e o fluxo de informação.

3.1 Componentes do Sistema

Para o projeto real, sem ser no contexto da simulação, serão necessários diversos componentes, entre os quais:

- **Controlador de segurança** — Ligado às barreiras de segurança, permite ativar o estado de emergência e desligar o sistema caso haja alguma situação de perigo eminente, impedindo assim potenciais incidentes.

- **Controlador Codesys** — Componente principal do sistema, apresenta ligações, diretas ou indiretas, a todas as restantes componentes. Integra-se também neste componente o pacote Softmotion, que é responsável pelo controlo de movimento e segue a biblioteca de controlo de movimento da PLCopen.

- **MetroID** — Software que, conjuntamente com o pós processador a desenvolver, trata da geração automática de código G para corte e furação de vigas, sendo assim fornecedor do código a montante da aplicação a desenvolver.

- **Simulador** — Fornecido pela SARKKIS, funciona a jusante do controlador e permite a visualização de resultados em 3D, além da realização de testes da solução desenvolvida.

- **Atuadores e Drives** — Conectados ao Softmotion, permitem atuar o sistema. É necessário ter em conta que drives serão escolhidas, pois nem todas são compatíveis com o Softmotion.

- **Módulo de I/O** — Transfere a informação do controlador de segurança, em modo Input-Output, para o CPU onde se encontra o Codesys, por Ethernet/IP.

- **HMI** — Permite a visualização e o comando por parte do utilizador. Através da HMI, o utilizador poderá comandar manualmente os eixos da máquina, assim como mandar executar o programa CNC e visualizar o estado atual da execução.

3.2 Arquitetura do Sistema

Tendo em conta os componentes necessários, o esquema da arquitetura do software do sistema será o seguinte, retratado na figura 3.1.

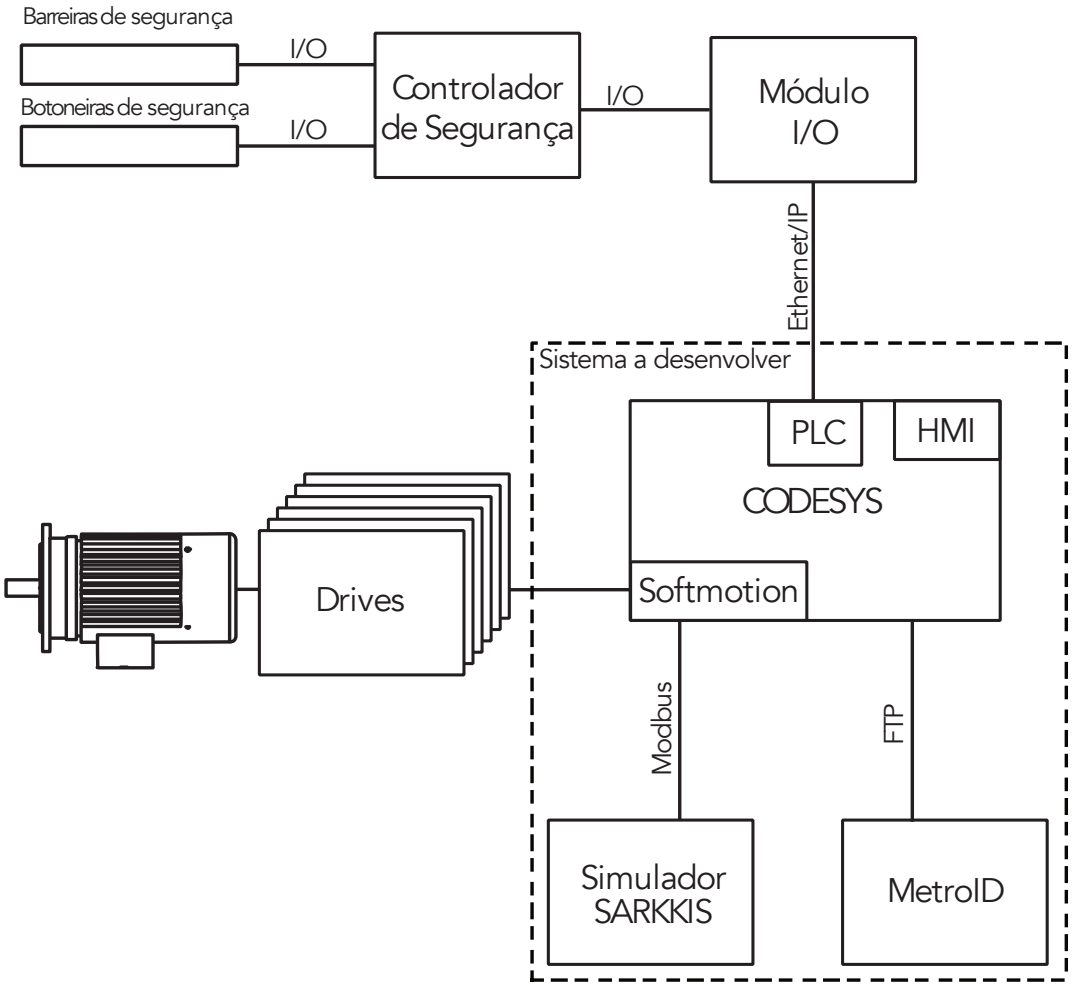

Figura 3.1: Arquitetura proposta do software do sistema

Como núcleo do sistema, o controlador Codesys comunica com todos os outros componentes do sistema, como se pode verificar na imagem acima.

Fazendo analogia ao que é dito em 2.2 e mostrado na figura 2.8, é no Codesys que se encontram os 3 principais componentes dum sistema CNC genérico: a HMI que será a visualização criada, o PLC que será a sequenciação de comandos por parte do controlador e o NCK, que será o controlo de movimento por parte do pacote Softmotion.

O controlador de segurança desligará o sistema em caso de emergência. Além disso, envia a informação para o controlador central que o sistema foi desligado, de forma a ser salvo o estado de execução do programa.

A transmissão de informação entre o controlador central e o controlador de segurança será efetuada por meio de um módulo de input/output ligado ao controlador através de Ethernet/IP, um protocolo de redes industriais que adapta o CIP (Common Industrial Protocol) ao standard EtherNet, permitindo comunicação caso o sistema entre em emergência.

A Ethernet/IP recorre a todos os protocolos de EtherNet tradicional incluindo o TCP (Transport Control Protocol), o IP (Internet Protocol) e as tecnologias de sinalização e acesso ao meio físico encontradas em todas as interfaces de rede EtherNet. Assim, como se baseia na tecnologia standard EtherNet, este protocolo é compatível com todos os dispositivos que utilizam EtherNet padrão (15).

A nível do MetroID irá existir comunicação com o Codesys a nível da transferência do ficheiro de texto que contém o código G. Este ficheiro será transferido por File Transfer Protocol (FTP), um protocolo de transferência de arquivos, incluído na suite do Internet Protocol, usado para transferir ficheiros de computador entre um cliente e um servidor numa rede de computadores que corre em TCP/IP (16).

Por sua vez, a comunicação entre o controlador e o simulador será feita através de Modbus TCP/IP, embora, neste caso, seja feita através de registos e não de entradas/saídas digitais. Esses registos transmitirão a posição atual dos atuadores de forma a se visualizar e simular o comportamento da máquina.

Como referenciado em 2.5.1.5, o Codesys abrange vários protocolos de comunicação. A escolha do Modbus prendeu-se com o facto de este estar também totalmente incorporado na plataforma de simulação, sendo apenas necessário referenciar entre o controlador e o simulador o significado de cada registo. Caso se tivesse optado por outro protocolo integrado no Codesys, como PROFINET por exemplo, seria necessário implementar a comunicação no simulador, o que se revela desnecessário neste caso.

A comunicação Modbus segue uma estrutura definida que contém o endereço do escravo, a função a ser executada, uma quantidade variável de dados complementares e uma verificação de consistência de dados (CRC) (17).

O protocolo Modbus TCP/IP baseia-se em relações cliente/servidor, mantendo as mesmas funções do Modbus RTU mas através de endereçamentos e CRCs diferentes, além de que utiliza o meio físico Ethernet. Permite também trocas de informação mais completas, rápidas e a maiores distâncias. Suporta proteção do servidor por gateway e proporciona serviços de rede mais abrangentes (18).

Em Modbus TCP/IP, a comunicação faz-se através de dois tipos de dados: Digitais, de 1 bit, e registos, de 16 bits.

Algumas das funções mais importantes e utilizadas são as seguintes:

0x01	Ler estado On/Off de uma saída digital (bit)
0x02	Ler estado On/Off de uma entrada digital (bit)
0x03	Ler conteúdo do registo (16 bits) de entrada
0x04	Ler conteúdo do registo (16 bits) de saída
0x05	Escrever estado de uma saída digital
0x06	Escrever conteúdo de um registo de saída
0x15	Escrever estado de múltiplas saídas digitais
0x16	Escrever conteúdo de múltiplos registos de saída

Tabela 3.1: Funções presentes no protocolo Modbus

3.3 Fluxo de Informação

Após ter uma ideia da arquitetura do sistema e das comunicações a realizar, é agora possível delinear o modo como os dados serão transferidos e tratados, esquematizado no fluxo de informação da figura 3.2.

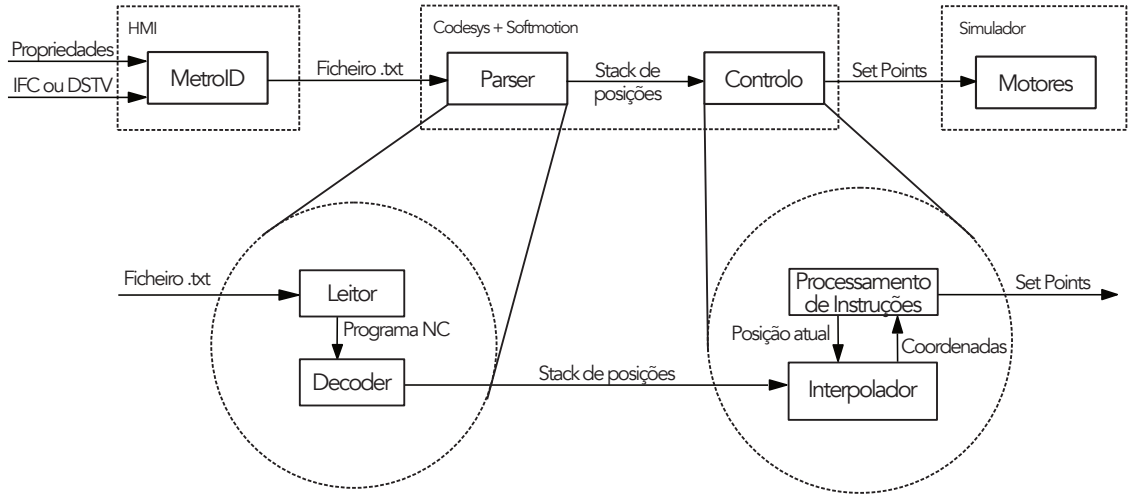

Figura 3.2: Fluxo de informação

O fluxo de informação inicia-se no MetroID, onde o utilizador importa o ficheiro IFC ou DSTV para a criação da viga e as propriedades da máquina necessárias para as operações a realizar.

Após a importação do ficheiro, o MetroID faz o tratamento da informação e gera uma quantidade de operações a realizar, que vai posteriormente ser interpretada pelo pós-processador que ,por sua vez, tratará de gerar um ficheiro de texto contendo o código G com as trajetórias a alcançar.

De seguida, após a geração do código G por parte do pós-processador, é enviado para o parser, já no Codesys, um ficheiro .txt com as instruções em código G nele contidas.

No entanto, existiam outras opções para comunicar a informação proveniente do MetroID, como, por exemplo, XML (Extensible Markup Language). Apesar disso, o ficheiro de texto (.txt) foi o escolhido devido à fácil utilização desse tipo de ficheiros por parte do Codesys.

O Parser tem a finalidade de ler o ficheiro .txt e retornar um *stack* de posições possíveis de ser lidas pelo controlador. Para tal, este processo é feito em dois passos intermédios. Primeiro, através do leitor, traduz o ficheiro de texto num programa NC propriamente dito. De seguida, pega nesse programa NC e transforma-o numa estrutura que contém todas as posições, como desejado.

Na parte do controlador, o interpolador recebe a informação de todas as posições a atingir, assim como das velocidades pretendidas. Com esta informação, é possível dar agora início ao processamento de instruções e consequente atuação dos motores. De modo a ser feito um controlo eficiente, é fornecida ao interpolador a posição atual dos motores para referenciação, contribuindo assim para um movimento regular dos motores.

Assim, o interpolador, tendo em conta a posição atual dos atuadores, atualiza de forma cíclica as coordenadas x, y e z, que são interpretadas pelo controlador e posteriormente enviadas para as drives corretas, consoante os comandos ativados.

De forma a emular o funcionamento da célula, estes comandos serão transmitidos ao simulador, onde será gerado o movimento.

3.4 Sumário

Neste capítulo abordou-se de uma forma generalizada como é que será solucionado o problema descrito nos dois capítulos anteriores, além de se ter também abordado em maior pormenor os componentes necessários para esta solução, a arquitetura do sistema e o fluxo de informação.

Capítulo 4

Desenvolvimento do Pós-Processador

Neste capítulo abordar-se-á todo o processo de desenvolvimento do pós-processador, mais concretamente, o seu funcionamento intrínseco, a sequenciação de tarefas e os códigos G e M incluídos.

4.1 Funcionamento do MetroID

De forma a que o desenvolvimento do pós-processador seja feito de forma correta, é importante conhecer o funcionamento do MetroID. Desse modo, é necessário analisar o fluxo de informação entre o MetroID e o pós-processador, retratado na figura 4.1.

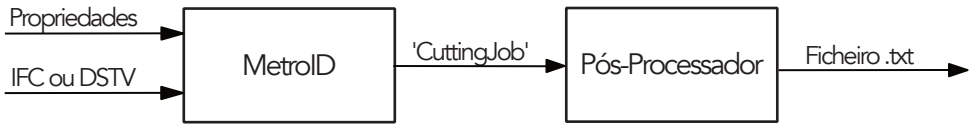

Figura 4.1: Fluxo de informação entre MetroID e pós-processador

Assim, analisando os *inputs* do software é importado um ficheiro IFC ou DSTV, que contem todas as informações relativas às dimensões da viga e operações a realizar. Além disso, o utilizador insere no software todas as propriedades da máquina relevantes à execução do programa.

Dentro do MetroID, a informação contida nos ficheiros IFC ou DSTV é fragmentada por operações e são geradas as trajetórias a fazer pela máquina industrial, denominadas 'Tasks'. Essas 'Tasks' são sequenciadas de acordo com as especificações de cada máquina, ditadas pelas propriedades inseridas pelo utilizador no início da operação e inseridas no objeto de memória 'CuttingJob' que será o *input* do pós-processador.

Por sua vez, o pós-processador, tendo recebido o objeto 'CuttingJob', trata de traduzir a informação recebida em C# para um ficheiro de texto em código G.

De forma a entender o modo de funcionamento do pós-processador, é em primeiro lugar necessária a compreensão da estrutura de dados que este recebe, como retratado na figura 4.2.

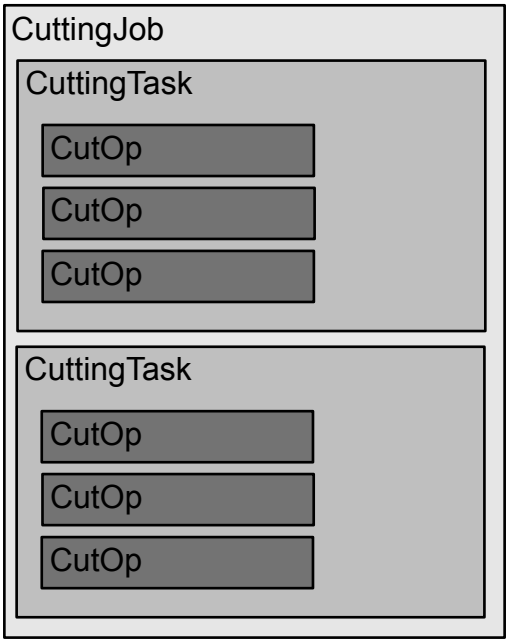

Figura 4.2: Estrutura de dados traduzida pelo pós-processador

Começando pelo nível mais elementar, cada objeto 'CutOp' representa uma operação a fazer na viga, seja um corte ou uma furação.

Por sua vez, cada 'CuttingTask' é composta por vários objetos 'CutOp', que representam as operações a realizar que, neste caso, são as furações e os cortes necessários para o fabrico de uma viga.

Tendo em conta que é possível fabricar mais do que uma viga a partir de uma só viga, o 'CuttingJob' engloba todas as 'CuttingTask', permitindo assim um fabrico mais rápido de várias vigas e um menor desperdício de material.

Em suma, o objeto 'CuttingJob' trata-se de uma lista que retrata as operações a fazer pelo máquina ao longo de toda a viga de forma a fabricar as vigas projetadas, sendo composto por uma ou mais listas 'CuttingTask'.

4.2 Estrutura do pós-processador

O pós-processador desenvolvido tem uma arquitetura modular, o que permite que o seu funcionamento possa ser facilmente descrito pelas duas classes principais que o constituem, conforme ilustrado na figura 4.3.

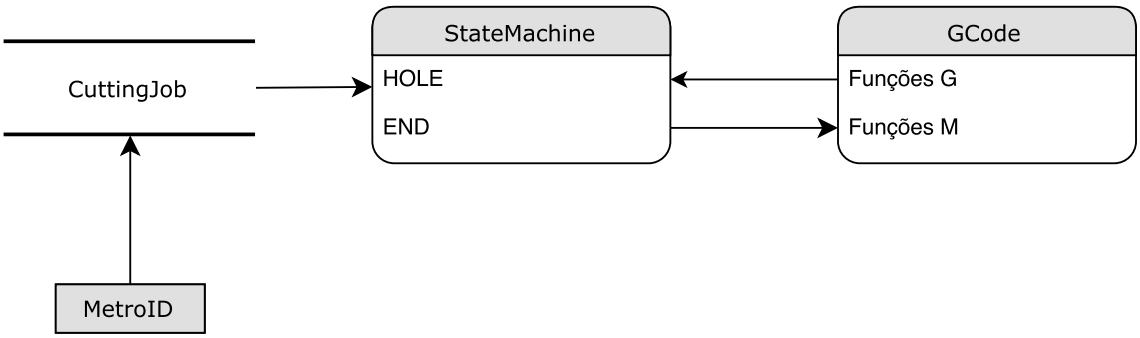

Figura 4.3: Diagrama organizacional do pós-processador

A primeira, 'GCode' consiste numa biblioteca de todas as funções disponíveis no código G e descritas nos subcapítulos abaixo. Permite a chamada das funções por parte da classe 'StateMachine'.

Por sua vez, a classe 'StateMachine' trata de receber o objeto 'CuttingJob' do MetroID, segmentar e ordenar a informação relativa a cada 'CutOp', como descrito em 4.3. De seguida interpreta cada operação e determina se se trata de uma operação de furação ('HOLE') ou de corte ('END'). Consoante o tipo de operação, vai buscar à biblioteca 'Gcode' as funções necessárias para definir um certo conjunto de ações pré-definidas.

4.3 Sequenciação de Operações

Por conseguinte, o que o pós-processador faz é analisar, ordenar e processar a informação de cada objeto 'CutOp' consoante o tipo de operação a realizar. Esta ordenação de operações é feita de maneira a que as operações sejam realizadas da forma mais eficiente e rápida possível.

O fluxo de funcionamento do pós-processador, representado na figura 4.4 tem então de ser feito de forma a que o produto final, isto é, o código G, reflita estas caraterísticas.

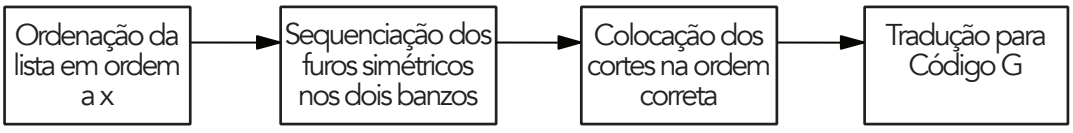

Figura 4.4: Fluxo de funcionamento do pós-processador

Em primeiro lugar, todos os objetos da lista são ordenados em relação ao eixo x, permitindo a sequenciação correta de todos os furos.

De seguida, elimina-se a duplicação de furos simétricos (mesmo x e y) nos dois banzos e adiciona-se a informação de simetria para assegurar a tradução correta posteriormente.

Em terceiro lugar, devido às restrições levantadas em 2.1 referentes ao layout da célula, é feita a reordenação dos cortes, de forma a que as operações sejam feitas de forma continua em relação ao eixo OX da viga e permitindo que o batente apenas se desloque numa direção.

Finalmente, é feita a tradução dos objetos 'CutOps' presentes na lista 'CuttingTask' para código G, complacente com as instruções descritas nos próximos subcapítulos, de forma a que a máquina execute os movimentos em conformidade com o pretendido.

4.4 Código G criado

O código G é uma linguagem de programação de máquinas por endereçamento de palavras. Sendo uma linguagem de comando de movimentos, permite o controlo e automatização de operações. Em suma, as instruções do código G informam o controlador das posições a atingir, do caminho a percorrer e de quão rápido deve ser percorrido esse caminho.

Apesar da existência de normalização deste processo (ISO 6983), a norma não garante que todos os programas de controlo de máquinas interpretem das mesma forma o mesmo código.

4.4.1 Funções G

Assim, de forma a que houvesse uma interpretação correta por parte do controlador, recorreu-se ao manual presente no Codesys para a especificação das Funções G a ser interpretadas. Assim, incluiu-se as funções G presentes na tabela 4.1.

G00	Posicionamento rápido
G01	Interpolação linear
G02	Interpolação circular no sentido horário (CW)
G03	Interpolação circular no sentido anti-horário (CCW)
G06	Interpolação parabólica
G08	Segmento de elipse no sentido horário
G09	Segmento de elipse no sentido anti-horário
G40	Desativação da compensação do raio da ferramenta
G41	Ativação da compensação do raio da ferramenta à esquerda da peça
G42	Ativação da compensação do raio da ferramenta à direita da peça
G50	Desativação das funções de arredondamento e alisamento das velocidades no percurso
G51	Ativação da função de alisamento das velocidades no percurso
G52	Ativação da função de arredondamento das velocidades no percurso
G53	Desativação da função de offset coordenadas
G54	Definição do offset para coordenadas absolutas
G55	Incremento do offset
G56	Definição do offset para coordenadas relativas a outro ponto
G98	Interpretação das coordenadas como absolutas
G99	Interpretação das coordenadas como relativas

Tabela 4.1: Funções G passíveis de ser interpretadas pelo controlador

4.4.2 Funções M

Visto que foi necessário criar um interpretador de funções M de origem, pois o Codesys não possui um interpretador genérico de funções M, havia liberdade total no que toca à criação de funções M por parte do pós-processador. Mesmo assim, optou-se por recriar algumas funções M genéricas do controlo numérico dos processos de fresagem e torneamento, presentes na tabela 4.2.

M03	Ativação da rotação da árvore no sentido horário
M04	Ativação da rotação da árvore no sentido anti-horário
M05	Desativação da rotação da árvore
M06	Mudança automática de ferramenta
M07	Ativação do liquido refrigerador em modo spray
M08	Ativação do liquido refrigerador em modo liquido
M09	Desativação do liquido refrigerador
M30	Fim de programa
M98	Chamada de subprograma
M99	Fim de subprograma

Tabela 4.2: Funções M genéricas

Houve também necessidade de incluir no processo de pós-processamento funções M para executar tarefas específicas ao corte e furação de vigas, como as indicadas na tabela 4.3.

M40	Seleção da broca
M42	Ativação da serra
M43	Seleção do perfil da viga
M44	Inserção das dimensões da viga

Tabela 4.3: Funções M criadas

Assim, são passados juntamente com a função os parâmetros K e L, de forma a transmitir mais informação específica a cada função M. Para as funções M presentes na tabela 4.3 usou-se a seguinte especificação:

- **M40** — De forma a selecionar qual a broca a escolher, é declarada a mudança de broca com o comando M40. Contudo, é com o parâmetro K que esta é efetivamente escolhida. Por exemplo, caso exista um furo no banzo do lado esquerdo, será gerado o código 'M40 K1', na alma 'M40 K2' e no banzo direito 'M40 K3'. Além disso, é passado no parâmetro L o diâmetro do furo a executar.

- **M42** — Nos casos em que existe um corte, a viga é movida em X para a posição correta e é posteriormente ativada a serra com a função M42, sendo que é necessário especificar o ângulo do corte. Esta especificação é novamente feita através do parâmetro K. Exemplificando, se existir um corte de 20º será gerado o código 'M42 K20'.
- **M43** — Através desta função é passada a informação relativa ao perfil da viga, alocada no parâmetro K. Assim gerar-se-ia o código 'M43' seguido de 'K1', 'K2' ou 'K3', caso se tratasse de uma viga de perfil 'I', 'U' ou 'L', respetivamente.
- **M44** — De forma análoga à função anterior, também é usado o parâmetro K para a transmissão de informação, neste caso da largura da viga. Contudo, neste caso é também usado o parâmetro L para transmissão da altura da viga. Assim, para uma viga de dimensões 160x120 seria gerado o código 'M44 K160 L120'

4.4.3 Parâmetros Adicionais

Além das funções G e M, existem também parâmetros a ter em conta no desenvolvimento do pós-processador. Estes parâmetros são passados conjuntamente com uma função G ou M específica e de acordo com o indicado na tabela 4.4.

D	Raio para correção das funções G40 a G42 Distância para arredondamento das funções G50 e G51
E	Definição da aceleração/desaceleração
F	Definição da velocidade de avanço
I	X do centro do círculo ou elipse das funções G02, G03, G08 e G09
J	Y do centro do círculo ou elipse das funções G02, G03, G08 e G09
K	Primeiro parâmetro passado nas funções M Z do círculo G02 e G03 em modo 3D Direção do eixo principal da elipse
L	Segundo parâmetro passado nas funções M
R	Raio usado em alternativa a I,J nas funções G02 e G03 Rácio entre o menor e o maior eixo da elipse das funções G08 e G09

Tabela 4.4: Parâmetros a inserir

4.5 Sumário

Em suma, foi abordado ao longo deste capítulo o modo como se alcançou o funcionamento atual do pós-processador, assim como todas as funções G e M incluídas na tradução entre os objetos do MetroID a montante e o código G a jusante.

Capítulo 5

Desenvolvimento do Controlador

Neste capítulo serão abordadas em maior detalhe as soluções desenvolvidas relativamente ao controlador, nomeadamente programas e blocos funcionais importantes ao funcionamento deste, a interface utilizador máquina e a sequenciação das tarefas.

5.1 Linguagens Escolhidas

As linguagens escolhidas para o desenvolvimento do controlador foram Controlo Contínuo de Funções, Texto Estruturado e Diagrama de Contactos.

Na maior parte dos casos, devido à facilidade de compreensão da sequenciação de comandos por terceiros, optou-se por **Controlo Contínuo de Funções**. Esta escolha prende-se também com o facto de esta linguagem permitir a execução de *loops* fechados e *feedback*, sem o uso de variáveis intermediárias, o que leva a um ganho em termos de memória e rapidez de execução.

Devido à sua flexibilidade em termos de soluções, optou-se por **Texto Estruturado** sempre que foi necessário haver tratamento de números, como no bloco funcional 'MCheck' em que se interpreta as funções M do código NC, pois o Controlo Contínuo de Funções é limitado neste campo.

Por fim, utilizou-se **Diagrama de Contactos** para o programa 'Main' que trata dos estados principais da máquina, descritos em 5.3.1.1. Esta opção foi feita devido ao seu determinismo e facilidade de visualização dos estados da máquina aquando da execução de operações.

5.2 Requisitos do Controlador

Antes de se desenvolver o controlador, é necessário ter em conta todos os requisitos que este terá de cumprir. Estes são:

- **Controlo de 9 eixos** — É necessário que o controlador coordene e faça o controlo de posição dos 9 eixos presentes na máquina de forma independente, sendo 2 por cada broca (Eixo Y e Z), 2 da serra (Eixo Z e de rotação) e um batente (Eixo X).

- **Comunicação com controlador de segurança** — Quando a emergência é acionada e a célula é desligada por parte do controlador de segurança, é necessário haver procedimentos a nível de software por parte do controlador central de forma a ser possível retomar a execução. Assim é requerido ao controlador que, nestes casos, interrompa o programa e salve o estado atual de execução, para posteriormente ser retomado.
- **Interpretação do código G criado pelo pós-processador** — É requerido que o controlador interprete corretamente todas as instruções especificadas no capítulo 4, de forma a serem gerados os comandos corretos.
- **Interface Homem Máquina** — É requerido que a HMI disponibilize certas funcionalidades, como visualização do estado das máquinas, visualização do estado de execução do programa ou comando manual e semiautomático de operações
- **Sequenciamento de operações** — De forma a se obter uma solução válida e normalizada, requere-se que o controlo de movimento seja feito de forma complacente com as especificações da biblioteca de controlo de movimento da PLCopen. Assim, entre outros requisitos, é necessário cumprir os diagramas de estados retratados nas figuras 5.1 e 5.2.

No que toca a controlo de movimento, foram consultadas as especificações da biblioteca de controlo de movimento da PLCopen, onde se encontram os seguintes diagramas de estados para controlo de um só eixo e de grupos de eixos, nas figuras 5.1 e 5.2 respetivamente.

Como se pode ver nas figuras, existem cuidados a ter na sequenciação de comandos, sendo que essa sequenciação será diferente caso se trate de um grupo de eixos, como os casos das brocas ou de um eixo independente, como o eixo OX do track.

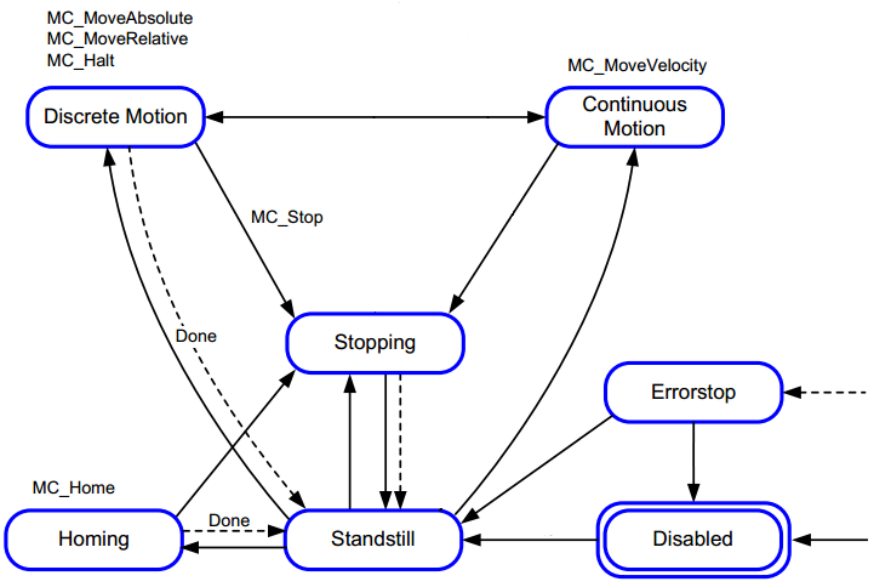

Figura 5.1: Diagrama de Estados para um só eixo (10)

Por exemplo, no diagrama acima, o controlo de movimento e a ativação dos blocos funcionais tem de ser feita de uma forma específica, cumprindo a sequenciação descrita no diagrama.

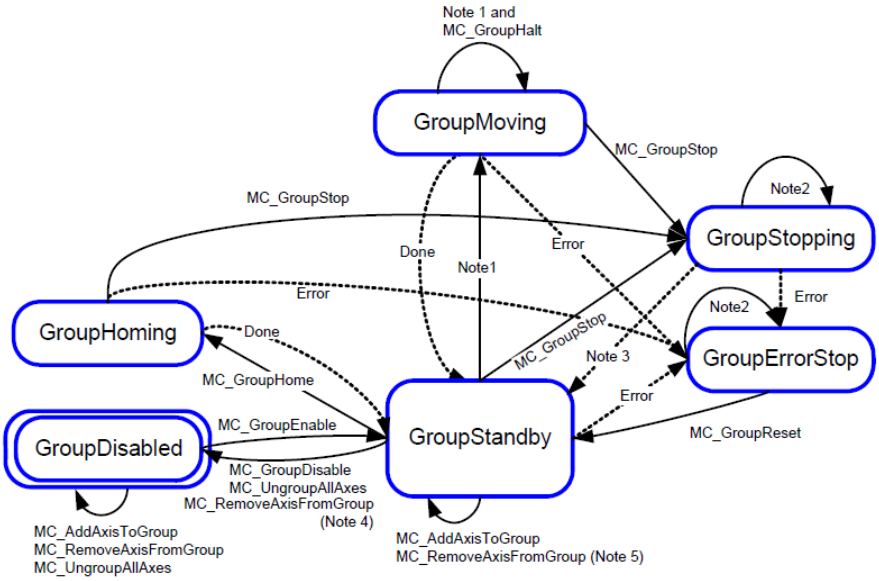

Figura 5.2: Diagrama de Estados para grupos de eixos (10)

Por sua vez, no segundo diagrama, onde se representa a sequenciação do controlo de movimento de múltiplos eixos, pode-se verificar a maior complexidade do diagrama, devido ao maior número de blocos funcionais disponíveis e cuidados a ter na coordenação de movimentos.

5.3 Soluções Implementadas

5.3.1 Programas

Como é necessário utilizar várias tarefas de forma a poder atualizar as variáveis em diferentes tempos de ciclo, foram criados diferentes programas. Além disso, devido à necessidade de modulação e flexibilização da aplicação, foi criado um programa para cada funcionalidade, seja a máquina de estados, a leitura do ficheiro de texto, a interpolação dos eixos ou o controlo manual da máquina.

5.3.1.1 Main

O programa 'Main' é, nada mais, nada menos do que a máquina de estados da aplicação, construida na forma de *Ladder Diagram*, mencionada em 2.3.1.3 e que tem como base o comportamento do Grafcet da figura 5.3, abaixo demonstrada:

O comportamento do software inclui a segurança, o ON/OFF do sistema e o comportamento geral da aplicação, divididos em 3 sub-Grafcets Modulares:

- **G00** — Assegura a segurança do sistema de forma a que, ao ser acionada a emergência é ativado o estado *Unsafe*, que força o estado X03 de G0 e desliga o sistema. Por sua vez, ao ser desativada a emergência e acionado o rearme, o sistema transita para o estado *Safe* que inicializa G0
- **G0** — Assegura o funcionamento correto do ligar e desligar do sistema. É inicializado em X00 (OFF) e só transita se o sistema não se encontrar em emergência e após a transição ascendente do interruptor 'PowerEnable'. Em X01 e X03 é ativado e desativado, respetivamente o bloco funcional 'Power', retratado em 5.3.3.1. Quando o sistema se encontra ligado, em X02, é inicializado G1, permitindo o funcionamento normal da aplicação.
- **G1** — Caracteriza o comportamento do sistema. Inicializa-se em X0 e permanece em espera até se ativar o comando de 'executeCNC' sendo necessário que o sistema esteja ligado, obviamente. Após entrar em X1, podem acontecer dois tipos de evento. Por um lado, se o programa CNC for completado ou abortado a meio da execução, o sistema volta ao estado inicial, em que está pronto para executar novamente outro ou até o mesmo programa CNC. Por outro lado, caso, a meio da execução, o sistema entre em estado de emergência (*Unsafe*), seja premido o comando 'Stop' ou a aplicação se desligue, o sistema entra no estado X2, em que é memorizado o ponto atual da execução do programa CNC, que será retomado quando o sistema voltar ao estado normal.

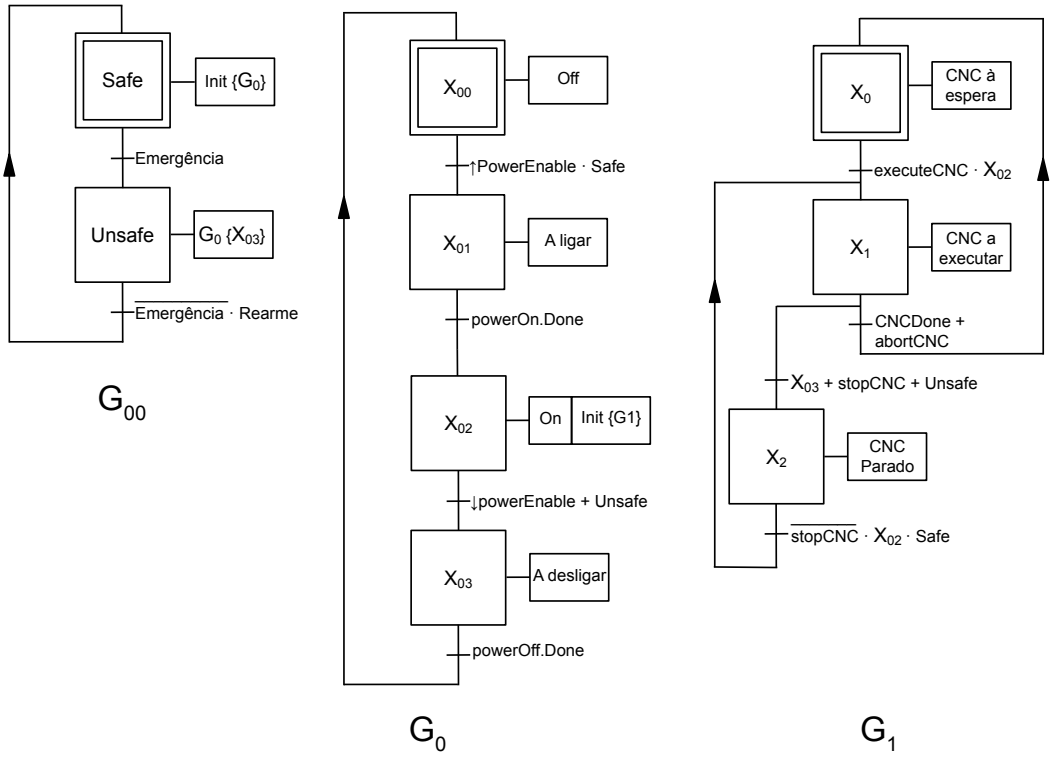

Figura 5.3: Grafcet comportamental do controlador

5.3.1.2 Path

No subprograma 'Path' é lido o ficheiro de texto transferido por FTP e transformado numa pilha (*stack*) de posições. Para tal, são executados três passos intermédios, representados na figura 5.4 por 3 blocos funcionais:

- **SMC_ReadNCFile** — Este primeiro bloco trata de descodificar o formato ASCII do ficheiro .txt para um programa CNC, passível de ser interpretado pelo 'SMC_NCDecoder'.

- **SMC_NCDecoder** — Este bloco funcional é usado para converter o código G do programa NC numa lista de objetos que seguem a estrutura GEOINFO do Softmotion, uma linha de código por ciclo. Este formato é o mesmo que o formato de *input* do interpolador.

- **SMC_CheckVelocities** — Por último, este bloco funcional faz um tratamento da lista de objetos acima mencionada, de forma a que seja assegurada a velocidade zero em mudanças de direção do trajeto do programa CNC. Visto que o programa vai ser adaptado para a máquina de corte e furação, que só executa movimentos ortogonais, este tratamento contribui para uma maior precisão do sistema.

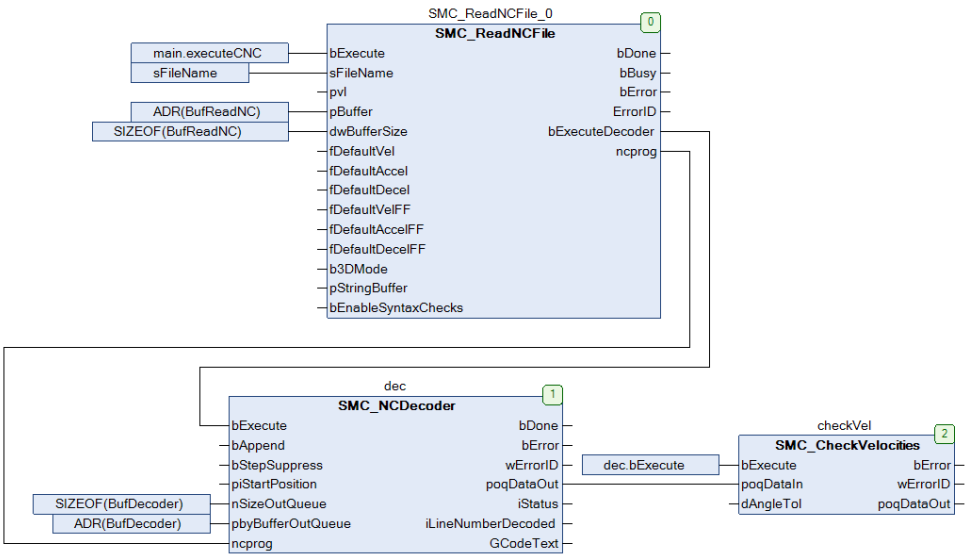

Figura 5.4: Programa de cálculo CNC do percurso

5.3.1.3 Interpolation

Este pode ser considerado o subprograma mais importante do controlador, pois é onde se efetua o controlo de movimento aquando da execução do programa CNC.

Para tal, o programa está dividido em duas partes, representadas na figura 5.5.

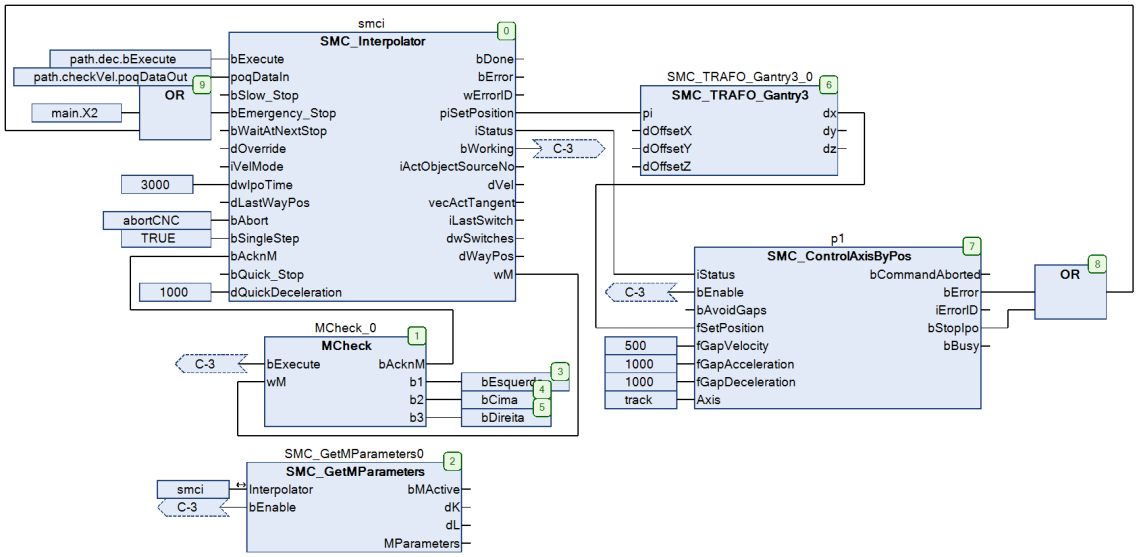

Figura 5.5: Versão simplificada para um só eixo do programa de interpolação

A primeira, trata de receber a informação tratada no 'Path' e interpolar as posições dos motores consoante os comandos atuados. Desta primeira parte fazem parte o interpolador 'SMC_Interpolator' e o bloco funcional MCheck conjuntamente com o 'SMC_GetMParameters'.

- **SMC_Interpolator** — Como o nome indica, é o componente principal do programa de interpolação. Recebe a trajétoria calculada no programa 'Path' e tem como *output* uma estrutura que contém as posições a atingir pelos motores.

- **MCheck e SMC_GetMParameters** — Estes dois blocos funcionais possuem a função de interpretar as funções M do código G. Ao longo da execução do programa, sempre que o interpolador recebe uma função M retorna-a como *output* e interrompe a execução até receber uma transição ascendente na entrada 'bAcknM'. Este output é apanhado pelo 'MCheck', que trata da realização do comando específico que essa função acarreta, tal como ligar/desligar a rotação da árvore ou escolher a broca a mover. Quando a ação é realizada, o 'MCheck' emite uma transição ascendente em 'bAcknM' que permite a continuação da execução do programa. Quanto ao SMC_GetMParameters, permite a interpretação dos parâmetros K e L adjacentes às funções M.

Por sua vez, a segunda parte trata do movimento real dos atuadores. Dependendo dos comandos ativados no código G, diferentes *Drivers* são ativadas através dos blocos funcionais 'SMC_ ControlAxisbyPos'. Estas posições são dadas pelo 'SMC_TRAFO_Gantry3'.

- **SMC_TRAFO_Gantry3** — Permite a tradução da estrutura de posições transmitida pelo interpolador para 3 valores reais que representam as coordenadas em X, Y e Z.

- **SMC_ControlAxisByPos** — Este bloco funcional, além de ativar as drivers, possui um controlo intrínseco de posição ao qual é fornecido um *feedback* da posição atual dos atuadores. É aplicado um bloco funcional deste tipo para cada um dos eixos a controlar.

5.3.1.4 PLC-PRG

Neste último programa foram incluídas todas as ações relacionadas com o controlo da célula através da interface criada, como o comando manual da máquina, a referenciação, o furo e a serração independente do programa CNC, o retorno à origem, a escolha do tipo de perfil da viga, entre outros.

Assim as principais funcionalidades presentes neste programa consistem em:

- **Movimento manual dos eixos** — Permite fazer o *Jog* de qualquer um dos eixos através do acionamento dos comandos a partir do HMI.

- **Movimento semiautomático dos eixos** — Permite realizar operações como a furação, o corte ou o movimento da viga, de forma semiautomática, isto é, apenas se requer que o operador insira as coordenadas necessárias no HMI e o resto da operação será feita de forma autónoma. Na imagem 5.6 é retratada a secção do programa responsável pela furação semiautomática da viga.

- **Referenciação do zero dos eixos** — Quando esta função é executada, a posição atual dos eixos é referenciada como a origem. A função pode ser acionada através do comando manual no HMI ou, no caso real, através de sensores, pois não é possível inserir sensores na plataforma de simulação.

- **Retorno à origem** — Permite o retorno automático de todos os eixos à respetiva origem.

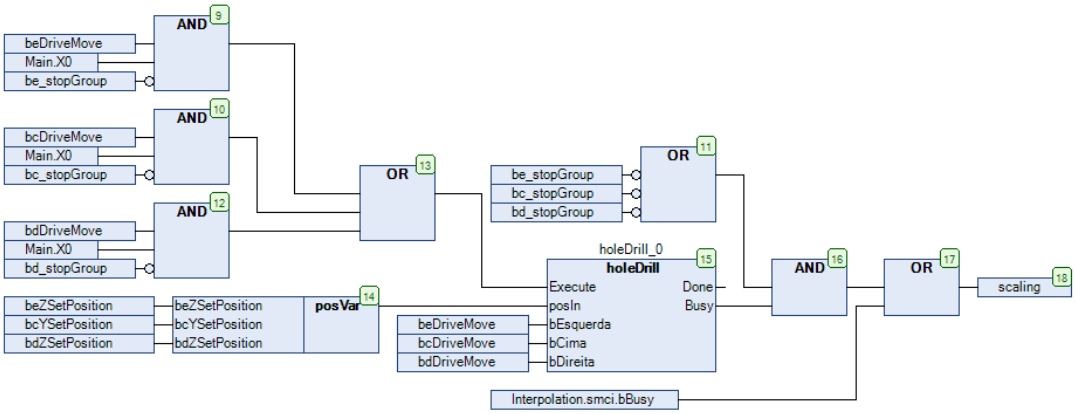

Figura 5.6: Secção do programa PLC_PRG responsável pela furação semiautomática

5.3.2 Sequenciação de Tarefas

Ao contrário de todos os outros programas presentes no controlador que se encontram na 'Main Task', o programa 'Path' encontra-se numa tarefa à parte, de menor prioridade e com um tempo de ciclo maior ('Path Task'). Isto deve-se ao facto de o 'Decoder' criar um objeto a cada chamada e se demorar mais tempo a processar esse objeto a nível da interpolação do que propriamente a criá-lo.

O mecanismo funciona da seguinte forma: Primeiramente, na 'Path Task' é criado um objeto 'GEOINFO', que contém as informações da posição e velocidade, por ciclo.

O objeto é armazenado na estrutura 'OUTQUEUE' do bloco funcional 'Decoder'. Este processo decorre até que a estrutura 'OUTQUEUE' esteja totalmente preenchida.

Quando isso acontece, todos os módulos da 'Path Task' são interrompidos e assim permanecem até que o interpolador tenha efetuado um movimento e removido um objeto 'GEOINFO' da 'OUTQUEUE'. Neste caso os módulos da 'Path Task' são reativados e a estrutura 'OUTQUEUE' volta a ser preenchida, repetindo-se o ciclo. Este comportamento encontra-se descrito na figura 5.7, que retrata a estrutura do 'OUTQUEUE' e a tarefa do interpolador de interpretar as posições dadas pelo 'GEOINFO' e interpolar o percurso entre pontos.

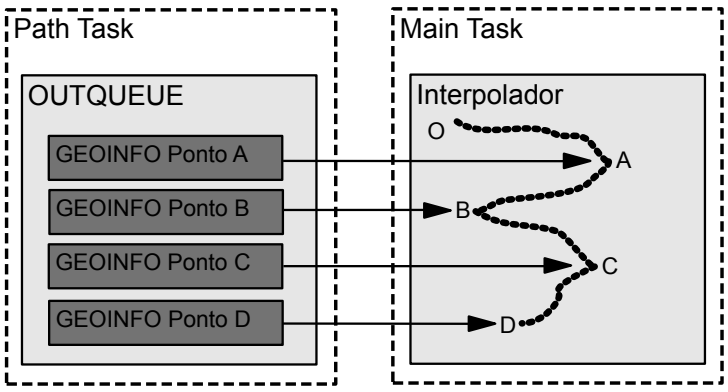

Figura 5.7: Estrutura da 'OUTQUEUE' e transmissão para o interpolador

Para cada ciclo da 'Main Task' o interpolador calcula e processa um ponto do percurso a fazer. Devido ao facto de um objeto 'GEOINFO' consiste geralmente em várias posições, irá demorar vários ciclos até que um objeto seja processado completamente e removido pelo interpolador. Posto isto, irá demorar mais ciclos a processar um objeto 'GEOINFO' do que a criá-lo. Desse modo, a melhor solução será definir as alterações entre tarefas como definido no grafcet da figura 5.8

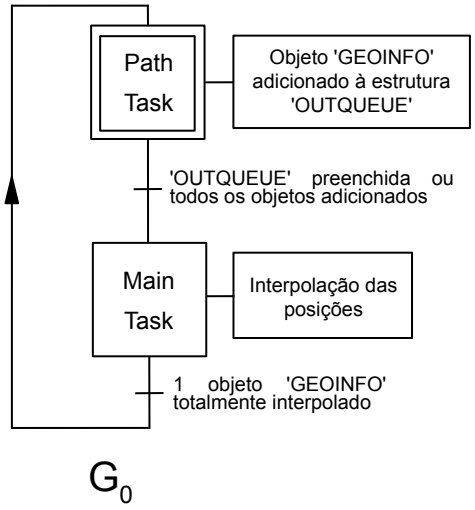

Figura 5.8: Diagrama comportamental da alteração entre tarefas

Assim, de modo a otimizar o controlador, é possível chamar a 'Path Task' com menos frequência do que a 'Main Task'. No entanto, os tempos de ciclo devem ser definidos de modo a que seja assegurado que existam sempre objetos disponíveis na estrutura, prontos a ser processados. Caso isto não aconteça, o interpolador entra em pausa até que existam novos elementos.

5.3.3 Blocos Funcionais

Como mencionado acima, procurou-se ao máximo que o programa fosse construído por módulos. Como tal, foram utilizados e criados bastantes blocos funcionais, que permitem a instanciação dos objetos, isto é, utilizar o mesmo bloco funcional em mais do que um programa.

5.3.3.1 Power

Este bloco funcional encontra-se intrinsecamente ligado às ações do sub Grafcet G0 da figura 5.3, pois os estados X01 e X03 ativam os *inputs* ON e OFF, respetivamente, deste bloco funcional.

Como também é possível verificar nos diagramas de estados das figuras 5.1 e 5.2, para existir um correto funcionamento do controlador é necessário que, antes de desligar, ele esteja na posição de 'Standstill' antes de se desligar a potência dos eixos. Para tal, chegou-se à solução da figura 5.9, representada numa versão simplificada.

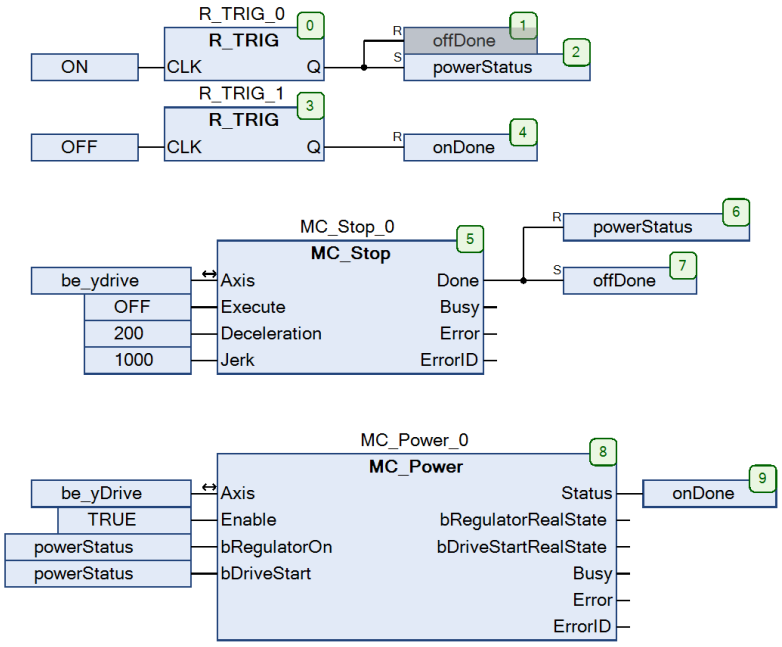

Figura 5.9: Versão simplificada do bloco funcional 'Power'

Assim, quando a entrada ON é ativada, é feito o *set* de 'powerStatus' que ativa o bloco funcional 'MC_Power' que, por sua vez ativa o eixo e envia para o exterior a informação de cumprimento da tarefa através de 'onDone'. Contudo, quando é ativado o OFF e agindo em cumprimento com as especificações da biblioteca de movimentos da PLCopen, o eixo é primeiramente colocado num estado de 'Standstill' através do bloco funcional 'MC_Stop' e só depois é que é feita a desativação do bloco 'MC_Power' e consequentemente do eixo enquanto é simultaneamente ativada a saída que indica o cumprimento da ação de desligar.

5.3.3.2 HoleDtrill e Saw

De forma a que o utilizador possa fazer operações de forma semiautomática, com o auxílio da HMI, foram feitos dois blocos funcionais 'HoleDrill' e 'Saw' relacionados com as operações de furação e serração, respetivamente.

O que se pretende é que o utilizador apenas insira as posições dos furos ou do corte e ative o comando para a operação respetiva, sendo que os blocos funcionais tratarão de gerar o movimento de forma automática.

Este bloco funcional é composto por blocos funcionais de movimento e blocos lógicos que permitem sequenciar, iniciar ou até abortar a operação.

5.3.4 Funções

As funções criadas tratam-se de funções de auxílio no cálculo numérico e não de controlo de movimento dos eixos, uma vez que não possuem capacidade de memória. Abaixo na figura 5.10,

é dado como exemplo uma das funções usadas.

Esta função tem o propósito de permitir a interpretação e posterior transformação da variável inteira 'beamInt' em 3 variáveis booleanas que permitem selecionar o tipo de perfil que está a ser tratado, permitindo assim a mudança do perfil na pré visualização.

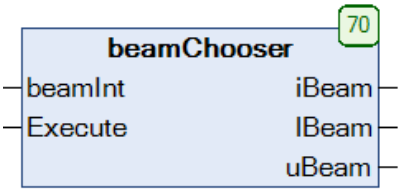

Figura 5.10: Função 'beamChooser'

Posto isto, não é necessário um elemento com memória como o bloco funcional, pois as saídas são inteiramente dependentes das entradas, não interessando o estado anterior da função. Assim, a opção pela função permite uma solução válida neste caso e que ocupa menos memória do controlador.

5.4 Interface Homem Máquina

A HMI foi desenvolvida de forma a a permitir que o utilizador final possua a capacidade de visualizar o estado atual da máquina, ativar alguns comandos e comandar manualmente as máquinas.

Assim, procurou desenvolver-se uma interface que fosse intuitiva e simples de usar, sem no entanto excluir ou descurar qualquer funcionalidade que fosse relevante para o controlo da máquina e para a visualização por parte do utilizador, tal como se vê na figura 5.11.

Começando pela visualização, criou-se uma visualização 2D das brocas (a preto), da serra (a cinzento) e do track (a azul) que se movimentam relativamente consoante a sua posição real e permitindo ao operador de chão de fábrica ter uma noção da execução dos comandos sem ter de estar a olhar diretamente para a máquina. Esta possibilidade revela-se especialmente importante no momento em que a máquina estiver a ser operada manualmente. No entanto, tem de ser assegurada a fiabilidade máxima entre o movimento real e o simulado.

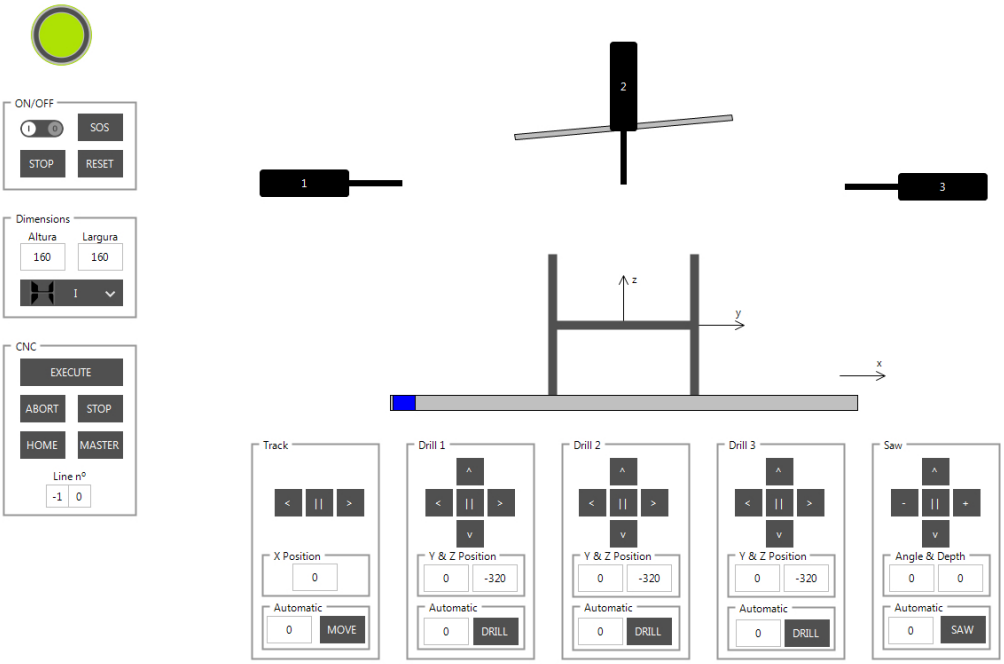

Figura 5.11: HMI Criada

Também está presente um indicador do estado da máquina no canto superior esquerdo, que fica verde quando a máquina está ligada e operacional, vermelho quando entra no estado de emergência e amarela quando está pronta para o rearme.

Em relação aos comandos que a HMI permite executar, além dos mais básicos como On/Off, o botão emergência, rearme e de stop, foram incluídos também comandos ao nível de dimensionamento da viga, comandos relacionados com a execução do programa NC e todos os comandos manuais e semi automáticos das ferramentas.

A nível de dimensionamento, caso se esteja a efetuar operações manuais, o operador pode inserir as dimensões de altura e largura de forma a permitir uma melhor visualização na simulação 2D. No entanto, no momento anterior ao corte é feita uma medição automática por parte da máquina de modo a evitar erros e potenciais acidentes. Também na execução do programa são enviadas, por meio de uma função M, as dimensões da viga de modo a serem atualizadas e permitirem uma melhor visualização das operações, visto que as dimensões da viga presente no HMI serão atualizadas consoante o input nestes campos.

Além disso, é da mesma forma possível selecionar o tipo de viga presente na máquina através de uma listbox, como representado na figura 5.12.

Figura 5.12: ListBox dos possíveis perfis da viga

Em termos dos comandos de execução do programa NC, inseriram-se funções para executar o programa, abortar ou pausar a execução, assim como funções de masterização e 'Homing', isto é, o retorno dos eixos à origem. De forma ao operador saber a linha do código NC que está a ser interpretada no momento, procurou-se incorporar uma função que o permitisse. No entanto, devido a ser necessário uma licença extra, não foi possível fazê-lo, optando-se então por integrar o número da linha de código correntemente a ser interpretada e o número de linhas total.

Chegando aos comandos manuais, presentes por baixo da representação do viga permitem fazer o *Jog* das brocas, da serra e do track. Além disso, é também possível visualizar as coordenadas atuais dos eixos e fazer operações em modo semiautomático de movimento do track, de furação e de corte.

5.5 Sumário

Neste capítulo procurou-se dar a entender o processo de desenvolvimento do controlador, assim como o funcionamento deste em maior detalhe relativamente ao 3 níveis das unidades organizacionais de programas: Programa, Bloco Funcional e Função.

Capítulo 6

Avaliação da Integração do Sistema e Resultados

6.1 Desenvolvimento do Simulador

Devido à impossibilidade de testar o software num equipamento real, foi construído um simulador da célula de forma a testar o funcionamento do controlador. Assim, foram desenhados de raiz, no software CAD 'SOLIDWORKS', mecanismos que permitissem emular as máquinas da célula real.

Após a montagem de toda a célula e posterior transformação para o formato STEP, a célula foi inserida na plataforma de simulação desenvolvida pela SARKKIS Robotics, sendo o resultado final o ilustrado na figura 6.1.

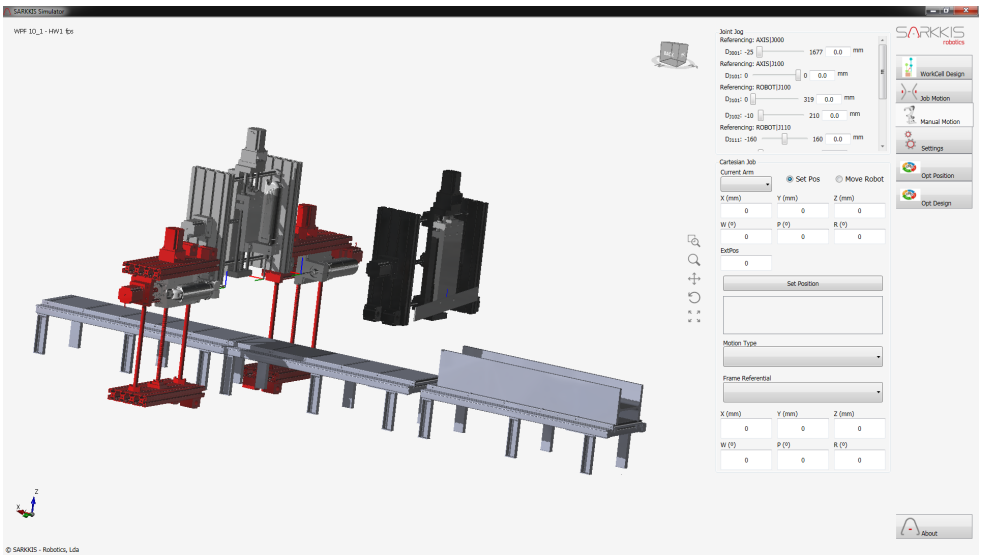

Figura 6.1: Simulador da Célula

Por fim, de forma a que os movimentos do simulador fossem controlados pelo sistema desenvolvido, implementou-se o protocolo de comunicação Modbus entre ambos.

Desse modo, foi criado um servidor Modbus TCP no controlador e um cliente no simulador, que irão comunicar entre si.

A comunicação fez-se toda através de registos e o endereçamento dos registo referentes a cada um dos eixos por sub-canal foi feita da forma descrita na tabela 6.1.

Output	Movimento correspondente
0	Batente
1	Vertical da Broca Esquerda
2	Horizontal da Broca Esquerda
3	Horizontal da Broca Superior
4	Vertical da Broca Superior
5	Vertical da Broca Direita
6	Horizontal da Broca Direita
7	Rotacional da Serra
8	Vertical da Serra

Tabela 6.1: Endereçamento dos sub-canais

6.2 Validação e Teste do Sistema

Relativamente ao pós-processador, os testes a fazer são relativos à sequenciação dos comandos dados pelo código G. Assim, de forma a validar os resultados obtidos foram geradas várias vigas no MetroID e analisados os códigos G gerados, comparando-o com os ficheiros IFC's correspondentes. Abaixo é descrito o processo de teste do sistema, com um exemplo de uma das várias vigas testadas.

A viga demonstrada na figura 6.2 foi gerada através da importação de um ficheiro IFC usado num caso real de fabrico de vigas por corte a plasma. Trata-se, por isso, de um teste significativo à eficácia do software desenvolvido, pois possui todo o tipo de operações a desenvolver pela máquina, tais como, furações isoladas, cortes paralelos e oblíquos à secção da viga e furações simultâneas dos banzos.

De salientar também que a partir deste ficheiro serão geradas 3 vigas, o que também permite testar o modo como o pós-processador trata a informação, caso o 'CuttingJob' inclua mais do que uma 'CuttingTask'.

Assim, de forma a testar o funcionamento do software, foi gerado o fabrico de 3 vigas, a partir de 2 ficheiros DSTV presentes em B e comparado com o código G correspondente, presente em A, podendo-se então analisar capacidade do pós-processador em lidar com mais do que uma 'task'.

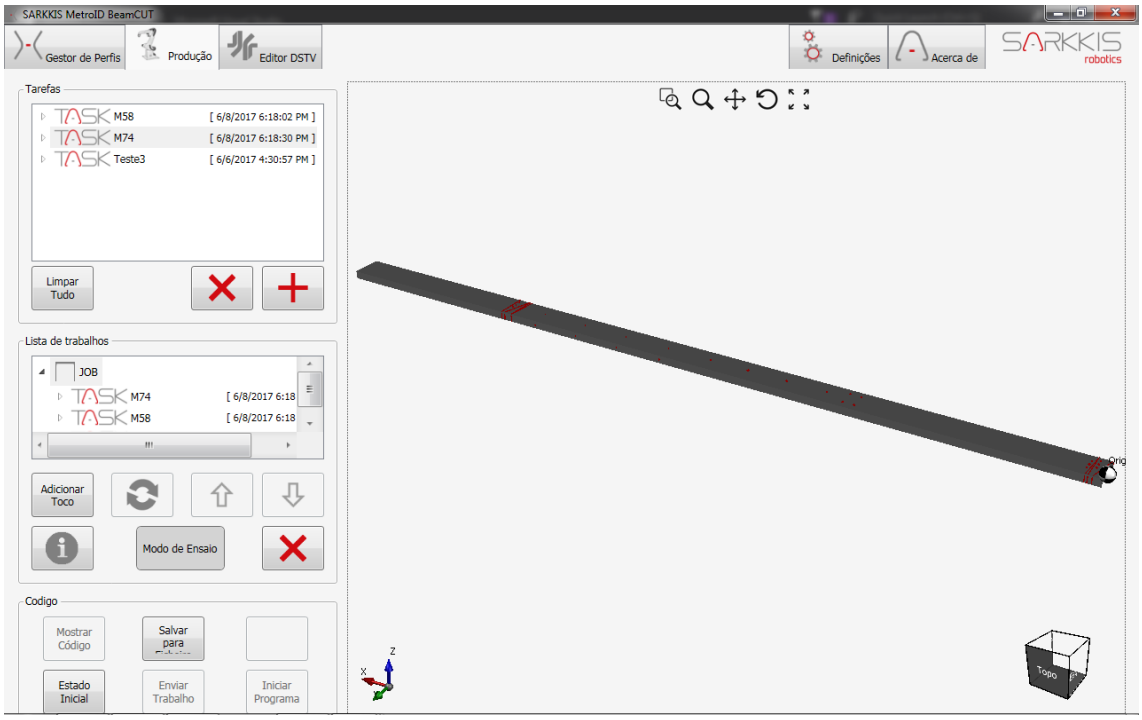

Figura 6.2: Exemplo da geração de uma viga

A partir da análise do ficheiro importado, neste caso um ficheiro DSTV, a célula terá de executar certas operações a partir do comando feito a partir do código G. Na tabela 6.2 faz-se a analogia entre as operações retratadas no ficheiro IFC e o ficheiro GCode, ambos presentes em anexo.

Task	Operação a realizar	Linhas do código G
1	Corte 15º 3 furos horizontais na alma Sangramento	N4-N5 N10-N33 N8-N9
2	Corte 5º 3 furos oblíquos na alma 2 pares de furos horizontais na alma 4 furos ao longo da viga na alma 12 furos ao longo da viga, 4 em cada banzo e 4 na alma Corte 5º Sangramento	N6-N7 N34-N57 N58-N89 N90-N120 N121-N169 e N178-N193 N170-N171 N172-N173
3	Corte 15º 3 furos horizontais na alma Sangramento	N174-N175 N194-N217 N176-N177

Tabela 6.2: Relação entre operações e código G

É de notar que a sequenciação do código G nem sempre segue a ordem das operações a realizar. Este fenómeno deve-se às restrições referentes ao esquema da célula. Por causa destas restrições, faz-se a reordenação dos cortes, que permitem que as operações sejam feitas de forma continua em relação ao eixo OX da viga e que o batente apenas se desloque numa direção, como mencionado em 4.3.

Após a análise da sequenciação do código G, foi simulada a célula de forma a analisar se, de facto, todos os comandos gerados eram cumpridos, o que se verificou.

Assim, pode-se assumir este teste como um sucesso pois, não só o pós-processador gerou corretamente o código G correspondente a cada uma das operações, como também as sequenciou de forma a que as operações fossem feitas de forma continua, sem haver regressão da viga.

Quanto ao tempo de implementação, estima-se que para implementar inteiramente esta solução numa máquina real levaria cerca de três semanas de engenharia. No entanto, segundo o Engenheiro Rui Cancelo da 'Motofil Robotics', uma empresa especializada em automação industrial e robótica, levaria dois meses para um engenheiro experiente implementar uma solução semelhante, usando controlo NC genérico. Desta forma, a solução desenvolvida permite uma implementação não só mais rápida, em 35% do tempo, como também mais flexível. Devido a estes fatores, houve um feedback muito positivo por parte da 'Motofil Robotics' em relação ao sistema desenvolvido, de tal forma que se disponibilizaram a fornecer os motores para avaliar e, consoante o resultado dos testes, implementar o sistema no futuro.

Capítulo 7

Conclusões

Esta dissertação teve como principal objetivo o desenvolvimento de um controlador capaz de automatizar o processo de corte e furação de vigas, desde a inserção do ficheiro IFC no sistema, passando pela geração de código G e acabando no comando dos movimentos necessários para o fabrico das vigas.

O desenvolvimento desta dissertação iniciou-se por um estudo dos requisitos da instalação, tais como especificações de células de corte e furação de vigas e do seu esquema construtivo, uma análise das normas aplicadas na programação de controladores lógicos programáveis, as tecnologias mais atuais necessárias para o desenvolvimento da dissertação. Após este estudo, foi projetado o sistema relativamente a componentes, arquitetura e fluxo de informação.

Depois de estudados os conceitos necessários à resolução do problema e projetada a arquitetura do sistema, foi realizado o desenvolvimento propriamente dito do sistema. Em primeiro lugar, foi desenvolvido um pós-processador para o 'MetroID BeamCUT' que gerasse código G a ser interpretado pelo controlador e uma HMI que permitisse o comando manual da máquina.

Posteriormente, utilizando o software 'Codesys', foi criado um softPLC tendo em conta todas as especificações da norma IEC 61131-3 e da biblioteca de controlo de movimento da PLCopen.

Concluída com êxito a fase de desenvolvimento do controlador e pós-processador, prosseguiu-se para a etapa seguinte: teste e validação do sistema. Como demonstrado no capítulo 6, o sistema está funcional e representa uma solução válida e pretendida no mercado, com um tempo de implementação muito menor relativamente à implementação de sistemas CNC genéricos.

Assim, ao longo da realização do trabalho, foi possível chegar a várias conclusões:

No decurso do extenso estudo realizado no início da dissertação, concluiu-se que existe um interesse muito grande, por parte da indústria, na integração de processos que conectem outros ramos da Engenharia, como a Engenharia Civil neste caso, com a automação industrial de forma a automatizar processos e atualizar indústrias.

Os conceitos da norma IEC 61131-3 aplicados durante a dissertação, revelaram-se valiosos para a realização deste software. Com o emprego de blocos funcionais simples, compatíveis e intermutáveis, como previsto na norma, assegurou-se a modularidade, a flexibilidade e a gestão da complexidade do sistema. Com esta abordagem foi possível evoluir rapidamente e de forma

organizada de um sistema simples para um sistema complexo. Deste modo realça-se a grande vantagem da programação de autómatos de acordo com a norma IEC 61131-3.

É também de salientar que a escolha do 'Codesys' como plataforma de desenvolvimento do controlador se revelou ser bastante frutífera, não só devido à grande quantidade de documentação disponível, que permitiu uma fácil integração com o pós-processador, mas também devido aos protocolos de comunicação já incorporados no software que permitiram o acoplamento entre os movimentos gerados pelo controlador e os movimentos executados no simulador de forma simples e descomplicada, neste caso utilizando o protocolo Modbus TCP.

7.1 Trabalhos Futuros

Uma vez demonstrada a validade da integração do software desenvolvido no fabrico de cortes e furações em vigas metálicas, referem-se algumas ideias interessantes de possíveis trabalhos futuros:

- **Implementação em célula real** — Tendo em conta que apenas se aplicou o sistema no contexto de simulação, seria vantajoso implementar este sistema de forma a controlar uma célula de corte e furação em contexto real.
- **Controlo de máquinas mais complexas** — Devido à arquitetura modular do controlador desenvolvido, será uma tarefa interessante explorar o controlo de máquinas mais complexas como, por exemplo, máquinas de furação que permitem o movimento das brocas nos 3 eixos coordenados.
- **Diferente estrutura da célula** — Seria também interessante aplicar o trabalho desenvolvido a células de layout simétrico e testar o seu desempenho a nível de tempos de execução.
- **Aumento do número de perfis disponíveis no pós-processador** — Visto que na presente dissertação apenas se abordou o corte de vigas metálicas com perfil I, L ou U, seria também interessante implementar o pós-processador para o fabrico de outros perfis metálicos, sejam vigas de perfis mais diversos, chapas ou tubos, integrando assim as outras vertentes do 'MetroID': 'PlanCUT' e 'TubeCUT'

7.2 Balanço pessoal

A nível pessoal, esta dissertação foi extremamente interessante e permitiu-me adquirir conhecimentos valiosos nas áreas de automação industrial, produção e programação.

A parceria com a SARKKIS Robotics foi um grande fator de motivação e inspiração no decurso do trabalho realizado, tendo aprendido muito. Além disso, a boa disposição, a entreajuda e o profissionalismo marcaram sem dúvida esta experiência única.

O desenvolvimento desta tese conduziu à aprendizagem de importantes conceitos relacionados com o desenvolvimento de software, a integração de aplicações e controlo numérico. Para

concluir é de destacar o desenvolvimento de outras capacidades pessoais como pensamento crítico e capacidade de resolução de problemas, entre muitas outras.

Foi extremamente recompensador ver reconhecimento e interesse no sistema desenvolvido, por parte de uma empresa de renome na área de automação e robótica.

Foi também muito gratificante poder dar o meu contributo académico, como coautor, na realização de um artigo científico denominado 'Flexible Work Cell Simulator using Digital Twin Methodology for Highly Complex Systems in Industry 4.0', submetido para a conferência 'ROBOT'2017: Third Iberian Robotics Conference'.

Referências

[1] André Monteiro e João Poças Martins. Building information modeling (bim) - teoria e aplicação. *INTERNATIONAL CONFERENCE ON ENGINEERING UBI2011*, Novembro 2011.

[2] areo. What is IFC and what do you need to know about it?, 2016. Disponível em http://blog.areo.io/what-is-ifc/, acedido pela última vez em 1 de junho de 2017.

[3] Max Lira Veras X. de Andrade e Regina Coeli Ruschel. Interoperabilidade de Aplicativos BIM Usados em Arquitetura por Meio do Formato IFC. *Gestão e Tecnologia de Projetos*, páginas 86–111, Novembro 2009.

[4] Rob Howard e Bo-Christer Bjork. Building Information Models – Experts' Views on BIM/IFC Developments. Relatório técnico, Construction Communications, Cambridge e Swedish School of Economics and Business Administration, Junho 2007.

[5] Dae-Hyuk Chung Ian Stroud Suk-Hwan Suh, Seong-Kyoon Kang. *Theory and Design of CNC Systems*. Springer-Verlag London Limited, Primeira edição, 2008. Páginas 1–27.

[6] M.R. Wright, D.E. Platts, D.B. French, G. Traicoff, M.A. Dupont, e G.A. Head. CNC Control Systems, Setembro 1995. US Patent 5,453,933. URL: https://www.google.com/patents/US5453933.

[7] Karl-Heinz John e Michael Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems*. Springer-Verlag GmbH, Primeira edição, 1998. Páginas 10–34, 41–54 e 99-154.

[8] R. W. Lewis. *Programming industrial control systems using IEC 1131-3*. The Institution of Electrical Engineers, primeira edição, 2001.

[9] 3S-Smart Software Solutions GmbH. User manual for plc programming with codesys 2.3, 2017. Disponível em https://www.parkermotion.com/manuals/Hauser/Compax3/CoDeSys_Manual_V2p3.pdf, acedido pela última vez em 19 de Abril de 2017.

[10] MODICON Inc. Modbus Protocol Reference Guide. Relatório técnico, MODICON, Inc., Junho 1996.

[11] 3S-Smart Software Solutions GmbH. Codesys — the comprehensive software suite for automation technology, 2016. Disponível em https://www.codesys.com/the-system.html, acedido pela última vez em 28 de Dezembro de 2016.

[12] 3S-Smart Software Solutions GmbH. Codesys device diretory, 2016. Disponível em http://devices.codesys.com/device-directory.html, acedido pela última vez em 28 de Dezembro de 2016.

[13] Robert W. Senesta. *Concepts of Programming Languages*. Pearson Education, Inc, Décima edição, 2012.

[14] Microsoft. What is a DLL?, 2017. Disponível em https://support.microsoft.com/en-us/help/815065/what-is-a-dll, acedido pela última vez em 30 de Maio de 2017. URL: https://support.microsoft.com/en-us/help/815065/what-is-a-dll.

[15] ODVA. EtherNet/IP™ – CIP on Ethernet Technology, 2016. Disponível em https://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00138R6_Tech-Series-EtherNetIP.pdf, acedido pela última vez em 28 de Dezembro de 2016.

[16] J. Reynolds. J. Postel. Rfc959. Relatório técnico, The Internet Engineering Task Force (IETF), Outubro 1985. Disponível em https://tools.ietf.org/pdf/rfc959.pdf, acedido pela última vez em 28 de Dezembro de 2016.

[17] Technical Committee 2. Motion control version 2.0. Relatório técnico, PLCopen, Março 2011. Disponível em http://www.plcopen.org/pages/tc2_motion_control/downloads/, acedido pela última vez em 31 de Maio de 2017.

[18] A. P. de Magalhães. Diapositivos de apoio ás aulas teóricas: Redes industriais. Relatório técnico, Faculdade de Engenharia da Universidade do Porto, 2016.

Anexo A

Ficheiros de texto gerados pelo pós-processador

```
% JOB
N1 G90
N2 M44 K380 L380
N3 M43 K3
N4 G01 X69.1
N5 M41 K15.0
N6 G01 X215.4
N7 M41 K5.03
N8 G01 X217.3
N9 M41 K0
N10 M40 K2 L22
N11 G01 X960.3 Y18 Z0
N12 M04
N13 M08
N14 G01 X960.3 Y18 Z150
N15 G01 X960.3 Y18 Z0
N16 M09
N17 M05
N18 M40 K2 L22
N19 G01 X960.3 Y-52 Z0
N20 M04
N21 M08
N22 G01 X960.3 Y-52 Z150
N23 G01 X960.3 Y-52 Z0
N24 M09
N25 M05
N26 M40 K2 L22
N27 G01 X960.3 Y-122 Z0
N28 M04
N29 M08
N30 G01 X960.3 Y-122 Z150
N31 G01 X960.3 Y-122 Z0
N32 M09
N33 M05
N34 M40 K2 L22
N35 G01 X1040.7 Y-77.8 Z0
N36 M04
N37 M08
N38 G01 X1040.7 Y-77.8 Z150
N39 G01 X1040.7 Y-77.8 Z0
N40 M09
N41 M05
N42 M40 K2 L22
N43 G01 X1046.8 Y-8 Z0
N44 M04
N45 M08
N46 G01 X1046.8 Y-8 Z150
N47 G01 X1046.8 Y-8 Z0
N48 M09
N49 M05
```

```
N50 M40 K2 L22
N51 G01 X1052.9 Y61.7 Z0
N52 M04
N53 M08
N54 G01 X1052.9 Y61.7 Z150
N55 G01 X1052.9 Y61.7 Z0
N56 M09
N57 M05
N58 M40 K2 L24
N59 G01 X3632.7 Y100.1 Z0
N60 M04
N61 M08
N62 G01 X3632.7 Y100.1 Z150
N63 G01 X3632.7 Y100.1 Z0
N64 M09
N65 M05
N66 M40 K2 L24
N67 G01 X3632.7 Y-99.9 Z0
N68 M04
N69 M08
N70 G01 X3632.7 Y-99.9 Z150
N71 G01 X3632.7 Y-99.9 Z0
N72 M09
N73 M05
N74 M40 K2 L24
N75 G01 X3772.7 Y100.1 Z0
N76 M04
N77 M08
N78 G01 X3772.7 Y100.1 Z150
N79 G01 X3772.7 Y100.1 Z0
N80 M09
N81 M05
N82 M40 K2 L24
N83 G01 X3772.7 Y-99.9 Z0
N84 M04
N85 M08
N86 G01 X3772.7 Y-99.9 Z150
N87 G01 X3772.7 Y-99.9 Z0
N88 M09
N89 M05
N90 M40 K2 L24
N91 G01 X4022.7 Y-9.8 Z0
N92 M04
N93 M08
N94 G01 X4022.7 Y-9.8 Z150
N95 G01 X4022.7 Y-9.8 Z0
N96 M09
N97 M05
N98 M40 K2 L24
N99 G01 X4522.7 Y-9.8 Z0
N100 M04
N101 M08
N102 G01 X4522.7 Y-9.8 Z150
N103 G01 X4522.7 Y-9.8 Z0
N104 M09
N105 M05
N106 M40 K2 L24
N107 G01 X5022.7 Y-9.8 Z0
N108 M04
N109 M08
N110 G01 X5022.7 Y-9.8 Z150
N111 G01 X5022.7 Y-9.8 Z0
N112 M09
N113 M05
N114 M40 K2 L24
N115 G01 X5522.7 Y-9.8 Z0
N116 M04
N117 M08
N118 G01 X5522.7 Y-9.8 Z150
N119 G01 X5522.7 Y-9.8 Z0
N120 M09
N121 M05
N122 M40 K2 L14
N123 G01 X6103.4 Y-9.9 Z0
N124 M04
N125 M08
N126 G01 X6103.4 Y-9.9 Z150
N127 G01 X6103.4 Y-9.9 Z0
N128 M09
N129 M05
```

N130 M40 K13 L14
N131 G01 X6103.4 Y0 Z138.6
N132 M04
N133 M08
N134 G01 X6103.4 Y190 Z138.6
N135 G01 X6103.4 Y0 Z138.6
N136 M09
N137 M05
N138 M40 K13 L14
N139 G01 X6703.4 Y0 Z138.6
N140 M04
N141 M08
N142 G01 X6703.4 Y190 Z138.6
N143 G01 X6703.4 Y0 Z138.6
N144 M09
N145 M05
N146 M40 K2 L14
N147 G01 X6703.4 Y-9.9 Z0
N148 M04
N149 M08
N150 G01 X6703.4 Y-9.9 Z150
N151 G01 X6703.4 Y-9.9 Z0
N152 M09
N153 M05
N154 M40 K13 L14
N155 G01 X7303.4 Y0 Z138.6
N156 M04
N157 M08
N158 G01 X7303.4 Y190 Z138.6
N159 G01 X7303.4 Y0 Z138.6
N160 M09
N161 M05
N162 M40 K2 L14
N163 G01 X7303.4 Y-9.9 Z0
N164 M04
N165 M08
N166 G01 X7303.4 Y-9.9 Z150
N167 G01 X7303.4 Y-9.9 Z0
N168 M09
N169 M05
N170 G01 X7457.9
N171 M41 K5.03
N172 G01 X7493.1
N173 M41 K0
N174 G01 X7495.9
N175 M41 K15.01
N176 G01 X7644.1
N177 M41 K0
N178 M40 K13 L14
N179 G01 X7903.4 Y0 Z138.6
N180 M04
N181 M08
N182 G01 X7903.4 Y190 Z138.6
N183 G01 X7903.4 Y0 Z138.6
N184 M09
N185 M05
N186 M40 K2 L14
N187 G01 X7903.4 Y-9.9 Z0
N188 M04
N189 M08
N190 G01 X7903.4 Y-9.9 Z150
N191 G01 X7903.4 Y-9.9 Z0
N192 M09
N193 M05
N194 M40 K2 L22
N195 G01 X8387.1 Y18 Z0
N196 M04
N197 M08
N198 G01 X8387.1 Y18 Z150
N199 G01 X8387.1 Y18 Z0
N200 M09
N201 M05
N202 M40 K2 L22
N203 G01 X8387.1 Y-52 Z0
N204 M04
N205 M08
N206 G01 X8387.1 Y-52 Z150
N207 G01 X8387.1 Y-52 Z0
N208 M09
N209 M05

```
N210 M40 K2 L22
N211 G01 X8387.1 Y-122 Z0
N212 M04
N213 M08
N214 G01 X8387.1 Y-122 Z150
N215 G01 X8387.1 Y-122 Z0
N216 M09
N217 M05
N218 M30
%
```

Anexo B

Ficheiros DSTV importados

B.1 Ficheiro 1

```
ST
** M58.nc1
  1384
  1
  M58
  M58
  300
  1
  PFC380*100
  U
     7273.30
      380.00
      100.00
       17.50
       10.00
       14.00
      55.200
       1.130
       5.000
       5.000
       0.000
       0.000
```

```
AK
  v       0.00u    380.00       0.00       0.00       0.00       0.00       0.00
         33.25       0.00       0.00       0.00       0.00       0.00       0.00
       7273.30       0.00       0.00       0.00       0.00       0.00       0.00
       7240.05     380.00       0.00       0.00       0.00       0.00       0.00
          0.00     380.00       0.00       0.00       0.00       0.00       0.00
AK
  o       0.00o      0.00       0.00      -5.00      17.50       0.00       0.00
          0.00     100.00       0.00       0.00       0.00       0.00       0.00
       7241.58     100.00       0.00       5.00       0.00       0.00       0.00
       7241.58       0.00       0.00       0.00       0.00       0.00       0.00
          0.00       0.00       0.00       0.00       0.00       0.00       0.00
AK
  u      31.71o      0.00       0.00       0.00       0.00       0.00       0.00
       7273.30       0.00       0.00      -5.00      17.50       0.00       0.00
       7273.30     100.00       0.00       0.00       0.00       0.00       0.00
         31.71     100.00       0.00       5.00       0.00       0.00       0.00
```

```
BO
  o     5107.65u      49.95       14.00
  o     5707.65u      49.95       14.00
  o     6307.65u      49.95       14.00
  o     6907.65u      49.95       14.00
BO
  u     5107.65u      50.00       14.00
  u     5707.65u      50.00       14.00
  u     6307.65u      50.00       14.00
  u     6907.65u      50.00       14.00
BO
  v       44.94u     267.93       22.00
  v       51.04u     198.20       22.00
  v       57.14u     128.46       22.00
  v     2637.80u      90.05       24.00
  v     2637.80u     290.05       24.00
  v     2777.80u      90.05       24.00
  v     2777.80u     290.05       24.00
  v     3027.80u     199.98       24.00
  v     3527.80u     199.98       24.00
  v     4027.80u     199.98       24.00
  v     4527.80u     199.98       24.00
  v     5107.65u     199.97       14.00
  v     5707.65u     199.97       14.00
  v     6307.65u     199.97       14.00
  v     6907.65u     199.97       14.00
EN
```

B.2 Ficheiro 2

```
ST
** M74.nc1
  1384
  1
  M74
  M74
  300
  1
  PFC380*100
  U
     145.42
     380.00
     100.00
      17.50
      10.00
      14.00
     55.200
      1.130
     14.922
      0.000
      0.000
      0.000



AK
  v       0.00u    380.00       0.00       0.00       0.00       0.00       0.00
        101.27       0.00       0.00       0.00       0.00       0.00       0.00
        145.42       0.00       0.00       0.00       0.00       0.00       0.00
        145.42     380.00       0.00       0.00       0.00       0.00       0.00
          0.00     380.00       0.00       0.00       0.00       0.00       0.00
AK
  o       0.00o      0.00       0.00     -14.92      17.50       0.00       0.00
          0.00     100.00       0.00       0.00       0.00       0.00       0.00
        145.42     100.00       0.00       0.00       0.00       0.00       0.00
        145.42       0.00       0.00       0.00       0.00       0.00       0.00
          0.00       0.00       0.00       0.00       0.00       0.00       0.00
AK
  u      96.60o      0.00       0.00       0.00       0.00       0.00       0.00
        145.42       0.00       0.00       0.00       0.00       0.00       0.00
        145.42     100.00       0.00       0.00       0.00       0.00       0.00
         96.60     100.00       0.00      14.92       0.00       0.00       0.00
         96.60       0.00       0.00       0.00       0.00       0.00       0.00
BO
  v     110.62o    172.14      22.00
  v     110.62o    242.14      22.00
  v     110.62o    312.14      22.00
EN
```