

Supermercado Automático

BRUNO MIGUEL DA SILVA MAIA
julho de 2025

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Supermercado Automático

Bruno Maia

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

Julho, 2025

*Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de
Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em
Engenharia Electrotécnica e de Computadores, Área de Especialização em
Automação e Sistemas.*

Candidato: Bruno Maia, Nº 1190453, 1190453@isep.ipp.pt

Orientação Científica: Ramiro Barbosa, rsb@isep.ipp.pt



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

Julho, 2025

*Dedico este trabalho à minha família, que sempre acreditou em mim, e à minha
namorada, pela presença incansável em cada etapa desta jornada.*

Agradecimentos

A conclusão deste trabalho representa o culminar de um percurso académico e pessoal que só foi possível graças ao apoio de várias pessoas, às quais quero expressar a minha profunda gratidão.

Em primeiro lugar, agradeço à minha família, pelo amor, paciência e constante incentivo ao longo de todos estes anos. Foram o meu pilar, mesmo nos momentos mais desafiantes, e é com muito orgulho que partilho esta conquista convosco.

Agradeço também à minha namorada, pelo carinho, compreensão e apoio incondicional em todas as fases deste percurso.

Por fim, deixo ainda um agradecimento aos colegas, professores e todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho e para o meu crescimento académico e profissional.

Resumo

A crescente digitalização do setor do retalho tem impulsionado a adoção de tecnologias inovadoras para otimizar operações e melhorar a experiência do cliente. Supermercados e mercearias enfrentam desafios significativos, como tempos de espera elevados, gestão ineficiente de inventário e custos operacionais elevados. Em resposta a estas questões, a automação tem sido uma das principais abordagens para modernizar estes estabelecimentos.

Esta dissertação apresenta o desenvolvimento de um sistema de automação para supermercados baseado na integração de dispensadores automáticos e tapetes rolantes. O objetivo é permitir que os clientes realizem as suas compras de forma rápida e eficiente, sem necessidade de interação direta com operadores. O sistema proposto utiliza um conjunto de tecnologias, incluindo sensores, atuadores e um módulo de interface para facilitar a interação com o utilizador.

Para validar a eficácia da solução, foi desenvolvido um protótipo funcional, no qual foram analisados aspetos como tempo de processamento, precisão na entrega dos produtos e usabilidade do sistema. Os resultados obtidos demonstram o potencial da automação na otimização da experiência de compra e na redução de custos operacionais para os retalhistas.

Palavras-Chave: Automação, Supermercados, Dispensadores Automáticos, Tapetes Rolantes, Otimização de Processos.

Abstract

The increasing digitalization of the retail sector has driven the adoption of innovative technologies to optimize operations and enhance the customer experience. Supermarkets and convenience stores face significant challenges, such as long waiting times, inefficient inventory management, and high operational costs. In response to these issues, automation has emerged as a key approach to modernizing these establishments.

This dissertation presents the development of an automation system for supermarkets based on the integration of automatic dispensers and conveyor belts. The goal is to enable customers to complete their purchases quickly and efficiently without the need for direct interaction with staff. The proposed system incorporates a combination of technologies, including sensors, actuators, and a user interface module to facilitate interaction.

To validate the effectiveness of the solution, a functional prototype was developed, analyzing aspects such as processing time, product delivery accuracy, and system usability. The results demonstrate the potential of automation in optimizing the shopping experience and reducing operational costs for retailers.

Keywords: Automation, Supermarkets, Automatic Dispensers, Conveyor Belts, Process Optimization.

Índice

Lista de Figuras	ix
Lista de Tabelas	xi
Lista de Acrónimos	xiii
1 Introdução	1
1.1 Contextualização	1
1.2 Definição do Problema	1
1.2.1 Objetivos	2
1.2.2 Análise e Identificação de Requisitos	3
Requisitos Funcionais	3
Requisitos Técnicos	4
1.3 Plano de Trabalho	5
1.4 Organização do Relatório	6
2 Estado da Arte	9
2.1 Contexto Atual	9
2.1.1 Digitalização e Automação no Retalho	9
2.1.2 Desafios do Setor	10
2.1.3 Tendências e Soluções Emergentes	10
2.2 Soluções Existentes e Diferenciação	11
2.3 Infraestrutura Tecnológica e Protocolos de Comunicação	14
2.3.1 <i>WiFi</i>	14
2.3.2 <i>Bluetooth</i>	15
2.3.3 Tecnologias de Curta Distância	15
2.3.4 Comunicação Via MQTT	16
2.3.5 Protocolos Série: UART, I2C e SPI	17
2.4 Revisão da Literatura	19
3 Arquitetura do Sistema	23
3.1 Arquitetura Geral	23
3.2 <i>Hardware</i>	25
3.2.1 Raspberry Pi 5	26

3.2.2	<i>Espressif32</i> (ESP32)	26
3.2.3	Sensor Ultrassónico	27
3.2.4	Tapete Rolante	28
3.2.5	<i>Display In-Plane Switching</i> (IPS)	29
3.2.6	Servomotores	30
3.2.7	Módulo <i>Radio-Frequency Identification</i> (RFID)	31
3.3	Metodologias de Teste	32
3.3.1	Testes Funcionais dos Componentes Individuais	32
3.3.2	Testes de Integração do Sistema	32
3.3.3	Testes de Segurança e Fiabilidade	32
3.3.4	Testes de Precisão e Quantidade	33
3.4	Lista de Componentes	33
4	Desenvolvimento do Protótipo	35
4.1	Visão Geral da Arquitetura	35
4.2	Desenvolvimento da Estrutura Física	37
4.2.1	Visão Geral do Protótipo	37
4.2.2	Modelo Tridimensional	38
	Peça de Ligação entre a Mola e o Servomotor	38
	Funil de Direcionamento para o Tapete Rolante	38
	Caixa de Receção no Fim do Tapete Rolante	39
	Suporte da Mola e do Motor do Dispensador	40
4.2.3	Preparação para Impressão	41
	Peça de Ligação entre a Mola e o Servomotor	41
	Funil de Direcionamento para o Tapete Rolante	41
	Caixa de Receção no Fim do Tapete	42
	Suporte da Mola e do Motor	43
4.2.4	Impressão 3D das Peças	43
	Peça de Ligação entre a Mola e o Servomotor	44
	Funil de Direcionamento para o Tapete Rolante	44
	Caixa de Receção no Fim do Tapete	45
	Suporte da Mola e do Motor	45
4.3	Implementação do <i>Hardware</i>	46
4.3.1	Sensores e Atuadores	46
4.3.2	Ligações e Esquemas Elétricos	46
4.4	Implementação do <i>Software</i>	48
4.4.1	Configuração do Ambiente de Desenvolvimento	48
4.4.2	Visão Geral do <i>Software</i>	48
4.4.3	Aplicação <i>Web</i>	48
	Comunicação entre o <i>Backend</i> e o <i>Frontend</i>	52

4.4.4	Integração do ESP32 no Sistema	59
	Conexão <i>Wi-Fi</i> e Inicialização do Cliente <i>Message Queuing</i> <i>Telemetry Transport</i> (MQTT)	59
4.4.5	Cliente ESP32 no Raspberry Pi	61
	Ficheiro <code>mqtt_client.py</code>	61
	Envio de Comandos pela Aplicação <i>Web</i>	62
4.4.6	Programa do ESP32	63
	Mapeamento de Produtos e Controlo de Servos	63
	Função de Dispensa e Tapete Rolante	64
	Controlo do Tapete Rolante com Sensor no Fim de Curso . .	65
4.5	Resultado Final do Protótipo	68
5	Validação do Sistema e Resolução de Problemas	69
5.1	Testes e Validação	69
5.1.1	Testes de Integração	69
5.1.2	Ajustes de Integração	70
5.1.3	Testes de Precisão e Quantidade	70
5.1.4	Testes de Robustez e Fiabilidade	71
5.1.5	Análise Final dos Testes	71
5.2	Desafios e Soluções	71
6	Conclusões	75
6.1	Trabalho Futuro	76
	Referências	77
	Anexo A Código Fonte da Aplicação <i>Web</i>	83
A.1	Ficheiro <code>app.py</code>	83
A.2	Ficheiro <code>mqtt_client.py</code>	93
A.3	Ficheiro <code>main.py</code>	94
A.4	Ficheiro <code>base.html</code>	95
A.5	Ficheiro <code>cart.html</code>	98
A.6	Ficheiro <code>category.html</code>	102
A.7	Ficheiro <code>checkout.html</code>	103
A.8	Ficheiro <code>index.html</code>	108
A.9	Ficheiro <code>search_results.html</code>	113
	Anexo B Código Fonte do Programa do ESP32	115
B.1	Ficheiro <code>main.cpp</code>	115

Lista de Figuras

1.1	Diagrama de Gantt	6
2.1	Loja da Amazon Go [2]	11
2.2	Armazém da Ocado [5]	12
2.3	Robomart [8]	12
2.4	Comparação entre Suning e BingoBox	13
2.5	Armazém da Takeoff Technologies [13]	14
2.6	Evolução do <i>WiFi</i> [15]	14
2.7	<i>Bluetooth mesh</i> [18]	15
2.8	Funcionamento do <i>Radio-Frequency Identification</i> (RFID) [20]	16
2.9	Funcionamento do <i>Near Field Communication</i> (NFC) [22]	16
2.10	Arquitetura <i>Publish/Subscribe</i> do <i>Message Queuing Telemetry Transport</i> (MQTT) [24]	17
2.11	Protocolo de Comunicação <i>Universal Asynchronous Receiver-Transmitter</i> (UART) [26]	18
2.12	Protocolo de Comunicação <i>Inter-Integrated Circuit</i> (I2C) [29]	18
2.13	Protocolo de Comunicação <i>Serial Peripheral Interface</i> (SPI) [32] . .	19
3.1	Arquitetura geral do sistema	24
3.2	Raspberry Pi 5 [43]	26
3.3	<i>Espressif32</i> (ESP32) [45]	27
3.4	Sensores ultrassônicos utilizados no projeto	28
3.5	Tapete rolante	29
3.6	<i>Liquid Crystal Display</i> (LCD) [52]	29
3.7	Servomotor SG90 [53]	30
3.8	Servomotor MG996R [56]	31
3.9	Módulo RFID [57]	31
4.1	Diagrama de blocos da arquitetura do sistema	36
4.2	Impressora BambuLab A1	37
4.3	Modelo tridimensional da peça de ligação entre a mola helicoidal e o servomotor, desenvolvido no Autodesk Fusion 360.	39
4.4	Modelo 3D do funil de direcionamento dos produtos do dispensador para o tapete rolante.	39

4.5	Modelo 3D da caixa de receção dos produtos situada no fim do tapete rolante.	40
4.6	Modelo 3D do suporte da mola e do motor do dispensador.	41
4.7	Preparação da peça de ligação no Bambu Studio com parâmetros otimizados.	42
4.8	Visualização da preparação do funil no Bambu Studio.	42
4.9	Preparação da caixa de receção para impressão.	43
4.10	Peça do suporte da mola pronta para impressão.	43
4.11	Peça final impressa em <i>Polyethylene Terephthalate Glycol</i> (PETG), montada na mola helicoidal.	44
4.12	Funil de direcionamento impresso em PETG.	45
4.13	Caixa de receção impressa, colocada no fim do tapete rolante.	45
4.14	Suporte da mola e do motor após impressão e montagem.	46
4.15	Esquema das ligações entre o ESP32, servos, motor do tapete e sensor	47
4.16	Página inicial	50
4.17	Página das categorias	50
4.18	Página do carrinho	51
4.19	Página de <i>checkout</i> do <i>website</i>	51
4.20	Página de Resultados de Pesquisa	52
4.21	Fluxograma do método de pesquisa de produtos	53
4.22	Fluxograma do método para obter os produtos	54
4.23	Fluxograma do método atualização de produtos do carrinho	55
4.24	Fluxograma do método remoção de produtos do carrinho	57
4.25	Fluxograma do método de finalização de compra	58
4.26	Fluxograma do método de conexão ao <i>Wi-Fi</i>	60
4.27	Fluxograma do método de receção de mensagens e tratamento de dados	62
4.28	Fluxograma do método de dispensador de produtos	65
4.29	Fluxograma do método de controlo do tapete rolante	66
4.30	Fluxograma do método de controlo do sensor ultrassónico	67
4.31	Fotografia do protótipo final do sistema	68

Lista de Tabelas

3.1	Testes funcionais dos componentes individuais	32
3.2	Testes de integração do sistema	32
3.3	Testes de segurança e fiabilidade	33
3.4	Testes de precisão e quantidade	33
3.5	Lista de componentes	34
3.6	Lista de componentes utilizados no protótipo	34
4.1	Ligações dos pinos do ESP32 aos componentes	47
5.1	Resultados dos Testes de Integração	70
5.2	Ajustes Durante a Integração	70
5.3	Resultados dos Testes de Precisão	70
5.4	Testes de Robustez e Recuperação	71

Lista de Acrónimos

API	<i>Application Programming Interface</i>
ARC	<i>Automatic Retail Checkout</i>
BOM	<i>Bill of Materials</i>
CSS	<i>Cascading Style Sheets</i>
DC	<i>Direct Current</i>
DEE	Departamento de Engenharia Electrotécnica
DOM	<i>Document Object Model</i>
ESP32	<i>Espressif32</i>
GB	<i>GigaByte</i>
GPIO	<i>General-Purpose Input/Output</i>
GPS	<i>Global Positioning System</i>
HC-SR04	<i>High-Conductance Ultrasonic Sensor</i>
HDMI	<i>High Definition Multimedia Interface</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol Secure</i>
I2C	<i>Inter-Integrated Circuit</i>
IoT	<i>Internet of Things</i>
IPS	<i>In-Plane Switching</i>
ISEP	Instituto Superior de Engenharia do Porto
JSN-SR04T	<i>Waterproof Ultrasonic Distance Sensor Module</i>
LCD	<i>Liquid Crystal Display</i>
MEEC	Mestrado em Engenharia Electrotécnica e Computadores

MQTT	<i>Message Queuing Telemetry Transport</i>
NFC	<i>Near Field Communication</i>
NVMe	<i>Non-Volatile Memory Express</i>
PCIe	<i>Peripheral Component Interconnect Express</i>
PETG	<i>Polyethylene Terephthalate Glycol</i>
PWM	<i>Pulse-width Modulation</i>
QoS	<i>Quality of Service</i>
QR	<i>Quick-Response</i>
REST	<i>Representational State Transfer</i>
RFID	<i>Radio-Frequency Identification</i>
RS-232	<i>Recommended Standard 232</i>
RS-485	<i>Recommended Standard 485</i>
RX	<i>Receive Data</i>
SD	<i>Secure Digital</i>
SPI	<i>Serial Peripheral Interface</i>
SSD	<i>Solid-State Drive</i>
TCP	<i>Transmission Control Protocol</i>
TEDI	<i>Tese/Dissertação</i>
TLS	<i>Transport Layer Security</i>
TX	<i>Transmit Data</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>
USB	<i>Universal Serial Bus</i>

Capítulo 1

Introdução

Com este capítulo introdutório pretende-se fazer a contextualização do projeto a desenvolver, sendo, para tal, definidos os objetivos propostos e a calendarização das atividades necessárias para os cumprir, bem como a organização do relatório.

1.1 Contextualização

O projeto apresentado enquadra-se no âmbito do trabalho proposto na unidade curricular de Tese/Dissertação (TEDI) do Mestrado em Engenharia Electrotécnica e Computadores (MEEC), do Departamento de Engenharia Electrotécnica (DEE), do Instituto Superior de Engenharia do Porto (ISEP).

1.2 Definição do Problema

Nos últimos anos, o setor do retalho, particularmente os supermercados, tem enfrentado uma crescente pressão para melhorar a eficiência operacional e oferecer uma experiência de compra mais satisfatória aos clientes. Este desafio surge em grande parte devido ao aumento da procura por conveniência e rapidez, fatores que nem sempre são atendidos pelos métodos tradicionais de operação.

Atualmente, a maioria dos supermercados ainda depende de processos manuais ou parcialmente automatizados para gerir o fluxo de produtos e clientes. Esta abordagem resulta frequentemente em filas longas, atrasos no atendimento e falhas logísticas, que não só diminuem a satisfação do cliente como também impactam

negativamente a produtividade do supermercado. Além disso, erros humanos no manuseamento de produtos ou na contagem de inventário continuam a ser uma fonte significativa de ineficiência e custos desnecessários.

Outro problema relevante é a sustentabilidade e o custo operacional. A dependência de trabalho intensivo em tarefas repetitivas contribui para o desperdício de recursos, enquanto o aumento da competitividade no mercado exige soluções mais económicas e escaláveis. Por exemplo, a cadeia de supermercados italiana Iper eliminou os talões de compra em papel, adotando recibos digitais para reduzir o impacto ambiental e os custos associados ao uso de papel [1]. Embora esta prática já seja algo visto em Portugal, ainda não é frequente.

Embora algumas grandes cadeias de supermercados tenham adotado tecnologias avançadas, como caixas de autoatendimento ou sistemas automatizados de inventário, estas soluções são frequentemente dispendiosas e difíceis de implementar em supermercados de menor escala ou em redes com margens reduzidas. Neste contexto, existe uma lacuna clara para o desenvolvimento de sistemas de automação integrados, acessíveis e eficientes, que consigam melhorar o processo de compra e entrega de produtos, minimizando simultaneamente os erros, o desperdício e principalmente o custo de implementação. A criação de um sistema automatizado com dispensadores de produtos e tapetes rolantes para entrega direta na caixa surge como uma solução potencialmente transformadora para estes problemas, permitindo uma operação mais rápida, precisa e sustentável.

1.2.1 Objetivos

O objetivo principal desta tese é desenvolver um sistema automatizado para supermercados, com foco na melhoria da eficiência operacional e na experiência do cliente. O projeto visa automatizar processos que atualmente são realizados de forma manual, com o intuito de reduzir erros humanos, otimizar o fluxo de produtos e aumentar a rapidez no atendimento aos clientes.

Os objetivos específicos desta tese incluem:

- **Desenvolver um sistema automatizado de dispensadores de produtos e tapetes rolantes:** O sistema deverá permitir a entrega eficiente de produtos aos clientes, com uma interface simples e eficaz que minimize o tempo de espera e a interação humana.
- **Integrar sensores para garantir que os produtos chegam aos clientes:** O uso de sensores e possivelmente RFID permitirá o rastreamento em tempo real dos produtos e a verificação do inventário, reduzindo erros e melhorando a precisão do processo de *checkout*.

- **Implementar uma interface para o utilizador:** O desenvolvimento de uma interface de utilizador intuitiva será fundamental para garantir uma interação fácil e rápida com o sistema. A interface permitirá aos clientes escolher os produtos ou encomendar e escolher um horário de levantamento.
- **Melhorar a eficiência logística:** O sistema deverá ser capaz de otimizar o movimento dos produtos dentro do supermercado, reduzindo a necessidade de intervenções manuais e agilizando o processo de compra.
- **Analisar a viabilidade económica:** Será feita uma análise económica para determinar os custos de implementação do sistema automatizado em supermercados e as possíveis vantagens financeiras em termos de redução de custos operacionais.

A tese procura, assim, apresentar uma solução prática e acessível para melhorar a operação dos supermercados, focando-se em sistemas que podem ser implementados em supermercados de pequena e média dimensão, com um custo de implementação razoável e um retorno sobre o investimento tangível a médio e longo prazo.

1.2.2 Análise e Identificação de Requisitos

Para o desenvolvimento de um sistema automatizado de distribuição de produtos em supermercados, como proposto neste projeto de tese, é necessário identificar os principais requisitos técnicos e funcionais. A análise de requisitos tem como objetivo garantir que o sistema atenda às necessidades do utilizador final, como a eficiência do processo de compra, a experiência do cliente, a precisão no controlo de inventário e a viabilidade económica.

Requisitos Funcionais

Os requisitos funcionais descrevem as funções que o sistema deve ser capaz de realizar. Para automatizar o sistema do supermercado, foram identificados os seguintes requisitos funcionais:

1. Processo de Compra:
 - O sistema deve permitir ao cliente escolher os produtos desejados de forma rápida e fácil.
 - O pagamento deve ser feito de forma automática, sem a necessidade de caixas físicas.
 - O cliente poderá realizar o pagamento através de cartões de crédito/débito ou pagamento móvel.
2. Interface do Utilizador:

- O sistema deve possuir uma interface de utilizador simples e intuitiva, permitindo que o cliente explorar facilmente todos os produtos da loja.
- A interface deve permitir ao cliente seleccionar produtos, visualizar informações do produto e verificar o total da compra antes de efetuar o pagamento.

3. Automação do Processo de Entrega:

- O sistema deve garantir que os produtos sejam entregues ao cliente de forma eficiente através de tapetes rolantes automáticos, minimizando o tempo de espera.
- A entrega dos produtos deve ser feita sem a necessidade de intervenção humana.

4. Gestão de Produtos e Dispensadores:

- O sistema deve controlar automaticamente os dispensadores para entregar a quantidade correta de cada produto após o pagamento.
- O sistema deve identificar cada produto por um identificador único (ID) e relacioná-lo ao dispensador correspondente.
- O sistema deve ser capaz de processar múltiplos produtos numa única compra, acionando cada dispensador em sequência.

5. Comunicação entre Dispositivos:

- O sistema deve enviar, após o pagamento, uma mensagem com os dados dos produtos comprados para os dispensadores.

6. *Feedback* ao Cliente:

- O sistema deve apresentar ao cliente uma confirmação visual do pagamento e da preparação da entrega.

7. Modo de Encomenda Remota:

- O sistema deve permitir que os clientes realizem encomendas remotamente através de um website.
- O cliente deve poder seleccionar os produtos, confirmar o carrinho e agendar a recolha presencial.

Requisitos Técnicos

Para implementar o sistema automatizado de distribuição de produtos, os seguintes requisitos técnicos foram identificados:

1. *Hardware:*

- Dispensadores automáticos capazes de armazenar e distribuir os produtos de forma eficiente.
- Tapetes rolantes automáticos para movimentar os produtos até ao cliente.
- Microcontroladores (ESP32) para controlo local de dispensadores, sensores e atuadores.
- Sensores ultrassónicos para deteção de presença de produtos.
- Escolher um controlador como unidade central de processamento e gestão da interface de utilizador.

2. *Software:*

- Sistema de controlo centralizado que integra os dispensadores automáticos, sensores ultrassónicos, tapetes rolantes e interface do utilizador.
- Cliente e servidor MQTT para troca de mensagens entre o Raspberry Pi e os ESP32, garantindo sincronização do processo de entrega após pagamento.
- Sistema de pagamento automático com suporte a métodos digitais (ex. cartão de crédito ou pagamento móvel).

3. Redes de Comunicação:

- A comunicação entre os componentes do sistema (dispensadores, sensores, tapetes rolantes, etc.) deve ser realizada de forma rápida e segura, utilizando protocolos de rede confiáveis.
- A rede local deve suportar conectividade estável entre o Raspberry Pi e os ESP32 através de *Wi-Fi*.

1.3 Plano de Trabalho

A calendarização das tarefas inerentes a este projeto encontram-se representadas na Figura 1.1.

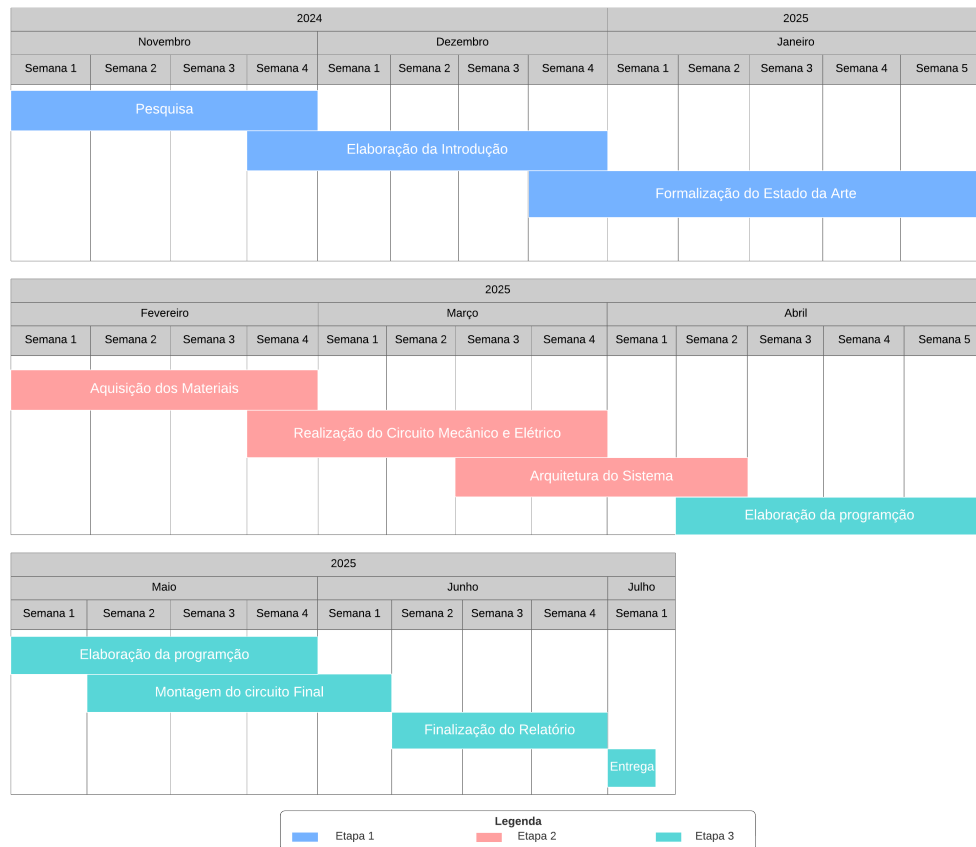


Figura 1.1: Diagrama de Gantt

1.4 Organização do Relatório

O presente relatório está estruturado em cinco capítulos, nos quais é descrito o processo de desenvolvimento do projeto. Neste primeiro capítulo, realiza-se uma contextualização do trabalho desenvolvido, delineando os objetivos a alcançar, a análise e identificação dos requisitos do problema, e, por fim, a calendarização das tarefas propostas.

No Capítulo dois, é apresentado um resumo do estudo realizado sobre diversas tecnologias que se revelaram fundamentais para a compreensão e desenvolvimento do projeto, particularmente no estudo dos protocolos de comunicação, nos dispositivos disponíveis no mercado e nos *softwares* relevantes para o trabalho em questão.

O Capítulo três descreve a arquitetura geral da solução desenvolvida, pormenorizando os componentes de *hardware* e *software* que a compõem.

O Capítulo quatro aborda o processo de desenvolvimento do protótipo, apresentando a arquitetura do *software*, dos sistemas e do mecanismo.

O Capítulo cinco apresenta a validação do sistema do dispensador e transporte de produtos, detalhando os testes realizados, os resultados obtidos e os principais

desafios técnicos enfrentados. Este capítulo confirma a fiabilidade do sistema e documenta os ajustes necessários para garantir a sua estabilidade.

Por fim, o Capítulo seis faz uma avaliação global do trabalho realizado, onde são apresentadas as conclusões e discutidas possíveis implementações futuras.

Capítulo 2

Estado da Arte

A automação tem transformado o retalho, com soluções como *self-checkout*, supermercados autónomos e robôs de inventário. Empresas como Amazon Go e Ocado adotam estas tecnologias, mas desafios como a gestão de *stock* e os custos operacionais persistem. Este capítulo analisa soluções atuais e tendências, enquadrando a inovação deste projeto com dispensadores automáticos e tapetes rolantes para otimizar a experiência de compra.

2.1 Contexto Atual

Nos últimos anos, o setor do retalho tem sofrido uma transformação significativa impulsionada pelo avanço das tecnologias digitais e pela mudança no comportamento dos consumidores. A crescente adoção de soluções automatizadas em supermercados e lojas de conveniência reflete a necessidade de maior eficiência operacional, redução de custos e melhoria da experiência do cliente.

2.1.1 Digitalização e Automação no Retalho

A digitalização do setor de retalho tem sido impulsionada pela crescente procura por soluções que reduzam o tempo de espera e eliminem tarefas repetitivas. Modelos de negócio inovadores, como supermercados sem caixas tradicionais e sistemas de *self-checkout*, tornaram-se cada vez mais comuns, permitindo aos consumidores um processo de compra mais rápido e conveniente.

Paralelamente, a automação desempenha um papel fundamental na otimização da gestão de inventário e na redução do desperdício. Sensores inteligentes, sistemas de monitorização em tempo real e algoritmos de previsão de procura permitem que os supermercados operem com maior eficiência, minimizando ruturas de *stock* e melhorando a cadeia de abastecimento.

2.1.2 Desafios do Setor

Apesar das vantagens da digitalização, os supermercados tradicionais ainda enfrentam desafios significativos, tais como:

- **Longas filas e tempos de espera:** Em horários de pico, os clientes podem enfrentar filas demoradas, afetando negativamente a experiência de compra.
- **Gestão ineficiente de inventário:** Erros na reposição de produtos e desperdícios são problemas recorrentes na operação de supermercados.
- **Custos operacionais elevados:** A necessidade de manter grandes equipas para reposição, atendimento e caixa aumenta os custos fixos das lojas.
- **Experiência do cliente:** Os consumidores procuram conveniência, rapidez e personalização, o que exige novas abordagens tecnológicas.

2.1.3 Tendências e Soluções Emergentes

Diante desses desafios, várias empresas têm apostado em soluções inovadoras para transformar a experiência de compra, tais como:

- **Supermercados autónomos:** Lojas sem caixas físicas, onde os produtos são identificados automaticamente através de sensores e cobrados diretamente ao cliente.
- **Sistemas de *self-checkout*:** Quiosques interativos que permitem aos clientes digitalizar e pagar pelos produtos sem necessidade de um operador de caixa.
- **Robótica e Inteligência Artificial:** Uso de robôs para reposição de prateleiras e algoritmos que analisam o comportamento dos consumidores para otimizar a disposição dos produtos.
- **Automação da recolha de produtos:** Implementação de tapetes rolantes e dispensadores automáticos para agilizar a entrega dos produtos aos clientes.

2.2 Soluções Existentes e Diferenciação

Empresas com projetos semelhantes, como Amazon Go e Takeoff Technologies, estão a explorar a automação para entregar produtos sem interação humana direta. A abordagem de dispensadores automáticos também é utilizada, principalmente em sistemas de distribuição e de armazenagem.

A maior diferença entre as soluções pré-existentes e o presente projeto de automação é que essas empresas geralmente focam na automatização de grandes armazéns ou na eliminação de caixas tradicionais. Em contraste, o conceito de utilização de tapetes rolantes e dispensadores automáticos diretamente para os clientes pode oferecer uma abordagem mais prática e inovadora para lojas físicas.

As soluções existentes incluem algumas empresas de destaque, tais como:

- **Amazon Go:** A Amazon Go é uma rede de lojas de proximidade sem caixas físicos, onde o cliente entra, retira os produtos desejados e os sensores e as câmaras monitorizam automaticamente os produtos retirados, tal como ilustrado na Figura 2.1 [2]. Ao sair da loja, o valor é debitado diretamente na conta da Amazon, sem necessidade de passar pela caixa física tradicional. Esta loja utiliza câmaras, sensores de peso e inteligência artificial para rastrear os produtos que os clientes retiram. A semelhança com o presente projeto está na automação do processo de retirada de produtos, pois embora a Amazon Go adote uma abordagem centrada em inteligência artificial e sensores avançados não utiliza, no entanto, dispensadores automáticos [3].



Figura 2.1: Loja da Amazon Go [2]

- **Ocado:** Esta empresa de supermercados *online* do Reino Unido implementou um sistema altamente automatizado de armazenamento e distribuição de produtos. No seu centro de distribuição, tal como pode ser observado na Figura 2.2, os robôs são usados para recolher e organizar produtos em cestos, que são depois entregues aos clientes através de um sistema de entregas convencional.

A tecnologia utilizada para a automação dos armazéns, inclui robôs automatizados em trilhos e sistemas de tapetes rolantes que movimentam os produtos dentro dos grandes armazéns. Embora a Ocado utilize robôs e tapetes para mover produtos, o foco está na automação de armazéns, em vez da interação direta com clientes em lojas físicas [4].



Figura 2.2: Armazém da Ocado [5]

- **Robomart:** O Robomart (Figura 2.3) é um supermercado móvel totalmente automatizado. Um veículo autônomo equipado com prateleiras de produtos desloca-se até aos clientes, que podem escolher os itens diretamente do veículo, sem necessidade de operadores humanos [6]. A tecnologia utilizada inclui um veículo autônomo e um sistema de pagamento automático, semelhante ao da Amazon Go. O Robomart oferece a conveniência da entrega automática de produtos, o que apresenta algumas semelhanças com o presente projeto, no que diz respeito à automação dos processos de venda sem interação humana [7].



Figura 2.3: Robomart [8]

- **Supermercados Autônomos na China:** Empresas chinesas, como BingoBox e Suning (Figuras 2.4a e 2.4b), têm vindo a testar supermercados sem funcionários. O cliente entra, faz a leitura de um código *Quick-Response* (QR) para aceder à loja, retira os produtos desejados e efetua o pagamento automaticamente ao sair. Estas lojas utilizam RFID, sensores de peso e sistemas de pagamento móvel integrados. Estes projetos partilham a ideia de eliminar caixas tradicionais e usar a automação para o controlo de inventário e pagamentos [9].



(a) Suning [10]



(b) BingoBox [11]

Figura 2.4: Comparação entre Suning e BingoBox

- **Takeoff Technologies:** Esta empresa trabalha com supermercados para automatizar a recolha de pedidos *online*. A empresa possui pequenos centros automatizados de distribuição dentro das próprias lojas, onde robôs recolhem os produtos para posterior entrega aos clientes. A tecnologia utilizada inclui tapetes rolantes, sistemas de entrega automática e robótica para automatizar o processamento dos pedidos, tal como representado na Figura 2.5. O conceito da Takeoff Technologies é semelhante ao presente projeto, no que diz respeito ao uso de tapetes rolantes automáticos e dispensadores de produtos para agilizar o atendimento ao cliente [12], mas usam operadores para fechar as caixas e terminar as encomendas.

Embora existam várias soluções no mercado, a maior diferença do projeto atual reside na combinação de tapetes rolantes e dispensadores automáticos diretamente até ao ponto de contacto com o cliente, uma abordagem mais prática e inovadora, focada em lojas físicas.

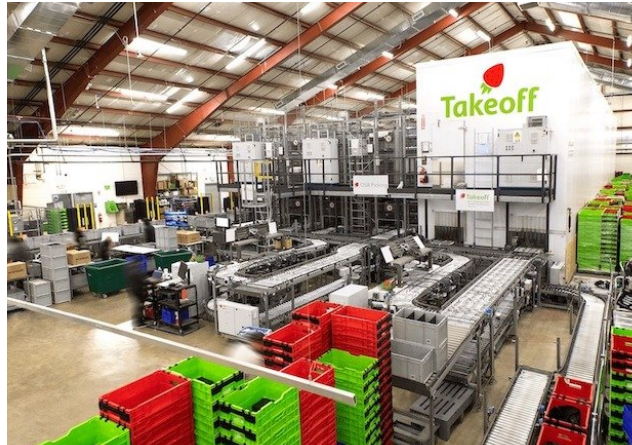


Figura 2.5: Armazém da Takeoff Technologies [13]

2.3 Infraestrutura Tecnológica e Protocolos de Comunicação

2.3.1 *WiFi*

O *WiFi* é uma tecnologia de comunicação *wireless* que possibilita a transmissão de dados através da utilização de ondas rádio, tipicamente nas frequências de 2.4 GHz e 5 GHz. O primeiro *standard* definido para o *WiFi* foi o IEEE 802.11, estreado em 1997, tendo sido alvo de várias revisões ao longo dos anos com várias melhorias, tais como melhor velocidade, alcance, e possibilidade de implementação de protocolos de segurança e encriptação. Esta evolução pode ser observada na Figura 2.6. O *WiFi* tem inúmeras aplicações práticas, nomeadamente o uso extensivo em redes *Internet of Things* (IoT), por ser altamente configurável, permitindo criar soluções seguras, com consumo energético mínimo e escaláveis, ou seja, que não são impeditivas à adição de mais dispositivos [14].

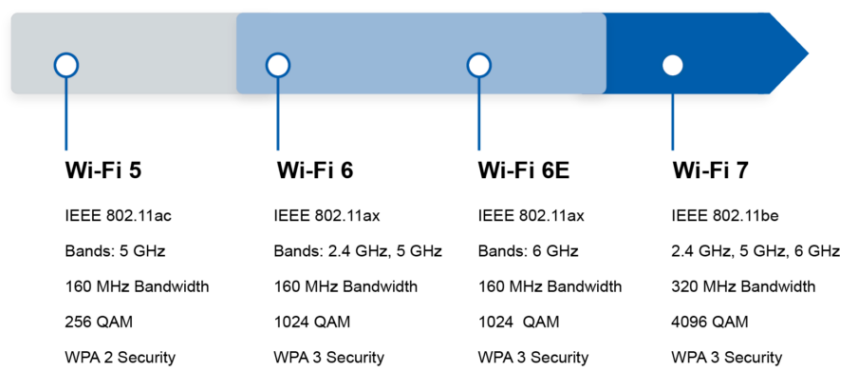


Figura 2.6: Evolução do *WiFi* [15]

2.3.2 Bluetooth

O *Bluetooth* é uma tecnologia de comunicação *wireless* que permite a troca de dados entre dispositivos a curta distância (Figura 2.7). Funciona na faixa de frequência de 2,4 GHz e é otimizada para baixo consumo de energia [16]. Esta tecnologia foi desenvolvida inicialmente para substituir cabos e conexões físicas, o *Bluetooth* é amplamente utilizado em diversos dispositivos, incluindo *smartphones*, computadores, impressoras e outros periféricos [17]. Estes, precisam de ser emparelhados para estabelecer uma conexão segura.

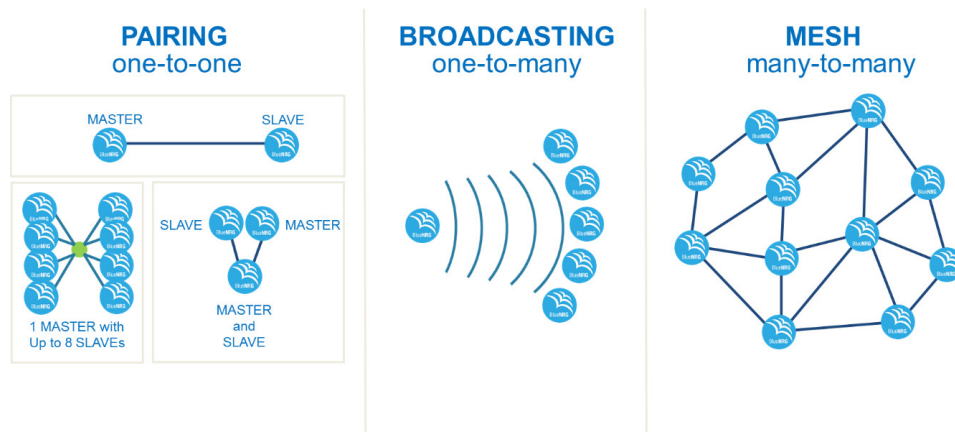


Figura 2.7: *Bluetooth mesh* [18]

2.3.3 Tecnologias de Curta Distância

As tecnologias de curta distância desempenham um papel fundamental em sistemas de comunicação e automação, especialmente em dispositivos IoT e aplicações de identificação e autenticação. Entre as mais conhecidas estão o *Radio-Frequency Identification* (RFID) e o *Near Field Communication* (NFC). Ambas as tecnologias utilizam comunicação sem fios para a transferência de dados entre dispositivos em proximidade, embora apresentem diferenças significativas nas suas aplicações e funcionalidades.

O RFID é uma tecnologia *wireless* que utiliza campos eletromagnéticos para a identificação de *tags* que são normalmente associadas a certos objetos. Estas *tags* contêm um *microchip* e uma pequena antena que transmite um identificador único quando a *tag* é lida por um leitor RFID. Quando uma *tag* entra na sua área de alcance, o leitor emite radiação eletromagnética que lhe induz uma corrente, o que faz com que esta envie o seu identificador único (Figura 3.9). Esta tecnologia tem muitas aplicações práticas, desde sistemas de controlo de acesso, sistemas de pagamento, gestão de inventário e identificação de animais e objetos [19].

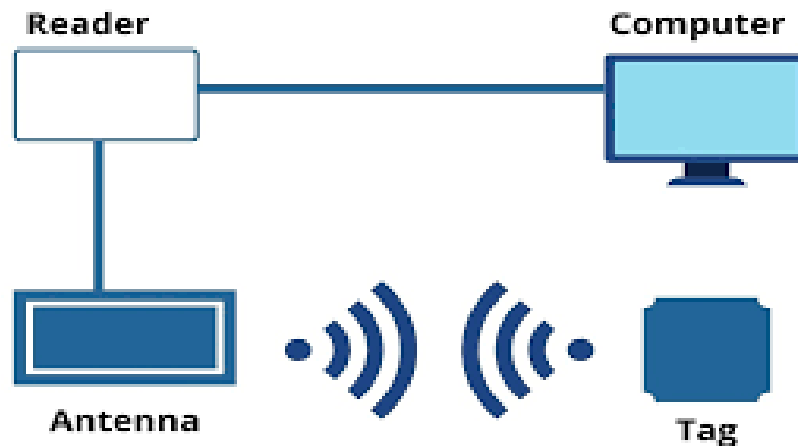


Figura 2.8: Funcionamento do RFID [20]

O NFC é uma tecnologia *wireless* que opera de uma forma muito semelhante ao RFID, no entanto existem algumas diferenças a salientar. O NFC opera à frequência de 13.56 MHz e tem um alcance de apenas alguns centímetros, enquanto que o RFID pode operar uma gama maior de frequências e pode ter um alcance de vários metros. Ao contrário do RFID, o NFC consegue incorporar vários protocolos criptográficos, resultando numa maior segurança na transferência de dados. Por este motivo, o NFC é preferido, por exemplo, ao RFID para sistemas de pagamento *contactless* (Figura 2.9) [21].

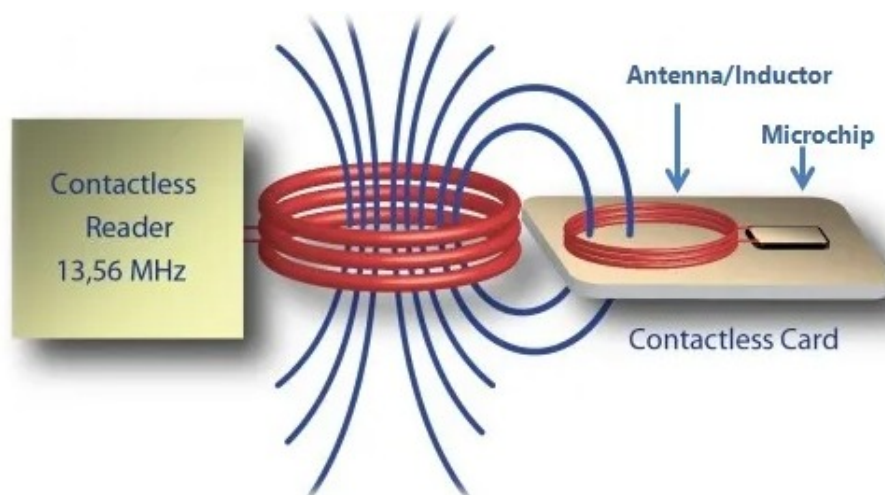


Figura 2.9: Funcionamento do NFC [22]

2.3.4 Comunicação Via MQTT

Este protocolo, também conhecido como *Message Queuing Telemetry Transport* (MQTT), é amplamente utilizado no domínio da IoT, projetado especificamente

para ser usado em ambientes com recursos limitados. Este protocolo faz uso de outros protocolos como o *Transmission Control Protocol* (TCP), ou *WebSockets* para estabelecer a comunicação entre vários dispositivos [23].

O modo de funcionamento do MQTT baseia-se na publicação de mensagens (*publish*), e a receção é realizada subscrevendo às mesmas (*subscribe*), tal como pode ser observado na Figura 2.10. Pelo meio, existe um *broker*, que recebe as mensagens publicadas pelos dispositivos e armazena-as numa *queue*, até que estas sejam entregues aos dispositivos que as subscrevem. As mensagens são organizadas em tópicos, que servem como uma estrutura para separar vários tipos de informação, tornando possível enviar e receber em simultâneo uma grande quantidade de dados [24].

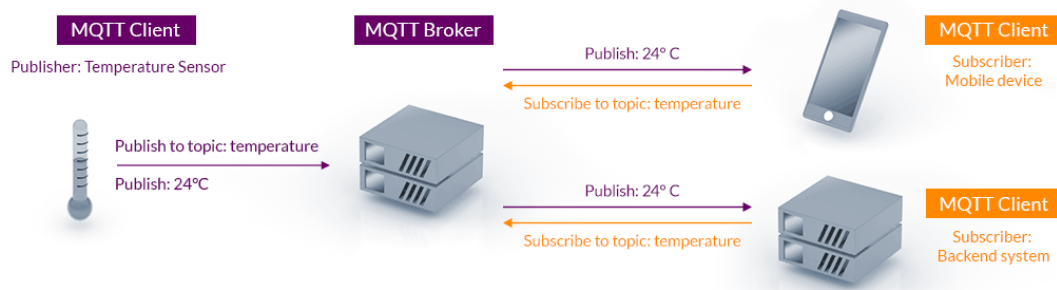


Figura 2.10: Arquitetura *Publish/Subscribe* do MQTT [24]

O protocolo MQTT suporta três níveis de *Quality of Service* (QoS):

- QoS 0: A mensagem é entregue no máximo uma vez, sem garantia de entrega.
- QoS 1: A mensagem é entregue pelo menos uma vez, garantindo que ela chegue ao destino, mas pode ser entregue mais de uma vez.
- QoS 2: A mensagem é entregue exatamente uma vez, garantindo que ela chegue ao destino apenas uma vez e na ordem correta.

O MQTT é vantajoso devido à sua leveza, baixa latência e flexibilidade, sendo utilizado em domótica, monitorização remota e saúde, entre outras áreas. A segurança pode ser reforçada com *Transport Layer Security* (TLS) para encriptação de dados.

2.3.5 Protocolos Série: UART, I2C e SPI

Os protocolos série desempenham um papel essencial na comunicação entre dispositivos embebidos. No contexto do projeto, podem ser utilizados para comunicação entre o Raspberry Pi e o ESP32, garantindo uma troca de dados eficiente e fiável.

UART: Protocolo de comunicação assíncrono que utiliza dois fios principais, *Transmit Data* (TX) e *Receive Data* (RX), para a transmissão e receção de dados

sem necessidade de um sinal de relógio compartilhado, tal como pode ser observado na Figura 2.11. A comunicação ocorre através da configuração de parâmetros como taxa de transmissão (*baud rate*), número de bits de dados (tipicamente 8 ou 9), paridade (opcional) e bits de paragem. O UART suporta comunicação ponto a ponto e pode ser usado para interligar microcontroladores, sensores, módulos *Global Positioning System* (GPS), módulos *Bluetooth* e interfaces *Recommended Standard 232* (RS-232) e *Recommended Standard 485* (RS-485) [25]. Apesar de ser simples e eficiente para curtas distâncias, o UART tem limitações em velocidade e número de dispositivos conectáveis quando comparado com protocolos como I2C e SPI.

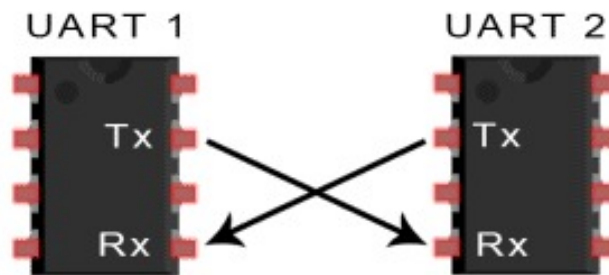


Figura 2.11: Protocolo de Comunicação UART [26]

I2C: Protocolo de comunicação síncrono que utiliza apenas dois fios, SDA e SCL (Figura 2.12), permitindo a interligação de múltiplos dispositivos num barramento comum. O I2C suporta comunicação entre um único mestre e múltiplos escravos, mas também pode operar em modo multi-mestre. A sua taxa de transferência pode variar entre 100 Kbit/s (modo padrão), 400 Kbit/s (modo rápido), 1 Mbit/s (modo rápido *plus*) e até 3.4 Mbit/s (modo de alta velocidade) [27, 28]. Esta tecnologia é amplamente utilizada em sistemas embebidos devido à sua simplicidade, baixo consumo de pinos e capacidade de endereçamento de múltiplos dispositivos num único barramento.

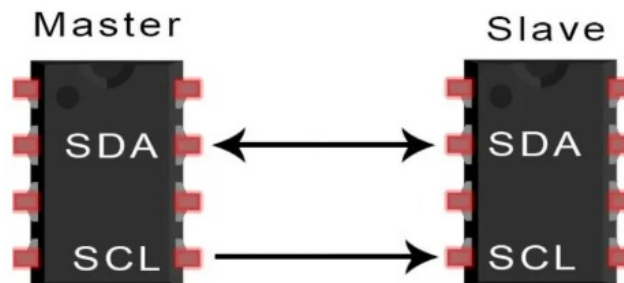


Figura 2.12: Protocolo de Comunicação I2C [29]

SPI: Protocolo de comunicação síncrono baseado num modelo mestre-escravo que utiliza quatro fios principais: MOSI, MISO, SCK e CS (Figura 2.13). O SPI destaca-se pela sua elevada velocidade de transferência, podendo atingir dezenas

de MHz, dependendo da implementação. Cada escravo necessita de uma linha de seleção exclusiva (CS), o que pode tornar a escalabilidade do sistema mais complexa em comparação com o I2C. No entanto, a sua taxa de transferência elevada e baixa latência fazem do SPI a escolha ideal para aplicações que requerem troca rápida de dados, como sensores de alta velocidade, cartões de memória e ecrãs [30, 31].

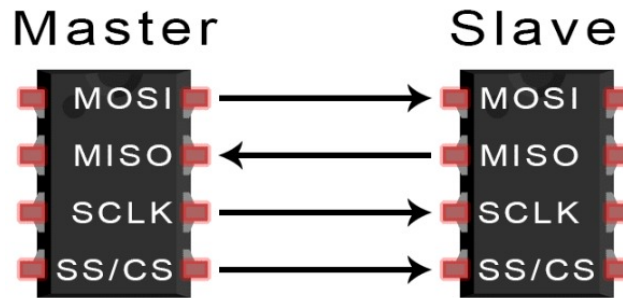


Figura 2.13: Protocolo de Comunicação SPI [32]

Para além disto, é possível aplicar um elevado nível de segurança recorrendo a alguns protocolos de segurança tais como o TLS, permitindo uma comunicação segura de dados, evitando o seu acesso indevido. Estes protocolos são fundamentais para a integração eficiente dos componentes do sistema, garantindo a comunicação robusta entre os dispositivos envolvidos no projeto.

2.4 Revisão da Literatura

O desenvolvimento de sistemas automáticos para lojas e supermercados enquadra-se no contexto da Indústria 4.0. Um estudo relevante nesta área é o estudo referido em [33], que analisa o impacto das tecnologias emergentes na automação de processos logísticos e comerciais, destacando a eficiência e a flexibilidade proporcionadas pelos sistemas inteligentes e conectados. A evolução das máquinas de venda automática e dos sistemas de autoatendimento é um reflexo direto das tendências tecnológicas que permitem maior autonomia e otimização dos processos. Diversos estudos abordam a aplicação de sensores, redes de comunicação e algoritmos de otimização em soluções para retalho automatizado, evidenciando a relevância deste projeto no contexto académico e industrial.

Outro estudo relevante é o trabalho de Manvi et al. [34], que apresenta um sistema de retalho automatizado baseado em tecnologia RFID. O artigo discute como a identificação por radiofrequência pode melhorar a gestão de inventário e a experiência do consumidor, eliminando a necessidade de *checkout* manual e reduzindo os erros operacionais. Esta abordagem está alinhada com o objetivo do presente projeto, que pretende desenvolver um sistema de automação para supermercados, otimizando a recolha e entrega de produtos ao cliente de forma eficiente.

No mundo atualmente, a implementação de tecnologias de comunicação como *Wi-Fi*, *Bluetooth* e *MQTT* em sistemas de retalho automatizado está a ser uma tendência crescente para garantir uma integração fluida entre dispositivos e servidores. O estudo de Manvi reforça a importância dessas soluções para criar um ecossistema interconectado, onde sensores, atuadores e unidades de processamento trabalham de forma coordenada para oferecer uma experiência de compra mais ágil e eficiente. Este conceito é também aplicado na conceção do presente projeto, garantindo um funcionamento robusto e adaptável às necessidades do mercado atual.

O projeto inclui uma interface gráfica para interagir com o cliente, sendo esta o principal meio de interação do utilizador. Por isso, é fundamental garantir uma experiência intuitiva e envolvente, facilitando a navegação, agilizando o processo de compra e proporcionando maior conveniência ao cliente. Deste modo, um estudo muito relevante nesta área é o de Brasel e Gips [35] que investiga o impacto das interfaces gráficas táteis no comportamento do consumidor, analisando como a interação com ecrãs sensíveis ao toque influencia as decisões de compra. A acessibilidade e um *design* intuitivo são fatores essenciais para garantir uma experiência de utilizador positiva e aumentar a adesão à solução. A implementação de um ecrã interativo permitirá que os clientes selecionem produtos, personalizem pedidos e efetuem pagamentos de forma eficiente, alinhando-se com as tendências de digitalização no retalho.

O estudo referido em [36], tem como objetivo demonstrar a implementação e sistematização de um mercado inteligente focado na conveniência. A pesquisa destaca os principais componentes de um mercado autónomo, incluindo tecnologias como sistemas de identificação por imagem, inteligência artificial e sensores, além de infraestrutura composta por câmaras e alarmes. O artigo também aborda os desafios e oportunidades para a implementação desses mercados no Brasil, ressaltando a necessidade de investimentos em tecnologia e a adaptação dos consumidores a novas formas de atendimento. A relevância deste trabalho para a área de gestão comercial reside na inovação que os supermercados autónomos representam, com potencial para transformar significativamente a maneira como os consumidores fazem as compras, oferecendo maior conveniência e horários de funcionamento mais flexíveis.

Além disso, um projeto citado em [37] envolve o sistema denominado *Automatic Retail Checkout* (ARC) (sistema *self-service kiosk* [38]), que visa tornar o processo de *checkout* em lojas mais rápido, autónomo e conveniente, reduzindo a dependência de operadores humanos. O sistema utiliza uma rede neuronal convolucional para identificar objetos colocados sob uma *webcam*. Para avaliar o sistema proposto, os autores criaram um conjunto de dados de imagens de 100 produtos de retalho locais de várias categorias, totalizando 31.000 imagens. Nos testes realizados, o modelo alcançou uma precisão de 94,5% na identificação dos produtos. Este nível de precisão

demonstra a eficácia do sistema ARC em reconhecer corretamente os itens, mesmo considerando variações de orientação e iluminação durante a captura das imagens. No entanto, os autores observaram que algumas confusões ocorreram entre produtos visualmente semelhantes, sugerindo que melhorias adicionais, como o uso de câmaras de maior qualidade e técnicas avançadas de pré-processamento de imagens, poderiam aumentar ainda mais a precisão do sistema. Este trabalho é relevante, pois demonstra a aplicação de técnicas de visão computacional e aprendizagem profunda para automatizar o processo de *checkout* em supermercados, alinhando-se com o conceito de supermercados automáticos.

De outro modo, neste projeto também se utiliza um Raspberry Pi para fazer de servidor para a página *web* e controlar a interface gráfica. Assim, o artigo [39] detalha o processo de configuração de um Raspberry Pi para atuar como um servidor *web* utilizando o NGINX. O autor destaca que o NGINX é rico em recursos e oferece um desempenho superior ao Apache, o que é particularmente importante ao trabalhar com *hardware* de recursos limitados, como o Raspberry Pi. O artigo fornece um guia passo a passo para a instalação e configuração do NGINX no Raspberry Pi, permitindo que ele hospede *websites* e aplicações *web*. Este recurso é valioso porque ajuda na implementação de servidores *web* eficientes em dispositivos de baixo custo. No entanto, além do Raspberry Pi servir de servidor neste projeto, também é utilizado para enviar mensagens ao ESP32 de modo a pedir os produtos. Um artigo interessante que refere a utilização do Raspberry Pi desta maneira é o [40], o que detalha o processo de conexão de um módulo de desenvolvimento ESP32 a um Raspberry Pi através de uma rede local, utilizando o protocolo MQTT para comunicação de dados leves. O MQTT é um protocolo projetado para dispositivos IoT, permitindo uma comunicação leve entre dispositivos e servidores. No exemplo apresentado, o Raspberry Pi atua como um *broker* MQTT (servidor), enquanto o ESP32 funciona como um cliente que publica dados de um sensor de luz para o *broker* MQTT. Este estudo é relevante para o projeto, pois demonstra a viabilidade de utilizar o Raspberry Pi para controlar e monitorizar dispositivos ESP32 através de comunicação MQTT, uma abordagem eficiente para aplicações IoT.

Por fim, o artigo [41] descreve o desenvolvimento de um sistema de controlo para um tapete rolante utilizando o microcontrolador ESP32 e comunicação *WiFi*. O sistema permite a monitorização e controlo remoto do tapete, proporcionando flexibilidade e eficiência no processo industrial. A implementação inclui a criação de uma interface *web* para o utilizador, permitindo o acesso e controlo da passadeira rolante através de dispositivos conectados à rede *WiFi*. Os resultados demonstram a viabilidade do uso do ESP32 em aplicações industriais, destacando sua capacidade de comunicação sem fio e processamento eficiente.

Capítulo 3

Arquitetura do Sistema

A concepção de um supermercado automático exige a integração eficiente de componentes elétricos, sistemas de *software* e mecanismos físicos. Este capítulo proporciona uma visão clara da infraestrutura do supermercado automático, destacando as soluções tecnológicas implementadas para garantir um processo autónomo e eficiente.

3.1 Arquitetura Geral

A Figura 3.1 apresenta o diagrama geral do sistema, no qual é possível observar a interconexão dos principais componentes e os protocolos de comunicação utilizados. O microcontrolador ESP32 comunica com os dispositivos periféricos, tais como os servomotores, o tapete rolante e outros sensores. O Raspberry Pi comunica diretamente com o *display* e por MQTT com o ESP32. A integração desses componentes permite uma operação coordenada e eficiente do sistema, sendo fundamental para o seu desempenho adequado.

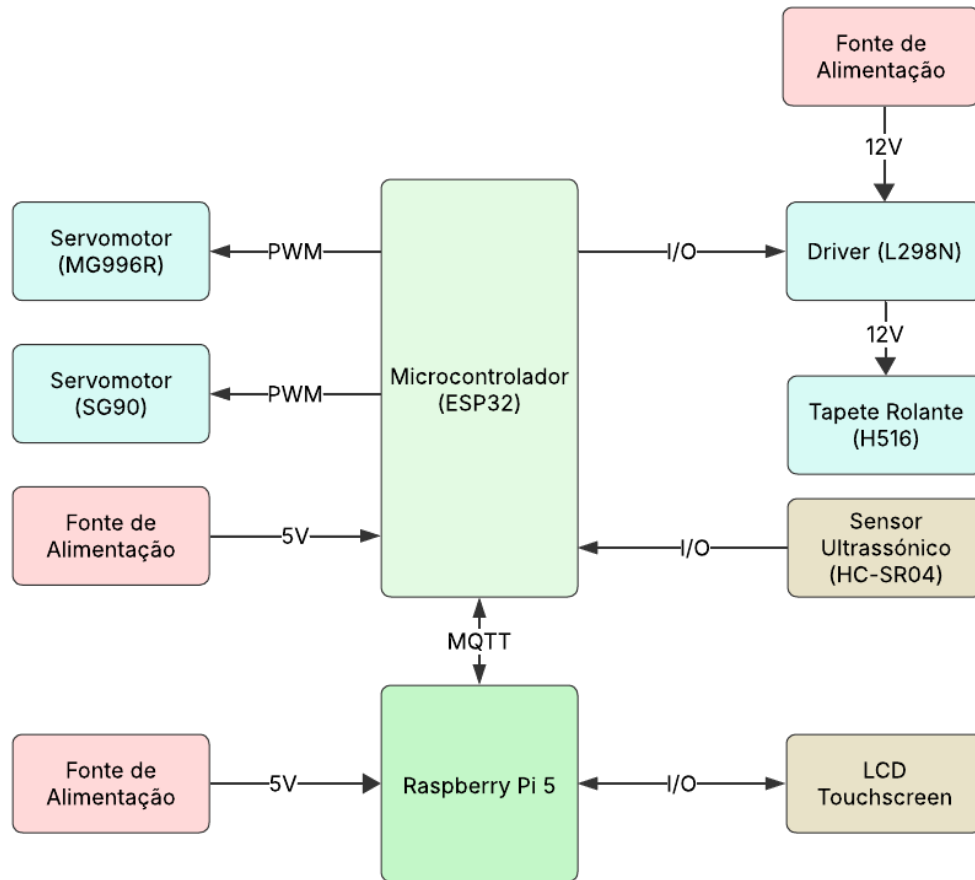


Figura 3.1: Arquitetura geral do sistema

A interligação dos componentes e o projeto do sistema têm como objetivo garantir uma operação eficiente e flexível, permitindo a adaptação do sistema a diferentes necessidades, bem como a sua escalabilidade em projetos futuros. A estrutura central do sistema é composta por dois elementos principais: um Raspberry Pi, que atua como servidor e interface gráfica do sistema, e um ESP32, responsável pelo controle dos mecanismos físicos, incluindo os dispensadores automáticos e os tapetes rolantes.

O Raspberry Pi desempenha um papel fundamental na gestão da interface com o utilizador e na coordenação dos pedidos. Ele hospeda uma página *web* interativa, acessível através do ecrã tátil instalado no supermercado. Esta interface permite ao cliente selecionar os produtos desejados, visualizar o carrinho e confirmar a compra. Além disso, o Raspberry Pi funciona como servidor MQTT, sendo responsável por encaminhar comandos e informações entre os diferentes componentes do sistema.

O ESP32 é o responsável pelo controlo físico da entrega dos produtos. Ele recebe os pedidos do Raspberry Pi através de mensagens MQTT, processa-os e aciona os dispensadores automáticos para libertar os produtos selecionados. Após a libertação dos produtos, o ESP32 controla o funcionamento do tapete rolante, que transporta

os produtos até à zona de recolha do cliente. A comunicação entre o Raspberry Pi e o ESP32 é baseada no protocolo MQTT, garantindo uma transmissão eficiente e em tempo real dos comandos e estados do sistema.

O funcionamento do sistema pode ser descrito em quatro etapas principais:

1. **Seleção e Confirmação do Pedido:** O cliente interage com a interface gráfica exibida no ecrã tátil, escolhe os produtos e finaliza a compra. O Raspberry Pi processa este pedido e envia as informações correspondentes ao ESP32 através do protocolo MQTT.
2. **Ativação dos Dispensadores:** O ESP32 interpreta os comandos recebidos e aciona os dispensadores automáticos, libertando os produtos selecionados.
3. **Transporte dos Produtos:** Após a libertação dos produtos, o ESP32 controla o funcionamento do tapete rolante, garantindo que os itens sejam deslocados corretamente até à zona de recolha.
4. **Conclusão e Notificação:** Quando todos os produtos chegam à zona de recolha, o ESP32 envia uma mensagem de confirmação ao Raspberry Pi, que pode notificar o cliente através da interface gráfica.

Esta abordagem modular permite um controlo eficiente e escalável do supermercado automático, garantindo que o sistema possa ser expandido para integrar novos módulos, dispensadores ou métodos de comunicação no futuro.

3.2 *Hardware*

Nesta secção, será apresentada a descrição dos principais componentes de *hardware* utilizados no desenvolvimento do projeto. Cada componente desempenha um papel crucial na operação do sistema, garantindo precisão, segurança e facilidade de uso.

O sistema para controlo dos dispensadores e tapetes rolantes foi concebido com base no microcontrolador ESP32-WROOM-32, que foi escolhido pela sua versatilidade e capacidade de suportar os requisitos do projeto. O ESP32 é alimentado com uma tensão de 3,3 V, sendo este o valor de referência para o funcionamento do microcontrolador e dos periféricos associados. Contudo, o *kit* de desenvolvimento utilizado no projeto, que inclui o ESP32, oferece a possibilidade de ser alimentado com tensões mais elevadas, nomeadamente 5,0 V, devido à presença de um regulador de tensão integrado. Este regulador permite garantir que a alimentação dos diversos componentes do sistema seja feita de forma estável e sem a necessidade de ajustes adicionais.

3.2.1 Raspberry Pi 5

O Raspberry Pi 5 foi escolhido para este projeto devido ao seu alto desempenho e capacidade de integração com diferentes dispositivos. Equipado com um processador Quad-core ARM Cortex-A76 a 2,4 GHz, oferece um processamento eficiente para aplicações exigentes, como a gestão da interface gráfica do utilizador. Conta com 4 GB de memória RAM LPDDR4X, permitindo a execução fluida de *software* sem comprometer o desempenho (Figura 3.2).

Para armazenamento, suporta cartões *micro-Secure Digital* (SD) e possui uma interface *Peripheral Component Interconnect Express* (PCIe) para conexão de *Solid-State Drive* (SSD) *Non-Volatile Memory Express* (NVMe), garantindo maior velocidade e confiabilidade no acesso aos dados. Em termos de conectividade, dispõe de *Wi-Fi* 802.11ac de banda dupla, *Bluetooth* 5.0, Gigabit Ethernet, além de duas portas *Universal Serial Bus* (USB) 3.0 e duas portas USB 2.0, facilitando a comunicação com periféricos. A saída de vídeo é suportada por duas portas *micro-High Definition Multimedia Interface* (HDMI), permitindo a exibição de gráficos em resolução 4K a 60 Hz [42].

O Raspberry Pi 5 também inclui um conector de 40 pinos *General-Purpose Input/Output* (GPIO), essencial para a comunicação com sensores e módulos externos, garantindo a integração com outros componentes do sistema. No contexto deste projeto, este microcontrolador será responsável por controlar a interface gráfica do utilizador, exibindo as opções de seleção dos produtos num ecrã tátil, e por enviar os pedidos ao ESP32 via comunicação série ou *Wi-Fi*, permitindo a execução precisa do dispensador dos itens. A sua escolha justifica-se pela combinação de desempenho avançado, conectividade versátil e capacidade de processamento gráfico, tornando-o ideal para a gestão eficiente do sistema automatizado.



Figura 3.2: Raspberry Pi 5 [43]

3.2.2 ESP32

O ESP32 é o responsável pelo processamento e pela comunicação entre os diferentes componentes de *hardware*. Este é alimentado com uma tensão de 3,3 V, mas pode alimentar com 5,0 V através do regulador de tensão integrado, o que proporciona flexibilidade nos componentes a integrar com este microcontrolador. O ESP32 utiliza

níveis lógicos de 3,3 V nos seus pinos GPIO, sendo compatível com a maioria dos sensores e atuadores de 3,3 V ou 5,0 V quando este nível lógico é necessário.

O ESP32, como ilustrado na Figura 3.3, suporta vários protocolos de comunicação, como SPI, I2C e UART, o que facilita a integração com dispositivos periféricos. No caso deste projeto, o ESP32 usa o protocolo GPIO para controlar o tapete rolante e servomotores, e o protocolo SPI para a comunicação com o módulo RFID. A conectividade *Wi-Fi* e *Bluetooth* integrada no ESP32 também permite expandir o sistema para suportar a monitorização remota ou a integração com redes IoT, o que, no caso do *Wi-Fi*, irá ser explorado no desenvolvimento do projeto para conectar com o Raspberry Pi [44].

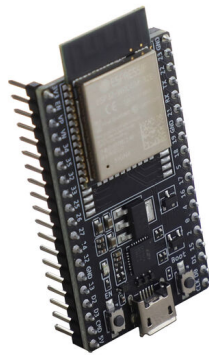


Figura 3.3: ESP32 [45]

3.2.3 Sensor Ultrassónico

O sensor ultrassónico utilizado no projeto é o *High-Conductance Ultrasonic Sensor* (HC-SR04), um sensor que lê distância através de ondas sonoras (Figura 3.4a). Este componente é responsável pela deteção da presença dos produtos antes da entrega ao cliente, enviando uma mensagem ao sistema de que o produto já foi entregue. O sensor mede a distância até ao objeto com base no tempo de retorno de um sinal ultrassónico, proporcionando uma medida confiável e precisa. O HC-SR04 utiliza dois pinos de comunicação: o pino TRIG, que recebe um sinal de 10µs para iniciar a emissão do pulso ultrassónico, e o pino ECHO, que envia de volta um sinal correspondente ao tempo de viagem do pulso. Este sensor opera com uma alimentação de 5,0 V, mas é compatível com o ESP32, pois os sinais de TRIG e ECHO são de 3,3 V, que é o nível lógico do ESP32. A precisão da medição depende da qualidade do sinal ultrassónico e do ambiente, e o HC-SR04 é ideal para detetar objetos em distâncias de 2 cm a 4 metros [46].

O sensor *Waterproof Ultrasonic Distance Sensor Module* (JSN-SR04T), apresentado na Figura 3.4b, é uma opção resistente à água e com características adequadas para ambientes adversos. No entanto, ele apresenta uma limitação significativa para

este projeto porque possui uma distância mínima de detecção de 20 cm, o que o torna inadequado para identificar objetos próximos, como produtos a passar no tapete rolante [47]. Essa sensibilidade reduzida é incompatível com os requisitos do sistema, que exige medições confiáveis em distâncias inferiores a 20 cm para garantir o posicionamento dos produtos no tapete rolante. Por esta razão, o sensor HC-SR04 (Figura 3.4a), com maior sensibilidade para curtas distâncias, foi selecionado como a solução mais adequada para o projeto [48].

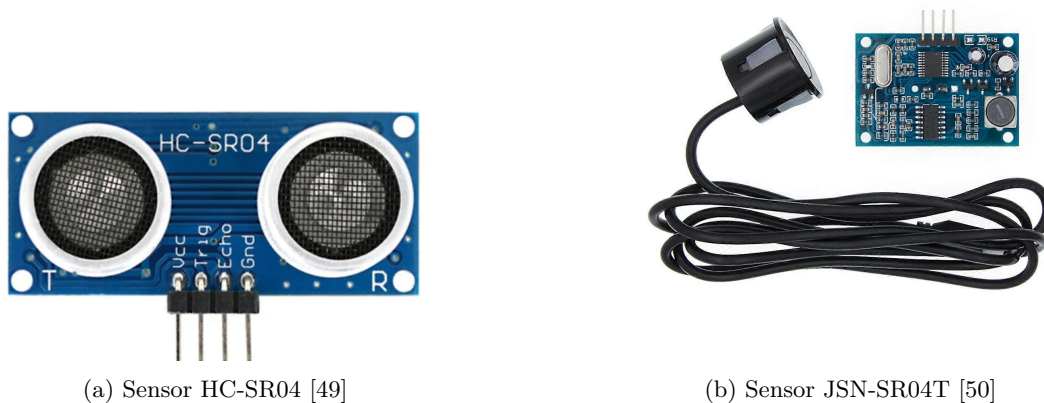


Figura 3.4: Sensores ultrassônicos utilizados no projeto

3.2.4 Tapete Rolante

O tapete rolante (Figura 3.5) é um componente essencial no sistemas de entrega automática de produtos alimentares, sendo responsável pelo transporte eficiente dos produtos desde a zona de armazenamento até à caixa. Este modelo específico apresenta um comprimento de 516 mm, dimensionado para integração em sistemas compactos, como *vending machines* e dispensadores automáticos.

Utiliza uma correia transportadora resistente e possui alimentação de 24,0 V, garantindo um funcionamento estável e seguro. Para otimizar a precisão na entrega dos produtos, o sistema conta com um *feedback*, através de uma placa de pressão, permitindo a detecção da presença dos produtos na passadeira. Este acessório é amplamente utilizado para automatizar a entrega de alimentos e medicamentos, garantindo um processo rápido e confiável nas *vending machines*.



Figura 3.5: Tapete rolante

3.2.5 Display In-Plane Switching (IPS)

O monitor tátil, ou no mundo da tecnologia mais conhecido por *touchscreen* escolhido para este projeto é um ecrã IPS de 7 polegadas com resolução de 1024x600 pixels, proporcionando uma excelente qualidade de imagem com cores vivas e amplo ângulo de visão (Figura 3.6). Equipado com tecnologia capacitiva de 5 toques simultâneos, permite uma interação fluida e responsiva, essencial para a interface gráfica do utilizador.

A conectividade é garantida através de uma porta HDMI para transmissão de vídeo e uma porta USB para alimentação e funcionalidade tátil. O ecrã é compatível com uma ampla variedade de dispositivos, incluindo o Raspberry Pi 5, garantindo integração sem necessidade de *drivers* adicionais. Além disso, o monitor possui um *design* compacto e leve, ideal para sistemas embebidos e aplicações interativas, como a seleção de produtos neste projeto.

Com suporte para o ajuste de brilho e um painel de vidro resistente, oferece durabilidade e visibilidade aprimorada em diferentes condições de iluminação. A combinação de alta qualidade de imagem, sensibilidade ao toque e fácil conectividade torna este monitor uma escolha eficiente para a interface do sistema de sistemas de entrega automatizada [51].



Figura 3.6: LCD [52]

3.2.6 Servomotores

Os servomotores SG90 (Figura 3.7) são utilizados para controlar o dispensador de produtos mais leves através de uma mola helicoidal. Estes atuadores permitem um controlo preciso do movimento, assegurando que a quantidade correta de produtos seja dispensada. Desta forma, cada servomotor pode ser configurado para rotacionar um ângulo específico, o que é essencial para libertar apenas um produto de cada vez e evitar erros.

Os servomotores SG90 operam com uma tensão de alimentação de 5,0 V e são controlados por sinais *Pulse-width Modulation* (PWM). O ESP32 gera o sinal PWM para controlar a posição do servo, com um ciclo de trabalho ajustável que determina a posição do braço do servomotor. O controlo de posição é feito variando a largura do pulso, com um intervalo de 1 ms a 2 ms, o que corresponde a um ângulo de movimento de 0° a 180°. O servo utiliza um nível lógico de 3,3 V para os sinais de controlo, sendo plenamente compatível com o ESP32, que pode gerar sinais PWM diretamente nos pinos configurados [53].



Figura 3.7: Servomotor SG90 [53]

O servomotor MG996R é um atuador de alto binário amplamente utilizado em aplicações que exigem força e precisão (Figura 3.8). Este modelo opera com uma tensão de alimentação entre 4,8 V e 7,2 V, sendo otimizado para 6,0 V, que corresponde a um melhor desempenho. Com um binário máximo de aproximadamente 9,4 kg · cm a 6,0 V, este servomotor é adequado para empurrar produtos mais pesados no sistemas de entrega automatizada [54].

O MG996R possui uma engrenagem metálica que garante maior durabilidade e resistência ao desgaste, sendo ideal para operações contínuas e de alto esforço mecânico. A sua velocidade de resposta é de 0,17 s/60° a 6,0 V, o que proporciona um movimento rápido e eficiente. Além disso, tem um ângulo de rotação típico de 0° a 180°, controlado por um sinal PWM com largura de pulso entre 500 µs e 2500 µs [55].

No contexto deste projeto, o MG996R será utilizado para empurrar produtos mais pesados, garantindo que a dispensa seja realizada de forma precisa e confiável. A sua robustez e alto binário fazem dele a escolha ideal para esta aplicação, assegurando um funcionamento eficiente e consistente ao longo do tempo.



Figura 3.8: Servomotor MG996R [56]

3.2.7 Módulo RFID

O módulo RFID MFRC-522 (Figura 3.9) pode ser utilizado para o controlo de acesso ao sistema, garantindo que apenas utilizadores autorizados possam parar a dispensa de produtos para se poder encher os dispensadores novamente. Este módulo funciona através da identificação de cartões RFID, tornando o sistema mais seguro e confiável. Ao identificar um cartão válido, o módulo permite o acesso às funcionalidades do dispensador, proporcionando uma camada extra de segurança no uso do dispositivo.

O módulo RFID utiliza o protocolo SPI para comunicação com o ESP32 e opera com uma alimentação de 3,3 V. O módulo RFID comunica com o ESP32 através de 5 pinos: SDA, SCK, MISO, MOSI e RST, todos com níveis lógicos de 3,3 V. O cartão RFID é lido através da tecnologia de indução magnética, permitindo uma autenticação rápida e segura [57].



Figura 3.9: Módulo RFID [57]

3.3 Metodologias de Teste

Nesta secção, descrevem-se as metodologias de teste definidas para avaliar o desempenho, fiabilidade e integração dos diversos componentes do sistema de automação do supermercado. A abordagem adotada segue uma lógica incremental, começando pela validação individual dos componentes, passando pela integração dos módulos e culminando com testes ao sistema completo em condições próximas da operação real.

3.3.1 Testes Funcionais dos Componentes Individuais

Estes testes têm como objetivo validar o correto funcionamento de cada componente do sistema, isoladamente. A Tabela 3.1 resume os principais componentes e os respetivos aspetos a testar.

Tabela 3.1: Testes funcionais dos componentes individuais

Componente	Aspetos a Testar
Servomotores	Precisão do ângulo, repetição do movimento, fiabilidade
Sensor ultrassónico	Precisão da medição, deteção de diferentes objetos e materiais
Tapete rolante	Transporte de diferentes produtos, estabilidade e aderência
Monitor tátil	Precisão do toque, responsividade e estabilidade da ligação

3.3.2 Testes de Integração do Sistema

Após a validação individual dos componentes, serão realizados testes para verificar o comportamento do sistema como um todo, focando-se na comunicação entre módulos e na execução de tarefas coordenadas. A Tabela 3.2 apresenta os principais pontos a testar.

Tabela 3.2: Testes de integração do sistema

Aspetos a Testar
Comunicação entre o <i>website</i> e o ESP32 via MQTT
Coordenação entre dispensador, tapete rolante e sensores
Funcionamento em cenários de uso real com múltiplos produtos
Comportamento perante falhas (comunicação, produto em falta)

3.3.3 Testes de Segurança e Fiabilidade

Estes testes visam garantir que o sistema é seguro, robusto e capaz de recuperar de falhas. A Tabela 3.3 resume os aspetos a validar.

Tabela 3.3: Testes de segurança e fiabilidade

Aspetos a Testar
Comportamento em caso de falha de energia ou comunicação
Capacidade de recuperação automática do sistema
Estabilidade durante funcionamento prolongado

3.3.4 Testes de Precisão e Quantidade

Nesta fase, o objetivo é validar a correta dispensa e transporte dos produtos, conforme previsto no sistema. A Tabela 3.4 resume os aspetos a considerar.

Tabela 3.4: Testes de precisão e quantidade

Aspetos a Testar
Calibração dos servomotores para libertar quantidades corretas
Transporte eficiente de produtos com diferentes pesos e tamanhos
Estabilidade do transporte sem quedas ou bloqueios
Validação do sistema em sequências de pedidos consecutivos

Esta estratégia de testes visa assegurar que o sistema cumpre os requisitos definidos, funcionando de forma previsível, segura e fiável, tanto na operação normal como em situações excecionais.

3.4 Lista de Componentes

A lista de componentes ou também conhecida por *Bill of Materials* (BOM) apresentado na Tabela 3.5 descreve os componentes comprados para o desenvolvimento do supermercado automático. A seleção dos componentes foi realizada com base nas funcionalidades necessárias, no custo-benefício e na disponibilidade no mercado.

No entanto, é de salientar que nem todos os componentes comprados vão ser utilizados no projeto final, alguns deles foram adquiridos para testes, a fim de determinar qual seria o mais adequado para o uso pretendido (como é o exemplo do JSN-SR04T ou do módulo RFID (MFRC-522)). Desta forma, os principais elementos observados na Tabela 3.1 incluem o sensor ultrassónico (HC-SR04) para deteção dos produtos, e servomotores (SG90 e MG996R) para a dispensa de diferentes produtos. Adicionalmente, foram utilizados o ESP32-WROOM-32D como microcontrolador secundário e o Raspberry Pi 5 como microcontrolador principal, um *display* IPS (GeekPi 10) para interação do utilizador e um tapete rolante para o transporte de alimentos até ao cliente.

Tabela 3.5: Lista de componentes

Nome	Descrição	Quantidade	Preço p/uni	Total p/uni
MFRC-522	Módulo RFID	1	1,71€	1,71€
SG90	Servomotor	3	1,10€	3,30€
MG996R	Servomotor	4	4,25€	17,01€
Mola	Mola Helicoidal	1	9,51€	9,51€
H516	Tapete Rolante	1	27,27€	27,27€
HC-SR04	Sensor ultrassónico	2	0,84 €	1,68
JSN-SR04T	Sensor ultrassónico	1	2,62 €	2,62 €
GeekPi 10	Monitor	1	66,11€	66,11€
Raspberry Pi 5	Raspberry Pi	1	87,40€	87,40€

Por fim, é apresentado o custo aproximado do protótipo final na Tabela 3.6. O custo total dos componentes necessários para o protótipo é de 200,73 €, demonstrando ser uma solução acessível para o sistema de um supermercado inteligente.

Tabela 3.6: Lista de componentes utilizados no protótipo

Nome	Descrição	Quantidade	Preço p/uni	Total
SG90	Servomotor	1	1,10€	1,10€
MG996R	Servomotor	2	4,25€	8,50€
Mola	Mola Helicoidal	1	9,51€	9,51€
H516	Tapete Rolante	1	27,27€	27,27€
HC-SR04	Sensor ultrassónico	1	0,84 €	0,84€
GeekPi 10	Monitor	1	66,11€	66,11€
Raspberry Pi 5	Raspberry Pi	1	87,40€	87,40€
			Total	200,73€

Capítulo 4

Desenvolvimento do Protótipo

Este capítulo descreve o processo de materialização do projeto, detalhando as seguintes etapas para integrar os diferentes componentes e assegurar o funcionamento do supermercado inteligente. São abordados os principais desafios enfrentados e as soluções implementadas, incluindo a configuração utilizada de *hardware*, o desenvolvimento do *firmware* e a integração de sensores e servomotores. A implementação foi conduzida com base no planeamento delineado anteriormente, garantindo que os objetivos definidos fossem cumpridos de forma eficaz e segura.

4.1 Visão Geral da Arquitetura

Este projeto integra componentes de *hardware* e *software* com o objetivo de automatizar o processo de compra num supermercado, oferecendo uma experiência mais eficiente e autónoma ao cliente. A arquitetura do sistema é composta por:

- **Microcontrolador ESP32:** Responsável pelo controlo local dos dispensadores automáticos e do tapete rolante, garantindo a interação com sensores e atuadores, bem como a comunicação com o sistema central através de *Wi-Fi*.
- **Sensores e Atuadores:** Incluem sensores de presença ultrassónicos para deteção de produtos, motores servo para acionar mecanismos do dispensador, e motores *Direct Current* (DC) para controlo dos tapetes rolantes.
- **Raspberry Pi 5:** Atua como servidor central e interface gráfica para os utilizadores, permitindo a seleção de produtos através de um monitor *touchscreen*.

Também gere a comunicação com a interface *web* externa para compras remotas.

- **Interface Web (Flask/Fast Application Programming Interface (API)):**
Permite aos clientes consultar produtos, efetuar encomendas e gerir o seu carrinho, tanto na loja física como remotamente através da Internet.

De forma a ilustrar a interligação entre os diversos componentes descritos, a Figura 4.1 apresenta o diagrama de blocos da arquitetura do sistema. Neste diagrama, é possível visualizar a comunicação entre o ESP32, responsável pelo controlo dos dispensadores e dos tapetes rolantes, o Raspberry Pi 5 que atua como servidor central e interface gráfica, e a aplicação *web* que permite ao cliente interagir com o sistema. A integração destes elementos é feita através de comunicação *Wi-Fi* e protocolos como *Hypertext Transfer Protocol Secure* (HTTPS) e MQTT, garantindo a sincronização de dados em tempo real entre as diferentes partes do sistema.

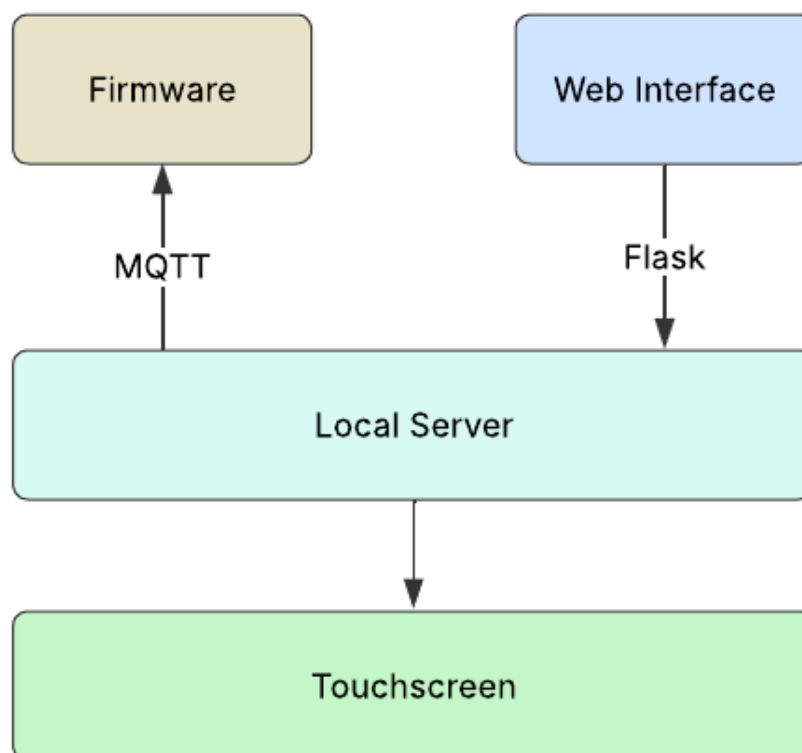


Figura 4.1: Diagrama de blocos da arquitetura do sistema

4.2 Desenvolvimento da Estrutura Física

4.2.1 Visão Geral do Protótipo

A ideia inicial para o desenvolvimento da estrutura do dispensador de produtos consistia em construir todo o modelo utilizando cartão. Este primeiro passo revelou-se extremamente útil para obter um *design* inicial do projeto, permitindo uma visualização preliminar das dimensões e funcionalidades desejadas. Contudo, dado o grande interesse em modelação tridimensional e a necessidade de uma solução mais robusta e precisa, optou-se por adquirir uma impressora 3D para dar continuidade ao desenvolvimento.

Assim, foi iniciada uma pesquisa sobre os melhores *softwares* para criar as peças que iriam ser necessárias para este projeto. Após a análise, foi concluído que o Autodesk Fusion 360 seria a opção mais adequada, dado o seu vasto conjunto de ferramentas e possibilidades para o *design* do projeto. Com o uso do Fusion 360, foi projetado um mecanismo para montar a mola no motor, a caixa de recolha dos produtos, entre outras peças que conduzem os produtos até à mesma, enquanto o Bambu Studio foi utilizado para preparar os arquivos para impressão 3D. A impressora escolhida foi a BambuLab A1 [58], cujas capacidades tecnológicas nos permitiram realizar impressões de alta precisão e qualidade (Figura 4.2).

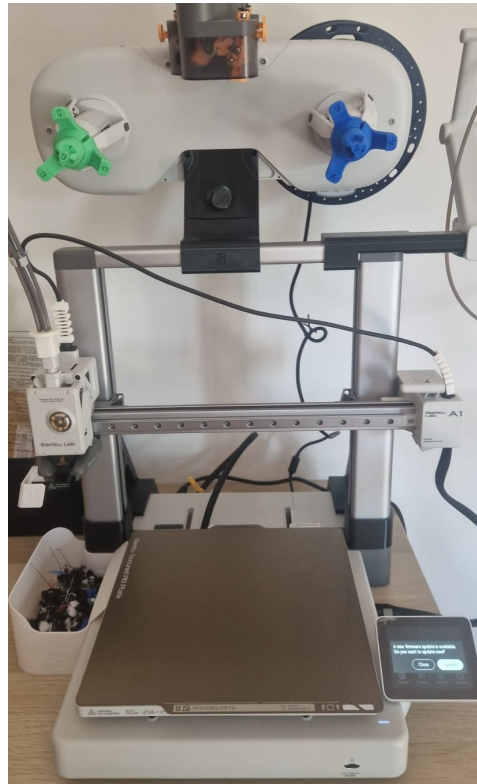


Figura 4.2: Impressora BambuLab A1

4.2.2 Modelo Tridimensional

Para a concepção tridimensional da estrutura física do dispensador, foram desenhadas diversas peças fundamentais para garantir o correto funcionamento do sistema. O Autodesk Fusion 360 [59] foi a ferramenta utilizada para esta tarefa, permitindo desenhar com elevada precisão todos os componentes que fazem parte do protótipo.

Esta modelação tridimensional teve em conta não só as dimensões dos componentes eletrónicos e mecânicos, mas também o tipo de esforços a que as peças estariam sujeitas, bem como a facilidade de montagem e impressão 3D.

Abaixo são apresentadas algumas das peças tridimensionais mais relevantes concebidas para o protótipo.

Peça de Ligação entre a Mola e o Servomotor

Uma das peças desenvolvidas com mais cuidado e atenção foi o suporte de ligação entre a mola helicoidal — onde os produtos são inseridos — e o servomotor responsável pela sua rotação. Esta peça tem um papel crucial na mecânica do dispensador, pois garante uma transferência eficiente do movimento do motor para a mola, possibilitando o avanço controlado dos produtos até ao tapete rolante.

O desenho tridimensional teve em conta não só as dimensões e posicionamento do servomotor, como também o tipo de esforço mecânico a que a peça estaria sujeita durante o funcionamento do sistema. Para tal, foram realizadas diversas interações no modelo até se alcançar uma geometria que garantisse robustez estrutural e uma montagem simples.

A Figura 4.3 apresenta duas vistas distintas da peça desenvolvida: a vista frontal e a vista traseira.

Funil de Direcionamento para o Tapete Rolante

Outra peça fundamental no desenho tridimensional do dispensador é o funil de direcionamento dos produtos. Esta estrutura foi desenvolvida com o objetivo de garantir que os produtos libertados pela mola helicoidal sejam corretamente canalizados para o tapete rolante, minimizando o risco de obstruções ou quedas fora da zona pretendida.

O modelo inclui duas abas laterais inclinadas que conduzem o produto para o centro da base, funcionando como um guia. A geometria em forma de funil foi cuidadosamente dimensionada para se adaptar ao volume dos produtos típicos utilizados no sistema, proporcionando uma transição suave e eficiente entre o dispensador e o mecanismo de transporte.

A Figura 4.4 mostra a vista isométrica do modelo tridimensional do funil de direcionamento, desenvolvido no Autodesk Fusion 360.

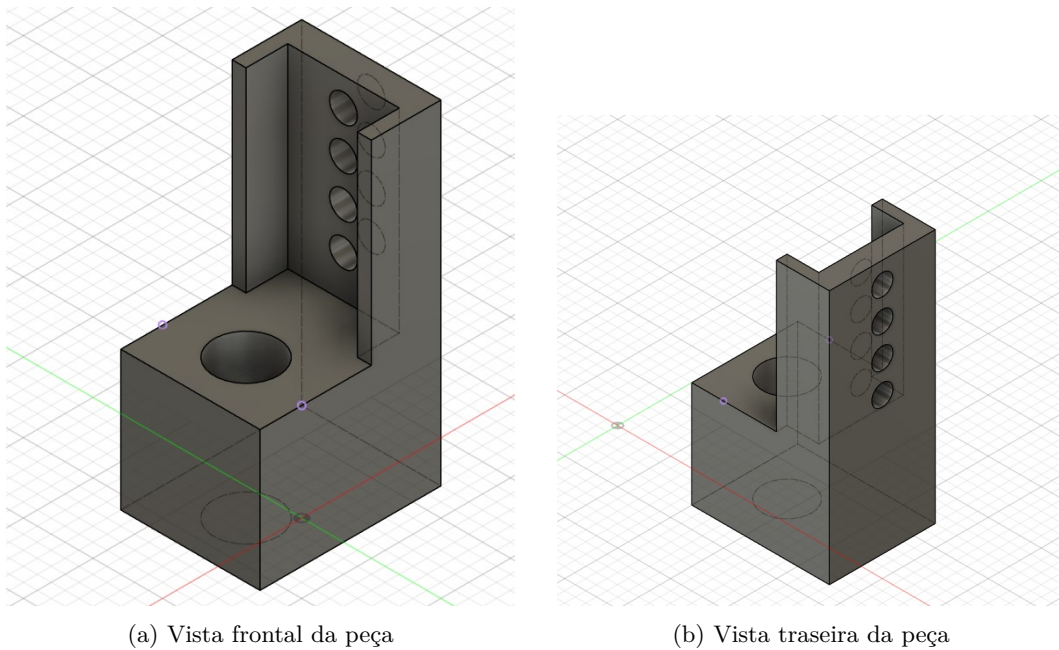


Figura 4.3: Modelo tridimensional da peça de ligação entre a mola helicoidal e o servomotor, desenvolvido no Autodesk Fusion 360.

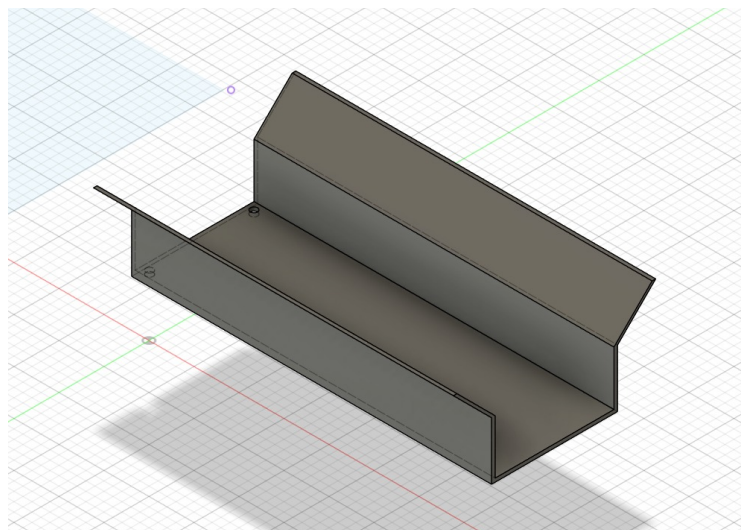


Figura 4.4: Modelo 3D do funil de direcionamento dos produtos do dispensador para o tapete rolante.

Caixa de Receção no Fim do Tapete Rolante

A última peça do percurso físico dos produtos é a caixa de receção, situada na extremidade do tapete rolante. Esta estrutura foi desenhada com o objetivo de recolher com segurança os produtos dispensados, garantindo que permanecem contidos até serem recolhidos pelo utilizador.

A caixa apresenta um formato cúbico com paredes suficientemente altas para evitar que os produtos saltem para fora, devido à inércia do movimento ou a pequenas irregularidades no avanço do tapete. A sua geometria simples permite uma impressão 3D rápida e uma integração fácil na estrutura geral do sistema.

Além disso, foram adicionadas zonas de fixação na lateral da peça para facilitar a montagem na base do protótipo e manter a estabilidade da caixa durante a operação.

A Figura 4.5 mostra a visualização tridimensional da caixa de receção, modelada no Autodesk Fusion 360.

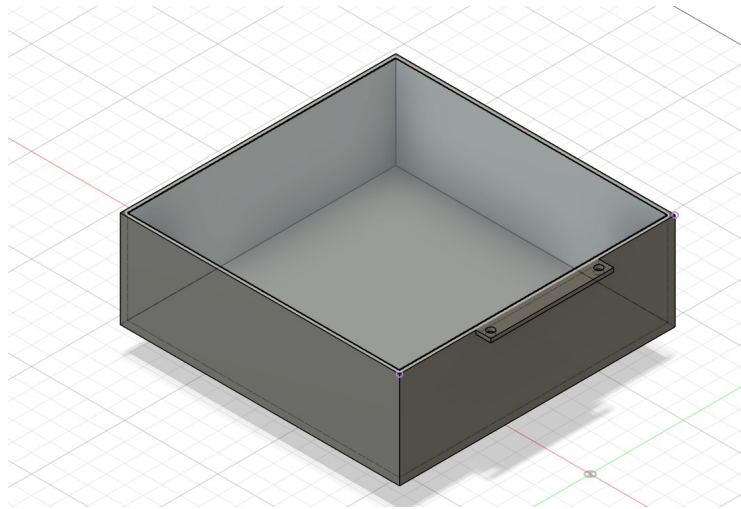


Figura 4.5: Modelo 3D da caixa de receção dos produtos situada no fim do tapete rolante.

Suporte da Mola e do Motor do Dispensador

Para assegurar a estabilidade da mola helicoidal durante a sua rotação e garantir o correto posicionamento do motor, foi desenvolvida uma estrutura de suporte específica. Esta peça tem como principais funções manter o eixo da mola centrado, evitar que esta se desloque lateralmente durante o movimento e proporcionar uma base firme para a fixação do servomotor.

Além disso, inclui uma abertura frontal onde o motor é montado, bem como guias laterais que impedem que o produto saia do percurso previsto antes de chegar ao funil e, posteriormente, ao tapete rolante. O encaixe da mola foi cuidadosamente ajustado à sua geometria, assegurando um contacto firme e estável com o eixo do motor.

A conceção teve também em conta a facilidade de montagem, recorrendo a fixações simples e garantindo compatibilidade com as outras peças do sistema.

A Figura 4.6 apresenta o modelo tridimensional do suporte da mola e do motor, desenhado no Autodesk Fusion 360.

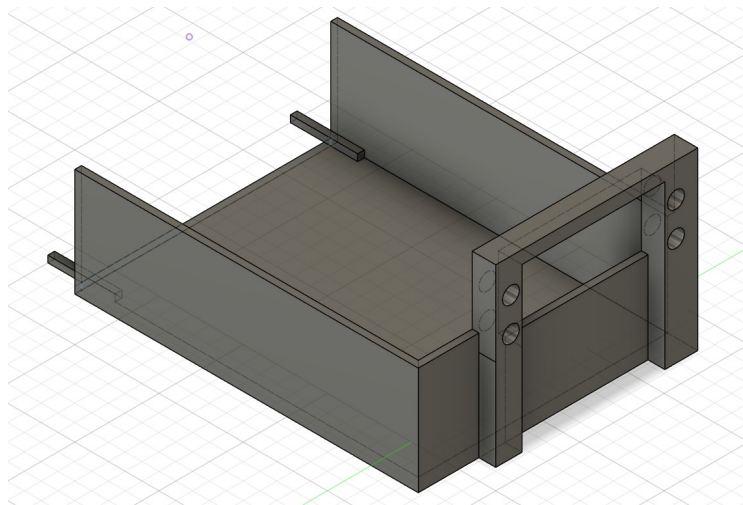


Figura 4.6: Modelo 3D do suporte da mola e do motor do dispensador.

4.2.3 Preparação para Impressão

Com os modelos tridimensionais finalizados no Fusion 360, o passo seguinte consistiu na conversão para o formato `.stl` e respetiva preparação para impressão 3D no *slicer* Bambu Studio — o *software* oficial da BambuLab.

Durante esta fase foram ajustados diversos parâmetros, como a resolução das camadas, o tipo de preenchimento (*infill*), a velocidade de impressão e a necessidade de suportes. A correta orientação das peças na mesa de impressão foi também fundamental para garantir qualidade dimensional, redução de deformações e otimização dos tempos de impressão.

Peça de Ligação entre a Mola e o Servomotor

Para garantir a robustez mecânica desta peça, foi aplicado um preenchimento de 50%, com camadas de 0,2 mm e paredes reforçadas. A orientação foi ajustada de forma a que a zona de encaixe no motor ficasse virada para cima, reduzindo o risco de imperfeições nas áreas de contacto funcional (Figura 4.7).

Funil de Direcionamento para o Tapete Rolante

O funil foi preparado com um preenchimento de 20% e camadas de 0,28 mm, tendo em conta que não está sujeito a esforços mecânicos elevados. A orientação do modelo foi ajustada para que a base inclinada ficasse virada para a mesa, o que permitiu eliminar a necessidade de suportes e garantir uma superfície suave na zona de deslizamento dos produtos (Figura 4.8).

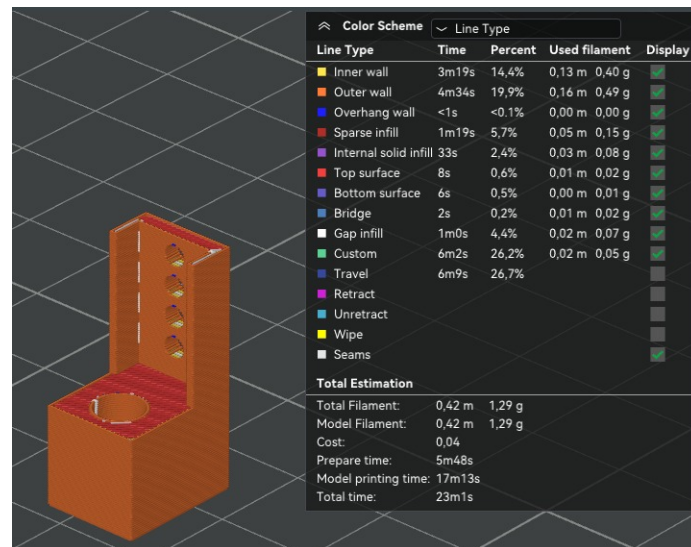


Figura 4.7: Preparação da peça de ligação no Bambu Studio com parâmetros otimizados.

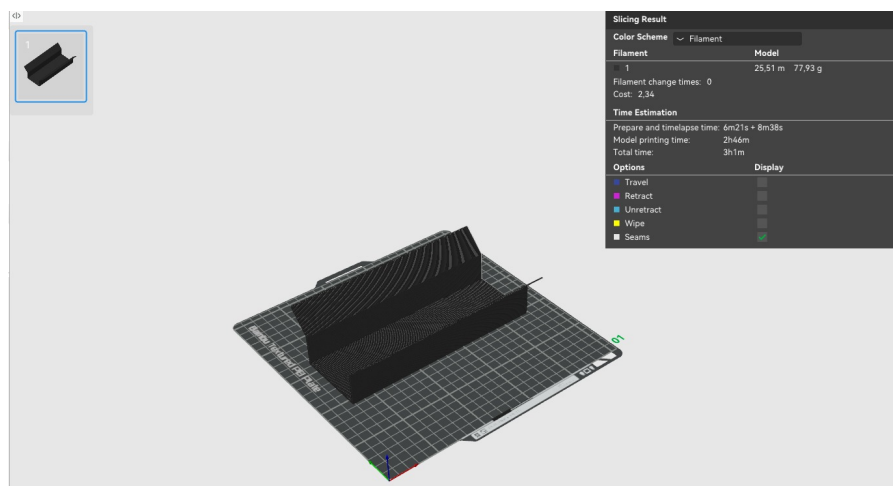


Figura 4.8: Visualização da preparação do funil no Bambu Studio.

Caixa de Receção no Fim do Tapete

A caixa foi preparada com um preenchimento de 15%, dado que a sua função é apenas conter os produtos. A orientação da peça foi mantida na posição natural (com a base na mesa), o que facilitou a impressão e evitou suportes desnecessários, necessitando apenas de suporte no patamar que liga a caixa ao funil anteriormente mostrado (Figura 4.9).

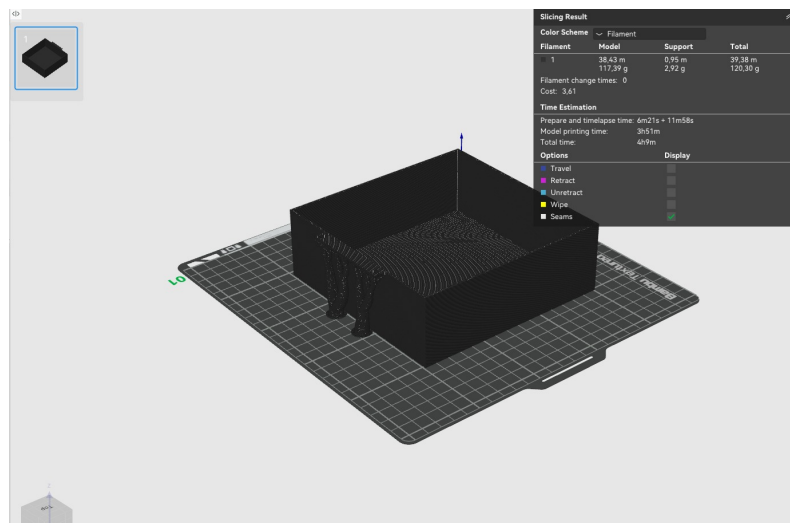


Figura 4.9: Preparação da caixa de receção para impressão.

Suporte da Mola e do Motor

Dado o papel estrutural desta peça, foi utilizado um preenchimento de 40% com paredes reforçadas e orientação otimizada para garantir estabilidade na área de montagem do motor. A impressão foi feita com suportes apenas na zona da montagem do motor por haver diversos buracos, tal como pode ser observado na Figura 4.10.

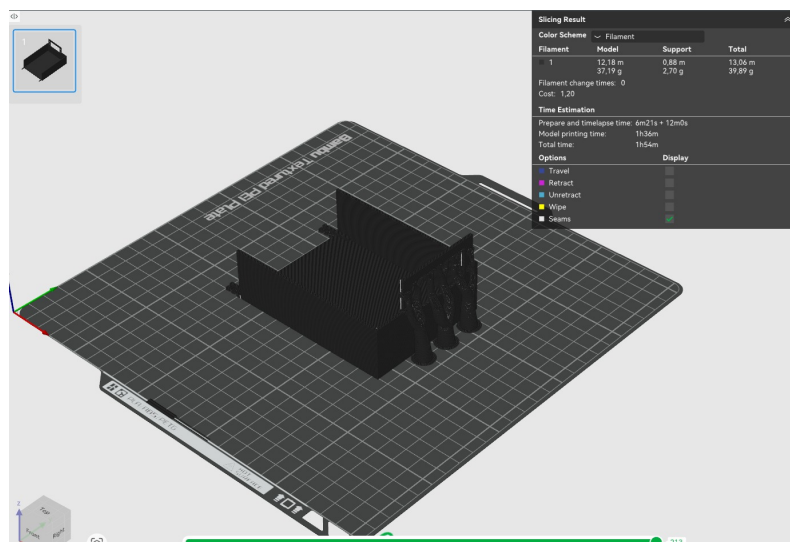


Figura 4.10: Peça do suporte da mola pronta para impressão.

4.2.4 Impressão 3D das Peças

A impressão das peças foi realizada com a impressora 3D BambuLab A1, conhecida pela sua elevada precisão e velocidade. Todos os modelos foram exportados a partir

do Autodesk Fusion 360 e preparados para impressão com parâmetros adaptados a cada tipo de peça e material.

Os materiais utilizados foram escolhidos com base nos requisitos específicos de cada componente, como resistência mecânica, flexibilidade ou facilidade de montagem. Após a impressão, foram realizados pequenos acabamentos manuais, como a remoção de suportes e rebarbas, para garantir a qualidade e compatibilidade com os restantes elementos do sistema.

Peça de Ligação entre a Mola e o Servomotor

Esta peça foi impressa em PETG, um material resistente ao impacto e à deformação, ideal para componentes que estão sujeitos a forças de torção e atrito. A escolha do PETG deveu-se à necessidade de garantir uma boa rigidez estrutural e durabilidade ao longo do tempo.

A impressão decorreu sem complicações, com elevada qualidade na definição das camadas e excelente aderência à mesa. Após a remoção dos suportes, a peça apresentou bom encaixe no eixo do motor e na extremidade da mola helicoidal (Figura 4.11).



Figura 4.11: Peça final impressa em PETG, montada na mola helicoidal.

Funil de Direcionamento para o Tapete Rolante

O funil foi impresso em PETG, dado que não está sujeito a esforços mecânicos relevantes e apenas serve como guia passivo para os produtos. O PETG foi escolhido

pela sua facilidade de impressão, baixo custo e boa definição de detalhe.

A geometria inclinada foi respeitada com sucesso, e a superfície interna apresentou boa suavidade, permitindo o deslizamento eficiente dos produtos em direção ao tapete (Figura 4.12).



Figura 4.12: Funil de direcionamento impresso em PETG.

Caixa de Receção no Fim do Tapete

A caixa de receção foi também impressa em PETG, uma vez que a sua função é simplesmente conter os produtos após a entrega. A simplicidade do modelo permitiu uma impressão rápida, com bom acabamento nas superfícies e sem necessidade de suportes.

Os furos laterais permitiram a fixação segura da peça na base do protótipo (Figura 4.13).



Figura 4.13: Caixa de receção impressa, colocada no fim do tapete rolante.

Suporte da Mola e do Motor

Esta peça exigia maior robustez, pois sustenta o motor e orienta o movimento da mola helicoidal. Assim, foi impressa em PETG, conferindo-lhe maior resistência estrutural e estabilidade dimensional.

A montagem do motor foi facilitada pelas zonas de encaixe frontal e pelas colunas laterais para parafusos. A impressão correu como esperado, com boas tolerâncias para a montagem do motor e da mola (Figura 4.14).



Figura 4.14: Suporte da mola e do motor após impressão e montagem.

4.3 Implementação do *Hardware*

4.3.1 Sensores e Atuadores

O sistema desenvolvido integra diversos sensores e atuadores conectados ao ESP32, funcionando como controlador essencial da componente física do projeto.

Entre os atuadores, destacam-se quatro servomotores, cada um responsável por accionar o mecanismo de dispensa de um tipo distinto de produto. Estes servos estão ligados aos pinos 18, 19, 21 e 22 do ESP32, e são controlados individualmente conforme o *ID* (Identificador) do produto recebido via comunicação MQTT.

Para o transporte dos produtos dispensados, é utilizado um motor DC controlado através de dois pinos digitais (26 e 27), que permitem acionar e inverter a direção do tapete rolante. O tapete é ativado após cada operação de dispensa, garantindo que o produto seja transportado até à zona de recolha.

A deteção do fim do tapete é feita através de um sensor ultrassónico, ligado aos pino 12 e 13. Este sensor serve para parar o tapete assim que o produto seja detetado, assegurando o correto posicionamento dos produtos.

4.3.2 Ligações e Esquemas Elétricos

As ligações elétricas do sistema foram realizadas com base nos requisitos de alimentação e sinal de cada componente, respeitando as especificações do ESP32.

A Tabela 4.1 apresenta um resumo dos pinos utilizados no ESP32 e a sua respetiva função.

Tabela 4.1: Ligações dos pinos do ESP32 aos componentes

Pino do ESP32	Componente
18	Servo do produto 1
19	Servo do produto 2
21	Servo do produto 3
22	Servo do produto 4
26	Motor do tapete (IN1)
27	Motor do tapete (IN2)
12	Sensor de fim de tapete (TRIG)
13	Sensor de fim de tapete (ECHO)

Todos os servos são alimentados através de uma fonte externa de 5 V, comum aos quatro, de forma a garantir corrente suficiente para o acionamento simultâneo. A terra (GND) da fonte externa é ligada ao GND do ESP32 para garantir uma referência comum de sinal.

O circuito do motor do tapete utiliza um controlador H-bridge (o L298N), com as entradas digitais a partir do ESP32 e alimentação própria de 12 V, adequada ao motor DC em uso.

A Figura 4.15 apresenta o esquema geral das ligações entre os componentes.

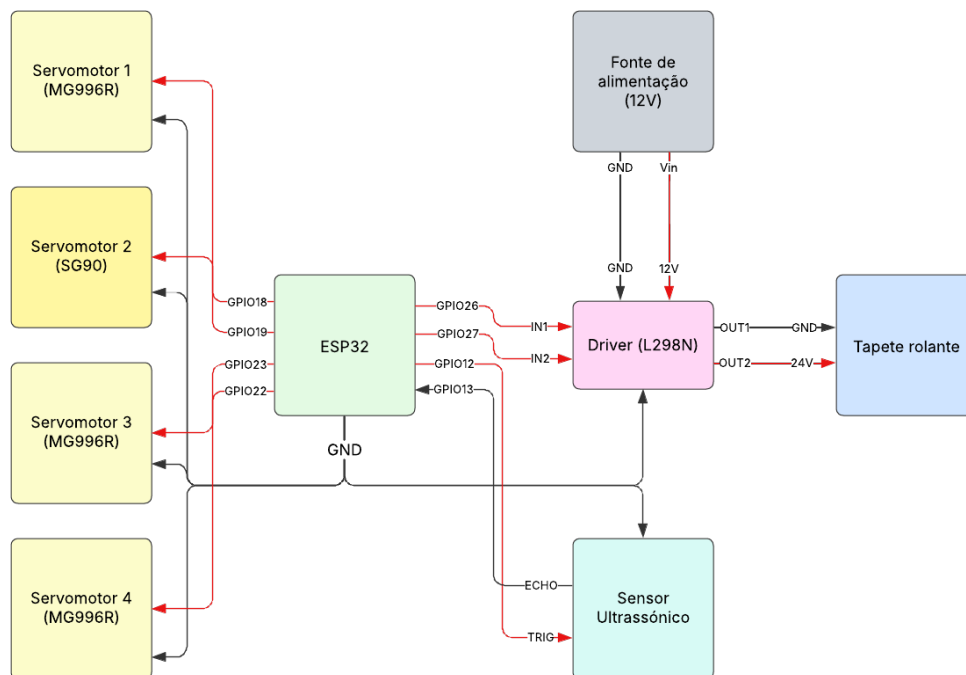


Figura 4.15: Esquema das ligações entre o ESP32, servos, motor do tapete e sensor

4.4 Implementação do *Software*

A componente de *software* deste projeto desempenha um papel fundamental na coordenação entre o sistema físico e a interação com o utilizador, tanto localmente como remotamente. Esta secção descreve a configuração do ambiente de desenvolvimento, o funcionamento interno das aplicações desenvolvidas e a interface *web* destinada aos clientes.

4.4.1 Configuração do Ambiente de Desenvolvimento

Para o desenvolvimento do sistema embarcado, foi utilizado o Visual Studio Code com a extensão PlatformIO, permitindo programar o ESP32 com recurso à linguagem C++ e à biblioteca Arduino.h. Esta combinação oferece flexibilidade e compatibilidade com diversas bibliotecas dedicadas ao controlo de sensores, atuadores e comunicação por *Wi-Fi*.

Já para a parte do servidor central e da interface gráfica no Raspberry Pi 5, o ambiente de desenvolvimento baseou-se em Python, com recurso ao *micro-framework* Flask, possibilitando a criação de um servidor local que gere as funcionalidades principais do sistema e a comunicação com o microcontrolador.

4.4.2 Visão Geral do *Software*

O sistema está dividido em dois grandes blocos de *software*: o *firmware* do microcontrolador e a aplicação *web* no servidor local.

- **Firmware do ESP32:** Responsável pela leitura de sensores (como o sensor ultrassónico), controlo dos motores (servos e tapetes rolantes), e comunicação com o servidor principal através do protocolo MQTT ou HTTP. O código foi desenvolvido de forma modular para permitir fácil manutenção e escalabilidade.
- **Servidor Local:** O Raspberry Pi 5 executa o servidor principal em Python, que recebe comandos do microcontrolador e regista os eventos num sistema local.

4.4.3 Aplicação *Web*

A interface *web* do sistema foi desenvolvida utilizando o *framework* Flask, que permite a criação de aplicações robustas e eficientes para gestão dos processos do supermercado. Para o *design* e responsividade, foi adotado o Bootstrap, que fornece componentes reutilizáveis e uma estrutura flexível para construção da interface. Complementarmente, a camada de *frontend* foi implementada em *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript, proporcionando

uma experiência de utilizador intuitiva e interativa. As funcionalidades incluem a visualização de produtos, gestão do carrinho de compras, *checkout* e pesquisa de produtos.

A interface *web* oferece um conjunto de funcionalidades que visam otimizar a experiência do utilizador:

- **Listagem de Produtos:** Os produtos estão organizados por categorias e são apresentados com imagens e informações detalhadas.
- **Gestão do Carrinho:** O utilizador pode adicionar, remover e atualizar quantidades de produtos no carrinho.
- **Finalização de Compra:** A interface permite ao utilizador introduzir os dados de pagamento e seleccionar a loja de recolha.
- **Pesquisa Dinâmica:** A aplicação permite a pesquisa em tempo real por nome de produto.

A interface está acessível tanto no monitor *touchscreen* do quiosque físico, ligado ao Raspberry Pi, como remotamente, possibilitando aos clientes realizar encomendas *online* e proceder à recolha no supermercado. Para garantir uma manutenção simplificada e escalabilidade otimizada, a aplicação foi desenvolvida de forma modular, estruturando-se nos seguintes componentes:

- **Backend (Flask):** Responsável pela lógica de negócio, gestão de rotas e comunicação com a interface *web*.
- **Frontend (HTML, CSS e JavaScript):** Interface gráfica para interação com o utilizador, construída sobre templates HTML.
- **Ficheiros Estáticos:** Os estilos visuais e interações dinâmicas são geridos através de ficheiros dedicados de CSS e JavaScript.
- **Páginas Dinâmicas:** As páginas são renderizadas dinamicamente de acordo com as requisições do utilizador, nomeadamente:
 - **index.html:** Página principal com listagem dos produtos (Figura 4.16).



Figura 4.16: Página inicial

- **category.html:** Apresenta os produtos filtrados por categoria (Figura 4.17).

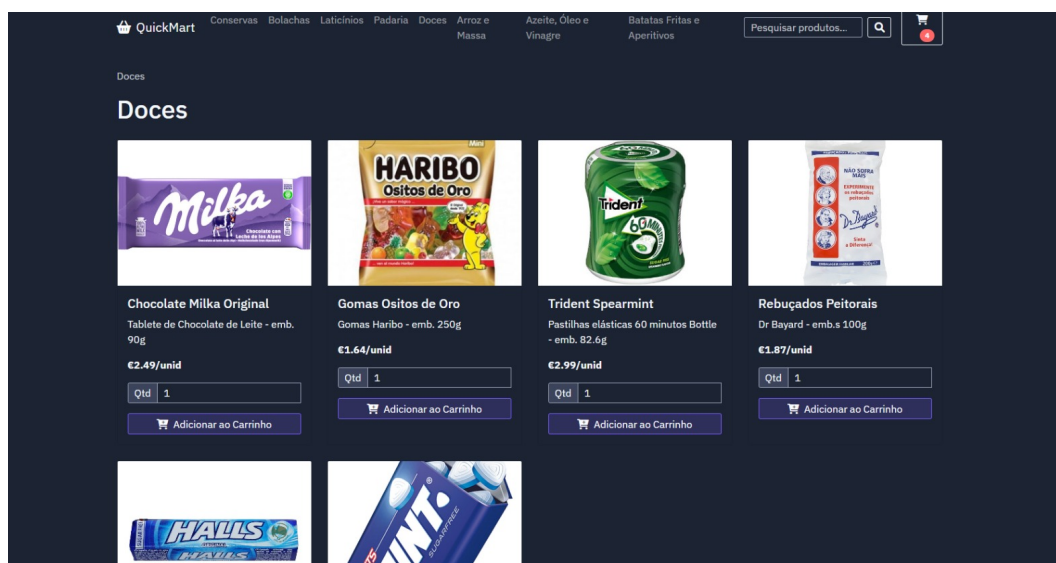


Figura 4.17: Página das categorias

- **cart.html:** Mostra os produtos adicionados ao carrinho de compras (Figura 4.18).

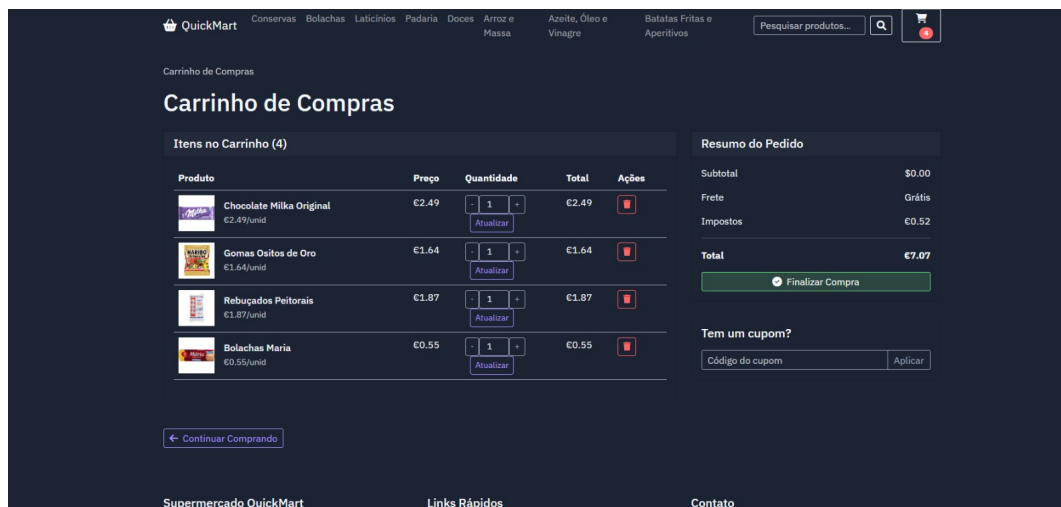


Figura 4.18: Página do carrinho

- **checkout.html**: Página para finalização da compra (Figura 4.19).

QuickMart

Massa Vinagre Aperitivos

Pesquisar produtos...

Carrinho de Compras / Finalizar Compra

Finalizar Compra

Informações do Cliente

Nome Sobrenome

Email Telefone

Loja de Levantamento

Loja

Informações de Pagamento

Nome no Cartão Número do Cartão

Mês de Validade Ano de Validade CVV

Resumo do Pedido

Produto	Qtd	Preço
Chocolate Milka Original	1	€2.49
Gomas Ositos de Oro	1	€1.64
Rebuçados Peitorais	1	€1.87
Bolachas Maria	1	€0.55
Subtotal		€6.55
Frete		Grátis
Impostos		€0.52
Total		€7.07

Figura 4.19: Página de *checkout* do website

- **search_results.html**: Mostra os resultados da pesquisa (Figura 4.20).

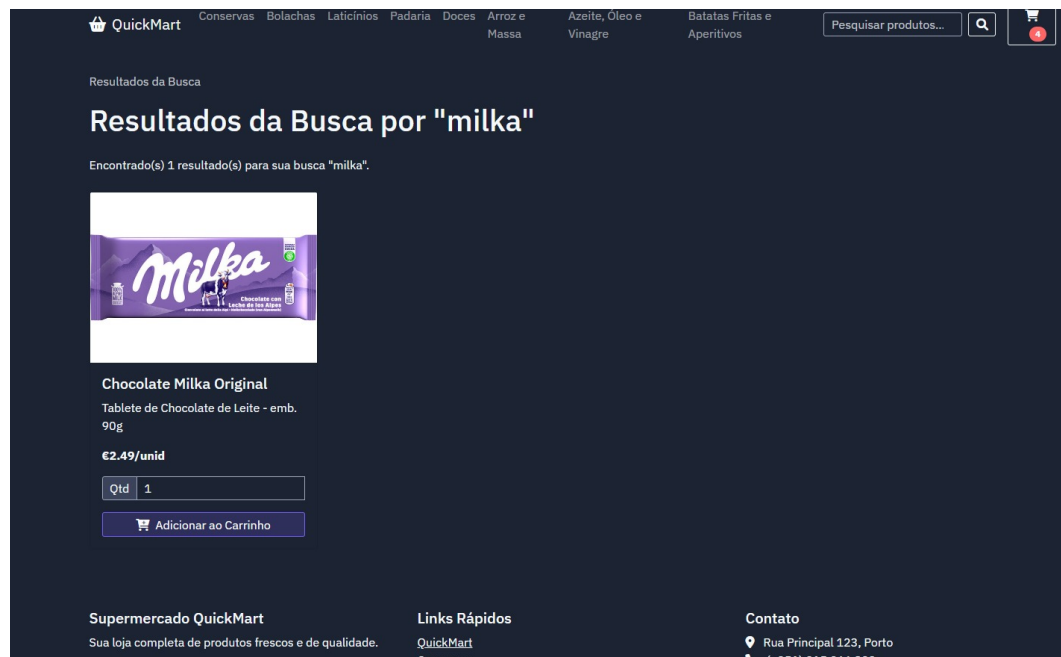


Figura 4.20: Página de Resultados de Pesquisa

- **CSS:** Definido em `custom.css`, responsável pela aparência e responsividade da interface.
- **JavaScript:** Definido em `main.js`, para manipulação do *Document Object Model* (DOM) e integração com o *backend*.

Comunicação entre o *Backend* e o *Frontend*

A interface comunica com o servidor através de requisições HTTP, utilizando o protocolo *Representational State Transfer* (REST). As principais rotas implementadas são:

- **GET /products:** Retorna a lista de produtos disponíveis para visualização. A Listagem 4.1 apresenta o código HTML responsável por permitir ao utilizador realizar pesquisas por produtos. Este formulário encontra-se na barra de navegação e, ao ser submetido, envia uma requisição do tipo GET para a rota `/search`, definida no *backend*. O parâmetro enviado é o texto inserido no campo de pesquisa, identificado pelo atributo `name="query"` (Figura 4.21).

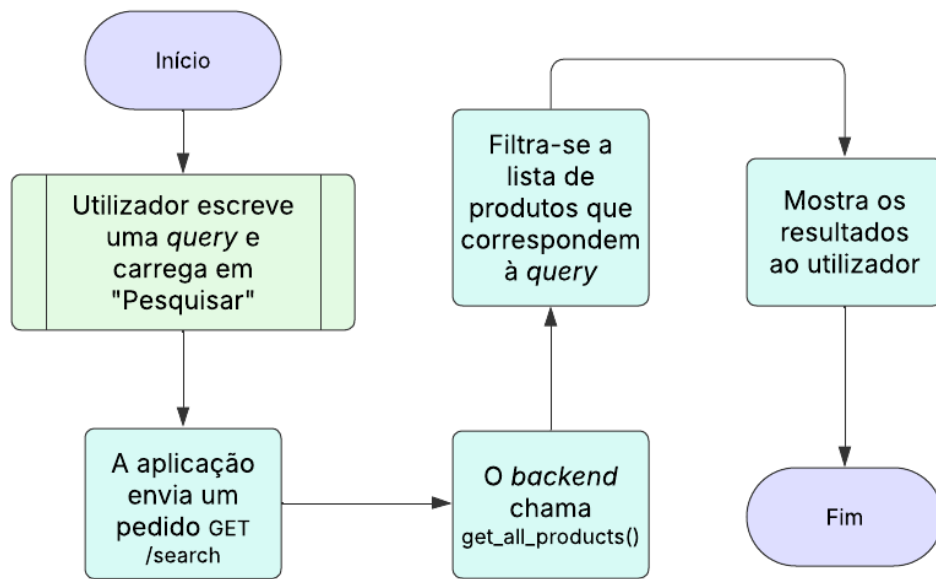


Figura 4.21: Fluxograma do método de pesquisa de produtos

```

1 <form class="d-flex me-3" id="search-form" action="{% url_for('
  search') }}" method="get">
2   <input class="form-control me-2" id="search-input" type="
  search" name="query" placeholder="Pesquisar produtos..." aria
  -label="Pesquisar">
3   <button class="btn btn-outline-light" type="submit">
4     <i class="fas fa-search"></i>
5   </button>
6 </form>

```

Listagem 4.1: Formulário de pesquisa de produtos na interface

O dicionário PRODUCTS está estruturado por categorias, cada uma contém uma lista de produtos, onde cada produto é representado por um dicionário com atributos como id, nome e preco. Um exemplo simplificado é apresentado na Listagem 4.2.

```

1 PRODUCTS = {
2   "Frutas": [
3     {"id": 1, "nome": "Banana", "preco": 0.29},
4     {"id": 2, "nome": "Maca", "preco": 0.35}
5   ],
6   "Bebidas": [
7     {"id": 3, "nome": "Agua 1.5L", "preco": 0.49},
8     {"id": 4, "nome": "Sumo de Laranja", "preco": 1.20}
9   ],

```

```

10     "Mercearia": [
11         {"id": 5, "nome": "Arroz 1kg", "preco": 1.10},
12         {"id": 6, "nome": "Massa Fusilli", "preco": 0.95}
13     ]
14 }

```

Listagem 4.2: Exemplo simplificado do dicionário PRODUCTS

Do lado do *backend*, existe uma função auxiliar que permite obter uma lista linear de todos os produtos, independentemente da sua categoria. Esta função percorre o dicionário PRODUCTS, que agrupa os produtos por categorias, e concatena todas as listas individuais numa única. O código Python correspondente é apresentado na Listagem 4.3 e descrito no fluxograma da Figura 4.22.

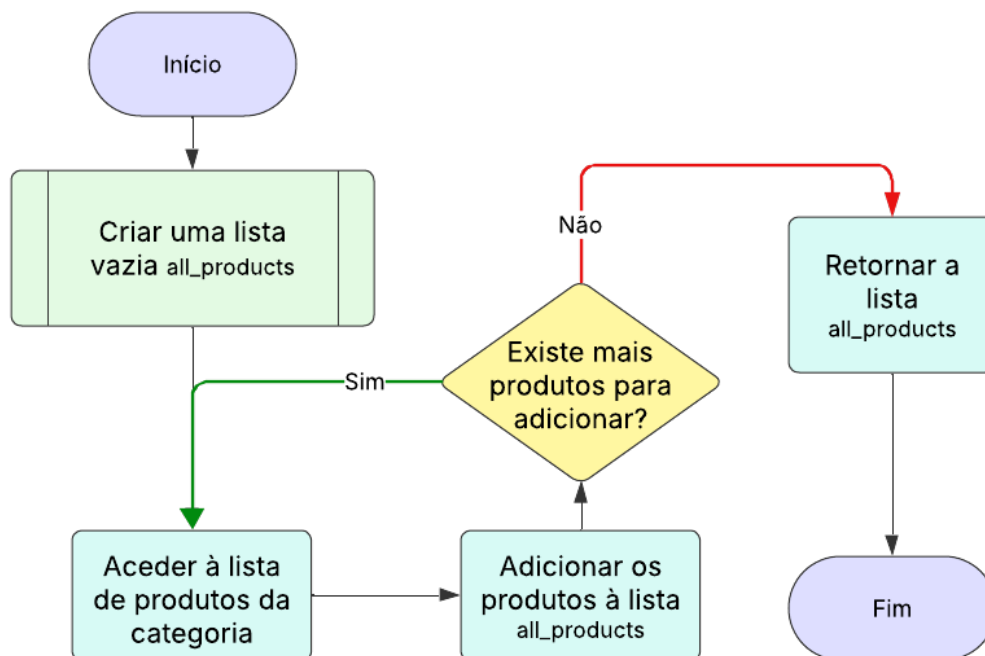


Figura 4.22: Fluxograma do método para obter os produtos

```

1  # Helper function to get a flat list of all products
2  def get_all_products():
3      all_products = []
4      for category_products in PRODUCTS.values():
5          all_products.extend(category_products)
6      return all_products

```

Listagem 4.3: Função auxiliar para obter todos os produtos

Desta forma, sempre que é realizada uma pesquisa, o servidor pode utilizar esta função para aceder rapidamente a todos os produtos disponíveis e devolver apenas aqueles que correspondem à *query* submetida. Assim, a interface permanece dinâmica e responsiva às interações do utilizador.

- **POST /cart:** Adiciona produtos ao carrinho de compras.

A funcionalidade de gestão do carrinho de compras é assegurada por várias rotas do tipo POST, que permitem adicionar, atualizar ou remover produtos. A interface comunica com o *backend* através de formulários HTML que enviam os dados dos produtos seleccionados.

A Listagem 4.4 e a Figura 4.23 apresentam um excerto da interface onde se encontram três formulários distintos:

- Um formulário para **atualizar a quantidade** de um produto no carrinho.
- Um formulário para **remover** um produto do carrinho.
- Um campo oculto (*hidden*) para garantir o envio do `product_id` associado.

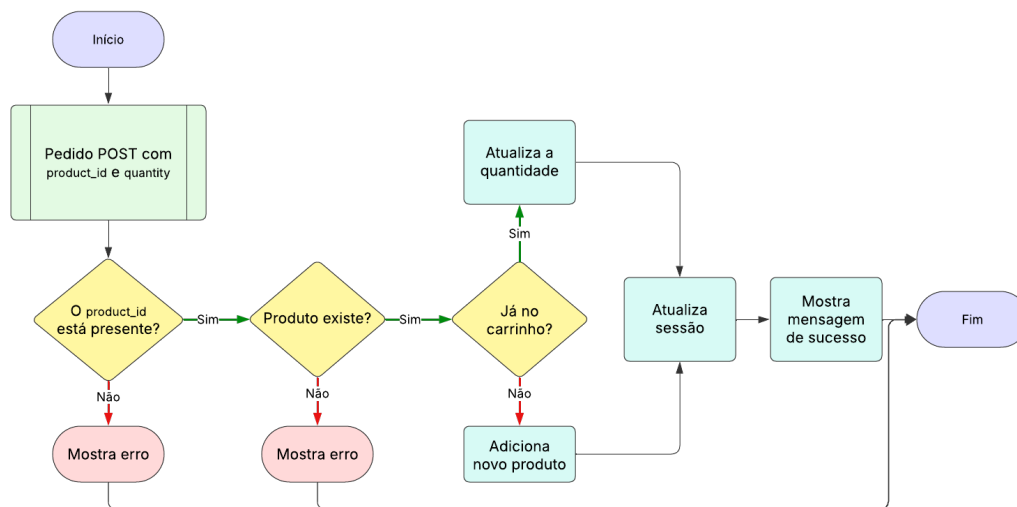


Figura 4.23: Fluxograma do método atualização de produtos do carrinho

```

1 <td>
2   <form action="{{ url_for('update_cart') }}" method="post"
3     class="d-inline">
4     <div class="quantity-control">
5       <button type="button" class="btn btn-sm btn-outline-
6         secondary btn-decrement"></button>

```

```

5         <input type="number" name="quantity" value="{{ item.
        quantity }}" min="1" max="99" class="form-control form-
        control-sm quantity-input" data-price="{{ item.product.price
        }}">
6         <button type="button" class="btn btn-sm btn-outline-
        secondary btn-increment">+</button>
7         <input type="hidden" name="product_id" value="{{ item
        .product.id }}">
8     </div>
9     <button type="submit" class="btn btn-sm btn-outline-
        primary ms-2">Atualizar</button>
10    </form>
11 </td>
12 <td class="item-total">{{ "%.2f"|format(item.total) }}</td>
13 <td>
14     <form action="{{ url_for('remove_from_cart') }}" method="post
        ">
15         <input type="hidden" name="product_id" value="{{ item.
        product.id }}">
16         <button type="submit" class="btn btn-sm btn-outline-
        danger">
17             <i class="fas fa-trash"></i>
18         </button>
19     </form>
20 </td>

```

Listagem 4.4: Formulários de atualização e remoção no carrinho

A rota que permite adicionar produtos ao carrinho está implementada no *backend* com o nome `/add_to_cart`, e está descrita na Listagem 4.5 e na Figura 4.24.

```

1  # Route for adding an item to the cart
2  @app.route('/add_to_cart', methods=['POST'])
3  def add_to_cart():
4      product_id = request.form.get('product_id')
5      quantity = int(request.form.get('quantity', 1))
6
7      if not product_id:
8          flash("Produto nao especificado", "danger")
9          return redirect(request.referrer or url_for('index'))
10
11     product = find_product_by_id(product_id)
12     if not product:
13         flash("Produto nao encontrado", "danger")
14         return redirect(request.referrer or url_for('index'))
15
16     cart = session.get('cart', {})

```

```

17
18     # Update quantity if product already in cart
19     if product_id in cart:
20         cart[product_id] += quantity
21     else:
22         cart[product_id] = quantity
23
24     session['cart'] = cart
25     session.modified = True
26
27     flash(f"Adicionado {quantity} {product['name']} ao seu
28     carrinho", "success")
29     return redirect(request.referrer or url_for('index'))

```

Listagem 4.5: Função para adicionar produtos ao carrinho

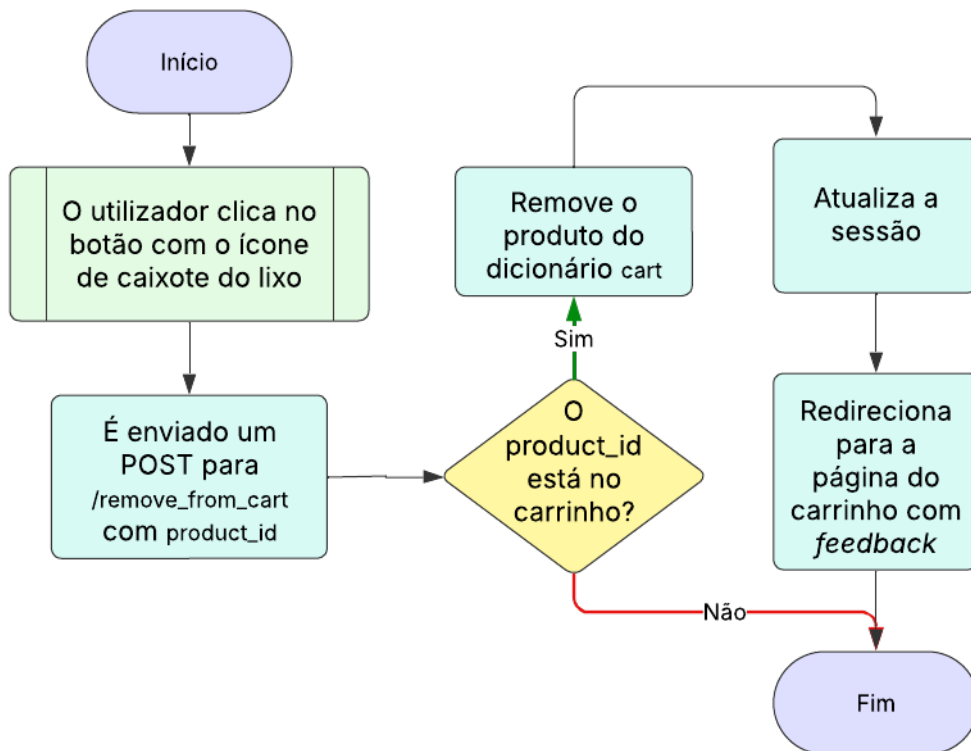


Figura 4.24: Fluxograma do método remoção de produtos do carrinho

As funcionalidades de atualização (`update_cart`) e remoção (`remove_from_cart`) seguem uma lógica semelhante, manipulando o conteúdo da variável de sessão `cart` consoante os dados recebidos. Por motivos de brevidade, essas implementações encontram-se detalhadas no Anexo A.5.

- **POST /checkout:** Finaliza a compra e gera um registo da encomenda.

Após o utilizador concluir a seleção dos produtos pretendidos, é possível finalizar a compra através da submissão de um formulário que desencadeia uma requisição POST para a rota `/place_order`. Esta rota, apresentada na Listagem 4.6, representa o processo de *checkout*, durante o qual os dados do carrinho são processados e enviados ao sistema físico (Figura 4.25).

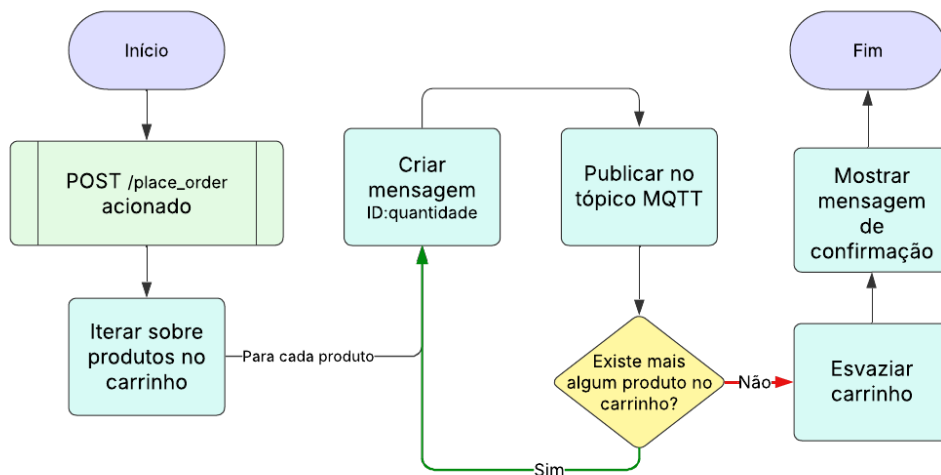


Figura 4.25: Fluxograma do método de finalização de compra

```

1  # Route for completing the checkout process
2  @app.route('/place_order', methods=['POST'])
3  def place_order():
4      cart = session.get('cart', {})
5      for product_id, quantidade in cart.items():
6          mensagem = f"{product_id}:{quantidade}"
7          client.publish(TOPIC_COMMANDS, mensagem)
8          print(f"[MQTT] Publicado: {mensagem}")
9
10     session['cart'] = {}
11     session.modified = True
12     flash("Seu pedido foi realizado! Obrigado por comprar conosco.", "success")
13     return redirect(url_for('index'))
  
```

Listagem 4.6: Função de finalização da compra (*checkout*)

Nesta função, todos os produtos presentes no carrinho são iterados e, para cada um, é construída uma mensagem no formato `<ID_produto>:<quantidade>`. Esta mensagem é então publicada no tópico MQTT definido como `TOPIC_COMMANDS`, sendo enviada para o microcontrolador responsável por processar

fisicamente a encomenda — por exemplo, ativa o dispensador do produto A, B ou C.

Após o envio dos dados, o carrinho é limpo e é apresentada ao utilizador uma mensagem de confirmação, indicando que a encomenda foi registada com sucesso. Esta abordagem garante uma comunicação eficaz entre a interface *web* e os dispositivos embarcados do sistema automatizado.

Esta abordagem assegura uma sincronização rápida e eficiente dos dados entre a interface gráfica e o servidor, proporcionando uma experiência de compra fluida e intuitiva, em total alinhamento com os objetivos do projeto.

4.4.4 Integração do ESP32 no Sistema

O microcontrolador ESP32 desempenha um papel essencial no sistema, sendo o responsável pela execução física dos comandos de dispensa de produtos através dos servomotores, bem como pela ativação do tapete rolante. A comunicação entre o Raspberry Pi 5 e o ESP32 é feita via protocolo MQTT, que permite o envio de comandos de forma leve e eficiente.

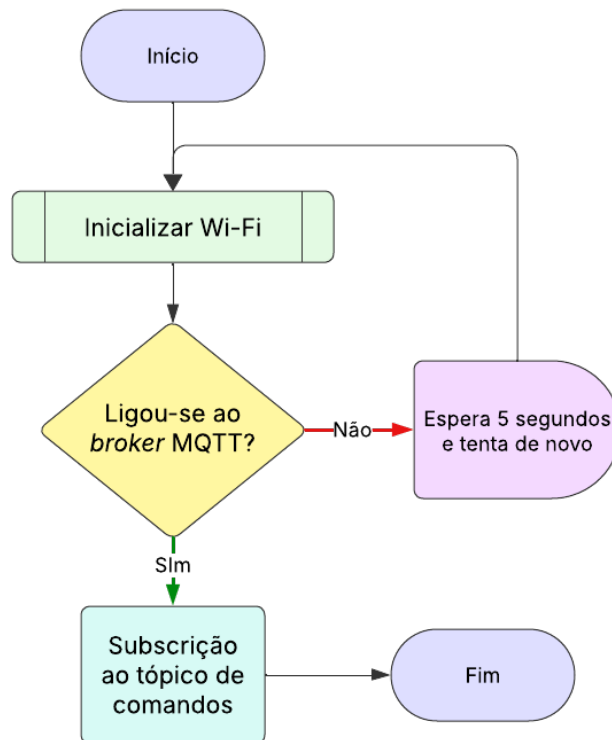
Conexão *Wi-Fi* e Inicialização do Cliente MQTT

O ESP32 conecta-se a uma rede *Wi-Fi* e estabelece ligação com o *broker* MQTT local. A configuração do *Wi-Fi* feita na função `setup_wifi()`, enquanto o cliente MQTT `PubSubClient` é inicializado com os parâmetros adequados (Listagem 4.7 e Figura 4.26).

```
1 const char* ssid = "DIGIFIBRA-PLUS-0719";
2 const char* password = "*****";
3 const char* mqttServer = "192.168.1.164";
4 const int mqttPort = 1883;
5 const char* mqttUser = "usuario";
6 const char* mqttPassword = "senha";
7
8 WiFiClient espClient.
9 PubSubClient client(espClient).
```

Listagem 4.7: Conexão *Wi-Fi* e configuração MQTT

Para monitorização do sistema, o ESP32 publica uma mensagem de estado a cada 10 segundos no tópico `supermercado/estado`, indicando que está *online* e a funcionar corretamente (Listagem 4.8).

Figura 4.26: Fluxograma do método de conexão ao *Wi-Fi*

```
1 static unsigned long lastPublish = 0;
2 if (millis() - lastPublish > 10000) {
3     client.publish("supermercado/estado", "ESP32 Online");
4     lastPublish = millis();
5 }
```

Listagem 4.8: Publicação de estado

A função `setup_wifi()` gere a conexão à rede, enquanto a função `reconnect()` trata da reconexão automática ao *broker* caso a ligação caia (Listagem 4.9).

```
1 void reconnect() {
2     while (!client.connected()) {
3         if (client.connect("ESP32Client", mqttUser, mqttPassword)) {
4             client.subscribe("supermercado/comandos");
5         } else {
6             delay(5000);
7         }
8     }
9 }
```

Listagem 4.9: Reconexão ao MQTT *broker*

4.4.5 Cliente ESP32 no Raspberry Pi

O Raspberry Pi, além de albergar o *broker* local, também executa um cliente responsável por comunicar com o ESP32. Este cliente é implementado em Python utilizando a biblioteca `paho-mqtt` e é utilizado pela aplicação *web* para enviar comandos de compra e monitorizar o estado do sistema.

Ficheiro `mqtt_client.py`

Este módulo é carregado juntamente com a aplicação Flask e é o responsável pela comunicação assíncrona com o ESP32. Abaixo apresenta-se parte do ficheiro (Listagem 4.10).

```
1 import paho.mqtt.client as mqtt
2
3 # Configuracoes do Broker
4 BROKER_IP = "192.168.1.164" # IP do Raspberry Pi (ou broker local)
5 BROKER_PORT = 1883
6 TOPIC_COMMANDS = "supermercado/comandos"
7 TOPIC_STATE = "supermercado/estado"
8
9 # Funcao chamada quando conectar
10 def on_connect(client, userdata, flags, rc):
11     print(f"Conectado ao broker MQTT com codigo: {rc}")
12     client.subscribe(TOPIC_STATE)
```

Listagem 4.10: Configuração do cliente MQTT no Raspberry Pi

O cliente subscreve-se ao tópico `supermercado/estado` para receber atualizações do ESP32, como indicações de disponibilidade ou funcionamento. A função `on_message()` é chamada sempre que uma nova mensagem é recebida (Listagem 4.11 e Figura 4.27).

```
1 def on_message(client, userdata, msg):
2     print(f"Mensagem recebida no topico {msg.topic}: {msg.payload.decode()}")
```

Listagem 4.11: Função de receção de mensagens do ESP32

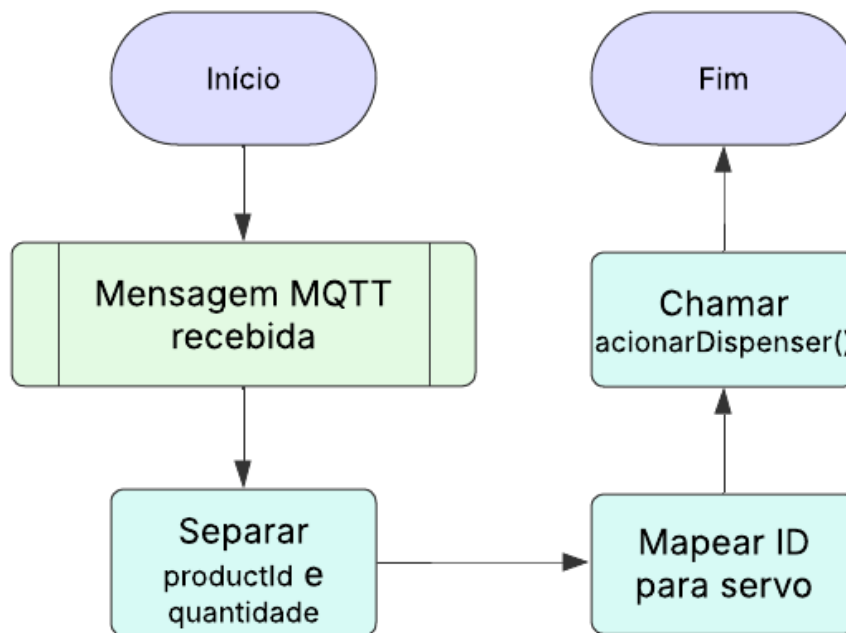


Figura 4.27: Fluxograma do método de recepção de mensagens e tratamento de dados

Por fim, o cliente é instanciado, configurado e ligado ao *broker* local, iniciando o *loop* de recepção de mensagens em segundo plano (Listagem 4.12).

```
1 client = mqtt.Client("FlaskClient")
2 client.on_connect = on_connect
3 client.on_message = on_message
4
5 client.connect(BROKER_IP, BROKER_PORT, 60)
6 client.loop_start()
```

Listagem 4.12: Inicialização do cliente MQTT

Envio de Comandos pela Aplicação Web

Com este cliente ESP32 em execução no Raspberry Pi, a aplicação *web* desenvolvida em Flask pode invocar métodos como (Listagem do cliente 4.13).

```
1 client.publish("supermercado/comandos", "102:2")
```

Listagem 4.13: Exemplo do envio de mensagem do cliente MQTT

Este comando indica ao ESP32 que deve dispensar duas unidades do produto com *ID* (Identificador) 102, desencadeando assim o processo físico de entrega dos produtos no sistema automático.

O ficheiro `mqtt_client.py` permite integrar o servidor da aplicação *web* com o sistema físico de entrega automatizada, garantindo uma comunicação eficaz, em tempo real e fiável com o ESP32. Através da separação de tópicos comandos e estado, obtém-se um canal de controlo e outro de monitorização, contribuindo para uma arquitetura robusta e escalável.

4.4.6 Programa do ESP32

O microcontrolador ESP32 é responsável pela execução das ações físicas do sistema de automação, nomeadamente a ativação dos dispensadores e o controlo do tapete rolante. Posto isto, já se sabe como a mensagem chega ao ESP32 do Raspberry Pi, portanto, passa-se à parte de automatização dos motores e sensores. O programa foi desenvolvido na *framework* do Arduino e carrega automaticamente no arranque do dispositivo.

Mapeamento de Produtos e Controlo de Servos

Cada produto disponível no sistema está associado a um dispensador físico, controlado por um servomotor. O programa define um *array* com os pinos utilizados e inicializa todos os motores (Listagem 4.14).

```
1 const int NUM_PRODUTOS = 4;
2 Servo servos[NUM_PRODUTOS];
3 int servo_pins[NUM_PRODUTOS] = {18, 19, 21, 22};
```

Listagem 4.14: Associação dos produtos aos servos

O ESP32 possui vários servomotores ligados a diferentes GPIO, representando diferentes produtos. Cada produto identificado por um *ID* (Identificador), que é mapeado para o índice correspondente do servomotor (Listagem 4.15).

```
1 int mapearProdutoParaServo(String productId) {
2     if (productId == "101") return 0;
3     else if (productId == "102") return 1;
4     else if (productId == "103") return 2; // produto mais leve
5     else if (productId == "104") return 3;
6     else return -1; // ID invalido
7 }
```

Listagem 4.15: Mapeamento dos *ID* (Identificadores) de produtos para servos

Posto isto, quando um comando é recebido pelo ESP32 contendo o *ID* (Identificador) do produto e a quantidade, o mesmo executa o movimento do servo correspondente seguido da ativação do tapete rolante (Listagem 4.16).

```
1 void callback(char* topic, byte* message, unsigned int length) {
2     String msg;
3     for (int i = 0; i < length; i++) {
4         msg += (char)message[i];
5     }
6     int separatorIndex = msg.indexOf(':');
7     if (separatorIndex != -1) {
8         String productId = msg.substring(0, separatorIndex);
9         String quantidadeStr = msg.substring(separatorIndex + 1);
10        int quantidade = quantidadeStr.toInt();
11        int servoIndex = mapearProdutoParaServo(productId);
12        acionarDispenser(servoIndex, quantidade);
13    }
14 }
```

Listagem 4.16: *Callback* e execução do comando

Função de Dispensa e Tapete Rolante

A função `acionarDispenser()` move o servomotor até uma determinada posição, para dispensar a quantidade indicada de produtos, seguida da ativação do tapete (Listagem 4.17 e Figura 4.28).

```
1 void acionarDispenser(int servoIndex, int quantidade) {
2     if (servoIndex < 0 || servoIndex >= NUM_PRODUTOS) {
3         Serial.println("Servo invalido para este ID de produto.");
4         return;
5     }
6     for (int i = 0; i < quantidade; i++) {
7         servos[servoIndex].write(0);
8         delay(1000);
9         servos[servoIndex].write(90);
10        delay(1000);
11        acionarTapete();
12    }
13 }
```

Listagem 4.17: Dispensador de produtos

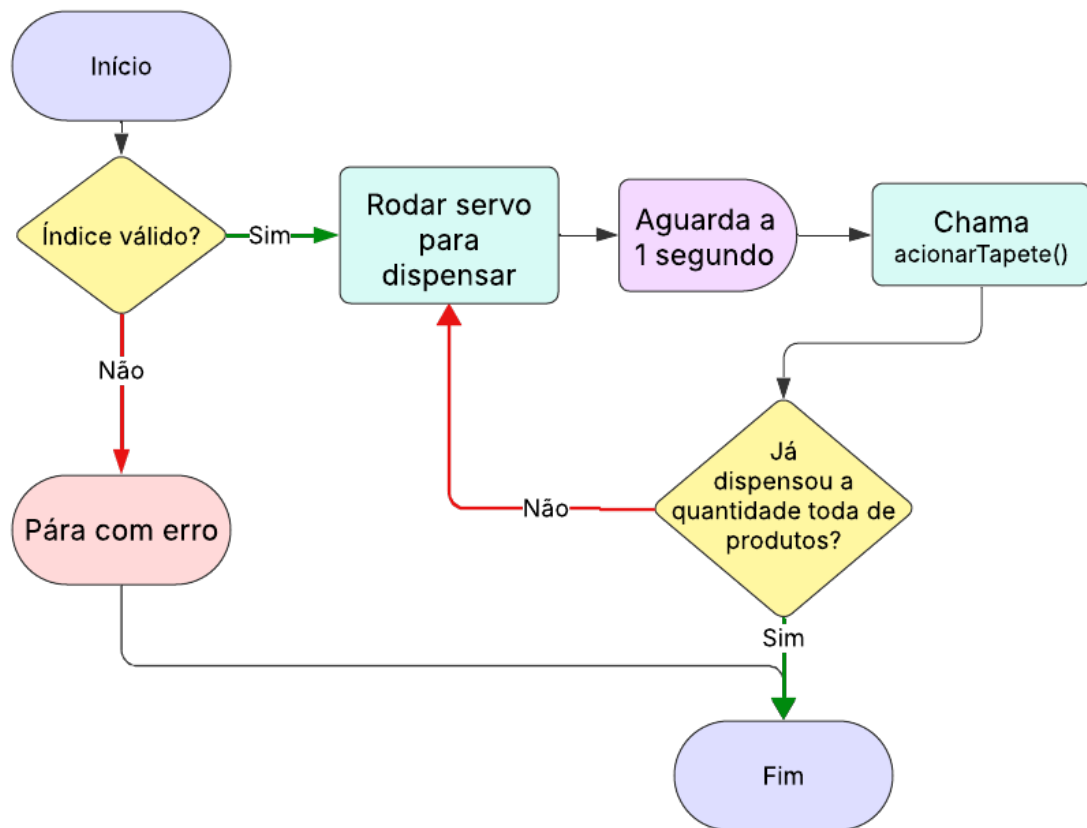


Figura 4.28: Fluxograma do método de dispensador de produtos

Controlo do Tapete Rolante com Sensor no Fim de Curso

O tapete rolante é ativado após cada dispensador de produto. A sua paragem é feita automaticamente quando o sensor de fim de tapete deteta a presença do produto ou após um tempo limite de segurança (Listagem 4.18 e Figura 4.29).

```

1 void acionarTapete() {
2     digitalWrite(motorIN1, HIGH);
3     digitalWrite(motorIN2, LOW);
4     Serial.println("Tapete rolante ativado");
5     unsigned long tempoInicio = millis();
6     const unsigned long TIMEOUT = 40000;
7
8     while (!detetarProdutoComUltrassonico() && millis() - tempoInicio
9           < TIMEOUT) {
10         delay(50);
11     }
12     if (detetarProdutoComUltrassonico()) {
13         Serial.println("Produto detetado com sensor ultrassonico.");
14     } else {

```

```
14     Serial.println("Timeout: produto nao detetado.");
15 }
16 digitalWrite(motorIN1, LOW);
17 digitalWrite(motorIN2, LOW);
18 Serial.println("Tapete rolante parado");
19 }
```

Listagem 4.18: Controlo do tapete rolante

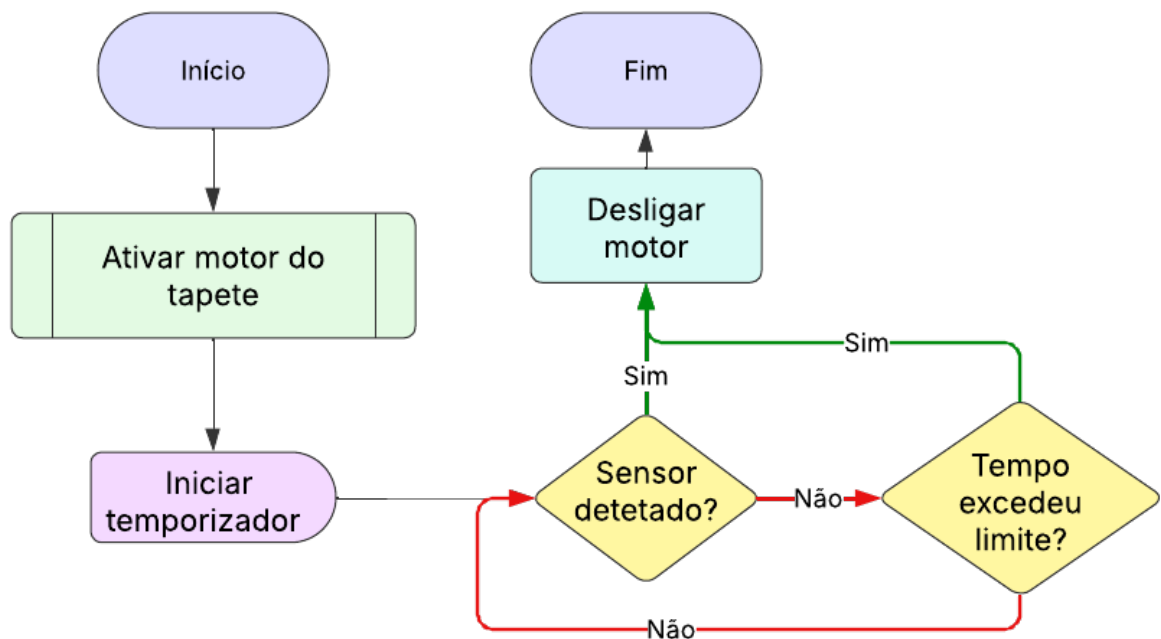


Figura 4.29: Fluxograma do método de controlo do tapete rolante

Para detetar a presença de um produto no final do tapete rolante, é utilizado um sensor ultrassónico. Este sensor emite um sinal de som e mede o tempo que este demora a refletir-se num obstáculo, calculando assim a distância ao objeto mais próximo. O método `detetarProdutoComUltrassonico()`, apresentado na Listagem 4.19 e na Figura 4.30, devolve `true` caso o produto seja detetado a uma distância inferior ou igual a 15 cm, permitindo assim parar o motor do tapete de forma automática quando este deteta o produto a entrar no recipiente.

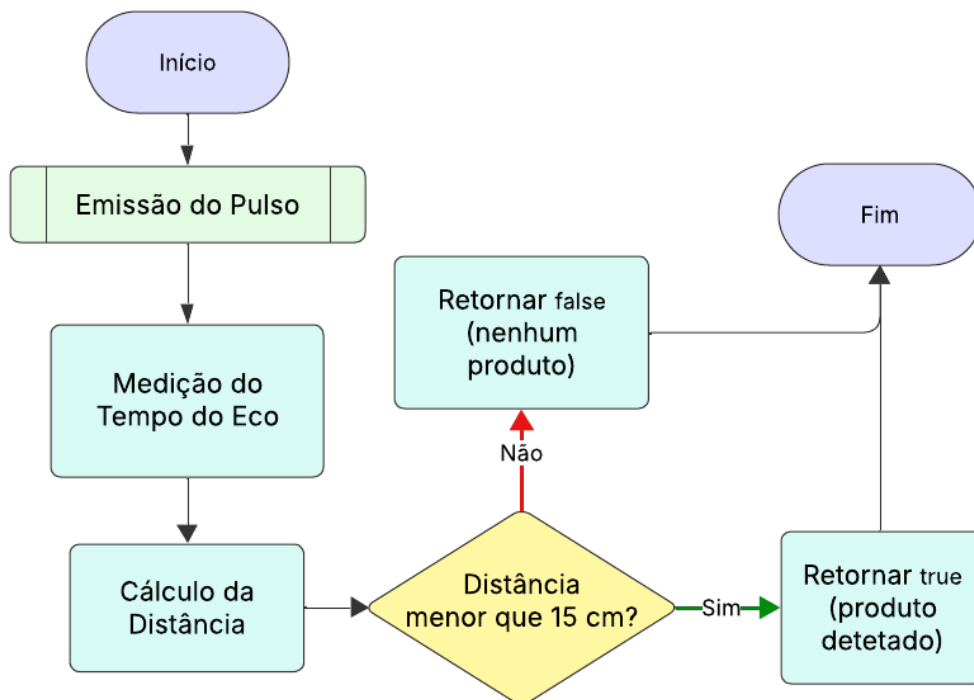


Figura 4.30: Fluxograma do método de controlo do sensor ultrassónico

```

1 bool detetarProdutoComUltrassonico() {
2     digitalWrite(TRIG_PIN, LOW);
3     delayMicroseconds(2);
4     digitalWrite(TRIG_PIN, HIGH);
5     delayMicroseconds(10);
6     digitalWrite(TRIG_PIN, LOW);
7
8     long duracao = pulseIn(ECHO_PIN, HIGH);
9     float distancia = duracao * 0.034 / 2;
10    Serial.print("Dist ncia medida: ");
11    Serial.print(distancia);
12    Serial.println(" cm");
13    return (distancia > 0 && distancia <= 15); // Produto detetado a
    menos de 15cm
14 }
  
```

Listagem 4.19: Leitura do sensor ultrassónico no fim do tapete

A integração do ESP32 no sistema permite executar fisicamente os comandos provenientes da aplicação *web*, garantindo uma resposta automatizada e sincronizada entre a seleção do utilizador e a entrega do produto. O uso do ESP32 como protocolo

de comunicação assegura leveza, rapidez e escalabilidade para a extensão do sistema com novos produtos e funcionalidades.

Este programa implementa uma lógica reativa simples, baseada em eventos recebidos por MQTT. Garante-se a execução sequencial do processo de dispensa, movimentação e detecção, permitindo uma integração eficaz entre a camada física e o sistema de gestão do supermercado.

4.5 Resultado Final do Protótipo

Após a implementação e integração de todos os componentes descritos nas secções anteriores, apresenta-se de seguida o sistema finalizado do protótipo do projeto.

Como se pode observar na Figura 4.31, o protótipo integra os dispensadores automáticos, o tapete rolante para transporte dos produtos e a interface gráfica que o cliente tem acesso, formando um sistema funcional terminado.

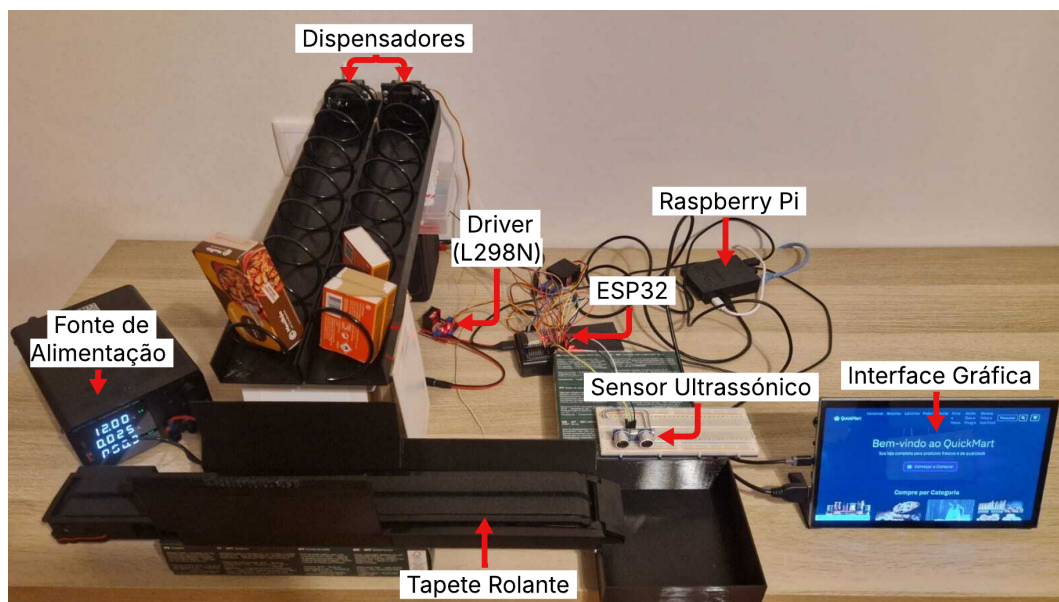


Figura 4.31: Fotografia do protótipo final do sistema

Este resultado demonstra a viabilidade da solução proposta, bem como a concretização dos objetivos definidos para o desenvolvimento do protótipo.

Capítulo 5

Validação do Sistema e Resolução de Problemas

Este capítulo apresenta a validação do sistemas de entrega automática e transporte de produtos, incluindo os testes realizados, os resultados obtidos e os principais desafios enfrentados durante o desenvolvimento. Além de confirmar a funcionalidade e fiabilidade do protótipo, foram identificados e resolvidos vários problemas técnicos que contribuíram para a evolução e estabilidade do sistema.

5.1 Testes e Validação

Com o protótipo funcional, foi fundamental realizar um conjunto de testes que permitissem validar a comunicação entre os diferentes módulos, o desempenho dos componentes físicos e a resposta do sistema em condições reais de utilização. Os testes foram organizados em categorias distintas, focando-se na integração do sistema, na resposta dos atuadores e na robustez do processo de dispensa e transporte dos produtos.

5.1.1 Testes de Integração

Os testes de integração permitiram validar o funcionamento coordenado entre a interface gráfica no Raspberry Pi 5, a comunicação via protocolo MQTT e o controlo de atuadores no ESP32. Estes testes, representados na Tabela 5.1, foram essenciais

para verificar a comunicação entre os módulos e a execução completa do ciclo de operação.

Tabela 5.1: Resultados dos Testes de Integração

Cenário Testado	Descrição do Teste	Resultado
Envio de pedido via UI	Mensagem MQTT recebida e interpretada pelo ESP32	Sucesso
Ação do servomotor	Produto corretamente dispensado após comando	Sucesso
Tapete rolante ativado	Transporte correto até ao sensor de fim	Sucesso
Sensor de deteção final	Produto detetado com sucesso a entrar na caixa de recolha	Sucesso

5.1.2 Ajustes de Integração

Durante a fase de testes, foram feitos ajustes finos no *hardware* e no *software* para melhorar o desempenho global do sistema (Tabela 5.2).

Tabela 5.2: Ajustes Durante a Integração

Elemento Ajustado	Motivo / Efeito do Ajuste
Ângulo do servomotor	Corrigido para garantir a saída controlada de uma unidade de produto
Tempo do tapete rolante	Ajustado para que o produto alcance o sensor sem cair antes
<i>Timeout</i> da mensagem MQTT	Reduzido para garantir resposta em tempo útil após pagamento

5.1.3 Testes de Precisão e Quantidade

Estes testes avaliaram a capacidade do sistema em dispensar corretamente o número e tipo de produtos solicitados, independentemente do seu peso e tamanho (Tabela 5.3).

Tabela 5.3: Resultados dos Testes de Precisão

Cenário	Esperado	Obtido
Produto leve (30 g)	1 unidade libertada	1 unidade
Produto pesado (2x250 g)	Transporte sem bloqueios	Sucesso
Transporte médio (250 g)	Tempo < 4 s	3,6 s
Formato largo	Transporte estável	Sucesso
10 dispensações seguidas	10/10 corretas	9/10

5.1.4 Testes de Robustez e Fiabilidade

Foram simuladas condições de falha e uso contínuo para garantir que o sistema se mantém estável e recupera de falhas pontuais (Tabela 5.4).

Tabela 5.4: Testes de Robustez e Recuperação

Falha Simulada	Comportamento Esperado	Resultado
Perda de energia	Retoma segura e estável	Sucesso
Falha no sensor	Alerta ao utilizador	Sucesso
Desligar ligação MQTT	Reconexão automática	Sucesso
Operação contínua (20 minutos)	Sem falhas ou perdas	Sucesso

5.1.5 Análise Final dos Testes

Os resultados obtidos nos diferentes testes comprovam que o sistema desenvolvido cumpre os requisitos definidos. A integração entre os componentes físicos e digitais revelou-se fiável, com tempos de resposta adequados e operação estável mesmo após longos períodos de utilização ou em situações de falha.

A precisão nas dispensações e o funcionamento do sistema de transporte confirmam a viabilidade técnica da solução proposta. Todos os testes planeados foram executados com sucesso, demonstrando que o protótipo está preparado para evoluir para uma versão funcional mais avançada.

5.2 Desafios e Soluções

Ao longo do desenvolvimento do projeto de automação do supermercado, surgiram vários desafios técnicos que exigiram abordagens específicas para serem ultrapassados. Um dos primeiros obstáculos foi a configuração inicial do Raspberry Pi 5, que apresentou dificuldades na instalação do sistema operativo e na deteção correta do monitor *touchscreen*. Para resolver estas questões, foi necessário recorrer diversos fóruns para perceber que por vezes o Raspberry Pi bloqueia e é preciso dar *setup* de um *software* de recuperação para depois garantir uma instalação limpa e otimizada do Raspberry Pi OS. Além disso, a configuração do monitor *touchscreen* exigiu ajustes no ficheiro `config.txt` para parametrização correta e a instalação de *drivers* específicos.

Outro desafio significativo surgiu durante a integração entre Flask e os *templates* HTML, onde ocorreram conflitos na passagem de dados, dificultando a renderização dinâmica e a gestão do estado do carrinho de compras. Para mitigar este problema, foi adotada uma abordagem baseada em API REST, que permitiu isolar a lógica de *backend* da apresentação no *frontend*. Com esta estratégia, a atualização dos dados

em tempo real foi otimizada através de *fetch requests* em JavaScript, garantindo uma experiência mais fluida para o utilizador.

À medida que o projeto evoluiu, a organização do código tornou-se um ponto crítico, com múltiplos ficheiros JavaScript, HTML e Python distribuídos de forma dispersa. Para solucionar esta questão, foi feita uma reestruturação do projeto, organizando-o em diretórios específicos: o diretório *backend* para o Flask e os serviços de API e o *frontend* para o código HTML e ficheiros estáticos. Esta organização proporcionou um desenvolvimento mais ágil e uma manutenção mais simples e estruturada do código (Listagem 5.1).

```
1 supermercado/  
2     static/  
3         css/  
4             custom.css  
5         images/  
6         js/  
7             main.js  
8     templates/  
9         base.html  
10        cart.html  
11        category.html  
12        checkout.html  
13        index.html  
14        search_results.html  
15    venv/  
16    __pycache__/  
17    app.py  
18    main.py  
19    mqtt_client.py  
20    pyproject.toml  
21    supermercado.service
```

Listagem 5.1: Estrutura de diretórios do projeto

Para além dos desafios ao nível do *software* e da integração de sistemas, também surgiram dificuldades no desenvolvimento físico do protótipo. Uma das peças impressas em 3D que serve de suporte ao motor do servomotor revelou-se estruturalmente frágil na sua primeira versão, apresentando flexões indesejadas durante o funcionamento chegando mesmo a partir. Esta fragilidade comprometeu o alinhamento do motor com a mola e, consequentemente, o desempenho do sistemas de entrega automatizada dos produtos. Para resolver este problema, foi necessário reforçar a zona de encaixe do motor e dos parafusos no modelo CAD e proceder a uma nova impressão, garantindo assim maior robustez e estabilidade mecânica.

Durante a fase de testes do sistema embebido, verificou-se um problema crítico com o microcontrolador ESP32, que deixou de responder e não era reconhecido por nenhuma porta USB. Após várias tentativas de reprogramação e diagnóstico, concluiu-se que o dispositivo estava danificado de forma irreversível, possivelmente devido a períodos longos de testes e com temperaturas de ambiente elevadas ou apenas uma descarga eletrostática. Este incidente obrigou à substituição do ESP32 por uma nova unidade, o que implicou um ligeiro atraso no cronograma, mas permitiu retomar os testes com o sistema funcional e com maior precaução na manipulação dos componentes eletrônicos.

Do mesmo modo, foram identificadas várias dificuldades no processo de comunicação entre o ESP32 e o *backend*, nomeadamente na ligação à rede *Wi-Fi* e na configuração do protocolo MQTT. Inicialmente, o ESP32 ficava indefinidamente no estado de tentativa de ligação, imprimindo apenas pontos no terminal, o que se revelou ser causado por instabilidade na rede local. Após a correção desses parâmetros, surgiram novos entraves na troca de mensagens via MQTT, sendo necessário garantir que tanto o *broker* como o cliente estivessem corretamente sincronizados e a operar na mesma rede. Para além disso, o sensor ultrassónico responsável por detetar a presença de produtos devolvia sistematicamente uma distância nula, o que indicava problemas nas ligações físicas ou na lógica de medição — situação que foi resolvida após validação dos pinos e da temporização dos sinais de `trigger` e `echo`. Por fim, ocorreram erros de compilação durante a programação do ESP32, nomeadamente mensagens associadas ao ficheiro `bootloader.bin`, que resultaram da existência de ficheiros temporários corrompidos. Este problema foi resolvido através da limpeza do ambiente de desenvolvimento e da recompilação completa do projeto.

Capítulo 6

Conclusões

O presente trabalho teve como principal objetivo o desenvolvimento de um sistema automatizado para supermercados, com o intuito de melhorar a eficiência operacional e a experiência de compra dos clientes. Para isso, foi concebido e implementado um protótipo funcional que integra dispensadores automáticos, sensores, um tapete rolante e uma interface gráfica acessível através de um ecrã tátil, permitindo a seleção, pagamento e recolha de produtos de forma autónoma.

Durante o desenvolvimento da solução, foi possível integrar com sucesso diversas tecnologias de *hardware* e *software*, como o microcontrolador ESP32 para controlo dos atuadores e sensores, o Raspberry Pi 5 para gestão da interface gráfica e servidor MQTT, bem como sensores ultrassónicos e servomotores. A comunicação entre os diversos componentes foi assegurada através do protocolo MQTT, garantindo um sistema eficiente, modular e escalável.

Os testes realizados demonstraram a viabilidade do sistema, validando a precisão da dispensa dos produtos, a fiabilidade do transporte até à zona de recolha e a robustez da comunicação entre os módulos. Apesar de alguns desafios encontrados, como a calibração dos motores ou a integração física dos componentes, as soluções implementadas permitiram alcançar os objetivos propostos.

Este projeto destaca-se pela sua abordagem inovadora ao combinar tecnologias de automação com uma interface amigável ao utilizador, propondo uma solução prática e acessível, especialmente adequada para supermercados de pequena e média dimensão. A automatização da entrega de produtos não só reduz os erros humanos

como também minimiza os tempos de espera e o esforço operacional, alinhando-se com as tendências atuais do setor do retalho.

6.1 Trabalho Futuro

Como trabalho futuro, propõe-se a expansão do sistema para múltiplos dispensadores em paralelo, a integração com sistemas de gestão de inventário em tempo real, a inclusão de métodos de pagamento *contactless* (como NFC) diretamente no terminal e a aplicação de algoritmos de otimização para o fluxo logístico. A adoção de técnicas de inteligência artificial poderá ainda contribuir para uma análise preditiva do comportamento dos consumidores e uma gestão dinâmica do *stock*.

Adicionalmente, a introdução de um sistema de autenticação por RFID dedicado aos operadores, permitindo o bloqueio temporário da máquina durante operações de *restock*, manutenção ou auditoria, garantindo assim a segurança do sistema e a integridade dos dados. Poderá também ser desenvolvida uma funcionalidade de criação de contas para utilizadores, com *login* protegido por palavra-passe, possibilitando um histórico de compras, preferências personalizadas e promoções direcionadas.

Outras melhorias a considerar incluem:

- A implementação de um modo *offline* com sincronização posterior com o servidor central, assegurando a operação contínua mesmo em caso de falha de rede.
- O desenvolvimento de uma aplicação móvel complementar, permitindo a pré-reserva de produtos, pagamento remoto ou levantamento rápido na loja.
- A utilização de visão por computador para deteção automática de falhas no abastecimento dos dispensadores ou produtos fora do lugar.
- A integração com sistemas de fidelização de clientes e campanhas promocionais inteligentes baseadas no perfil de consumo.
- A instalação de sensores ambientais (temperatura, humidade) para monitorização da qualidade de produtos sensíveis.
- A adaptação do sistema para suportar produtos refrigerados ou frescos, exigindo novos tipos de dispensadores com controlo térmico.

Estas extensões visam tornar o sistema mais robusto, escalável e adaptado às exigências reais do retalho moderno, potenciando uma experiência de compra mais eficiente, segura e personalizada.

Referências

- [1] Huffington Post, “Ni mercadona ni lidl: la cadena de supermercados que declara la guerra a los recibos,” *Huffington Post*, 2025. Accessed: 3 fev. 2025. [Citado na página 2]
- [2] StartSe, “Mercado amazon go une digital e analógico.” Available at <https://www.startse.com/artigos/mercado-amazon-go-une-digital-e-analogico/>, 2024. (Last accessed in 04/01/2025). [Citado nas páginas ix e 11]
- [3] A. Polacco and K. Backes, “The amazon go concept: Implications, applications, and sustainability,” *Journal of Business and Management*, vol. 24, no. 1, pp. 79–92, 2018. [Citado na página 11]
- [4] Ocado, “Ocado - aws.” <https://aws.amazon.com/solutions/case-studies/ocado-aws-is-how/>, n.d. Ocado Group Scales Up Innovation to Reimagine the Online Grocery Experience. [Citado na página 12]
- [5] M. W. Live, “Us retailer inks ocado warehouse robotics deal.” Available at <https://www.mobileworldlive.com/industry/us-retailer-inks-ocado-warehouse-robotics-deal/>, July 2024. (Last accessed in 02/03/2025). [Citado nas páginas ix e 12]
- [6] J. Shieber, “Robomart is the latest startup to try and unseat the local convenience store,” 2018. Last accessed on 17/01/2025. [Citado na página 12]
- [7] N. Mafi, “Forget amazon go, robomart will bring groceries to your front door.” <https://www.architecturaldigest.com/story/forget-amazon-go-robomart-will-bring-groceries-front-door>, 2018. Architectural Digest. [Citado na página 12]
- [8] N. Luna, “Robomart unveils ambitious expansion plans for autonomous store-hailing fleet,” 2024. Last accessed on 17/01/2025. [Citado nas páginas ix e 12]
- [9] Exame, “Mercado móvel vai até você com a ajuda da inteligência artificial,” *Exame*, 2021. Accessed: 3 de February de 2025. [Citado na página 13]

-
- [10] S. H. Group, “Suning unmanned ’biu’ store to make international debut at ces 2018 to showcase smart retail worldwide,” 2018. Last accessed on 27/02/2025. [Citado na página 13]
- [11] jason, “Deals | fosun leads a us80 million investment in china’s amazon go.” Available at <https://kr-asia.com/fosun-leads-a-us80-million-investment-in-chinas-amazon-go>, Jan. 2018. (Last accessed in 22/02/2025). [Citado na página 13]
- [12] T. Technologies, “Takeoff technologies developing grocery micro fulfillment market,” 2018. On My Own Technology. [Citado na página 13]
- [13] B. Ladd, “Retail technology influencer brittain ladd weighs in on takeoff technologies chapter 11 bankruptcy filing,” June 2024. (Last accessed in 16/02/2025). [Citado nas páginas ix e 14]
- [14] S. P. Jean Walrand, *Communication Networks A Concise Introduction*. Springer, 2022. [Citado na página 14]
- [15] D. Meaney, “The evolution of wi-fi technologies: Wi-fi 7 (802.11be).” <https://ecsxtal.com/news-resources/the-evolution-of-wi-fi-technologies-wi-fi-7-802-11be/>, 2024. Accessed: 2025-03-02. [Citado nas páginas ix e 14]
- [16] Bluetooth, “Introducing bluetooth® channel sounding.” Available at <https://www.bluetooth.com/>, 2024. (Last accessed in 11/10/2024). [Citado na página 15]
- [17] Sony, “O que é a tecnologia sem fios bluetooth?.” Available at <https://www.sony.pt/electronics/support/articles/00030769>, 2024. (Last accessed in 08/10/2024). [Citado na página 15]
- [18] M. Veneri, “Bluetooth mesh can result in a smarter industrial environment.” Available at <https://embeddedcomputing.com/application/industrial/industrial-networking-connectivity/blue-mesh-can-result-in-a-smarter-industrial-environment>, Nov. 2017. (Last accessed in 19/02/2025). [Citado nas páginas ix e 15]
- [19] R. Weinstein, “Rfid: a technical overview and its application to the enterprise,” *IEEE IT Professional*, vol. 7, no. 3, pp. 27–33, 2005. [Citado na página 15]
- [20] V. Pandita, “How does rfid work: Science technology behind the systems!.” Available at <https://libcon.in/blog/How-RFID-works.aspx>, 2020. (Last accessed in 25/02/2025). [Citado nas páginas ix e 16]

-
- [21] C. Foresman, “Near field communications: a technology primer.” Available at <https://arstechnica.com/gadgets/2011/02/near-field-communications-a-technology-primer/>, 2024. (Last accessed in 12/10/2024). [Citado na página 16]
- [22] J. Fallows, “Nfc tags how they work.” Available at <https://play.fallows.ca/wp/insights/nfc-tags-how-they-work/>, 2017. (Last accessed in 02/03/2025). [Citado nas páginas ix e 16]
- [23] E. Redmond, “Introduction to mqtt protocol,” *Journal of IoT Applications*, vol. 15, no. 4, pp. 123–134, 2020. [Citado na página 17]
- [24] MQTT.org, “Mqtt: The standard for iot messaging.” Available at <https://mqtt.org/>, 2024. (Last accessed in 12/10/2024). [Citado nas páginas ix e 17]
- [25] J. Axelson, *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*. Lakeview Research, 2007. [Citado na página 18]
- [26] C. Basics, “Basics of uart communication,” n.d. Accessed: 08.05.2025. [Citado nas páginas ix e 18]
- [27] D. Symes, *I2C Bus: Implementation and Programming*. Newnes, 2001. [Citado na página 18]
- [28] P. Semiconductors, “I2c-bus specification and user manual,” *NXP Semiconductors*, 2000. [Citado na página 18]
- [29] S. Studio, “I2c communication interface and protocol with diagrams.” <https://www.seeedstudio.com/blog/2019/09/26/i2c-communication-interface-and-protocol-with-diagrams/>, 2019. (Accessed: 08.05.2025). [Citado nas páginas ix e 18]
- [30] W. Matthes, *The SPI Bus: Theory and Applications*. Springer, 2016. [Citado na página 19]
- [31] M. Inc., “Spi block guide,” *Freescale Semiconductor*, 2003. [Citado na página 19]
- [32] C. Basics, “Basics of the spi communication protocol.” <https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/>, n.d. Accessed: 08.05.2025. [Citado nas páginas ix e 19]
- [33] T. M. Fernandez-Carames, O. Blanco-Novoa, I. Froiz-Míguez, and P. Fraga-Lamas, “Towards an autonomous industry 4.0 warehouse: A uav and blockchain-based system for inventory and traceability applications in big data-driven supply chain management,” *arXiv preprint arXiv:2402.00709*, 2024. [Citado na página 19]

-
- [34] S. S. Manvi, P. Venkataram, and S. Bhargavi, “Automated retail store based on rfid,” in *2013 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, pp. 1–6, 2013. [Citado na página 19]
- [35] S. A. Brasel and J. Gips, “Getting in touch with your thinking style: How touchscreens influence purchase,” *Journal of Consumer Psychology*, vol. 24, no. 2, pp. 226–235, 2014. [Citado na página 20]
- [36] E. S. e Fabio Barbosa, “Um estudo sobre a sistematização e implementação de supermercado autônomo ou de autoatendimento,” *Seven Publicações*, 2025. [Citado na página 20]
- [37] S. T. Bukhari, A. W. Amin, M. A. Naveed, and M. R. Abbas, “Arc: A vision-based automatic retail checkout system,” *arXiv preprint arXiv:2104.02832*, 2021. [Citado na página 20]
- [38] ARC Experience, “Arc - autonomous retail checkout experience.” Available at <https://experiencearc.com/>, 2025. Accessed: 04 Mar. 2025. [Citado na página 20]
- [39] C. Grime, “Transforme um raspberry pi em um servidor web,” *Medium*, 2021. [Citado na página 21]
- [40] Desconhecido, “Como conectar um microcontrolador esp32 a um raspberry pi usando iot mqtt,” *Predictable Designs*, 2022. [Citado na página 21]
- [41] J. Silva and M. Pereira, “Desenvolvimento de um sistema de controle de esteira transportadora utilizando esp32 e comunicação wi-fi,” *Revista de Tecnologia Industrial*, vol. 15, no. 3, pp. 45–53, 2023. [Citado na página 21]
- [42] E. Roth, “Raspberry pi 5 gets a cheaper 2gb option.” Available at https://www.theverge.com/2024/8/19/24223494/raspberry-pi-5-cheaper-2gb-option?utm_source=chatgpt.com, 2024. Accessed: 07-03-2025. [Citado na página 26]
- [43] Raspberry Pi Foundation, “Raspberry pi 5 (8gb) - minicomputador,” 2024. [Citado nas páginas ix e 26]
- [44] nabto, “Esp32 for iot.” Available at <https://www.nabto.com/guide-to-iot-esp-32/>, 2024. (Last accessed in 05/11/2024). [Citado na página 27]
- [45] Az/delivery, “Nodemcu-esp-32s-kit.” Available at <https://www.az-delivery.de/en/products/nodemcu-esp-32s-kit>, 2024. (Last accessed in 28/10/2024). [Citado nas páginas ix e 27]

-
- [46] ElecFreaks, “Ultrasonic ranging module hc - sr04.” Available at <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>, 2024. (Last accessed in 16/11/2024). [Citado na página 27]
- [47] jahankit, “Jsn-sr04t-2.0.” <https://www.makerguides.com/wp-content/uploads/2019/02/JSN-SR04T-Datasheet.pdf>, 2024. (Last accessed in 16/11/2024). [Citado na página 28]
- [48] Makerguides, “Interfacing esp32 and jsn-sr04t waterproof ultrasonic sensor,” 2024. (Last accessed in 13/11/2024). [Citado na página 28]
- [49] Geekbuying, “Módulo de medição de distância ultra-sônica de sensor hc-sr04 3.3v / 5vt,” 2024. Last accessed on 28/10/2024. [Citado na página 28]
- [50] Aliexpress, “Módulo ultrassônico jsn-sr04t.” Available at <https://pt.aliexpress.com/i/1005001780453753.html>, 2024. (Last accessed in 28/10/2024). [Citado na página 28]
- [51] Amazon, “Monitor tátil ips 7"1024x600 hdmi para raspberry pi.” Available at <https://www.amazon.es/gp/product/B0CPM1T1XZ/>, 2025. Accessed: 11-03-2025. [Citado na página 29]
- [52] Tiendamia, “Monitor raspberry pi honyond com tela tátil ips lcd,” 2025. [Citado nas páginas ix e 29]
- [53] 38-3d, “Micro servo motor sg90 gear 9g.” Available at <https://38-3d.co.uk/pt/products/micro-servo-motor-sg90-gear-9g>, 2024. (Last accessed in 28/10/2024). [Citado nas páginas ix e 30]
- [54] T. Pro, “Mg996r servo motor.” Available at <https://towerpro.com.tw/product/mg996r/>, 2025. Accessed: 11-03-2025. [Citado na página 30]
- [55] D. Electronics, “Mg996r datasheet.” Available at <https://www.digikey.com/en/htmldatasheets/production/5014637/0/0/1/mg996r.html>, 2025. Accessed: 11-03-2025. [Citado na página 30]
- [56] Waveshare, “Mg996r servo motor - high torque, metal gear.” Available at <https://www.waveshare.com/mg996r-servo.htm>, 2025. Accessed: 11-03-2025. [Citado nas páginas ix e 31]
- [57] botnroll, “Módulo rfid rc522.” Available at <https://www.botnroll.com/pt/rfid-nfc/2580-m-dulo-rfid-rc522-13-56mhz-6cm-c-tags-s50.html>, 2024. (Last accessed in 28/10/2024). [Citado nas páginas ix e 31]

- [58] Bambu Lab, “Impressora 3d Bambu Lab A1.” <https://bambulab.com/pt-br/a1>, 2025. Accessed: 30-05-2025. [Citado na página 37]
- [59] Autodesk, “Fusion 360 — Software CAD/CAM/CAE 3D baseado na nuvem.” <https://www.autodesk.com/pt/products/fusion-360/overview>, 2025. Accessed: 09-06-2025. [Citado na página 38]

Anexo A

Código Fonte da Aplicação *Web*

A.1 Ficheiro `app.py`

Abaixo apresenta-se o código principal da interface *web* desenvolvida em Flask, responsável por gerir a navegação, o carrinho de compras, o *checkout* e a integração com o MQTT.

```
1 import os
2 import json
3 import logging
4 from flask import Flask, render_template, request, session, redirect,
    url_for, jsonify, flash
5 from mqtt_client import client, TOPIC_COMMANDS
6
7 # Configure logging
8 logging.basicConfig(level=logging.DEBUG)
9
10 # Create the app
11 app = Flask(__name__, static_url_path="/static")
12 app.secret_key = os.environ.get("SESSION_SECRET", "
    supermarket_secret_key")
13
14 # Product data (in-memory database)
15 PRODUCTS = {
16     "preserves": [
17         {
```

```
18         "id": "preserves-1",
19         "name": "Lata de Milho",
20         "price": 2.69,
21         "unit": "unid",
22         "description": "Lata de milho com 150g",
23         "image": "/static/images/milho.jpg"
24     },
25     {
26         "id": "preserves-2",
27         "name": "Salsichas em Lata NOBRE",
28         "price": 1.39,
29         "unit": "unid",
30         "description": "8 Salsichas em Lata",
31         "image": "/static/images/salsichas.png"
32     },
33     {
34         "id": "preserves-3",
35         "name": "Feijao preto em lata",
36         "price": 1.39,
37         "unit": "unid",
38         "description": "Feijao preto em lata com 410g",
39         "image": "/static/images/feijao.jpg"
40     }
41 ],
42 "cookies": [
43     {
44         "id": "cookies-1",
45         "name": "Bolachas OREO",
46         "price": 1.49,
47         "unit": "unid",
48         "description": "Bolachas de chocolate com Recheio de Creme
49 , cada embalagem com 154g",
50         "image": "/static/images/bolachas_oreo.jpg"
51     },
52     {
53         "id": "cookies-2",
54         "name": "Bolachas Maria",
55         "price": 0.55,
56         "unit": "unid",
57         "description": "Bolacha Maria",
58         "image": "/static/images/bolachas_maria.jpg"
59     },
60     {
61         "id": "cookies-3",
62         "name": "Chips Ahoy",
63         "price": 2.49,
64         "unit": "unid",
65         "description": "Bolacha Mini Chips Ahoy Pedacos Chocolate
- emb. 160g",
```

```
65         "image": "/static/images/chips_ahoy.jpg"
66     },
67     {
68         "id": "cookies-4",
69         "name": "Choco Flakes",
70         "price": 4.49,
71         "unit": "unid",
72         "description": "Bolacha Cuetara Flakes Recheada Chocolate
- emb. 500g",
73         "image": "/static/images/choco_flakes.jpg"
74     }
75 ],
76 "dairy": [
77     {
78         "id": "dairy-1",
79         "name": "Leite Mimosa Meio-Gordo",
80         "price": 0.99,
81         "unit": "unid",
82         "description": "O leite UHT Mimosa Bem Essencial Meio-
Gordo oferece a riqueza nutricional do leite com baixo teor de
gordura.",
83         "image": "/static/images/leite_mimosa.jpg"
84     },
85     {
86         "id": "dairy-2",
87         "name": "Leite Mimosa Magro",
88         "price": 0.99,
89         "unit": "unid",
90         "description": "O leite UHT Mimosa Bem Essencial Magro e
um aliado da Linha, Saude e Ritmo, essenciais ao seu bem-estar.",
91         "image": "/static/images/leite_mimosa_magro.jpg"
92     },
93     {
94         "id": "dairy-3",
95         "name": "Iogurte Yopro Baunilha",
96         "price": 1.29,
97         "unit": "unid",
98         "description": "Iogurte grego natural",
99         "image": "/static/images/yopro_baunilha.png"
100     },
101     {
102         "id": "dairy-4",
103         "name": "Manteiga Mimosa com Sal",
104         "price": 2.39,
105         "unit": "unid",
106         "description": "Manteiga com sal - emb. 250g",
107         "image": "/static/images/manteiga_mimosa.jpg"
108     }
109 ],
```

```
110     "bakery": [  
111         {  
112             "id": "bakery-1",  
113             "name": "Pao Garcia Cacete Sem Gluten",  
114             "price": 0.69,  
115             "unit": "unid",  
116             "description": "Pao sem gluten",  
117             "image": "/static/images/pao_garcia.jpg"  
118         },  
119         {  
120             "id": "bakery-2",  
121             "name": "Bollycao",  
122             "price": 0.80,  
123             "unit": "unid",  
124             "description": "Bolo Bollycao Recheado",  
125             "image": "/static/images/bollycao.jpg"  
126         }  
127     ],  
128     "candy": [  
129         {  
130             "id": "candy-1",  
131             "name": "Chocolate Milka Original",  
132             "price": 2.49,  
133             "unit": "unid",  
134             "description": "Tablete de Chocolate de Leite - emb. 90g",  
135             "image": "/static/images/milka_original.jpg"  
136         },  
137         {  
138             "id": "candy-2",  
139             "name": "Gomas Ositos de Oro",  
140             "price": 1.64,  
141             "unit": "unid",  
142             "description": "Gomas Haribo - emb. 250g",  
143             "image": "/static/images/haribo_gomas.jpeg"  
144         },  
145         {  
146             "id": "candy-3",  
147             "name": "Trident Spearmint",  
148             "price": 2.99,  
149             "unit": "unid",  
150             "description": "Pastilhas elasticas 60 minutos Bottle -  
emb. 82.6g",  
151             "image": "/static/images/trident.jpg"  
152         },  
153         {  
154             "id": "candy-4",  
155             "name": "Rebucados Peitorais",  
156             "price": 1.87,  
157             "unit": "unid",
```

```
158         "description": "Dr Bayard - emb.s 100g",
159         "image": "/static/images/rebucados_peitorais.jpg"
160     },
161     {
162         "id": "candy-5",
163         "name": "Halls",
164         "price": 0.69,
165         "unit": "unid",
166         "description": "Rebucados Originais Halls - emb. 32g",
167         "image": "/static/images/halls.jpg"
168     },
169     {
170         "id": "candy-6",
171         "name": "Smint",
172         "price": 2.00,
173         "unit": "unid",
174         "description": "Drageias mint de peppermint Smint - emb 35
175 g",
176         "image": "/static/images/smint.jpg"
177     }
178 ],
179 "carbs": [
180     {
181         "id": "101",
182         "name": "Arroz Agulha Cigala",
183         "price": 1.69,
184         "unit": "unid",
185         "description": "Arroz Agulha Cigala - emb. 1Kg",
186         "image": "/static/images/arroz_cigala.png"
187     },
188     {
189         "id": "102",
190         "name": "Massa Milaneza Helices",
191         "price": 1.57,
192         "unit": "unid",
193         "description": "Massa Milaneza Helices - emb. 500g",
194         "image": "/static/images/massa_helices.jpg"
195     }
196 ],
197 "oil": [
198     {
199         "id": "103",
200         "name": "Azeite Oliveira da Serra Virgem Extra Classico",
201         "price": 6.29,
202         "unit": "unid",
203         "description": "Em boca e um azeite suave com uma
204 intensidade media a ligeira de amargo e picante - emb. 750ml",
205         "image": "/static/images/azeite_oliveira.png"
206     }
207 ],
```

```
205     {
206         "id": "104",
207         "name": "oleo Alimentar Fula",
208         "price": 1.57,
209         "unit": "unid",
210         "description": "O Fula de sempre agora inclui oleos
essenciais de ervas aromaticas que reduzem o cheiro a fritos. -
emb. 1L",
211         "image": "/static/images/oleo_fula.jpg"
212     },
213 ],
214 "snacks": [
215     {
216         "id": "snacks-1",
217         "name": "Aperitivo Sunbites Sal Marinho",
218         "price": 2.15,
219         "unit": "unid",
220         "description": "Aperitivo Sunbites Sal Marinho - emb. 95g"
221     },
222     {
223         "id": "snacks-2",
224         "name": "Batatas Fritas Lays Gourmet",
225         "price": 1.59,
226         "unit": "unid",
227         "description": "Lays Gourmet - emb. 170g",
228         "image": "/static/images/lays.jpg"
229     },
230     {
231         "id": "snacks-3",
232         "name": "Lorenz Saltletts Brezel",
233         "price": 2.59,
234         "unit": "unid",
235         "description": "Aperitivos Salgados - emb. 150g",
236         "image": "/static/images/brezel.jpg"
237     }
238 ],
239 ]
240 }
241
242 # Categories with images
243 CATEGORIES = [
244     {
245         "id": "preserves",
246         "name": "Conservas",
247         "image": "/static/images/conservas.jpg"
248     },
249     {
250         "id": "cookies",
```

```
251         "name": "Bolachas",
252         "image": "/static/images/bolachas.jpg"
253     },
254     {
255         "id": "dairy",
256         "name": "Laticínios",
257         "image": "/static/images/laticinios.jpg"
258     },
259     {
260         "id": "bakery",
261         "name": "Padaria",
262         "image": "/static/images/padaria.jpg"
263     },
264     {
265         "id": "candy",
266         "name": "Doces",
267         "image": "/static/images/doces.jpg"
268     },
269     {
270         "id": "carbs",
271         "name": "Arroz e massa",
272         "image": "/static/images/arroz_massa.jpeg"
273     },
274     {
275         "id": "oil",
276         "name": "Azeite, óleo e Vinagre",
277         "image": "/static/images/azeite_oleo.png"
278     },
279     {
280         "id": "snacks",
281         "name": "Batatas Fritas e Aperitivos",
282         "image": "/static/images/snacks.jpg"
283     }
284 ]
285
286 # Helper function to get a flat list of all products
287 def get_all_products():
288     all_products = []
289     for category_products in PRODUCTS.values():
290         all_products.extend(category_products)
291     return all_products
292
293 # Helper function to find a product by ID
294 def find_product_by_id(product_id):
295     all_products = get_all_products()
296     for product in all_products:
297         if product["id"] == product_id:
298             return product
299     return None
```

```
300
301 # Initialize the cart in session if not present
302 @app.before_request
303 def init_cart():
304     if 'cart' not in session:
305         session['cart'] = {}
306
307 # Route for the home page
308 @app.route('/')
309 def index():
310     return render_template('index.html', categories=CATEGORIES)
311
312 # Route for viewing a specific category
313 @app.route('/category/<category_id>')
314 def category(category_id):
315     if category_id in PRODUCTS:
316         category_name = next((cat['name'] for cat in CATEGORIES if cat
317                               ['id'] == category_id), "Produtos")
318         return render_template('category.html',
319                                category_id=category_id,
320                                category_name=category_name,
321                                products=PRODUCTS[category_id])
322     else:
323         flash(f"Categoria {category_id} nao encontrada", "danger")
324         return redirect(url_for('index'))
325
326 # Route for viewing the cart
327 @app.route('/cart')
328 def cart():
329     cart_items = []
330     total = 0
331
332     for product_id, quantity in session.get('cart', {}).items():
333         product = find_product_by_id(product_id)
334         if product:
335             item_total = product['price'] * quantity
336             cart_items.append({
337                 'product': product,
338                 'quantity': quantity,
339                 'total': item_total
340             })
341             total += item_total
342
343     return render_template('cart.html', cart_items=cart_items, total=
344                             total)
345
346 # Route for adding an item to the cart
347 @app.route('/add_to_cart', methods=['POST'])
348 def add_to_cart():
```

```
347     product_id = request.form.get('product_id')
348     quantity = int(request.form.get('quantity', 1))
349
350     if not product_id:
351         flash("Produto nao especificado", "danger")
352         return redirect(request.referrer or url_for('index'))
353
354     product = find_product_by_id(product_id)
355     if not product:
356         flash("Produto nao encontrado", "danger")
357         return redirect(request.referrer or url_for('index'))
358
359     cart = session.get('cart', {})
360
361     # Update quantity if product already in cart
362     if product_id in cart:
363         cart[product_id] += quantity
364     else:
365         cart[product_id] = quantity
366
367     session['cart'] = cart
368     session.modified = True
369
370     flash(f"Adicionado {quantity} {product['name']} ao seu carrinho",
371           "success")
372     return redirect(request.referrer or url_for('index'))
373
374 # Route for updating cart item quantities
375 @app.route('/update_cart', methods=['POST'])
376 def update_cart():
377     product_id = request.form.get('product_id')
378     quantity = int(request.form.get('quantity', 0))
379
380     if not product_id:
381         flash("Produto nao especificado", "danger")
382         return redirect(url_for('cart'))
383
384     cart = session.get('cart', {})
385
386     if quantity > 0:
387         cart[product_id] = quantity
388     else:
389         # Remove item if quantity is 0
390         if product_id in cart:
391             del cart[product_id]
392
393     session['cart'] = cart
394     session.modified = True
```

```
395     flash("Carrinho atualizado", "success")
396     return redirect(url_for('cart'))
397
398 # Route for removing an item from the cart
399 @app.route('/remove_from_cart', methods=['POST'])
400 def remove_from_cart():
401     product_id = request.form.get('product_id')
402
403     if not product_id:
404         flash("Produto nao especificado", "danger")
405         return redirect(url_for('cart'))
406
407     cart = session.get('cart', {})
408
409     if product_id in cart:
410         del cart[product_id]
411
412     session['cart'] = cart
413     session.modified = True
414
415     flash("Item removido do carrinho", "success")
416     return redirect(url_for('cart'))
417
418
419 # Route for checking out
420 @app.route('/checkout')
421 def checkout():
422     cart_items = []
423     total = 0
424
425     for product_id, quantity in session.get('cart', {}).items():
426         product = find_product_by_id(product_id)
427         if product:
428             item_total = product['price'] * quantity
429             cart_items.append({
430                 'product': product,
431                 'quantity': quantity,
432                 'total': item_total
433             })
434             total += item_total
435
436     return render_template('checkout.html', cart_items=cart_items,
437                           total=total)
438
439 # Route for completing the checkout process
440 @app.route('/place_order', methods=['POST'])
441 def place_order():
442     cart = session.get('cart', {})
```

```
443     for product_id, quantidade in cart.items():
444         mensagem = f"{product_id}:{quantidade}"
445         client.publish(TOPIC_COMMANDS, mensagem)
446         print(f"[MQTT] Publicado: {mensagem}")
447
448     session['cart'] = {}
449     session.modified = True
450     flash("Seu pedido foi realizado! Obrigado por comprar conosco.", "
451     success")
452     return redirect(url_for('index'))
453
454 # Route for searching products
455 @app.route('/search')
456 def search():
457     query = request.args.get('query', '').lower()
458     if not query:
459         return redirect(url_for('index'))
460
461     # Search all products that contain the query in name or
462     # description
463     all_products = get_all_products()
464     results = [
465         product for product in all_products
466         if query in product['name'].lower() or query in product['
467         description'].lower()
468     ]
469
470     return render_template('search_results.html', query=query,
471     products=results)
472
473 # API endpoint to get cart count (for updating the cart indicator)
474 @app.route('/api/cart_count')
475 def cart_count():
476     cart = session.get('cart', {})
477     count = sum(cart.values())
478     return jsonify({'count': count})
479
480 if __name__ == '__main__':
481     app.run(host='0.0.0.0', port=5000, debug=True)
```

Listagem A.1: Ficheiro `app.py`

A.2 Ficheiro `mqtt_client.py`

Este ficheiro implementa a configuração e gestão do cliente MQTT utilizado na aplicação *web* desenvolvida com Flask.

```
1 # mqtt_client.py
2 import paho.mqtt.client as mqtt
3
4 # Configuracoes do Broker
5 BROKER_IP = "192.168.1.164" # IP do Raspberry Pi
6 BROKER_PORT = 1883
7 TOPIC_COMMANDS = "supermercado/comandos"
8 TOPIC_STATE = "supermercado/estado"
9
10 # Funcao chamada quando conectar
11 def on_connect(client, userdata, flags, rc):
12     print(f"Conectado ao broker MQTT com codigo: {rc}")
13     client.subscribe(TOPIC_STATE)
14
15 # Funcao chamada ao receber mensagens
16 def on_message(client, userdata, msg):
17     print(f"Mensagem recebida no topico {msg.topic}: {msg.payload.
18         decode()}")
19
20 # Inicializacao do cliente
21 client = mqtt.Client("FlaskClient") # ID diferente se o outro ainda
22     estiver a correr
23 client.on_connect = on_connect
24 client.on_message = on_message
25
26 # Conectar e iniciar loop em background
27 client.connect(BROKER_IP, BROKER_PORT, 60)
28 client.loop_start()
```

Listagem A.2: Ficheiro mqtt_client.py

A.3 Ficheiro main.py

Este ficheiro é responsável por iniciar simultaneamente o servidor *web*, desenvolvido com Flask, e o cliente MQTT. A comunicação MQTT é executada numa *thread* separada, garantindo que o sistema mantém responsividade na interface *web* enquanto gere a receção de mensagens provenientes dos microcontroladores.

```
1 from app import app
2 from mqtt_client import start_mqtt
3 import threading
4
5 if __name__ == '__main__':
6     print("A iniciar MQTT e Flask...")
7
8     # Thread para MQTT
```

```
9     mqtt_thread = threading.Thread(target=start_mqtt)
10     mqtt_thread.daemon = True # termina com o processo principal
11     mqtt_thread.start()
12
13     # Iniciar Flask
14     app.run(host='0.0.0.0', port=5000, debug=True)
```

Listagem A.3: Ficheiro *main.py*

A.4 Ficheiro *base.html*

Este ficheiro HTML define o base utilizado por todas as páginas da interface *web*.

```
1 <!DOCTYPE html>
2 <html lang="pt" data-bs-theme="dark">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale
6     =1.0">
7     <title>{% block title %}Supermercado QuickMart{% endblock %}</
8     title>
9
10     <!-- Bootstrap CSS from Replit CDN -->
11     <link rel="stylesheet" href="https://cdn.replit.com/agent/
12     bootstrap-agent-dark-theme.min.css">
13
14     <!-- Font Awesome for icons -->
15     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
16     libs/font-awesome/6.0.0/css/all.min.css">
17
18     <!-- Custom CSS -->
19     <link rel="stylesheet" href="{{ url_for('static', filename='css/
20     custom.css') }}">
21
22     {% block extra_css %}{% endblock %}
23 </head>
24 <body>
25     <!-- Navigation Bar -->
26     <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
27         <div class="container">
28             <a class="navbar-brand" href="{{ url_for('index') }}">
29                 <i class="fas fa-shopping-basket me-2"></i>QuickMart
30             </a>
31             <button class="navbar-toggler" type="button" data-bs-
32             toggle="collapse" data-bs-target="#navbarNav" aria-controls="
33             navbarNav" aria-expanded="false" aria-label="Toggle navigation">
34                 <span class="navbar-toggler-icon"></span>
```

```

28         </button>
29         <div class="collapse navbar-collapse" id="navbarNav">
30             <ul class="navbar-nav me-auto">
31                 {% for category in [
32                     {'id': 'preserves', 'name': 'Conservas'},
33                     {'id': 'cookies', 'name': 'Bolachas'},
34                     {'id': 'dairy', 'name': 'Laticínios'},
35                     {'id': 'bakery', 'name': 'Padaria'},
36                     {'id': 'candy', 'name': 'Doces'},
37                     {'id': 'carbs', 'name': 'Arroz e Massa'},
38                     {'id': 'oil', 'name': 'Azeite, óleo e Vinagre'},
39                     {'id': 'snacks', 'name': 'Batatas Fritas e
Aperitivos'}
40                 ] %}
41                 <li class="nav-item">
42                     <a class="nav-link" href="{% url_for('category
', category_id=category.id) %}">{{ category.name }}</a>
43                 </li>
44                 {% endfor %}
45             </ul>
46             <!-- Search Form -->
47             <form class="d-flex me-3" id="search-form" action="{%
url_for('search') %}" method="get">
48                 <input class="form-control me-2" id="search-input"
type="search" name="query" placeholder="Pesquisar produtos..."
aria-label="Pesquisar">
49                 <button class="btn btn-outline-light" type="submit
">
50                     <i class="fas fa-search"></i>
51                 </button>
52             </form>
53             <!-- Cart Link -->
54             <a href="{% url_for('cart') %}" class="btn btn-outline
-light position-relative">
55                 <i class="fas fa-shopping-cart"></i>
56                 <span class="badge bg-danger rounded-pill cart-
badge d-none" id="cart-count">0</span>
57             </a>
58         </div>
59     </div>
60 </nav>
61
62 <!-- Flash Messages -->
63 {% with messages = get_flashed_messages(with_categories=true) %}
64     {% if messages %}
65         <div class="container mt-3">
66             {% for category, message in messages %}

```

```

67         <div class="alert alert-{{ category }} alert-
dismissible fade show" role="alert">
68             {{ message }}
69             <button type="button" class="btn-close" data-
bs-dismiss="alert" aria-label="Close"></button>
70         </div>
71         {% endfor %}
72     </div>
73     {% endif %}
74 {% endwith %}
75
76 <!-- Main Content -->
77 <main>
78     {% block content %}{% endblock %}
79 </main>
80
81 <!-- Footer -->
82 <footer class="footer mt-5">
83     <div class="container">
84         <div class="row">
85             <div class="col-md-4 mb-3">
86                 <h5>Supermercado QuickMart</h5>
87                 <p>Sua loja completa de produtos frescos e de
qualidade.</p>
88             </div>
89             <div class="col-md-4 mb-3">
90                 <h5>Links Rápidos</h5>
91                 <ul class="list-unstyled">
92                     <li><a href="{{ url_for('index') }}" class="
text-light">QuickMart</a></li>
93                     <li><a href="{{ url_for('category',
category_id='preserves') }}" class="text-light">Conservas</a></li>
94                     <li><a href="{{ url_for('category',
category_id='cookies') }}" class="text-light">Bolachas</a></li>
95                     <li><a href="{{ url_for('cart') }}" class="
text-light">Carrinho de Compras</a></li>
96                 </ul>
97             </div>
98             <div class="col-md-4">
99                 <h5>Contato</h5>
100                 <address>
101                     <i class="fas fa-map-marker-alt me-2"></i> Rua
Principal, Porto<br>
102                     <i class="fas fa-phone me-2"></i> (+351) 915
066 802<br>
103                     <i class="fas fa-envelope me-2"></i>
info@quickmart.com
104                 </address>
105             </div>

```

```
106         </div>
107         <hr>
108         <div class="text-center">
109             <p>&copy; 2025 Supermercado QuickMart. Todos os
direitos reservados.</p>
110         </div>
111     </div>
112 </footer>
113
114 <!-- Bootstrap JS Bundle with Popper -->
115 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/
bootstrap.bundle.min.js"></script>
116
117 <!-- Custom JavaScript -->
118 <script src="{ { url_for('static', filename='js/main.js') } }"></
script>
119
120 {% block extra_js %}{% endblock %}
121 </body>
122 </html>
```

Listagem A.4: Ficheiro base.html

A.5 Ficheiro cart.html

Este ficheiro define a estrutura e o conteúdo da página do carrinho de compras da aplicação *web* do supermercado.

```
1 {% extends 'cart.html' %}
2
3 {% block title %}Carrinho de Compras - Supermercado QuickMart{%
endblock %}
4
5 {% block content %}
6     <div class="container my-4">
7         <nav aria-label="breadcrumb">
8             <ol class="breadcrumb">
9                 <li class="breadcrumb-item active" aria-current="page"
>Carrinho de Compras</li>
10             </ol>
11         </nav>
12
13         <h1 class="mb-4">Carrinho de Compras</h1>
14
15         {% if not cart_items %}
16             <div class="empty-state">
```

```

17         <i class="fas fa-shopping-cart fa-4x mb-3 text-muted">
18     </i>
19     <h3>Seu carrinho esta vazio</h3>
20     <p>Parece que voce ainda nao adicionou nenhum produto
    ao seu carrinho.</p>
21     <a href="{{ url_for('index') }}" class="btn btn-
    primary">Comecar a Comprar</a>
22     </div>
23     {% else %}
24     <div class="row">
25         <div class="col-lg-8">
26             <div class="card mb-4">
27                 <div class="card-header">
28                     <h5 class="mb-0">Itens no Carrinho ({{
    cart_items|length }})</h5>
29                 </div>
30                 <div class="card-body">
31                     <div class="table-responsive">
32                         <table class="table">
33                             <thead>
34                                 <tr>
35                                     <th>Produto</th>
36                                     <th>Preco</th>
37                                     <th>Quantidade</th>
38                                     <th>Total</th>
39                                     <th>Acoes</th>
40                                 </tr>
41                             </thead>
42                             <tbody>
43                                 {% for item in cart_items %}
44                                 <tr>
45                                     <td>
46                                         <div class="d-flex
    align-items-center">
47                                             
48                                             <div>
49                                                 <h6 class="mb
    -0">{{ item.product.name }}</h6>
50                                                 <small class="
    text-muted">{{ "%.2f"|format(item.product.price) }}/{{ item.
    product.unit }}</small>
51                                             </div>
52                                         </div>
53                                     </td>
54                                     <td> {{ "%.2f"|format(item
    .product.price) }}</td>

```

```

55         <form action="{{
url_for('update_cart') }}" method="post" class="d-inline">
56             <div class="
quantity-control">
57                 <button type="
button" class="btn btn-sm btn-outline-secondary btn-decrement"></
button>
58                 <input type="
number" name="quantity" value="{{ item.quantity }}" min="1" max="
99" class="form-control form-control-sm quantity-input" data-price
="{{ item.product.price }}">
59                 <button type="
button" class="btn btn-sm btn-outline-secondary btn-increment"></
button>
60                 <input type="
hidden" name="product_id" value="{{ item.product.id }}">
61             </div>
62             <button type="
submit" class="btn btn-sm btn-outline-primary ms-2">Atualizar</
button>
63         </form>
64     </td>
65     <td class="item-total"> {{
        "%.2f"|format(item.total) }}</td>
66     <td>
67         <form action="{{
url_for('remove_from_cart') }}" method="post">
68             <input type="
hidden" name="product_id" value="{{ item.product.id }}">
69             <button type="
submit" class="btn btn-sm btn-outline-danger">
70             <i class="fas
fa-trash"></i>
71             </button>
72         </form>
73     </td>
74 </tr>
75 {% endfor %}
76 </tbody>
77 </table>
78 </div>
79 </div>
80 </div>
81 </div>
82 <div class="col-lg-4">
83     <div class="card mb-4">
84         <div class="card-header">
85             <h5 class="mb-0">Resumo do Pedido</h5>
86         </div>

```

```

87         <div class="card-body">
88             <div class="d-flex justify-content-between
mb-3">
89                 <span>Subtotal</span>
90                 <span id="cart-total"> {{ "%.2f"|
format(total) }}</span>
91             </div>
92             <div class="d-flex justify-content-between
mb-3">
93                 <span>Frete</span>
94                 <span>Gratis</span>
95             </div>
96             <div class="d-flex justify-content-between
mb-3">
97                 <span>Impostos</span>
98                 <span> {{ "%.2f"|format(total * 0.08)
}}</span>
99             </div>
100             <hr>
101             <div class="d-flex justify-content-between
mb-3 fw-bold">
102                 <span>Total</span>
103                 <span> {{ "%.2f"|format(total * 1.08)
}}</span>
104             </div>
105             <a href="{{ url_for('checkout') }}" class=
"btn btn-success w-100">
106                 <i class="fas fa-check-circle me-2"></
i>Finalizar Compra
107             </a>
108         </div>
109     </div>
110     <div class="card">
111         <div class="card-body">
112             <h5 class="mb-3">Tem um cupom?</h5>
113             <div class="input-group">
114                 <input type="text" class="form-control
" placeholder="Codigo do cupom">
115                 <button class="btn btn-outline-
secondary" type="button">Aplicar</button>
116             </div>
117         </div>
118     </div>
119 </div>
120 </div>
121 <div class="mt-4">
122     <a href="{{ url_for('index') }}" class="btn btn-
outline-primary">

```

```

123             <i class="fas fa-arrow-left me-2"></i>Continuar
        Comprando
124             </a>
125         </div>
126         {% endif %}
127     </div>
128 {% endblock %}
129
130 {% block extra_js %}
131 {% endblock %}

```

Listagem A.5: Ficheiro cart.html

A.6 Ficheiro category.html

Este ficheiro define a estrutura e o conteúdo na página das diferentes categorias de produtos na aplicação *web*.

```

1 {% extends 'category.html' %}
2
3 {% block title %}{{ category_name }} - Supermercado QuickMart{%
    endblock %}
4
5 {% block content %}
6     <div class="container my-4">
7         <nav aria-label="breadcrumb">
8             <ol class="breadcrumb">
9                 <li class="breadcrumb-item active" aria-current="page"
10 >{{ category_name }}</li>
11             </ol>
12         </nav>
13
14         <h1 class="mb-4">{{ category_name }}</h1>
15
16         <!-- Display message if no products are available -->
17         {% if not products %}
18             <div class="empty-state">
19                 <i class="fas fa-search fa-4x mb-3 text-muted"></i>
20                 <h3>Nenhum produto encontrado</h3>
21                 <p>Nao conseguimos encontrar nenhum produto nesta
22 categoria. Por favor, volte mais tarde.</p>
23                 <a href="{{ url_for('index') }}" class="btn btn-
24 primary">Voltar ao Inicio</a>
25             </div>
26         {% else %}
27             <div class="row g-4">
28                 {% for product in products %}

```

```

26         <div class="col-sm-6 col-md-4 col-lg-3">
27             <div class="card product-card h-100">
28                 <div class="product-img-container">
29                     
30                 </div>
31                 <div class="card-body">
32                     <h5 class="card-title">{{ product.name }}<
/h5>
33                     <p class="card-text">{{ product.
description }}</p>
34                     <p class="card-text fw-bold"> {{ "%.2f"|
format(product.price) }}/{{ product.unit }}</p>
35                     <form action="{{ url_for('add_to_cart') }}"
" method="post" class="d-flex flex-column">
36                         <div class="input-group mb-3">
37                             <span class="input-group-text">Qtd
</span>
38                             <input type="number" class="form-
control" name="quantity" value="1" min="1" max="99">
39                             <input type="hidden" name="
product_id" value="{{ product.id }}">
40                         </div>
41                         <button type="submit" class="btn btn-
primary">
42                             <i class="fas fa-cart-plus me-2"><
/i>Adicionar ao Carrinho
43                         </button>
44                     </form>
45                 </div>
46             </div>
47         </div>
48     {% endfor %}
49 </div>
50 {% endif %}
51 </div>
52 {% endblock %}

```

Listagem A.6: Ficheiro `category.html`

A.7 Ficheiro `checkout.html`

Este ficheiro define a estrutura e o conteúdo da página de *checkout* na aplicação *web*.

```

1 {% extends 'base.html' %}
2

```

```

3 {% block title %}Finalizar Compra - Supermercado QuickMart{% endblock
  %}
4
5 {% block content %}
6     <div class="container my-4">
7         <nav aria-label="breadcrumb">
8             <ol class="breadcrumb">
9                 <li class="breadcrumb-item"><a href="{{ url_for('cart
10                 ')) }}">Carrinho de Compras</a></li>
11                 <li class="breadcrumb-item active" aria-current="page">
12                 >Finalizar Compra</li>
13             </ol>
14         </nav>
15
16         <h1 class="mb-4">Finalizar Compra</h1>
17
18         {% if not cart_items %}
19             <div class="empty-state">
20                 <i class="fas fa-shopping-cart fa-4x mb-3 text-muted">
21                 </i>
22                 <h3>Seu carrinho esta vazio</h3>
23                 <p>Voce nao pode finalizar a compra com um carrinho
24                 vazio.</p>
25                 <a href="{{ url_for('index') }}" class="btn btn-
26                 primary">Comecar a Comprar</a>
27             </div>
28         {% else %}
29             <div class="row">
30                 <div class="col-lg-8">
31                     <form id="checkout-form" action="{{ url_for('
32                     place_order') }}" method="post">
33                         <!-- Customer Information -->
34                         <div class="form-section mb-4">
35                             <h3 class="mb-3">Informacoes do Cliente</
36                             h3>
37                             <div class="row g-3">
38                                 <div class="col-md-6">
39                                     <label for="firstName" class="form-

```

```

40                                     <label for="email" class="form-
label">Email</label>
41                                     <input type="email" class="form-
control" id="email" required>
42                                     </div>
43                                     <div class="col-md-6">
44                                     <label for="phone" class="form-
label">Telefone</label>
45                                     <input type="tel" class="form-
control" id="phone" required>
46                                     </div>
47                                 </div>
48 </div>
49
50 <!-- Pickup Address -->
51 <div class="form-section mb-4">
52     <h3 class="mb-3">Loja de Levantamento</h3>
53     <div class="row g-3">
54         <div class="col-12">
55             <label for="address" class="form-
label">Loja</label>
56             <select class="form-select" id="
state" required>
57                 <option value="">Escolha...</
option>
58                 <option>QuickMart Lisboa</
option>
59                 <option>QuickMart Porto</
option>
60                 <option>QuickMart Faro</option
>
61                 <option>QuickMart Braga</
option>
62             </select>
63         </div>
64     </div>
65 </div>
66
67 <!-- Payment Information -->
68 <div class="form-section mb-4">
69     <h3 class="mb-3">Informacoes de Pagamento<
/h3>
70     <div class="row g-3">
71         <div class="col-md-6">
72             <label for="cardName" class="form-
label">Nome no Cartao</label>
73             <input type="text" class="form-
control" id="cardName" required>
74         </div>

```

```

75         <div class="col-md-6">
76             <label for="cardNumber" class="
form-label">Numero do Cartao</label>
77             <input type="text" class="form-
control" id="cardNumber" required>
78         </div>
79         <div class="col-md-4">
80             <label for="expMonth" class="form-
label">Mes de Validade</label>
81             <select class="form-select" id="
expMonth" required>
82                 <option value="">Escolha...</
option>
83                 <option>01</option>
84                 <option>02</option>
85                 <option>12</option>
86             </select>
87         </div>
88         <div class="col-md-4">
89             <label for="expYear" class="form-
label">Ano de Validade</label>
90             <select class="form-select" id="
expYear" required>
91                 <option value="">Escolha...</
option>
92                 <option>2023</option>
93                 <option>2024</option>
94                 <option>2025</option>
95                 <option>2026</option>
96                 <option>2027</option>
97             </select>
98         </div>
99         <div class="col-md-4">
100             <label for="cvv" class="form-label
">CVV</label>
101             <input type="text" class="form-
control" id="cvv" required>
102         </div>
103     </div>
104 </div>
105
106     <div class="d-grid gap-2">
107         <button type="submit" class="btn btn-
success btn-lg">
108             <i class="fas fa-lock me-2"></i>
Finalizar Pedido
109         </button>
110     </div>
111 </form>

```



```

152             <span> {{ "%.2f"|format(total * 0.08)
    }}</span>
153         </div>
154         <hr>
155         <div class="d-flex justify-content-between
    mb-0 fw-bold">
156             <span>Total</span>
157             <span> {{ "%.2f"|format(total * 1.08)
    }}</span>
158         </div>
159     </div>
160 </div>
161 </div>
162 </div>
163 {% endif %}
164 </div>
165 {% endblock %}

```

Listagem A.7: Ficheiro checkout.html

A.8 Ficheiro index.html

Este ficheiro define a estrutura e o conteúdo da página inicial na aplicação *web*.

```

1 {% extends 'base.html' %}
2
3 {% block title %}Supermercado QuickMart - Produtos Frescos & Mais{%
    endblock %}
4
5 {% block content %}
6     <!-- Hero Section -->
7     <section class="hero-section">
8         <div class="container text-center">
9             <h1 class="display-4">Bem-vindo ao QuickMart</h1>
10            <p class="lead">Sua loja completa para produtos frescos e
    de qualidade</p>
11            <a href="#categories" class="btn btn-primary btn-lg mt-3">
12                <i class="fas fa-shopping-basket me-2"></i> Começar a
    Comprar
13            </a>
14        </div>
15    </section>
16
17    <!-- Featured Categories -->
18    <section class="container my-5" id="categories">
19        <h2 class="text-center mb-4">Compre por Categoria</h2>
20        <div class="row g-4">

```

```

21         {% for category in categories %}
22         <div class="col-md-6 col-lg-3">
23             <a href="{% url_for('category', category_id=category.
24 id) %}" class="text-decoration-none">
25                 <div class="card category-card h-100">
26                     <div class="category-img-container">
27                         
29                     </div>
30                     <div class="category-overlay">
31                         <h3 class="h5 mb-0">{% category.name %}</
32 h3>
33                     </div>
34                 </div>
35             </a>
36         </div>
37         {% endfor %}
38     </div>
39 </section>
40
41 <!-- Featured Specials -->
42 <section class="container my-5 py-4 bg-dark rounded">
43     <h2 class="text-center mb-4">Promoco'es da Semana</h2>
44     <div class="row g-4">
45         <div class="col-md-6 col-lg-3">
46             <div class="card product-card h-100">
47                 <div class="product-img-container">
48                     
51                 </div>
52                 <div class="card-body">
53                     <h3 class="card-title">Bananas</h3>
54                     <p class="card-text text-success">Preco
55 especial: 0,49/kg</p>
56                     <p class="card-text"><small class="text-muted"
57 >Preco normal: 0,59/kg</small></p>
58                     <form action="{% url_for('add_to_cart') %}"
59 method="post">
60                         <input type="hidden" name="product_id"
61 value="preserves-2">
62                         <input type="hidden" name="quantity" value
63 =1">
64                         <button type="submit" class="btn btn-
65 primary">
66                             <i class="fas fa-cart-plus me-2"></i>
67                             Adicionar ao Carrinho
68                         </button>
69                     </form>

```

```

58         </div>
59     </div>
60 </div>
61 <div class="col-md-6 col-lg-3">
62     <div class="card product-card h-100">
63         <div class="product-img-container">
64             
65         </div>
66         <div class="card-body">
67             <h5 class="card-title">Brocolis</h5>
68             <p class="card-text text-success">Preço
especial: 1,99/unid</p>
69             <p class="card-text"><small class="text-muted"
>Preço normal: 2,49/unid</small></p>
70             <form action="{ { url_for('add_to_cart') } }"
method="post">
71                 <input type="hidden" name="product_id"
value="veg-2">
72                 <input type="hidden" name="quantity" value
="1">
73                 <button type="submit" class="btn btn-
primary">
74                     <i class="fas fa-cart-plus me-2"></i>
Adicionar ao Carrinho
75                 </button>
76             </form>
77         </div>
78     </div>
79 </div>
80 <div class="col-md-6 col-lg-3">
81     <div class="card product-card h-100">
82         <div class="product-img-container">
83             
84         </div>
85         <div class="card-body">
86             <h5 class="card-title">Leite</h5>
87             <p class="card-text text-success">Preço
especial: 2,99/L</p>
88             <p class="card-text"><small class="text-muted"
>Preço normal: 3,49/L</small></p>
89             <form action="{ { url_for('add_to_cart') } }"
method="post">
90                 <input type="hidden" name="product_id"
value="dairy-1">

```

```

91         <input type="hidden" name="quantity" value
    = "1">
92         <button type="submit" class="btn btn-
    primary">
93             <i class="fas fa-cart-plus me-2"></i>
    Adicionar ao Carrinho
94         </button>
95     </form>
96 </div>
97 </div>
98 </div>
99 <div class="col-md-6 col-lg-3">
100     <div class="card product-card h-100">
101         <div class="product-img-container">
102             
103         </div>
104         <div class="card-body">
105             <h5 class="card-title">Pao</h5>
106             <p class="card-text text-success">Preco
    especial: 2,49/unid</p>
107             <p class="card-text"><small class="text-muted"
    >Preco normal: 2,99/unid</small></p>
108             <form action="{ { url_for('add_to_cart') } }"
    method="post">
109                 <input type="hidden" name="product_id"
    value="bakery-1">
110                 <input type="hidden" name="quantity" value
    = "1">
111                 <button type="submit" class="btn btn-
    primary">
112                     <i class="fas fa-cart-plus me-2"></i>
    Adicionar ao Carrinho
113                 </button>
114             </form>
115         </div>
116     </div>
117 </div>
118 </div>
119 </section>
120
121 <!-- Shopping Features -->
122 <section class="container my-5">
123     <h2 class="text-center mb-4">Por que Comprar Conosco?</h2>
124     <div class="row g-4">
125         <div class="col-md-4">
126             <div class="card h-100 text-center">
127                 <div class="card-body">

```

```

128         <i class="fas fa-truck fa-3x mb-3 text-primary
"></i>
129         <h5 class="card-title">Entrega Rápida</h5>
130         <p class="card-text">Receba suas compras em
sua casa em poucas horas.</p>
131     </div>
132 </div>
133 </div>
134 <div class="col-md-4">
135     <div class="card h-100 text-center">
136         <div class="card-body">
137             <i class="fas fa-leaf fa-3x mb-3 text-success"
></i>
138             <h5 class="card-title">Produtos Frescos</h5>
139             <p class="card-text">Fornecemos os produtos
mais frescos e locais para sua família.</p>
140         </div>
141     </div>
142 </div>
143 <div class="col-md-4">
144     <div class="card h-100 text-center">
145         <div class="card-body">
146             <i class="fas fa-euro-sign fa-3x mb-3 text-
warning"></i>
147             <h5 class="card-title">Melhores Precos</h5>
148             <p class="card-text">Aproveite precos
competitivos e promocoões semanais.</p>
149         </div>
150     </div>
151 </div>
152 </div>
153 </section>
154
155 <!-- Newsletter Signup -->
156 <section class="container my-5 py-4 bg-dark rounded">
157     <div class="row justify-content-center">
158         <div class="col-md-8 text-center">
159             <h3>Inscreva-se em Nossa Newsletter</h3>
160             <p>Receba atualizações semanais sobre novos produtos e
ofertas especiais!</p>
161             <form class="row g-3 justify-content-center">
162                 <div class="col-md-8">
163                     <input type="email" class="form-control"
placeholder="Digite seu email">
164                 </div>
165                 <div class="col-md-4">
166                     <button type="submit" class="btn btn-primary w
-100">Inscrever-se</button>
167                 </div>

```

```

168         </form>
169     </div>
170 </div>
171 </section>
172 {% endblock %}

```

Listagem A.8: Ficheiro `index.html`

A.9 Ficheiro `search_results.html`

Este ficheiro permite a procura de um determinado produto na aplicação *web*.

```

1 {% extends 'base.html' %}
2
3 {% block title %}Resultados da Busca por "{{ query }}" - Supermercado
   QuickMart{% endblock %}
4
5 {% block content %}
6     <div class="container my-4">
7         <nav aria-label="breadcrumb">
8             <ol class="breadcrumb">
9                 <li class="breadcrumb-item active" aria-current="page">
>Resultados da Busca</li>
10             </ol>
11         </nav>
12
13         <h1 class="mb-4">Resultados da Busca por "{{ query }}"</h1>
14
15         <!-- Display message if no products are found -->
16         {% if not products %}
17             <div class="empty-state">
18                 <i class="fas fa-search fa-4x mb-3 text-muted"></i>
19                 <h3>Nenhum produto encontrado</h3>
20                 <p>Nao conseguimos encontrar nenhum produto que
corresponda aos seus criterios de busca.</p>
21                 <a href="{{ url_for('index') }}" class="btn btn-
primary">Voltar ao Inicio</a>
22             </div>
23         {% else %}
24             <p class="mb-4">Encontrado(s) {{ products|length }}
resultado(s) para sua busca "{{ query }}"</p>
25
26             <div class="row g-4">
27                 {% for product in products %}
28                 <div class="col-sm-6 col-md-4 col-lg-3">
29                     <div class="card product-card h-100">
30                         <div class="product-img-container">

```

```

31         
32     </div>
33     <div class="card-body">
34         <h5 class="card-title">{{ product.name }}<
/h5>
35         <p class="card-text">{{ product.
description }}</p>
36         <p class="card-text fw-bold"> {{ "%.2f"|
format(product.price) }}/{{ product.unit }}</p>
37         <form action="{{ url_for('add_to_cart') }}"
" method="post" class="d-flex flex-column">
38             <div class="input-group mb-3">
39                 <span class="input-group-text">Qtd
</span>
40                 <input type="number" class="form-
control" name="quantity" value="1" min="1" max="99">
41                 <input type="hidden" name="
product_id" value="{{ product.id }}">
42             </div>
43             <button type="submit" class="btn btn-
primary">
44                 <i class="fas fa-cart-plus me-2"><
/i>Adicionar ao Carrinho
45             </button>
46         </form>
47     </div>
48 </div>
49 </div>
50 {% endfor %}
51 </div>
52 {% endif %}
53 </div>
54 {% endblock %}

```

Listagem A.9: Ficheiro search_results.html.html

Anexo B

Código Fonte do Programa do ESP32

B.1 Ficheiro `main.cpp`

O código apresentado de seguida corresponde ao ficheiro `main.cpp`, responsável por controlar os dispensadores automáticos, tapete rolante e sensores. Este ficheiro implementa a receção de mensagens via MQTT, a ativação de servomotores e o controlo do tapete rolante.

```
1
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4 #include <ESP32Servo.h>
5
6 // Wi-Fi e MQTT
7 const char* ssid = "DIGIFIBRA-PLUS-0719";
8 const char* password = "M7+eN3eDBL";
9 const char* mqttServer = "192.168.1.164";
10 const int mqttPort = 1883;
11 const char* mqttUser = "usuario";
12 const char* mqttPassword = "senha";
13
14 // Motores e sensores
15 const int motorIN1 = 26;
```

```
16 const int motorIN2 = 27;
17
18 // Substituímos o sensorFimTapete pelo sensor ultrassônico
19 const int TRIG_PIN = 12;
20 const int ECHO_PIN = 13;
21
22 // Servo
23 const int NUM_PRODUTOS = 4;
24 Servo servos[NUM_PRODUTOS];
25 int servo_pins[NUM_PRODUTOS] = {18, 19, 21, 22};
26
27 WiFiClient espClient;
28 PubSubClient client(espClient);
29
30 // Função para mapear IDs dos produtos
31 int mapearProdutoParaServo(String productId) {
32     if (productId == "101") return 0;
33     else if (productId == "102") return 1;
34     else if (productId == "103") return 2;
35     else if (productId == "104") return 3;
36     else return -1;
37 }
38
39 // Leitura do sensor ultrassônico
40 bool detectarProdutoComUltrassônico() {
41     digitalWrite(TRIG_PIN, LOW);
42     delayMicroseconds(2);
43     digitalWrite(TRIG_PIN, HIGH);
44     delayMicroseconds(10);
45     digitalWrite(TRIG_PIN, LOW);
46
47     long duracao = pulseIn(ECHO_PIN, HIGH);
48     float distancia = duracao * 0.034 / 2;
49
50     Serial.print("Distância medida: ");
51     Serial.print(distancia);
52     Serial.println(" cm");
53
54     return (distancia > 0 && distancia <= 15); // Produto detectado a
        menos de 15cm
55 }
56
57 void setup_wifi() {
58     WiFi.begin(ssid, password);
59     while (WiFi.status() != WL_CONNECTED) {
60         delay(1000);
61         Serial.print(".");
62     }
63     Serial.println("Wi-Fi conectado");
```

```
64     Serial.println(WiFi.localIP());
65 }
66
67 void acionarTapete() {
68     digitalWrite(motorIN1, HIGH);
69     digitalWrite(motorIN2, LOW);
70     Serial.println("Tapete rolante ativado");
71
72     unsigned long tempoInicio = millis();
73     const unsigned long TIMEOUT = 40000;
74
75     while (!detetarProdutoComUltrassonico() && millis() - tempoInicio
76 < TIMEOUT) {
77         delay(50);
78     }
79
80     if (detetarProdutoComUltrassonico()) {
81         Serial.println("Produto detetado com sensor ultrassonico.");
82     } else {
83         Serial.println("Timeout: produto n o detetado.");
84     }
85
86     digitalWrite(motorIN1, LOW);
87     digitalWrite(motorIN2, LOW);
88     Serial.println("Tapete rolante parado");
89 }
90
91 void acionarDispenser(int servoIndex, int quantidade) {
92     if (servoIndex < 0 || servoIndex >= NUM_PRODUTOS) return;
93
94     for (int i = 0; i < quantidade; i++) {
95         Serial.println("A Dispensar produto");
96         servos[servoIndex].write(90);
97         delay(1000);
98         servos[servoIndex].write(180);
99         delay(1000);
100         servos[servoIndex].write(90);
101         acionarTapete();
102     }
103 }
104
105 void callback(char* topic, byte* message, unsigned int length) {
106     String msg;
107     for (int i = 0; i < length; i++) msg += (char)message[i];
108
109     int sep = msg.indexOf(':');
110     if (sep != -1) {
111         String id = msg.substring(0, sep);
112         int q = msg.substring(sep + 1).toInt();
```

```
112     int idx = mapearProdutoParaServo(id);
113     acionarDispenser(idx, q);
114 }
115 }
116
117 void reconnect() {
118     while (!client.connected()) {
119         if (client.connect("ESP32Client", mqttUser, mqttPassword)) {
120             client.subscribe("supermercado/comandos");
121         } else {
122             delay(5000);
123         }
124     }
125 }
126
127 void setup() {
128     Serial.begin(115200);
129
130     pinMode(TRIG_PIN, OUTPUT);
131     pinMode(ECHO_PIN, INPUT);
132
133     for (int i = 0; i < NUM_PRODUTOS; i++) {
134         servos[i].attach(servo_pins[i]);
135     }
136
137     pinMode(motorIN1, OUTPUT);
138     pinMode(motorIN2, OUTPUT);
139     digitalWrite(motorIN1, LOW);
140     digitalWrite(motorIN2, LOW);
141
142     setup_wifi();
143     client.setServer(mqttServer, mqttPort);
144     client.setCallback(callback);
145 }
146
147 void loop() {
148     if (!client.connected()) reconnect();
149     client.loop();
150
151     static unsigned long lastPublish = 0;
152     if (millis() - lastPublish > 10000) {
153         client.publish("supermercado/estado", "ESP32 Online");
154         lastPublish = millis();
155     }
156 }
```

Listagem B.1: Ficheiro main.cpp do ESP32