Aplicação de OPC UA e Asset Administration Shell na adaptação de um equipamento CNC à indústria 4.0

Projeto para obtenção do Grau de Mestre em
Engenharia para Fabricação Digital Direta

Pedro Rafael Marques de Sousa

Leiria, abril de 2024

escola superior de tecnologia e gestão
instituto politécnico de leiria

Aplicação de OPC UA e Asset Administration Shell na adaptação de um equipamento CNC à indústria 4.0

Projeto para obtenção do Grau de Mestre em
Engenharia para Fabricação Digital Direta

Pedro Rafael Marques de Sousa
Licenciado em Engenharia Mecânica

Trabalho de Projeto realizado sob a orientação do Professor Doutor Carlos Fernando Couceiro de Sousa Neves e do Professor Doutor Mário António Simões Correia

Leiria, abril de 2024

Originalidade e Direitos de Autor

Agradecimentos

Este trabalho foi realizado no âmbito do projeto S4Plast - *Sustainable Plastics Advanced Solutions* que foi financiado pelo Fundo Europeu de Desenvolvimento Regional através do Programa Operacional Competitividade e Internacionalização do Portugal2020.

Agradeço ao Professor Doutor Carlos Fernando Couceiro de Sousa Neves e ao Professor Doutor Mário António Simões Correia pelo apoio, ânimo, conhecimentos e orientação que me transmitiram ao longo deste curso, sem os quais não teria sido possível concluir com sucesso este trabalho.

Agradeço também aos meus colegas Marcella Cavalcanti e André Martins pelas valiosas ajudas que me prestaram ao longo desta jornada académica.

Agradeço à minha família pelo apoio que me deu, ao longo dos anos, para que conseguisse concretizar este meu objetivo.

Esta página foi intencionalmente deixada em branco.

Resumo

A aplicação de novas tecnologias como, por exemplo, Gémeos Digitais e Sistemas Cíber-Físicos, no setor industrial, desencadeou a 4ª Revolução Industrial, também designada por Indústria 4.0. Para padronizar a aplicação dos sistemas da indústria 4.0, surgiu a Arquitetura de Referência do Modelo Indústria 4.0 (RAMI4.0) que indica a utilização das tecnologias Open Platform Communication Unified Architecture (OPC UA) e Asset Administration Shell (AAS).

O projeto apresentado neste relatório consistiu em desenvolver um sistema de controlo e aquisição de dados, baseado nas tecnologias EdingCNC, OPC UA e AAS e no conceito *Plug&Produce*, que foi aplicado à fresadora CNC ProLIGHT1000 presente na ESTG.

Criou-se em C++ uma *classe* que declara vários *métodos* abstratos para representar um controlador CNC genérico, da qual foi derivada uma *classe* específica para os controladores EdingCNC que define esses *métodos* recorrendo às funções da API do respetivo software de controlo. Desenvolveu-se um servidor OPC UA e modelou-se o seu *address space* para descrever a ProLIGHT1000. Neste servidor OPC UA incluíram-se as bibliotecas que compõem a API do EdingCNC, o que permitiu criar uma solução de código para controlar e monitorizar esse equipamento CNC. Através dos *softwares* AASX Package Explorer e AASX Server, criou-se uma AAS que descreve a ProLIGHT1000 e que foi conectada ao servidor OPC UA. Esta AAS é o modelo virtual padronizado deste equipamento CNC e serve para o adaptar aos sistemas de informação da indústria 4.0, de acordo com o RAMI4.0.

O funcionamento deste servidor OPC UA foi testado através do *software* UAModeler e também se verificou a sua correta interação tanto com a AAS construída quanto com a máquina, através do *software* EdingCNC.

O resultado obtido deste trabalho foi que a monitorização e controlo deste equipamento CNC, que antes era feita apenas no computador local através do EdingCNC, passou a ser feita numa rede IP através de um servidor OPC UA e cliente(s) OPC UA, recorrendo à chamada de *métodos* de uma *classe* genérica comum. Por fim, demonstrou-se a aplicação de uma AAS e o estabelecimento da sua conexão ao servidor OPC UA.

Palavras-chave: Indústria 4.0, OPC UA, AAS, Gémeo Digital, *Plug&Produce*

Esta página foi intencionalmente deixada em branco.

Abstract

The application of new technologies, such as Digital Twin and Cyber-Physical Systems in industrial systems triggered the 4th Industrial Revolution, also known as Industry 4.0. To standardize the application of industry 4.0 systems, the Reference Architecture Model Industry 4.0 (RAMI4.0) was created, which indicates the use of Open Platform Communication Unified Architecture (OPC UA) and Asset Administration Shell (AAS) technologies.

The project presented in this report consisted of developing a control and data acquisition system, embodying the Plug&Produce concept and based on EdingCNC, OPC UA and AAS technologies, which was applied to the ProLIGHT1000 CNC milling machine present at ESTG.

A class was developed in C++ that declares several abstract methods to represent a generic CNC controller, from which a specific class for EdingCNC controllers was derived which calls the functions of the respective control software. An OPC UA server was developed, and its address space was modeled to describe the ProLIGHT1000. In this OPC UA server, the libraries that make up the EdingCNC API were included, which allowed to create a code solution to control and monitor this CNC device. Using the AASX Package Explorer and AASX Server software, an AAS was created that describes the ProLIGHT1000 and was connected to the OPC UA server. This AAS is the standardized virtual model of this CNC device allowing it to be compatible with industry 4.0 information systems, according to the RAMI4.0 model.

The operation of this OPC UA server was tested using the UAModeler software and was verified its correct interaction with both the built AAS and the CNC device.

The result obtained from this work was that the monitoring and control of this CNC device, which was previously done solely on the local computer through the EdingCNC user interface, can now be done on an IP network through an OPC UA server and OPC UA client(s), using the call of methods of a common generic class. Finally, the application of an AAS and the establishment of its connection to the OPC UA server was demonstrated.

Keywords: Industry 4.0, OPC UA, AAS, Digital Twin, Plug&Produce

Esta página foi intencionalmente deixada em branco.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

3D	Tridimensional
5G	Quinta Geração de Telecomunicações
AS	*Address Space*

AAS	*Asset Administration Shell* ; Consola de Administração do Ativo
API	*Application Programming Interface* ; Interface de Programação da Aplicação
BITKOM	*Bundesverband Informationswirtschaft, Telekommunikation und neue Medien* ; Associação Federal para Tecnologias de Informação, Telecomunicações e Novos Media
CAD	*Computer Aided Design* ; Desenho Auxiliado por Computador
CAE	*Computer Aided Engineering* ; Engenharia Auxiliada por Computador
CAM	*Computer Aided Manufacturing* ; Fabricação Auxiliada por Computador
CNC	*Computer Numerical Control* ; Controlo Numérico Computarizado
COM	*Component Object Model*
CPS	*Cyber-Physical System* ; Sistema Cíber-Físico
CPPS	*Cyber-Physical Production System* ; Sistema de Produção Cíber-Físico
CS	*Companion Specification*
DHCP	*Dynamic Host Configuration Protocol*
DT	*Digital Twin* ; Gémeo Digital
ERP	*Enterprise Resource Planning* ; Planeamento dos Recursos da Empresa
ESTG	Escola Superior de Tecnologia e Gestão
EUA	Estados Unidos da América
FA	Fabricação Aditiva ; *Additive Manufacturing* (AM)
FDD	Fabricação Digital Direta ; *Digital Direct Manufacturing* (DDM)
FDM	*Filament Deposition Modelling* ; Modelação por Deposição de Filamento
GPRS	*General Packet Radio Service*
HMI	*Human/Machine Interface* ; Interface Humano/Máquina

HTTPS	*Hyper Text Transfer Protocol Secure*
I4.0	Indústria4.0 ; *Industrie*4.0 ; *Industry*4.0
IEC	*International Electrotechnical Commission* Comissão Eletrotécnica Internacional
IoT	*Internet of Things* ; Internet das Coisas
IP	*Internet Protocol* ; Protocolo de Internet
JSON	*JavaScript Object Notation*
MES	*Manufacturing Execution System* ; Sistema de Execução da Produção
MQTT	*Message Queuing Telemetry Transport*
OPC UA	*Open Platform Communications Unified Architecture*
PLC	*Programmable Logic Controller*; Autómato Programável
QR	*Quick Response*
RAMI4.0	*Reference Architecture Model Industrie4.0*; Arquitetura de Referência para o Modelo de Indústria4.0
REST	*Representational State Transfer*
RFID	*Radio-Frequency Identification*; Identificação por Radiofrequência
RPM	Rotações por minuto
SCADA	*Supervisory Control and Data Acquisition*; Sistema de Supervisão e Aquisição de Dados
TCP	*Transmission Control Protocol*;
TIC	Tecnologias de Informação e Comunicação
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
VDMA	*Verband Deutscher Maschinen- und Anlagenbau*; Associação Alemã de Engenharia Mecânica e de Instalações Industriais
XML	*Extensible Markup Language*
ZVEI	*Verband der Elektro- und Digitalindustrie*; Associação Alemã da Indústria Elétrica e Digital

1. Introdução

Diversos autores consideram que, atualmente, assistimos à 4ª Revolução Industrial, também designada por Indústria 4.0 (I4.0) [1, 2, 3]. O conceito de I4.0 tem sido tema de discurso quer de políticos, quer de representantes do setor industrial, tanto ao nível nacional como internacional, tendo se verificado grandes desenvolvimentos neste âmbito [3].

Atualmente, as indústrias de manufatura enfrentam o desafio de se manterem rentáveis, perante um mercado globalizado, diversificado, competitivo e volátil. Verifica-se, também, a procura por produtos personalizados com ciclos de vida mais curtos. Assim, o paradigma de Produção em Massa apresenta-se desatualizado e inadequado para fornecer uma gama de produtos que satisfaçam as necessidades individuais de cada cliente [4].

Para responder a estas necessidades, surgiu o paradigma Indústria 4.0 que se baseia nos conceitos de *Smart Factory* e *Smart Manufacturing* e tem por base Sistemas Cíber-Físicos e tecnologias de Fabricação Digital que permitem recolher e processar enormes quantidades de dados (*big data*) ao longo de todo o ciclo de vida do produto e permitem aumentar a flexibilidade de produção e o controlo da qualidade, enquanto mantêm o *time-to-market* e os custos de produção competitivos com o modelo de Produção em Massa [4].

1.1. Motivação

O Mestrado em Engenharia para Fabricação Digital Direta [1], aborda os seguintes temas:

- engenharia, materiais e tecnologias aplicadas na Fabricação Digital Direta (FDD);
- áreas de aplicação das tecnologias de FDD, economia circular e sustentabilidade;
- engenharia de conceção do produto;
- diversas tecnologias relacionadas com o paradigma de Indústria 4.0.

Este projeto de mestrado foi realizado no âmbito de um subprojeto do projeto mobilizador S4Plast - *Sustainable Plastics Advanced Solutions* [2]. Projetos Mobilizadores são programas nacionais de investigação que pretendem criar novos produtos, processos ou serviços, ou melhorar aqueles já existentes [5]. O projeto S4Plast pretende produzir peças plásticas com estética metálica e elevada reciclabilidade, através de sistemas de produção autónomos, ágeis e precisos [6, 7, 8].

1 - https://www.ipleiria.pt/curso/mestrado-em-engenharia-para-fabricacao-digital-direta/
2 - https://s4plast.toolingportugal.com/

Este subprojeto do S4Plast pretende desenvolver sistemas de *hardware* e *software* que permitam integrar as informações provenientes do chão de fábrica, recolhidas de dispositivos industriais, nomeadamente os comummente encontrados na indústria de moldes e plásticos, em sistemas de automação de níveis hierárquicos superiores, segundo o modelo de I4.0.

1.2. Problemática

Atualmente, nos diversos setores industriais, é muito frequente as empresas apresentarem, no seu chão de fábrica, diversos equipamentos automáticos industriais que são integrados numa arquitetura de sistemas, designada por Pirâmide da Automação, que permite gerir, de forma centralizada e hierarquizada, as operações de produção, logística e gestão.

Máquinas-Ferramenta de Controlo Numérico Computorizado (CNC) estão presentes em inúmeras indústrias porque permitem sistemas de produção muito mais eficientes. Uma máquina-ferramenta CNC (Figuras 1 e 2) recebe comandos, em código baseado na norma ISO 6983 e designado por Código G, para acionar os seus motores de eixo e atuadores, e movimentar uma ferramenta em relação à posição da peça a ser maquinada, para realizar diversas operações de fabricação [9]. Existem máquinas-ferramenta CNC específicas para realizar trabalhos de fresagem, torneamento, impressão3D, entre outros.

Estes equipamentos geralmente apresentam diferentes interfaces e *softwares* de controlo, consoante o seu fabricante, o que pode restringir o acessos aos dados e a interoperabilidade.

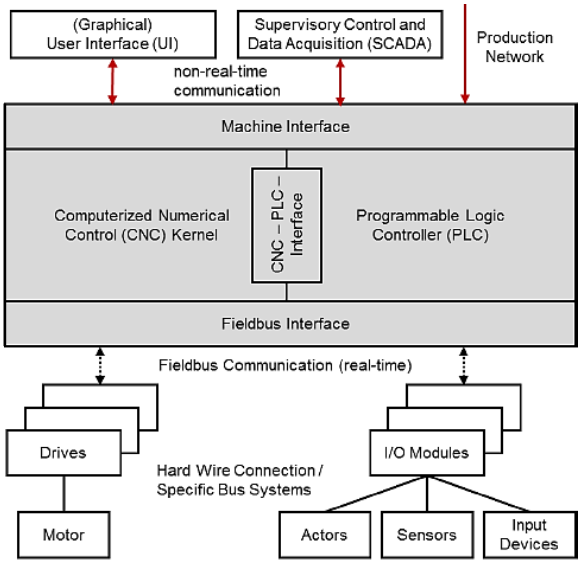


Figura 1 - Arquitetura de um sistema CNC [10].

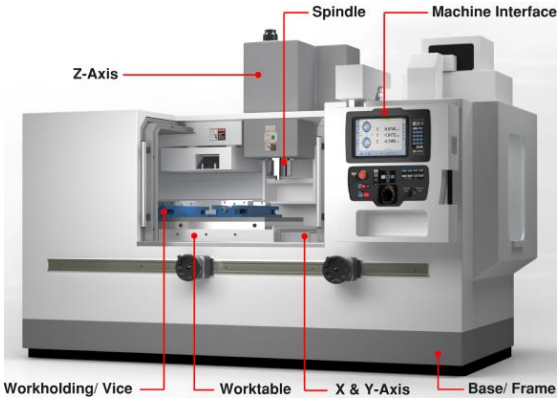


Figura 2 - Representação de uma máquina-ferramenta CNC (Exemplo de Fresadora) [11].

Assim, o chão de fábrica carateriza-se por ser composto por um conjunto de equipamentos que apresenta uma grande heterogeneidade, isto porque:

- este conjunto pode incluir equipamentos modernos, mas de diferentes fornecedores, que podem utilizar protocolos de comunicação proprietários ou distintos sistemas operativos, Interfaces de Programação das Aplicações (APIs) ou formatos de dados [12];
- devido ao elevado custo de aquisição destes equipamentos, à medida que vão sendo lançados modelos mais sofisticados, continua a fazer sentido manter os equipamentos antigos, que ainda se apresentam muito produtivos, contudo desatualizados para atender às novas necessidades de aquisição de dados ou de interoperabilidade [11, 12].
  Estes equipamentos desatualizados são designados por equipamentos *legacy*.

A maioria dos modernos equipamentos automáticos industriais disponibilizam interface de comunicação por Ethernet, enquanto que os mais antigos disponibilizam, normalmente, interface série [12].

Alguns dos métodos utilizados para integrar estes equipamentos na Pirâmide da Automação consistem em utilizar *gateways* ou *middleware* (camada de *software*) para traduzir os diferentes protocolos de comunicação ou formatos de dados, permitindo que os sistemas comuniquem de forma interoperável [3]. Contudo, estas abordagens baseiam-se em distintas redes de comunicação, específicas para cada nível da Pirâmide da Automação, o que cria uma interdependência nos processos e pode restringir o acesso às informações.

Em contrapartida, à medida que novas tecnologias vão sendo integradas nos sistemas de produção, a Pirâmide da Automação tem vindo a ser repensada para poder representar novas dinâmicas conceptualizadas para a I4.0 [14]. A arquitetura proposta para a I4.0 baseia-se na Pirâmide da Automação, mas também inclui sistemas de controlo distribuído e equipamentos capazes de comunicarem entre si diretamente, em tempo real e de forma padronizada, o que permite implementar sistemas de produção capazes de se auto-reconfigurarem, logo, mais eficientes.

Esta nova abordagem implica novos desafios como, por exemplo, com o mesmo *software* de controlo ou aquisição de dados, conseguir operar máquinas de diferentes fabricantes, independentemente de qualquer protocolo de comunicação proprietário. Nesta abordagem, utilizam-se preferencialmente protocolos de comunicação padronizados, como o *Open Platform Communications Unified Architecture* (OPC UA), *Message Queuing Telemetry Transport* (MQTT) ou *Representational State Transfer* (REST) [3].

Para estabelecer comunicação interoperável com máquinas *legacy*, pode-se optar por as substituir por outras mais sofisticadas ou por uma combinação das seguintes opções [13] :

- alterar o seu sistema elétrico, por exemplo, acrescentando sensores, microcontroladores ou outros componentes de *hardware*;
- estabelecer conectividade através de tecnologias de Internet das Coisas (IoT);
- criar o seu gémeo digital;
- aplicar *middleware* para aceder aos dados do seu controlador, recorrendo às suas APIs, caso existam, para aplicar uma camada de abstração sobre os seus sistemas de controlo.

Considerando esta heterogeneidade de equipamentos (modernos e *legacy*), surgem duas possíveis soluções, indicadas abaixo:

(1) - o(s) *software*(*s*) de controlo / aquisição de dados, conseguem interpretar cada um dos diferentes protocolos de comunicação proprietários;

(2) - cada equipamento requer um *gateway* para traduzir o seu protocolo de comunicação proprietário para um protocolo de comunicação aberto, padronizado e interoperável.

A necessidade de integrar diversos dispositivos e aplicações pode ser resolvida através da implementação de uma Arquitetura Orientada a Serviços [15]. A Arquitetura Orientada a Serviços funciona com base na descoberta dos *serviços* disponíveis na rede e na comunicação por solicitação ou resposta (Figura 3). Assim, este modelo permite estabelecer a interoperabilidade entre sistemas heterogéneos, através da interação com as APIs desses equipamentos, que correspondem a *serviços*. Aqui, o conceito de *serviço* corresponde a um módulo de *software* que apresenta uma lógica de controlo ou funções que respondem a solicitações específicas [15].

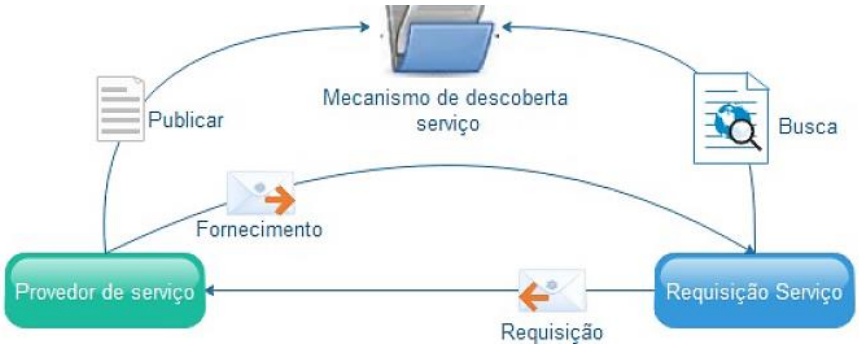


Figura 3 - Mecanismos de Comunicação numa Arquitetura Orientada a Serviços [15].

Esta necessidade de soluções que visassem a reconfigurabilidade e interoperabilidade dos sistemas de produção conduziu ao desenvolvimento do conceito *Plug&Produce* (P&P).

Arai *et al*. [16] introduziram o conceito *Plug&Produce* (P&P), que é uma metodologia para integrar equipamentos industriais nos sistemas de produção, de forma rápida e com a mínima intervenção humana, à semelhança do conceito *Plug&Play* (que se refere à forma automática de instalar dispositivos nos computadores como, por exemplo, a instalação de um rato). Desde então, o conceito de P&P tem vido a ser amplamente desenvolvido.

Weyer *et al*. [17] apresentaram o conceito de uma interface física "*Plug&Produce*" para equipamentos industriais (que integra ligações padrão para: Ethernet, eletricidade, sinal de paragem de emergência e ar comprimido), disponibilizando, assim, conectividade padrão.

Panda *et al*. [18] desenvolveram uma arquitetura P&P baseada nos protocolos *Dynamic Host Configuration Protocol* (DHCP), OPC UA e *Electronic Device Description Language*.
Assim, para equipamentos industriais conectados à rede através de Ethernet, o protocolo DHCP pode ser utilizado para lhes atribuir um endereço IP e gerir a comunicação TCP/IP. Para além disso, os *serviços Discovery* do protocolo OPC UA podem ser utilizados para, automaticamente, detetar os equipamentos conectados e integrá-los na rede do sistema de produção, sendo que cada equipamento deve disponibilizar no modelo de dados (*address space* no caso do OPC UA) do seu servidor, a sua descrição, de acordo com um modelo de informação padronizado.

Martins *et al*. [19] desenvolveram um sistema P&P, baseado no protocolo OPC UA, para monitorizar máquinas-ferramenta CNC, sendo que para essa finalidade desenvolveram uma interface genérica (Figura 4) para um conjunto de diferentes controladores CNC.

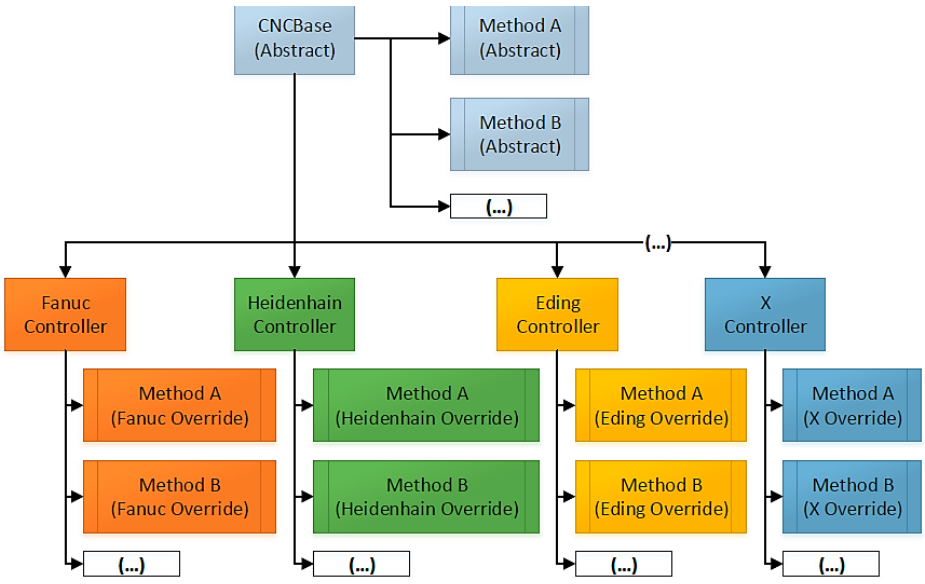


Figura 4 - Classe genérica e classes específicas para representar os diferentes controladores CNC [19].

Outra abordagem para a implementação de sistemas P&P baseia-se na representação virtual dos ativos, que é feita através da implementação das suas Consolas de Administração. Ye *et al*. [20] desenvolveram um sistema P&P, com base no protocolo de comunicação OPC UA, no formato AutomationML e na Consola de Administração do Ativo.

Portanto, esta problemática relaciona-se com a necessidade e o desafio de interconectar distintos equipamentos ou sistemas industriais de forma a estabelecer a troca de informações em tempo real e interoperabilidade, o que pode implicar: traduzir formatos de dados ou protocolos; adotar protocolos de comunicação abertos e padronizados; desenvolver soluções de *software* ou *hardware*; ou aplicar tecnologias de IoT, *middleware* ou virtualização.

1.3. Objetivo Geral

Este projeto pretende dar continuidade a outros projetos realizados na Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria, nomeadamente [21] e [22].

Para se integrar no referido subprojeto do S4Plast, este trabalho pretende desenvolver e implementar uma arquitetura de sistemas que permita conectar e operar, de forma igual, equipamentos industriais de distintos fabricantes, de acordo com conceitos de base da I4.0 como a Arquitetura Orientada a Serviços e o *Plug&Produce*. Para validar a solução proposta, será demonstrado um caso de implementação, baseado num equipamento CNC presente no Laboratório de Robótica Avançada e Fábricas Inteligentes da ESTG.

Assim, este trabalho pretende responder à seguinte questão de investigação: "Como tornar interoperáveis os equipamentos CNC industriais, para os monitorizar e controlar de forma agnóstica, segundo o modelo de Indústria 4.0 ?".

1.4. Organização deste documento

O capítulo 1 introduz a problemática, o objetivo geral e a questão de investigação.
O capítulo 2 consiste num enquadramento à Indústria 4.0 e à Fabricação Digital Direta.
O capítulo 3 descreve o estado da arte da interoperabilização de equipamentos industriais.
De seguida, apresenta-se o relatório deste projeto.
O capítulo 4 descreve as tecnologias utilizadas.
O capítulo 5 indica a metodologia adotada e descreve o desenvolvimento deste projeto.
O capítulo 6 indica os resultados obtidos.
O capítulo 7 apresenta as conclusões e as sugestões de trabalhos futuros.

2. Enquadramento

2.1. Revoluções Industriais

A Primeira Revolução Industrial ocorreu entre 1760 e 1840 e corresponde a uma transição para novos processos industriais, que aconteceu primeiro na Inglaterra e depois disseminou-se gradualmente para a Europa continental e América do Norte.

No início deste período, as fontes de energia disponíveis para realizar as atividades socioeconómicas limitavam-se ao trabalho de seres vivos ou ao trabalho mecânico de sistemas eólicos ou hídricos. Durante este período ocorreu um grande desenvolvimento da Engenharia Mecânica, destacando-se os teares mecânicos, motores a vapor, locomotivas, navios a vapor e máquinas-ferramenta como o torno mecânico [23].

Assim, as indústrias artesanais, dispersas entre vilas e aldeias, foram substituídas por indústrias urbanas, concentradas em fábricas movidas a vapor, verificando-se um aumento da produtividade [23].

A Segunda Revolução Industrial corresponde a uma fase de rápido desenvolvimento científico e industrial que aconteceu no Reino Unido, Estados Unidos, Alemanha, França, Itália e Japão, entre 1870 e 1914, que está associada à eletrificação de cidades e fábricas, à melhoria das condições de vida e de trabalho e à utilização do petróleo como combustível. As indústrias passaram a utilizar a energia elétrica, sendo os circuitos de comando ainda baseados em tecnologia pneumática ou hidráulica, o que se designa por Automação Fixa [3].

Nesta época, Henry Ford organizou as suas fábricas, para que máquinas e trabalhadores fossem posicionados segundo a sequência de trabalhos, formando uma Linha de Montagem, o que permitiu fabricar, de forma rápida e eficiente, uma maior quantidade de automóveis, que se tornaram mais baratos, difundindo, assim, o modelo de Produção em Massa.

A Terceira Revolução Industrial, ocorrida entre 1960 e 2000, foi um fenómeno global de consolidação da automação baseada em eletrónica e tecnologias de informação, associado ao desenvolvimento de semicondutores, microprocessadores, computadores e Internet.

A partir de 1960, surgiram sistemas CNC, robôs industriais, protocolos de comunicação industrial, Autómatos Programáveis (PLC) e *softwares* para auxílio em Desenho (CAD), Fabricação (CAM) e Engenharia (CAE) [3].

Neste período surgiu a Automação Flexível, que permite que alterações aos produtos sejam concretizadas em tempo real, sem a necessidade de parar toda a linha de produção. Para além disso, introduziu-se a Automação Industrial por Níveis Hierárquicos, na forma de pirâmide, em que cada nível comunica apenas com o nível que lhe está diretamente acima ou abaixo [3]. Estes níveis estão identificados na Figura 5 e descritos abaixo:

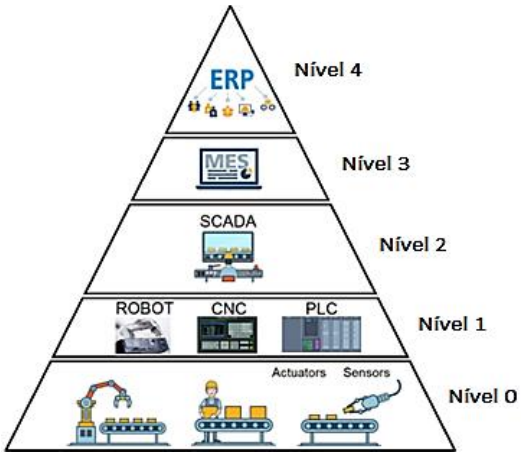


Figura 5 - Pirâmide da Automação e respetivos níveis hierárquicos (adaptado de [24]).

- **Nível de Campo / Chão de Fábrica** (**Nível 0**): máquinas e seus sensores ou atuadores.
- **Nível de Controlo** (**Nível 1**): diversos controladores como, por exemplo, CNC ou PLC.
- **Nível de Supervisão** (**Nível 2**): sistema designado por *Supervisory Control and Data Acquisition* (SCADA), composto por *hardware*, como computadores industriais e redes de comunicações e componentes de *software*, como Interfaces Humano / Máquina (HMI) e bases de dados. Este sistema recolhe dados em tempo-real para monitorizar as máquinas no chão de fábrica ou acionar os seus controladores.
- **Nível de Gestão da Produção** (**Nível 3**): sistema informático designado de *Manufacturing Execution System* (MES) que recebe informações provenientes dos níveis 2 e 4, e que gere as funções de controlo das operações de produção, alocação de recursos, gestão da manutenção de equipamentos e controlo de qualidade [3].
- **Nível de Planeamento da Empresa** (**Nível 4**): *software* designado de *Enterprise Resource Planning* (ERP) que permite gerir as finanças, logística, recursos humanos e materiais, projetos, encomendas, etapas de fabricação e ciclo de vida dos produtos, para a(s) unidade(s) de produção industrial da empresa [3].

Hoje assistimos à quarta revolução industrial, que teve início na 2ª década do século 21, desencadeada por avanços informáticos relacionados com o grande aumento da capacidade de cálculo dos equipamentos e está associada ao desenvolvimento novas tecnologias como,

por exemplo, Internet das Coisas, Inteligência Artificial, Computação na Nuvem, Analítica de *Big Data*, Robótica e Fabricação Aditiva [1, 3].

Em contrapartida, diversas indústrias têm vindo a implementar mudanças estruturais para se adaptarem a várias tendências globais, como políticas de desenvolvimento sustentável, mercados voláteis e crescente procura por produtos personalizados [25].

Neste contexto, em julho de 2010, o governo alemão anunciou o seu programa *High-Tech Strategy 2020*, para assegurar a sua posição de líder em desenvolvimento tecnológico, em áreas como setor industrial, ação climática, energia e condições sociais [26]. Em 2011, o mesmo governo convocou um grupo composto por representantes das áreas científica, política e industrial para desenvolver esse programa no setor industrial [26]. Em 2013 este grupo de trabalho propôs algumas recomendações, através da publicação do relatório "*Recommendations for implementing the strategic initiative INDUSTRIE4.0*", que o governo alemão adotou e que serviram de base para lançar a iniciativa nacional designada de *Industrie4.0*, para ser implementada no prazo de 15 anos [26]. Contudo, esta designação foi usada, pela primeira vez, pela chanceler Angela Merkel, na Feira de Hannover, em 2011.

Os objetivos do programa *Industrie4.0* são: tornar a economia alemã mais competitiva e obter crescimento económico; sofisticar o seu sector industrial, através de tecnologias como Internet das Coisas e Sistemas Cíber-Físicos; e desenvolver tecnologias para digitalizar e interconectar produtos, processos, negócios e cadeias de valor. Para além disso, este programa pretende apoiar a investigação e padronização de tecnologias, criar modelos de implementação, criar uma rede de parceiros e financiar empresas [26].

As Associações de Engenharia Mecânica e Instalações Industriais (VDMA), de Indústria Elétrica e Digital (ZVEI), de Telecomunicações, Tecnologias de Informação e Novos Media (BITKOM), o Ministério Federal da Educação e Investigação e o Ministério Federal da Economia e Energia fundaram, em 2013, o *website* PlattformIndustrie4.0 [3], que serve para coordenar e divulgar o programa *Industrie4.0* [25, 26]. Esse programa tem demonstrado grande sucesso, por exemplo, através do desenvolvimento da Arquitetura de Referência do Modelo Indústria 4.0 (RAMI4.0) [26].

3 - https://www.plattform-i40.de

Atualmente, a Alemanha representa a maior economia da União Europeia e a quarta maior do mundo, com um setor de exportação extremamente desenvolvido, nomeadamente de máquinas, automóveis, produtos eletrónicos e produtos químicos [28].

À semelhança do governo alemão, vários países abordaram esta temática e criaram o seu programa nacional de desenvolvimento, adaptado ao perfil da sua economia. Como exemplo, pode referir-se que, nos Estados Unidos da América, em 2014, foi fundada a *Industrial Internet Consortium*; na China, em 2015, foi lançado o programa *Made in China 2025*; na França, em 2015, foi lançado o programa *Alliance pour l'Industrie du Futur*; e em Portugal, em 2017, foi lançado o programa *Indústria4.0*.

De acordo com Hall *et al*. [29], estes programas de desenvolvimento baseiam-se num conjunto de conceitos comuns, nomeadamente:

- **Interconexão -** máquinas, dispositivos e pessoas estarão conectadas através da IoT;
- **Modularidade -** sistemas modulares permitem uma adaptação fácil e flexível a novas necessidades através da troca, remoção ou adição de módulos individuais;
- **Interoperabilidade -** sistemas cíber-físicos estarão conectados entre si e cooperarão através da IoT, de protocolos de comunicação e de modelos de informação padronizados;
- **Virtualização -** implementação de modelos e simulações de sistemas ou processos físicos, formando uma réplica digital precisa e continuamente atualizada, o que promove a otimização, rastreabilidade, autonomia e flexibilidade dos sistemas de produção;
- **Descentralização -** implementação de sistemas cíber-físicos, caraterizados pelas suas capacidades de se auto-configurarem e de tomarem decisões de forma autónoma;
- **Capacidade em Tempo-Real -** a informação será recolhida e processada em tempo real. Todos os participantes da cadeia de valor poderão interagir entre si, em tempo real;
- **Orientação para Serviços -** aplicações e *serviços* bem tipificados, com base na *cloud* ou noutros sistemas cíber-físicos, estarão disponíveis na internet.

A Internet e sensores mais sofisticados podem ser utilizados em sistemas embebidos, o que permite que, ao longo do ciclo de vida de um produto ou processo, uma enorme quantidade de dados seja recolhida, processada e transmitida, para automatizar o processo produtivo a todos os níveis da cadeia de valor [1].

A Indústria 4.0 não rejeita a tradicional Pirâmide da Automação, que acaba por englobar, mas permite uma rede em que todos os elementos podem comunicar com todos os níveis, sem intermediação, e também com a *cloud*, tal como mostra a Figura 6 [30].

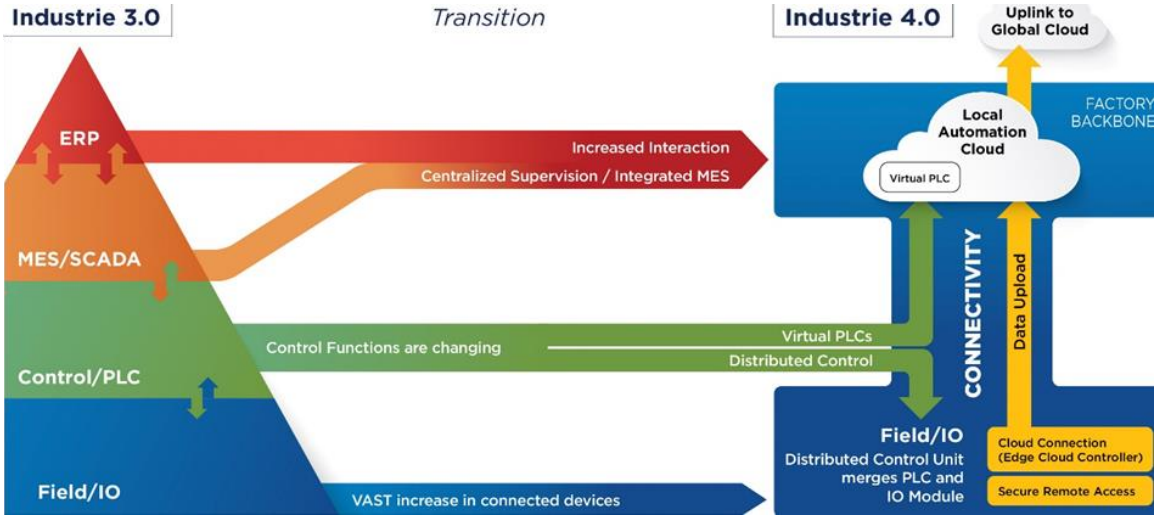


Figura 6 - Evolução da Pirâmide da Automação para um novo modelo: Pilar da Indústria4.0 [31].

2.2. Fabricação Digital Direta

Fabricação Digital designa a aplicação de sistemas informáticos a serviços de fabricação, produtos ou processos, para os interligar e criar uma abordagem de fabricação integrada digitalmente, desde a etapa de *design* até à produção e operações de logística [32].

Fabricação Digital Direta (FDD) designa a fabricação, descentralizada e colaborativa, de produtos funcionais para o seu uso-final, a partir de um ficheiro digital que contém o correspondente modelo 3D, através de máquinas de Fabricação Aditiva conectadas por Tecnologias de Informação e Comunicação (TIC) [32, 33].

Fabricação Aditiva (FA) é definida pela a norma ISO/ASTM 52900 : 2015 [35] como "o processo de juntar materiais, camada sobre camada, para produzir objetos, de acordo com a informação de um modelo 3D digital" (Figura 7). As tecnologias de FA permitem fabricar objetos em materiais poliméricos, metálicos, cerâmicos ou compósitos (Tabela 1).

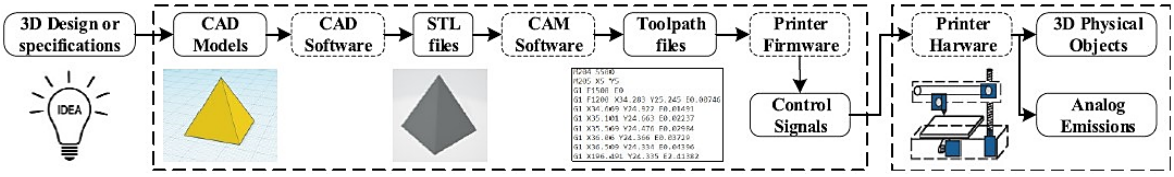


Figura 7 - Etapas que compõem o fluxo de trabalho dos processos de fabricação aditiva [36].

Tabela 1 - Categorias de processos de Fabricação Aditiva [35].

Categoria de Processos de Fabricação Aditiva	Exemplos de Processos de Fabricação	Materiais Processáveis
Fotopolimerização em Câmara	*Stereolithography* ; *Digital Light Processing* ; *Digital Light Synthesis*	resinas fotopoliméricas
Impressão de Material	*Material Jet Printing* ; *Polyjet*	polímeros ou ceras
Impressão de Aglomerante	*Color Jet Printing* ; *High Speed Sintering*	polímeros, metais ou cerâmicos
Extrusão de Material	*Fused Deposition Modeling*	polímeros ou compósitos
Fusão em Camada de Pó	*Electron Beam Melting* ; *Selective Laser Melting* ; *Selective Laser Sintering*	polímeros, metais ou cerâmicos
Laminação em Folhas	*Laminated Object Manufacturing*	metais, papel ou polímeros
Deposição Direcionada de Energia	*Laser Engineered Net Shaping* ; *Wire Arc Additive Manufacturing*	metais

Os processos de fabricação aditiva são considerados ideais para [37] :

- aplicações de prototipagem rápida;
- produzir pequenas séries de produtos;
- fabricar produtos com geometria muito complexa;
- fabricar objetos com um gradiente controlado de propriedades mecânicas ou de materiais;

Atualmente, os processos de fabricação aditiva têm aplicação em várias industrias, como automóvel, aeroespacial, medicina, arquitetura, construção civil e bens de consumo [37].

Assim, as tecnologias de fabricação digital direta permitem fabricar produtos personalizados e promover a sustentabilidade e modelos de fabricação mais flexíveis [33].

3. Estado da Arte

3.1. Gémeos Digitais

O conceito de Gémeo Digital (*Digital Twin*, DT) surgiu a partir do modelo "*Conceptual Ideal for PLM*" apresentado por Dr. Michael Grieves numa formação sobre *Product Lifecycle Management*, na Universidade do Michigan, em 2002.

Em 2003, o Dr. Grieves introduziu, pela primeira vez, o termo *Digital Twin* para se referir a um sistema composto por três elementos:

- produto físico;
- produto virtual no espaço digital;
- conexões de dados e informações, que unem o produto físico ao produto virtual.

Este produto virtual seria um modelo detalhado de um produto físico, de forma que ambos fossem virtualmente indistinguíveis. Através das conexões de dados, as características e o comportamento do produto físico seriam replicadas no produto virtual, e vice versa [36, 37]. O produto físico e o produto virtual estariam conectados durante todo o seu ciclo de vida. No produto virtual estaria representado:

- modelo 3D do produto físico, com dimensões e tolerâncias;
- lista de materiais, e lista de processos de fabricação;
- histórico de serviços desempenhados e de componentes substituídos;
- dados do estado operacional do produto, o seu histórico e previsões de estados futuros.

Kritzinger *et al*. [40] distinguiram os conceitos Modelo Digital (*Digital Model*), Sombra Digital (*Digital Shadow*) e Gémeo Digital (*Digital Twin*), sendo este último caraterizado por fluxos bidirecionais de dados, feitos de forma automática, tal como mostra a Figura 8.

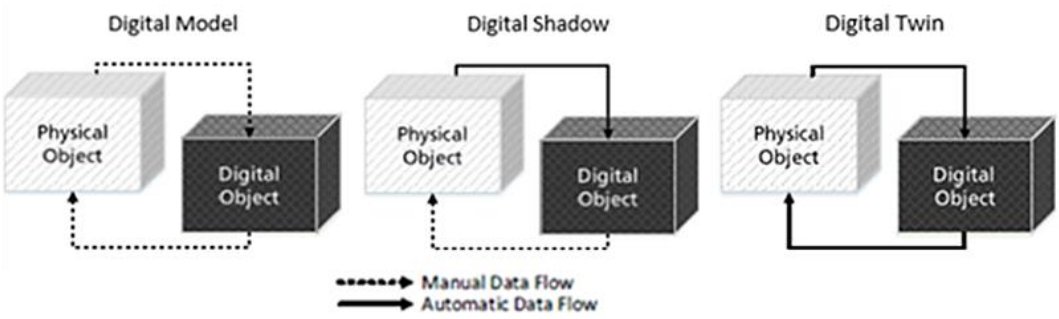


Figura 8 - Distinção entre os conceitos *Digital Model*, *Digital Shadow* e *Digital Twin* [40].

A primeira referência à implementação de DT surge em 2012 e é atribuída à NASA e à *US Air Force* que aplicaram este conceito para prever a vida útil dos seus veículos aeroespaciais, integrando simulações estruturais, dados provenientes do sistema de gestão da condição do veículo integrado a bordo e o histórico de manutenção. Nesta aplicação, DT foi definido como: "uma simulação probabilística, integrada em várias escalas e físicas, de um veículo ou sistema, que integra o histórico da frota, dados provenientes de sensores e os melhores modelos físicos à disposição, para replicar o estado do seu gémeo físico" [41].

Tao & Zhang [42] introduziram e descreveram o funcionamento e implementação do conceito de gémeo digital do chão de fábrica, DT *Shop-Floor*, integrando os elementos: chão de fábrica físico; chão de fábrica virtual; dados; e sistema de serviços.

Vrabič *et al*. [43] definiram DT como: "uma representação digital de um produto, ou sistema, que integra simulações e serviços de dados. Durante todo o ciclo de vida do produto, esta representação contém informações, provenientes de várias fontes, que são continuamente atualizadas e visualizadas, para analisar estados atuais ou futuros e otimizar processos de tomada de decisão. Um DT pode apresentar-se como uma simulação razoável da operação de um sistema físico ou como uma simulação de ultra precisão".

Tao *et al*. [44] propuseram uma arquitetura de DT, com 5 dimensões, incluindo:
(1) - entidade física; (2) - modelo digital; (3) - dados; (4) - serviços digitais;
(5) - conexões entre o produto físico, o produto virtual, dados e os serviços digitais.

Outra noção interessante apresentada por Grieves [38] é que, para além de se poder simular os vários processos de fabricação, também se poderiam integrar na simulação as informações provenientes do sistema MES relativas à conclusão de cada processo, deixando de se tratar de uma pura simulação para ser um réplica precisa do funcionamento real da fábrica, obtendo-se uma emulação através de tecnologias de *Hardware-in-the-Loop*.

A partir de 2016, surgiram as primeiras aplicações de DT no âmbito da I4.0, destacando-se o desenvolvimento pelas empresas Siemens, GE e IBM de plataformas baseadas na *cloud* para criar e monitorizar em tempo-real DTs, como MindSphere, Predix e Watson IoT Platform, respetivamente [43, 44, 45].

Atualmente, a tecnologia DT é aplicada em vários setores, como as indústrias automóvel, de manufatura e aeroespacial, construção civil, redes de eletricidade e saúde [38, 45, 46].

No setor industrial, o DT é aplicado principalmente para:

- planeamento de linhas de produção;
- simulações e emulações de processos de fabricação;
- monitorização, controlo e otimização, em tempo-real, de processos de fabricação;
- manutenção preventiva da condição de ativos;
- deteção e análise de tendências e controlo de qualidade.

De acordo com Kritzinger *et al*. [40], Juarez *et al*. [49], e Qi *et al*. [47], um DT pode ser desenvolvido com diferentes níveis de integração e não existe um método padrão para a sua implementação. A implementação de um DT é um processos moroso e interdisciplinar, porque pode envolver conhecimentos de Engenharia Mecânica, Eletrotécnica e Informática e implica a modelação virtual de uma entidade física, o que pode requerer vários submodelos, como modelação geométrica, física, de comportamento e de regras (Figura 9).

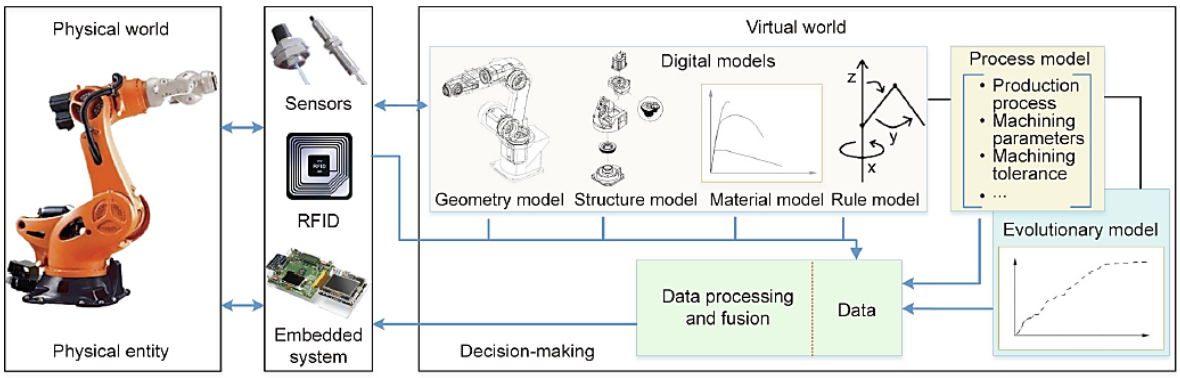


Figura 9 - Representação conceptual do funcionamento de um Gémeo Digital [48].

Segundo Tao & Zhang [42], Hu *et al*. [46], e Qi *et al*. [47], um DT completo inclui:

- **Modelo Geométrico** que descreve a forma, dimensões, posição e relações de montagem dos componentes da entidade física. Inúmeros *softwares* de CAD permitem modelar e visualizar estas informações como, por exemplo, AutoCAD ou SolidWorks.
- **Modelo Físico** que descreve as solicitações ou constrangimentos do sistema físico e as propriedades dos materiais que o compõem. A maioria dos *softwares* de CAD oferece uma abordagem integrada com ferramentas de CAE que permitem realizar simulações através de Análise por Elementos Finitos. Também se refere o uso de outras ferramentas como, por exemplo, o *software* Simulink.
- **Modelo de Comportamento** que define o funcionamento do sistema, de acordo com programas de Controlo Numérico, e define mecanismos de resposta a estímulos externos (interferência humana ou interação com objetos, físicos ou digitais).

Desta forma, é o mesmo código que controla o sistema físico e o sistema virtual.

Aqui, incluem-se *softwares* de programação e simulação de robots, como o RobotStudio ou o RoboDK, e também softwares de desenvolvimento de jogos, como o Unity3D. Refere-se também que, através do *software* CODESYS, normalizado para automação de PLCs, é possível modelar o sistema de controlo de uma máquina CNC. Utilizando um *software* de simulação de sistemas físicos (exemplo: Simumatik), o modelo virtual dessa máquina pode interagir com o *software* de automação através de *socket communication*.

- **Modelo de Regras** que define associações e constrangimentos, para que o DT possua capacidade de avaliar e realizar a otimização de parâmetros do processo, com base no atual estado operacional, registo histórico e outros dados provenientes da rede. Para criar estes modelos, pode-se recorrer a representações lógicas, usando descrições em linguagem *Extensible Markup Language* (XML), modelos de Ontologia ou Inteligência Artificial (como redes neuronais e algoritmos de *machine learning*).

Para facilitar a integração destes modelos, pode-se recorrer a uma plataforma de *software*, comum a todos, que permita realizar as trocas de dados e defina as relações de *input*/*output*, e de influência. Exemplo deste tipo de plataforma é o *Robot Operating System*. Vrabič *et al*. [43] demonstraram a implementação do Gémeo Digital de um robô, através da integração de vários submodelos no *Robot Operating System*.

Fuller *et al*. [45] descreveram a arquitetura de um DT, segundo uma organização em quatro níveis. Estes níveis estão identificados na Tabela 2 e descritos abaixo.

Tabela 2 - Níveis da arquitetura de um Gémeo Digital e respetivas tecnologias de base [45].

Nível	Tecnologias de Base
N4 - Aplicação	Modelação, Visualização e Serviços
	Software e APIs
	Processamento de dados
N3 - Middleware	Armazenamento de Dados
	Pré-processamento de Dados
N2 - Redes	Tecnologias de Comunicações
N1 - Objeto	Plataformas de Hardware
	Tecnologias de sensores

- **Nível 1 -** relacionado com os componentes físicos do sistema, nomeadamente: *hardware* de processamento e controlo; leitores de códigos de barras, de QR *codes* ou de RFID; câmaras, sensores ou atuadores.
- **Nível 2 -** que inclui protocolos e tecnologias de comunicações, por cabo, ou sem fios.
  As tecnologias de comunicação por fios são: cabo coaxial, de par entrançado e fibra ótica.
  As tecnologias de comunicação sem fios incluem: *Zig-Bee*, *Bluetooth*, *Wi-Fi*, *Ultra-Wide Band*, *Near-Field Communication*, rádio digital, satélite e comunicações móveis GPRS, 3G, 4G e, mais recentemente, 5G.
- **Nível 3 -** corresponde ao *middleware*, ou seja, às tecnologias que permitem a conexão entre vários sistemas operativos, bases de dados, ou aplicações.
  Este nível está relacionado com as tecnologias que permitem o armazenamento de dados (exemplos: MongoDB, NoSQL, MySQL, NewSQL) e o seu processamento através de algoritmos, por exemplo, de *data mining* ou *machine learning*, em servidores *web* "*Platform-as-a-Service*" que disponibilizam APIs e serviços de pré-processamento ou visualização (exemplos: BigQuery e Apache).
- **Nível 4 -** que corresponde aos submodelos de um DT. A interface e os submodelos de um DT podem ser implementados através de plataformas de *software* baseadas na *cloud* e desenvolvidas especificamente para criar e monitorizar DTs como, por exemplo, a Predix, Mindsphere ou Twin Builder.

De forma complementar, refere-se o trabalho de Onaji *et al*. [50] que constitui um modelo para a implementação do DT dos processos de fabricação, na indústria de manufatura:

- **Nível 1 -** elementos integrados por microcontroladores, PLCs, HMI, identificação por radiofrequência (RFID) e os protocolos TCP/IP, Ethernet, OPC UA, e MTConnect;
- **Nível 2 -** *softwares* de simulação que integram e disponibilizam um servidor OPC UA. (exemplos: Simumatik; Tecnomatix);
- **Nível 3 -** algoritmos de inteligência artificial;
- **Nível 4 -** plataformas *cloud*, para processar e armazenar dados;
- **Nível 5 -** integração com o sistema ERP.

No âmbito da Fabricação Digital Direta, refere-se o trabalho de Pantelidakis *et al*. [51] que consistiu em desenvolver o DT de uma impressora3D de *Fused Deposition Modeling* (FDM), implementado na plataforma de *software* Unity3D, a partir de dados recolhidos pelo *software* de controlo Octoprint, o que possibilitou simular e otimizar o processo.

3.2. Sistemas Cíber-Físicos

Em 2006, na *National Science Foundation*, nos EUA, a Drª. Helen Gill introduziu o conceito de Sistema Cíber-Físico (CPS) para se referir a uma nova geração de sistemas, derivados de sistemas embebidos e sistemas mecatrónicos, caraterizados pela integração de processos computacionais e controlo de processos físicos (Figura 10), cujos componentes possuem capacidade para comunicar em rede, sendo projetados para responder em tempo-real [52]. Nestes sistemas, o domínio cíber inclui *software*, sistemas embebidos e redes de sistemas, integrando processos de comunicação, computação e controlo [53]. Os dados obtidos por sensores influenciam os processos computacionais, e vice versa, ou seja, a computação afeta o controlo dos processos físicos, feito através de atuadores.

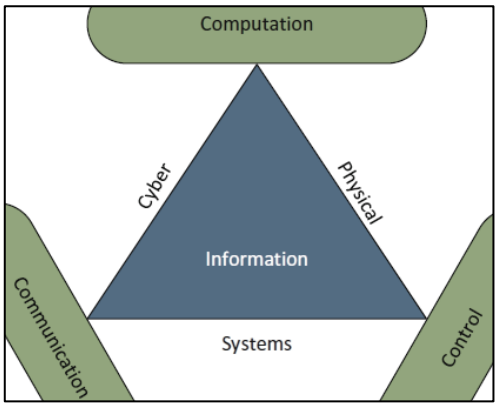


Figura 10 - Conceito de Sistema Cíber-Físico [53].

Assim, conclui-se que os conceitos de gémeo digital e sistema cíber-físico estão fortemente relacionados. Contudo, diferem porque são projetados para concretizar diferentes objetivos. Os DTs são projetados para simular, monitorizar, controlar e prever o estado de um processo / componente / sistema físico, enquanto os CPSs são projetados para serem equipamentos autónomos (*self-X capabilities*) que estão associados a sistemas de controlo distribuído e às tecnologias de SOA, IoT, comunicação máquina-a-máquina, computação *fog* / *edge* / *cloud*, *big data analytics* e modelos de informação padronizados [53].

Os atuais protocolos industriais para comunicação, em tempo-real, são redes baseadas em Ethernet, como Modbus/TCP, EtherCAT, PROFINET, Ethernet/IP, Ethernet Powerlink e SERCOS III. Estes protocolos podem ser interligados através de *gateways* [3].

Atualmente, os CPSs são aplicados nas indústrias automóvel e aeroespacial [53]. Um sistema cíber-físico implementado num sistema de produção designa-se por Sistema de Produção Cíber-Físico (*Cyber-Physical Production System,* CPPS).

Lee *et al*. [54] desenvolveram uma arquitetura conceptual (Figura 11), composta por 5 níveis, para a implementação de CPPSs. Estes níveis podem ser descritos da seguinte forma:

- **Nível 1 -** aquisição de dados (provenientes de sensores, controladores, ou sistemas MES e ERP) e transferência para um servidor central. Neste nível, o protocolo MTConnect revela-se muito útil para recolher, em tempo-real, as informações de máquinas CNC, independentemente dos seus protocolos proprietários.
- **Nível 2 -** utilização de aplicações de prognóstico e gestão da condição do ativo, baseadas em algoritmos, para os equipamentos analisarem o seu próprio estado (*self-awareness*).
- **Nível 3 -** utilização de um servidor central, que recolhe informação de todas as máquinas CNC, o que permite comparar os seus desempenhos (*self-compare*).
- **Nível 4 -** apresenta informações aos utilizadores e apoia processos de tomada de decisão.
- **Nível 5 -** a partir do *feedback* proveniente dos níveis inferiores, este nível permite que máquinas CNC sejam capazes de se auto-configurarem (*self-configure and self-optimize*)

Mourtzis *et al*. [55] e Liu *et al*. [56] demonstraram a implementação de CPPSs, tendo por base os protocolos OPC UA e MTConnect.

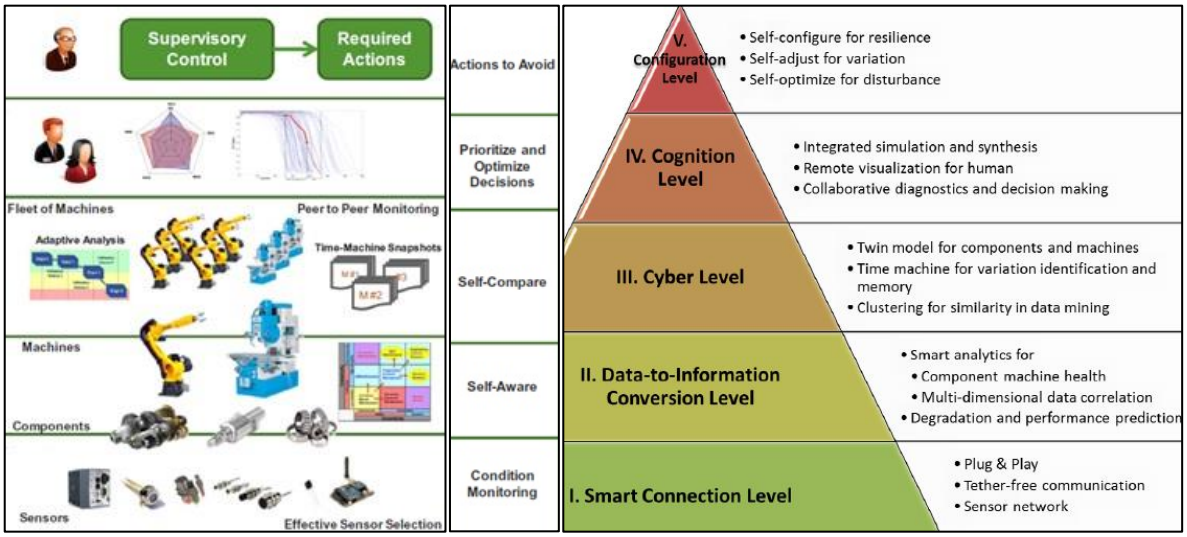


Figura 11 - Modelo de arquitetura de um Sistema de Produção Cíber-Físico (adaptado de [54]).

3.3. Modelo de Arquitetura de Referência Indústria 4.0

O Modelo de Arquitetura de Referência Indústria 4.0 (RAMI4.0) foi desenvolvido pela PlatformIndustrie4.0 e a ZVEI e surgiu da necessidade de um modelo de referência para os setores de produção e automação industrial que permitisse estruturar e representar as complexas infraestruturas, conexões, funcionalidades e dinâmicas que caraterizam a I4.0.

De acordo com Adolphs *et al*. [57], o modelo RAMI4.0 (Figura 12) permite:

- descrever os elementos do sistema de produção, designados por *ativos*, de acordo com as suas funcionalidades ou com níveis hierárquicos.
- organizar as várias tecnologias informáticas que servem de base à I4.0, com base nos seus domínios de aplicação e em normas já existentes;
- padronizar e representar as diversas operações e a contínua aquisição de dados, que ocorrem ao longo do ciclo de vida de um produto ou sistema.

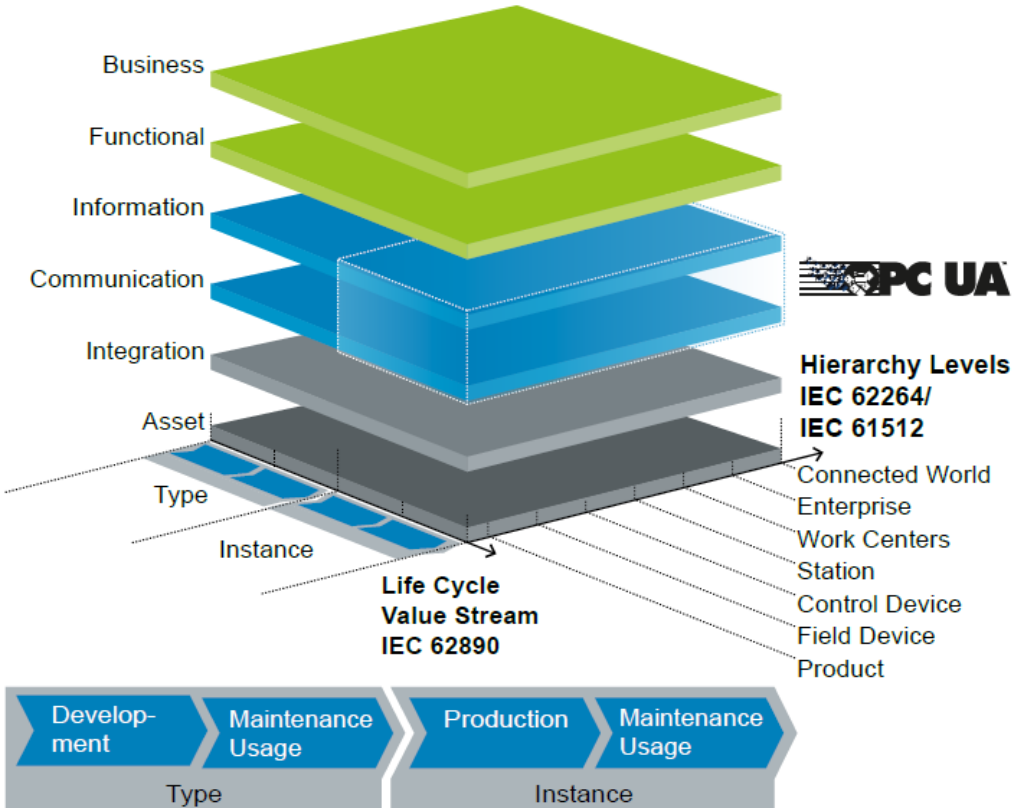


Figura 12 - Representação 3D da Arquitetura de Referência do Modelo Indústria 4.0 [58].

O RAMI4.0 foi apresentado publicamente em 2015, na feira de Hannover, na Alemanha, e foi proposto como norma sob a designação alemã DIN SPEC 91345 e sob a designação internacional IEC PAS 63088 : 2017.

O RAMI4.0 é representado por um modelo 3D (Figura 12) que é constituído, na base, pelo Eixo do Ciclo de Vida do Produto e da Cadeia de Valor e pelo Eixo de Níveis Hierárquicos, e, na vertical, pelo Eixo das Camadas de Tecnologias de Informação e Comunicação.

O **Eixo de Níveis Hierárquicos** representa os elementos do sistema de produção e está organizado de acordo com as normas IEC 62264 - *Enterprise-Control Systems Integration* e IEC 61512 - *Batch Control*. À terminologia definida na norma IEC 62264, a RAMI4.0 acrescenta os termos *Produto*, *Dispositivo de Campo* e *Mundo Conectado*, para descrever novas dinâmicas da I4.0 [57].

O **Eixo de Níveis Hierárquicos** distingue:

- **Produto** - designa um objeto físico, uma peça a ser processada ou um produto acabado, que pode comunicar com o sistema de produção, através de comunicação ativa ou passiva, como, por exemplo, códigos de barras ou QR *codes*.
- **Dispositivo de Campo** - designa um dispositivo, como um sensor ou atuador.
- **Dispositivo de Controlo** - representa unidades de controlo de sistemas industriais, tais como PLCs ou Sistemas de Controlo Distribuído. Estes dispositivos, adaptados à I4.0, devem apresentar novas funcionalidades, nomeadamente: permitirem utilizar Modelos de Informação e uma Arquitetura Orientada para Serviços, permitindo a interoperabilidade e reconfigurabilidade necessárias para implementar o conceito *Plug&Produce*.
- **Estação** - local onde operadores realizam diversas atividades para administrar operações industriais, através, por exemplo, do SCADA.
- **Centro de Trabalho** - descreve a unidade de produção, que monitoriza as informações da produção, define o estado da produção e gere o fluxo de materiais a serem processados.
- **Empresa** - corresponde ao sistema de gestão da empresa como, por exemplo, o ERP.
- **Mundo Conectado** - representa um conjunto de instalações industriais e a sua integração e colaboração com clientes, fornecedores e outras empresas externas.

Na base deste modelo, está também o **Eixo do Ciclo de Vida e da Cadeia de Valor**, que se baseia na norma IEC 62890 - *Life-Cycle-Management for Systems and Component*s e que permite organizar, normalizar e visualizar as operações e relações que compõem o ciclo de vida de um produto ou sistema. Neste eixo, é feita a distinção entre um produto em desenvolvimento, designado por *tipo*, e cada um dos produtos fabricados, designados por *instâncias*.

O *tipo* de um produto é sempre criado na etapa inicial de conceção desse produto. No modelo RAMI4.0, é feita a distinção entre as etapas de desenvolvimento e as etapas de manutenção do *tipo*. O desenvolvimento do *tipo* do produto inclui: conceção, simulações, e a produção dos primeiros protótipos. Concluída a conceção, teste e validação do produto, o seu *tipo* é lançado para ser produzido em série. Em contrapartida, a manutenção do *tipo* do produto está relacionada com a manutenção do modelo virtual que ocorre ao longo do seu ciclo de vida, incluindo modificações, atualizações de *software*, *etc*. [57].

Depois de validado o *tipo*, é iniciada a produção do respetivo produto, sendo que cada produto fabricado representa uma *instância* do produto, que é representada, por exemplo, com um número de série, código de barras, ou QR *code*, único. O modelo RAMI4.0 também distingue entre as etapas de produção e de manutenção de uma *instância*. Na etapa de produção de uma *instância*, são guardados dados relativos ao seu controlo de qualidade e à sua identificação única, entre outros. Na etapa de manutenção da *instância* são guardados, e partilhados, dados relativos às operações de manutenção, otimização ou reciclagem.
Estes dados obtidos ao longo do ciclo de vida, podem ser utilizados para realizar modificações no modelo do *tipo* [57]. Desta forma, o modelo RAMI4.0 permite padronizar e visualizar as diversas operações que ocorrem ao longo do ciclo de vida de um produto e integrar todos os participantes, de forma que todos se compreendam.

Relativamente ao **Eixo Vertical**, este representa a infraestrutura, normas e protocolos de comunicação necessários para implementar a representação digital de um ativo e a integração dos sistemas de informação [57]. Este eixo define as seguintes 6 camadas:

- **Camada do Ativo -** representa objetos físicos como, por exemplo, produtos acabados, peças a serem fabricadas, máquinas, dispositivos ou até documentos. Um requisito essencial para as aplicações da I4.0 é a identificação dos produtos e das respetivas atividades a serem realizadas pelos sistemas de produção, podendo estes objetos apresentarem comunicação passiva, por exemplo, através de QR *codes*. Nesta camada, estão também incluídos os colaboradores humanos.
- **Camada de Integração -** representa as tecnologias que permitem a transmissão de dados entre o domínio físico e o domínio digital, como por exemplo: sensores, leitores de RFID, sistemas de execução de controlo lógico, atuadores, *software* para programação de PLCs e HMI para monitorização do processo e geração de eventos. Esta camada transmite, às camadas superiores, dados relacionados com os ativos técnicos, físicos ou digitais.

- **Camada de Comunicação -** esta camada permite a comunicação entre os elementos conectados à rede, através de protocolos de comunicação, para fornecer às camadas superiores um formato de dados uniforme. Para além disso, fornece *serviços* para controlar a Camada de Integração e é responsável pelas funcionalidades mínimas a que um utilizador deve ter acesso numa infraestrutura de sistemas de informação para a I4.0. Para implementar esta camada, a RAMI4.0 recomenda unicamente o protocolo de comunicação OPC UA que disponibiliza as informações através do seu *address space*.
- **Camada de Informação -** esta camada contém os dados relevantes do ativo ou processo em execução, atualizados em tempo-real, isto é, o seu *run-time enviroment*. Aqui, são definidas e aplicadas regras para efetuar o pré-processamento dos eventos recebidos da Camada de Comunicação e gerar eventos na Camada de Funcionalidade. Nesta camada, as informações provenientes das camadas inferiores são analisadas e sumariadas em estruturas de dados normalizadas, ou seja, num Modelo de Informação. A RAMI4.0 indica, como Modelos de Informação a utilizar, as normas IEC 61360 - *Common Data Dictionary*; IEC 61804 - *Electronic Device Description*; IEC 62453 - *Field Device Tool*; ECLASS; e OPC UA e suas *Companion Specifications*.
- **Camada de Funcionalidade -** esta camada integra os dados do *run-time enviroment* do ativo para interagir com os diversos *serviços*, aplicações, simulações ou processos na Camada do Negócio, estabelecendo a Integração Vertical ou Horizontal. Nesta camada, são definidas as funcionalidades do ativo e também as regras e operações lógicas relacionadas com processos autónomos de tomada de decisão. A RAMI4.0 indica que se implemente esta camada, de acordo com a norma IEC 62769 - *Field Device Integration*, para definir os parâmetros e funcionalidades dos dispositivos.
- **Camada do Negócio -** esta camada garante a integração das funções dos ativos, ao longo do ciclo de vida, de acordo com os processos de negócio definidos nesta camada, a partir dos eventos que são recebidos da camada de Funcionalidade. Aqui são definidos parâmetros comerciais, como custos de produção, tempo de entrega, parâmetros de controlo de qualidade, para automatizar e descentralizar os processos e orquestrar serviços na Camadas de Funcionalidade.

A RAMI4.0 apresenta também as definições de *comunicação*I4.0 e de *componente*I4.0. A comunicaçãoI4.0 designa conceitos novos para a indústria, que são:

- **Modelos de Informação** para descrever os dispositivos e as suas capacidades, de forma padronizada;

- **Arquitetura Orientada a Serviços**. Este tipo de protocolos de comunicação permite que os equipamentos passem a disponibilizar na rede as suas capacidades como *serviços*.

Um pré-requisito para a comunicaçãoI4.0 é uma rede de comunicação baseada no Protocolo de Internet. Existindo uma rede IP, o OPC UA pode ser aplicado como uma interface padrão, para máquinas de diferentes fornecedores [58].

Para implementar a RAMI4.0, um ativo tem de ser considerado uma *entidade*, ou seja, este têm de apresentar: (1) - um identificador único (exemplos: código de barras, QR *code*, ou endereço IP); (2) - uma representação virtual; e (3) - capacidade de comunicação (passiva, ativa, ou comunicaçãoI4.0), o que é feito através de uma Consola de Administração (AAS).

Se um, ou mais, ativos estiveram conectados, através de um sistema de informação, à Consola de Administração que o(s) representa, esse conjunto forma um componenteI4.0, como mostra a Figura 13. Um componenteI4.0, através da sua AAS, disponibiliza permanentemente à rede as informações do seu estado atual, a sua representação virtual, documentação técnica e as suas funcionalidades na forma de *serviços* [55, 57].

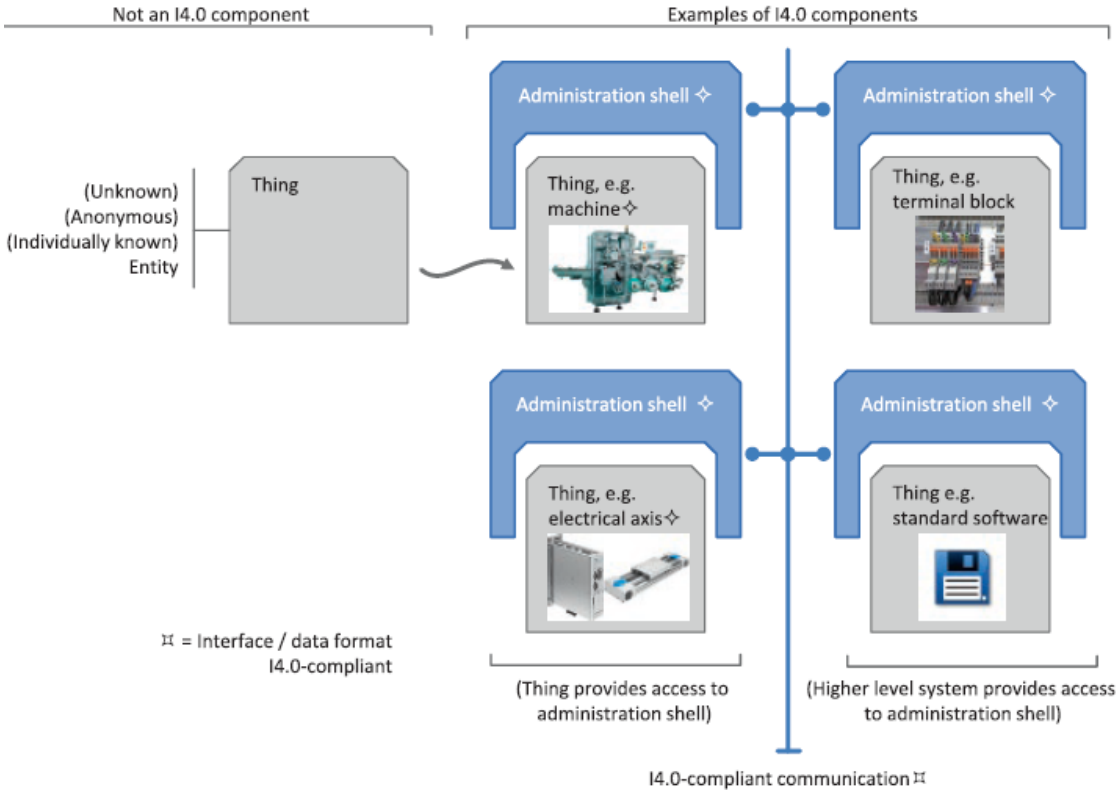


Figura 13 - Exemplos de componentesI4.0 [57].

3.4. Protocolo Open Platform Communications Unified Architecture

O final do século 20 caraterizou-se pela difusão de sistemas de automação baseados em computadores e em *software* e pela proliferação de protocolos e interfaces proprietárias para dispositivos industriais, o que condicionava o acesso aos dados e a interoperabilidade.

Em 1995, um conjunto de empresas formou um grupo de trabalho para desenvolver uma forma de acesso normalizada aos dados dos sistemas de automação [60]. Em 1996, este grupo publicou a sua primeira norma, designada por OPC *Data Access*. Esta especificação foi implementada com base nas tecnologias *Component Object Model* (COM) e *Distributed Component Object Model* (DCOM) do sistema operativo Windows, que permitem que um *cliente* possa chamar *métodos* e aceder aos dados de *objetos*-COM presentes no sistema local ou em outros computadores ligados à mesma rede. Desta forma, foi possível desenvolver esta norma, apenas especificando as APIs adaptadas aos vários dispositivos industriais [60]. Assim surgiu o acrónimo OPC, que inicialmente significava *Object Linking & Embedding for Process Control*, mas, de forma a acompanhar a evolução desta tecnologia, hoje designa *Open Platform Communications* [61].

Em 1996, este consorcio fundou a OPC Foundation [4], cujas funções são: desenvolver as respetivas normas; colaborar para desenvolver modelos de informação relativos a outras normas industriais; e prestar serviços de certificação. Hoje, a OPC Foundation integra mais de 920 membros, incluído todos os grandes fornecedores de sistemas de automação [60].

Nessa época, foram também publicadas outras normas OPC que funcionam segundo o modelo *cliente*/*servidor* e que, em conjunto, se designam por OPC *Classic*.

Em 2006, a OPC Foundation publicou uma nova norma designada por *Open Platform Communications Unified Architecture* (OPC UA) e definida pelo padrão IEC 62541.
O OPC UA é um protocolo de comunicação baseado numa Arquitetura Orientada a Serviços.

Segundo a OPC Foundation [62], o OPC UA carateriza-se por:

- conter todas as funcionalidades do OPC *Classic*;
- incluir um rico modelo de base, *meta model*, que define as regras de semântica, estrutura e comportamento, para definir os diversos modelos de informação OPC UA;
- permitir descobrir aplicações OPC UA a correr no mesmo computador ou na mesma rede;
- incluir mecanismos integrados de segurança, encriptação, autentificação e auditoria;

4 - https://opcfoundation.org/

- permitir os modelos de comunicação *cliente*/*servidor* e *publicador*/*subscritor*;
- funcionar em inúmeras plataformas de *hardware*, como PLCs, PCs, servidores na *cloud*, microcontroladores; e de *software*, como Microsoft Windows, MacOS, Android e Linux.

Atualmente, a norma OPC UA é composta por 24 partes (Tabela 3). Todas as partes desta norma podem ser acedidas livremente no *website* da OPC Foundation.

Tabela 3 - Partes que constituem a norma OPC UA [63].

Parte da Norma OPC UA	Título
Parte 1 (OPC 10000-1)	*Overview and Concepts*
Parte 2 (OPC 10000-2)	*Security Model*
Parte 3 (OPC 10000-3)	*Address Space Model*
Parte 4 (OPC 10000-4)	*Services*
Parte 5 (OPC 10000-5)	*Information Model*
Parte 6 (OPC 10000-6)	*Mappings*
Parte 7 (OPC 10000-7)	*Profiles*
Parte 8 (OPC 10000-8)	*Data Access*
Parte 9 (OPC 10000-9)	*Alarms and Conditions*
Parte 10 (OPC 10000-10)	*Programs*
Parte 11 (OPC 10000-11)	*Historical Access*
Parte 12 (OPC 10000-12)	*Discovery and Global Services*
Parte 13 (OPC 10000-13)	*Aggregates*
Parte 14 (OPC 10000-14)	*PubSub*
Parte 15 (OPC 10000-15)	*Safety*
Parte 16 (OPC 10000-16)	*State Machines*
Parte 17 (OPC 10000-17)	*Alias Names*
Parte 18 (OPC 10000-18)	*Role-Based Security*
Parte 19 (OPC 10000-19)	*Dictionary References*
Parte 20 (OPC 10000-20)	*File Transfer*
Parte 21 (OPC 10000-21)	*Device Onboarding*
Parte 22 (OPC 10000-22)	*Base Network Model*
Parte 23 (OPC 10000-23)	*Common ReferenceTypes*
Parte 24 (OPC 10000-24)	*Scheduler*

Conforme se observa na Figura 14, o protocolo OPC UA é definido por três camadas, que são: combinações de protocolos, modelos de comunicação e modelos de informação. Na base estão os mecanismos e formatos utilizados para garantir a codificação, transporte e segurança dos dados. Acima estão os modelos *cliente/servidor* e *publicador/subscritor*. No topo estão os modelos de informação OPC UA.

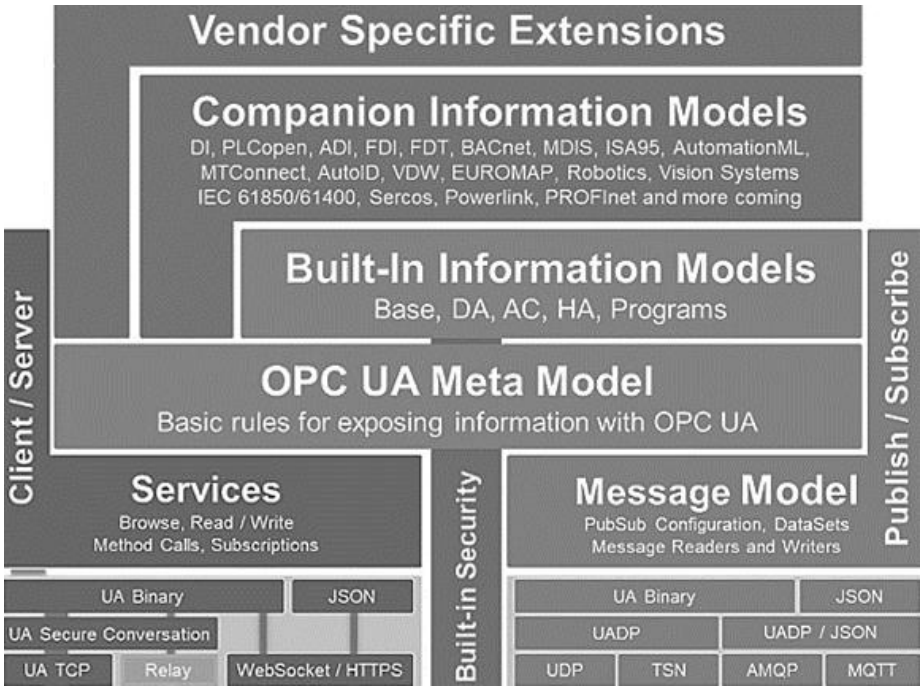


Figura 14 - Representação da arquitetura do protocolo OPC UA e seus modelos de informação [64].

Um meta-modelo é um modelo que serve de base para definir e descrever outros modelos. O meta-modelo OPC UA baseia-se em conceitos orientados a *objetos*, inclui modelos de *objeto*, *nó* e *classes de nó*, e permite definir *subtipos* e *hierarquias* de *tipos*, isto é, define as regras para definir e expor informação, incluindo dados complexos e metadados [65]. O meta-modelo OPC UA designa-se por *Address Space* (AS), é detalhado na parte 3 da sua norma, e é nele que todos os Modelos de Informação OPC UA se baseiam.

O OPC UA também contém o seu Modelo de Informação, que é definido na parte 5 da sua norma e integra *tipos de objeto*, *tipos de referências*, *tipos de variáveis*, *tipos de dados* e *métodos*, para descrever diferentes tipos de dados, expor a *qualidade de serviço* e suportar os seus serviços de *Data Access*, *Alarm&Conditions* ou *Historical Data Access* [66].

A OPC Foundation coopera com outras organizações para criar Modelos de Informação adicionais, designados por *Companion Specification* (CS), para casos específicos de setores industriais. Cada CS tem atribuído um ficheiro .pdf com a sua descrição e um ficheiro .xml

que o define, ambos disponíveis no *website* da OPC Foundation [5, 6]. Cada CS acrescenta a sua especificação no *address space*. Alguns exemplos de CS estão indicados na Tabela 4 e também nas Figuras 14, 15 e 16.

Tabela 4 - Exemplos de *companion specifications* [58].

Companion Specification	Descrição
Auto ID	Modelo de Informação que serve para integrar os sistemas de identificação automática, tais como QR *codes* ou *Near-Field Communication*.
CNC Systems	Modelo de Informação que serve para integrar sistemas CNC.
EUROMAP 77	Modelo de Informação que serve para integrar máquinas de injeção de plástico nos sistemas MES e SCADA.
Field Device Integration	Modelo de Informação que serve para integrar dispositivos de campo.
Robotics	Modelo de Informação que serve para integrar robôs nos sistemas industriais de produção.

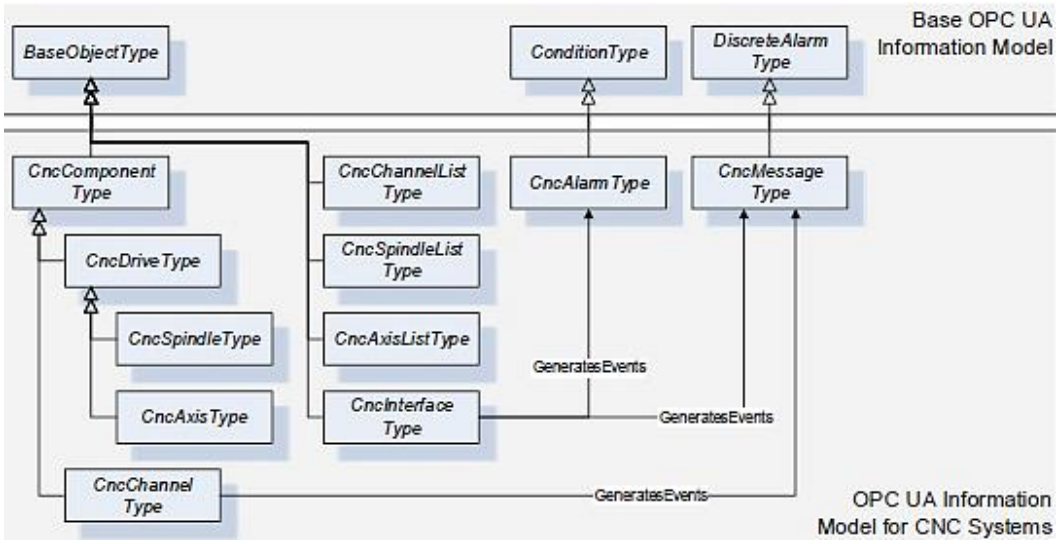


Figura 15 - Representação da companion specification "OPC UA for CNC Systems" [10].

A adoção de uma *companion specification* permite estabelecer uma interface padrão para uma determinada categoria de dispositivos e, assim, integrar equipamentos de diferentes fornecedores, tal como se pode verificar no exemplo ilustrado na Figura 16.

5 - https://opcfoundation.org/about/working-groups/
6 - https://files.opcfoundation.org/schemas/

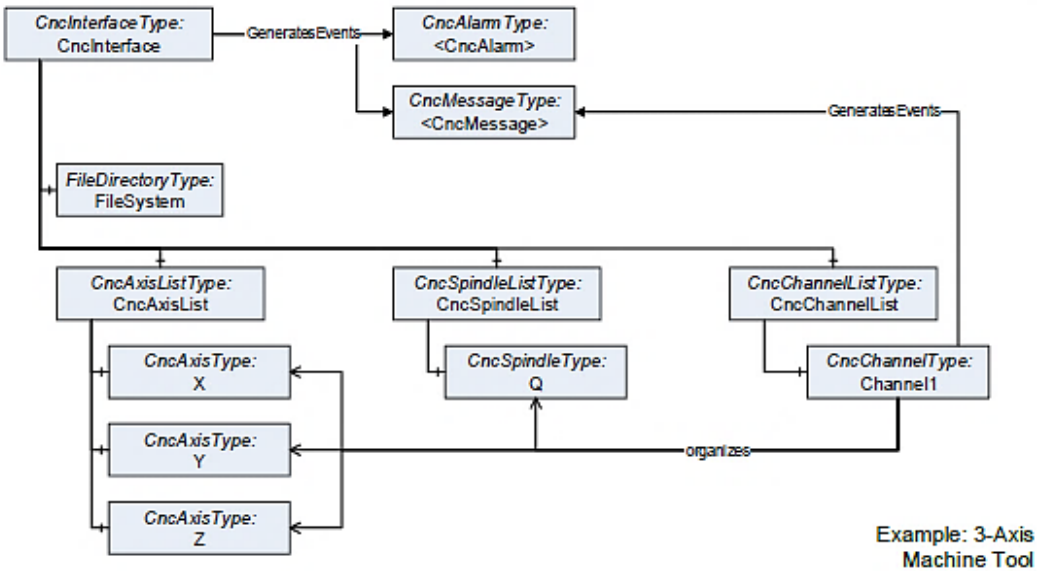


Figura 16 - Aplicação da CS "CNC Systems" para descrever uma máquina-ferramenta CNC [10].

Finalmente, com base numa *companion specification*, um vendedor ou utilizador, por exemplo, tem a possibilidade de criar a sua Extensão Específica, acrescentando, assim, o seu modelo de informação personalizado.

Mahnke *et al*. [60] descrevem a arquitetura de uma aplicação OPC UA (Figura 17), como o conjunto das seguintes camadas de *software* :

- **Communication Stack** ou **Pilha de Comunicação** - é um conjunto de bibliotecas de *software* que estabelecem a ligação entre a aplicação e o *hardware*; permitem invocar *serviços* na rede; codificar, encriptar e formatar uma mensagem a enviar; ou descarregar, desencriptar e descodificar uma mensagem recebida.
- **Software Development Kit** (SDK) - é uma camada de *software* construída no topo da *stack* que disponibiliza as APIs específicas para permitir a comunicação entre servidor, cliente e *stack*. Tanto a SDK como a *stack* são camadas de *software* que lidam com tarefas de baixo nível (SDK para a camada de aplicação e *stack* para a de transporte) e permitem ocultar a complexidade desnecessária para o desenvolvimento de aplicações. As APIs variam consoante a linguagem de programação, sistema operativo e *stack* utilizada.
- **Aplicação** - designa o código que carateriza o servidor OPC UA ou cliente OPC UA, e as suas funcionalidades, de alto nível, utilizando a respetiva API específica.

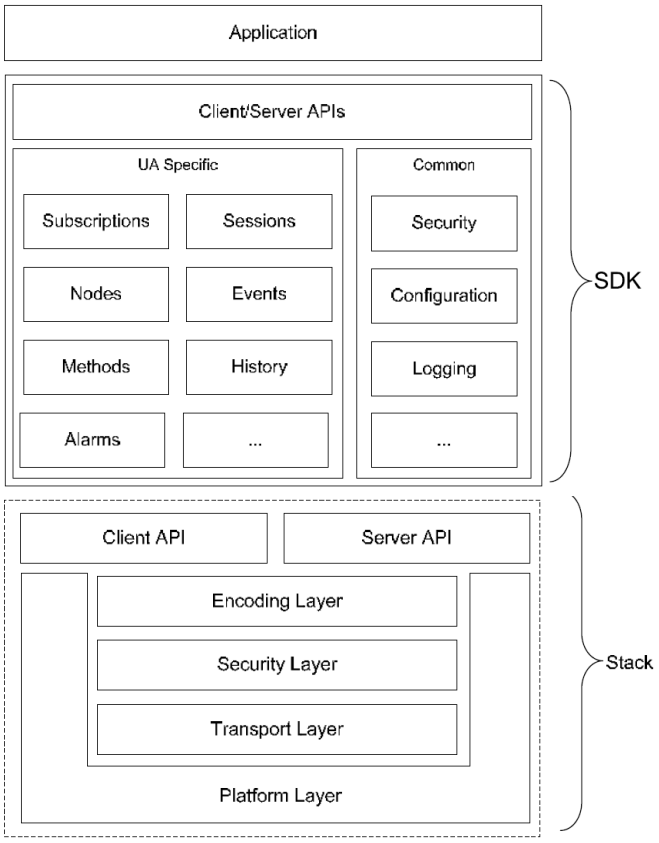


Figura 17 - As 3 camadas de software que compõem uma aplicação OPC UA (adaptado de [60]).

Existem diversas implementações, em código aberto, da *stack* OPC UA, algumas já com SDKs integradas, como open62541 em C/C++, UA.NETStandard em C#, entre outras [7]. A OPC Foundation disponibiliza livremente várias *stacks* OPC UA [8].

3.5. Consola de Administração do Ativo

Em 2016, a ZVEI introduziu a Consola de Administração do Ativo (*Asset Administration Shell*, AAS) que consiste numa ferramenta de *software* que permite criar as representações virtuais padronizadas dos ativos, administra a sua comunicaçãoI4.0, permite arquivar e transferir diversos formatos de ficheiros e integra mecanismos de identificação, segurança, auditoria e gestão de direitos de acesso. Assim, a RAMI4.0 passou a definir componenteI4.0 como um ativo conectado à sua AAS. Neste âmbito, a AAS representa a ferramenta utilizada para criar os padronizados gémeos digitais dos ativos [63, 64]. A especificação da AAS pode ser consultada em [69].

7 - https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations
8 - https://github.com/OPCFoundation

De acordo com a PlattformIndustrie4.0 [70] e Ye *et al*. [71], as AAS são classificadas numa de três categorias (Figura 18), consoante a seu modo de interação, nomeadamente:

- **AAS passiva -** consiste num pacote de ficheiros, onde as várias informações do ativo são guardadas segundo estruturas de dados padronizadas, em linguagem XML ou *JavaScript Object Notation* (JSON). Para além disso, esta serve como um repositório para diversos formatos de ficheiros.
- **AAS reativa -** para além das características anteriores, esta carateriza-se pela capacidade de trocar informações com outras AASs ou aplicações de *software* através de uma API. Estas APIs são ligadas por protocolos como, por exemplo, *Hyper Text Transfer Protocol* (HTTP), MQTT ou *Transmission Control Protocol* (TCP) / *Internet Protocol* (IP). Tipicamente, esta interação é estabelecida segundo o modelo *cliente*/*servidor*.
- **AAS proativa -** carateriza-se por poder autonomamente comunicar e cooperar com outras AASs, utilizando uma semântica e sintaxe padronizadas. Esta interação entre AASs é estabelecida através do modelo de comunicação *peer-to-peer*.

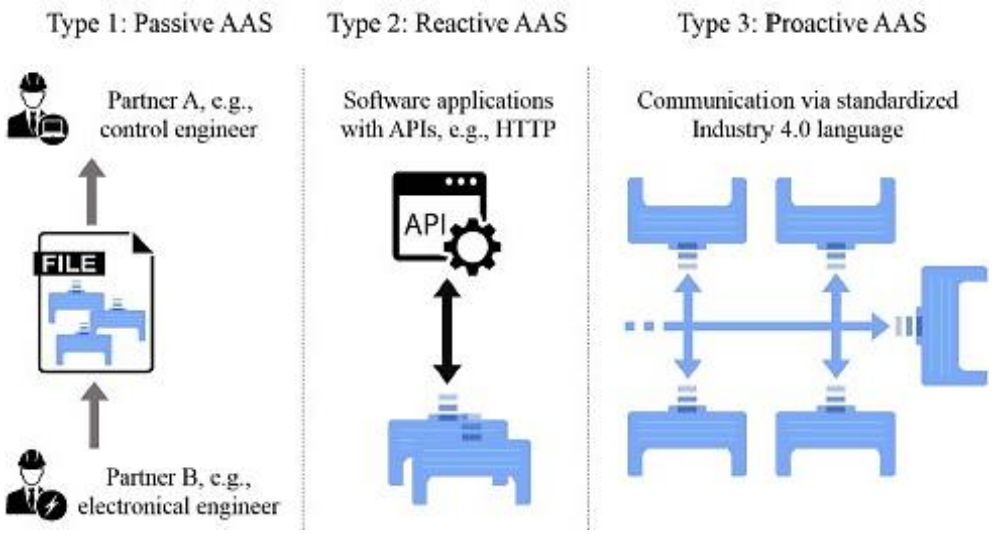


Figura 18 - As três categorias de Consola de Administração do Ativo [71].

No âmbito da Fabricação Digital Direta, destaca-se o trabalho de Erp *et al*. [68] que consistiu no desenvolvimento de um modelo conceptual para a troca de *serviços* entre ativos e na sua demonstração através da implementação, ao nível dos equipamentos, de uma AAS para uma impressora3D e outra AAS para um robô móvel e, ao nível do sistema, uma AAS para a célula de produção. Desta forma, implementaram a troca de *serviços* entre estes ativos através da definição de condições lógicas e integração dos IDs semânticos únicos das AASs, e respetivos *submodelos*, na AAS da célula de produção.

4. Tecnologias

4.1. ProLIGHT1000

A primeira especificação deste projeto foi a seleção do equipamento industrial. Selecionou-se a máquina-ferramenta CNC ProLIGHT1000 porque era a única disponível no laboratório de Robótica Avançada e Fábricas Inteligentes da ESTG e, no entanto, adequada para o efeito pretendido. A ProLIGHT1000 (Figura 19) é um modelo de fresadora CNC fabricado na década de 1990 pela empresa Light Machines Corporation, que hoje corresponde à empresa Intelitek [9].

Este equipamento carateriza-se por [72] :

- possuir 3 eixos de movimento (graus de liberdade) para maquinação;
- possuir uma mesa de trabalho cuja área disponível é 495,30 mm x 158,75 mm;
- permitir até 5000 RPMs no seu *spindle*;
- utilizar códigos G&M para a execução de trabalhos de maquinação;
- permitir realizar trabalhos de maquinação com troca automática de ferramentas.

Figura 19 - ProLIGHT1000 presente na ESTG.

4.2. EdingCNC

A ProLIGHT1000 sofreu algumas modificações nos seus sistemas de controlo, antes de se iniciarem os trabalhos deste projeto de mestrado. Uma dessas modificações foi a substituição do *software* de controlo WPL1000 (fornecido junto com o equipamento) pelo EdingCNC (versão 4.03.60) (Figura 20), que é um *software* de controlo para máquinas-ferramenta CNC desenvolvido pela empresa EdingCNC [10]. Esta modificação foi feita para poder atualizar a máquina com um *software* mais moderno e, principalmente, porque este *software* disponibiliza uma API, ao contrário do WPL1000. Além disso, a carta de controlo original da ProLIGHT1000 foi substituída pela carta CPU5A3, também desenvolvida pela empresa EdingCNC, porque estes produtos de *software* e de *hardware* são interdependentes.

9 - https://intelitek.com
10 - https://edingcnc.com

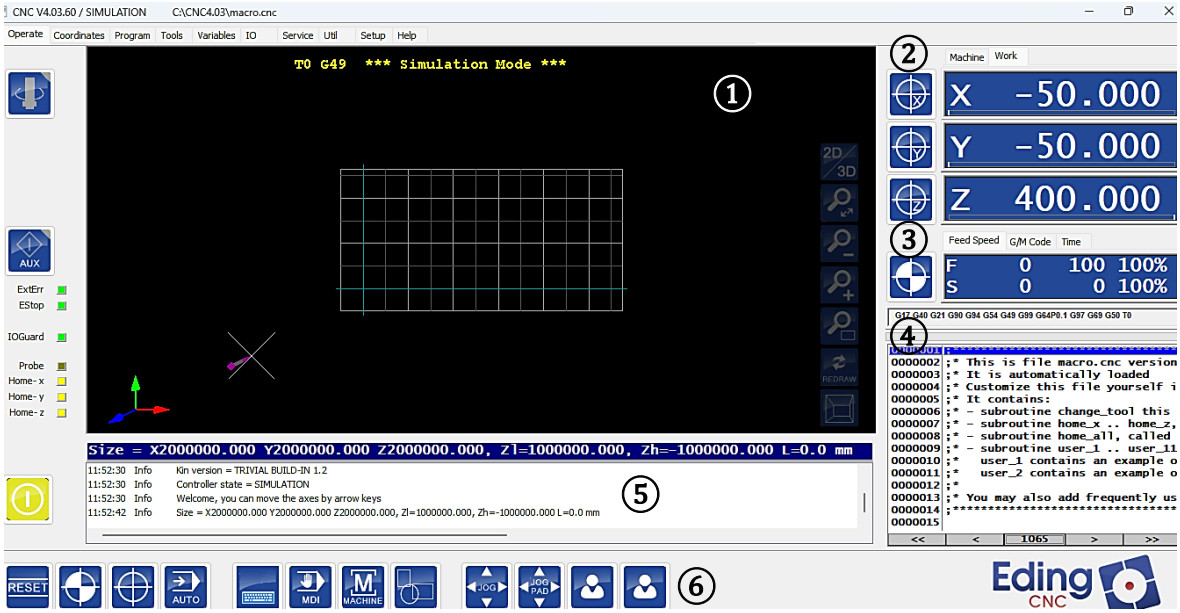


Figura 20 - Interface gráfica do software EdingCNC.

A Figura 20 mostra a interface gráfica do *software* EdingCNC, na qual se distingue:

1 - secção que permite visualizar as trajetórias de maquinação;

2 - secção que indica os valores das atuais coordenadas para os eixos X, Y e Z;

3 - secção que indica o valor do avanço da ferramenta e da velocidade de rotação do *spindle*;

4 - secção que mostra o código do ficheiro *macro.cnc* ou programa de maquinação CNC;

5 - secção que mostra as mensagens de aviso ou de erro;

6 - secção que contém diversos botões para operar a máquina, através desta interface.

O programa EdingCNC disponibiliza uma API, designada por *cncapi*, desenvolvida em C/C++, que permite aceder às funções do seu componente CNCServer, responsável pela gestão da maquinação e pela ligação à placa de controlo. É através desta API que a sua interface gráfica se interliga com o CNCServer [73]. O CNCServer interpreta os programas de maquinação e transmite-os à carta de controlo que, por sua vez, aciona controladores dos motores dos eixos X, Y e Z e *Spindle* e gere o controlo lógico da máquina, como os sinais de fim de curso ou sinal de emergência.

Nesta máquina, também foi acrescentado um microcontrolador *Arduino*, para controlar o seu sistema de troca automática de ferramenta, como mostram as Figuras 21 e 22. O motivo desta modificação é que a carta CPU5A3 não disponibiliza saídas suficientes para gerir, além dos motores e controlo lógico, o sistema de troca automática de ferramenta.

Figura 21 - Microcontrolador Arduino Mega 2560 [74].

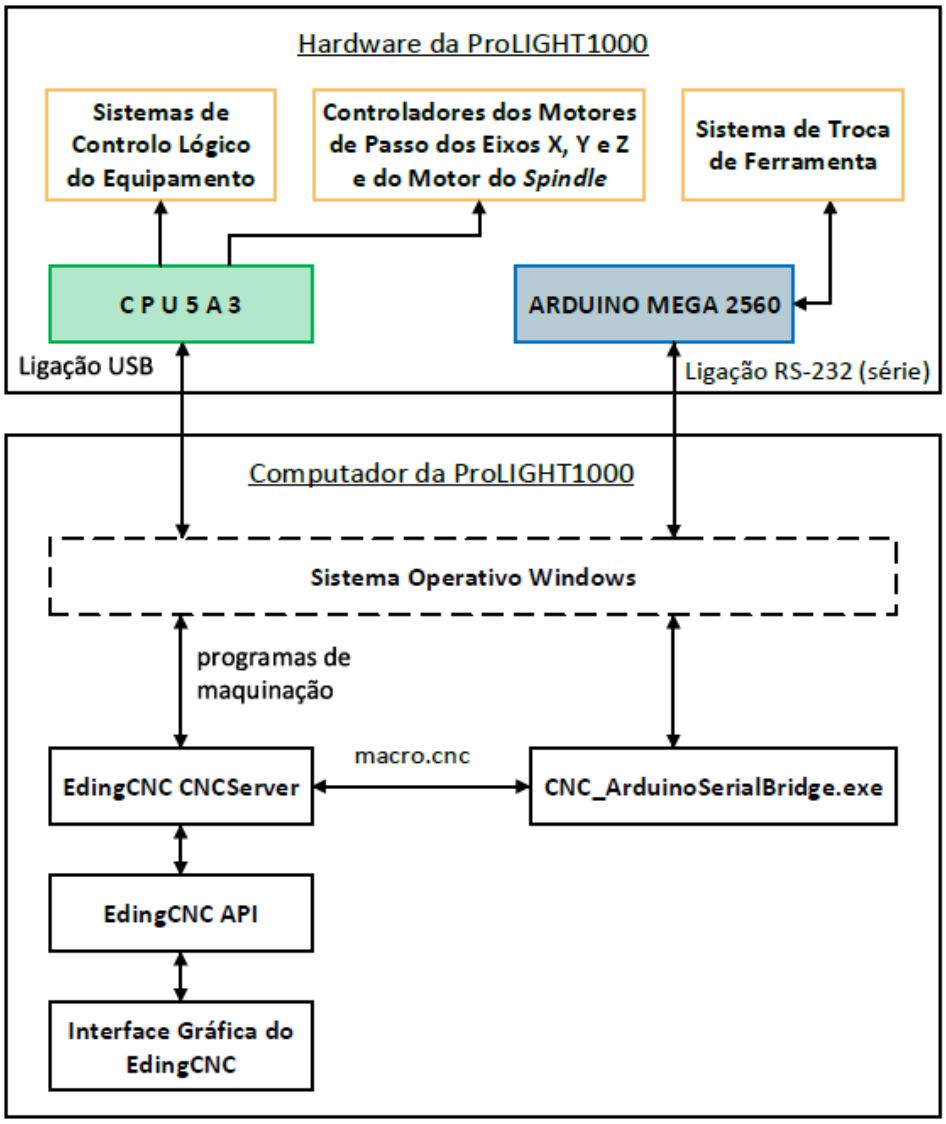


Figura 22 - Sistemas de controlo da ProLIGHT1000 (antes de se iniciar este projeto de mestrado).

O ficheiro *macro.cnc* é um ficheiro de texto editável a que o interpretador de programas CNC do EdingCNC recorre para chamar as diversas sub-rotinas de maquinação [73].

4.3. Sistema de Troca Automática de Ferramenta

O sistema de troca automática de ferramenta aplicado na ProLIGHT1000 e os ficheiros que esse sistema utiliza foram desenvolvidos antes de se iniciar este projeto de mestrado.

Para implementar este sistema no computador da ProLIGHT1000, na diretoria da pasta *CNC4.03* que contém o programa EdingCNC, foi modificado o ficheiro *macro.cnc* e foram guardados os ficheiros *CNC_ArduinoSerialBridge.exe* e *PortaCOM.txt* (Figuras 23 e 24).

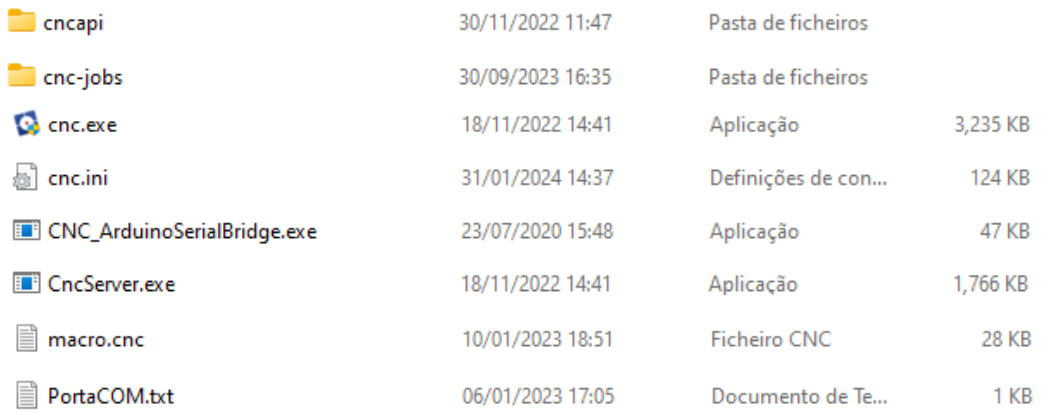


Figura 23 - Excerto do conteúdo da pasta *CNC4.03* .

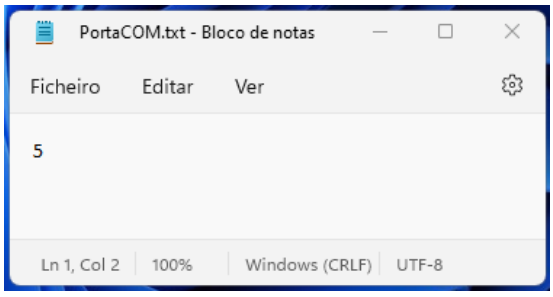


Figura 24 - Ficheiro *PortaCOM.txt* .

O ficheiro *PortaCOM.txt* é um ficheiro de texto que contém apenas um número, o que permite facilmente identificar, ou alterar, a porta COM utilizada para comunicar com o *Arduino*.

O ficheiro *macro.cnc* foi editado para que cada sub-rotina da troca automática de ferramenta chame o programa *CNC_ArduinoSerialBridge.exe*, introduza um determinado caractere e espere, neste caso até 5 segundos, que termine a execução desse programa. Assim, o interpretador do EdingCNC, ao ler o comando M6 T2, chamará, por exemplo, as sub-rotinas Arrumar Ferramenta 1 (DropTool1) e Colocar Ferramenta 2 (PickTool2) e cada uma destas sub-rotinas chamará o programa *CNC_ArduinoSerialBridge.exe* que irá escrever, na porta COM, uma certa sequência de caracteres que será lida e interpretada pelo *Arduino*.

A Figura 25 mostra um excerto do ficheiro *macro.cnc*, onde se vê uma parte da sub-rotina DropTool1 que comanda subir a plataforma de armazenamento para arrumar a ferramenta 1 através do comando `EXEC "CNC_ArduinoSerialBridge.exe" "@1up#" 5000`.

```
Sub DropTool1
    msg "Dropping tool 1"
    msg "Moving for position"
    G53 G1 Z-78 F600;
    G53 Y268.5;
    G53 X474;
    msg "Tool Holder 1 Up"
    #5399 = 0;
    EXEC "CNC_ArduinoSerialBridge.exe" "@1up#" 5000
    ...
```

Figura 25 - Sub-rotina DropTool 1.

4.3.1. CNC_ArduinoSerialBridge

O ficheiro *CNC_ArduinoSerialBridge.exe* é um ficheiro que foi desenvolvido em C/C++, cujo código tem como argumentos de entrada o `vetor de strings` utilizado no ficheiro *macro.cnc* para o chamar, que lhe é fornecido através do sistema operativo. A Figura 26 mostra as instruções do programa *CNC_ArduinoSerialBridge.exe*.

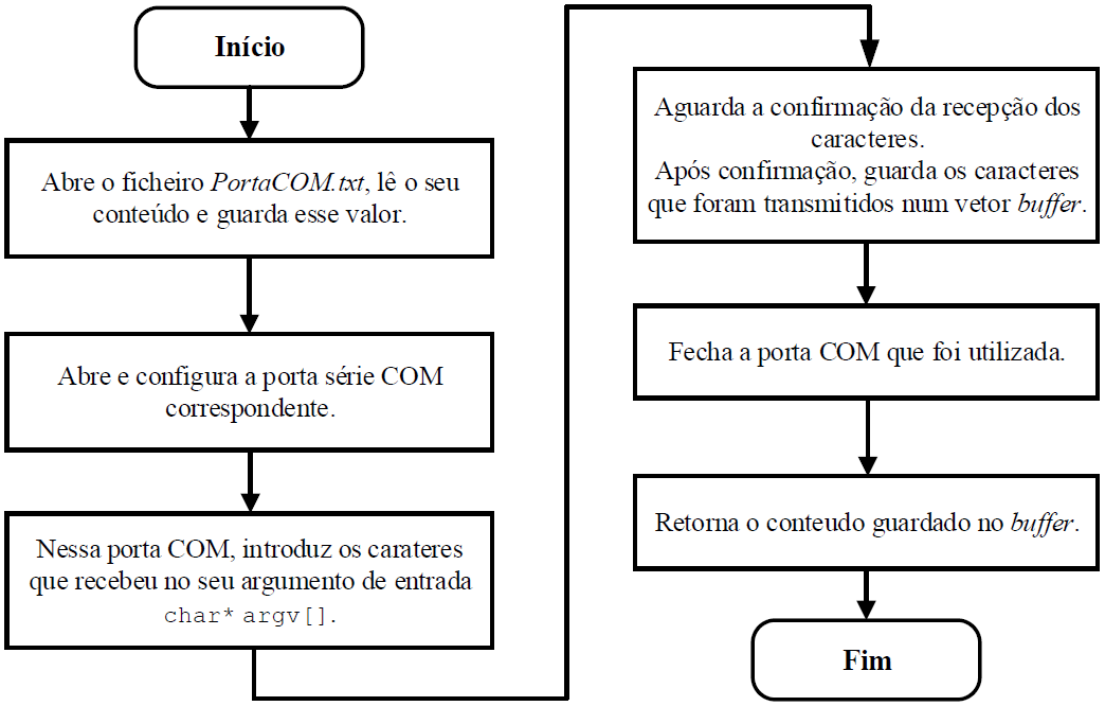


Figura 26 - Fluxograma das instruções do programa *ArduinoSerialBridge.exe*.

4.3.2. Programa no Arduino

Para que o *Arduino* possa interpretar a mensagem que lhe foi transmitida pelo programa *CNC_ArduinoSerialBridge.exe*, nele foi guardado o programa *AutomaticToolChanger.ino*, que foi desenvolvido através do *software Arduino IDE* em C/C++. Este programa permite controlar as electroválvulas pneumáticas da ProLIGHT1000 para subir, ou descer, as plataformas de armazenamento de ferramenta e abrir, ou fechar, a cabeça do *spindle*, consoante a sequência de caracteres que lê. Este programa identifica os pinos do *Arduino* cujos valores de tensão elétrica acionam as electroválvulas ou indicam o estado dos sensores e, além disso, associa os caracteres "@" e "#" ao início e ao fim, respetivamente, de uma mensagem recebida e, por fim, implementa uma máquina de estados (Figura 26).

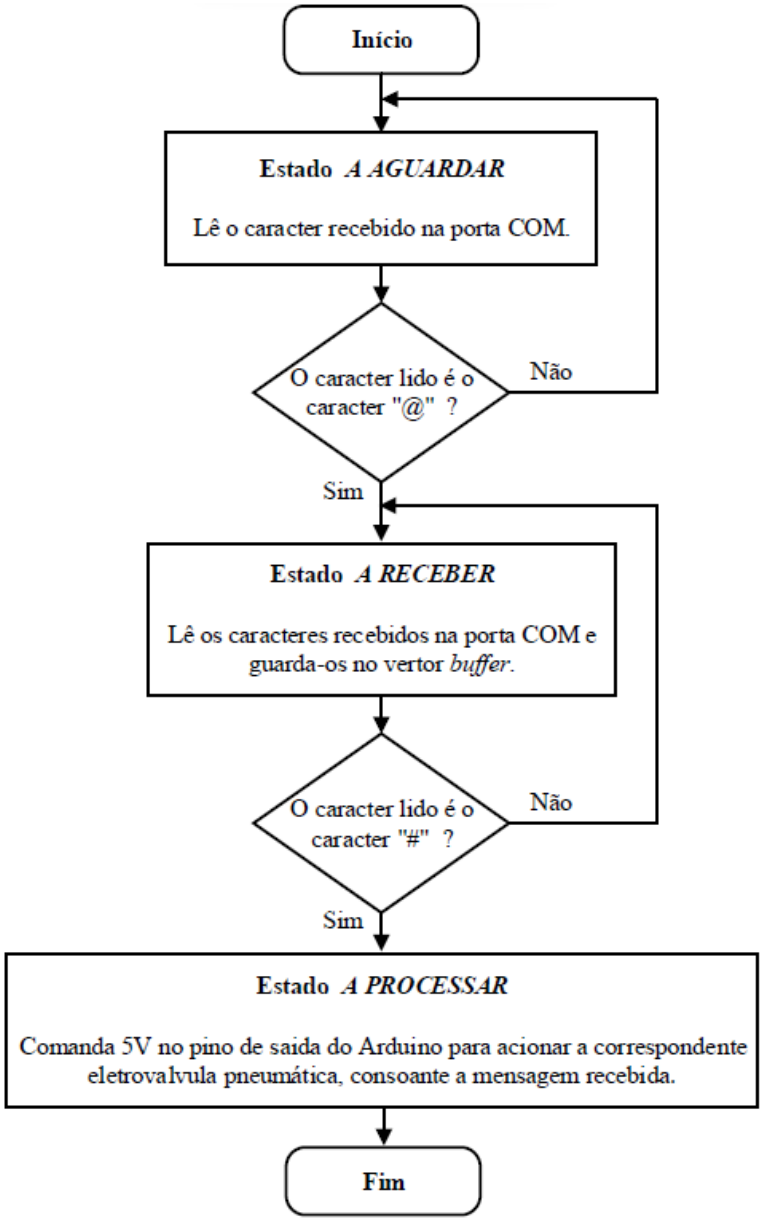


Figura 27 - Máquina de estados do programa *AutomaticToolChanger.ino*.

4.4. Protocolo OPC UA

4.4.1. Address Space

A norma OPC UA define *address space* (AS) como "o conjunto de *objetos*, e respetivas informações, que um servidor disponibiliza para os clientes" [63]. O *address space* é um modelo para representar informações, que é estruturado, padronizado, orientado a *objetos*, fortemente tipificado e que permite descrever exaustivamente produtos, processos ou equipamentos, para que todos os dispositivos conectados possuam a mesma interface, não só ao nível de descrição do *tipo de dados* mas também do *tipo de objetos*.

O OPC UA define o elemento básico que compõe e organiza o *address space*, o *objeto* (Figura 23), que pode conter *variáveis*, *métodos* e *referências* a outros *objetos* [65]. Todos os *objetos* que compõem o *address space* referem o seu *tipo de objeto* e são definidos por *nós* (*nodes*) que, por sua vez, consoante a *classe de nó*, são descritos por determinados *atributos* (uns obrigatórios e outros opcionais) e interconectados por *referências* [65]. O *nó* (Figura 24) é a unidade atómica, utilizada construir todas as outras entidades no AS.

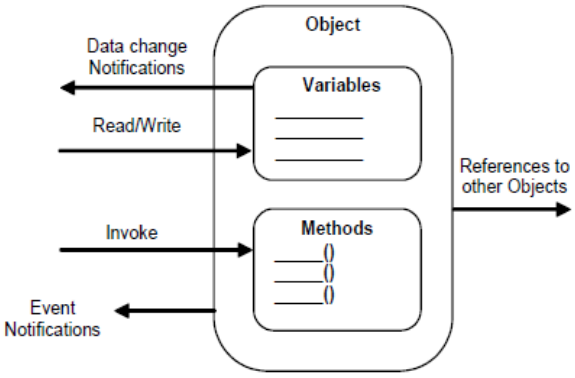


Figura 28 - Modelo de objeto do OPC UA [65].

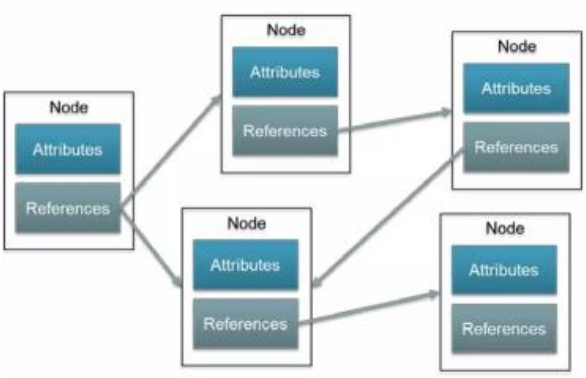


Figura 29 - Elementos (*nós* / *nodes*) que modelam as informações contidas num *address space* [75].

A parte 3 da norma OPC UA define uma *classe de nó*, a *BaseNodeClass*, da qual todas as *classes de nó* são derivadas, ou seja, herdam os seus *atributos* e podem acrescentar outros. A *BaseNodeClass* contém os seguintes *atributos* obrigatórios [65] :

- *BrowseName* - Nome utilizado para navegar pelo *address space* e criar diretorias;
- *DisplayName* - Nome utilizado para exibir o nome do *nó* ao utilizador;
- *NodeClass* - Enumeração que identifica a *classe do nó*;
- *NodeId* - Identificador único do *nó*.

A norma OPC UA define as seguintes *classes de nó* (*NodeClass*) [65] :

- *DataType* - define a estrutura dos dados, para os valores de uma classe *Variable*;
- *Method* - elemento utilizado para definir funções executáveis;
- *Object* - elemento que representa um sistema, componente, objeto físico ou de *software*;
- *ObjectType* - elemento que define um objeto genérico que pode ser instanciado;
- *ReferenceType* - elemento que define o significado das *referências* entre *nós*.
- *Variable* - elemento que contém um valor;
- *VariableType* - elemento utilizado para definir tipos para a classe *Variable*;
- *View* - elemento que restringe o acesso de um *cliente* a um definido subconjunto de *nós*;

A norma OPC UA refere que cada *nó* é identificado por um *NodeId* que é composto pelo *index* do seu *namespace* e um valor, em `string` ou `int`, do seu identificador único [65], tal como mostra a Figura 25. Cada *namespace* corresponde a um domínio de *nós* [65].

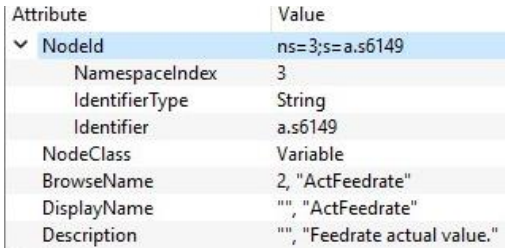


Figura 30 - Exemplo de um *NodeId*, visualizado através do cliente OPC UA UAExpert.

Para modelar um *address space* de acordo com uma *companion specification*, deve-se recorrer ao ficheiro .xml que esta fornece e que contém o seu modelo de informação.
Associar um conjunto de *objetos* a um *namespace* e URI, como mostra a Tabela 5, permite que estes se diferenciem, no caso de serem adicionados *objetos* que tenham nomes iguais.
No exemplo da Tabela 5, o *namespace* de *index* 0 contém o Modelo de Informação OPC UA do qual outros *nós* irão derivar. O *namespace* de *index* 1 é específico do servidor local. O último *namespace* diz respeito à CS utilizada e contém o respetivo Modelo de Informação.
É também possível adicionar outros *namespaces* personalizados, baseados nos anteriores

Tabela 5 - Namespaces obrigatórios para um servidor modelado segundo a CS "CNC Systems" [10].

Namespace	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowsNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for *Nodes* defined in the local *Server*. This may include types and instances used in a device represented by the *Server*. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/CNC/	Namespace for *NodeIds* and *BrowsNames* defined in this specification. The namespace index is *Server* specific.	Mandatory

4.4.2. Servidores e Clientes

A Figura 31 mostra os elementos que compõem as aplicações servidor e cliente OPC UA.

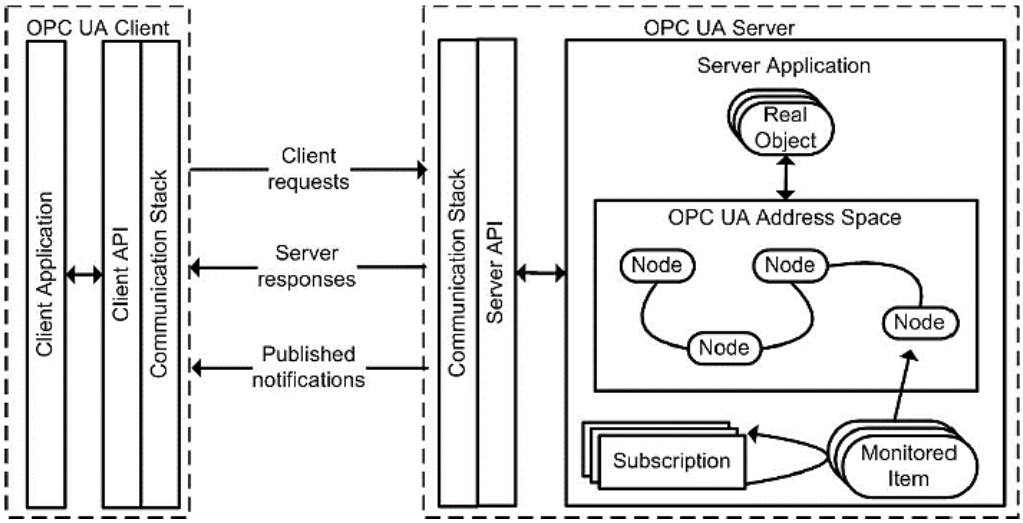


Figura 31 - Aplicações servidor OPC UA, cliente OPC UA e a forma como estas interagem [76].

Uma aplicação servidor OPC UA encapsula, através da *stack* e sua API, os objetos reais, que são dispositivos físicos ou *software*, e disponibiliza as informações no seu *address space*. A aplicação cliente OPC UA encontra as URLs das aplicações OPC UA disponíveis na rede e estabelece um canal seguro e uma sessão com o servidor, aplicando os mecanismos de autentificação e encriptação definidos pelo servidor [60]. A aplicação cliente utiliza a sua API para se ligar à *stack*, receber e enviar informações de e para o servidor e navegar pelo *address space*.

4.5. Asset Administration Shell

Uma AAS serve para conter, de forma padronizada, as informações, propriedades e dados de um ativo e encapsular as suas funcionalidades [77]. A estrutura da AAS é baseada na norma IEC TS 62832 - 1 : 2016 - *Digital Factory Framework*, é organizada num cabeçalho (*header*) e num corpo (*body*) e é constituída por um *manifesto* e um *gestor de componentes*, como se vê na Figura 27 [78].

O corpo da AAS pode conter um conjunto de diferentes *submodelos* padronizados, que servem para descrever o ativo em diversos domínios, como identificação, documentação técnica, referência de subcomponentes, ficheiros CAD, *Plug&Produce* ou descrição das suas *capacidades* (como, por exemplo, maquinação ou transporte) que recorrem a *métodos* e desencadeiam eventos, permitindo, assim, automatizar e otimizar a utilização do ativo [78].

O *manifesto* é uma lista das propriedades contidas no cabeçalho ou corpo da AAS, tais como, as identificações do ativo e da AAS ou referências dos submodelos incluídos [78].

O *gestor de componentes* gere os submodelos da AAS e respetivas propriedades, dados ou funcionalidades e é o elo de ligação do componenteI4.0 a serviços de TIC, para que este possa integrar-se numa Arquitetura Orientada a Serviços ou em repositórios de AASs [79].

O meta-modelo da Consola de Administração do Ativo está representado na Figura 28.

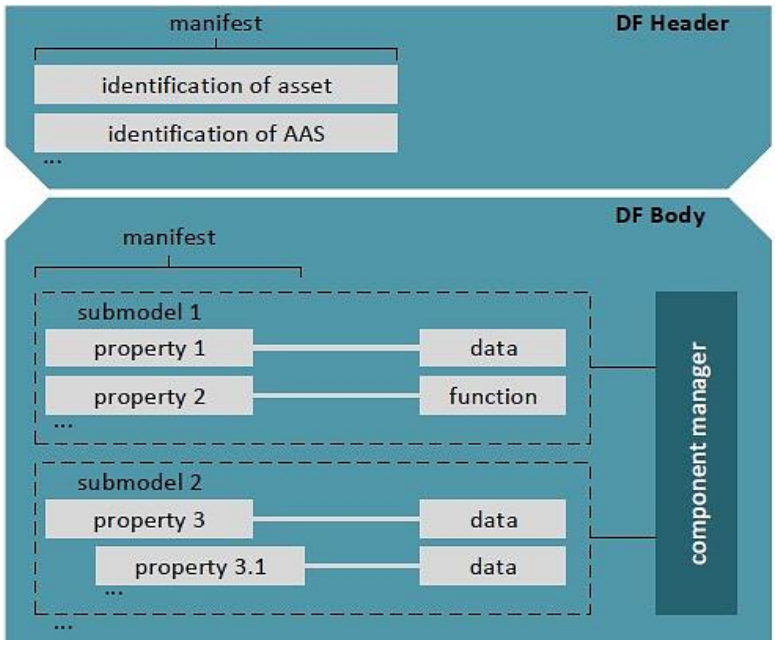


Figura 32 - Estrutura da Consola de Administração do Ativo [77].

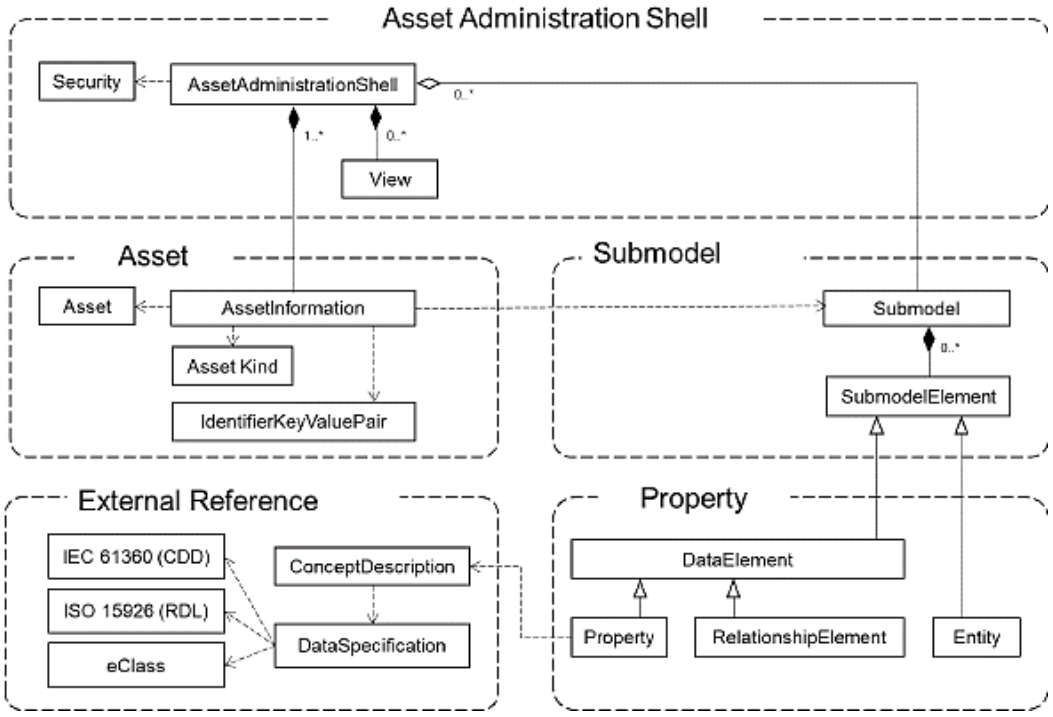


Figura 33 - Meta-modelo da Consola de Administração do Ativo [69].

Cada *submodelo* contém uma quantidade estruturada de *propriedades*, que são *modelos de dados* padronizados para descrever o ativo [69]. Cada AAS, ativo, *submodelo*, *classe* e *propriedade* tem uma identificação inequívoca [78]. Os ativos são associados a *classes* e as suas caraterísticas são descritas por *propriedades*. Estes termos são definidos por dicionários padronizados, como, por exemplo, o ECLASS, IEC 61360 - *Common Data Dictionary* ou o IEC 15926 - *Reference Data Library* [78].

Vários submodelos padronizados e alguns exemplos da sua aplicação estão disponíveis nos *websites* da Industrial Digital Twin Association (IDTA) [11, 12 e 13]. Em 2020, a IDTA foi fundada pela PlattformIndustrie4.0, VDMA, ZVEI, BITKOM e 20 grandes empresas alemãs do setor industrial [79, 80]. O objetivo da IDTA é centralizar, gerir e padronizar o desenvolvimento dos gémeos digitais e impulsionar a implementação da tecnologia AAS, no âmbito da I4.0 [80, 81].

A PlattformIndustrie4.0 especificou um formato de pacote de ficheiros, cuja extensão é .aasx, para representar uma AAS [69]. Atualmente existem diversas aplicações de *software* que implementam as especificações da tecnologia AAS, para aceder e editar ficheiros .aasx.

O AASX Package Explorer [14] é um *software* desenvolvido em C# e disponível em fonte aberta, que disponibiliza uma interface gráfica para editar e visualizar informações de AASs. Este *software* integra mecanismos de segurança e várias APIs, como REST, MQTT e OPC UA, o que permite que outras aplicações possam aceder às informações das AASs [71].

Existem outras implementações da especificação da AAS em diferentes linguagens de código, tais como BaSyx, NOVAAS, entre outras [83].

Por fim, refere-se o AASX Server [15], que é um *software* desenvolvido em C# e disponível em fonte aberta, que implementa um servidor que permite aceder às informações de um repositório de ficheiros .aasx. O AASX Server integra mecanismos de segurança e pode ser conectado através dos protocolos REST, MQTT e OPC UA [71].

11 - https://industrialdigitaltwin.org/en/content-hub/submodels
12 - https://admin-shell-io.com/samples/
13 - https://github.com/admin-shell-io/submodel-templates
14 - https://github.com/admin-shell-io/aasx-package-explorer
15 - https://github.com/admin-shell-io/aasx-server

5. Desenvolvimento do Projeto

Este projeto consistiu em adaptar a fresadora CNC ProLIGHT1000, presente no Laboratório de Robótica Avançada e Fábricas Inteligentes da Escola Superior de Tecnologia e Gestão do Politécnico de Leiria, ao paradigma indústria 4.0. A Figura 29 apresenta o sistema que foi desenvolvido e a forma como este interage com os sistemas de controlo da ProLIGHT1000.

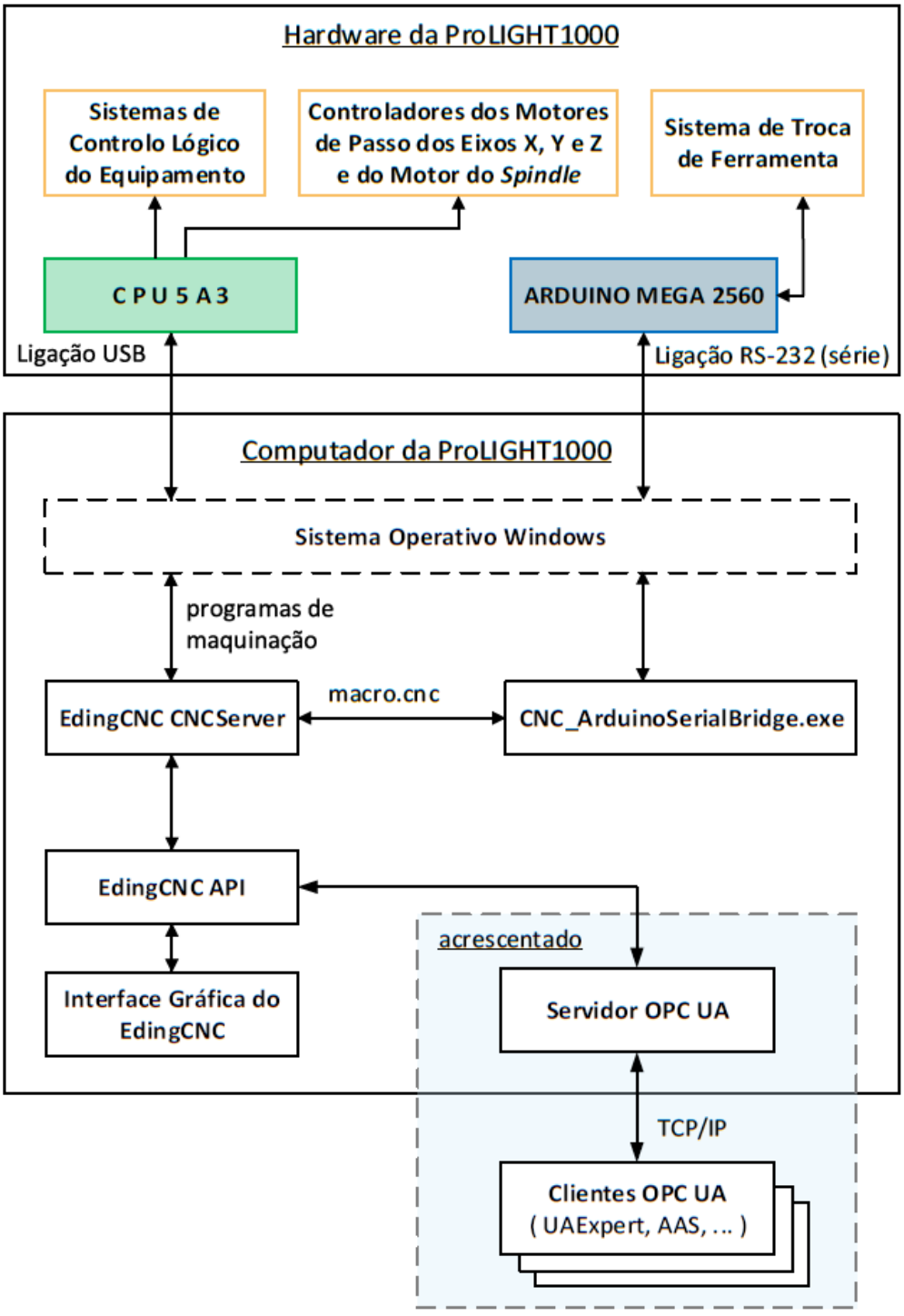


Figura 34 - Sistema acrescentado aos sistemas de controlo da ProLIGHT1000.

5.1. Seleção da stack OPC UA

Optou-se por utilizar a implementação do OPC UA através da *stack* open62541 [16] porque :

- esta *stack* foi desenvolvida em linguagem de programação C99;
- está disponível de forma gratuita, em fonte aberta e está bem documentada;
- inclui as SDKs para *servidor* e *cliente* integradas;
- permite a compilação para vários sistemas operativos, tendo-se optado pelo Windows, aquele em que a interface do EdingCNC está desenvolvida;

O facto de a *stack* open62541 ter sido desenvolvida em C99 foi um fator decisivo para a sua seleção porque, assim, esta é capaz de comunicar diretamente com a API do EdingCNC.

Lucas [21] desenvolveu um servidor OPC UA a partir da *stack* UA.NETStandard em C# e conectou-o à API do EdingCNC, recorrendo a um *code wrapper* desenvolvido por Sander Oosterhof. Esta abordagem foi descartada porque este *code wrapper* foi descontinuado, é incompatível com a última versão do EdingCNC (4.03.60) e já não se encontra disponível.

5.2. Modelação do address space

Um servidor OPC UA criado a partir de uma *stack* contém apenas uma estrutura mínima de *address space*. Assim, surgiu a necessidade de modelar o *address space*, isto é, definir a organização de uma estrutura de *nós* que correspondam às pretendidas *instâncias* de *objetos*, com o intuito de descrever a fresadora CNC ProLIGHT1000. Este trabalho baseou-se no trabalho de Lucas [21] e na *companion specification* "OPC UA for CNC Systems" que indica as especificações a adotar para descrever sistemas CNC.

Para modelar o *address space*, optou-se utilizar o *software* UAModeler, fornecido pela empresa Unified Automation [17], aproveitando o conhecimento anterior desta ferramenta no Laboratório de Robótica Avançada e Fábricas Inteligentes da ESTG. O UAModeler é um *software* que serve para modelar, através da sua interface gráfica, a estrutura de um *address space* e exportar essa informação em código. A Figura 30 mostra o *address space* que foi modelado e exportado para um ficheiro em linguagem XML que se designou por *cnc*1.*xml*. Após se compararem os *tipos* de *objetos* definidos pela CS com os valores retornados pelas funções do EdingCNC, atribuiu-se um valor a cada *objeto* deste *address space*, exceto aos *objetos* CNCAlarm e CNCMessage porque não foram considerados relevantes.

16 - https://www.open62541.org/
17 - https://www.unified-automation.com/

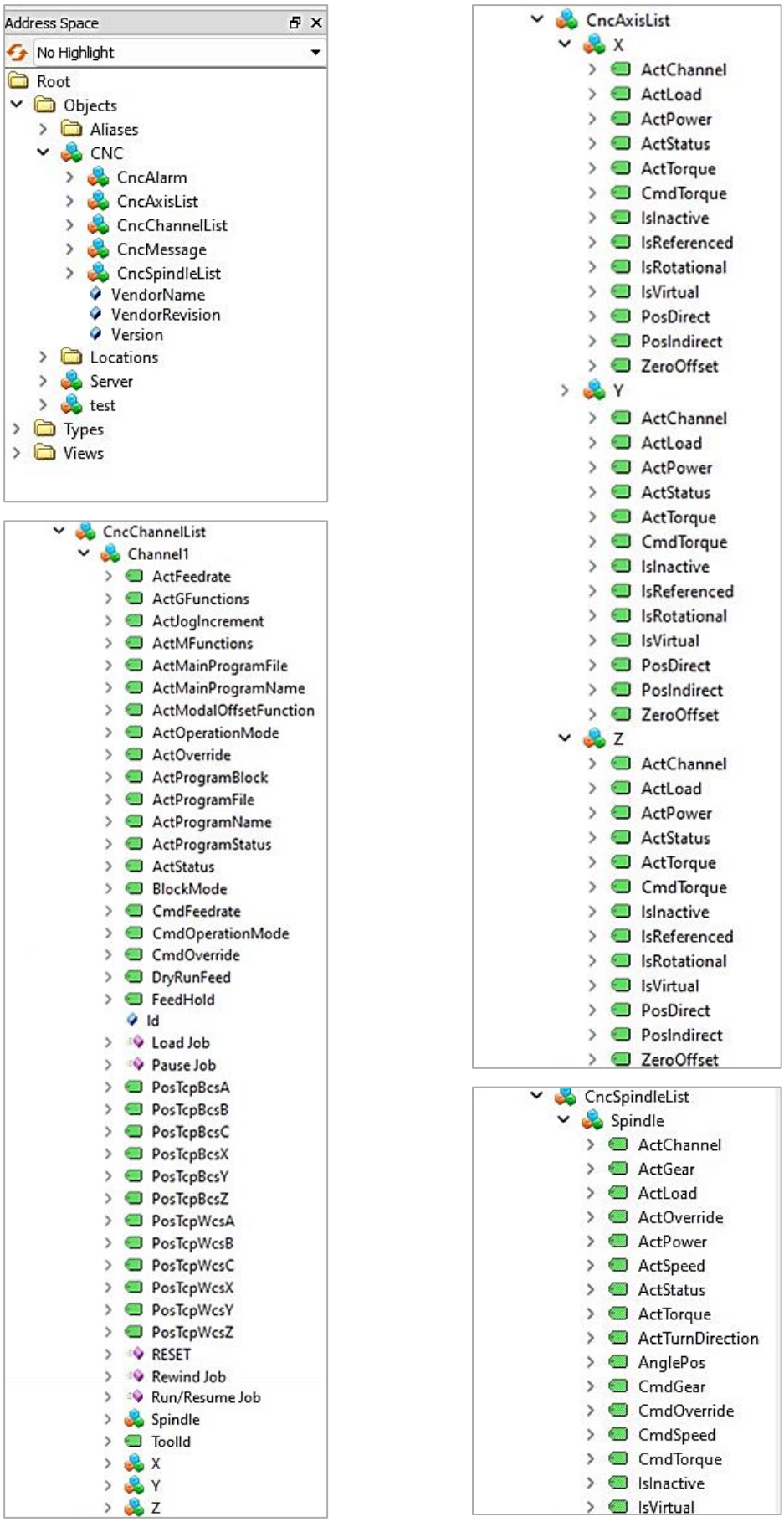


Figura 35 - Diferentes vistas do *address space* modelado, visualizadas no UAExpert.

5.3. Projeto de Software

Para desenvolver este projeto, utilizaram-se os *softwares* CMake e Microsoft Visual Studio. O CMake é uma ferramenta de desenvolvimento de *software* que permite, em múltiplas plataformas, gerir as configurações de projetos de *software*, em código C/C++ [84].
O Microsoft Visual Studio é um ambiente integrado para desenvolvimento de *software* [85].

Começou-se por utilizar o CMake para compilar as bibliotecas de *software* que compõem a *stack* open62541, de onde resultaram os ficheiros *open62541.c* e *open62541.h* . De seguida, utilizando o CMake, compilaram-se os ficheiros *open62541.c* e *open62541.h*, os ficheiros que definem o *nodeset* da *companion specification* "OPC UA for CNC Systems" (*Opc.Ua.CNC.NodeSet.bsd*, *Opc.Ua.CNC.NodeSet.csv* e *Opc.Ua.CNC.NodeSet.xml*) e o ficheiro que contém o *address space* modelado (*cnc*1.*xml*) numa solução de *software* em C++. Por fim, através do Microsoft Visual Studio, esta solução foi editada e compilada.

5.3.1. Desenvolvimento da classe CNCBase e da classe CNCEding

Lucas [21] e Martins [22] desenvolveram, em código C#, uma *classe* que representa um controlador CNC genérico e *classes* derivadas que implementam os *métodos* específicos das diferentes marcas de controladores CNC. Desta forma, criaram um modelo de dados comum e uma interface genérica, onde cada marca de controlador CNC responde aos *métodos* da *classe* genérica com as suas funções específicas, concretizando, assim, uma solução *P&P*. Para além disso, com alterações mínimas a um servidor base que contenha estas *classes*, é possível obter servidores para diferentes controladores, já que a implementação específica de cada marca de controlador CNC é encapsulada numa *classe* derivada específica.

Neste projeto, optou-se por aplicar a abordagem *P&P* desenvolvida por Lucas [21] e Martins [22], mas em C++ porque o EdingCNC disponibiliza a sua API nessa linguagem.

Através do Visual Studio, criou-se uma *classe* que se designou de CNCBase, que declara vários *métodos* abstratos para representar um controlador CNC genérico e da qual podem ser derivadas *classes* específicas para cada marca de controlador CNC que definem esses *métodos* recorrendo às funções das APIs dos respetivos *softwares* de controlo. A seguir, desenvolveu-se uma *classe*, derivada da CNCBase, que se designou de CNCEding e que a cada um dos seus *métodos* associa a chamada de uma função da API do EdingCNC. A Figura 36 ilustra a relação entre a *classe* CNCBase e a *classe* CNCEding.

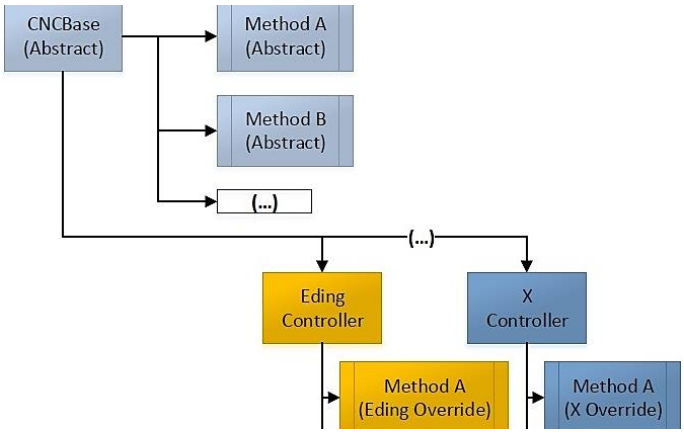


Figura 36 - Representação da *classe* CNCBase e da *classe* CNCEding (adaptado de [22]).

A Figura 37 mostra um excerto da *classe* CNCBase, onde se vê a declaração de um *método* abstrato e genérico que, como os demais *métodos* nela declarados, se apresenta vazio. Essa figura mostra o *método* `Get_tcp_wcs_x_actpos`, que é utilizado para chamar o valor do parâmetro "coordenada X em relação ao Zero-Peça" a qualquer controlador CNC.

A implementação dos *métodos* da *classe* CNCBase é feita pela *classe* CNCEding. A Figura 38 mostra um excerto da *classe* CNCEding, onde o *método* `Get_tcp_wcs_x_actpos` é definido pela chamada da função `CncGetWorkPosition().x` da API do EdingCNC.

```cpp
#pragma once
#ifndef CNCBASE_H
#define CNCBASE_H
using namespace std;
class CNCBase {
      public: double Get_tcp_wcs_x_actpos();
      ...
}
```

Figura 37 - Excerto do código da *classe* CNCBase.

```cpp
#pragma once
#include "CNCBase.h"
#include "cncapi.h"

class CNCEding : public CNCBase {
      public: double Get_tcp_wcs_x_actpos() {
      double WorkCoordtsX = CncGetWorkPosition().x;
      return WorkCoordtsX;
      };
      ...
}
```

Figura 38 - Excerto do código da *classe* CNCEding.

5.3.2. Ficheiro XML acessório para o servidor OPC UA

Para este projeto, criou-se em linguagem XML um ficheiro acessório que se designou como *cnc.xml*. Este ficheiro consiste numa lista de configurações, onde o conteúdo de cada linha se apresenta organizado segundo a estrutura **<add key="Parâmetro" value="Valor"/>**.

Depois de o servidor OPC UA ter sido iniciado, este lê o ficheiro *cnc.xml* e guarda em memória os valores dos diversos parâmetros de configuração, o que permite, de forma fácil, alterar o seu modo de funcionamento, sem a necessidade de editar o seu código e recompilar.

A Figura 39 mostra um excerto do ficheiro *cnc.xml*. As primeiras linhas deste ficheiro indicam a marca do controlador CNC e o seu endereço IP e, eventualmente, as configurações de dispositivos conectados ao servidor OPC UA como, por exemplo, uma câmara de vídeo. Por fim, para cada *método* da *classe* CNCBase, este ficheiro contém uma linha caraterizada pela estrutura **<add key="NomeDoMétodo" value="NodeId, Atualiza, Frequência"/>** . Cada método está, assim, associado a uma `string` "NomeDoMétodo", que o identifica, e a uma `string` "NodeId, Atualiza, Frequência" que será dividida pelas suas "," e o resultado será atribuído aos respetivos `int` *NodeId*, `bool` *Atualiza* e `int` *Frequência*.

```
<parameters>
  <add key="CNC_CONTROLLER" value="EDING"/>
  <add key="CNC_IP" value="192.168.56.101"/>
  <add key="CNC_PORT" value="19000"/>
  <add key="Get_VendorName" value="6001,false,1"/>
  <add key="Get_Version" value="6003,false,1"/>
  <add key="Get_X1_ActStatus" value="6057,true,1"/>
  <add key="Get_X1_Referenced" value="6065,true,1"/>
  <add key="Get_Ch1_PosTcpWcsX_Act" value="6237,true,1"/>
  ...
</parameters>
```

Figura 39 - Excerto do código do ficheiro *cnc.xml* .

Recorrendo às funções `UA_Server_addRepeatedCallBack( )` e `switch(...) case ... :`, o servidor OPC UA irá executar periodicamente cada *método* cujo `bool` *Atualiza* seja `true` e o valor retornado será escrito no *nó* identificado pelo respetivo `int` *NodeId*.

Assim, editando as linhas do ficheiro *cnc.xml*, facilmente se pode configurar qual a *classe* específica aplicada ou quais os *métodos* da *classe* genérica cujos valores retornados serão periodicamente atualizados e escritos nos correspondentes *nós* do servidor OPC UA.

5.3.3. Código da aplicação servidor OPC UA

Através do Visual Studio, foi desenvolvido o código do servidor OPC UA, em C/C++. A Figura 40 mostra a sequência de instruções que carateriza o código do servidor OPC UA desenvolvido.

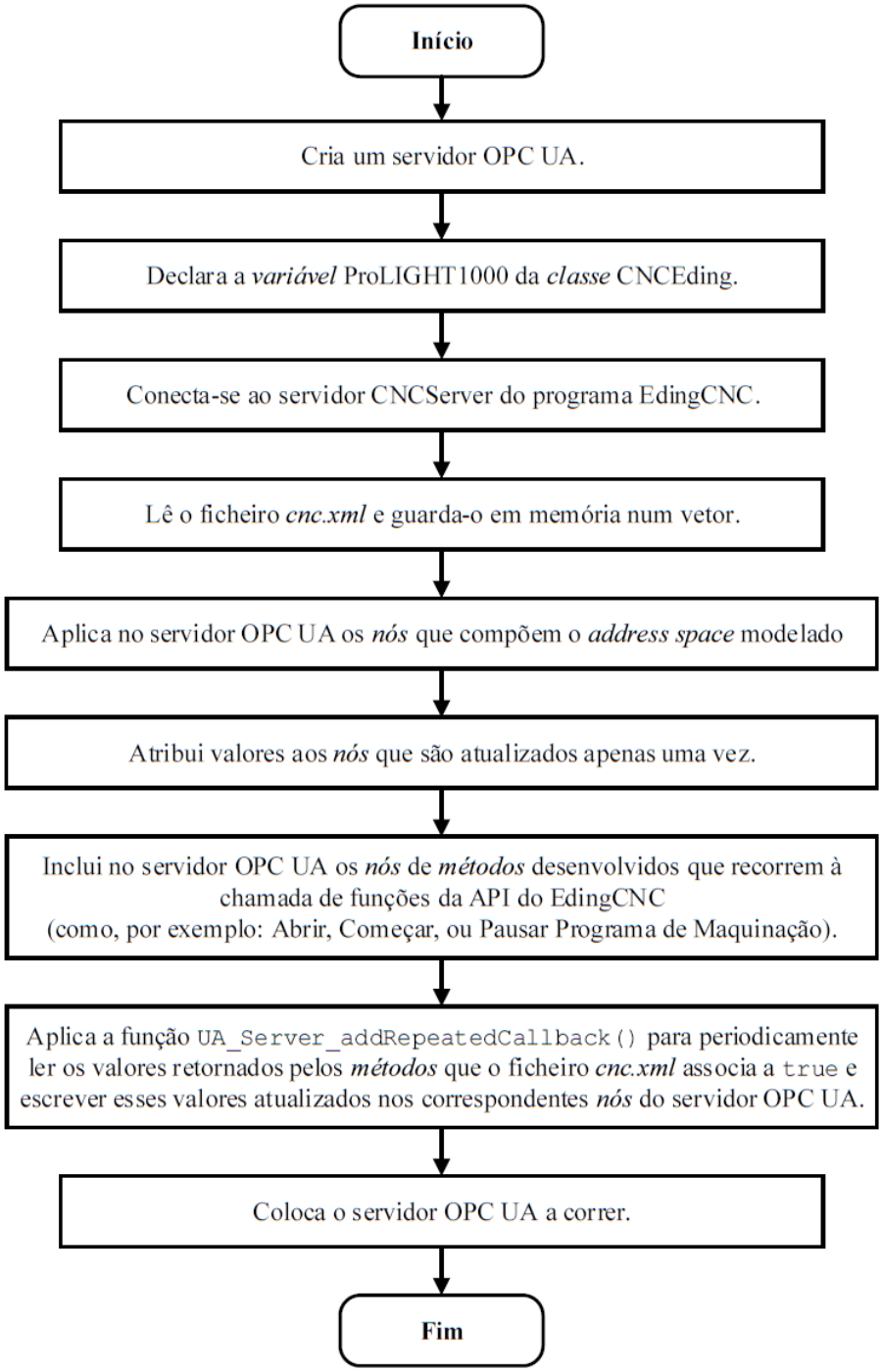


Figura 40 - Fluxograma das instruções da aplicação servidor OPC UA.

5.4. Implementação de uma Asset Administration Shell

Para criar um gémeo digital da fresadora CNC ProLIGHT1000, implementou-se uma *Asset Administration Shell*, o que foi feito recorrendo ao programa AASX Package Explorer para manter a compatibilidade com outros ativos no laboratório, uma vez que esse programa já sido aplicado em projetos anteriores.

Nesta AAS incluíram-se vários submodelos padronizados para referenciar diferentes aspetos do ativo como, por exemplo, os contactos do seu fabricante ou documentação técnica, como mostra a Figura 41. Nesta AAS foi incluído o submodelo *Operational Data*, que serve para continuamente ler os dados do funcionamento de um ativo. Os *nós* que compõem o *address space* do servidor OPC UA foram associados às *propriedades* desse submodelo.

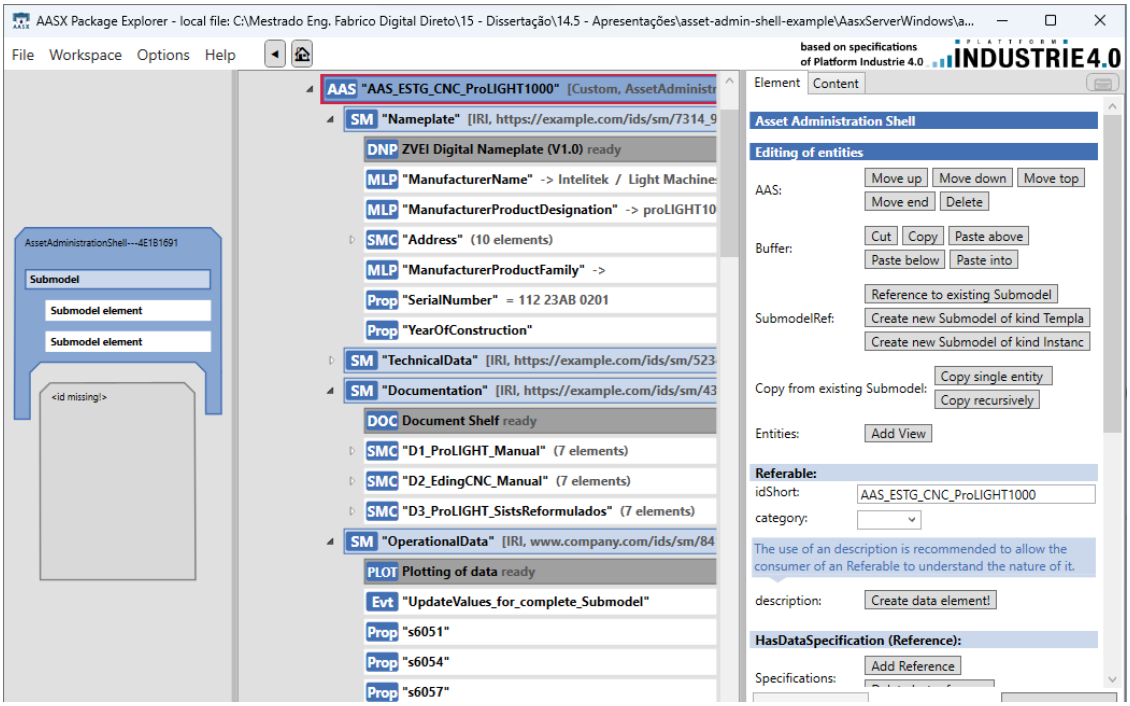


Figura 41 - Ficheiro .aasx criado e seus instanciados submodelos preenchidos.

O *software* AASX Server foi utilizado para iniciar um servidor REST que integra um cliente OPC UA e que contém as informações de um repositório de ficheiros .aasx, onde se incluiu esta AAS. O AASX Package Explorer e o AASX Server foram conectados, via TCP/IP, ao servidor OPC UA, o que permitiu que esta AAS lesse periodicamente os valores escritos nos *nós* do servidor OPC UA, que são continuamente atualizados. A Figura 42 mostra as aplicações que compõem o sistema projetado e mostra que os dados revelados na interface gráfica do EdingCNC são escritos no servidor OPC UA e lidos pela AAS.

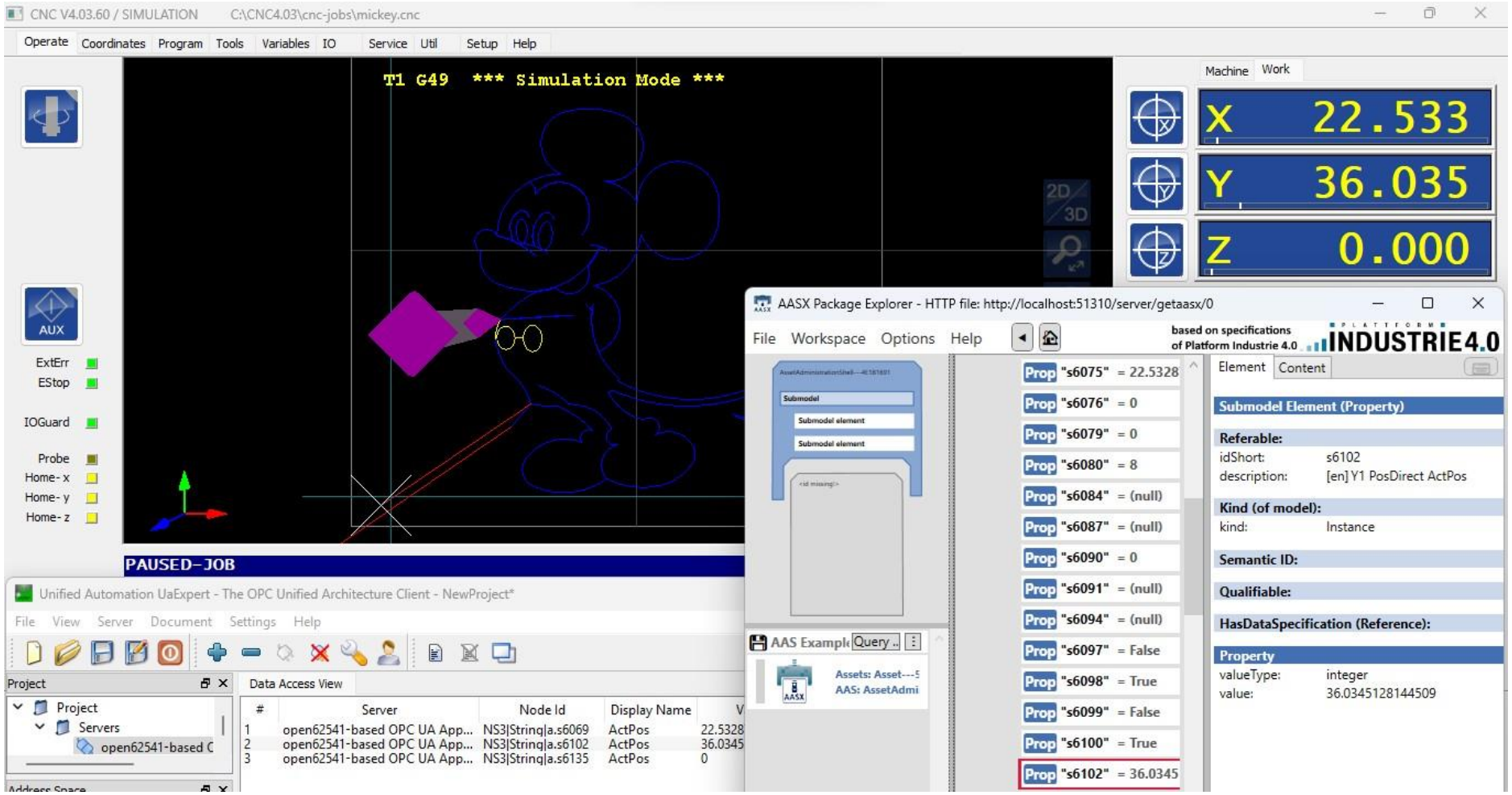


Figura 42 - UAExpert e AAS conectadas ao servidor OPC UA que está conectado ao EdingCNC.

5.5. Documentação da implementação do projeto e dos testes realizados

Foram incluídos alguns anexos neste relatório de projeto, para facilitar a sua compreensão.

O Anexo A descreve a implementação deste sistema e explica o seu funcionamento. Os capítulos 5 e 7 desse anexo, apresentam os testes que foram realizados para averiguar o funcionamento do servidor OPC UA e da AAS, respetivamente. Para testar o servidor OPC UA desenvolvido, utilizou-se o *software* UAExpert que implementa um cliente OPC UA genérico com uma interface gráfica e através do qual é possível ler ou escrever os valores dos *objetos* do servidor OPC UA e validar o funcionamento dos *métodos* adicionais que foram criados, recorrendo às funções da API do EdingCNC, tais como Carregar Programa, Correr Programa, Pausar Programa, Recomeçar Programa e RESET.

O Anexo B esquematiza os sistemas de controlo da ProLIGHT1000 presente na ESTG.

Finalmente, o Anexo C consiste numa tabela que sumariza os valores que foram atribuídos aos *nós* que compõem o *address space* que foi modelado para o servidor OPC UA deste projeto.

6. Resultados

Este projeto demonstrou uma forma de desenvolver e implementar um sistema de controlo e aquisição de dados para um equipamento industrial, que atende às exigências do modelo RAMI4.0, nomeadamente a utilização das tecnologias OPC UA e AAS, e que aplica o conceito *Plug&Produce*. Assim, conseguiu-se apresentar uma resposta válida para a questão de investigação e atingir os objetivos propostos.

Passou a ser possível adquirir dados da ProLIGHT1000 de forma interoperável, porque os valores devolvidos pelas funções da API do EdingCNC passaram a ser escritos nos *objetos* de um servidor OPC UA. Para além disso, passou a ser possível controlar este equipamento de forma interoperável, porque as funções da API do EdingCNC foram associadas a *objetos* e *métodos* de um servidor OPC UA, aos quais e com os quais um cliente OPC UA pode aceder e interagir.

Este sistema foi testado e verificou-se o seu correto funcionamento, no caso do EdingCNC estar a operar em modo de simulação e no caso de estar a operar em modo de trabalho real.

Por fim, demonstrou-se a integração da fresadora CNC ProLIGHT1000 com a sua AAS, passando, assim, este ativo a ser considerado um componenteI4.0. Esta AAS é classificada como reativa porque interage com a API do EdingCNC. Esta AAS, para além de conter a descrição deste ativo, é, de forma automática e periódica, atualizada com os valores escritos nos *nós* do servidor OPC UA, que correspondem aos parâmetros do *software* EdingCNC. Assim, esta AAS constitui um gémeo digital deste equipamento, em concordância com o modelo RAMI4.0.

7. Conclusões e Trabalhos Futuros

7.1. Conclusões

A 4ª Revolução Industrial já impactou a sociedade e espera-se que, ao longo dos próximos anos, este impacto venha a ter maior magnitude. As inovadoras aplicações de tecnologias de Fabricação Digital Direta e de novos sistemas e equipamentos automáticos industriais, vêm permitir implementar sistemas de produção caraterizados por uma maior interoperabilidade, rastreabilidade e eficiência, o que permite repensar sua a forma de gestão e que terá impactos na economia, sustentabilidade e sociedade. Esta evolução tem por base a aplicação de protocolos de comunicação padronizados, modelos de informação padronizados e gémeos digitais padronizados.

Na Europa, o modelo padrão de arquitetura para sistemas da indústria 4.0 é o RAMI4.0, que indica a utilização das tecnologias OPC UA e AAS. O protocolo de comunicação OPC UA permite transmitir e expor estruturas complexas de informação derivadas de modelos de informação padronizados. A tecnologia AAS serve para descrever um ativo e encapsular as suas funcionalidades e criar, de forma padronizada, o seu gémeo digital.

Concluiu-se o que o protocolo OPC UA permite estabelecer a camada de comunicação entre distintos sistemas, de forma fiável e segura. Além disso, concluiu-se que a tecnologia AAS apresenta-se como uma solução completa, fiável e segura, que permite descrever diversos aspetos dos ativos e que permite recorrer a diversos mecanismos de comunicação, incluindo o OPC UA.

Os programas que servem para editar e visualizar ficheiros .aasx e diversos dos submodelos padronizados de AAS ainda estavam em desenvolvimento e, durante a elaboração deste trabalho, foram publicadas novas versões do AASX Package Explorer e do AASX Server.

7.2. Trabalhos Futuros

Com base neste projeto, como trabalhos futuros, sugere-se:

- Implementar nesta AAS o submodelo *Capability Description* que permitirá integrar os termos que descrevem as *capacidades* (*capabilities*) abstratas desse equipamento e as *habilidades* (*skills*) concretas que esse equipamento possui para efetuar operações de trabalho. Assim, esta AAS também permitirá controlar o respetivo equipamento CNC.

- Implementar neste servidor OPC UA a funcionalidade de registo histórico;
- Preencher os *objetos* deste *address space* associados ao CNCAlarm ou ao CNCMessage;
- Conectar o servidor OPC UA ao *software* RobotStudio para simular os movimentos deste equipamento CNC, utilizando o seu modelo 3D.

Referências Bibliográficas

[1] K. Schwab, “The Fourth Industrial Revolution”. World Economic Forum, 2016. Geneva, Switzerland.

[2] C. J. Bartodziej, “The concept Industry 4.0”. Springer Gabler Wiesbaden, 2017. [Online]. Disponível em: https://doi.org/10.1007/978-3-658-16502-4

[3] J. R. C. Pinto, “Tecnologias de Automação na Indústria4.0”. 1ª edição. Lidel - Edições Técnicas, 2021. Lisboa.

[4] D. Mourtzis (Ed.), “Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology”. Elsevier, 2022. ISBN 978-0-12-823657-4. [Online]. Disponível em: https://doi.org/10.1016/C2019-0-05325-3

[5] Agência Nacional de Inovação, “Mobilizadores”, 2023. Acedido em 6 de Março de 2023, em: https://www.ani.pt/pt/financiamento/incentivos-financeiros-pt-2020/mobilizadores/

[6] EDILÁSIO, “Projeto S4Plast”, 2023. Acedido em 6 de Março de 2023, em: https://www.edilasio.pt/s4plast.html

[7] OLI - Sistemas Sanitários, S.A., “S4Plast – Sustainable Plastics Advanced solutions”, 2023. Acedido em 7 de Março de 2023, em: https://www.oli-world.com/pt/empresa/projetos/s4plast-e2-80-93-sustainable-plastics-advanced-solutions/

[8] Projeto S4Plast, “Projeto”, 2023. Acedido em 4 de Março de 2023, em: https://s4plast.toolingportugal.com/projeto.html

[9] International Organization for Standardization, “ISO 6983-1:2009 - Automation systems and integration -- Numerical control of machines,” 2024. Acedido em 16 de Março de 2024, em: https://www.iso.org/standard/34608.html

[10] OPC Foundation, “OPC UA Companion-Specification: OPC UA for Computerized Numerical Control (CNC) Systems (OPC 40502)”, 2017. Acedido em 1 de Janeiro de 2023, em: https://opcfoundation.org/developer-tools/documents/view/216

[11] P. Jacobs, “CNC Machine Buyer’s Guide: Types, Uses, Price, & Definitions”, 2022. Acedido em 1 de Janeiro de 2023, em: https://www.cncmasters.com/cnc-machine-buyers-guide/

[12] A. Martins, B. Silva, H. Costelha, C. Neves, J. Lyons, and J. Cosgrove, "An approach to integrating manufacturing data from legacy Injection Moulding Machines using OPC UA". Trabalho apresentado na *IMC37 - 37th International Manufacturing Conference,* Athlone, 2021

[13] D. Jaspert, M. Ebel, A. Eckhardt, and J. Poeppelbuss, "Smart retrofitting in manufacturing : A systematic review".
Em: *Journal of Cleaner Production*, vol. 312, 2021.
[Online]. Disponível em: https://doi.org/10.1016/j.jclepro.2021.127555

[14] M. Soori, B. Arezoo, and R. Dastres, "Internet of things for smart factories in industry 4.0, a review". Em: *Internet of Things and Cyber-Physical Systems*, vol. 3, 2023.
[Online]. Disponível em: doi: https://doi.org/10.1016/j.iotcps.2023.04.006

[15] T. O. Silva, "Uma Arquitetura Orientada a Serviços visando o suporte à automação industrial integrada e distribuída".
Dissertação de Mestrado (Programa de Pós-Graduação em Mecatrónica).
Universidade Federal da Bahia, 2018.

[16] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, "Agile assembly system by "Plug and Produce"". Em: *CIRP Annals*, vol. 49, no. 1, 2000.
[Online]. Disponível em: doi: https://doi.org/10.1016/S0007-8506(07)62883-2

[17] S. Weyer, M. Schmitt, M. Ohmer, and D. Gorecky, "Towards industry 4.0 - Standardization as the crucial challenge for highly modular, multi-vendor production systems". Em: *IFAC-PapersOnLine*, vol. 28, no. 3, 2015.
[Online]. Disponível em: https://doi.org/10.1016/j.ifacol.2015.06.143

[18] S. K. Panda, T. Schroder, L. Wisniewski, and C. Diedrich, "PlugProduce Integration of Components into OPC UA based data-space".
Em: *2018 IEEE 23rd International Conferennce on Emerging Technologies and Factory Automation (ETFA)*, 2018. Turin, Itália.
[Online]. Disponível em: https://doi.org/10.1109/ETFA.2018.8502663

[19] A. Martins, J. Lucas, H. Costelha, and C. Neves, "Developing an OPC UA Server for CNC Machines". Em: *Procedia Computer Science*, vol. 180, 2021.
[Online]. Disponível em: doi: 10.1016/j.procs.2021.01.276

[20] X. Ye, J. Jiang, C. Lee, N. Kim, M. Yu, and S. H. Hong, "Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach".
Em: *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, 2020.
[Online]. Disponível em: https://doi.org/10.1109/MIE.2020.3010492

[21] J. L. F. Lucas, "Desenvolvimento de servidor OPC UA para sistema CNC".
Projeto de mestrado em Engenharia Mecânica - Produção Industrial.
Instituto Politécnico de Leiria, 2019.

[22] A. F. Martins, “Industrial device integration and virtualization for smart factories”, Projeto de mestrado em Engenharia Eletrotécnica.
Instituto Politécnico de Leiria, 2020.

[23] Encyclopædia Britannica, 2023.
“History of Technology - Development of Industries”.
Acedido em 4 de janeiro de 2023, em: https://www.britannica.com/technology/history-of-technology/Development-of-industries

[24] J. Dornelles, 2021.
“Internet das Coisas na Manufatura Industrial: uma ferramenta da Indústria 4.0 para interação dos processos”. Acedido em 18 de Fevereiro de 2023, em: https://redeindustria40.com.br/internet-das-coisas-na-manufatura-industrial-uma-ferramenta-da-industria-4-0-para-interacao-dos-processos/

[25] E. Westkämper, D. Spath, C. Constantinescu, and J. Lentes, “Digitale Produktion”. Springer Berlin, 2013. Heidelberg. ISBN 978-3-642-20258-2.
[Online]. Disponível em: https://doi.org/10.1007/978-3-642-20259-9

[26] D. Klitou, J. Conrads, M. Rasmussen, L. Probst, B. Pedersen, and PwC, “Digital Transformation Monitor - Germany: Industrie4.0”. European Comission, 2017.
[Online]. Disponível em:
https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie 4.0_DE.pdf.

[27] C. Steinhoff, “Aktueller Begriff: Industrie 4.0”. Deutscher Bundestag, 2016.

[28] Perfil da Alemanha (2023), “Perfil da Alemanha”.
Acedido em 18 de fevereiro de 2023, em:
https://www.tatsachen-ueber-deutschland.de/pt-br/alemanha-livro-edicao-2023.

[29] R. Hall, S. Schumacher, and A. Bildstein, “Systematic Analysis of Industrie 4.0 Design Principles”. Em: *Procedia CIRP*, vol. 107, 2022.
[Online]. Disponível em: https://doi.org/10.1016/j.procir.2022.05.005

[30] C. Neves, H. Costelha, and L. C. Bento, “Indústria4.0 - A Quarta Revolução Industrial”. Em: *O Molde*, 2018. Acedido em 18 de Fevereiro de 2023, em: https://issuu.com/cefamol/docs/molde_118.

[31] Belden Inc., “TSN Technology”, 2023. Acedido em 19 de Fevereiro de 2023, em: https://www.belden.com/Solutions/TSN-Technology

[32] TWI Ltd., “What is Digital Manufacturing ?”, 2023.
Acedido em 4 de Março de 2023, em: https://www.twi-global.com/technical-knowledge/faqs/what-is-digital-manufacturing

[33] D. Chen, S. Heyer, S. Ibbotson, K. Salonitis, J. G. Steingrímsson, S. Thiede, "Direct digital manufacturing: Definition, evolution, and sustainability implications". Em: *Journal of Cleaner Production*, vol. 107, 2015. [Online]. Disponível em: doi: 10.1016/j.jclepro.2015.05.009

[34] I. Gibson, D. Rosen, and B. Stucker, "Additive Manufacturing Technologies". 2ª edição. Springer New York, 2015.

[35] International Organization for Standardization, "ISO/ASTM 52900:2015.", 2015. Acedido em 30 de Novembro de 2020, em: https://www.iso.org/obp/ui/#iso:std:iso-astm:52900:ed-1:v1:en.

[36] S.-Y. Yu, A. V. Malawade, S. R. Chhetri, and M. A. Al Faruque, "Sabotage Attack Detection for Additive Manufacturing Systems". Em: *IEEE Access*, vol. 8, 2020. [Online]. Disponível em: https://doi.org/10.1109/ACCESS.2020.2971947

[37] 3Dnatives, "Additive Manufacturing: Applications by sector," 2024. Acedido em 16 de Março de 2024, em: https://www.3dnatives.com/en/applications-by-sector/

[38] M. Grieves, "Digital Twin : Manufacturing Excellence through Virtual Factory Replication", 2015. [Online]. Disponível em: https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication

[39] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". Em: *Transdisciplinary perspectives on complex systems: New findings and approaches*, 2017. Springer, Cham. [Online]. Disponível em: https://doi.org/10.1007/978-3-319-38756-7_4

[40] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification". Em: *IFAC-PapersOnLine*, vol. 51, no. 11, 2018. [Online]. Disponível em: https://doi.org/10.1016/j.ifacol.2018.08.474

[41] E. H. Glaessgen and D. S. Stargel, "The digital twin paradigm for future NASA and U.S. Air force vehicles". Em: *53rd Structures, Structural Dynamics and Materials Conference*, 2012. Honolulu, Hawaii. [Online]. Disponível em: https://doi.org/10.2514/6.2012-1818

[42] F. Tao and M. Zhang, "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing". Em: *IEEE Access*, vol. 5, 2017. [Online]. Disponível em: doi: 10.1109/ACCESS.2017.2756069

[43] R. Vrabič, J. A. Erkoyuncu, P. Butala, and R. Roy, "Digital twins: Understanding the added value of integrated models for through-life engineering services".

Em: *Procedia Manufacturing*, vol. 16, 2018.
[Online]. Disponível em: https://doi.org/10.1016/j.promfg.2018.10.167

[44] F. Tao, M. Zhang, Y. Liu, and A. Y. C. Nee , "Digital twin driven prognostics and health management for complex equipment". Em: *CIRP Annals*, vol. 67, no. 1, 2018.
[Online]. Disponível em: https://doi.org/10.1016/j.cirp.2018.04.055

[45] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research". Em: *IEEE Access*, vol. 8, 2020.
[Online]. Disponível em: https://doi.org/10.1109/ACCESS.2020.2998358

[46] W. Hu, T. Zhang, X. Deng, Z. Liu, and J. Tan, "Digital twin: a state-of-the-art review of its enabling technologies, applications and challenges". Em: *Journal of Intelligent Manufacturing and Special Equipment*, vol. 2, no. 1, 2021.
[Online]. Disponível em: https://doi.org/10.1108/jimse-12-2020-010

[47] Q. Qi et al., "Enabling technologies and tools for digital twin".
Em: *Journal of Manufacturing Systems*, vol. 58, 2021.
[Online]. Disponível em: https://doi.org/10.1016/j.jmsy.2019.10.001

[48] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, "Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry4.0: Correlation and Comparison".
Em: *Engineering*, vol. 5, no. 4, 2019.
[Online]. Disponível em: https://doi.org/10.1016/j.eng.2019.01.014

[49] M. G. Juarez, V. J. Botti, and A. S. Giret, "Digital Twins : Review and Challenges".
Em: *ASME Journal of Computing Information Science in Engineering*.
June 2021; 21(3), 2012. [Online]. Disponível em: https://doi.org/10.1115/1.4050244

[50] I. Onaji, D. Tiwari, P. Soulatiantork, B. Song, and A. Tiwari,
"Digital twin in manufacturing: conceptual framework and case studies". Em:
International Journal of Computer Integrated Manufacturing, vol. 35, no. 8, 2022.
[Online]. Disponível em: https://doi.org/10.1080/0951192X.2022.2027014

[51] M. Pantelidakis, K. Mykoniatis, J. Liu, and G. Harris, "A digital twin ecosystem for additive manufacturing using a real-time development platform". Em: *The International Journal of Advanced Manufacturing Technology*, vol. 120, no. 9–10, 2022. [Online]. Disponível em: https://doi.org/10.1007/s00170-022-09164-6

[52] E. A. Lee and S. A. Seshia, "Introduction to Embedded Systems - A Cyber-Physical Systems Approach". 2ª edição. MIT Press, 2017. ISBN: 978-0-262-53381-2

[53] Y. Tian and D. C. Levy (Ed.), "Handbook of Real-Time Computing".
Springer Nature Singapore, 2022. ISBN 978-981-287-250-0
[Online]. Disponível em: https://doi.org/10.1007/978-981-287-251-7

[54] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems”.
Em: *Manufacturing Letters*, vol. 3, 2015.
[Online]. Disponível em: https://doi.org/10.1016/j.mfglet.2014.12.001

[55] D. Mourtzis, N. Milas, and N. Athinaios, “Towards Machine Shop 4.0: A General Machine Model for CNC machine-tools through OPC UA”.
Em : *Procedia CIRP*, vol. 78, 2018.
[Online]. Disponível em: https://doi.org/10.1016/j.procir.2018.09.045

[56] C. Liu, H. Vengayil, Y. Lu, and X. Xu, “A Cyber-Physical Machine Tools Platform using OPC UA and MTConnect”. Em: *Journal of Manufacturing Systems*, vol. 51, 2019. [Online]. Disponível em: https://doi.org/10.1016/j.jmsy.2019.04.006

[57] P. Adolphs *et al*., “Status Report - Reference Architecture Model Industrie4.0”.
VDI, ZVEI. 2015.
[Online]. Disponível em: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/GMA_Status_Report__Reference_Archtitecture_Model_Industrie_4.0__RAMI_4.0_/GMA-Status-Report-RAMI-40-July-2015.pdf

[58] VDMA, Fraunhofer Application Center Industrial Automation (IOSB-INA), “Industrie 4.0 Communication Guideline - Based on OPC UA”, 2017.
[Online]. Disponível em: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/vdma-i40-communication-opc.html

[59] T. Bangemenn *et al*., “Status Report - Industrie 4.0 - Technical Assets: Basic terminology concepts, life cycles and administration models”.
VDI, ZVEI. Düsseldorf, 2016.
[Online]. Disponível em: https://www.vdi.de/ueber-uns/presse/publikationen/details/industrie-40-technical-assets-basic-terminology-concepts-life-cycles-and-administration-models-english-version

[60] W. Mahnke, S.-H. Leitner, and M. Damm, “OPC Unified Architecture”.
Springer-Verlag, 2009. Berlin. ISBN 978-3-540-68898-3.
[Online]. Disponível em: https://doi.org/10.1007/978-3-540-68899-0

[61] OPC Foundation (2023), “What is OPC ?”. Acedido em 17 de Abril de 2023, em: https://opcfoundation.org/about/what-is-opc/

[62] OPC Foundation (2023), “Unified Architecture”. Acedido em 18 de Abril de 2023, em: https://opcfoundation.org/about/opc-technologies/opc-ua/

[63] OPC Foundation, “OPC Unified Architecture Part 1: Overview and Concepts (OPC 10000-1)”, 2022. Acedido em 1 de Janeiro de 2023, em: https://opcfoundation.org/developer-tools/documents/view/158

[64] Unified Automation, "C++ Based OPC UA Client/Server/PubSub SDK: OPC UA Overview", 2023. Acedido em 20 de Abril de 2023, em: https://documentation.unified-automation.com/uasdkcpp/1.7.3/html/L2OpcUaFundamentalsOverview.html

[65] OPC Foundation, "OPC Unified Architecture Part 3: Adress Space Model (OPC 10000-3)", 2022. Acedido em 1 de Janeiro de 2023, em: https://opcfoundation.org/developer-tools/documents/view/160

[66] OPC Foundation, "OPC Unified Architecture Part 5: Information Model (OPC 10000-5)", 2022. Acedido em 1 de Janeiro de 2023, em: https://opcfoundation.org/developer-tools/documents/view/162

[67] T. A. Abdel-Aty, E. Negri, and S. Galparoli, "Asset Administration Shell in Manufacturing: Applications and Relationship with Digital Twin". Em: *IFAC-PapersOnLine*, vol. 55, no. 10, 2022. [Online]. Disponível em: https://doi.org/10.1016/j.ifacol.2022.10.090

[68] T. van Erp, F. B. Pedersen, N. P. L. Larsen, and R. B. Lund, "Industrial Digital Twin in Industry 4.0: Enabling Service Exchange Between Assets". Em: *Manufacturing Driving Circular Economy*, 2023. [Online]. Disponível em: https://doi.org/10.1007/978-3-031-28839-5_64

[69] S. Bader, E. Barnstedt, H. Bedenbender, M. Billman, B. Boss, and A. Braunmandl, "SPECIFICATION Details of the Asset Administration Shell: Part 1 - The exchange of information between partners in the value chain of Industrie4.0". Federal Ministry for Economic Affairs and Energy (BMWi), 2020. Berlin. [Online]. Disponível em: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html

[70] PlattformIndustrie4.0 (2022), "Asset Administration Shell Reading Guide".

[71] X. Ye, S. H. Hong, W. S. Song, Y. C. Kim, and X. Zhang, "An Industry 4.0 Asset Administration Shell-Enabled Digital Solution for Robot-Based Manufacturing Systems". Em: *IEEE Access*, vol. 9, 2021 [Online]. Disponível em: https://doi.org/10.1109/ACCESS.2021.3128580

[72] Light Machines Corporation, "ProLIGHT1000 Machining Center User's Guide", 1996. Manchester, New Hampshire, USA.

[73] Bert Eding, "User Manual Eding CNC Software". versão 4.03. 2021.

[74] Arduino, "Arduino MEGA 2560 Rev3", 2023. Acedido em 23 de Dezembro de 2023, em: https://docs.arduino.cc/hardware/mega-2560

[75] S. Potier, "OPC UA - Information Models & Companion Specifications", 2023. Acedido em 15 de Junho de 2023, em: https://www.slideshare.net/stephane_potier/

opc-ua-information-models-companion-specifications

[76] M. Blume *et al.*, “An OPC UA based approach for dynamic-configuration of security credentials and integrating a vendor independent digital product memory”. Em: *Kommunikation in der Automation (KommA)*, 2014.

[77] L. Sakurada and P. Leitao, “Multi-Agent Systems to Implement Industry4.0 components”. Em: *IEEE Conference on Industrial Cyberphysical Systems* (*ICPS*), 2020. [Online]. Disponível em: https://doi.org/10.1109/ICPS48405.2020.9274745

[78] PlattformIndustrie4.0, “Relationships between I4.0 Components - Composite Components and Smart Production”. Federal Ministry for Economic Affairs and Energy (BMWi), 2017. Berlin. [Online]. Disponível em: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/hm-2018-relationship.html

[79] PlattformIndustrie4.0, “Structure of the Asset Administration Shell”. Federal Ministry for Economic Affairs and Energy (BMWi), 2016. Berlin. [Online]. Disponível em: https://industrialdigitaltwin.org/en/wp-content/uploads/2021/09/01_structure_of_the_administration_shell_en_2016.pd

[80] Industrial Digital Twin Association, “IDTA - working together to promote the Digital Twin”, 2023. Acedido em 24 de Dezembro de 2023, em: https://industrialdigitaltwin.org/en/

[81] PlattformIndustrie4.0, “Soft shell, hard centre”, 2020. Acedido em 24 de Dezembro de 2023, em: https://www.plattform-i40.de/IP/Redaktion/EN/PressReleases/2020/2020-09-24-soft-shell-hard-centre.html

[82] ZVEI, “User organization Industrial Digital Twin Association founded”, 2020. Acedido em 24 de Dezembro de 2023, em: https://www.zvei.org/en/press-media/pressarea/user-organization-industrial-digital-twin-association-founded

[83] admin-shell-io, “aasx-package-explorer: C# based viewer / editor for the Asset Administration Shell”, 2023. Acedido em 15 de Dezembro de 2023, em: https://github.com/admin-shell-io/aasx-package-explorer

[84] Kitware (2023), “CMake - Upgrade Your Software Build System”. Acedido em 24 de Dezembro de 2023, em: https://cmake.org/

[85] Microsoft (2023), “Visual Studio: IDE and Code Editor for Software Developers”. Acedido em 24 de Dezembro de 2023, em: https://visualstudio.microsoft.com/

Lista de Anexos

Anexo A - Manual de implementação do projeto.

GUIÃO PARA A IMPLEMENTAÇÃO DO SERVIDOR OPC UA E DA AAS

Pedro Rafael Marques Sousa

Leiria, abril de 2024

Índice

Lista de Figuras

1. Introdução

Este anexo descreve os procedimentos necessários para implementar o sistema apresentado neste relatório de projeto de mestrado (Figura A1).

Respeitando as especificações do modelo RAMI4.0, este sistema foi projetado para interagir com o *software* e *hardware* de controlo CNC do fabricante EdingCNC e permitir controlar e adquirir dados de equipamentos industriais, de acordo com a abordagem *Plug&Produce* adotada (Figura A2). Este sistema pode ser personalizado, visto que se pode alterar o seu *address space* e adicionar funcionalidades ou interfaces para outros controladores CNC.

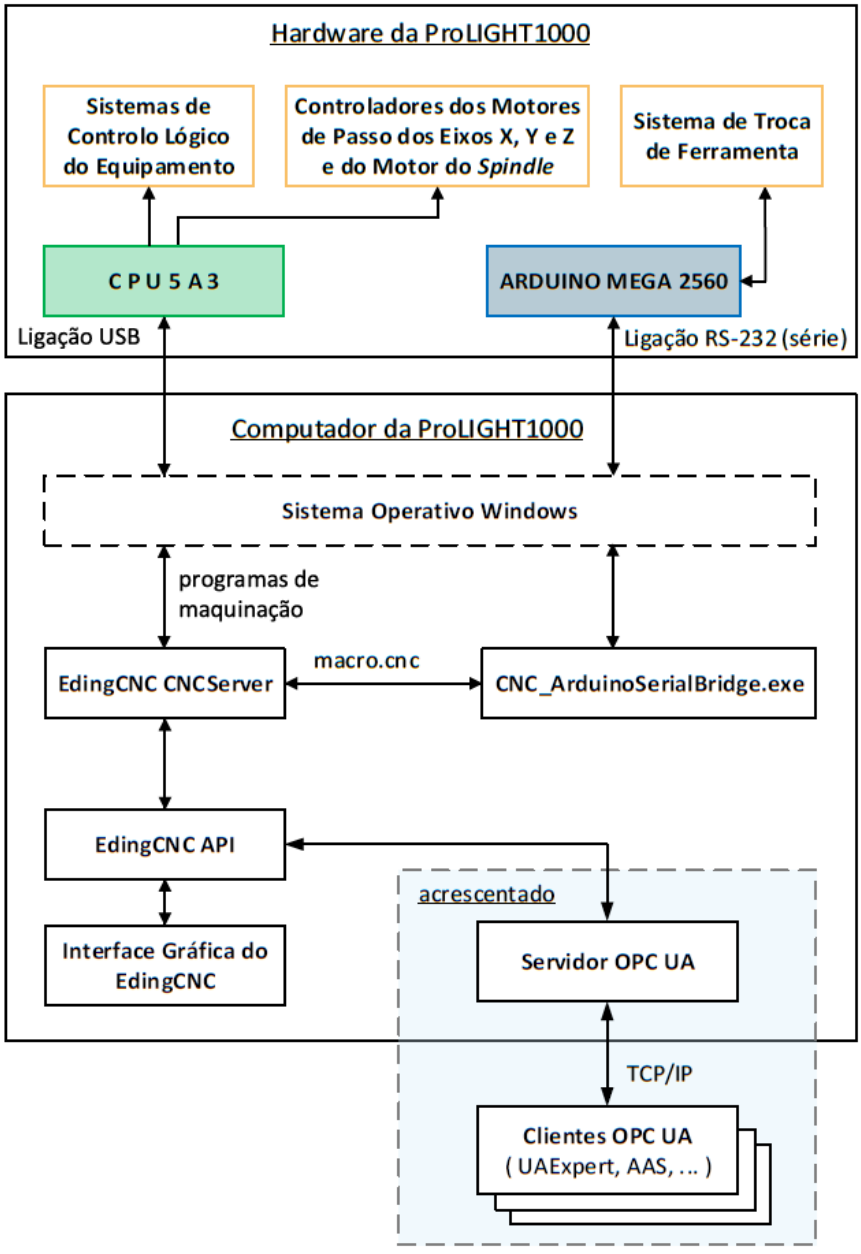


Figura A1 - Sistema de controlo e aquisição de dados, cuja implementação é demonstrada neste manual.

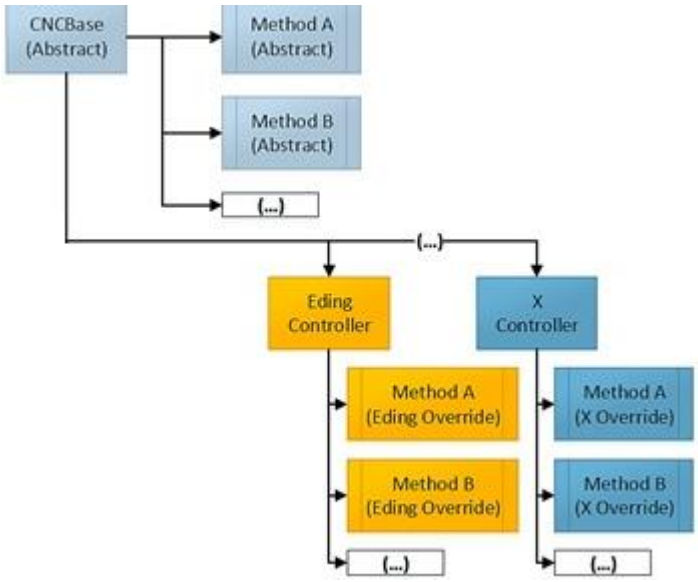


Figura A2 - Representação da *classe* CNCBase e da *classe* CNCEding (adaptado de [22]).

Para seguir este manual, será necessário instalar os seguintes programas:

- **EdingCNC** (foi utilizada a versão 4.03.60), https://edingcnc.cncsoftware.download/ ;
- **CMake** (foi utilizada a versão 3.26.0-rc3), https://cmake.org/download/ ;
- **Microsoft Visual Studio** (foi utilizada a versão *Enterprise* 2019) https://visualstudio.microsoft.com/pt-br/downloads/ ;
- **UAExpert** (foi utilizada a versão 1.6.3 448), https://www.unified-automation.com/products/development-tools/uaexpert.html ;
- **UAModeler** (foi utilizada a versão 1.6.8 515), https://www.unified-automation.com/products/development-tools/uamodeler.html ;

  Nota: Tanto o UAExpert como o UAModeler requerem um registo de utilizador no *site* da UnifiedAutomation.
- **AASX Package Explorer** (foi utilizada a versão v2022-08-06) https://github.com/admin-shell-io/aasx-package-explorer/releases
- **AASX Server** (foi utilizada a versão v2022-07-25.alpha), https://github.com/admin-shell-io/

2. Procedimentos Iniciais

2.1. Descarregar e compilar a stack OPC UA

O primeiro passo consiste em aceder ao *website* https://github.com/open62541/open62541 (Figura A3) e descarregar o ficheiro .zip (Figura A4) que contém o código desta *stack*. Para isso, nesse *site*, com o botão esquerdo do rato, clique em **Code > Download Zip**.

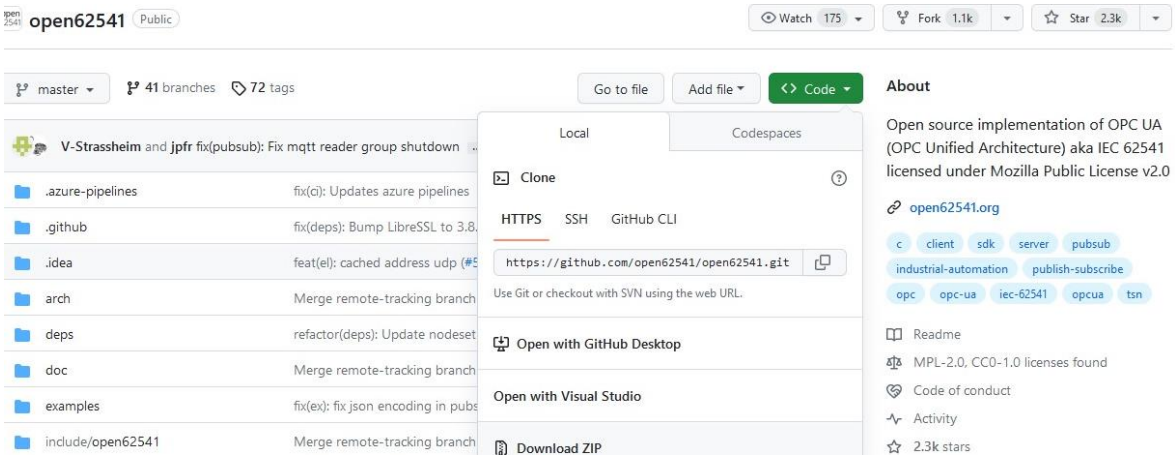


Figura A3 - Página *web* do repositório *github* da open62541.org que contém a respetiva *stack* OPC UA.

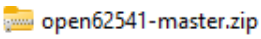


Figura A4 - Ficheiro .zip descarregado do repositório *github* da open62541.org.

Clique, com o botão direito do rato, nesse ficheiro e, com o botão esquerdo, clique em **Propriedades**, selecione **Desbloquear** e clique em **Aplicar** e em **OK**. Clique, com o botão direito, nesse ficheiro e, com o botão esquerdo, clique em **Extrair Todos > Extrair**. Desta forma, obtém-se o conteúdo do ficheiro *open62541-master.zip*, que se vê na Figura A5.

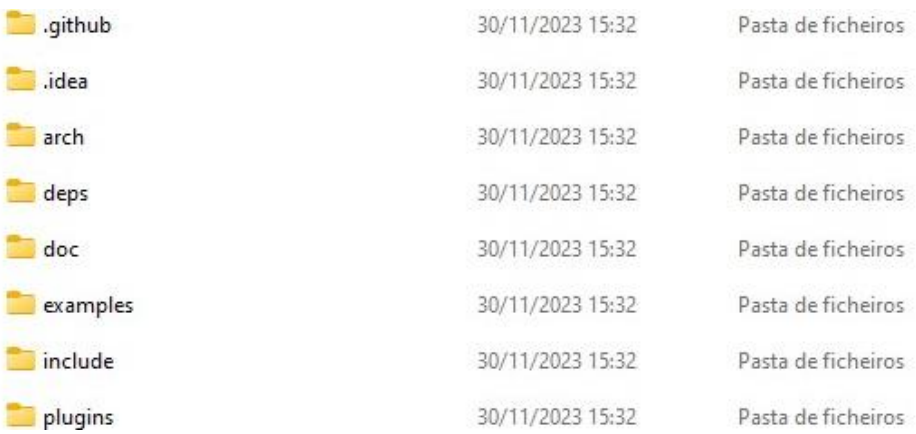


Figura A5 - Excerto do conteúdo extraído do ficheiro *open62541-master.zip*.

De seguida, deve-se abrir o CMake, obtendo-se uma janela como mostra a Figura A6.

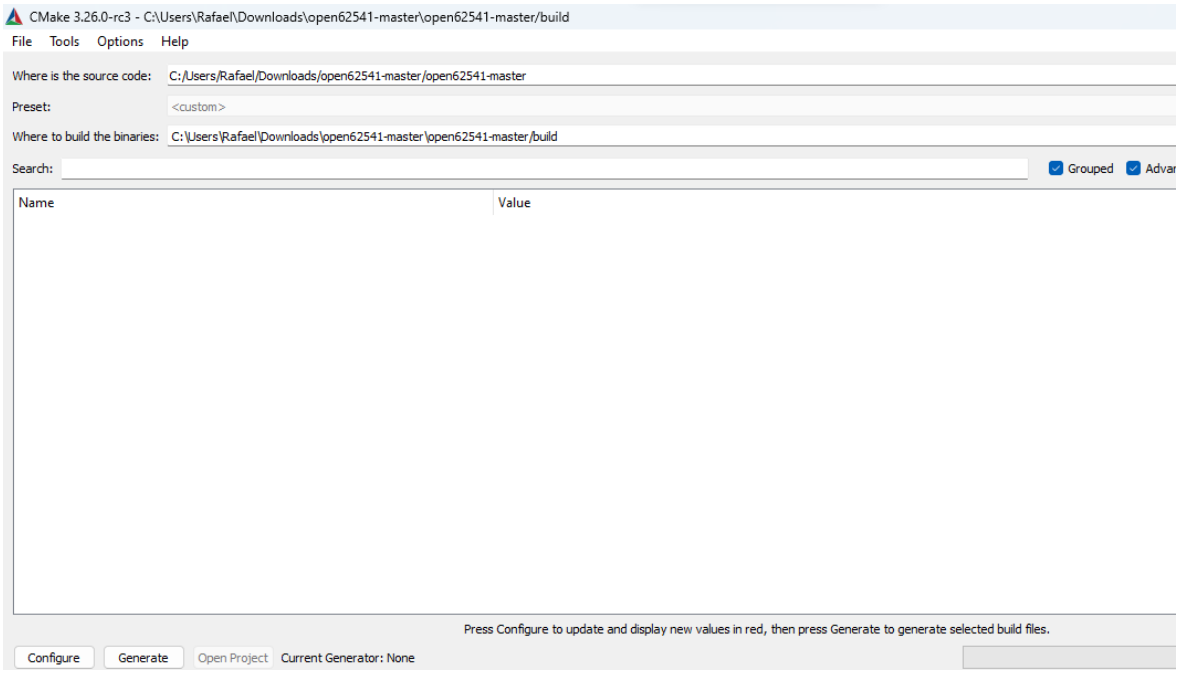


Figura A6 - Interface gráfica do CMake.

Clique em **File > Delete Cache**. No campo **"Where is the source code"** deve-se introduzir o caminho da pasta extraída do ficheiro *open62541-master.zip*. No campo **"Where to build the binaries"** deve-se introduzir esse mesmo caminho, ao qual se deve acrescentar no final "/build". Clique em **Configure** e em **Yes**, o que faz abrir uma janela (Figura A7). As opções a selecionar, neste caso, são aquelas indicadas na Figura A7. Clique em **Finish**.

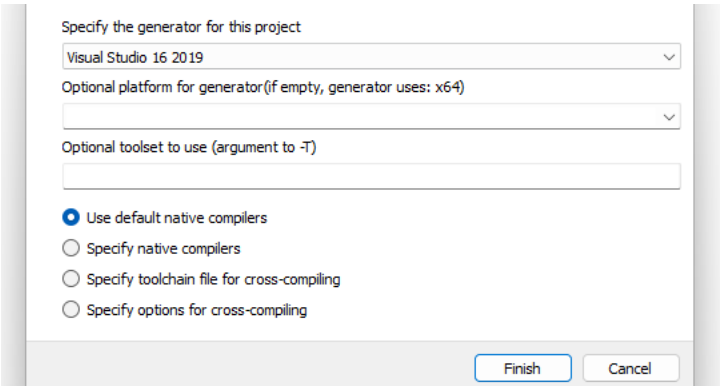


Figura A7 - Janela aberta no CMake, depois de se clicar em *Configure*.

Como mostra a Figura A8, o CMake passa a disponibilizar diversas opções, mas apenas se deve alterar a opção **"UA_ENABLE_AMALGAMATION"** para ativada.

Figura A8 - Interface gráfica do CMake, depois de se clicar em *Finish*.

Clique em **Generate** e feche o CMake. Na diretoria que se vê na Figura A5 passa a existir uma nova pasta designada por *build* que contém novos ficheiros, como mostra a Figura A9.

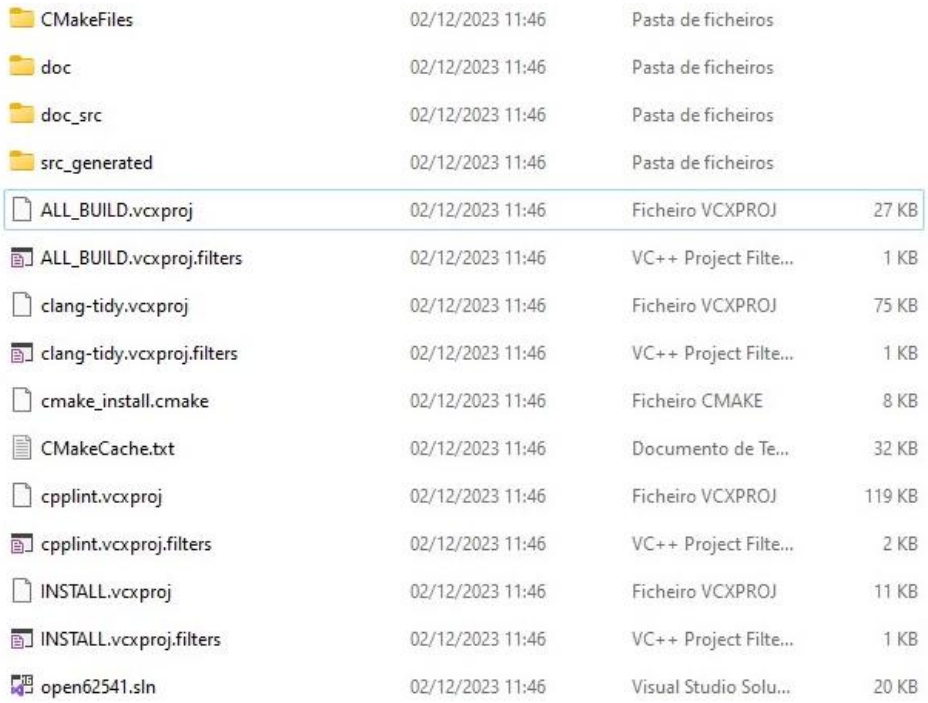


Figura A9 - Excerto do conteúdo da pasta *build*.

Através do Visual Studio, abra o ficheiro *open62541.sln* que se localiza na pasta *build*. No **Gerenciador de Soluções** do Visual Studio, clique, com o botão direito, sobre **Solução 'open62541'** e selecione **Compilar Solução** (Figura A10). Feche o Visual Studio.

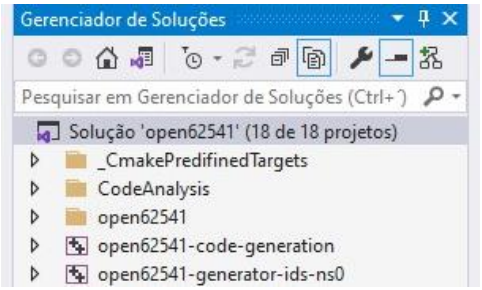


Figura A10 - Excerto da interface gráfica do Visual Studio, depois de aberto o ficheiro open62541.sln.

Na pasta *build* surgem dois novos ficheiros (Figura A11), resultantes desta compilação.

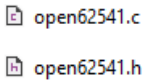


Figura A11 - Ficheiros *open62541.c* e *open62541.h*, localizados na pasta *build*.

Mais adiante, estes dois ficheiros serão utilizados para desenvolver o servidor OPC UA.

3. Address Space

O próximo passo consiste em modelar o *address space* do nosso servidor OPC UA.

Para realizar esse trabalho, é necessário, para além do programa UAModeler, descarregar:

- o ficheiro .pdf que descreve a especificação da CS "OPC UA for CNC Systems", que se encontra disponível em https://opcfoundation.org/developer-tools/documents/view/216. **Nota**: Para isso, é necessário fazer um registo de utilizador no *site* da OPC Foundation ;
- o ficheiro .xml que descreve essa *companion specification* e que se encontra disponível em https://files.opcfoundation.org/schemas/CNC/1.0/Opc.Ua.CNC.NodeSet.xml. Com o botão direito do rato, clique nessa página, selecione **guardar página como**, em **Guardar página com o tipo:** selecione *.xml e, por fim, clique em **Guardar**;

3.1. Criar um projeto no UAModeler

Comece por abrir o UAModeler e selecione **File > New Project**. Abre-se uma janela, como se vê na Figura A12.

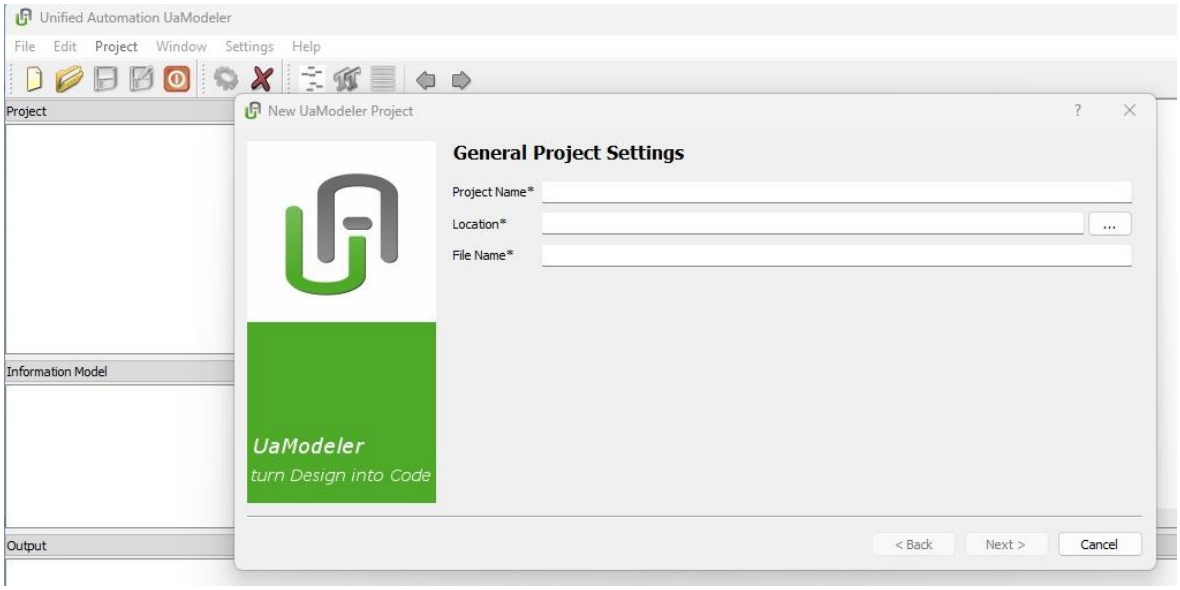


Figura A12 - Iniciar um novo projeto no UAModeler (parte I).

Atribua um nome a este novo projeto do UAModeler (**Project Name**), uma diretoria (**Location**) onde o projeto será guardado e um nome (**File Name**) para o ficheiro a exportar. Ou seja, é feita a distinção entre o projeto, que guarda as configurações aplicadas para modelar o *address space*, e o próprio ficheiro exportado que é o *address space* modelado. Clique em **Next**.

Como mostra a Figura A13, deve-se selecionar a opção **Modelling**, indicar a diretoria onde será guardado o *address space* modelado e clicar em **Next**.

Figura A13 - Iniciar um novo projeto no UAModeler (parte II).

O UAModeler já inclui o meta-modelo OPC UA. Para importar outros modelos de informação, neste caso a CS "OPC UA for CNC Systems", deve-se clicar em **Find another model**, selecionar o ficheiro, que neste caso é o *Opc.Ua.CNC.NodeSet.xml*, clicar em **Abrir** e ativá-lo, deixando um visto, como mostra a Figura A14. Clique em **Next**.

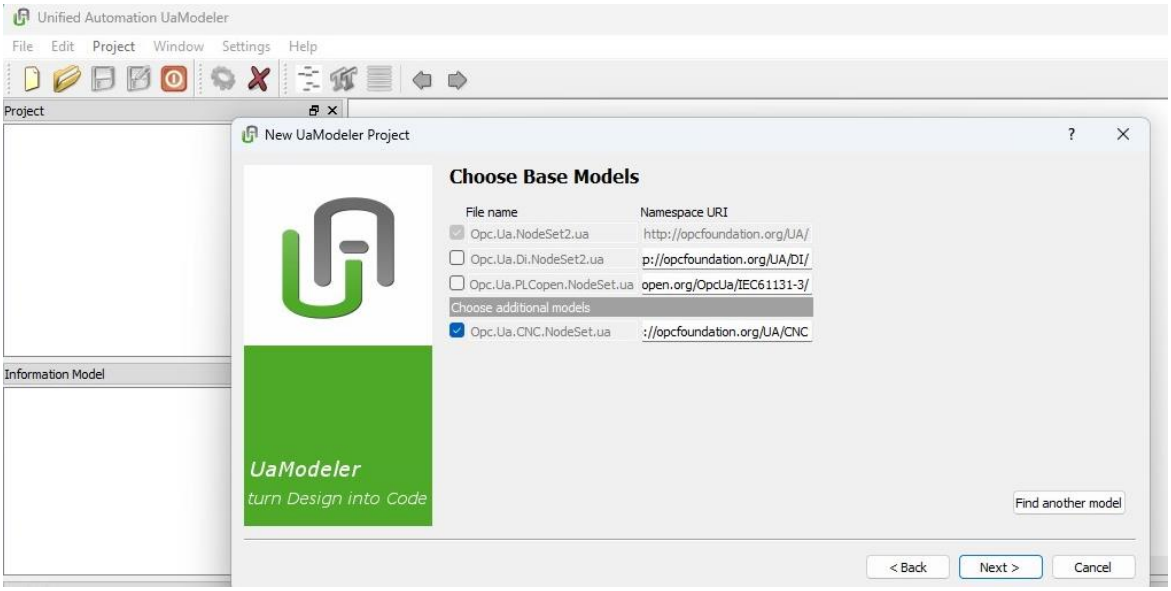


Figura A14 - Iniciar um novo projeto no UAModeler (parte III).

De seguida, como se observa na Figura A15, deve-se atribuir um nome ao *address space* e ao URI do respetivo *namespace*. Clique em **Next**.

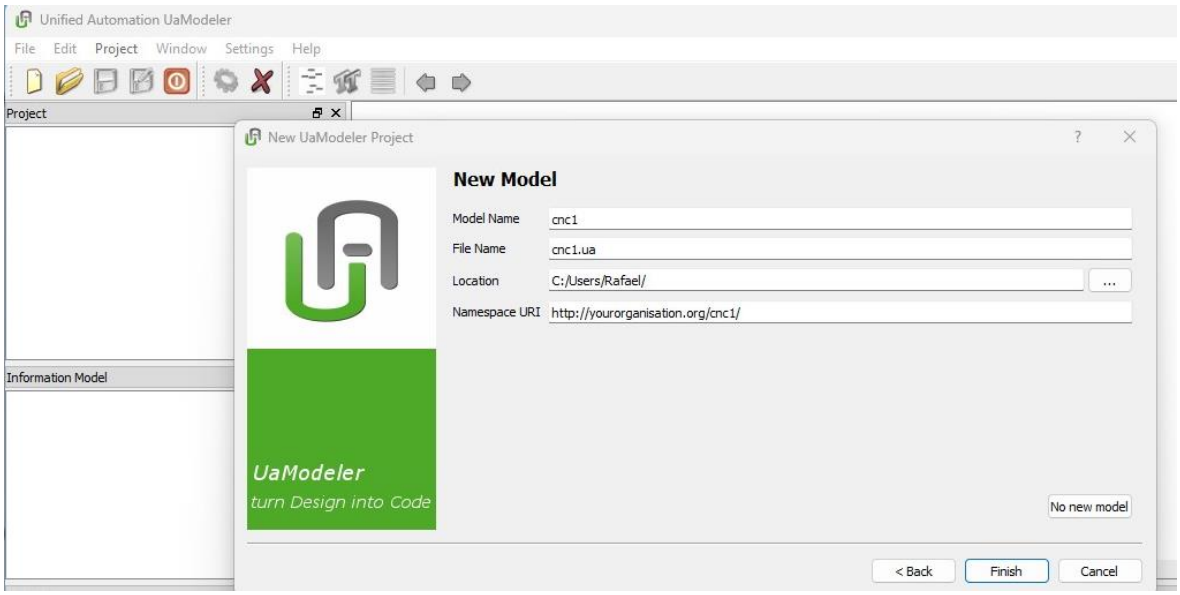


Figura A15 - Iniciar um novo projeto no UAModeler (parte IV).

O UAModeler passa a apresentar-se como a Figura A16, onde se distinguem 3 seções:

A - seção que identifica os modelos de informação OPC UA incluídos no projeto;

B - seção que apresenta graficamente o *address space* modelado;

C - seção que serve para criar *nós*, definir instâncias de *objetos* e editar os seus atributos ou referências, para modelar o *address space*;

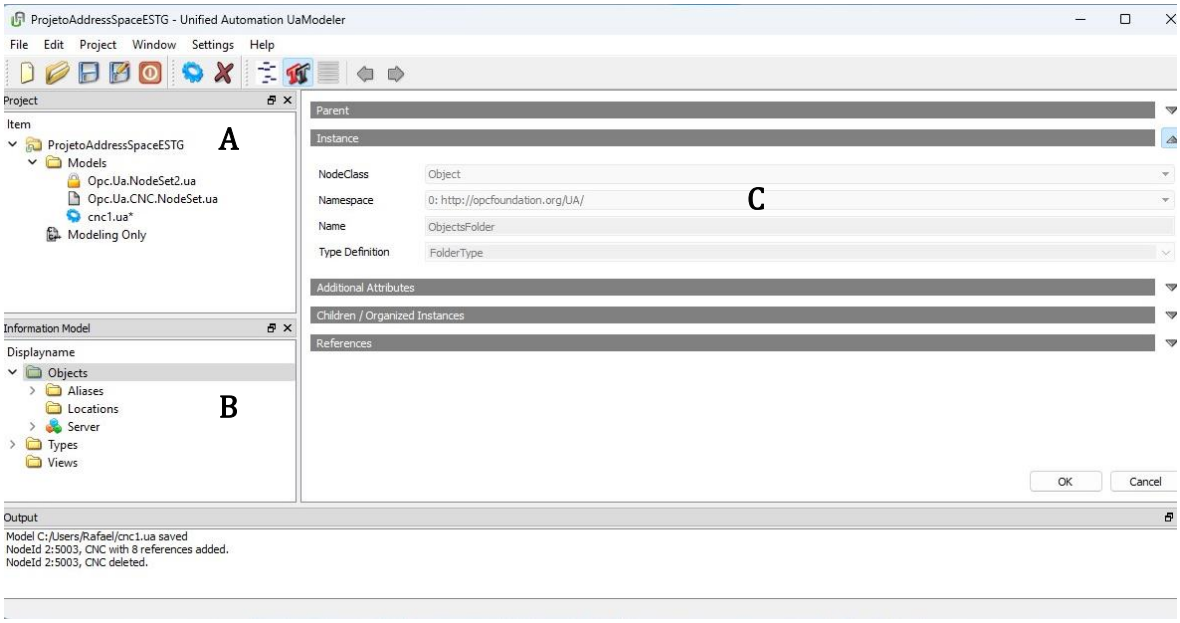


Figura A16 - Iniciar um novo projeto no UAModeler (parte V).

Na secção B da Figura A16 vê-se a mínima estrutura que, por convenção, carateriza um *address space*, é derivada do meta-modelo OPC UA e será personalizada respeitando modelos de informação padronizados, neste caso a CS "OPC UA for CNC Systems".

3.2. Modelar um address space através do UAModeler

Pretende-se obter um modelo de *address space*, cujos *objetos* estão organizados de acordo com o exemplo de descrição de uma máquina-ferramenta com 3 eixos indicado por esta CS. Para além disso, consultou-se o documento "OPC UA for CNC Systems" para averiguar quais são os *objetos* obrigatórios deste modelo de informação e como estes se caraterizam. Adicionalmente, optou-se por incluir neste *address space* alguns *métodos* disponibilizados pela API do EdingCNC, tais como Load Job, Run Job, Pause Job ou RESET.

A partir dessas especificações, inicia-se a modelação do *address space* no UAModeler. Para isso, no UAModeler, serão adicionados *nós* que definirão *instâncias* de *tipos* de *objetos* definidos por esta CS e serão preenchidos os seus atributos ou referências.

Na secção B, clique, com o botão direito do rato, sobre **Objects** e selecione **Add Instance**. O UAModeler passa a apresentar-se como a Figura A17.

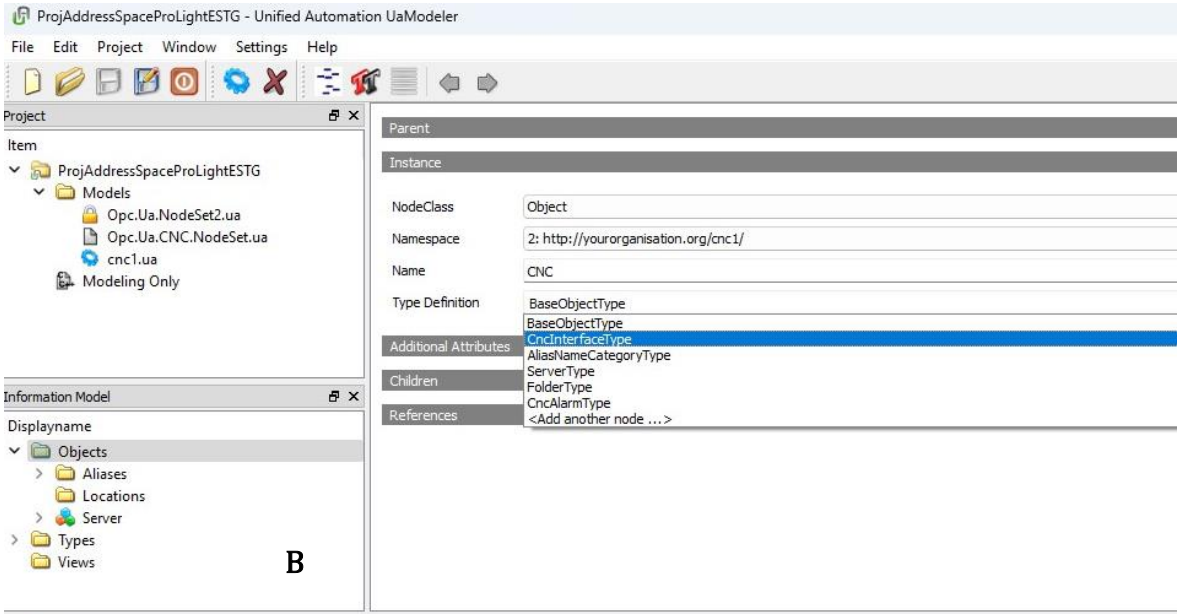


Figura A17 - Modelação de um *address space*, no UAModeler (adição de um *nó*).

Na secção B da Figura A17 surgem parâmetros que podem ser editados para definir as propriedades do *nó* a adicionar, como a sua *NodeClass*, *namespace*, *nome*, entre outros. Para adicionar um *nó* que é *instância* de um pretendido e determinado tipo de *objeto* da CS "CNC Systems", deve-se, no campo **NodeClass**, selecionar **Object** e, no campo **Type Definition**, selecionar **<Add another node…>**, o que faz abrir outra janela.

Nesta janela do UAModeler (Figura A18), é possível selecionar um determinado *tipo* de *objeto*, de entre aqueles definidos pelos modelos de informação OPC UA importados. Os *tipos* de *objetos* definidos pela CS "CNC Systems" distinguem-se porque estão realçados.

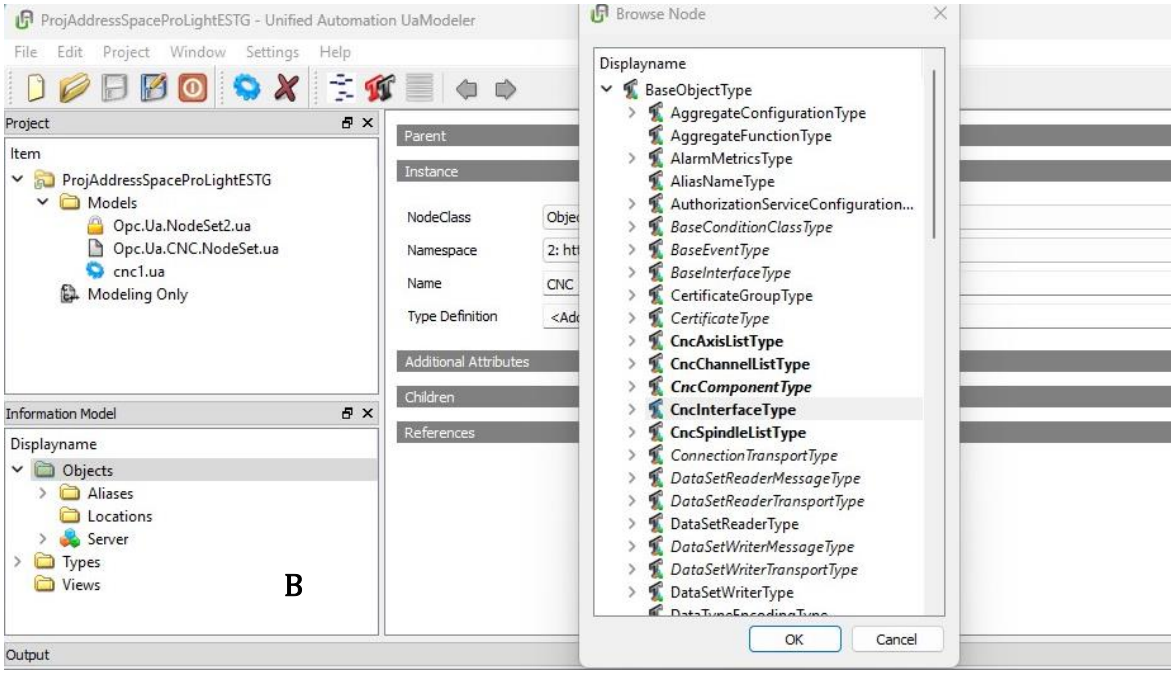


Figura A18 - Modelação de um *address space*, no UAModeler (seleção do *tipo* de *objeto* para um *nó*).

Deve-se selecionar o *tipo* de *objeto* pretendido, clicar em **OK** para fechar essa janela e, novamente, clicar em **OK**. Esse *objeto* passa a estar visível na seção B. Desta forma, são adicionados ao *address space* os elementos pretendidos, para criar uma estrutura personalizada em concordância com os modelos de informação OPC UA. De seguida, deve-se clicar em **File** e em **Save**.

Como resultado, o *address space*, obtido deste processo de modelação, deve ser igual aquele apresentado na Figura A19.

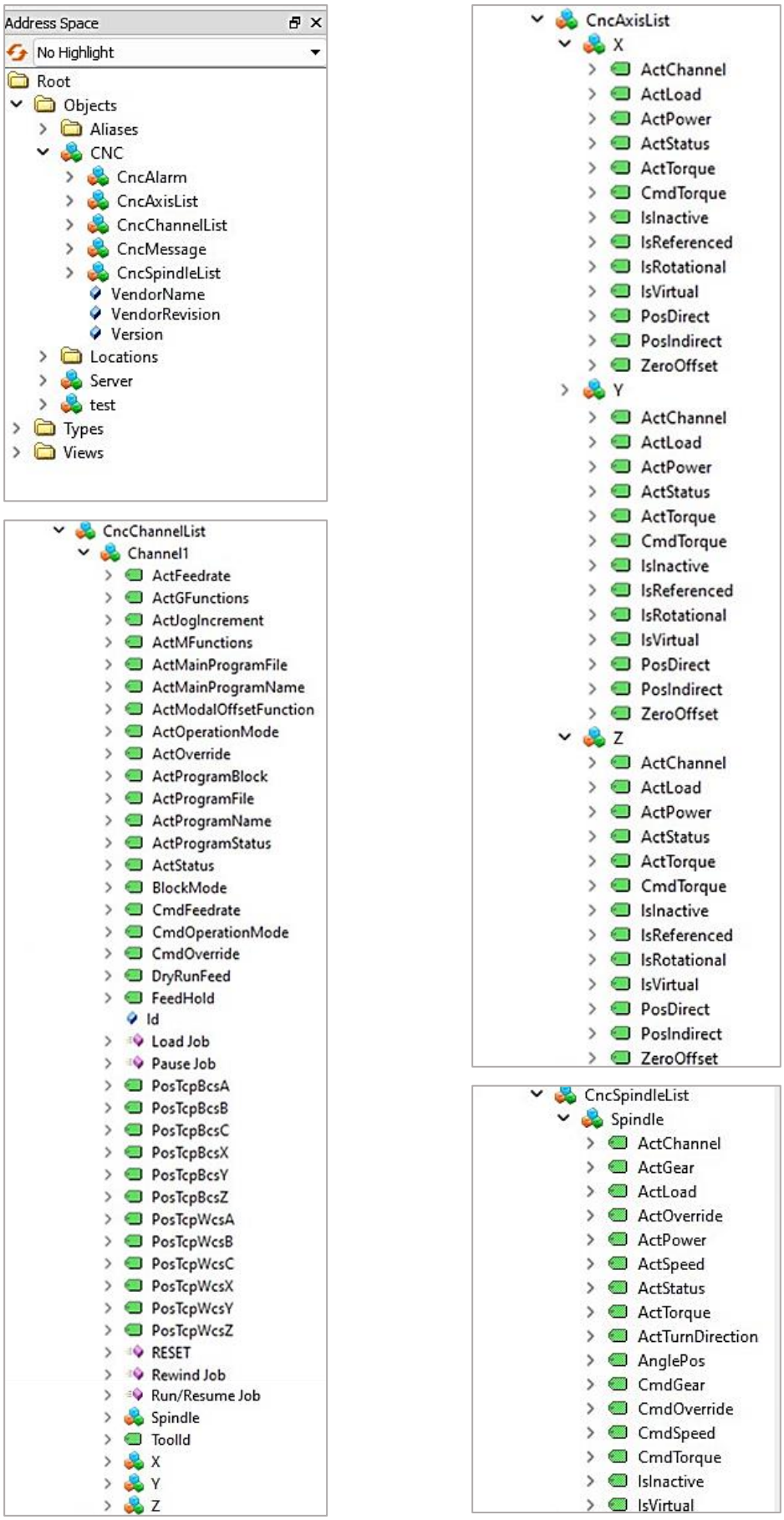


Figura A19 - Diferentes vistas do *address space* modelado.

3.3. Exportar o address space

Depois de concluída a modelação do *address space*, o código que define a sua estrutura pode ser exportado, num ficheiro em linguagem XML, como mostra a Figura A20. Para o fazer, deve-se clicar, com o botão direito do rato, na secção A, sobre o ícone que representa este modelo e clicar, com o botão esquerdo, em **Export XML**. Por fim, feche o UAModeler

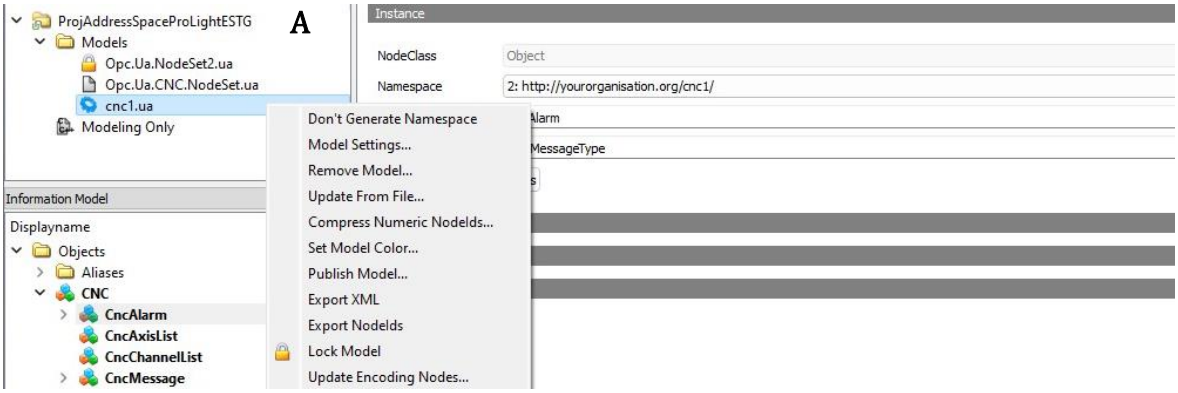


Figura A20 - Exportar um *address space* em formato *.xml*.

Para uma licença gratuita, que neste caso foi utilizada, este *software* tem um limite para o numero de *nós* que um modelo a exportar pode ter. No entanto, isso pode ser contornado, editando manualmente o ficheiro XML. A Figura A21 mostra um excerto desse ficheiro.

```
▼<UAVariable DataType="Boolean" ParentNodeId="ns=1;s=a.s5009" NodeId="ns=1;s=a.s6169" BrowseName="2:Block
   <DisplayName Locale="en">BlockMode</DisplayName>
   <Description Locale="en">Block mode status (true in case of block mode is active, else false).</Descri
  ▼<References>
     <Reference ReferenceType="HasTypeDefinition">i=2365</Reference>
     <Reference ReferenceType="HasComponent" IsForward="false">ns=1;s=a.s5009</Reference>
   </References>
  ▼<Value>
     <uax:Boolean>false</uax:Boolean>
   </Value>
 </UAVariable>
▼<UAVariable DataType="Double" ParentNodeId="ns=1;s=a.s5009" NodeId="ns=1;s=a.s6170" BrowseName="2:CmdFee
   <DisplayName Locale="en">CmdFeedrate</DisplayName>
```

Figura A21 - Excerto do ficheiro *cnc1.xml* que foi exportado do UAModeler.

Pelo motivo que é explicado na página 97, optou-se por recorrer a *NodeIds* identificados por uma `string` para identificar os *nós* deste *address space* modelado. Através do UAExpert, é possível verificar previamente o valor `int` de cada *NodeId*. Para editar estes *NodeIds*, abra o ficheiro *cnc*1.*xml* através da aplicação **Bloco de Notas** e clique em **Editar** > **Substituir**. No campo **Localizar** introduza, por exemplo, "i=6069" e no campo **Substituir** introduza a nova descrição, que neste exemplo seria "s=a.s6069", e clique em **Substituir Tudo**. Deve-se repetir este processo para todos os *NodeIds* necessários.

4. Servidor OPC UA

4.1. Ficheiros a serem compilados

Para criar o servidor OPC UA, é necessário criar uma pasta e nela colocar os ficheiros que serão usados para compilar uma *solução* com o CMake e, de seguida, com o Visual Studio. O nome atribuído a esta pasta será o nome da *solução*, no CMake e no Visual Studio.
Foi atribuído o nome *open62541-nodeset-compiler-example*.

Como mostra a Figura A22, a pasta *open62541-nodeset-compiler-example* tem de conter os seguintes elementos:

- os ficheiros *open62541.c* e *open62541.h*;
- uma pasta designada por *build* e outra pasta designada por *model*;
- um ficheiro designado por *CMakeLists.txt*;
- um ficheiro designado por *main.c*;
- um ficheiro designado por *cnc.xml* (ficheiro de configurações do servidor OPC UA)

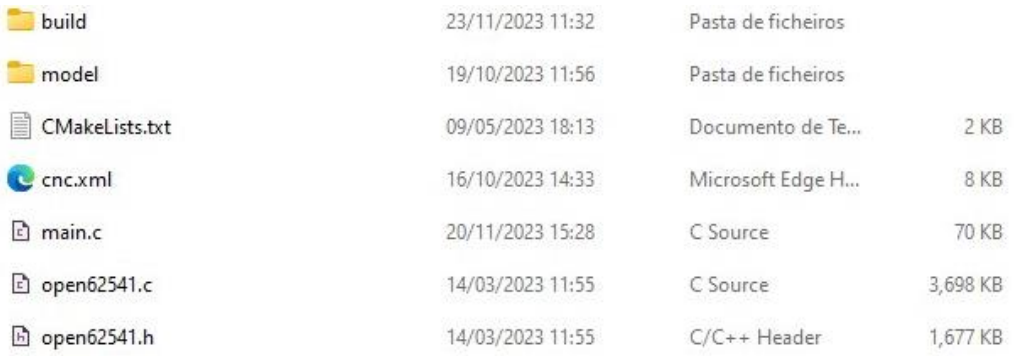


Figura A22 - Ficheiros que a pasta *open62541-nodeset-compiler-example* deve conter.

Opc.Ua.CNC.NodeSet.bsd, *Opc.Ua.CNC.NodeSet.csv* e *Opc.Ua.CNC.NodeSet.xml* são ficheiros que caraterizam o conjunto de *nós* (*NodeSet*) desta *companion specification* e que devem ser guardados na pasta *model*, juntos com o ficheiro *cnc*1.*xml*, como mostra a Figura A23. Os ficheiros designados *Opc.Ua.CNC.NodeSet*.* devem ser descarregados do *website* https://github.com/OPCFoundation/UA-Nodeset/tree/latest/CNC.

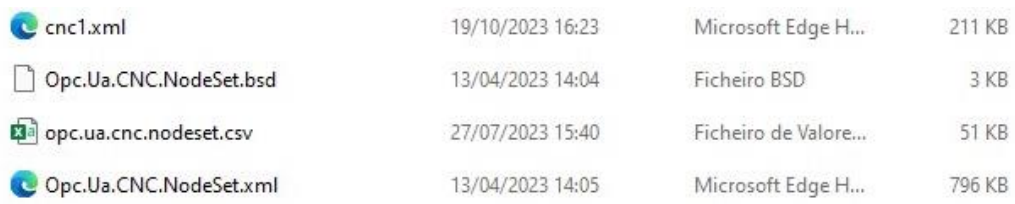


Figura A23 - Ficheiros que a pasta *model* deve conter.

O ficheiro *CMakeLists.txt* contém as instruções que o CMake deve utilizar para compilar o conteúdo da pasta *open62541-nodeset-compiler-example* e foi descarregado do repositório *github* de Stefan Profanter, um colaborador do projeto open62541. Deve-se aceder ao *site* https://github.com/Pro/opcua-modeling-tutorial-server/blob/master/CMakeLists.txt, com o botão esquerdo do rato, clicar sobre **Raw**, o que faz carregar uma nova página, na qual, com o botão direito, se deve clicar, selecionar **guardar página como**, no campo **Guardar página com o tipo:** selecionar *.txt e, por fim, clicar em **Guardar**. A Figura A24 mostra o ficheiro *CMakeLists.txt* que foi descarregado.

```
```
cmake_minimum_required(VERSION 3.2)
project(opcua-modeling-tutorial-server)

# open62541 must be installed.
# If in custom path, then use -DCMAKE_PREFIX_PATH=/home/user/install
find_package(open62541 1.1 REQUIRED COMPONENTS FullNamespace)

# Output directory for Nodeset Compiler
set(GENERATE_OUTPUT_DIR "${CMAKE_BINARY_DIR}/src_generated/")
file(MAKE_DIRECTORY "${GENERATE_OUTPUT_DIR}")
include_directories("${GENERATE_OUTPUT_DIR}")

ua_generate_nodeset_and_datatypes(
NAME "foo_flt"
TARGET_PREFIX "${PROJECT_NAME}"
FILE_CSV "${PROJECT_SOURCE_DIR}/model/Published/FooFlt/FooFltModel.csv"
FILE_BSD "${PROJECT_SOURCE_DIR}/model/Published/FooFlt/FooFlt.Types.bsd"
OUTPUT_DIR "${GENERATE_OUTPUT_DIR}"

This namespace index must match the order in which you are adding the
nodeset in the source code
NAMESPACE_MAP "2:https://new.foo.com/zebra-compression/flattening-and-
subspacefolding/UA/"
FILE_NS "${PROJECT_SOURCE_DIR}/model/Published/FooFlt/FooFlt.NodeSet2.xml"

INTERNAL
)

Previous macro automatically sets some variables which hold the generated
source code files using the provided NAME
add_executable(opcua-modeling-tutorial-server
${UA_NODESET_FOO_FLT_SOURCES}
${UA_TYPES_FOO_FLT_SOURCES}
main.c
)

Make sure the nodeset compiler is execute before compiling the main file

add_dependencies(opcua-modeling-tutorial-server
${PROJECT_NAME}-ns-foo_flt
)

target_link_libraries(opcua-modeling-tutorial-server open62541::open62541)
```
```

Figura A24 - Conteúdo do ficheiro *CMakeLists.txt* descarregado.

O código desse ficheiro *CMakeLists.txt* deve ser adaptado a este projeto em particular, ou seja, deve ser adaptado para compilar o *nodeset* da CS "CNC Systems".

Neste código, todas as ocorrências da palavra *opcua-modeling-tutorial-server* devem ser substituídas pelo nome desta solução, que é *open62541-nodeset-compiler-example*, e todas as ocorrências da palavra *FooFit* devem ser substituídas pelo nome do modelo utilizado, que é *Opc.Ua.CNC.NodeSet*. Este ficheiro deve passar a apresentar-se como a Figura A25.

```
```
cmake_minimum_required(VERSION 3.2)
project(open62541-nodeset-compiler-example)

# open62541 must be installed.
# If in custom path, then use -DCMAKE_PREFIX_PATH=/home/user/install
find_package(open62541 1.1 REQUIRED COMPONENTS FullNamespace)

# Output directory for Nodeset Compiler
set(GENERATE_OUTPUT_DIR "${CMAKE_BINARY_DIR}/src_generated/")
file(MAKE_DIRECTORY "${GENERATE_OUTPUT_DIR}")
include_directories("${GENERATE_OUTPUT_DIR}")

ua_generate_nodeset_and_datatypes(
NAME "cnc"
TARGET_PREFIX "${PROJECT_NAME}"
FILE_CSV "${PROJECT_SOURCE_DIR}/model/opc.ua.cnc.nodeset.csv"
FILE_BSD "${PROJECT_SOURCE_DIR}/model/Opc.Ua.CNC.NodeSet.bsd"
OUTPUT_DIR "${GENERATE_OUTPUT_DIR}"

This namespace index must match the order in which you are adding the
nodeset in the source code
NAMESPACE_MAP "2:http://opcfoundation.org/UA/CNC"
FILE_NS "${PROJECT_SOURCE_DIR}/model/Opc.Ua.CNC.NodeSet.xml"
INTERNAL
)

ua_generate_nodeset_and_datatypes(
NAME "cnctest"
TARGET_PREFIX "${PROJECT_NAME}"
OUTPUT_DIR "${GENERATE_OUTPUT_DIR}"

This namespace index must match the order in which you are adding the
nodeset in the source code
NAMESPACE_IDX 3
FILE_NS "${PROJECT_SOURCE_DIR}/model/cnc1.xml"
DEPENDS "cnc"
INTERNAL
)

Previous macro automatically sets some variables which hold the generated
source code files using the provided NAME
add_executable(open62541-nodeset-compiler-example
${UA_NODESET_CNC_SOURCES}
${UA_TYPES_CNC_SOURCES}
${UA_NODESET_CNCTEST_SOURCES}
main.c
)

Make sure the nodeset compiler is execute before compiling the main file
add_dependencies(open62541-nodeset-compiler-example
${PROJECT_NAME}-ns-cnc
)

target_link_libraries(open62541-nodeset-compiler-example open62541::open62541)
```
```

Figura A25 - Conteúdo do ficheiro *CMakeLists.txt* utilizado.

O ficheiro *CMakeLists.txt* deve ser guardado na pasta *open62541-nodeset-compiler-example*.

O passo seguinte consiste em criar o ficheiro *main*.c que se vê na Figura A22. Para o fazer, clique, com o botão direito do rato, dentro da pasta *open62541-nodeset-compiler-example* e, com o botão esquerdo, selecione **Novo > Documento de Texto**. Com o botão direito do rato, clique nesse ficheiro, selecione **Mudar o Nome**, escreva *main.c*, pressione **Enter** e clique em **Sim**.

A seguir, deve-se consultar o tutorial de como criar um servidor OPC UA, na documentação do open62541, no *site* https://www.open62541.org/doc/1.3/tutorial_server_firststeps.html. A Figura A26 mostra o código que é indicado nesse tutorial e que deve ser copiado.

```cpp
#include <open62541/plugin/log_stdout.h>
#include <open62541/server.h>
#include <open62541/server_config_default.h>
#include <signal.h>
#include <stdlib.h>

static volatile UA_Boolean running = true;

static void stopHandler(int sig) {
    UA_LOG_INFO(UA_Log_Stdout, UA_LOGCATEGORY_USERLAND, "received ctrl-c");
    running = false;
}

int main(void) {
    signal(SIGINT, stopHandler);
    signal(SIGTERM, stopHandler);

    UA_Server *server = UA_Server_new();
    UA_ServerConfig_setDefault(UA_Server_getConfig(server));

    UA_StatusCode retval = UA_Server_run(server, &running);

    UA_Server_delete(server);
    return retval == UA_STATUSCODE_GOOD ? EXIT_SUCCESS : EXIT_FAILURE;
}
```

Figura A26 - Código para implementar um servidor OPC UA, de acordo com a documentação da *stack open62541*.

Com o botão esquerdo do rato, clique duas vezes sobre o ficheiro *main*.c, para o abrir com o Visual Studio. Nesse ficheiro deve-se colar o código copiado. Com o botão esquerdo do rato, selecione **Arquivo > Salvar Tudo** e feche o Visual Studio.

Na pasta *open62541-nodeset-compiler-example* foi também guardado o ficheiro *cnc.xml*, como mostra a Figura A22. A Figura A27 mostra um excerto do ficheiro *cnc.xml* .

```
```

<parameters>

<add key="CNC_CONTROLLER" value="EDING"/>
<!-- Values available FANUC, HEIDENHAIN or EDING -->

<add key="CNC_IP" value="192.168.56.101"/>
<!-- Enter IP from the CNC controller you want to connect -->

<add key="CNC_PORT" value="19000"/>
<!-- Enter PORT from the CNC controller you want to connect, common ports
8193 (FANUC) 19000 (HEIDENHAIN) -->

<add key="POWER_METER_IP" value="192.168.20.4"/>
<!-- Enter IP from the power meter you want to connect or NONE,
192.168.20.4 -->

<add key="POWER_METER_PORT" value="502"/>
<!-- Enter PORT from the power meter you want to connect or 0, 502 -->

<add key="VIDEO_IN_PATH" value="0"/>
<!-- Enter the path from source stream "DALSA" or
http://24.181.120.98:8080/mjpg/video.mjpg -->

<add key="VIDEO_OUT_PATH" value="../../../videos/"/>
<!-- Enter the path to save error videos, ../../videos/ -->

<add key="VIDEO_FPS" value="30"/>
<!-- Enter the number of frames per second to save error videos, 30 -->

<add key="VIDEO_LENGHT" value="10"/>
<!-- Enter the video length (in seconds) to save error videos, 10 -->

<add key="INFLUXDB" value="true"/>
<!-- Saves data do InfluxDB -->

<add key="Get_VendorName" value="6001,false,1"/>

<add key="Get_VendorRevision" value="6002,false,1"/>

<add key="Get_Version" value="6003,false,1"/>

<add key="Get_AlarmMessage" value="6044,true,1"/>

<add key="Get_X1_ActStatus" value="6057,true,3"/>

<add key="Get_x1_Referenced" value="6065,true,1"/>

<add key="Get_X1_Rotational" value="6066,false,1"/>

<add key="Get_X1_PosDirectAct" value="6069,true,1"/>

<add key="Get_X1_PosDirectCmd" value="6070,false,1"/>

<add key="Get_X1_PosDirectRemD" value="6073,false,1"/>

<add key="Get_X1_PosIndirectAct" value="6075,true,1"/>

<add key="Get_X1_PosIndirectCmd" value="6076,false,1"/>

...

<parameters>

```
```

Figura A27 - Excerto do ficheiro *cnc.xml*.

4.2. Compilar o servidor OPC UA

Depois de todos os ficheiros, que estão apresentados na Figura A22, terem sido guardados na pasta *open62541-nodeset-compiler-example*, conforme os procedimentos indicados no subcapítulo anterior, dá se inicio à sua compilação, para se obter o servidor OPC UA.

Abra o CMake e clique em **File** e em **Delete Cache**. No campo **"Where is the source code:"** introduza o caminho da pasta *open62541-nodeset-compiler-example*. No campo **"Where to build the binaries:"** introduza esse mesmo caminho, ao qual deve-se acrescentar no final: /build . Clique em **Configure**, **Finish** e **Generate** e feche o CMake. A pasta *build*, que estava vazia, passou a conter vários ficheiros, como se vê na Figura A28.

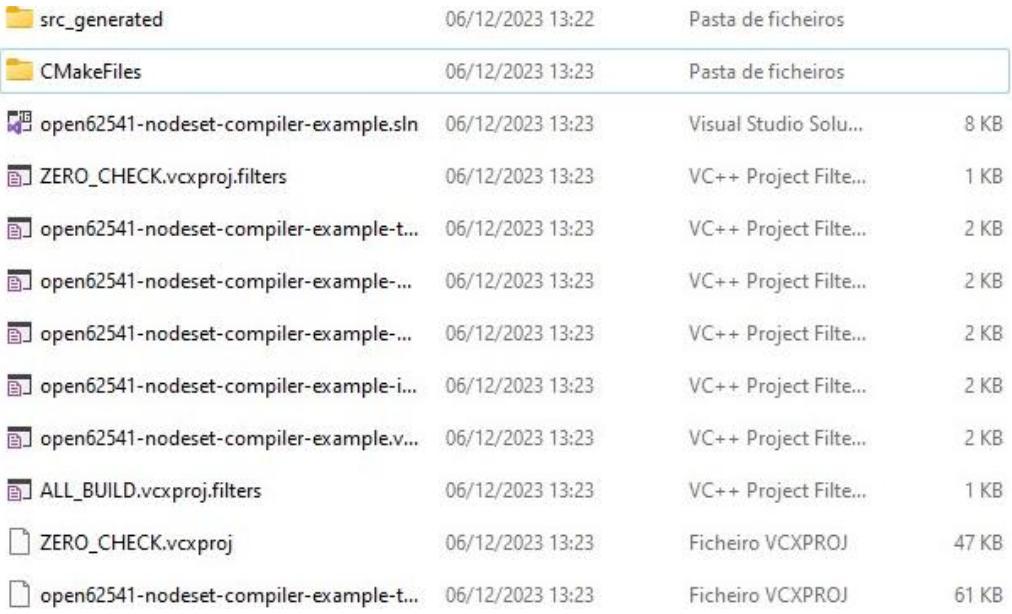


Figura A28 - Excerto do conteúdo da pasta *build*.

Com o botão esquerdo do rato, clique duas vezes sobre o ficheiro *open62541-nodeset-compiler-example.sln*, para o abrir através do Visual Studio. No **Gerenciador de Soluções** (Figura A29) do Visual Studio, com o botão direito do rato, clique no projeto *open62541-nodeset-compiler-example* e selecione **Definir como Projeto de Inicialização**.

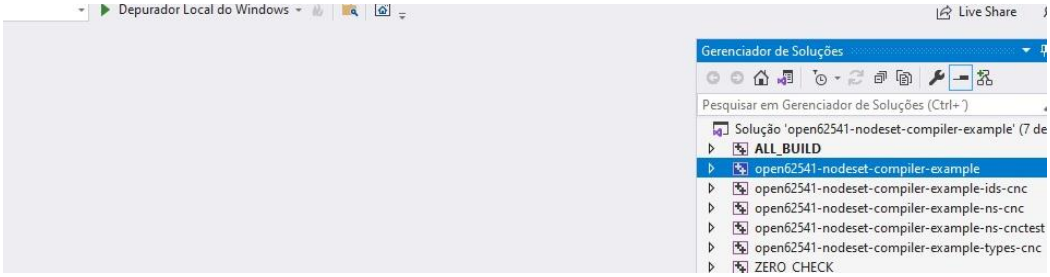


Figura A29 - Janela do Visual Studio que mostra o ficheiro *open62541-nodeset-compiler-example.sln*.

No **Gerenciador de Soluções**, com o botão direito do rato, clique sobre o projeto *open62541-nodeset-compiler-example* e selecione **Propriedades**. Aceda a **Vinculador > Entrada** e escreva *ws2_32.lib* no campo **Dependências Adicionais**.

Clique, com o botão direito do rato, sobre **Solução open62541-nodeset-compiler-example**, selecione **Compilar Solução** e feche o Visual Studio. Como mostra a Figura A30, a pasta *src_generated*, que estava vazia, passou a conter vários ficheiros.

Figura A30 - Ficheiros resultantes da compilação da *solução* no Visual Studio.

Os ficheiros indicados na Figura A30 resultam da compilação dos ficheiros *cnc*1.*xml* e *Opc.Ua.CNC.NodeSet* (*.bsd, .csv* e *.xml*) e contêm o *namespace*, *NodeIds* e *tipos* de *objetos* da *companion specification* "OPC UA for CNC Systems", geradas em código C/C++.

Para além disso, como resultado desta compilação, é guardado na pasta *Debug* o servidor OPC UA no formato de um ficheiro executável (*.exe*). No entanto, como o ficheiro *main*.c ainda contém apenas o código de um servidor OPC UA vazio, o nosso servidor OPC UA ainda não possui nenhuma funcionalidade. Para que o servidor OPC UA em formato *.exe* funcione corretamente, deve-se guardar o uma cópia do ficheiro *cnc.xml* na pasta *Debug*.

Além disso, como se vê na Figura A31, a pasta *src_generated* passa a ser utilizada para guardar os ficheiros que serão importados para o Visual Studio.

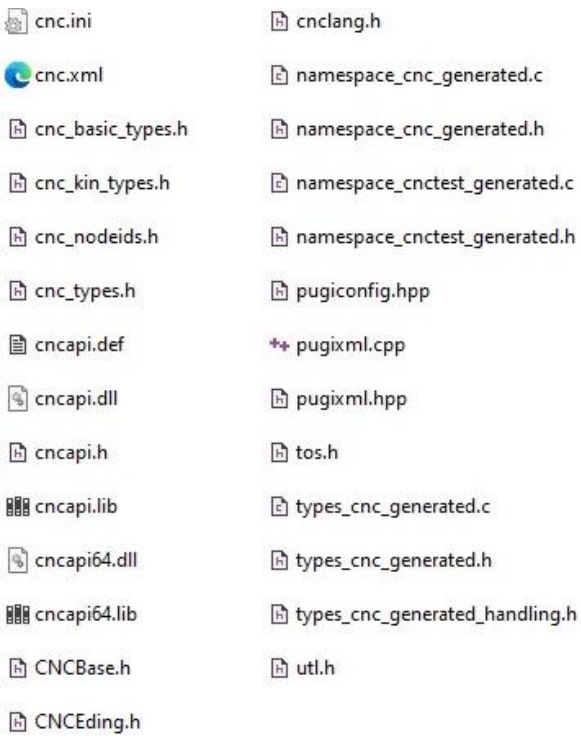


Figura A31 - Ficheiros guardados na pasta *src_generated*.

Para que o servidor OPC UA possa comunicar com a API do EdingCNC, utilizaram-se os ficheiros localizados na diretoria "(…)\CNC4.03\cncapi" (Figura A32). À exceção da pasta *python* e seu conteúdo, todos estes ficheiros foram copiados para a pasta *src_generated*. Estes ficheiros, visíveis na Figura A32, são um conjunto de bibliotecas, em linguagem C, que servem para interagir com o servidor CNCServer do EdingCNC.

python	30/11/2022 11:47	Pasta de ficheiros	
cnc_basic_types.h	18/11/2022 14:41	C/C++ Header	32 KB
cnc_kin_types.h	18/11/2022 14:41	C/C++ Header	9 KB
cnc_types.h	18/11/2022 14:41	C/C++ Header	95 KB
cncapi.h	18/11/2022 14:41	C/C++ Header	71 KB
cnclang.h	18/11/2022 14:41	C/C++ Header	3 KB
tos.h	18/11/2022 14:41	C/C++ Header	20 KB
utl.h	18/11/2022 14:41	C/C++ Header	11 KB
cncapi.def	18/11/2022 14:41	Export Definition ...	5 KB
cncapi.dll	18/11/2022 14:41	Extensão da aplica...	478 KB
cncapi64.dll	18/11/2022 14:41	Extensão da aplica...	535 KB
cncapi.lib	18/11/2022 14:41	Object File Library	49 KB
cncapi64.lib	18/11/2022 14:41	Object File Library	44 KB

Figura A32 - Conteúdo da pasta *cncapi*, do EdingCNC.

Foram guardados na pasta *src_generated* os ficheiros *cnc.xml*, *pugixml* (*.cpp* e *.hpp*) e *pugiconfig.hpp*. Os três ficheiros designados de "*pugi*" correspondem a uma biblioteca que foi utilizada para escrever o código das funcionalidades do servidor OPC UA. A biblioteca *pugi* foi desenvolvida em C++, serve para ler ficheiros XML e pode ser descarregada dos *websites* https://github.com/zeux/pugixml ou https://pugixml.org/.

A Figura A31 mostra os ficheiros *CNCBase.h* e *CNCEding.h*, que correspondem à *classe* genérica abstrata e à *classe* especifica para o controlador EdingCNC, respetivamente.

Para criar essas classes, no Visual Studio, neste projeto, com o botão direito do rato, clique sobre **Source Files** e, com o botão esquerdo, selecione **Adicionar > Novo Item > Classe do C++**. Abre-se uma janela, onde se deve indicar que estas *classes* serão guardadas na pasta *src_generated* e que a *classe* base da CNCEding é a CNCBase.

Deve-se preencher os ficheiros *main.c*, *CNCBase.h* e *CNCEding.h* com o respetivo código que se encontra disponível no repositório *github* do autor (https://github.com/pedro-rafael-sousa/OPC-UA-Server-for-Eding-ProLIGHT1000/).

A seguir, deve-se importar o conteúdo da pasta *src_generated* para o Visual Studio. Para fazer isso, abra o ficheiro *open62541-nodeset-compiler-example.sln* através do Visual Studio, com o botão direito do rato, clique sobre **Source Files** e, com o botão esquerdo, selecione **Adicionar > Item Existente** e, por fim, selecione todo o conteúdo da pasta *src_generated* e clique em **Adicionar**.

O **Gerenciador de Soluções** deste projeto deve passar a apresentar-se como se vê na Figura A33.

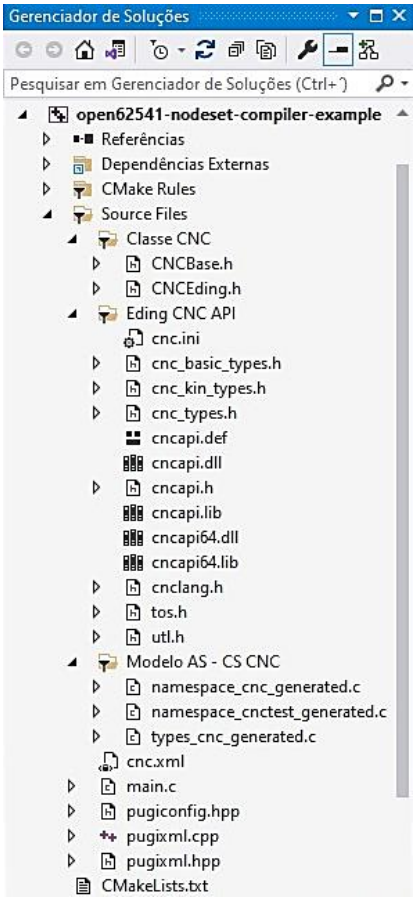


Figura A33 - Projeto *open62541-nodeset-compiler-example*, no Visual Studio.

Com o botão direito do rato, clique sobre **Solução 'open62541-nodeset-compiler-example'** e, com o botão esquerdo, clique em **Recompilar Solução**. O desenvolvimento deste servidor OPC UA passa a estar concluído e o ficheiro executável (Figura A34), guardado na pasta *Debug*, passa a conter todas as funcionalidades definidas.

Para testar este servidor, pode-se executar o servidor executável ou, no Visual Studio, clicar sobre **Depurador Local do Windows** para o correr, o que faz abrir a Consola do Windows (Figura A36), onde se visualiza o servidor OPC UA a chamar periodicamente o *método PeriodicUpdate* que foi desenvolvido para ler parâmetros do EdingCNC e escrever esses valores nos *objetos* do seu *address space*.

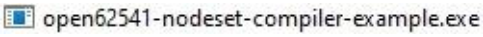


Figura A34 - Ficheiro *open62541-nodeset-compiler-example.exe*, guardado na pasta *Debug*.

Figura A35 - Excerto de uma janela do Visual Studio, onde se vê o comando *Depurador Local do Windows*.

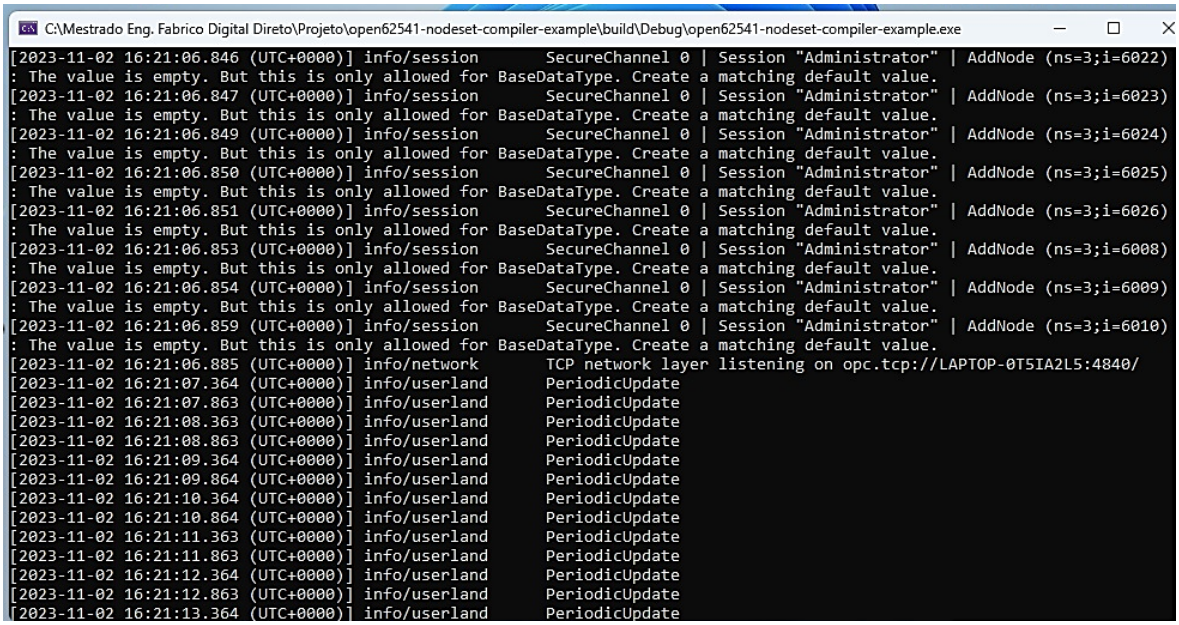


Figura A36 - Consola do Windows, onde se vê o servidor OPC UA a correr.

Na Figura A36 é possível ver o endereço URL deste servidor OPC UA, que é utilizado por clientes OPC UA para estabelecerem a ligação com este servidor.

5. Testes ao funcionamento do servidor OPC UA

Para testar o funcionamento do servidor OPC UA, deve-se recorrer ao programa UAExpert, que é fornecido pela empresa UnifiedAutomation e que implementa um cliente OPC UA genérico com uma interface gráfica.

Deve-se começar por executar o servidor OPC UA, abrir o UAExpert e clicar, com o botão esquerdo do rato, no ícone **Add Server** (Figura A37). Uma nova janela abre-se.

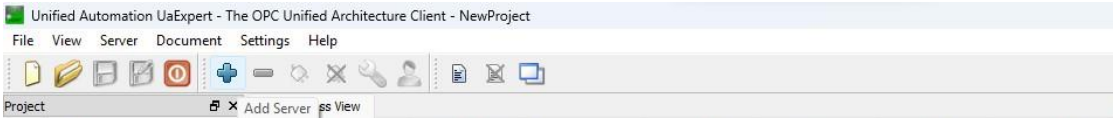


Figura A37 - Excerto da interface gráfica do UAExpert, onde se vê o comando *Add Server*.

De seguida, deve-se fazer duplo clique no ícone abaixo de **Custom Discovery** e inserir o URL do servidor OPC UA, tal como mostra a Figura A38.

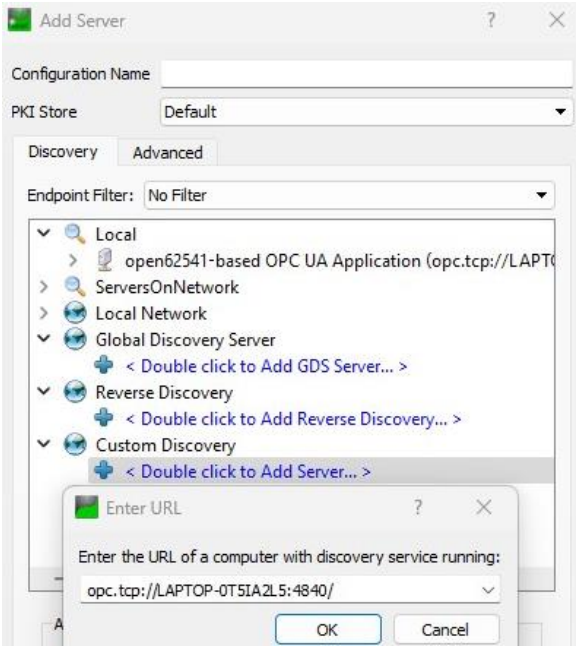


Figura A38 - Janela *Add Server* do UAExpert.

Deve-se expandir o ícone deste servidor OPC UA, selecionar o seu modo de segurança, que neste caso é **None**, e clicar em **OK**. O UAExpert passa a estar conectado ao servidor OPC UA e a expor o seu *address space*, que pode ser navegado, os seus *métodos* podem ser acionados e os seus o*bjetos* podem ser arrastados para a janela **Data Access View** para os seus valores serem continuamente visualizados, como se vê nas Figuras A39 a A45.

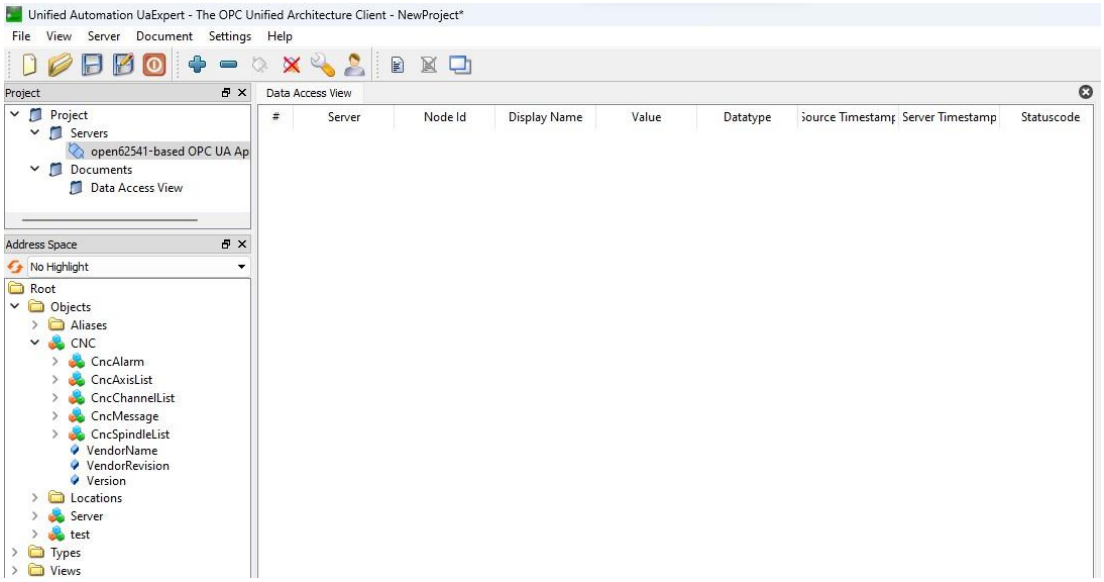


Figura A39 - UAExpert conectado ao servidor OPC UA desenvolvido.

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp
1	open62541-based OPC UA App...	NS3|String|a.s6050	ActChannel	3|String|a.s5009	NodeId	14:12:12.121	14:12:12.121
2	open62541-based OPC UA App...	NS3|String|a.s6051	ActLoad		Null	14:12:12.121	14:12:12.121
3	open62541-based OPC UA App...	NS3|String|a.s6054	ActPower		Null	14:12:12.121	14:12:12.121
4	open62541-based OPC UA App...	NS3|String|a.s6057	ActStatus	0 (InPosition)	Int32	14:15:08.623	14:15:08.623
5	open62541-based OPC UA App...	NS3|String|a.s6058	ActTorque		Null	14:12:12.121	14:12:12.121
6	open62541-based OPC UA App...	NS3|String|a.s6061	CmdTorque		Null	14:12:12.121	14:12:12.121
7	open62541-based OPC UA App...	NS3|String|a.s6064	IsInactive	false	Boolean	14:12:12.121	14:12:12.121
8	open62541-based OPC UA App...	NS3|String|a.s6065	IsReferenced	true	Boolean	14:14:07.123	14:14:07.123
9	open62541-based OPC UA App...	NS3|String|a.s6066	IsRotational	false	Boolean	14:12:12.121	14:12:12.121
10	open62541-based OPC UA App...	NS3|String|a.s6067	IsVirtual	true	Boolean	14:12:12.121	14:12:12.121
11	open62541-based OPC UA App...	NS3|String|a.s6080	ZeroOffset	8	Double	14:14:07.123	14:14:07.123
12	open62541-based OPC UA App...	NS3|String|a.s6075	ActPos	41.510632546	Double	14:15:09.123	14:15:09.123
13	open62541-based OPC UA App...	NS3|String|a.s6076	CmdPos	0	Double	14:14:07.364	14:14:07.364
14	open62541-based OPC UA App...	NS3|String|a.s6079	RemDist	0	Double	14:14:07.365	14:14:07.365
15	open62541-based OPC UA App...	NS3|String|a.s6069	ActPos	41.510632546	Double	14:15:09.123	14:15:09.123
16	open62541-based OPC UA App...	NS3|String|a.s6070	CmdPos	0	Double	14:14:07.366	14:14:07.366
17	open62541-based OPC UA App...	NS3|String|a.s6073	RemDist	0	Double	14:14:07.367	14:14:07.367
18	open62541-based OPC UA App...	NS3|String|a.s6083	ActChannel	3|String|a.s5009	NodeId	14:12:12.121	14:12:12.121
19	open62541-based OPC UA App...	NS3|String|a.s6084	ActLoad		Null	14:12:12.121	14:12:12.121
20	open62541-based OPC UA App...	NS3|String|a.s6087	ActPower		Null	14:12:12.121	14:12:12.121
21	open62541-based OPC UA App...	NS3|String|a.s6090	ActStatus	0 (InPosition)	Int32	14:15:08.623	14:15:08.623
22	open62541-based OPC UA App...	NS3|String|a.s6091	ActTorque		Null	14:12:12.121	14:12:12.121
23	open62541-based OPC UA App...	NS3|String|a.s6094	CmdTorque		Null	14:12:12.121	14:12:12.121
24	open62541-based OPC UA App...	NS3|String|a.s6097	IsInactive	false	Boolean	14:12:12.121	14:12:12.121
25	open62541-based OPC UA App...	NS3|String|a.s6098	IsReferenced	true	Boolean	14:14:52.124	14:14:52.124
26	open62541-based OPC UA App...	NS3|String|a.s6099	IsRotational	false	Boolean	14:12:12.121	14:12:12.121
27	open62541-based OPC UA App...	NS3|String|a.s6100	IsVirtual	true	Boolean	14:12:12.121	14:12:12.121
28	open62541-based OPC UA App...	NS3|String|a.s6113	ZeroOffset	6	Double	14:14:52.125	14:14:52.125
29	open62541-based OPC UA App...	NS3|String|a.s6108	ActPos	48.7354771324	Double	14:15:09.123	14:15:09.123
30	open62541-based OPC UA App...	NS3|String|a.s6109	CmdPos	0	Double	14:14:52.573	14:14:52.573
31	open62541-based OPC UA App...	NS3|String|a.s6112	RemDist	0	Double	14:14:52.575	14:14:52.575
32	open62541-based OPC UA App...	NS3|String|a.s6102	ActPos	48.7354771324	Double	14:15:09.123	14:15:09.123
33	open62541-based OPC UA App...	NS3|String|a.s6103	CmdPos	0	Double	14:14:52.575	14:14:52.575
34	open62541-based OPC UA App...	NS3|String|a.s6106	RemDist	0	Double	14:14:52.577	14:14:52.577
35	open62541-based OPC UA App...	NS3|String|a.s6116	ActChannel	3|String|a.s5009	NodeId	14:12:12.121	14:12:12.121
36	open62541-based OPC UA App...	NS3|String|a.s6117	ActLoad		Null	14:12:12.121	14:12:12.121
37	open62541-based OPC UA App...	NS3|String|a.s6120	ActPower		Null	14:12:12.121	14:12:12.121
38	open62541-based OPC UA App...	NS3|String|a.s6123	ActStatus	0 (InPosition)	Int32	14:15:53.123	14:15:53.123
39	open62541-based OPC UA App...	NS3|String|a.s6124	ActTorque		Null	14:12:12.121	14:12:12.121
40	open62541-based OPC UA App...	NS3|String|a.s6127	CmdTorque		Null	14:12:12.121	14:12:12.121
41	open62541-based OPC UA App...	NS3|String|a.s6130	IsInactive	false	Boolean	14:12:12.121	14:12:12.121
42	open62541-based OPC UA App...	NS3|String|a.s6131	IsReferenced	true	Boolean	14:15:53.123	14:15:53.123
43	open62541-based OPC UA App...	NS3|String|a.s6132	IsRotational	false	Boolean	14:12:12.121	14:12:12.121
44	open62541-based OPC UA App...	NS3|String|a.s6133	IsVirtual	true	Boolean	14:12:12.121	14:12:12.121
45	open62541-based OPC UA App...	NS3|String|a.s6146	ZeroOffset	-27.621841	Double	14:15:53.123	14:15:53.123
46	open62541-based OPC UA App...	NS3|String|a.s6141	ActPos	0	Double	14:15:53.123	14:15:53.123
47	open62541-based OPC UA App...	NS3|String|a.s6142	CmdPos	0	Double	14:15:53.194	14:15:53.194
48	open62541-based OPC UA App...	NS3|String|a.s6145	RemDist	0	Double	14:15:53.195	14:15:53.195
49	open62541-based OPC UA App...	NS3|String|a.s6135	ActPos	0	Double	14:15:53.123	14:15:53.123
50	open62541-based OPC UA App...	NS3|String|a.s6136	CmdPos	0	Double	14:15:53.196	14:15:53.196
51	open62541-based OPC UA App...	NS3|String|a.s6139	RemDist	0	Double	14:15:53.197	14:15:53.197
52	open62541-based OPC UA App...	NS3|String|a.s6149	ActFeedrate		Null	14:12:12.121	14:12:12.121
53	open62541-based OPC UA App...	NS3|String|a.s6152	ActGFunctions	{0,17,40,21,90,...	UInt32	14:16:33.124	14:16:33.124
54	open62541-based OPC UA App...	NS3|String|a.s6153	ActJogIncrement	0	Double	14:16:33.249	14:16:33.249
55	open62541-based OPC UA App...	NS3|String|a.s6158	ActMFunctions	{5,9,27,90}	UInt32	14:16:33.124	14:16:33.124
56	open62541-based OPC UA App...	NS3|String|a.s6156	ActMainProgramFile	C:\CNC4.03\c...	String	14:16:33.124	14:16:33.124
57	open62541-based OPC UA App...	NS3|String|a.s6157	ActMainProgramName	mickey.cnc	String	14:16:33.124	14:16:33.124
58	open62541-based OPC UA App...	NS3|String|a.s6159	ActModalOffsetFunction	54	UInt32	14:16:33.124	14:16:33.124
59	open62541-based OPC UA App...	NS3|String|a.s6160	ActOperationMode	0 (Manual)	Int32	14:16:33.124	14:16:33.124
60	open62541-based OPC UA App...	NS3|String|a.s6161	ActOverride		Null	14:12:12.121	14:12:12.121
61	open62541-based OPC UA App...	NS3|String|a.s6164	ActProgramBlock	{'n18 g01 z0 (s...	String	14:14:27.123	14:14:27.123
62	open62541-based OPC UA App...	NS3|String|a.s6165	ActProgramFile		Null	14:12:12.121	14:12:12.121
63	open62541-based OPC UA App...	NS3|String|a.s6166	ActProgramName		Null	14:12:12.121	14:12:12.121
64	open62541-based OPC UA App...	NS3|String|a.s6167	ActProgramStatus	0 (Stopped)	Int32	14:16:33.124	14:16:33.124
65	open62541-based OPC UA App...	NS3|String|a.s6168	ActStatus	0 (Active)	Int32	14:16:33.124	14:16:33.124
66	open62541-based OPC UA App...	NS3|String|a.s6169	BlockMode	false	Boolean	14:16:33.124	14:16:33.124
67	open62541-based OPC UA App...	NS3|String|a.s6170	CmdFeedrate	1250	Double	14:16:33.124	14:16:33.124

Figura A40 - Exemplo de utilização do UAExpert para ler valores de *nós* do servidor OPC UA.

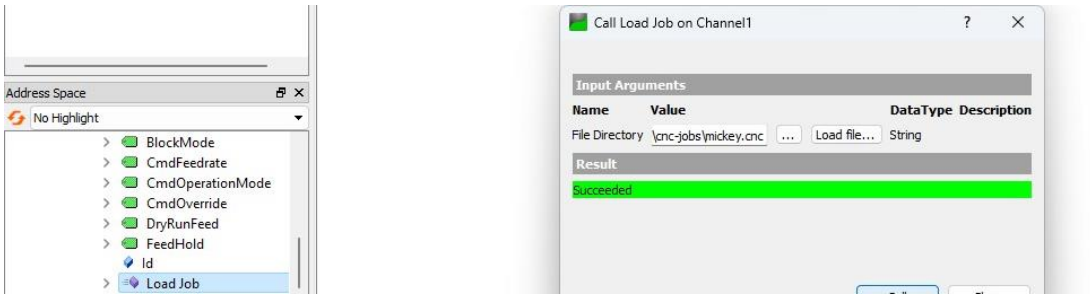


Figura A41 - Execução do *método* Load Job, através do UAExpert.

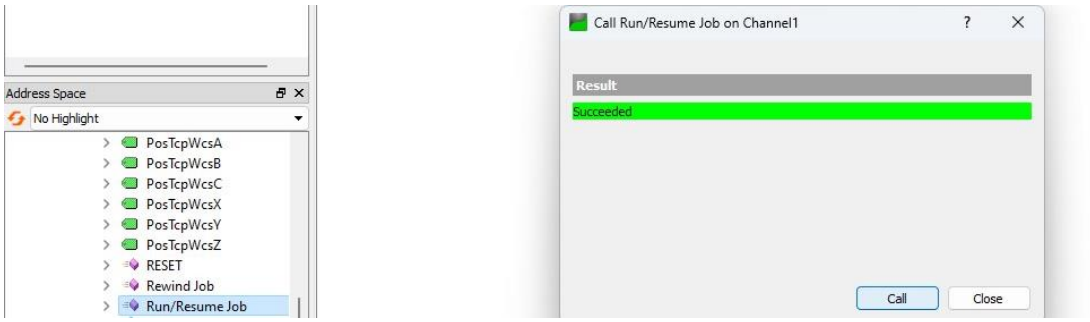


Figura A42 - Execução do *método* Run / Resume Job, através do UAExpert.

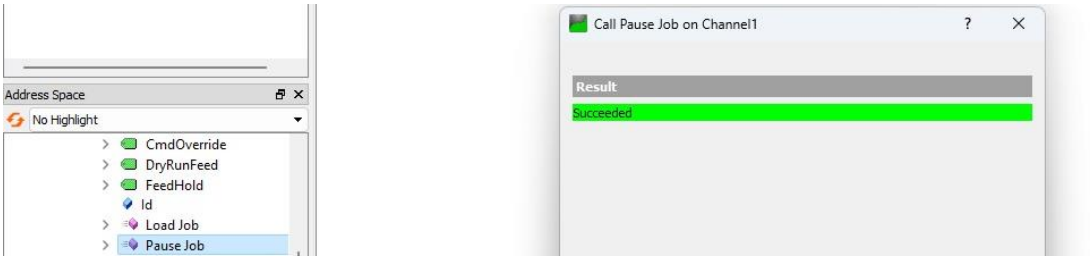


Figura A43 - Execução do *método* Pause Job, através do UAExpert.

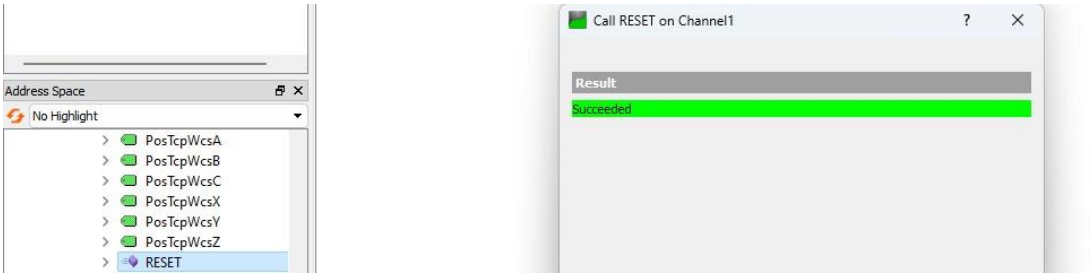


Figura A44 - Execução do *método* RESET, através do UAExpert.

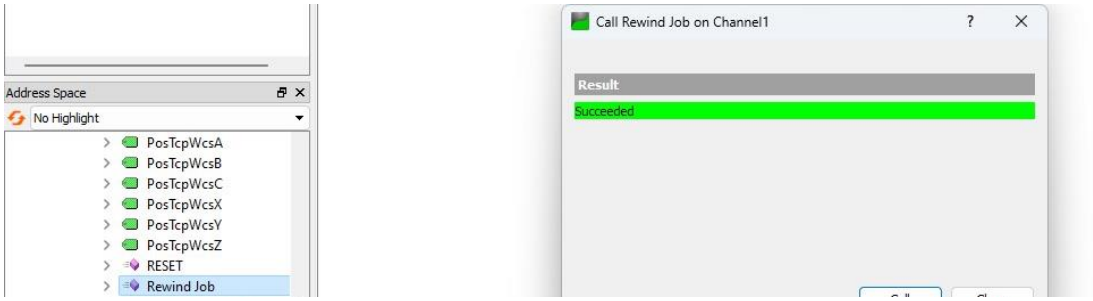


Figura A45 - Execução do *método* Rewind Job, através do UAExpert.

6. Consola de Administração do Ativo

O último passo consiste em criar a Consola de Administração do Ativo (*Asset Administration Shell*, AAS) que representa a Fresadora CNC ProLIGHT1000 presente na ESTG.

Deve-se começar por criar uma pasta, que neste caso se nomeou de *asset-admin-shell-example*, para guardar os ficheiros relacionados com a implementação desta AAS. A seguir, deve-se descarregar os ficheiros .zip que contêm os programas AASX Server Windows e AASX Package Explorer e deve-se os descompactar e guardar na pasta *asset-admin-shell-example*, tal como se vê nas Figuras A46, A47 e A48.

Figura A46 - Conteúdo da pasta *asset-admin-shell-example*.

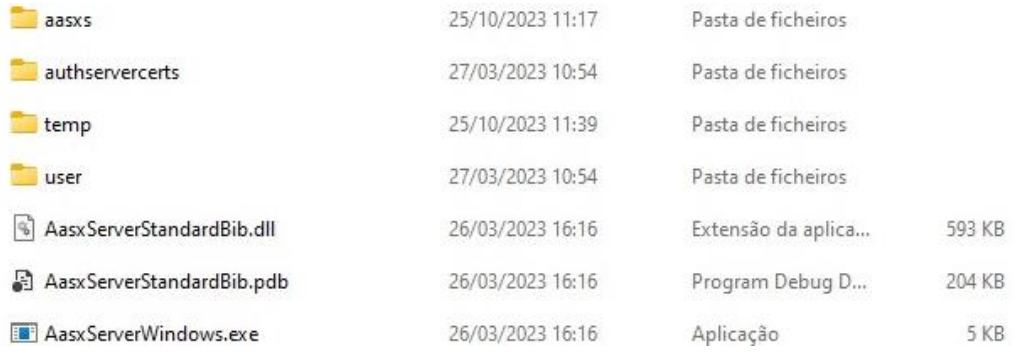


Figura A47 - Excerto do conteúdo da pasta *AasxServerWindows*.

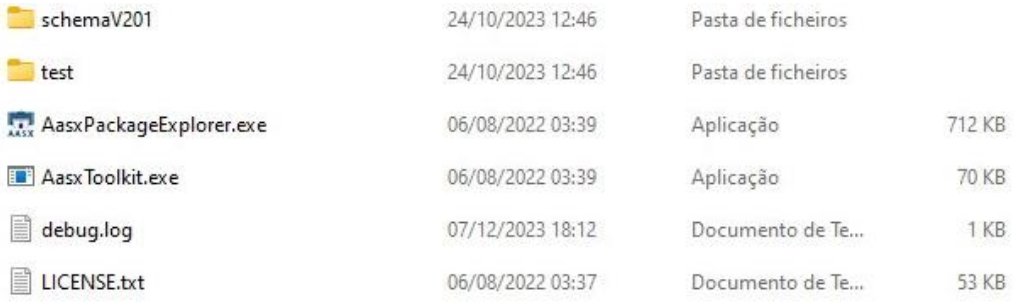


Figura A48 - Excerto do conteúdo da pasta *AasxPackageExplorer*.

Com o botão direito do rato, clique dentro da pasta *asset-admin-shell-example* e, com o botão esquerdo, selecione **Novo > Documento de Texto**. Abra esse ficheiro, escreva:

```
cd AasxServerWindows
AasxServerWindows.exe --rest --no-security --data-path aasxs --opc-
client-rate 500
```

, clique em **Guardar** e feche o Bloco de Notas. Com o botão direito do rato, clique nesse ficheiro e, com o botão esquerdo, selecione **Mudar o Nome**, escreva *START_AasxServer.bat*, pressione **Enter** e clique em **Sim**.

Uma vez executado, este ficheiro seleciona a diretoria *AasxServerWindows* e abre o ficheiro *AasxServerWindows.exe*, aplicando as seguintes configurações:

- inicia um servidor REST;
- especifica a diretoria onde se localiza(m) a(s) AAS(s), que neste caso é a pasta *aasxs*;
- integra um cliente OPC, com uma definida taxa de atualização, que neste caso é 500 ms, sem necessidade de autentificação de segurança.

A seguir, crie o ficheiro *START_CNC.bat*. Com o botão direito do rato, clique dentro da pasta *asset-admin-shell-example* e, com o botão esquerdo, selecione **Novo** > **Documento de Texto**. Abra esse ficheiro, escreva:

```
cd AasxPackageExplorer
.\AasxPackageExplorer.exe -load-without-prompt -aasxrepo ..\aas-
repository.json
```

, clique em **Guardar** e feche o Bloco de Notas. A seguir, com o botão direito do rato, clique nesse ficheiro e, com o botão esquerdo, selecione **Mudar o Nome**, escreva *START_CNC.bat*, pressione **Enter** e clique em **Sim**.

Uma vez executado, este ficheiro seleciona a diretoria da pasta *AasxPackageExplorer* e executa o ficheiro *AasxPackageExplorer.exe*, que abre o repositório de AASs que é definido pelo ficheiro *aas-repository.json*.

O passo seguinte consiste em criar um ficheiro .aasx que descreve a ProLIGHT1000. Par realizar isso, abra a aplicação *AasxPackageExplorer.exe*, que se vê na Figura A48. Nesse programa, com o botão esquerdo do rato, clique em **File** > **New** > **Yes**, para criar um novo ficheiro .aasx. A seguir, com o botão esquerdo, clique em **Workspace** > **Edit**, o que faz este programa trocar do seu Modo de Visualização para o seu Modo de Edição de ficheiros .aasx, passando este a apresentar-se como a Figura A49.

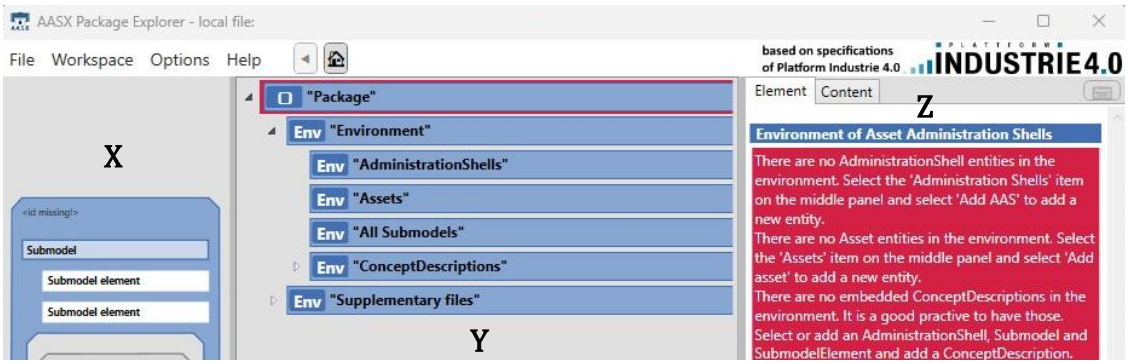


Figura A49 - Janela da aplicação AASX Package Explorer, em modo de edição, a expor um ficheiro .aasx vazio.

Observando a Figura A49, nota-se que a interface gráfica desta aplicação é dividida em 3 secções, nomeadamente:

- **secção X -** secção que permite expor uma representação gráfica do ativo e da AAS;
- **secção Y -** seção que serve para navegar pela estrutura de dados que representam os ativos ou AASs instanciadas, bem como os seus submodelos ou propriedades;
- **secção Z -** consoante o modo da aplicação selecionado, esta secção permite visualizar os submodelos incluídos e o seu conteúdo, ou editar as suas propriedades.

Para guardar este ficheiro .aasx, com o botão esquerdo do rato, clique em **File > Save as**, indique que este será guardado na diretoria "(…)\AasxServerWindows\aasxs" e atribua-lhe um nome, que neste exemplo correspondeu a "AAS_ESTG_CNC_ProLIGHT1000".

Na secção Y, selecione o ícone **Environment** e, na secção Z, clique em **Add Asset**. O ativo (*asset*) instanciado passa a estar visível na seção Y, como mostra a Figura A50. Quando um ativo é instanciado, é lhe atribuído automaticamente um identificador único como, por exemplo, "Asset---3D747924".

Um ativo instanciado requer uma designação. No parâmetro **idShort** do campo **Referable**, introduza, por exemplo, "ESTG_CNC_ProLIGHT1000" e clique em **Take over changes**.

A seguir, selecione o ícone **Environment** e clique em **Add AAS**. A AAS instanciada passa a estar visível na seção Y, como mostra a Figura A50. Quando uma AAS é instanciada, é lhe atribuído automaticamente um identificador único como, por exemplo, "AssetAdministrationShell---4E1B1691".

Uma AAS instanciada requer uma designação. No parâmetro **idShort**, introduza, por exemplo, "AAS_ESTG_CNC_ProLIGHT1000" e clique em **Take over changes**.

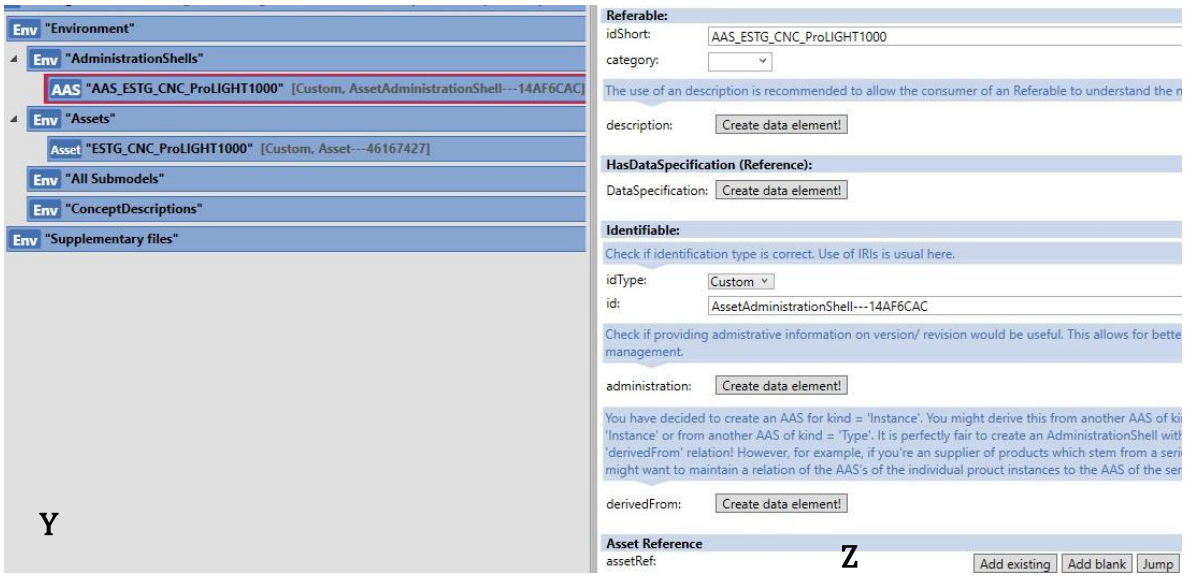


Figura A50 - Excerto de uma janela do AASX Package Explorer, onde se vê um ativo e uma AAS instanciadas.

A seguir, para associar a AAS ao correspondente ativo, na secção Y selecione o ícone da AAS e, na secção Z, no campo **AssetReference**, clique em **Add Existing**, selecione o ativo ESTG_CNC_ProLIGHT1000 e clique em **Select!**.

O próximo passo consiste em adicionar à AAS os submodelos padronizados. Para adicionar um submodelo, na seção Y, com o botão esquerdo do rato, selecione o ícone da AAS instanciada e clique em **Workspace > plugins > New Submodel**. Neste caso, o primeiro submodelo selecionado foi o *ZVEI Digital Nameplate (V1.0)*.

Depois de um submodelo ter sido instanciado, é lhe atribuído uma designação e um identificador único. Para além disso, na secção Z, surge um conjunto de parâmetros que podem ser preenchidos para descrever o ativo, como se vê na Figura A51.

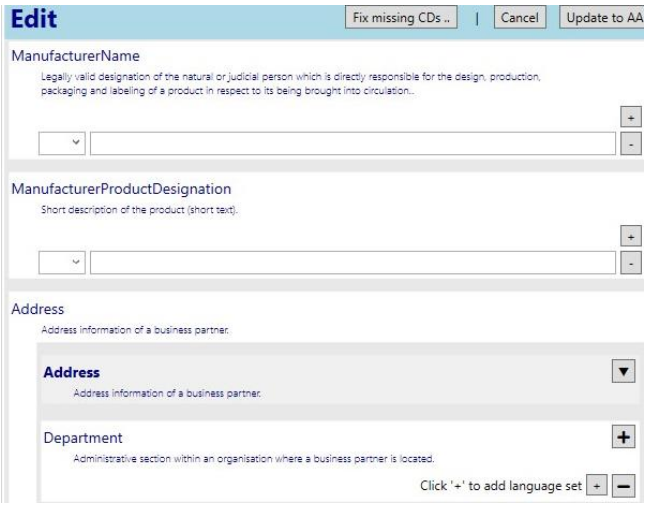


Figura A51 - Secção onde é possível preencher um submodelo, neste caso o *ZVEI Digital Nameplate*.

Deve-se preencher o submodelo *ZVEI Digital Nameplate* com as informações dos meios de contacto do fabricante destes equipamento e obter o resultado que se vê na Figura A52.

Figura A52 - Submodelo *Digital Nameplate* preenchido com informações do fabricante.

A seguir, deve-se importar o submodelo *Technical Data (sheet) model (V1.1)* e preenchê-lo com as informações que descrevem este ativo como, por exemplo, uma designação, o seu número de série ou uma fotografia, e obter o resultado que se vê na Figura A53.

Figura A53 - Submodelo *Technical Data* preenchido com as referencias do equipamento.

A seguir, importe o submodelo *Aasx Plugin Document Shelf | Document (recommended version)*. Através desse submodelo, é possível importar ficheiros e atribuir-lhes um conjunto de *propriedades* que os descrevam.

Para preencher esse submodelo, na secção Y, clique no ícone **Document Shelf** e, de seguida, na secção Z, clique em **Add Document**. Neste caso, foram importados os ficheiros .pdf que contêm os manuais de utilização da ProLIGHT1000 e do EdingCNC, e o esquema dos sistemas de controlo desse equipamento e obteve-se o resultado visível na Figura A54.

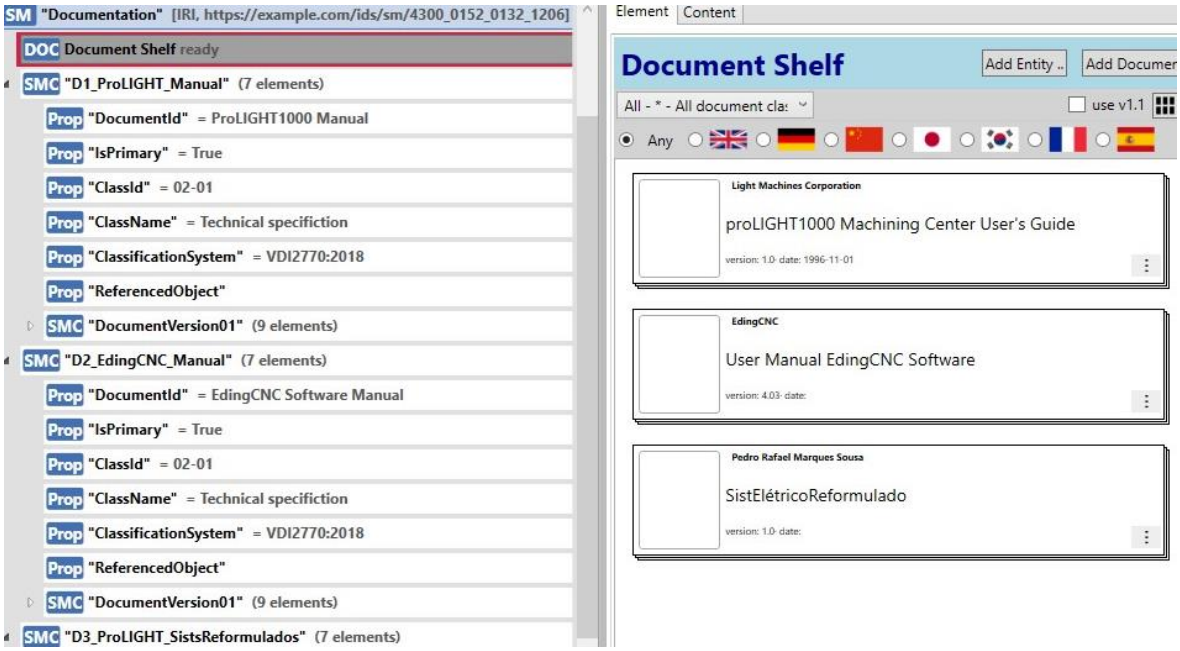


Figura A54 - Submodelo *Documentation* preenchido.

Por fim, deve-se importar o submodelo *Operational Data*. No entanto, tal deve ser feito através de um procedimento diferente, visto que, ao contrário dos submodelos anteriores, na altura em que este texto foi escrito, a aplicação AASX Package Explorer não disponibilizava o *plugin* desse submodelo. Este submodelo serve para conter os dados obtidos durante o tempo de execução de um processo.

Para obter esse submodelo, aceda ao *site* https://admin-shell-io.com/samples/ e descarregue e descompacte o ficheiro *Festo_Demo_Box.zip*. Com a aplicação AASX Package Explorer, abra o ficheiro *00_FestoDemoBox-Module-2.aasx*, que foi extraído, selecione o submodelo **Operational Data** e, no campo **Editing of Entities**, clique em **Buffer > Copy**. Na janela da aplicação AASX Package Explorer que contém o ficheiro .aasx que se está a preencher, selecione, na seção Y, o ícone da AAS e, na secção Z, clique em **Buffer > Paste into**. Os elementos deste submodelo, que também foram copiados, foram apagados.

Deve-se preencher os **qualifiers** deste submodelo (Figura A55), para permitir conectá-lo ao servidor OPC UA.

O **qualifier 1** deve ser preenchido com o endereço URL do servidor OPC UA.

Os **qualifiers 2** e **3** não devem ser preenchidos porque, neste caso, optou-se por não utilizar uma autentificação de segurança.

O **qualifier 4** deve ser preenchido com o *index* do *namespace* do *address space* modelado.

O **qualifier 5** é um parâmetro obrigatório que indica uma diretoria, no servidor OPC UA, para aceder aos *nós* a ler. Neste caso, optou-se por introduzir "a." e o motivo é explicado de seguida.

Neste submodelo, adicione também novas *propriedades* que correspondem aos *nós* do servidor OPC UA, em que se escreveu valores de parâmetros do EdingCNC. Para o fazer, clique em **Submodel Element > Add Property**. Cada *property* adicionada corresponde a um *nó* do servidor OPC UA, cujo valor será lido por esta AAS.

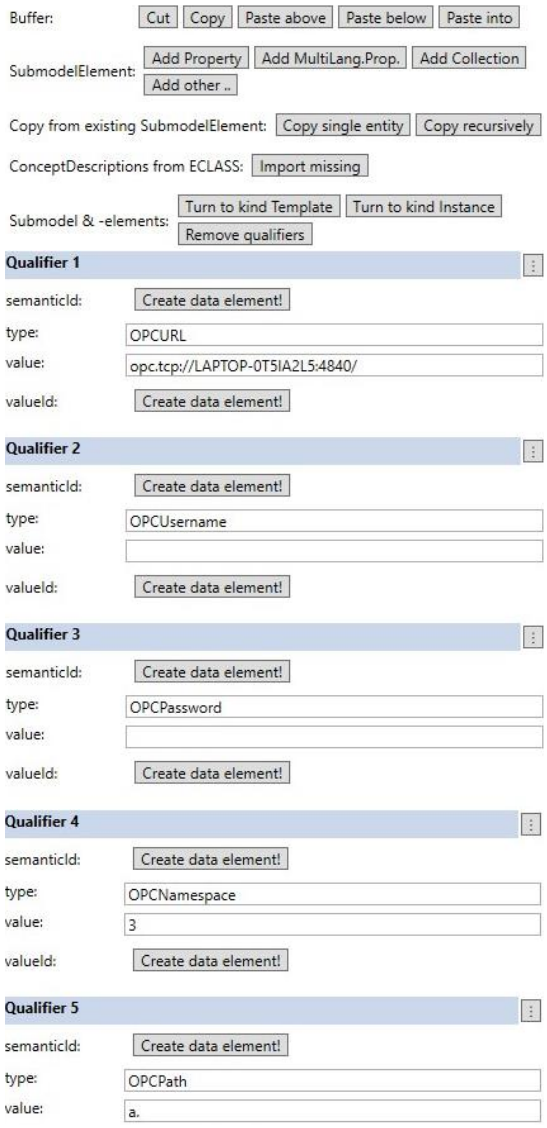


Figura A55 - Submodelo *Operational Data*.

Como se vê na Figura A56, cada *property* requer uma designação (**idShort**) e uma descrição. O termo atribuído a este **idShort** deve começar sempre por uma letra, o que é uma limitação. Por essa razão, os *NodeIds* dos *nós* que compõem o *address space* modelado, que inicialmente estavam associados a um valor `int` como, por exemplo, "6051", foram alterados para serem identificados por uma `string` como "a.s6051". Optou-se por atribuir a designação "a.s", acrónimo de *administration shell*.

Este submodelo une a `string` "a.", que foi introduzida no **qualifier 5** *OPCPath*, ao **idShort** de cada uma das sua *properties*, formando novas `strings` utilizadas para aceder aos respetivos *nós* do servidor OPC UA. Assim, para cada *nó* que compõe o *address space* modelado, deve-se adicionar uma *property* a este submodelo, o que permitiu ler todos esses valores na AAS.

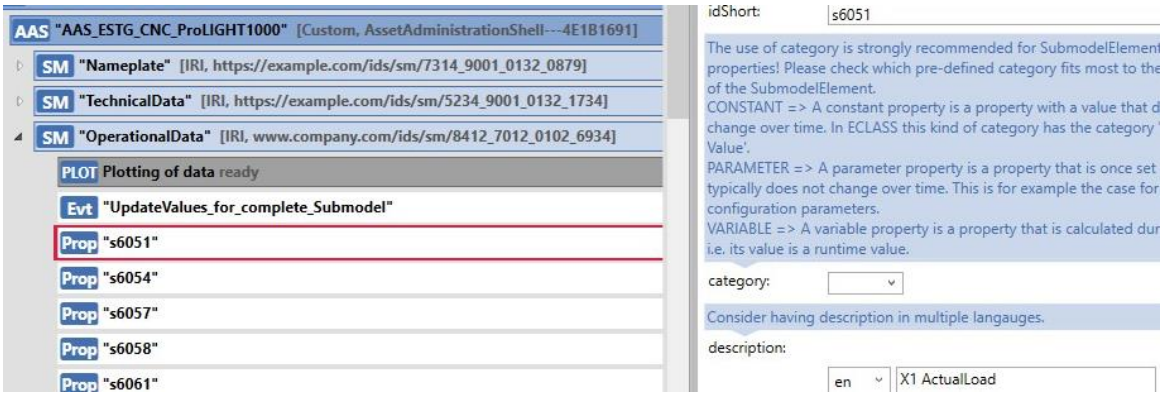


Figura A56 - *Properties* adicionadas ao submodelo *Operational Data* da AAS.

Por fim, deve-se criar o ficheiro *aas-repository.json*, que indica quais são as AASs que compõem um determinado repositório de AASs. Para o fazer, clique, com o botão direito do rato, dentro da pasta *asset-admin-shell-example* e, com o botão esquerdo, selecione **Novo** > **Documento de Texto**. Abra esse ficheiro, escreva:

```
{
  "Header": "AAS Example via HTTP",
  "filemaps": [
    {
      "AssetIds": [
        "Asset---5C66D8E1"
      ],
      "AasIds": [
        "AssetAdministrationShell---75D5CC87"
      ],
      "SubmodelIds": [],
      "description": "",
      "tag": "UpdateValues",
      "code": "Asset---5C66D8E1",
      "Options": {
      "LoadResident": false,
        "StayConnected": true,
        "UpdatePeriod": 500
      },
      "Location": "http://localhost:51310/server/getaasx/0"
    }
  ]
}
```

, clique em **Guardar** e feche o Bloco de Notas. Com o botão direito do rato, clique nesse ficheiro e, com o botão esquerdo, selecione **Mudar o Nome**, escreva *aas-repository.json*, pressione **Enter** e clique em **Sim**.

7. Testes ao funcionamento do sistema

Para testar o funcionamento do sistema desenvolvido, ou seja, para testar a interação entre o servidor OPC UA e a AAS, deve realizar-se o procedimento descrito abaixo.

Primeiro abra a aplicação EdingCNC e, de seguida, inicie a aplicação do servidor OPC UA. Através da interface gráfica do EdingCNC ou dos *métodos* incluído no *address space*, deve-se importar e iniciar um programa de maquinação. O servidor OPC UA passa a ler os parâmetros do EdingCNC, o que se pode verificar recorrendo a um cliente OPC UA como, por exemplo, o UAExpert.

A seguir, na pasta *asset-admin-shell-example*, com o botão esquerdo do rato, faça duplo clique no ficheiro *START_AasxServer.bat* para o executar. O servidor AASX Server é iniciado e passa a ler continuamente os valores dos *nós* do servidor OPC UA, como mostra a Figura A57.

Finalmente, com o botão esquerdo do rato, faça duplo clique no ficheiro *START_CNC.bat* para o executar, o que faz abrir a aplicação AASX Package Explorer e carregar o repositório definido pelo ficheiro *aas-repository.json*. O resultado deve ser semelhante àquele que se vê nas Figuras A57 e A58.

Enquanto que os valores dos *nós* do servidor OPC UA são lidos continuamente pelo servidor AASX Server, para os atualizar recorrendo à interface gráfica do AASX Package Explorer é necessário fazer duplo clique no ícone do repositório de AASs (Figura A58).

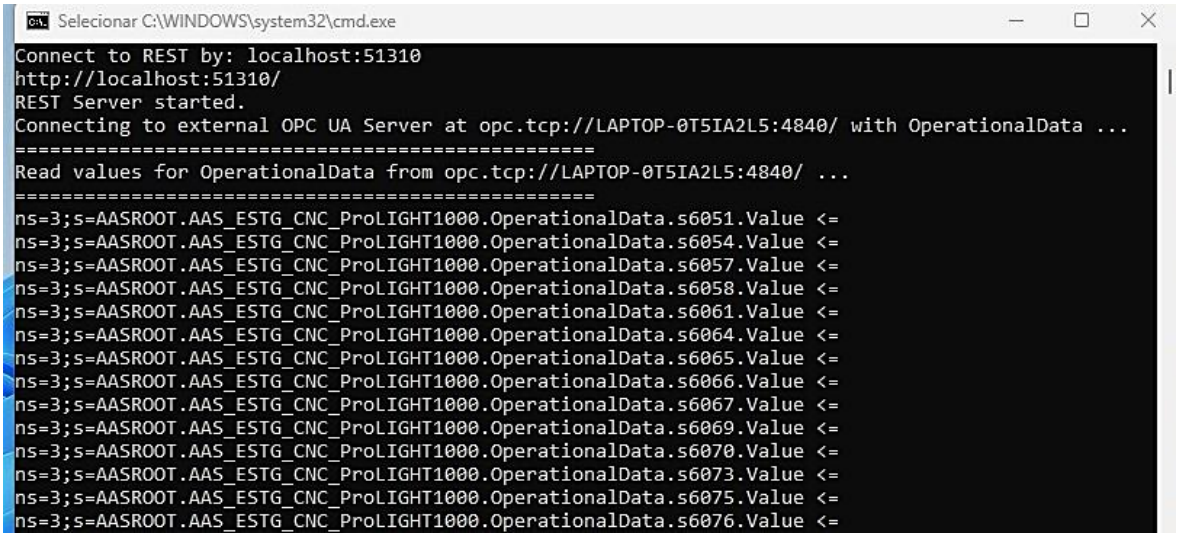


Figura A57 - Servidor AASX Server a ler continuamente os valores de *nós* do servidor OPC UA.

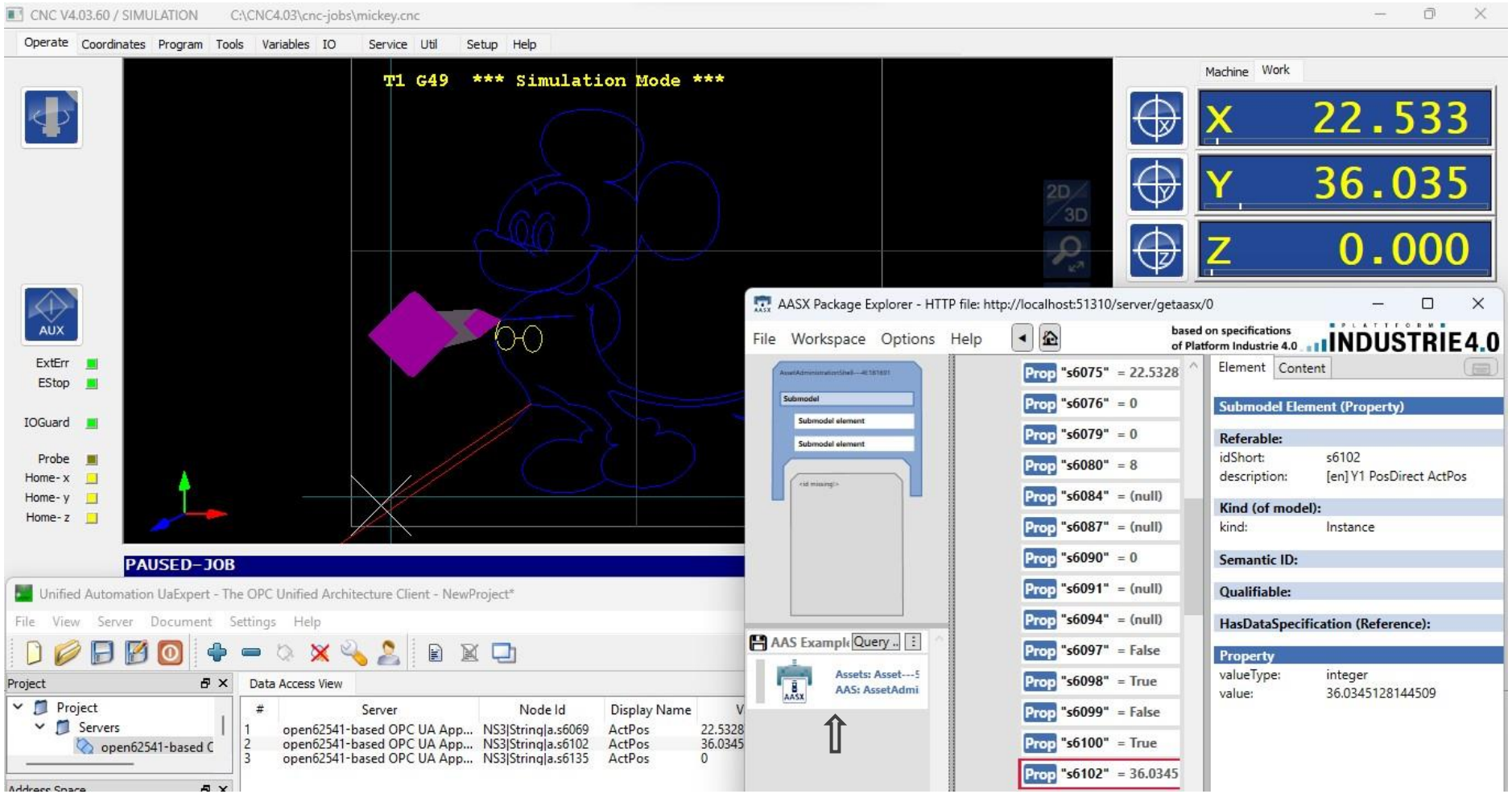
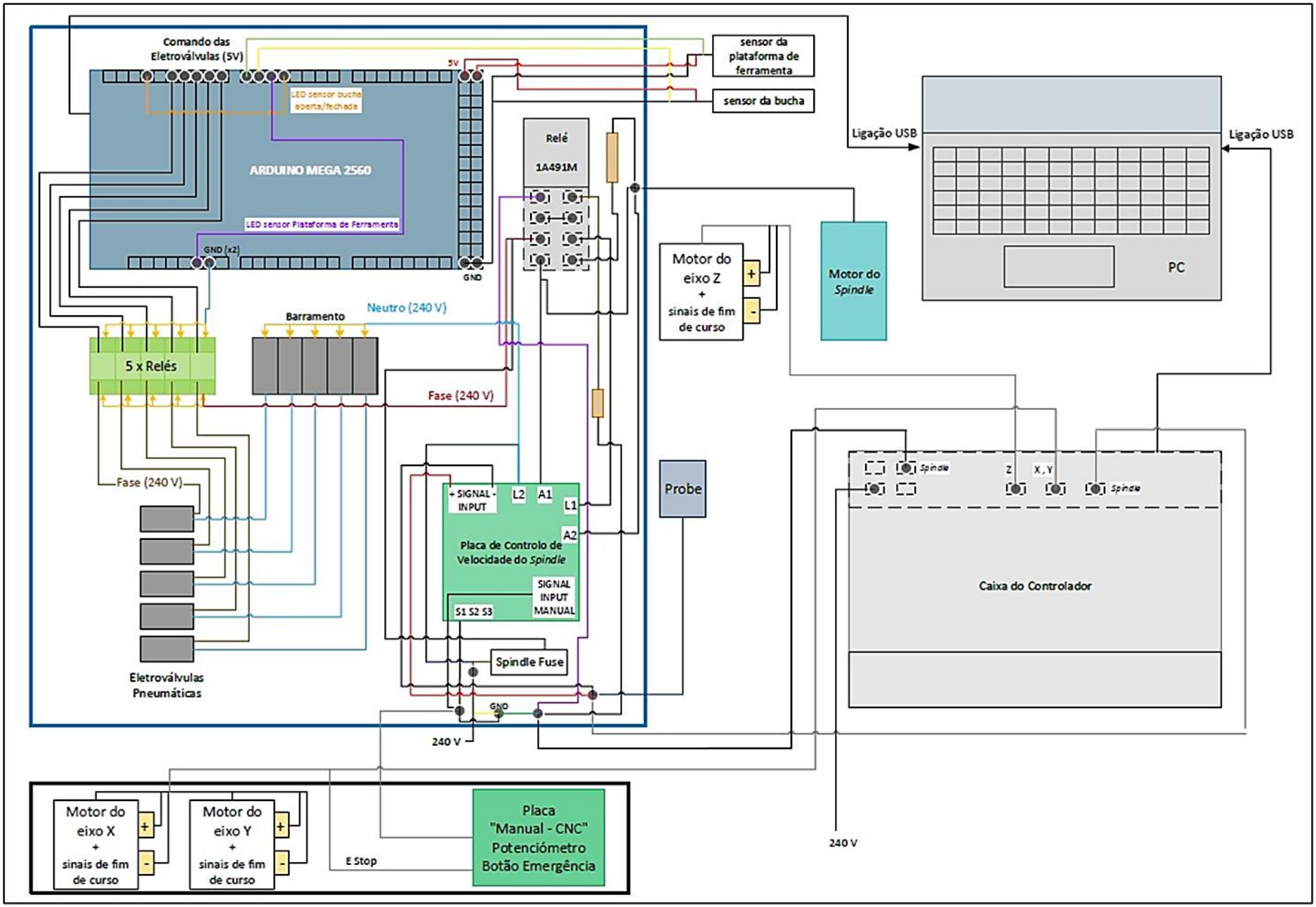


Figura A58 - Clientes OPC UA (UAExpert e AAS) conectados ao servidor OPC UA que, por sua vez, está conectado ao CNCServer do EdingCNC.

Anexo B - Representação dos sistemas de controlo da ProLIGHT1000 presente na ESTG.

(*Na página seguinte*)


Comando das Eletroválvulas (5V)
5V
LED sensor bucha aberta/fechada
ARDUINO MEGA 2560
LED sensor Plataforma de Ferramenta
GND (x2)
GND
sensor da plataforma de ferramenta
sensor da bucha
Relé
1A491M
Ligação USB
Ligação USB
PC
Motor do eixo Z + sinais de fim de curso
Motor do Spindle
Barramento
Neutro (240 V)
5 x Relés
Fase (240 V)
Fase (240 V)
+ SIGNAL - INPUT
L2
A1
L1
A2
Placa de Controlo de Velocidade do Spindle
SIGNAL INPUT MANUAL
S1 S2 S3
Probe
Spindle
Z
X , Y
Spindle
Caixa do Controlador
Spindle Fuse
GND
240 V
Eletroválvulas Pneumáticas
Motor do eixo X + sinais de fim de curso
Motor do eixo Y + sinais de fim de curso
E Stop
Placa "Manual - CNC" Potenciómetro Botão Emergência
240 V

Anexo C - Descrição dos *nós* do *address space* modelado.

CncAxis X		
BrowseName	**DataType**	**Value**
ActChannel	NodeId	3 , "a.s5009"
ActLoad	Null	- - - - - - - - - - - - - - - - - - - -
ActPower	Null	- - - - - - - - - - - - - - - - - - - -
ActStatus	Int32	CncGetJointStatus(0)->state
ActTorque	Null	- - - - - - - - - - - - - - - - - - - -
CmdTorque	Null	- - - - - - - - - - - - - - - - - - - -
IsInactive	Boolean	false
IsReferenced	Boolean	CheckJointHomed(0)
IsRotational	Boolean	false
IsVirtual	Boolean	Get_SimulationMode()
PosDirect	Null	- - - - - - - - - - - - - - - - - - - -
PosIndirect	Null	- - - - - - - - - - - - - - - - - - - -
ZeroOffset	Double	Get_MachineOriginXOffset()

CncAxis Y		
BrowseName	**DataType**	**Value**
ActChannel	NodeId	3 , "a.s5009"
ActLoad	Null	- - - - - - - - - - - - - - - - - - - -
ActPower	Null	- - - - - - - - - - - - - - - - - - - -
ActStatus	Int32	CncGetJointStatus(1)->state
ActTorque	Null	- - - - - - - - - - - - - - - - - - - -
CmdTorque	Null	- - - - - - - - - - - - - - - - - - - -
IsInactive	Boolean	false
IsReferenced	Boolean	CheckJointHomed(1)
IsRotational	Boolean	false
IsVirtual	Boolean	Get_SimulationMode()
PosDirect	Null	- - - - - - - - - - - - - - - - - - - -
PosIndirect	Null	- - - - - - - - - - - - - - - - - - - -
ZeroOffset	Double	Get_MachineOriginYOffset()

CncAxis Z		
BrowseName	**DataType**	**Value**
ActChannel	NodeId	3 , "a.s5009"
ActLoad	Null	- - - - - - - - - - - - - - - - - - - -
ActPower	Null	- - - - - - - - - - - - - - - - - - - -
ActStatus	Int32	CncGetJointStatus(2)->state
ActTorque	Null	- - - - - - - - - - - - - - - - - - - -
CmdTorque	Null	- - - - - - - - - - - - - - - - - - - -
IsInactive	Boolean	false
IsReferenced	Boolean	CheckJointHomed(2)
IsRotational	Boolean	false
IsVirtual	Boolean	Get_SimulationMode()
PosDirect	Null	- - - - - - - - - - - - - - - - - - - -
PosIndirect	Null	- - - - - - - - - - - - - - - - - - - -
ZeroOffset	Double	Get_MachineOriginZOffset()

CncSpindle Spindle		
BrowseName	**DataType**	**Value**
ActChannel	NodeId	3 , "a.s5009"
ActGear	UInt32	1
ActLoad	Null	- - - - - - - - - - - - - - - - - - - -
ActOverride	Null	- - - - - - - - - - - - - - - - - - - -
ActPower	Null	- - - - - - - - - - - - - - - - - - - -
ActSpeed	Null	- - - - - - - - - - - - - - - - - - - -
ActStatus	Int32	
ActTorque	Null	- - - - - - - - - - - - - - - - - - - -
ActTurnDirection	Int32	Get_S1_TurnDirection()
AnglePos (Act, Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
CmdGear	UInt32	1
CmdOverride	Double	Get_S1_CmdSpeedOverride()
CmdSpeed	Double	Get_S1_ProgrammedSpeed()
CmdTorque	Null	- - - - - - - - - - - - - - - - - - - -
IsInactive	Boolean	CncGetSpindleStatus()->spindleIsOn
IsVirtual	Boolean	Get_SimulationMode()

CncChannel Channel		
BrowseName	**DataType**	**Value**
ActFeedrate	Null	- - - - - - - - - - - - - - - - - - - -
ActGFunctions	UInt32	Get_ActGFunctions()
ActJogIncrement	Null	- - - - - - - - - - - - - - - - - - - -
ActMFunctions	UInt32	Get_ActMFunctions()
ActMainProgramFile	String	Get_act_main_program_file()
ActMainProgramFileName	String	Get_act_main_program_file()
ActModalOffsetFunction	UInt32	Get_ActGFunctions()
ActOperationMode	Int32	CncGetState()
ActOverride	Null	- - - - - - - - - - - - - - - - - - - -
ActProgramBlock	String	CurrentILineText()
ActProgramFile	Null	- - - - - - - - - - - - - - - - - - - -
ActProgramName	Null	- - - - - - - - - - - - - - - - - - - -
ActProgramStatus	Int32	CncGetState()
ActStatus	Int32	Get_ChannelActStatus()
BlockMode	Boolean	Get_SingleStepMode()
CmdFeedrate	Double	Get_S1_ProgrammedFeed()
CmdOperationMode	Int32	CncGetState()
CmdOverride	Double	Get_S1_CmdFeedOverride()
DryRunFeed	Double	- - - - - - - - - - - - - - - - - - - -
FeedHold	Boolean	- - - - - - - - - - - - - - - - - - - -
Id	UInt32	1
PosTcpBcsA (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsA (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsB (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsB (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsC (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsC (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsX (Act)	Double	Get_tcp_bcs_c_actpos()
PosTcpBcsX (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsY (Act)	Double	Get_tcp_bcs_y_actpos()
PosTcpBcsY (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpBcsZ (Act)	Double	Get_tcp_bcs_z_actpos()

PosTcpBcsZ (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsA (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsA (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsB (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsB (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsC (Act)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsC (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsX (Act)	Double	Get_tcp_wcs_x_actpos()
PosTcpWcsX (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsY (Act)	Double	Get_tcp_wcs_y_actpos()
PosTcpWcsY (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
PosTcpWcsZ (Act)	Double	Get_tcp_wcs_z_actpos()
PosTcpWcsZ (Cmd & RemD)	Null	- - - - - - - - - - - - - - - - - - - -
ToolId	UInt32	Get_ToolIDinSpindle()