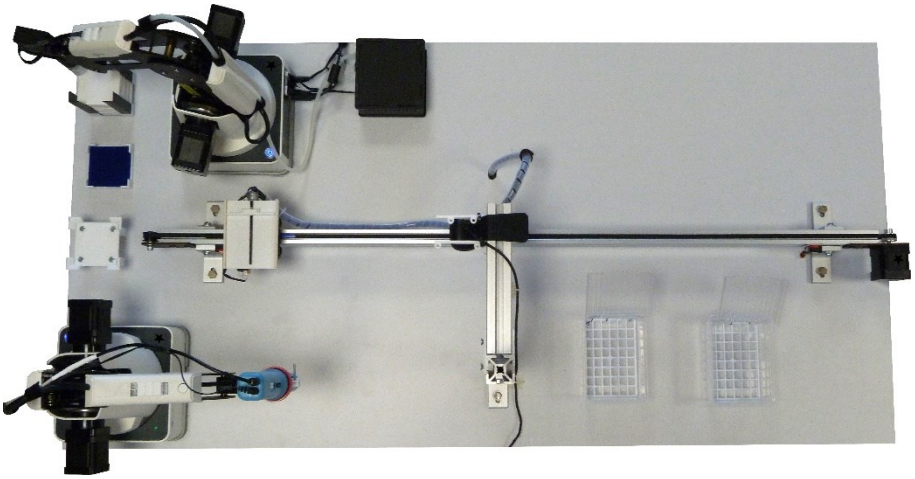

Desenvolvimento de um Processo Automático de Montagem de Painéis Solares à Escala Reduzida

MARCO COSTA LUCAS
(Licenciado em Engenharia Mecânica)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Mecânica, na Área de Especialização de Energia, Refrigeração e Climatização

Orientadores:
Doutor Francisco Mateus Marnoto de Oliveira Campos
Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira

Júri:
Presidente: Doutor André Rui Dantas Carvalho
Vogais:
Doutora Maria Graça Vieira Brito Almeida
Doutor Francisco Mateus Marnoto de Oliveira Campos

Novembro de 2024

Desenvolvimento de um Processo Automático de Montagem de Painéis Solares à Escala Reduzida

MARCO COSTA LUCAS
(Licenciado em Engenharia Mecânica)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Mecânica, na Área de Especialização de Energia, Refrigeração e Climatização

Orientadores:
Doutor Francisco Mateus Marnoto de Oliveira Campos, ISEL/IPL
Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira, ISEL/IPL

Júri:
Presidente: Doutor André Rui Dantas Carvalho, ISEL/IPL
Vogais:
Doutora Maria Graça Vieira Brito Almeida, ISEL/IPL
Doutor Francisco Mateus Marnoto de Oliveira Campos, ISEL/IPL

Novembro de 2024

Agradecimentos

O desenvolvimento deste projeto não teria sido possível sem o apoio de inúmeras pessoas que se cruzaram no meu caminho, em ambiente académico e pessoal.

Quero agradecer ao Professor Doutor Francisco Campos, que desde o primeiro contacto mostrou o seu apoio e motivação para a realização do projeto. Agradeço por toda a orientação, especialmente no desenvolvimento das peças e impressão 3D, bem como nos desafios que surgiram na integração dos diversos equipamentos.

Agradeço ao Professor Doutor Fernando Carreira pelas sugestões e orientações providenciadas durante a realização do projeto desde a proposta de implementação de visão computacional numa fase muito embrionária do projeto até ao apoio na revisão do relatório na sua fase final.

À minha família, pela permanente preocupação, motivação e aconselhamento de decisões a nível pessoal e académico. Agradeço de forma especial ao meu pai por toda a ajuda oferecida no âmbito da eletrónica do projeto.

À minha metade, Catarina, agradeço o apoio incondicional em todos os momentos de insegurança e incerteza, as horas passadas a ouvir os meus desabafos e planos para o projeto. Sem a sua presença e estabilidade que proporciona, nunca chegaria onde cheguei hoje. Esta página A4 é insuficiente para listar todos os momentos em que esteve presente para me motivar e fazer ver as coisas por outro prisma, mais otimista. Agradeço a ajuda na fase final do projeto, com a aquisição dos diversos materiais, apoio na montagem do laboratório e as soluções criativas para os desafios que surgiram.

A todos os meus amigos, colegas e professores que se cruzaram no meu percurso e ajudaram-me a tornar a pessoa que sou hoje, o meu profundo agradecimento.

Declaração de integridade

Declaro que este trabalho de projeto é o resultado da minha investigação pessoal e independente. O seu conteúdo é original e todas as fontes listadas nas referências bibliográficas foram consultadas e estão devidamente mencionadas no texto. Mais declaro que todas as referências científicas e técnicas relevantes para o desenvolvimento do trabalho estão devidamente citadas e constam das referências bibliográficas.

O autor

Assinado por: **MARCO COSTA LUCAS**
Num. de Identificação: 13477690
Data: 2024.11.22 16:52:26+00'00'

Lisboa, 22 de Novembro de 2024

Desenvolvimento de um Processo Automático de Montagem de Painéis Solares à Escala Reduzida

Resumo

A aplicação crescente da robótica nas mais diversas indústrias encontra-se relacionada com a evolução tecnológica e a adaptabilidade para a realização de tarefas num espetro diversificado. A robótica, em ambiente académico, conhecida como robótica educacional, revela grandes vantagens como ferramenta de aprendizagem inovadora com o potencial de transformar a educação e apoiar os educandos nos mais diversos contextos de aprendizagem. Na robótica educacional os educandos são incentivados a pôr em prática os seus projetos através de linguagens de programação gráficas ou textuais, enriquecendo as suas capacidades de resolução de problemas. Perspetivando-se um forte crescimento no mercado das inovações tecnológicas associadas à robótica, inteligência artificial e programação, torna-se premente uma aposta na formação de futuros engenheiros nestas temáticas.

A finalidade deste projeto foi providenciar o laboratório de robótica do Departamento de Engenharia Mecânica do Instituto Superior de Engenharia de Lisboa com novas valências de modelos de contextos de produção industrial através da cooperação entre vários equipamentos com recurso à linguagem de programação Python, promovendo atividades interdisciplinares.

O projeto focou-se na implementação de um modelo de um processo automático, em escala reduzida, inspirado na montagem de painéis solares. Para este efeito foram usados dois robôs Dobot Magician e dois eixos lineares, acionados por motores de passo e controlados por um microprocessador Arduino UNO. Para o apoio da produção foram projetadas diversas peças, recorrendo a técnicas de desenho assistido por computador, que posteriormente foram impressas em 3D. Foi desenvolvido um procedimento de verificação de qualidade da produção, recorrendo a visão computacional, para distinguir o destino da peça produzida em função do resultado do teste.

Antevê-se que as ações realizadas na integração e controlo dos variados equipamentos, num ambiente de programação em forte crescimento, constituam o ponto de partida para futuros projetos no âmbito da temática desenvolvida.

Palavras-chave: Robótica Educacional, Python, Controlo de Produção, Automação, Supervisão

Development of an Automated Process for Small-Scale Solar Panel Assembly

Abstract

The growing use of robotics in the most diverse industries is related to technological evolution and adaptability in the application of solutions that allow tasks to be carried out across a diverse spectrum. Robotics, in an academic environment is known as educational robotics and reveals great benefits as an innovative learning tool with the potential to transform education and support students in the most diverse learning contexts, encouraging them to put their projects into practice through graphical or textual programming languages, allowing the enhancement of problem-solving skills. With the prospect of significant growth in the market for technological innovations associated with robotics, artificial intelligence and programming, there is an urgent need to invest in the training of future engineers in these topics.

The purpose of this project was to provide the robotics laboratory of the Department of Mechanical Engineering of the Instituto Superior de Engenharia de Lisboa with new models of industrial production contexts through cooperation between various equipment using the Python programming language, thus promoting interdisciplinary activities.

The project focused on implementing a model of an automatic process, on a small-scale, inspired by the assembly of solar panels. For this purpose, two Dobot Magician robots and two linear axes were used, driven by stepper motors and controlled by an Arduino UNO microprocessor. To support production, several parts were designed using computer-aided design techniques, which were later printed in 3D. A production quality verification procedure was developed using computer vision to distinguish the destination of the produced part, depending on the result of the quality test.

It is anticipated that the outcomes in the integration and control of the various equipment, in a rapidly growing programming environment, constitute the starting point for future projects within the scope of the studied topic.

Keywords: Educational Robotics, Python, Production Control, Automation, Supervision

Lista de símbolos e siglas

Alfabeto romano

V_{REF}	*tensão de referência*
I_{MAX}	*corrente máxima*
R_s	*resistência instalada (driver A4988)*

Siglas

3D	*Três Dimensões*
API	*Application Programming Interface*
ATX	*Advanced Technology Extended*
CAD	*Computer-Aided Design*
CNC	*Computer Numerical Control*
CSV	*Comma Separated Values*
COM	*Porta de Comunicação*
CTEM	*Ciência, Tecnologia, Engenharia e Matemática*
DEM	*Departamento de Engenharia Mecânica*
DLL	*Dynamic Link Library*
EIO	*Extended Input and Output*
EVA	*Acetato-Vinilo de Etileno*
GDL	*Graus de Liberdade*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
HDMI	*High-Definition Multimedia Interface*
IDE	*Ambiente de Desenvolvimento Integrado*
IOT	*Internet of Things*
I/O	*Input/Output*
ISEL	*Instituto Superior de Engenharia de Lisboa*
PIB	*Produto Interno Bruto*
PTP	*Point to Point*
PVC	*Policloreto de Vinilo*
RE	*Robótica Educacional*
RGB	*Vermelho, Verde, Azul*
SCARA	*Selective Compliance Assembly Robot Arm*
UART	*Universal Asynchronous Receiver/Transmitter*
UC	*Unidade Curricular*
UE	*União Europeia*

USB *Universal Serial Bus*

YOLO *You Only Look Once*

Índice

Índice de figuras

Índice de quadros

Índice de listagens

1 Introdução

1.1 Enquadramento e motivação

A evolução da robótica tem transformado as indústrias e o nosso quotidiano em geral. Na indústria os robôs têm desempenhado um papel da maior relevância, observada desde os primeiros passos da automatização de processos, ocorrida na Indústria 3.0 até à interligação de sistemas que permitem a colaboração entre humanos e robôs na Indústria 4.0. Com a antecipação da chegada, em breve, da Indústria 5.0, prevê-se uma simbiose mais estreita entre o homem e sistemas robóticos (Zafar et al., 2024). Segundo Niku (2020), os robôs são elementos extremamente poderosos na indústria atual, capazes de realizar uma vasta gama de tarefas e operações com precisão, sem a necessidade de se garantir medidas de segurança e conforto que são exigidas para um operador humano, podendo ser utilizados em cenários de elevado risco, como o espaço ou ambientes submarinos.

Nos últimos anos tem-se verificado uma tendência crescente de introdução da robótica na educação, potenciada pelo desenvolvimento contínuo da ciência, da tecnologia e pela chegada da era da inteligência artificial (Yang et al., 2020). O Instituto Superior de Engenharia de Lisboa (ISEL) não é exceção a esta tendência, possuindo um Laboratório de Robótica, integrado no Departamento de Engenharia Mecânica (DEM), onde diariamente são desenvolvidas atividades nas diversas áreas abrangidas pela robótica como, por exemplo, a programação, construção mecânica e eletrónica. Com o objetivo de aumentar as capacidades do laboratório e introduzir novos contextos de aprendizagem, foi proposta a realização do presente projeto, no âmbito do Trabalho Final de Mestrado, consistindo na integração de diversos equipamentos existentes para a realização de um processo de produção automatizado.

A implementação de novas ferramentas, no contexto de ensino, relacionadas com a robótica e a programação, promove a formação de profissionais mais bem preparados para os desafios tecnológicos do futuro, incentivando a inovação e a capacidade de resolução de problemas.

1.2 Objetivos do projeto

O presente projeto possui como principal objetivo criar um processo automático que possa ser usado como aplicação educacional no âmbito da robótica, eletrônica e programação, através da integração de diversos equipamentos, para a execução de uma operação de montagem. Foi também estabelecido como objetivo desenvolver uma aplicação de controlo e supervisão do processo com recurso à linguagem de programação Python. A escolha desta linguagem deve-se à sua popularidade e à sua adoção na unidade curricular (UC) de Introdução à Programação na Licenciatura de Engenharia Mecânica, o que justifica o seu uso nas outras UCs do curso, nomeadamente em trabalhos na área da robótica e automação.

Para o cumprimento destes objetivos foram estabelecidos os seguintes objetivos secundários:

- Definir o processo automático a desenvolver;
- Realizar o projeto mecânico e fabricar os elementos necessários;
- Configurar diversos equipamentos para automatizar o processo;
- Preparar um espaço de trabalho representativo do processo e integrar os equipamentos disponíveis no Laboratório de Robótica com os elementos fabricados;
- Implementar novas capacidades de comunicação com os equipamentos;
- Programar o sistema de controlo e automação do processo.

1.3 Metodologia aplicada

A realização de um projeto requer uma definição clara dos objetivos a concretizar e a adoção de uma metodologia para apoiar e alcançar as metas estabelecidas. A metodologia aplicada na realização do presente projeto consistiu em dividir em etapas as atividades a realizar. Tratando-se de um projeto com construção de protótipo, o estabelecimento de precedências entre as etapas revelou-se fundamental para garantir a disponibilidade dos recursos no momento adequado. As etapas estabelecidas para o projeto foram: a recolha de informação; planeamento e definição do processo; fabricação de equipamentos necessários; preparação e montagem da área de trabalho; programação de equipamentos e testes do protótipo. Devido à complexidade de algumas etapas, houve a necessidade de as subdividir em tarefas, de modo a geri-las mais facilmente.

A validação do cumprimento dos objetivos do trabalho foi realizada através de uma análise do processo recorrendo a vários testes do protótipo.

Foi desenhado um cronograma que organiza as tarefas chave de cada etapa, garantindo uma distribuição equitativa do volume de trabalho ao longo do período de realização do projeto, a precedência das tarefas e a antecipação de possíveis dificuldades. Durante a execução do plano foram efetuados pequenos ajustes ao cronograma para a sua adequação ao ritmo das atividades realizadas. Na Figura 1.1 pode ser observado o cronograma das tarefas chave, elaborado com as adaptações finais já incluídas.

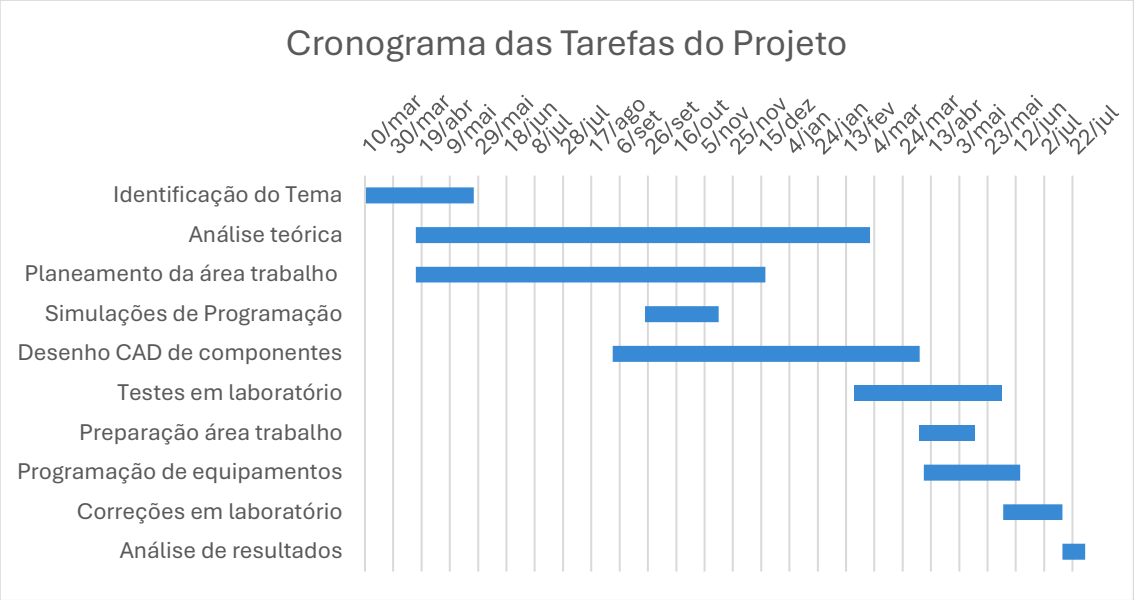

Figura 1.1 – Cronograma de realização das tarefas chave do projeto.

1.4 Estrutura do projeto

O presente documento encontra-se estruturado em seis capítulos, compostos pelos conteúdos seguidamente descritos.

O primeiro capítulo, de caracter introdutório, enquadra a temática do Trabalho Final de Mestrado, os seus principais objetivos e a metodologia aplicada na sua elaboração.

No segundo capítulo é estudado o estado da arte e enquadramento teórico dos temas de maior relevância abordados no trabalho, nomeadamente, a Robótica, Eletrónica, Tecnologia e o contexto destas temáticas no ensino.

No terceiro capítulo é descrito o desenvolvimento da conceção do protótipo do processo, composto pela descrição do local onde foi implantado, caracterização e explicação do processo e equipamentos intervenientes, apresentação dos componentes produzidos no âmbito do trabalho e a montagem dos equipamentos alusivos ao processo. Ao longo do capítulo são descritas as estratégias e soluções adotadas para a resolução de desafios e dificuldades emergentes durante a realização do projeto.

No quarto capítulo é exposto o sistema de controlo e supervisão que foi desenvolvido, através de uma explicação detalhada dos programas que o constituem. Adicionalmente

é apresentada a interface gráfica concebida para a interação entre o utilizador e o processo.

No quinto capítulo é efetuada a apresentação e análise critica dos resultados obtidos do trabalho desenvolvido, com o foco na verificação da operacionalidade das várias etapas do processo e a análise de parâmetros de produção.

No sexto capítulo são apresentadas as conclusões retiradas do trabalho e a confrontação com os objetivos propostos. Adicionalmente são descritas as dificuldades identificadas e apresentadas propostas de trabalho futuro.

2 Enquadramento teórico

2.1 Robótica

2.1.1 Definição e evolução histórica da robótica

A robótica, de acordo com a Norma Internacional ISO 8373:2021, é a ciência e a prática de projeto, manufatura e utilização de robôs. Os robôs, por seu lado, são descritos como mecanismos comandados por programas com autonomia para realizar locomoção, manipulação ou posicionamento (International Organization for Standardization, 2021).
O termo robô foi utilizado pela primeira vez em 1921 pelo escritor checo Karel Capek na sua peça de teatro intitulada R.U.R. (*Rossum's Universal Robots*). A origem do termo deriva da palavra checa *robota* com o significando de "trabalho forçado" ou "escravo" (Klafter et al., 1989). Na peça, os robôs, surgem com semelhanças com a fisionomia humana, mas dispõem de uma capacidade de produzir um maior volume de trabalho (Poynton, 2022).
No conto Runaround, de 1942, escrito pelo americano Isaac Asimov, surge pela primeira vez o termo "robótica" e as três regras ou leis da robótica (Asimov, 1942):

1. "*Um robô não pode ferir um ser humano ou, por inação, permitir que um ser humano se venha a ferir";*
2. *"Um robô deve obedecer a ordens dadas por seres humanos, exceto quando tais ordens entrem em conflito com a Primeira Lei.";*
3. *"Um robô deve proteger a sua própria existência desde que tal proteção não entre em conflito com a Primeira ou Segunda Lei*.".

Embora o termo e conceito de robô e robótica surja apenas no século XX, a existência de máquinas com a capacidade de operar autonomamente, ou autómatos, remontam à Grécia Antiga. No ano de 350 a.C. o filósofo e matemático Archytas de Tarentum (428 a.C. - 347 a.C.) efetuou uma demonstração de uma máquina intitulada "O Pombo". A máquina possuía a capacidade de voar uma distância superior a 200 metros com a utilização de um jato de vapor e ar comprimido (Yates et al., 2011).

Igualmente na Grécia Antiga, no ano de 250 a.C., foram produzidos, pela primeira vez, relógios de água, ou clepsidras, com peças móveis, pelo engenheiro civil Ctesibius de Alexandria (285 a.C – 222 a.C.) (Jeoung-Myoung, 2023).

No século XV, Leonardo da Vinci (1452 – 1519) apresentou um robô com semelhanças a formas humanas, denominado humanóide, conhecido como o “Cavaleiro Mecânico” que acabou por evidenciar a necessidade de aplicação de grande rigor na construção de máquinas automáticas bem como a exigência de uma fonte permanente de energia, podendo ser pneumática, hidráulica ou elétrica. Segundo Pires (2007), a ideia do “Cavaleiro Mecânico” era tão revolucionária para o seu tempo que Leonardo da Vinci optou por fazê-lo desaparecer.

A contribuição do Nicola Tesla (1856 – 1943) para a robótica surge no século XIX, com a utilização da descoberta das ondas de rádio de Heinrich Hertz (1857 – 1894) para comandar um autómato. Para demonstrar a ideia, Tesla construiu um submarino telecomandado em miniatura (Pires, 2007).

Após as contribuições de um elevado número de engenheiros, cientistas e inventores, até ao século XIX, persistia o problema da falta de inteligência por parte das máquinas. De forma a desempenharam as tarefas autonomamente, os robôs deverão possibilitar a sua prévia programação.

A “idade do robô”, segundo Klafter et al. (1989), teve início quando George Devol patenteou o primeiro robô industrial, conhecido como o “Unimate” em 1961, transformando a indústria de produção (Ballard et al., 2012). O robô possuía a capacidade de efetuar um movimento controlado de um ponto para o outro, de forma autónoma (Gasparetto & Scalera, 2019).

O desenvolvimento da robótica e o número de equipamentos introduzidos na indústria cresceu de forma exponencial a partir de meados do século XX. Na indústria atual, a robótica é de importância basilar; sem ela não seria possível o elevado rigor e velocidade de produção características da indústria moderna.

Segundo o World Robotics 2023 publicado pelo International Federation of Robotics, em 2022 existiam 3,9 milhões de robôs industriais a operar em fábricas pelo mundo, um aumento de 13% face a 2017. A Ásia é a região com maior expressão de instalações de robôs industriais no ano de 2022 com 405 000 unidades, com a China a apresentar uma evolução significativa face ao resto do mundo, tendo o número de instalações anuais de robôs evoluído de 14% em 2012 para 52% em 2022. A Europa registou um número de 84 000 robôs industriais instalados em 2022, com a Alemanha a liderar o mercado neste segmento. O setor industrial com o maior número de instalações de robôs no ano de 2022 foi o da eletricidade e eletrónica, seguida de forma muito próxima pelo setor automóvel. Na Figura 2.2 é possível observar a evolução do número de robôs industriais em operação ao longo dos anos recentes, com representação da Taxa de Crescimento Anual Composta de 13% entre os anos 2017 e 2022 (Bill et al., 2023).

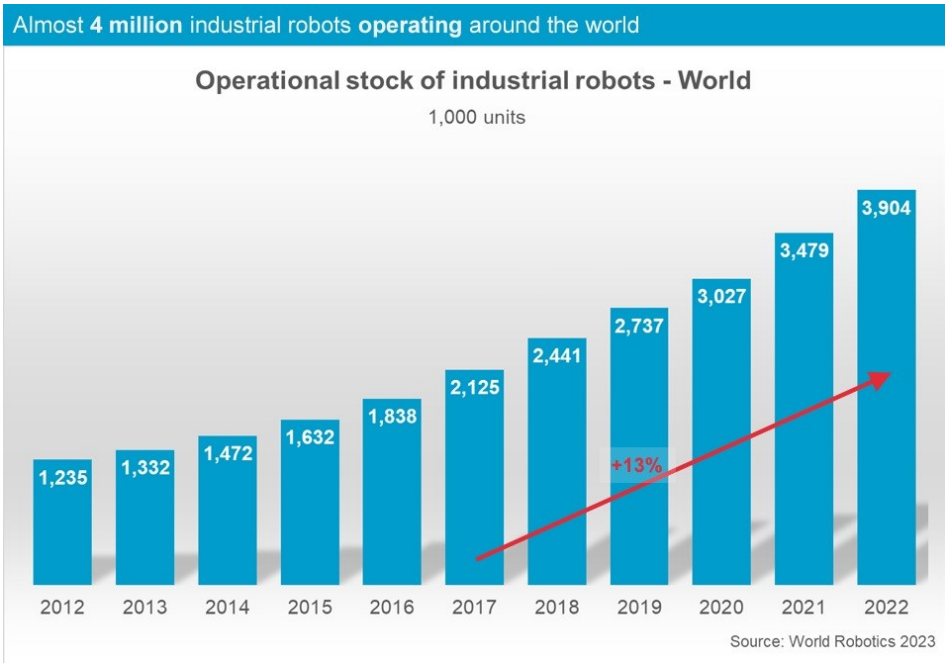

Figura 2.2 – Número de robôs industriais em operação (em milhares), adaptada de Bill et al. (2023).

2.1.2 Futuro da robótica

A evolução crescente de utilização da robótica nas mais diversas indústrias encontra-se essencialmente associada à evolução da tecnologia e à sua adaptabilidade na realização de tarefas de elevada complexidade ou de funções simples com uma elevada cadência (Verl, 2023). O objetivo foca-se essencialmente na melhoria de eficiência e produtividade. No futuro, prevê-se que a robótica, auxiliada por outros avanços tecnológicos, como por exemplo a inteligência artificial, proporcione elevados benefícios para a indústria e para a economia mundial (PwC, 2017).

Segundo a estimativa de Hawksworth & Berriman (2018), as inovações tecnológicas associadas à robótica e inteligência artificial poderão representar, em 2030, uma contribuição de até 14% do Produto Interno Bruto (PIB) mundial, sendo equivalente a 13,8 triliões de euros, no valor atual.

Os campos de aplicação da robótica estendem-se além da indústria. Atualmente encontram-se em desenvolvimento robôs com a capacidade de interpretar instruções dadas com linguagem natural bem como a habilidade de reconhecer certas emoções humanas e reagir perante as mesmas. O desenvolvimento desta tipologia de robótica encontra-se em franco crescimento para a aplicação nas áreas de serviço ao cliente e de assistência médica (Verl, 2023).

Segundo Boesl (2016) os vários estudos elaborados para analisar a tendência prevista para o desenvolvimento da área da robótica focam-se essencialmente nas seguintes aplicações:

- Robótica de produção industrial;
- Robótica médica e de cuidados de saúde;
- Robótica associada à agricultura;
- Robótica militar;
- Robótica associada à logística e transporte;
- Robótica aeroespacial;
- Robótica de apoio ao serviço a consumidores.

Além dos tópicos enumerados também foram destacadas as áreas de investigação na robótica, aplicações educacionais da robótica, veículos autónomos e o impacto da robótica na sociedade.

Segundo o autor espera-se, especificamente nos robôs, uma evolução das vertentes tecnológicas e de construção. Os pontos em que se antevê uma evolução futura são resumidamente dados por:

- Modificações no hardware utilizado, como por exemplo ao nível dos atuadores, sensores, materiais utilizados e elementos terminais;
- Evoluções nas tipologias de software utilizadas, nomeadamente com a aplicação de sistemas com utilização da nuvem e segurança de dados;
- Interação entre humanos e robôs;
- Avanços no estado de consciência e perceção dos robôs;
- Melhorias na cognição automática e inteligência artificial.

A evolução da robótica possui uma relação com as diversas revoluções industriais, existindo um aumento da sua aplicação e importância com as sucessivas transições. Durante a 3ª revolução industrial ocorreu um aumento significativo na capacidade de fabrico em linhas de montagem, em grande parte fruto dos avanços tecnológicos, de computação e da robótica (Sherwani et al., 2020). A 4ª revolução industrial, ou Indústria 4.0 é caracterizada como a rápida transformação da tecnologia, das indústrias e dos padrões sociais atuais, dos processos e sistemas e a interconexão com a automação inteligente (Chu et al., 2022). Na Figura 2.3 é possível observar as diversas revoluções industriais, bem como as particularidades que as caracterizam.

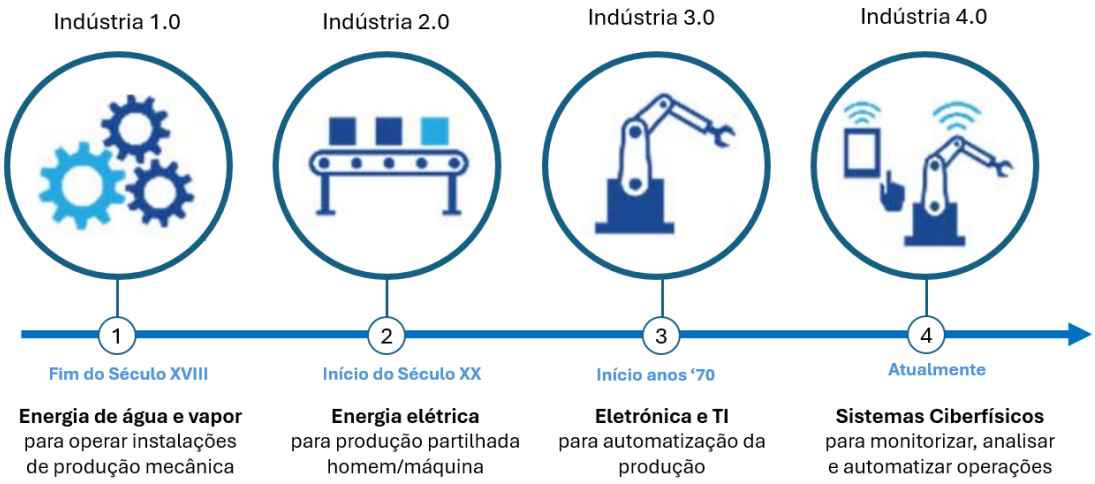

Figura 2.3 – As quatro revoluções industriais, adaptada de Chu et al. (2022).

Com diversos autores a perspetivarem a chegada da Indústria 5.0, prevê-se que esta nova revolução industrial proporcione, à semelhança das anteriores, um elevado aumento das potencialidades para a evolução da robótica, desta vez, na diversificação de robôs e máquinas inteligentes que permitirão aos humanos trabalhar de forma mais eficiente. Prevêem-se que a Indústria 5.0 empregará em 60% das áreas de logística, manufatura, mineração, agricultura e energia de trabalhos diretamente relacionados com robótica (Kabeyi & Olanrewaju, 2023).

2.1.3 Classificações de robôs

A classificação dos robôs é complexa e existem várias abordagens, dependendo dos autores. Os robôs podem ser classificados segundo a tipologia de tarefas que efetuam, segundo as suas aplicações ou segundo a forma como se movimentam.

Como classificação mais ampla e mais popular, os robôs são divididos pela sua aplicação, podendo ser separados em dois conjuntos, os robôs industriais e os robôs de serviço. Na Norma Internacional ISO 8373:2021, um robô industrial é descrito como um manipulador multifuncional reprogramável e controlado automaticamente, programável em três ou mais eixos que podem encontrar-se fixos ou montados sobre uma plataforma móvel (International Organization for Standardization, 2021), para uso em aplicações de automação em ambiente industrial. Por sua vez, a norma descreve um robô de serviço como um robô de uso pessoal ou profissional que executa tarefas úteis para humanos ou equipamentos.

Um robô industrial é, simplificadamente, composto pelos seguintes componentes:

- **Base**: Suporte do robô que permite efetuar o varrimento da área de trabalho em segurança e sem vibrações. Frequentemente aloja os componentes eletrónicos necessários ao funcionamento do robô. Caso não seja um robô móvel, é na base que se fixa o robô ao seu local de operação;
- **Elo**: Consiste em ligações quase rígidas conectadas por juntas que permitem o movimento relativo dos elos vizinhos (Craig, 2005);
- **Junta**: Gere a relação dos movimentos entre os elos do robô, funcionando de forma semelhante ao cotovelo humano. Representam uma função critica no controlo da postura e movimentos bem como na determinação da carga máxima que pode ser transportada. Nas juntas de rotação a deslocação é determinada através de ângulos, nas juntas lineares a deslocação é determinada através de um valor de translação (Duan et al., 2021);
- **Elemento Terminal**: Está localizado na extremidade livre da cadeia de elos que compõem o braço robótico. Este componente pode variar dependendo da aplicação em causa, podendo ser uma garra, dispositivo de sucção, mecanismo de impressão em três dimensões (3D), entre outros (Craig, 2005).

Na Figura 2.4 encontram-se identificados os vários componentes que tipicamente constituem um robô.

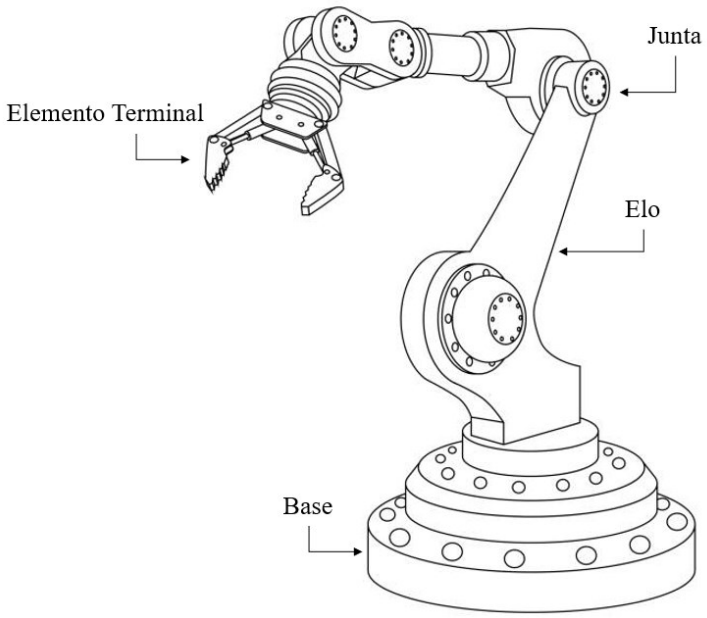

Figura 2.4 – Componentes constituintes de um robô industrial, adaptada de Kazakov (2021).

Para a classificação dos robôs indústrias podem ser adotadas várias abordagens, as mais populares que são as que avaliam o robô quanto ao número de graus de liberdade (GDL) e outra que classifica segundo a sua estrutura mecânica ou cinemática.

O número de GDL é determinado pelo número de juntas do robô. Tipicamente um robô industrial possui seis GDL, permitindo a sua movimentação em qualquer direção no espaço e a orientação do elemento terminal (Neves, 2022).

A classificação segundo a cinemática de robôs divide-se em seis categorias: cartesiano, cilíndrico, esférico, Selective Compliance Assembly Robot Arm (SCARA), vertical articulado e paralelo (Dobra, 2014). Seguidamente encontram-se descritas as particularidades de cada categoria.

Robô cartesiano

Tipicamente compostos por juntas de translação que permitem a deslocação nos três eixos do sistema de coordenadas cartesiano. Na Figura 2.5 é possível observar um exemplo de um robô com classificação cartesiano.

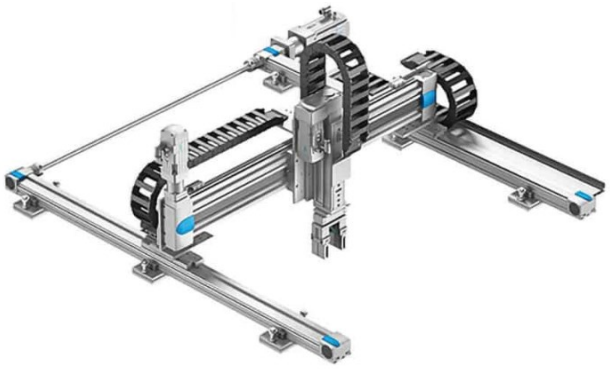

Figura 2.5 – Exemplo de robô cartesiano, adaptada de Festo (2024).

Robô cilíndrico

Compostos por um braço horizontal que se encontra instalado numa coluna vertical que, por sua vez, se encontra instalada numa base rotativa, como se pode observar na Figura 2.6.

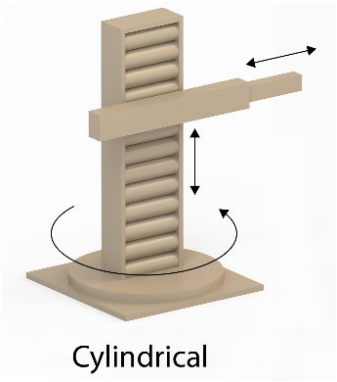

Figura 2.6 – Exemplo de robô cilíndrico, adaptada de Fairchild (2024).

Robô esférico

Possuem um braço de translação que está montado sobre uma junta de rotação, através da qual se modifica a orientação relativamente ao plano horizontal. Estes robôs podem

rodar em torno de uma base, resultando numa área de trabalho do robô com a forma de uma porção de esfera. Na Figura 2.7 pode-se observar os eixos de rotação e translação dos robôs com classificação esférica.

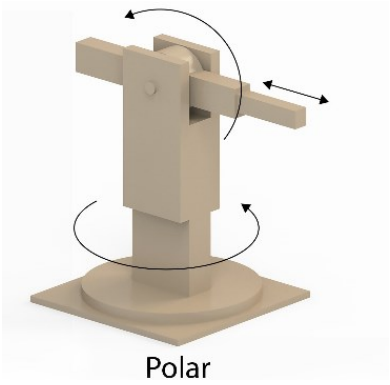

Figura 2.7 – Exemplo de robô esférico, adaptada de Fairchild (2024).

Robô SCARA

Esta tipologia de robô caracteriza-se por uma boa flexibilidade de movimentos segundos os eixos X e Y e maior rigidez na direção Z, que são vantajosas nas tarefas de montagem. Tipicamente o robô SCARA possui três juntas de rotação e uma junta prismática. O posicionamento do elemento terminal no plano X-Y é efetuado pelas juntas rotativas enquanto o movimento no eixo Z é efetuado pela junta prismática (Chen et al., 2020). Na Figura 2.8 é possível observar os eixos de rotação e translação de um robô SCARA.

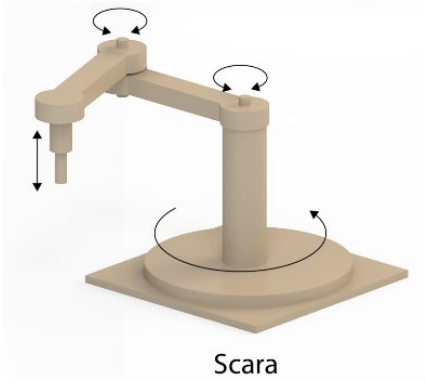

Figura 2.8 – Exemplo de robô SCARA, adaptada de Fairchild (2024).

Robô articulado vertical

Este é um robô em que todas as juntas são de rotação. As vantagens desta configuração são a possibilidade de chegar perto da base do robô, permitir uma boa relação entre a sua ocupação e a área de trabalho bem como possibilitar uma elevada flexibilidade. Estas características motivam a sua ampla utilização na indústria para a realização de tarefas de manuseamento (Gümbel et al., 2022). Na Figura 2.9 é possível observar um exemplo de robô de classificação articulado vertical.

Figura 2.9 – Exemplo de robô articulado vertical, adaptada de Delta Electronics Inc. (2024).

Robô paralelo

Os robôs paralelos distinguem-se dos anteriores por serem formados por uma estrutura cinemática fechada. Estes robôs movimentam uma pequena plataforma, onde é montado o elemento terminal, que está ligada à base por vários braços articulados. Os robôs com esta classificação permitem uma elevada eficiência nas funções de deslocação de objetos devido à capacidade de atingir velocidades superiores aos outros robôs (Khalil & Dombre, 2002). Na Figura 2.10 é possível observar um exemplo de robô paralelo.

Figura 2.10 – Exemplo de robô paralelo, adaptada de P. Li (2020).

2.2 Robótica educacional e experimental

A Robótica Educacional (RE) define-se, segundo Gabriele et al. (2012), como a área da investigação que visa promover uma aprendizagem ativa e envolvente através da utilização de artefactos criados pelos educandos e da exploração dos fenómenos que eles simulam. Com o desenvolvimento contínuo da ciência, da tecnologia e o advento da era da inteligência artificial, a introdução da prática e aplicação da robótica na educação tem-se tornado cada vez mais comum (Yang et al., 2020). O recurso à RE revela grandes vantagens como ferramenta de aprendizagem inovadora com o potencial de transformar a educação e apoiar os educandos em vários contextos de

aprendizagem (Evripidou et al., 2020). Adicionalmente, a RE incentiva os alunos a operar os seus modelos com a utilização de linguagens de programação gráficas ou textuais promovendo a capacidade de resolução de problemas (Alimisis, 2009).

Segundo Benitti (2012), a robótica é uma ferramenta de apoio ao ensino de conteúdos relacionados com as temáticas da robótica, nomeadamente a programação, a construção mecânica e a eletrónica.

A educação em ciência, tecnologia, engenharia e matemática (CTEM) tem demonstrado o potencial para influenciar positivamente a qualidade da educação e contribuir para a preparação da geração atual para a resolução de problemas, quer em contexto educacional, quer em contexto profissional (Kurniati et al., 2022).

Um dos motivos para a crescente utilização de robôs na educação provém da capacidade dos robôs para despertarem a imaginação das crianças e dos adultos devido às suas semelhanças com humanos ou à sua capacidade de as substituir na realização de variadas tarefas. A robótica educacional é frequentemente apresentada como uma Plataforma para alcançar três objetivos principais (Barak & Assal, 2018):

- O ensino de temáticas CTEM;
- O desenvolvimento de capacidade de aprendizagem como investigação científica, projeto de engenharia, resoluções de problemas, pensamento criativo e trabalho em equipa;
- Promoção da motivação dos educandos para o envolvimento na ciência e nas tecnologias.

O desenvolvimento do interesse pelas temáticas CTEM, na idade infantil, permite o desenvolvimento de várias valências que são exploradas e potenciadas na RE, conforme se pode observar na Figura 2.11. Os benéficos não são exclusivos às áreas da ciência; também na aprendizagem de línguas, a utilização de robôs para o treino da fala tem demonstrado melhorias na motivação e aprendizagem dos alunos (Atman Uslu et al., 2023).

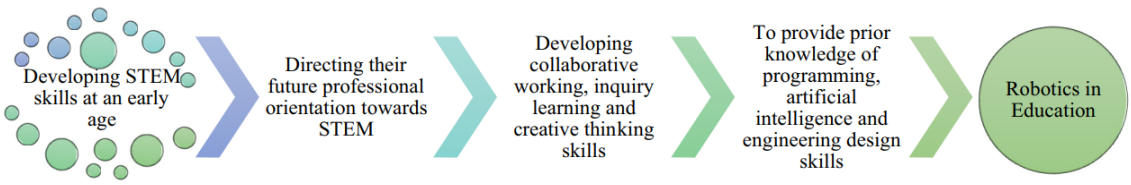

Figura 2.11 – Resultados educacionais de aplicações robóticas, adaptada de Gunes & Kucuk (2022).

2.2.1 Robôs educacionais

Os robôs educacionais são definidos como máquinas programadas para cumprir diversas tarefas, projetadas com vários componentes e com funcionamento controlado por operador ou de forma autônoma. Os robôs educacionais podem possuir a forma de humanos, animais ou veículos, variando na sua forma e tamanho (Alimisis, 2009).

Segundo Kılıç & Gökoğlu (2022), a ideia de utilizar robôs no estudo das ciências computacionais pode remontar ao trabalho produzido por Seymour Papert. Papert desenvolveu a linguagem de programação Logo na década de 1960, com o objetivo de compreender como as crianças aprendiam, quando a utilização de computadores não era uma possibilidade para todos.

Após a introdução do *Turtle Robot* produzido pelo próprio Papert, outros robôs educacionais foram produzidos para várias gamas de idades, como por exemplo o WeDo, Lego Mindstorms, KiwiRobotic, BeeBor, Cubelets e TangibleK (Yolcu & Demirer, 2023). Todos estes robôs possuem um controlador, que normalmente pode ser programado ou configurado pelo educando. Os controladores *open-source* mais populares da atualidade na RE são o Arduino e Raspberry Pi, que têm como função o processamento de informação do robô, funcionando como o seu cérbero (Chebotareva & Gavrilova, 2019). Alguns exemplos dos robôs enumerados podem ser observados na Figura 2.12 e Figura 2.13.

Figura 2.12 – Robô educacional Lego Mindstorms, adaptada de Lego (2024).

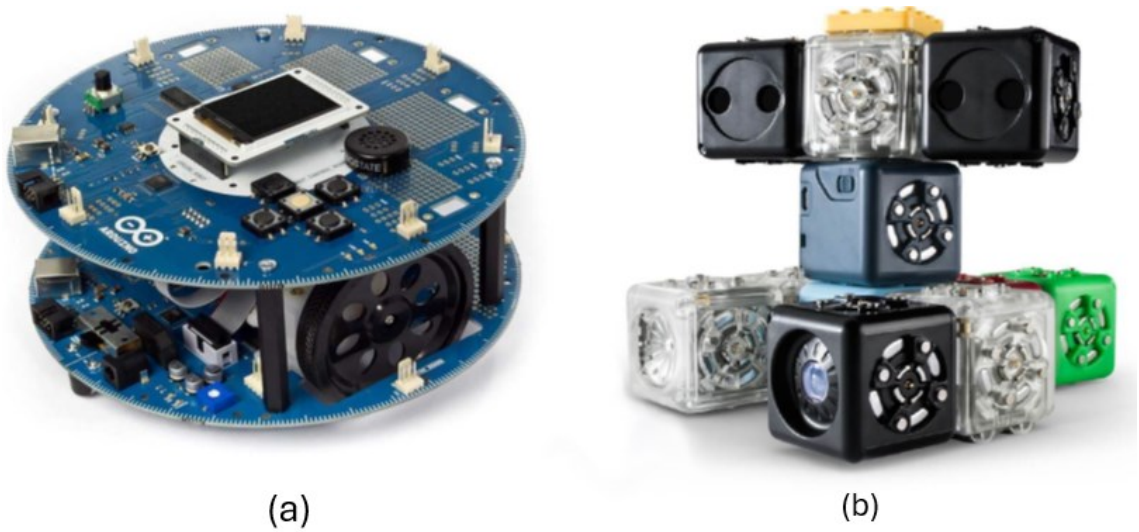

Figura 2.13 – Robôs educacional Arduino (a) e Cubelets (b), adaptada de Arduino (2024c) e Modular Robotics (2024).

Os sensores e motores dos robôs possibilitam aos educandos a oportunidade de definir comportamentos para os objetos através da programação e simultaneamente obter interação visual e física com o resultado. A programação de robôs educacionais é considerada relativamente simples através da utilização da programação baseada em blocos, como por exemplo o Blocky, Scratch, Mblock, entre outros. A simplicidade permite a utilização de robôs educacionais em todos os níveis do ensino (Yolcu & Demirer, 2023).

2.2.2 Microcontrolador Arduino

Grande parte da identidade de um robô provém da sua capacidade de realizar ações de forma autónoma. Esta capacidade provém da execução de instruções implementadas no programa que o regula. Para a execução das instruções é necessário um processador que descodifica o código e o converte numa ação física, criando os movimentos articulados característicos dos robôs. Existem diversas opções de processadores no mercado aptos para serem usados em robôs, contudo numa vertente mais educacional é dada preferência a soluções de programação mais intuitivas e versáteis. O microcontrolador Arduino surge como uma opção muito atrativa dada a sua capacidade de adaptação a diversos cenários e ao seu baixo custo. A utilização do Arduino na programação de robôs surge em diversos cursos e unidades curriculares de robótica e programação (Çakır et al., 2022).

O Arduino dispõe de uma ampla variedade de produtos, tendo sido lançados mais de 100 produtos de hardware desde a sua criação. Os produtos mais conhecidos e utilizados são as placas Nano, MKR, Classic e Mega (Arduino, 2024a).

Uma das placas que mais se destaca no âmbito da RE é o Arduino UNO, nome que provém do número “um” em italiano, e caracteriza-se por ser um microcontrolador

económico, flexível e que permite uma programação *open-source* de fácil aplicação. O microcontrolador pode ser integrado numa variedade de projetos eletrónicos e de robótica. O Arduino UNO possui o microcontrolador ATmega328 com 6 pinos de entradas analógicos e 14 pinos de entradas e saídas digitais. A placa possui uma capacidade de armazenamento de 32KB onde pode ser armazenado o programa (Špaková et al., 2024). Na Figura 2.14 pode ser observado um exemplo de um microcontrolador Arduino UNO.

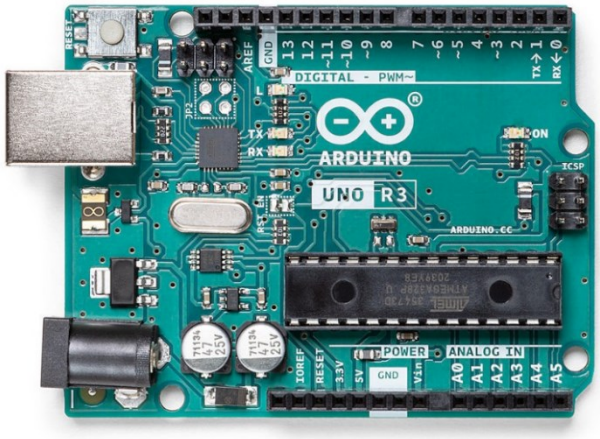

Figura 2.14 – Microcontrolador Arduino UNO, adaptada de Arduino (2024b).

2.2.3 Raspberry Pi

A Raspberry Pi Foundation, fundada no Reino Unido em 2009, produz uma linha de computadores de reduzida dimensão com o nome Raspberry Pi (Harini et al., 2023). À semelhança do microprocessador Arduino, o Raspberry Pi surge como uma solução acessível e flexível para diversas aplicações e projetos educacionais. Com mais semelhanças com um computador *desktop*, o Raspberry Pi permite a conectividade com monitores ou televisões através da sua saída *High-Definition Multimedia Interface* (HDMI) e com dispositivos de interface, como o rato e o teclado, caso o sistema operativo o permita. O Raspberry Pi permite a utilização de linguagens de programação como o Python e o Scratch, este último sendo mais vocacionado para crianças como ferramenta educacional (Karthikeyan et al., 2023).

A versatilidade do Raspberry Pi torna-o uma solução interessante nos mais diversos projetos. Os autores Harini et al. (2023), propuseram a utilização de um Raspberry Pi, com ligação a uma câmara de vídeo para a detetar e afugentar, através de som, corvos em campos de amendoins. O projeto foi proposto como solução para resolver um problema de perda de cultivo devido ao ataque por corvos.

Outro exemplo de projeto com recurso ao Raspberry Pi foi proposto por Jahangir Badashah et al. (2022), com o objetivo de conceber um robô com telepresença, ou seja,

com a capacidade de interagir com o ambiente ao seu redor. Neste projeto foi incorporado tanto um Raspberry Pi como um Arduino, sendo que este último era o responsável por operar os servomotores para o controlo do movimento da câmara e dos motores que movimentam o robô.

Dados recentes indicam que o Raspberry Pi é o computador de placa única mais popular. Segundo um inquérito a engenheiros efetuado pela Farnell, 44% preferiam utilizar o Raspberry Pi, contra 28% que escolheriam o Arduino, surgindo este em segundo lugar e o Beagleboard em terceiro com 6% (Manners, 2021). Ao contrário do Arduino, existe uma variedade reduzida de famílias e modelos do Raspberry Pi, sendo as famílias designadas por Raspberry Pi, Raspberry Pi 2, Raspberry Pi Zero, Raspberry Pi 3, Raspberry Pi 4, Raspberry Pi Pico e Raspberry Pi 5. Os modelos mais populares são o A, A+, B e B+. Com o desenvolvimento da tecnologia têm sido lançadas, com alguma regularidade, novas versões. O modelo inicial, denominado Raspberry Pi Model B+ foi lançado em 2014, enquanto o modelo mais recente, o Raspberry Pi 5 foi lançado em outubro de 2023 (Karthikeyan et al., 2023). Na Figura 2.15 pode ser observado um exemplo do controlador de placa única Raspberry Pi.

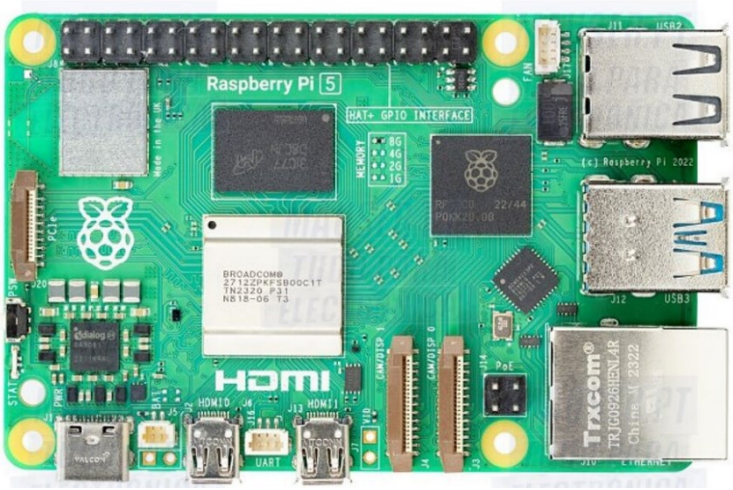

Figura 2.15 – Raspberry Pi 5, adaptada de Raspberry Pi (2024).

2.2.4 Dobot Magician

O Dobot Magician, produzido pelo fabricante Dobot Robotics, é um braço robótico multifuncional para aplicações educacionais (Dobot, 2024a). Permite a integração de programação, mecânica, eletrónica e automação, sendo uma ferramenta de excelência no ensino das CTEM. O Dobot Magician é disponibilizado com uma variedade de elementos terminais para diversas finalidades como escrita, sucção, impressão 3D, entre outros. Além das funcionalidades implementadas, o Dobot Magician suporta o desenvolvimento de até 13 dispositivos de interfaces extensíveis e mais de 20 linguagens de programação e possui uma elevada precisão, de 0,2 mm. Fruto do seu

sucesso, o Dobot Magician venceu o prémio “CES 2018 Innovation Award” da Consumer Technology Association e o “Global Design Award 2018” da Industrie Forum Design (Dobot, 2024b).

O Dobot Magician possui, de origem, uma aplicação que pode ser utilizada para o envio de instruções e para programação do robô denominada “DobotBlockly”. A utilização desta aplicação é opcional dada a possibilidade de comunicar com recurso a diversas linguagens de programação como, por exemplo, o Python. O envio de comandos para o braço robótico pode surgir de várias fontes, podendo ser por Bluetooth, Wi-Fi ou ligação *Universal Serial Bus* (USB). Na Figura 2.16 pode ser visualizado o Dobot Magician com os seus diversos acessórios e elementos terminais.

Figura 2.16 – Dobot Magician com acessórios e elementos terminais, adaptada de STEM Education Works (2024).

2.3 Projetos de robótica

Neste tópico são enumerados e descritos alguns projetos, elaborados em âmbito académico, com a utilização de elementos da RE, como por exemplo o microprocessador Arduino, a placa Raspberry Pi e o braço robótico Dobot Magician.

Sistema automático de classificação e transporte de frutas e vegetais

A vertente educacional do Dobot Magician torna-o atrativo em projetos inovadores para a simulação e prototipagem de soluções. Um exemplo de utilização do Dobot Magician num projeto foi criado pelos autores H. Li et al. (2023) em que foi proposto um sistema automático de classificação e transporte de frutas e vegetais para apresentar uma solução em resposta a uma necessidade levantada pelo setor agrícola local. O sistema era composto por um computador que executava a função de unidade de controlo que coordenava o Dobot Magician, um tapete transportador e uma câmara industrial. O

controlo do braço robótico foi desenvolvido com recurso ao *Application Programming Interface* (API) do Dobot, com o nome "DobotDllType", que permitiu estipular os movimentos do braço com o modo *Point to Point* (PTP). O tapete transportador era acionado com um motor de passo com uma placa de controlo do modelo A4950. Relativamente à análise das imagens captadas pela câmara industrial foi utilizado o modelo *You Only Look Once* (YOLO) versão 5s. O YOLO foi desenvolvido para a deteção e segmentação de imagens. Criado pelo Joseph Redmon e Ali Farhadi na Universidade de Washington em 2015. Atualmente o módulo é mantido é desenvolvido pela Ultralytics (Ultralytics, 2023).

O processo dividia-se em três fases:

- Transporte pelo tapete transportador: transporta a fruta e os legumes até à zona de deteção;
- Deteção do artigo: a câmara envia uma imagem do artigo para o computador que utiliza o modelo YOLO para identificar o tipo de artigo e transmite a informação de ação para o braço robótico;
- Classificação: após o envio da instrução pelo computador, o Dobot Magician movimenta o artigo para a localização desejada.

Na Figura 2.17 pode ser observado o espaço de trabalho e os elementos utilizados no projeto.

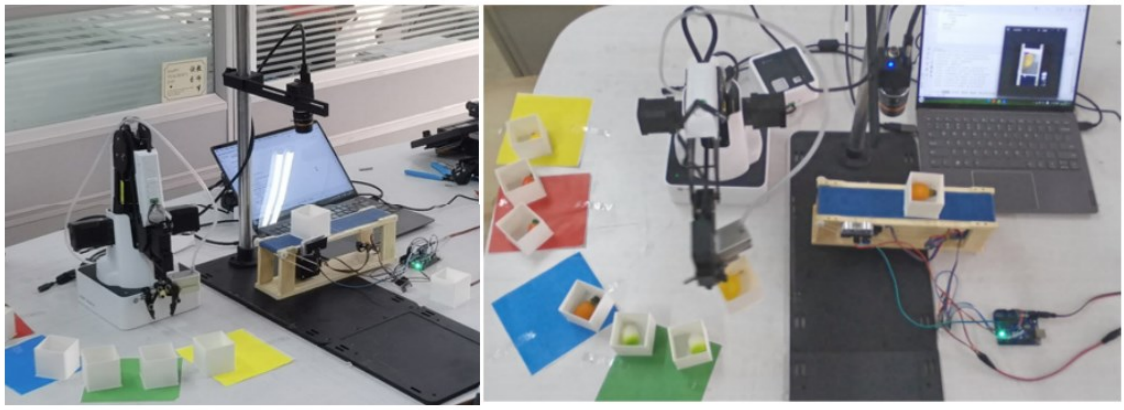

Figura 2.17 – Projeto de sistema automático de classificação e transporte de frutas e vegetais, adaptada de H. Li et al. (2023).

Braço robótico para a deslocação de objetos por controlo de voz para apoio a Veteranos com deficiência visual

Os autores Natharani et al. (2021) implementaram outro projeto composto por um braço robótico para a deslocação de objetos, por controlo de voz, para apoio a Veteranos com deficiência visual. Este projeto, à semelhança do anterior, também utiliza o Dobot Magician com a garra como elemento terminal. Neste trabalho uma câmara envia imagens da envolvência para o controlador, que por sua vez efetua a deteção de objetos, utilizando também o modelo YOLO. O sistema contém uma entrada de áudio

que capta os comandos proferidos pelo utilizador para serem descodificados com recurso à biblioteca de reconhecimento de fala da Google *Speech to Text.*

Na Figura 2.18 pode ser observado o *framework* proposto para o funcionamento do modelo.

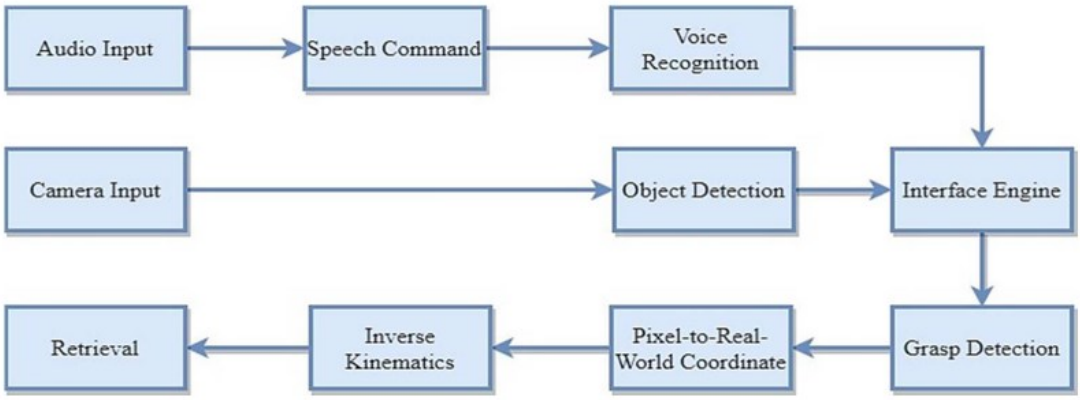

Figura 2.18 – Framework do projeto de braço robótico para a deslocação de objetos por controlo de voz para apoio a Veteranos com deficiência visual, adaptada de Natharani et al. (2021).

Controlo de um robô com quatro GDL para mover objetos em função da cor e peso utilizando a lógica de controlo Fuzzy

Os autores Nugroho et al. (2023) utilizaram o braço robótico Dobot Magician, com o controlo realizado pela lógica *Fuzzy*, para a movimentação de objetos mediante a cor e peso. Adicionalmente foi utilizado um Arduino Mega 2560 como controlador, uma célula de carga do modelo HX711 como sensor de peso dos objetos e o sensor TCS-3200 para a deteção de cores. Os *softwares* utilizados para a implementação do sistema foram o Dobot Studio para a programação do robô e o Matlab para a implementação do controlo com lógica *Fuzzy*, a qual foi posteriormente incorporada no Arduino.

A utilização da lógica de controlo *Fuzzy* trouxe diversas vantagens, nomeadamente: a capacidade de utilizar entradas analógicas em regras lógicas que permitem ao robô responder de forma apropriada; e a facilidade de uma implementação no microcontrolador Arduino Mega 2560.

O funcionamento do sistema inicia-se com a chegada de uma peça, com uma determinada cor e peso, sobre o tapete transportador. A peça, ao ser detetada por um sensor de proximidade, permanece imóvel no local de recolha. O Dobot Magician, com o elemento terminal de sucção, desloca a peça para o sensor de leitura de cor e de seguida para o sensor de verificação do peso. Por fim, o Dobot Magician transportará a peça para o local apropriado, em função da cor e peso. Na Figura 2.19 podem ser observados os vários elementos constituintes do sistema.

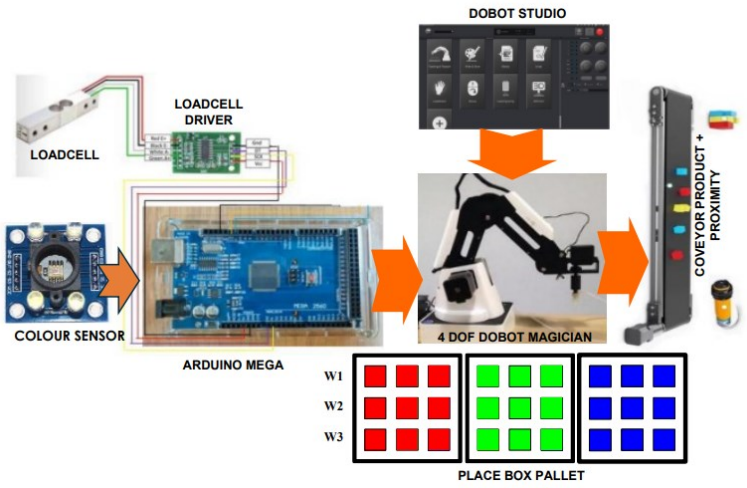

Figura 2.19 – Elementos do sistema de controlo de robô de quatro GDL com lógica *Fuzzy*, adaptada de Nugroho et al. (2023).

Com este projeto, os autores concluíram que a implementação da lógica de controlo *Fuzzy* melhorou a proficiência do Dobot Magician através da capacidade de executar respostas em função de duas variáveis de entrada (cor e peso). Os dados recolhidos dos testes permitiram concluir que o sistema possuía um erro de 1,8% na precisão das ações executadas. O erro presente era consequência da leitura da cor por parte do sensor devido a diferenças causadas pela intensidade da luz produzida pelas lâmpadas. Na Figura 2.20 é possível observar o espaço de trabalho do sistema.

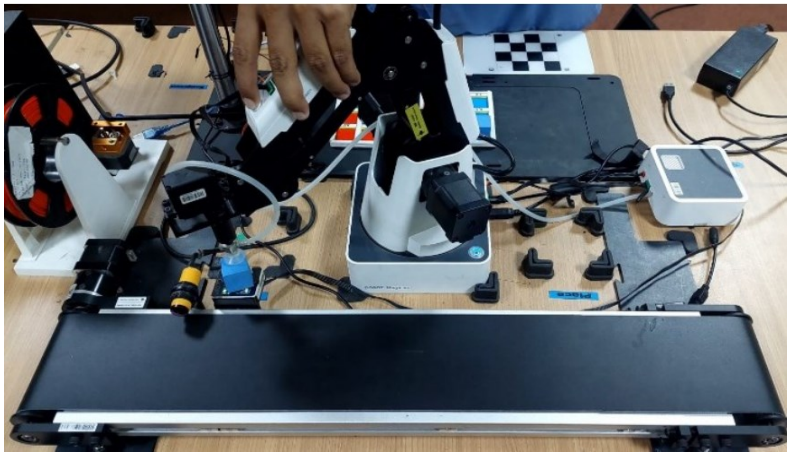

Figura 2.20 – Espaço de trabalho do sistema de controlo de robô de quatro GDL com lógica *Fuzzy*, adaptada de Nugroho et al. (2023).

Desenvolvimento de um protótipo de veículo autónomo a partir de um carro telecomandado

No âmbito de um trabalho de projeto para obtenção de grau de Mestre em Engenharia Mecânica, Marçal (2023) desenvolveu o protótipo de um veículo autónomo a partir de um carro telecomandado. O objetivo do projeto consistia em produzir uma réplica de um veículo autónomo, com algumas capacidades estabelecidas, nomeadamente a capacidade de se deslocar entre várias coordenadas geográficas e evitar a colisão com qualquer obstáculo no seu percurso.

O sistema era composto por vários sensores, um módulo Bluetooth e um sistema *Global Positioning System* (GPS). O Arduino Mega foi escolhido como o microprocessador de controlo. Na Figura 2.21 é possível observar o protótipo do veículo telecomandado modificado para incorporar os elementos eletrónicos de automação.

Figura 2.21 – Protótipo de veículo autónomo modificado, adaptada de Marçal (2023).

O sistema operava com o sistema de navegação por coordenadas geográficas com recurso a um módulo GPS e um sensor magnetómetro. Com a introdução das coordenadas geográficos do percurso, o veículo deslocava-se pelo caminho mais curto entre os pontos. Sempre que era detetado, pelos sensores ultrassónicos, um obstáculo no percurso, o veículo efetuava o contorno ao obstáculo e regressava à trajetória estabelecida para a coordenada seguinte. O módulo Bluetooth era utilizado exclusivamente para a monitorização de dados do veículo.

Desenvolvimento de um robô para combate a fogos num cenário de simulação

O autor Marques (2017) propôs o desenvolvimento de um robô para o combate a fogos num cenário de simulação no âmbito do Trabalho Final de Mestrado para obtenção do grau de mestre em Engenharia Mecânica. O objetivo do projeto consistia no desenvolvimento de um robô móvel demonstrativo com a capacidade de explorar um cenário desconhecido para a deteção e extinção de um foco de incêndio.

O sistema era composto por um *rangefinder* RPLIDAR 360º, responsável pelo mapeamento do local, uma câmara termográfica FLIR LEPTON para a deteção de chama, uma ventoinha para extinção da chama, motores e um Raspberry Pi como unidade de processamento. Na Figura 2.22 pode ser observado o protótipo do robô desenvolvido.

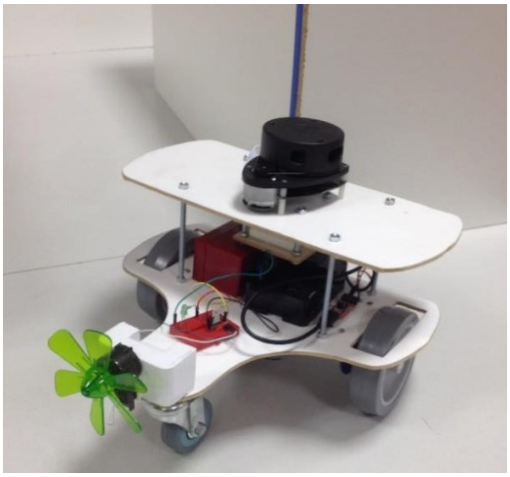

Figura 2.22 – Protótipo de robô para combate a fogos num cenário de simulação, adaptada de Marques (2017).

O autor do projeto concluiu que o robô conseguia fazer a deteção da chama com facilidade a uma distância de dois metros, sendo a taxa de sucesso de 100% nessas condições. Com uma probabilidade de 77,5%, o robô parava para efetuar a extinção da chama no raio de segurança de 10 cm a 20 cm, definidos pelo autor.

Robô de limpeza de painéis solares

Com o objetivo de explorar a aplicabilidade e melhoria de eficiência na utilização de um robô para a limpeza de painéis solares controlado por *Internet of Things* (IoT) os autores Sutam et al. (2023) propuseram um trabalho de projeto para a construção e programação do robô.

O sistema era composto por uma estrutura em aço inoxidável com uma escova de limpeza, dois motores para a deslocação e outros motores para o controlo da escova. O microprocessador utilizado no projeto foi um ESP32 com as funcionalidades de Wi-Fi e Bluetooth. O módulo utilizado para o controlo dos motores foi o IBT-2. Na Figura 2.23 é possível observar o protótipo da estrutura do robô e os componentes eletrónicos utilizados.

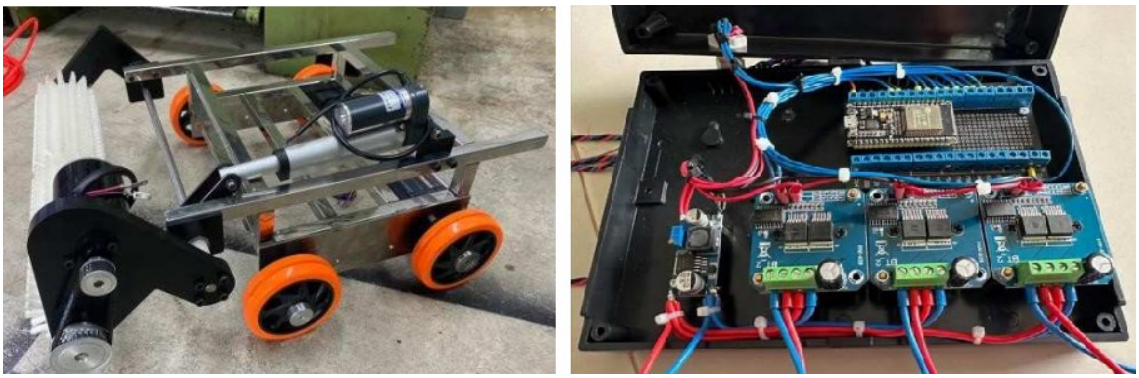

Figura 2.23 – Protótipo de robô de limpeza de painéis solares, adaptada de Sutam et al. (2023).

A programação do robô foi efetuada com recurso à programação Arduino e foi utilizada a plataforma Ant.io para a comunicação com o equipamento através da internet. Para o funcionamento do robô foi implementado um cenário com doze pontos distribuídos por uma área de 1,65 m^2 do painel solar a limpar.

2.4 Programação de robôs

A programação computacional exige do programador um conjunto de aptidões, entre as quais a capacidade de resolução de problemas e o pensamento lógico, que a tornam um desafio, especialmente no início da sua aprendizagem.

A utilização da programação em blocos é frequentemente recomendada para a iniciação na programação de robôs, devido à sua facilidade de aplicação e versatilidade. Exemplos destes ambientes são o Blockly, Scratch e Alice que podem ser aplicados em kits de robótica (Yilmaz Ince & Koc, 2021). Na Figura 2.24 pode ser observado um exemplo de um programa na linguagem Blocky.

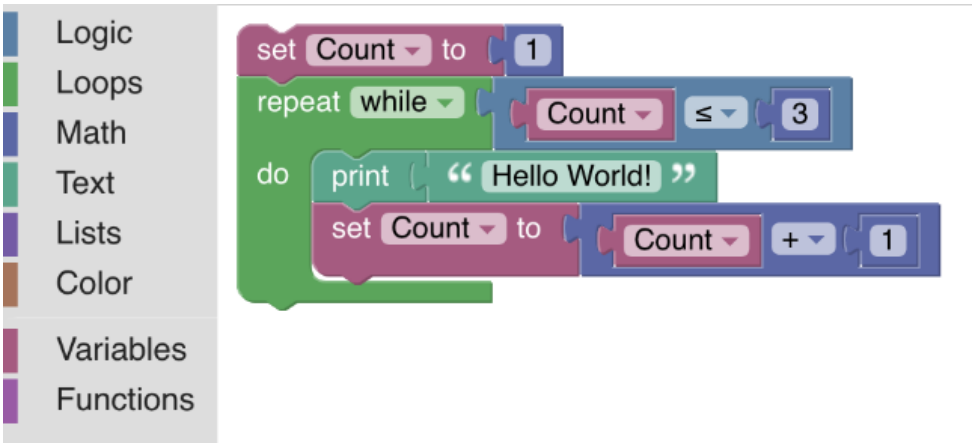

Figura 2.24 – Exemplo de programação Blocky, adaptada de Sandbox Academy (2024).

Existem diversas linguagens de programação, cada uma com as suas vantagens e aplicações. Algumas das linguagens de programação mais comuns em 2023, segundo a Vailshery (2024), foram o JavaScript, HTML/CSS, Python, SQL e TypeScript. Para a aplicação na área da robótica, as linguagens de programação historicamente mais utilizadas são o C, C++, Python, Java, C#, Matlab, Lisp e Pascal. Embora algumas destas linguagens de programação se encontrem desatualizadas, como por exemplo o Pascal e o Lisp, continuam a existir aplicações na indústria com a sua utilização (Biba, 2022).

A escolha da metodologia e linguagem de programação a aplicar num projeto de robótica assenta essencialmente nos objetivos do projeto, no tipo de programação ou código lido pelo processador utilizado e nas ferramentas que a própria linguagem de programação fornece.

Embora não permitam a programação de robôs físicos, existem diversas plataformas virtuais para criação e programação de robôs como por exemplo o Robot Virtual Worlds, Tinkercad Circuits e Webots. Estas soluções revelam-se úteis em fases preliminares e de testes de projeto de robótica para a análise de soluções com menor custo (Yolcu & Demirer, 2023).

O avanço da capacidade de aprendizagem por parte das máquinas, conhecido como o *machine learning*, tem potenciado diversas capacidades de melhoria de produtividade e eficiência da indústria. Uma das potencialidades é a visão computacional, que consiste na capacidade de uma máquina observar e interpretar o ambiente físico ao seu redor. Esta capacidade é normalmente possível com recurso a imagens de fotografia e vídeo. Através da visão computacional, uma máquina pode identificar objetos e estimar a sua localização bem como as suas propriedades. Esta identificação pode ser utilizada posteriormente para a execução de ações por parte de um robô ou equipamento semelhante perante as instruções implementadas (Jacobs & Nel, 2023).

2.5 Robótica no setor energético

A utilização e implementação crescente da robótica nos mais diversos setores da indústria tem contribuído para elevar os padrões de qualidade e eficiência. Os benefícios para o setor energético não se verificam apenas do lado da produção, mas também na sua utilização. A energia representa um pilar crucial na indústria e na sociedade atual dada a vasta utilização e dependência energética. Desde o final da Segunda Guerra Mundial o desenvolvimento mundial cresceu rapidamente, o que originou um aumento significativo do consumo de energia. Embora a produção de energias renováveis tenha aumentado ao longo dos últimos 10 anos, ainda representa uma parcela muito pequena, 7%, da energia total produzida. Prevendo-se o contínuo aumento da exigência energética, a situação não é sustentável. Para a resolução do problema, três abordagens têm sido apresentadas e implementadas nos países mais desenvolvidos, nomeadamente (Caetano et al., 2017):

- Redução da quantidade de energia utilizada através da melhoria da eficiência das redes de distribuição de energia;
- Utilização de energias renováveis;
- Desenvolvimento de novas formas de armazenamento de energia de forma eficiente.

Como exemplo de investimento nas energias renováveis, Portugal estipulou a aplicação de 23 biliões de euros de fundos europeus para o financiamento, até 2029, de projetos de transição energética e aplicação de energias renováveis na produção de eletricidade (S&P Global: Country/Territory Report - Portugal., 2023).

A robótica pode contribuir de diversas formas na aplicação das medidas de sustentabilidade energética nomeadamente na automatização de tarefas de produção e manutenção de equipamentos relacionados com a produção energética como oleodutos e condutas, turbinas eólicas e painéis solares e fotovoltaicos. A robótica apresenta-se como uma mais valia dada a sua capacidade de trabalho a elevadas velocidades, melhorando a produtividade (Verl, 2023). A produção e utilização de veículos elétricos tem vindo a sofrer um grande aumento nos últimos anos. A utilização da robótica na produção das viaturas tem vindo a revelar-se fundamental e amplamente estudada por forma a aplicar novas metodologias. O autor Sharma (2024), propôs um trabalho para a melhoria do tempo e eficiência de inspeção de baterias de lítio, após produção, com destino ao ramo automóvel através da utilização de robôs e visão computacional. Atualmente a inspeção é efetuada de forma manual o que permite a ocorrência de erros, consome mais tempo e representa um estrangulamento na produção. O autor concluiu, com o estudo, que a aplicação da robótica na inspeção das baterias contribui de forma significativa para a melhora da eficiência e qualidade do produto.

2.6 Energia Solar

A origem do interesse no estudo da energia provém de ser um dos principais fatores para o desenvolvimento económico da sociedade atual. Fruto desta necessidade de progresso, o recurso e utilização de energias de fontes não renováveis provocam danos irreversíveis no ambiente. Conscientes na necessidade de mitigar os problemas ambientais, diversos organismos governamentais, instituições e agências promoveram uma profunda revisão das políticas de desenvolvimento. A União Europeia (UE) aprovou, em dezembro de 2019, o Acordo Verde Europeu. O acordo é constituído por diversos pontos com o objetivo de resolver os problemas relacionados com a poluição ambiental e alterações climáticas. De entre as medidas, destacam-se o compromisso da EU em reduzir as emissões de carbono, em pelo menos 55% até 2030 e tornar-se o primeiro continente climaticamente neutro até 2050 (European Commission, 2019).

Uma das principais estratégias para o êxito dos compromissos europeus, consiste na promoção e incentivo de utilização de energias renováveis. Das diversas energias renováveis disponíveis, a energia solar tem vindo a revelar-se como um grande trunfo estratégico devido à diminuição consistente dos custos desde o início de 2010 (Filiz Baştürk, 2024).

A energia presente na luz proveniente do sol que incide na Terra durante noventa minutos tem a capacidade de garantir todas as necessidades de energia de todos os seres humanos. Com o objetivo de tirar o maior proveito possível desta energia, a

ciência tem vindo a escrutinar as suas potencialidades e a desenvolver equipamentos capazes de converter a energia proveniente do sol em energia que possa ser utilizada diretamente. A energia proveniente do sol pode ser convertida em energia térmica, conhecida como o solar térmico ou em energia elétrica, conhecida como solar fotovoltaica. A instalação de sistemas solares fotovoltaicos tem sofrido um crescimento exponencial, quer para soluções de pequena dimensão como por exemplo habitações, quer para soluções industriais através de parques solares de elevada dimensão (Carter, 2018). Segundo o documento Energia em Número, edição de 2023, do Observatório da Energia et al. (2023) a potência instalada relativa ao solar fotovoltaico foi a que mais cresceu no ano de 2022, atingindo os 2562 MW. Na Figura 2.25 é possível observar a evolução de utilização das energias renováveis hídrica, eólica, biomassa e solar em Portugal entre 2011 e 2022.

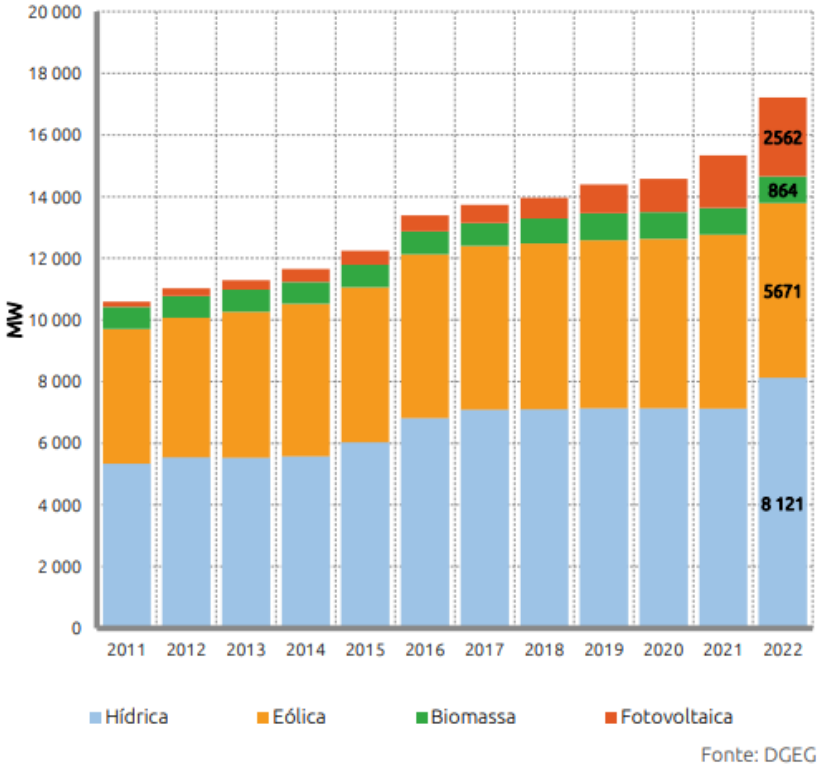

Figura 2.25 – Evolução de utilização de energias renováveis em Portugal entre 2011 e 2022, adaptada de Observatório da Energia et al. (2023).

As células solares são os elementos do painel solar fotovoltaico responsáveis pela conversão da energia luminosa em energia elétrica. Além das células solares, um painel fotovoltaico genérico é composto por outros componentes com as funções de proteger as células solares e promover a sua eficiência. Na Figura 2.26 encontram-se representados os diversos componentes constituintes do painel fotovoltaico. Na parte inferior do painel encontra-se um suporte, também conhecido como fundo protetor ou *backsheet*, com cor branca e composta pelas matérias Tedlar (fluoreto de polivinila) e tereftalato de polietileno. Seguidamente encontram-se duas placas de acetato-vinilo de

etileno (EVA) com as células solares entre as mesmas. Por fim é colocado um vidro na parte superior do painel. Geralmente todos os componentes constituintes do painel são incorporados numa moldura de alumínio (Padoan et al., 2019).

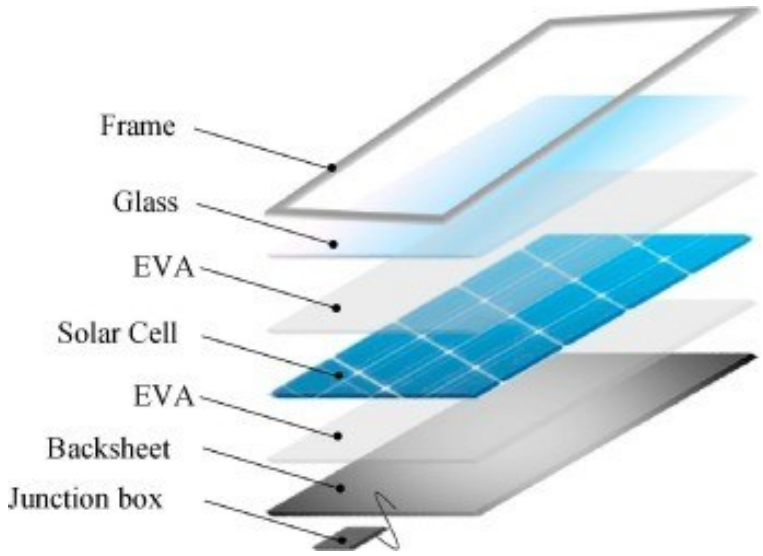

Figura 2.26 – Constituição genérica de um painel solar fotovoltaico, adaptada de Padoan et al. (2019).

3 Conceção do protótipo do processo

3.1 Laboratório de robótica

Grande parte do conhecimento humano é adquirido através da prática. Tomando como exemplos duas ações básicas da vida quotidiana, aprende-se a escrever, escrevendo e a cozinhar, cozinhando (Piñeros Glasscock, 2022). A adoção da aplicação prática dos ensinamentos teóricos, em todos os níveis do sistema educativo, enriquece a sua qualidade e torna-o diferenciador.

O ISEL é reconhecido pela sua forte componente prática no ensino, estando equipado com diversos laboratórios das áreas de engenharia lecionadas (ISEL, 2024). Os laboratórios são utilizados em contexto da componente prática de unidades curriculares e em projetos de investigação e inovação.

Incorporado no DEM do ISEL, o Laboratório de Robótica encontra-se equipado com diversos equipamentos e *softwares* de robótica e eletrónica, destacando-se os de maior dimensão, nomeadamente um tapete transportador de paletes em circuito fechado, um armazém automático em carrossel e um robô manipulador Scorbot ER IX. Para além destes existe uma diversidade de outros equipamentos como dois robôs Dobot Magician, microcontroladores, eixos lineares, bem como diverso material eletrónico e acessórios para criação de projetos em âmbito académico. Os equipamentos são utilizados para as componentes práticas de unidades curriculares como Robótica Industrial e Eletrónica e Instrumentação dos cursos de Engenharia Mecânica.

O presente trabalho de projeto foi desenvolvido nas instalações do Laboratório de Robótica pela utilidade e necessidade dos diversos equipamentos presentes bem como pelo amplo espaço, que permite o desenvolvimento do projeto sem interferência na operação quotidiana do laboratório.

3.2 Descrição do processo, equipamentos e acessórios

3.2.1 Processo

A definição do processo sofreu uma evolução constante durante a fase de preparação e elaboração do projeto. Numa fase inicial foram determinados os objetivos principais do processo e os equipamentos utilizados. Em paralelo à estipulação e descrição das várias etapas do processo surgiram novas ideias e propostas para aumentar o rigor do processo e torná-lo o mais eficiente possível.

A automação do modelo de produção de painéis solares à escala reduzida encontra-se centrada em torno de duas tipologias de equipamentos principais. Como primeiro elemento, responsável pela manipulação das várias peças constituintes dos painéis solares foram escolhidos dois robôs Dobot Magician. Como segundo elemento, responsável pela deslocação dos painéis solares ao longo da linha de produção, foi escolhido um motor de passo do modelo Nema 17.

Como elementos secundários, foram definidos vários equipamentos para garantir o fornecimento dos elementos constituintes dos painéis solares bem como a melhoria de eficiência do processo. Exemplos destes equipamentos são os alimentadores das peças dos painéis solares, os detetores de fim de curso do motor de passo, a câmara para a verificação de qualidade e o mecanismo de expulsão do painel solar no final da produção, entre outros.

Para efeitos de simplificação de nomenclaturas, daqui em diante, o Dobot Magician com o elemento terminal ventosa atuada por uma bomba de vácuo é designado como “Dobot 1” e o Dobot Magician com o elemento terminal I-Extruder é designado como “Dobot 2”.

A disposição dos equipamentos no protótipo do processo encontra-se representada na Figura 3.27.

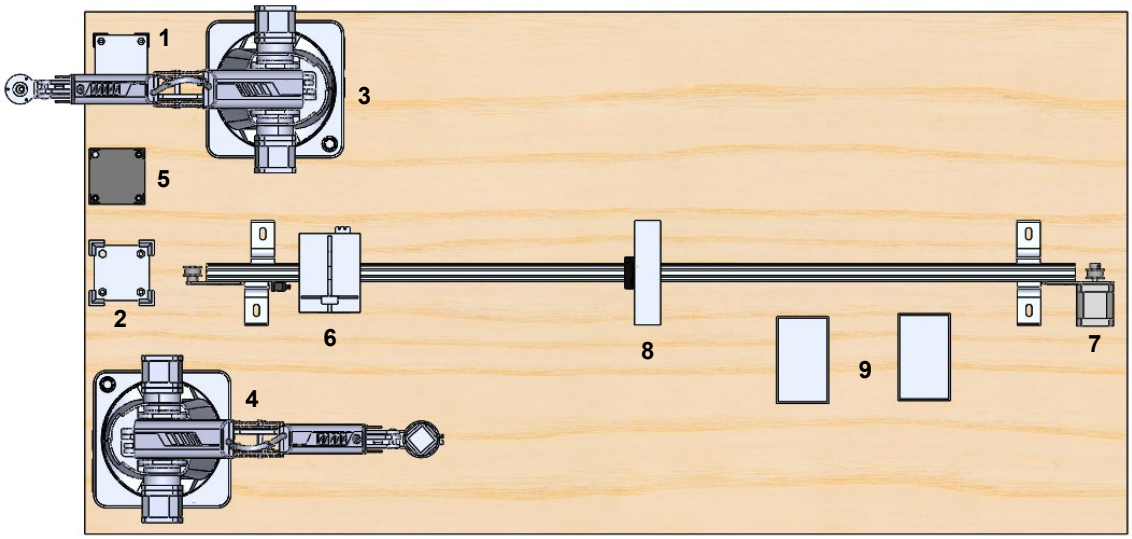

Figura 3.27 – Disposição dos equipamentos no protótipo do processo.

O processo automático de montagem de painéis solares à escala reduzida inicia-se com a movimentação da base do painel solar desde o seu alimentador (ponto 1 da Figura 3.27) até à mesa de montagem (ponto 2 da Figura 3.27), pelo Dobot 1 (ponto 3 da Figura 3.27). Seguidamente o Dobot 2 (ponto 4 da Figura 3.27), aplica uma camada de cola sobre a base do painel solar. Após a aplicação da cola, o Dobot 1 movimenta a segunda peça constituinte do painel solar, desde o seu alimentador (ponto 5 da Figura 3.27) até à base do painel solar, efetuando desta forma a união das duas peças. A montagem das peças e aplicação da cola ocorre sobre a mesa de montagem. Após a montagem, o Dobot 1 desloca o painel solar para a mesa de transporte (ponto 6 da Figura 3.27), sobre o eixo linear. Um motor de passo (ponto 7 da Figura 3.27) efetua a deslocação da mesa ao longo do eixo linear. Na deslocação, o painel solar efetua uma passagem por uma zona de controlo de qualidade, onde se encontra instalada uma câmara que capta uma fotografia do painel solar para tratamento computacional (ponto 8 da Figura 3.27). Seguidamente o painel solar continuará a deslocação ao longo do eixo linear. Esse deslocamento depende do resultado do controlo de qualidade, podendo ser a aceitação da peça ou a sua rejeição, e termina junto a um recetor de peças aceites ou rejeitadas (ponto 9 da Figura 3.27). A expulsão é efetuada por um segundo motor de passo localizado na mesa de transporte.

Finalizado o processo este pode ser reiniciado, com funcionamento em ciclo contínuo. Na Figura 3.28 pode ser observado o fluxograma do processo descrito.

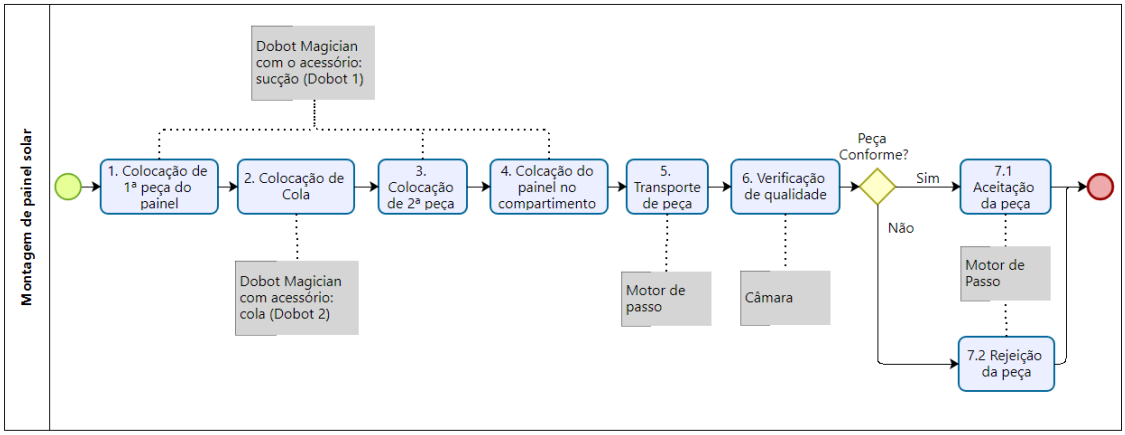

Figura 3.28 – Fluxograma do processo.

3.2.2 Arquitetura de hardware e protocolos de comunicação do sistema

A existência de automatismos só é possível através da comunicação entre equipamentos. O comando para uma ação parte maioritariamente de um estímulo, através de um sinal, vindo do próprio equipamento ou de outro. Os protocolos de comunicação surgem como um mecanismo para que possa ocorrer a troca de informação entre vários dispositivos. O sucesso para uma comunicação está assente

nas regras específicas do protocolo que deverão ser compreendidas pelo emissor e pelo recetor da informação.

A arquitetura de hardware e protocolos de comunicação do processo do presente projeto pode ser observada na Figura 3.29. O elemento principal de emissão e receção de informação é o computador, é neste que o utilizador pode iniciar ou interromper o processo automático e verificar o seu estado. Os equipamentos que se encontram ligados diretamente ao computador são o Arduino UNO, os dois robôs Dobot Magician e a Câmara. O protocolo de comunicação utilizado entre o computador e os equipamentos é o USB.

O Dobot 2 efetua a comunicação com o I-Extruder através de uma saída digital de *input/output* (I/O), sendo que neste caso ocorre apenas a saída de sinais do Dobot Magician para o I-Extruder, não havendo retorno de informação.

O Arduino UNO encontra-se conectado a uma placa de *Computer Numerical Control* (CNC) conhecida como CNC Shield V3 através de diversas ligações digitais I/O. A placa CNC Shield V3 foi projetada para soluções específicas envolvendo motores de passo, tendo o objetivo de simplificar as ligações necessárias. A grande maioria das ligações da placa possui conexões diretas a saídas na placa Arduino UNO. As correspondências das ligações do CNC Shield V3 e a placa Arduino UNO podem ser observadas na Figura 3.30. Na placa CNC Shield V3 encontram-se montados dois *drivers* A4988 para controlo dos dois motores de passo Nema 17 e D8-MOTOR80, sendo que a comunicação entre estes é efetuada através de ligações digitais I/O. Os sensores de fim de curso encontram-se ligados à placa CNC Shield V3 através de ligações digitais I/O.

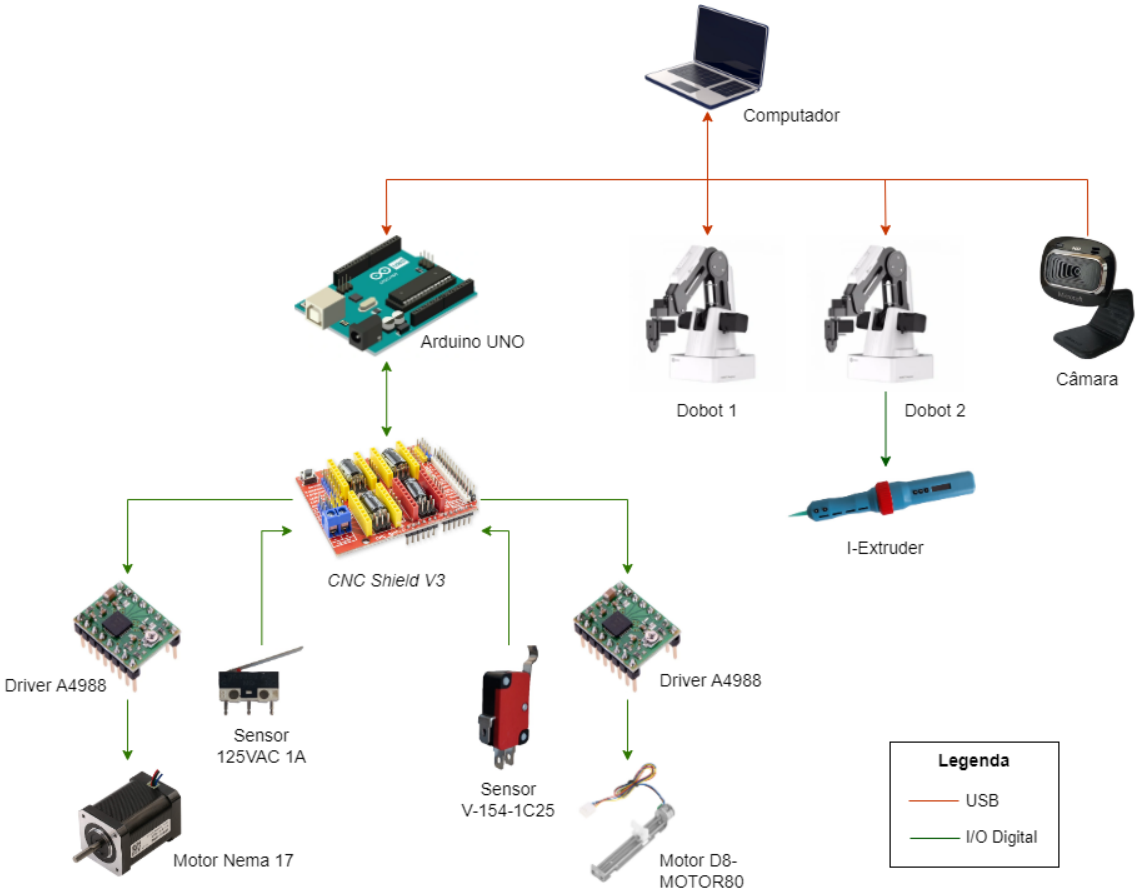

Figura 3.29 – Arquitetura de hardware e protocolos de comunicação.

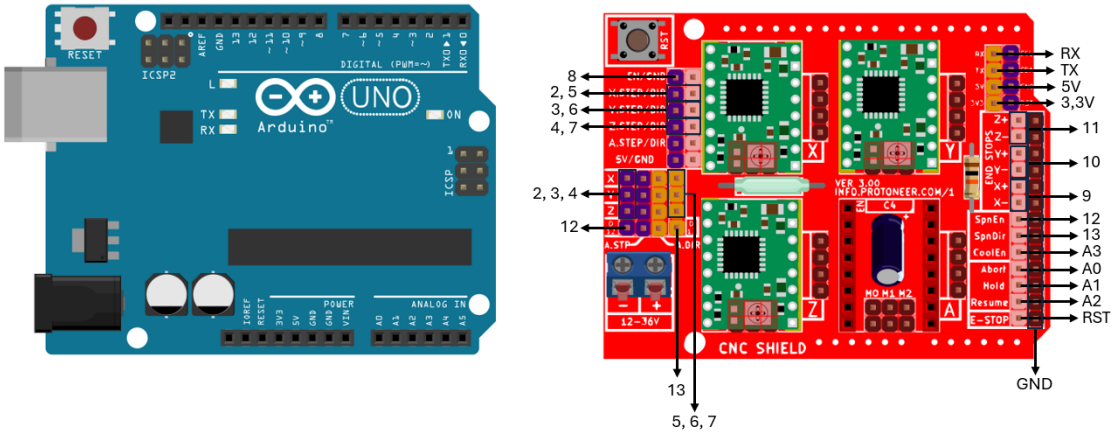

Figura 3.30 – Ligações entre a placa CNC Shield V3 e Arduino UNO, adaptada de Mikroelectron (2024).

3.2.3 Equipamentos e acessórios

Dobot Magician

O Dobot Magician, à semelhança de outros robôs educacionais, possui componentes estruturais básicos para a realização de diversas operações, nomeadamente, três

juntas, diversos elementos terminais, motores e uma base que garante a sua estabilidade. Na Figura 3.31 é possível observar os principais componentes do robô.

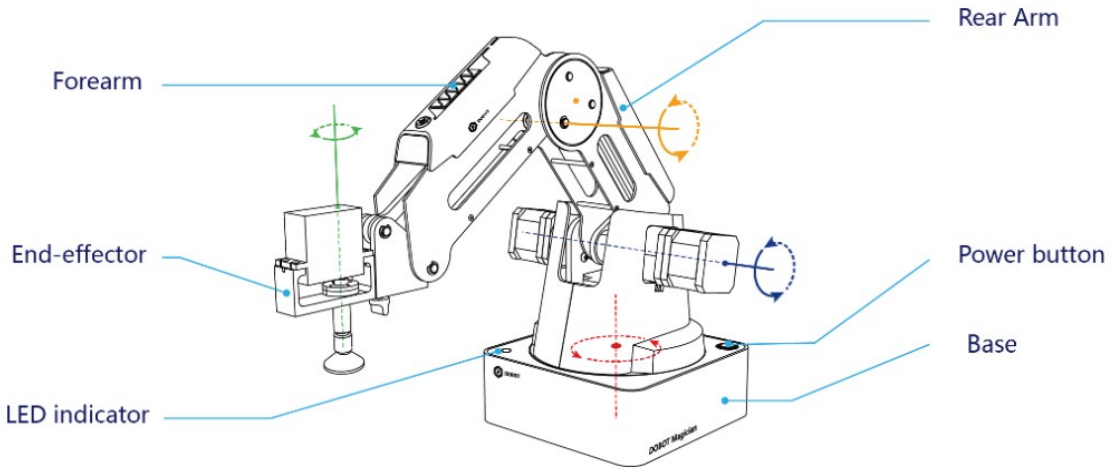

Figura 3.31 – Componentes principais do robô Dobot Magician, adaptada de Dobot Magician User Guide (2020).

O espaço de trabalho do robô encontra-se limitado pelo alcance dos elos e pelas barreiras físicas impostas pelos próprios componentes do robô.

No plano vertical do robô, a segunda junta permite uma amplitude de movimentos de 85º, com origem no centro dos motores laterais do robô. A terceira junta permite uma amplitude de movimentos de 100º, com origem na junta dos dois elos.

No plano horizontal do robô, a junta 1 permite uma amplitude de rotação de 180º com origem no ponto central do robô e um alcance máximo de 320 mm. Na Figura 3.32 é possível observar, através da coloração amarela, o espaço de trabalho do robô Dobot Magician no plano vertical e horizontal (*Dobot Magician User Guide*, 2020).

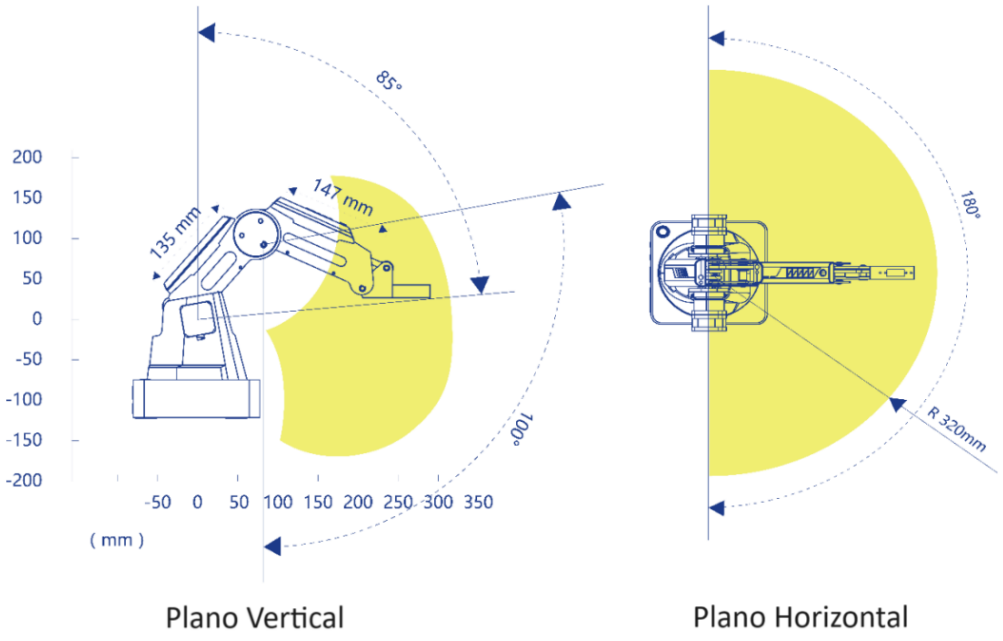

Figura 3.32 – Espaço de trabalho do robô Dobot Magician nos planos vertical e horizontal, adaptada de Dobot Magician User Guide (2020).

As características técnicas do robô Dobot Magician podem ser observadas no Quadro 3.1.

Quadro 3.1 – Características técnicas do robô Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Característica		Valor
Carga Máxima		500 g
Velocidade Máxima (Com carga de 250 g)	Velocidade de rotação das juntas 1, 2 e 3.	320 º/s
	Velocidade de rotação dos servomotores.	480 º/s
Precisão de movimentos repetidos		0,2 mm
Alimentação elétrica		12 V / 7 A DC
Protocolos de comunicação		USB, Wi-Fi e Bluetooth
Entradas e saídas		20 extensões I/O

As interfaces do Dobot Magician encontram-se localizadas na parte traseira da base e no elo dianteiro. Na base encontram-se essencialmente as portas de entradas e saídas de interfaces de comunicação, acessórios dos elementos terminais e a alimentação elétrica do robô. No elo dianteiro localizam-se as interfaces de comunicação com os elementos terminais passíveis de serem instalados no robô (*Dobot Magician User Guide*, 2020). Na Figura 3.33 podem observar-se as interfaces localizadas na base do robô. No Quadro 3.2 encontram-se descritas as características das interfaces.

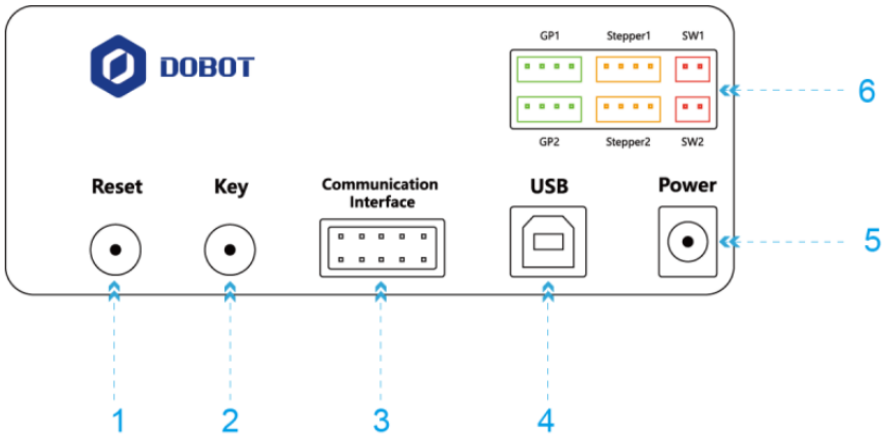

Figura 3.33 – Interfaces localizadas na base do Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Quadro 3.2 – Descrição das interfaces localizadas na base do Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Número	Descrição	
1	Botão de reiniciar. Reinicia o programa carregado na unidade microcontrolador. Durante o reiniciar, o LED indicador na base do robô permanece com cor amarela, durante aproximadamente 5 segundos, após os quais, se o reiniciar for bem-sucedido, altera para a cor verde.	
2	Botão funcional: • Pressionado durante uma curta duração: Robô inicia o programa offline carregado; • Pressionado durante dois segundos: Robô inicia a execução da instrução *Home*.	
3	Interface de comunicação/ Interface UART: Conexão por Bluetooth, Wi-Fi, entre outros.	
4	Interface USB: Ligação com computador.	
5	Interface de ligação de alimentação elétrica.	
6	SW1	Interface de alimentação da bomba de ar. Saída de 12 V para utilização diversa.
	SW2	Saída de 12 V para utilização diversa.
	Stepper1	Interface para ligação a motor de passo. Ligação para a interface do módulo de impressão 3D.
	Stepper2	Interface para ligação de motor de passo.
	GP1	Interface de sinal da bomba de ar, sensores de cor e infravermelhos. Interface de ligação de equipamento customizado.
	GP2	Interface de ligação de equipamento customizado.

Na Figura 3.34 podem ser observadas as interfaces localizadas no elo dianteiro do robô.

No Quadro 3.3 encontram-se descritas as características das interfaces.

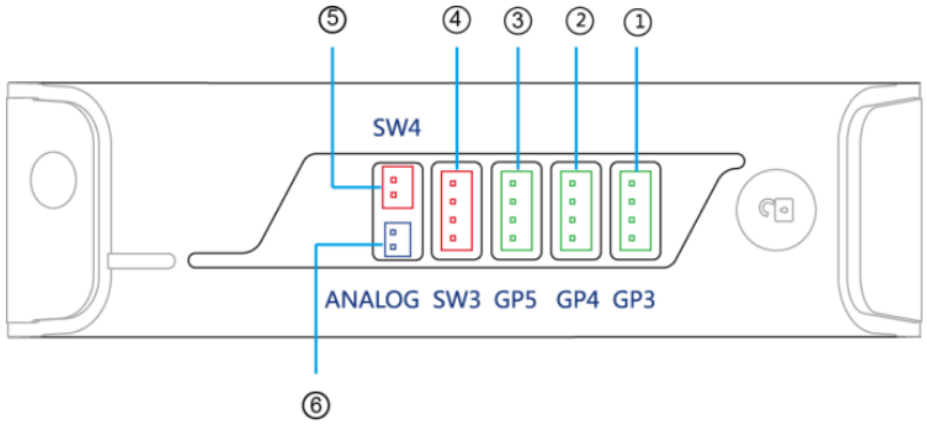

Figura 3.34 – Interfaces localizadas no elo dianteiro do Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Quadro 3.3 – Descrição das interfaces localizadas no elo dianteiro do Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Número	Descrição	
1	GP3	Interface de ligação de elemento terminal. Interface do Eixo-R do servo (eixo de rotação do elemento terminal). Interface de ligação de equipamento customizado.
2	GP4	Interface de ligação de equipamento de calibração *auto-levelling.* Interface de ligação de equipamento customizado.
3	GP5	Interface de sinal para gravação a laser. Interface de ligação de equipamento customizado.
4	SW3	Interface de ligação de terminal quente para o módulo de impressão 3D. Saída de 12 V para utilização diversa.
5	SW4	Interface de ligação de ventilação para o módulo de impressão 3D. Interface de alimentação elétrica para gravação a laser. Saída de 12 V para utilização diversa.
6	ANALOG	Interface de ligação de termístor para o módulo de impressão 3D.

I-Extruder

Projetado para a aplicação, com precisão, de solda, colas e graxas, o I-Extruder é um equipamento extrusor benéfico para diversas aplicações de eletrónica como, por exemplo, a montagem de componentes em circuitos impressos. Na base do seu funcionamento encontra-se um motor de passo que pode ser programado para um deslocamento fixo, garantindo desta forma o depósito de material de forma controlada. Além da funcionalidade de depósito de material, o equipamento possui a capacidade de deslocar e colocar (*Pick & Place*) peças sensíveis ao choque, à contaminação exterior ou de reduzida dimensão. O I-Extruder caracteriza-se por ser compacto, de reduzido peso (105 g) e com operação silenciosa (I-EXTRUDER, 2024a). Além da operação manual do equipamento é possível a automatização de depósito de material através da aplicação de impulsos de 3,3 V e 5 V na entrada *trigger* do equipamento. Na Figura 3.35 é possível observar o equipamento I-Extruder utilizado no projeto, preparado para o depósito de cola.

Figura 3.35 – I-Extruder.

Motor de passo Nema 17

O motor de passo de modelo Nema 17 é frequentemente utilizado para equipamentos de CNC, como por exemplo impressoras 3D, dada a sua robustez e precisão de

funcionamento. O motor possui um binário elevado, contudo, encontra-se projetado para minimizar a vibração e ruído. De seguida encontram-se algumas características do motor (STEPPERONLINE, 2024):

- Motor bipolar;
- Dimensão do veio: 5 mm;
- Binário: 0,54 Nm;
- Ângulo de passo: 1,8 º.

Na Figura 3.36 pode-se observar o motor de passo de modelo Nema 17 utilizado no projeto.

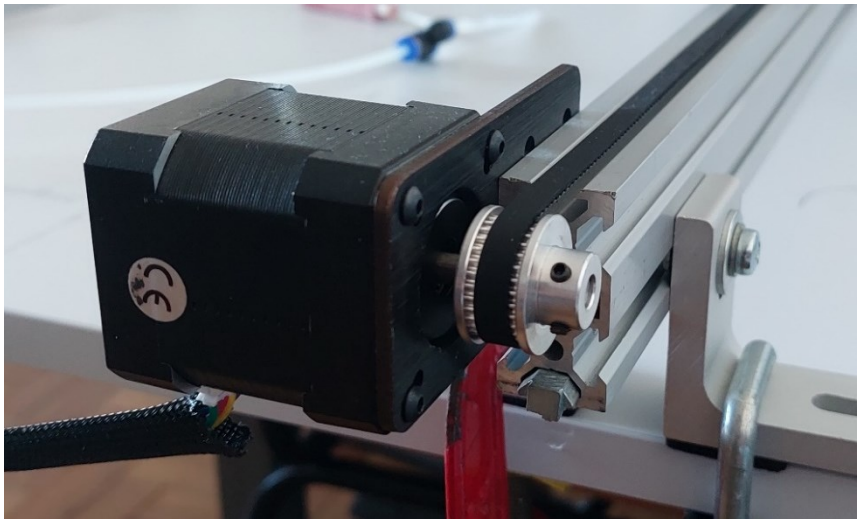

Figura 3.36 – Motor de passo Nema 17.

O motor de passo Nema 17 encontra-se acoplado a uma polia e correia que permitem efetuar o movimento do V-Slot Gantry sobre o eixo linear. Na Figura 3.37 pode ser observado o V-Slot Gantry instalado sobre o eixo linear.

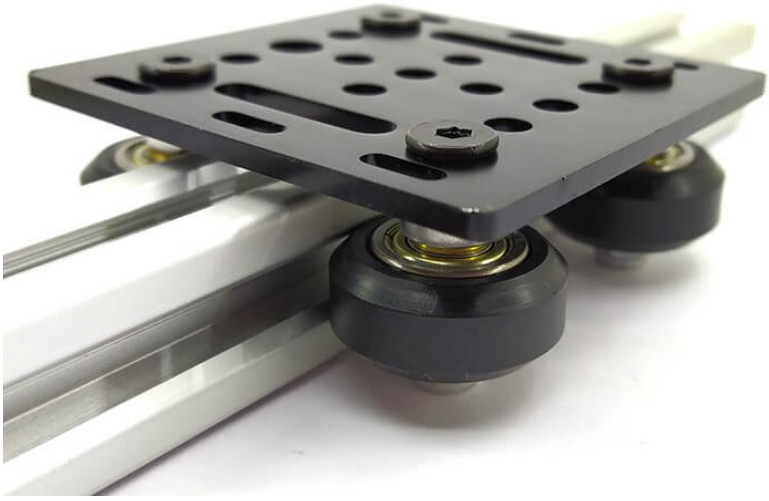

Figura 3.37 – V-Slot Gantry sobre eixo linear, adaptada de Fillment3D (2024).

Motor de passo D8-MOTOR80

Pela sua reduzida dimensão e características de funcionamento, o motor de passo modelo D8-MOTOR80 é frequentemente utilizado em aplicações de leitores de CD e DVD para permitir a introdução e remoção de discos. O motor possui um diâmetro de apenas 15 mm, é bipolar e alimentado com corrente contínua com tensão de alimentação máxima de 12 V. O atuador linear de parafuso possui um curso de 80 mm. Na Figura 3.38 pode-se observar o motor de passo D8-MOTOR80 utilizado no projeto.

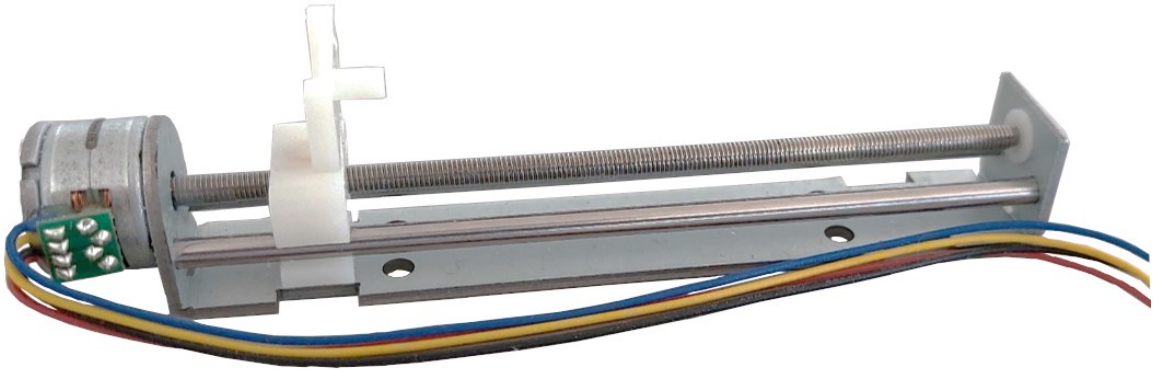

Figura 3.38 – Motor de passo D8-MOTOR80.

CNC Shield V3

O equipamento CNC Shield V3 permite o controlo, com elevada precisão, de equipamentos de impressão 3D, fresadoras e outros dispositivos de CNC que utilizem motores de passo. A placa foi projetada para permitir o encaixe direto sobre um Arduino, evitando a necessidade de ligações adicionais. A possibilidade da adição de quatro *drivers* A4988 ou DRV8825 permite o controlo de três eixos X, Y, Z e um quarto eixo, permitindo a duplicação de um dos anteriores ou a criação de um eixo customizado. A placa possui ligações preparadas para a implementação de sensores de fim de curso e de botões para o controlo do equipamento. Seguidamente encontram-se listadas algumas características adicionais relevantes do equipamento (Rajguru Electronics, 2024):

- Dimensões: 68 x 53 x 18 mm;
- Tensão de alimentação: 12 ~ 36 V DC;
- Compatível com GRBL 0,9;
- *Microstepping* até 1/16 para o *driver* A4988 e 1/32 para DRV8825;
- Proteção para sobreaquecimento e subtensão;
- Design compacto.

Na Figura 3.39 pode ser observado o CNC Shield V3 utilizado no projeto, instalado no Arduino e com dois *drivers* A4988 incorporados.

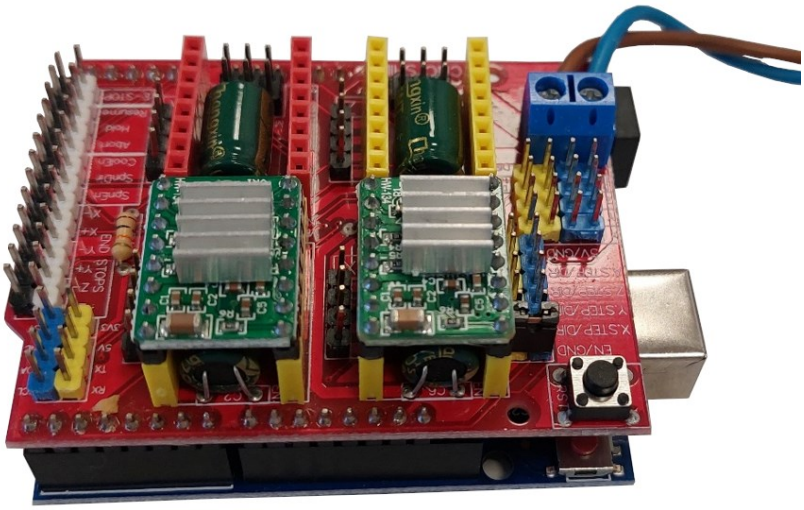

Figura 3.39 – CNC Shield V3 com *drivers* A4988.

Driver A4988

O *driver* A4988 foi projetado para o controlo de motores de passo bipolares. O *driver* possibilita a personalização do passo do motor, podendo o *Microstepping* corresponder a um passo inteiro até corresponder a um dezasseis avo de passo. O equipamento possui uma capacidade de saída, regulável, de 35 V e 2 A (Allegro MicroSystems, 2022). Na Figura 3.40 pode-se observar o *driver* A4988 utilizado no projeto para o controlo dos motores de passo.

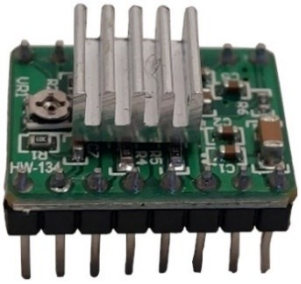

Figura 3.40 – *Driver* A4988.

Câmara

A câmara de modelo LifeCAM HD-3000 de fabrico da Microsoft permite a gravação de vídeo até à resolução de 720p com um campo de visão diagonal de 68,5 º. A interface de ligação da câmara é através de USB e permite um zoom digital até quatro vezes. A câmara permite tirar fotografias com uma resolução até um megapixel. Na Figura 3.41 pode ser observada a câmara utilizada no projeto.

Figura 3.41 – Câmara Microsoft LifeCam HD-3000.

Equipamentos auxiliares

Para além dos apresentados atrás, foram necessários vários equipamentos auxiliares que se descrevem seguidamente.

- Fontes de Alimentação e Transformadores – Responsáveis pela conversão da energia da rede, fornecida em corrente alternada, em corrente contínua para o correto funcionamento dos equipamentos, nomeadamente 12 V para a alimentação do CNC Shield V3 e os robôs Dobot Magician;
- Computador Portátil – Equipamento responsável pela interação entre o utilizador e os diversos equipamentos. O computador contém as instruções de comando para o automatismo do processo;
- Sensores de Fim de Curso (*Microswitches*) – O automatismo de movimentos em que não se usam sensores absolutos, como é o caso dos eixos lineares, que operam em anel aberto, exige que se detete uma coordenada de referência a partir da qual as posições seguintes são inferidas, por acumulação dos movimentos realizados. Para o presente projeto foram utilizados dois modelos de sensores de fim de curso para restringir a amplitude de movimentos dos motores de passo Nema 17 e D8-MOTOR80. Os modelos de *microswitches* utilizados foram o V-154-1C25 no motor de passo Nema 17 e o 125VAC 1A no motor de passo D8-MOTOR80. Os sensores utilizados podem ser observados na Figura 3.42 e Figura 3.43 respetivamente. As patilhas instaladas em cada modelo foram escolhidas tendo em conta a posição de instalação e o alcance necessário.

Figura 3.42 – Sensor de fim de curso de modelo V-154-1C25.

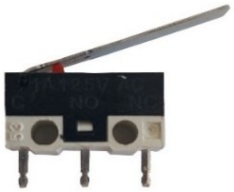

Figura 3.43 – Sensor de fim de curso de modelo 125VAC 1A.

Painel solar

Com base na proposta do projeto, de criar um modelo inspirado no processo de montagem de painéis solares, foram estudadas várias soluções para a manipulação de materiais pelos equipamentos disponíveis para a obtenção de um produto final representativo de um painel solar. A estratégia adotada foi a utilização de dois materiais distintos, representando diferentes elementos estruturais dos painéis. O primeiro elemento, representando o fundo protetor ou *backsheet*, foi construído com cartão kapaline, ou K-Line, de cor branca. O segundo elemento, representativo das células solares e do material EVA, foi construído com a utilização de folhas de acetato de poliéster, com coloração. Para a união dos dois materiais foi escolhida uma cola transparente.

A dimensão de ambos os componentes representativos do painel solar, é de 60 mm por 60 mm. Na Figura 3.44 ilustram-se os dois componentes utilizados para a montagem da representação do painel solar.

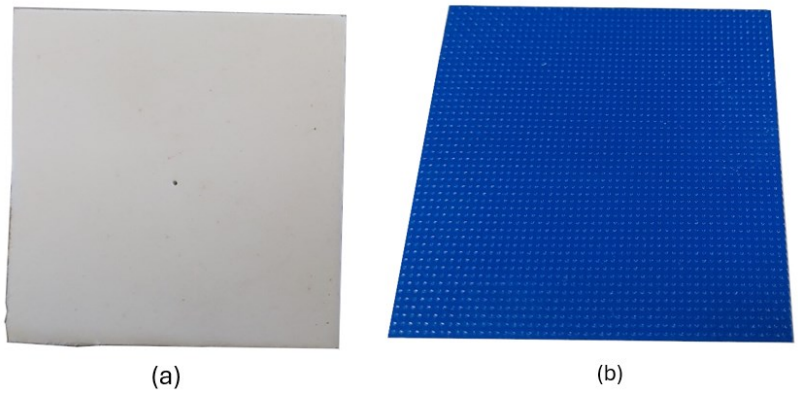

Figura 3.44 – Material kapaline (a) e acetato de poliéster (b) constituintes da representação do painel solar.

3.3 Desenho e fabricação de equipamentos

Durante a fase de planeamento e preparação do projeto foram identificadas várias necessidades de peças que não existiam no mercado ou que não correspondiam na totalidade às particularidades do projeto. A título de exemplo, não foi possível encontrar um adaptador do robô Dobot Magician para o I-Extruder, um dos equipamentos fundamentais ao projeto. Como solução para o problema optou-se por efetuar o projeto da peça, recorrendo ao programa de *Computer-Aided Design* (CAD) SolidWorks da Dassault Systèmes, e posteriormente fazer a impressão 3D da peça. Para a realização da impressão 3D foi utilizada uma impressora Anycubic Vyper. A impressora possibilita um tamanho de impressão de 245 x 245 x 260 mm.

De seguida encontram-se descritos os vários equipamentos projetados e fabricados.

3.3.1 Adaptador Dobot Magician e I-Extruder

A incorporação do I-Extruder como elemento terminal do robô Dobot Magician permite a capacidade de automatizar a sequência de movimentos de depósito de material no processo de montagem do painel solar. Devido ao I-Extruder representar um equipamento externo ao Dobot Magician, não existia um adaptador adequado para o encaixe do I-Extruder no robô. Como solução do problema foi adotada a estratégia de fabricar o adaptador com recurso a impressão 3D.

Na fase inicial de conceção do adaptador foram efetuadas medições do diâmetro do I-Extruder, bem como da furação existente no robô para a incorporação dos elementos terminais. Foi decidido o ponto ideal para o encaixe do adaptador no I-Extruder, por forma a garantir sempre a mesma posição de montagem, bem como o mínimo de movimento do equipamento durante o deslocamento do robô. Posteriormente foi efetuado o modelo CAD do protótipo do adaptador onde foram adicionados alguns pormenores como a implementação do formato da cabeça do parafuso para facilitar a operação de montagem e uma ranhura para a passagem dos botões existentes no I-Extruder.

O modelo do adaptador sofreu várias revisões até à obtenção do modelo final. As revisões constituíram essencialmente no ajuste de dimensões de encaixe do adaptador no Dobot Magician e no I-Extruder. Na Figura 3.45 é possível observar a revisão final do modelo CAD do adaptador.

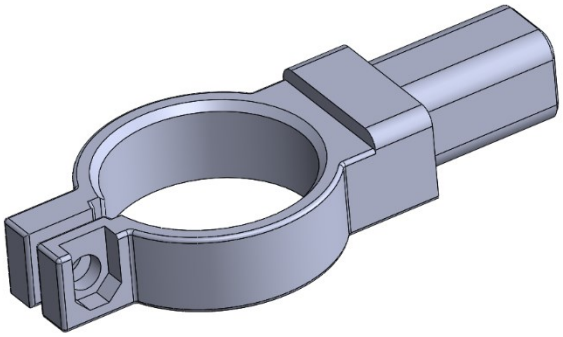

Figura 3.45 – Modelo CAD do adaptador Dobot Magician e I-Extruder.

Na Figura 3.46 é possível observar as várias iterações do adaptador impresso em 3D, sendo a versão final a peça à direita.

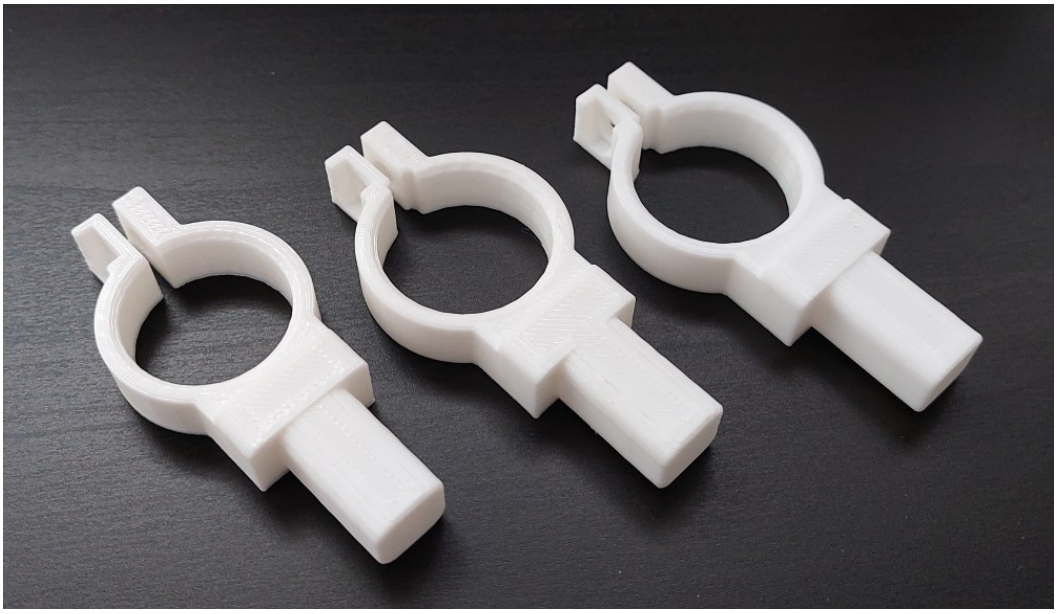

Figura 3.46 – Impressão 3D das iterações do adaptador Dobot Magician e I-Extruder.

3.3.2 Adaptador sensor de fim de curso e eixo linear

Para garantir que não são ultrapassados os limites de deslocamento da mesa de transporte, foram instalados *microswitches* com patilha nas extremidades do eixo linear. A implementação dos sensores revelou-se fundamental para garantir a deteção da mesa e desencadear a sua paragem previamente à colisão com o limite físico do mecanismo. A instalação dos sensores nas extremidades do eixo linear foi feita com recurso a um adaptador produzido de forma personalizada, que permite simultaneamente o encaixe com o eixo linear, tirando proveito do seu perfil V-Slot, popular em projetos de impressoras 3D e CNC. Posteriormente às medições o adaptador foi modelado recorrendo ao software SolidWorks.

Foram projetadas duas configurações do adaptador para permitir o seu encaixe em ambas as extremidades do eixo linear, onde as posições dos *microswitches* são

simétricas entre si. Na Figura 3.47 pode ser observado o modelo CAD elaborado do adaptador.

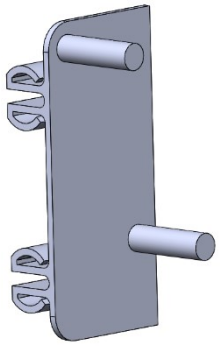

Figura 3.47 – Modelo CAD do adaptador sensor de fim de curso e eixo linear.

Na Figura 3.48 é possível observar o adaptador impresso em 3D instalado no eixo linear e com o sensor de fim de curso incorporado.

Figura 3.48 – Impressão 3D do adaptador sensor de fim de curso e eixo linear.

3.3.3 Mesa de transporte e expulsão

Durante o processo de produção do painel solar existe uma fase de transferência do painel entre a estação da montagem e o recetor de peças final, passando pelo controlo de qualidade. A movimentação do painel solar, neste processo de transferência, é efetuada com recurso ao V-Slot Gantry, montado no perfil V-Slot, cujo movimento é acionado através do motor de passo Nema 17. O V-Slot Gantry não possui uma superfície estável para o transporte do painel solar, nem um mecanismo de extração do painel. Para colmatar essas necessidades efetuou-se o projeto da mesa de transporte e expulsão, sendo esta o elemento mais complexo das peças produzidas por impressão 3D no âmbito do presente projeto. A essa mesa de transporte atribuíram-se dois objetivos: transporte do painel solar entre a fase de montagem e o recetor de peças final e a expulsão do painel solar para o respetivo recetor. Para efetuar o movimento de

expulsão da peça optou-se pela instalação do motor de passo D8-MOTOR80 na estrutura da mesa de transporte, bem como de sensores de fim de curso.

A mesa de transporte foi projetada em quatro peças. Seguidamente serão descritas as peças e a sua função.

- Peça superior (ponto 1 da Figura 3.49) – apresenta um topo liso para acomodar e transportar o painel solar após montagem e duas barras laterais de Policloreto de Vinilo (PVC) (fixadas posteriormente à impressão 3D) para guiar o painel durante o movimento. A peça possui um rasgo central para permitir a passagem da haste acoplada ao motor D8-MOTOR80. Adicionalmente a peça possui quatro pernas de elevação para permitir a instalação do motor de passo na parte inferior.
- Parte inferior (ponto 2 da Figura 3.49) – responsável por suportar o motor de passo D8-MOTOR80 bem como a instalação de dois sensores de fim de curso do modelo 125VAC 1A nas extremidades do eixo do motor. A peça foi projetada para efetuar um encaixe com o V-Slot Gantry através de ganchos e usar a furação existente, garantindo a estabilidade da mesa de transporte durante a deslocação. A peça superior é aparafusada à peça inferior.
- Peça intermédia de fixação (ponto 3 da Figura 3.49) – efetua a fixação do motor de passo D8-MOTOR80 à peça inferior, evitando a sua oscilação durante a operação.
- Barra de expulsão (ponto 4 da Figura 3.49) – responsável por conduzir a expulsão do painel solar da peça superior da mesa de transporte. A peça foi projetada para encaixar na haste acoplada ao motor de passo D8-MOTOR80 e com a extensão do painel solar.

Na Figura 3.49 pode ser observada uma vista explodida do modelo CAD projetado, com as diversas peças.

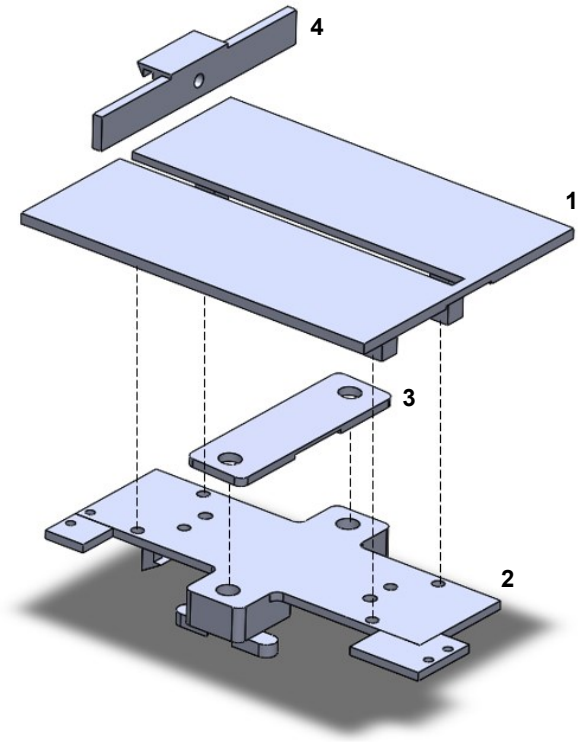

Figura 3.49 – Vista explodida das peças constituintes da mesa de transporte e expulsão.

Na Figura 3.50 pode ser observado o modelo CAD da mesa de transporte e expulsão conjuntamente com o motor de passo D8-MOTOR80 e os sensores de fim de curso 125VAC 1A instalados.

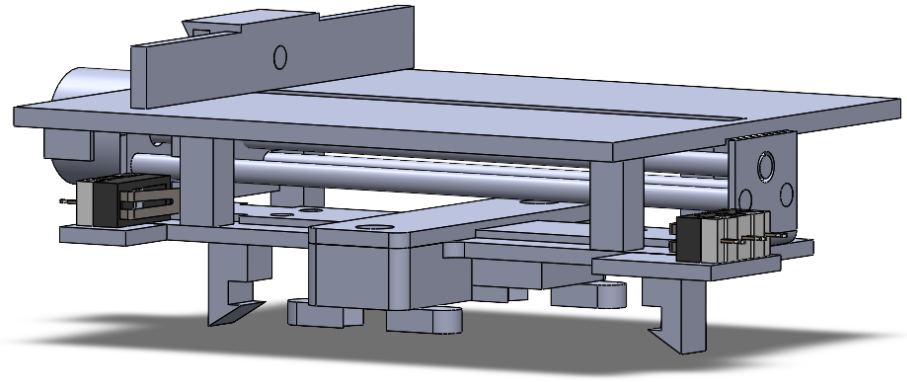

Figura 3.50 – Modelo CAD da mesa de transporte e expulsão com componentes instalados.

Na Figura 3.51 pode ser observada a montagem da mesa de transporte e expulsão impressa em 3D, com os equipamentos instalados e sobre o V-Slot Gantry.

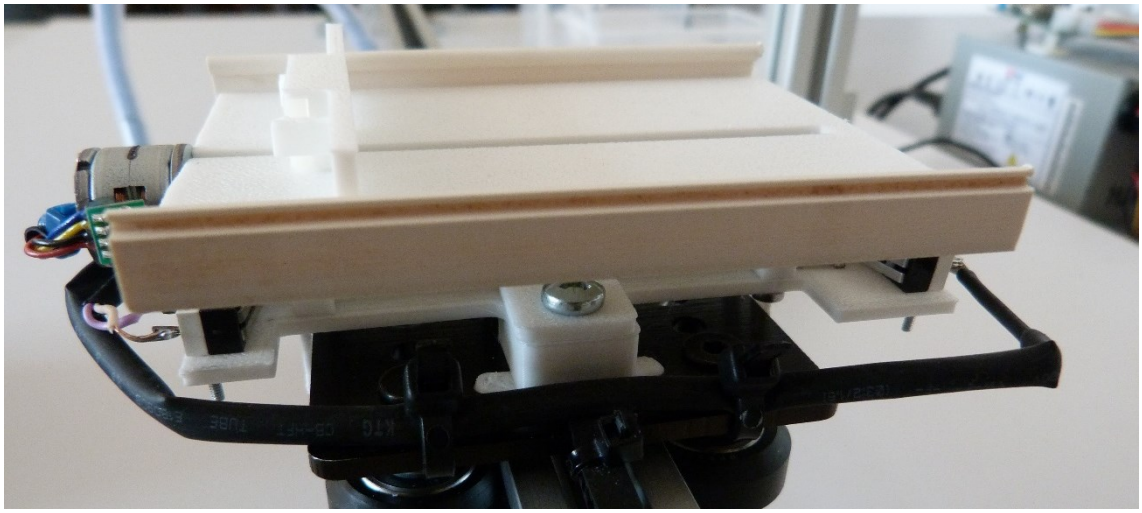

Figura 3.51 – Mesa de transporte e expulsão impressa e componentes instalados.

3.3.4 Mesa de montagem

Para garantir a melhor qualidade e precisão no processo de produção do painel solar surgiu a necessidade de criar uma estação para a colocação das peças e aplicação de cola. Tratando-se do ponto do processo onde ocorre a união dos componentes, foi fundamental garantir que a mesa de montagem permite um posicionamento constante das peças constituintes do painel solar, através de um encaixe à medida das peças.

A mesa de montagem possui uma zona de colocação das peças, com rampa para promover o seu alinhamento para uma cavidade preparada com as dimensões do painel solar. Adicionalmente possui furação e cavidade para a inserção de parafusos de fixação de tamanho M5. Para a redução de tempo de deslocação dos robôs na colocação das peças sobre a mesa de montagem, na aplicação da cola e no transporte da peça finalizada para a mesa de transporte, foram projetadas pernas de elevação da mesa. Na Figura 3.52 é possível observar o modelo CAD das duas partes da mesa de montagem que foram projetadas. Do lado esquerdo pode ser observada a parte superior da mesa de montagem e do lado direito as pernas de elevação.

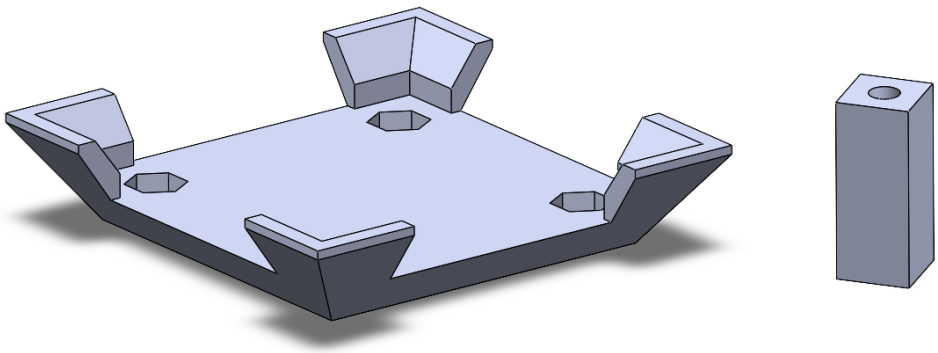

Figura 3.52 – Modelo CAD das partes constituintes da mesa de montagem.

A mesa de montagem foi projetada em duas partes para reduzir o tempo de impressão. A impressão 3D do modelo projetado para a mesa de montagem e a sua configuração final pode ser observada na Figura 3.53.

Figura 3.53 – Mesa de montagem impressa e instalada sobre base.

3.3.5 Alimentadores

Os elementos constituintes do painel solar, nomeadamente o kapaline representando o fundo protetor ou *backsheet* e as folhas de acetato de poliéster representando as células solares e o material EVA são transportados pelo Dobot 1 durante as diversas fases de produção. No primeiro passo, existe a necessidade de fornecer os elementos constituintes do painel solar. Para garantir a padronização do processo de produção, é fundamental garantir que a localização e o tamanho das peças são constantes. Para cumprir esses pressupostos foram projetados dois alimentadores, um para cada tipo de peça, recorrendo a um software de modelação CAD, para posteriormente se efetuar a impressão 3D.

Com o objetivo de reduzir o tempo de impressão e face à maior espessura do *backsheet* em comparação com as células solares, optou-se por implementar um perfil de PVC em "L" no alimentador do *backsheet*. O perfil permite garantir um posicionamento fixo do elemento.

Ambos os alimentadores foram projetados com cavidades com o formato das cabeças dos parafusos de fixação à mesa para facilitar o aperto e impedir a obstrução. Na Figura 3.54 podem ser observados os modelos CAD do alimentador do *backsheet* (lado esquerdo) e do alimentador das células solares (lado direito).

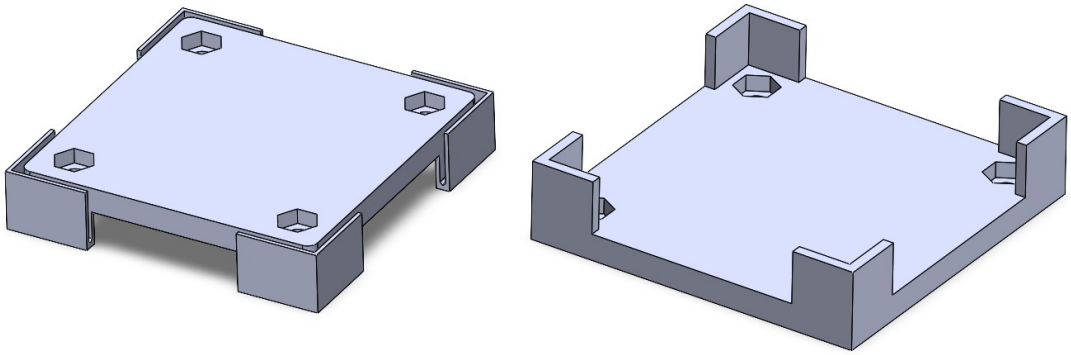

Figura 3.54 – Modelo CAD dos alimentadores.

Na Figura 3.55 é possível observar o alimentador do *backsheet*, com o perfil PVC em "L" instalado (lado esquerdo) e o alimentador das células solares (lado direito).

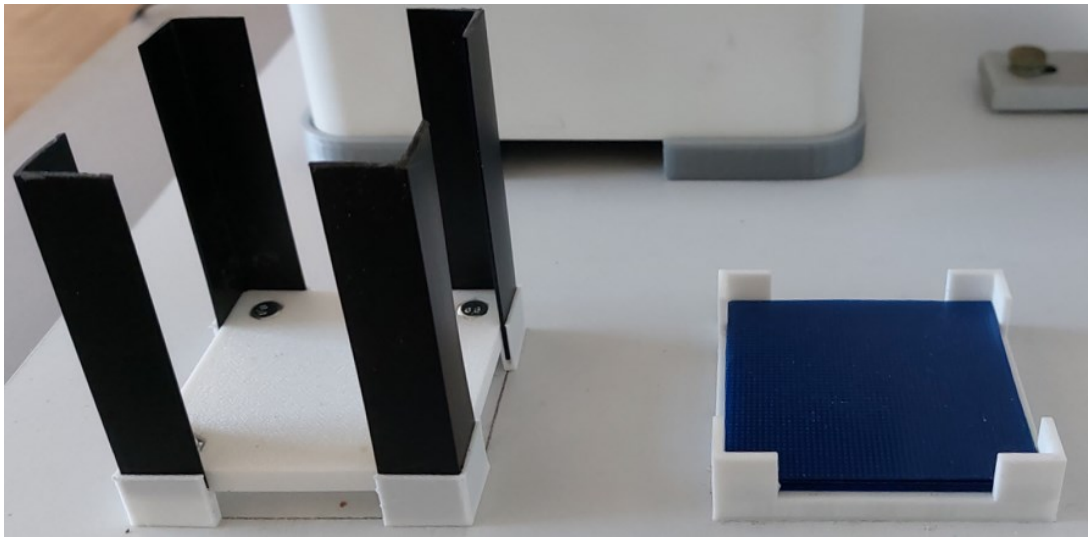

Figura 3.55 – Impressão 3D dos alimentadores.

3.4 Preparação e montagem de equipamentos

3.4.1 Planeamento da área de trabalho

Para a otimização dos equipamentos a selecionar e a gestão do espaço, foi fundamental a realização do planeamento e representação da área de trabalho do projeto. Os robôs Dobot Magician selecionados para integrar o projeto possuem uma área de trabalho limitada, dado que se encontraram fixos na mesa. O planeamento da área de trabalho permitiu precaver situações de possíveis colisões na execução de movimentos dos robôs e motores de passo. Adicionalmente, a incorporação de outros equipamentos móveis, como por exemplo o motor de passo, responsável pela deslocação da mesa de transporte, introduz complexidade adicional na gestão de espaço e interação dos diversos equipamentos.

Como estratégia para a planificação e representação da área de trabalho do projeto, recorreu-se ao *software* de CAD SolidWorks. O *software* permite a representação dos equipamentos e de diversos acessórios do projeto com elevado rigor e detalhe, melhorando a validade do modelo computacional face ao modelo real. A fase inicial do desenvolvimento do modelo CAD consistiu na incorporação dos modelos dos robôs Dobot Magician e do eixo linear, com a mesa de transporte incorporada, para validar a capacidade de deslocação dos elos dos robôs para as diversas estações de montagem do painel solar, e avaliar a dimensão da base que seria necessária para instalar todo o equipamento. Após validação do modelo inicial foram incorporados os restantes equipamentos, como por exemplo, os alimentadores, mesa de montagem e diversos outros equipamentos auxiliares.

Para a construção do modelo CAD recorreu-se a diversos modelos existentes e disponíveis em sítios da internet, nomeadamente dos componentes:

- Robô Dobot Magician (Gero, 2018);
- V-Slot Gantry (OpenBuilds, 2024);
- Motor de passo Nema 17, polia e placa de fixação (OpenBuilds, 2024);
- Motor de passo D8-MOTOR80 (Rodpan, 2022);
- Sensores de fim de curso do modelo V-154-1C25 (SolarTurtle, 2023);
- Sensores de fim de curso do modelo 125VAC 1A (Igor Lirtsman, 2021).

As restantes peças do modelo CAD foram elaboradas com base em medições efetuadas às peças disponíveis no Laboratório de Robótica e em peças projetadas e modeladas especificamente para o projeto. A modelação computacional revelou-se de grande utilidade na planificação e teste dos equipamentos modelados para impressão 3D. A visualização das peças na sua posição de instalação permitiu identificar possibilidades de melhorá-las.

Com a conclusão da modelação CAD, foram efetuadas simulações de movimentos das partes móveis, nomeadamente dos elementos terminais do Dobot Magician e dos motores de passo. Estas simulações permitiram verificar a existência de eventuais interferências ou colisões bem como a abrangência dos equipamentos. Na Figura 3.56 pode ser observado o modelo CAD do protótipo do processo.

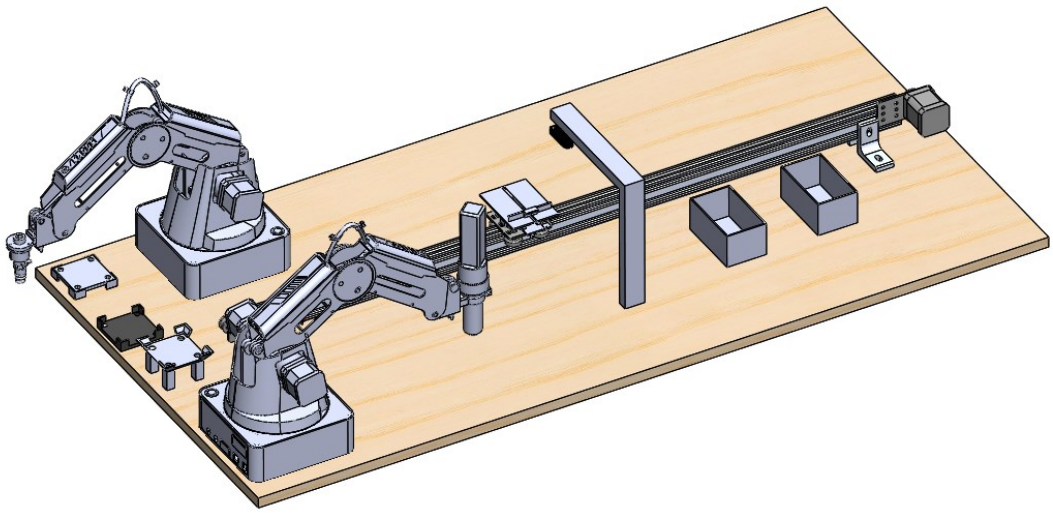

Figura 3.56 – Modelo CAD do protótipo do processo.

3.4.2 Preparação de equipamentos

Subsequentemente à planificação e representação, por recurso computacional, da área de trabalho, procedeu-se à preparação dos equipamentos eletrónicos e auxiliares integrantes do projeto. As configurações de programação efetuadas encontram-se descritas no capítulo 4.

Como base e ponto de fixação dos equipamentos do projeto foi selecionada uma placa de aglomerado revestido a melamina de dimensão 1200 mm por 600 mm, com 16 mm de espessura. A escolha deste material proveio da sua boa resistência, proteção contra choque, facilidade de trabalhar e baixo custo. Seguindo a planificação da área de trabalho realizada na secção anterior, efetuaram-se as marcações das posições de instalação dos diversos equipamentos sobre a base bem como os pontos guia para a furação. Na Figura 3.57 podem ser observadas as marcações efetuadas sobre a base.

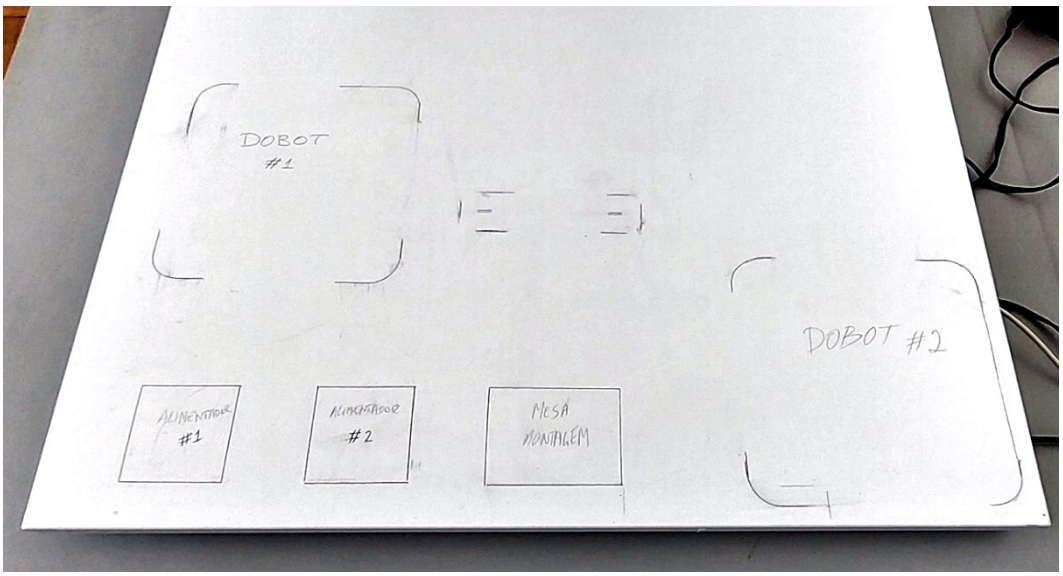

Figura 3.57 – Marcações efetuadas sobre base de aglomerado revestido a melamina.

O eixo linear, com perfil V-Slot 2040, permite incorporar diversos elementos como, por exemplo, um motor de passo e respetivos acessórios como a placa de montagem, correia e polia. O eixo linear, de 1 metro de comprimento, foi fixado à base com o apoio de suportes em forma de “L”. O eixo linear e suportes podem ser observados na Figura 3.58.

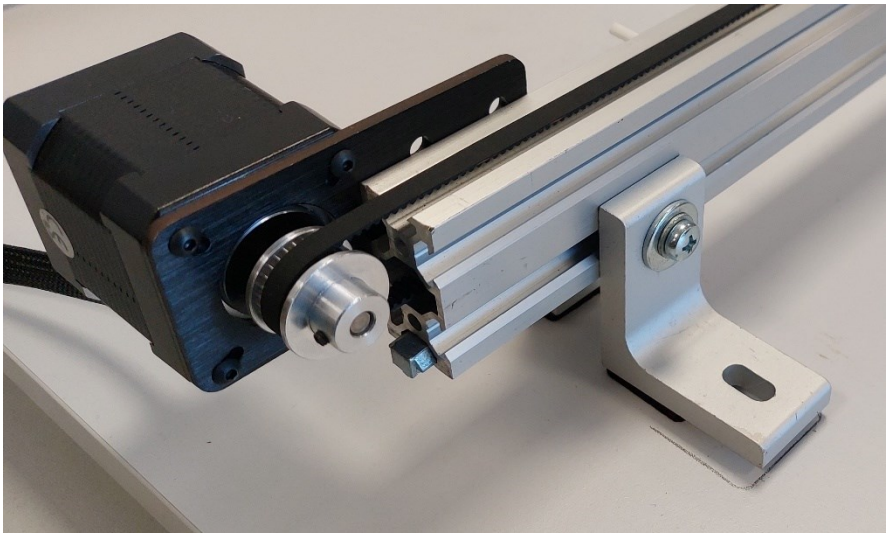

Figura 3.58 – Eixo linear V-Slot 2040 e suportes.

Para a fixação dos robôs Dobot Magician foram utilizados dois suportes projetados e produzidos no âmbito do projeto “*Precision on a mobile robot platform tool with dual arm”* realizado por Pereira et al. (2018). Os suportes possuem a vantagem de permitir o posicionamento rigoroso dos robôs sem impor a sua fixação à base, possibilitando a movimentação dos robôs, nomeadamente para o seu armazenamento, quando não se encontrem em utilização. A furação existente nos suportes foi utilizada para a marcação e posterior furação da base. Foram utilizados parafusos de tamanho M3 para a fixação do suporte. Um exemplo de um destes suportes pode ser observado na Figura 3.59.

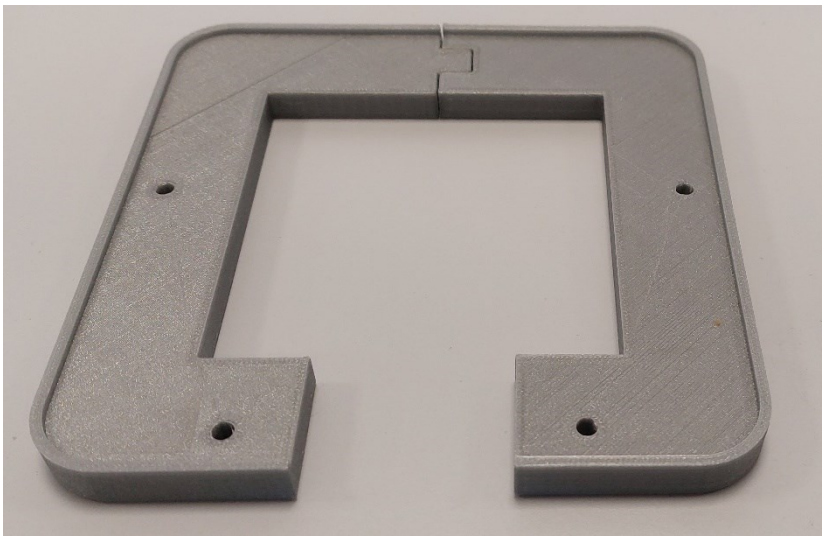

Figura 3.59 – Suporte de fixação do robô Dobot Magician.

A utilização de componentes eletrônicos, como as placas Arduino UNO, o CNC Shield V3, os sensores e o I-Extruder, introduziu a necessidade da conceção de ligações elétricas, nomeadamente de cablagens e terminais apropriados. Seguidamente encontram-se descritos os elementos produzidos.

Ligação dos sensores de fim de curso

A placa CNC Shield V3 permite a ligação de dois sensores de fim de curso por cada eixo, para além de alimentar e controlar os motores. Os terminais de ligação dos sensores fim de curso estão assinalados na Figura 3.60.

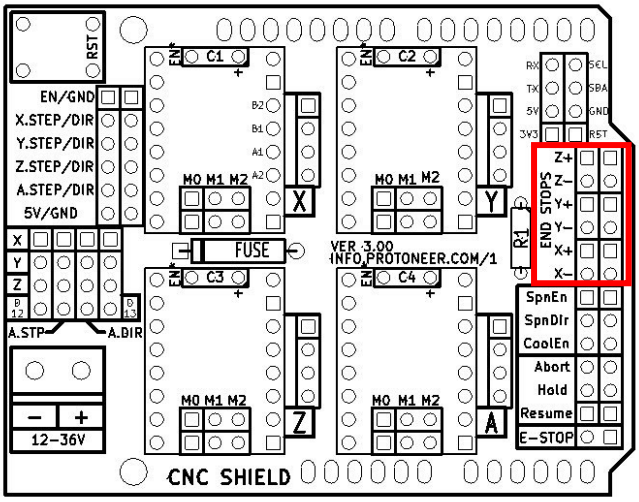

Figura 3.60 – Diagrama de ligações da placa CNC Shield V3, adaptada de Osoyoo (2017).

A ligação dos sensores de fim de curso, relativos ao eixo X (motor do eixo linear), foram preparados através da cravação da cablagem em terminais do tipo *Dupont*, para ligação à placa e em terminais *Faston* para ligação aos sensores.

Relativamente à ligação dos sensores de fim de curso do eixo Y (motor da mesa de transporte), os terminais de ligação à placa foram cravados com terminais do tipo *Dupont* e na extremidade oposta os fios foram soldados diretamente aos terminais do sensor. Na Figura 3.61 pode ser observada a cablagem e terminais produzidos para ligação dos sensores de fim de curso à placa CNC Shield V3.

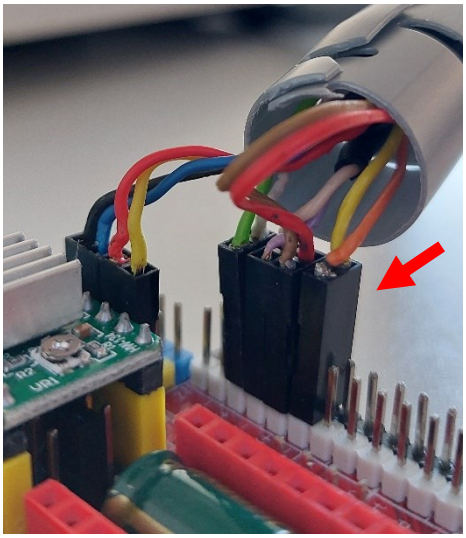

Figura 3.61 – Cablagem e terminais da cablagem dos sensores de fim de curso.

Ligação do I-Extruder ao robô Dobot Magician

O comando do I-Extruder pode ser efetuado manualmente no equipamento ou através do envio de um sinal elétrico através da entrada *trigger* existente no equipamento. Tratando-se de um sinal que pode ser de 3,3 V ou 5 V optou-se por ligar o I-Extruder a uma das portas de interface do robô Dobot Magician, denominadas por portas *Extended Input and Output* (EIO). Esta solução permite o envio de sinais elétricos ao I-Extruder através de comandos enviados pelo robô.

A entrada *trigger* do I-Extruder é efetuada através de uma ligação *jack* macho de 2,5 mm de três pólos. As características da ligação do *trigger* foram obtidas do manual de utilizador do I-Extruder, de onde foi extraído o excerto apresentado na Figura 3.62.

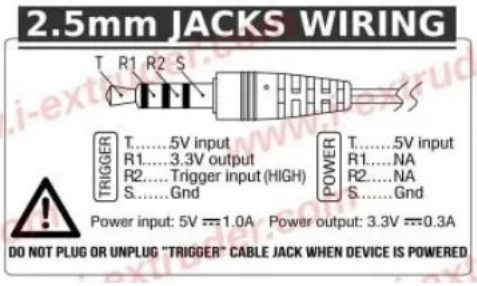

Figura 3.62 – Instruções de ligação do *trigger* do I-Extruder, adaptada de I-EXTRUDER (2024b).

Para transmitir o sinal *trigger* foi selecionada a porta de interface GP2 do Dobot Magician, mais especificamente a entrada EIO14, por permitir uma saída de 3,3 V / 20mA (*Dobot Magician User Guide*, 2020). Na Figura 3.63 podem ser observadas as várias interfaces de ligação com equipamentos externos, disponíveis no Dobot Magician, com destaque para a porta EIO14.

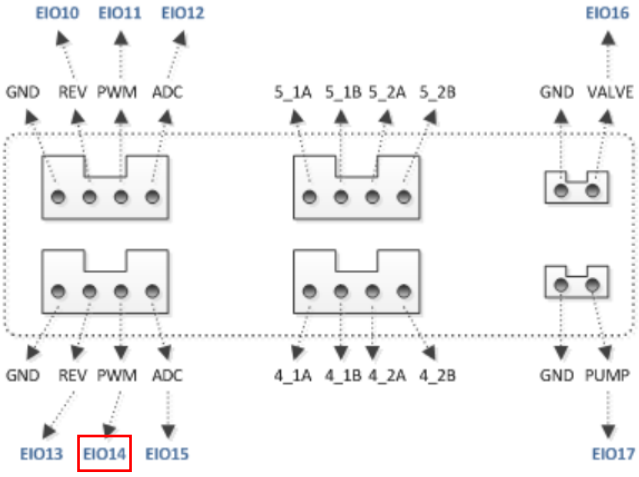

Figura 3.63 – Portas de interface disponíveis no Dobot Magician, com destaque para a EIO14, adaptada de Dobot Magician User Guide (2020).

O cabo de ligação do I-Extruder ao Dobot Magician foi adquirido com a ligação 2,5 mm já incorporada, tendo sido soldado ao cabo um terminal *Dupont* para ligação à entrada EIO14 e terra. Na Figura 3.64 pode ser observada a cablagem preparada para a ligação do I-Extruder ao Dobot Magician.

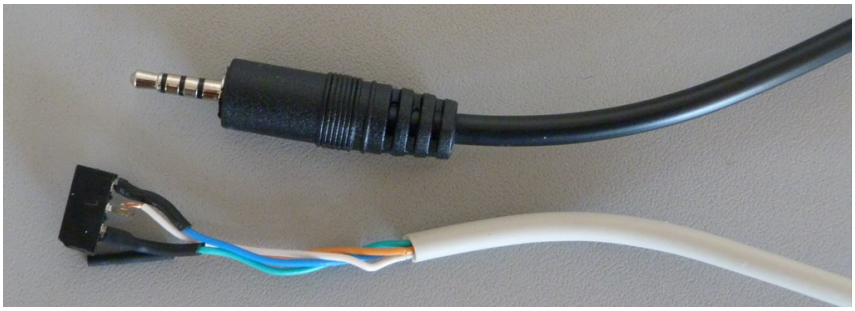

Figura 3.64 – Cablagem de ligação do I-Extruder ao Dobot Magician.

Ligação do motor de passo D8-MOTOR80 à placa CNC Shield V3

O motor de passo de modelo D8-MOTOR80 possui originalmente uma ligação com terminal do tipo XH2.54, contudo, este tipo de terminal não é passível de ligação aos pinos da placa CNC Shield V3. Além disso, o cabo de origem apresentava uma dimensão curta, sendo insuficiente para as amplitudes de deslocação previstas para o motor. Para a solução do problema recorreu-se à extensão da cablagem e cravação de terminais do tipo *Dupont* para permitir a ligação ao CNC Shield V3. Numa primeira iteração da cablagem de ligação do motor de passo D8-MOTOR80 e dos sensores de fim de curso do eixo Y, os cabos foram colocados numa única manga. Posteriormente, na fase de testes operacionais, foram detetadas indicações erráticas dos sensores de fim de curso. Com o apoio de um voltímetro foi possível concluir que o acionamento do motor de passo influenciava as indicações dos sensores. A resolução do problema passou por individualizar e isolar cada cabo de equipamento numa manga termoretractil.

Na Figura 3.65 pode ser observada a cablagem preparada para a ligação do motor de passo D8-MOTOR80 à placa CNC Shield V3.

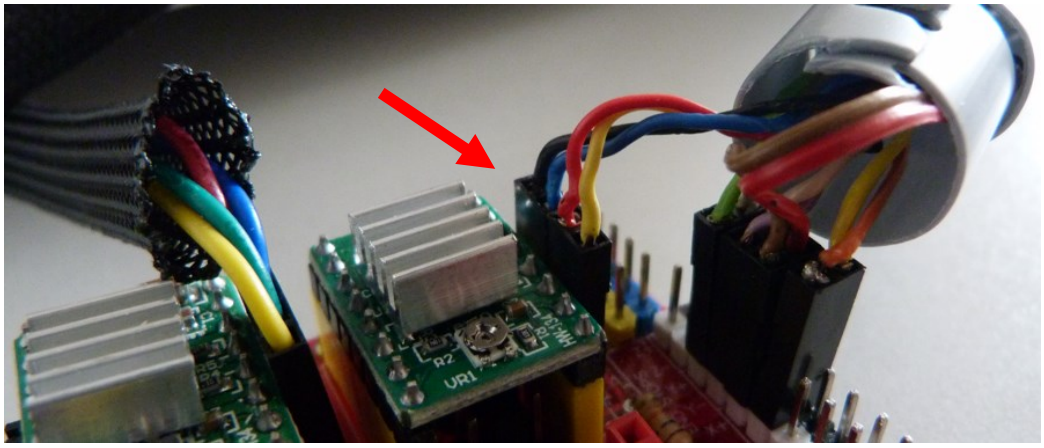

Figura 3.65 – Cablagem de ligação do motor de passo D8-MOTOR80 à placa CNC Shield V3.

Ligações e configuração do Arduino UNO, CNC Shield V3 e drivers A4988

A solução construtiva da placa CNC Shield V3 permite o encaixe direto sobre o Arduino UNO, evitando a necessidade de cablagem adicional. A alimentação elétrica da placa CNC Shield V3 é providenciada através do Arduino UNO, contudo, a alimentação elétrica requerida pelos motores de passo deve ser garantida por uma ligação externa de 12 V a 36 V. A alimentação elétrica foi providenciada por uma fonte de alimentação do tipo *Advanced Technology Extended* (ATX).

De forma semelhante ao CNC Shield V3 e Arduino UNO, os *drivers* A4988 podem ser instalados diretamente no CNC Shield V3, através de terminais próprios existentes na placa. Foram instalados na placa dois *drivers* A4988 para o controlo dos dois motores de passo existentes no projeto.

Os *drivers* A4988 permitem alguma personalização de parâmetros para ajustar o modo de funcionamento dos motores de passo. Os dois parâmetros configurados nos *drivers* foram o *Microstepping* e a tensão de referência.

O *Microstepping* constitui uma forma de diminuir o tamanho de cada passo do motor de passo, permitindo uma melhoria na precisão do movimento e redução das vibrações, nomeadamente a baixas velocidades. Os *drivers* A4988 permitem a configuração de *Microstepping* de um passo completo até 1/16 do passo. A implementação do *Microstepping* nos *drivers* é efetuada através da instalação de *jumpers* numa determinada configuração conforme o valor de *Microstepping* desejado. Na Figura 3.66 pode ser observado o local de instalação dos jumpers na placa CNC Shield V3.

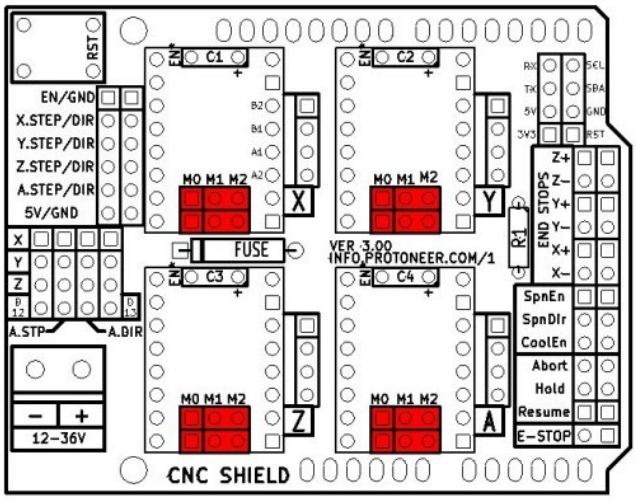

Figura 3.66 – Local de instalação de *jumpers* para configuração de *Microstepping* na placa CNC Shield V3, adaptada de Kruger (2013).

No Quadro 3.4 pode ser consultada a configuração de *jumpers* requerida para cada valor de *Microstepping*, em que "High" representa a instalação de *jumper* e "Low" significa sem instalação de *jumper*.

Quadro 3.4 – Configuração de *Microstepping* para o *driver* A4988, adaptada de Kruger (2013).

MS0	MS1	MS2	Valor de *Microstepping*
Low	Low	Low	Passo completo
High	Low	Low	Meio passo (1/2)
High	High	Low	Um quarto de passo (1/4)
Low	High	Low	Um oitavo de passo (1/8)
High	High	High	Um dezasseis avos de passo (1/16)

Após vários ensaios experimentais, com ambos os motores de passo utilizados no projeto, selecionaram-se os valores de *Microstepping* de 1/16 para o motor Nema 17 e 1/4 para o motor D8-MOTOR80.

O segundo parâmetro configurado foi a limitação da tensão de referência, correspondente à tensão de alimentação dos motores de passo. A tensão é regulada através de um potenciómetro localizado no *driver* A4988. A localização do potenciómetro pode ser observada na Figura 3.67.

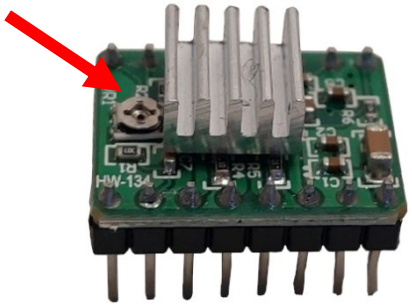

Figura 3.67 – Localização do potenciómetro no driver A4988.

A fórmula de cálculo da tensão de referência é dada pela equação (1) (Allegro MicroSystems, 2022).

$$V_{REF} = I_{MAX} \times 8 \times R_s \quad (1)$$

onde, V_{REF} corresponde à tensão de referência, I_{MAX} à corrente máxima do motor de passo e R_s a resistência instalada no *driver*.

Em ambos os *driver*s A4988 o valor da resistência é de 0,1 Ω. Para o motor de passo Nema 17 a corrente máxima é de 0,9 A e para o motor de passo D8-MOTOR80 é de 0,8 A.

Com a determinação dos parâmetros necessários, procedeu-se ao cálculo do valor de tensão de referência, tendo sido obtido o valor de V_{REF} de 0,72 V para o motor Nema 17 e V_{REF} de 0,64 V para o motor D8-MOTOR80. A regulação da tensão de referência nos *drivers* A4988 foi efetuado com o apoio de um voltímetro.

Instalação dos elementos terminais nos robôs Dobot Magician

Após a impressão 3D do adaptador do I-Extruder para o Dobot Magician procedeu-se à sua instalação no Dobot 2. Adicionalmente foi instalado o elemento terminal de ventosa no Dobot 1. Na Figura 3.68 podem ser observados os Dobot Magician com os respetivos elementos terminais instalados.

(a) (b)

Figura 3.68 – Elemento terminal ventosa (a) e I-Extruder (b) instalados nos robôs Dobot Magician.

3.4.3 Montagem de equipamentos e área de trabalho

Com o planeamento da área de trabalho e a preparação dos diversos equipamentos já efetuados, procedeu-se à montagem e instalação dos elementos do projeto no

laboratório de robótica do DEM. A base do protótipo foi colocada sobre mesas unidas, disponíveis no laboratório, tendo-se colocado vários blocos de acrílico por baixo para proporcionar uma ligeira elevação da base e permitir a passagem e ocultação de cablagem.

Com as marcações efetuadas sobre a base procedeu-se à furação da mesma e fixação dos diversos equipamentos, nomeadamente as bases dos robôs Dobot Magician, o eixo linear, os alimentadores, a mesa de montagem entre outros.

Os cabos elétricos representam um papel fundamental de alimentação elétrica dos equipamentos e transmissão de informação, contudo, quando desorganizados, podem constituir obstáculos ao bom funcionamento do processo, dificultar a manutenção ou alteração do processo, para além de constituírem uma imperfeição no aspeto visual do projeto. Com o objetivo de organizar e ocultar a cablagem procurou-se, sempre que possível, passar a cablagem pela zona inferior da base. Nas situações onde não era possível foram utilizados alguns acessórios de organização de cabos para melhorar o aspeto visual e promover uma área de trabalho livre. Na zona inferior da base as cablagens foram organizadas em função do seu destino e afixadas com abraçadeiras e fitas de velcro.

Devido à entrada de várias ligações na placa CNC Shield V3, optou-se por deixar a mesma fora da base e diretamente sobre as mesas de apoio, permitindo um percurso mais curto das cablagens até à zona inferior da base. Por outro lado, optou-se por manter a fonte de alimentação dos motores de passo sobre as mesas para permitir um acesso rápido ao interruptor de corte na eventualidade de ocorrer algum incidente ou movimento inesperado dos motores. Na Figura 3.69 pode ser observada a disposição dos equipamentos após a finalização da preparação da área de trabalho.

Figura 3.69 – Área de trabalho após montagem e preparação dos equipamentos.

3.4.4 Calibração dos robôs Dobot Magician

A calibração dos robôs é um fator crucial para o correto funcionamento do processo automático. A execução dos diversos movimentos do robô encontra-se assente na indicação de valores de coordenadas para onde se pretende a sua deslocação, pelo que se torna fundamental garantir que as coordenadas indicadas no software de controlo do robô correspondam à posição física correta do robô, de acordo com os padrões de fabrico.

O Dobot Magician possui dois tipos de sistemas de coordenadas, um sistema que utiliza as coordenadas das juntas enquanto outra utiliza as coordenadas cartesianas. No sistema de coordenadas das juntas são utilizadas as coordenadas J1, J2, J3 e J4. No sistema de coordenadas cartesianas são utilizadas as coordenadas X, Y, Z e R, sendo que o ponto da origem (0, 0, 0) se encontra no centro dos três motores do robô (note-se que o motor que aciona a Junta 3 não tem o eixo coincidente com o ponto de origem – ver figura 3.71). Na Figura 3.70 e Figura 3.71 é possível observar a localização dos eixos de cada um dos sistemas de coordenadas.

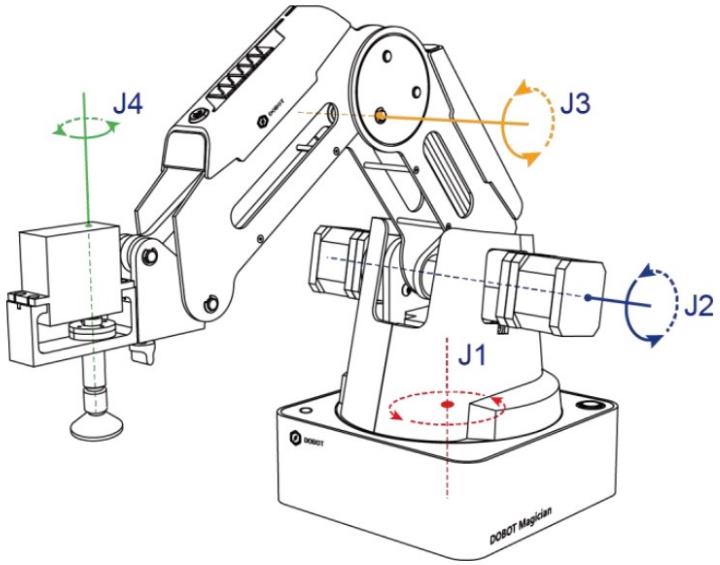

Figura 3.70 – Sistema de Coordenadas das Juntas no Dobot Magician, adaptada de: Dobot Magician User Guide (2020).

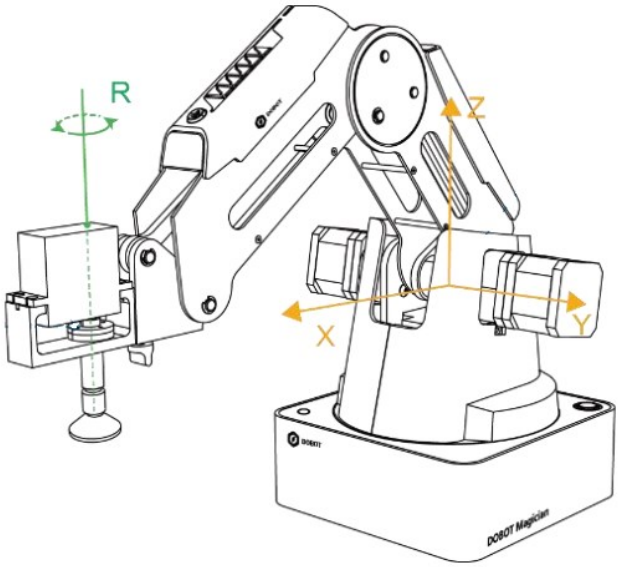

Figura 3.71 – Sistema de Coordenadas Cartesianas no Dobot Magician, adaptada de Dobot Magician User Guide (2020).

Existem três parâmetros passíveis de calibração no Dobot Magician, a base, os sensores e o *home*.

A calibração da base permite corrigir os valores do Encoder da base. Segundo o Manual de Utilizador do Dobot Magician, após a execução da instrução *home* a coordenada J1 deverá estar localizada nos 0º, com erro permissível entre -3º e 1º. Caso o valor não se encontre dentro desse intervalo, o robô requer calibração da base.

A calibração dos sensores dos ângulos dos elos dianteiro e traseiro permite a correta movimentação dos elos do robô. Segundo o Manual de Utilizador do Dobot Magician, a coordenada Z deve permanecer a mesma ao movimentar o Dobot Magician no mesmo plano horizontal. Se este comportamento não se verificar, será necessário calibrar os sensores dos ângulos dos elos. Os sensores poderão ser calibrados de forma manual ou automática.

O *home* corresponde às coordenadas de referência para o robô, sendo utilizadas como o ponto de partida para as restantes movimentações. No início de qualquer operação com o robô, deve ser executada a ação *home* por forma ao robô deslocar-se para a posição de referência. A execução do *home* tem o objetivo de repor as coordenadas de referência na eventualidade de o robô sofrer uma colisão ou se ocorrer a perda de passos por parte de um dos motores.

Na fase inicial de desenvolvimento deste projecto foi verificado um comportamento incorreto pelos robôs quando era executada a instrução de *Home,* nomeadamente na coordenada J1, onde não eram obtidos os valores nos parâmetros indicados pelo fabricante. Por esta razão, segundo o manual Dobot Magician User Guide (2020), procedeu-se à calibração do Encoder da base com recurso ao *Writing and drawing kit* bem como dos sensores através de uma calibração automática com recurso ao

adaptador *auto-levelling* fornecido com os robôs. Para garantir um correto ponto de referência para as movimentações dos robôs foi também calibrada a coordenada *home*. Todas as ações de calibração dos robôs foram executadas seguindo o Manual de Utilizador do Dobot Magician.

4 Programação do sistema de controlo e supervisão

4.1 Controlo e programação dos equipamentos

As secções anteriores tornaram patente que o processo automático aqui desenvolvido se apoia num conjunto de dispositivos mecatrónicos que possuem ou estão associados a um controlador próprio. Recordando, os robôs Dobot possuem um controlador interno, o I-Extruder possui também eletrónica interna de controlo e os motores de passo são comandados por um controlador Arduino UNO. Desta forma, uma boa parte das operações de controlo dos equipamentos está a cargo desses controladores. A par desses controladores foi introduzido um computador cuja função é integrar, sincronizar e sequenciar as operações dos diversos equipamentos. Este computador, daqui em diante designado *controlador central* estabelece a ordem das operações necessária para cumprir os requisitos do processo, para além de proporcionar a interface com o utilizador. A comunicação entre o controlador central e os controladores periféricos está assente num conjunto de instruções preparadas para desencadear as tarefas necessárias. Cada instrução consiste numa mensagem que é interpretada pelo elemento recetor com base em protocolos de comunicação que foram estabelecidos neste trabalho.

Tendo em consideração a popularidade atual da linguagem de programação Python e a sua introdução em unidades curriculares de diversos cursos de engenharia (inclusivamente na LEM do ISEL), esta foi escolhida como a linguagem de desenvolvimento do programa do controlador central. Contudo, a programação em Python representa uma programação de alto nível, servindo aqui essencialmente como organizador de mensagens e envio das mesmas. Estas mensagens são descodificadas por parte dos controladores periféricos, tendo estes mecanismos próprios para o acionamento dos motores do respetivo equipamento.

Para a programação do Arduino UNO, usou-se a linguagem C++. O programa criado foi gravado na memória flash do Arduino UNO e corre sempre que o equipamento se

encontra alimentado. A arquitetura de controlo do processo encontra-se representada na Figura 4.72.

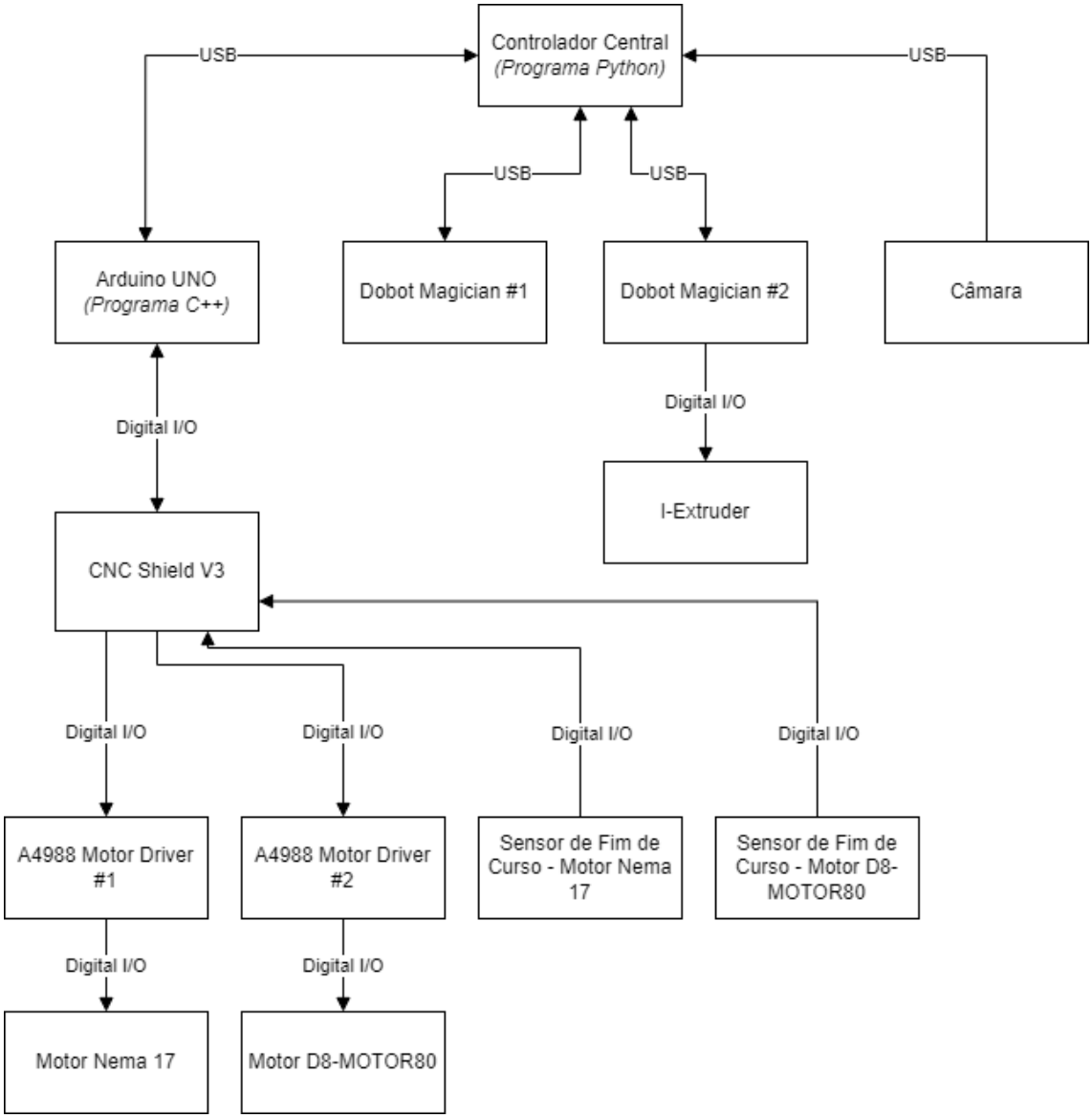

Figura 4.72 – Arquitetura de controlo do processo.

A escrita de programas pode ser elaborada de diversas formas, desde a escrita num simples bloco de notas até à utilização de aplicações de ambiente de desenvolvimento integrado (IDE) que permite a escrita de uma forma mais eficiente através da deteção automática de erros e de automatismos na estruturação do código.

Para a programação do Arduino UNO foi utilizado o *software* Arduino IDE Versão 2.3.2. O software permite a escrita do código, a sua verificação e o seu envio para o Arduino UNO. Além disso, disponibiliza uma interface de comunicação série com o hardware.

Para a escrita de programas em Python existem diversos IDE, variando, entre outros aspetos, na sua complexidade, nos automatismos de deteção de erros e no apoio ao utilizador. Para a realização deste projeto foi escolhido o IDE PyCharm Community Edition Versão 2024.1.2 da JetBrains, dado que, para além de ser uma aplicação de

utilização livre, oferece um bom equilíbrio entre as funcionalidades de automatismo da escrita de código e a flexibilidade dada ao utilizador para a organização de um projeto e estruturação do código.

A programação em Python popularizou-se muito graças à possibilidade de utilização de módulos. No âmbito do presente projeto, os módulos utilizados foram tipificados em duas categorias, módulos externos e módulos internos. Os módulos externos são elaborados por outros utilizadores e podem ser importados para o programa em desenvolvimento, enquanto os módulos internos são conjuntos de instruções que fazem parte integrante do presente projeto. O uso de módulos permite a escrita de blocos de código independentes, mas que podem ser interligados, auxiliando a organização de projetos de grandes dimensões através de uma divisão lógica das partes constituintes.

No Quadro 4.5 encontram-se listados os módulos externos utilizados na programação Python deste projeto.

Quadro 4.5 – Módulos externos Python.

Módulo	Autor	Descrição
TkInter	Python	Integrante da biblioteca de Python. Módulo utilizado para o desenvolvimento de *Graphical User Interface* (GUI).
webbrowser	Python	Integrante da biblioteca de Python Módulo para executar e controlar *browsers* de internet.
time	Python	Integrante da biblioteca de Python. Módulo fornece várias funcionalidades relacionadas com o tempo.
datetime	Python	Integrante da biblioteca de Python. Módulo fornece várias funcionalidades relacionadas com datas e horas.
csv	Python	Integrante da biblioteca de Python. Módulo permite a manipulação de ficheiros no formato *Comma Separated Values* (CSV).
os	Python	Integrante da biblioteca de Python. Módulo que permite a execução de tarefas diversas relacionadas com o sistema operativo.
DobotDllType	Dobot	Módulo de interface de comunicação com o robô Dobot Magician. Página oficial: https://www.dobot-robots.com/
pyserial	Chris Liechti	Módulo permite a comunicação através das interfaces série do controlador central. Página Oficial: https://pyserial.readthedocs.io/en/latest/index.html
cv2	OpenCV Team	Módulo que permite a ligação do Python com a biblioteca de visão computacional e aprendizagem automática OpenCV. Página oficial: https://opencv.org/
numpy	NumPy	Módulo para criação e manipulação, com funções matemáticas, de matrizes multidimensionais. Página oficial: https://numpy.org/

Relativamente aos módulos internos, estes foram organizados de acordo com o âmbito de aplicação das funções que implementam, procurando-se separar as funções de comunicação com os equipamentos das de interface com o utilizador.

No total foram criados oito módulos, podendo ser divididos em dois grandes grupos. O Grupo I é composto pelos módulos de interface gráfica para a interação do utilizador

com o programa. O Grupo II é constituído pelos módulos de comunicação e de operação dos controladores periféricos, bem como por módulos de execução de tarefas secundárias.

No Quadro 4.6 encontram-se listados os módulos internos que foram criados e o âmbito de aplicação de cada um.

Quadro 4.6 – Módulos internos Python.

Grupo	Módulo	Âmbito de aplicação
I	*main*	Módulo inicial do programa, composto pela interface gráfica de interação com o utilizador para configuração da produção em modo automático.
	manualcontrol	Interface gráfica de interação com o utilizador para controlo manual das tarefas dos diversos equipamentos do projeto.
II	*dobotcommunication*	Instruções de ligação e manipulação dos robôs Dobot Magician para execução das tarefas do projeto.
	motorcommunication	Instruções de ligação e manipulação com os motores de passo para execução das tarefas do projeto.
	automaticproduction	Sequência de instruções para a execução do processo em modo automático.
	cameravision	Instruções de captura e análise de imagem através de câmara para análise de qualidade do produto.
	detecterror	Deteção de erros de comunicação com equipamentos e omissões de preenchimento de campos nas interfaces gráficas.
	productionhistory	Elaboração de um ficheiro de registo histórico das peças produzidas pelo processo.

As funções constituintes de cada módulo interno e as respetivas dependências encontram-se representadas na Figura 4.73.

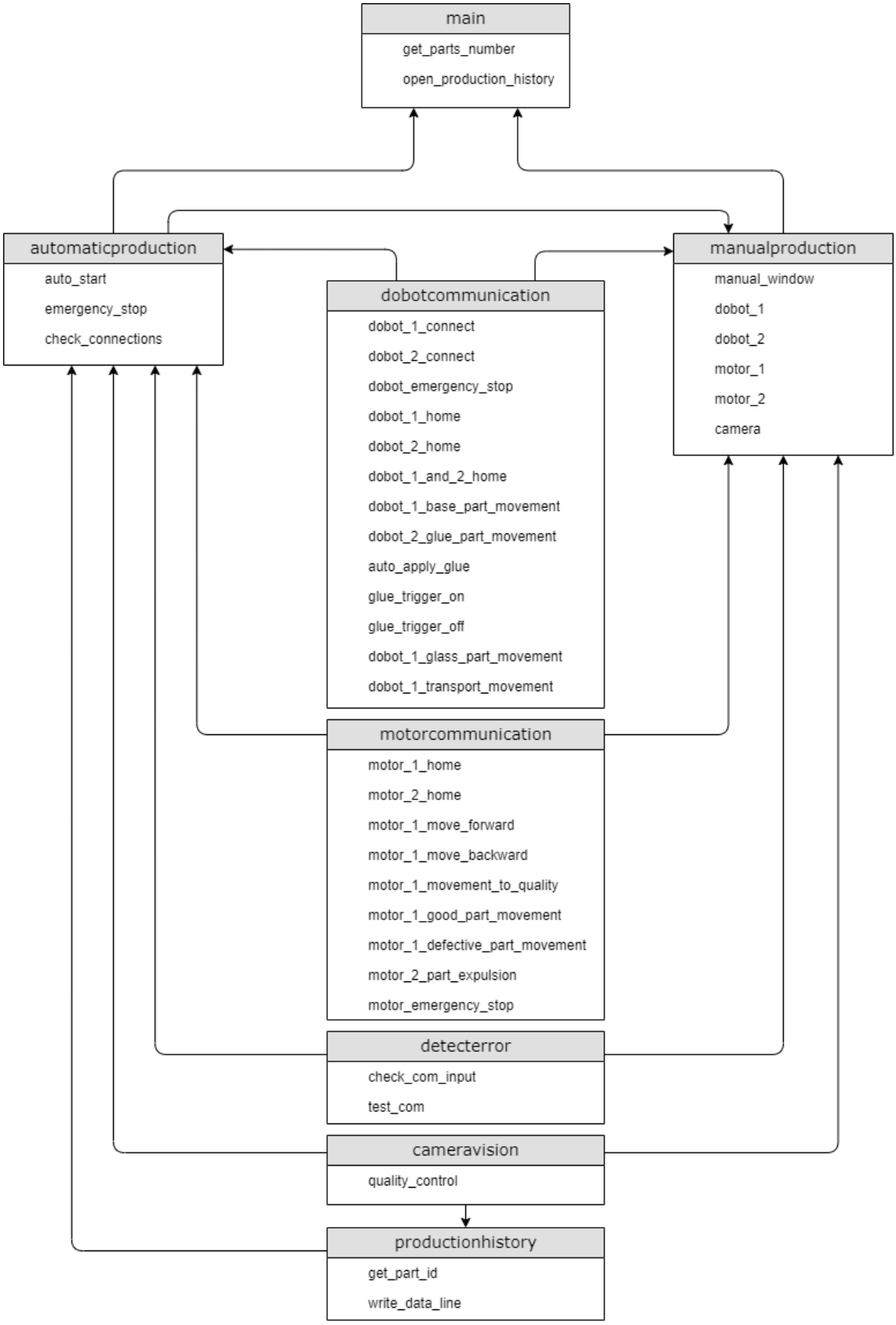

Figura 4.73 – Funções e interligação dos módulos internos.

É de referir que foi criada, na aplicação, a possibilidade de comandar cada um dos equipamentos individualmente, no chamado *modo manual*, útil para a realização de testes e operação dos equipamentos em caso de falhas. Em contraponto, a aplicação

oferece o modo automático, em que o processo de produção é executado autonomamente.

Outro aspeto a salientar no desenvolvimento desta aplicação são os mecanismos de sincronismo entre equipamentos. Este é um aspeto importante dado que existem 4 controladores que operam em paralelo: o controlador do Dobot 1, o controlador do Dobot 2, o Arduino UNO e o controlador central, sendo este responsável por coordenar os restantes. Para garantir a sincronização das operações do processo e evitar a execução da tarefa seguinte sem que a anterior esteja concluída, foram implementadas instruções de espera no controlador central. A determinação de conclusão de uma etapa do processo variou em função do equipamento que a realiza: no Dobot Magician é realizada uma pausa até à conclusão da última instrução enviada; na atuação dos motores de passo, caso esteja prevista a ativação de um sensor de fim de curso, é aguardada uma mensagem vinda do controlador periférico; nos restantes casos é efetuada uma pausa de um tempo pré-estabelecido pelo autor.

Nas secções seguintes é abordado mais aprofundadamente o código desenvolvido no presente projeto, dividindo-o por temas, que resultam em quatro secções dedicadas respetivamente a i) Comando dos Dobot Magician, ii) programação do Arduino UNO, iii) teste de controlo de qualidade e iv) interface gráfica. Cada uma destas debruça-se sobre um ou vários módulos internos; mais concretamente teremos a correspondência: i) *dobotcommunication*, ii) *motorcommunication*, iii) *cameravision* e iv) *main*, *manualcontrol*, *automaticproduction*, *productionhistory*, *detecterror*. Em cada uma das secções serão descritos os conceitos fundamentais que estão na base da programação daqueles módulos.

4.1.1 Programação dos robôs Dobot Magician

A programação da interação com os robôs Dobot Magician foi elaborada no módulo interno *dobotcommunication*. Este módulo é constituído por um conjunto de funções dedicadas à abertura/fecho de comunicação e movimentação dos robôs. A divisão dos movimentos em funções permitiu que, no modo manual, seja possível o acionamento de movimentos específicos do processo. A execução do processo em modo automático foi constituída pela sequência de funções de acordo com a ordem de produção.

O principal módulo externo importado na programação dos robôs Dobot Magician foi o “DobotDllType”, um módulo criado pelo fabricante dos robôs e onde estão definidas diversas funções para a interface com os robôs. O módulo permite duas formas de envio de instruções para os robôs: através de uma fila de instruções ou individualmente, instrução a instrução. Devido à estruturação do programa em várias funções, dois modos de funcionamento (manual e automático) e ao facto de, nalgumas situações, o acionamento de uma instrução ser pendente de uma resposta, foi estabelecida a forma

de envio de instrução a instrução, através da configuração do parâmetro “isQueued” com o valor nulo. Foi igualmente utilizado o módulo externo “time” para a introdução de alguns períodos de pausa no processo.

Uma dificuldade identificada durante a fase de desenvolvimento do programa de controlo consistiu na comunicação simultânea com os dois robôs Dobot Magician. Embora estivesse definida uma porta série de comunicação diferente para cada robô, o módulo DobotDllType apenas não permite a criação de dois objetos de comunicação com os robôs. A solução desenvolvida para ultrapassar esta limitação consistiu em abrir e fechar a comunicação sempre que as instruções eram enviadas para um robô, possibilitando a comunicação com o outro de seguida, pelo mesmo procedimento. A frequência de ligações não acarretou problemas ao funcionamento do processo.

Foram criadas duas variáveis globais que são utilizadas nas diversas funções, nomeadamente a variável “api” que armazena em memória o *Dynamic Link Library* (DLL) para comunicação com os robôs e a variável “dobot_baudrate”, onde se estabelece a taxa de transmissão de informação em bits por segundo, sendo o valor de 115200 nos robôs Dobot Magician.

Função “dobot_1_connect”

A criação do canal de comunicação com o Dobot 1 é efetuada na função “dobot_1_connect”. A função possui como variável de entrada o número da porta série de comunicação (COM) com o robô. A primeira instrução da função consiste na tentativa de estabelecer a ligação com o Dobot 1; caso a ligação seja bem-sucedida, são atribuídos valores de referência a vários parâmetros do robô, nomeadamente às coordenadas do ponto de referência, conhecido como *home position*, às velocidades e acelerações dos movimentos de juntas e cartesianos e a parâmetros comuns. Na Listagem 4.1 pode ser observada a programação da função “dobot_1_connect”.

```
def dobot_1_connect(com):
    state1 = dType.ConnectDobot(api, com, dobot_baudrate)[0]
    if state1 == dType.DobotConnect.DobotConnect_NoError:
        # Async Motion Params Setting
        dType.SetHOMEParams(api, x: 250.00, y: 0.00, z: 50.00, r: 0.00, isQueued=0)
        dType.SetPTPJointParams(api, j1Velocity: 40, j1Acceleration: 40, j2Velocity: 40, j2Acceleration: 40, j3Velocity: 40,
                                j3Acceleration: 40, j4Velocity: 40, j4Acceleration: 40, isQueued=0)
        dType.SetPTPCommonParams(api, velocityRatio: 100, accelerationRatio: 100, isQueued=0)
        dType.SetPTPCoordinateParams(api, xyzVelocity: 100, xyzAcceleration: 100, rVelocity: 100, rAcceleration: 100, isQueued=0)
```

Listagem 4.1 – Programação da função “dobot_1_connect”.

Função “dobot_2_connect”

A função “dobot_2_connect” cria o canal de comunicação com o Dobot 2 e estabelece os mesmos parâmetros encontrados na função de ligação do Dobot 1. O Dobot 2 possui um parâmetro adicional que define a saída EIO14 como uma saída digital (GPIOTypeDO). Esse parâmetro é designado “multiplex” e toma o valor 1 quando se

trata de uma saída digital. Esta configuração permite acionar o I-Extruder através do robô. Na Listagem 4.2 pode ser observada a programação da função "dobot_2_connect".

```
def dobot_2_connect(com):
    state2 = dType.ConnectDobot(api, com, dobot_baudrate)[0]
    if state2 == dType.DobotConnect.DobotConnect_NoError:
        # Async Motion Params Setting
        dType.SetHOMEParams(api, x: 250.00, y: 0.00, z: 50.00, r: 0.00, isQueued=0)
        dType.SetPTPJointParams(api, j1Velocity: 40, j1Acceleration: 40, j2Velocity: 40, j2Acceleration: 40, j3Velocity: 40,
                                j3Acceleration: 40, j4Velocity: 40, j4Acceleration: 40, isQueued=0)
        dType.SetPTPCommonParams(api, velocityRatio: 100, accelerationRatio: 100, isQueued=0)
        dType.SetPTPCoordinateParams(api, xyzVelocity: 100, xyzAcceleration: 100, rVelocity: 100, rAcceleration: 100, isQueued=0)
        dType.SetIOMultiplexing(api, address: 14, multiplex: 1, isQueued=0)
```

Listagem 4.2 – Programação da função "dobot_2_connect".

Função "dobot_emergency_stop"

Para contemplar a eventualidade de ocorrência de erros na execução do processo de produção, ou a necessidade de uma interrupção da movimentação dos robôs, foi criada a função "dobot_emergency_stop". A função possui como variáveis de entrada os números das portas de comunicação com os robôs, efetua a ligação com os Dobot 1 e 2 e executa instruções que forçam a interrupção da sequência de movimentos. No final é terminada a ligação entre o controlador central e os robôs. Na Listagem 4.3 pode ser observada a programação da função "dobot_emergency_stop".

```
def dobot_emergency_stop(port1, port2):
    dobot_1_connect(port1)
    dType.SetQueuedCmdStopExec(api)
    dType.SetQueuedCmdForceStopExec(api)
    dType.DisconnectDobot(api)
    dobot_2_connect(port2)
    dType.SetQueuedCmdStopExec(api)
    dType.SetQueuedCmdForceStopExec(api)
    dType.DisconnectDobot(api)
```

Listagem 4.3 – Programação da função "dobot_emergency_stop".

Função "dobot_1_home"

Para a movimentação do Dobot 1 para a posição de referência, conhecida como *home position*, foi criada a função "dobot_1_home". A função começa por estabelecer a ligação com o robô e de seguida executa a movimentação das juntas para as posições de referência. Por fim o ciclo *while* obriga o processador a aguardar pelo término da movimentação do robô, antes de fechar a comunicação. Na Listagem 4.4 pode ser observada a programação da função "dobot_1_home".

```
def dobot_1_home(port1):
    dobot_1_connect(port1)
    go_home = dType.SetHOMECmd(api, temp=0, isQueued=0)[0]
    while go_home > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.4 – Programação da função “dobot_1_home”.

Função “dobot_2_home”

O envio do Dobot 2 para a posição de referência é efetuado através da função “dobot_2_home”. A função tem uma estrutura idêntica à função de envio do Dobot 1 para a posição de referência, variando apenas a variável de entrada e a função de estabelecimento de ligação. Na Listagem 4.5 pode ser observada a programação da função “dobot_2_home”.

```
def dobot_2_home(port2):
    dobot_2_connect(port2)
    go_home = dType.SetHOMECmd(api, temp=0, isQueued=0)[0]
    while go_home > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.5 – Programação da função “dobot_2_home”.

Função “dobot_1_and_2_home”

As funções anteriores, de envio dos robôs para as respetivas posições de referência, foram estabelecidas para o controlo manual do processo, permitindo a manipulação de cada robô de forma individualizada. Para a execução do processo de produção em modo automático foi desenvolvida a função “dobot_1_and_2_home”. A função permite a otimização do tempo do processo através da execução da instrução de envio de ambos os robôs de forma simultânea para a posição de referência.

A função é composta, essencialmente, pela junção das funções anteriores, contudo com uma restruturação da sequência das instruções. A função efetua o envio do Dobot 1 e de seguida o Dobot 2 para as posições de referência, ficando posteriormente, num ciclo de verificação a aguardar o término de ambos os movimentos. Na Listagem 4.6 pode ser observada a programação da função “dobot_1_and_2_home”.

```python
def dobot_1_and_2_home(port1, port2):
    dobot_1_connect(port1)
    first_go_home = dType.SetHOMECmd(api, temp=0, isQueued=0)[0]
    dType.DisconnectDobot(api)
    dobot_2_connect(port2)
    last_go_home = dType.SetHOMECmd(api, temp=0, isQueued=0)[0]
    while last_go_home > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
    dobot_1_connect(port1)
    while first_go_home > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.6 – Programação da função “dobot_1_and_2_home”.

Função “dobot_1_base_part_movement”

O primeiro movimento de montagem do painel solar consiste na movimentação da peça correspondente ao fundo protetor ou *backsheet* desde o seu alimentador (alimentador 1), até à mesa de montagem, executada pelo Dobot 1. Na Figura 4.74 pode ser observada uma vista em planta do espaço de trabalho onde é representada com setas azuis a trajetória efetuada pelo robô.

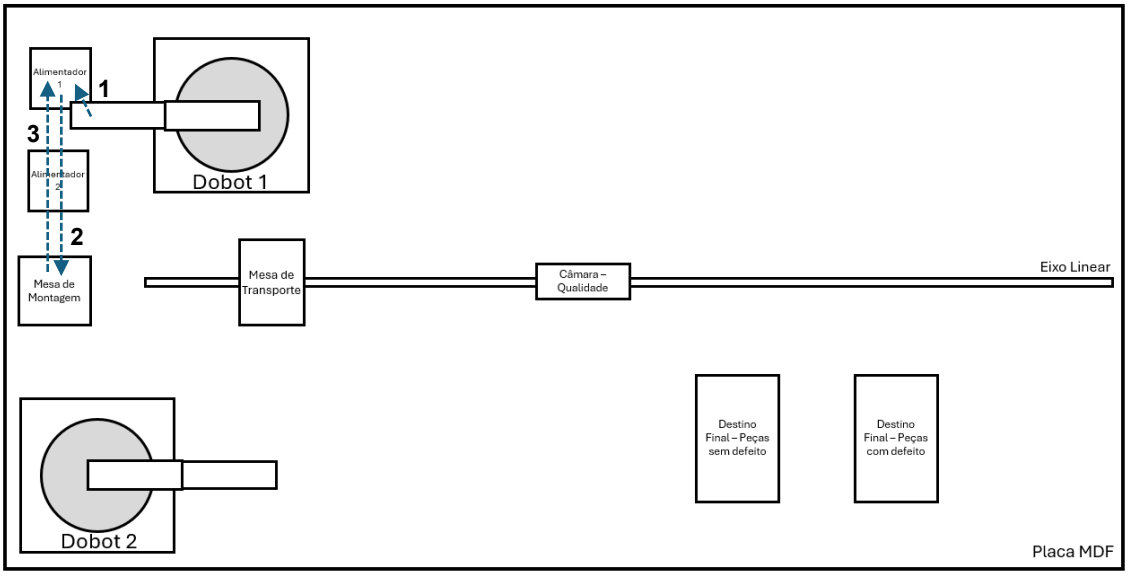

Figura 4.74 – Movimentos executados na função “dobot_1_base_part_movement”.

Para a execução deste movimento foi criada a função “dobot_1_base_part_movement”. A função possui como variáveis de entrada o número da porta de comunicação e o número da peça, no lote que está a ser produzido. A função começa por efetuar a ligação com o Dobot 1 e a definição de duas variáveis: “part_number” que efetua a subtração de uma unidade ao número da peça a ser produzida e “base_thickness” que determina a espessura entre cada placa do *backsheet*, existente no alimentador. O primeiro movimento do robô realiza o alinhamento da ventosa com o centro do alimentador (linha 1 da Figura 4.74). Este movimento ocorre apenas na produção da primeira peça do lote;

nas peças subsequentes o robô já se encontra alinhado com o alimentador. Seguidamente é efetuada a descida do elemento terminal para uma altura calculada em função do número da peça que está a ser produzida e a espessura de cada placa, permitindo assim a captação da peça em função do consumo no alimentador, sendo que neste cálculo se assume inicialmente a existência de dez peças no alimentador. Após a conclusão da descida do elemento terminal é ativada a sucção no elemento terminal. Seguidamente é efetuada a elevação da peça e a sua deslocação até à mesa de montagem onde é encaixada na cavidade (linha 2 da Figura 4.74). Após a conclusão do movimento, é desativada a sucção no elemento terminal e o robô regressa à posição centrada com o alimentador, ficando preparado para as próximas instruções (linha 3 da Figura 4.74). Na Listagem 4.7 pode ser observada a programação da função "dobot_1_base_part_movement".

```
def dobot_1_base_part_movement(port1, part_number):
    dobot_1_connect(port1)
    part_number -=1
    base_thinkness = 3.1
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.34, -37, z: 50, rHead: 0.00, isQueued=0)
    index_a = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.34, -37, (-34.5-base_thinkness*part_number),
                              rHead: 0.00, isQueued=0)[0]
    while index_a > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 1, isQueued=0)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.34, -37, z: 44, rHead: 0.00, isQueued=0)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 175.1816, y: 193.0589, z: 44, -8.8320, isQueued=0)
    index_b = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 184.02, y: 191.12, -40, -9.1320, isQueued=0)[0]
    while index_b > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 0, isQueued=0)
    time.sleep(1)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 175.1816, y: 193.0589, z: 44, rHead: 0.00, isQueued=0)
    index_c = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.34, -37, z: 50, rHead: 0.00, isQueued=0)[0]
    while index_c > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.7 – Programação da função "dobot_1_base_part_movement".

Função "auto_apply_glue"

O acionamento do I-Extruder para aplicação da cola, em modo automático, foi controlado com a função "auto_apply_glue". A função é constituída pelo acionamento da extrusão da cola durante 1.2 segundos através da ativação da interface EIO14 do Dobot 2. Foi inserida uma pausa de dois segundos após a desativação do extrusor para permitir a queda de algum excedente de cola na saída do I-Extruder. Na Listagem 4.8 pode ser observada a programação da função "auto_apply_glue".

```
def auto_apply_glue():
    dType.SetIODO(api, address: 14, level: 1, isQueued=0)
    time.sleep(1.2)
    dType.SetIODO(api, address: 14, level: 0, isQueued=0)
    time.sleep(2)
```

Listagem 4.8 – Programação da função "auto_apply_glue".

Função "glue_trigger_on"

O acionamento de saída de cola do I-Extruder, para utilização no controlo em modo manual, foi programado na função "glue_trigger_on". A Função inicia a ligação com o Dobot 2 e a ativação da interface EIO14 do robô. A função criada pode ser observada na Listagem 4.9.

```
def glue_trigger_on(port2):
    dobot_2_connect(port2)
    dType.SetIODO(api,  address: 14,  level: 1, isQueued=0)
```

Listagem 4.9 – Programação da função "glue_trigger_on".

Função "glue_trigger_off"

De forma complementar à função anterior, foi criada a função "glue_trigger_off" para a desativação da interface EIO14 do robô, desativando a saída de cola do I-Extruder. Na Listagem 4.10 pode ser observada a programação da função "glue_trigger_off".

```
def glue_trigger_off(port2):
    dobot_2_connect(port2)
    dType.SetIODO(api,  address: 14,  level: 0, isQueued=0)
```

Listagem 4.10 – Programação da função "glue_trigger_off".

Função "dobot_2_glue_part_movement"

O segundo movimento de montagem do painel solar consiste na aplicação de cola sobre o *backsheet*, efetuada pelo Dobot 2. Na Figura 4.75 pode ser observada a trajetória efetuada pelo robô.

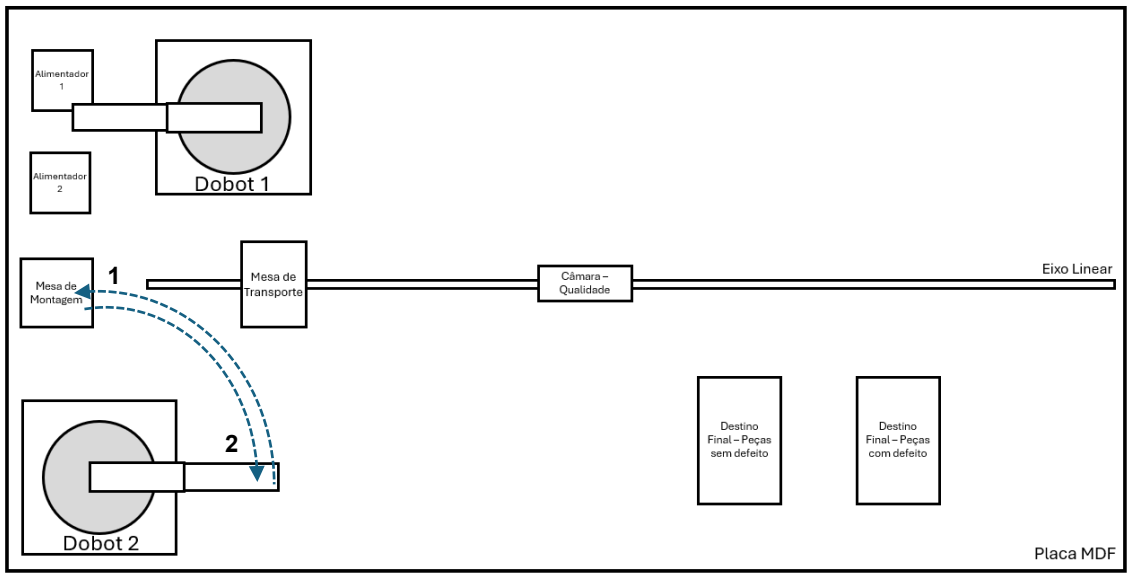

Figura 4.75 – Movimentos executados na função "dobot_2_glue_part_movement".

Para a execução do movimento foi criada a função "dobot_2_glue_part_moviment" que possui como variável de entrada o número da porta de comunicação. Esta função começa por efetuar a ligação com o Dobot 2; de seguida desloca o robô até ao centro

da mesa de montagem e desce-o para o alinhamento da extremidade de saída de cola do I-Extruder com o centro da mesa de montagem (linha 1 da Figura 4.75). Para o acionamento do I-Extruder, para a aplicação de cola, é utilizada a função "auto_apply_glue", anteriormente descrita. Após a aplicação da cola o robô regressa à sua posição de repouso (linha 2 da Figura 4.75). Na Listagem 4.11 pode ser observada a programação da função "dobot_2_glue_part_movement".

```
def dobot_2_glue_part_movement(port2):
    dobot_2_connect(port2)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVJXYZMode, -30.45, y: 195, z: 60, rHead: 0.00, isQueued=0)
    index_a = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, -41.097, y: 186.8651, -15.7949,
                              rHead: 0.00, isQueued=0)[0]
    while index_a > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    auto_apply_glue()
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, -30.45, y: 195, z: 60, rHead: 0.00, isQueued=0)
    index_b = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVJXYZMode, x: 250, y: 0, z: 50, rHead: 0.00, isQueued=0)[0]
    while index_b > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.11 – Programação da função "dobot_2_glue_part_movement".

Função "dobot_1_glass_part_movement"

O terceiro movimento de montagem do painel solar consiste na movimentação da peça correspondente às células solares desde o seu alimentador (alimentador 2) até à mesa de montagem, efetuada pelo Dobot 1. Na Figura 4.76 pode ser observada a trajetória efetuada pelo robô.

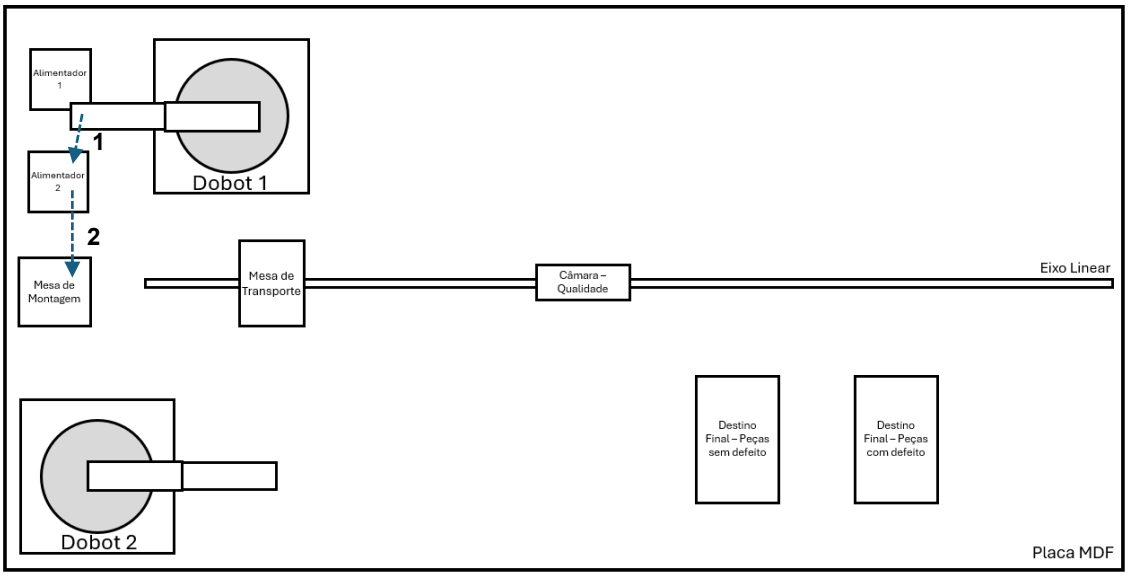

Figura 4.76 – Movimentos executados na função "dobot_1_glass_part_movement".

Para a execução do movimento foi criada a função "dobot_1_glass_part_movement". As instruções da função possuem semelhanças com a função de movimento do *backsheet*, nomeadamente, as variáveis de entrada, a ligação com o Dobot 1 e a definição das duas variáveis "part_number" e "glass_thickness" que determina a espessura entre cada uma das peças das células solares no alimentador. O robô inicia

a movimentação com a deslocação do elemento terminal até ao centro do alimentador 2, efetuando seguidamente a descida para uma posição definida com base no número da peça a ser produzida no lote e a espessura das peças das células solares (linha 1 da Figura 4.76). Com o elemento terminal posicionado sobre a peça é acionada a sucção para a prender e deslocá-la até à mesa de montagem, onde é posicionada na cavidade (linha 2 da Figura 4.76). Uma vez terminado o movimento, a ação de sucção no elemento terminal é desativada, terminando esta etapa do processo. Na Listagem 4.12 pode ser observada a programação da função “dobot_1_glass_part_movement”.

```
def dobot_1_glass_part_movement(port1, part_number):
    dobot_1_connect(port1)
    part_number -=1
    glass_thinkness = 0.1
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.3393, y: 71.3479, z: 19, rHead: 0.00, isQueued=0)
    index_a = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.3393, y: 71.3479,
                              (-73.1198-glass_thinkness*part_number), rHead: 0.00, isQueued=0)[0]
    while index_a > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 1, isQueued=0)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.3393, y: 71.3479, z: 19, rHead: 0.00, isQueued=0)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 175.1816, y: 193.0589, z: 44, rHead: 0.00, isQueued=0)
    index_b = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 187.53, y: 191.00, -41, -1, isQueued=0)[0]
    while index_b > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    index_c = dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 0, isQueued=0)[0]
    time.sleep(1)
    while index_c > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.12 – Programação da função “dobot_1_glass_part_movement”.

Função “dobot_1_transport_movement”

Após a montagem do painel solar existe a necessidade de efetuar a sua deslocação desde a mesa de montagem até à mesa de transporte, que o irá movimentar até ao seu recetor de peças final. Na Figura 4.77 pode ser observado o movimento efetuado pelo robô.

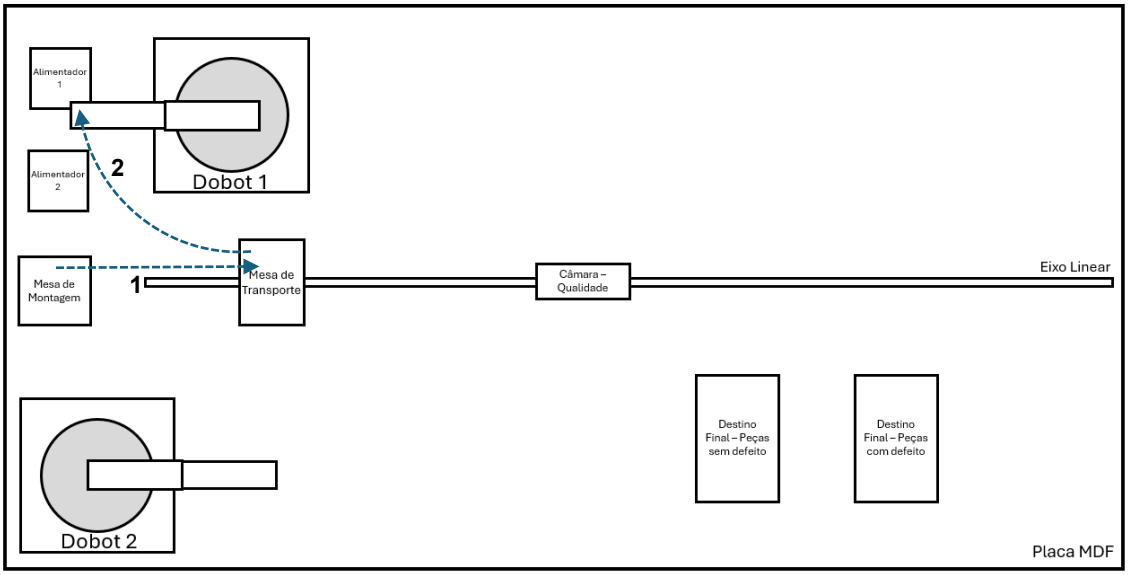

Figura 4.77 – Movimentos executados na função “dobot_1_transport_movement”.

Para a realização do movimento foi criada a função “dobot_1_transport_movement”. O movimento inicia-se na posição onde terminou a função “dobot_1_glass_part_movement”, sendo ativada a sucção no elemento terminal para prender o painel solar e deslocá-lo até à mesa de transporte onde é largado (linha 1 da Figura 4.77). Uma vez desativada a sucção no elemento terminal, o robô regressa à posição de repouso (linha 2 da Figura 4.77). Na Listagem 4.13 pode ser observada a programação da função “dobot_1_transport_movement”.

```
def dobot_1_transport_movement(port1):
    dobot_1_connect(port1)
    dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 1, isQueued=0)
    time.sleep(0.5)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, x: 188.3393, y: 191.8523, z: 54.630, rHead: 0.00, isQueued=0)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, -51.0842, y: 193.9331, z: 54.630, rHead: 0.00, isQueued=0)
    index_b = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, -51.0842, y: 193.9331, z: 26.6709,
                              rHead: 0.00, isQueued=0)[0]
    while index_b > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.SetEndEffectorSuctionCup(api, enableCtrl: 1, on: 0, isQueued=0)
    time.sleep(1)
    dType.SetPTPCmd(api, dType.PTPMode.PTPMOVLXYZMode, -51.0842, y: 189.9331, z: 54.630, rHead: 0.00, isQueued=0)
    index_c = dType.SetPTPCmd(api, dType.PTPMode.PTPMOVJXYZMode, x: 187.34, -37, z: 50, rHead: 0.00, isQueued=0)[0]
    while index_c > dType.GetQueuedCmdCurrentIndex(api)[0]:
        dType.dSleep(100)
    dType.DisconnectDobot(api)
```

Listagem 4.13 – Programação da função “dobot_1_transport_movement”.

4.1.2 Programação do Arduino

A programação dos movimentos dos motores de passo foi estruturada em dois níveis, um associado ao Arduino UNO e o outro ao controlador central. Na Figura 4.78 pode ser observada a hierarquia de controlo dos motores de passo.

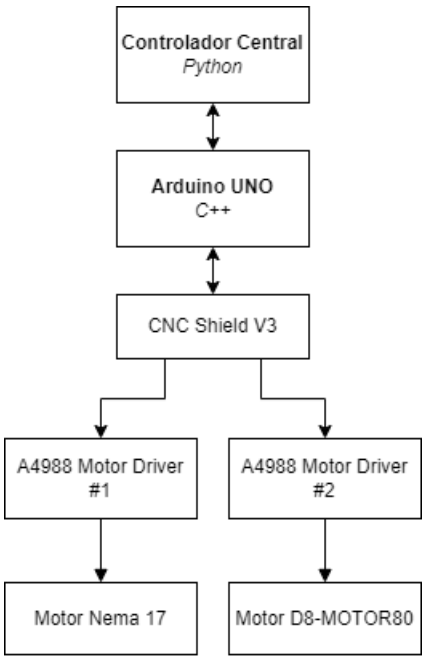

Figura 4.78 – Hierarquia de controlo dos motores de passo.

O comando de movimentos dos motores de passo, em Python, está a cargo de funções do módulo interno *motorcommunication*.

Por simplicidade, foi adotada a nomenclatura de Motor 1 para o motor de passo Nema 17 e Motor 2 para o motor de passo D8-MOTOR80 e os respectivos eixos lineares foram designados eixos 1 e 2. Adicionalmente foi convencionado o sentido de movimento do Motor 1 e eixo linear 1 da esquerda para a direita como sentido positivo (ou frente), e o movimento do Motor 2 e eixo linear 2 de cima para baixo como o sentido positivo. Na Figura 4.79 pode ser observado os sentidos convencionados para os motores e eixos lineares.

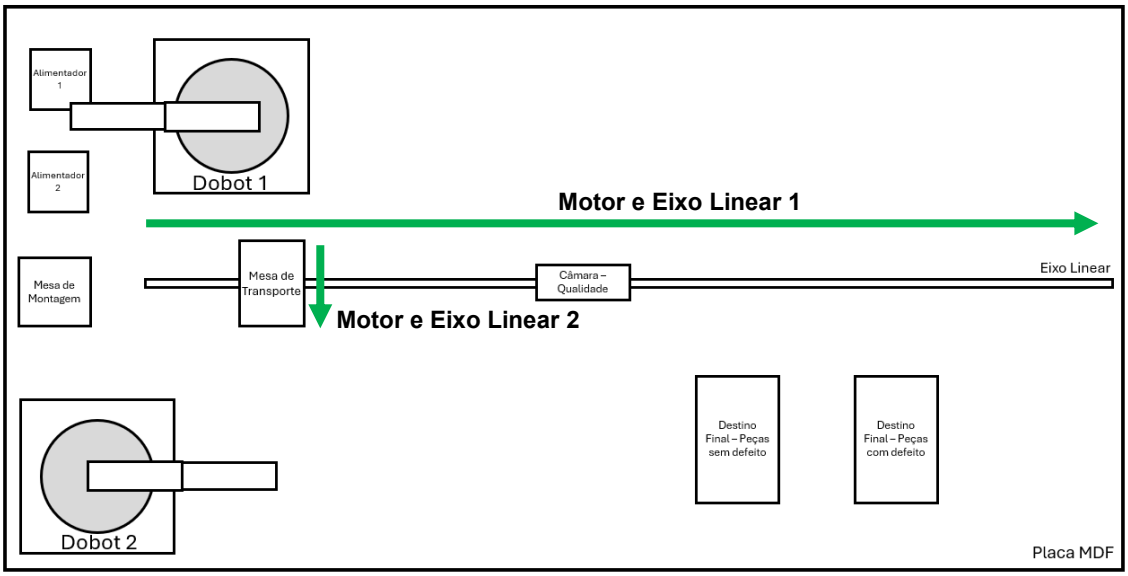

Figura 4.79 – Sentidos convencionados para os motores e eixos lineares.

Programação Arduino

O programa executado no Arduino UNO é constituído essencialmente por três partes: uma primeira parte onde são importadas bibliotecas e onde são definidas as constantes e variáveis e os seus respetivos valores; uma segunda parte, constituída pela função "setup", que é executada no arranque do equipamento e onde são estabelecidas as características das entradas e saídas do Arduino UNO e, por último a função "loop", que executa instruções num ciclo infinito, enquanto o sistema se encontra em funcionamento. Os motores de passo mantiveram a nomenclatura existente na placa CNC Shield V3, onde os eixos coordenados são designados X, Y e Z. No presente projeto foi utilizado o eixo X para o controlo do Motor 1 e o eixo Y para o controlo do Motor 2.

Bibliotecas e variáveis

A primeira biblioteca externa importada foi o “Arduino.h” que possui as definições e funções necessárias para controlar o microprocessador Arduino UNO. A segunda biblioteca externa importada foi o “A4988.h”, contendo as funções necessárias para o controlo dos *drivers* A4988 instalados na placa CNC Shield V3.

As constantes criadas dividem-se em dois grupos: as constantes transversais a todos os motores que estão a ser controlados e as específicas para cada motor. Para as constantes específicas foi introduzido, na nomenclatura, o sufixo “_X” ou “_Y” consoante o motor. Daqui em diante as constantes especificas poderão ser referidas sem sufixo, por forma a manter a generalidade. Foram estabelecidas as seguintes constantes:

- “STOPPER_PIN” – Pinos do Arduino UNO ligados aos sensores de fim de curso de cada eixo;
- “MOTOR_STEPS” – Passos do motor por cada rotação;
- “RPM” – Rotações por minutos dos motores de passo;
- “MICROSTEPS” – Configuração do *Microstepping* dos motores de passo;
- “ENABLE”, “SLEEP”, “DIR” e “STEP” – Pinos do Arduino UNO ligados a propriedades gerais do controlo dos motores de passo.

Para a configuração dos *drivers* A4988 foram criadas duas variáveis que representam cada uma um motor de passo, designadas “stepperX” e “stepperY”.

Adicionalmente foram criadas variáveis de apoio à execução do programa, nomeadamente o “input” para permitir a introdução de uma instrução por parte do utilizador e as variáveis “direction” que representam o sentido de deslocação dos motores. Na Listagem 4.14 pode ser observado o código de declaração das bibliotecas das constantes e das variáveis do programa carregado no Arduino UNO.

```
#include <Arduino.h>

// Pin on Arduino correspondent to CNC Shield END STOPS
#define STOPPER_PIN_X 9 // X+/X- END STOPS
#define STOPPER_PIN_Y 10 // Y+/Y- END STOPS
#define STOPPER_PIN_Z 11 // Z+/Z- END STOPS

// Motor steps per revolution
#define MOTOR_STEPS 200
// Target RPM for X and Y axis motores
#define RPM_X 50
#define RPM_Y 100

// Microsteps for X and Y axis motores
#define MICROSTEPS_X 16 // 3 Jumpers
#define MICROSTEPS_Y 4 // 1 Jumper

// General Settings
#define ENABLE 8
#define SLEEP 13

// X motor settings
#define DIR_X 5
#define STEP_X 2
// Y motor settings
#define DIR_Y 6
#define STEP_Y 3

#include "A4988.h"
A4988 stepperX(MOTOR_STEPS, DIR_X, STEP_X, ENABLE);
A4988 stepperY(MOTOR_STEPS, DIR_Y, STEP_Y, ENABLE);
char input;
int direction_x;
int direction_y;
```

Listagem 4.14 – Declaração das bibliotecas e variáveis do programa carregado no Arduino UNO.

Função "setup"

A função inicia-se com a definição da taxa de transmissão de informação de 500000 bits/s para a comunicação série com o Arduino UNO. Seguidamente são definidas as configurações de alguns pinos digitais do Arduino UNO, nomeadamente os pinos de ligação aos sensores de fim de curso e os pinos de acionamento dos *drivers*. O CNC Shield V3 permite a ligação de dois sensores de fim de curso, por eixo, contudo, no Arduino UNO, estes dois sensores encontram-se ligados a um único pino, sendo o comportamento igual no acionamento de um ou outro sensor. Dada esta característica é definido apenas uma variável correspondente ao sensor de fim de curso por eixo. No caso de acionamento de um sensor, a sua distinção é efetuada com recurso à variável "direction" definida durante o "loop". Os motores de passo são configurados com os parâmetros de rotações por minuto e *Microstepping*, e o seu estado de ativação é inicializado com o valor de "LOW". Na Listagem 4.15 pode ser observado o código da função "setup" do programa.

```
void setup() {

  Serial.begin(500000);

  pinMode(STOPPER_PIN_X, INPUT_PULLUP);
  pinMode(STOPPER_PIN_Y, INPUT_PULLUP);
  pinMode(STOPPER_PIN_Z, INPUT_PULLUP);
  pinMode(ENABLE, OUTPUT);

  stepperX.begin(RPM_X, MICROSTEPS_X);
  stepperY.begin(RPM_Y, MICROSTEPS_Y);

  stepperX.setEnableActiveState(LOW);
  stepperY.setEnableActiveState(LOW);

  Serial.println("ARDUINO READY");
}
```

Listagem 4.15 – Código da função “setup” do programa.

Função “loop”

O bloco de instruções que executa o controlo dos motores, encontra-se na função “loop”. No caso do presente projeto as ações a realizar são determinadas por uma informação de entrada no Arduino, podendo tratar-se do sinal de um sensor de fim de curso ou uma mensagem recebida através da comunicação série. Face a esta distinção, a função “loop” encontra-se estruturada em duas partes, uma parte relativa aos comandos a executar em resposta à variação do sinal de um sensor e outra parte, relativa aos movimentos a realizar por um motor em resposta a uma instrução recebida.

Os sensores de fim de curso que foram utilizados foram conectados na configuração de normalmente abertos, de modo que o acionamento do sensor feche o circuito. No código, as entradas de sinal dos sensores foram definidas como “INPUT_PULLUP” que ativam uma resistência de *pull-up* interna do Arduino. Por definição os pinos de ligação dos sensores, quando não estão acionados, permanecem com o valor “HIGH”. No acionamento de um sensor a tensão de entrada reduz-se para 0 V alterando-se o valor lógico para “LOW”. Neste sentido, a programação foi elaborada para a deteção do acionamento do sensor de fim de curso. Quando é acionado o sensor de fim de curso do eixo X ou Y, é emitida uma mensagem, para o controlador central, indicativa de “STOPPER REACHED”, com a indicação do motor respetivo. Seguidamente é efetuado o corte de alimentação do motor, provocando a sua paragem. Contudo, para evitar o acionamento constante do sensor de fim de curso depois deste ser atingido, foi incluído um movimento no sentido oposto ao anterior, por forma a desativar o sensor. A amplitude desse deslocamento foi ajustada de modo a ser suficiente para desativar o sensor. Na Listagem 4.16 pode ser observada a programação elaborada para os sensores, na função “loop”.

```
void loop() {
  // Check if X axis END STOP was hit
  if (digitalRead(STOPPER_PIN_X) == LOW){
    Serial.println("MOTOR 1 STOPPER REACHED");
    stepperX.stop();
    stepperX.enable();
    if (direction_x == 2){
      stepperX.startRotate(-10);
    }
    if (direction_x == 1){
      stepperX.startRotate(10);
    }
  }
  // Check if Y axis END STOP was hit
  if (digitalRead(STOPPER_PIN_Y) == LOW){
    Serial.println("MOTOR 2 STOPPER REACHED");
    stepperY.stop();
    stepperY.enable();
      if (direction_y == 3){
        stepperY.startRotate(10);
      }
      if (direction_y == 4){
        stepperY.startRotate(-10);
      }
  }
```

Listagem 4.16 – Programação dos sensores de fim de curso na função “loop” do programa carregado no Arduino UNO.

A operação dos motores de passo pelo Arduino é feita em resposta a mensagens enviadas pelo controlador central. As mensagens relativas a diferentes ações são distinguidas através de uma letra. Em resposta à instrução, o Arduino envia ao controlador central uma mensagem com a instrução recebida e outra com uma frase que descreve a instrução a ser realizada. Adicionalmente, nas instruções de movimentação dos motores, são definidas as variáveis “direction” para distinguir o sentido de deslocação dos motores. Este dado é necessário para o movimento subsequente ao acionamento de um sensor de fim de curso. No Quadro 4.7 encontram-se resumidas as instruções programadas para o funcionamento dos motores de passo e respetivas letras de acionamento.

Quadro 4.7 – Acionamento e instruções de funcionamento dos motores de passo.

Descrição da ação	Letra de acionamento	Ação realizada
Motor 1 para a posição de referência	*h*	100000 passos no sentido negativo
Motor 1 no sentido positivo	*f*	5000 passos no sentido positivo
Motor 1 no sentido negativo	*b*	5000 passos no sentido negativo
Motor 1 para o controlo de qualidade	*c*	1580 passos no sentido positivo
Motor 2 para a posição de referência	*t*	10000 passos no sentido negativo
Motor 2 no sentido positivo	*r*	6000 passos no sentido positivo
Motor 1 até zona de expulsão de peça conforme	*g*	1400 passos no sentido positivo
Motor 1 até zona de expulsão de peça com defeito	*d*	2500 passos no sentido positivo
Corte de alimentação elétrica do Motor 1 e 2	*x*	Corte de alimentação de ligação dos motores através da alteração do valor do pino “ENABLE” para “HIGH”

Na Listagem 4.17 pode ser observada a programação elaborada para o comando dos motores de passo, na função “loop”.

```
if(Serial.available()){
    input = Serial.read();
    Serial.println(input);

    if (input == 'h'){
      stepperX.enable();
      Serial.print("MOTOR 1 GOING TO HOME");
      direction_x = 1;
      stepperX.startRotate(-100000);
    }
    if (input == 'f'){
      Serial.print("MOTOR 1 MOVING FORWARD");
      direction_x = 1;
      stepperX.startRotate(5000);
    }
    if (input == 'b'){
      Serial.print("MOTOR 1 MOVING BACKWARD");
      direction_x = 2;
      stepperX.startRotate(-6000);
    }
    if (input == 'c'){
      Serial.print("MOTOR 1 GOING TO CAMERA");
      direction_x = 2;
      stepperX.startRotate(1580);
    }
    if (input == 't'){
      Serial.print("MOTOR 2 MOVING BACKWARD");
      direction_y = 3;
      stepperY.startRotate(-10000);
    }
    if (input == 'r'){
      Serial.print("MOTOR 2 MOVING FORWARD");
      direction_y = 4;
      stepperY.startRotate(6000);
    }
    if (input == 'g'){
      Serial.print("MOTOR 1 MOVING TO GOOD PART EXPULSION");
      direction_y = 2;
      stepperX.startRotate(1400);
    }
    if (input == 'd'){
      Serial.print("MOTOR 1 MOVING TO DEFECTIVE PART EXPULSION");
      direction_y = 2;
      stepperX.startRotate(2500);
    }
    if (input == 'x'){
      Serial.print("TURNING MOTOR 1 & 2 OFF");
      digitalWrite(ENABLE, HIGH);
    }
  }
}
```

Listagem 4.17 – Bloco de comando dos motores de passo na função “loop”.

Programação Python

O comando dos motores de passo pelo controlador central consiste essencialmente no envio das mensagens descritas na função “loop” do ponto anterior.

Função “motor_1_home”

Com o objetivo de garantir a consistência nas deslocações dos motores de passo foi necessário estabelecer, à semelhança do efetuado para os robôs, uma posição de referência de partida dos motores. A colocação do Motor 1 na posição de referência foi programada na função “motor_1_home”. A função tem como variável de entrada o número da porta de comunicação com o Arduino. A primeira instrução da função consiste em estabelecer a ligação com o Arduino UNO. Seguidamente é enviada a

instrução de *home* do motor de passo, através da letra “h”. Por fim, quando for ativado o sensor de fim de curso, é recebida a mensagem “MOTOR 1 STOPPER REACHED” que indica a chegada do motor à posição de referência. Na Figura 4.80 pode ser observada a posição de referência do Motor 1, representada pela linha azul e o sentido de deslocação do motor até à posição de referência, representada pela seta verde.

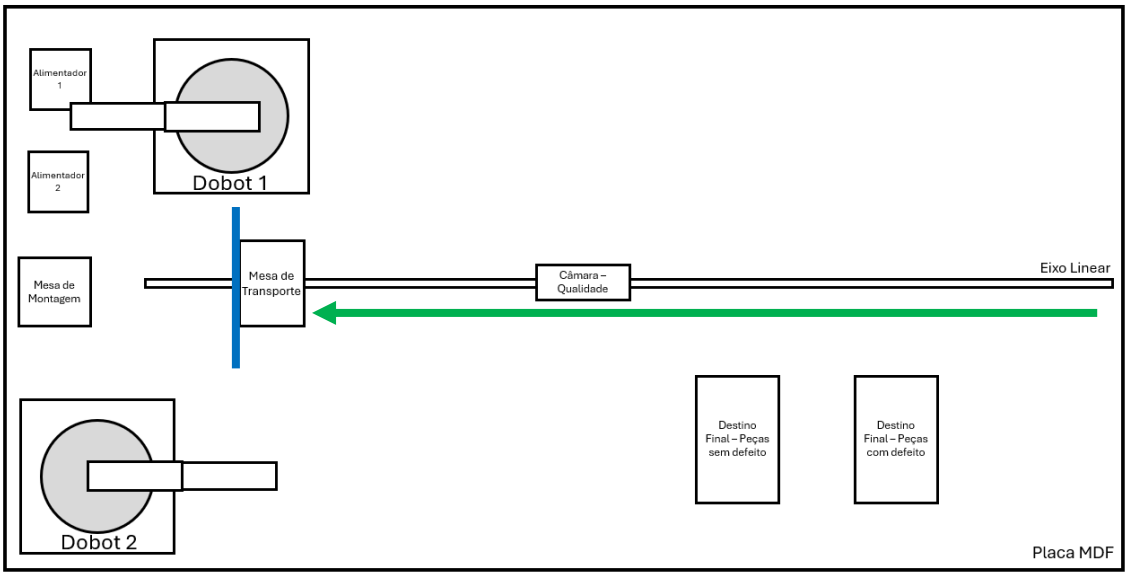

Figura 4.80 – Posição de referência e sentido de deslocação do Motor 1 na função “motor_1_home”.

Na Listagem 4.18 pode ser observada a programação da função “motor_1_home”.

```
def motor_1_home(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('h', 'utf-8'))
    while spark.readline().strip().decode('utf-8') != "MOTOR 1 STOPPER REACHED":
        time.sleep(0.2)
    spark.close()
```

Listagem 4.18 – Programação da função “motor_1_home”.

Função “motor_2_home”

À semelhança do efetuado para o Motor 1, foi criada a função “motor_2_home” para o envio do Motor 2 para a posição de referência. A função possui uma estrutura idêntica à do Motor 1 variando o código enviado para o Arduino UNO que passa a ter a letra “t”. Quando for ativado o sensor de fim de curso é recebida a mensagem “MOTOR 2 STOPPER REACHED”, indicativa de que o Motor 2 atingiu a posição de referência. Na Figura 4.81 pode ser observada a posição de referência do Motor 2, representada pela linha azul e o sentido de deslocação do motor até à posição de referência, representada pela seta verde.

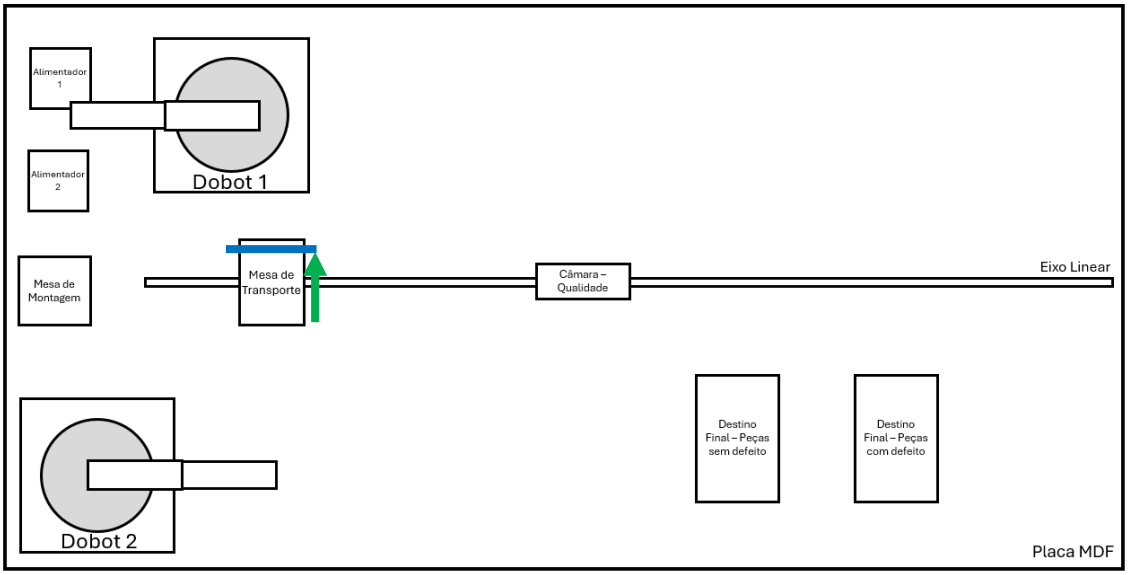

Figura 4.81 – Posição de referência e sentido de deslocação do Motor 2 na função "motor_2_home".

Na Listagem 4.19 pode ser observada a programação da função "motor_2_home".

```
def motor_2_home(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('t', 'utf-8'))
    while spark.readline().strip().decode('utf-8') != "MOTOR 2 STOPPER REACHED":
        time.sleep(0.2)
    spark.close()
```

Listagem 4.19 – Programação da função "motor_2_home".

Função "motor_1_move_forward"

Para utilização no controlo em modo manual, foi implementada uma função para acionamento do Motor 1 no sentido positivo. A função consiste no envio da instrução "f" que aciona o motor para efetuar um número pré-definido (5000) de passos no sentido positivo. Na Listagem 4.20 pode ser observada a programação da função "motor_1_move_forward".

```
def motor_1_move_forward(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('f', 'utf-8'))
    spark.close()
```

Listagem 4.20 – Programação da função "motor_1_move_forward".

Função "motor_1_move_backward"

À semelhança da função "motor_1_move_forward", foi criada a função "motor_1_move_backward" para utilização no controlo em modo manual, para efeitos de testes do movimento do Motor 1 no sentido negativo. A função tem uma estrutura idêntica, variando apenas a instrução "b" que aciona o motor para efetuar um número

estipulado de passos (5000) no sentido negativo. Na Listagem 4.21 pode ser observada a programação da função “motor_1_move_backward”.

```
def motor_1_move_backward(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('b', 'utf-8'))
    spark.close()
```

Listagem 4.21 – Programação da função “motor_1_move_backward”.

Função “motor_1_movement_to_quality”

No processo de produção do painel solar existe uma etapa de verificação de qualidade, efetuada com recurso a uma câmara localizada sensivelmente a meio do eixo linear 1. Foi criada a função “motor_1_movement_to_quality” para a deslocação do Motor 1 e respetiva peça colocada na mesa de transporte, até ao local designado para a verificação. A função é constituída pelas etapas: estabelecimento de ligação com o Arduino UNO, envio de instrução para a deslocação do Motor 1 e pausa para garantir a conclusão da deslocação antes de efetuar a desconexão com o Arduino UNO. Na Figura 4.82 pode ser observada a deslocação efetuada pelo Motor 1 até à posição de verificação de qualidade.

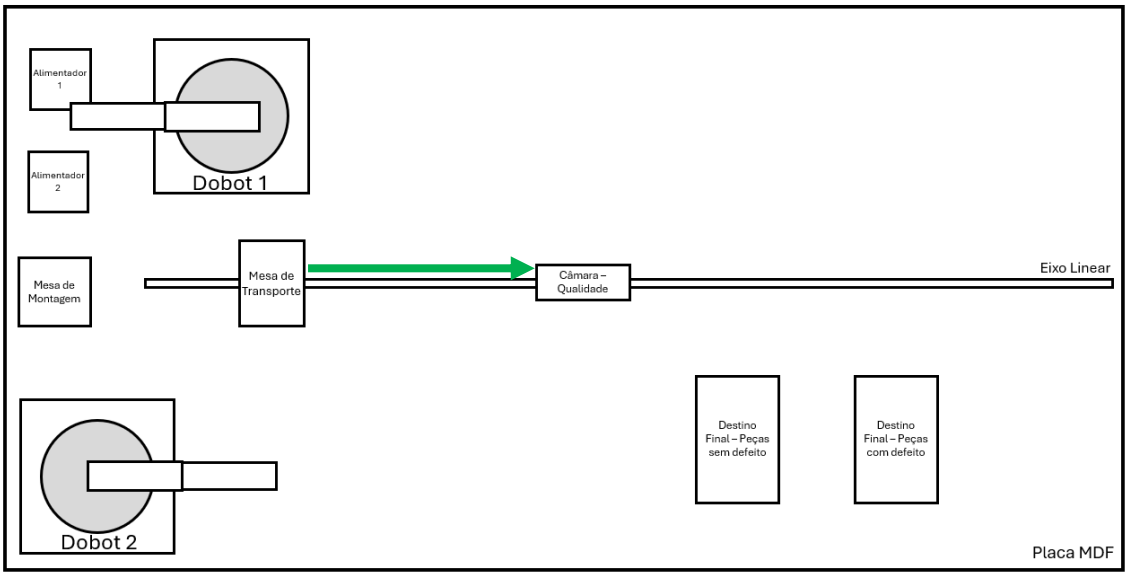

Figura 4.82 – Sentido de deslocação do Motor 1 na função “motor_1_movement_to_quality”.

Na Listagem 4.22 pode ser observada a programação da função “motor_1_movement_to_quality”.

```
def motor_1_movement_to_quality(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('c', 'utf-8'))
    time.sleep(6)
    spark.close()
```

Listagem 4.22 – Programação da função “motor_1_movement_to_quality”.

Função "motor_1_good_part_movement"

Após a verificação de qualidade e mediante o resultado, a peça produzida é deslocada para o seu recetor final. Para tratar as peças que passam no teste de qualidade foi criada a função "motor_1_good_part_movement" que desloca o Motor 1 desde o ponto de controlo de qualidade até ao recetor de peças conformes. Neste caso o código da mensagem e enviar é a letra "g". Foi calculado um tempo de cinco segundo para a execução da deslocação, findos os quais é terminada a ligação com o Arduino UNO. Na Figura 4.83 pode ser observada a deslocação efetuada pelo Motor 1 até à posição expulsão de peça conforme.

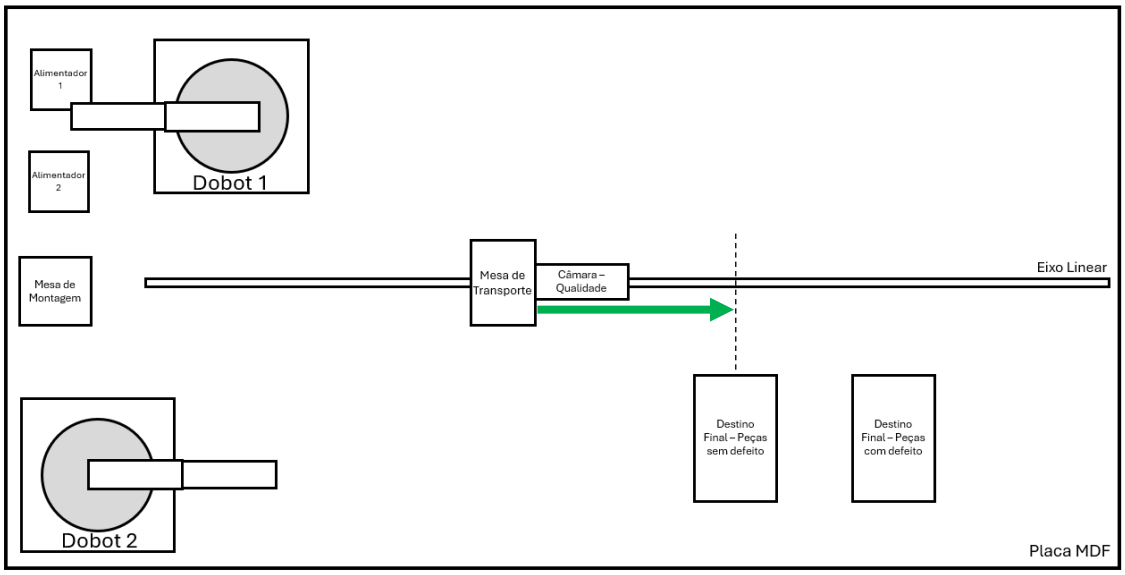

Figura 4.83 – Sentido de deslocação do Motor 1 na função "motor_1_good_part_movement".

Na Listagem 4.23 pode ser observada a programação da função "motor_1_good_part_movement".

```
def motor_1_good_part_movement(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('g', 'utf-8'))
    time.sleep(5)
    spark.close()
```

Listagem 4.23 – Programação da função "motor_1_good_part_movement".

Função "motor_1_defective_part_movement"

À semelhança do implementado para o destino de peça conforme, foi criada a função "motor_1_defective_part_movement" para a deslocação de peças com defeito. A deslocação é efetuada pelo Motor 1 tendo como ponto de partida o local de verificação de qualidade. A função envia a instrução de letra "d" ao Arduino UNO, sendo a instrução que ativa a movimentação do Motor 1 até ao local de expulsão de peça com defeito. Na Figura 4.84 pode ser observada a deslocação efetuada pelo Motor 1 até à posição de expulsão de peça com defeito.

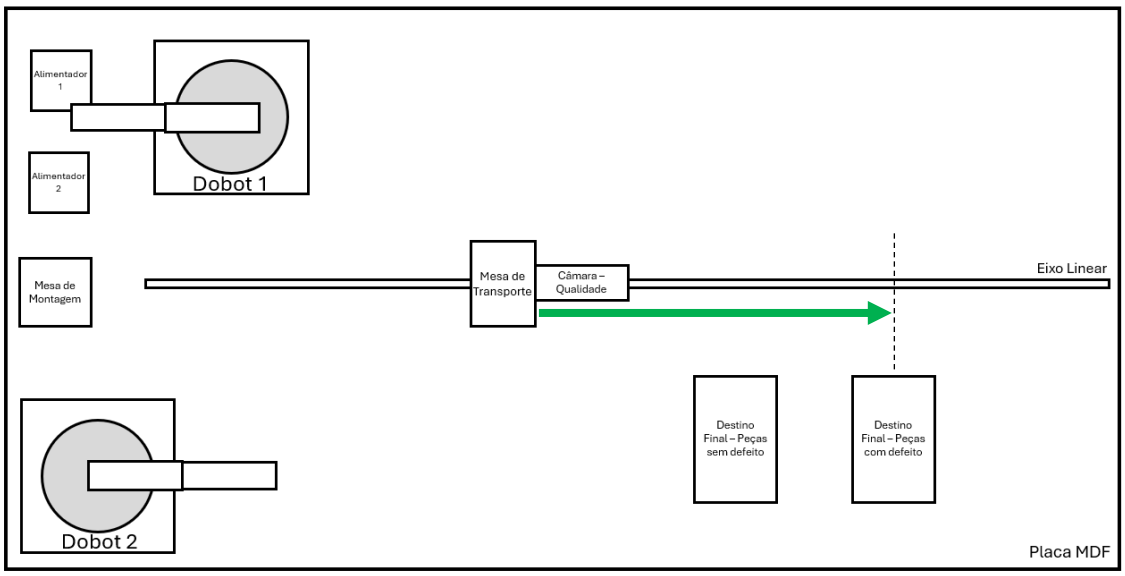

Figura 4.84 – Sentido de deslocação do Motor 1 na função "motor_1_defective_part_movement".

Na Listagem 4.24 pode ser observada a programação da função "motor_1_defective_part_movement".

```
def motor_1_defective_part_movement(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('d', 'utf-8'))
    time.sleep(9)
    spark.close()
```

Listagem 4.24 – Programação da função "motor_1_defective_part_movement".

Função "motor_2_part_expulsion"

A colocação do painel solar produzido no seu recetor final resulta da expulsão da peça da mesa de transporte, através do acionamento do Motor 2 no sentido positivo. Para a realização da ação de expulsão foi criada a função "motor_2_part_expulsion". Nesta função envia-se a instrução de letra 'r' para o Arduino UNO. A função permanece em espera até à receção da informação que foi acionado o sensor de fim de curso associado ao Motor 2, efetuando-se posteriormente o corte de ligação com o Arduino UNO. Na Listagem 4.25 pode ser observada a programação da função "motor_2_part_expulsion".

```
def motor_2_part_expulsion(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('r', 'utf-8'))
    while spark.readline().strip().decode('utf-8') != "MOTOR 2 STOPPER REACHED":
        time.sleep(0.2)
    spark.close()
```

Listagem 4.25 – Programação da função "motor_2_part_expulsion".

Função "motor_emergency_stop"

Para contemplar a ocorrência de erros ou necessidade de uma interrupção da movimentação dos motores, foi criada a função "motor_emergency_stop". A função inclui o envio da instrução 'x' que efetua a interrupção do movimento dos motores. Na Listagem 4.26 pode ser observada a programação da função "motor_emergency_stop".

```
def motor_emergency_stop(com):
    spark = serial.Serial(port=com, baudrate=motor_baudrate, timeout=None)
    print(spark.readline().strip().decode('utf-8'))
    spark.write(bytes('x', 'utf-8'))
    spark.close()
```

Listagem 4.26 – Programação da função "motor_emergency_stop".

4.1.3 Programação do teste de controlo de qualidade

A inclusão de uma etapa de verificação de qualidade neste processo automático levou à incorporação de técnicas de visão computacional. Por forma a simular a ocorrência de defeitos, decidiu-se usar placas com diferentes cores na construção do painel solar à escala. Concretamente, consideram-se os painéis de cor azul como produtos conformes, enquanto painéis com qualquer outra cor são considerados produtos com defeito. A captura da imagem da peça é efetuada por uma câmara ligada ao controlador central através de USB. A câmara foi posicionada de modo a garantir que a peça preencha a maior parte da imagem captada.

O procedimento de validação por visão computacional encontra-se no módulo interno *cameravision*. A programação deste módulo é sustentada por dois módulos externos, nomeadamente o "cv2" e o "numpy". As variáveis globais que foram criadas no módulo interno *cameravision* foram o "cam_port", um número inteiro que identifica a câmara a utilizar para a captura de imagem, a variável "cam" que representa a câmara escolhida e que tem associados métodos de captura de imagem (segundo a implementação no módulo "cv2") e a variável "path" que contém o caminho de destino para a gravação das imagens captadas pela câmara. Na Listagem 4.27 podem ser observadas as variáveis globais do módulo interno *cameravision*.

```
cam_port = 1
cam = cv2.VideoCapture(cam_port)
path = 'C:/solarproduction/'
```

Listagem 4.27 – Variáveis globais do módulo *cameravision*.

Função "quality_control"

Para desencadear a verificação de qualidade da peça produzida quer no modo manual, quer no modo automático foi criada a função "quality_control". A função é composta por

três partes, uma primeira consistindo no método de captura de imagem da câmara, efetuada na instrução “cam.read”, onde são devolvidas duas variáveis, um valor booleano, que indica se a imagem foi captada com sucesso e uma variável “image”, contendo a imagem captada pela câmara.

Numa segunda parte é efetuada a gravação da imagem captada pela câmara, numa pasta de destino definida na variável global “path”. É atribuído como nome do ficheiro o número sequencial da peça. O armazenamento das imagens permite a sua consulta em qualquer momento, após a produção, para eventual análise mais detalhada.

Na terceira parte da função é analisada a variável temporária “image”. A variável é composta por uma imagem RGB (Vermelho, Verde, Azul), onde a cor de cada pixel é representada por três componentes: um para o vermelho, um para o verde e outro para o azul. Estes componentes são armazenados sob a forma de três matrizes 2D, sendo que cada matriz corresponde a um canal de cor. O valor em cada elemento da matriz indica a intensidade do respetivo canal de cor nesse pixel. Com recurso à técnica de *slicing* do Python, é efetuada a separação dos canais de cores em três variáveis “b”, “g” e “r” dos valores correspondentes às cores azul, verde e vermelho, respetivamente. Posteriormente é calculada a intensidade média de cada canal de cor através da função “mean” do módulo externo “numpy”. Por fim, mediante a cor predominante, é devolvida a resposta na variável “part_ok” sendo que se a coloração predominante for azul, a variável é devolvida como verdadeira, valor indicativo de peça conforme. Caso seja outra cor a peça encontra-se com defeito e é atribuído o valor falso à variável. Na Listagem 4.28 pode ser observada a programação da função “quality_control”.

```python
def quality_control():
    part_ok = False
    # read input from camera
    result, image = cam.read()

    part_id = productionhistory.get_part_id()
    file_name = str(part_id) + ".png"
    cv2.imwrite(os.path.join(path, file_name), image)

    b = image[:, :, 0]
    g = image[:, :, 1]
    r = image[:, :, 2]

    b_mean = np.mean(b)
    g_mean = np.mean(g)
    r_mean = np.mean(r)

    # find most prominent color
    if b_mean > g_mean and b_mean > r_mean:
        print("Good Part")
        part_ok = True
        return part_ok
    if g_mean > r_mean and g_mean > b_mean:
        print("Defective Part")
        return part_ok
    if r_mean > b_mean and r_mean > g_mean:
        print("Defective Part")
        return part_ok
```

Listagem 4.28 – Programação da função “quality_control”.

Na Figura 4.85 podem ser observados dois exemplos de captura de imagens pela câmara para a verificação de qualidade. Do lado esquerdo um exemplo de peça conforme (cor azul) e do lado direto uma peça com defeito (cor diferente de azul).

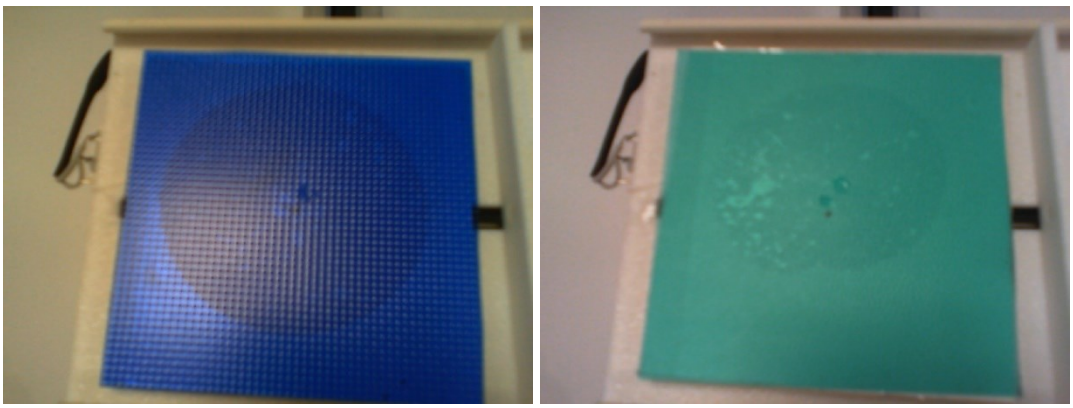

Figura 4.85 – Exemplos de imagens para verificação de qualidade. Peça conforme (esquerda) peça com defeito (direita).

4.1.4 Programação da interface gráfica e controlo de produção

O programa que permite o controlo e supervisão do processo por um operador foi implementado na forma de uma interface gráfica, permitindo uma interação mais intuitiva e expedita entre o utilizador e os equipamentos ou processo. A programação das janelas da interface gráfica foi realizada em Python, utilizando o módulo externo "Tkinter". Além da interação do utilizador com os equipamentos, foi idealizada a implementação de alguns indicadores auxiliares para o controlo de produção, tendo sido estipulados os seguintes objetivos da interface gráfica:

1. Acionamento da produção em modo automático com indicação do número de peças a produzir;
2. Visualização do estado da produção, nomeadamente sobre a fase atual do processo, o número de peças produzidas e o número de peças com defeito;
3. Controlo manual e individualizado de cada equipamento do processo para efeitos de teste e correção de erros;
4. Interrupção manual de todos os equipamentos em caso de falha ou erro.

Perante os objetivos estabelecidos a aplicação foi desenvolvida com cinco módulos internos, sendo dois de apoio para execução de pequenas tarefas. O módulo principal da aplicação é designado por *main* e possui a primeira janela da interface gráfica, onde é configurada e acionada a produção automática. O código que comanda a produção automática encontra-se no módulo *automaticproduction*. O módulo *main* contém uma opção que permite ao utilizador visualizar uma segunda janela onde pode ser realizado o controlo manual dos equipamentos. Esta janela foi programada no módulo *manualcontrol*. Para a deteção de erros de ligação aos equipamentos e de preenchimento de campos no controlo manual, foi criado o módulo *detecterror*. A criação de um registo histórico das peças produzidas foi programada no módulo *productionhistory*. A interface gráfica permite acionar o processo de produção, a partir dos quais são executados os módulos internos e funções desenvolvidas no projeto.

Módulo main

A primeira janela da interface gráfica, apresentada ao iniciar-se a aplicação, foi programada no módulo interno *main*. Foi criado o objeto "root", que contém toda a informação e componentes da interface gráfica. Foram configurados os parâmetros da janela, nomeadamente o nome, a sua dimensão e cor de fundo. Na Listagem 4.29 pode ser observada a parametrização da janela.

```
root = tkinter.Tk()
root.title("Solar Panel Production Control")
root.minsize(width=515, height=420)
root.configure(bg='white')
```

Listagem 4.29 – Parametrização da janela do módulo “Tkinter”.

No módulo *main* foram criadas duas variáveis globais, a variável “parts_produced” que indica o número de peças produzidas, em modo automático e a variável “defective_parts” que contabiliza o número de peças que foram produzidas e detetadas com defeito no controlo de qualidade. Ambas as variáveis são iniciadas com o valor zero no arranque do programa e são incrementadas durante a execução da produção.

Após serem feitas as parametrizações gerais são adicionados diversos elementos à janela, nomeadamente: texto (*Label*), botões (*Button*), imagens (*PhotoImage*) e entradas de texto (*Entry*). O módulo externo “Tkinter” permite três formas de distribuição de elementos numa janela, conhecidos como “Pack”, “Grid” e “Place”. O “Pack” distribui os elementos uns a seguir aos outros na vertical ou na horizontal. A forma “Grid” permite distribuir os elementos na forma de grelha, utilizando colunas e linhas. Já a forma “Place” permite a colocação dos elementos em qualquer localização na janela, definida por coordenadas de pixel. Neste projeto foi escolhida a forma “Grid” por permitir uma organização estruturada e controlada dos diversos elementos presentes na janela.

A janela programada no módulo *main* pode ser decomposta em duas partes. Uma primeira parte relativa ao acionamento, controlo e visualização sobre o processo de produção em modo automático e uma segunda parte de controlo dos equipamentos em modo manual. Na secção relativa à configuração da produção em modo automático foi criada uma caixa de entrada de texto que permite definir o número de peças que se pretende produzir. Adicionalmente foram programados três botões, um botão “Start” através do qual se inicia o processo de produção, um botão de “Emergency Stop” que efetua a paragem de todos os equipamentos, nomeadamente robôs e motores, e por fim o botão “Check Connections” para verificar se os equipamentos necessários ao processo de produção se encontram corretamente ligados ao controlador central. Para cada um dos elementos da janela foram atribuídas propriedades escolhidas para fins estéticos e de organização. Na Listagem 4.30 pode ser observado o código referente à configuração da janela de produção em modo automático.

```
automatic_label = tkinter.Label(text="Automatic Mode", font=("Arial", 18, "bold"), bg='white')
automatic_label.grid(column=0, row=0, columnspan=12)

exit_button = tkinter.Button(text="Exit", command=lambda: root.destroy(), padx=3, pady=3, borderwidth=0.5)
exit_button.grid(column=9, row=0)

parts_number_label = tkinter.Label(text="Parts to Produce:", bg='white')
parts_number_label.grid(column=0, row=8, columnspan=6, sticky="E")
parts_number = tkinter.Entry(root, width=5)
parts_number.grid(column=6, row=8, columnspan=6, sticky="W")

start_button = tkinter.Button(text="Start", command=lambda: get_parts_number(), padx=5, pady=5, bg="green",
                              fg='white', borderwidth=0.5)
start_button.grid(column=0, row=10, columnspan=12)

stop_button = tkinter.Button(text="Emergency Stop", command=lambda: automaticproduction.emergency_stop(),
                             padx=5, pady=5, bg="red", borderwidth=0.5)
stop_button.grid(column=0, row=15, columnspan=12)

check_connections = tkinter.Button(text="Check Connections", command=lambda: automaticproduction.check_connections(),
                                   padx=5, pady=5, bg="orange", fg='black', borderwidth=0.5)
check_connections.grid(column=0, row=16, columnspan=12)
```

Listagem 4.30 – Código da configuração da janela de produção em modo automático.

A supervisão do estado da produção foi implementada através da apresentação do número de peças produzidas, o número de peças produzidas com defeitos e as fases pela qual a peça que se encontra a ser produzida já passou. O número de peças produzidas é apresentado através do valor da variável “parts_produced” e, de forma semelhante, o número de peças produzidas com defeito é apresentado pelo valor da variável “defective_parts”. As fases do processo de produção foram divididas em seis pontos chave:

1. Elemento referente ao *backsheet* colocado sobre a mesa de montagem (*Base in Place*);
2. Aplicação de cola sobre o *backsheet* terminada (*Glue Applied*);
3. Elemento referente às células solares colocado sobre o *backsheet* na mesa de montagem (*Glass in Place*);
4. Peça colocada sobre a mesa de transporte (*Movement Ready*);
5. Controlo de qualidade terminado (*Quality Control*);
6. Ejeção da peça para o recetor realizada (*Part Eject*).

A apresentação da fase de produção é efetuada com recurso a uma imagem que é modificada em pontos chave da produção, por forma a refletir as fases de produção pela qual a peça já passou. No arranque da aplicação e início de produção de uma peça, todas as fases se encontram com a coloração cinzenta, indicativa de que ainda não foi realizada. Com o decorrer do processo e passagem pelos pontos chave, as fases concluídas apresentam-se com a coloração verde. Na Figura 4.86 pode ser observada a sequência de atualização da imagem referente às fases de produção.

Figura 4.86 – Sequência de atualização de imagem referente às fases de produção.

Como elemento complementar, foi implementado um botão de ligação a uma folha de cálculo com vários dados relativos ao histórico de produção. Este ficheiro é produzido e atualizado no módulo interno *productionhistory*. Na Listagem 4.31 pode ser observada a programação efetuada para o controlo e visualização do processo de produção e histórico.

```
parts_produced_label = tkinter.Label(text='Parts Produced: ' + str(parts_produced), bg='white')
parts_produced_label.grid(column=0, row=20, columnspan=12)

defective_parts_label = tkinter.Label(text='Defective Parts: ' + str(defective_parts), bg='white')
defective_parts_label.grid(column=0, row=30, columnspan=12)

production_phase_label = tkinter.Label(text='Production Phase', font=("Arial", 12, "bold"), bg='white')
production_phase_label.grid(column=0, row=31, columnspan=12)

phase = tkinter.PhotoImage(file="images/phase_all_off.png")
phase_image = ttk.Label(root, width=509, background="white", image=phase, borderwidth=0)
phase_image.grid(column=0, row=32, columnspan=10)

production_history_button = tkinter.Button(text="Production History", command=lambda: open_production_history(),
                                           padx=5, pady=5, bg="grey", fg='white', borderwidth=0.5)
production_history_button.grid(column=0, row=33, columnspan=12)
```

Listagem 4.31 – Programação do controlo e visualização do processo de produção.

Para a abertura de uma nova janela para o controlo do processo em modo manual foi criado o botão “Enter Manual Control” que executa a função “manual_window” no módulo interno *manualcontrol*. Na Listagem 4.32 pode ser observada a programação realizada para a criação do botão “Enter Manual Control”.

```
manual_label = tkinter.Label(text="Manual Control", font=("Arial", 18, "bold"), bg='white')
manual_label.grid(column=0, row=60, columnspan=12)

manual_control_button = tkinter.Button(text="Enter Manual Control", command=lambda: manualcontrol.manual_window(root),
                                       padx=5, pady=5, bg="lightblue", borderwidth=0.5)
manual_control_button.grid(column=0, row=70, columnspan=12)
```

Listagem 4.32 – Programação do botão “Enter Manual Control”.

Na Figura 4.87 pode ser observada a janela da interface gráfica programada no módulo *main* com a configuração, acionamento, controlo e visualização do processo em modo automático, bem como o botão para entrada no modo manual.

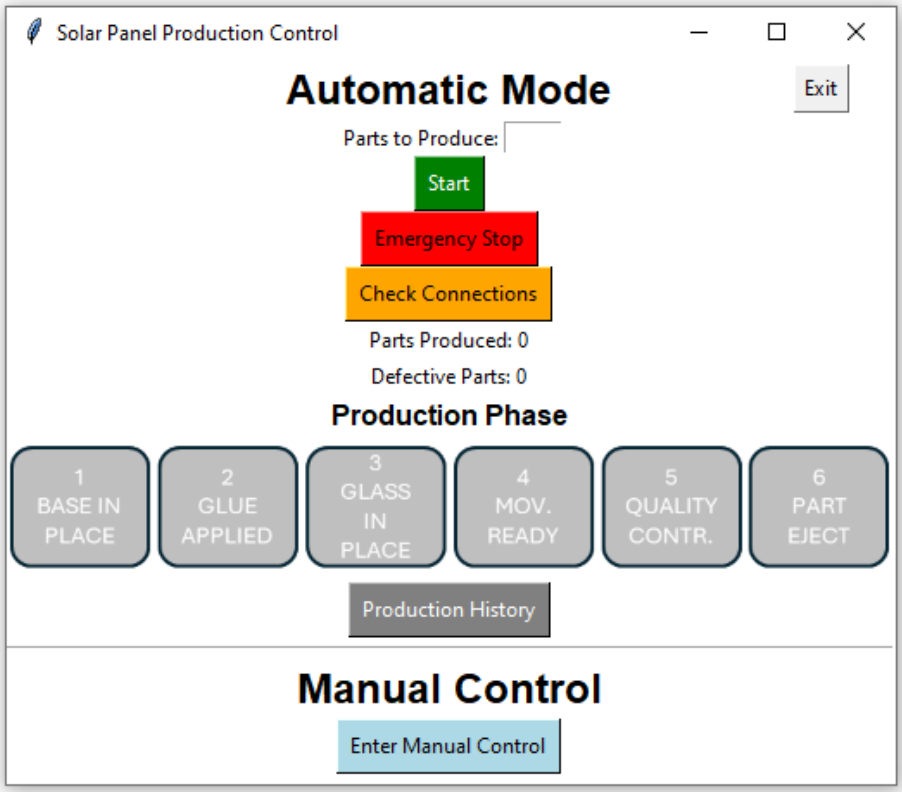

Figura 4.87 – Janela de interface gráfica do módulo *main*.

Módulo manualcontrol

O controlo dos diversos equipamentos em modo manual é realizado numa segunda janela que é acedida através do botão “Enter Manual Control” da janela principal. Na Listagem 4.33 pode ser observada a parametrização da janela de controlo manual.

```
def manual_window(root):
    child_root = tkinter.Toplevel(root)
    child_root.title("Solar Panel Production Manual Control")
    child_root.minsize(width=860, height=500)
    child_root.configure(bg='white')
```

Listagem 4.33 – Parametrização da janela de controlo manual do módulo “Tkinter”.

Os elementos presentes na janela são semelhantes aos utilizados na janela principal, nomeadamente: texto, botões, imagens e entradas de texto. Adicionalmente foi inserido o botão “Emergency Stop” para efetuar a paragem de todos os equipamentos e o botão “Check Connections” para verificar as ligações dos equipamentos ao controlador central.

A estrutura de organização da janela do controlo manual consiste na apresentação de uma imagem referente a cada equipamento, bem como botões para executar as ações

realizadas por cada um no âmbito do projeto. Os botões executam funções programadas nos módulos *dobotcommunication*, *motorcommunication* e *cameravision*. Os Dobot Magician, os motores de passo Nema 17 e D8-MOTOR80 e a câmara podem ser controlados individualmente.

A programação referente a cada equipamento foi efetuada de forma semelhante pelo que, a título exemplificativo, é explicada a implementação do controlo manual do Dobot 1. Na janela do controlo manual é carregada uma imagem referente ao equipamento (ponto 1 da Figura 4.88), para ajudar o utilizador a identificar o equipamento a operar. Por baixo é apresentada uma caixa de entrada de texto para a indicação da porta de comunicação com o equipamento (ponto 2 da Figura 4.88). Por defeito a caixa vem preenchida com um número pré-determinado de porta, no entanto este poderá se alterado manualmente. Finalmente, existem botões para cada ação executada pelo robô, nomeadamente: o envio do robô para a posição de referência (*home position*), movimentação do *backsheet* desde o alimentador 1 até à mesa de montagem (*Base Movement*), movimentação do elemento referente às células solares desde o alimentador 2 até à mesa de montagem (*Glass Movement*) e a movimentação da peça completa desde a mesa de montagem até à mesa de transporte (*Complete Part Movement*) (ponto 3 da Figura 4.88). Na Figura 4.88 pode ser observada a janela da interface gráfica do controlo manual dos equipamentos.

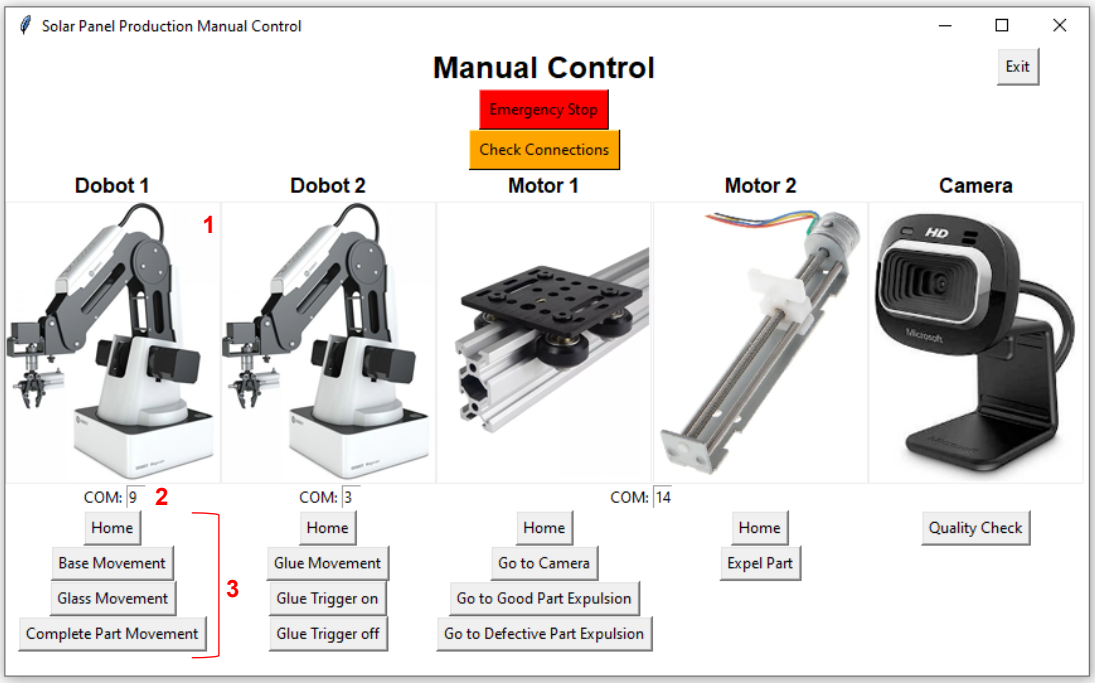

Figura 4.88 – Janela de interface gráfica do módulo *manualcontrol*.

Na Listagem 4.34 pode ser observada a programação efetuada para o controlo manual do Dobot 1.

```
dobot_1_picture = tkinter.Canvas(child_root, width=169, height=222, bg='white', bd=0, highlightthickness=1)
dobot_1_picture.grid(column=0, row=80, columnspan=2)
img_dobot = tkinter.PhotoImage(master=child_root, file="images\dobot.png")
dobot_1_picture.create_image( *args: 84, 111, image=img_dobot)
dobot_1_picture.image = img_dobot
dobot_1_com_label = tkinter.Label(child_root, text="COM:", bg='white')
dobot_1_com_label.grid(column=0, row=90, sticky="E")
dobot_1_com = tkinter.Entry(child_root, width=2)
dobot_1_com.insert( index: 0,  string: '9')
dobot_1_com.grid(column=1, row=90, sticky="W")
dobot_1_home_button = tkinter.Button(child_root, text="Home", command=lambda: dobot_1(dobot_1_com, action='home'),
                                padx=2, pady=2)
dobot_1_home_button.grid(column=0, row=100, columnspan=2)
dobot_1_base_movement = tkinter.Button(child_root, text="Base Movement", command=lambda: dobot_1(dobot_1_com,
                                action='base_part_movement'), padx=2, pady=2)
dobot_1_base_movement.grid(column=0, row=101, columnspan=2)
dobot_1_glass_movement = tkinter.Button(child_root, text="Glass Movement", command=lambda: dobot_1(dobot_1_com,
                                action='glass_part_movement'), padx=2, pady=2)
dobot_1_glass_movement.grid(column=0, row=102, columnspan=2)
dobot_1_complete_part_movement = tkinter.Button(child_root, text="Complete Part Movement",
                                command=lambda: dobot_1(dobot_1_com, action='transport_movement'), padx=2, pady=2)
dobot_1_complete_part_movement.grid(column=0, row=103, columnspan=2)
```

Listagem 4.34 – Programação do controlo manual do Dobot 1.

A execução das ações do robô mencionadas é condicionada pela existência de um valor válido na caixa de texto. Na Listagem 4.35 pode ser observada a programação da verificação do preenchimento da caixa de texto bem como a chamada das funções referentes às ações do equipamento.

```
def dobot_1(dobot_1_com, action):
    com = 'COM' + dobot_1_com.get()
    equipment = 'Dobot 1'
    label_ok = detecterror.check_com_input(com, equipment)
    if label_ok:
        match action:
            case 'home':
                dobotcommunication.dobot_1_home(com)
            case 'base_part_movement':
                dobotcommunication.dobot_1_base_part_movement(com, part_number=1)
            case 'glass_part_movement':
                dobotcommunication.dobot_1_glass_part_movement(com, part_number=1)
            case 'transport_movement':
                dobotcommunication.dobot_1_transport_movement(com)
```

Listagem 4.35 – Programação da verificação de erros e chamada de funções.

No Quadro 4.8 encontram-se enumeradas as funções e respetivos módulos correspondentes aos botões presentes na janela de controlo manual.

Quadro 4.8 – Correspondência de botões e funções executadas no controlo manual.

Equipamento	Botão	Módulo	Função
Dobot 1	*Home*	*dobotcommunication*	"dobot_1_home"
Dobot 1	*Base Movement*	*dobotcommunication*	"dobot_1_base_part_movement"
Dobot 1	*Glass Movement*	*dobotcommunication*	"dobot_1_glass_part_movement"
Dobot 1	*Complete Part Movement*	*dobotcommunication*	"dobot_1_transport_movement"
Dobot 2	*Home*	*dobotcommunication*	"dobot_2_home"
Dobot 2	*Glue Movement*	*dobotcommunication*	"dobot_2_glue_part_movement"
Dobot 2	*Glue Trigger on*	*dobotcommunication*	"glue_trigger_on"
Dobot 2	*Glue Trigger off*	*dobotcommunication*	"glue_trigger_off"
Motor 1	*Home*	*motorcommunication*	"motor_1_home"
Motor 1	*Go to Camera*	*motorcommunication*	"motor_1_movement_to_quality"
Motor 1	*Go to Good Part Expulsion*	*motorcommunication*	"motor_1_good_part_movement"
Motor 1	*Go to Defective Part Expulsion*	*motorcommunication*	"motor_1_defective_part_movement"
Motor 2	*Home*	*motorcommunication*	"motor_2_home"
Motor 2	*Expel Part*	*motorcommunication*	"motor_2_part_expulsion"
Câmara	*Quality Check*	*cameravision*	"quality_control"

Módulo automaticproduction

A automatização do processo de produção foi programada no módulo *automaticproduction*. O módulo é composto essencialmente pela organização da sequência de produção de uma peça, sendo o processo completamente autónomo para a produção do número de peças estipuladas pelo utilizador.

Para a execução do módulo são utilizados os módulos externos "Tkinter", "time" e "datetime". A utilização do módulo "Tkinter" neste âmbito deveu-se à necessidade de atualizar a imagem da fase do processo, o número de peças produzidas e o número das peças com defeito na janela do módulo *main*.

O módulo *automaticproduction* possui cinco variáveis globais que representam: as portas de ligação aos dois robôs Dobot Magician e ao Arduino UNO ("port1", "port2 e "port3") e a taxa de transmissão de informação em bits por segundo dos robôs e do Arduino UNO ("dobot_baudrate" e "motor_baudrate"). Na Listagem 4.36 pode ser observada a programação das variáveis globais do módulo *automaticproduction*.

```
port1 = "COM9"  # COM Port for Dobot 1 (Suction)
port2 = "COM3"  # COM Port for Dobot 2 (Extruder)
port3 = "COM14"  # COM Port for CNC Shield
dobot_baudrate = 115200
motor_baudrate = 50000
```

Listagem 4.36 – Variáveis globais do módulo *automaticproduction*.

O controlo da operação automática de produção encontra-se implementado na função "auto_start". Esta função é chamada quando é pressionado o botão "Start" na janela do módulo *main*. A função possui sete variáveis de entrada:

- "phase_image" – imagem das fases presente na janela do módulo *main*;
- "number_of_parts" – número de peças a ser produzido no lote;
- "root" – objeto da janela do módulo *main*;
- "parts_produced_label" – texto apresentado na janela do módulo *main* referente ao número de peças produzidas;
- "parts_produced" – número de peças produzidas;
- "defective_parts_label" – texto apresentado na janela do módulo *main* referente ao número de peças produzidas com defeito;
- "defective_parts" – número de peças produzidas com defeito.

A sequência de produção em modo automático inicia-se com o envio do Motor 1, seguidamente o Motor 2 e posteriormente os robôs Dobot 1 e 2 para as posições de referência (*home position*). Esta ação é apenas executada na produção da primeira peça do lote, assumindo-se que a partir daí não ocorrem colisões ou cortes de alimentação elétrica. Na Listagem 4.37 pode ser observada a programação efetuada para o início da função "auto_start", nomeadamente das funções de entrada e envio dos equipamentos para as posições de referência.

```
def auto_start(phase_image, number_of_parts, root, parts_produced_label, parts_produced,
               defective_parts_label, defective_parts):
    motorcommunication.motor_1_home(port3)
    motorcommunication.motor_2_home(port3)
    dobotcommunication.dobot_1_and_2_home(port1, port2)
```

Listagem 4.37 – Programação da parte inicial da função "auto_start".

Para a produção dos painéis solares em modo automático, foi criado um ciclo que é repetido conforme o número de peças a produzir. Ao longo do ciclo são recolhidas várias informações necessárias para a criação do registo histórico da produção, nomeadamente a data e as horas de início e fim da produção da peça e o resultado da verificação de qualidade. No Quadro 4.9 apresenta-se a sequência de ações que constituem a produção da peça, com a indicação do equipamento responsável por cada ação e a função que é chamada a executar.

Quadro 4.9 – Sequência de ações no ciclo de produção automática.

Nº Ação	Equipamento	Módulo	Função
1	Dobot 1	*dobotcommunication*	“dobot_1_base_part_movement”
2	Dobot 2	*dobotcommunication*	“dobot_2_glue_part_movement”
3	Dobot 1	*dobotcommunication*	“dobot_1_glass_part_movement”
4	Dobot 1	*dobotcommunication*	“dobot_1_transport_movement”
5	Motor 1	*motorcommunication*	“motor_1_movement_to_quality”
6	Câmara	*cameravision*	“quality_control”
7	Motor 1	*motorcommunication*	“motor_1_good_part_movement” ou “motor_1_defective_part_movement”
8	Motor 2	*motorcommunication*	“motor_2_part_expulsion”
9	Motor 2	*motorcommunication*	"motor_2_home”
10	Motor 1	*motorcommunication*	"motor_1_home”

A programação de todas as ações daquela sequência foi efetuada de forma semelhante. A título exemplificativo será descrita a programação da ação de movimentação do *backsheet* do alimentador 1 até à mesa de montagem. A ação inicia-se pela chamada da função “dobot_1_base_part_movement” do módulo *dobotcommunication*. Após a execução da função e tratando-se de um ponto chave da produção, é atualizada a imagem na janela do módulo *main*, referente às fases de produção efetuadas. A atualização da imagem é efetuada através da indicação do caminho da imagem seguido de uma atualização forçada ao objeto “root”. Na Listagem 4.38 pode ser observado o excerto de programação da função “auto_start” referente à execução da ação de transporte do *backsheet*.

```
print('Starting Dobot 1 Base Movement')
dobotcommunication.dobot_1_base_part_movement(port1, part_number)
phase = tkinter.PhotoImage(file="images/phase_1_on.png")
phase_image.configure(image=phase)
phase_image.image = phase
time.sleep(0.5)
root.update()
```

Listagem 4.38 – Excerto da programação da função “auto_start”.

Foram criadas duas funções adicionais no módulo *automaticproduction*. Embora as ações não sejam específicas do módulo, optou-se por incorporá-las, pois os módulos e variáveis necessários já se encontravam importados e criados no módulo. Foi criada a função “emergency_stop”, que efetua a paragem dos equipamentos e a função “check_connections” que permite a verificação da disponibilidade das portas de ligação aos equipamentos através da função “test_com” do módulo *detecterror*. Na Listagem 4.39 pode ser observado o código das funções “emergency_stop” e “check_connections”.

```
def emergency_stop():
    dobotcommunication.dobot_emergency_stop(port1, port2)
    motorcommunication.motor_emergency_stop(port3)


def check_connections():
    detecterror.test_com(port1, dobot_baudrate)
    detecterror.test_com(port2, dobot_baudrate)
    detecterror.test_com(port3, motor_baudrate)
```

Listagem 4.39 – Programação das funções “emergency_stop” e “check_connections”.

Módulo productionhistory

Com o objetivo de criar um histórico das peças produzidas foi criado o módulo *productionhistory* composto por duas funções. A função “get_part_id” tem por objetivo determinar o índice da peça atual, analisando, para tal, o ficheiro de histórico e calculando o número de peças já inseridas. A função “write_data_line” efetua a entrada dos dados da presente peça no ficheiro de histórico. Para a leitura e escrita do ficheiro foi utilizado o módulo externo “csv”. O módulo *productionhistory* possui duas variáveis globais, a variável “filename” que indica o nome do ficheiro utilizado para o controlo do histórico e a variável “dir_path” que indica o caminho da pasta onde se encontram armazenadas as imagens captadas durante o controlo de qualidade.

Função “get_part_id”

A função “get_part_id” inicia-se com a leitura do ficheiro de histórico e guarda o seu conteúdo no objeto “reader_file”. Seguidamente, é percorrida cada linha de entrada no objeto e é incrementada a variável “i” por cada linha analisada. Por fim é devolvido o valor da variável “i” que contém o número sequencial da próxima entrada. Na Listagem 4.40 pode ser observada a programação efetuada para a função “get_part_id”.

```
def get_part_id():
    i = 0
    with open(filename, 'r', newline='') as csvfile:
        reader_file = csv.reader(csvfile)
        for _ in reader_file:
            i += 1
        return i
```

Listagem 4.40 – Programação da função “get_part_id”.

Função “write_data_line”

A escrita de uma nova entrada no ficheiro de histórico de produção é realizada pela função “write_data_line”. A função possui como variáveis de entrada “start_time” indicativa da data e hora de início de produção da peça, “part_number”, o número, dentro do lote, da peça que está a ser produzida, “end_time” indicativa da data e hora

de fim de produção da peça e “part_quality” com o resultado da verificação de qualidade da peça.

A função inicia-se com a abertura do ficheiro de histórico no modo de adição de linha e guarda o seu conteúdo no objeto “writer”. Seguidamente é obtido o número sequencial da peça que foi produzida, com recurso à função “get_part_id” e armazenada na variável “count”. É obtido o caminho completo da imagem captada pela câmara na verificação da qualidade, sendo o caminho composto pela variável “dir_path”, o número sequencial da peça produzida e a extensão “.png” da imagem. O caminho da imagem é guardado na variável “image”. Por fim a função efetua a inserção de uma nova linha no ficheiro do histórico, com o preenchimento dos campos com as variáveis “count”, “part_number”, “start_time”, “end_time”, “part_quality” e “image” descritos anteriormente.

Na Listagem 4.41 pode ser observada a programação efetuada para a função “write_data_line”.

```
def write_data_line(start_time, part_number, end_time, part_quality):
    with open(filename, 'a', newline='') as csvfile:
        writer = csv.writer(csvfile, delimiter=";")
        count = get_part_id()
        image = dir_path+str(count)+'.png'
        writer.writerow((count, part_number, start_time, end_time, part_quality, image))
```

Listagem 4.41 – Programação da função “write_data_line”.

Módulo detecterror

A deteção atempada de erros evita que ocorram interrupções na fase de produção e pode proporcionar orientações claras ao utilizador sobre a origem do erro para que a sua resolução seja mais expedita. No programa elaborado para o processo de produção dos painéis solares foram identificados dois erros frequentes que impediam um correto funcionamento da aplicação. O primeiro erro identificado foi a falta de preenchimento dos campos de entrada de texto para a indicação do número da porta de ligação dos equipamentos. A falta desta indicação origina um erro no estabelecimento da ligação entre o controlador central e o equipamento, originando uma interrupção na execução do programa. Para a deteção deste erro foi criada a função “check_com_input”. O segundo erro detetado encontra-se relacionado com o primeiro, contudo já é na fase de comunicação entre o controlador central e o equipamento. Apesar de o número da porta de ligação se encontrar correta, poderá ocorrer um erro na comunicação pelo facto da porta estar ocupada, como por exemplo, estar a ser utilizada por outra aplicação. Para a verificação deste erro de comunicação foi criada a função “test_com”. A verificação deste erro não ocorre de forma automática, devendo ser acionada pelo utilizador no início da aplicação.

Função "check_com_input"

A função "check_com_input" tem como variáveis de entrada o número da porta de ligação ("com"), bem como o nome do equipamento referente à porta que está a ser verificada ("equipment"). A função inicia com a determinação da variável "label_ok" com o valor verdadeiro. Seguidamente verifica-se se a variável de entrada "com" possui apenas o texto "COM". No caso verdadeiro, isso é indicativo de que falta informação; nesse caso é despoletado um erro indicando que é necessária a inserção do número da porta de ligação do referido equipamento. Caso a variável de entrada contenha texto diferente de "COM", isso é indicativo de que foi preenchida a entrada de texto com o número da porta de ligação e então é devolvida a resposta da variável "label_ok" com o valor verdadeiro. Na Listagem 4.42 pode ser observada a programação da função "check_com_input".

```
def check_com_input(com, equipment):
    label_ok = True
    if com == 'COM':
        tkinter.messagebox.showerror( title: 'Error', 'Please input a COM port for '+equipment)
    else:
        return label_ok
```

Listagem 4.42 – Programação da função "check_com_input".

Função "test_com"

A verificação da disponibilidade da porta de ligação de determinado equipamento é efetuada na função "test_com". A função possui como variáveis de entrada o número da porta de ligação ("com") e a taxa de transmissão de informação em bits por segundo ("baudrate"). A função inicia-se com a tentativa de estabelecimento de comunicação com a porta indicada pela variável "com" e a respetiva taxa de transmissão de informação. Caso a comunicação não seja estabelecida, é devolvido um erro a indicar que ocorreu um problema de comunicação e que a porta de ligação indicada se encontra inacessível. Caso a comunicação seja bem estabelecida é devolvida uma informação indicativa de que a ligação se encontra disponível. Na Listagem 4.43 pode ser observada a programação da função "test_com".

```
def test_com(com, baudrate):
    try:
        serial.Serial(port=com, baudrate=baudrate, timeout=1)
    except SerialException:
        tkinter.messagebox.showerror( title: 'Error', 'Connection Error. '+com+ ' port access is denied.')
    else:
        tkinter.messagebox.showinfo( title: 'Information', com+ ' available.')
```

Listagem 4.43 – Programação da função "test_com".

5 Análise de resultados obtidos

5.1 Introdução e objetivos da análise de resultados

A análise de resultados constitui uma etapa essencial na realização de um projeto. Após a execução das diversas etapas do projeto, nomeadamente a preparação, montagem e testes, é fundamental a análise dos resultados para a avaliação do cumprimento dos objetivos propostos assim como a identificação de pontos de melhoria. A adoção deste procedimento permite a implementação da melhoria contínua no processo, através do ciclo *Plan, Do, Check, Act* (PDCA). Com as etapas do Planeamento (*Plan*) e Execução (*Do*) concluídas, surge a necessidade de realizar a Verificação (*Check).* Esta etapa consiste em analisar o processo e verificar os seus resultados quando comparados com métricas de desempenho estabelecidos previamente. Perante as conclusões que forem retiradas é possível passar para a etapa de Atuar (*Act)*, através da implementação das oportunidades de melhoria que forem identificadas. Tratando-se de um ciclo, a melhoria contínua não termina, pois é uma técnica iterativa que procura a otimização constante dos processos.

Neste capítulo é realizada uma análise detalhada dos resultados obtidos a partir de testes do processo de produção proposto. O objetivo principal desta análise consiste em avaliar o resultado da montagem dos diversos equipamentos do sistema, o desempenho do sistema, através da análise das métricas de duração da execução do processo, e a precisão do teste de qualidade implementado. Através desta análise são identificadas oportunidades de melhoria.

5.2 Metodologia de análise e recolha de dados

A metodologia da análise de resultados foi repartida em duas partes, uma relativa à análise da montagem da área de trabalho, que consiste numa comparação entre o modelo CAD que foi elaborado e o protótipo do processo. A segunda parte é relativa à execução do processo e inclui a avaliação do desempenho dos movimentos executados por cada equipamento e a avaliação da execução do programa de controlo e supervisão,

através da análise dos dados apresentados ao utilizador durante a execução. Adicionalmente serão efetuadas análises quantitativas da duração da montagem de cada painel solar e da precisão do processo de verificação de qualidade.
A recolha de dados da execução do processo de produção foi efetuada a partir de 5 lotes de 10 peças, totalizando 50 ciclos de produção realizadas em modo automático. Cada execução consiste na montagem completa de um painel solar e a sua ejeção, mediante o resultado da verificação de qualidade. Os dados foram obtidos a partir do ficheiro de histórico de produção gerado automaticamente.

5.3 Resultados

5.3.1 Área de trabalho

Durante a montagem da área de trabalho no laboratório foi utilizado, como base, o modelo CAD previamente elaborado. No decorrer de testes de execução do processo foi identificada a necessidade de algumas alterações. As modificações principais envolveram a localização dos robôs Dobot Magician, para permitir a movimentação das juntas dos robôs dentro das suas áreas de trabalho bem como melhorar a fluidez de movimentos de transporte das peças. Uma correção adicional incidiu sobre o suporte implementado para a câmara. Inicialmente foi considerado um suporte em forma de “U”, contudo, para facilitar a movimentação da cablagem do motor de passo e sensores instalados na mesa de transporte, decidiu-se manter a zona de deslocação desobstruída, impedindo desta forma interferências no deslocamento. Neste sentido foi instalado um suporte para a câmara em forma de “L” invertido. Os recetores das peças completas não foram rigorosamente implementados no modelo CAD dado não se tratar de elementos determinantes na gestão do espaço. A sua implementação foi executada na fase final da montagem da área de trabalho para garantir a escolha da localização mais favorável. Na Figura 5.89 pode ser observada uma comparação entre a área de trabalho projetada no modelo CAD e a do protótipo real.

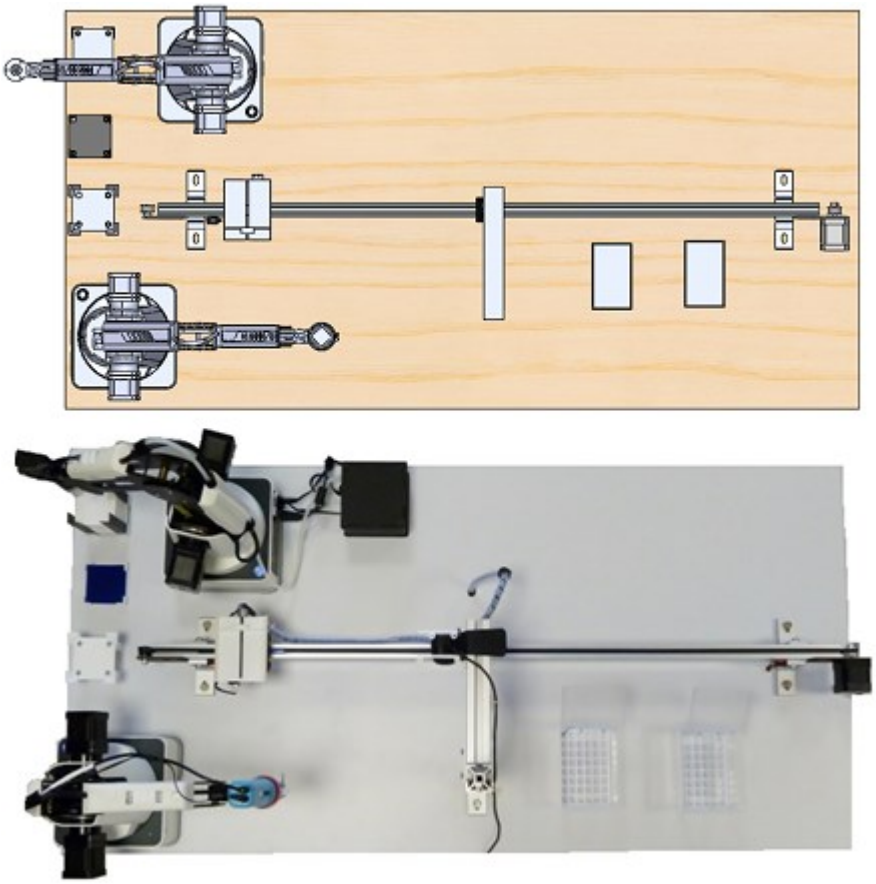

Figura 5.89 – Área de trabalho do processo: modelo CAD (superior); protótipo real (inferior).

5.3.2 Execução de testes do protótipo

A execução de 50 ciclos de produção permitiu recolher dados suficientes para a análise das ações executadas pelos equipamentos e das informações apresentadas ao utilizador e para verificar o correto funcionamento do processo em modo automático.

No início de produção de cada lote, os robôs Dobot Magician e os motores de passo, realizaram corretamente a movimentação até às suas posições de referência. A movimentação da primeira peça para a montagem do painel solar, consistindo na recolha do *backsheet* do alimentador 1 até à mesa de montagem, ocorreu de forma prevista. Na Figura 5.90 é possível observar o início do movimento, com a recolha através de sucção, realizada pelo Dobot 1, do *backsheet* do alimentador 1.

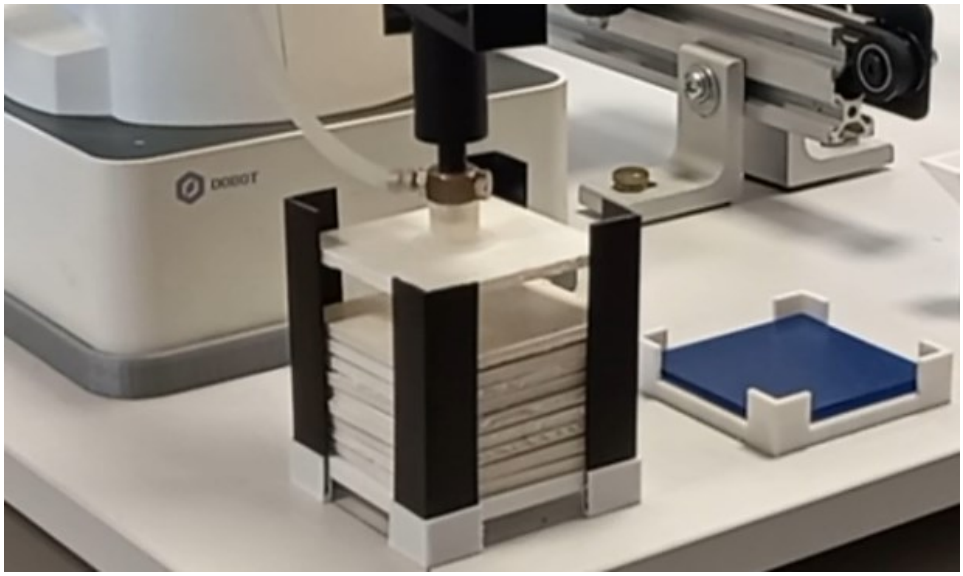

Figura 5.90 – Movimento do *backsheet* do alimentador 1 para a mesa de montagem.

A segunda etapa do processo de montagem consistiu na aplicação de cola sobre o *backsheet*, realizada pelo Dobot 2, com o elemento terminal I-Extruder. A quantidade de cola e a sua localização de aplicação revelaram-se apropriados para garantir uma boa ligação entre as duas peças constituintes do painel solar. Na Figura 5.91 pode ser observada a aplicação de cola sobre o *backsheet*.

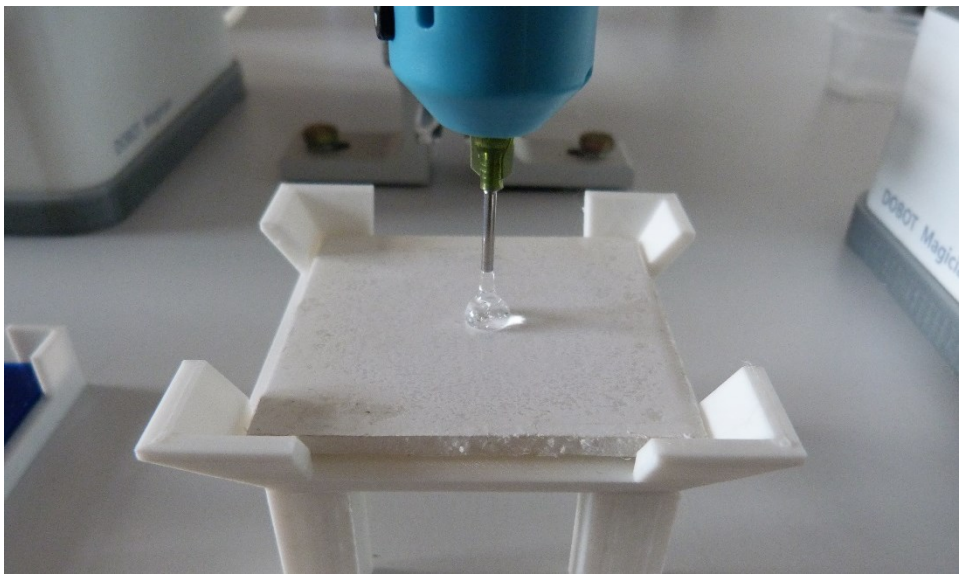

Figura 5.91 – Aplicação de cola sobre o *backsheet*.

Posteriormente à aplicação da cola, o Dobot 1 desloca a peça referente às células solares do alimentador 2 até à mesa de montagem. Quando a peça é posicionada sobre o *backsheet*, a cola aplicada anteriormente espalha-se de forma uniforme, ligando os dois materiais. Verificou-se que o posicionamento da peça, bem como o tempo de aplicação de pressão, foram adequadamente configurados o que permitiu uma adequada união entre as peças, sem a ocorrência de separação nos movimentos

seguintes. Na Figura 5.92 pode ser observado o momento da colagem da peça referente às células solares e o *backsheet* na mesa de montagem.

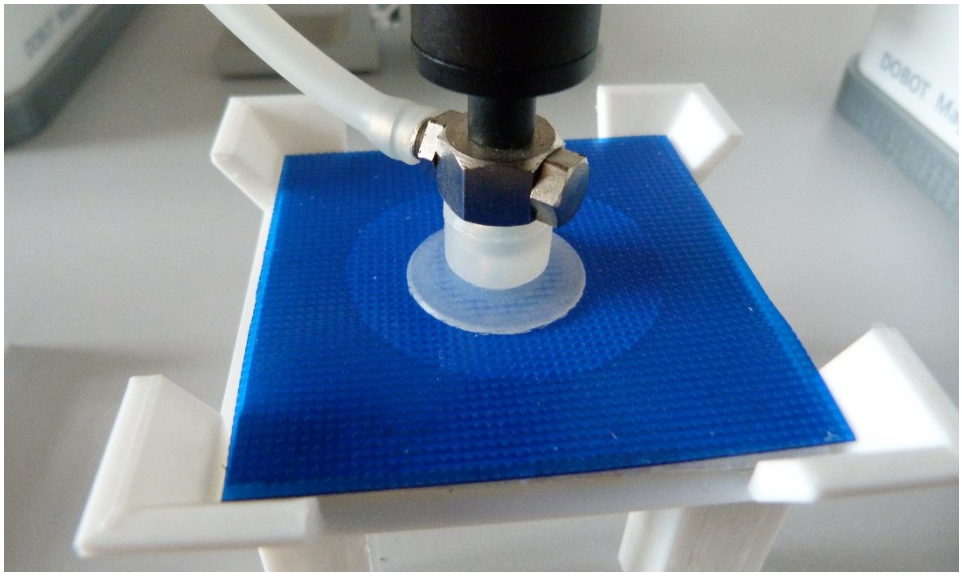

Figura 5.92 – Colagem das células solares e *backsheet* na mesa de montagem.

A colocação da peça completa sobre a mesa de transporte ocorreu de forma prevista, tendo a peça sido colocada sempre no centro da mesa de transporte (Figura 5.93).

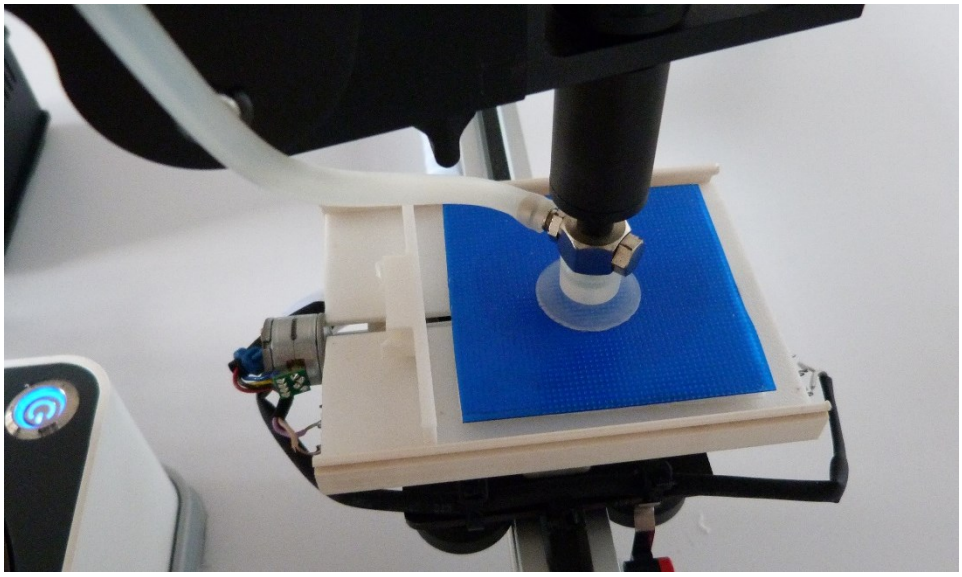

Figura 5.93 – Colocação do painel solar sobre a mesa de transporte.

O posicionamento da mesa de transporte (Figura 5.94) para captura de imagem pela câmara revelou-se adequado, com as imagens a cobrirem a totalidade do painel solar.

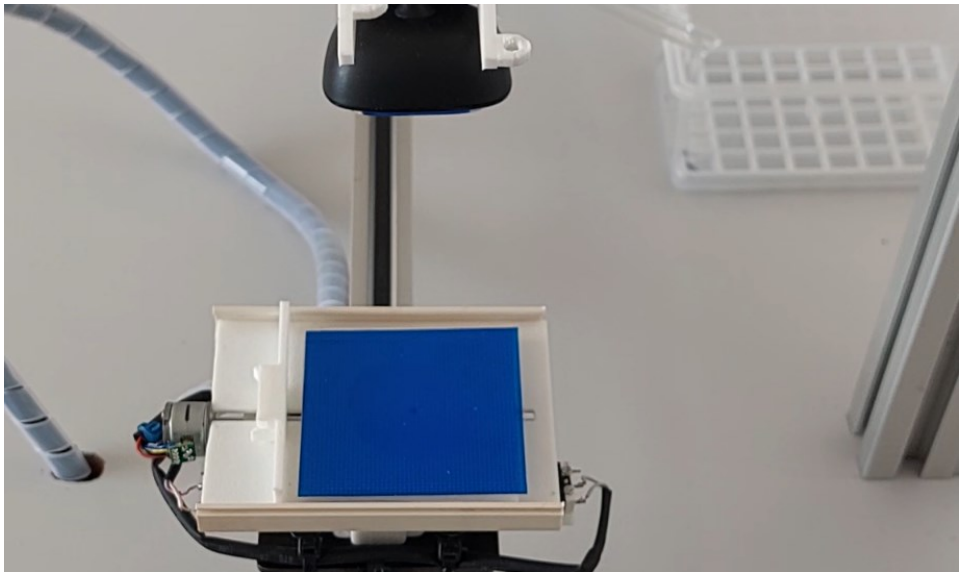

Figura 5.94 – Posicionamento da mesa de transporte para etapa de verificação de qualidade.

Na Figura 5.95 encontra-se uma amostra das imagens captadas para verificação de qualidade de quatro peças painéis solares.

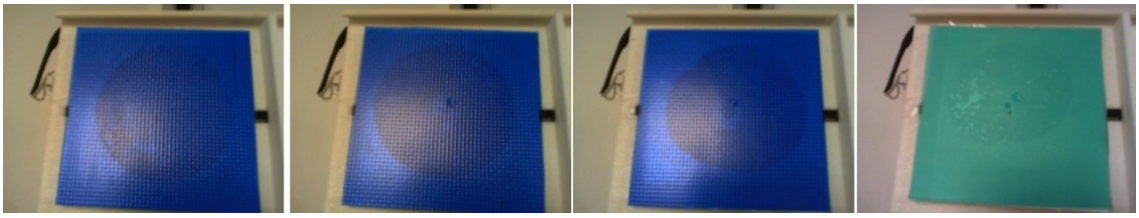

Figura 5.95 – Amostra de imagens captadas para verificação de qualidade.

A última etapa do processo é a ejeção do painel solar para um recetor de peças sendo este escolhido conforme o resultado da verificação de qualidade. O movimento de ejeção da peça completa apresentou nalguns ciclos pequenos problemas, nomeadamente no movimento do painel solar ao longo da rampa de acesso ao recetor, que por vezes não foi realizado na totalidade. Dois dos motivos para estes problemas foram a baixa inclinação da rampa e a sua rugosidade. O movimento de ejeção do painel solar da mesa de transporte para o recetor de peças encontra-se ilustrado na Figura 5.96.

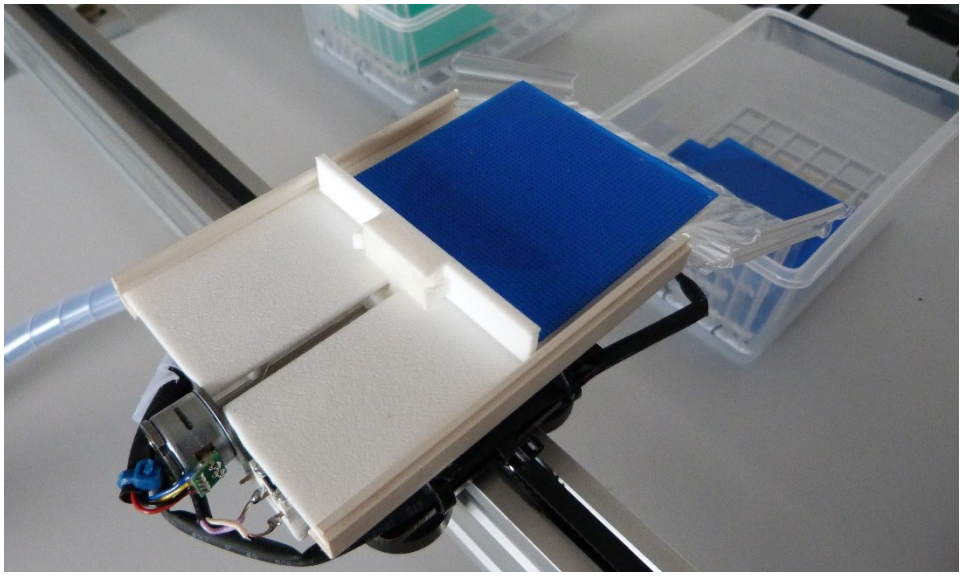

Figura 5.96 – Movimento de ejeção do painel solar da mesa de transporte.

Durante a execução dos testes, a interface gráfica do programa de controlo foi apresentando de forma correta as informações do estado do processo de produção, em coerência com o ocorrido na prática. Na Figura 5.97 pode ser observado o exemplo de um momento da produção onde sete peças já tinham sido produzidas e onde uma nova peça já se encontrava a ser produzia, estando esta na etapa de verificação de qualidade.

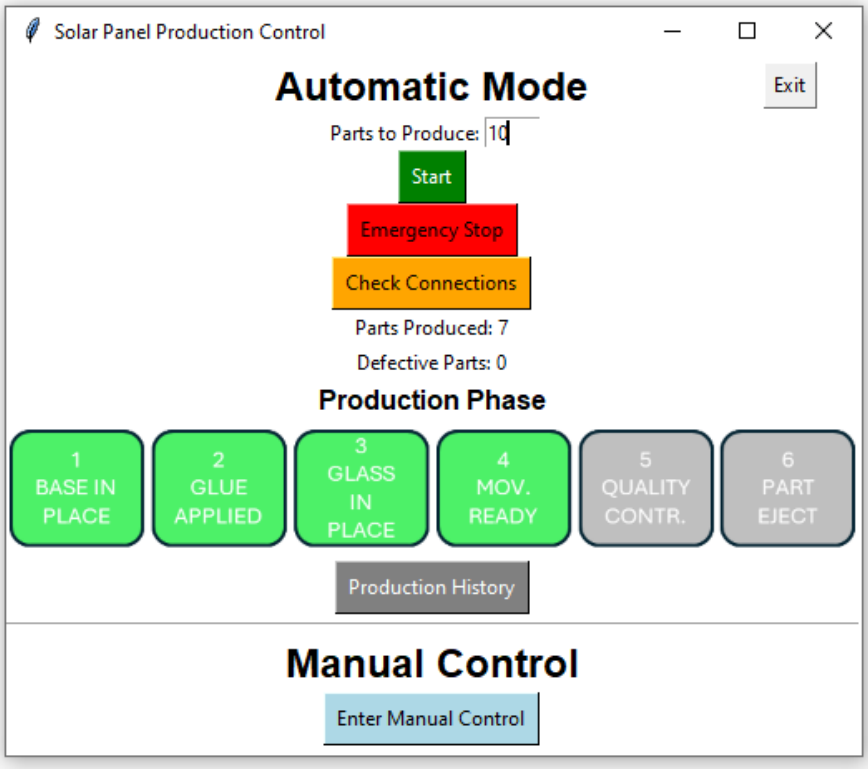

Figura 5.97 – Interface gráfica do estado de produção de lote em progresso.

Na Figura 5.98 pode ser observada a interface gráfica do programa de controlo no final de produção de um dos lotes simulados. O lote solicitado era constituído por dez painéis solares, sendo que três apresentavam defeito.

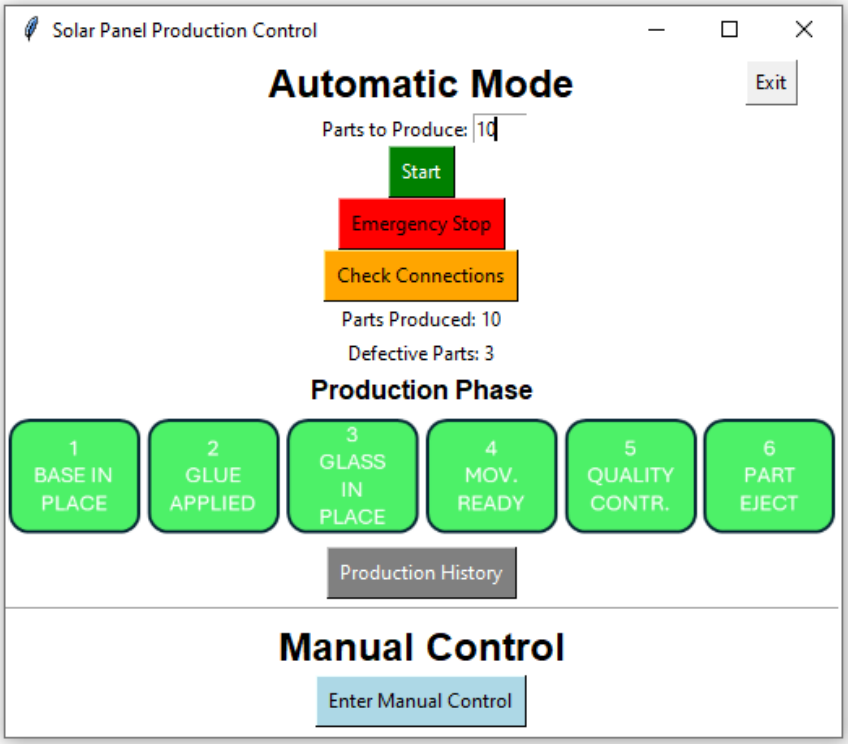

Figura 5.98 – Interface gráfica do estado de produção de lote completo.

Após a execução dos testes foi consultado o ficheiro de histórico de produção. Verificou-se que, para cada painel solar executado, todas as informações foram corretamente preenchidas, tal como pode ser observado na Figura 5.99, tratando-se esta de um excerto do ficheiro de histórico de produção.

part_id	number_in_batch	start_time	end_time	quality_check	image
62	1	20/07/2024 08:52	20/07/2024 08:54	TRUE	file:///C:/solarproduction/62.png
63	2	20/07/2024 08:54	20/07/2024 08:56	TRUE	file:///C:/solarproduction/63.png
64	3	20/07/2024 08:56	20/07/2024 08:58	FALSE	file:///C:/solarproduction/64.png
65	4	20/07/2024 08:58	20/07/2024 08:59	TRUE	file:///C:/solarproduction/65.png
66	5	20/07/2024 08:59	20/07/2024 09:01	FALSE	file:///C:/solarproduction/66.png
67	6	20/07/2024 09:01	20/07/2024 09:03	TRUE	file:///C:/solarproduction/67.png
68	7	20/07/2024 09:03	20/07/2024 09:05	TRUE	file:///C:/solarproduction/68.png
69	8	20/07/2024 09:05	20/07/2024 09:07	FALSE	file:///C:/solarproduction/69.png
70	9	20/07/2024 09:07	20/07/2024 09:10	FALSE	file:///C:/solarproduction/70.png
71	10	20/07/2024 09:10	20/07/2024 09:12	FALSE	file:///C:/solarproduction/71.png

Figura 5.99 – Excerto do ficheiro de histórico de produção.

Com recurso aos dados recolhidos no histórico de produção, foi possível efetuar uma análise estatística de alguns parâmetros do processo de produção. A primeira métrica analisada foi o tempo de produção de cada painel solar, calculado pela diferença entre as horas de fim e de início de produção. Os testes mostraram que o tempo médio de produção por unidade foi de 1 minuto e 54 segundos, com um desvio padrão de 6 segundos. Foi possível concluir que os painéis solares com um maior tempo de produção são referentes às peças com defeito. Este aumento deveu-se ao facto de o recetor das peças com defeito se encontrar mais distante e, consequentemente, o movimento da mesa de transporte demorar mais tempo, em ambos os sentidos. Na

Figura 5.100 mostra-se o gráfico do tempo de produção por cada painel solar onde se observa claramente duas tendências distintas para os tempos de produção.

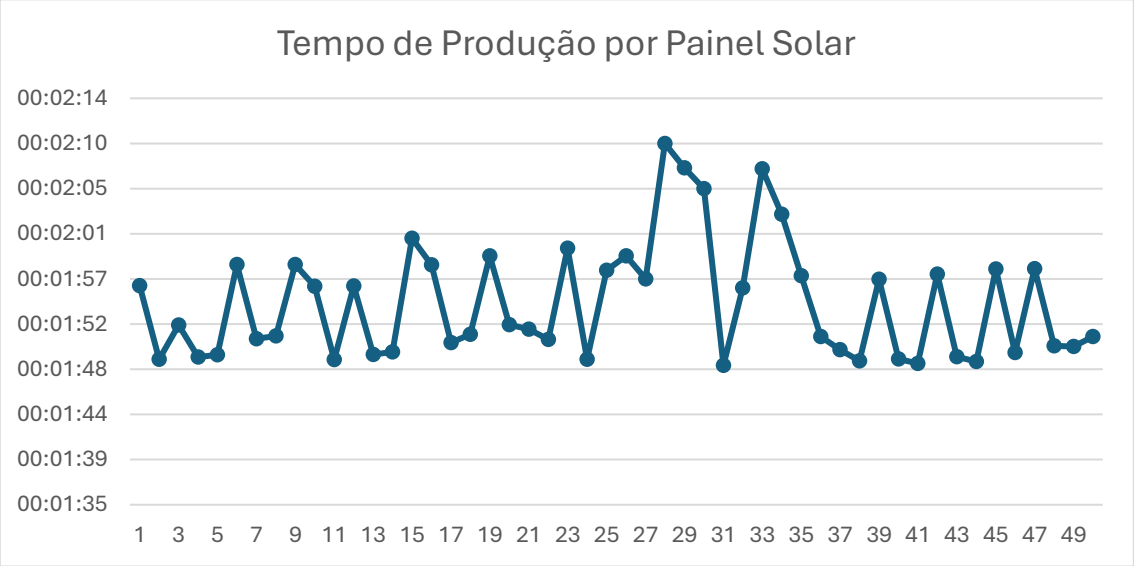

Figura 5.100 – Gráfico do tempo de produção por painel solar.

A segunda métrica analisada foi a precisão do controlo de qualidade. Previamente à execução de cada lote do teste, foram intencionalmente inseridas peças com defeito. O resultado da verificação de qualidade foi confirmado durante a execução da produção e posteriormente confirmado através dos dados guardados no ficheiro de histórico de produção. Verificou-se uma taxa de sucesso de 100% na identificação da conformidade e inconformidade das peças, tendo sido contabilizadas 32 peças sem defeito e 18 com defeito.

5.4 Propostas de Melhoria

Os testes realizados permitiram a identificação de algumas melhorias a implementar de modo a tornar o processo mais eficiente e reduzir a probabilidade de ocorrência de falhas que interrompam a produção. Seguidamente serão listados os pontos identificados como possíveis melhorias:

- Substituição das rampas de acesso aos recetores de peças. As rampas deverão possuir uma maior inclinação e apresentar a mínima rugosidade possível para evitar o atrito na deslocação do painel solar;
- Deslocação do sensor de fim de curso do motor de passo D8-MOTOR80, que deteta o final do percurso de ejeção do painel solar, para uma posição mais adiantada. O reposicionamento do sensor não irá interferir com a ejeção da peça, contudo irá permitir a poupança de tempo;

- Alteração da distribuição da cola sobre o *backsheet* através da modificação da trajetória realizada pelo Dobot 2 para um movimento circular. A alteração da forma de distribuição irá permitir uma distribuição mais uniforme da cola.
- Implementação de um sensor de medição da quantidade de cola no I-Extruder, e apresentar o valor na interface gráfica, incluindo uma informação de alerta sempre que for detetado que a quantidade de cola não é suficiente para concluir o lote em produção.
- Alteração do tipo de defeito detetado na etapa de verificação de qualidade para um acontecimento mais frequente na produção de um painel solar, em contexto real, como por exemplo a existência de riscos no vidro.

6 Conclusões e trabalho futuro

O projeto realizado consistiu no aumento das capacidades do laboratório de robótica do DEM com vista a proporcionar novas ferramentas de aprendizagem a alunos do Mestrado em Engenharia Mecânica, nas temáticas da robótica, automação e programação.

Na fase inicial de realização do projeto foi efetuado um resumo do estado da arte da robótica geral, educacional e da sua programação. Este estudo permitiu um enquadramento teórico do tema e a verificação das tendências nas estratégias adotadas em projetos semelhantes. Seguidamente foi desenvolvido o processo automático de montagem de painéis à escala reduzida. O projeto abrangeu as fases de planeamento, desenho e fabricação de equipamentos, montagem dos mesmos, programação do processo e testes do protótipo produzido composto pela produção de 50 painéis solares para a recolha de dados e verificação do desempenho do processo.

O desenvolvimento do projeto consistiu na integração de dois robôs Dobot Magician, com duas ferramentas diferentes no elemento terminal: uma ventosa atuada por uma bomba de vácuo e um I-Extruder, um equipamento que não pertence à gama disponibilizada pelo fabricante do robô e que requereu a fabricação de um adaptador para fazer o seu acoplamento. Foram integrados dois motores de passo, comandados por um microprocessador Arduino UNO. Adicionalmente foi implementada uma câmara para aferição da qualidade da produção através da aplicação de técnicas de visão computacional.

Os objetivos propostos no início do projeto foram amplamente alcançados com a implementação de um processo de produção automático, integrando diversos equipamentos disponibilizados no laboratório de robótica, fabricação de novas peças e o desenvolvimento de uma aplicação de controlo e supervisão com recurso à linguagem de programação Python.

Na análise de resultados dos testes foram registadas algumas métricas da produção, nomeadamente a duração média de 1 minuto e 54 segundos por cada painel solar e uma precisão de 100% no sistema de aferição da qualidade recorrendo à visão computacional. O mecanismo de deteção de peças conformes e com defeito teve como

objetivo implementar um modelo simples de visão computacional, as peças foram artificialmente implementadas e de fácil deteção, num contexto real a identificação de defeitos, como por exemplo riscos no vidro e módulos solares defeituosos, é mais complexo e requer o desenvolvimento de algoritmos mais sofisticados. Adicionalmente foi realizada uma análise qualitativa do processo de produção, para verificação da conformidade de todas as etapas e identificação de possíveis oportunidades de melhoria.

Durante a realização do projeto surgiram alguns desafios, nomeadamente no estabelecimento de comunicação e na programação dos robôs Dobot Magician. De facto, a documentação oficial não refere a possibilidade de utilização de mais de um robô em simultâneo comandados pelo mesmo computador. Esta dificuldade foi contornada abrindo e fechando a comunicação sempre que era necessário enviar instruções a um dos robôs. Outro desafio encontrado foi a implementação dos sensores de fim de curso: devido à proximidade das cablagens dos sensores e dos motores, ocorriam indicações erráticas de ativação dos sensores. O desafio foi ultrapassado através da adoção de soluções construtivas na área de trabalho, nomeadamente na disposição das cablagens através da individualização e isolamento dos cabos sensores de fim de curso e de alimentação dos motores de passo.

Face às especificidades do projeto existiram algumas limitações, maioritariamente nas ligações elétricas. Devido à reduzida dimensão dos terminais dos sensores de fim de curso e à falta de equipamento apropriado para soldadura, os terminais encontravam-se expostos a vibrações suscetíveis de provocar a quebra da solda. Para a mitigação desta limitação e minimização das vibrações, as cablagens dos terminais foram fixas em pontos estratégicos, nomeadamente ao V-Slot Gantry e à base.

Tratando-se de um projeto de robótica educacional, as possibilidades de expansão e melhoria são inúmeras em função da criatividade do autor. Como trabalho futuro do presente projeto, foram identificados alguns pontos, nomeadamente:

- Implementação das propostas de melhorias enumeradas na análise de resultados dos testes do protótipo. A implementação das melhorias irá reduzir a probabilidade de ocorrência de erros na execução da produção e melhorar a eficiência do processo;
- Melhoria nas soldaduras dos terminais dos sensores de fim de curso dos motores de passo;
- Expansão das capacidades de controlo dos robôs Dobot Magician no modo manual, permitindo, por exemplo, a movimentação das juntas conforme as solicitações do utilizador, nos diversos eixos bem como a obtenção das coordenadas do elemento terminal a todo o momento. Esta funcionalidade irá

reduzir a necessidade de utilização de aplicações complementares, como por exemplo o Dobot Studio, para a obtenção de dados relativos aos robôs;

- Integração de mais equipamentos disponíveis no laboratório de robótica no processo de produção, nomeadamente o armazém automático rotativo e o robô manipulador Scorbot ER IX. A incorporação de novos equipamentos permite aumentar as capacidades do processo, possibilitando, por exemplo, uma maior disponibilização de materiais para a produção ou a deslocação da peça fabricada para outro processo, como por exemplo o embalamento;
- Conversão da aplicação elaborada para o ambiente *web*, recorrendo a módulos de *Frameworks Web* para Python, como por exemplo *Django*. A implementação do controlo num ambiente *web* permitirá o comando remoto do processo de produção.

O autor considera que o projeto desenvolvido no âmbito do Trabalho Final de Mestrado, trouxe ao laboratório de robótica do DEM novas valências para a transmissão de conhecimentos na área da robótica, programação, eletrónica e inteligência artificial, podendo servir de base para futuros projetos. Sendo as temáticas abordadas de espectável relevância no futuro, um laboratório mais bem equipado irá permitir que professores transmitam a futuros alunos conhecimentos mais aprofundados e a realização de ensaios práticos que contribuem para uma melhor compreensão dos conceitos teóricos, equipando o mercado de trabalho com especialistas mais preparados para a resolução de desafios.

Referências bibliográficas

Alimisis, D. (2009). *Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods*. School of Pedagogical and Technological Education (ASPETE).

Allegro MicroSystems. (2022, Abril 5). *A4988 DMOS Microstepping Driver with Translator and Overcurrent Protection Datasheet*. 4988-DS, Rev. 8. Obtido 7 de Setembro de 2024, de https://www.allegromicro.com/-/media/files/datasheets/a4988-datasheet.pdf

Arduino. (2024a). *Arduino Hardware*. Obtido 15 de Março de 2024, de https://www.arduino.cc/en/hardware

Arduino. (2024b). *Arduino Uno Rev3*. Obtido 4 de Março de 2024, de https://store.arduino.cc/products/arduino-uno-rev3

Arduino. (2024c, Março 14). *Arduino Robot*. Obtido 26 de Março de 2024, de https://docs.arduino.cc/retired/other/arduino-robot/

Asimov, I. (1942). *Runaround*. Street & Smith.

Atman Uslu, N., Yavuz, G. Ö., & Koçak Usluel, Y. (2023). A systematic review study on educational robotics and robots. *Interactive Learning Environments*, *31*(9), 5874–5898. https://doi.org/10.1080/10494820.2021.2023890

Ballard, L., Sabanovic, S., Kaur, J., & Milojevic, S. (2012). George Charles Devol, Jr. [History]. *IEEE Robotics & Automation Magazine*, *19*(3), 114–119. https://doi.org/10.1109/MRA.2012.2206672

Barak, M., & Assal, M. (2018). Robotics and STEM learning: students' achievements in assignments according to the P3 Task Taxonomy—practice, problem solving, and projects. *International Journal of Technology and Design Education*, *28*(1), 121–144. https://doi.org/10.1007/s10798-016-9385-9

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, *58*(3), 978–988. https://doi.org/10.1016/j.compedu.2011.10.006

Biba, J. (2022, Setembro 20). *Top 8 Robotic Programming Languages*. Built in. Obtido 10 de Março de 2024, de https://builtin.com/robotics/robotic-programming-language

Bill, M., Müller, C., Kraus, W., & Bieller, S. (2023). *World Robotics 2023*.

Boesl, D. B. O. (2016). A Look into the Crystal Ball: Predicting the Future of Robotics [Industrial Activities]. *IEEE Robotics & Automation Magazine*, *23*(4), 10–19. https://doi.org/10.1109/MRA.2016.2614374

Caetano, N. S., Mata, T. M., Martins, A. A., & Felgueiras, M. C. (2017). New Trends in Energy Production and Utilization. *Energy Procedia*, *107*, 7–14. https://doi.org/10.1016/j.egypro.2016.12.122

Çakır, N. K., Yurdakul, S., & Çetin, G. (2022). Arduino-assisted Robotic And Coding Applications In Science Teaching: Bulb Brightness. Em *JIBA) / Araştırma Temelli Etkinlik Dergisi (ATED)* (Vol. 12, Número 2). Obtido de https://orcid.org/0000-0003-3186-2781

Carter, B. (2018). The Horn Book Magazine. *Vol. 94 Issue 4*, 130–131.

Chebotareva, E., & Gavrilova, L. (2019). Educational Mobile Robotics Project «ROS-Controlled Balancing Robot» Based on Arduino and Raspberry Pi. *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, 209–214. https://doi.org/10.1109/DeSE.2019.00047

Chen, D., Zeng, Z., Guan, Y., Zhu, H., & Zhang, T. (2020). SCARA Robots Developed with Modular Method. *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, 454–459. https://doi.org/10.1109/ICMA49215.2020.9233716

Chu, T. S., Culaba, A. B., & Jose, J. A. C. (2022). Robotics in the Fifth Industrial Revolution. *2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 1–6. https://doi.org/10.1109/HNICEM57413.2022.10109473

Craig, J. J. (2005). *Introduction to Robotics Mechanics and Control* (3ª ed.). Pearson Education Inc.

Delta Electronics Inc. (2024). *Robô Articulado*. Obtido 13 de Abril de 2024, de https://delta-electronics.com.br/produtos/robo-articulado/

Dobot. (2024a). *About Dobot Robotics*. Obtido 2 de Abril de 2024, de https://www.dobot-robots.com/about/about-dobot

Dobot. (2024b). *All-in-One Robot Education - Dobot Product Catalog*. Obtido 2 de Abril de 2024, de https://www.alcom.no/wp-content/uploads/2019/11/DOBOT-Product-Catalog-Educational-Version.pdf

Dobot Magician User Guide (V2.0). (2020). Shenzhen Yuejiang Technology Co., Ltd.

Dobra, A. (2014). General classification of robots. Size criteria. *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 1–6. https://doi.org/10.1109/RAAD.2014.7002249

Duan, S., Wang, L., Wang, F., Han, X., & Liu, G. (2021). A technique for inversely identifying joint stiffnesses of robot arms via two-way TubeNets. *Inverse Problems in Science and Engineering*, *29*(13), 3041–3061. https://doi.org/10.1080/17415977.2021.1967344

European Commission. (2019, Dezembro 11). *The European Green Deal*. Document 52019DC0640. Obtido 2 de Julho de 2024, de https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2019:640:FIN

Evripidou, S., Georgiou, K., Doitsidis, L., Amanatiadis, A. A., Zinonos, Z., & Chatzichristofis, S. A. (2020). Educational Robotics: Platforms, Competitions and Expected Learning Outcomes. *IEEE Access*, *8*, 219534–219562. https://doi.org/10.1109/ACCESS.2020.3042555

Fairchild, M. (2024). *Types of industrial robots and their different uses*. Obtido 14 de Abril de 2024, de https://howtorobot.com/expert-insight/industrial-robot-types-and-their-different-uses

Festo. (2024). *Robôs cartesianos*. Obtido 13 de Abril de 2024, de https://www.festo.com/pt/pt/c/produtos/automacao-industrial/sistemas-de-manipulacao/robos-cartesianos-id_pim108/

Filiz Baştürk, M. (2024). Solar Energy Production and Economic Growth: An Analysis for EU Countries. *Sosyoekonomi*, *32*(60), 95–109. https://doi.org/10.17233/sosyoekonomi.2024.02.05

Fillment3D. (2024). *V Gantry Plate | Full Kit*. Obtido 9 de Julho de 2024, de https://fillment3d.pt/produto/v-gantry-plate-full-kit/

Gabriele, L., Tavernise, A., & Bertacchini, F. (2012). *Active Learning in a Robotics Laboratory with University Students* (pp. 315–339). https://doi.org/10.1108/S2044-9968(2012)000006C014

Gasparetto, A., & Scalera, L. (2019). *From the Unimate to the Delta Robot: The Early Decades of Industrial Robotics* (pp. 284–295). https://doi.org/10.1007/978-3-030-03538-9_23

Gero. (2018, Agosto 22). *3D Model of Dobot Magician with Tools (STEP & STL)*. Obtido 25 de Fevereiro de 2024, de https://forum.dobot.cc/t/3d-model-of-dobot-magician-with-tools-step-stl/1126

Gümbel, P., He, X., & Dröder, K. (2022). Precision optimized pose and trajectory planning for vertically articulated robot arms. *Procedia CIRP*, *106*, 185–190. https://doi.org/10.1016/j.procir.2022.02.176

Gunes, H., & Kucuk, S. (2022). A systematic review of educational robotics studies for the period 2010–2021. *Review of Education*, *10*(3). https://doi.org/10.1002/rev3.3381

Harini, D., Sri, K. B., Durga, M. M., & Brahmaiah, O. V. (2023). Crow Detection In Peanut Field Using Raspberry Pi. *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 260–266. https://doi.org/10.1109/ICACCS57279.2023.10112813

Hawksworth, J., & Berriman, R. (2018). *Will robots really steal our jobs?*

I-EXTRUDER. (2024a). *I-EXTRUDER Solder Paste Dispenser*. Obtido 4 de Maio de 2024, de https://www.i-extruder.com/en/20-44-i-extruder-solder-paste-dispenser.html#/27-usb_power_adapter_type-eu_plug/29-product_pack-standard

I-EXTRUDER. (2024b). *I-EXTRUDER User Manual*. Obtido 10 de Julho de 2024, de https://www.i-extruder.com/img/cms/100.webp

Igor Lirtsman. (2021, Agosto 16). *DM1 Series limit switch (1A 125V)*. Obtido 19 de Maio de 2024, de https://grabcad.com/library/dm1_series-limit-switch-1a-125v-1

International Organization for Standardization. (2021). *Robotics - Vocabulary (ISO 8373:2021)*.

ISEL. (2024). *Estudar no ISEL*. Obtido 5 de Maio de 2024, de https://www.isel.pt/o-isel-apresenta-se/estudar-no-ISEL

Jacobs, J., & Nel, S. (2023). A GENERIC COMPUTER VISION TOOL FOR RECOGNISING HUMAN ACTIVITIES. *South African Journal of Industrial Engineering*, *34*(3). https://doi.org/10.7166/34-3-2956

Jahangir Badashah, S., Reddy, B. K., & Usha, N. (2022). Telepresence Robot Using Raspberry Pi. *JOURNAL OF ALGEBRAIC STATISTICS*, *13*(2), 2165–2172. Obtido de https://publishoa.com

Jeoung-Myoung, K. (2023). AUTOMATA TECHNOLOGY IN THE MEDIEVAL ISLAMIC WORLD AND ITS INFLUENCE ON KOREA. *Journal of Oriental Studies*, *105*(2). https://doi.org/10.26577/JOS.2023.v105.i2.011

Kabeyi, M. J. B., & Olanrewaju, O. A. (2023). Smart grid technologies and application in the sustainable energy transition: a review. *International Journal of Sustainable Energy*, *42*(1), 685–758. https://doi.org/10.1080/14786451.2023.2222298

Karthikeyan, S., Raj, R. A., Cruz, M. V., Chen, L., Vishal, J. L. A., & Rohith, V. S. (2023). A Systematic Analysis on Raspberry Pi Prototyping: Uses, Challenges, Benefits, and Drawbacks. *IEEE Internet of Things Journal*, *10*(16), 14397–14417. https://doi.org/10.1109/JIOT.2023.3262942

Kazakov, V. (2021). *Outline mechanical robotic arm with gripper isolated on white. Vector illustration*. Obtido 27 de Fevereiro de 2024, de

https://www.dreamstime.com/outline-mechanical-robotic-arm-gripper-isolated-white-vector-illustration-image224951668

Khalil, W., & Dombre, E. (2002). Introduction to geometric and kinematic modeling of parallel robots. Em *Modeling, Identification and Control of Robots* (pp. 171–190). Elsevier. https://doi.org/10.1016/B978-190399666-9/50008-7

Kılıç, S., & Gökoğlu, S. (2022). Exploring the Usability of Virtual Robotics Programming Curriculum for Robotics Programming Teaching. *Informatics in Education*, *21*(3), 523–540. https://doi.org/10.15388/infedu.2022.20

Klafter, R. D., Chmielewski, T. A., & Negin, M. (1989). *Robotic Engineering: An Integrated Approach*. Prentice-Hall International, Inc.

Kruger, B. (2013, Setembro 29). *Arduino CNC Shield V3.XX – Assembly Guide*. Obtido 16 de Julho de 2024, de https://blog.protoneer.co.nz/arduino-cnc-shield-v3-00-assembly-guide/

Kurniati, E., Suwono, H., Ibrohim, I., Suryadi, A., & Saefi, M. (2022). International Scientific Collaboration and Research Topics on STEM Education: A Systematic Review. *Eurasia Journal of Mathematics, Science and Technology Education*, *18*(4), em2095. https://doi.org/10.29333/ejmste/11903

Lego. (2024). *Robô Inventor*. Obtido 12 de Março de 2024, de https://www.lego.com/pt-pt/product/robot-inventor-51515

Li, H., Dong, C., Jia, X., Xiang, S., & Hu, Z. (2023). Design of Automatic Sorting and Transportation System for Fruits and Vegetables Based on Dobot Magician Robotic Arm. *2023 2nd International Symposium on Control Engineering and Robotics (ISCER)*, 229–234. https://doi.org/10.1109/ISCER58777.2023.00045

Li, P. (2020). *Visual Calibration, Identification and Control of 6-RSS Parallel Robots* [Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Mechanical Engineering)]. Concordia University.

Manners, D. (2021). *Raspberry Pi is the most popular SBC.* Electronics Weekly.

Marçal, R. M. N. (2023). *Desenvolvimento de um protótipo de veículo autónomo a partir de um carro telecomandado* [Trabalho de projeto para obtenção do grau de Mestre em Engenharia Mecânica]. Instituto Superior de Engenharia de Lisboa.

Marques, M. Â. G. (2017). *Desenvolvimento de um Robô para Combate a Fogos num Cenário de Simulação* [Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Mecânica]. Instituto Superior de Engenharia de Lisboa.

Mikroelectron. (2024). *ARDUINO CNC SHIELD V3*. Obtido 14 de Julho de 2024, de https://mikroelectron.com/Product/ARDUINO-CNC-SHIELD-V3/

Modular Robotics. (2024). *Cubelets robot blocks - Modular Robotics*.

Natharani, R., Liri, F., Samawi, J., Lin, H., Lee, K., Ruppert, N., George, K., & Panangadan, A. (2021). Voice Controlled Object Grasping Robotic Arm for Visually Impaired Disabled Veterans. *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1–6. https://doi.org/10.1109/CONECCT52877.2021.9622354

Neves, H. R. M. (2022). *Projetos de automação utilizando robótica industrial* [Relatório de Estágio de Natureza Profissional para a obtenção do grau de Mestre em Engenharia Eletrotécnica]. Instituto Superior de Engenharia de Coimbra.

Niku, S. B. (2020). *Introduction to Robotics: analysis, control, applications* (3.ª ed.). John Wiley & Sons Ltf.

Nugroho, E. A., Setiawan, J. D., & Munadi, M. (2023). Handling Four DOF Robot to Move Objects Based on Color and Weight using Fuzzy Logic Control. *Journal of Robotics and Control (JRC)*, *4*(6), 769–779. https://doi.org/10.18196/jrc.v4i6.20087

Observatório da Energia, DGEG – Direção Geral de Energia e Geologia, D. de S. de P. E. e E., & ADENE – Agência para a Energia, D. de F. I. e E. (2023). *Energia em Números - Edição 2023*. ADENE – Agência para a Energia.

OpenBuilds. (2024, Setembro 26). *OpenBuilds GrabCAD*. Obtido 23 de Janeiro de 2024, de https://grabcad.com/openbuilds-1

Osoyoo. (2017, Abril 7). *CNC Shield V3.0+A4988 Installation Guide*. Obtido 10 de Julho de 2024, de https://osoyoo.com/2017/04/07/arduino-uno-cnc-shield-v3-0-a4988/

Padoan, F. C. S. M., Altimari, P., & Pagnanelli, F. (2019). Recycling of end of life photovoltaic panels: A chemical prospective on process development. *Solar Energy*, *177*, 746–761. https://doi.org/10.1016/j.solener.2018.12.003

Pereira, A., Carreira, F., Campos, F., & Calado, J. (2018). *Precision on a mobile robot platform tool with dual arm*.

Piñeros Glasscock, J. S. (2022). The puzzle of learning by doing and the gradability of knowledge-how. *Philosophy and Phenomenological Research*, *105*(3), 619–637. https://doi.org/10.1111/phpr.12832

Pires, J. N. (2007). *Industrial Robots Programming: Building Applications for the Factories of the Future*. Springer. Obtido 2 de Março de 2024, de https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html

Poynton, B. (2022). From R.U.R. to Westworld: Personal Revolt, Digital Technology, and the Making of a New Robot Ur-text. *Comparative Drama*, *56*(4), 363–388. https://doi.org/10.1353/cdr.2022.0023

PwC. (2017). *Global Artificial Intelligence Study: Sizing the prize*.

Rajguru Electronics. (2024). *CNC Shield For A4988 Datasheet*. Obtido 4 de Maio de 2024, de https://rajguruelectronics.com/Product/434/CNC%20shield%20for%20A4988.pdf

Raspberry Pi. (2024). *Raspberry Pi 5*. Obtido 1 de Abril de 2024, de https://www.raspberrypi.com/products/raspberry-pi-5/

Rodpan, S. (2022, Fevereiro 9). *Stepper Linear Slider 80mm*. Obtido 11 de Abril de 2024, de https://grabcad.com/library/stepper-linear-slider-80mm-1

Sandbox Academy. (2024). *BP101 Blockly Programming Beginner 101*. Obtido 3 de Abril de 2024, de https://sandboxacademy.ca/events/bl101-blockly-programming-beginner-101/

Sharma, A. (2024). Making electric vehicle batteries safer through better inspection using artificial intelligence and cobots. *International Journal of Production Research*, *62*(4), 1277–1296. https://doi.org/10.1080/00207543.2023.2180308

Sherwani, F., Asad, M. M., & Ibrahim, B. S. K. K. (2020). Collaborative Robots and Industrial Revolution 4.0 (IR 4.0). *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, 1–5. https://doi.org/10.1109/ICETST49965.2020.9080724

SolarTurtle. (2023, Agosto 9). *V-154-1C25 Micro switch on off Single Pole Double Throw*. Obtido 8 de Maio de 2024, de https://grabcad.com/library/v-154-1c25-micro-switch-on-off-single-pole-double-throw-1

S&P Global: Country/Territory Report - Portugal. (2023). Em *Portugal Country Monitor*. Obtido de https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=173349751&site=eds-live

Špaková, A., Ferenčík, N., Sedláková, V., Kolembusová, P., Steingartner, W., & Hudák, R. (2024). Device for monitoring the vital functions of athletes using Arduino UNO development board. *2024 IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 000357–000364. https://doi.org/10.1109/SAMI60510.2024.10432865

STEM Education Works. (2024). *Dobot Magician*. Obtido 2 de Abril de 2024, de https://stemeducationworks.com/product/dobot-magician/

STEPPERONLINE. (2024). *Nema 17*. Obtido 5 de Maio de 2024, de https://www.omc-stepperonline.com/nema-17-stepper-motor

Sutam, B., Maneetham, D., Crisnapati, P. N., & Srichaipanya, W. (2023). The Solar Panels Cleaning Robot Control via IoT. *2023 11th International Conference on Cyber and IT Service Management (CITSM)*, 1–6. https://doi.org/10.1109/CITSM60085.2023.10455327

Ultralytics. (2023, Novembro 12). *Ultralytics YOLO Docs*. Obtido 1 de Setembro de 2024, de https://docs.ultralytics.com/

Vailshery, L. S. (2024, Abril 3). *Most used programming languages among developers worldwide as of 2023*. Statista. Obtido 10 de Março de 2024, de https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

Verl, A. (2023). Aktuelle Trends in der Robotik. *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, *118*(11), 736–737. https://doi.org/10.1515/zwf-2023-1159

Yang, K., Liu, X., & Chen, G. (2020). Global Research Trends in Robot Education in 2009-2019: A Bibliometric Analysis. *International Journal of Information and Education Technology*, *10*(6), 476–481. https://doi.org/10.18178/ijiet.2020.10.6.1410

Yates, D. R., Vaessen, C., & Roupret, M. (2011). From Leonardo to da Vinci: the history of robot-assisted surgery in urology. *BJU International*, *108*(11), 1708–1713. https://doi.org/10.1111/j.1464-410X.2011.10576.x

Yilmaz Ince, E., & Koc, M. (2021). The consequences of robotics programming education on computational thinking skills: An intervention of the Young Engineer's Workshop (YEW). *Computer Applications in Engineering Education*, *29*(1), 191–208. https://doi.org/10.1002/cae.22321

Yolcu, V., & Demirer, V. (2023). The effects of educational robotics in programming education on students' programming success, computational thinking, and transfer of learning. *Computer Applications in Engineering Education*, *31*(6), 1633–1647. https://doi.org/10.1002/cae.22664

Zafar, M. H., Langås, E. F., & Sanfilippo, F. (2024). Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review. *Robotics and Computer-Integrated Manufacturing*, *89*, 102769. https://doi.org/10.1016/j.rcim.2024.102769