FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Robô de baixo custo para mini hortas

Francisco Terra

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Filipe Santos

Co-orientador: Armando Sousa

28 de Junho de 2021

Resumo

O crescimento da população mundial tem vindo, incrementalmente, a aumentar a pressão na produção e no fornecimento de alimentos. De forma a aumentar o volume de produção é necessário aumentar a eficácia dos métodos de cultivo atuais. A integração da *Internet of things* (IoT) no processo agrícola permite a monitorização de diversas grandezas físicas, o que possibilita um controlo automatizado e muitas vezes localizado. Desta necessidade surgiu o projeto *open-source* Farmbot que se insere na categoria da agricultura de precisão e que é constituído por um robô cartesiano (semelhante a uma impressora 3D).

Com este trabalho, pretende-se alcançar uma solução de baixo custo e *open-source* que consiga automatizar os processos de monitorização, plantação e nutrição das plantas em hortas de pequena dimensão. É uma solução que pretende fomentar o cultivo de alimentos perto dos centros urbanos e que sendo automatizada, permite a sua utilização por quem não tem conhecimentos agrícolas.

O PixelCropBot foi inspirado na solução Farmbot, à qual foi acrescentada a mobilidade de um robô diferencial, tornando o robô mais versátil. A localização do robô é determinada recorrendo à deteção e descodificação de QR codes. O controlo do robô é conseguido com o recurso a uma Raspberry Pi 4, conectada a uma AgIoT. As aplicações que visam o controlo da estrutura foram desenvolvidas tendo como base a arquitetura ROS2.

Abstract

Increasing population is exerting pressure day by day on available limited food supply. In order to increase the volume of production, it is necessary to increase the effectiveness of current cultivation methods. The integration of the IoT in the agricultural process allows monitoring different parameters (e.g. temperature, soil moisture), which allows an automated and localized control. FarmBot is an open source precision agriculture CNC farming project consisting of a Cartesian coordinate robot farming machine (similar to a 3D printer).

This work aims to achieve a low-cost solution that can automate the processes of monitoring, planting and nourishing plants in small gardens. A solution that allows cultivation of food close to urban centers and that, being automated, can be used by anyone without agriculture expertise.

The PixelCropBot was inspired by the Farmbot solution, to which the mobility of a differential robot was added, making the robot more versatile. The location of the robot is determined by detecting and decoding QR codes. Control of the robot is achieved using a Raspberry Pi 4 connected to an AgIoT. The applications developed to control the structure were based on the ROS2 architecture.

Agradecimentos

Ao meu orientador, Filipe Santos, e ao meu coorientador, Armando Sousa, gostava de agradecer a oportunidade de desenvolver este projeto, o apoio, a disponibilidade e o aconselhamento prestado ao longo do semestre.

Aos meus amigos Pedro Moura, André Baltazar e Alexandre Carvalho, por todo o incentivo e ajuda prestada no decorrer desta dissertação. Obrigado por acreditarem em mim e no meu trabalho.

Ao professor Mário Cunha, ao André Aguiar e ao Sandro Magalhães, por seguirem de perto o desenvolvimento deste projeto. Ao Leandro Rodrigues pela partilha de conhecimento agronómico e pelo interesse demonstrado pelo trabalho desenvolvido, estando sempre disponível para ajudar.

Ao Daniel Silva e ao Vitor Tinoco e à restante equipa do CRIIS, por proporcionarem um ótimo ambiente de trabalho e por estarem disponíveis para ajudar sempre que foi preciso.

Ao senhor José Carlos, pelo apoio no desenho da PCB, pela ajuda a soldar diversos componentes e pelas dicas e informações transmitidas sobre este processo.

Aos meus pais e irmão, pelo seu amor, paciência e apoio incondicional durante este período de aprendizagem.

Francisco Terra

Conteúdo

1 Introdução **1**
- 1.1 Objetivo . . . 2
- 1.2 Motivação . . . 2
- 1.3 Estrutura da dissertação . . . 2

2 Revisão Bibliográfica **3**
- 2.1 Introdução . . . 3
- 2.2 Estado da arte . . . 3
- 2.3 Localização utilizando QR codes . . . 10

3 Estrutura **13**

4 Eletrónica **19**

5 Software **25**
- 5.0.1 Nós ROS2 . . . 25
- 5.0.2 Microcontroladores . . . 30

6 Localização **33**
- 6.1 Introdução . . . 33
- 6.2 Implementação . . . 34
- 6.3 Resultados . . . 39

7 Testes **43**

8 Conclusão e Trabalho futuro **47**
- 8.1 Conclusão . . . 47
- 8.2 Trabalho futuro . . . 47

A Informação adicional sobre o Farmbot **49**
- A.1 Hardware . . . 49
  - A.1.1 Trilhos . . . 50
  - A.1.2 Ferramentas . . . 51
  - A.1.3 Controlo . . . 53
- A.2 Comparação entre os modelos Genesis e Express . . . 54
- A.3 Software . . . 55

B Lista de materiais **61**

C Esquemático da PCB **63**

Referências **65**

Lista de Figuras

2.1 Farmbot Genesis . . . 3
2.2 MicroCEA . . . 4
2.3 OpenAg Personal Food computer . . . 5
2.4 Thorvard . . . 5
2.5 Thorvard II . . . 6
2.6 Agribot . . . 6
2.7 RobotFarm . . . 7
2.8 Titan Smartponics . . . 8
2.9 PlantTalk . . . 8
2.10 Plantação onde foi implementado o AgriTalk e conjunto de sensores usados na estação meteorológica . . . 9
2.11 Agrobot . . . 9

3.1 Esquerda: Desenho CAD do PixelCropBot em fase de projeto. Direita: Robô construído. . . . 13
3.2 Peça móvel responsável pelo movimento ao longo do eixo Y . . . 14
3.3 Farmbot e PixelCropBot, imagem da direita e esquerda, respetivamente . . . 15
3.4 Elementos que permitem a fixação do motor e da roda ao perfil V-slot da estrutura 15
3.5 Fixação da caixa do motor . . . 16
3.6 Peças utilizadas para a fixação da roda ao eixo . . . 16
3.7 Peças criadas para a fixação da roda livre . . . 17
3.8 Elementos que constituem a caixa onde é fixada a roda livre . . . 17
3.9 Rodas após a fixação à estrutura. Esquerda: roda livre. Direita: Roda acoplada ao motor. . . . 17
3.10 Elementos responsáveis pelo movimento nos eixos Y e Z . . . 18
3.11 Caixa criada para a fixação da câmara à estrutura . . . 18

4.1 Visão geral dos componentes de controlo . . . 19
4.2 Esquema de ligações mínimas para controlo de um motor através de um DRV8825 20
4.3 Esquerda: Design final da PCB desenhada no EasyEda. Direita: PCB com todos os componentes soldados . . . 21
4.4 Circuito de potencia controlado por onda PWM presente na AgIoT . . . 22
4.5 Interior da caixa de electrónica . . . 23
4.6 Ligações elétricas da caixa de electrónica . . . 23

5.1 Comunicação entre todos os intervenientes do sistema . . . 25
5.2 Conjunto de teclas que permitem o controlo dos motores DC da estrutura. O duty cycle do PWM do controlo dos motores irá ter sempre os mesmo valor (30%) . . . 26
5.3 Teclas que permitem o movimento no eixo Y e Z . . . 26

5.4 Códigos que indicam como é que será o movimento do robô 27
5.5 Exemplo de trama enviada para o ATmega. Neste caso, as duas rodas irão girar no sentido positivo (com valores de PWM de 30% e 25%, roda direita e esquerda respetivamente) 27
5.6 Trama de dados que permite o movimento do eixo y até à posição 1000 com uma velocidade de 400 28
5.7 Sequência de controlo dos motores de passo 28
5.8 Representação do robô após a sequência de inicialização (vista de cima) 29
5.9 Máquinas de estados usadas para o controlo do motor de passo. Esquerda: máquina de estado responsável pelo movimento dos eixos. Direita: máquina de estados que sinaliza as anomalias 31

6.1 Elementos que compõem o QR code 34
6.2 Resultados da aplicação dos diferentes processos de binarização: Otsu, *Mean*, *Guassian* e Canny (da esquerda para a direita) 34
6.3 Contornos obtidos, a vermelho, após a fase de pré-processamento 35
6.4 Região de interesse obtida 36
6.5 Relaçaõ entre o ROI e o vetor orientação 36
6.6 Padrão que mostra como é que os vértices devem ser ordenados 37
6.7 QR codes detetados com os vértices numerados segundo o padrão e com os eixos representados 37
6.8 Vetor L, a vermelho, e informação sobre o ângulo deste vetor com o eixo x 38
6.9 Região de interesse após terem sido identificados os cantos e a azul o resultado descodificado 39
6.10 Equipamento de teste 40
6.11 Exemplo do resultado obtido: ângulo em graus no canto superior direito e as coordenadas da câmara abaixo do ponto central 40
6.12 Da esquerda para a direita e de cima para baixo: High Qualitycamera, camera V2 1280x720 e camera V2 1920x1080 41

7.1 Local onde os testes foram realizados 44
7.2 Representação do cenário usado durante os testes 44
7.3 Imagem obtida durante o processo de localização 45

A.1 Visão geral do Hardware do Farmbot 50
A.2 Ferramentas incluídas no modelo Genesis 51
A.3 Visão geral do Hardware do Farmbot 53
A.4 Visão high level de todo o sistema 55
A.5 Aplicação web em vários dispositivos 56
A.6 Menu de criação de sequencia 56
A.7 Menu de criação de sequencia 57
A.8 Comunicação entre as diferentes partes que compõe o sistema 57

B.1 Lista de materiais utilizada para a construção do PixelCropBot 62

C.1 Ligações elétricas da PCB desenhada para o controlo dos motores de passo . . . 63

Lista de Tabelas

6.1 Resultados do algoritmo de deteção dos QR codes 41

A.1 Comparação das dimensões entre o modelo Express e Genesis 55

Abreviaturas e Símbolos

AgIoT	IoT solution for Agrifood Sector
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BLE	Bluethoot Low Energy
CAD	Computer Aided Design
CAN	Controller Area Network
CNC	Computer Numeric Control
DC	Direct current
GPS	Global Positioning System
GUI	Graphical User Interface
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
ID	Identity
IoT	Internet of Things
JSON	JavaScript Object Notation
LED	Light-emitting diode
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MQTT	Message Queuing Telemetry Transport
PCB	Printed circuit board
PFC	Personal Food Computer
PWM	Pulse Width Modulation
QR	Quick Response
ROS	Robot Operating System
RPC	Remote Procedure Call
RTK	Real Time Kinematic
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
UTM	Universal Tool Mount
UV	Ultravioleta

Capítulo 1

Introdução

O crescimento da população mundial tem vindo, incrementalmente, a aumentar a pressão na produção e no fornecimento de alimentos. Os vegetais são um elemento central da dieta do ser humano, existindo uma enorme variedade de espécies que podem ser cultivadas[1]. Uma das prioridades das Nações Unidas centra-se em promover uma agricultura sustentável, recorrendo à integração de novas tecnologias, aliada a uma distribuição justa dos alimentos, de forma a sustentar a população mundial[2].

A maioria das áreas adequadas ao cultivo de alimentos já estão a ser utilizadas. De forma a aumentar o volume de produção, com recursos limitados, é necessário aumentar a eficácia dos métodos atuais[3]. Adicionalmente, a poluição e as alterações climáticas estão cada vez mais a ameaçar a quantidade e a qualidade dos alimentos em todo o planeta. É expectável que o desenvolvimento da *smart farming* seja uma contramedida efetiva, incentivando diversas pesquisas sobre a integração da IoT (Internet of Things) em aplicações agrícolas[4].

A água é um bem essencial cada vez mais escasso, fruto das alterações climáticas, sendo necessário tornar o processo de irrigação mais eficiente de forma a otimizar o consumo de água. A presença de água no solo representa uma condição necessária para o crescimento ideal de uma planta, no entanto o seu excesso é prejudicial[5].

Os métodos agrícolas convencionais apresentam algumas desvantagens/inconvenientes que podem ser resolvidos trocando a maquinaria pesada por soluções robóticas mais leves. Algumas destas vantagens são a diminuição da compactação do solo, a diminuição do consumo de combustível e o aumento de precisão[6]. A agricultura de precisão pode ser definida como a arte e a ciência de usar a tecnologia para melhorar o rendimento da colheita. A agricultura de precisão tem como objetivo otimizar o rendimento e o investimento, procurando adaptar-se à variabilidade ambiental, melhorando as condições de crescimento das plantas, caso a caso. Este objetivo é alcançado recorrendo a um controlo baseado na análise da informação de diversos sensores colocados no terreno. As ações podem ser desenvolvidas localmente como na aplicação de fertilizantes ou na deteção de doenças[7].

Apesar de muitas casas possuírem jardim, a produção dos alimentos acaba normalmente por não ser realizada por ser necessário uma continua observação e atenção da horta, o que muitas

vezes não se coaduna com a rotina diária. Por outro lado, as pessoas muitas vezes não têm o conhecimento suficiente, na área da agricultura, para poderem tirar o máximo proveito das suas plantações. Por esta razão têm aparecido muitas soluções que permitem automatizar este processo. Desta necessidade surgiu o projeto *open-source* Farmbot que se insere na categoria da agricultura de precisão e que é constituído por um robô cartesiano (semelhante a uma impressora 3D), sendo uma solução tecnológica e acessível com o intuito de facilitar a produção de alimentos[8].

1.1 Objetivo

O objetivo principal deste trabalho é alcançar uma solução de baixo custo que possa ser utilizada no meio agrícola, com o intuito de automatizar os processos de monitorizar, plantar e nutrir as plantas em hortas de pequena dimensão.

Existindo ainda um conjunto de objetivos secundários que seriam importantes atingir:

- Melhorar uma solução *open-source* com a incorporação do ROS2.
- Alcançar uma solução modular para fins de investigação.
- Permitir a utilização por quem não tem conhecimentos sobre a agricultura.

1.2 Motivação

Com base no que foi referido anteriormente, é preciso mudar rapidamente a forma como se produz os alimentos, para que seja possível atender às necessidades de uma população em crescimento. Adicionalmente é importante uma gestão calculada e sustentável que possa garantir as melhores condições para o desenvolvimento das plantas, sem que haja desperdício.

Esta é uma área onde existe uma comunidade que ao longo dos últimos anos partilha do mesmo desejo, sendo que as soluções desenvolvidas são normalmente *open-source*.

Este projeto é uma oportunidade para aprofundar o estudo sobre o desenvolvimento tecnológico e agronómico, conciliando as duas vertentes numa solução que pretende melhorar as já existentes.

1.3 Estrutura da dissertação

Esta dissertação está dividida em 8 capítulos. O capítulo 2 apresenta diversas soluções criadas para o cultivo automático de alimentos e plantas. Os capítulos 3 e 4 explicam o Hardware utilizados para a construção do PixelCropBot. O capítulo 3 explica como a estrutura do robô foi construída e o capítulo 4 diz respeito à parte eletrónica. O capítulo 5 explica a arquitetura de controlo criada neste projeto. O capítulo 6 explica o método de localização utilizado neste robô, que é baseado no reconhecimento de QR codes. O capítulo 7 mostra os testes e alguns resultados alcançados pelo PixelCropBot. Por último, o capítulo 8 apresenta as considerações finais sobre o trabalho desenvolvido e sobre o que precisa de ser melhorado.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

Esta secção tem como objetivo analisar as diversas soluções criadas para o cultivo automático de alimentos e plantas. A solução Farmbot apresenta uma análise mais exaustiva, visto que será a base do trabalho desta dissertação, encontrando-se no anexo A informação adicional que foi recolhida durante a pesquisa. A informação referente ao Farmbot foi retirada da extensa documentação encontrada no *site* oficial e no *White Paper* pulicado em 2013[8]. No final deste capítulo são apresentados alguns exemplos sobre o método de localização baseado em QR codes (Quick Response Code).

2.2 Estado da arte

Farmbot (figura 2.1)



Figura 2.1: Farmbot Genesis

O Farmbot é um projeto que se insere na categoria da agricultura de precisão. É constituído por um robô cartesiano (semelhante a uma impressora 3D) e software necessário para a monitorização de uma pequena horta. Foi criado em 2011 por um estudante americano, Rory Aronson,

com o objetivo de criar uma solução tecnológica e acessível com o intuito de facilitar a produção de alimentos. Esta ideia está prsente no seu lema: *"Create an open and accessible technology aiding everyone to grow food and to grow food for everyone"*. Por este motivo, esta é uma solução totalmente *open-source*, sendo público o hardware utilizado bem como os ficheiros de CAD criados, o software e toda a documentação associada ao projeto. O utilizador pode controlar a horta através de uma aplicação web, podendo ser acedida através de qualquer *web browser*.

O Farmbot apresenta uma nova abordagem relativamente à agricultura de precisão, colocando todo o equipamento sobre trilhos. Desta forma o sistema tem a capacidade de ser extremamente preciso e de reposicionar as ferramentas em locais exatos, repetidamente, ao longo do tempo. Esta tecnologia é semelhante à utilizada em impressoras 3D e fresadoras. O Farmbot apresenta um conjunto de ferramentas com diferentes funcionalidades, que podem ser encaixadas na *Universal Tool Mount* (UTM). Estas ferramentas incluem um injetor de sementes, um emissor de água, um removedor de infestantes e um sensor de humidade do solo. Existe uma câmara que é utilizada para a deteção de infestantes e acompanhamento do crescimento das plantas. Os elementos fundamentais para o controlo são o Farmduino (placa que contem um ATmega 2560 e as ligações a todos os periféricos) e a Raspberry Pi 3B+.

MicroCEA (figura 2.2)



Figura 2.2: MicroCEA

O MicroCEA (Micro Controlled Environment Agriculture) é baseado no MIT Food computer, sendo um dispositivo que automatiza o crescimento de vegetais, ideal para ser usado dentro de casa. Os alimentos são cultivados dentro de uma caixa selada e que apresenta os componentes necessários para o controlo das luzes LED, da humidade no ar, do nível de C02, da temperatura, do pH e da condutividade da água. O centro da arquitetura da parte eletrónica é a Raspberry Pi 2, que é responsável pelo controlo do dispositivo. Este elemento recebe a informação recolhida pelos sensores através do Arduino (ATmega 2560), sendo responsável pela parte lógica e atuação.

É ainda responsável pelo envio de informação para a base de dados, sendo comunicação efetuada utilizando MQTT e os dados são transmitidos e armazenados no formato JSON (facilitando a sua extração e análise). A GUI é uma página web que permite ao utilizador alterar as definições, visualizar os valores atuais dos sensores e gráficos que refletem as medições ao longo do tempo[2].

OpenAg Personal Food computer (figura 2.3)

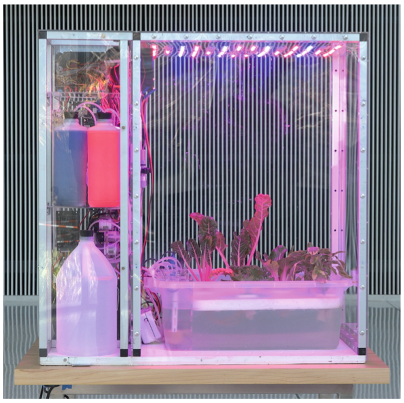


Figura 2.3: OpenAg Personal Food computer

O OpenAg Personal Food computer (PFC) é uma plataforma que permite o cultivo de plantas num ambiente controlado. Foi criado com o objetivo de ser *low-cost*, customizável, *open-source*, *user friendly* e de tamanho reduzido (806x502x809) mm. A parte eletrónica foi construída em torno de uma Raspberry Pi 3, sendo o hardware do PFC controlado através de ROS. A leitura dos diferentes sensores (temperatura, humidade, sensor de CO2, pH, condutividade elétrica e temperatura da água) e a atuação (iluminação, humidificador, ar condicionado, entre outros) são realizadas através de ATmega 2560. A interface gráfica, permite ao utilizador, visualizar a informação de diferentes sensores ao longo do crescimento da planta, criação de vídeos *time-lapse* e atuação manual dos atuadores[9].

Thorvard (figura 2.4)



Figura 2.4: Thorvard

O Thorvard é um robô móvel e versátil para operações em campos agrícolas. Este pretende ser uma solução *low-cost* que possa realizar operações durante o ciclo de crescimento da planta, em diferentes tipos de plantações. Esta plataforma foi desenvolvida utilizando ROS e comunica com os motores utilizando o protocolo CANopen. A localização do robô é conseguida através da utilização de RTK-GPS. Este robô apresenta ferramentas para semear, para remover infestantes e de calagem de forma a aumentar o nível de pH dos solos[6].

Thorvald II (figura 2.5)



Figura 2.5: Thorvard II

O Thorvald II é um robô móvel modular, altamente reconfigurável, adaptado a qualquer tipo de ambiente para ser utilizado em aplicações agrícolas. É possível criar um conjunto de robôs diferentes, usando os mesmos módulos e permitindo adicionar propriedades ou controlar a altura do robô. O Thorvald I tinha uma altura fixa de 59cm que era menor do que altura máxima das plantas que estava a cultivar, não sendo a solução ideal para acompanhar o crescimento da mesma até ao fim. A configuração base é composta por um robô com 4 rodas, existindo também versões diferenciais ou em triciclo. O computador de bordo controla os periféricos através de ROS a correr no sistema operativo Linux Ubuntu e as comunicações são realizadas usando o protocolo CANopen. A arquitetura modelar do ROS permite que o código funcione nas diferentes versões do robô[10].

Agribot (figura 2.6)



Figura 2.6: Agribot

O Agribot é um robô móvel criado para ser uma plataforma experimental de controlo, navegação e aquisição de dados em ambiente agrícola. A principal finalidade desta aplicação é a aquisição remota de diferentes parâmetros agronómicos. Este robô foi projetado para ser utilizado durante toda a fase de crescimento das plantas, apresentando assim, uma estrutura em pórtico assente sobre 4 rodas. O robô tem uma altura de 1.8m, motor a diesel, um sistema de propulsão hidráulico e uma rede de controlo baseada em CAN[11].

Growth Monitor Pi

O Growth Monitor Pi é uma solução económica para a monitorização de plantas em ambientes controlados e em estufas. É composto por uma Raspberry Pi, uma câmara e um conjunto de sensores (temperatura, humidade e intensidade luminosa) economicamente acessíveis. Para além da função de monitorização, este dispositivo é responsável por avisar o utilizador se algum dos parâmetros sair do intervalo desejado e pela criação de *datasets* referentes às condições medidas dentro da câmara de crescimento. O upload dos dados é feito através da aplicação *third-party* Rclone que é conectada ao serviço GoogleDrive[12].

RobotFarm (figura 2.7)



Figura 2.7: RobotFarm

O RobotFarm é uma solução automática para o cultivo de plantas, utilizando a técnica da hidroponia, ou seja, o solo é substituído por um substrato estéril e as plantas são irrigadas através de uma solução nutritiva formada por água e sais minerais. A sua reduzida dimensão e facilidade de utilização, torna-o uma solução atrativa para o uso doméstico. O ambiente de cultivo é totalmente independente do exterior, sendo o ar e a água esterilizados com radiação UV. O clima interno é regulado, controlando a temperatura, a humidade relativa e o nível de CO2. O RobotFarm apresenta uma interface HMI (human machine interface) constituída por um conjunto de botões e um display[4].

Titan Smartponics (figura 2.8)

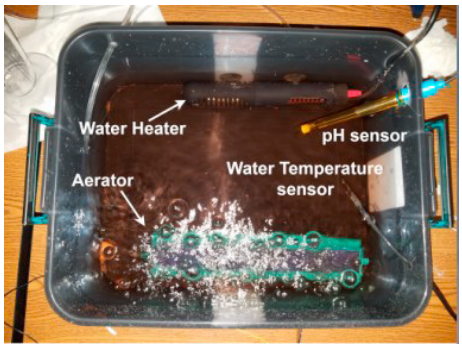


Figura 2.8: Titan Smartponics

O Titan Smartponics foi criado para ser um sistema para o cultivo de plantas e vegetais que possa operar independentemente das condições climatéricas exteriores. Isto foi conseguido através da hidroponia. O sistema foi automatizado, utilizando microcontroladores e sensores (condutividade elétrica, pH, temperatura da água e temperatura/humidade do ar) e tornando a intervenção humana quase inexistente, apenas necessária no início, aquando da plantação. O sistema utiliza dois arduinos responsáveis pela aquisição de dados e controlo. É utilizada uma Raspberry Pi que utiliza o software *open-source* Domoticz para monitorização dos dados que podem também ser acedidos através de uma aplicação para telemóvel[13].

PlantTalk (figura 2.9)

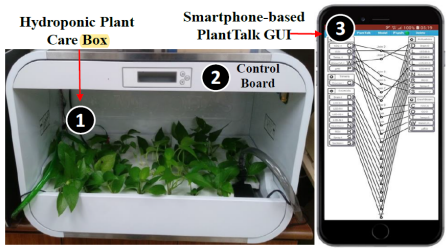


Figura 2.9: PlantTalk

O PlantTalk é uma solução baseada no Iot, que utiliza a técnica da hidroponia para o cultivo das plantas. O PlantTalk foi desenvolvido recorrendo à plataforma IoTtalk e o sistema é constituído por uma placa de controlo que recebe os dados de diversos sensores e os envia para o servidor através de uma ligação wi-fi ou ethernet. São várias as placas de controlo suportadas (como por exemplo o arduino ou o ESP8266). O utilizador tem acesso aos dados e ao controlo dos atuadores através de uma página web que pode ser facilmente acedida em qualquer smartphone[14].

iHydroIoT

Como a solução anterior, o iHydroIoT é um sistema baseado em hidroponia. O iHydrIoT é composto por um Arduino Uno, um módulo BLE (bluethoot low energy) para comunicar com o telemóvel, um conjunto de sensores e uma Raspberry Pi que é utilizada como sevidor. Estes dados são também enviados através da Raspberry Pi, via ethernet, para o serviço Ploty que permite a visualização dos mesmos. A aplicação móvel foi criada usando a linguagem de programação Swift e só se encontra disponível para versões superiores ao iOS10. Esta solução pode ser utilizada no modo automático ou manual[15].

AgriTalk (figura 2.10)

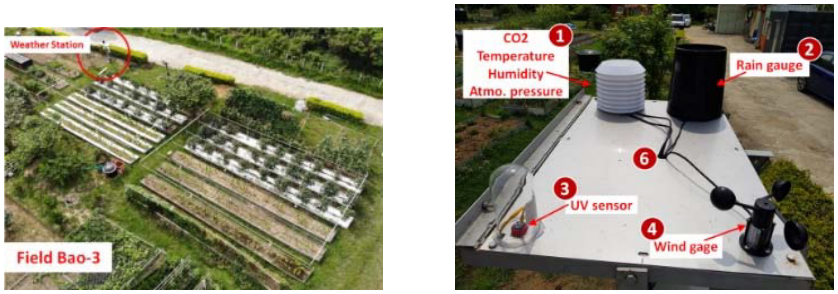


Figura 2.10: Plantação onde foi implementado o AgriTalk e conjunto de sensores usados na estação meteorológica

O AgriTalk é uma plataforma que usa IoT com o objetivo de ser uma solução dentro da vertente da agricultura de precisão. Esta solução apresenta uma enorme variedade de sensores, que estão espalhados em diferentes estações meteorológicas (temperatura, humidade, pressão atmosférica, nível de C02 e pluviosidade, sensor UV e de velocidade do vento) ou inseridos no solo (humidade, temperatura, condutividade elétrica e pH). Esta solução apresenta 3 atuadores: sistema de irrigação, spray de bio pesticida e fertilizante. Os dados podem ser analisados recorrendo a uma aplicação móvel que também permite controlar os atuadores[3].

Agrobot (figura 2.11)



Figura 2.11: Agrobot

O Agrobot foi desenvolvido para ser um robô móvel autónomo capaz de realizar diferentes funções em meio agrícola, como aplicação de sprays contendo remédio ou fertilizantes, análise do solo e da saúde da planta. O Agrobot usa sensores GPS para estimar a sua posição, no entanto, se for usado em espaços como estufas, terá como base o mapa da instalação. O Agrobot pode também ser controlado à distancia através de um computador remoto[16].

AeroGarden

O AeroGarden é uma pequena solução comercial para o cultivo de plantas dentro de casa. De forma a promover o crescimento das plantas faz uso das técnicas de hidroponia e aeroponia combinadas com a iluminação LED. O ambiente desta solução não é controlado sendo afetado pelo clima envolvente[17].

Leaf

Leaf é outra solução comercial ideal para o cultivo de plantas no interior. Ao contrário do AeroGarden possui a habilidade de controlar e adaptar o ambiente ao redor da planta. O controlo é realizado através da análise de dados de diversos sensores que também podem ser consultados pelo utilizador a partir de uma aplicação móvel[18].

2.3 Localização utilizando QR codes

Os QR codes já foram alvo de várias aplicações e experiências. Estes podem, por exemplo, ser utilizados para obter a localização de uma câmara[19]. De forma a processar as imagens recolhidas pela câmara foi utilizada a biblioteca Open Source Computer Vision Library (OpenCV), que contém um conjunto de funções para a manipulação de imagens. Para a descodificação do QR code foi usada a biblioteca Zbar, uma biblioteca *open-source*, que permite ler vários tipos de códigos como QR e o código de barras. Nesta proposta, a localização era realizada recorrendo à trilateração, sendo, portanto, necessário que no campo de visão da câmara estejam presentes no mínimo três QR codes.

Estes códigos foram também usados em experiências de localização e navegação[20]. Neste caso o método usado para inferir a localização foi a triangulação.

Noutra aplicação[21], os QR codes foram colocados no teto e foram lidos através da câmara de um smartphone. Os QR codes são utilizados como *landmarks* artificiais, constituindo pontos de referência que ajudam a estimar a posição do robô. É também analisada a orientação do QR code de forma a ser possível inferir a direção do movimento do robô.

Nesta aplicação, as coordenadas do QR code estão guardadas numa base de dados e podem ser acedidas através do ID lido no mesmo. A posição do robô é posteriormente calculada através desta informação e de cálculos geométricos aplicados à imagem. Foram detetados problemas na deteção quando o robô se movia rapidamente, pelo que foram colocados círculos em volta dos QR codes. Quando o robô deteta os círculos, abranda, de forma a conseguir descodificar a informação. Esta

abordagem acabou no final por ser substituída pela utilização de QR codes vermelhos, trocando assim a localização de círculos pela localização de cor. Esta técnica permitiu que a velocidade de movimentação do robô pudesse ser maior do que no caso interior.

Na solução apresentada em [22], o robô está equipado com uma câmara que aponta para o teto, onde estão colocados os QR codes. Os QR codes foram uniformemente distribuídos pelo teto, em forma de grelha, sendo possível em qualquer posição detetar sempre pelo menos um QR code.

Capítulo 3

Estrutura

No presente capítulo é apresentado o trabalho de projeção e desenvolvimento da estrutura utilizada no robô PixelCropBot. No decorrer do mesmo, é abordado em detalhe o processo de decisão, ao nível do desenho e escolha de materiais e, posterior construção. A lista de materiais utilizadas para a construção do robô pode ser consultada na figura B.1 presente no apêndice B.

Atendendo às necessidades da agricultura de precisão, o PixelCropBot foi projetado para ser um robô versátil, capaz de se movimentar em ambientes de *microfarming* e de se adaptar a diferentes atividades agrícolas. Tendo ainda como base de estudo as tecnologias já existentes no mercado, o PixelCropBot foi desenhado com inspiração particular no Farmbot. O desenho da solução foi elaborado com recurso ao software Fusion360 e pode ser observado na figura 3.1.



Figura 3.1: Esquerda: Desenho CAD do PixelCropBot em fase de projeto. Direita: Robô construído.

Durante o processo de construção do robô, foram considerados vários materiais. Numa fase inicial, tendo em vista a obtenção de uma solução de baixo custo, ponderou-se a utilização de uma

estrutura baseada em tubos 4lean[1]. No entanto, pelo seu formato circular verificou-se uma maior complexidade em integrar os eixos relativos ao movimento da ferramenta. Nesse seguimento, considerou-se a utilização de perfis de alumínio V-slot, a partir dos quais é possível deslocar partes móveis da estrutura, como o eixo da ferramenta, com o apoio de V-wheels. Pelas semelhanças que apresenta à construção de uma impressora 3D, existiram uma variedade de exemplos que puderam ser explorados de forma a definir os materiais a utilizar.

Por sua vez, a peça que é responsável pelo movimento no eixo Y e onde é acoplado o eixo Z, foi projetado diretamente para esta estrutura e fabricada com base no CAD retirado da página de documentação do Farmbot[2]. De modo a utilizar uma peça de acoplamento dos eixos mais robusta, esta foi fabricada em alumínio. A peça pode ser observada na figura 3.2.

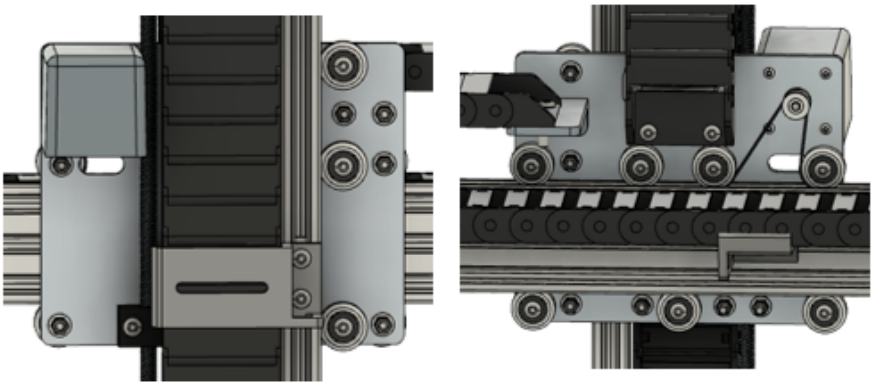


Figura 3.2: Peça móvel responsável pelo movimento ao longo do eixo Y

A diferença estrutural mais relevante face ao Farmbot, está presente na movimentação segundo o eixo X. No caso do Farmbot, esta é realizada sobre carris, o que implica a construção, por parte do utilizador, de uma estrutura em que estes possam assentar. No caso do Farmbot, é recomendado a construção de uma *Raised Bed*, uma estrutura de madeira elevada onde as plantas e alimentos são plantados. Desta forma a área de atuação do Farmbot está limitada ao local onde esta estrutura é instalada. Por sua vez, o PixelCropBot comporta-se como um robô diferencial, sendo o carril substituído por um movimento livre sobre quatro rodas, sem confinamento a uma área de atuação fixa. Embora esta solução ofereça uma maior versatilidade ao utilizador, a mesma agrega em si uma maior complexidade no controlo da direção e uma inferior precisão, quando comparada com um sistema movido por motores de passo. A diferença entre as duas soluções pode ser observada na figura 3.3.

[1] www.4lean.net
[2] genesis.farm.bot/v1.5/Extras/cad.html



Figura 3.3: Farmbot e PixelCropBot, imagem da direita e esquerda, respetivamente

A locomoção no eixo X é realizada recorrendo a dois motores DC (recorrentemente utilizados nas janelas dos automóveis), os quais estão acopladas a duas das rodas da estrutura. Como os motores apresentam a mesma orientação, estes estão colocados em rodas opostas e cruzadas na estrutura. Um dos motores encontra-se acoplado à roda da frente de um lado, enquanto que o outro se encontra acoplado à roda traseira do lado oposto. As duas rodas restantes estão presas a um eixo através de rolamentos, deslizando livremente.

De forma a tornar possível acoplar a roda ao eixo do motor, foi desenhada e impressa uma peça, com recurso a uma impressora 3D, colocada entre o motor e a roda, onde as duas partes são aparafusadas. A mesma foi desenhada, aproveitando para a sua fixação, as furações existentes tanto no eixo do motor, como na roda. Os parafusos que atravessam a roda foram presos utilizando porcas embutidas nas peças após a sua impressão.

Adicionalmente, foi desenhada uma caixa para integrar o motor e a roda com o perfil V-slot da estrutura. A figura 3.4 mostra como é que é feita a fixação da roda e do motor à estrutura principal.

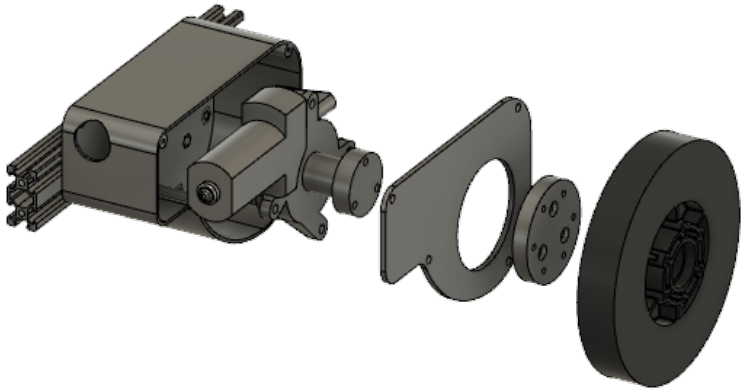


Figura 3.4: Elementos que permitem a fixação do motor e da roda ao perfil V-slot da estrutura

A caixa é fixada à estrutura através da utilização de 4 pequenas peças que são normalmente usadas em uniões perpendiculares entre os perfis de alumínio. A montagem desta caixa ao perfil de alumínio é apresentada na figura 3.5.

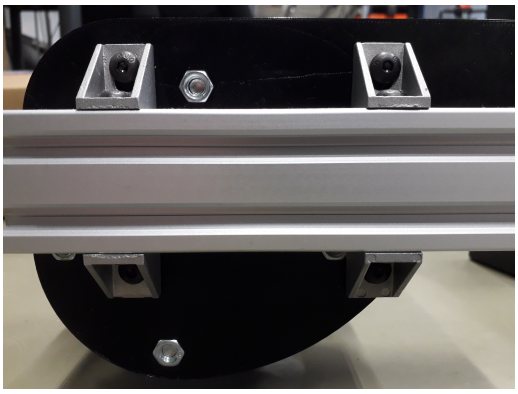


Figura 3.5: Fixação da caixa do motor

Às rodas traseiras foram acoplados dois rolamentos, um em cada lado da roda. Para esta fixação foram desenhadas duas peças adicionais que foram unidas através de seis parafusos, aproveitando assim as furações existentes na roda. A figura 3.6 apresenta as peças utilizadas para prender os rolamentos à roda.

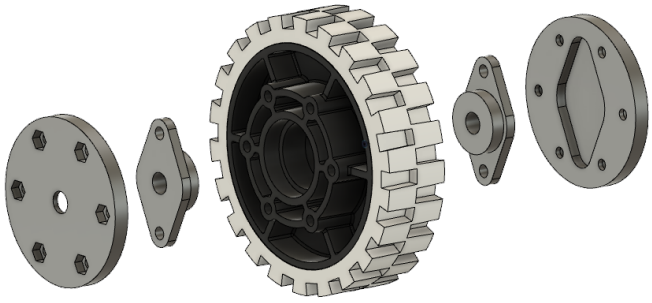


Figura 3.6: Peças utilizadas para a fixação da roda ao eixo

Estes rolamentos são atravessados por um eixo de 8mm que está fixo a uma caixa apertada à estrutura principal. Os rolamentos escolhidos têm dois parafusos que servem para prender o eixo ao rolamento, desta forma foi possível fixar a posição do rolamento, assim como a posição da roda para que esta não se mova ao longo do eixo. O rolamento que é preso ao eixo é o rolamento que se encontra na parte interior. Após a fixação do rolamento foi colocada a roda, o outro rolamento e a peça exterior, sendo tudo aparafusado. Conseguiu-se, assim, que as rodas traseiras deslizem quando o robô está em movimento. A figura 3.7 mostra algumas das peças impressas para este fim.

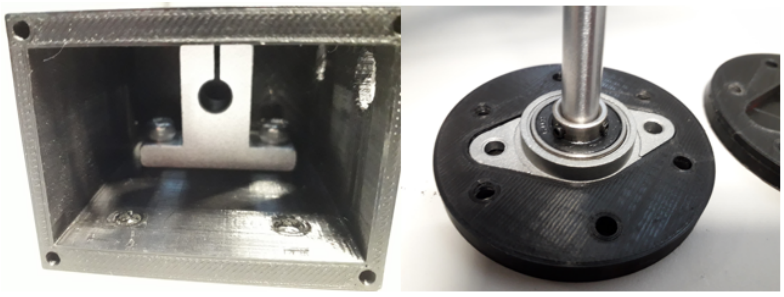


Figura 3.7: Peças criadas para a fixação da roda livre

É de realçar a presença de furações nas paredes laterais da caixa. Estas permitem aparafusar todos os elementos no interior. A montagem dos mesmos é ilustrada na figura 3.8.

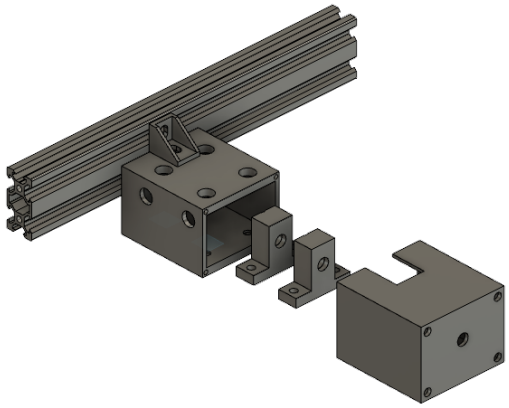


Figura 3.8: Elementos que constituem a caixa onde é fixada a roda livre

A integração das rodas com a caixa, o veio e o perfil V-slot é apresentada na figura 3.9.

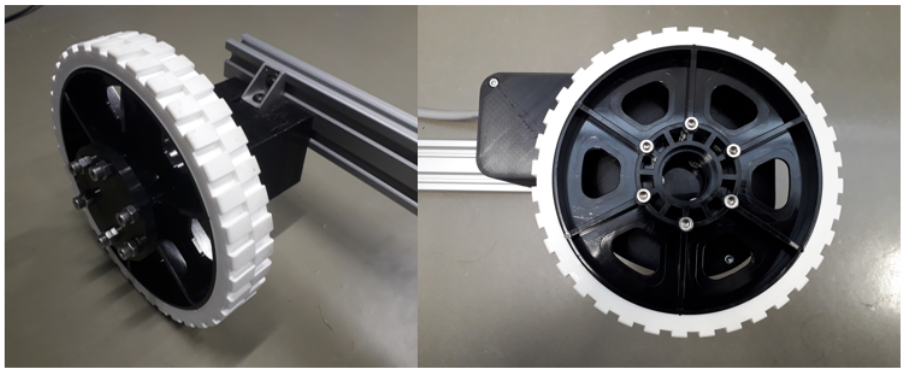


Figura 3.9: Rodas após a fixação à estrutura. Esquerda: roda livre. Direita: Roda acoplada ao motor.

Focando a atenção no eixo Y, o movimento da peça móvel é conseguido através de um motor de passo acoplado a uma polia que percorre uma correia fixa às duas extremidades do perfil, garantindo, assim, um movimento da ferramenta em toda a largura do robô.

O movimento do eixo Z é controlado também através de um motor de passo que está acoplado a um varão roscado. O movimento do motor permite o aparafusar e desaparafusar do varão roscado movendo o eixo Z para cima e para baixo. Este eixo desliza através de um conjunto de rodas que se encontram fixadas na placa móvel. A figura 3.10 mostra em detalhe os elementos responsáveis pelo movimento da ferramenta, observado de dois pontos de vista diferentes.

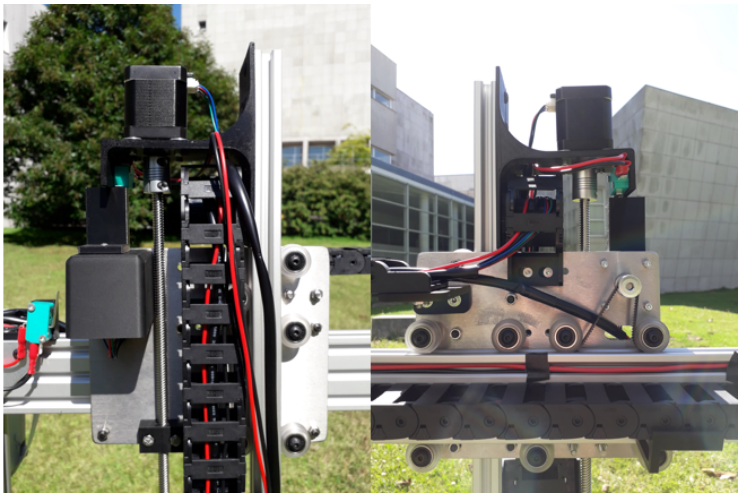


Figura 3.10: Elementos responsáveis pelo movimento nos eixos Y e Z

Um dos batentes que existe no Farmbot de forma a impedir o movimento para além dos limites, no eixo Y foi alterado, o que permitiu a fixação um detetor de fim de curso. O outro detetor de fim de curso, do eixo Z, foi colocado na peça que suporta o motor de passo que está acoplado ao varão roscado.

A câmara usada para a localização do robô através dos QR codes foi fixada por baixo da caixa de electrónica, assegurando a captura de uma faixa colocada sobre o solo e paralelamente ao movimento do robô. A figura 3.11 mostra a caixa criada para este efeito.

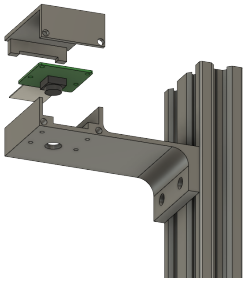


Figura 3.11: Caixa criada para a fixação da câmara à estrutura

Capítulo 4

Eletrónica

No presente capítulo é abordada a vertente de eletrónica que constitui o PixelCropBot. São apresentados os componentes necessários ao controlo, os atuadores e os sensores de todo o sistema, bem como as ligações estabelecidas entre si.

Primeiramente, pode ser observada na figura 4.1 a interligação entre os elementos de controlo, bem como como as suas ligações aos motores, câmara e fins-de-curso. O controlo do robô é conseguido com recurso a uma Raspberry Pi 4 conectada a uma AgIoT, hardware desenvolvido no INESC TEC que confere ao sistema a capacidade de atuação em potência dos motores. Adicionalmente, foi integrada no circuito uma nova PCB (Printed Circuit Board), desenhada com o fim de controlar individualmente os motores de passo, presentes nos eixos Y e Z.

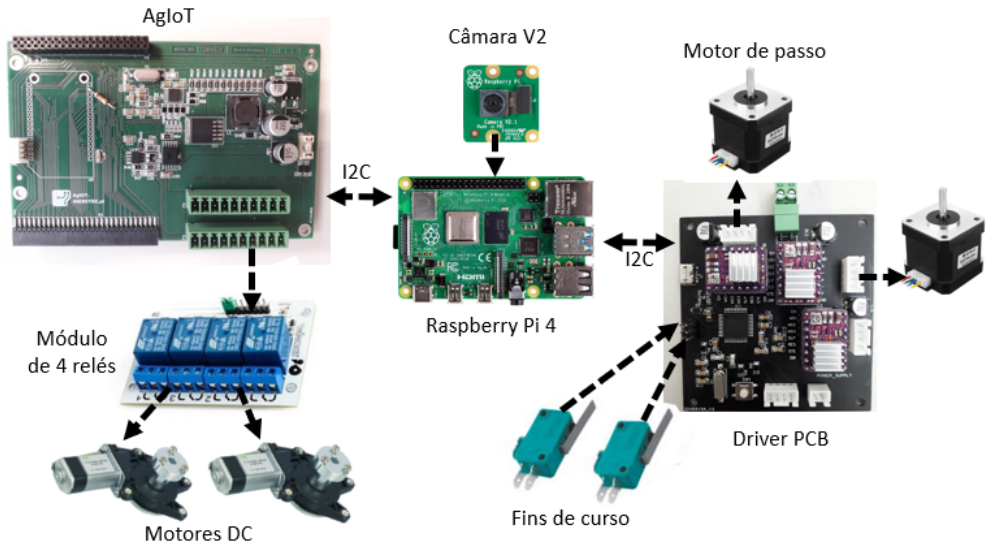


Figura 4.1: Visão geral dos componentes de controlo

AgIoT (IoT solution for Agrifood Sector)

A AgIoT[1] é uma solução modular e *open-source*, projetada e desenvolvida para satisfazer as necessidades tecnológicas do setor agro-alimentar. Além de apresentar a possibilidade de sensorizar e atuar sobre os componentes do sistema em tempo real (a partir de um ATmega32U4), é constituída por um circuito de potência, o qual recorre a um conjunto de MOSFET's com o objetivo de permitir a interrupção ou condução de corrente suficiente para fazer movimentar motores DC, atuar sobre válvulas, entre outros componentes. A AgIoT oferece ainda versatilidade ao utilizador na escolha do microprocessador a incorporar, podendo este diferir entre as várias vertentes existentes da Raspberry Pi e ainda num Omega 2, componente projetado e produzido pela Onion. No corrente projeto optou-se pela integração com uma Raspberry Pi 4 com o objetivo de utilizar uma câmara compatível com este dispositivo, bem como a utilização da arquitetura ROS2.

Motores de passo

Os motores de passo foram incluídos no robô para assegurar o movimento da ferramenta segundo os eixos Y e Z. Na expectativa de garantir um movimento preciso a baixo custo, a opção recaiu sobre os motores Nema 17. Estes possuem um passo de 1.8 graus, ou seja, 200 passos por revolução. O controlo dos motores foi efetuado com recurso ao driver DRV8825, o qual estabelece um limite de corrente pelo qual o motor será alimentado. Este driver permite que sejam usados *steps* mais pequenos (*microsteps*) de forma a se obter uma maior resolução no movimento. A resolução é controlada através de três pinos, sendo seis os modos existentes (full step, 1/2 step, 1/4 step, 1/8 step, 1/16 step e 1/32 step). Este driver é capaz de fornecer uma corrente até 1.5A, no entanto se for utilizado um dissipador de calor, este valor pode chegar até aproximadamente 2.2A. O valor de corrente limite imposta por este circuito integrado pode ser calculado em função da tensão medida entre o potenciómetro à massa do circuito. O valor limite será o dobro da tensão medida neste ponto.

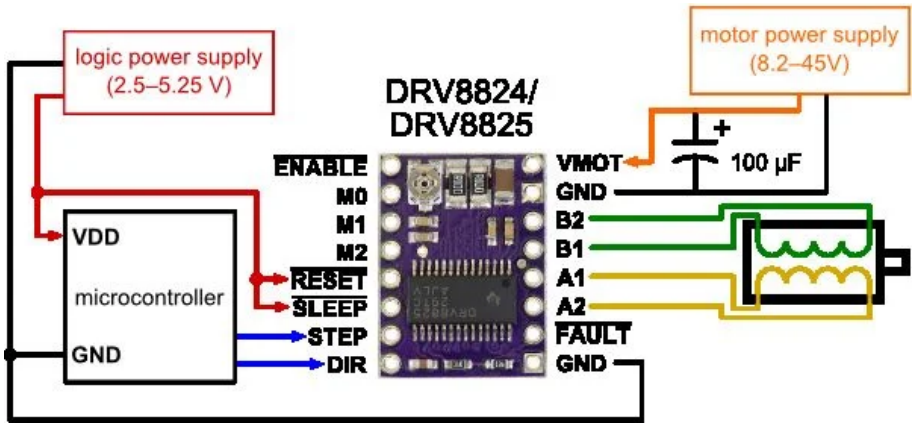


Figura 4.2: Esquema de ligações mínimas para controlo de um motor através de um DRV8825

Para o controlo dos motores de passo foi desenvolvida uma PCB capaz de integrar um microcontrolador ATmega32U4, responsável pelo controlo individual de três *drivers*. O circuito

[1] http://agiot.inesctec.pt/

permite o controlo de até 3 motores e mantém, em simultâneo, a possibilidade de comunicação com o microcontrolador através de SPI, USB, série e I2C. A comunicação SPI é usada inicialmente de forma a programar o *bootloader* no ATmega. A partir dessa altura, o ATmega pode ser programado através de um conetor micro USB. O microcontrolador está ligado aos pinos M0, M1 e M2 para permitir a utilização de *microsteps*, ao pino de *sleep* para que seja possível minimizar o consumo, ao pino STEP que deve receber informação sempre que seja necessário mover o motor e, finalmente, ao pino DIR que indica a direção do movimento.

O desenho do circuito e da PCB foi feito recorrendo ao software EasyEDA. O resultado final pode ser observado na figura 4.3. O desenho da PCB seguiu um processo iterativo e evolutivo. Em primeiro lugar, foi necessário reorganizar a localização dos componentes para que os condensadores ficassem mais perto dos circuitos integrados. De seguida, houve a alteração da *footprint* das resistências e condensadores para tornar possível a utilização de um tamanho de caixa 1206 e, assim, aproveitar material já existente no laboratório. Foi aumentada a largura das linhas que alimentam os motores para suportarem valores de corrente mais elevados e alteradas as regras de DRC (design rule check) para estas linhas. Evitou-se, ainda, a existência de curvas de 90º e foram criados planos de alimentação. A PCB foi criada com duas camadas e com quatro planos de alimentação (5V, GND, 12V, GND do motor).

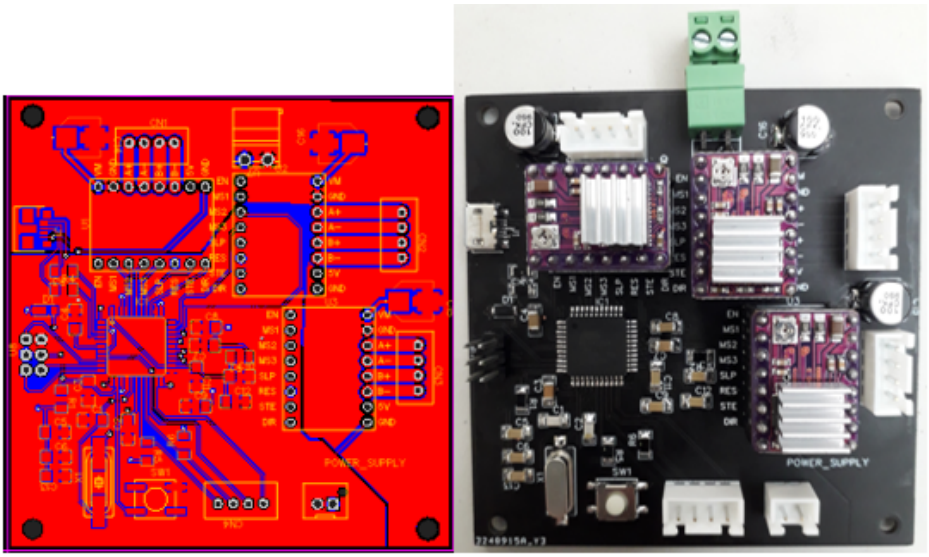


Figura 4.3: Esquerda: Design final da PCB desenhada no EasyEda. Direita: PCB com todos os componentes soldados

Para a integração de dois fins de curso, recorreu-se a pinos SPI, definidos com o propósito de serem utilizados como portas digitais. Os detetores são fundamentais para que seja possível detetar a posição limite dos eixos móveis Y e Z. Estes detetores foram posicionados na origem dos eixos. Desta forma, se a localização dos eixos for perdida ou acumular erro ao longo do tempo, executa-se o processo de inicialização dos eixos, fazendo com que estes se desloquem até aos detetores de fim de curso.

Motores DC

A estrutura é movida através de dois motores que estão acoplados a duas das quatros rodas da estrutura, sendo a mesma controlada como um robô móvel diferencial. O controlo destes motores é feito através de um sinal PWM, cujo valor permite ajustar a velocidade de rotação do motor. A partir do microcontrolador presente na AgIoT, é possível gerar o sinal que serve de comando à abertura e fecho do circuito de potência do motor. Quando o MOSFET entra em condução, é gerada uma diferença de potencial aos seus terminais. O período de tempo em que o motor é alimentado define a velocidade de rotação. O circuito pode ser observado na figura 4.4.

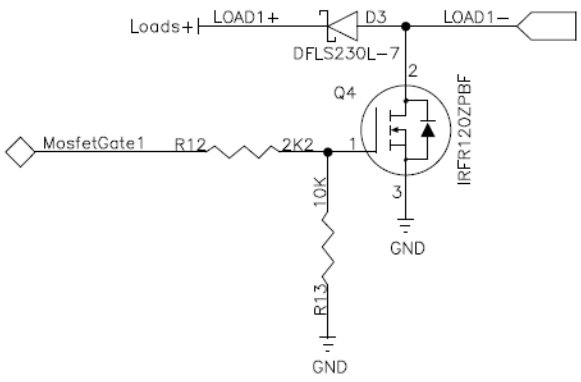


Figura 4.4: Circuito de potencia controlado por onda PWM presente na AgIoT

O circuito de controlo utilizado apresenta uma desvantagem para a sua utilização neste contexto, visto que permite a rotação do motor apenas num sentido. De forma a ultrapassar este problema, optou-se por usar um conjunto de relés em conjunto com o circuito de potência já mencionado, com o intuito de mudar a polaridade da alimentação do motor. Os terminais do motor encontram-se ligados ao terminal comum de dois reles diferentes, estando num relé o contacto normalmente fechado ligado ao load+ e no outro ligado ao load-. O sinal de controlo dos dois relés é o mesmo e aquando da sua ativação a polaridade da alimentação do motor é invertida. A ligação entre os motores e a AgIoT por intermédio do módulo de relés pode ser consultada na figura 4.6, presente no final deste capítulo. A distribuição da alimentação para as diferentes *boards* e motores é feita recorrendo a uma régua de bornes, sendo uma forma simples e organizada de criar um barramento.

Caixa de eletrónica

A caixa de eletrónica criada para este robô inclui a AgioT, uma Raspberry Pi, a PCB desenvolvida, o módulo de relés, uma régua de bornes e uma placa de prototipagem. Esta placa de prototipagem foi adicionada para realizar de forma simples o circuito envolvendo os dois fins de curso. O circuito é constituído por um conector para as ligações que alimentam os fins de curso, duas resistências de *pull-down* e ainda as ligações para os pinos da PCB. Desta caixa, saem as ligações para a bateria, os motores DC, os motores de passo, a câmara e os fins de curso. O

circuito é alimentado através de uma bateria recarregável de 12V. Foi criado um suporte que permitiu a fixação da PCB por cima da Raspberry Pi que, por sua vez, se encontra encaixada sobre a AgIoT. A figura 4.5 mostra os componentes presentes na caixa de eletrónica e as suas conexões. Posteriormente, a figura 4.6 mostra o esquemático das ligações existentes nesta caixa.

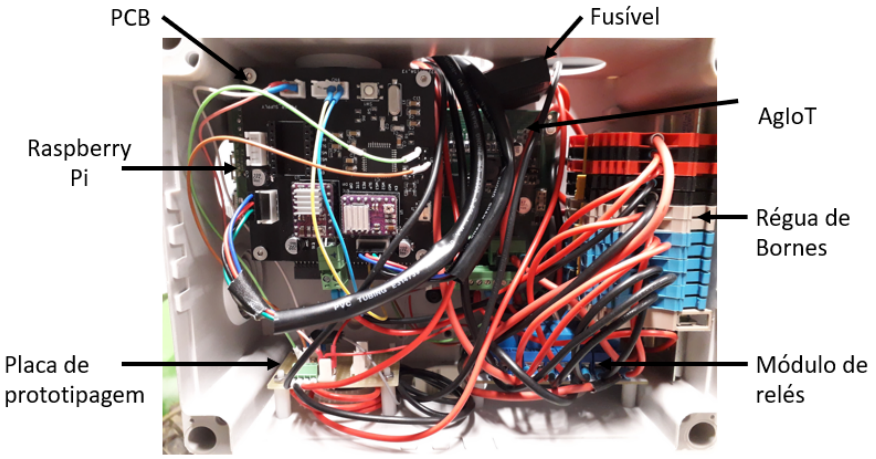


Figura 4.5: Interior da caixa de electrónica

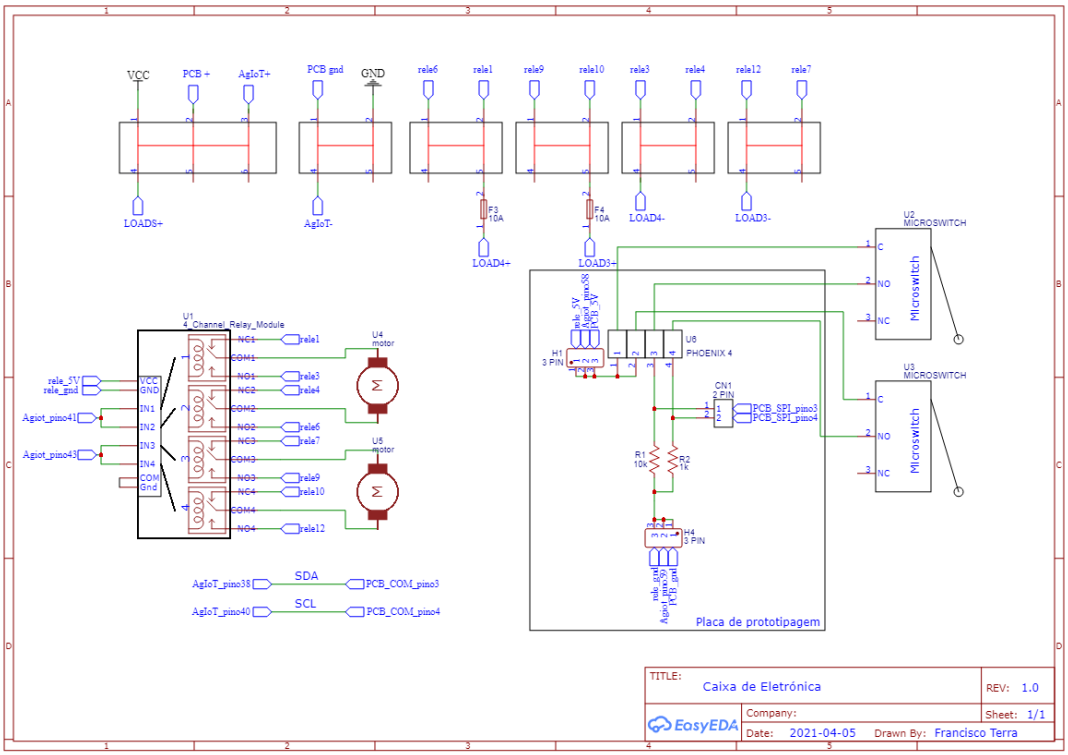


Figura 4.6: Ligações elétricas da caixa de electrónica

Capítulo 5

Software

As aplicações que visam o controlo da estrutura foram desenvolvidas tendo como base a arquitetura ROS2. Nesse seguimento e, considerando a representação em grafo dos processos em execução, foi necessário projetar um conjunto de nós, os quais estão interligados a partir da publicação e subscrição de tópicos ou através de serviços. Os nós foram desenvolvidos com recurso, principalmente, à linguagem de programação C++, com exceção do nó responsável pela deteção de QR codes, o qual foi escrito em Python. A figura 5.1 ilustra o modo como o sistema está organizado e também como os diferentes nós comunicam entre si. Ainda referente à utilização de ROS2, a distribuição Foxy foi escolhida e, posteriormente instalada no raspbian (Raspberry Pi OS).

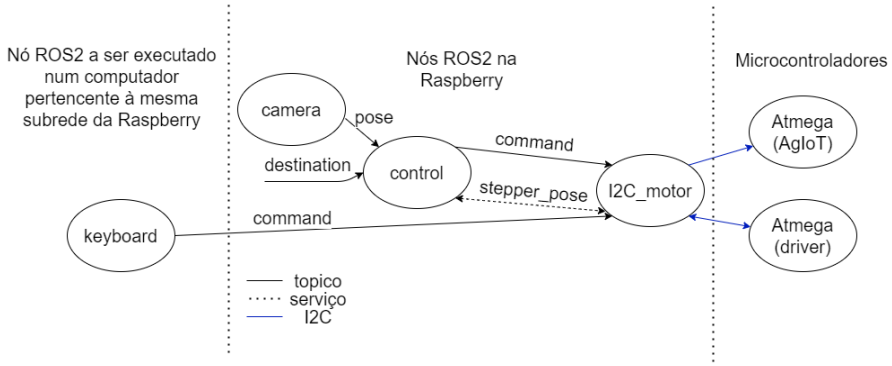


Figura 5.1: Comunicação entre todos os intervenientes do sistema

5.0.1 Nós ROS2

Keyboard

Com intuito de realizar testes ao movimento do robô e da ferramenta, de um modo manual, foi desenvolvida uma pequena aplicação que permitiu o controlo remoto da estrutura, através de um teclado. Os diferentes comandos que podem ser enviados para os microcontroladores (ATmega32U4), foram mapeados em diferentes teclas. Por exemplo, um clique sobre a tecla W, irá

provocar um movimento do robô no sentido positivo do eixo X. A forma como os diferentes movimentos foram mapeados pode ser conferida nas figuras 5.2 e 5.3. Qualquer um dos movimentos será cancelado no momento em que a ação sobre a tecla terminar. De forma a saber que tecla está clicada em cada instante foi usada a biblioteca conio.h[1]. A função *getch*, é uma função bloqueante, que retorna o código ASCII da tecla pressionada. De modo a evitar o congelamento da restante aplicação, recorreu-se ao uso de *threads*. Nesse seguimento, foi necessário medir, recorrentemente, a diferença entre o tempo atual e o último clique, definindo ainda um *threshold*, a partir do qual se saberá que a tecla foi solta.

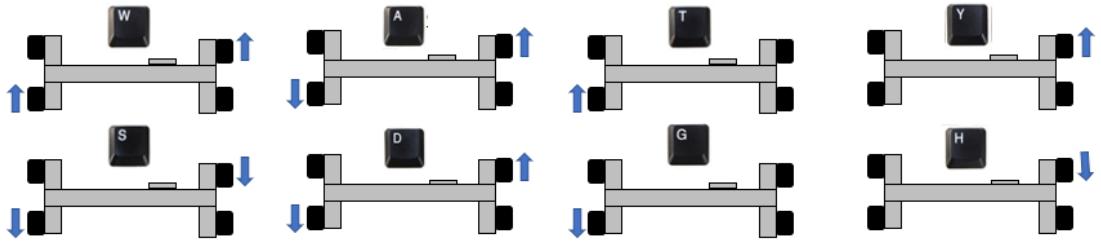


Figura 5.2: Conjunto de teclas que permitem o controlo dos motores DC da estrutura. O duty cycle do PWM do controlo dos motores irá ter sempre os mesmo valor (30%)

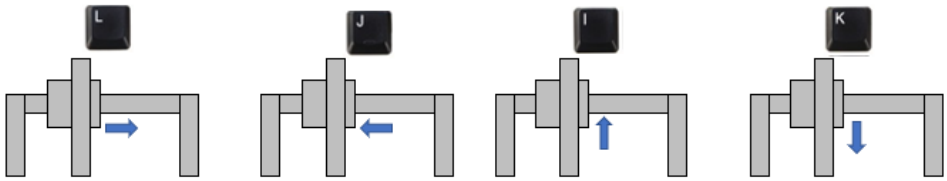


Figura 5.3: Teclas que permitem o movimento no eixo Y e Z

Sempre que uma tecla é pressionada ou solta é publicada uma instrução num tópico. Este será subscrito, e a informação textual será convertida em comandos e, posteriormente, comunicada, via I2C, para o microcontrolador. Este nó é executado num computador à parte, porque não funciona corretamente na Raspberry Pi (sempre que uma tecla era clicada o programa terminava a sua execução). A biblioteca usada depende do sistema operativo utilizado. Estando os dois dispositivos na mesma sub-rede, o nó I2C consegue receber a informação que é publicada no computador.

I2C

O nó I2C é responsável pela comunicação entre a Raspberry Pi e os dois microcontroladores (ATmega32U4). Este receberá instruções através do tópico *command* e reencaminhará o comando

[1] github.com/zoelabbb/conio.h

correspondente através de I2C. Para além da função já mencionada, possui ainda 2 *threads* cada um responsável por enviar uma mensagem periodicamente para os ATmegas. Se o ATmega perder a ligação com a Raspberry, ou seja, se passou um tempo superior ao estipulado entre recessões de mensagens I2C, o ATmega desligará todas as suas saídas como medida de segurança. De forma a impedir o possível envio de mensagens I2C por 2 *threads* em simultâneo, o excerto de código responsável pela comunicação (o recurso partilhado) encontra-se protegido por um *mutex*. Através desta partilha mutuamente exclusiva, garante-se, a cada momento, a existência de um único *slave* (microcontrolador ATMega) a comunicar com o *master*, evitando, assim, conflito de dados.

Dada a utilização de dois tipos de motores, DC e de passo, foram considerados dois formatos de mensagem distintos. No primeiro caso (motores DC), foram utilizados 3 bytes, o primeiro byte é utilizado para informar como é que os relés devem estar atracados (figura 5.4), o segundo representa o *duty cycle* que deverá ser utilizado na onda PWM que controla o motor da direita e o terceiro representa o *duty cycle* do motor da esquerda. A figura 5.5 mostra um exemplo de uma mensagem que pode ser enviada.

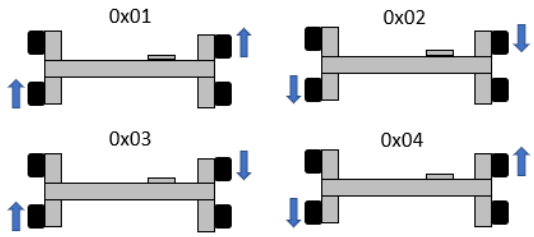


Figura 5.4: Códigos que indicam como é que será o movimento do robô

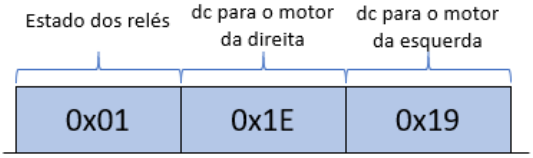


Figura 5.5: Exemplo de trama enviada para o ATmega. Neste caso, as duas rodas irão girar no sentido positivo (com valores de PWM de 30% e 25%, roda direita e esquerda respetivamente)

Por outro lado, as mensagens enviadas para o microcontrolador responsável pelo controlo dos motores de passo ocupam no máximo 5 bytes. O primeiro byte é utilizado para indicar a operação desejada (como por exemplo: movimento segundo um eixo, sequência de inicialização, entre outros), o segundo e o terceiro byte dizem respeito à velocidade a que os motores de passo devem rodar. Por fim, os últimos 2 bytes correspondem à posição até à qual o motor se deve mover. Os valores de velocidade e posição foram divididos em 2 bytes, com o objetivo de estender

a representação decimal de 255 para 65535. A figura 5.6 ilustra uma possível trama de dados que pode ser enviada para o controlo dos motores de passo por posição e velocidade.

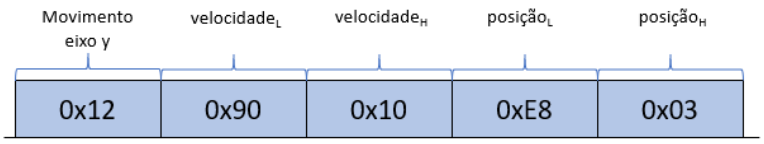


Figura 5.6: Trama de dados que permite o movimento do eixo y até à posição 1000 com uma velocidade de 400

Para saber a posição de cada motor de passo, foi utilizada uma rotina que permite a leitura da posição dos motores. Durante a implementação desta rotina, houve problemas em enviar e receber mais que um byte. A forma de contornar este pequeno problema de forma simples, mas não a mais eficaz, foi repetir o pedido de leitura 4 vezes.

Os dois primeiros bytes enviados pelo microcontrolador correspondem à posição do motor que controla o movimento no eixo Y e os dois últimos bytes correspondem ao motor do eixo Z. Esta rotina de leitura encontra-se presente na *thread* responsável pela comunicação periódica com o ATmega e é apenas ativa através do serviço stepper_pose. Até o motor chegar à posição pretendida, é efetuada uma leitura periódica da posição do motor. Quando este tiver chegado ao destino é enviada a resposta para o nó de controlo e é interrompido o processo de leitura. A figura 5.7 ilustra o processo acima detalhado.

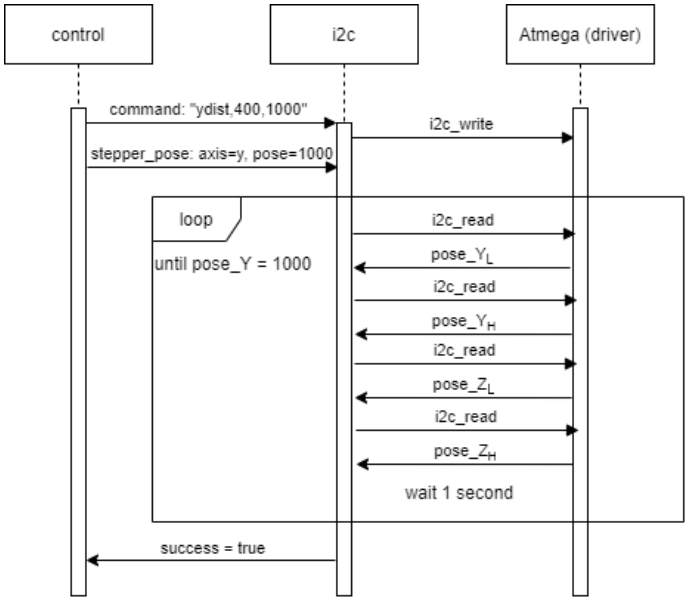


Figura 5.7: Sequência de controlo dos motores de passo

Câmara

O nó da câmara é responsável por detetar os QR codes existentes no chão e inferir a posição da câmara a partir da sua descodificação. O processo de obtenção da posição é referido no capitulo 6, sobre a localização. Os resultados (coordenadas X, Y e a orientação do robô) obtidos são publicados no tópico pose.

Control

O nó de controlo é responsável pelo controlo de mais alto nível do robô. Neste é estabelecida a lógica de movimento que irá ser responsável por conduzir a estrutura e a ferramenta de um ponto inicial até um ponto destino. No momento em que o nó é inicializado, são enviados comandos para os motores em Y e Z, de modo a proceder à inicialização dos respetivos eixos. Posteriormente, são recebidas as coordenadas de destino da ferramenta e são produzidos os comandos adequados ao movimento da estrutura, sendo este dividido em três fases distintas. A primeira consiste em usar a informação cedida pela câmara para fazer com que o robô se mova para a frente ou para trás de forma a alcançar a coordenada pretendida no eixo X. É de realçar que, neste momento, o movimento do robô carece de qualquer tipo de controlador através do qual será possível controlar a sua velocidade, corrigir a orientação e possíveis desalinhamentos, garantindo o bom funcionamento, num regime periódico. A próxima etapa foi implementada partindo do princípio que esta carência irá ser suprimida futuramente. Desse modo, as implementações posteriores assumem que o robô está alinhado com a horta. No que respeita à segunda fase do movimento, a posição em *steps* é calculada através da equação 5.1. De seguida é comunicada, via I2C, para o microcontrolador, o qual dará ordem de comando aos motores. O sentido da deslocação é determinado pela diferença entre a posição atual da ferramenta e a sua posição de destino. A figura 5.8 mostra a representação da variável $offset_y$, que corresponde à distância entre a câmara e a ferramenta quando esta se encontra na posição inicial. A constante K corresponde à distancia em milímetros percorrida entre *steps*.

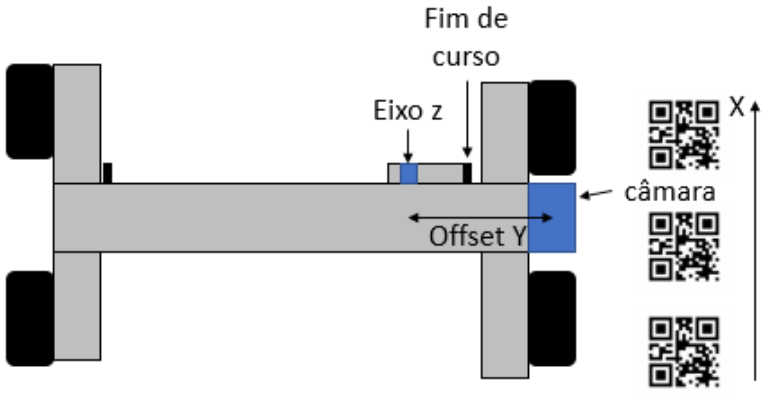


Figura 5.8: Representação do robô após a sequência de inicialização (vista de cima)

$$steps_y = \frac{goal_y - (pose_y + offset_y)}{K_y} \tag{5.1}$$

Se o valor dos *steps* calculado for inferior a 0 ou superior ao número máximo de *steps* permitidos em cada um dos eixos, o movimento é cancelado. Nesta situação é emitido um aviso a dar conhecimento de que não é possível alcançar a posição desejada. Por outro lado, se o movimento for pré-estabelecido com sucesso, a informação sobre a posição e a velocidade é publicada no tópico command, sendo posteriormente usado o serviço stepper_pose com o intuito de obter a informação de quando o movimento for concluído. O processo de posicionamento da ferramenta segundo o eixo Z é semelhante ao explicado para o eixo Y. Após a ferramenta estar posicionada no local pretendido, o nó permanece num ciclo de espera por novas coordenadas da ferramenta. É importante realçar que, sendo os QR codes a única informação de localização disponível no robô, se esta, porventura, falhar durante um período maior que um intervalo de tempo previamente definido, é gerado um comando para cessar o movimento no eixo X.

5.0.2 Microcontroladores

ATmega (driver)

O controlo dos motores de passo é feito recorrendo à execução de duas máquinas de estado para cada um deles. As máquinas de estados implementadas para este propósito estão representadas na figura 5.9.

A máquina de estados da esquerda é usada para controlar o movimento do motor e nela são considerados três estados distintos: paragem, inicialização do motor (fazendo-o girar no sentido negativo até atingir a posição zero dada pelo fim de curso) e o movimento do mesmo para a posição desejada com a velocidade pretendida. A informação sobre a posição de destino e a velocidade do movimento são adquiridas através das mensagens enviadas por I2C. Após a recessão destes dados é ativada uma *flag* para que a transição entre estados aconteça durante o ciclo em que o programa principal está a ser executado. Quando o controlo é feito através do teclado é apenas enviado um comando com a informação da velocidade. A variável distância ficará com o valor max_pose que corresponde ao valor máximo que o motor de passo deverá alcançar. O movimento neste caso é interrompido através da mensagem I2C que sinaliza a inativação da *flag*.

A outra máquina de estado utilizada tem como objetivo sinalizar a existência de qualquer anomalia no sistema, como por exemplo, o alcance dos limites na área de atuação da estrutura, dado pelo sinal de saída dos fim de curso presentes em cada eixo. Adicionalmente, o estado de emergência pode ainda ser alcançado quando a comunicação com a Raspberry Pi for interrompida durante um período maior do que um tempo pré-definido. Como previamente mencionado, também é efetuada a resposta a pedidos de leitura I2C sendo enviados as posições atuais dos dois motores de passo.

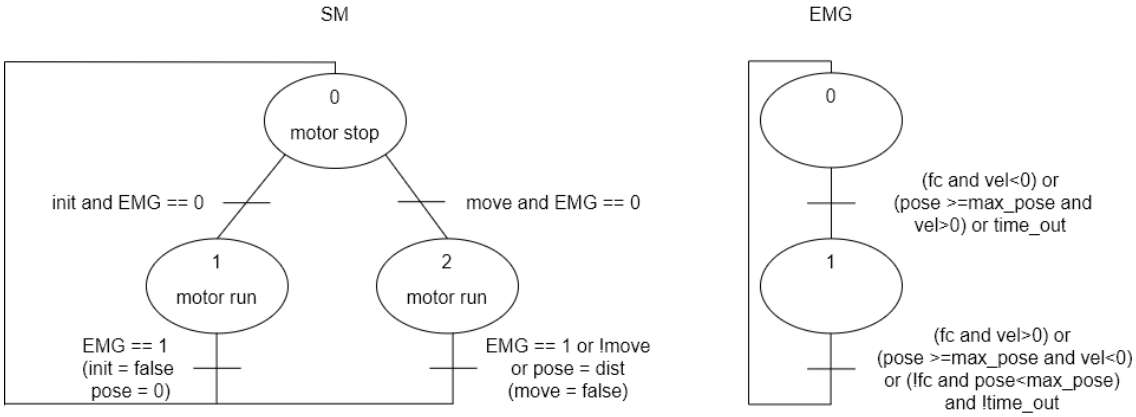


Figura 5.9: Máquinas de estados usadas para o controlo do motor de passo. Esquerda: máquina de estado responsável pelo movimento dos eixos. Direita: máquina de estados que sinaliza as anomalias

Os dois motores estão a trabalhar a 1/4 de *step*, sendo que este valor foi escolhido de forma a alcançar um movimento com maior resolução e ainda uma maior suavidade no deslocamento. Com os motores a funcionar com um *step* completo, a estrutura apresentava uma maior oscilação durante o movimento, causando erros de alguns milímetros na estimação da posição, a qual é determinada apenas pela contagem do número de *steps*. Com o motor a funcionar com um 1/4 de *step* este problema é atenuado. O erro na estimação da posição, que se verifica ser cumulativo no tempo, pode ser resolvido, periodicamente, fazendo a ferramenta voltar à posição zero (posição do fim de curso).

ATmega (AgIoT)

Este microcontrolador recebe o valor do *duty cycle* de ambos os motores, bem como o código que identifica o estado dos relés. Verificando a transição do código dos relés face ao atual, é possível detetar a necessidade de inversão dos motores. Nesse momento, o estado dos relés é atualizado e, em simultâneo, é congelada a execução do movimento do robô, com o intuito de assegurar que a comutação é concluída. Após este processo, as saídas PWM são atualizadas para refletir os novos valores de *duty cycle*. Não havendo necessidade de alterar a disposição dos relés, a atualização dos valores do *duty cycle* é realizada imediatamente. Como no caso previamente referido, se a comunicação com a Raspberry Pi for interrompida, o movimento dos dois motores é cancelado por segurança.

Capítulo 6

Localização

Neste capítulo será abordado o sistema de localização implementado no PixelCropBot, sendo apresentada uma pequena introdução sobre a utilização de QR codes para a localização de robôs móveis, a explicação do algoritmo implementado e os resultados obtidos.

6.1 Introdução

Para que seja possível calcular e realizar trajetórias corretamente e de forma precisa, é necessário conseguir estimar a posição do robô. Para se obter esta posição pode-se recorrer a diferentes métodos como por exemplo GPS ou utilização de *landmarks* [19].

Um dos principais desafios na área dos robôs móveis passa por encontrar uma solução eficiente e com uma boa relação de custo-benefício, que permita a localização precisa do robô. A utilização de QR codes é um método baseado em visão que utiliza *landmarks* artificiais e processamento de imagem. Estas *landmarks* são colocadas estrategicamente de forma a servirem como pontos de referência globais, facilitando o processo de localização.

A utilização de landmarks artificiais é feita recorrendo a imagens, formas, cores ou códigos únicos e facilmente reconhecíveis para facilitar o processo de visão computacional. Em comparação com as *landmarks* naturais, estas são mais fáceis de detetar, o que torna o processo mais rápido. No entanto, existe um maior esforço para planear e colocar as respetivas *landmarks* no terreno [21].

QR code

O QR Code é uma tecnologia que evoluiu do código de barras. Surgiu em 1994, criado pela empresa Denso-Wave, a pedido da indústria automóvel Japonesa, que pretendia um sistema melhor para catalogar as peças nas linhas de produção. Em oposição ao código de barras que apenas guarda informação segundo uma direção, o QR Code guarda informação segundo duas dimensões, permitindo um maior armazenamento da mesma[23].

Quando os QR codes são examinados estruturalmente é possível notar que cada código apresenta três padrões diferentes: deteção, alinhamento e *timing* (figura 6.1). O padrão de deteção encontra-se localizado em três dos quatro cantos, o que permite uma rápida leitura do código em

qualquer direção. O padrão de alinhamento permite a correção do QR code se este estiver curvado. O padrão de timing é composto por duas linhas que unem os três padrões de localização, sendo constituída por pontos pretos e brancos, alternadamente. Desta forma é possível ter uma noção da localização de cada célula de informação individualmente[24].

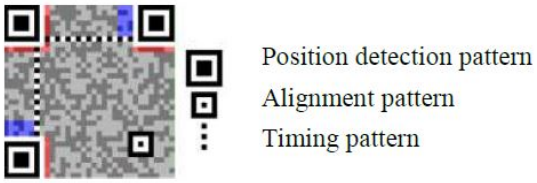


Figura 6.1: Elementos que compõem o QR code

6.2 Implementação

A análise e manipulação das imagens foi feita recorrendo à biblioteca OpenCV[1] e a informação contida nos QR codes foi descodificada usando a biblioteca pyZbar[2].

Pré-processamento

O primeiro passo, consiste em converter a imagem original numa imagem em *grayscale*, de forma a simplificar o processamento e reduzir a quantidade de informação analisada. De seguida é aplicado um filtro gaussiano à imagem para reduzir o ruído que possa existir.

De forma a obter os contornos dos QR codes foram testados alguns métodos de binarização da imagem. Inicialmente, foi testado o método de binariazção Otsu, que define um valor de *threshold* global, que não é muito eficaz se existirem sombras sobre o QR code. Este entrave, já tinha sido verificado e mencionado noutro trabalho[25], pelo que a alternativa usada foi o *Mean Adaptive Threshold*. Adicionalmente foi testado o uso do Guassian Adaptive Threshold e o método de Canny para a deteção dos contornos. O resultado da aplicação dos diferentes métodos pode ser visualizada na figura 6.2. O *Mean Adaptive Threshold*, foi o método capaz de identificar um maior número de QR codes, pelo que foi o escolhido.

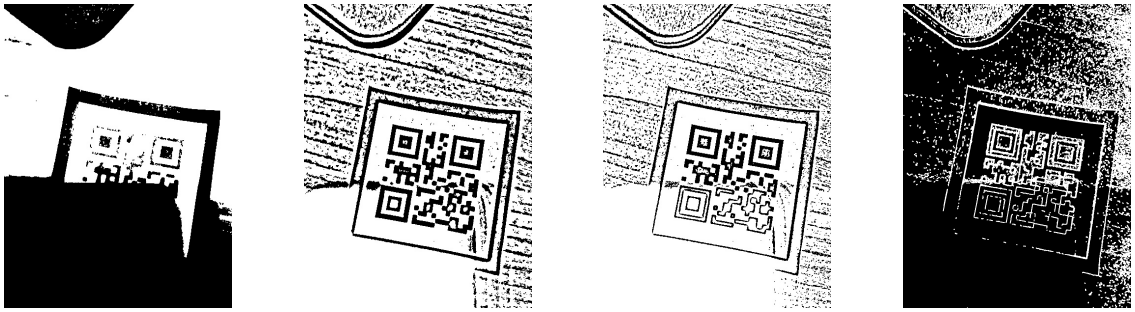


Figura 6.2: Resultados da aplicação dos diferentes processos de binarização: Otsu, *Mean*, *Guassian* e Canny (da esquerda para a direita)

[1] opencv.org
[2] pypi.org/project/pyzbar

Utilizando a função *findContours* é possível obter todos os contornos existentes na imagem. Recorrendo à função *ApproxPolyDP* foi possível aproximar os contornos obtidos em figuras com menos vértices. A implementação deste processo é baseada no algoritmo Douglas-Peucker. A partir deste ponto só são considerados os polígonos que apresentam quatro lados, um *aspect ratio* perto de um e uma área dentro de um valor máximo e mínimo, que dependem da área esperada do QR code.

Os contornos que cumprirem estes requisitos são guardados, sendo que se existirem dois contornos cujo centro seja muito próximo, guarda-se apenas o de menor área. Isto acontece porque cada QR code poderá gerar dois contornos, um entre o ambiente onde está colocado e a moldura preta e o outro entre esta e o fundo do código, que é branco. Ao ser escolhido o contorno com menor área, ou seja, o contorno interior, não se considera a moldura preta para as restantes fases do processo. Esta moldura foi colocada para aumentar a eficácia da deteção dos contornos e tornar evidente a transição entre o preto e o branco.



Figura 6.3: Contornos obtidos, a vermelho, após a fase de pré-processamento

Obtenção da região de interesse

O próximo passo consiste em corrigir a perspetiva do polígono obtido, gerando-se assim, a região de interesse em forma de um quadrado. É necessário analisar os pontos deste polígono de forma a conseguir identificar os pontos que corresponderão a cada vértice deste quadrado. Por exemplo, o canto superior esquerdo corresponderá ao ponto do polígono cuja a soma das suas coordenadas for menor. As Equações 6.1 - 6.4 mostram como é que é associado cada ponto do polígono a cada vértice do quadrado que se espera obter com esta transformação.

$$\text{canto superior esquerdo} = ponto_{min(x+y)} \tag{6.1}$$

$$\text{canto superior direito} = ponto_{max(x-y)} \tag{6.2}$$

$$\text{canto inferior esquerdo} = ponto_{min(x-y)} \tag{6.3}$$

$$\text{canto inferior esquerdo} = ponto_{max(x+y)} \tag{6.4}$$

Com esta correlação entre os pontos é possível criar a matriz de transformação recorrendo à função *getPerspectiveTransform*. Esta matriz, juntamente com a imagem são fornecidos como parâmetros à função *warpPerspective*, cujo resultado será a região de interesse (ROI). Na figura 6.4 é apresentado o resultado obtido no final deste processo.



Figura 6.4: Região de interesse obtida

Orientação

O passo seguinte tem como objetivo descobrir a orientação do QR code, permitindo assim obter a informação sobre o referencial global. A região de interesse é dividida em quatro quadrantes, sendo realizado um processo semelhante ao da fase de pré-processamento com o intuito de procurar a existência de dois quadrados concêntricos. Na versão inicial era necessário a identificação de três quadrados, no entanto este requisito foi diminuído para dois, visto que os resultados do processamento de imagens com sombra apresentam resultados piores. Com base nesta informação é preenchido um vetor de orientação, com o valor um se forem encontrados os dois quadrados ou com o valor zero no caso contrário. Os possíveis vetores de orientação estão exemplificados na figura 6.5.



Figura 6.5: Relaçaõ entre o ROI e o vetor orientação

Após a caracterização da orientação do QR code, os vértices do polígono identificados na imagem são reordenados de forma a seguirem o padrão representado na figura 6.6.



Figura 6.6: Padrão que mostra como é que os vértices devem ser ordenados

Após a identificação dos vértices do QR code é possível definir os vetores que caracterizam o referencial global. As equações 6.5 e 6.6 mostram como é calculado o vetor que define o eixo X e Y, respetivamente. O resultado deste passo é mostrado na figura 6.7.

$$\vec{vx} = (corner_3.x - corner_2.x, corner_3.y - corner_2.y) \tag{6.5}$$

$$\vec{vy} = (corner_0.x - corner_2.x, corner_0.y - corner_2.y) \tag{6.6}$$

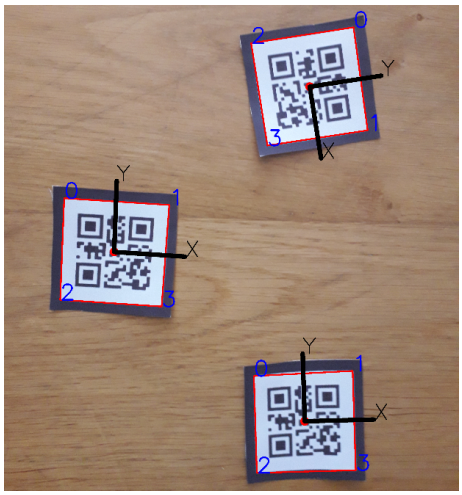


Figura 6.7: QR codes detetados com os vértices numerados segundo o padrão e com os eixos representados

Posição

Um dos passos necessários para determinar a posição da câmara, passa por calcular a posição do QR code ao centro da imagem no referencial global. O primeiro passo consiste em calcular o vetor que tem origem no centro do QR code e a extremidade no centro da imagem. Este vetor é calculado através da expressão presente na equação 6.7.

$$\vec{L} = (center_image.x - center_QR.x, center_image.y - center_QR.y) \tag{6.7}$$

As equações 6.8 - 6.10 mostram o procedimento para a escrita deste vetor na base do referencial global.

$$angle = arctan2(vx_1, vx_0) - arctan(L_1, L_0) \tag{6.8}$$

$$L_{distance} = \sqrt{L_0^2 + L_1^2} \tag{6.9}$$

$$\vec{L}_{global} = (L_{distance} * cos(angle), L_{distance} * sin(angle)) \tag{6.10}$$

Este vetor está representado na figura 6.8, bem como a informação sobre o ângulo formada entre o vetor $\vec{L}$ e o eixo X.



Figura 6.8: Vetor L, a vermelho, e informação sobre o ângulo deste vetor com o eixo x

Este vetor é posteriormente multiplicado por K, uma constante que correlaciona a distância em píxeis com a distância em metros. Esta correlação é realizada através da medida do lado do QR code, que é conhecida e constante, com a sua distância em píxeis. Para este último valor é utilizado a média dos quatro lados do quadrado.

$$d_1 = \sqrt{(corner_0.x - corner_1.x)^2 + (corner_0.y - corner_1.y)^2} \tag{6.11}$$

$$d_2 = \sqrt{(corner_0.x - corner_2.x)^2 + (corner_0.y - corner_2.y)^2} \tag{6.12}$$

$$d_3 = \sqrt{(corner_3.x - corner_1.x)^2 + (corner_3.y - corner_1.y)^2} \tag{6.13}$$

$$d_4 = \sqrt{(corner_2.x - corner_3.x)^2 + (corner_2.y - corner_3.y)^2} \tag{6.14}$$

$$d_{avg} = \frac{d_1 + d_2 + d_3 + d_4}{4} \tag{6.15}$$

$$K = \frac{QR_code_l}{d_{avg}} \tag{6.16}$$

A informação sobre as coordenadas do QR code está guardada no mesmo e são obtidas através da aplicação da função *decode* nas diferentes regiões de interesse obtidas. A figura 6.9 mostra o resultado desta etapa. Sendo a posição da câmara calculada através da expressão 6.17

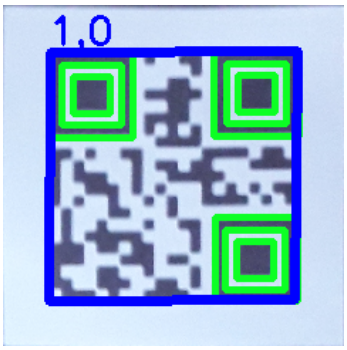


Figura 6.9: Região de interesse após terem sido identificados os cantos e a azul o resultado descodificado

$$estimated_pose = QR_pose + K * \vec{L}_{global} \tag{6.17}$$

Devido à existência de sombras, a função *decode* pode não ser capaz de descodificar o QR code. Neste caso é feita a binarização da zona de interesse recorrendo à função *Guassian Adaptive Threshold*. Esta nova imagem será passada à função *decode*. Este processo permite a deteção de QR codes que doutra forma não seria possível, mas não permite uma deteção infalível dos mesmos.

Orientação do robô

A orientação do robô é calculada através do ângulo entre o eixo X global e o eixo X da imagem, como mostrado na equação 6.18

$$theta = arctan2(vx_1, vx_2) - arctan2(0,1) \tag{6.18}$$

Se houver mais que uma região de interesse o resultado da posição estimada e da orientação é a média dos resultados obtidos para cada uma.

6.3 Resultados

A implementação do algoritmo foi validada com a utilização de duas câmaras diferentes: a Raspberry Pi Camera Module V2 e a Raspberry Pi High Quality Camera. Com estes testes pretende-se obter a distância máxima na qual ainda é possível descodificar a informação dos QR codes, e também o erro associado à posição estimada com a utilização de um ou três QR codes.

Os testes foram realizados recorrendo a uma plataforma móvel, onde foram fixados os QR codes, que pode ser observada na figura 6.10. Esta plataforma foi sendo afastada da câmara até ao limiar onde ainda era possível descodificar a informação. Nesta distância máxima, o centro da placa (com coordenadas (0,0)) foi alinhado com o centro da imagem. Na figura 6.11 observa-se a informação sobre a coordenada estimada para a câmara, no centro da imagem, e no canto superior direito, a orientação da mesma. Sobre cada QR code é apresentado o valor das coordenadas, em metros.



Figura 6.10: Equipamento de teste

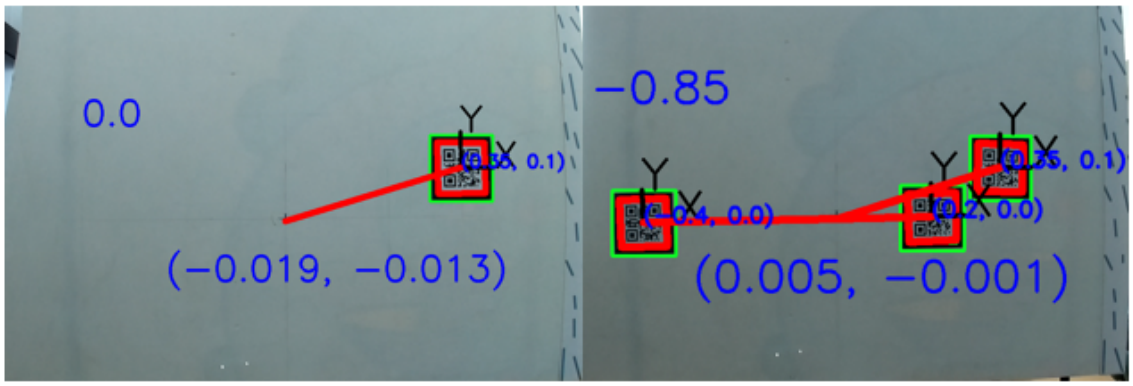


Figura 6.11: Exemplo do resultado obtido: ângulo em graus no canto superior direito e as coordenadas da câmara abaixo do ponto central

Os resultados estão presentes na tabela 6.1. As resoluções testadas foram 1920x1080, 1280x720 e 640x480. Na câmara V2 a definição mais baixa não foi documentada, visto que obteve resultados muito insatisfatórios.

Tabela 6.1: Resultados do algoritmo de deteção dos QR codes

Camara e resoluções	Dimensão dos QR codes (cm)					
	7.5	8.5	9.5	10.6	11.7	12.8
High Quality Camera 1920X1080						
distância máxima / m	1.54	1.86	2.05	2.25	2.52	3.02
erro (1 QR code) / cm	2.2	2.6	2.2	2.1	1.7	1.7
erro (3 QR codes) / cm	0.7	0.6	0.3	0.1	0.6	0.5
High Quality Camera 1280X720						
distância máxima / m	1.06	1.31	1.58	1.82	1.95	2.19
erro (1 QR code) / cm	1.6	2.1	2.6	1.9	1.6	2.1
erro (3 QR codes) / cm	0.8	1.0	0.5	0.5	0.4	0.7
High Quality Camera 640X480						
distância máxima / m	0.66	0.75	0.85	0.92	1.00	1.12
erro (1 QR code) / cm	2.2	3.1	3.2	3.6	3.6	3.7
erro 3 (QR codes) / cm	1.7	2.1	2.2	2.4	2.2	2.2
V2 Camera 1920x1080						
distância máxima / m	1.71	1.98	2.21	2.38	2.61	2.90
erro (1 QR code) / cm	1.8	1.5	2.2	1.9	1.7	1.4
erro (3 QR codes) / cm	0.8	0.8	0.9	1.0	0.6	0.6
V2 Camera 1280x720						
distância máxima / m	0.86	1.16	1.29	1.42	1.56	1.74
erro (1 QR code) / cm	1.7	1.9	1.5	1.7	1.4	1.7
erro (3 QR codes) / cm	0.9	0.8	0.6	0.9	0.9	0.7

A câmara V2 apresenta melhor resultados do que a High Quality camera na resolução 1920x1080, no entanto, é importante notar que o campo de visão desta câmara é menor nesta resolução. O campo de visão observável para as diferentes câmaras encontra-se representado na figura 6.12



Figura 6.12: Da esquerda para a direita e de cima para baixo: High Qualitycamera, camera V2 1280x720 e camera V2 1920x1080

O tempo de processamento é maior quanto maior for a resolução, por isso, apesar de uma maior resolução permitir o uso de QR codes mais pequenos, pode ser necessário optar por uma

resolução mais baixa de forma a permitir melhores resultados de localização em tempo real. Com a resolução mais alta obteve-se leituras a uma taxa de 2.07 FPS(frames per second), enquanto que a 1280x720 esse valor duplica, obtendo-se 5.12 FPS. Através dos resultados obtidos é possível constatar que o erro nas medições é semelhante para todos os casos, com exceção das medições com baixa resolução. O erro diminui significativamente se estiverem presentes no campo de visão da câmara mais do que um QR code.

Pelos motivos apresentados anteriormente, a câmara utilizada foi a câmara V2, privilegiando uma solução mais económica. A resolução escolhida foi de 1280x720, o que permitiu uma velocidade maior, sendo necessário a utilização de QR codes maiores. Como a câmara está afixada a uma altura de 92 cm, com base nos testes realizados, podem ser utilizados QR codes com tamanho igual ou superior a 8.5cm. Foi escolhido o valor de 9.5cm, não o valor mínimo, com o objetivo de tornar o sistema de deteção mais robusto.

Capítulo 7

Testes

Neste capítulo, são efetuadas algumas considerações relativamente aos testes que foram realizados, bem como a alguns resultados obtidos.

Os testes realizados durante a construção do robô seguiram uma abordagem *bottom-up*. Os componentes que constituem a parte de eletrónica foram testados separadamente. A sua integração no sistema foi feita após a verificação do funcionamento destas partes independentes. Estes testes individuais serviram para validar o controlo dos motores DC e de passo, bem como o módulo da câmara, cujos resultados estão presentes no capitulo 6. Após a verificação do funcionamento dos diferentes subsistemas e o processo de comunicação entre os mesmos, começaram a ser realizados os testes ao funcionamento do robô.

Os primeiros testes efetuados na estrutura, consistiram em averiguar o erro existente no posicionamento da ferramenta ao longo do eixo Y e Z. Durante estes testes, foi estudada a velocidade de deslocação destes eixos, bem como a utilização de *micro-stepping*. Como mencionado previamente foi utilizado 1/4 de *step* de forma a tornar o movimento mais suave e minimizar as oscilações causadas na estrutura. A velocidade utilizada para que o erro da estimação da posição baseado na contagem de *steps* seja fiável foi 2 cm/s. Este eixo percorre uma distância de 1,182m.

O movimento no eixo Z é feito a uma velocidade de 0.4 cm/s. Os limites de altura que este eixo pode atingir, estão por definir, visto que vai depender da ferramenta e de como esta será fixada. A velocidade neste eixo teve que ser reduzida para minimizar a oscilação do varão roscado. Esta velocidade poderá ser aumentada, criando um suporte que conecte o varão roscado ao perfil do eixo Z, aumentando assim a sua estabilidade. Nestes testes de posicionamento, o ponto escolhido para representar a ferramenta foi o centro da face inferior do perfil de alumínio do eixo Z. Os testes foram realizados num corredor, sendo o movimento do robô efetuado apenas para a frente e para trás, ou seja, no sentido positivo e negativo do eixo X. O local onde os testes foram realizados encontra-se representado na figura 7.1.



Figura 7.1: Local onde os testes foram realizados

Os testes finais foram realizados utilizando o robô e uma tira de papel contendo vários QR codes para a localização do robô. A tira de papel apresenta QR codes de 20cm em 20cm colocados ao longo do eixo X. Este espaçamento entre os QR codes foi escolhido de forma a existirem sempre 3 QR codes no campo de visão da câmara, o que melhorou os resultados obtidos como mencionado no capítulo 6 . A figura 7.2 exemplifica o cenário usado durante o teste.

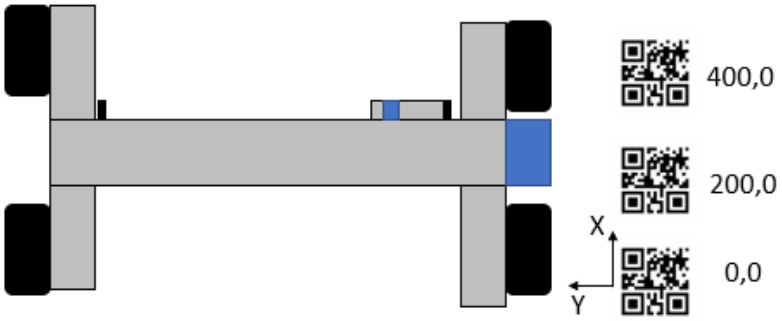


Figura 7.2: Representação do cenário usado durante os testes

A figura 7.3 mostra os QR codes reconhecidos durante o movimento do robô. Esta imagem inclui as coordenadas da câmara representadas no centro da imagem e orientação do robô no canto superior direito. Os valores das coordenadas estão expressos em milímetros e o ângulo em graus.

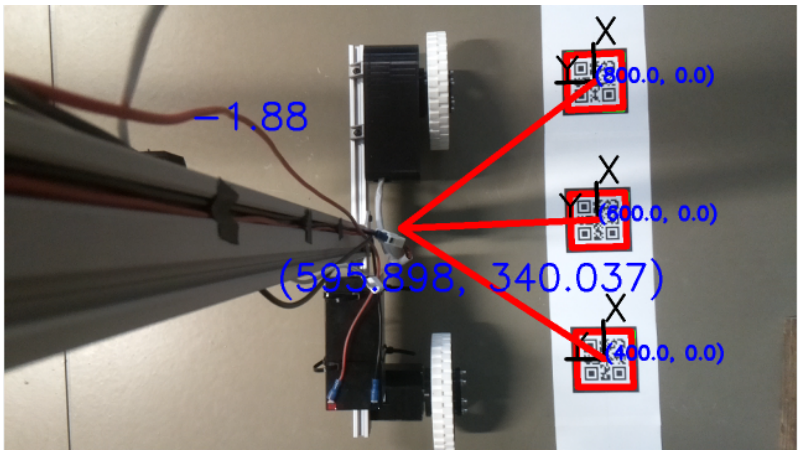


Figura 7.3: Imagem obtida durante o processo de localização

Estes testes serviram para perceber o erro existente no movimento ao longo do eixo X e a velocidade máxima a que o robô pode andar de forma a conseguir detetar os QR code. A velocidade máxima que o robô pode andar para este fim é de 7.67 cm/s, o que se traduz num *duty cycle* de 23%. No entanto usar este valor de *duty cycle* faz com que os motores tenham dificuldade em mover a estrutura, ficando o robô por vezes parado a meio do seu percurso. Dos testes realizados quando o robô conseguiu concluir o percurso sem problemas o valor de erro da posição final variou entre 1cm e 3cm. Controlando o robô através do teclado este problema não é manifestado, visto que é usado um valor de *duty cycle* mais elevado. Apesar dos erros presentes no eixo X não serem muito grandes, o robô não estaria preparado para constantes movimentações, visto que não houve correção da orientação. Este processo de correção da orientação foi efetuado manualmente, entre ensaios. Estes testes foram realizados publicando as coordenadas de destino da ferramenta no tópico *destination* através da linha de comandos.

Capítulo 8

Conclusão e Trabalho futuro

8.1 Conclusão

O robô construído no âmbito desta dissertação ainda não está apto para ser utilizado no contexto agrícola. No entanto foi possível cumprir os objetivos de criar uma solução totalmente *open-source* e de baixo custo. O repositório público criado para este trabalho no GitLab[1] alberga os ficheiros CAD das peças criadas, os esquemas das ligações elétricas, a lista de material e o software desenvolvido. O custo do PixelCropBot neste momento ronda os 621€, conforme indicado na figura B.1 presente no apêndice B.

Todavia é importante ressalvar a ausência de ferramentas para plantação, rega e remoção de infestantes que outras soluções apresentam, como é o caso do Farmbot. De forma a tornar o robô útil para o domínio agrícola, o preço do robô ainda aumentará com a necessidade da instalação de novo Hardware para que seja possível efetuar as tarefas acima mencionadas. Por outro lado, será importante rever a estrutura do robô de forma a torná-la mais robusta às oscilações existentes aquando do movimento do robô, o que também poderá influenciar o preço.

Os testes realizados serviram como prova de conceito para esta evolução do Farmbot, no que diz respeito à liberdade de movimento. A localização através de QR codes mostrou-se um método eficaz, no entanto é necessário ter em atenção o contexto onde o robô será utilizado, visto que o algoritmo de deteção não é independente das condições de iluminação. É também necessário fazer uma intervenção no local onde será instalado o robô para a colocação de uma tira de QR codes ao longo de horta que este irá cuidar. A velocidade de movimento segundo os eixos Y e Z é lenta, devido ao facto de se ter tentado alcançar uma solução de baixo custo, eliminando assim, o uso de *encoders* para a contagem de passos.

8.2 Trabalho futuro

Como mencionado em cima, o trabalho realizado serviu como prova de conceito. Sendo ainda é necessário realizar várias alterações e melhorias para que o robô se torne autónomo e apto para

[1] gitlab.inesctec.pt/agrob/pixelcroprobot

a realização de tarefas agrícolas.

Uma das tarefas que ficou por concluir e que é necessária para o bom funcionamento do PixelCropBot é a criação de um controlador de velocidade, responsável por gerar os valores de PWM pretendidos, de forma a que o robô consiga movimentar-se com a velocidade desejada. É necessário utilizar a informação enviada pela câmara relativa à orientação do robô para que este permanece alinhado com a horta.

Adicionalmente, é necessário criar as ferramentas que serão fixadas na extremidade do eixo Z. Estas ferramentas, a serem desenvolvidas, devem incluir uma sementeira, um emissor de água e um removedor de infestantes.

Para além das ferramentas, será importante colocar uma câmara neste eixo. Esta servirá para recolher imagens da horta, de forma a criar um *dataset* das espécies plantadas e para que seja possível detetar o surgimento de infestantes.

A nível de Software, será necessário criar um nó que enviará os comandos de posição com base no estado da horta. Este nó terá que ter conhecimento das plantas existentes na horta, bem como as necessidades de rega de cada uma, para que sejam gerados os comandos necessários para realizar esta operação.

O algoritmo de deteção de QR codes precisa de ser melhorado para permitir a descodificação de informação em ambientes onde a iluminação não é controlada. Poderá também ser otimizado, com o intuito de aumentar o *frame rate* para que seja possível elevar a velocidade do robô. Será importante rever a estrutura do robô e torná-lo mais robusto, com o intuito de minimizar as oscilações existentes durante o movimento e após a travagem de um dos eixos.

Anexo A

Informação adicional sobre o Farmbot

A utilização de maquinaria na agricultura costuma estar normalmente associada a monoculturas. O Farmbot declara-se como o primeiro equipamento agrícola escalável capaz de lidar com policulturas. Incorpora a eficácia, a escalabilidade e o uso mínimo de mão de obra enquanto permite a gestão de policulturas com elevada eficiência do ponto de vista biológico. Estas competências associadas ao conhecimento técnico sobre as plantas (espaçamento entre as mesmas, potenciais sinergias) tornam a horta sustentável e eficiente como o mínimo de intervenção humana.

A.1 Hardware

Farmbot apresenta uma nova abordagem relativamente à agricultura de precisão colocando todo o equipamento sobre trilhos, desta forma o sistema tem a capacidade de ser extremamente preciso e de reposicionar as ferramentas em locais exatos repetidamente ao longo do tempo. Esta tecnologia é semelhante à utilizada em impressoras 3D e fresadoras. A precisão pode aumentar a eficiência no sistema agrícola de várias maneiras, incluindo as seguintes:

- As infestantes podem ser eliminados por exemplo através de pesticidas ou remoção mecânica sem danificar o que foi plantado.

- Cada planta pode ser regada e fertilizada individualmente e com um regime otimizado que é alterado conforme a fase de crescimento da planta.

- Os ciclos de vida das diferentes plantas não precisam começar e terminar ao mesmo tempo. Portanto, qualquer espaço livre pode ser replantado imediatamente.

- Um mapeamento preciso da horta permite a seleção das plantas mais convenientes para as condições espaciais e temporais.

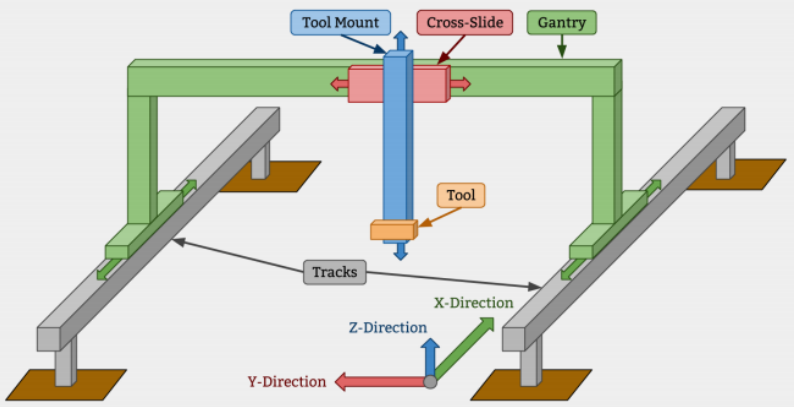


Figura A.1: Visão geral do Hardware do Farmbot

A figura A.1 representa de forma simplificada a estrutura do Farmbot. Existem dois trilhos paralelos que representam a direção do eixo X, existindo um pórtico sobre estes trilhos e que se move ao longo dos mesmos. Inserido neste pórtico está um *cross-slide* responsável pela movimentação segundo o eixo Y. A UTM (*Univelsal Tool Mount*) encontra-se acoplada a uma estrutura que representa o eixo Z e que se encontra acoplada ao *cross-slide*, permitindo movimentos em altura.

A.1.1 Trilhos

Como referido anteriormente a utilização de trilhos é um fator que diferencia o Farmbot das diferentes soluções já existentes. Esta implementação apresenta vantagens relativamente ao uso de um veiculo (por exemplo um trator):

- Grande precisão em todos os seus movimentos.
- Qualquer geometria de plantação pode ser escolhida visto que não há necessidade de caminhos adicionais para a passagem de hardware.
- Os trilhos ocupam menos espaço do que os acessos utilizados pelos tratores e não compactam o solo.
- Controlo simplificado, no caso do trator seria necessário criar um sistema de piloto automático.

Os trilhos são formados utilizando extrusões de alumínio que devem ser colocadas preferencialmente ao longo de uma *raised Bed* ou podem ser colocados perto do solo recorrendo ao uso de *Pier Blocks*. A segunda opção apresenta maior dificuldades durante o processo de instalação que está relacionado com o alinhamento das diferentes extrusões e é preciso existir um maior cuidado para garantir a estabilidade de toda a estrutura.

O transporte de toda a cablagem é conseguida através de da utilização de esteiras porta cabos que se encontram em torno dos 3 eixos existentes. Inclui não só as ligações elétricas mas também

o tubo responsável pelo fornecimento de água e ao longo do eixo Z o tubo de ar que está ligado à bomba de vácuo.

Os motores utilizados para permitir a deslocação do robô são os *NEMA 17 Stepper Motor*. Este motor foi escolhido por ser acessível, utilizado em aplicaçõe semelhantes (algumas empressoras 3D), facilidade de controlo, precisão, velocidade e binário.

A.1.2 Ferramentas

O Farmbot apresenta um conjunto de ferramentas com diferentes funcionalidades que podem ser encaixadas na UTM. A UTM está fixo à extrusão de alumínio que representa o eixo Z. A UTM possui 12 ligações eléctricas das quais apenas 5 estão a ser utilizadas, sendo possível usar as restantes para acrescentar funcionalidades e/ou permitir o uso de novas ferramentas/sensores criados pelo utilizador. Possui ainda ligação a 3 cabos responsáveis pelo transporte de gases ou líquidos. Na solução comercial neste momento apenas duas destas ligações estão a ser usadas, podendo a terceira ser utilizada para o fornecimento, por exemplo, de fertilizantes. O encaixe das ferramentas é conseguido através da utilização de 3 ímanes.



Figura A.2: Ferramentas incluídas no modelo Genesis

- Injetor de sementes - É controlado por uma bomba de vacuo alimentada a 24V e possui precisão milimétrica ao posicionar as sementes no solo. Possui ainda agulhas com diferentes tamanhos que podem ser trocadas de forma a se adaptarem ao tamanho da semente a ser semeada.

- Emissor de água - O Fluxo de água é controlado por uma eletroválvula. Esta ferramenta é divida em duas peças distintas. A peça superior é responsável pela interligação entre a ferramenta e a UTM, bem como ao tubo de transporte de água. A parte inferior tem a função de dividir a corrente de aguá por vários orifícios de forma a ser possível criar um padrão de rega mais uniforme, reduzindo a pressão para não danificar as plantas. Se for preciso alterar o padrão de rega é necessário apenas recriar a peça inferior com o formato desejado.

- Removedor de infestantes - Esta ferramenta é capaz de esmagar e destroçar pequenos infestantes. Esta ferramenta é configurável sendo possível mudar o tipo de lâminas ou espigões e também a disposição dos mesmos, adaptando-se assim ao tipo de solo e infestante.

- Sensor de humidade do solo - Este sensor possui duas pontas metalizadas que são enterradas no solo, que funcionam como uma resistência variável cujo valor depende da quantidade de água do solo. A quantidade de água no solo é proporcional à condutividade do mesmo, o que irá resultar numa resistência menor.

- Câmera - Ao contrário das outras ferramentas a câmara não é encaixada na UTM, encontrando-se fixada em torno do eixo Z (ao lado da UTM). Esta é uma câmera boroscopica, à prova de água e com ligação USB. De momento é utilizada para a deteção de infestantes e acompanhamento do crescimento das plantas.Os criadores pretendem no futuro através de algoritmos de *machine learning* detetar possíveis doenças que estejam a afetar as plantas.

Estas ferramentas são utilizadas uma de cada vez sendo posteriormente armazenadas num compartimento para ferramentas que pode ser inserido em qualquer parte da estrutura desde que esteja ao alcance da UTM. Os modelos do Farmbot incluem 2 compartimentos para ferramentas (cada um com capacidade para 3 ferramentas). Este componente foi projetado para que o processo de montagem/desmontagem da ferramenta seja automático. Para além das ferramentas previamente apresentadas, também é neste compartimento que são guardados os recipientes contendo as sementes a serem plantadas.

- *Seed Bin* - Um único recipiente usado para sementes de grande dimensão ou para sementes todas do mesmo tipo (monocultura).

- *Seed Tray* - Recipiente dividido em 16 orifícios, permitindo ter um controlo sobre o número e tipo de sementes a plantar.

Os modelos CAD das diferentes ferramentas são disponibilizados no site, facilitando a modificação de ferramentas existentes ou até mesmo a criação de novas compatíveis com a UTM. Estes modelos foram criados usando a versão grátis do *OnShape*. O *OnShape* funciona com base no modelo SaaS (*software as a service*) e permite a partilha de ficheiros, bem como a edição em simultâneo de um mesmo design. Com o software atual do farmbot é apenas possível acrescentar novos sensores analógicos. Para ler sensores digitais (ex. I2C) é necessário modificar o código base.

A.1.3 Controlo

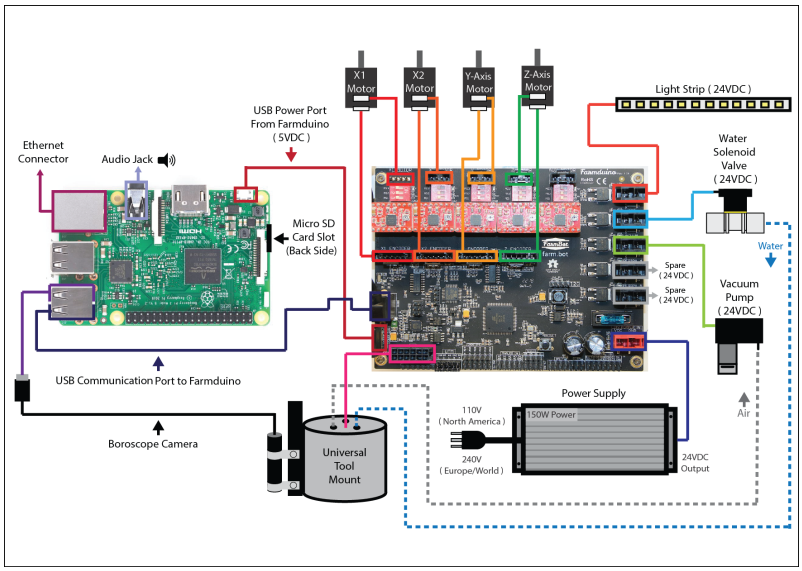


Figura A.3: Visão geral do Hardware do Farmbot

O controlo dos motores, da eletroválvula, da bomba de vácuo e sensores é feito através de um Arduino. Este microcontrolador foi escolhido por ser *open source*, pelo seu baixo custo, uma grande comunidade que utiliza esta plataforma e a linguagem de programação *C*.

A versão mais recente do farmbot (v1.5) vem equipada com uma placa denominada Farmduino que foi desenvolvida e otimizada para ser integrada neste projeto. Esta placa é constituida por um ATmega2560, 5 drivers (Trinamic TMC2130) para o controlo dos motores, um co-processador (STM32), com o objetivo de processar os dados dos encoders mais rapidamente e ligações aos vários periféricos, à Raspberry Pi e ao cabo de dados da UTM. A placa Farmduino é alimentada a 24V, para que para além da parte de controlo consiga alimentar a eletroválvula e bomba de vácuo.

Os elementos fundamentais para o controlo são o Farmduino e a Raspeberry Pi 3B+, figura A.3 representa as ligações dos outros componentes elétricos a estas placas, bem como a ligação entre ambas. A comunicação com a Raspberry Pi é feita utilizando uma linguagem semelhante ao *G-code*. O *G-code* (*Geometric Code*) é uma linguagem de programação usada em máquinas que utilizam CNC (controlo numérico computorizado). Tem como objetivo informar a máquina do que fazer e como o fazer, inclui instruções sobre o local para onde a máquina deve-se mover, a velocidade do movimento e o tipo de trajectória.

A Raspeberry Pi é responsável pela monitorização e gestão da horta. É ainda encarregue por de realizar a ligação à internet, analisar a informação dos sensores, enviar comandos para o arduino, tirar fotos utilizando a câmara e monitorizar e controlar a botoneira e os LEDs que se encontram no topo da caixa que envolve estes dispositivos de controlo.

A.2 Comparação entre os modelos Genesis e Express

Atualmente são comercializados dois modelos do Farmbot, o Express (v1.0) e o Genesis(v1.5). O modelo Genesis é o mais recente, é mais costumizável, tem maior poder computacional, é mais resistente e apresenta uma maior área de atuação, o que se traduz num preço mais elevado. A seguir serão detalhadas as principais diferenças entre os dois modelos:

- Trilhos - O modelo Express não apresenta trilhos segundo o eixo X. As rodas presentes nos dois lados do pórtico deslizam diretamente na superficie de madeira da *raised bed*, sendo da responsabilidade do utilizador guarantir o bom funcionamento do mesmo. Adicionalmente, o Farmbot Express não foi testado em condições de ventos fortes ao contrário do Genesis que se mostrou adequado a diferentes condições climatéricas.

- Poder computacional - O modelo Express vem equipado com uma Raspberry Pi Zero, com apenas um processador com uma velocidade de 1GHz e 512MB de RAM. O modelo Genesis apresenta maior poder computacional, este vem equipado com uma versão mais recente, a Raspberry Pi 3 B+ com um processador *quad-core* que opera a 1.4GHz.

- Precisão - O modelo Express vem equipado com drivers TMC2130 que permitem detetar a paragem do motor e assim detetar os limites dos eixos de movimento. O modelo Genesis inclui em cada motor um *encoder*, sendo possível obter uma estimativa melhor para a posição do robô, aumentando a precisão das operações realizadas.

- Conexão à internet - O modelo Express tem apenas a opção de conexão através da rede Wi-fi, enquanto que o Genesis tem a também a opção de ligação usando o cabo Ethernet. Em termos de protocolos usados na comunicação Wi-fi o Express utiliza o 802.11n enquanto que a o Genesis suporta uma conexão mais rápida através do protocolo 802.11ac.

- Compatibilidade e customização - O modelo Express não tem a capacidade de trocar ferramentas, estando permanentemente fixado ao eixo Z uma ferramenta 3 em 1 (injetor de sementes, ferramenta de rega e o removedor de infestantes). O modelo Genesis possui várias ferramentas que podem ser fixadas facilmente ao UTM. Para além das ferramentas existentes, existe a possibilidade do utilizador desenvolver as próprias ferramentas compatíveis com a UTM ou esperar por novas criações da equipa do farmbot (que garantem a compatibilidade com o modelo Genesis).

- Tempo de montagem - O modelo Express vem 95% pré montado o que reduz imenso o tempo de montagem. O Modelo Genesis vem apenas 25% pré montado pelo que se estima um tempo de montagem entre 20 a 25 horas.

- Tamanho - Tanto o modelo Express como o Genesis estão disponíveis na versão normal e XL, estando o valor das suas dimensões especificados na Tabela A.1

Tabela A.1: Comparação das dimensões entre o modelo Express e Genesis

	Express	Genesis
Normal	1.2m x 3m (3.6m^2)	1.5m x 3m (4.5m^2)
XL	2.4m x 6m (14.4m^2)	3m x 6m (18m^2)

A.3 Software

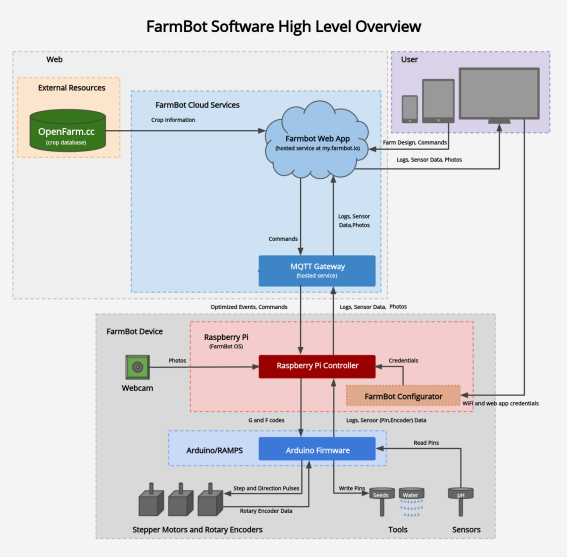


Figura A.4: Visão high level de todo o sistema

Arduino Firmware

O firmware que é carregado para o Arduino é responsável por controlar os motores, ferramentas, sensores e outros dispositivos electrónicos. Este recebe códigos G e F, atuando nos motores e lendo os valores dos sensores de acordo com as instruções recebidas.

FarmBot OS

O FarmBot OS é o sistema operativo personalizado criado para correr na Raspberry Pi e permitir a comunicação com o servidor web (através de wi-fi), sincronizar sequências, regimes, eventos, estado da horta e dados de sensores. Comunica com o arduino através de códigos G e F e recebe medições dos sensores e dados dos encoders. Recebe fotos via USB tiradas pela camâra e envia-as para a aplicação web. Permite uma configuração através de uma ligação wi-fi evitando assim o uso de um rato, teclado e monitor. O Farmbot OS está disponivel para Download no site oficial, sendo apenas necessário colocá-lo num cartão microSD, utilizando por exemplo o programa *balenaEtcher*.

OpenFarm

O OpenFarm é uma base de dados grátis e de livre acesso contendo informações sobre diversas plantas e informação relativa às suas necessidades. Começou por ser uma parte do Farmbot, no entanto agora é projeto independente.

Aplicação Web

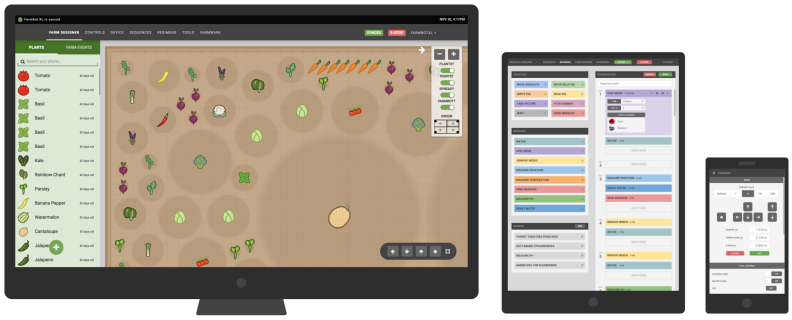


Figura A.5: Aplicação web em vários dispositivos

A aplicação Web permite ao utilizador controlar e configurar facilmente o Farmbot a partir do *web browser*, *tablet* ou *smartphone*. A aplicação permite um controlo manual em tempo real, acesso ao log dos sensores, a criação de sequencias e regimes/rotinas para serem executadas em datas especificas. A colocação das plantas na horta é feita através de um mecanismo de *drag-and-drop*, permitindo uma visualização gráfica da horta.

Criação de uma sequência

A forma do utilizador programar o farmbot para que este execute determinada ação (como plantar ou regar) é feita através da criação de uma sequência. A criação de uma sequência é conseguida criando uma lista ordenada de ações a realizar pelo farmbot, este conjunto de ações está definido no painel *Commands*. A figura A.6 mostra um exemplo da criação de uma sequência.

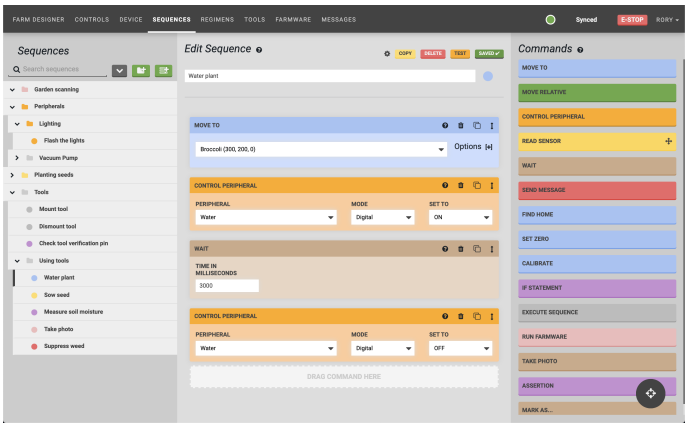


Figura A.6: Menu de criação de sequencia

A lista de comandos inclui instruções que permitem mover o robo, controlar alguns periféricos como a bomba de vácuo ou a eletróvalvula, leitura do sensor de humidade do solo, paragem de

emergência. Existe também um comando que permite criar *If statments*, dependendo a realização da sequência da avaliação da condição testada.

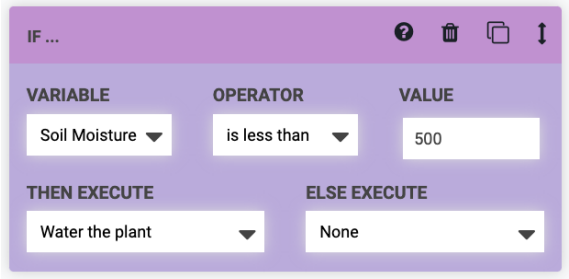


Figura A.7: Menu de criação de sequencia

A.3.0.1 Comunicação

Um Arduino tem memória e poder de processamento limitado para tarefas não relacionadas ao hardware. De forma a combater estas limitações, é utilizado a Raspberry Pi que apresenta também periféricos de conexão à rede. Esta apresenta assim as seguintes funções:

- Enviar a informação sobre a horta e os sensores para o utilizador e a *REST API*.
- Garantir a execução das tarefas agendadas.
- Aceitar chamadas remotas de procedimento (RPC *Remote Procedure Call*) enviadas pelo utilizador.

Apesar da Raspberry Pi possuir um poder de processamento razoável, apresenta como o Arduino alguns problemas no que diz respeito à integridade e disponibilidade dos dados. Os dados deixam de estar disponiveis se o equipamento estiver offline e alguma informação pode ser perdida devido a comportamentos inesperados como a falta ou um pico na alimentação.

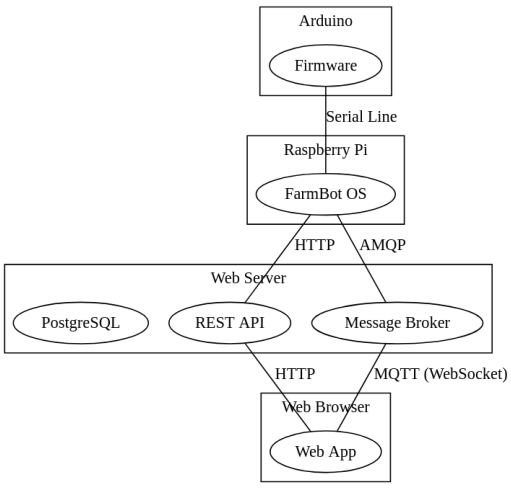


Figura A.8: Comunicação entre as diferentes partes que compõe o sistema

Por estas razões o Farmbot não pode ser 100% autónomo. De forma a mitigar os problemas de integridade e disponiblidade, este funciona sempre em conjunto com um *Web Server*, também denominado como "*The Web App*". A aplicação web é composta principalmente por 3 componentes:

- Um *message broker* em tempo real responsável pela troca de mensagens entre a base de dados, o utilizador e a Raspberry Pi.
- Uma interface gráfica para facilitar o controlo do farmbot por parte do utilizador.
- REST API para armazenamento de dados/informação.

REST API

A REST API é um *HTTP web server* que apresenta as seguintes funções:

- Armazenamento de informação, validação e segurança. A informação é armazenada numa *centralized database* o que permite a edição de informação quando o dispositivo está offline, a validação da informação e controlo de acesso é conseguido através de mecanismos de autentificação,
- Envio de notificações por email para o utilizador.
- Upload e tratamento das imagens - Redimensionamento e armazenamento das imagens capturadas pela câmara do Farmbot.

A REST API não tem como objetivo controlar o Farmbot, este processo é realizado pelo Message Broker.

Message Broker

A utilização de HTTP não se adequa à utilização de RPC. Se o farmbot utilizasse apenas HTTP como um mecanismo de comunicação, este seria forçado a estar num estado de *polling*, ou seja, teria que enviar constantemente novos pedidos HTTP para a API para ter conhecimento de um novo RPC.

Por este motivo é utilizado um message broker, que permite a troca de mensagens entre o cliente, o dispositivo e o servidor sem a necessidade de um pedido.

O message broker facilita a comunicação entre as diferentes partes do sistema. O Farmbot OS e a REST API podem enviar mensagens para as entidades que estiverem nesse momento conectadas ao message broker.

Data sync

Quando o utilizador ativa a função de auto-sync, a API irá constantemente fazer o upload das modificações em tempo real para o robô. Neste caso o utilizador não precisa de clicar no botão sync para confirmar as suas alterações na interface gráfica. Como mencionado anteriormente a API não consegue enviar mensagens via HTTP, visto que este protocolo assenta num paradigma de comunicação baseado no pedido/resposta. Desta forma em caso de updates, o Farmbot não

tem forma de saber que a informação guardada na API sofreu alterações e não faz um pedido para obter estas modificações. A solucão arranjada para este problema foi permitir o envio de mensagens pela API através do message broker. Estas mensagens são geradas através de um processo em *background*. Neste caso a web API assume os papeis de servidor web e também de cliente do ponto de vista do message broker.

A conexão entre o message broker e a aplicação web é realiada com recurso a web sockets se este estiver a correr num browser ou através de MQTT(Message Queuing Telemetry Transport) se for uma aplicação. A conexão entre o Farmbot OS e o message broker é realizada através de AMQP (Advanced Message Queuing Protocol).

Anexo B

Lista de materiais

		Quantidade	Preço	Total
RatRig	V-slot 2060 1.5m	1	19.37	19.37
	V-slot 2040 0.5m	2	5.15	10.3
	V-slot 2040 1.0m	2	10.3	20.6
	V-slot 2020 1m	1	7.5	7.5
	Nema 17 stepper cables	2	4.28	8.56
	Nema 17 stepper	2	11.99	23.98
	Leadscrew block	1	4.79	4.79
	Leadscrew 1m	1	29.99	29.99
	Flexible Coupling 5mm * 8mm	1	4.15	4.15
	V-wheels	12	3.9	46.8
	Spacer 6mm	6	0.23	1.38
	Eccentric spacer 6mm	6	1.5	9
	M5 screws 30mm	12	0.13	1.56
	Hex nut M5	12	0.15	1.8
	Gt2 belt 6mm	1	2.49	2.49
	Gt2 pulley	1	1.64	1.64
	Belt clamp	2	0.42	0.84
	Cable carrier 1.5m	1	17.71	17.71
	Cable carrier 1.0m	1	12.14	12.14
	M5 tnuts	85	0.2	17
	M5 8mm screw (25 pack)	4	2.02	8.08
	Cast 90 Degree Corner Bracket for 2020	16	0.65	10.4
	Joining Plate - T	4	2.58	10.32
	Joining Plate - 90 Degree	2	2.58	5.16
Outros	dvr8825	2	3.9	7.8
	Velleman VMA400	1	10.4	10.4
	M2 10mm screw	4	0.04	0.16
	M3 10 mm screw	22	0.03	0.66
	M3 15mm screw	8	0.03	0.24
	M4 50mm screw	24	0.1	2.4
	M5 12mm screw	8	0.03	0.24
	M5 20mm screw	6	0.03	0.18
	M6 35 mm screw	8	0.08	0.64
	M2 nuts	4	0.01	0.04
	M6 nuts	8	0.01	0.08
	M5 nuts	12	0.01	0.12
	M4 nuts	24	0.01	0.24
	SK8 shaft support	4	1.1	4.4
	Linear shaft rod 8mm	1	5.29	5.29
	KFL08 ball bearing	4	2.95	11.8
	Raspberry Pi 4	1	44.9	44.9
	Camara V2	1	30.49	30.49
	AgIoT	1	75	75
	Driver PCB	1	16.14	16.14
	Microswitch	2	1.08	2.16
	High Grip Wheels	2	10.08	20.16
	Power Window Motor	2	21.01	42.02
	12V battery	1	1	19.85
	Eletrical connections	1	10	10
	Cross-slide plate	1	20	20
3d printing	Motor_wheel_connector	2		
	Motor_enclosure	2		
	Motor_enclosure_cover	2		
	Free_wheel_enclosure	2		
	Free_wheel_enclosure_cover	2		
	Free_wheel_fixation_interior	2		
	Free_wheel_fixation_exterior	2		20.16
	StepperZ_base	1		
	StepperY_enclosure	1		
	Microswitch_support	1		
	Belt_end	1		
	Vertical_support	3		
	Horizontal_support	5		
	Camera_support	1		
			Total	621.13

Figura B.1: Lista de materiais utilizada para a construção do PixelCropBot

Anexo C

Esquemático da PCB

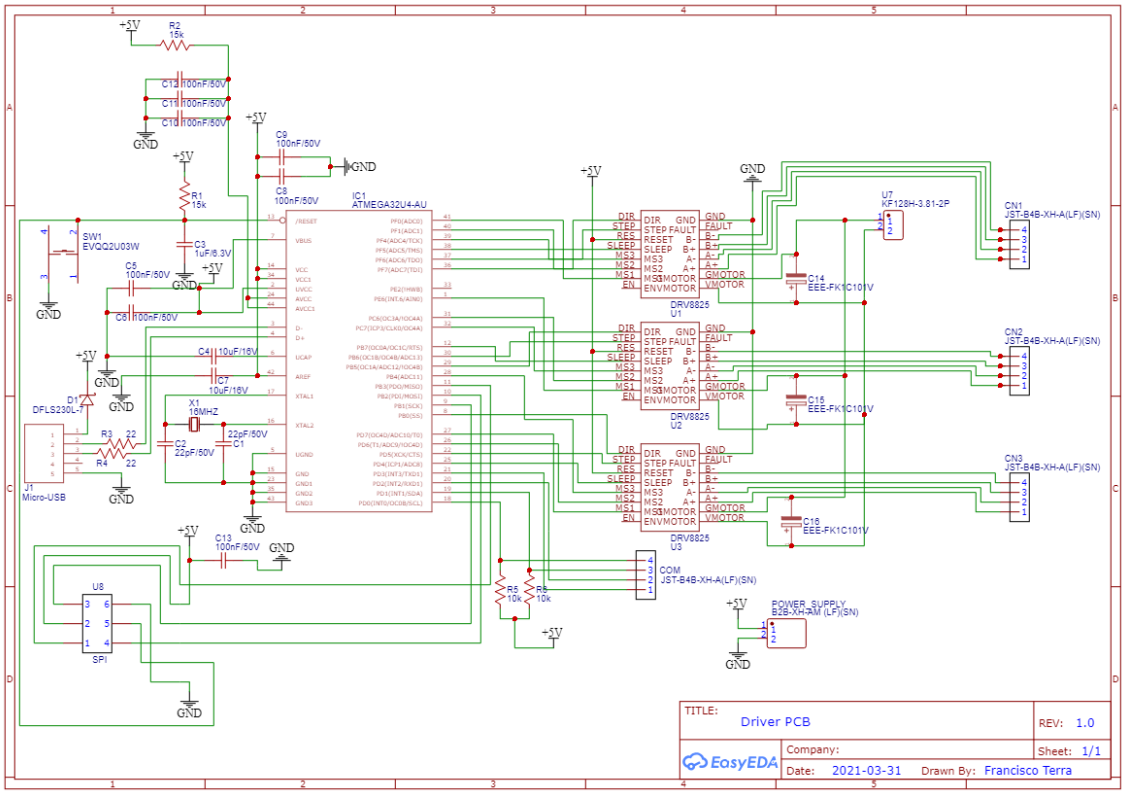


Figura C.1: Ligações elétricas da PCB desenhada para o controlo dos motores de passo

Referências

[1] Sehrish Munawar Cheema, Museb Khalid, Abdur Rehman, e Nadeem Sarwar. Plant Irrigation and Recommender System–IoT Based Digital Solution for Home Garden. Em *Communications in Computer and Information Science*, volume 932, páginas 513–525. Springer Verlag, 2019. URL: http://link.springer.com/10.1007/978-981-13-6052-7{_}44, doi:10.1007/978-981-13-6052-$7_4$4.

[2] Joseph D. Stevens e Talal Shaikh. MicroCEA: Developing a Personal Urban Smart Farming Device. Em *2nd International Conference on Smart Grid and Smart Cities, ICSGSC 2018*, páginas 49–56. Institute of Electrical and Electronics Engineers Inc., nov 2018. doi:10.1109/ICSGSC.2018.8541311.

[3] Wen Liang Chen, Yi Bing Lin, Yun Wei Lin, Robert Chen, Jyun Kai Liao, Fung Ling Ng, Yuan Yao Chan, You Cheng Liu, Chin Cheng Wang, Cheng Hsun Chiu, e Tai Hsiang Yen. AgriTalk: IoT for precision soil farming of turmeric cultivation. *IEEE Internet of Things Journal*, 6(3):5209–5223, jun 2019. doi:10.1109/JIOT.2019.2899128.

[4] Silvia Angeloni e Giorgia Pontetti. RobotFarm: A Smart and Sustainable Hydroponic Appliance for Meeting Individual and Collective Needs. Em *Advances in Intelligent Systems and Computing*, volume 994, páginas 268–280. Springer Verlag, 2020. doi:10.1007/978-3-030-22263-$5_2$6.

[5] Foughali Karim, Fathalah Karim, e Ali Frihida. Monitoring system using web of things in precision agriculture. *Procedia Computer Science*, 110:402–409, 2017. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050917312590, doi:10.1016/j.procs.2017.06.083.

[6] Lars Grimstad, Cong D. Pham, Huynh T. Phan, e Pal J. From. On the design of a low-cost, lightweight, and highly versatile agricultural robot. Em *Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts, ARSO*, volume 2016-March. IEEE Computer Society, mar 2016. doi:10.1109/ARSO.2015.7428210.

[7] J. A. López-Riquelme, N. Pavón-Pulido, H. Navarro-Hellín, F. Soto-Valles, e R. Torres-Sánchez. A software architecture based on FIWARE cloud for Precision Agriculture. *Agricultural Water Management*, 183:123–135, mar 2017. doi:10.1016/j.agwat.2016.10.020.

[8] FarmBot | Open-Source CNC Farming. URL: https://farm.bot/.

[9] Eduardo Castelló Ferrer, Jake Rye, Gordon Brander, Tim Savas, Douglas Chambers, Hildreth England, e Caleb Harper. Personal food computer: A new device for controlled-environment agriculture. Em *Advances in Intelligent Systems and Computing*, volume 881, páginas 1077–1096. Springer Verlag, 2019. arXiv:1706.05104, doi:10.1007/978-3-030-02683-$7_7$9.

[10] Lars Grimstad e Pål Johan From. Thorvald II - a Modular and Re-configurable Agricultural Robot. *IFAC-PapersOnLine*, 50(1):4588–4593, jul 2017. doi:10.1016/j.ifacol.2017.08.1005.

[11] Rubens Andre Tabile, Eduardo Paciencia Godoy, Giovana Tripoloni Tangerino, Arthur José Vieira Porto, Ricardo Yassushi Inamasu, e Rafael Vieira De Sousa. Application of systematic methods in the electromechanical design of an agricultural mobile robot. Em *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 4, páginas 276–281. Elsevier, aug 2013. doi:10.3182/20130828-2-SF-3019.00052.

[12] Brandin Grindstaff, Makenzie E. Mabry, Paul D. Blischak, Micheal Quinn, e J. Chris Pires. Affordable remote monitoring of plant growth in facilities using Raspberry Pi computers. *Applications in Plant Sciences*, 7(8), 2019. doi:10.1002/aps3.11280.

[13] Vaibhav Palande, Adam Zaheer, e Kiran George. Fully Automated Hydroponic System for Indoor Plant Growth. Em *Procedia Computer Science*, volume 129, páginas 482–488. Elsevier B.V., jan 2018. doi:10.1016/j.procs.2018.03.028.

[14] Lan Da Van, Yi Bing Lin, Tsung Han Wu, Yun Wei Lin, Syuan Ru Peng, Lin Hang Kao, e Chun Hao Chang. Planttalk: A smartphone-based intelligent hydroponic plant box. *Sensors (Switzerland)*, 19(8), apr 2019. doi:10.3390/s19081763.

[15] Gonçalo Marques, Diogo Aleixo, e Rui Pitarma. Enhanced Hydroponic Agriculture Environmental Monitoring: An Internet of Things Approach. Em *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11538 LNCS, páginas 658–669. Springer Verlag, 2019. doi:10.1007/978-3-030-22744-$9_5$1.

[16] Halil Durmus, Ece Olcay Gunes, Murvet Kirci, e Burak Berk Ustundag. The design of general purpose autonomous agricultural mobile-robot: 'AGROBOT'. Em *2015 4th International Conference on Agro-Geoinformatics, Agro-Geoinformatics 2015*, páginas 49–53. Institute of Electrical and Electronics Engineers Inc., sep 2015. doi:10.1109/Agro-Geoinformatics.2015.7248088.

[17] AeroGarden: Indoor Gardens, Grow Lights, Seed Kits, & More. URL: https://www.aerogarden.com/.

[18] Leaf. URL: http://www.getleaf.co.

[19] Zheqi Li e Jidong Huang. Study on the use of Q-R codes as landmarks for indoor positioning: Preliminary results. Em *2018 IEEE/ION Position, Location and Navigation Symposium, PLANS 2018 - Proceedings*, páginas 1270–1276. Institute of Electrical and Electronics Engineers Inc., jun 2018. doi:10.1109/PLANS.2018.8373516.

[20] A Sneha, V Sai, Lakshmi Teja, Kanti Mishra, e Kuppili N Satya Chitra. QR Code based Indoor Navigation system for Attender Robot. 2020. doi:10.4108/eai.13-7-2018.165519.

[21] Seok Ju Lee, Girma Tewolde, Jongil Lim, e Jaerock Kwon. QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument. Em *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, volume 2015-August, páginas 965–970. Institute of Electrical and Electronics Engineers Inc., aug 2015. doi:10.1109/AIM.2015.7222664.

[22] Huijuan Zhang, Chengning Zhang, Wei Yang, e Chin Yin Chen. Localization and navigation using QR code for mobile robot in indoor environment. Em *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, páginas 2501–2506. Institute of Electrical and Electronics Engineers Inc., 2015. doi:10.1109/ROBIO.2015.7419715.

[23] Hugo Mendes Magalhães. Smart Packaging e estudo de tecnologias emergentes aplicáveis ao mercado de embalagens. *undefined*, 2017.

[24] Gokhan Atali, Zeynep Garip, S Serdar Ozkan, e Durmuş Karayel. PATH PLANNING OF MOBILE ROBOTS BASED ON QR CODE. Relatório técnico, 2018.

[25] Ústav Mechaniky Těles, Mechatroniky A Biomechaniky, e Bc Milan Hurban. FACULTY OF MECHANICAL ENGINEERING FAKULTA STROJNÍHO INŽENÝRSTVÍ INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS. Relatório técnico.