INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




Aplicação de Sistema Ciberfísico em Design Aberto: Integração de Sistemas de Controlo Versáteis e Abertos

JOÃO NELSON ALVES DA SILVA
outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

APLICAÇÃO DE SISTEMA CIBERFÍSICO EM DESIGN ABERTO: INTEGRAÇÃO DE SISTEMAS DE CONTROLO VERSÁTEIS E ABERTOS

João Nelson Alves da Silva

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

Instituto Superior de Engenharia do Porto

APLICAÇÃO DE SISTEMA CIBERFÍSICO EM DESIGN ABERTO: INTEGRAÇÃO DE SISTEMAS DE CONTROLO VERSÁTEIS E ABERTOS

João Nelson Alves da Silva

1171847

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação do Doutor Hélio Cristiano Gomes Alves de Castro e coorientação do Doutor João Augusto de Sousa Bastos.

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

JÚRI

Presidente

Doutor António Manuel Pereira da Silva Amaral
Professor Adjunto, Instituto Superior de Engenharia do Porto

Orientador

Doutor Hélio Cristiano Gomes Alves de Castro
Professor Adjunto, Instituto Superior de Engenharia do Porto

Vogal

Doutor Luís Ferreira
Professor Adjunto, Politécnico do Cávado e do Ave

página propositadamente em branco

AGRADECIMENTOS

Ao longo da execução deste trabalho, muitos foram os que me ofereceram apoio e motivação indispensáveis. Assim, expresso o meu agradecimento a:

Ao Prof. Dr. Hélio Castro e ao Prof. Dr. João Bastos, os meus orientadores, agradeço pela orientação e disponibilidade constante ao longo deste trabalho.

À Liliana, minha esposa e companheira, cuja força e determinação foram a minha base. Desde os primeiros passos até a conclusão deste trabalho, a sua presença foi inabalável e indispensável, sendo a verdadeira âncora nos momentos mais desafiantes. Sem ela, este caminho não teria sido possível.

Ao meu pai e à minha mãe, agradeço pela educação, compreensão e amor incondicional, bem como pelo constante apoio.

Ao Simão, meu irmão, pelo sua inestimável suporte e compreensão ao longo de todo este percurso.

A todos aqueles que, de forma direta ou indireta, influenciaram e apoiaram o percurso e materialização desta dissertação, expresso o meu profundo reconhecimento e agradecimento.

página propositadamente em branco

RESUMO

Neste trabalho, partindo de uma arquitetura base de um sistema ciberfísico (CPS) previamente estabelecida, *Open Science Lab for Manufacturing* (OSLab4Man), procura-se aprimorar e adaptar a estrutura ao considerar as necessidades e requisitos específicos discutidos anteriormente. Este empreendimento encontra-se profundamente enraizado no paradigma do Open Design, garantindo que a arquitetura emergente não apenas segue os princípios do design aberto, mas também está alinhada com os critérios estabelecidos.

Após uma revisão bibliográfica do estado da arte no domínio dos sistemas ciberfísicos e do Open Design, é proposta uma arquitetura mais detalhada e adaptada. Esta nova estrutura enfatiza a integração e comunicação fluida entre as componentes, promovendo uma maior eficiência e adaptabilidade. Com a arquitetura definida, o trabalho segue com uma implementação prática, utilizando um armazém automático como prova de conceito. Este exemplo serve não apenas para exemplificar a sinergia entre componentes digitais e físicos, mas também para demonstrar a aplicabilidade e robustez da estrutura proposta em cenários reais de produção.

Ao concluir, o trabalho representa uma contribuição para a área, oferecendo uma abordagem inovadora que alia teoria e prática. O resultado é uma proposta arquitetónica sólida, que encarna os princípios do Open Design e oferece soluções tangíveis para os desafios associados à produção aberta e colaborativa.

PALAVRAS-CHAVE

Sistema Ciberfísico; Sistema de Controlo; Protótipo; Open Design.

página propositadamente em branco

ABSTRACT

In this work, drawing from a foundational architecture of a previously defined cyber-physical system (CPS), *Open Science Lab for Manufacturing* (OSLab4Man), there's an endeavor to refine and adapt this structure while accounting for the specific needs and requirements previously discussed. This venture is deeply anchored in the Open Design paradigm, ensuring that the emerging architecture not only stays true to open design principles but is also tailored to meet the outlined criteria.

Following a literature review on the state of the art in cyber-physical systems and Open Design, a more detailed and adapted architecture is put forth. This renewed structure emphasizes seamless integration and communication among components, fostering greater efficiency and adaptability. With the architecture set, the work transitions to a practical implementation, using an automated warehouse as its proof of concept. This illustration serves not just to highlight the synergy between digital and physical facets but also to showcase the applicability and sturdiness of the proposed framework in real-world manufacturing scenarios.

In conclusion, this work stands as a contribution to the field, offering an innovative approach that bridges theory and practice. The result is a solid architectural proposal that embodies the Open Design principles and delivers tangible solutions to challenges inherent to open and collaborative manufacturing environments.

KEYWORDS

Cyber-Physical System; Control System; Prototype; Open Design.

página propositadamente em branco

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

página propositadamente em branco

LISTAS DE SIGLAS

Lista de Siglas

AMQP	*Advanced Message Queuing Protocol*
AR	*Augmented Reality*
ASRS	*Automatic Storage and Retrieval System*
BOM	*Bills of Materials*
CoAP	*Constrained Application Protocol*
CPS	*Cyber Physical System*
ERP	*Enterprise Resource Planning*
GUI	*Graphic User Interface*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
ISEP	Instituto Superior de Engenharia do Porto
M2M	*Machine to Machine*
MES	*Manufacturing Execution System*
MQTT	*Message Queuing Telemetry Transport*
MS	*Manufacturing Systems*
P. Porto	Instituto Politécnico do Porto
PLC	*Programmable Logic Controller*
PS	*Production System*
OD	*Open Design*
OI	*Open Innovation*
OS	*Open Source*
OSD	*Open Design Definition*
OSLab4Man	*Open Science Lab for Manufacturing*
R&R	*Reproducibility and Replicability*
UI	*User Interface*
VR	*Virtual Reality*

página propositadamente em branco

1. INTRODUÇÃO

Nesta introdução é apresentada a justificação do trabalho a desenvolver assim como os seus objetivos e como os concretizar. É também delineada a estrutura da dissertação.

1.1. Contextualização

Vivemos num momento em que a fusão do mundo físico com o digital está a redefinir os contornos da nossa realidade. Os sistemas ciberfísicos, resultantes desta confluência, prometem transformações profundas em inúmeros setores, alavancando capacidades de computação, comunicação e controlo para criar soluções inovadoras. No entanto, o desenvolvimento desses sistemas não é apenas uma questão de tecnologia, mas também de abordagem e filosofia. E é aqui que o conceito de "open design" entra em cena.

O "open design", ou design aberto, vai além da mera abertura do código ou das especificações técnicas. Propõe uma abertura do processo de design em si, encorajando a colaboração, a partilha de ideias e a cocriação entre diversas partes interessadas. Esta filosofia, aplicada aos sistemas ciberfísicos, tem o potencial de acelerar a inovação, democratizar a tecnologia e criar soluções mais resilientes e adaptáveis.

Procura-se entender como uma abordagem de design aberto pode influenciar a conceção, desenvolvimento e implementação de sistemas ciberfísicos, bem como os benefícios e desafios associados.

1.2. Objetivos

Os objetivos são os seguintes:

a) Revisão do estado da arte relativo aos sistemas ciberfísicos e ao Open Design;

b) Definir um modelo que agregue as matérias em estudo;

c) Analisar os requisitos de um sistema segundo o modelo definidos e estabelecer uma arquitetura de solução;

d) Implementar um protótipo que sirva de prova de conceito para a solução encontrada;

e) Identificar potenciais lacunas na informação existente e sugerir trabalhos futuros para a sua mitigação.

1.3. Metodologia

Neste estudo, adota-se uma metodologia estruturada para garantir uma análise abrangente e rigorosa do tema em questão. A metodologia segue os seguintes passos:

i. **Revisão Bibliográfica**: Realiza-se uma investigação da literatura existente sobre sistemas ciberfísicos e Open Design, com o intuito de compreender os avanços recentes, técnicas, desafios e soluções.

ii. **Definição do Modelo**: Com base na revisão bibliográfica, define-se um modelo que integra os conceitos de sistemas ciberfísicos e Open Design, tendo em conta as melhores práticas e inovações mais recentes.

iii. **Análise de Requisitos**: Para a conceção de uma arquitetura eficaz e robusta, procede-se a uma análise detalhada dos requisitos do sistema. Esta etapa envolve a identificação das necessidades e especificações fundamentais para que o sistema opere de forma otimizada.

iv. **Desenho da Arquitetura**: Tendo por base os requisitos identificados, estabelece-se uma arquitetura de solução.

v. **Implementação do Protótipo**: Na fase de implementação, cria-se um protótipo funcional como prova de conceito para a arquitetura proposta

vi. **Avaliação e Feedback**: Posterior à implementação, o protótipo é submetido a testes e avaliações em relação à sua eficácia, desempenho e conformidade com os requisitos iniciais. Este processo inclui a recolha de feedback para identificar potenciais melhorias.

vii. **Identificação de Lacunas e Trabalhos Futuros**: Com base na avaliação e feedback, identificam-se lacunas no estudo e propõem-se recomendações para trabalhos futuros.

1.4. Estrutura

Neste relatório existem 5 capítulos principais.

1. **INTRODUÇÃO**: Este capítulo oferece uma visão geral do tema em questão, delineando o escopo do trabalho e apresentando um plano de execução.
2. **REVISÃO BIBLIOGRÁFICA**: No segundo capítulo, realiza-se uma análise da literatura existente, explorando os conceitos e avanços mais recentes relacionados ao tema.
3. **PROPOSTA DE UM SISTEMA CIBERFÍSICO**: Nesta secção, discute-se o enquadramento do tema, definindo-se os requisitos essenciais do sistema e delineando-se a arquitetura da solução proposta.
4. **IMPLEMENTAÇÃO DO PROTÓTIPO**: Este capítulo aborda o projeto e a integração das componentes de hardware e software. Ao longo deste processo, recolhem-se e analisam-se os resultados emergentes.
5. **CONCLUSÃO**: Finalmente, realiza-se uma avaliação crítica dos resultados alcançados, determinando-se se os objetivos iniciais foram atingidos. Adicionalmente, identificam-se áreas de melhoria e delineiam-se sugestões para trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica vai ser dividida em 3 partes. A primeira será focada nos sistemas ciberfísicos, a segunda no conceito de Open Design, e a terceira na interligação entre estes dois domínios.

2.1. Sistemas Ciberfísicos

Um sistema ciberfísico (Cyber Physical System – CPS) é uma integração de processos físicos com processos de computação. Sistemas de computação monitorizam e controlam os processos físicos, usando na sua maioria *loops* de *feedback* onde esses processos físicos afetam a computação e vice-versa. O CPS diz respeito, portanto, à interseção do mundo físico ao cibernético e não à sua união. Não é suficiente compreender separadamente os componentes computacionais e os físicos, é necessário compreender a sua interação [1].

2.1.1. Origens e conceito

Um sistema ciberfísico é um sistema multidisciplinar, que integra processos cibernéticos e físicos ao combinar tecnologias de computação, comunicação e controlo, normalmente em circuito fechado [2][3]. Como mostra na Figura 1, a parte física é constituída pelos recursos de fabrico (máquinas, materiais, humanos e ambiente). A parte digital incorpora as capacidades de gestão de dados, análise e computação. A parte física recolhe dados, deteta e executa, enquanto a parte digital, analisa, processa informação e toma decisões. Através desta conexão a parte física pode afetar os processos digitais e a parte digital pode afetar os processos físicos [4].

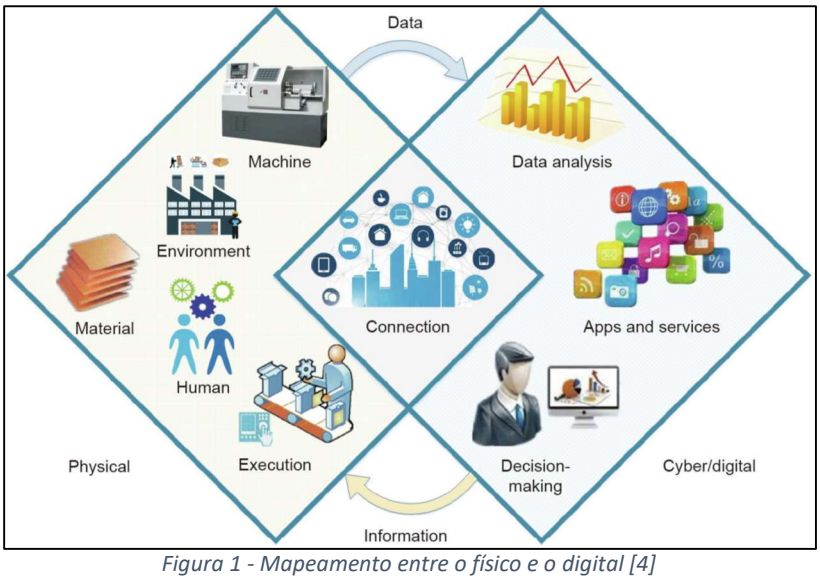


Figura 1 - Mapeamento entre o físico e o digital [4]

Esta integração não implica a simples convergência do ciberespaço com o mundo físico, mas sim uma sinergia de interconexões entre os componentes de ambos os espaços [3].

Os CPS´s ajudam os sistemas mecânicos a perceber o mundo físico. Estas perceções são processadas como dados, são feitos cálculos de forma que se possa informar os sistemas e com isto que eles tomem medidas para alterar os resultados do processo [5].

As aplicações dos CPS´s são muitas com a possibilidade de serem expandidas para todas as áreas da vida humana e não só. Exemplos de aplicações atuais incluem controlo ambiental, sistemas de defesa, produção fabril, geração e conservação de energia, controlo de tráfego, segurança, sistemas e dispositivos médicos de alta confiança e estruturas inteligentes [6].

Na indústria os CPS´s são particularmente importantes. São o elemento diferenciador entre a Indústria 3.0 e Indústria 4.0 com maior relevo. Apresentam características como a autonomia de operação, confiabilidade, ampla distribuição, segurança e boas tolerâncias ao erro. Em conjunto, estas características resultam numa ferramenta que ajuda na superação de desafios presentes em todas as facetas da indústria. Estes desafios podem ir do controlo de custo, segurança individual assim como gestão de qualidade. Para além disso, estes sistemas permitem estender o ciclo de vida de equipamentos existente [5].

Com o constante desenvolvimento de sensores, atuadores, dispositivos interligados em rede e unidades autónomas que combinam um pouco de tudo resultam numa geração de quantidades enormes de informação. A essa informação é dado o nome de *Big Data* [7]. Sendo a gestão desta *Big Data* um desafio por si mesmo, os CPS's podem ser utilizados para ajudar nessa gestão, assim como tornar as máquinas mais flexíveis e adaptáveis autonomamente, em suma, mais inteligentes [7].

2.1.2. Construção de um sistema ciberfísico

Para a construção e execução de CPS´s é necessária a criação de modelos. Estes modelos são complexos e necessitam de usar engenharia de software, controlo, redes de sensores entre outros. Estas disciplinas vão incidir sobre o desafio que é incorporar sistemas computacionais, físicos, focados em humanos e dinâmicos que necessitam de dar respostas em tempo real, organizando-se assim de forma cíclica e dinâmica. O maior desafio é criar portanto uma ligação entre o mundo computacional, que trata dos eventos sequencialmente, com o mundo físico onde os eventos ocorrem simultaneamente [7].

2.1.3. Modelação

Em relação à forma como são abordados estes sistemas, os modelos assim como as ferramentas para CPS têm aumentado. Os modelos são uma das abordagens à criação de CPS's. Outras são, entre outras, a computação a tempo contínuo, máquinas de estados finitos, redes de processo, eventos discretos ou de semântica multiagente [8].

Na Figura 2 [3] pode-se observar a base de funcionamento para as diferentes abordagens de modelação, o *feedback loop*. Com uma parte física e com uma parte digital e uma componente de comunicação entre as duas.

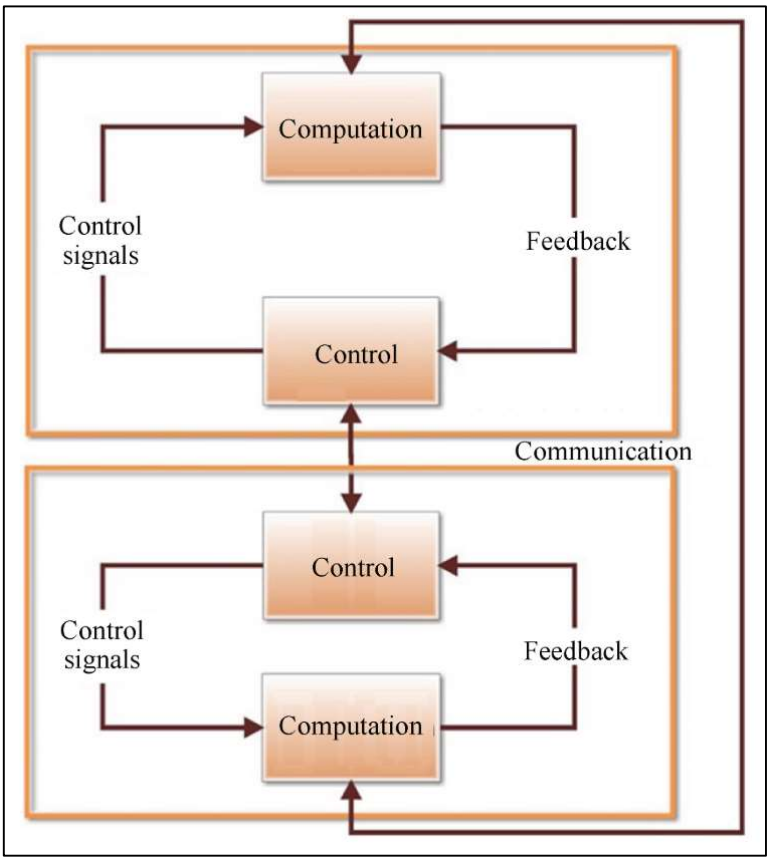


Figura 2 - Modelo em loop fechado [3]

Os sistemas CPS podem ser agrupados conforme as suas características e ordenados conforme a sua complexidade e usabilidade. Esta escala está dividida entre o domínio linear e o domínio dinâmico [7].

2.1.3.1. Domínio Linear

No domínio linear tem-se um espectro de modelos ou definições que é ordenado consoante a adição de novos módulos ou funcionalidades [7]. Na Figura 3 pode se ver que mais à esquerda se encontram os modelos PS/MS aos quais se atribui uma definição mais clássica de CPS, seguido dos modelos CPS^0, que se apresentam como sistemas sem capacidade de aprendizagem, até aos modelos CPS^1 e CPS^2 denominados como inteligentes por incluírem capacidades de aprendizagem [7].

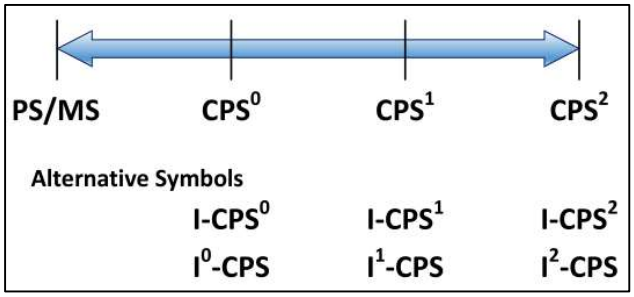


Figura 3 - Domínio linear do espectro dos modelos e definições do CPS [7]

Em cenários de elevada complexidade, estes modelos perdem capacidade de resposta válida [9]. Neste contexto é necessário alargar o domínio linear dos modelos CPS. Com a adição de modelos CPS^n onde a aprendizagem funcionaria com n *loops* conforme a complexidade do cenário. Na Figura 4 tem-se a representação do crescimento linear, consequente exponencial com a passagem do domínio linear para o dinâmico e por fim passando para um modelo "quântico".

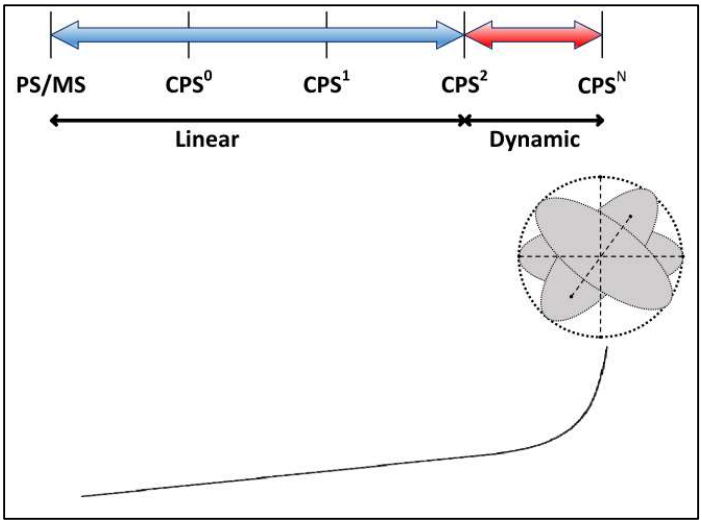


Figura 4 - Os modelos lineares e dinâmicos do CPS [7]

2.1.3.1.1. PS/MS

Com os Sistemas de Produção (PS), ou Sistemas de Manufatura, Figura 5 [7], pode se ver esquematizado que o domínio físico é gerido e controlado por vários sistemas de informação como os Planeamento de Recursos Empresariais (ERP), as Listas de Materiais (BOM) e Sistemas de Execução de Fabrico (MES). Estes sistemas de informação são a parte digital do sistema pois fornecem os *feedbacks loops*. No entanto não são considerados CPS pois enquanto os *loops* de mais baixo nível podem ser realizados em tempo real, os de nível superior são realizados offline. Isto é considerado por Putnik e Ferreira [7] como uma limitação.

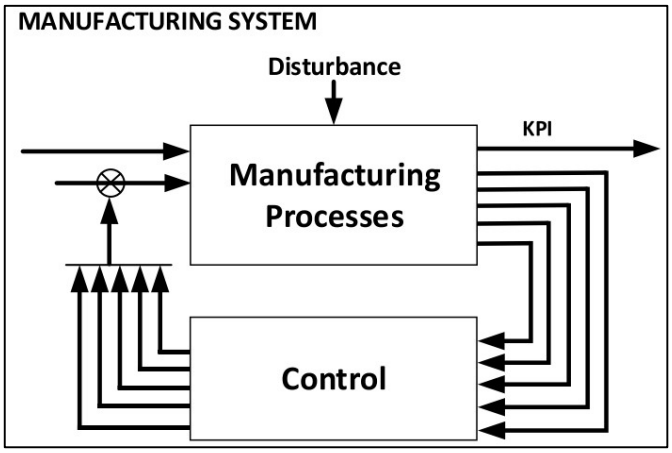


Figura 5 – Arquitetura lógica de um PS/MS clássico ou convencional [7]

2.1.3.1.2. CPS^0

Os CPS clássicos (CPS^0) referem-se a sistemas onde os sistemas físicos os e sistemas digitais se relacionam e são controlados de forma ciclica (*feedback loop*), proporcionando adaptabilidade, capacidade de reconfiguração, resposta rápida e robustez. Os sistemas tradicionais de controlo não conseguem responder de forma suficientemente rápida a um sistema, que devido à sua complexidade e /ou escala, toma dimensões onde é necessário flexibilidade, agilidade e dinamismo para lidar com esta constante mudança [10]. Isto quer dizer que a principal diferença entre os sistemas de produção clássicos já referidos e os sistemas com CPS é a capacidade de controlar ciclicamente em tempo real. Esta funcionalidade não é nova, mas adquire estas propriedades de responsividade em tempo real com o recurso a novas tecnologias como a computação em "nuvem", a internet das coisas (*Internet of Things* - IoT), internet rápida, Web 4.0 e outros [7]. Quase todas estas tecnologias de computação requerem, com poucas exceções, o uso de serviços externos enquanto os *loops* de controlo de mais baixo nível ainda se encontram incorporados nos módulos de controlo "tradicionais" PS/MS. Por esta razão, a parte digital específica do CPS é separada. Na Figura 6 essa parte é apresentada como "DIGITAL PROCESSES 0 LAYER" (de acordo com a denominação de CPS^0). A parte de controlo tradicional, é a parte física do sistema e apresenta-se como "PHYSICAL PROCESS LAYER" [7].

A "SIMULATION", que reage à entrada (*SYSTEM* INPUT), faz a simulação de modificações aplicadas a todas os CPS, permitindo assim a análise das repercussões das alterações do ambiente (dados do processamento anterior, alterações, perturbações, etc.) no processo de fabrico. As simulações continuadas, são uma necessidade, e advêm da avaliação dos resultados por um período de tempo, que é feita de forma constante, em que cada estado deve ser visto como uma nova iteração,

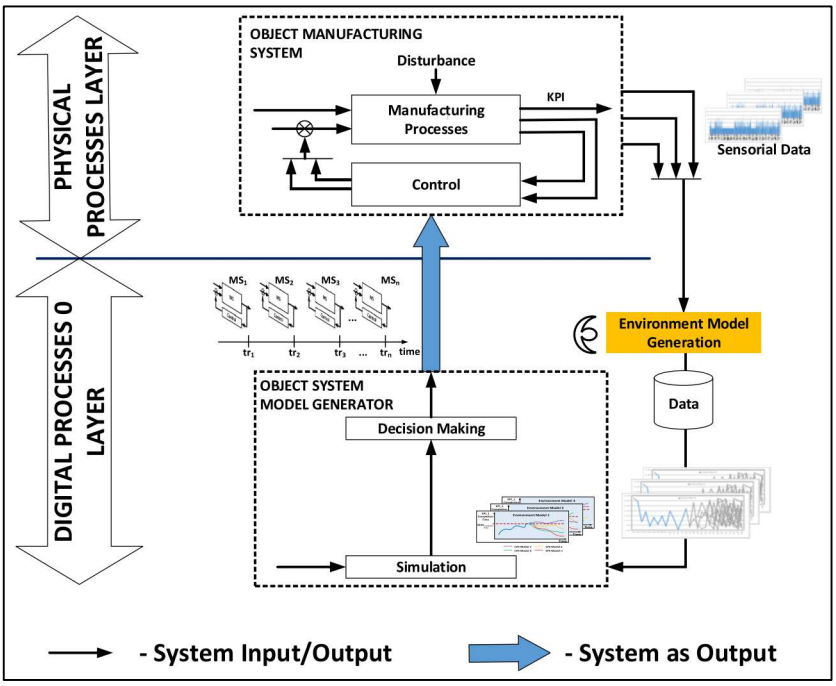


Figura 6 - Arquitetura lógica de um modelo CPS^0 [7]

exterior a todo o sistema e consequentemente aparecendo o estado anterior como *input*. Estes *loops* irão causar alterações na parte física permitindo assim que os sistemas tratem as mudanças continuada e dinamicamente [7].

2.1.3.1.3. CPS[1]

Estes CPS's têm a componente física e componente digital integradas, onde uma aprendizagem em *loop* único pode influenciar ou alterar a componente física. Esta arquitetura lógica é uma prorrogação da arquitetura lógica CPS[0] com a integração do módulo de aprendizagem, como se pode ver na Figura 8. Com este módulo é possível reconfigurar estruturalmente o módulo MS [7]. A aprendizagem está, de forma idêntica à da modelação anterior, a recolher dados passados, presentes e calculados e a descobrir algoritmos avançados de estratégias de aprendizagem de interface de inteligência artificial [11].

A necessidade de aprendizagem contínua (à semelhança dos modelos CPS[0]) prende-se com a análise constante dos resultados ao longo do tempo, em que, novamente, cada estado deve ser visto como uma nova iteração, exterior a todo o sistema e consequentemente aparecendo o estado anterior como *input*, bem como novas condições, desejadas (variações induzidas) ou não (perturbações).

A mudança estrutural é proporcionada pelos algoritmos de aprendizagem, dentro do sub-módulo "LEARNING". O novo módulo de controlo, que irá substituir o antigo, é gerado através do processo de aprendizagem que produz o novo módulo de controlo como o representado na Figura 7.

2.1.3.1.4. CPS[2]

Sistemas Ciberfísicos com aprendizagem de *loop* duplo ou CPS[2], onde de forma a controlar mais eficazmente a parte física, modifica-se a parte digital também. Com este modelo tem-se uma nova parte digital que tem capacidade de aprender com o módulo de aprendizagem do modelo CPS[1].

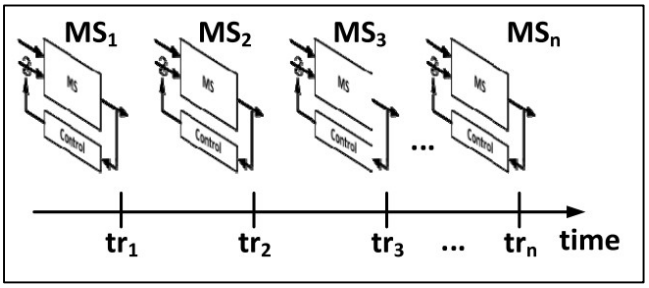


Figura 7 - Aprendizagem de loop único, juntamente com processos de simulação [7]

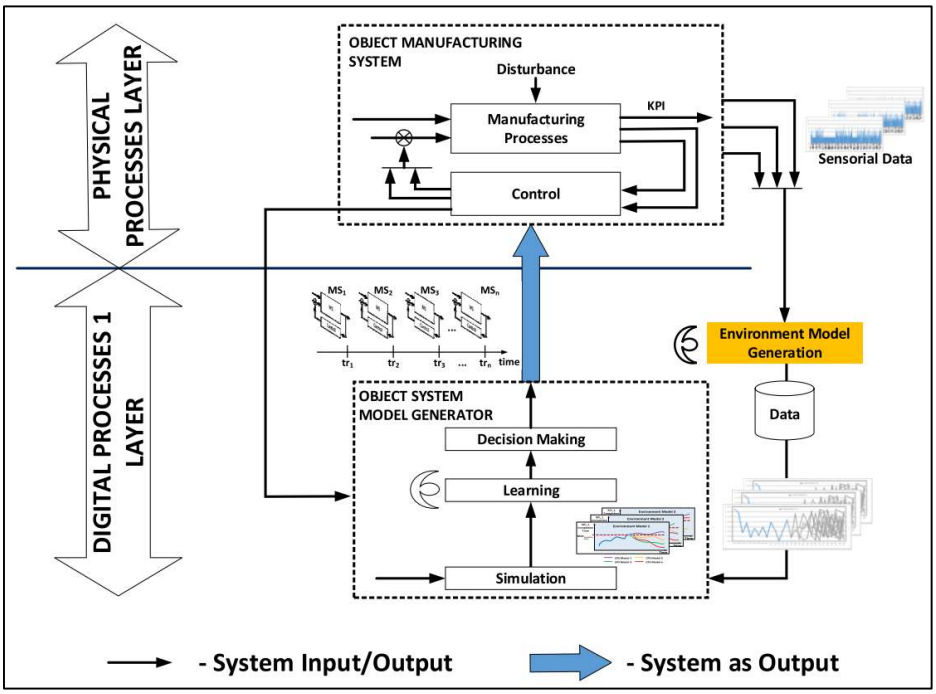


Figura 8 - Arquitetura lógica de um modelo CPS[1] [7]

Este processo de aprendizagem sobre outro processo de aprendizagem é chamado na literatura de *second loop learning* [12]. Como mostra na Figura 9, este segundo módulo de aprendizagem está incorporado no módulo denominado "COMPUTACIONAL SYSTEM MODEL GENERATOR". O novo módulo de computação, juntamente com o módulo de computação descrito na arquitetura lógica do CPS[1] formam um mais complexo terceiro módulo da parte digital do CPS [7].

Desta forma, prevê-se que os dados da parte física dos CPS (OBJECT MANUFACTURING SYSTEM) afete o próprio processo de aprendizagem [7]. A necessidade de aprendizagem contínua (como nos modelos CPS[1]) provém da avaliação permanente da aprendizagem dentro do primeiro ciclo. A mudança estrutural da parte digital do CPS é fornecida pelos algoritmos de aprendizagem, dentro do sub-módulo "LEARNING" dentro do "COMPUTACIONAL SYSTEM MODEL GENERATOR".

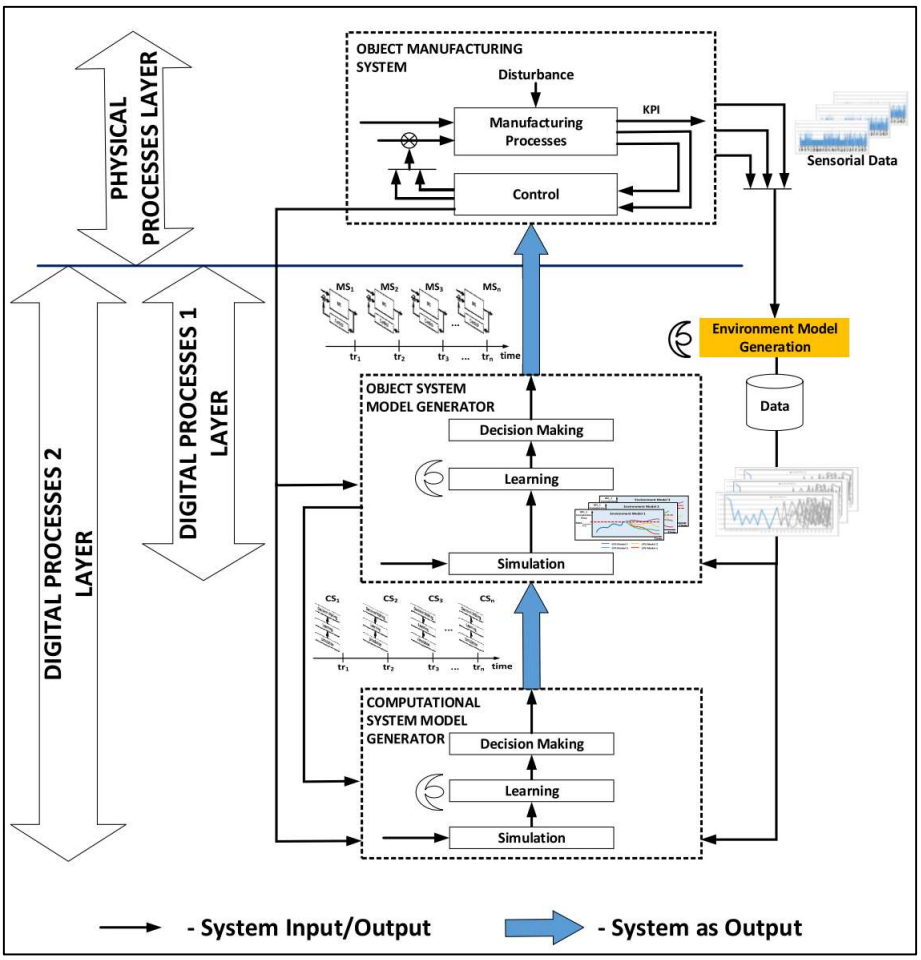


Figura 9 - Arquitetura lógica de um modelo CPS[2] [7]

O novo "OBJECT SYSTEM MODEL GENERATOR" irá substituir o anterior. Isto acontece sequencialmente como ilustrado na Figura 10 [7].

Este modelo satisfaz as definições, visão e requisitos existentes para os CPS. Com este modelo temos os *loops* de *feedback* onde os processos físicos afetam a computação, e de forma inversa, os processos digitais (computação) afetam os processos físicos [7] [13].

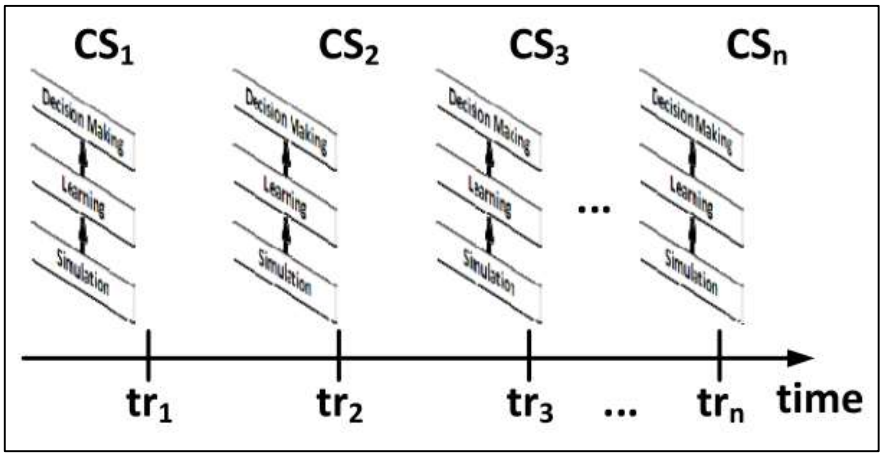


Figura 10 - Aprendizagem com um segundo loop [7]

2.1.3.2. Domínio dinâmico

O domínio dinâmico é considerando os sistemas industriais presentes. Segundo Putnik e Ferreira [7] os modelos no domínio dinâmico distribuem-se de forma síncrona, como se pode ver na Figura 11, da seguinte forma: CPS (sistemas ciberfísicos clássicos), CPS-AMS (Augmented Manufacturing Systems CPS), M-CPS (Mixed CPS), CPS-VMS (Virtual Manufacturing Systems CPS) e I-CPS (Intelligent).

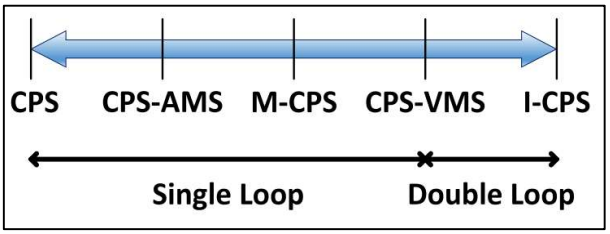


Figura 11 - Domínio dinâmico do espectro dos modelos e definições do CPS [7]

Dentro deste espectro tem-se a distinção entre sistemas de *loop* único ou duplo *loop*.

De uma forma geral, os CPS´s que existem são baseados nos modelos clássicos (CPS^0 e CPS^1). De uma forma simplificada são a parte física e seus sistemas de controlo que modificam o módulo de controlo do sistema de fabrico [7]. O CPS-AMS dá uso a tecnologias como a realidade aumentada para explorar simulações usando dados do ambiente em tempo real com origem em sensores integrados para o efeito [7]. O M-CPS é, provavelmente, um dos CPS'a mais comuns em uso. Onde a realidade mista (RM) é aplicada junto com a tecnologia e os sistemas de controlo autónomos de forma a gerar correções ao processo[7][14]. O CPS-VMS centra-se na realidade virtual para simular a realidade[7].

A grande limitação destes modelos é serem baseados numa aprendizagem num único *loop* e por isso não conseguem processar a realidade na sua verdadeira grandeza. Para que haja uma verdadeira eficácia tem de haver aprendizagem em *loop* duplo [15].

Os modelos I-CPS são modelos CPS^2 usando o duplo *loop* de aprendizagem já referido, e assim tornando capazes de abordar cenários de maior complexidade de forma mais adequada.

2.1.3.2.1. CPS-AMS

OS CPS-AMS são sistemas que integram a realidade aumentada (AR). Esta tecnologia permite combinar o virtual com o real em tempo real e registar-se em 3D [16][17]. A AR é usada para complementar o mundo real com informações ou objetos virtuais de forma a parecerem coexistirem no mesmo espaço [18].

Em Khalid et al. [18] é feita uma listagem dos componentes chave necessários para que a integração da AR com os CPS's possa existir. Do lado dos CPS's é necessário:

1. Prototipagem virtual
2. Sensores sem fios
3. Dispositivos móveis
4. Rede de comunicações

Do lado da AR é necessário:

1. Representação dos meios de comunicação
2. Mecanismo de *input*
3. Continuo de mecanismo de *output*
4. Localizadores

2.1.3.2.2. M-CPS

No *Mixed* CPS tem se a implementação de sistemas CPS com a realidade mista (MR). Nesta situação vai para além da realidade aumentada do ponto de vista em que pode haver interação entre o utilizador, as componentes digitais e o ambiente circundante [19].

Não existe uma definição universalmente aceite de MR. O significado de MR depende de diferentes contextos de aplicação, tais como um sinónimo de AR, a experiência que pode fazer a transição entre AR e VR, ou um ambiente de simulação com dados do mundo real como entrada [20]. Na Figura 12 pode-se ver o espetro da realidade virtual onde se pode encontrar uma definição de MR.

Nesta revisão é adotada a aplicação de MR como uma experiência interativa que combina real e virtual que são semelhantes a AR.

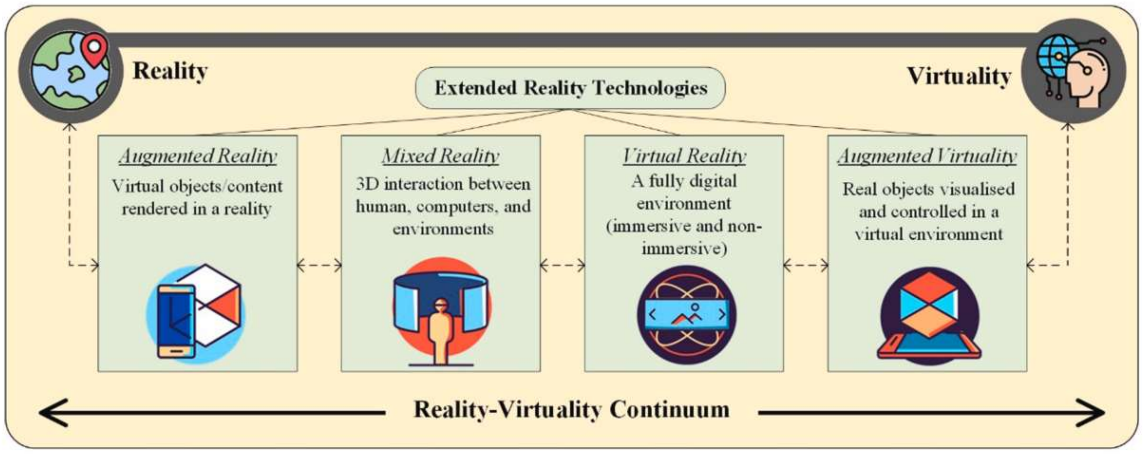


Figura 12 - Espectro da realidade aumentada [20]

2.1.3.2.3. CPS-VMS

Seguindo o espectro da realidade aumentada já visto na Figura 12, tem-se a existência da realidade virtual (VR) [21]. Em Hu et al. [22] é apresentada uma versão deste sistema em que é usada a VR com deteção multifacetada. Esta deteção é multifacetada ao poder ser usada, implantada ou não invasiva (*wearable*, *implantable*, non-invasive (WIN)). A identificação e comunicação com estes equipamentos WIN é feita por radiofrequência (RFID). Os *inputs* e *outputs* de cada equipamento são comunicados entre si e ao mesmo tempo enviados para um servidor que vai aplicar os ciclos de aprendizagem. Dentro de um ambiente de realidade virtual existe a possibilidade de se navegar o espaço 3D e fazer zoom em áreas de interesse específico. Este tipo de navegação permite a verificação de recursos com um grande nível de precisão e detalhe [23].

2.1.3.2.4. I-CPS

Os CPS inteligentes ou I-CPS são modelos que têm uma aprendizagem continuada ao longo do tempo de forma a maximizar a sua eficácia, colaboração e conectividade quer com entidades abstratas como físicas. Estão preparados para serem integrados em domínios de grande escala [7]. Esta capacidade é atributo do uso de aprendizagem de duplo *loop* como já se apresentou nos modelos CPS^2 [15].

2.1.4. Outras arquiteturas

As arquiteturas de CPS não se limitam apenas aos modelos apresentados. É de referir a existência de semânticas, Meta-Arquitetura e Meta-Programação e *Codesign* [18].

2.1.4.1. Semânticas

Em relação às semânticas pode-se ver na Figura 13 uma esquematização de um processo em que informação é retirada de sensores, interpretada pela semântica do sistema e enviada para os agentes conforme a sua localização no sistema [19].

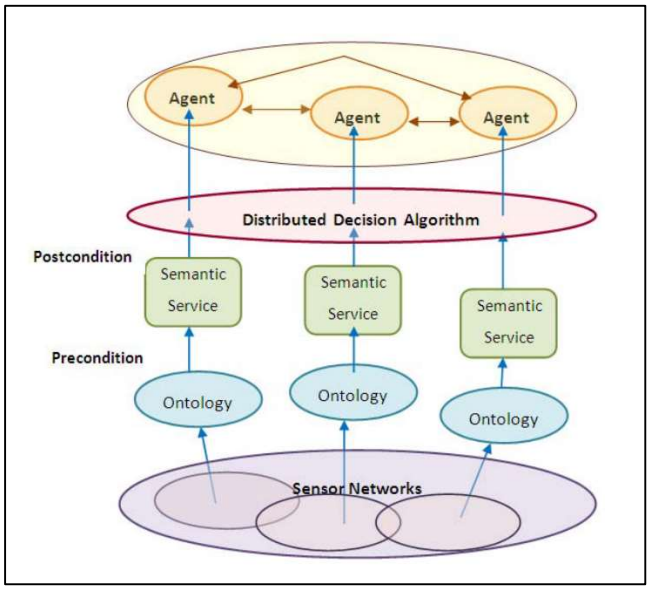


Figura 13 - Enquadramento de agentes semânticos [19]

Bujorianu et al. [24] apresenta um quadro de um modelo de referência para CPS's multiagentes e uma lógica formal para expressar propriedades de segurança em CPS's. No seu enquadramento, os agentes têm mobilidade física contínua e evoluem num ambiente físico incerto, que modeliza de perto o mundo real. A sua estrutura modeliza tanto o controlo humano como o controlo automático, tornando assim o modelo centrado no utilizador. Talcott [25] desenvolve uma semântica baseada em eventos para as CPS's. É classificado diferentes eventos de acordo com as suas características, por exemplo, pontuais versus duradouros e únicos versus correntes. O autor observou que a utilização da semântica baseada em eventos proporciona uma forma natural de especificar componentes de sistemas abertos em termos de interfaces e comportamento observável. Além disso, também fornece uma base para a conceção, monitorização e implementação de CPS's. Bujorianu et al. [26] propõe uma abordagem formal chamada método formal Hibertean para fornecer uma semântica denotativa para os CPS. Combina a semântica denotativa com um modelo algébrico para processos físicos para modelar a causalidade física e a observabilidade [8].

2.1.4.1.1. Meta-Arquitetura e Meta-Programação

Embora existam várias linguagens arquitetónicas e de programação para fazer a modelação das características de componentes digitais ou físicos individuais num CPS, há necessidade de metodologias para especificar e verificar as propriedades dos diferentes componentes do sistema. Hang et al. [27] propõe técnicas que permitem sintetizar os algoritmos dos modelos de CPS com limitações em tempo real. Rajhans et al. [28] apresenta uma metodologia de conceção meta arquitetónica para facilitar a conceção e avaliação de arquiteturas alternativas para CPS, que alarga

as linguagens de especificação arquitetónica existentes, tais como a *Unified Modeling Language* (UML), *Systems Modeling Language* (SysML), e *Abstract Architecture Description Language* (AADL), permitindo a modelação de componentes utilizando métodos formais, tais como processos de estado finito, sistemas de transição etiquetados, e autómatos híbridos. A sua abordagem permite o uso de plugins para realizar análises de comportamento. Este trabalho permite uma estrutura unificada para modelar tanto elementos físicos como digitais num estilo arquitetónico CPS. Dabholkar et al. [28] apresenta uma abordagem de especialização sistemática de *middleware* de uso geral para satisfazer as exigências dos CPS's utilizadas em diferentes domínios. Esta abordagem baseia-se nos princípios do desenvolvimento de software orientado para as características, que emprega uma estrutura algébrica do *middleware* existente. Ao elevar o nível de abstração para o nível de características que o *middleware* oferece, em vez de se concentrar em detalhes ao nível do código fonte, a sua abordagem especializa o *middleware* de uso geral para as necessidades de CPS específicas do domínio.

2.1.4.1.2. Codesign

Goswami et al. [29] discute uma arquitetura para abordar as considerações de criação para facilitar a interação entre as aplicações de controlo que funcionam em unidades de processamento distribuídas espacialmente. A arquitetura de controlo e comunicação que concebida deu origem a aplicações de controlo com capacidade de decidir um calendário de comunicação. No caso de um atraso nessa comunicação, existe flexibilidade que evita a origem de falhas e erros. Miller et al. [30] apresenta uma estrutura de *codesign* para a elaboração de equipamentos ciberfísicos onde os aparelhos físicos estão conectados a modelos em tempo real e em que é simulado o ambiente de comunicação entre os dois. Zhang et al. [13] apresenta o problema do agendamento de tarefas para CPS's cujos comportamentos são regulados por leis de controlo de feedback. Ao codificar a lei de controlo e o algoritmo de agendamento de tarefas, o seu método consegue um desempenho previsível e uma dissipação de energia para o sistema onde múltiplos pêndulos invertidos são controlados por um único processador.

2.2. Open Design

Os benefícios do software de código aberto têm sido reconhecidos ao longo do tempo na indústria. Este tipo de software é concedido a qualquer pessoa para usar, estudar, modificar, e distribuir o código fonte para qualquer finalidade. Estas liberdades permitem os seguintes benefícios: flexibilidade e liberdade, a capacidade de autoaprendizagem, fiabilidade, apoio e responsabilização, estabilidade e manutenção. Estes benefícios levaram a grandes sucessos industriais: por exemplo, o Linux é um sistema operativo no qual dois terços dos servidores Web foram executados em 2017 [31]. No entanto, devido à digitalização global e à disseminação de um custo de acesso à internet eficiente e de baixo custo, este fenómeno propagou-se a outros campos industriais. Pode-se citar dados abertos, educação aberta, hardware aberto, entre outros [31].

2.2.1. Definição e contexto

O termo *open design* (OD) tem sido utilizado desde o final dos anos 90. Define-se o *open design* como design ao qual é permitida a sua distribuição gratuita e a sua modificação e derivações pelos seus criadores. O OD utiliza duas alavancas, o poder das massas, a soma de todas as contribuições efetuadas gera um grande desenvolvimento e produção, e também a capacidade de apenas ser necessário melhorar as soluções existente em vez de ter de desenvolver tudo de raiz [31]. O OD situa-se onde a abordagem aberta encontra o design do produto. Contudo, este termo único está mais ou menos intimamente relacionado com múltiplas práticas já existentes. Portanto, para cunhar uma definição de OD, temos primeiro de ser capazes de avaliar a abertura de um projeto de design. Isto permite definir esta noção em relação aos conceitos existentes. Na Figura 14 podemos ver um gráfico com a localização relativa de diversos conceitos num sistema de eixos com processos e planeamentos [32].

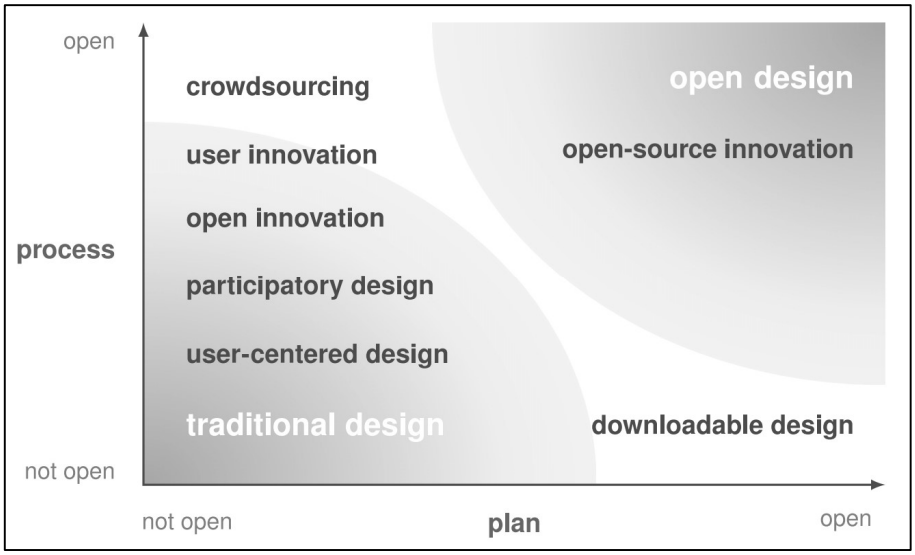


Figura 14 - Open Design e conceitos relacionados [31]

2.2.2. Openness

A *Openness* diz respeito à acessibilidade. É uma qualidade que mede o quão alguma coisa se pode aceder, quer para visualizar, modificar ou utilizar. A capacidade de visualização prende-se com o consumo de informação assim como a sua partilha, a capacidade de modificação conecta-se com a potencial modificação, melhoramento ou adição de conteúdo a algo já existente, e por fim a capacidade de uso de algo que se refere à possibilidade de reutilização com ou sem restrições. Estas são as três operações fundamentais que estão implicadas pela acessibilidade. Da perspetiva da teoria de sistemas, a *openness* relaciona-se com a transparência e a permeabilidade de quaisquer fronteiras naturais ou construídas [32]. Segundo Avital, M. [32] existem três arquétipos de *openness*, ou *Open-X*. Esses arquétipos são o *Open Innovation*, o *Open Source* e o *Open Design.* Outros autores sugerem ainda mais variações como *Open Data*, *Open City*, *Open Government*, *Open Education* e *Open Science* [33]. Na Tabela 1 vê-se a relação que cada arquétipo, avançados por Avital, M. [32], tem com a proposta de valor e impulso, orientação central e principais agentes envolvidos.

Tabela 1 - Os arquétipos do Openness, adaptado de Avital, M. [32]

	Open Innovation	***Open Source***	***Open Design***
Proposta de valor e impulso	Conhecimento distribuído	Desenvolvimento distribuído	Fabrico distribuído
Orientação central	Ver	Modificar	Usar
Principais agentes envolvidos	Organizações	Comunidade de *developers*	Consumidor

2.2.2.1. Open Innovation

Open Innovation (OI) é um conceito relativamente recente, mas rico. Numa revisão feita em 2011 de 150 artigos acerca deste tema foi descoberto que os investigadores usam definições diferentes e aplicadas às suas investigações o que torna difícil a construção de um corpo coerente de conhecimento [34]. Henry Chesbrough em 2003 define *Open Innovation* desta forma "*Open innovation is a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology. Open innovation combines internal and external Ideas into architectures and systems whose requirements are defined by a business model."* [35]. Por outras palavras, é expectável um melhor resultado entre uma empresa e o seu meio circundante caso elas permitam um mais livre fluxo de ideias.

OI é uma área de cada vez mais interesse da parte de investigadores e profissionais. Esta tendência é clara ao se ver o gráfico, Figura 15, onde se tem o número de artigos feitos por ano com o tema de OI [36].

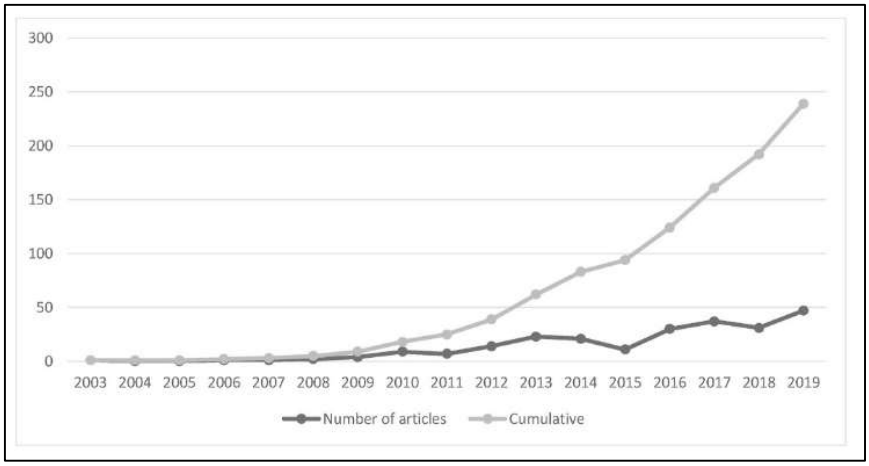


Figura 15 - Número de artigos em Open Innovation por ano [36]

Chiaroni, Chiesa e Frattini [37] são propostas 3 eixos ao longo dos quais se desenvolve o enquadramento teórico de OI. Esses eixos são as dimensões de OI, as alavancas da gestão, e os processos de adoção de OI. Na Figura 16 tem-se uma representação destes eixos.

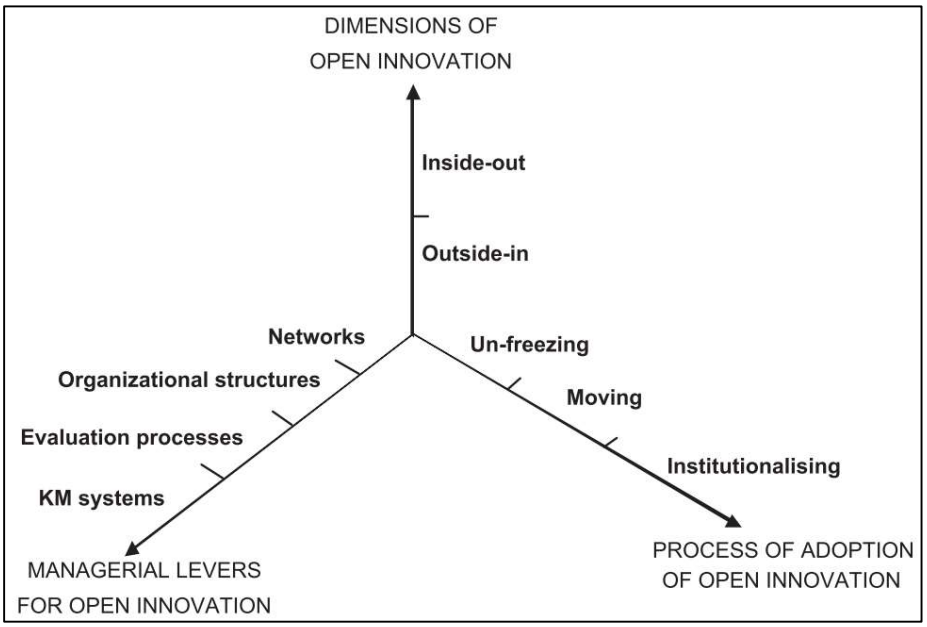


Figura 16 - Enquadramento teórico de Open Innovation [36]

Em suma, pode-se resumir o OI nos seguintes pontos [38]:

- O conhecimento exterior à organização é tão importante quanto o interior;
- Para criar valor é necessário criar modelos de negócios que invistam em pesquisa e desenvolvimento;
- Existência de canais de exposição do conhecimento interno;
- O conhecimento desenvolvido numa organização supera os limites da mesma;
- A gestão da propriedade intelectual deve ter uma abordagem diferenciada e proativa e não defensiva.

2.2.2.2. Open Source

A mais-valia e o incentivo que o *Open Source* (OS) traz são os processos o desenvolvimento distribuído que realçam as capacidades de modificação do conceito de *Openness*. O conceito de OS tem as suas origens na indústria software e os principais intervenientes são os programadores. O software é, tradicionalmente, construído pelos programadores, dentro das empresas comerciais com o intuito de ser vendido ou licenciado e por isso mesmo é fortemente protegido quer tecnologicamente como legalmente também [32]. A contraposição deste paradigma é do OS que é desenvolvido de forma coordenada por várias entidades independentes e voluntárias. Este desenvolvimento é desempenhado através do acesso e partilha pelos seus intervenientes do código fonte (*source code*). Este código pode ser modificado e redistribuído de acordo com termos estabelecidos e assim fomentar a melhoria continua assim como a expansão deste. Alguns exemplos de software desenvolvido desta forma são o Linux e o Mozilla Firefox [32]. Existem plataformas de OS software onde se pode encontrar muitos outros tipos de programas para todos os tipos de utilização. Uma delas é o SourceForge que é usada por Castro, H. et al. [33] para encontrar software de CAD a ser usado numa iniciativa de OD.

Na Tabela 2 tem-se um resumo das diferenças entre o software fechado e o OS [39].

Tabela 2 - Abordagem aos avanços tecnológicos e tendências futuras da propriedade de dados, adaptado de Castro, H. et al. [39]

	Fechado e Proprietário	***Open Source***
Propriedade dos dados	**Institucional**	**Descentralizada**
Abordagem aos avanços tecnológicos	Monetização de dados mantendo uma abordagem que mantém a propriedade intelectual privada e inacessível ao utilizador final	Desenvolvido e testado através de colaboração "aberta"
	O software é propriedade do individuo ou da organização que o desenvolveu	O código fonte pode ser acedido, modificado e redistribuído pela comunidade de *developers* e programadores
	O mercado é limitado pelo número de *developers* e consumidores finais, influenciado pelos custos e flexibilidade	Estimula a inovação de pequenas e médias empresas assim como de indivíduos ao acederem a plataformas de OS sem nenhum custo
Tendências futuras	Várias organizações governamentais têm regulamentado a proteção e privacidade dos dados, dando aos consumidores mais controlo sobre as informações pessoais que as empresas recolhem sobre eles. Com a crescente consciencialização e discussão pública em torno da privacidade e propriedade dos dados, é provável que o futuro das abordagens fechadas e de propriedade sobre software e tecnologias emergentes seja cada vez mais descentralizado.	As recentes mudanças para modelos OS são indicativas da natureza cada vez mais colaborativa dos avanços tecnológicos, e do interesse crescente dos consumidores em compreender como as tecnologias que utilizamos têm impacto nas nossas vidas. Os principais desafios a uma adoção mais ampla de plataformas OS são as vulnerabilidades de financiamento e segurança, mas é provável que as tecnologias descentralizadas e a propriedade de dados venham a desempenhar um papel mais importante no futuro.

2.2.2.3. Open Design

Como já referido, o OD é uma das variações de *Open-X*. O seu propósito é abordar o desenvolvimento de software e hardware, sendo desta forma mais abrangente que OS, usando métodos de design em OS, apoiado por uma comunidade que compartilha conhecimento [33]. Em suma, o conceito de OD baseia-se no design OS e na redefinição da propriedade intelectual e industrial para facilitar o desenvolvimento colaborativo de hardware e software [40].

No OD os consumidores são os principais "atores". Os designers, programadores e *developers* desempenham um papel importante ao produzirem, partilharem e promoverem, mas são os consumidores finais que fazem a sua distribuição [32]. Em contraste com o design tradicional, que é apenas uma etapa no processo de fabrico, umo OD é direcionado ao consumidor que participa nesse fabrico [32]. OD deixa implícito que os projetos sejam publicamente disponibilizados, licenciados seguindo uma abordagem de *open-access*, partilhados, e distribuídos digitalmente usando ficheiros comuns [32]. Como já referido, o carácter livre desta abordagem não significa que haja uma anarquia em relação ao acesso a conteúdos e informação. Existem acordos de licença asseguram que as liberdades, descritas na *Open Design Definition* (ODD), são protegidas à medida que o desenho ou modelo é modificado e redistribuído. Qualquer pessoa é livre de utilizar o desenho ou modelo aberto como elemento funcional de um sistema proprietário ou não proprietário, desde que o elemento do design ou modelo seja claramente identificado. Contudo, qualquer indivíduo ou organização que utilize ou modifique um design ou modelo deve concordar com os termos especificados numa licença [41]:

- a documentação de um design ou modelo está disponível gratuitamente;
- qualquer pessoa é livre de utilizar ou modificar o design, alterando a documentação do desenho;
- qualquer pessoa é livre de distribuir os designs originais ou modificados (gratuitamente ou não);
- as modificações ao design devem ser devolvidas à comunidade (se redistribuídas)

Em relação aos processos do OD é possível haver variações em ralação a como são gerados os produtos. Na Figura 17 é possível ver a forma como contribuições abertas e não abertas se combinam para a criação de soluções de design [42].

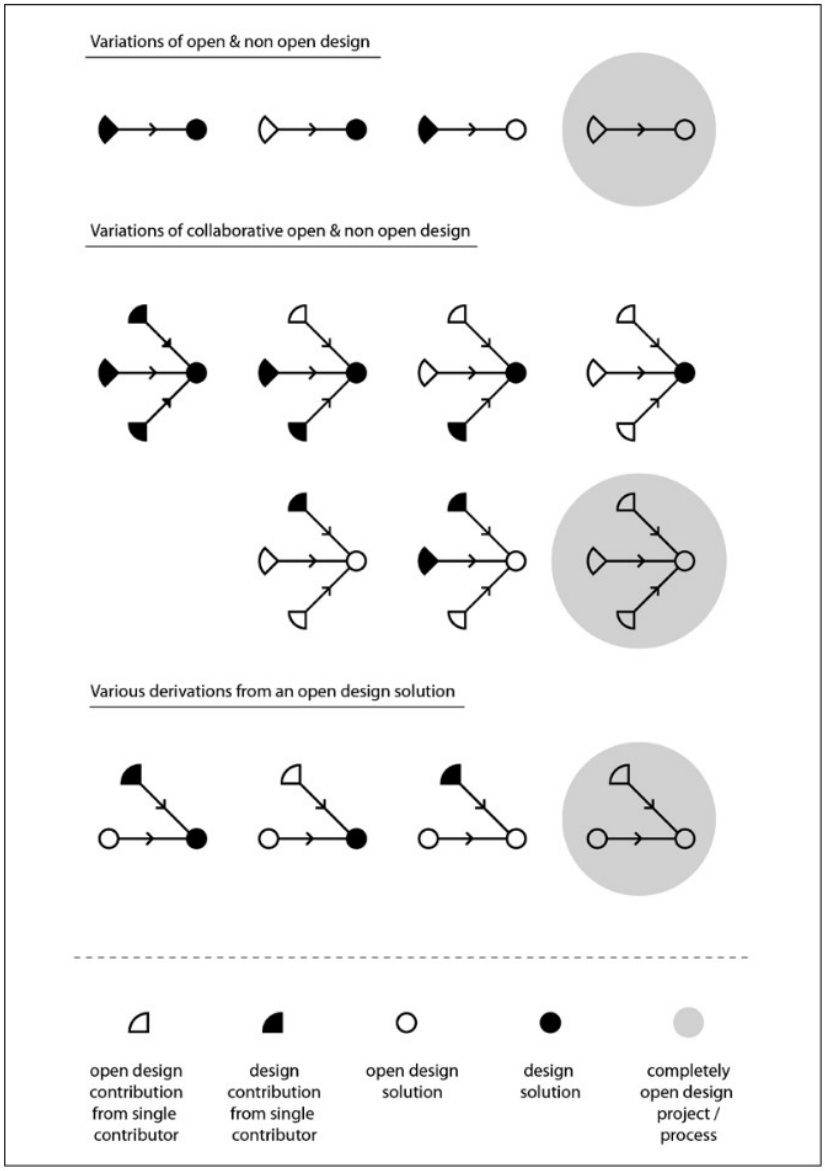


Figura 17 - Variações de processo de Open Design [42]

As metodologias de OD e a sua categorização de acordo com os seus níveis de utilização proporcionam suficiente informação sobre como estão relacionados com a gestão do conhecimento em termos de distribuição e democratização de acesso aos dados. De modo a simplificar pode-se categorizar essas metodologias em 3 níveis (Sistemas, Processos e Ferramentas) como se pode ver na Tabela 3. O nível do sistema, através da abordagem peer-to-peer, lida com infraestruturas e age como camada principal de execução de sistemas baseados em dados para a gestão de soluções baseadas no conhecimento. A nível do processo, o *co-design* sugere o envolvimento de não-designers e outros intervenientes no processo global de processos de conceção dentro da perspetiva da capacidade do OD de transformar diversos conhecimentos em componente integrada nas equipas e organizações de design. O terceiro nível, ferramentas, e os seus sub-tópicos, cocriação e desenho participativo, centra-se nos intervenientes que participam em massa em qualquer projeto nas organizações [43].

Tabela 3 - Metodologias do OD e os seus condutores [43]

Nível	Metodologias	Condutores existente de OD
Sistemas	*Peer-to-Peer*	Sistemas Difusos Sistemas Distribuidos Sistemas Descentralizados Sistemas Centralizados
Processos	*Co-design*	Diálogo Interação Relacionamentos Troca de conhecimento
Ferramentas	Design Participativo	Equidade, Experiência, Personalização, Partilha de Conhecimento
	Co-criação	Democracia, Participação

2.3. Os sistemas ciberfísicos e o Open Design

O encadeamento dos sistemas ciberfísicos no Open Design não é uma matéria ainda muito desenvolvida na literatura tendo sido encontrado apenas um exemplo de trabalho desenvolvido nesse sentido.

Nesse trabalho [44], o objetivo é criar um laboratório de manufatura adotando práticas de *reproducibility and replicability* (R&R), orientado para a comunidade de investigadores, usando práticas de OD e numa dinâmica de produção controlada por um CPS. O nome do projeto é *Open Science Lab for Manufacturing* (OSLab4Man). Na Figura 18 é mostrada a arquitetura do CPS usado neste projeto, e como já foi referido, apoiado no OD. Aos membros de uma comunidade é dado livre acesso a um repositório que se encontram vários produtos geridos por essa comunidade. A ligação com o mundo físico é feita através de interfaces CPS e concretizada pelo meio de dispositivos IoT.

OS CPS's ao usarem cada vez maiores quantidades de sensores e atuadores, em redes cada vez mais complexas, necessitam de formas de gerir essa informação. O desenvolvimento e operação de software tornaram-se cada vez mais dependentes de dados e estes dados podem ser mais a

cessíveis às organizações e indivíduos através de tecnologias de partilha de dados OS [39][45].

Por causa da natureza a aberto do OD, os produtos desenvolvidos tanto podem ser simples ou de grande complexidade. Um CPS deve ser capaz de acompanhar essa complexidade. Para isso pode-se referir de novo o nível de aprendizagem de um sistema. A aprendizagem como já referido pode ser feita num simples *loop*, em duplo *loop* ou teoricamente n *loop*. O paralelismo com o OD

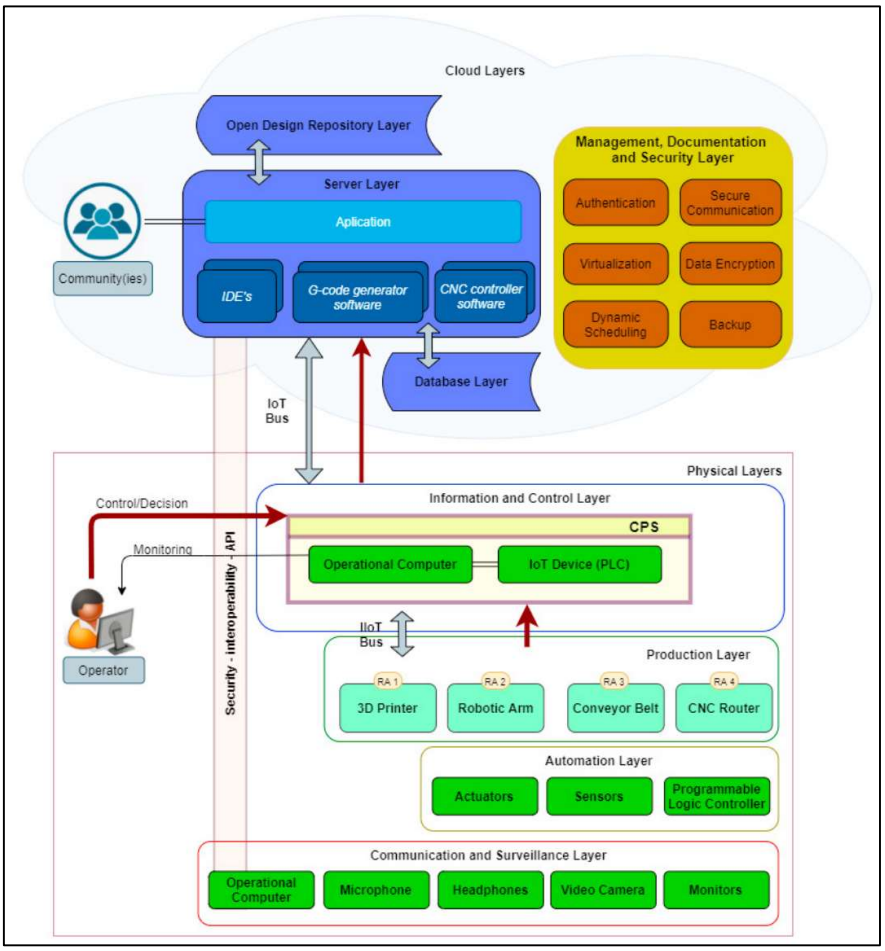


Figura 18 - Arquitetura de um CPS suportado por OD [44]

encontra-se com o uso de aprendizagem *single-loop* para detetar e corrigir erros e assim expandir as competências e conhecimento de uma organização sem alterações da natureza fundamental da mesma [15]. Com a aprendizagem *double-loop* para alem da deteção e correção de erros e problemas, a organização é sujeita a potenciais mudanças de normas, procedimentos, políticas e entre outros aspetos [15]. Um CPS assente em OD terá portanto que ser capaz de se adequar à complexidade de um ambiente em constante mudança e por isso usar uma aprendizagem de duplo *loop*.

página propositadamente em branco

3. PROPOSTA DE UM SISTEMA CIBERFÍSICO

3.1. Enquadramento

O sistema ciberfísico proposto será versão do sistema proposto em Castro et al. [44], o *Open Science Lab for Manufacturing* (OSLab4Man).

A arquitetura divide-se em duas componentes principais:

- Componente Computacional: Baseia-se em soluções de computação em nuvem, explorando as vantagens da computação distribuída e assegurando acesso remoto. A opção pela nuvem facilita a integração de diferentes agentes, permitindo a colaboração e participação ativa da comunidade no aprimoramento e evolução do sistema.
- Componente Física: Consiste numa célula de produção integrada que inclui um computador operacional responsável pela gestão de várias ferramentas, nomeadamente um braço robótico, uma fresadora CNC, uma impressora 3D e um tapete de recolha. Adiciona-se a esta configuração elementos de automação, tais como PLCs, sensores e atuadores, complementados por sistemas de monitorização audiovisual. Na Figura 19 tem-se o layout da célula de produção proposta.

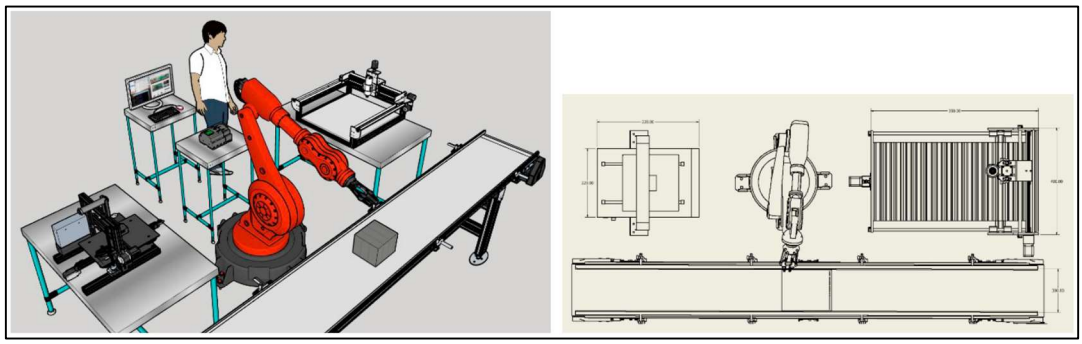


Figura 19 - Layout da célula de produção [44]

O acesso ao sistema é diferenciado, permitindo a intervenção de múltiplos agentes. A comunidade interage principalmente através da componente em nuvem, enquanto o operador tem uma interação mais direta e local com a célula de produção. A presença de elementos produtivos, combinada com dispositivos de automação e monitorização, cria um ambiente altamente adaptável e preparado para enfrentar os desafios da produção moderna. O Diagrama de Fluxo do Sistema Físico do CPS, representado na Figura 20, ilustra as relações e interações entre os vários componentes físicos da arquitetura. Este diagrama visa facilitar a compreensão das operações e das dinâmicas existentes na célula de produção.

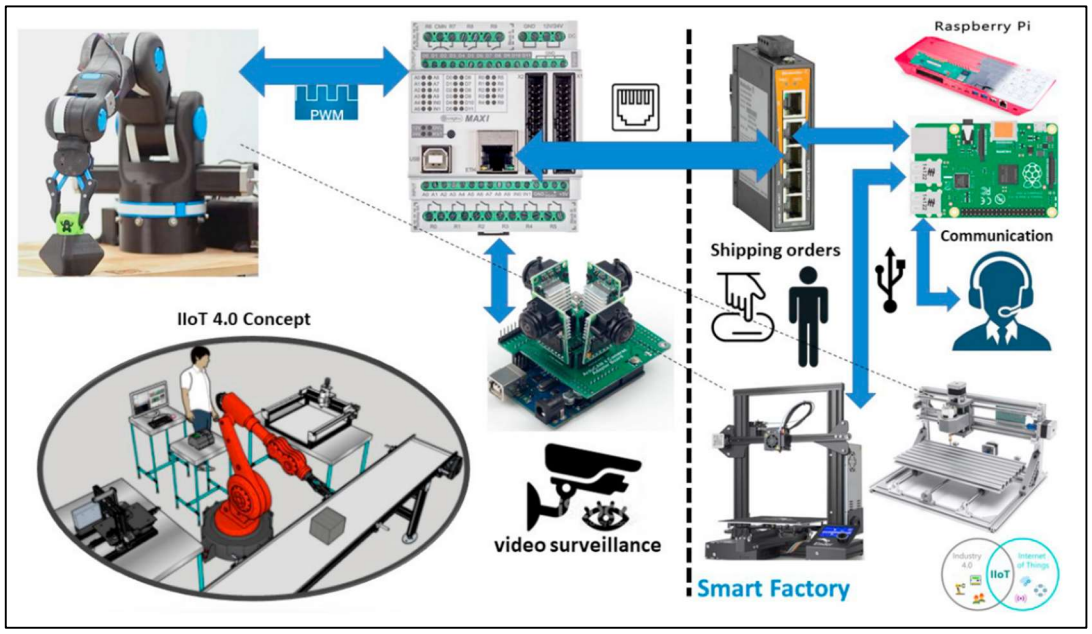


Figura 20 - Diagrama de fluxo do Sistema Físico [44]

A arquitetura proposta para o sistema ciberfísico caracteriza-se pela sua dualidade, composta por uma vertente computacional, alojada na *cloud*, e uma vertente física, centrada nos equipamentos e dispositivos de produção. Toda esta estrutura e interação entre os diferentes componentes encontram-se representadas de forma esquemática na Figura 21.

Parte Computacional na *Cloud*:

No núcleo da vertente computacional encontra-se o servidor. Este alberga a aplicação principal, diversas IDEs (Ambientes de Desenvolvimento Integrado) e *softwares* dedicados à geração de linguagem standard de comando numérico.

Ligado ao servidor, localiza-se um repositório Open Design, promovendo a partilha e a colaboração em projetos de design aberto.

Adjacente ao servidor, a base de dados armazena e gere a informação relevante, garantindo a integridade e a acessibilidade dos dados.

A sustentar toda esta estrutura, o gestor de serviço *cloud* desempenha funções cruciais como autenticação, comunicação segura, virtualização, encriptação de dados, agendamento dinâmico e operações de backup, garantindo a robustez e a segurança da componente computacional.

Domínio Físico:

O sistema de informação e controlo, composto pelo computador operacional e dispositivos IoT (Internet das Coisas), assume-se como a ponte entre a vertente computacional e a vertente física.

Ligado ao sistema de informação e controlo, o sistema de produção integra uma série de equipamentos: uma impressora 3D, um braço robótico, um tapete transportador e uma fresadora

CNC. Estes equipamentos trabalham de forma interligada e coordenada, sob a orientação do sistema de informação e controlo.

Em paralelo, o sistema de automação, composto por autómatos, atuadores e sensores, facilita e monitoriza o processo produtivo, assegurando a eficiência e a precisão das operações.

Complementarmente, existe um sistema de comunicação e vigilância, que integra elementos audiovisuais, permitindo a supervisão e monitorização remota das atividades, assim como a interação entre operadores e máquinas.

Interligação e Acesso:

O operador local interage diretamente com o sistema de informação e controlo, beneficiando de uma interface direta e intuitiva.

Este sistema, por sua vez, comunica com o servidor na *cloud* através de uma ligação IoT e através do próprio computador operacional.

Já os utilizadores, cuja interação se deseja mais abrangente e remota, acedem ao sistema predominantemente via servidor na *cloud*, beneficiando de uma plataforma versátil e segura.

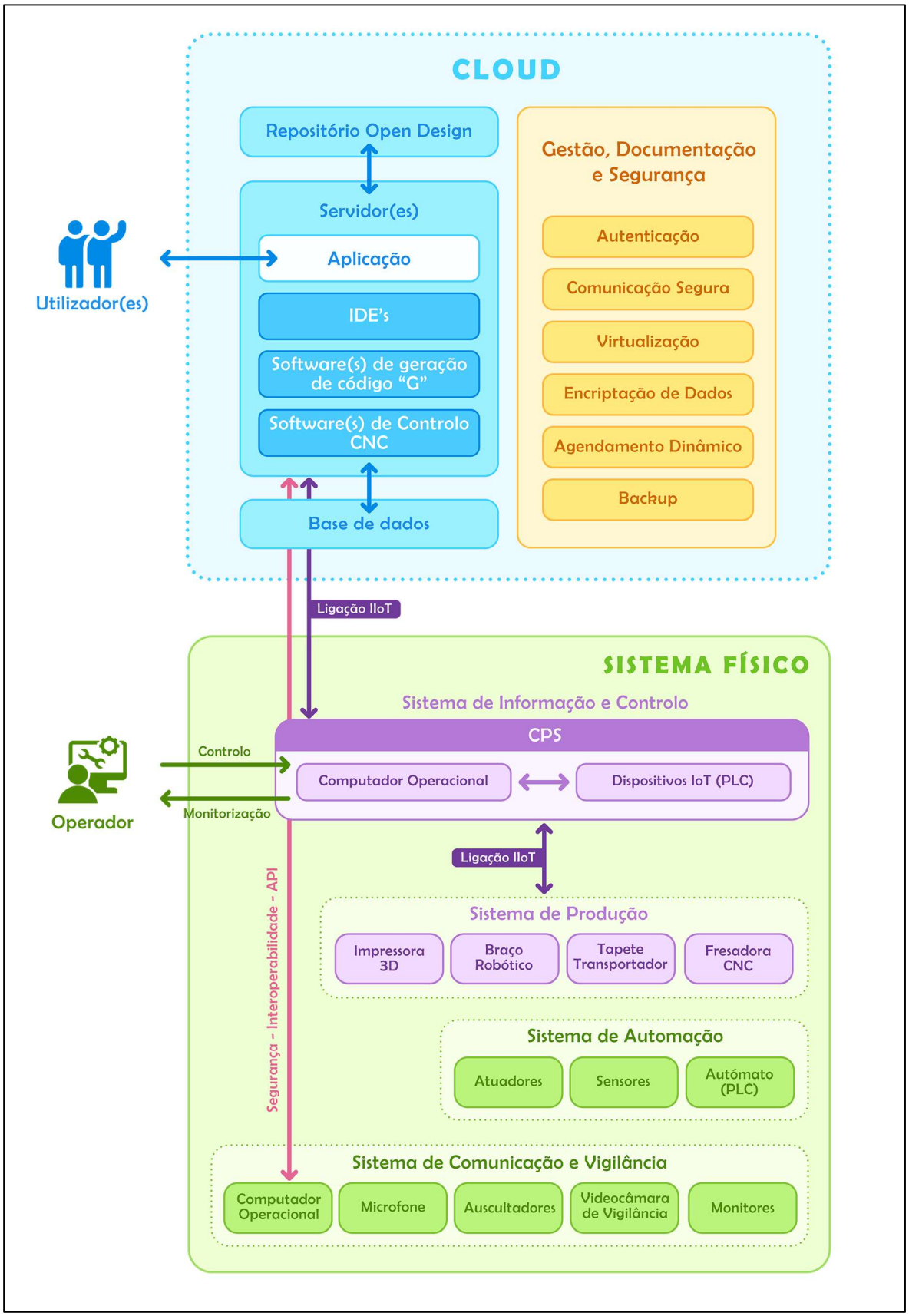


Figura 21 - Arquitetura OSLab4Man [44]

3.2. Análise de requisitos

De forma a ir de encontro com as necessidades de um sistema com a arquitetura proposta, e tendo em conta uma abordagem aberta no desenvolvimento das ferramentas necessárias, as seguintes condições apresentam-se:

1. Sistemas de controlo versáteis e abertos
2. Interoperabilidade entre sistemas
3. Componentes modulares
4. Configurabilidade
5. Protocolos de comunicação e interface standard

3.2.1. Sistemas de controlo versáteis e abertos

Dado a necessidade de uma plataforma aberta e colaborativa, este software adota uma natureza *open source*. Em contraste com soluções proprietárias, o *software open source* não só permite a adaptação e melhoria por outros investigadores e profissionais, mas também promove a transparência e replicabilidade, aspetos cruciais em qualquer investigação científica.

Para a construção da aplicação principal, diversas linguagens de programação são consideradas, incluindo Python, Java e C++.

Em relação ao *firmware*, existem várias alternativas, como o *Repetier* [46] ou o *Smoothie* e o *Marlin*.

Os comandos numéricos constituem a base das operações de impressoras 3D e fresadoras CNC. Esta linguagem de programação, específica para o contexto de produção, foi concebida para transmitir comandos detalhados e precisos às máquinas, orientando-as em relação a movimentos, trajetórias, velocidades e um leque de outras operações essenciais. Ao contrário de linguagens como *Python* ou *Java*, que são linguagens de programação de alto nível, os comandos numéricos são considerados de baixo nível. Enquanto *Python* e *Java* detêm um grau de abstração que os aproxima da linguagem humana, os comandos numéricos são diretos e específicos, disponibilizando instruções que são prontamente traduzidas em movimentos físicos pela máquina.

Quanto à comunicação, são considerados protocolos como o MQTT Enfileiramento de Mensagens por Transporte de Telemetria (*Message Queuing Telemetry Transport*), o AMQP (Protocolo avançado de enfileiramento de mensagens - *Advanced Message Queuing Protocol*) [47], e o CoAP (Protocolo de Aplicação Restrita - *Constrained Application Protocol*) [48]. Estes protocolos apresentam-se como soluções viáveis devido às suas características distintas.

Para a interface gráfica, consideram-se opções como o *Node-RED*, o *Grafana* e o *Home Assistant*. O Node-RED apresenta uma abordagem baseada em fluxos que é altamente intuitiva, promovendo uma integração facilitada com os diferentes componentes do sistema e as comunicações adequadas para a sua operacionalidade [49]. Por outro lado, o *Grafana* e o *Home Assistant* também possuem características distintas que os tornam soluções viáveis para diferentes aplicações e contextos.

3.2.2. Interoperabilidade entre sistemas

A interoperabilidade garante que diferentes sistemas ou componentes possam trabalhar juntos, trocar informações e utilizar as informações trocadas para operar eficazmente. No cenário proposto, cada sistema físico tem suas especificidades e finalidades, mas alguns princípios comuns devem ser observados:

- Estrutura Unificada de Construção: Uma estrutura consistente ao nível da construção física, por exemplo, na forma de interfaces modulares ou padrões de montagem, facilita a integração e manutenção de diferentes sistemas. Esta uniformidade pode ser alcançada através do uso de componentes padronizados ou técnicas de montagem similares;
- *Software* e Protocolos Comuns: Ao nível da programação, a adoção de uma linguagem comum proporciona um ambiente coeso. Esta coesão permite que os sistemas partilhem informações, interpretando-as de maneira uniforme, e respondam em concerto, atingindo objetivos globais;
- Comunicação Eficiente: A natureza distribuída destes sistemas requer uma comunicação eficaz;
- Flexibilidade e Escalabilidade: Com uma base comum em termos de hardware e software, é mais fácil expandir ou modificar o sistema no futuro. Se, por exemplo, um novo sistema físico necessitar de ser integrado, a estrutura existente pode acomodá-lo com mínimas modificações;
- Monitorização e Manutenção Centralizadas: Com um *framework* comum, é possível implementar sistemas de monitorização e manutenção que abranjam todos os componentes do CPS. Isso facilita o diagnóstico e a resolução de problemas, garantindo um funcionamento contínuo e eficaz.

Ao assegurar que estes sistemas físicos sejam interoperáveis entre si, não só se maximiza a eficiência individual de cada componente, como também se amplifica o potencial coletivo do sistema ciberfísico global. Esta integração bem-sucedida promete revolucionar o modo como as operações são realizadas, levando a avanços significativos em termos de produção, automação e inovação. Na Figura 22 está exemplificado a forma como o CPS virtualiza os componentes físicos do sistema, e depois estabelece a comunicação com estes sendo que a linguagem, protocolos e elementos físicos são comuns.

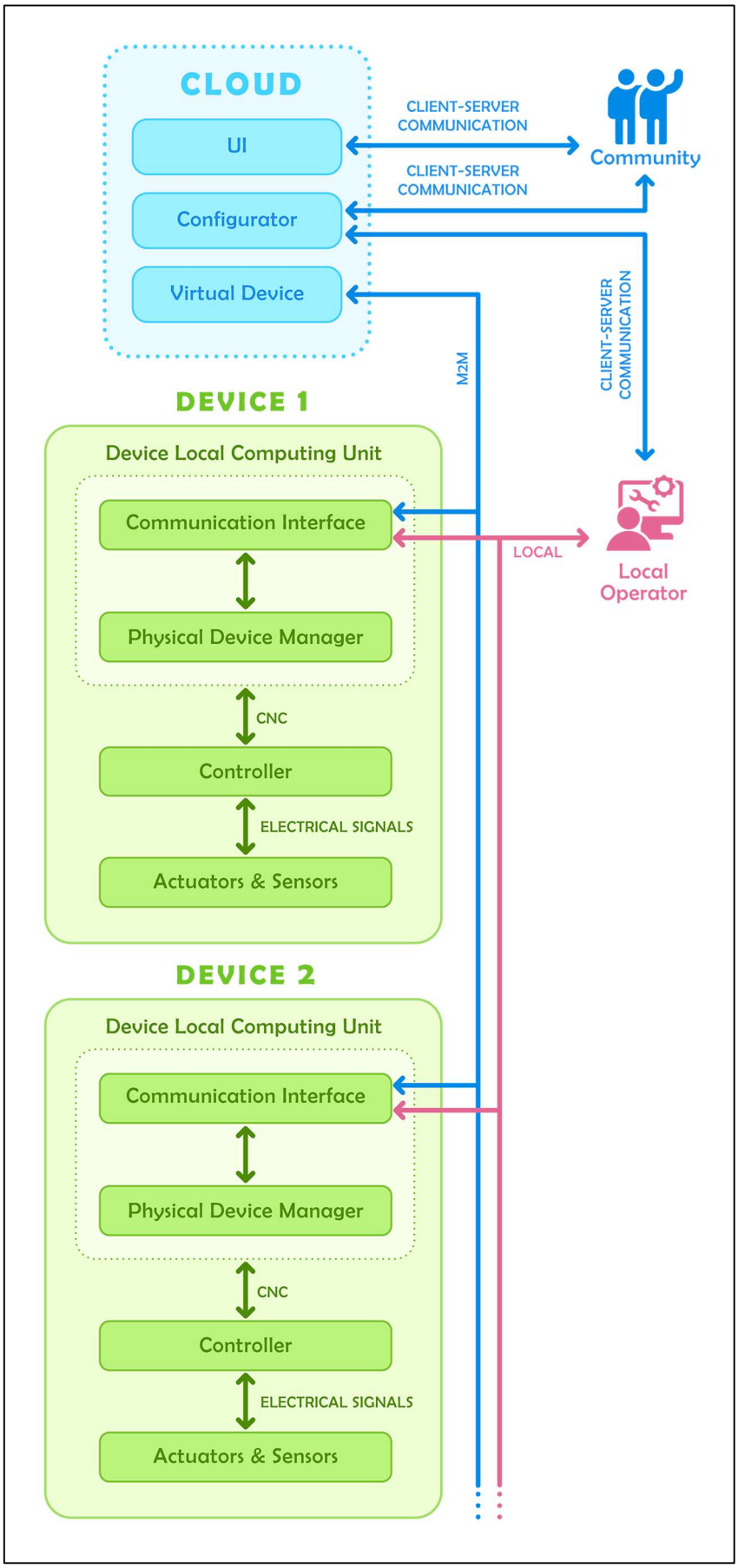


Figura 22 - Interoperabilidade entre sistemas

3.2.3. Componentes modulares

No desenvolvimento de sistemas ciberfísicos, a modularidade emerge como uma característica crucial para garantir adaptabilidade, escalabilidade e manutenção eficiente. No projeto em questão, a modularidade é abordada tanto ao nível físico quanto ao nível da programação, estabelecendo uma arquitetura coesa e flexível.

3.2.3.1. Nível Físico

A essência da modularidade no hardware é manifestada pelo uso de componentes intercambiáveis e configuráveis. Neste sistema, isso é evidenciado pela adoção de controladores uniformes para diferentes dispositivos, como a impressora 3D, a fresadora CNC e o braço robótico. Utilizar os mesmos controladores para diversos elementos físicos proporciona diversas vantagens. Primeiro, simplifica o processo de integração e substituição de componentes, caso seja necessário. Além disso, esta abordagem uniformizada facilita a manutenção, uma vez que a equipe técnica precisa familiarizar-se apenas com um tipo de controlador, independentemente do dispositivo que ele opera. Esta uniformidade, portanto, não apenas maximiza a eficiência operacional, mas também otimiza custos e tempos de formação.

3.2.3.2. Nível de Programação

A modularidade no software é implementada através da criação de módulos ou funções específicas que podem ser reutilizadas em diferentes partes do sistema ou mesmo em projetos futuros. Cada módulo é desenvolvido para executar uma tarefa ou função específica, seja ela relacionada com o controlo da impressora 3D, a operação do braço robótico ou a gestão do armazém automático. Esta abordagem modular no código facilita a depuração, manutenção e expansão do software. Além disso, a reutilização de módulos pode acelerar o desenvolvimento de novas funcionalidades ou a integração de novos dispositivos ao sistema.

A abordagem modular adotada neste sistema ciberfísico representa uma visão avançada e estratégica de design e desenvolvimento. Ao enfatizar a modularidade em ambos os níveis, físico e de programação, este projeto posiciona-se não apenas para atender às necessidades atuais, mas também para adaptar-se e evoluir face aos desafios e oportunidades futuras.

A Figura 23 mostra implementação desta abordagem por módulos. A aplicação, o ponto de interação principal com o utilizador, é composta por diversos módulos que visam segmentar funções e características específicas. Esta modularidade facilita a manutenção, a integração de novas funcionalidades e garante que qualquer alteração em um módulo tenha impacto mínimo nos outros. A base de dados, crucial para o armazenamento e recuperação de informações, é igualmente modular. Esta estrutura permite uma gestão otimizada dos dados, com capacidade para expandir, reconfigurar ou até mesmo substituir módulos específicos sem comprometer a integridade global dos dados.

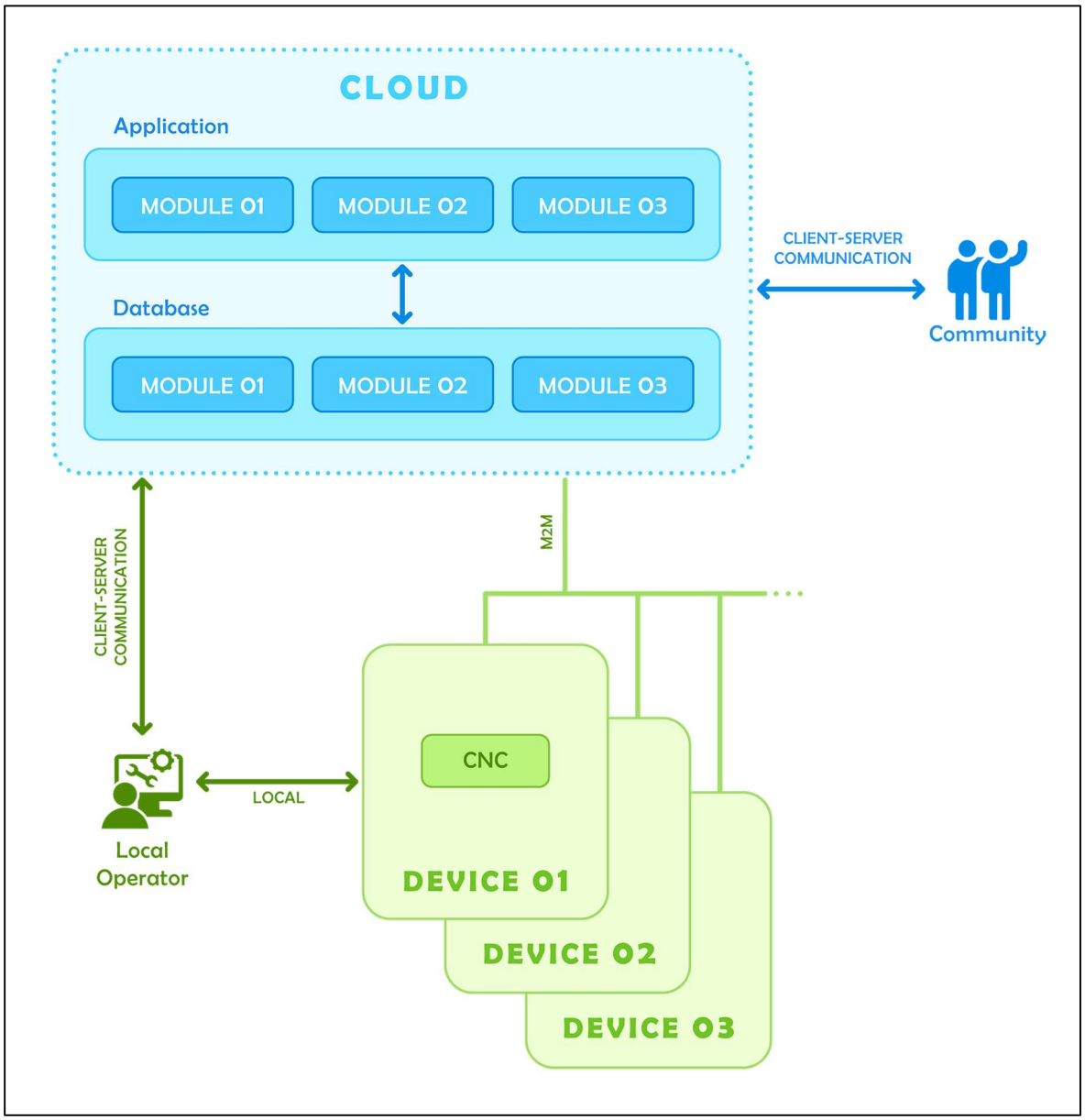


Figura 23 - Módulos computacionais e físicos

3.2.4. Configurabilidade

A configurabilidade, neste contexto, refere-se à capacidade do sistema de ser configurado de acordo com requisitos específicos, permitindo mudanças em seu comportamento, funcionalidades ou operações sem intervenção direta no código-fonte ou na estrutura física base. Este é um requisito vital, pois, para um sistema ciberfísico manter-se relevante e funcional ao longo do tempo, ele deve ser capaz de adaptar-se às novas necessidades ou circunstâncias.

No caso do projeto em questão, a configurabilidade é mais do que apenas um recurso desejável; é uma necessidade. Se existir uma mudança nas prioridades de operação ou na lógica de controlo, o sistema deve permitir a reconfiguração adequada sem a necessidade de reconstruir todo o *software* ou *hardware*.

Na Figura 22 é possível ver que a capacidade de configuração do sistema, quer pelo operador local como pela comunidade está bem estabelecida através das comunicações apresentadas.

3.2.5. Protocolos de comunicação e interface standard

A comunicação eficiente as interfaces padronizadas são elementos importantes que devem ser considerados na conceção de um CPS. Estes elementos garantem não apenas a coesão e operacionalidade do sistema, mas também a sua adaptabilidade e abrangência em um ambiente cada vez mais globalizado e conectado.

3.2.5.1. Protocolos de Comunicação

A adoção de protocolos como o MQTT assegura a transmissão contínua e sem erros de dados entre dispositivos. Estes protocolos fornecem mecanismos que garantem a entrega de mensagens, gestão da largura de banda e segurança dos dados transmitidos. Num contexto onde precisão e tempo real são vitais, protocolos de comunicação robustos são a base de operações fiáveis.

3.2.5.2. Interfaces Padrão

A relevância de interfaces padronizadas é destacada em diversos softwares. Estas interfaces garantem que diferentes componentes de um sistema sejam capazes de interpretar e atuar com base nos dados recebidos, independentemente da linguagem ou plataforma em que foram desenvolvidos. Esta padronização contribui para uma maior interoperabilidade e facilita a integração entre sistemas e componentes diversos.

3.2.5.3. Interconexão com a *Cloud*

A comunicação cliente-servidor é uma pedra angular na arquitetura de sistemas modernos e desempenha um papel crucial na integração com serviços de *cloud computing*. Este modelo de comunicação facilita o acesso e a interação com dados e serviços armazenados na nuvem. Através dele, é possível realizar monitoramento e controlo remoto de sistemas ciberfísicos, permitindo que os utilizadores interajam com estes sistemas a partir de qualquer localização e dispositivo. Esta interconexão garante a agilidade, escalabilidade e flexibilidade necessárias para responder às demandas dinâmicas do mundo atual.

3.3. Arquitetura da solução

Ao desenvolver um sistema ciberfísico, é essencial considerar a harmonia e integração entre os componentes computacionais e físicos. No caso deste projeto, o sistema foi delineado em dois domínios principais: o computacional, alojado na *cloud*, e o físico, responsável por ações concretas e monitorização.

3.3.1. Sistema computacional (*Cloud*)

O advento da computação em nuvem revolucionou a forma como processamos e armazenamos informação. Este paradigma trouxe uma eficiência e escalabilidade sem precedentes, permitindo que os sistemas operassem de maneira mais dinâmica e adaptável. Central para esta transformação está a combinação de diversos componentes que juntos formam a espinha dorsal da infraestrutura de *cloud*.

Componentes-Chave do Sistema Computacional de *Cloud*:

- **Servidor**: Atua como o núcleo da infraestrutura, processando solicitações e distribuindo informações de forma eficiente. É onde as aplicações são alojadas e executadas;
- **Base de Dados**: Responsável pelo armazenamento, recuperação e gestão de dados. É uma componente crucial para garantir que a informação seja acessível e esteja organizada de forma eficiente;
- **Gestor de Serviços**: Coordena e gere os diversos serviços oferecidos pela infraestrutura de *cloud*, assegurando que cada serviço seja executado de forma otimizada e alinhada com as necessidades específicas.

Além das componentes principais, várias soluções de software têm sido desenvolvidas para facilitar a gestão e operacionalização dos sistemas de *cloud*. O ***OpenStack*** é um exemplo notável, oferecendo um conjunto de ferramentas de código aberto para construir e gerir plataformas de *cloud* [50]. Outras alternativas poderiam ser:

- ***OpenNebula*****:** Uma solução robusta que fornece uma gestão simples, mas flexível, de infraestruturas híbridas [51];
- ***CloudStack***: Uma plataforma que oferece serviços de infraestrutura de *cloud* (IaaS), permitindo aos utilizadores criar, gerir e implementar infraestruturas de computação em larga escala [52];
- ***Eucalyptus***: Compatível com a API do AWS, é uma solução que permite a construção de *clouds* privadas, mantendo a flexibilidade de migrar para *clouds* públicas [53];
- ***Proxmox VE***: Foca-se na gestão de máquinas virtuais e na virtualização de armazenamento, permitindo uma gestão centralizada de vários nós [54].

Estas alternativas *open source* oferecem uma vantagem significativa em relação às soluções comerciais, não só pelo custo reduzido, mas também pela flexibilidade e transparência que proporcionam. Optar por soluções de código aberto significa adotar uma abordagem mais transparente, flexível e personalizável para a gestão de recursos de *cloud*.

Ao considerar todas estas componentes e ferramentas disponíveis, torna-se evidente o potencial e a versatilidade dos sistemas computacionais de *cloud*. A capacidade de adaptar, escalar e personalizar as infraestruturas torna a computação em nuvem uma escolha preferencial para aplicação em sistemas ciberfísicos.

3.3.1.1. Servidor (*Server*)

No contexto de um sistema ciberfísico, o servidor desempenha um papel central, garantindo uma operacionalidade eficaz e segura. Para uma compreensão mais profunda, descrevem-se a seguir as suas funções e componentes essenciais.

Função: O servidor emerge como o núcleo central da infraestrutura computacional, desempenhando uma função primordial na gestão de solicitações e na orquestração de informações entre a base de dados e o gestor de serviço.

Importância: A eficiência na execução das operações e a segurança na distribuição de dados são imperativas em qualquer sistema ciberfísico. O servidor assegura que esses padrões sejam mantidos, garantindo que os processos sejam eficazmente geridos e que a informação seja disseminada de forma coesa e segura.

Na Figura 24 estão representados os componentes do servidor:

- **Aplicação Principal**: Esta é a espinha dorsal do servidor, responsável por gerir as funções centrais e processar solicitações. Garante que todas as operações sejam executadas de forma fluída e sincronizada;
- **Interface do Utilizador (*User Interface* - UI)**: É a face visível do sistema para os utilizadores. Uma interface bem desenhada é crucial para garantir que os utilizadores possam interagir com o sistema de forma intuitiva e eficaz;
- **Configurador**: Ferramenta que permite aos administradores ajustar e personalizar o funcionamento do sistema, adequando-o às necessidades específicas do ambiente em que está inserido;
- **Ambientes de Desenvolvimento Integrado (*Integrated development environments* - IDEs)**: Estas plataformas fornecem as ferramentas necessárias para que os desenvolvedores possam criar, testar e implementar soluções de software, permitindo a adaptação e expansão contínua do sistema;
- **Equipamento Virtual (*Virtual Device*)**: Elemento crucial que permite a simulação e teste de operações antes de serem aplicadas ao ambiente real. Isto não só garante a eficiência das operações, mas também minimiza riscos associados a novas implementações.

A organização e colaboração eficaz entre estes componentes asseguram que o servidor não só atenda às necessidades imediatas, mas também esteja preparado para adaptações e evoluções futuras, fortalecendo a resiliência e adaptabilidade do sistema ciberfísico proposto.

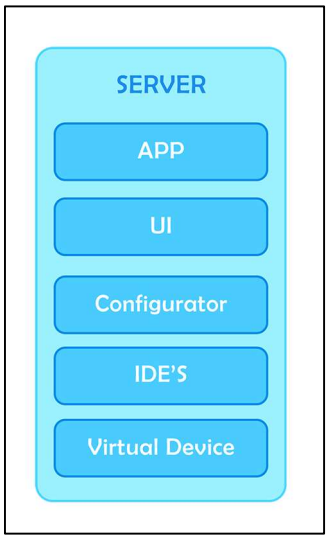


Figura 24 - Servidor Cloud

3.3.1.2. Base de Dados (*Database*)

Para a integral compreensão do sistema, é imperativo focar numa componente que serve de alicerce à gestão de informações: a Base de Dados. Assim, segue-se uma breve explanação da sua função e do seu papel crítico no contexto do sistema.

Função: Armazena todos os dados relevantes do sistema, desde configurações de máquinas até *logs* de operações.

Importância: Proporciona um acesso rápido e confiável às informações, permitindo que o sistema tome decisões informadas e supervisione as atividades.

3.3.1.3. Gestor de Serviço

Uma componente importante para a operacionalidade e segurança do sistema ciberfísico é o Gestor de Serviço. Desempenhando funções cruciais e garantindo interações eficientes e seguras.

Função: O Gestor de Serviço posiciona-se como a entidade central na orquestração dos serviços oferecidos pelo sistema. A sua principal responsabilidade é assegurar que cada serviço seja realizado conforme as especificações estabelecidas, bem como atendendo às particularidades e necessidades do momento.

Importância: Num sistema ciberfísico, a rapidez, eficácia e personalização são elementos-chave. O Gestor de Serviço garante que os serviços não só cumpram com os padrões de excelência, mas também que sejam capazes de adaptar-se e responder prontamente às demandas e alterações que possam surgir.

A Figura 25 representa o módulo dos componentes do Gestor de Serviços:

- **Autenticação**: Garante que apenas utilizadores autorizados tenham acesso ao sistema, proporcionando um ambiente seguro e minimizando o risco de intrusões maliciosas;
- **Comunicação Segura**: Estabelece canais de comunicação protegidos, assegurando que a informação trocada entre os diversos componentes do sistema ocorra de forma íntegra e confidencial;
- **Virtualização**: Permite a criação de versões virtuais dos recursos, o que facilita a gestão, escalabilidade e testes de novas funcionalidades sem interferir na operação real;
- **Encriptação de Dados**: Cuida da segurança da informação armazenada e transferida, convertendo-a em códigos para prevenir acesso não autorizado;
- **Agendamento Dinâmico**: Adapta-se em tempo real às demandas do sistema, programando tarefas e alocação de recursos de forma eficiente;
- **Backup**: Mantém cópias de segurança da informação, assegurando que, em caso de falhas ou perdas, os dados possam ser rapidamente recuperados.

O Gestor de Serviço, com todos estes componentes, não só otimiza a operação corrente, como também fornece uma fundação sólida que permite ao sistema ciberfísico adaptar-se e evoluir face aos desafios e inovações futuras.

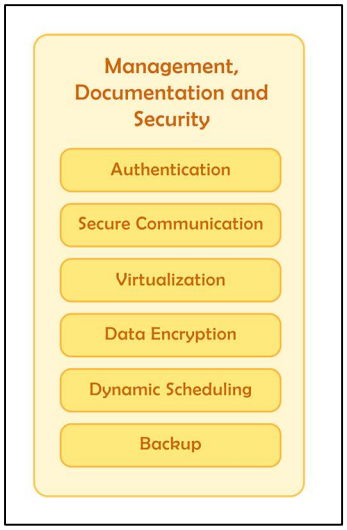


Figura 25 - Gestor do serviço de cloud

3.3.2. Sistema Físico (*Physical System*)

O sistema ciberfísico estrutura-se em diversas unidades funcionais, cada uma com um papel específico. Estas unidades, interligadas, garantem uma transição fluida entre o ambiente digital e o

físico. Exploram-se três unidades fundamentais para compreender como as operações do sistema se materializam: Unidades de Produção, Unidade de Automação e Unidade de Monitorização.

3.3.2.1. Unidades de Produção (*Manufacturing Units*)

As unidades de Produção são caracterizadas, nesta secção, pela sua função, importância e estrutura.

Função: As unidades de produção são vitais para o sistema ciberfísico, pois são responsáveis pela materialização das tarefas de produção. Estas unidades têm como papel a execução de funções específicas relacionadas à fabricação, montagem ou processamento de componentes.

Importância: Estas unidades servem como o "braço ativo" do sistema ciberfísico, convertendo as instruções digitais e comandos computacionais em ações tangíveis e reais no mundo físico. Elas garantem que os processos de produção sejam realizados de acordo com os parâmetros estabelecidos.

Estrutura:

A unidade de produção é segmentada em três componentes principais:

- **Unidade de Computação Local (*Local Computing Unit*)**: Esta atua como o núcleo de comunicação entre o servidor central e o ambiente de produção. A Figura 26 destaca uma clara distinção dentro desta unidade, dividindo-a entre o Interface de Comunicação (*Communication Interface*) e o Elemento de Controlo Físico (*Physical Device Manager*). Enquanto o software do Interface de Comunicação estabelece a ponte de diálogo com outras unidades e o servidor central, o Elemento de Controlo Físico recebe comandos de alto nível do Interface de Comunicação, traduzindo-os em instruções de baixo nível de comando numérico, que são então enviadas ao controlador;

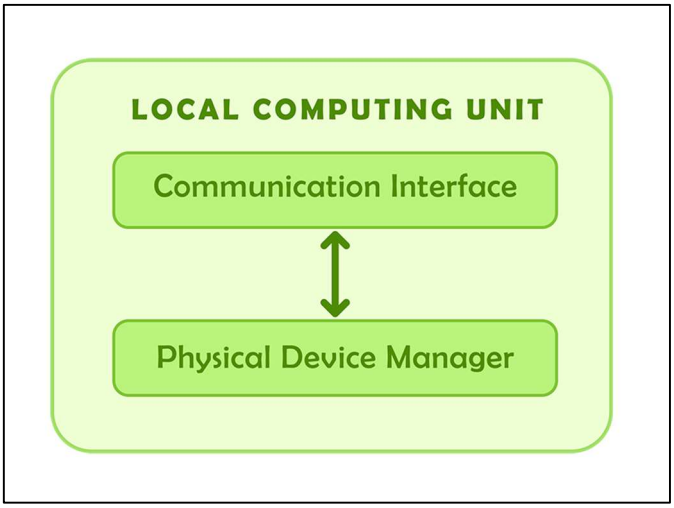


Figura 26 - Unidade de Computação Local

- **Controlador (*Controller*)**: O controlador é responsável por interpretar as instruções de baixo nível recebidas e dirigir as operações físicas apropriadas nos atuadores e sensores;
- **Atuadores e Sensores (*Actuators & Sensors*):** Estes componentes são os que executam ou monitorizam ações no mundo físico. Eles respondem diretamente às instruções provenientes do controlador, garantindo que as operações sejam realizadas conforme comandado.

A Figura 27 ilustra a comunicação intrínseca aos componentes de uma unidade de produção. Esta unidade, de forma integrada, consegue processar instruções de alto nível, adaptando-as para que sejam compreendidas pelo controlador. Este, por sua vez, converte as instruções recebidas em sinais elétricos, transmitindo-os ao sistema físico para efetuar a operação desejada. Da mesma forma, o fluxo inverso de informações também ocorre: o sistema físico, ao detetar ou processar uma ação, envia sinais elétricos ao controlador. Este, ao interpretar os sinais, comunica-se com a unidade de computação local através de instruções de baixo nível, fornecendo feedback ou atualizações sobre o estado atual do sistema.

Ao conceber estas unidades com uma arquitetura modular e interfaces padronizadas, a flexibilidade e adaptabilidade são maximizadas, permitindo a implementação de melhorias, ajustes ou expansões no futuro com menor esforço e complexidade.

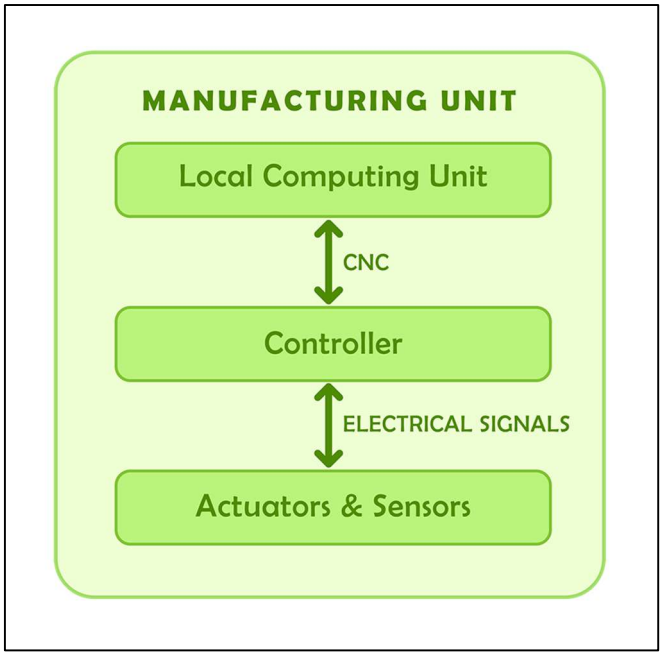


Figura 27 - Unidade de Produção

3.3.2.2. Unidade de Automação (*Automation Unit*)

As unidades de Automação são caracterizadas, nesta secção, pela sua função, importância e estrutura.

Função: A Unidade de Automação assume a responsabilidade central de gerir e controlar de forma automatizada os diversos processos e operações, garantindo que estas decorram de forma otimizada, tanto em termos de eficiência como de precisão.

Importância: Uma das grandes valias desta unidade é a sua capacidade de minimizar a necessidade de intervenção humana, tornando as operações mais autónomas. Esta autonomia não só liberta os operadores para se concentrarem em tarefas de maior valor agregado, mas também garante que todas as operações sejam realizadas de forma consistente, reduzindo a margem de erro e potenciando uma maior precisão nas tarefas executadas.

Composição e Funcionamento: Esta unidade é estruturada em dois elementos-chave: o Controlador Lógico Programável (*Programmable Logic Controller* – PLC) e o conjunto de Atuadores e Sensores. O PLC, que serve como o cérebro da operação, interage diretamente com os atuadores e sensores através de sinais elétricos. Esta interação, que é fundamental para a coordenação e execução eficaz das tarefas, está detalhadamente exemplificada na Figura 28. O PLC interpreta os comandos e instruções, convertendo-os em ações concretas ao enviar os sinais elétricos apropriados para os atuadores. Em simultâneo, recebe *feedback* dos sensores, o que lhe permite ajustar e refinar continuamente as operações em tempo real. Esta dinâmica assegura que o sistema responde eficazmente às demandas e adapta-se às variáveis do processo**.**

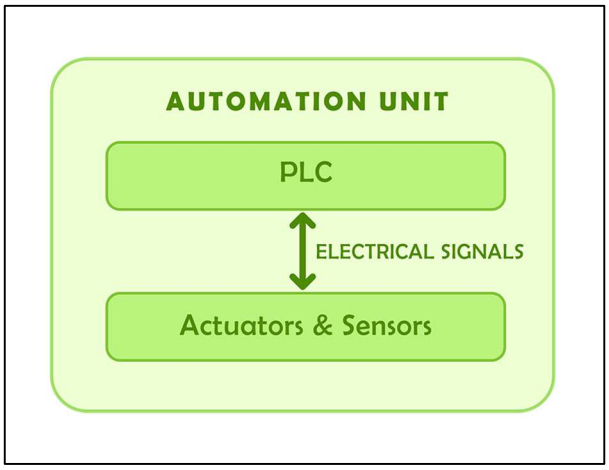


Figura 28 - Unidade de automação

3.3.2.3. Unidade de Monitorização (*Monitoring Unit*)

As unidades de Monitorização são caracterizadas, nesta secção, pela sua função, importância e estrutura.

Função: A Unidade de Monitorização desempenha o papel vital de acompanhar e avaliar constantemente o status, o desempenho e a condição geral de diversos componentes e operações que compõem o sistema. Esta vigilância contínua abrange desde a operação de maquinário até as interações e comunicações entre diferentes componentes.

Importância: Esta unidade é crucial para a integridade geral do sistema, uma vez que fornece feedback contínuo sobre todas as operações em andamento. Ao monitorizar em tempo real, é possível detetar e corrigir rapidamente qualquer desvio ou anomalia. Assim, não só se assegura que os padrões de qualidade e eficiência sejam consistentemente mantidos, mas também se maximiza a segurança e a fiabilidade do sistema.

Composição e Funcionamento: A estrutura desta unidade apresentada na Figura 29 tem no nível superior, a Unidade de Computação Local, cuja arquitetura e funcionalidade são idênticas às encontradas nas unidades de produção. Esta unidade serve como o principal núcleo de processamento e gestão de dados.

Complementando a Unidade de Computação Local, temos uma variedade de equipamentos especializados dedicados à captura e apresentação de dados.

Vídeo: As câmaras, responsáveis por captar imagens em tempo real, estão ligadas diretamente à Unidade de Computação Local. Estas câmaras recebem comandos desta unidade e, em retorno, enviam-lhe dados visuais. Os monitores, por sua vez, recebem o *feedback* visual processado da Unidade de Computação Local, apresentando-o de forma compreensível para monitorização ou análise.

Áudio: Os microfones captam informações sonoras do ambiente ou das operações em curso. Estes sinais são depois transmitidos à Unidade de Computação Local para processamento. As colunas, por outro lado, recebem instruções ou alertas sonoros da unidade, permitindo uma comunicação bidirecional.

Outros Dispositivos: Esta categoria abrange uma gama diversificada de equipamentos, incluindo dispositivos hápticos. Estes dispositivos podem, por exemplo, receber informações táteis do ambiente operacional e transmiti-las à Unidade de Computação Local. Em certos contextos, dispositivos como estes podem ser cruciais para perceber anomalias ou mudanças sutis no ambiente que não são facilmente detetáveis através de meios visuais ou sonoros.

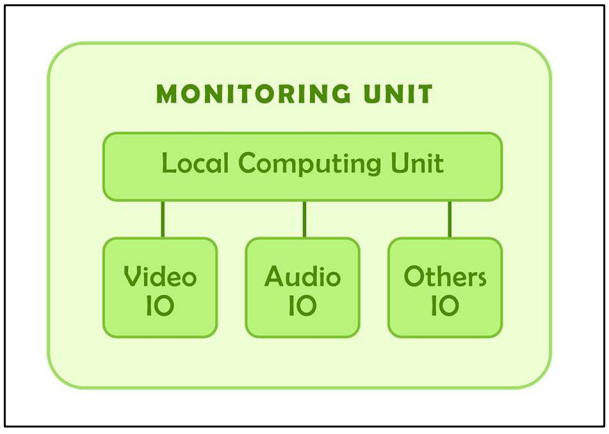


Figura 29 - Unidade de monitorização

3.3.3. Comunicação e acessos

A integração eficiente entre comunicação e acessos determina a eficácia operacional. Este segmento examina as modalidades de interação entre comunidade, operadores locais e unidades distintas. Utilizando protocolos padronizados e interfaces otimizadas, prioriza-se a precisão e a fluidez na produção.

3.3.3.1. Acesso da Comunidade

O acesso da comunidade ao sistema ciberfísico baseia-se em duas modalidades de comunicação: o acesso através da interface de utilizador e o acesso ao conteúdo do servidor. Ambas as modalidades garantem a eficácia e fluidez da interação entre os utilizadores e o sistema.

Os acessos são feitos das seguintes formas:

- **Acesso através de Interface de Utilizador (UI):** Este tipo de acesso permite que a comunidade interaja diretamente com os componentes físicos do sistema. Através de uma interface gráfica intuitiva, os utilizadores podem monitorizar, controlar e configurar os componentes do sistema ciberfísico, facilitando a manipulação e gestão destes elementos em tempo real. Usando ferramentas como o *Node-Red* que é orientada por modelos baseada no navegador para ligar os fluxos entre dispositivos IoT [55]. A ferramenta permite uma abordagem de um clique para implantar as capacidades de execução. O Node-RED utiliza o *Node.js* (um motor de execução *JavaScript*). Os fluxos são armazenados como objetos JSON [56];
- **Acesso ao Conteúdo do Servidor:** Além da interação direta com os componentes, a comunidade tem a capacidade de aceder ao conteúdo armazenado no servidor. Este acesso proporciona uma flexibilidade maior, pois permite aos utilizadores introduzirem modificações, implementarem melhorias e acrescentarem informações ao sistema. Este nível de interação fomenta a colaboração e a partilha de conhecimentos, enriquecendo o ecossistema do projeto e potenciando o seu desenvolvimento contínuo.

3.3.3.2. Acesso do Operador Local

O operador local interage com o sistema através de dois métodos principais: via servidor, alinhando-se com os protocolos da comunidade, e acesso direto aos sistemas físicos. Este último, permitindo-lhe uma intervenção direta sem intermediação do servidor:

- **Via Servidor (Semelhante à Comunidade)**: O operador local tem a opção de interagir com o sistema da mesma maneira que a comunidade. Esta uniformidade na interface e protocolo garante que o operador esteja familiarizado com os processos padrão que a comunidade utiliza.
- **Acesso Direto ao Sistema Físico**: O diferencial crítico para o operador local é a capacidade de interagir diretamente com os sistemas físicos. Dada a sua proximidade e responsabilidade, o operador tem as autorizações e os meios para aceder, diagnosticar e até intervir nos sistemas físicos sem a necessidade de passar pelo servidor.

3.3.3.3. Comunicação entre unidades e sistemas

A comunicação entre sistemas deve usar protocolos IoT e M2M (*Machine-to-Machine*). Estes protocolos são projetados especificamente para facilitar a interação entre máquinas, minimizando o envolvimento humano direto. As vantagens destes protocolos incluem a capacidade de operar em ambientes de rede instáveis, consumir menos largura de banda, garantir entregas de mensagens mais confiáveis e fornecer comunicações mais seguras. Ao adotar tais protocolos, os sistemas podem interagir de maneira mais autônoma, otimizando processos e melhorando a eficiência geral:

- **Unidades de Produção**: São responsáveis pela execução de tarefas físicas. Para que operem de forma otimizada, necessitam de instruções claras e atualizadas;
- **Unidades de Monitorização**: Constantemente recolhem dados sobre o estado e desempenho dos sistemas físicos. A transmissão eficaz destes dados, em tempo real, para a *cloud* e outras unidades é vital para manter o sistema em condições ideais;
- **Unidades de Automação**: Dependem de uma comunicação robusta para receber instruções e enviar feedback. A automação eficaz só é possível quando as instruções são recebidas e implementadas sem erros nem atrasos;
- A comunicação das unidades com o servidor na *cloud* não só permite uma gestão centralizada dos diversos componentes, como também garante que os dados recolhidos sejam armazenados, analisados e utilizados para melhorar continuamente o sistema. O uso de protocolos IoT facilita esta comunicação, oferecendo uma plataforma unificada, independente da natureza específica de cada unidade ou dispositivo.

A comunicação entre as unidades e sistemas não é apenas uma funcionalidade, é uma necessidade intrínseca. Garantir que esta comunicação ocorra de forma eficaz e eficiente é essencial para a funcionalidade, otimização e evolução do sistema ciberfísico em questão.

3.3.4. Integração da arquitetura

O esquema apresentado na Figura 30 fornece uma perspetiva holística da estruturação do sistema ciberfísico. Através deste, é possível discernir claramente a composição intrínseca do sistema e as interligações vitais entre os seus componentes. Esta representação visa facilitar a compreensão da interação entre as partes computacional e física, bem como das interações externas com os utilizadores.

No âmbito do sistema físico, identificam-se três unidades principais: unidades de produção, unidade de automação e unidade de monitorização. Estas constituem os elementos ativos e sensoriais que materializam as operações e recolhem dados em tempo real.

Paralelamente, a *cloud*, parte computacional do sistema, é estruturada em torno de três componentes principais: servidor, base de dados e sistema de gestão. No servidor, diversas ferramentas, como a aplicação, UI, configurador, IDEs e dispositivos virtuais, convergem para garantir a eficácia das operações e a acessibilidade aos utilizadores. Este servidor é interconectado com o restante sistema computacional, permitindo a transferência de dados e informações, oriundas tanto das unidades físicas como dos utilizadores e do próprio sistema de gestão.

Em cenários tecnológicos contemporâneos, marcados por uma crescente interconexão de sistemas, a seleção criteriosa de protocolos de comunicação assume uma importância preponderante, visando garantir tanto a eficiência como a segurança nas interações entre dispositivos. Neste enquadramento, é crucial esclarecer as estratégias de comunicação adotadas por este sistema: as unidades físicas interagem entre si e com o servidor central através de protocolos especializados em IoT e M2M. Estes protocolos foram concebidos especificamente para atender às exigências de dispositivos interligados, assegurando trocas de informação ágeis e protegidas.

Por outro lado, quer a comunidade, quer o operador local, ao interagirem com o sistema, utilizam uma abordagem baseada em comunicações cliente-servidor. Enquanto a comunidade acede predominantemente através de uma interface de utilizador (UI) alojada na plataforma do servidor, o operador local usufrui de um acesso mais diversificado. A comunicação cliente-servidor garante uma transmissão de dados estruturada e robusta, proporcionando uma interação fluída e consistente para todos os envolvidos.

Especificamente, o operador local, para além de comunicar com o servidor, detém também a capacidade de intervir diretamente nas unidades físicas. Este nível de acesso oferece uma maior flexibilidade operacional, permitindo uma resposta ágil a eventuais contingências ou alterações necessárias no decurso das operações.

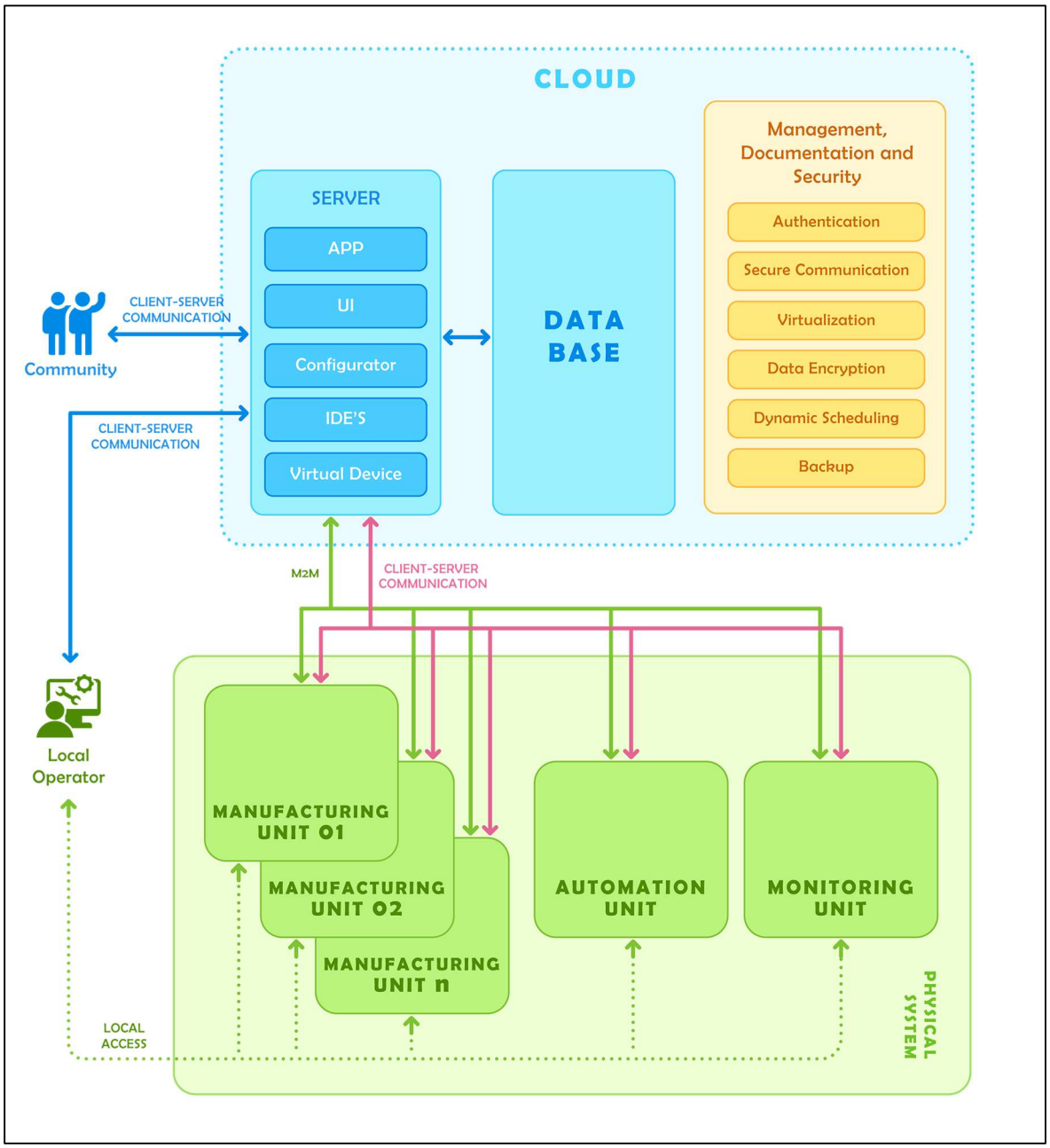


Figura 30 - Arquitetura do CPS proposta

4. IMPLEMENTAÇÃO DO PROTÓTIPO

Definido a arquitetura conceptual do sistema ciberfísico, de forma a se obter uma prova de conceito do funcionamento deste, é desenvolvido um protótipo de sistema de armazenamento e recolha automático (*Automatic Storage and Retrieval System* – **ASRS**).

A implementação está diretamente relacionada com os requisitos apresentados no capítulo anterior.

1. Sistemas de Controlo Versáteis e Abertos:

   O ASRS é alimentado por um software que encoraja a inovação e adaptação. Como uma plataforma aberta, convida à colaboração e contribuição da comunidade, garantindo uma base sólida para futuras evoluções.

2. Interoperabilidade Entre Sistemas:

   Visando um futuro onde os sistemas comunicam-se de forma transparente, o ASRS foi concebido para se integrar e dialogar com diversas plataformas, estabelecendo as bases para um ambiente ciberfísico coeso.

3. Componentes Modulares:

   O design modular do ASRS é fundamental para garantir sua adaptabilidade. Cada módulo pode ser visto como um bloco de construção, pronto para ser reconfigurado ou expandido à medida que a tecnologia e as necessidades evoluem.

4. Configurabilidade:

   Mesmo como uma prova de conceito, o ASRS foi projetado para ser flexível. Esta abordagem garante que podemos testar e redefinir parâmetros, aprendendo e otimizando para implementações futuras.

5. Protocolos de Comunicação e Interface Standard:

   A preparação para o futuro também implica em estabelecer padrões. Com protocolos e interfaces standard, o ASRS demonstra sua prontidão para se integrar em ecossistemas ciberfísicos mais amplos.

Ao centrar-se nos requisitos-chave desde esta fase inicial, o ASRS demonstra o que é possível desenvolver, enquanto abre a porta a novos desenvolvimentos.

4.1. Sistemas de Controlo Versáteis e Abertos

Para a implementação de um sistema ciberfísico como o armazém automatizado, a escolha dos sistemas de comando e controlo é tão crucial quanto a dos sistemas físicos. O comando e controlo não só comanda o funcionamento físico da operação, mas também oferece a interface para interação com o utilizador e possíveis sistemas externos. Nesta seção, exploramos as razões pelas quais foram escolhidos e como eles se alinham aos requisitos estabelecidos para o projeto.

1. *Marlin*: *Marlin* é um *firmware* de código aberto utilizado principalmente em impressoras 3D e CNC. Ele se destaca por sua capacidade de controlo refinado dos componentes de hardware, oferecendo uma ampla gama de configurações que são essenciais para garantir a precisão e eficiência necessárias em ambientes de armazém. A escolha do *Marlin* foi motivada não apenas por sua versatilidade, mas também pela sua natureza de código aberto. Isso permite modificações conforme necessário, proporcionando uma adaptabilidade que é essencial em projetos em evolução.
2. Código G: O código G é uma linguagem de programação amplamente adotada para controlar máquinas automatizadas. Ao optar por utilizar o código G, garante-se um nível de padronização, uma vez que é uma linguagem amplamente conhecida e adotada na indústria. Além disso, ao usar uma linguagem comum como o código G, a interoperabilidade é aumentada, permitindo uma possível integração com outros sistemas que também compreendem e interpretam o código G.
3. *Python*: *Python* é uma linguagem de programação de alto nível, conhecida por sua clareza e versatilidade. Sua adoção neste projeto se deu pela sua ampla base de bibliotecas e sua comunidade ativa, o que facilita a integração com diversos sistemas e componentes. Além disso, *Python*, sendo de código aberto, alinha-se ao requisito de ter um software de controlo versátil e aberto. A capacidade de modificar e adaptar o código às necessidades específicas do projeto, sem restrições de licenciamento, oferece uma vantagem significativa em termos de flexibilidade e escalabilidade.

Neste contexto, foram desenvolvidos dois softwares distintos para controlar o sistema:

1**. Sistema de gestão *offline***:

Este software foi desenvolvido utilizando o *Visual Studio Code* em *Python*, com uma interface de utilizador (UI) integrado. Ele é direcionado principalmente ao operador local e tem como objetivo principal a ligação direta ao armazém. Este software permite que o operador efetue trabalhos de melhoria, manutenção, calibração e resolução de problemas. Graças à sua interface intuitiva, as operações diárias e de manutenção são facilitadas.

2. **Sistema de gestão integrado**:

Visando a comunidade mais ampla de utilizadores, desenvolveu-se um segundo software, também baseado no *Visual Studio Code* em *Python*, mas com uma ênfase na comunicação IoT, especificamente utilizando o protocolo MQTT. Este software permite aos utilizadores remotos interagirem com o sistema ciberfísico e efetuarem operações. Para garantir uma experiência de

utilizador otimizada e amigável, integrou-se um programa *Node-RED*, proporcionando uma interface gráfica para os utilizadores remotos.

Complementar a estes softwares, foram desenvolvidos Códigos G para o controlo do armazém. Estes códigos representam um conjunto de instruções para controlar as operações e movimentos precisos do armazém, garantindo a eficiência e precisão na execução das tarefas.

O desenvolvimento destes softwares e códigos não apenas fortalece o sistema ciberfísico como um todo, mas também assegura que ele seja escalável, adaptável e capaz de atender a uma ampla variedade de necessidades e cenários operacionais.

4.1.1. Sistema de gestão *offline*

Este sistema tem como objetivo gerir um armazém automatizado, permitindo a carga e descarga de objetos. As principais características e funções do sistema são detalhadas a seguir:

1. **Inicialização do Sistema**:

O sistema verifica se existe uma ligação disponível com o hardware através da porta série 'COM6' a uma velocidade de transmissão de 250.000 bits por segundo. Se a ligação não estiver disponível, o sistema opera em modo de simulação.

2. **Comunicação com o sistema físico**:

O sistema envia comandos *G-Code* para o hardware através da ligação série. Estes comandos são utilizados para controlar o movimento do mecanismo de carga e descarga.

3. **Definição do Armazém**:

O armazém é definido por um número de filas e colunas. O espaço entre as prateleiras é representado em milímetros (mm) com uma altura de 90 mm e uma largura de 100 mm. A origem do armazém (posição 0,0) situa-se a 50 mm no eixo x e 10 mm no eixo y.

4. **Gestão da Base de Dados**:

O sistema utiliza uma base de dados em formato Excel denominada "*WarehouseDB3.xlsx*" para armazenar informações sobre a ocupação das prateleiras. Se a base de dados não existir, uma nova é criada com todas as células vazias.

5. **Interface Gráfica de Utilizador (GUI)**:

O GUI, presente na Figura 31, permite ao utilizador:

- Carregar objetos para o armazém, onde o nome do objeto é solicitado.
- Descarregar objetos com base no nome.

- Executar um processo de *homing* para calibrar o sistema.
- Testar os motores passo a passo ao inserir coordenadas x e y.
- Enviar comandos *G-Code* manualmente.

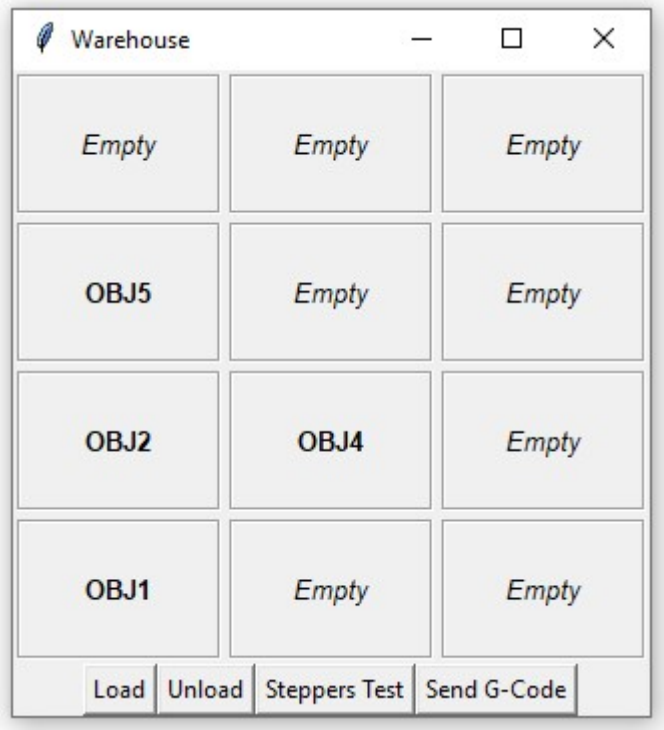


Figura 31 - Interface gráfico do sistema de gestão offline (não ligado ao equipamento)

6. **Processo de *Homing***:

Antes de carregar ou descarregar objetos, é necessário realizar um processo de *homing*. Este processo move o mecanismo para a sua posição inicial, garantindo que todos os movimentos subsequentes sejam precisos.

7. **Carregamento e Descarregamento**:

Ao carregar um objeto, o sistema procura o espaço vazio mais próximo do local de carga no armazém, move o mecanismo para essa posição, deposita o objeto e atualiza a base de dados. Ao descarregar, o sistema localiza o objeto na base de dados, move o mecanismo para a posição correspondente, retira o objeto, e deposita-o no local de carga/descarga e atualiza a base de dados.

Os componentes chave do programa são:

Importações (Bibliotecas *Python*):

- *tkinter* e suas subclasses: Usado para criar a interface gráfica;
- *pandas*: Para gerenciar e manipular os dados do armazém;

- *serial*: Para comunicar com o hardware do armazém;
- *openpyxl*: Um pacote para trabalhar com arquivos Excel, que é o formato escolhido para a base de dados do armazém;
- *datetime*: Possivelmente para adicionar *timestamps*.

Configuração Inicial:

O código define o tamanho do armazém em termos de linhas e colunas.

Há uma tentativa de estabelecer uma conexão serial com o hardware. Se a conexão falhar, o sistema funciona em modo de simulação.

Funções de Movimento e Controlo:

- *send_gcode*(): Envie um comando *G-Code* para o hardware via conexão serial;
- *homing_routine*(): Uma rotina de *homing* para calibrar o hardware antes de ser usado;
- *move_to_position*(): Move o hardware para uma posição especificada;
- *calculate_position*(): Calcula a posição física no armazém com base na linha e coluna.

Gestão da Base de Dados:

- *create_database*(): Cria uma base de dados inicial se não existir uma;
- *load_object*(): Carrega um objeto no armazém e atualiza a base de dados;
- *unload_object*(): Descarrega um objeto do armazém e atualiza a base de dados.

Interface Gráfica do Utilizador (GUI):

- *update_warehouse_display*(): Atualiza a visualização do armazém na GUI;
- *on_load_button_click*(), *on_unload_button_click*(), etc.: Funções de *callback* que são chamadas quando certos botões são pressionados na GUI;
- *main*(): A função principal que configura a GUI e entra no *loop* principal do evento.

O funcionamento do sistema de gestão *offline* pode ser visualizado de uma forma mais global na Figura 32 onde está representado num fluxograma.

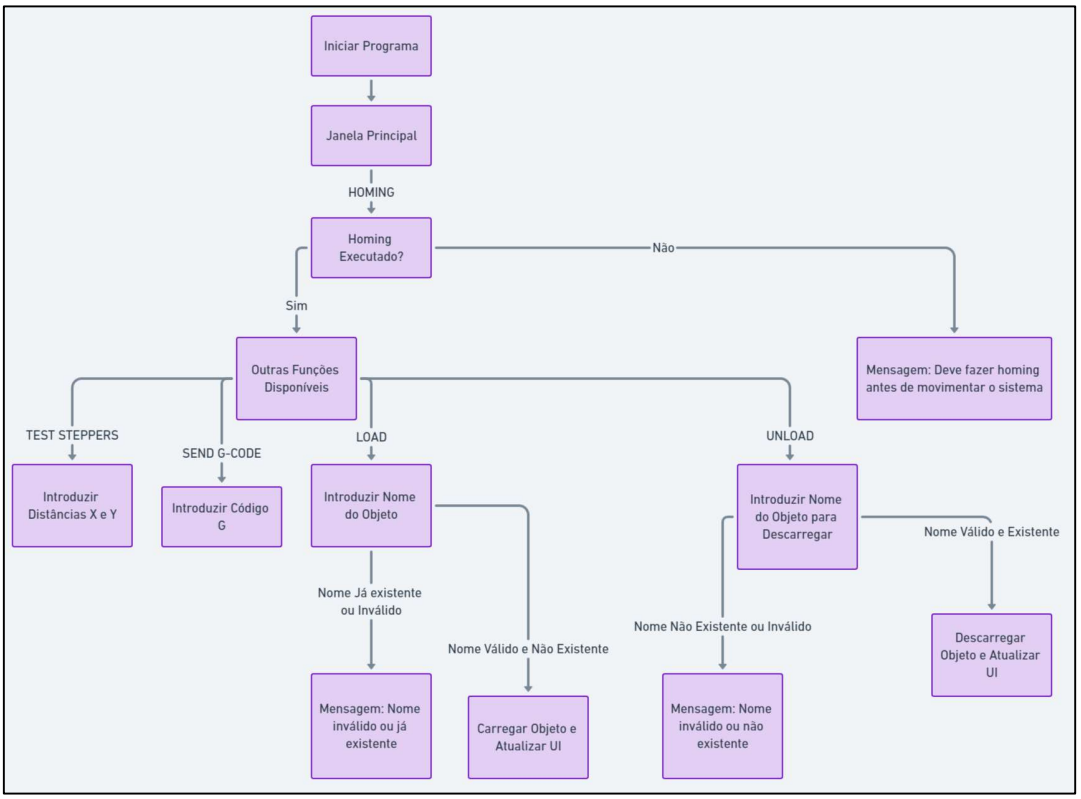


Figura 32 - Fluxograma do sistema de gestão offline

O código completo do sistema de gestão *offline* pode ser consultado no APÊNDICE A.

4.1.2. Sistema de gestão integrado

4.1.2.1. Aplicação do sistema

Este sistema está concebido para operar num armazém automatizado, utilizando um braço robótico e comunicando-se através de um *broker* MQTT. As principais características e funções do sistema são detalhadas a seguir:

1. **Inicialização do Sistema**:

Verifica-se uma conexão com o braço robótico através da porta serial. Caso esta conexão não seja estabelecida, o sistema opera em modo de simulação.

2. **Comunicação com o Hardware**:

O sistema utiliza comandos *G-Code* transmitidos via conexão serial para controlar os movimentos e ações do braço robótico.

3. **Definição do Armazém**:

O armazém é organizado em linhas e colunas. Cada célula pode conter um objeto. A função *calculate_position* é responsável por determinar as coordenadas destas células.

4. **Gestão da Base de Dados**:

O programa utiliza uma base de dados em formato Excel, onde se regista o inventário atual do armazém. Se o ficheiro Excel não existir, um novo é gerado com todas as células vazias.

5. **Comunicação MQTT**:

O sistema estabelece ligação com um broker MQTT. Há vários tópicos aos quais o sistema se subscreve, como "*INVENTORY_Q*", "*HOMING*", "*LOAD*" e "*UNLOAD*". Dependendo das mensagens recebidas, o sistema irá responder com ações específicas.

6. **Processo de *Homing***:

Através do tópico "*HOMING*", posiciona-se o braço robótico na sua posição inicial, assegurando que todos os movimentos subsequentes sejam precisos.

7. **Carregamento e Descarregamento**:

O sistema, ao receber instruções via MQTT, carrega ou descarrega objetos no armazém, atualizando posteriormente a base de dados.

Os componentes chave do programa são:

Importações (Bibliotecas Python):

- *paho.mqtt.client*: Biblioteca para estabelecer a comunicação via MQTT;
- *openpyxl*: Pacote utilizado para ler e escrever em arquivos Excel;
- *serial*: Para estabelecer a comunicação via porta serial com o braço robótico;
- *pandas*: Utilizado para a manipulação e análise dos dados do inventário;
- *datetime:* Possivelmente para adicionar *timestamps*.

Configuração Inicial:

Estabelece-se a conexão MQTT e tenta-se ligar ao braço robótico via conexão serial.

Funções de Movimento e Controlo:

- *send_gcode*(): Envie um comando *G-Code* para o braço robótico;

- *handle_homing*(): Uma rotina de *homing*;
- *handle_load*(): Carrega um objeto no armazém;
- *handle_unload*(): Descarrega um objeto do armazém.

Gestão da Base de Dados:

- *publish_inventory*(): Envia o estado atual do inventário.

Comunicação MQTT:

Funções específicas que lidam com diferentes tópicos MQTT e garantem a resposta adequada a cada mensagem recebida.

Função Principal:

- *main()*: Esta função configura a comunicação MQTT e entra num *loop*, aguardando por mensagens do broker.

O funcionamento do sistema de gestão integrado pode ser visualizado de uma forma mais global na Figura 33 onde está representado num fluxograma.

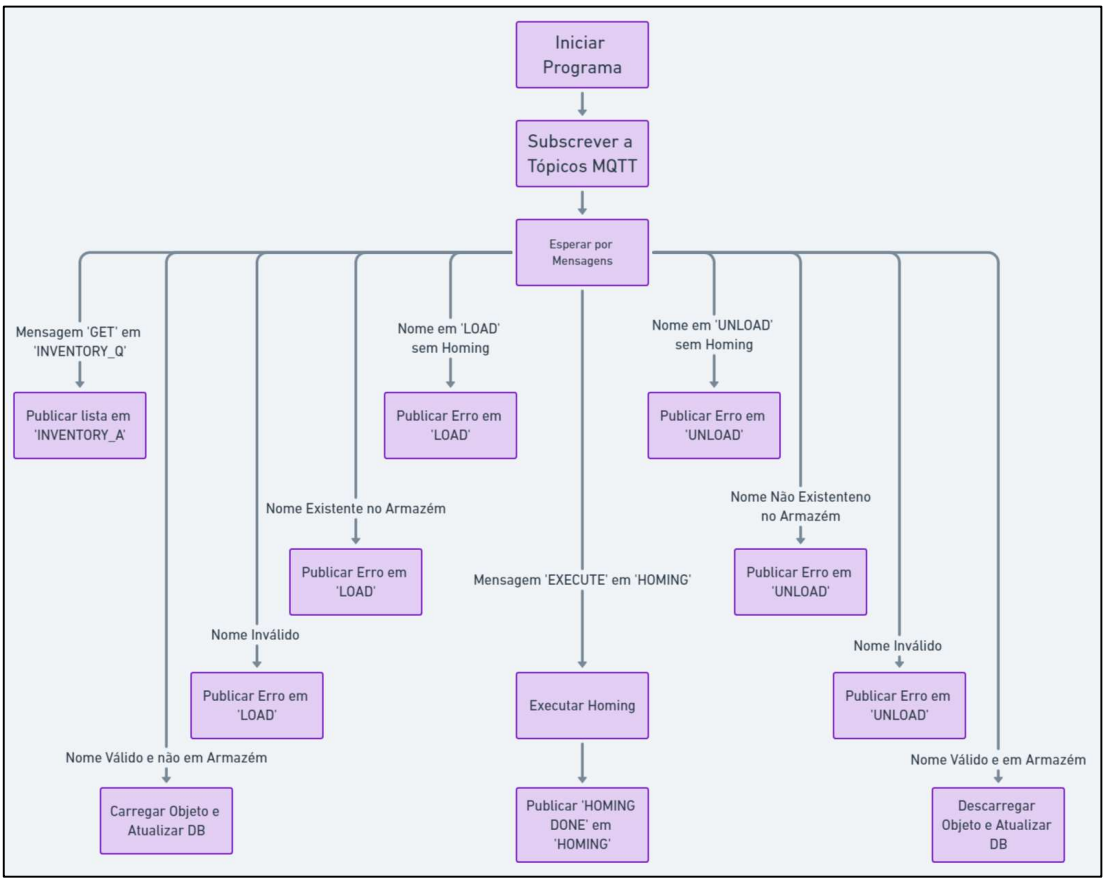


Figura 33 - Fluxograma do sistema de gestão integrado

O código completo do sistema de gestão integrado pode ser consultado no APÊNDICE B.

4.1.2.2. Interface do sistema

A servir de ponte entre a comunicação MQTT e a comunidade de utilizadores é usado então o *Node-RED* que é uma ferramenta de programação visual, baseada em fluxos, destinada ao desenvolvimento de aplicações interligadas para IoT.

Na Figura 34 é possível ver-se o interface gráfico desenvolvido e adaptado a uma janela de um *browser*.

Na secção de *Homing* tem apenas um botão que ao ser pressionado executa o ciclo de *homing*.

Na secção de *Inventory* tem um botão que ao ser pressionado pede uma resposta ao sistema para que seja devolvido o nome dos objetos em armazém assim como o número de lugares vazios. Essa informação aparece acima do botão na caixa de texto.

Na secção *Load* tem-se um botão, uma caixa de introdução de texto acima do botão e uma caixa de texto abaixo. A caixa de texto devolve mensagens de erro e a caixa de introdução de texto serva para definir o nome do objeto a carregar.

De forma equivalente a secção *Unload* tem as mesmas características de forma a escolher o nome do objeto a descarregar e a receber mensagens de erro do sistema.

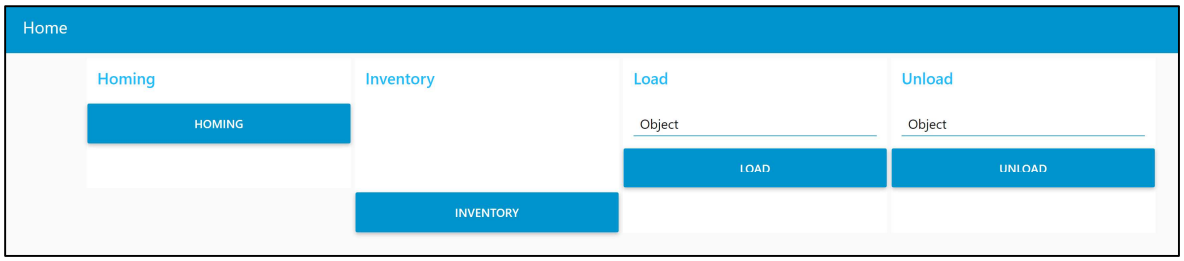


Figura 34 - Interface do utilizador no Node-Red num browser

De forma a se obter esta interface com estas funcionalidades usa-se o editor de fluxograma que funciona através de qualquer browser.
Com este editor é possível acrescentar e modificar módulos com as funcionalidades desejadas e interligar esses módulos para se obter a interface de utilizador pretendido.
Na Figura 35 pode-se ver um exemplo do ambiente de trabalho do Node-Red. O fluxograma da Figura 35 origina o UI da Figura 34.

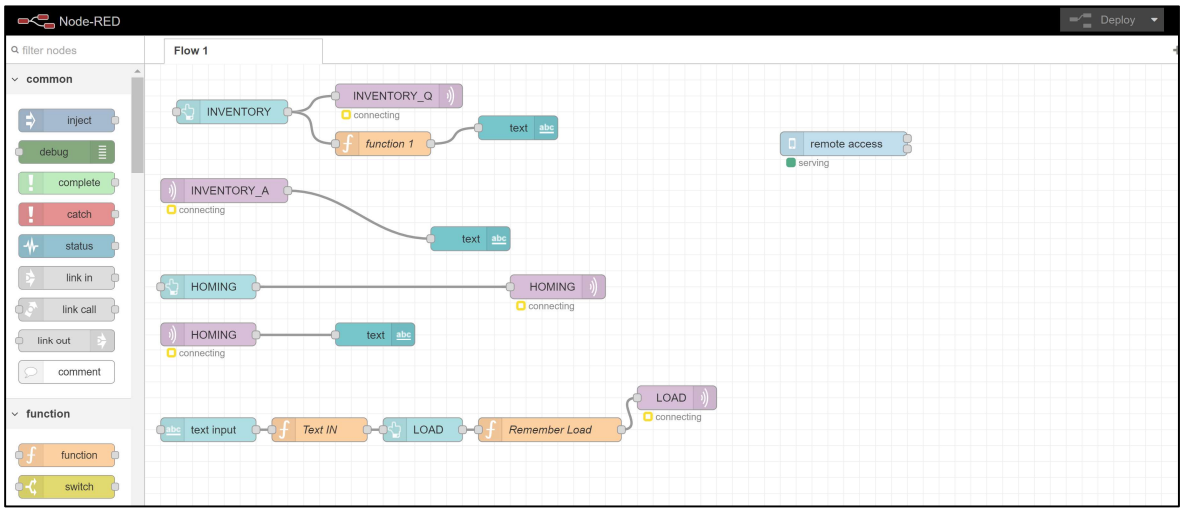


Figura 35 - Ambiente de trabalho do Node-Red

4.1.3. Sistema de controlo de hardware - Firmware

O *firmware Marlin* é frequentemente utilizado em projetos de impressão 3D e CNC pela sua versatilidade. Contudo, para responder às necessidades específicas deste sistema, realizam-se diversas alterações:

- Configuração para *RAMPS 1.4*: Define-se a utilização da placa *RAMPS 1.4* no *firmware*, garantindo que todas as portas e funcionalidades estejam adequadas para esta placa;
- Dois motores de passo: Configura-se a gestão de dois motores de passo de forma independente. Esta personalização torna-se crucial para assegurar a movimentação precisa do sistema;
- Implementação de três servomotores: Integram-se três servomotores, ou servos, ajustando-se as portas de saída e os ciclos de funcionamento para um controlo eficaz;
- Funcionamento dos fins de curso em X e em Y: Estabelece-se o funcionamento dos fins de curso nas direções X e Y, assegurando uma operação segura e dentro dos limites predefinidos.

4.2. Interoperabilidade Entre Sistemas

No âmbito da implementação e operacionalização de um armazém automatizado, a seleção de componentes físicos é uma parte importante que determina não apenas o desempenho do sistema, mas também a sua capacidade de se integrar com outros sistemas, garantindo interoperabilidade.

4.2.1. Construção Física

A estrutura física do ASRS é composta por vários componentes essenciais para o seu funcionamento. A visão global do armazém pode ser observada na Figura 36.

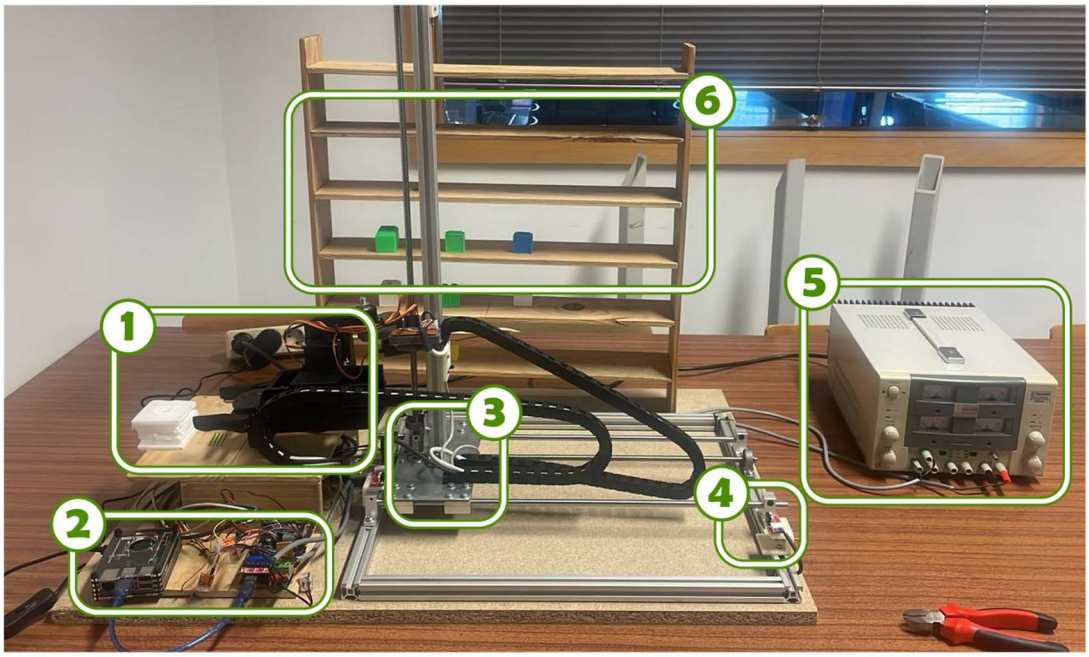


Figura 36 - Visão global do ASRS

A estrutura física do armazém inclui:

1. **Garra e zona de carga/descarga –** A garra é o componente que fisicamente carrega os objetos que são depositados e levantados do ASRS. Integralmente concebida através da técnica de impressão 3D. A zona de carga descarga é o local onde o objeto inicia a rotina de carga e termina a rotina de descarga: Na Figura 37 é possível ver em pormenor as três partes principais da garra:

    - 1.2 - O conjunto de pinças da garra, que se ajustam em torno dos objetos, baseia-se no design utilizado pelo projeto da célula de produção OSLab4Man. É usado um servomotor para controlar a abertura e fecho da garra;
    - 1.3 - A servir de base de fixação ao conjunto de pinças, encontra-se uma estrutura extensível que dá à garra a capacidade de alcançar os objetos quando estes estão no armazém. Este controlo da extensão da garra é também feito por um servomotor;

- 1.4 - Por fim, a base rotativa da garra garante que esta possa ser orientada tanto para o interior do armazém como para a zona de carga e descarga. O servomotor 1.4 é diferente dos outros dois. Mas a poderá ser aplicado em qualquer dos outros subsistemas e vice-versa.;

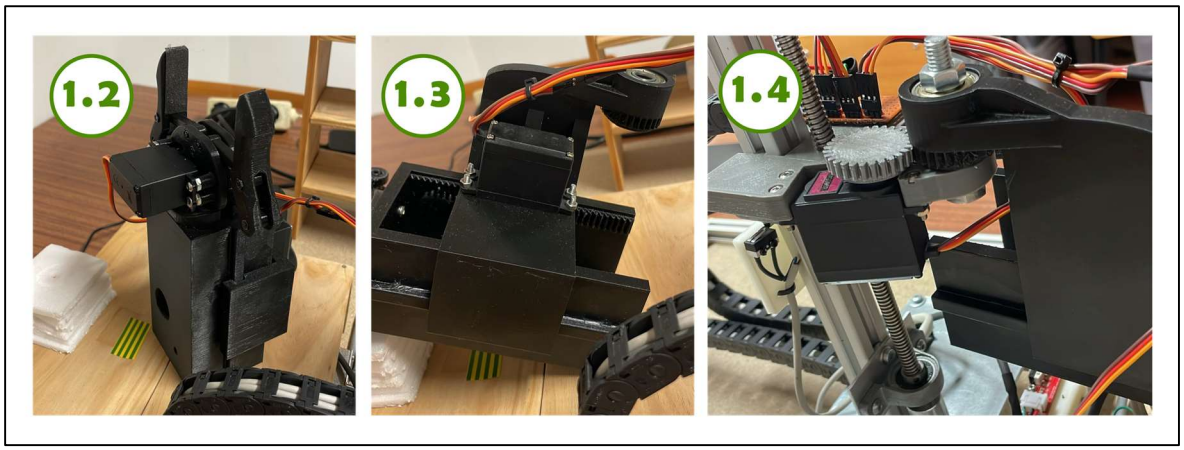


Figura 37 - Componentes da garra

2. Componentes de controlo e computação

Para uma operação eficaz do ASRS, é indispensável uma combinação dos sistemas atuadores com os de controlo. Com o objetivo de ilustrar os componentes de controlo e computação que fazem parte da arquitetura, a Figura 38 exibe o *Raspberry Pi*, o *Arduino Mega* e a *placa RAMPS 1.4* utilizados para esse efeito.

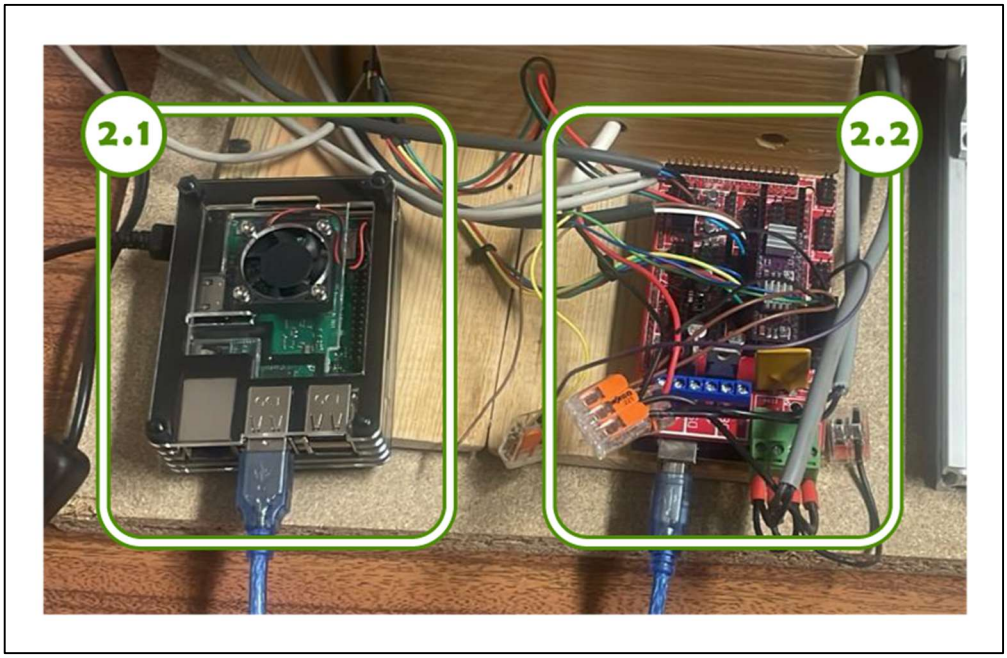


Figura 38 - Raspberry Pi, Arduino Mega e Ramps 1.4

2.1 ***Raspberry Pi***: Este microcomputador é o responsável por executar o software principal, gerenciando as operações do armazém e comunicando-se com o *Arduino* para controlar os componentes físicos. Como um dos microcomputadores mais populares, o *Raspberry Pi* oferece versatilidade e facilidade de integração, tornando-o uma escolha adequada para projetos que buscam interoperabilidade.

2.2 ***Arduino Mega***: É o cérebro do sistema de controlo, interpretando os comandos do software e transmitindo-os para o hardware apropriado. O uso do *Arduino Mega*, uma placa de controlo amplamente adotada, garante a possibilidade de integração com diversos sistemas e componentes, dada a sua vasta comunidade e suporte.

RAMPS 1.4: Esta placa serve como interface entre o *Arduino Mega* e os motores de passo. Esta placa, juntamente com os controladores de motores de passo, é frequentemente utilizada em impressoras 3D e outras máquinas automatizadas, destacando-se pela sua compatibilidade e adaptabilidade a diferentes sistemas.

3. Motores de passo

Estes motores passo a passo, ou *steppers*, são responsáveis pelo movimento do sistema ao longo dos eixos X e Y. Proporcionam precisão no posicionamento e são fundamentais para a localização e transporte de itens no armazém:

3.1 Motor do eixo X: A Figura 39 mostra o *stepper* responsável pelo movimento do sistema ao longo do eixo X. Este faz movimentar a base no qual está integrado o eixo Y.

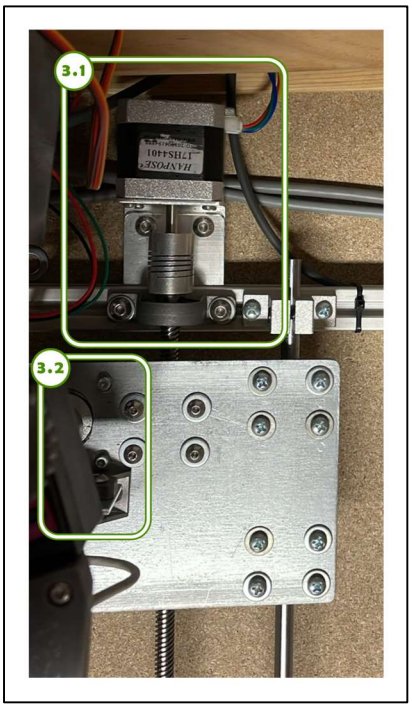


Figura 39 - Stepper do eixo X

3.2 Motor do eixo Y: A Figura 40 ilustra o *stepper* encarregado do movimento ao longo do eixo Y. Este eixo faz movimentar diretamente o subsistema da garra.

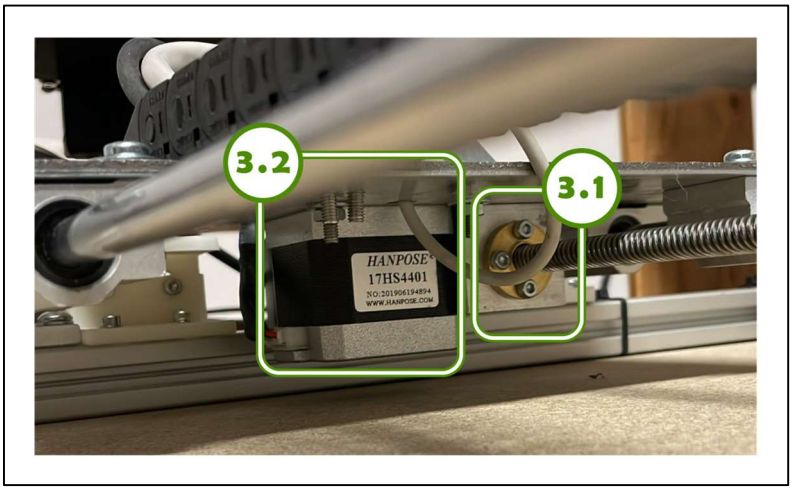


Figura 40 - Stepper do eixo Y

4. Fins de curso

Estes sensores são essenciais para o processo de *homing* e para garantir a segurança do sistema, prevenindo que os componentes móveis ultrapassem os seus limites de operação e se danifiquem. Na Figura 41 pode-se ver em pormenor o fim de curso utilizado para servir como segurança na movimentação no eixo X. Como se trata de um equipamento de segurança a ligação elétrica é feita no contacto normalmente fechado. Praticamente qualquer fim de curso de qualquer fabricante poderá ser aplicado para efetuar estas funções. É necessário apenas garantir a sua fixação no local mais apropriado.

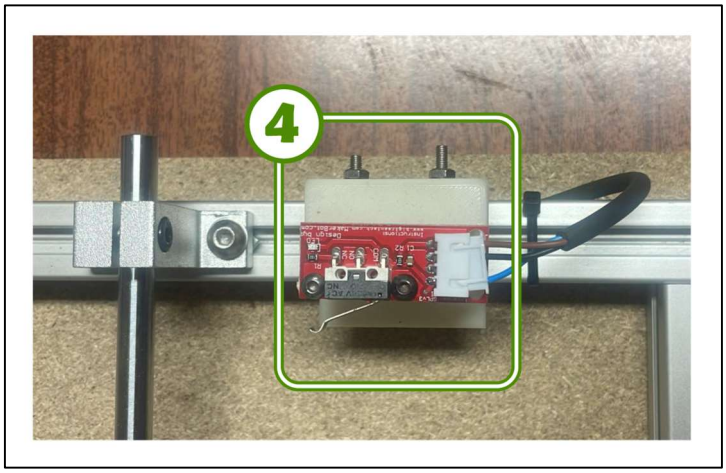


Figura 41 - Fim de curso X+

Os quatro sensores de fim de curso usados permitem implementar as seguintes funções:

- X-: Define o limite mínimo no eixo X;
- X+: Define o limite máximo no eixo X;
- Y-: Define o limite mínimo no eixo Y;
- Y+: Define o limite máximo no eixo Y.

5. Fonte de alimentação

A fonte de alimentação escolhida para este sistema é versátil e fundamental para o seu funcionamento, fornecendo simultaneamente 12V DC e 5V DC.

Os 12V DC têm um papel crucial, uma vez que alimentam o *Arduino* e a placa *RAMPS 1.4*. Esta última, por sua vez, garante a operacionalidade dos motores passo-a-passo. Adicionalmente, os 12V DC também são responsáveis pela ativação dos comandos dos sensores de fim de curso, assegurando que o sistema opere dentro dos seus limites predefinidos e evitando potenciais danos. Por outro lado, os 5V DC servem para alimentar os servomotores.

A alimentação também pode ser feita pelo *Arduino*. A Figura 42 mostra a fonte de alimentação em uso. Em 5.1 faz o fornecimento dos 5V DC e em 5.2 faz o fornecimento dos 5V DC diretamente aos servomotores. A Figura 43 mostra o pormenor de uma placa de ligações feita para a ligação dos servomotores. Esta placa possibilita que a substituição dos servomotores seja feita de forma mais simples.

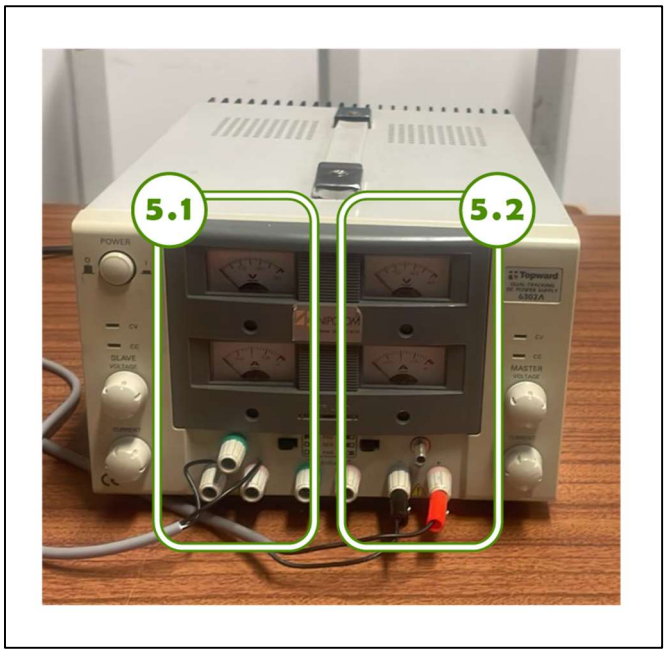


Figura 42 - Fonte de alimentação

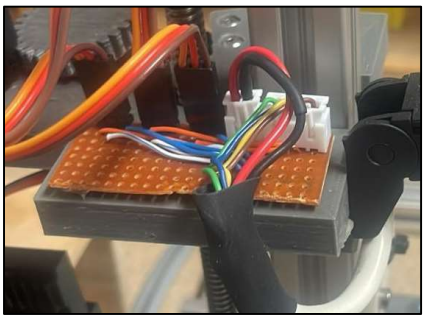


Figura 43 - Placa de ligação dos servomotores

O esquema elétrico que detalha as ligações necessárias para a conexão de todos os equipamentos pode ser consultado no APÊNDICE C.

6. Prateleiras do armazém

A Figura 36 ilustra também as prateleiras. A construção específica destas prateleiras não é importante, mas dever feita de forma que as suas dimensões sejam conhecidas para a configuração do sistema. As linhas das prateleiras devem ser paralelas, evitando assim possíveis obstáculos ou desalinhamentos durante as operações. Esta orientação geométrica assegura não só uma otimização do espaço de armazenamento, mas também contribui para a precisão e eficiência do sistema como um todo.

Na Figura 44 esquematiza-se o layout dos componentes do sistema físico.

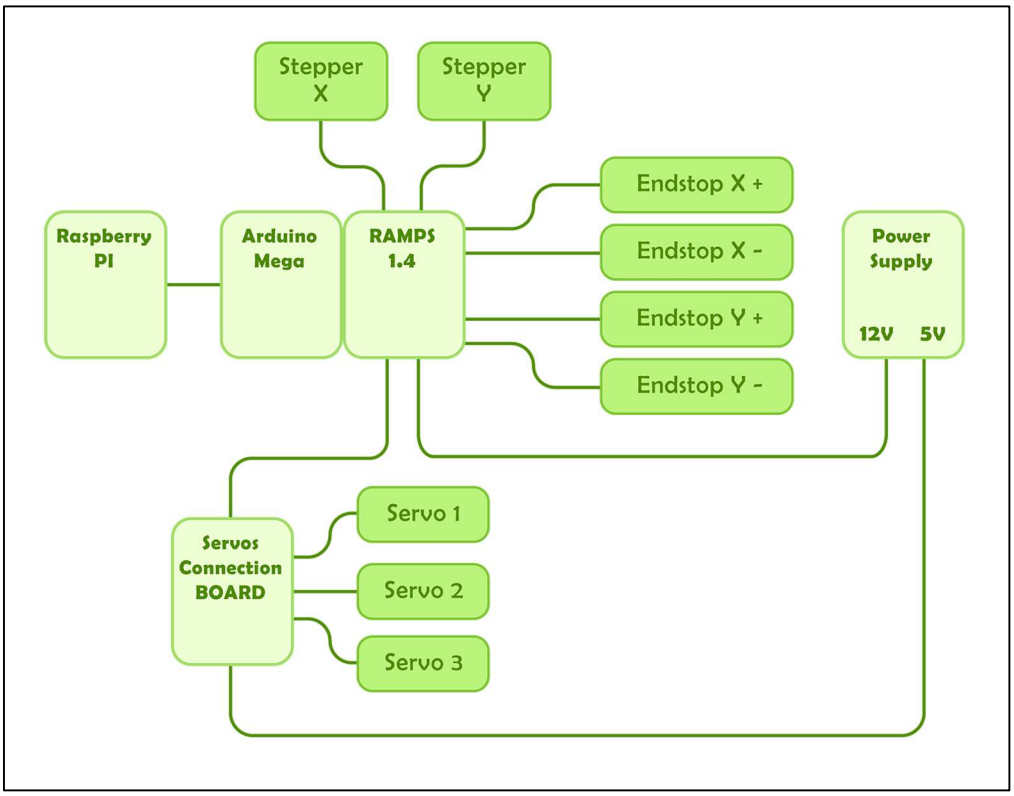


Figura 44 - Layout do sistema físico.

4.3. Componentes Modulares

A modularidade é um princípio fundamental na construção de sistemas ciberfísicos eficientes e flexíveis. A capacidade de intercambiar componentes - sejam eles físicos ou digitais - sem comprometer a funcionalidade geral do sistema, torna-o mais resiliente, adaptável e fácil de manter.

4.3.1. Modularidade digital

No domínio digital a modularidade traduz-se em blocos de código que definem estados e funções e que podem ser utilizados em qualquer outro programa que esteja a controlar equipamentos compatíveis.

A aplicação desenvolvida conta com módulos como, um módulo de conexão, módulo de dimensionamento do armazém ou módulo de posicionamento dos *steppers*. Estes podem ser omitidos, trocados, e intercambiados com módulos desta aplicação e de outras equivalentes. Na Figura 45 é representada esta modularidade na aplicação do sistema de controlo (*offline* e integrado).

Figura 45 - Compartimentalização modular da aplicação

Um exemplo destes módulos é a função '*homing()*' apresentada na Figura 46.

Este módulo define uma rotina de referência (*homing*) para um sistema. A função *homing_routine* inicializa a sequência de comandos de referência quando chamada.

Funcionalidades Principais:

- *show_homing_complete*(): Apresenta uma mensagem informando que a rotina de referência foi concluída;
- *homing_commands*(): Envia sequencialmente comandos *G-code* para a máquina, responsáveis pela operação de referência.

Sequência de Operações:

- São emitidos quatro comandos específicos através da função *send_gcode* para calibrar e posicionar a máquina;
- O código aguarda uma resposta da conexão após o comando "G28", garantindo que a rotina foi realizada. Uma vez confirmada a conclusão da rotina, é emitido o comando "M84" e, após um atraso de 3000 milissegundos, é mostrada a mensagem de conclusão.

Variável Global:

- *homing_done*: Esta variável global é definida como verdadeira após a conclusão da rotina de referência, indicando que o processo foi realizado com sucesso.

```python
# Define homing routine
def homing_routine(root):
    def show_homing_complete():
        messagebox.showinfo("Homing", "Homing routine completed. You can now load or unload objects.")

    def homing_commands():
        send_gcode("M280 P2 S0")
        send_gcode("M280 P0 S180")
        send_gcode("M280 P1 S0")
        send_gcode("G28")

        # Wait for the homing routine to complete.
        while True:
            response = connection.readline().decode()
            if response.strip() == "ok":
                break

        send_gcode("M84")
        root.after(3000, show_homing_complete)  # Delay the message display by 3000 milliseconds

    homing_commands()

    global homing_done
    homing_done = True
```

Figura 46 - Função de homing do ASRS

4.3.2. Modularidade física

Na esfera física, a modularidade é ainda mais tangível.

- ***Manufacturing Unit***: As unidades de produção são em si módulos que podem ser acrescentados, retirados ou substituídos conforme seja necessário;
- ***Local Computing Unit***: O *Raspberry P*i atua como a unidade de computação local, processando dados e tomando decisões em tempo real. Pode ser facilmente substituído por outra unidade computacional se necessário, devido à sua natureza *plug-and-play*;
- ***Controller***: O *Arduino*, em conjunto com o *RAMPS 1.4*, funciona como o controlador. Esta combinação pode ser intercambiada com outros controladores compatíveis, oferecendo flexibilidade e adaptabilidade ao sistema;
- ***Actuators***: Como é mostrado na Figura 43, devido à aplicação de elementos como a placa de ligações dos servomotores, estes podem ser substituídos ou trocados entre si no caso de uma falha ou numa tentativa de resolução de um problema.

Pode-se então ter uma situação em que o módulo garra se mostre ineficiente ou o tipo de objeto a transportar seja diferente, a solução poderá ser a sua substituição por um módulo eletroíman, uma vez que ambos os módulos exigem idênticos requisitos técnicos para integração e operacionalização, salvo mínimas configurações. Da mesma forma o sistema pode apenas funcionar num eixo caso o armazém seja uniaxial como é esquematizado na Figura 47.

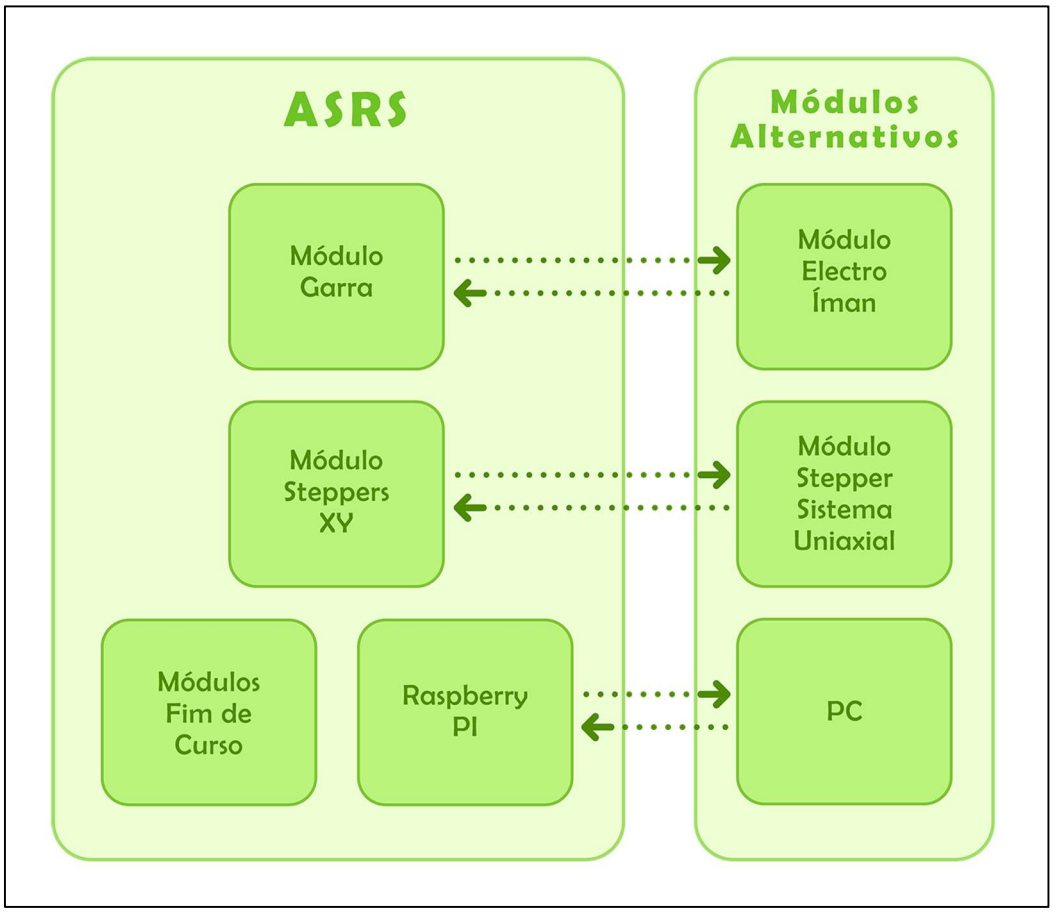


Figura 47 - Exemplo de troca de módulos no sistema físico

A modularidade física do sistema é evidente na forma como os componentes podem ser desconectados e reconectados sem comprometer a funcionalidade geral. Além disso, sistemas semelhantes com os mesmos componentes podem ceder módulos para este sistema e vice-versa.

4.4. Configurabilidade

A configuração é uma propriedade essencial em sistemas modernos, uma vez que permite uma maior adaptabilidade e evolução. No contexto do armazém, a configurabilidade é observada tanto no aspeto físico quanto no digital, permitindo assim que o sistema se adapte às mudanças nas necessidades ou à integração de novas tecnologias.

Configurabilidade Física:

- Componentes Comuns: Dada a utilização de componentes comuns como *steppers*, servomotores, e placas como o *Arduino* e *RAMPS 1.4*, a substituição de peças torna-se simples. Estes componentes estão amplamente disponíveis no mercado, o que facilita a sua aquisição e integração;

- Flexibilidade de Montagem: A estrutura do armazém pode ser facilmente ajustada para acomodar novos componentes ou adaptar-se a diferentes configurações, graças ao seu design modular.

Configurabilidade Digital:

- Software Aberto: O uso de softwares abertos como *Marlin* e *Python* permite que o código seja facilmente acedido e modificado. Esta abertura é fundamental para permitir adaptações rápidas às necessidades específicas do utilizador ou para integrar novas funcionalidades;
- Blocos de Código Independentes: O código é estruturado em funções distintas, permitindo que cada bloco tenha um propósito específico. Por exemplo, a função *homing*() pode ser facilmente adaptada ou expandida para incorporar novas lógicas ou responder a diferentes tipos de hardware;
- Integração de Novas Bibliotecas: Dada a natureza versátil do *Python*, novas bibliotecas ou *frameworks* podem ser integrados ao sistema para expandir suas capacidades ou melhorar a eficiência.

4.5. Protocolos de Comunicação e Interface Standard

A eficiência na comunicação entre os componentes de um sistema ciberfísico é crucial para assegurar um funcionamento adequado. No sistema de armazém em questão, adotam-se vários protocolos para garantir esta comunicação, desde conexões locais para configuração até protocolos mais robustos para comunicação remota.

Ligação Série Local:

- **Descrição**: Estabelece-se uma ligação ponto-a-ponto entre o *Raspberry Pi* e o *Arduino*;
- **Utilidade**: Permite configuração e diagnóstico rápido de problemas, garantindo uma comunicação direta e eficaz entre o controlador principal e o hardware do armazém.

Implementação do MQTT:

- **Utilidade**: Facilita a troca de mensagens em tempo real, especialmente útil em sistemas ciberfísicos distribuídos. Além disso, o MQTT suporta mensagens publicadas/subscritas, tornando-o ideal para ambientes onde múltiplos dispositivos necessitam de comunicar entre si;
- **Lógica do funcionamento**:
  - Tópico: *INVENTORY_Q*

    Mensagem: "*GET*"

Ação: Quando a mensagem "*GE*T" é recebida neste tópico, a função *publish_inventory(client)* é chamada. Esta função carrega os dados do armazém do arquivo Excel, conta o número de espaços vazios, obtém os nomes dos objetos no armazém, e depois publica uma mensagem no tópico *INVENTORY_A* com os nomes dos objetos e o número de espaços vazios;

- Tópico: *HOMING*

  Mensagem: "*EXECUTE*"

  Ação: Se a mensagem "*EXECUTE*" é recebida neste tópico, a sequência de *homing* (centralização) é iniciada usando a função *handle_homing(msg, client*). Uma série de comandos *G-Code* são enviados para executar a sequência de *homing*. Após a sequência de *homing* estar completa, uma mensagem "*HOMING DONE*" é publicada de volta no tópico *HOMING*;

- Tópico: *LOAD*

  Mensagem: Qualquer nome de item (texto arbitrário)

  Ação: Quando uma mensagem (nome do item) é recebida neste tópico, a função *handle_load(msg, client)* é chamada. O programa verifica se o item com esse nome já existe no armazém. Se existir, ele publica uma mensagem de erro no tópico LOAD. Se não houver espaço vazio no armazém, ele publica "*FULL*" no tópico LOAD. Se o *hardware* não estiver disponível ou o *homing* não tiver sido realizado, também publica mensagens de erro. Caso contrário, ele executa uma sequência para carregar o item no local vazio mais próximo no armazém;

- Tópico: *UNLOAD*

  Mensagem: Qualquer nome de item (texto arbitrário)

  Ação: Quando uma mensagem (nome do item) é recebida neste tópico, a função *handle_unload(msg, client)* é chamada. Se o item não estiver presente no armazém, publica uma mensagem de erro no tópico *UNLOAD*. Se o *hardware* não estiver disponível ou o *homing* não tiver sido realizado, também são publicadas mensagens de erro. Caso contrário, ele executa uma sequência para descarregar o item do armazém.

4.6. Integração

A integração de um sistema ciberfísico representa um momento fundamental na validação da operacionalidade e viabilidade do conceito. Aqui, procede-se à análise de como o protótipo espelha cada componente da arquitetura proposta.

Enquadramento Inicial:

- Considerando o carácter demonstrativo deste projeto, não se recorre à subscrição de serviços de *cloud computing* ou *cloud server*. Desta forma, centraliza-se a computação e o armazenamento a nível local;

- *Manufacturing Unit*: A unidade de produção desempenha um papel chave no contexto do sistema ciberfísico e engloba:
    - *Local Computing Unit*: O *Raspberry Pi* desempenha esta função, processando dados e coordenando operações que, em contextos convencionais, seriam a cargo de um serviço de *cloud*;
    - *Controller*: O *Arduino Mega*, em conjunto com o *shield RAMPS 1.4*, assegura a tradução das instruções da LCU em ações concretas, enviando comandos aos actuadores e recolhendo informações dos sensores;
- *Actuators & Sensors*: Os motores de passo, os servomotores e os fins de curso constituem esta categoria, responsáveis pela interação física com o meio envolvente;
- Computação e Armazenamento Local: Na ausência de um serviço de *cloud* externo, o *Raspberry Pi* encarrega-se das funções computacionais. Neste âmbito, destaca-se o seu papel em:
    - Servir de plataforma para a aplicação principal;
    - Alojar o servidor de interface de utilizador com recurso ao *Node-Red*;
    - Atuar como broker do protocolo MQTT, assegurando a comunicação entre dispositivos;
    - Armazenar a base de dados, garantindo o registo contínuo de operações e estados.

Concluindo, este exercício de integração demonstra a concretização da arquitetura de sistema ciberfísico proposta num protótipo operacional, onde cada componente desempenha uma função específica e essencial para a coesão do sistema.

4.7. Resultados

Nesta secção, apresentam-se os resultados alcançados no desenvolvimento do projeto, dividindo-se em duas categorias principais: o sistema de gestão e controlo e o sistema físico.

4.7.1. Sistema de gestão e controlo

No âmbito do sistema de gestão e controlo, há uma subdivisão que abrange o sistema de gestão offline e o sistema de gestão integrado. Estas categorias e subcategorias refletem os principais componentes e funcionalidades desenvolvidas, permitindo uma análise estruturada e detalhada das soluções implementadas e dos resultados obtidos.

4.7.1.1. Sistema de gestão *offline*

O sistema permite uma interação eficaz entre o utilizador e o hardware do armazém. Através da GUI, o utilizador pode carregar e descarregar objetos, visualizar o estado do armazém, enviar comandos *G-Code* e testar os motores.

A escolha de usar um arquivo Excel como base de dados é prática e fácil de implementar. No entanto, para um sistema de armazém em grande escala, um sistema de gestão de base de dados mais robusto poderia ser mais adequado.

A integração com a conexão serial e comandos *G-Code* sugere que o hardware é controlado através de um *firmware* baseado em *G-Code*, típico em CNCs e impressoras 3D.

O código também inclui vários mecanismos de segurança, como verificar se o processo de *homing* foi realizado antes de executar certos comandos, o que é crucial para evitar danos ao hardware ou à carga.

Seria benéfico adicionar comentários mais detalhados ou uma documentação, especialmente em torno das funções relacionadas com o hardware, para ajudar na manutenção e na compreensão do código por terceiros.

Em suma, este é um sistema de gestão de armazém bem pensado, com uma GUI amigável e mecanismos de interação eficazes com o *hardware* do armazém. A modularidade do código também facilita a expansão ou modificação do sistema no futuro.

4.7.1.2. Sistema de gestão integrado

O Sistema de Gestão Integrado demonstra uma combinação poderosa de tecnologias, permitindo uma integração completa e robusta entre diferentes plataformas e dispositivos. Graças à GUI, os utilizadores têm acesso a funcionalidades abrangentes, como a visualização em tempo real do inventário e o controlo do braço robótico.

A decisão de usar a biblioteca *paho.mqtt.client* revela uma tentativa de estabelecer uma comunicação baseada em MQTT. Esta escolha é estratégica para o IoT, pois o MQTT é um protocolo leve de mensagens que é ideal para dispositivos com recursos limitados e em ambientes de rede de baixa largura de banda.

Da mesma forma, a utilização da biblioteca serial sinaliza a capacidade do sistema de interagir diretamente com *hardware*, nomeadamente com o braço robótico. Isso indica uma integração direta com equipamentos, proporcionando maior controlo e precisão nas operações do armazém.

A opção por manter o armazenamento de dados em um arquivo Excel, embora prático, pode ser limitado para um sistema de armazém em grande escala. Uma recomendação seria avaliar a migração para um sistema de gestão de base de dados mais robusto à medida que o sistema cresce e se torna mais complexo.

No que diz respeito à segurança, é fundamental garantir que todos os comandos enviados para o braço robótico sejam validados para evitar movimentos inesperados ou potencialmente perigosos. Além disso, a integração de sistemas de autenticação e autorização fortaleceria a segurança do sistema contra acessos não autorizados.

Assim como mencionado na análise anterior, seria proveitoso introduzir uma documentação mais extensa e comentários detalhados para facilitar a manutenção e compreensão do código por terceiros.

Em resumo, o Sistema de Gestão Integrado é uma solução avançada que aproveita as tecnologias atuais para oferecer uma experiência de gestão de armazém integrada e dinâmica. As capacidades de interação em tempo real e a modularidade do código garantem que o sistema seja escalável e adaptável às futuras necessidades.

4.7.2. Sistema físico

O hardware, como complemento crucial ao software, desempenha um papel fundamental na configuração e funcionamento de um sistema ciberfísico. A integração e interação entre diferentes componentes determinam a eficácia global do sistema. Nesta secção, procede-se à análise detalhada dos componentes de hardware utilizados, incluindo as suas vantagens e limitações.

1. ***Raspberry Pi* (*Local Computing Unit*):**

A opção pelo *Raspberry P*i deve-se à sua versatilidade, baixo custo e vasta comunidade de apoio. Sendo capaz de processar dados e alojar diversas aplicações em simultâneo, mostra-se como uma solução adequada para as necessidades do projeto.

Vantagens:

- Custo-eficácia: O *Raspberry Pi* apresenta uma excelente relação preço/desempenho;
- Flexibilidade: Suporta várias linguagens de programação e sistemas operativos;
- Comunidade: Uma ampla comunidade de utilizadores auxilia na resolução de problemas.

Limitações:

- Desempenho: Em comparação com soluções de computação mais robustas, apresenta limitações em tarefas intensivas;
- Expansibilidade: As possibilidades de expansão de hardware são limitadas.

2. ***Arduino Mega* com *RAMPS 1.4* (*Controller*):**

A combinação *Arduino Mega* e *RAMPS 1.4* oferece uma base estável e testada para o controlo dos atuadores e a leitura dos sensores.

Vantagens:

- Modularidade: O *RAMPS 1.4* permite uma fácil substituição e adição de componentes;
- Compatibilidade: Existe uma grande variedade de periféricos e módulos desenvolvidos especificamente para o *Arduino*;

- Programabilidade: O ambiente de desenvolvimento do *Arduino* é acessível e bem documentado.

Limitações:

- Desempenho: Em determinadas tarefas, o *Arduino Mega* pode mostrar-se menos capaz do que outras soluções de microcontroladores mais recentes;
- Consumo energético: Em aplicações de baixo consumo, outras soluções poderiam ser mais eficientes.
- O *Shield RAMPS 1.4* não permite a ligação de mais de cinco motores de passo e limita a integração direta de fins de curso a três eixos.

3. **Atuadores e Sensores:**

Motores de passo, servo motores e fins de curso materializam a interface direta com o ambiente físico.

Vantagens:

- Precisão: Motores de passo asseguram uma movimentação precisa;
- *Feedback*: Fins de curso garantem que os movimentos ocorram dentro dos limites estabelecidos;
- Versatilidade: Servomotores são adequados para uma variedade de aplicações em que o controlo da amplitude de movimentos sejam necessário.

Limitações:

- Desgaste: Componentes físicos estão sujeitos a desgaste com o uso contínuo.
- Calibração: Requerem calibrações periódicas para manter a precisão.

5. CONCLUSÃO

No decorrer deste estudo, identifica-se o alcance de vários objetivos previamente estabelecidos, bem como a existência de áreas que necessitam de um aprofundamento ou de uma abordagem distinta. Assim, a conclusão desta dissertação está estruturada em dois subcapítulos:

- **Conclusões Finais**: Este subcapítulo reflete sobre os resultados e descobertas principais alcançados ao longo da pesquisa. Os objetivos iniciais são revisitados, discutindo-se o grau de cumprimento dos mesmos e as implicações práticas e teóricas das contribuições apresentadas.
- **Limitações e Trabalhos Futuros**: Nesta seção, são reconhecidas as limitações identificadas durante o desenvolvimento do trabalho, quer sejam inerentes à metodologia adotada, aos recursos disponíveis ou a outros fatores pertinentes. Adicionalmente, são identificadas oportunidades para pesquisa e desenvolvimento subsequente, propondo-se abordagens e áreas que podem ser exploradas para aprimorar ou expandir o que foi aqui delineado.

5.1. Conclusões finais

Ao desenvolver sistemas ciberfísicos com uma perspetiva de *Open Design*, a principal meta é democratizar e inovar o processo de construção. Esta abordagem não só permite uma participação mais abrangente da comunidade, mas também viabiliza um sistema ciberfísico mais robusto e adaptável.

Benefícios do Desenvolvimento de Sistemas Ciberfísicos com *Open Design*:

- **Colaboração Aumentada**: A abertura do design promove que especialistas de diferentes áreas contribuam para o desenvolvimento conjunto, maximizando as funcionalidades;
- **Adaptabilidade**: Estes sistemas demonstram flexibilidade em face de novas necessidades ou contextos, potenciando uma evolução constante;
- **Modularidade Reforçada**: A integração do *Open Design* nos sistemas ciberfísicos favorece uma estrutura modular, facilitando o desenvolvimento autónomo de componentes.

A respeito dos objetivos deste trabalho:

- **Revisão do Estado da Arte**: Realizou-se uma análise abrangente relacionada a sistemas ciberfísicos e ao *Open Design*;
- **Modelo Definido**: Estabeleceu-se um modelo que incorpora ambos os campos de estudo.
- **Requisitos e Arquitetura**: Foram analisados os requisitos necessários segundo o modelo estabelecido, definindo uma arquitetura de solução apropriada;
- **Protótipo**: Implementou-se um protótipo funcional, servindo como prova de conceito da solução proposta;
- **Lacunas e Trabalhos Futuros**: Identificaram-se algumas áreas que necessitam de investigação adicional, apontando direções para estudos subsequentes.

Concluindo, a abordagem de *Open Design* no desenvolvimento de sistemas ciberfísicos oferece uma combinação poderosa. Através desta integração, não só se realça a inovação e adaptabilidade do sistema, mas também se solidifica o conceito como uma ferramenta acessível e democrática, alinhada com as necessidades modernas e emergentes.

5.2. Limitações e trabalhos futuros

Apesar dos avanços e benefícios alcançados com a integração de sistemas ciberfísicos e *Open Design*, é fundamental reconhecer os desafios existentes e identificar oportunidades de pesquisa e desenvolvimento futuros. O trabalho desenvolvido revelou algumas limitações, assim como direções promissoras para explorar em trabalhos subsequentes.

Limitações:

- **Maturidade Tecnológica**: Embora os sistemas ciberfísicos estejam em ascensão, a sua maturidade tecnológica, quando combinada com a abordagem *Open Design*, ainda pode ser um obstáculo, principalmente em contextos industriais mais tradicionais;
- **Gestão de Dados**: O fluxo contínuo de dados, característico de sistemas ciberfísicos, pode ser desafiante para manter e gerir, sobretudo quando se procura transparência e abertura;
- **Interoperabilidade**: A integração de diferentes sistemas e módulos de várias fontes, especialmente em um contexto aberto, pode resultar em desafios de compatibilidade.

Trabalhos Futuros:

- **Integração:** Complementar o trabalho do ASRS e do OSLab4Man integrando-o e construir a célula de produção idealizada;
- **Normas e Protocolos**: Estabelecer normas e protocolos claros para garantir que os contributos em *Open Design* sejam não apenas inovadores, mas também consistentes e compatíveis com os sistemas ciberfísicos existentes;
- **Segurança Avançada**: Desenvolver métodos mais robustos de segurança que protejam tanto a integridade dos sistemas ciberfísicos como os contributos em ambiente aberto;
- **Comunicação entre sistemas:** Sugere-se a investigação de comunicação síncrona versus assíncrona em sistemas ciberfísicos, focando no impacto da latência e eficiência, bem como na otimização de transações em ambientes de alta latência e sua integração segura em sistemas distribuídos;
- **Manual de construção e operação**: Elaborar um manual que detalhadamente especifique todo o processo de construção, programação e configuração de um sistema;
- **Teste e Validação**: Realizar mais estudos e testes no campo real para validar a eficácia, eficiência e robustez de sistemas ciberfísicos desenvolvidos sob a ótica do *Open Design*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] E. A. Lee, “CPS foundations,” in *Design Automation Conference*, 2010, pp. 737–742, doi: 10.1145/1837274.1837462.

[2] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, “Review on cyber-physical systems,” *IEEE/CAA J. Autom. Sin.*, vol. 4, no. 1, pp. 27–40, 2017, doi: 10.1109/JAS.2017.7510349.

[3] X. Guan, B. Yang, C. Chen, W. Dai, and Y. Wang, “A comprehensive overview of cyber-physical systems: From perspective of feedback system,” *IEEE/CAA J. Autom. Sin.*, vol. 3, no. 1, pp. 1–14, 2016, doi: 10.1109/JAS.2016.7373766.

[4] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, “Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison,” *Engineering*, vol. 5, no. 4, pp. 653–661, 2019, doi: 10.1016/j.eng.2019.01.014.

[5] A. J. C. Trappey, C. V. Trappey, U. H. Govindarajan, J. J. Sun, and A. C. Chuang, “A Review of Technology Standards and Patent Portfolios for Enabling Cyber-Physical Systems in Advanced Manufacturing,” *IEEE Access*, vol. 4, pp. 7356–7382, 2016, doi: 10.1109/ACCESS.2016.2619360.

[6] E. A. Lee, “Cyber physical systems: Design challenges,” *Proc. - 11th IEEE Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput. ISORC 2008*, pp. 363–369, 2008, doi: 10.1109/ISORC.2008.25.

[7] G. D. Putnik, L. Ferreira, N. Lopes, and Z. Putnik, “What is a Cyber-Physical System: Definitions and models spectrum,” *FME Trans.*, vol. 47, no. 4, pp. 663–674, 2019, doi: 10.5937/fmet1904663P.

[8] S. K. Khaitan and J. D. McCalley, “Design techniques and applications of cyberphysical systems: A survey,” *IEEE Syst. J.*, vol. 9, no. 2, pp. 350–365, 2015, doi: 10.1109/JSYST.2014.2322503.

[9] S. Sousa, E. Nunes, and I. Lopes, “Measuring and managing operational risk in industrial processes,” *FME Trans.*, vol. 43, no. 4, pp. 295–302, 2015, doi: 10.5937/fmet1504295S.

[10] S. C. Suh, U. J. Tanik, J. N. Carbone, and A. Eroglu, “Applied cyber-physical systems,” *Appl. Cyber-Physical Syst.*, vol. 9781461473, no. January 2013, pp. 1–253, 2014, doi: 10.1007/978-1-4614-7336-7.

[11] V. Dhawan, K. Debnath, I. Singh, and S. Singh, “Prediction of forces during drilling of composite laminates using artificial neural network: A new approach,” *FME Trans.*, vol. 44, no. 1, pp. 36–42, 2016, doi: 10.5937/fmet1601036D.

[12] G. D. Putnik, V. Shah, Z. Putnik, and L. Ferreira, “Machine learning in cyber-physical systems and manufacturing singularity – it does not mean total automation, human is still in the centre Part ii – in-cps and a view from community on industry 4.0 impact on society,” *J. Mach. Eng.*, vol. 21, no. 1, pp. 133–153, 2021, doi: 10.36897/jme/134245.

[13] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney, “Task scheduling for control oriented requirements for cyber-physical systems,” *Proc. - Real-Time Syst. Symp.*, pp. 47–56, 2008, doi: 10.1109/RTSS.2008.52.

[14] Q. Pan, J. Wu, X. Lin, and J. Li, “Side-Channel Analysis-Based Model Extraction on Intelligent CPS: An Information Theory Perspective,” *Proc. - IEEE Congr. Cybermatics 2021 IEEE Int. Conf. Internet Things, iThings 2021, IEEE Green Comput. Commun. GreenCom 2021, IEEE Cyber, Phys. Soc. Comput. CPSCom 2021 IEEE Smart Data, SmartD*, pp. 254–261, 2021, doi:

10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00050.

[15] V. V. Ractham and P. Kantamara, “Single-loop vs. double-loop learning: an obstacle or a success factor for organizational learning,” *Edulearn11 3rd Int. Conf. Educ. New Learn. Technol.*, vol. 2, no. 7, pp. 1586–1591, 2011.

[16] Y. Yin, P. Zheng, C. Li, and L. Wang, “A State-of-the-Art Survey on Augmented Reality-Assisted Digital Twin for Futuristic Human-Centric Industry Transformation,” *Robot. Comput. Integr. Manuf.*, vol. 81, no. 102515, pp. 1–21, 2023, doi: 10.1016/j.rcim.2022.102515.

[17] R. T. Azuma, “A Survey of Augmented Reality,” *Presence Teleoperators Virtual Environ.*, vol. 6, no. 4, pp. 355–385, Aug. 1997, doi: 10.1162/pres.1997.6.4.355.

[18] C. M. Lukman Khalid, M. S. Fathi, and Z. Mohamed, “Integration of cyber-physical systems technology with augmented reality in the pre-construction stage,” in *2014 2nd International Conference on Technology, Informatics, Management, Engineering & Environment*, 2014, pp. 151–156, doi: 10.1109/TIME-E.2014.7011609.

[19] J. Lin, S. Sedigh, and A. Miller, “Modeling cyber-physical systems with semantic agents,” *Proc. - Int. Comput. Softw. Appl. Conf.*, no. September, pp. 13–18, 2010, doi: 10.1109/COMPSACW.2010.13.

[20] Y. P. Tsang, T. Yang, Z. S. Chen, C. H. Wu, and K. H. Tan, “How is extended reality bridging human and cyber-physical systems in the IoT-empowered logistics and supply chain management?,” *Internet of Things (Netherlands)*, vol. 20, no. August, 2022, doi: 10.1016/j.iot.2022.100623.

[21] S. Adwernat, M. Wolf, and D. Gerhard, “Optimizing the Design Review Process for Cyber-Physical Systems using Virtual Reality,” *Procedia CIRP*, vol. 91, pp. 710–715, 2020, doi: 10.1016/j.procir.2020.03.115.

[22] F. Hu *et al.*, “Cyberphysical System with Virtual Reality for Intelligent Motion Recognition and Training,” *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 2, pp. 347–363, 2017, doi: 10.1109/TSMC.2016.2560127.

[23] L. Ferreira *et al.*, “Disruptive data visualization towards zero-defects diagnostics,” *Procedia CIRP*, vol. 67, pp. 374–379, 2018, doi: https://doi.org/10.1016/j.procir.2017.12.270.

[24] M. C. Bujorianu, M. L. Bujorianu, and H. Barringer, “A Formal Framework for User-centric Control of Multi-Agent Cyber-physical Systems Marius Bujorianu , Manuela Bujorianu and Howard Barringer February 2009 Manchester Institute for Mathematical Sciences School of Mathematics The University of Manchester A F,” *Secretary*, no. February, 2009, [Online]. Available: http://www.manchester.ac.uk/mims/eprints.

[25] C. Talcott, “Cyber-physical systems and events,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5380 LNCS, pp. 101–115, 2008, doi: 10.1007/978-3-540-89437-7_6.

[26] M. C. Bujorianu and H. Barringer, “An integrated specification logic for cyber-physical systems,” *Proc. IEEE Int. Conf. Eng. Complex Comput. Syst. ICECCS*, pp. 291–300, 2009, doi: 10.1109/ICECCS.2009.36.

[27] C. Hang, P. Manolios, and V. Papavasileiou, “Synthesizing cyber-physical architectural models with real-time constraints,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6806 LNCS, pp. 441–456, 2011, doi: 10.1007/978-3-642-22110-1_35.

[28] A. Rajhans *et al.*, “An architectural approach to the design and analysis of cyber-physical

systems," *Electron. Commun. EASST*, vol. 21, 2009, doi: 10.14279/tuj.eceasst.21.286.278.

[29] D. Goswami, R. Schneider, and S. Chakraborty, "Co-design of cyber-physical systems via controllers with flexible delay constraints," *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC*, pp. 225–230, 2011, doi: 10.1109/ASPDAC.2011.5722188.

[30] B. Miller, F. VahicK, and T. Givargis, "Application-specific codesign platform generation for digital mockups in cyber-physical systems," in *2011 Electronic System Level Synthesis Conference (ESLsyn)*, 2011, pp. 1–6, doi: 10.1109/ESLsyn.2011.5952295.

[31] É. Boisseau, J.-F. Omhover, and C. Bouchard, "Open-design: A state of the art review," *Des. Sci.*, vol. 4, 2018.

[32] M. Avital, "The generative bedrock of open design," *Open Des. now Why Des. cannot Remain Exclus.*, no. August, pp. 48–58, 2011.

[33] H. Castro, G. Putnik, A. Castro, and R. D. Bosco Fontana, "Open Design initiatives: an evaluation of CAD Open Source Software," *Procedia CIRP*, vol. 84, pp. 1116–1119, 2019, doi: https://doi.org/10.1016/j.procir.2019.08.001.

[34] E. K. R. E. Huizingh, "Open innovation : State of the art and future perspectives," *Technovation*, vol. 31, no. 1, pp. 2–9, 2011, doi: 10.1016/j.technovation.2010.10.002.

[35] H. Chesbrough, W. Vanhaverbeke, and J. West, "Open innovation," *new Imp. Creat. Profit. from Technol.*, vol. 1, 2006.

[36] T. Obradović, B. Vlačić, and M. Dabić, "Open innovation in the manufacturing industry: A review and research agenda," *Technovation*, vol. 102, p. 102221, 2021, doi: https://doi.org/10.1016/j.technovation.2021.102221.

[37] D. Chiaroni, V. Chiesa, and F. Frattini, "The Open Innovation Journey: How firms dynamically implement the emerging innovation management paradigm," *Technovation*, vol. 31, no. 1, pp. 34–43, 2011, doi: https://doi.org/10.1016/j.technovation.2009.08.007.

[38] M. Greco, G. Locatelli, and S. Lisi, "Open innovation in the power & energy sector: Bringing together government policies, companies' interests, and academic essence," *Energy Policy*, vol. 104, pp. 316–324, 2017, doi: https://doi.org/10.1016/j.enpol.2017.01.049.

[39] H. Castro, F. Costa, L. Ferreira, P. Ávila, G. D. Putnik, and M. Cruz-Cunha, "Data Science for Industry 4.0: A Literature Review on Open Design Approach," *Procedia Comput. Sci.*, vol. 204, pp. 877–884, 2022, doi: https://doi.org/10.1016/j.procs.2022.08.106.

[40] H. Castro, G. Putnik, A. Castro, and R. D. B. Fontana, "Could Open Design learn from Wikipedia?," *Procedia CIRP*, vol. 84, pp. 1112–1115, 2019, doi: https://doi.org/10.1016/j.procir.2019.07.001.

[41] R. Vallance, S. Kiani, and S. Nayfeh, "Open design of manufacturing equipment," in *Proceedings of the CHIRP 1st International Conference on Agile, Reconfigurable Manufacturing*, 2001, no. January 2001, pp. 33–43.

[42] J. Tooze *et al.*, "Open design: contributions, solutions, processes and projects," *Des. J.*, vol. 17, no. 4, pp. 538–559, 2014.

[43] A. D. Demirbilek and O. Mengi, "Open Design: Exploring the Use of Open Knowledge in Service Design," in *European Conference on Knowledge Management*, 2022, vol. 23, no. 1, pp. 315–322.

[44] H. Castro *et al.*, "Cyber-Physical Systems using Open Design: an approach towards an Open Science Lab for Manufacturing," *Procedia Comput. Sci.*, vol. 196, pp. 381–388, 2022, doi: https://doi.org/10.1016/j.procs.2021.12.027.

[45] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” *Int. J. Inf. Manage.*, vol. 35, no. 2, pp. 137–144, 2015, doi: https://doi.org/10.1016/j.ijinfomgt.2014.10.007.

[46] S. S. Vadodaria, E. Warner, I. Norton, and T. B. Mills, “Design data for the 3D printer modification to print gels and pastes and the corresponding firmware,” *Data Br.*, vol. 36, p. 106974, 2021, doi: 10.1016/j.dib.2021.106974.

[47] F. Iqbal, M. Gohar, H. Alquhayz, S. J. Koh, and J. G. Choi, “Performance evaluation of AMQP over QUIC in the internet-of-thing networks,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 4, pp. 1–9, 2023, doi: 10.1016/j.jksuci.2023.02.018.

[48] V. Seoane, C. Garcia-Rubio, F. Almenares, and C. Campo, “Performance evaluation of CoAP and MQTT with security support for IoT environments,” *Comput. Networks*, vol. 197, no. July, p. 108338, 2021, doi: 10.1016/j.comnet.2021.108338.

[49] M. Domínguez, R. González-Herbón, J. R. Rodríguez-Ossorio, J. J. Fuertes, M. A. Prada, and A. Morán, “Development of a Remote Industrial Laboratory for Automatic Control based on Node-RED,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 17210–17215, 2020, doi: 10.1016/j.ifacol.2020.12.1741.

[50] A. Malik, J. Ahmed, J. Qadir, and M. U. Ilyas, “A measurement study of open source SDN layers in OpenStack under network perturbation,” *Comput. Commun.*, vol. 102, pp. 139–149, 2017, doi: https://doi.org/10.1016/j.comcom.2016.12.014.

[51] Y. Kessaci, N. Melab, and E.-G. Talbi, “A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager,” *Futur. Gener. Comput. Syst.*, vol. 36, pp. 237–256, 2014, doi: https://doi.org/10.1016/j.future.2013.07.007.

[52] Q. Huang *et al.*, “Evaluating open-source cloud computing solutions for geosciences,” *Comput. Geosci.*, vol. 59, pp. 41–52, 2013, doi: https://doi.org/10.1016/j.cageo.2013.05.001.

[53] P. S. Saikrishna and R. Pasumarthy, “Multi-objective switching controller for cloud computing systems,” *Control Eng. Pract.*, vol. 57, pp. 72–83, 2016, doi: https://doi.org/10.1016/j.conengprac.2016.09.001.

[54] Z. Wang, S. Yang, X. Xiang, A. Vasilijević, N. Mišković, and Đ. Nađ, “Cloud-based remote control framework for unmanned surface vehicles∗∗The research leading to these results was partially funded by the European Union’s Horizon 2020 research and innovation programme under the project EUMarinerobots (grant agreement No 7311,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14564–14569, 2020, doi: https://doi.org/10.1016/j.ifacol.2020.12.1462.

[55] M. Blackstock and R. Lea, “Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED),” in *Proceedings of the 5th International Workshop on Web of Things*, 2014, pp. 34–39, doi: 10.1145/2684432.2684439.

[56] M. García-Valls, A. Dubey, and V. Botti, “Introducing the new paradigm of Social Dispersed Computing: Applications, Technologies and Challenges,” *J. Syst. Archit.*, vol. 91, no. March, pp. 83–102, 2018, doi: 10.1016/j.sysarc.2018.05.007.

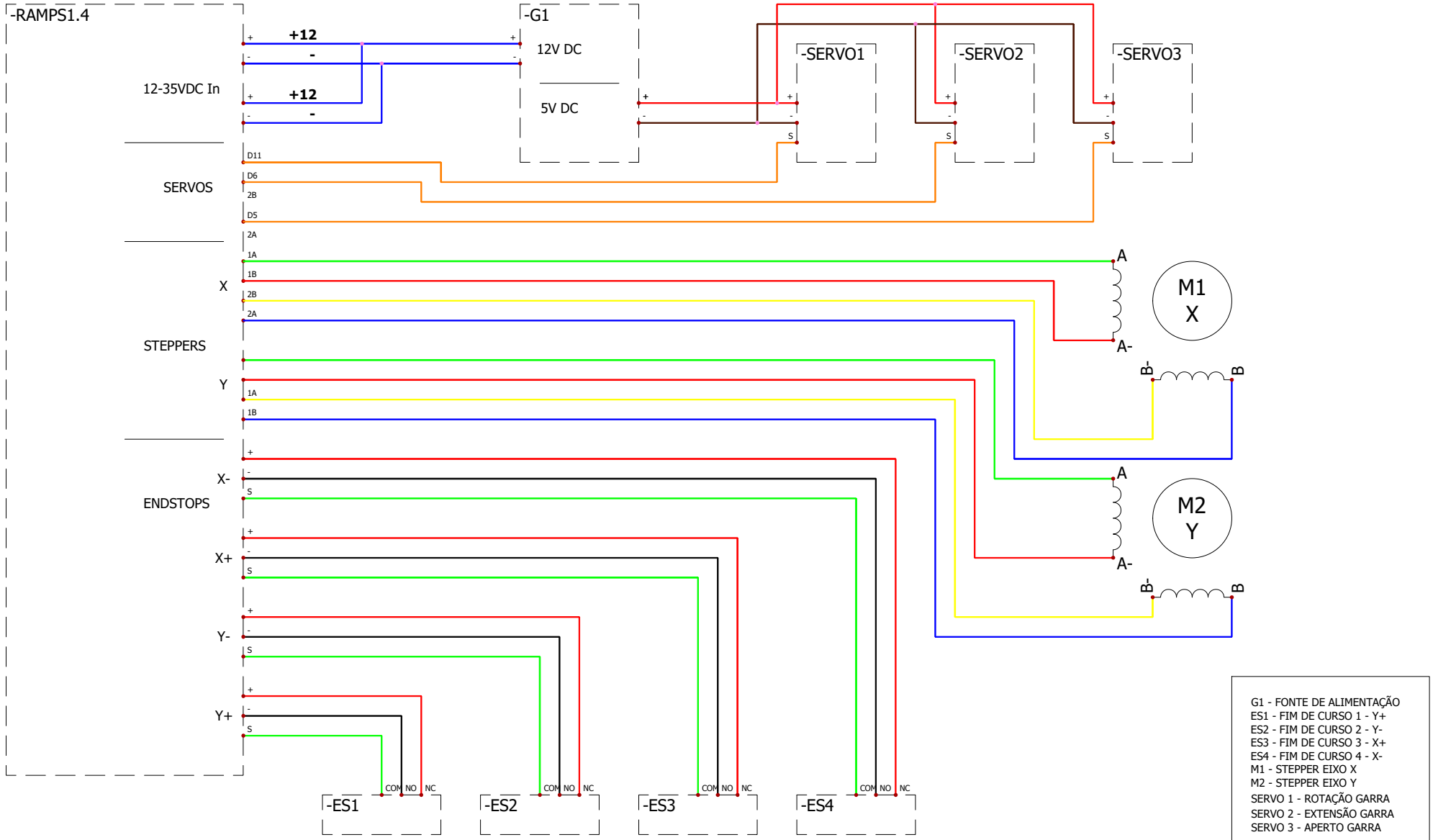
APÊNDICE A – CÓDIGO DO SISTEMA DE CONTROLO *OFFLINE*

https://pastebin.com/KUBWmHux

APÊNDICE B – CÓDIGO DO SISTEMA DE CONTROLO *INTEGRADO*

https://pastebin.com/YkWJFLnM

APÊNDICE C – ESQUEMA ELÉTRICO DO ASRS



ASRS

REV.	DATE	NAME	CHANGES
0	08/12/2021	Admin	

REVISION 0

SCHEME 01

CONTRACT:

LOCATION: +L1 Painel elétrico principal

User data 1

User data 2