FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# Sistema de Localização de Objetos para Gestão de Stocks

**Duarte Marques Aguiar**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: João Rodrigues

Co-orientador: Rui Esteves Araújo

29 de julho de 2021

# Resumo

Numa loja de venda a retalho, quando ocorre ruptura de stocks de um produto numa prateleira, por períodos de longa duração, podem ocorrer perdas consideráveis de lucro. Dessa forma, o sistema desenvolvido ambiciona alertar os repositores da loja dessas rupturas de stock de modo a preveni-las. Ademais, o sistema elaborado disponibiliza ao retalhista informação sobre: a quantidade de produtos retirados das prateleiras; a deslocação de produtos para outras prateleiras que não as designadas; possíveis furtos de produtos; e erros de posicionamento de produtos, de acordo com acordos comerciais que ditam a disposição dos produtos.

O sistema desenvolvido utiliza duas células de carga por cada prateleira de forma a detetar movimentos que ocorram e determinar o peso e a posição relativa associados a cada um. Dado que cada prateleira poderá conter mais do que um tipo de produto, é necessário inferir a quantidade e o tipo de produtos envolvidos num dado movimento. Para tal é utilizada a informação acerca da composição e disposição dos produtos na prateleira, designados por planogramas.

# Abstract

In a retail store, when stocks of a product run out on a shelf for long periods of time, potential losses of profit can occur. In this way, the developed system aims to alert store repositors of these stock outages in order to prevent them. In addition, the system developed provides the retailer with information on: the quantity of products removed from the shelves; the movement of products to other shelves other than those designated; and possible theft of products.

The developed system uses two load cells for each shelf in order to detect movements that occur and determine the weight and relative position associated with each one. Since each shelf may contain more than one type of product, it is necessary to infer the quantity and type of products involved in a given movement. For this, information about the disposition of products on the shelf, called planograms, is used.

# Agradecimentos

Gostaria de agradecer ao meu orientador o professor Rui Esteves Araújo e aos engenheiros João Rodrigues e Tiago Silva (da Xhockware) pelo apoio prestado e pela constante orientação e motivação. Ademais, agradeço aos elementos da Xhockware pela confiança e por me permitirem desenvolver este mesmo projeto.

Por fim, agradeço a todos os meus familiares e amigos pela motivação e apoio ao longo deste percurso.

Duarte Marques Aguiar

*"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time."*

Thomas Edison

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas e Símbolos

| | |
|---|---|
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| ISR | Interrupt Service Routine |
| MCU | Microcontroller Unit |
| REST | Representational state transfer |
| LDR | Light Dependent Resistor |
| QR | Quick Response |
| REST | Representational State Transfer |
| HTTP | HyperText Transfer Protocol |
| UTP | Unshielded twisted pair |
| ADC | Conversor analógico/digital |
| UART | Universal Asynchronous Receiver-Transmitter |
| I2C | Inter-Integrated Circuit |
| SPI | Serial Peripheral Interface |
| RAM | Random-access memory |
| SOC | System On a Chip |
| CAN | Controller Area Network |
| OSI | Open Systems Interconnection |
| CSMA/CD+AMP | Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority |
| RTR | Remote Transmission Request |
| DLC | Data Length Code |
| CRC | Cyclic Redundancy Check |
| ACK | Acknowledgement |
| TCP | Transmission Control Protocol |
| URI | Uniform Resource Identifier |
| MySQL | My Strucure Query Language |
| PHP | Hypertext Preprocessor |
| HTML | Hypertext Markup Language |
| USB | Universal Serial Bus |
| U/UTP | Unshielded/Unshielded twisted pair |
| PD_SCK | Power down control and serial clock input |
| DOUT | Serial data output |
| RATE | Output data rate control input |

# Capítulo 1

# Introdução

## 1.1 Enquadramento e Motivação

O tema para esta dissertação foi proposto pela empresa Xhockware. Uma *start-up* que visa oferecer soluções inovadoras para empresas/grupos existentes nas áreas do retalho e venda direta ao consumidor. Sendo assim, a empresa pretende desenvolver um produto que visa efetuar o controlo/gestão em tempo real de *stocks* de produtos de superfícies comerciais, disponível em prateleiras. Possibilitando, assim, a automatização de processos de reposição de *stock* e análise de preferências de consumo.

No âmbito desta tese, pretende-se desenvolver um sistema capaz de detetar movimentações de produtos realizadas em prateleiras de supermercado de forma a obter, em tempo real, o *stock* de cada tipo de produto existente. Pretende-se que as movimentações de produtos sejam detetadas através da monitorização de variações de peso na superfície da prateleira e atendendo também à disposição dos produtos na mesma - planogramas. O sistema deverá gerar alertas quando o *stock* de um dado produto é reduzido de forma a que a reposição de *stock* seja feita o mais rapidamente possível e assim evitar perdas a curto e longo prazo.

No mundo competitivo que se vive atualmente, as situações de ruptura de *stocks* são um problema específico e de longa duração que podem levar a perdas de curto e longo prazo. Porque, embora um cliente que seja fiel a um dado supermercado possa desculpar uma situação única de ruptura de *stock*, não o fará se a situação for recorrente. Significando que as superfícies comerciais conseguem prevenir que um cliente mude para uma empresa da competição ao evitar rupturas de *stock* nos seus expositores [1].

Os produtos quando colocados nas prateleiras dos supermercados seguem uma organização pré estabelecida, à qual se dá o nome de planograma. Um planograma é um diagrama que define: o local; a forma de exposição; o número de frentes e níveis dos produtos; o número de diferentes tipos de produtos; e os espaçamentos entre produtos.Tornando os produtos acessíveis aos clientes de uma maneira organizada e concisa [2].

A empresa já possui trabalho desenvolvido nesta área. A maior parte do trabalho está relacionado com a sensorização das prateleiras e a comunicação entre os sensores e o elemento que faz a

ligação entre os sensores e o servidor. Possuem igualmente trabalho desenvolvido na vertente da aplicação web e da base de dados relacional.

## 1.2 Objetivos

O objetivo principal desta dissertação passa por desenvolver uma aplicação passível de ser usada por cadeias de supermercados como forma de manter registo de *stocks* de produtos em tempo real e gerar alertas quando os níveis de *stock* são baixos.

Sendo assim, o sistema deverá ser capaz de determinar a posição numa superfície, a prateleira do supermercado, onde se verifica uma variação de peso e determinar a ocorrência de vários eventos distintos. Como por exemplo a remoção de objetos, adição de objetos, a deslocação de objetos pela superfície, erros de conformidade ao planograma e situações de erro inconsistência. Depois, com base nestes eventos serão inferidos os tipos e quantidades de produtos que estão envolvidos. Ademais, o algoritmo de deteção de eventos deverá possuir um grau de robustez elevado de forma a evitar a deteção de eventos erróneos e determinar a quantidade e tipos de produtos com um nível de certeza elevado.

Ainda, deverá ser desenvolvida uma *application programming interface* (API) capaz de fazer a gestão e visualização dos eventos que ocorrem tal como obter uma imagem em tempo real dos produtos que existam nas prateleiras. Por fim, o sistema deverá ser escalável e flexível.

## 1.3 Estrutura do documento

Tendo em consideração a informação anteriormente apresentada, o presente trabalho encontra-se organizado da seguinte forma.

Primeiramente, apresenta-se uma revisão bibliográfica que contemplará os seguintes tópicos: soluções existentes no mercado; arquitetura do sistema proposto; células de carga; *smart sensor*; controlador de prateleira; concentrador; comunicações; *web server* e lógica difusa. De seguida, no capítulo 3 é explorado o algoritmo de aquisição e processamento do sinal das células de carga, tal como os algoritmos de inferência de eventos e a arquitetura do controlador.

No capítulo 4, são discutidos os algoritmos utilizados para a gestão de stock. O algoritmo de inferência de movimentações de stock e as interfaces que permitem a interacção dos funcionários com as prateleiras. No final são demonstrados e discutidos os resultados obtidos de testes realizados a cada parte do sistema.

Por fim, no capítulo 5 são apresentadas as conclusões e as perspectivas de evolução do trabalho.

# Capítulo 2

# Revisão Bibliográfica

## 2.1 Soluções existentes no mercado

Nesta secção serão apresentados alguns produtos com características idênticas ao produto que é proposto desenvolver, destacando as funcionalidades que mais se assemelham aos objetivos do projeto. Nomeadamente, a *SmartShelf Weighting Pad*; *ShelfX Smart Fridge*; *WiseShelf*; e a *Amazon Go*. No final da secção é feita uma comparação entre cada uma destas.

***SmartShelf Weighing Pad***

A *SmartShelf Weighing Pad* é um produto desenvolvido pela empresa METTLER TOLEDO para ambiente industrial [3]. Este produto oferece a monitorização de stocks de forma fácil e com possibilidade de monitorização em tempo real a partir de uma conexão à Internet. Uma prateleira pode conter até seis células de carga conectadas a um só elemento de condicionamento de sinal e dispõem de um protocolo baseado em RS-485. Estas são ligadas entre si até uma caixa de conversão cujas funções são traduzir o protocolo RS-485 em RS-232 e fornecer alimentação às prateleiras conectadas ao barramento RS-485. Cada célula de carga consegue monitorizar um item, e a caixa de conversão dispõe de 4 barramentos RS-485 sendo possível conectar 20 prateleiras em cada um. No entanto, para conetar mais prateleiras ao barramento é necessário utilizar um repetidor de sinal. O barramento RS-232 permite fazer a conexão a um computador onde executa um *software* costumizável que controla as prateleiras numa configuração master-slave e disponibiliza a informação relacionada com os níveis de stock ao utilizador, seja através de um monitor ou numa aplicação baseada na *cloud*.

O sistema completo pode ser observado na figura 2.1. Na figura 2.1a pode-se observar a unidade conversora, dando destaque às portas RJ45 que fazem a ligação a outras unidades de balanças. Na figura 2.1b pode-se observar uma prateleira com seis células de carga.

(a) Unidade conversora. (b) Prateleira.

Figura 2.1: SmartShelf Weighing Pad.

### *ShelfX Smart Fridge*

A ShelfX foi estabelecida em 2011 e oferece um produto que permite que os clientes comprem qualquer item e qualquer quantidade diretamente na prateleira [4]. A tecnologia é baseada em placas de deteção de peso e pode ser instalado em qualquer frigorífico efetuando algumas alterações ao mesmo. Nomeadamente, é necessário instalar um sistema de bloqueio da porta; um sistema de pagamento e validação; e instalar as placas de deteção de peso nas prateleiras (figura 2.2). Analisando o produto, verifica-se que, embora cada prateleira possa conter várias placas de deteção de peso, cada placa apenas pode conter um tipo de produto.

Figura 2.2: ShelfX.

### *WiseShelf*

A WiseShelf é uma empresa que desenvolveu uma solução de controlo de stocks que pode ser implementada em qualquer prateleira de estante já existente [5]. A deteção de objetos é feita através de vários *Light dependent resistors* (LDRs) dispostos em séries ao longo da prateleira e envia os dados para a aplicação de gestão através de Wi-Fi, figura 2.3. Este produto permite o envio de alertas quando o nível de stock de um dado produto numa prateleira está baixo. Também permite a visualização da disposição dos produtos nas prateleiras.


Figura 2.3: Prateleira da *WiseShelf*.

### *Amazon Go*

A Amazon Go é um conceito de loja assistida por visão computacional, fusão sensorial e *deep learning* onde não existem caixas registadoras físicas, o cliente apenas tem de apresentar à entrada da loja um código *Quick response* (QR) gerado pela aplicação da *Amazon* (figura 2.4) [6]. O sistema faz uso de várias câmaras distribuídas pelo teto e de diversos sensores embutidos nas prateleiras, inclusive sensores de peso, de forma a monitorizar os movimentos de cada cliente. Com isto, o sistema é capaz de determinar quais os produtos que cada cliente retirou das prateleiras, adicionando-os automaticamente ao carrinho de compras virtual. Caso um cliente coloque de novo um produto na prateleira o sistema retira o produto automaticamente do carinho compras do cliente.

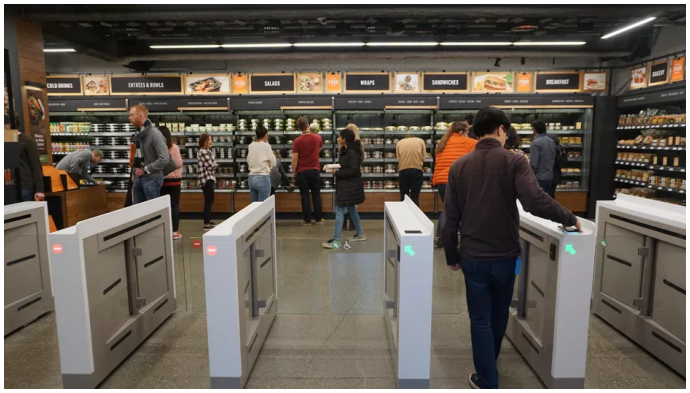


Figura 2.4: Amazon Go.

### Continente labs

O continente labs é o primeiro supermercado de marca europeia do género, situado em Lisboa (figura 2.5) [7]. É o resultado da parceria entre o Continente e a start up portuguesa sensei. O supermercado combina tecnologias capazes de identificar automaticamente os produtos que são retirados ou devolvidos às prateleiras lineares. O cliente tem de estar registado na aplicação do supermercado, ao entrar na loja apresenta um código QR gerado na aplicação que o identifica dentro da loja e ao seu carrinho virtual. O pagamento é feito de forma automática assim que

o cliente sai da loja, não existindo caixas de pagamento no interior. O supermercado é equipado com mais de 400 sensores de prateleiras e cerca de 230 câmaras no teto que acompanham o cliente e registam os produtos que são retirados das prateleiras, adicionando-os ao carrinho virtual de cada cliente.

Figura 2.5: Continente Labs.

### Comparação dos Produtos

Das cinco soluções apresentadas é possível encontrar semelhanças com o produto que se pretende desenvolver.

Analisando os produtos *SmartShelf* e *ShelfX* verifica-se que é possível usar células de carga para detetar quantidades de produtos existentes numa dada superfície e tornar esses dados acessíveis remotamente. Ademais, no caso da *SmartShelf* verifica-se que usando um barramento de comunicações série com suporte para comunicações multi ponto permite a conexão de prateleiras em cadeia e também a alimentação e comunicação a partir de um único cabo. Por fim, o produto *ShelfX* implementa o conceito de monitorização de stocks e venda de produtos numa aplicação em ambiente real do mundo do retalho. No entanto, apenas em ambientes controlados.

A solução *WiseShelf* levanta questões relacionadas com o comportamento do sistema quando os produtos a monitorizar são translúcidos. Não obstante, a solução permite que existam, numa só prateleira, diversos tipos de produtos que são monitorizados de forma independente. Como também possibilita a monitorização do sistema através de uma aplicação *web*.

Por fim a *Amazon Go* o Continente labs, com a combinação de vários elementos, como as células de carga e diversas câmaras permite determinar o tipo e a quantidade de produtos retirados de uma prateleira e associar um cliente a esse evento. No entanto, levanta questões acerca das políticas de privacidade, dado que todos os clientes que entram na loja estão a ser filmados e todos os seu movimentos são constantemente monitorizados.

O produto a desenvolver combina a solução de deteção de objetos da *SmartShelf*, através de sensores de peso, com o sistema de gestão de stocks de um supermercado baseado numa aplicação *web*, como a *Amazon Go* e o Continente labs.

## 2.2 Requisitos do Sistema

Os requisitos do sistema são os seguintes:

- Os dados fornecidos pelas prateleiras devem ser adquiridos e processados em tempo real.
- Os dados adquiridos pelos sensores devem permitir determinar a localização de eventos usando apenas dois sensores de carga.
- A aplicação deverá correr num servidor e ter acesso à *web*, fazendo uso de uma base de dados relacional.
- A aplicação deverá providenciar uma interface gráfica para monitorização, controlo e manutenção do sistema.
- O sistema deverá inferir as quantidades e tipos de produtos retirados ou colocados nas prateleiras.
- O sistema deverá gerar alarmes em situações de níveis baixos de stock ou em situações de erro.
- O sistema deve aprender e manter atualizada a informação de peso de referência de cada produto, já que não é passível de ser considerada a existência de informação fidedigna associada a pesos de artigos.

## 2.3 Arquitetura do Sistema Proposto

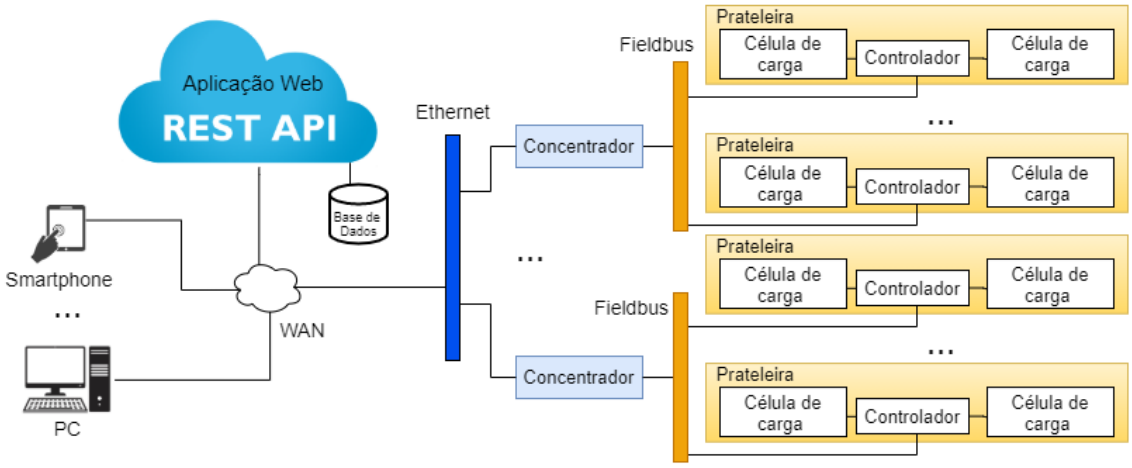



Figura 2.6: Arquitetura do sistema.

Tal como é possível observar na figura 2.6, esta representa a arquitetura do sistema, que vai de encontro à arquitetura já desenvolvida pela empresa. De seguida é apresentado uma descrição

sucinta de cada componente do sistema. Terminando com a identificação das diferenças em relação ao sistema implementado pela empresa.

### Arquitetura

A base do sistema é composta pelas prateleiras que são constituídas por duas células de carga e um microcontrolador. Estes dois elementos são explorados nas secções 2.4 e 2.6, respetivamente. O papel do microcontrolador é o de monitorizar as variações de peso significativas que ocorram na superfície da prateleira e comunicar ao concentrador o peso e a posição do centro de massa relativamente à variação que ocorreu.

O concentrador permite a conexão entre as prateleiras e o servidor. Este converte as mensagens provenientes das prateleiras em pedidos POST que são depois interpretados pela API. A ligação entre as prateleiras e o concentrador é feita através de um barramento série e a conexão ao servidor é feita por *internet*, sendo o barramento série melhor aprofundado na secção 2.8.

No servidor é executado uma API do estilo *Representational state transfer* (REST). Sendo assim, o concentrador e o serviço web usam métodos do protocolo de comunicação *Hypertext Transfer Protocol* (HTTP) para interagirem com o servidor. Como visto anteriormente, o concentrador usa o método POST para enviar mensagens sobre as alterações de estado que ocorram (contendo o peso, a posição e o id da prateleira). Estes temas são abordados na secção 2.9. Para além disso, há que ter em conta a possibilidade dos diferentes elementos do sistema se encontrarem em redes distintas. Uma vez que, as conexões ao servidor são feitas através de *Internet Protocol*.

Adicionalmente, uma aplicação *web* disponibiliza uma interface gráfica que permite um utilizador com credenciais válidas visualizar o conteúdo de uma prateleira, realizar reposições de stock e corrigir de erros de stock que possam a surgir.

O barramento permite estabelecer a comunicação entre várias prateleiras e o concentrador. Sendo que, utilizando um meio de comunicação por cabo baseado em RS485, juntamente com pares de fios entrelaçados, permite aumentar a robustez da comunicação - referente às interferências eletromagnéticas -. Sucessivamente, reduzir a complexidade e o custo do controlador de cada prateleira, tornando possível a utilização de apenas um cabo para a comunicação e alimentação. Por exemplo, um cabo *Unshielded/Unshielded twisted pair* (U/UTP) categoria 6A que providencia isolamento eletromagnético entre os vários pares de fios dentro dentro do próprio cabo [8].

### Comparação

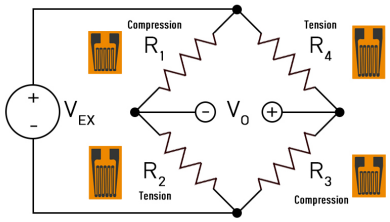
A empresa de momento dispõem de uma base já desenvolvida deste projeto. No entanto, foram identificadas algumas áreas passíveis de serem alteradas de forma a tornarem o sistema mais flexível e escalável. Na versão implementada pela empresa, existe um *master* na rede de comunicação das prateleiras, o concentrador, que questiona periodicamente as prateleiras pelo estado atual de peso em cada célula de carga, calculando depois a ocorrência de eventos em cada uma. As comunicações baseiam-se num barramento RS-485 com um protocolo de comunicação desenvolvido pelos mesmos, e a frequência de amostragem atual está limitada a 8 Hz.

A alteração proposta seria a de distribuir o processamento relativo à deteção da ocorrência de eventos por cada controlador das prateleiras. Isto permite que a amostragem às células de carga possa ser feita a uma frequência mais elevada e não limitada pelo número de elementos conectados ao barramento RS-485. Outra vantagem que a arquitetura proposta introduz em relação à já implementada é a de a rede de comunicação ser menos sobrecarregada pois cada controlador apenas tem a necessidade de enviar mensagens ao concentrador caso exista a deteção de eventos na prateleira.

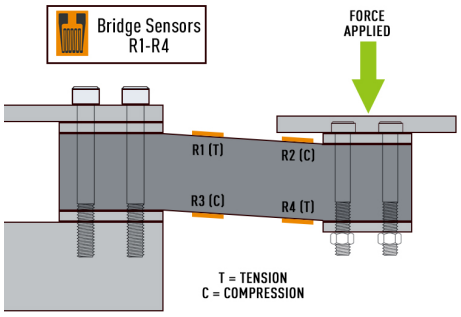
## 2.4 Células de Carga

### Princípio Físico

Uma célula de carga é um transdutor que converte a força aplicada, maioritariamente peso, num sinal elétrico. Embora existam várias soluções de células de carga, a mais comum é a célula de carga baseada em *strain gauges* [9], a resistência das mesmas varia com a força aplicada. Estas quando conectadas usando uma ponte de wheatstone numa configuração de quatro elementos, como a observada na figura 2.7a, permite aumentar a precisão das medições. As células de carga ficam embutidas numa estrutura que deforma, consoante a força aplicada. Nesta, normalmente, duas das células costumam ficar em tensão e as outras duas em compressão, figura 2.7b.




(a) Ponte de Wheatstone com 4 sensores de carga




(b) Posição das células de carga embutidas numa estrutura

Figura 2.7: Célula de carga.

Na figura 2.8 pode-se observar a variação das resistências numa ponte de wheatstone (numa configuração como na da figura 2.7b), como tal a variação da tensão de saída será proporcional à deformação aplicada à estrutura de suporte e é dada por:

$$V_{out} = \frac{\Delta R}{R} V_{in}$$

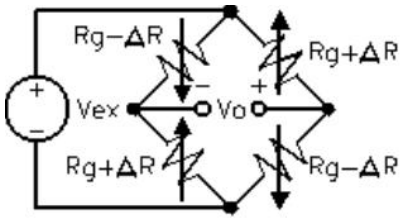



Figura 2.8: Variação das resistências numa configuração como a da figura 2.7b.

As células de carga disponibilizadas pela empresa são do tipo analógico, figura 2.9, estas requerem uma fase de condicionamento de sinal e outra de conversão de sinal analógico para digital antes dos dados poderem ser utilizados para processamento digital. Dado que, a sensibilidade das células de carga ronda os 2mV/V e com uma excitação de 5V apenas 10mV de escala completa estarão disponíveis. O sinal de saída deverá ser primeiramente amplificado por um amplificador de instrumentação com as seguintes características: elevada impedancia de entrada; elevado ganho diferencial; elevada razão de rejeição em modo comum e de fonte de alimentação; baixo *drift*; baixo *offset* e baixas correntes de polarização. A utilização de um filtro passa baixo com frequência de corte baixa permite remover as frequências indesejáveis introduzidas por fontes eletromagnéticas internas ou externas ao sistema. Ademais, a leitura de força exige uma frequência de amostragem baixa com baixo ruído e elevada resolução de quantificação. Para diminuir o ruído, uma média de quatro ou mais amostras deverá ser feita antes da informação poder ser usada. Resoluções superiores a 16 bits e frequência de amostragem desde 10 a 1000 amostras/segundo são adequadas [10].

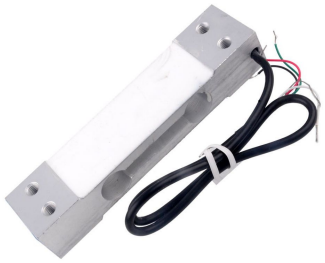


Figura 2.9: Célula de carga.

### Conversor Analógico/Digital (ADC)

A solução proposta para digitalização do sinal é o integrado HX711 [11], da empresa AVIA semicondutor, que é um conversor analógico-digital de 24 bits desenhado para fazer interface com células de carga. O integrado incorpora um amplificador de ganho programável de baixo ruído com duas frequências de amostragem seleccionáveis, 10 amostras/segundo e 80 amostras/-segundo. Para além disso, possui filtros de rejeição comum com frequências de 50Hz e 60Hz. Este conversor cumpre com as especificações já vistas. As características do sensor podem ser analisadas no extrato obtido da *datasheet* do componente presente na figura 2.10. A figura 2.11

mostra o integrado montado com os componentes externos necessários, como o filtro passa baixo para filtragem do sinal e condensadores de desacoplamento da fonte de alimentação para uma alimentação estável da ponte de *wheatstone*.

| Parameter | Notes | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|
| Full scale differential input range | V(inp)-V(inn) | | ±0.5(AVDD/GAIN) | | V |
| Common mode input | | AGND+1.2 | | AVDD-1.3 | V |
| Output data rate | Internal Oscillator, RATE = 0 | | 10 | | Hz |
| | Internal Oscillator, RATE = DVDD | | 80 | | |
| | Crystal or external clock, RATE = 0 | | $f_{clk}$/1,105,920 | | |
| | Crystal or external clock, RATE = DVDD | | $f_{clk}$/138,240 | | |
| Output data coding | 2's complement | 800000 | | 7FFFFF | HEX |
| Output settling time (1) | RATE = 0 | | 400 | | ms |
| | RATE = DVDD | | 50 | | |
| Input offset drift | Gain = 128 | | 0.2 | | mV |
| | Gain = 64 | | 0.4 | | |
| Input noise | Gain = 128， RATE = 0 | | 50 | | nV(rms) |
| | Gain = 128， RATE = DVDD | | 90 | | |
| Temperature drift | Input offset（Gain = 128） | | ±6 | | nV/℃ |
| | Gain（Gain = 128） | | ±5 | | ppm/℃ |
| Input common mode rejection | Gain = 128， RATE = 0 | | 100 | | dB |
| Power supply rejection | Gain = 128， RATE = 0 | | 100 | | dB |
| Reference bypass（$V_{BG}$） | | | 1.25 | | V |
| Crystal or external clock frequency | | 1 | 11.0592 | 20 | MHz |
| Power supply voltage | DVDD | 2.6 | | 5.5 | V |
| | AVDD，VSUP | 2.6 | | 5.5 | |
| Analog supply current (including regulator) | Normal | | 1400 | | μA |
| | Power down | | 0.3 | | |
| Digital supply current | Normal | | 100 | | μA |
| | Power down | | 0.2 | | |

Figura 2.10: Características elétricas do HX711.

O HX711 disponibiliza as medições a uma cadencia fixa, mas selecionável, e com ganho programável através de uma interface série. A frequência a que os dados são amostrados é definida através do nível lógico presente no pino RATE (*Output data rate control input*). Se o nível lógico corresponder a '0' então a frequência de amostragem é de 10 Hz e se for '1' a frequência de amostragem é de 80 Hz. A comunicação série é feita através dos pinos PD_SCK (*Power down control and serial clock input*) e DOUT (*Serial data output*). Quando o HX711 apresenta um valor disponível ocorre uma transição de '1' para '0' no pino DOUT, que após isto o microcontrolador pode aplicar 24 impulsos de relógio no pino PD_SCK de forma a obter o valor medido pelo HX711 no pino DOUT. De seguida, é necessário aplicar mais 1 a 3 impulsos de forma a definir o canal e o ganho da próxima conversão. Sendo que, 1 impulso corresponde ao canal A com ganho 128; 2

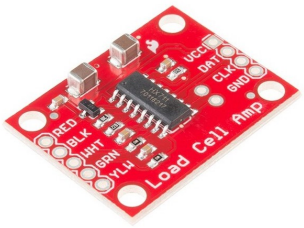



Figura 2.11: Módulo baseado no integrado HX711.

impulsos corresponde ao canal B com ganho 32; por fim, 3 impulsos corresponde a uma medição feita no canal A com ganho 64. A partir do 25 impulso de relógio o sinal do pino PD_SCK volta a '1' até à próxima medição estar pronta. Mantendo o sinal do pino PD_SCK a '1' por mais do que 60$\mu s$ o HX711 entra em modo de *power down* e só volta ao estado normal quando o sinal aplicado em PD_SCK voltar a '0'. Contudo, após o reinicio, por defeito é selecionado o canal A, com ganho 128. Na figura 2.12 pode-se observar o funcionamento do protocolo de comunicação série do HX711.

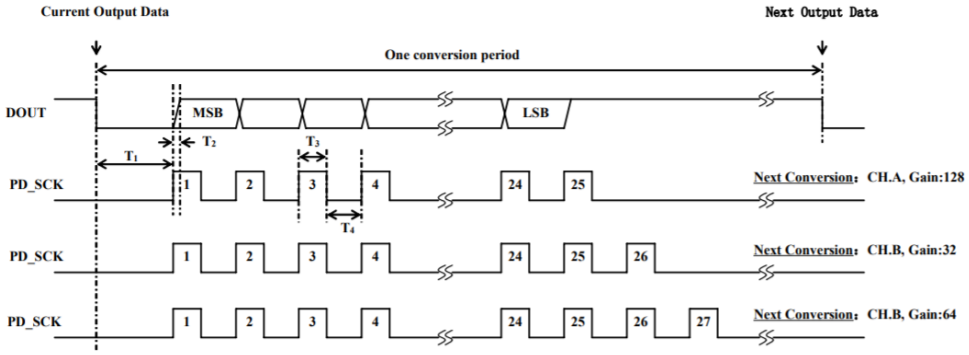



Figura 2.12: Interface de comunicação série do HX711.

### Calibração das células de carga

Uma vez que, a variação do sinal de saída de uma célula de carga é proporcional ao peso que lhe é aplicada quando estas são alimentadas o estado em que se encontram é desconhecido. Como tal é necessário tomar o valor que é apresentado, quando estável, como referencia de tara. Às medições que forem feitas posteriormente devem lhes ser subtraídas este valor de forma a obter o valor real do peso aplicado.

Após a determinação da referência é possível obter valores que representam o peso de um objeto. Contudo, é ainda necessário multiplicar este valor por um fator de correção. Este fator de correção é dependente da célula de carga utilizada, logo tem de ser determinado para todas as células de forma individual. Mais uma vez, dado que a variação do sinal é proporcional ao peso então o fator de multiplicação pode ser calculado a partir de uma equação linear $y = kx + b$, sendo

y o peso em gramas, k o fator de correção, x o valor lido nas células de carga e b a ordenada na origem. No entanto, como já foi determinado um valor de referencia para o zero relativo á ordenada na origem é zero, simplificando a equação a $y = kx$. O passo final do cálculo do fator de correção envolve colocar um objeto de peso conhecido em cima da célula de carga, fazendo coincidir o centro de massa do objeto com o centro da célula, obtendo assim o valor de saída da célula de carga para uma massa de valor conhecida. Sendo assim, o fator de correção é dado por $k = y/x$.

Como esta aplicação exige duas células de carga a suportar a superfície da prateleira, a determinação da massa de um objeto é dada pela equação $y = k_1x_1 + k_2x_2$, e os fatores de calibração são calculados nas mesmas condições do caso anterior.

### Princípio de determinação do centro de massa numa superfície

Quando duas células de carga são utilizadas, como na figura 2.13, o peso total de um objeto, colocado na superfície de medição, é distribuído por essas. Desta forma, o peso do objeto pode ser calculado pela soma dos pesos individuais de cada célula de carga [12]:

$$F_x = F_1 + F_2 \tag{2.1}$$

Considerando ainda a figura 2.13 e tomando como base a primeira lei de equilíbrio estático, o cálculo do centro de massa numa superfície pode ser determinado pela seguinte equação:

$$x = \frac{F_2}{F_x} d \tag{2.2}$$

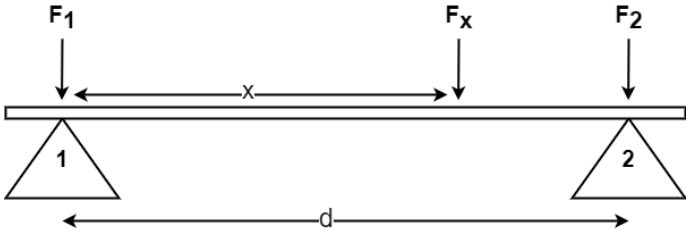



Figura 2.13: Forças exercidas numa superfície apoiada por duas células de carga.

### Processamento do sinal das células de carga

Dada a sensibilidade do ADC usado e a existência de possíveis interferências que possam afetar a leitura do sinal, este apresenta um ruído elevado. A solução passa por implementar um filtro passa baixo de forma a atenuar o ruído do sinal. Como este processamento terá de ser feito em tempo real por um microcontrolador com poucos recursos é necessário ter em consideração a complexidade do mesmo. Uma forma de suavizar o sinal passa por utilizar um filtro de média deslizante de tamanho N em que todas as amostras têm peso igual, que é $1/N$, [13]. A média deslizante é uma média que é aplicada aos últimos N valores medidos e que pode ser implementada num microcontrolador com poucos recursos resumindo-se a 3 operações aritméticas - uma

subtracção, uma adição e uma divisão - quando fazendo uso de um *buffer* circular onde ficam guardados os últimos N valores da janela deslizante. Contudo, a média deslizante simétrica aplica um atraso no sinal de $(N-1)/2$ amostras.

O tempo de resposta de uma célula de carga a uma variação de peso ronda os 800 ms e a forma de onda que pode ser observada na figura 2.14, esta forma de onda foi obtida a partir de um protótipo usado para testes. Atendendo ao teorema de *Nyquist* a frequência de amostragem deverá ser superior a duas vezes a frequência do sinal, $f_{amostragem} > 2 * f_{sinal}$. Sendo assim, uma frequência de amostragem de 10 Hz, a frequência mínima do HX711, permite detetar variações no sinal sem a ocorrência de distorção. Contudo, ao fazer uso de uma frequência de amostragem mais elevada é possível suprimir o ruído presente no sinal.

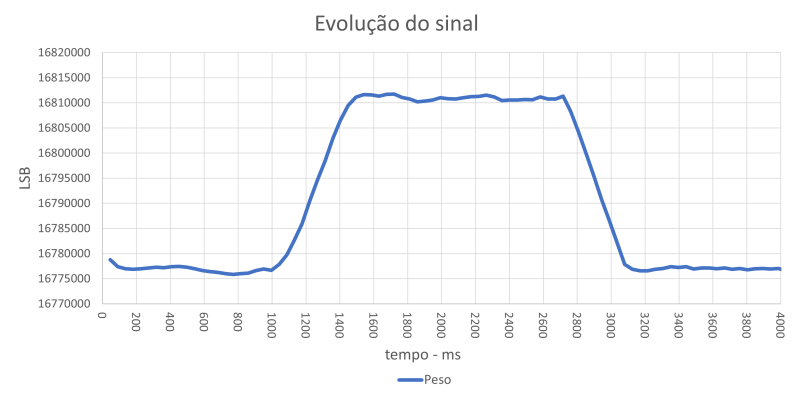



Figura 2.14: Evolução do sinal de uma célula de carga quando um objeto é colocado e retirado de cima de uma célula de carga.

Através da figura 2.14 verifica-se que a resposta da célula de carga aos eventos de colocar e retirar um objeto da superfície assemelha-se a uma onda quadrada. Desta forma, é possível verificar a ocorrência de eventos analisando mudanças bruscas no sinal.

## 2.5 *Smart Sensor*

Um sensor inteligente permite combinar os elementos de deteção, os sensores, o processamento de informação e tecnologia de comunicação de forma a equipar os sensores com funcionalidades mais avançadas do que apenas a deteção de dados brutos, [14].

Algumas das características que levam à criação de redes de sensores inteligentes são: redução de custos - é esperado que o serviço, operação e manutenção seja simplificado e a cablagem reduzida; monitorização remota - uma melhor gestão dos dispositivos e manutenção preventiva é alcançada através da monitorização remota; modularidade - a conectividade dos módulos é feita através de *software*; flexibilidade - a capacidade de realizar mudanças rápidas nos sensores possibilita a adaptação a diferentes aplicações e/ou a transmissão de informações de locais difíceis ou inacessíveis; medições mais precisas e codificação a taxas de amostragem mais elevadas de forma mais eficiente. Esta característica permite adicionar mais capacidades ao sensor inteligente, como

por exemplo diagnóstico remoto e ou auto diagnóstico; calibração remota; afinação dos parâmetros do sensor de forma remota; entre outras.

Do ponto de vista da arquitetura de um sensor inteligente, que pode ser observada na figura 2.15, identifica-se sete elementos importantes, [14]:

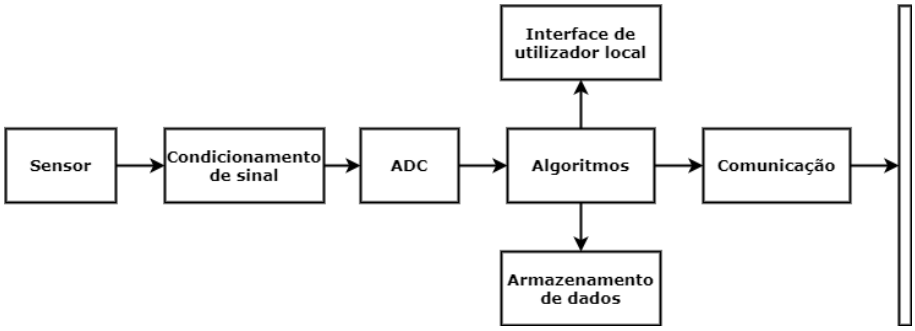



Figura 2.15: Arquitetura de um sensor inteligente.

- **Sensor** - o dispositivo que transforma a energia de um domínio para o domínio elétrico, transdutor.
- **Condicionamento de sinal** - o circuito que prepara o sinal antes da conversão para o domínio digital.
- **Conversor de analógico para digital** - dispositivo que converte o sinal de entrada analógico num código digital que representa a magnitude do sinal analógico. É importante que este se situe o mais próximo possível do ponto de medição.
- **Algoritmos** - são funções que incluem a conversão de dados para as unidades especificadas pelo utilizador, processamento de sinal, análise de dados e de redução, verificação de condições de alarme, entre outros. São estes algoritmos que fornecem a característica de inteligência aos sensores.
- **Armazenamento de dados** - armazenamento de dados relacionados com a identificação do sensor, configuração e de calibração.
- **Interface de utilizador** - apresentação correta de dados de forma previamente estabelecida.
- **Comunicação** - interface a um meio de comunicação para acesso remoto de forma a realizar várias acções nomeadamente: configurar do sensor; fazer calibração; fazer diagnósticos; capturar dados; entre outros.

## 2.6 Controlador da prateleira

### Microcontrolador

O microcontrolador que será utilizado como controlador das prateleiras será um Atmega328p, com frequência de relógio de 16 MHz, usando a plataforma Arduino, figura 2.16[15]. Este é um microcontrolador com arquitetura AVR de 8 bits e conta com: uma interface *Universal Asynchronous Receiver-Transmitter* (UART); uma interface *Inter-Integrated Circuit* (I2C); uma interface *Serial Peripheral Interface* (SPI); 2 *timers* de 8 bits e 1 *timer* de 16 bits. Apresentando múltiplos *Interrupt Service Routine* (ISR) que são desencadeados por diversos eventos relacionados com: interfaces de comunicação; alterações nos estados dos pinos digitais; *timers*; e outros que não são considerados de grande relevância tendo em conta esta aplicação.

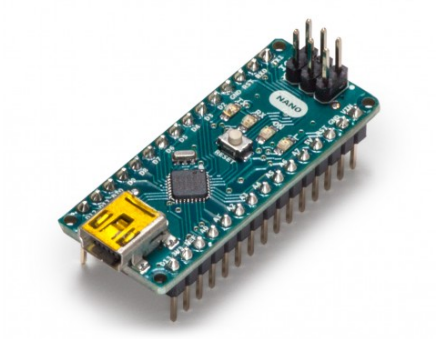


Figura 2.16: Atmega328p - Arduino nano.

### Escalonamento de Tarefas

O escalonamento de tarefas permite obter uma ordem pela qual as tarefas são executadas no processador, fazendo uma correspondência entre cada instante de tempo e a tarefa que está a executar. Um escalonamento diz-se fazível se todas as restrições impostas às tarefas são cumpridas, por exemplo as temporais. Com isso é feito o levantamento das tarefas executas pelo processador e determinadas as suas restrições.

Uma análise preliminar permite identificar 4 rotinas principais executadas pelo controlador da prateleira, sendo elas por ordem de prioridade: T1) leitura do ADC; T2) comunicações; T3) determinação e identificação de eventos e T4) controlo interno. As tarefas de prioridade mais baixa podem ser escalonadas utilizando o método de *Round Robin*, método de escalonamento em que as tarefas executam umas após as outras. No entanto, as tarefas de prioridade mais alta terão de interromper as tarefas de prioridade mais baixa e isso é possível através do uso de ISR. No caso das comunicações, como por exemplo fazer uso da ISR de dados recebidos na interface UART do microcontrolador para copiar os dados recebidos para um *buffer* interno que será analisado posteriormente por uma tarefa de menor prioridade, a tarefa T4. No caso da tarefa T1 o *trigger* da tarefa seria um *falling edge* no pino do microcontrolador que estivesse conectado ao pino DOUT do HX711, visto que, este transita do valor lógico ’1’ para o ’0’ quando os dados se encontram disponíveis, como discutido na secção 2.4.

## 2.7 Concentrador

O concentrador é o elemento que fará a ligação entre múltiplas prateleiras e a API, atuando como *gateway* entre o barramento RS485 e a rede IP. Como tal, este necessita de ter uma interface *Ethernet* ou WiFi e outra interface para com o barramento das prateleiras. Dado que, o concentrador terá de traduzir mensagens enviadas pelas prateleiras em pedidos HTTP pode-se tornar num *bottleneck* da rede. Sendo assim, é necessário que este apresente um elevado poder de processamento e ao mesmo tempo baixo custo de operação e manutenção.

### ESP32

Uma opção passa por usar o microcontrolador da empresa ESPRESSIF SYSTEMS, o ESP32 (figura 2.17). Este microcontrolador apresenta uma frequência de relógio de 240 MHz, até 16 MB de memória de programa e 520 KB de memória *Random-access memory* (RAM) com possibilidade de expansão para 4 MB, [16]. Ademais este microcontrolador está equipado com dois processadores lógicos tornando possível a divisão da carga de processamento por ambos. Como este não possui uma interface *Ethernet* é necessário utilizar um módulo adicional, ou se a aplicação assim o permitir utilizar o módulo WiFi embutido.

Figura 2.17: ESP32 .

### Raspberry Pi 3

Outra opção passaria pela utilização de um *System On a Chip* (SOC) da RaspberryPi Foundation, como por exemplo um Raspberry Pi 3 (figura 2.18)[17]. Este permite o desenvolvimento de programas baseados em *threads*, sendo a gestão das mesmas feitas pelo sistema operativo. O sistema operativo é uma distribuição Linux baseado em Debian, que pode requer ou não manutenção aquando da disponibilidade de novas atualizações.

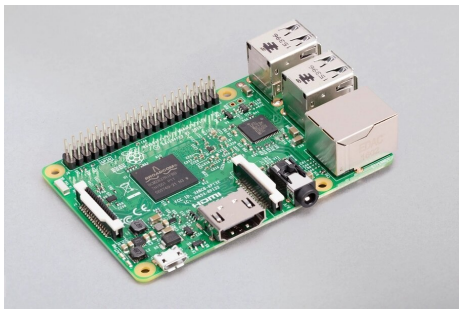


Figura 2.18: Raspberry Pi 3 .

## 2.8 Comunicações

### RS-485

O RS-485 é uma norma que especifica as características elétricas dos transmissores e receptores para uso em comunicações série e que permite comunicação bidireccional em modo *half-duplex* com conexões multi-ponto, a uma velocidade máxima de 1 Mbps. Esta norma especifica o uso de sinais diferenciais, o que torna a comunicação robusta e imune a interferências. Esta norma especifica que o número máximo de dispositivos conectados a um barramento RS-485 é 32. No entanto, há a possibilidade de conectar mais dispositivos, se os respetivos receptores apresentarem impedâncias de entrada mais elevadas. A interface entre um barramento destes e um microcontrolador pode ser feita através do periférico UART, presente na maioria dos microcontroladores [18]. Uma vez que, os microcontroladores não possuem uma interface RS-485 é necessário utilizar um periférico externo como o MAX485 da Maxim Integrate. Na figura 2.19 pode-se observar este componente implementado num módulo *ready to use*.

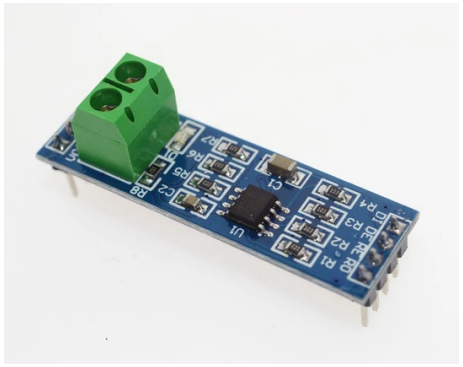


Figura 2.19: MAX485 module.

Existem vários protocolos desenvolvidos para comunicação entre vários aparelhos, como por exemplo o MODBUS. No entanto, apresenta um *overhead* muito grande. Como tal, é possível desenvolver protocolos simples - sendo o caso do protocolo que a empresa implementa.

### Controller Area Network (CAN)

O barramento CAN foi desenvolvido pela BOSCH como um sistema de transmissão de mensagens que especifica uma taxa de transmissão máxima de 1 Mbps até uma distância de 40 metros. Ao contrário de outros barramentos, CAN é baseado no envio de mensagens curtas, até 8 bytes de dados, por exemplo medições de sensores, que são enviadas para todos os dispositivos conectados à rede, [19]. CAN foi originalmente desenvolvido para a industria automóvel como forma de diminuir ligações complexas entre diferentes componentes por um barramento de dois fios. É especificado que CAN apresenta grande resiliência a interferências eletromagnéticas, habilidade de auto diagnóstico e recuperação de erros nos dados. Estas características levaram a outras industrias a adoptar o uso de CAN.

O protocolo de comunicação CAN, ISO-11898:2003, descreve como a informação é transmitida entre dispositivos na rede e este está de acordo com a especificação o modelo OSI - *Open Systems Interconnection* - da forma em que é definido em camadas. A arquitetura ISO-11898 especifica o formato das mensagens e as características elétricas dos transceptores, conectores e cabos.

O protocolo de comunicação CAN implementa *The Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority* (CSMA/CD+AMP) - com deteção de colisão e arbitragem baseada em prioridades das mensagens. CSMA significa que cada dispositivo deve esperar por um determinado período de inatividade antes de enviar uma mensagem. CD+AMP significa que as colisões são resolvidas por meio de uma arbitragem bit a bit com base na prioridade pré-programada em cada mensagem. Existem dois tipos de identificadores de mensagens, identificadores de 11 bits que permitem 2048 mensagens únicas e outro de 29 bits que permite que existam 537 milhões de mensagens únicas.


Figura 2.20: Mensagem CAN com identificador de 11 bits.

Uma mensagem CAN, com identificador de 11 bits, pode ser observada na figura 2.20. Os campos mais importantes são o identificar de 11 bits que identifica unicamente cada mensagem na rede CAN. O *Remote transmission request* (RTR) que é um bit que indica que uma dada informação é requerida por um dispositivo e dado que as mensagens são recebidas por todos os dispositivos garante que existe uniformidade nos dados utilizados pelo sistema todo. O campo *Data length code* (DLC) é representado por 4 bits e indica quantos bytes de informação a mensagem contém, até um máximo de 8 bytes. A mensagem CAN também incluí um *cyclic redundancy check* (CRC) de 16 bits da informação presente na mensagem para deteção de erros. O campo *Acknowledgement* (ACK) permite detetar se todos os dispositivo receberam a mensagem sem erros, e caso algum dispositivo tenha encontrado algum erro a mensagem é retransmitida.

### Comparação

Comparando estas duas soluções de comunicação verifica-se que tanto CAN como RS-485 apresentam longas distâncias de comunicação, imunidade a interferências eletromagnéticas e permitem comunicações multi ponto. No entanto, o protocolo CAN permite que vários dispositivos tentem aceder ao barramento sem que haja perda de mensagens, algo que usando RS-485 teria de ser implementado por software e com recurso a componentes externos.

## 2.9 *Web Server*

Nesta secção são expostas as tecnologias que permitem desenvolver serviços *web* como o HTTP, API e o estilo REST.

### HTTP

HTTP é um protocolo de comunicação da camada de aplicação que tem como base o modelo cliente-servidor. Neste, o cliente abre uma conexão TCP (*Transmission Control Protocol*) para fazer um pedido e esperar por uma resposta. O protocolo HTTP é *stateless* o que significa que o servidor não guarda qualquer dado entre pedidos sucessivos [20]. Na figura 2.21 pode-se observar um pedido HTTP e na imagem 2.22 uma resposta HTTP [21].

Uma mensagem de pedido HTTP é composta da seguinte forma:

- **método** - define a operação que o cliente quer realizar, que podem ser GET, POST, OPTIONS ou HEAD. Tipicamente o método GET é usado para obter um recurso do servidor, e o método POST é usado quando o cliente quer enviar valores para o servidor.
- **Endereço** - é o endereço do recurso a que o pedido HTTP se refere.
- **Versão** - a versão do protocolo HTTP usado.
- **Cabeçalho** - comporta informações adicionais para o servidor.
- **Corpo** - é opcional e está separado por uma alinha em branco do cabeçalho da mensagem. Este incluí o recurso que é enviado ao servidor.

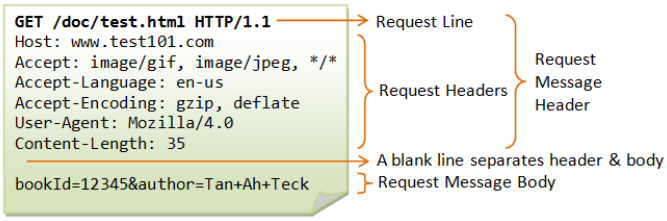



Figura 2.21: Pedido HTTP .

A divisão de uma resposta HTTP é a seguinte:

- **Versão** -a versão do protocolo HTTP usado.

- **Código de estado** - indica se o pedido teve sucesso ou não, e o porquê.
- **Mensagem de estado** - breve descrição do código de estado.
- **Cabeçalho** - comporta informações adicionais para o servidor.
- **Corpo** - opcional, e está separado por um alinha em branco do cabeçalho da mensagem. Este incluí o recurso que é pedido ao servidor.

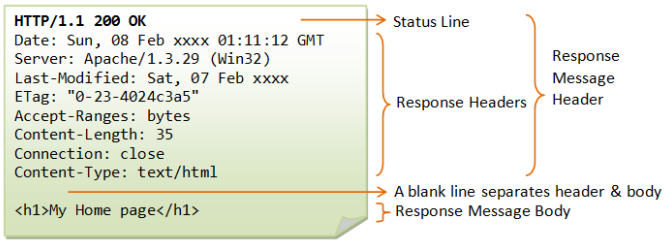



Figura 2.22: Resposta HTTP.

### API

API permite divulgar as funcionalidades e informação de um dado serviço a outras aplicações através de uma interface documentada. Assim, uma API é um conjunto de regras definidas que explicam como os computadores ou aplicações comunicam entre si. Uma API fica entre a aplicação e o servidor web, agindo como uma camada intermédia que processa a transferência de dados entre sistemas [22].

Maneira de funcionamento de uma API:

1. Uma aplicação cliente invoca uma chamada à API para obter informações, realiza um pedido. Esse pedido é processado desde a aplicação até ao servidor *web* a partir do *Uniform Resource Identifier* (URI) da API incluindo um verbo de pedido, cabeçalhos e, por vezes, um corpo de pedido.
2. Depois de receber um pedido válido a API faz uma chamada para o programa externo ou servidor *web*.
3. O servidor envia a resposta à API com as informações solicitadas.
4. A API transfere os dados para a aplicação que a requereu inicialmente.

Embora a transferência de dados seja diferente, dependendo do serviço *web* utilizado, o serviço de pedidos e respostas acontece por meio de uma API. As APIs oferecem segurança por *design*, dado que, agem como intermediários e facilitam a abstração de funcionalidades entre os dois sistemas. As chamadas às APIs geralmente incluem credenciais de autorização para reduzir os riscos de ataques ao servidor.

### REST

O estilo REST utilizado em arquitetura de software define um conjunto de normas que devem ser usadas para desenvolver serviços *web*. Existem 6 restrições que se aplicam quando desenvolvendo um serviço *web* [23] que são:

- Interface Uniforme - Os recurso devem ser unicamente identificados através de um URL e apenas através do uso de métodos especificados no protocolo de rede.

- Cliente-servidor - Deve haver uma distinção clara entre cliente e servidor. Por exemplo, a parte de interface gráfica fica encarregue do cliente e a parte de manipulação e armazenamento de dados fica a encargo do servidor. Isto permite que os componentes evoluam de forma independente tornando o sistema escalável.

- *Stateless* - A comunicação deve ser por natureza sem estado em que cada pedido do cliente ao servidor deve conter toda a informação necessária para o interpretar. Fazendo com que o cliente não possa tirar proveito de qualquer informação armazenada no servidor. Se for necessário manter um estado da sessão esta deve ser mantido pelo cliente.

- Cache - A restrição de *cache* exige que os dados dentro de uma resposta a um pedido sejam implicitamente ou explicitamente marcados como armazenáveis, ou não, em *cache*. Se uma resposta for armazenada em *cache* então o cliente tem o direito de usar a informação da resposta para pedidos similares futuros.

- Sistema por camadas - O sistema tem de estar dividido em camadas hierárquicas. O que restringe o comportamento do componente, de forma a que cada componente não possa ver além da camada com que está a interagir. Esta restrição limita a complexidade geral do sistema e promove a independência do substrato.

- Código sob Pedido - O REST permite que as funcionalidades do cliente sejam expansíveis ao fazer *download* e execução de código sob a forma de mini-aplicativos ou scripts. Permite que o cliente seja mais simplificado pois reduz o número de funcionalidades pré-implementadas. No entanto, isto reduz a visibilidade do sistema e, por isso, é uma restrição opcional.

Quando todas estas restrições são implementadas o sistema ganha propriedades não funcionais desejáveis, tais como: desempenho; escalabilidade; simplicidade; modificabilidade, visibilidade, portabilidade e confiabilidade. O estilo REST é uma arquitetura que utiliza padrões como o HTTP ou o URI.

### Servidor *Web*

Os componentes *docker-php-nginx-postgres-composer* juntos compõem um serviço *web*. Cada um destes componentes contribuí com recursos essenciais para a pilha:

- **Docker** - Esta é uma tecnologia de *containerization* de código aberto para compilar e correr aplicações de forma isolada e independente do seu ambiente, *coss-platform*. Isto é possível pois cada aplicação executa dentro de uma unidade de *software* padronizada denominada por *container*.
- **NGINX** - É o servidor *web*. O servidor *web* NGINX é um servidor web de código aberto. Este processa pedidos e distribui aplicativos através de pedidos HTTP de forma a que o serviço seja acessível a qualquer utilizador por meio de um URL.
- **postgres** - É a base de dados. Postgres é um sistema de gestão de base de dados relacionais de código aberto. Os dados são guardados e acessíveis de forma fácil através de *querys* SQL.
- **PHP** - É a linguagem de programação. A linguagem de programação PHP é orientada a *scripts* e em conjunto com o NGINX permite criar páginas *web* dinâmicas. Um dos processos que é possível fazer é o de interagir com a base de dados, algo que não é possível usando apenas HTML.
- **Compose** - O Compose é uma ferramenta para definir e correr múltiplos serviços *docker*. Cada serviço executa de forma isolada podendo interagir entre si quando necessário. A configuração do *docker* compose é feita através de um ficheiro YAML.

### Slim Framework

Slim é uma pequena *framework* PHP que permite desenvolver serviços *web* e API de forma simples e rápida. A *framework* permite fazer o mapeamento entre chamadas de retorno métodos específicos de pedidos HTTP e URIs. Suporta também a correspondência de parâmetros e padrões [24].

### Postman

Postman é uma plataforma que permite testar as API desenvolvidas de forma rápida e cómoda fornecendo diversas ferramentas para este fim. Nomeadamente, permite realizar pedidos HTTP com diferentes métodos de forma rápida [25].

## 2.10 Lógica Difusa

A lógica difusa é uma ferramenta utilizada para representar o conhecimento humano de uma forma fácil de ser manipulada por computadores. Desta forma, também é possível lidar com incerteza nas variáveis de entrada consideradas [26].

# Capítulo 3

# Sistema Inteligente de Inferência de Eventos

Neste capítulo é feita uma descrição do *hardware* utilizado para o desenvolvimento do projeto. De seguida, é explorada a melhor abordagem de adquirir o sinal das células de carga e o processamento digital aplicado ao mesmo.

Seguidamente, é explorada a forma de realizar a deteção e caracterização de movimentos na superfície da prateleira. Por fim, é discutida a arquitetura do controlador e do concentrador desenvolvido baseado em *software*.

## 3.1 Hardware

**Células de carga**

As células de carga de referência utilizadas para desenvolvimento da prateleira foram duas células de carga ZEMIC L6D class C3 de capacidade máxima 30 quilogramas. Resultando numa capacidade máxima combinada de 60 quilogramas.

**Estrutura**

A estrutura metálica utilizada, figura 3.1, representa uma prateleira de supermercado com 60 cm de comprimento. A superfície da prateleira é suportada por células de carga colocadas em ambas as extremidades da prateleira. Os suportes da mesma foram desenhados de forma a serem compatíveis com as prateleiras comerciais, permitindo uma fácil instalação nos suportes existentes.

Figura 3.1: Estrutura metálica utilizada para testes.

**ADC - HX711**

São utilizados, para análise, quatro HX711, sendo que dois deles estão munidos de proteção contra ruído eletromagnético e os outros dois não - figura 3.2.

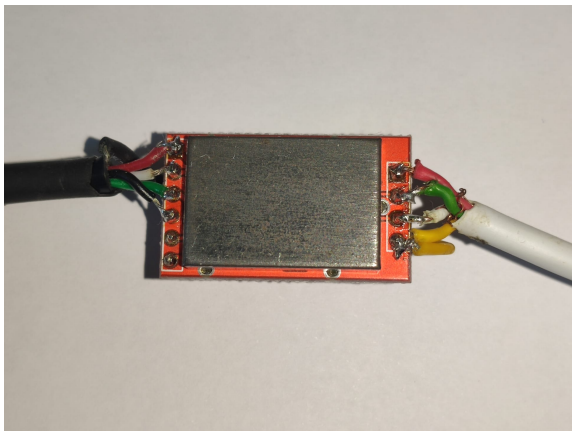


Figura 3.2: HX711 com escudo eletromagnético.

**Controlador**

O controlador é composto por um microcontrolador atmega328p e um integrado de comunicação série RS485, o MAX485, figura 3.3. A placa é alimentada a 12 volts e incluí reguladores lineares de tensão para realizar a alimentação dos ADC e células de carga de forma independente do circuito de controlo, reduzindo o impacto de interferências causadas por variações de carga no circuito de controlo. A placa de circuito integrado possuí duas interfaces para conectar dois HX711 e também duas interfaces RJ45 para alimentação e comunicação.

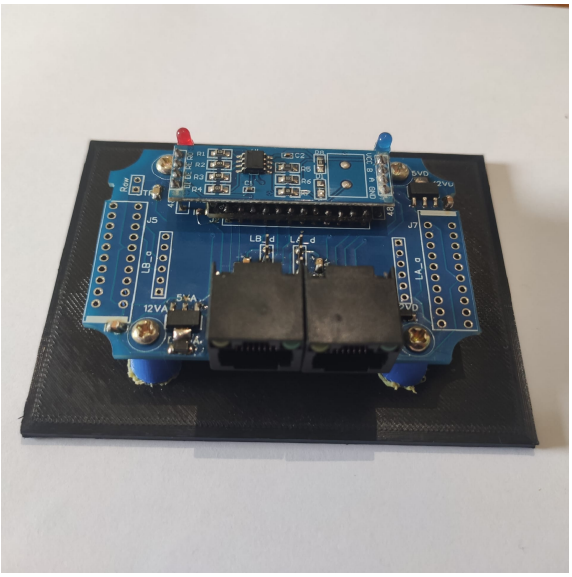


Figura 3.3: Controlador.

## 3.2 Processamento do sinal

**Parâmetros obtidos a partir das células de carga**

A configuração das células de carga utilizada permite obter os seguintes parâmetros:

- Peso: peso individual de cada célula de carga e peso total colocado no topo da prateleira;
- Posição: localização, segundo o eixo longitudinal da prateleira, do centro de massa dos objetos colocados no topo da prateleira;
- Interação: Evolução temporal do peso durante a ocorrência de movimentos.

**Aquisição do sinal**

As células de carga possuem, como referencia de sensibilidade 2mV/V. Sendo que neste caso dada a excitação de cada célula de carga ser de 5 volts, a variação do sinal de saída ocorre entre 0 e 10 mV, com sensibilidade de 0.33$\mu$V/grama. O erro combinado da célula de carga - $\leq 0.002\%$ da sensibilidade da célula de carga - é de, no máximo, 0.002 mV, correspondendo a 6 gramas.

Dado que, o IC utilizado é desenvolvido especificamente para fazer interface com células de carga, possuí de forma integrada um regulador de tensão para alimentar a ponte de *wheatstone* do sensor de carga e o ADC. De forma a minimizar o ruído eletromagnético captado pelo ADC o IC HX711 é equipado com um escudo que permite reduzir o ruído presente na aquisição de leituras em aproximadamente 10 vezes. Posteriormente será feita a comparação entre o HX711 com e sem escudo eletromagnético.

O HX711 possui 3 ganhos do sinal de entrada divididos por dois canais de entrada. O canal A possuí ganho de 64 e de 128 e o canal B possui apenas um ganho de 32. Referindo-se apenas ao canal A do integrado, na tabela 3.1 podemos observar o limite de entrada do HX711 para os

diferentes ganhos tal como a resolução referente a cada ganho. Verifica-se que a sensibilidade do ADC é maximizada utilizando o ganho do canal A é de 128, sendo de 2.38$nV$ por divisão.

| Ganho | Limite de tensão | Resolução (gramas/ divisão) |
|---|---|---|
| 64 | $\pm 40mV$ | 0.0143 |
| 128 | $\pm 20mV$ | 0.0072 |

Tabela 3.1: Resolução de leitura de peso com excitação da célula de carga a 5V para diferentes ganhos do canal A do HX711.

O integrado HX711 possuí uma interface de comunicação série de fácil integração com um microcontrolador, sendo que a leitura dos novos valores adquiridos e a configuração para a próxima leitura são feitos no mesmo acesso. A sequência de leitura do ADC encontra-se na figura 3.4.

A taxa de amostragem do IC, quando este não tem um cristal externo ligado é de 10 amostras/segundo ou 80 amostras/segundo. Quando é conectado um cristal externo a frequência de amostragem é imposta pelo cristal. Sendo que para obter uma frequência de exatamente 10 amostras/segundo o cristal tem de ter uma frequência de 11.0592 MHz. A taxa de amostragem dos HX711 utilizados é de aproximadamente 11 Hz.

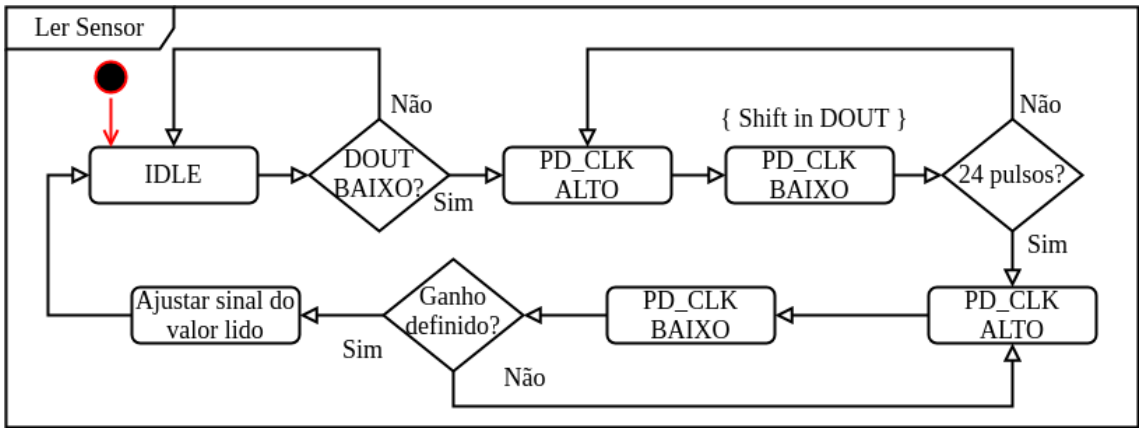



Figura 3.4: Sequência de leitura do HX711.

**Condicionamento digital do sinal**

O condicionamento digital do sinal é feito através de um filtro de média deslizante de tamanho fixo, N. Este filtro tem as características de um filtro passa baixo. Reduz o efeito de ruído aleatório, no entanto introduz um atraso a mudanças bruscas do sinal. É possível aplicar este filtro de forma a não utilizar muitos recursos do CPU em troca de utilização de memória RAM.

É utilizado um *buffer* circular para guardar as últimas N amostras do sinal em memória. Desta forma, o valor médio é dado pela equação 3.1, em que p é o índice do *buffer* que contém a leitura mais antiga. Após o cálculo da média o valor mais antigo guardado é substituído pelo valor mais

recente.

$$\bar{y}[i] = \bar{y}[i-1] + \frac{x[i] - x[p]}{N} \tag{3.1}$$

O tamanho, N, escolhido para o filtro de média deslizante é de 11. Equivale a uma janela temporal de aproximadamente 1 segundo.

**Medição do Ruído em tempo real**

A medição do ruído do sinal é feito em tempo real pelo microcontrolador através do cálculo do erro RMS do sinal de uma janela de tamanho fixo, $N_{RMS}$. Para cada instante, é utilizado o valor médio da janela considerada. Esta é calculada de forma recursiva no entanto, o erro de cada ponto da janela é recalculado tendo em consideração a média dos valores da janela, 3.2.

$$Erro_{RMS}[i] = \sqrt{\frac{\sum_{j=i-N_{RMS}+1}^{i} (\overline{x_i} - x_j)^2}{N_{RMS}}} \tag{3.2}$$

Na figura 3.5 verifica-se a evolução do ruído quando ocorre uma movimentação de massa numa célula de carga.

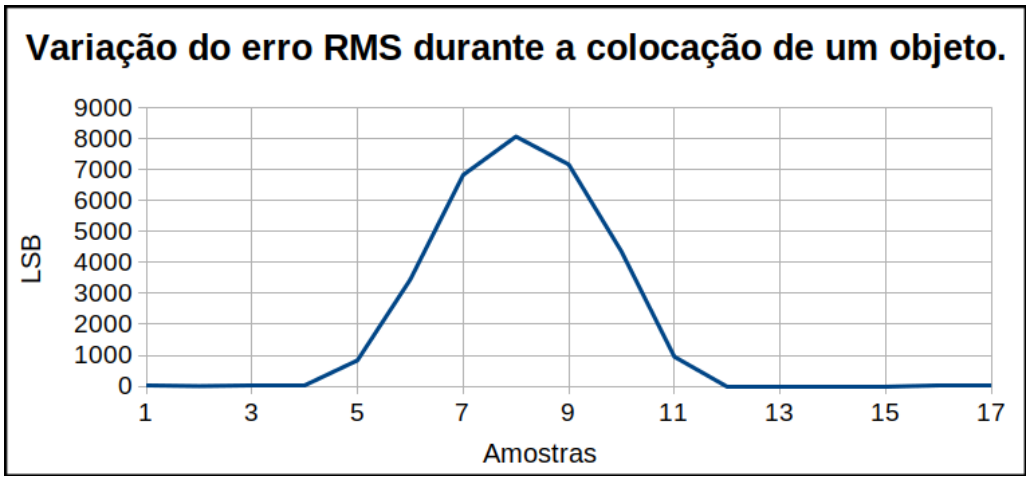



Figura 3.5: Erro RMS aquando do levantamento de uma massa de cima de uma célula de carga.

**Resposta a variações de peso**

Como visto anteriormente, é importante implementar o filtro de média deslizante de forma a reduzir o ruído presente nas medições. No entanto, este filtro impõe um atraso no sinal original igual ao tamanho do filtro, neste caso 11 períodos de amostragem. Ou seja, a uma frequência de amostragem de 11 Hz, o atraso imposto ao sinal é de $11 * \frac{1}{11} = 1 segundo$.

Dado que o tempo de estabilização do sinal de saída das células de carga é de aproximadamente 0.8 segundos, após o filtro de média deslizante, este passa a 1,8 segundos. Como forma de melhorar a resposta do filtro aquando de variações de peso é implementado o algoritmo representado no diagrama da figura 3.6 que permite conceder ao filtro a capacidade de seguir o sinal

original, e de ignorar valores pontuais de ruído. A implementação em código está presente no anexo A.1.

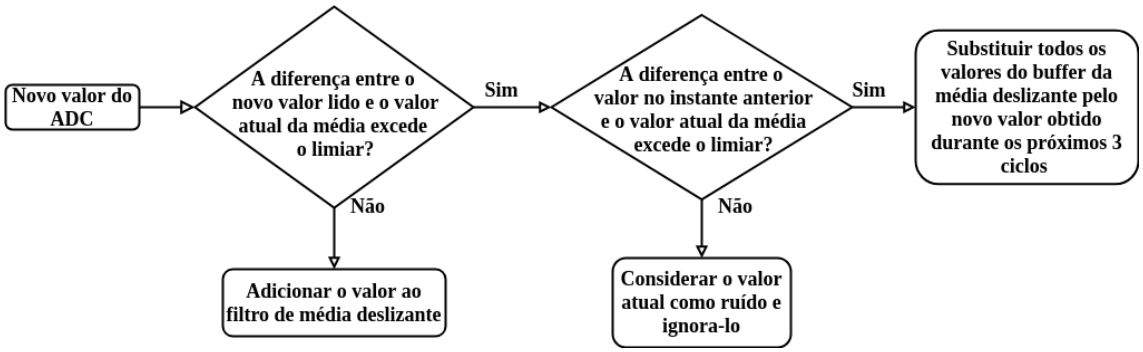



Figura 3.6: Algoritmo de resposta a variações de peso, .

Na figura 3.7 pode-se observar, através da curva amarela, o atraso imposto pela média deslizante. A curva a laranja demonstra a resposta do algoritmo anterior aquando de variações de peso.

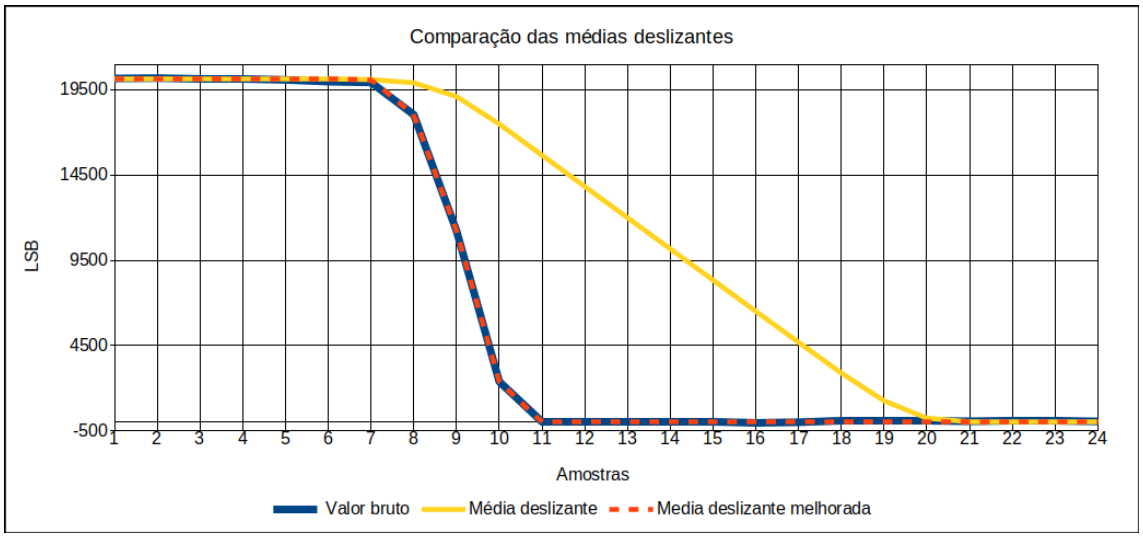



Figura 3.7: Desempenho do algoritmo de resposta a variações de peso.

## 3.3 Inferência de Eventos

**Conceito de Eventos**

Para permitir a gestão de *stocks* de forma autónoma, é necessário detetar quando ocorrem movimentações de produtos numa prateleira e inferir eventos. Sendo assim, um evento é caracterizado pelo peso do objeto e a posição relativa do centro de massa do movimento na prateleira.

Os tipos de eventos considerados são:

- **Colocar:** é caracterizado por um aumento global do peso na prateleira e um aumento de carga em ambas as células de carga, como mostra a figura 3.8;

- **Retirar:** é caracterizado por uma redução global do peso na prateleira e uma redução de carga em ambas as células de carga, como mostra a figura 3.9;
- **Deslocamento:** ao contrário dos eventos anteriores, a carga aplicada em cada célula de carga sofre uma variação de sentidos opostos e a carga total aplicada na superfície da prateleira mantém-se constante, como mostra a figura 3.10.
- **Indefinido:** a carga aplicada em cada célula sofre uma variação de sentidos opostos, no entanto a carga total aplicada na superfície da prateleira é diferente à carga inicialmente aplicada.

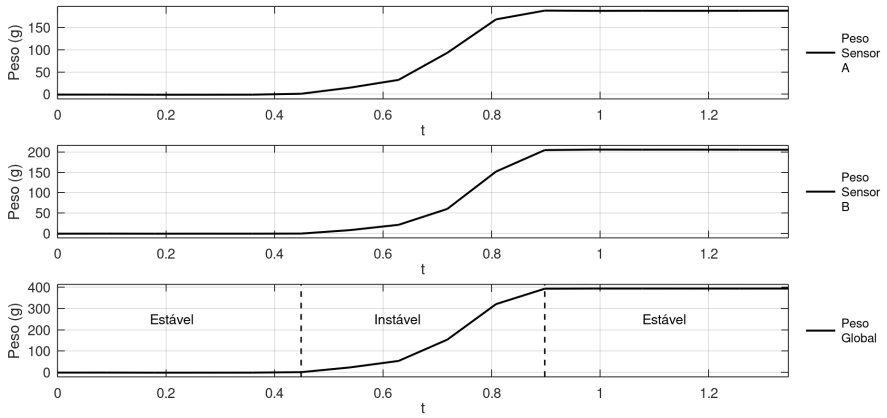



Figura 3.8: Variação do peso ao pousar um objeto na prateleira.

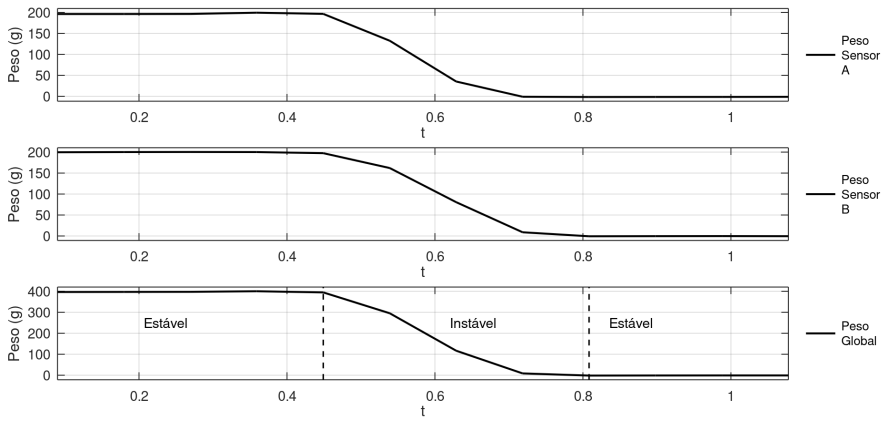



Figura 3.9: Variação do peso ao levantar um objeto na prateleira.

Partindo do princípio que, na ocorrência de um movimento na prateleira, o peso transita de um estado estável[1] para um estado instável [2], até por fim, atingir o estado estável no novo ponto de

[1] o momento antes do início do movimento

[2] o período em que as células de carga se deslocam para o novo ponto de equilíbrio

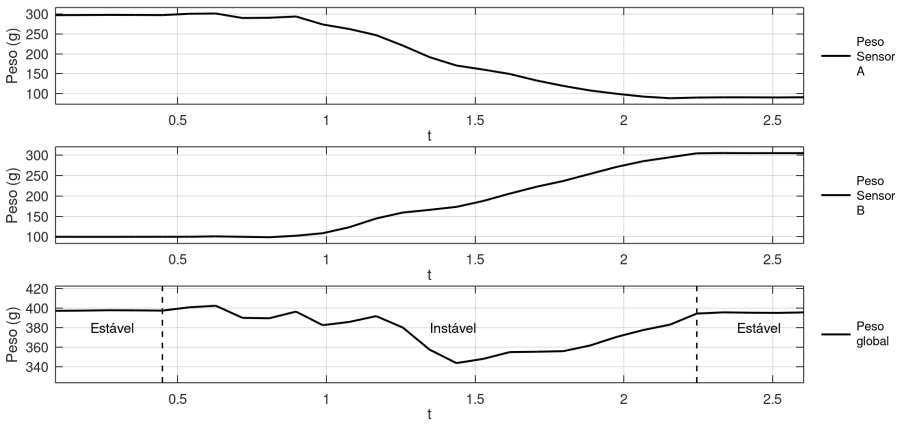



Figura 3.10: Variação do peso ao deslizar um objeto na prateleira.

equilíbrio após o movimento. Então, é necessário obter um parâmetro que permita caracterizar o peso em relação à sua variação em torno de um ponto de equilíbrio.

**Caracterização do peso**

De seguida é desenvolvido um sistema de decisão baseado em lógica difusa que permite caracterizar o sinal do peso relativamente ao seu nível de estabilidade.

É necessário avaliar a variação do sinal em tempo real, sendo utilizados os seguintes indicadores:

- O módulo da diferença finita de 1ª ordem - este parâmetro permite detetar variações imediatas de peso, normalmente o início de um movimento;
- O erro rms deslizante de tamanho $N_{rms}$ - este indicador permite obter o nível do ruído do sinal na janela temporal selecionada. Ademais, auxilia na deteção de pequenas oscilações, como por exemplo oscilações mecânicas que são observadas após um movimento.

As duas variáveis de entrada são classificadas utilizando quatro variáveis linguísticas: Muito Pequeno, Pequeno, Médio e Grande. Estas tem como abreviaturas, respetivamente: MP, P, M, G.

A variação do peso, é classificada através de três variáveis linguísticas: Estável, Pouco Estável e Instável, respetivamente: E, PE, I.

A partir disto, é possível constatar algumas relações entre as variáveis de entrada e de saída, nomeadamente:

- Se a derivada é Muito Pequeno e o erro rms é Muito Pequeno então o sinal é Estável;
- Se o erro rms é Muito Pequeno e a derivada é Médio ou Grande então o sinal é Pouco Estável e representa o caso em que o movimento tem início.
- Se o erro rms é Médio ou Grande e a derivada é Muito Pequena então o sinal é Pouco Estável, representa o instante em que o peso está a estabilizar após um movimento;

- Se o erro rms é Grande e a derivada é Grande então o sinal é Instável, representa o instante em que o peso está a transitar para o novo ponto de equilíbrio.

Baseado nestes princípios, a definição completa das regras que caracterizam o peso da prateleira no decorrer de movimentações de peso encontra-se na tabela 3.2.

| Sinal | | Derivada | | | |
|---|---|---|---|---|---|
| | | MP | P | M | G |
| Erro RMS | MP | E | E | PE | PE |
| | P | E | PE | PE | I |
| | M | PE | PE | I | I |
| | G | PE | I | I | I |

Tabela 3.2: Conjunto de regras difusas.

As funções de pertença das variáveis de entrada são uma mistura entre funções triangulares e trapezoidais. Na figura 3.11b e 3.11a estão representadas as funções de pertença para a variável derivada e o erro RMS, respetivamente.

As funções de pertença para a variável de saída são constantes e estão representadas nas figuras 3.11c.

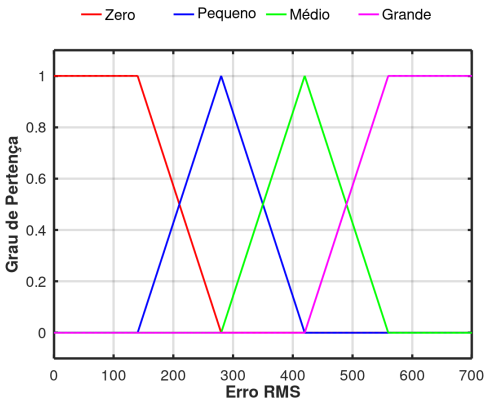
**Algoritmo de Inferência de Evento**

Num evento ideal, inicio de um evento é obtido quando o peso, transita do estado estável para um estado não estável e o fim do evento é dado pela transição de um estado não estável para o estado estável. No entanto, existem vários fatores que podem fazer com que um movimento não ocorra da forma convencional. Por exemplo, a ocorrência de vários eventos sobrepostos, sem permitir que as células de carga tenham tempo de estabilizarem entre os eventos. Desta forma, a abordagem seguida tem em consideração estes fatores.

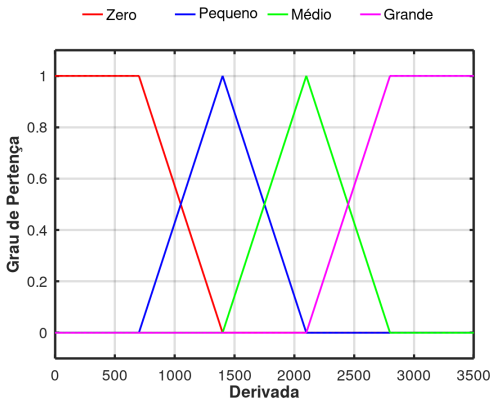
O algoritmo de deteção de eventos é baseado numa máquina de estados, onde as transições entre estados são ativadas por mudanças no estado do sinal de peso, como pode ser observado na figura 3.12. Adicionalmente, é implementado um método de atribuição de pontuação ao movimento baseado em *score*.

Quando a máquina de estados executa pela primeira vez, o estado do peso não é conhecido e, por isso, esta tem inicio no estado de inicialização (INIT) onde aguarda até que o peso se encontre estável.

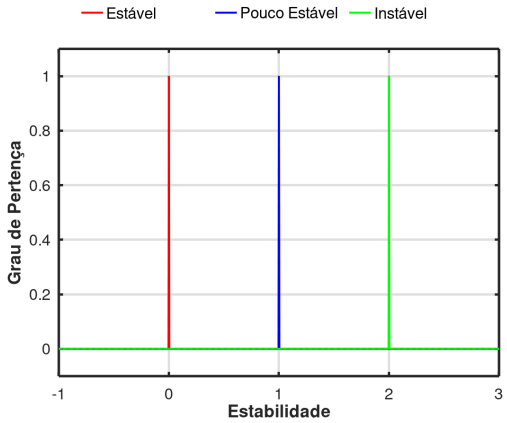
A transição de saída do estado 0 da máquina de estados para os estados 1 ou 2 é despoletado pela mudança de um sinal estável para pouco estável ou instável, respetivamente. Desta forma, o estado 1 representa o início de um movimento lento, em que o peso não varia de forma brusca. O estado 2 representa o ponto médio de um evento, em que a variação de peso é grande. A partir destes dois estados, se o peso se tornar estável é considerado que o movimento terminou.




(a) Funções de pertença da variável erro RMS.




(b) Funções de pertença da variável derivada.




(c) Funções de pertença da variável de saída.

Figura 3.11: Funções de pertença do método fuzzy.

E são então determinadas as características do evento e a pontuação do evento é máxima, pois o movimento partiu de um ponto estável até outro ponto estável.

Quando no estado 2, e a variação do peso passa a pouco instável, ocorre a transição para o estado 3. Este estado representa a parte final do movimento em que o peso sofre, maioritariamente, oscilações mecânicas antes de alcançar o ponto de equilíbrio. Aqui, a pontuação é incrementada em 1 valor, até que o peso esteja estável ou volte a ficar instável, este último significando que está a ocorrer uma movimentação nova não relaciona com a anterior. No último caso, se a pontuação for superior à pontuação mínima, o movimento é considerado como verdadeiro, caso não seja superior à pontuação mínima o movimento é agregado ao movimento seguinte.

**Caracterização do Evento**

Após a deteção de um movimento é necessário caracteriza-lo em um evento determinando o tipo de evento (Colocar, Retirar, Deslocamento, Indefinido), a variação de peso total, e a posição do centro de massa do evento.

O peso que caracteriza o evento, é dado pela variação global de peso desde o início até ao final do movimento, $P_e = P[i] - P[e_0]$. De seguida, se o peso do evento é superior a um dado limite, $|P_e| > \varepsilon$, e se este é positivo ocorre o evento de colocar um objeto. Pelo contrário, se o peso é

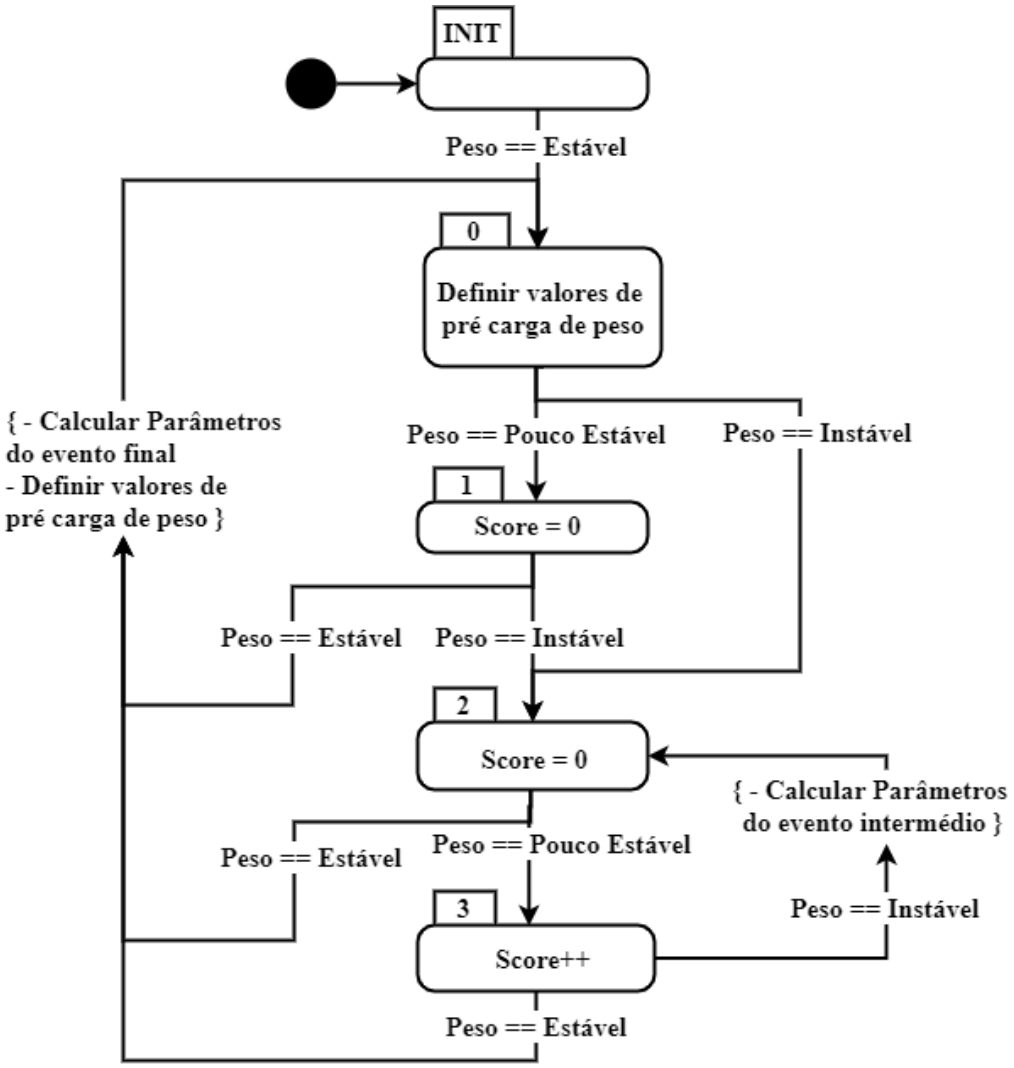



Figura 3.12: Algoritmo de inferência de eventos.

negativo ocorre o evento de retirar um objeto da superfície da prateleira.

O centro de massa do movimento é dado pela variação do peso no sensor B dividido pelo peso total do movimento, $CM_e = \frac{P_B - P_{B_0}}{P_e}$. Quando o peso do evento é menor que $\varepsilon$, e a mudança do centro de gravidade é diferente de zero ocorre um evento de deslocamento de objetos.

O caso especial de evento indefinido verifica-se na ocorrência simultânea dos três eventos anteriores e é caracterizado por ter uma variação total de peso diferente de zero e a variação do peso nas células de carga tem sentidos opostos - como no movimento de deslocamento. As caracterizações dos eventos encontram-se resumidas na tabela 3.3.

A implementação do algoritmo em código está presente no anexo A.2.

| | | Variação de carga | | |
|---|---|---|---|---|
| | | Sensor A | Sensor B | Global |
| Evento | Colocar | + | + | $P_e > \varepsilon$ |
| | Retirar | - | - | $P_e < -\varepsilon$ |
| | Deslocamento | + / - | - / + | $\lvert P_e \rvert < \varepsilon$ |
| | Indefinido | + / - | - / + | $\lvert P_e \rvert > \varepsilon$ |

Tabela 3.3: Características da variação de carga na determinação de um evento.

## 3.4 Arquitetura do Controlador

O *hardware* utilizado terá diversas aplicações em diversos produtos dentro do ceio da empresa. Como tal, o *software* desenvolvido é adaptável às situações de utilização do *hardware*. A arquitetura do sistema do controlador está representada no diagrama de objetos na figura 3.13.

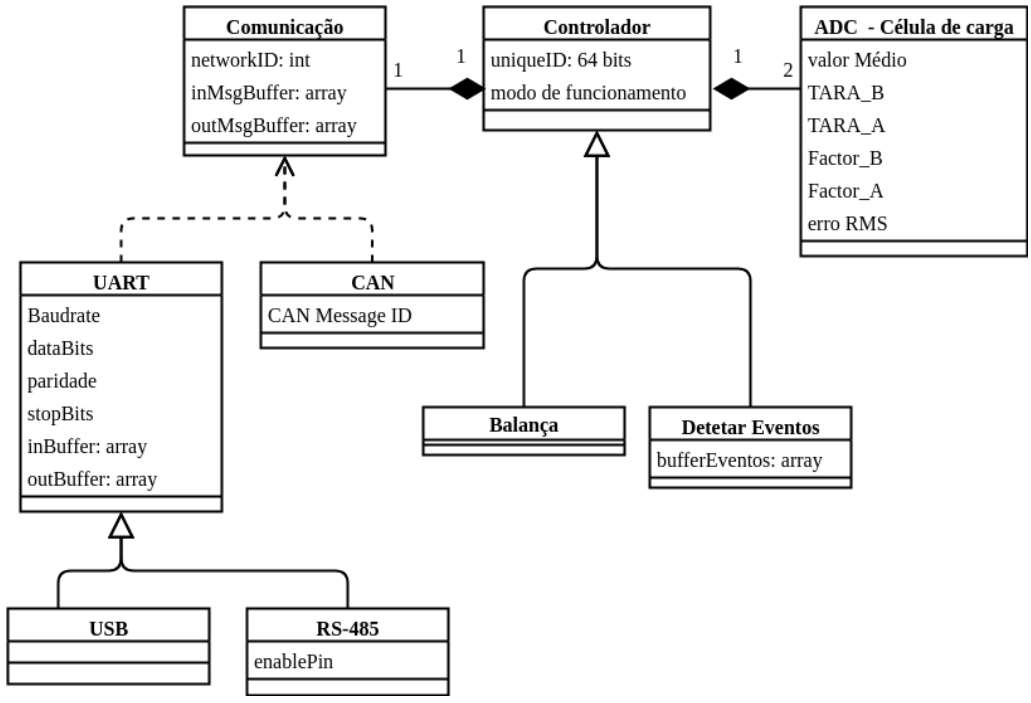



Figura 3.13: Diagrama de classes do controlador.

**Modo de detetar eventos**

O modo de deteção de eventos torna o controlador num dispositivo independente detetando eventos e comunicando-os ao concentrador para posterior análise.

**Modo de balança**

No modo de balança, o controlador continua a realizar as leituras das células de carga e a

fazer o processamento do sinal. No entanto, este não realiza processamento com os dados adquiridos afim de detetar eventos. Neste modo, o controlador funciona em modo *pollling*, em que o concentrador questiona o controlador pela medição de peso mais recente.

**Comandos**

O controlador disponibiliza comandos para realizar a configuração de parâmetros e alterar o funcionamento do controlador sendo estes: TARA, calibração, informação, configurar identificador de rede, pedido de medidas e definir modo de funcionamento. Os comandos TARA, calibração, informação, configurar identificador de rede e pedido de medidas já se encontram implementados no protocolo de comunicação desenvolvido pela empresa, sendo acrescentado o comando"definir modo de funcionamento" à lista de comandos. Estes comandos são baseados em ASCII sendo o primeiro caractere o identificador do comando, seguido pelo identificador de rede do controlador e, por fim, os parâmetros do comando.

- TARA: O comando de TARA permite definir a situação atual como a referência do zero absoluto de carga, para cada sensor. É essencial que o valor de TARA seja obtido em situações em que a balança esteja completamente estável. Desta forma, é calculado o valor médio de 32 amostras sucessivas [3] sendo que, se for detado uma variação brusca do sinal durante o período de recolha de amostras, o processo recomeça.

- Calibração: O comando de calibração é acompanhado pela indicação do peso de um objeto que será utilizado para determinar os fatores multiplicativos de peso de cada célula de carga. O fluxograma da figura 3.14 descreve o processo de calibração.

- Configurar Identificador de rede: Este comando permite indicar ao controlador um identificador pelo qual ele será identificado no barramento série.

- Pedido de medidas: O controlador responde com as medidas de peso mais recentes.

- Definir modo de funcionamento: Comando utilizado para alternar entre os dois modos de funcionamento.

## 3.5 Concentrador

O concentrador tem o objetivo de gerir a comunicação dos controladores conectados ao barramento série, atribuindo identificadores únicos a cada dispositivo. Este transmite os dados gerados pelas prateleiras à aplicação *web* e também de envia comandos de configuração até às prateleiras.

O concentrador, figura 3.15, foi desenvolvido em ambiente de simulação num PC utilizando a plataforma QT creator. Atuando como um *software* de apoio ao desenvolvimento do sistema e permitindo realizar *debug* de forma simples. Assim o concentrador implementado possui uma interface gráfica que permite acompanhar as comunicações e enviar comandos às prateleiras.

[3] Aproximadamente 2,9 segundos, admitindo que a taxa de amostragem dos ADC é 11 Hz

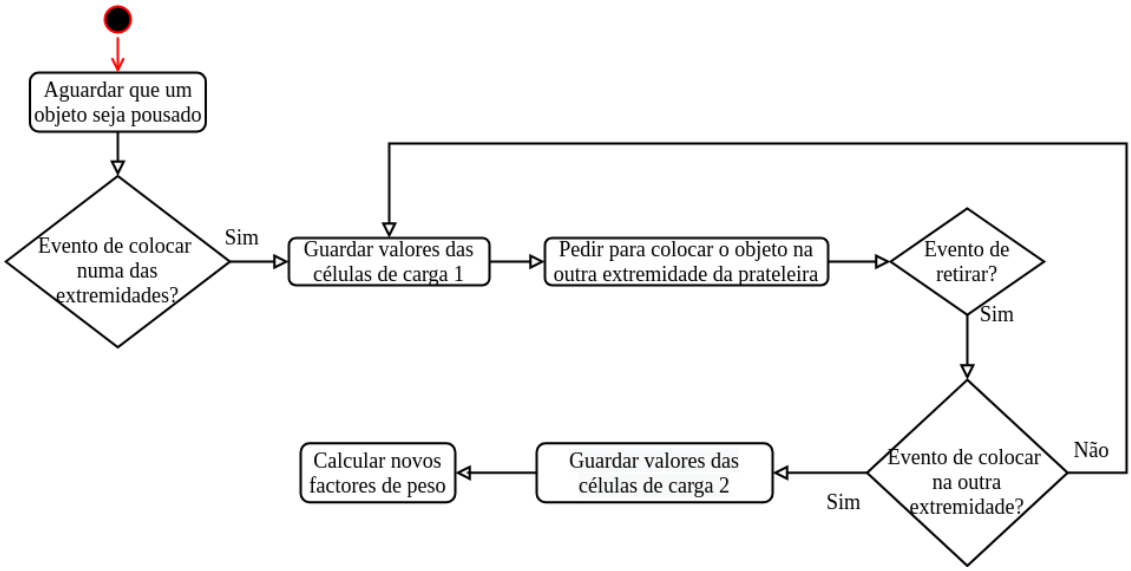



Figura 3.14: Processo de calibração.

No entanto, o sistema real utiliza um concentrador implementado em *hardware*, endereçável univocamente em rede e capaz de conectar até 32 unidades em cascata, utilizando cabos UTP padrão.

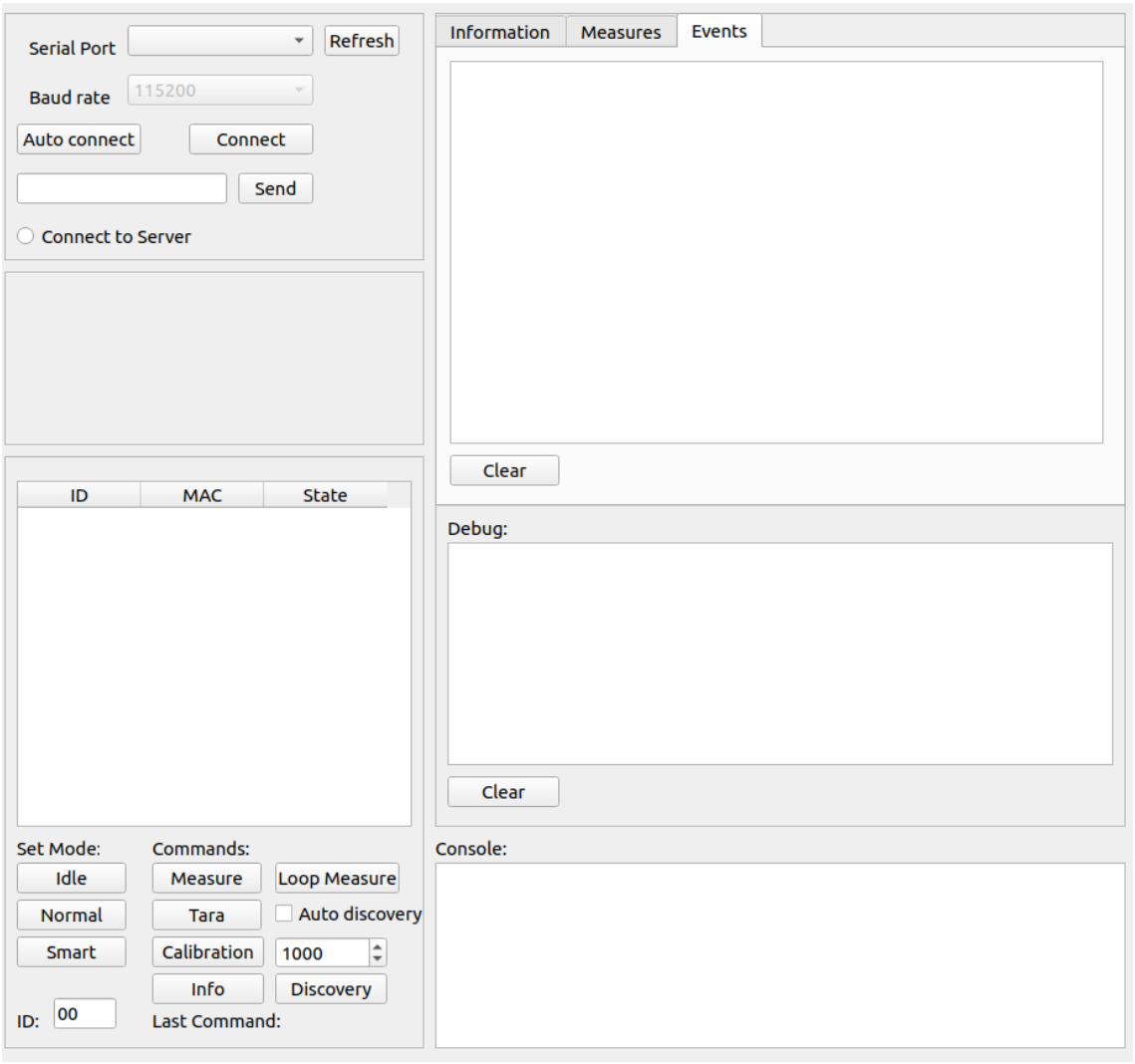



Figura 3.15: Interface de .

# Capítulo 4

# Algoritmo de Gestão de *Stock* e Resultados

Neste capitulo, é explorado o algoritmo de inferência de *stock*. Passando pela organização dos produtos numa prateleira de supermercado até à análise de quatro casos de estudo que caracterizam movimentos que ocorrem na prateleira e a forma de inferir o *stock* associado a um evento.

Seguidamente, na secção 4.2 são desenvolvidos os métodos da API, dispondo de uma interface entre as prateleiras e o servidor e outra para os utilizadores interagirem com as prateleiras.

Por último, são apresentados os resultados obtidos em vários testes ao sistema, nomeadamente, ao ruído do HX711, ao algoritmo de inferência de eventos e ao algoritmo de inferência de stock.

## 4.1 Algoritmo de Inferência de *Stock*

**Planograma**

Na figura 4.1 observa-se um exemplo de produtos expostos numa prateleira.

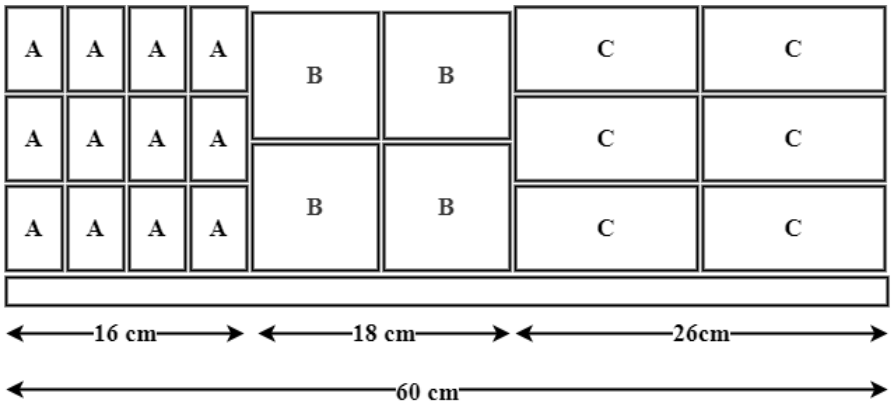



Figura 4.1: Exemplo de produtos expostos numa prateleira.

A exposição dos produtos numa prateleira é feita por zonas, em cada uma existe apenas um tipo de produto. A zona é caracterizada pelo número de produtos segundo o plano frontal ($q_f$), o

número de níveis de produtos ($q_a$) e o o número de produtos em profundidade ($q_p$) - figura 4.2. Assim, o stock máximo de produtos existentes em cada zona é dado por $Q_{max} = q_f * q_a * q_p$. Cada produto tem associado as suas dimensões físicas o que permite determinar na prateleira o início e o fim de uma zona, admitindo que não existem espaços vazios entre produtos, figura 4.3.

| *Planograma* |
| --- |
| Produto |
| Número de frentes |
| Número de níveis |
| Número profundidade |
| Posição de inicio |
| Posição de fim |
| Stock máximo |

Figura 4.2: Dados associados ao planograma.

| *Produto* |
| --- |
| Nome |
| largura |
| Altura |
| Profundidade |
| Peso |
| Peso estimado |
| erro padrão do peso |

Figura 4.3: Dados associados ao planograma.

**Premissas**

O algoritmo de determinação de produtos associados a um movimento baseia-se, inicialmente, nas seguintes limitações:

- Todos os produtos da mesma zona têm o mesmo peso teórico;
- As medições de peso e posição, realizadas pela prateleira, são precisas;
- As zonas de uma prateleira são subdivididas em filas de produtos, orientadas em profundidade;
- Nenhum produto é colocado fora da sua zona;
- Os eventos detetados envolvem apenas produtos pertencentes à prateleira, caso contrário ocorrem situações de erro.

**Casos de estudo**

Foram realizadas três análises, que de seguida serão expostas. Iniciando por uma análise simples que envolve a movimentação de produtos individuais, passando pela análise dos casos em que os movimentos incluem produtos diferentes, casos em que são retirados e colocados produtos num só movimento. Até, por fim, serem abordados casos em que são colocados objetos estranhos na prateleira.

**1º Caso**

No primeiro caso de análise apenas é permitida a movimentação de um tipo de produto de cada vez, não havendo limitações impostas na quantidade. Neste caso, verifica-se que é possível determinar o tipo de produto procurando pela zona que corresponde à posição do movimento. O peso

associado ao movimento é um valor múltiplo do peso individual do produto, logo a quantidade de produtos será dada pela divisão entre o peso medido e o peso do produto da zona i, $Q_i = \frac{P_e}{P_i}$, arredondado às unidades.

**2º Caso**

No segundo caso, é possível movimentar mais de um tipo de produtos de cada vez. No entanto, não sendo permitido que a um movimento esteja associada a colocação e/ou levantamento de objetos fora dos limites das suas zonas. Assim, verifica-se que o método anteriormente visto não tem os resultados esperados, pois o centro de massa de um movimento que inclua objetos de diferentes referências não identifica diretamente os objetos envolvidos.

Desta forma, o foco prende-se em encontrar uma combinação de produtos que descreva o evento ocorrido, $C_c = \{Q_{c1}, Q_{c2}, ..., Q_{c_{n_{zonas}}}\}$, cujo peso total de produtos coincida com a diferença de peso do evento, $P_e$, e o centro de massa da combinação coincida com o centro de massa do evento, $CM_e$ ($|Peso_e - Peso_c| \leq \varepsilon_{peso}$ e $|CM_e - CM_c| \leq \varepsilon_{posicao}$.). Assim, para cada combinação c, pode-se obter o peso total, $Peso_c$, e o centro de massa, $CM_c$, a partir das seguintes equações, em que $i = \{1, 2, ...n_{zonas}\}$:

$$Peso_c = \sum_{c=i}^{n_{zonas}} (Peso_i * Q_{ci}) \tag{4.1}$$

$$CM_c = \frac{\sum_{c=i}^{n_{zonas}} (Peso_i * Q_{ci} * CM_i)}{Peso_c} \tag{4.2}$$

O número de combinações de quantidades de objetos possíveis de obter é dado por:

$$Nmax_{comb} = 1 + \prod_{i=1}^{zonas} Nmax_i, \tag{4.3}$$

em que $Nmax_i = \lceil \frac{P_e}{P_i} \rceil$.

O número total de combinações rapidamente cresce com o número de frentes de produtos expostos numa prateleira. Contudo, é possível minimizá-lo calculando as combinações possíveis, que satisfaçam a condição de peso, de forma iterativa. Dando início com a combinação

$$Q_1 = \{0, 0, ..., mdc(P_{planograma}, P_e)\},$$

para cada iteração o algoritmo verifica se a condição de peso é satisfeita, e caso seja, esta combinação é guardada. No caso em que o peso da combinação ultrapassa o peso do evento, a quantidade do produto da direita é colocado a zero e a quantidade do produto seguinte, o da esquerda, é incrementada. Quando a quantidade de um produto atinge o limite máximo, é colocado a zero e a quantidade do produto à esquerda é incrementada. São utilizadas apenas quantidades positivas de produtos pois o peso do evento de retirar é simétrico ao evento de colocar e no caso de ocorrer um evento de retirar as quantidades de produtos são invertidas antes de determinar o centro de massa.

Após determinadas as combinações de quantidades de produtos que satisfaçam ambas as condições, é calculado o erro de peso e posição, sendo estes normalizados aos intervalos [0, max erro de peso] e [0, max erro de CM], respetivamente. O critério de decisão passa por encontrar a combinação que apresenta o valor mais baixo de 4.4.

$$\sqrt{(\text{erro Peso normalizado})^2 + (\text{erro CM normalizado})^2} \tag{4.4}$$

**3º Caso**

Quando ocorre um evento indefinido, por exemplo colocar e retirar simultâneamente objetos diferentes [1] o peso associado ao evento é a diferença dos pesos dos objetos associados. Contudo, o centro de massa do evento geralmente está posicionado fora dos limites da prateleira. Quando este evento ocorre próximo do centro da prateleira esta condição já não se verifica. A determinação das combinações ocorre da mesma forma que no caso anterior, no entanto, as quantidades iniciais de produtos são valores negativos.

**4º Caso**

Quando são colocados produtos estranhos na prateleira existem dois desfechos possíveis. O objeto pode ser reconhecido como um produto pertencente à prateleira, no caso em que o seu peso e posição coincide com uma combinação de produtos associada à prateleira, gerando assim uma situação de falso positivo.

O outro desfecho é o objeto não ser reconhecido como um produto da prateleira e neste caso é possível gerar um alerta de objeto colocado fora de posição para possível correção manual, desta forma eliminando o requisisto que que o sprodutos não podem ser colocados fora das suas zonas.

**Algoritmo**

Os passos seguidos após a deteção do evento na prateleira, e consequente envio da informação ao servidor, até à inferência dos produtos e respetivas quantidades está representado no fluxograma da figura 4.4 e a sua implementação no anexo B.1.

## 4.2 API

A API é desenvolvida em PHP, tendo como base a *framework slim*. Desta forma são disponibilizados vários métodos divididos em duas categorias: interface com as prateleiras e uma interface de utilizador.

**Interface com a prateleira**

A interface com a prateleira é realizada através de um método, utilizado para a comunicação do estado atual e eventos ocorridos na prateleira, sendo este identificado pelo seguinte URI:

/shelf/{identificador}/update/{frame}

Os parâmetros entre chavetas são parâmetros reconhecidos pela *framework slim*. O parâmetro "identificador"identifica, através do número de identificação único, a prateleira e o parâmetro

[1] troca de produtos de tipos diferentes

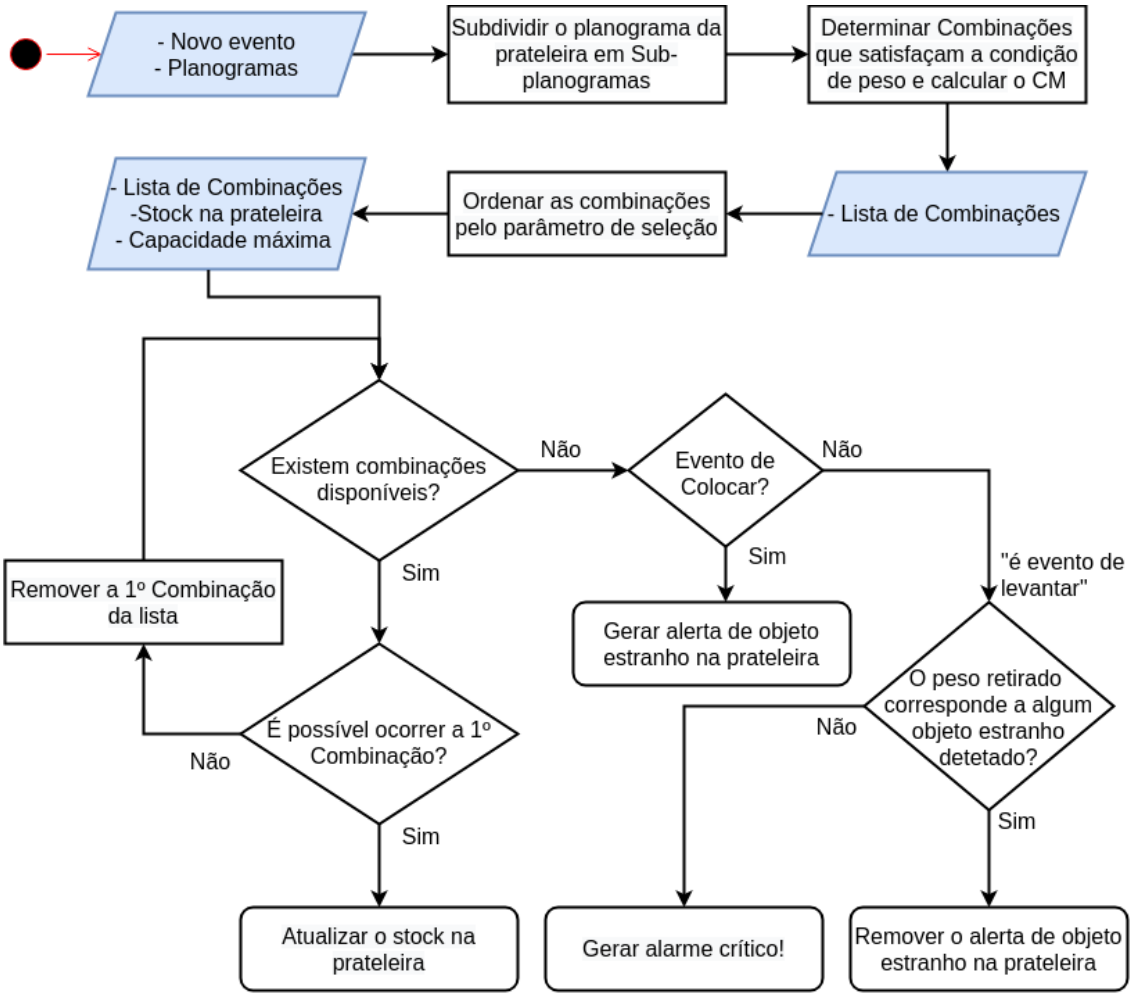



Figura 4.4: Processo de inferência de produtos associados a um evento.

"*frame*" é composto pelas seguintes variáveis que caracterizam o evento, separadas por ";": tipo de evento, peso inicial, peso Final, peso, CM Inicial, CM Final, CM, flag de evento indefinido, *score*. Utilizando um método GET do HTTP.

**Interface de utilizador**

A interface com o utilizador disponibiliza as seguintes funcionalidades:

- Adicionar produtos à base de dados;
- Definir a ordem de disposição dos produtos numa prateleira;
- Fazer reposição de produtos numa prateleira;
- Fazer correções de *stock* numa prateleira manualmente;
- Visualizar o *stock* de cada produto numa prateleira, com identificação de objetos colocados fora do sítio;

**Adicionar produtos**

A funcionalidade de adição de produtos requer que o utilizador introduza tanto a identificação do produto, SKU, como as dimensões físicas do mesmo. De forma opcional, pode introduzir o peso real do produto, caso contrário será inferido durante o processo de reposição do produto.

**Definir Disposição de produtos**

A interface de definição de disposição de produtos numa prateleira permite que o utilizador defina os produtos associados à prateleira, e a localização na prateleira de cada um.

**Reposição de produtos**

De forma a repor produtos na prateleira é necessário coloca-la em modo de reposião. Neste modo todos os eventos criados pela prateleira são ignorados pelo algoritmo de inferência de produtos. Após o início da reposição, o funcionário coloca os produtos na prateleira, nas suas devidas zonas. Para terminar o processo, o funcionário introduz na interface a quantidade total de produtos que colocou na prateleira e o sistema repõe o *stock* e faz uma estimativa do peso individual dos produtos c na prateleira. - figura 4.5.

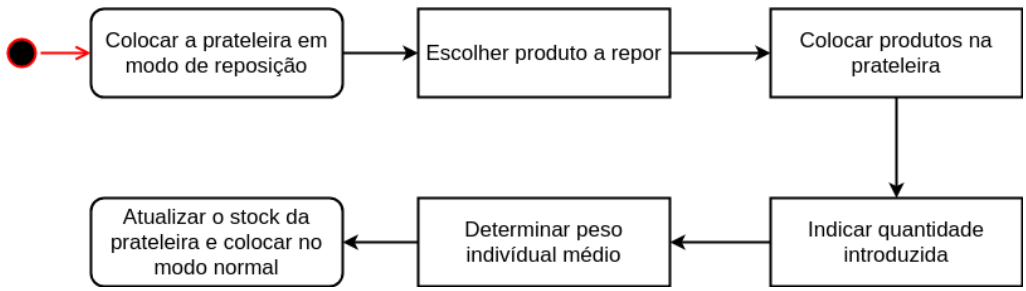



Figura 4.5: Algoritmo de reposição de produtos numa prateleira.

**Correção manual de *stock***

Em situações em que o algoritmo de inferência de *stock* não produz os resultados esperados, é possível corrigir o *stock* na prateleira de forma manual através desta interface. Aqui o funcionário introduz a quantidade de produtos a adicionar ou retirar de forma a obter a quantidade de produtos correta.

**Visualizar *Stock***

A interface de visualização de *stock* permite observar as entradas e saídas de *stock* nas prateleiras. Para além disso, sinaliza quando um dado produto está a esgotar na prateleira e permite obter visualmente a posição de um objeto quando este é colocado fora do seu lugar designado. Oferece também uma interface que permite corrigir, de forma simplificada, eventos associados a objetos fora do sítio.

Na figura 4.6 observa-se a representação dos produtos nas suas devidas posições na prateleira, com identificação do produto e a respetiva quantidade existente. A representação do *stock* existente de cada produto é feita através do desenho de colunas de produtos e de um esquema de cores. As colunas desenhadas representam a percentagem de produtos existentes, sendo o seu interior preenchido de forma a representar o nível atual de produtos existentes. É também utilizado um esquema de quatro cores para representar a quantidade de produtos em relação à capacidade

máxima, e alertar o funcionário nos casos de *stock* reduzido. As seguintes cores representam o *stock* existente: 0%-25% - vermelho; 25%-50% - amarelo; 50%-75% - verde claro; 75%-100% - verde escuro. Quando não existe *stock* de um dado produto na prateleira não são desenhadas colunas. Na figura 4.7 pode-se observar as representações de stock.

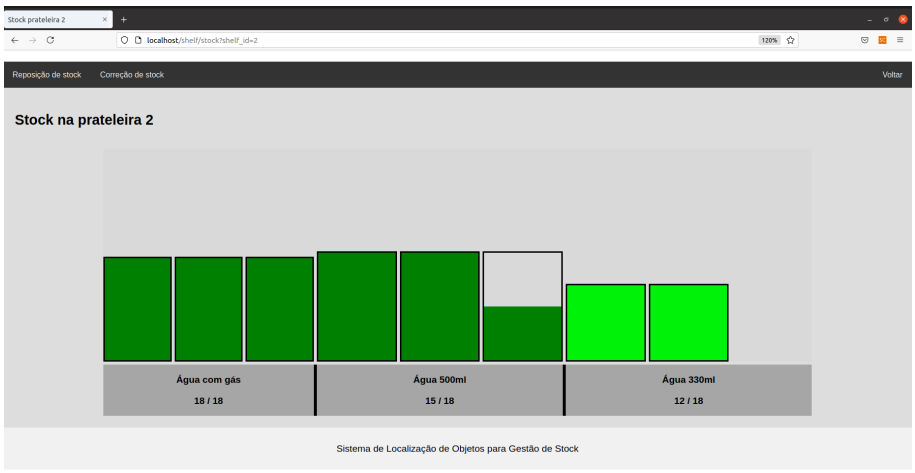



Figura 4.6: Representação dos produtos existentes numa prateleira após reposição completa e/ou parcial de produtos.

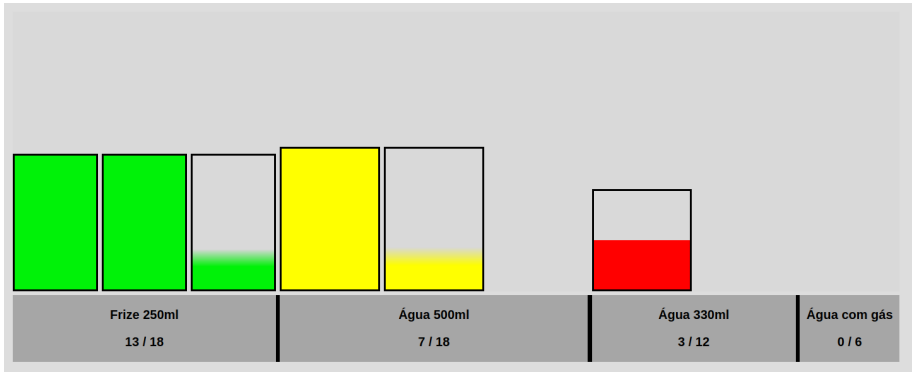



Figura 4.7: Representação do nível de *stock* existente numa prateleira.

A representação de objetos colocados fora de sítio é feita através do desenho de um ponto de interrogação na posição onde o objeto foi colocado. Para mais, é mostrada uma lista de objetos colocados fora de sítio possibilitando ao funcionário corrigir estes eventos nos casos em que o sistema não o consegue fazer de forma autónoma. O sistema disponibiliza a opção de ignorar o evento ou uma possível solução para resolução do evento. Caso a solução disponibilizada não represente o evento, o funcionário ignora o evento e corrige a incoerência na interface de correção manual de *stock*. Na figura 4.8 observa-se a representação dos objetos colocados fora de sítio.

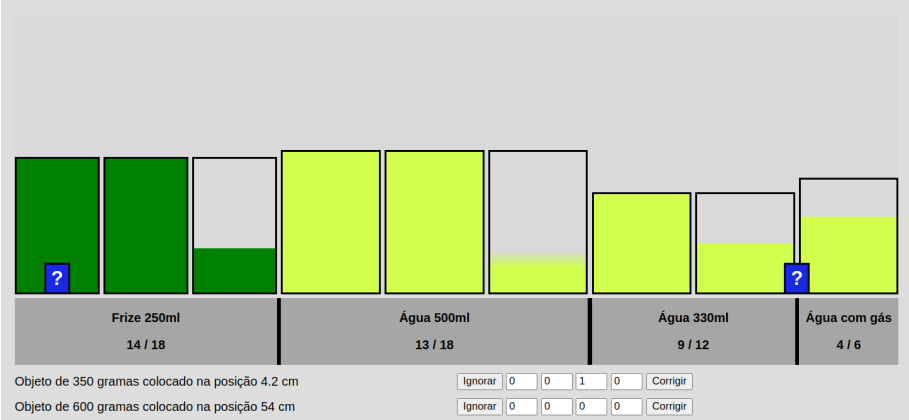



Figura 4.8: Representação do nível de *stock* existente numa prateleira, com indicação de dois objetos colocados fora do sítio.

## 4.3 Resultados

### 4.3.1 Ruído do sensor HX711

Nesta secção pretende-se determinar a diferença entre a utilização de um ADC equipado com blindagem e sem blindagem e a influência desta característica no ruído aleatório observado. Procura-se também verificar a influência da utilização de um filtro de média deslizante.

Para cada teste foram recolhidas amostras com o ADC conectado à célula de carga. Desta forma, para ambos os casos, os valores obtidos de ruído englobam o ruído da célula de carga e o ruído do ADC. No entanto, assume-se que o ruído das células de carga é muito inferior ao ruído de leitura do HX711.

**Sem blindagem**

Neste ensaio foram adquiridas 720 amostras contínuas de um HX711 sem blindagem conectado a uma célula de carga em regime de equilíbrio e sem carga aplicada.

Na figura 4.9 observa-se a distribuição dos valores lidos durante o período considerado. Verifica-se que o valor do desvio padrão é de 1521 LSB e o ruído pico-a-pico de 10039 LSB ($6.6 * ruidoRMS$). Convertendo em bits de ruído obtém-se 13.29 bits de ruído ($log_2(ruido_{p-p})$). Assim a resolução do ADC livre de ruído é de 10.71 bits (24 - ruído em bits).

**Com blindagem**

Neste ensaio foram adquiridas 1000 amostras contínuas de um HX711 com blindagem conectado a uma célula de carga em regime de equilíbrio e sem carga aplicada.

Na figura 4.10 observa-se a distribuição dos valores lidos durante o período considerado. Verifica-se que o valor do desvio padrão é de 28.39 LSB e o ruído pico-a-pico de 187.37 LSB ($6.6 * ruidoRMS$). Convertendo em bits de ruído obtém-se 7.55 bits de ruído ($log_2(ruido_{p-p})$). Assim a resolução do ADC livre de ruído é de 16.45 bits (24 - ruído em bits).

**Média deslizante**

Foi aplicado um filtro de média deslizante de tamanho 11 nos dois casos anteriores.

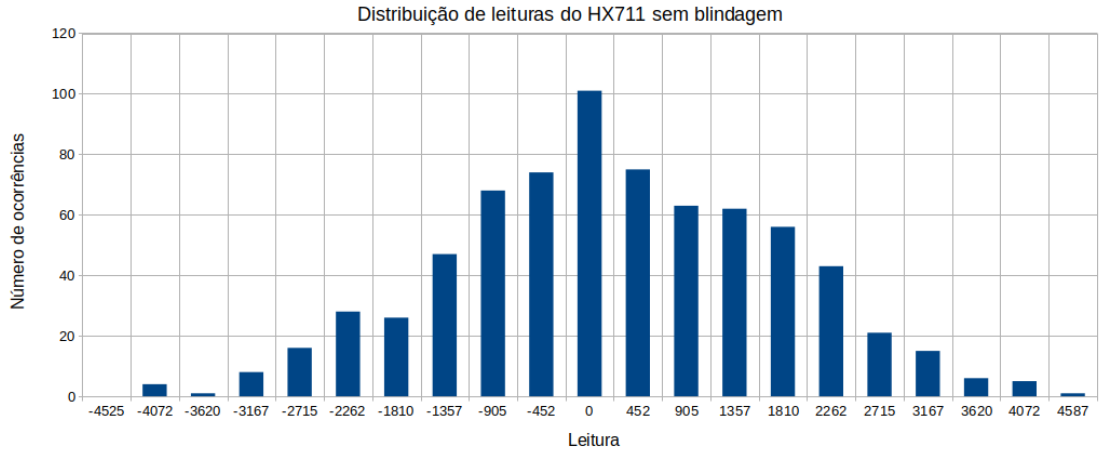



Figura 4.9: Distribuição em LSB das leituras realizadas por um HX711 sem blindagem.

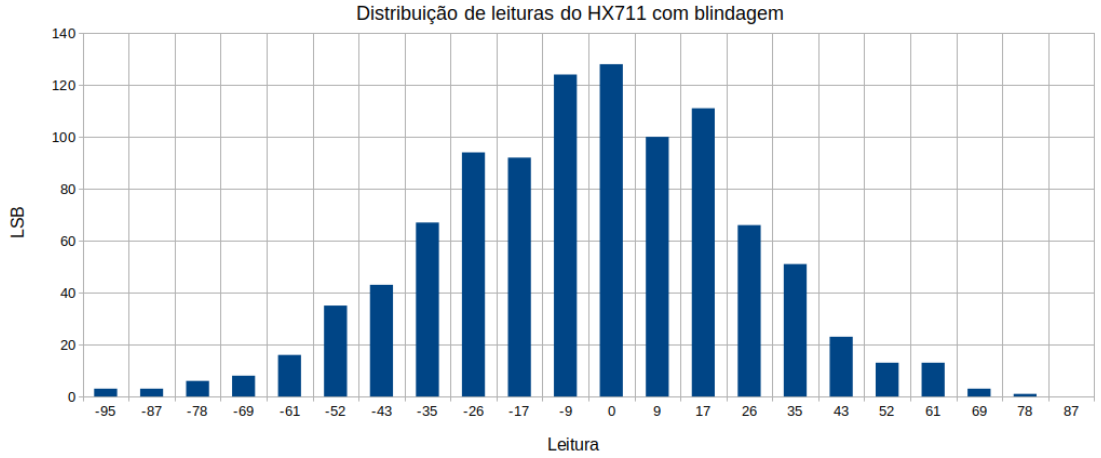



Figura 4.10: Distribuição em LSB das leituras realizadas por um HX711 com blindagem.

No ensaio do ADC sem blindagem, o desvio padrão baixou para 626 LSB levando ao aumento do número de bits do ADC sem ruído para 11.99 bits, um aumento de 1.28 bits.

No ensaio do ADC com blindagem, o desvio padrão do sinal reduziu para 12,40 LSB. O que aumenta o número de bits sem ruído do ADC em 1.20 bits para 17.65 bits sem ruído.

**Discussão**

Verifica-se que a utilização do escudo eletromagnético em volta do ADC reduz, de forma eficaz, o ruído das medições. Sendo que, o fator com mais impacto na redução do ruído de medição é a utilização de uma blindagem para proteger o ADC de interferências eletromagnéticas, com um aumento de 5.74 bits de resolução do ADC sem ruído. A média deslizante de tamanho 11, apresentou resultados similares nos dois casos, aumentando a resolução do ADC em aproximadamente 1.20 bits. Assim, através da utilização de um ADC blindado e da aplicação do filtro de média deslizante obtém-se um ruído pico a pico de aproximadamente $12.40 \times 6.6 \times 0.0072 = 0.59$

gramas (ruído LSB $\times 6.6 \times \frac{\text{gramas}}{\text{divisão}}$).

### 4.3.2 Deteção de eventos

De forma a avaliar o desempenho do algoritmo de deteção de eventos, foram realizados diversos movimentos utilizando objetos de pesos diferentes em diversas posições da superfície da prateleira. Os objetos utilizados foram: garrafas de água de 500 mililitros (aproximadamente 522 gramas) e garrafões de água de 5 litros (aproximadamente 5074 gramas). Foram feitas três marcações na superfície da prateleira que representa o centro de massa dos eventos, na posição 25%, 50% e 75%.

A seguinte análise avalia o desempenho do sistema na capacidade de detetar os eventos; a capacidade de medir o peso dos objetos de forma precisa; e a capacidade de detetar o centro de massa do evento.

**Eventos realizados**

Na tabela 4.1 observa-se o número total de eventos realizados durante o teste, categorizados pelo tipo de evento, peso e posição. Também são apresentados os eventos detetados pelo algoritmo.

| Evento | Peso (g) | Posição (%) | Ocorrências | Detetados |
|---|---|---|---|---|
| Colocar | 522 | 25 | 31 | 31 |
| | | 50 | 32 | 32 |
| | | 75 | 30 | 30 |
| | 5074 | 25 | 20 | 20 |
| | | 50 | 20 | 20 |
| | | 75 | 20 | 20 |
| Retirar | 522 | 25 | 31 | 31 |
| | | 50 | 32 | 32 |
| | | 75 | 29 | 29 |
| | 5074 | 25 | 20 | 20 |
| | | 50 | 20 | 20 |
| | | 75 | 20 | 20 |
| Outros | | | 0 | 5 |
| Total | | | 305 | 310 |

Tabela 4.1: Eventos realizados e detetados.

**Análise dos dados recolhidos**

Os resultados de deteção de eventos teve uma taxa de verdadeiros positivos de 100%, sendo que, em certas ocasiões, em que o tempo entre a colocação/levantamento de objetos era curto,

o evento continha informação dos movimentos juntos. Estes eventos foram considerados como corretos, pois identificavam os objetos corretamente em termos de peso e posição. Foram identificados 5 falsos positivos, estes foram verificados apenas nos ensaios com o garrafão de água. São eventos classificados como "Deslocamento"e que ocorrem devido a oscilações mecânicas geradas pela grande massa do garrafão de água.

Nos gráficos das figuras 4.11 e 4.12 observa-se a distribuição dos valores das medições realizadas para o peso de 522 gramas e o de 5074 gramas, respetivamente. Verifica-se que o desvio padrão das medições do objeto mais leve é de 2.7 gramas e do objeto mais pesado é de 4.8 gramas. Indicando que 99.7% das medições realizadas encontram-se dentro do intervalo 522 $\pm$ 8.1 gramas, no caso da garrafa de água, e 5074 $\pm$ 14.4 gramas para o caso do garrafão. O maior erro observado, para o peso de 522 gramas, foi de 14 gramas representando 2.7% do peso real. Para o peso de 5047, o maior erro foi de 28 gramas o representa 0.55% do peso real.

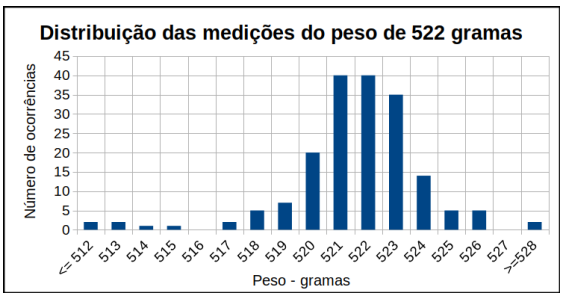



Figura 4.11: Distribuição das medições do peso de 522 gramas.

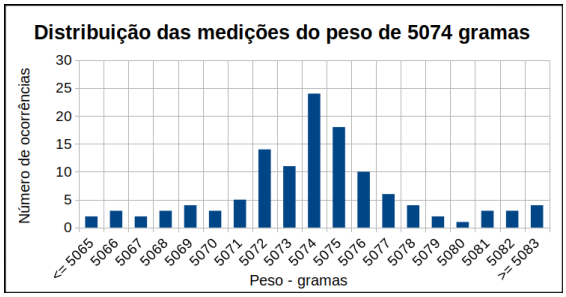



Figura 4.12: Distribuição das medições do peso de 5074 gramas.

Nos gráficos das figuras 4.13, 4.14 e 4.15 observa-se o erro das medições realizadas nas posições 25%, 50% e 75%, respetivamente. Para cada posição, o desvio padrão das medições é de 2.7 gramas, 4.8 gramas e de 3.1 gramas. Verificando-se que, no centro da prateleira, o erro de medição de peso é superior em relação às extremidades da prateleira. Tal ocorre pelo fato de que no centro, a base da prateleira está sujeita a uma maior deflexão que por sua vez origina grandes oscilações mecânicas.

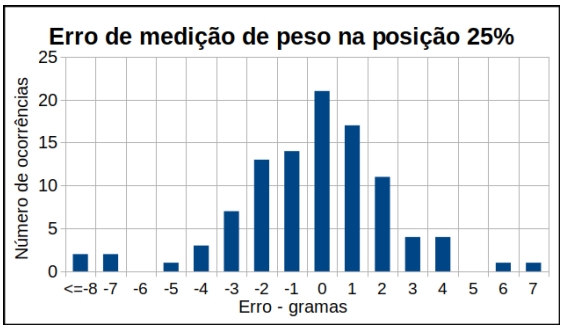



Figura 4.13: Distribuição do erro das medições do peso na posição 25%.

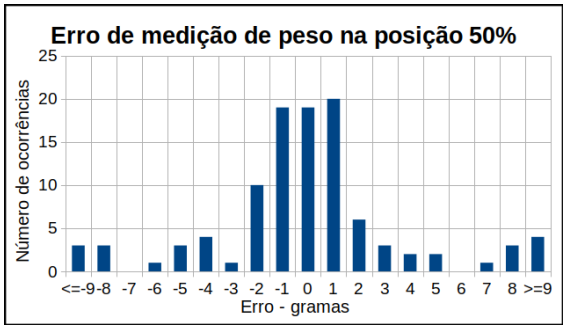



Figura 4.14: Distribuição do erro das medições do peso na posição 50%.

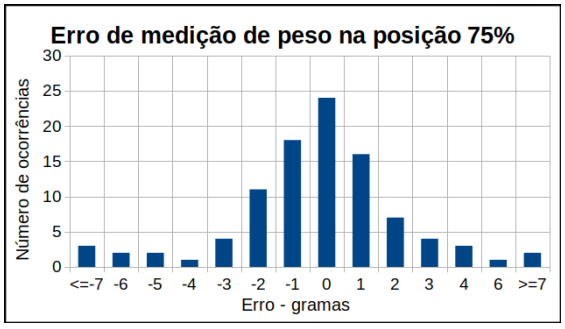



Figura 4.15: Distribuição do erro das medições do peso na posição 75%.

Nas figuras 4.16, 4.17 e 4.18 observa-se as medições de posição obtida para as posições 25%, 50% e 75% com os respetivos desvio padrão de 1.1%, 1.0% e 1.0%. Admitindo que o erro máximo é caracterizado por 3 desvios padrão temos para a posição 25% um erro de 2.0 cm e para as posições 50% e 75% um erro de 1.8 cm, para uma prateleira com 60 centímetros de comprimento.

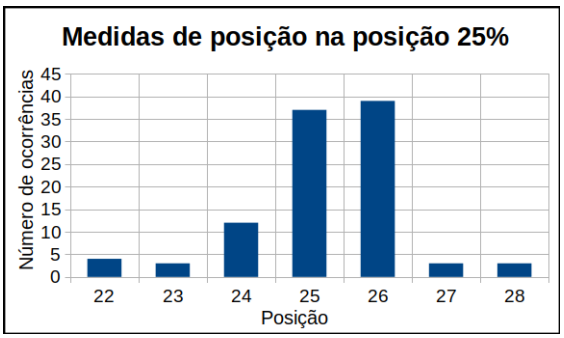



Figura 4.16: Distribuição das medições da posição na marca de 25%.

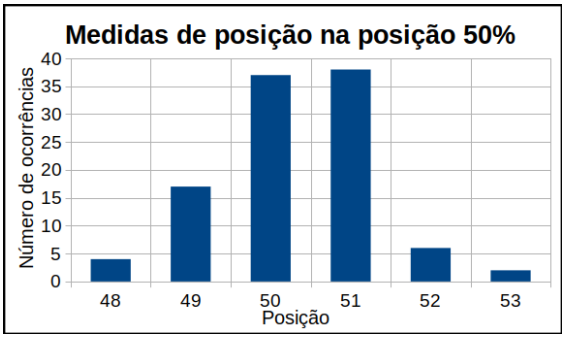



Figura 4.17: Distribuição das medições da posição na marca de 50%.

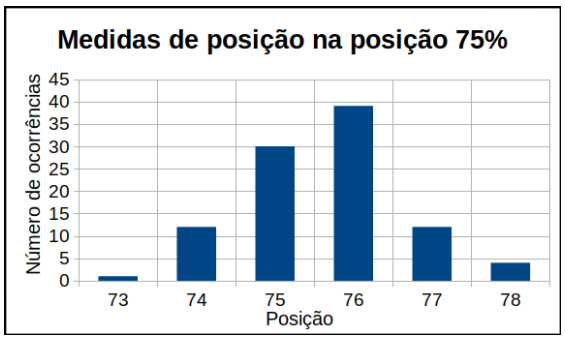



Figura 4.18: Distribuição das medições da posição na marca de 75%.

Nas figuras 4.19 e 4.20 observa-se o erro de medição da posição para os dois valores de peso utilizado, respetivamente 522 e 5074 gramas, e os seus desvio padrão de 0.9% (1.6 cm em 60 cm) e de 1.3% (2.3 cm em 60 cm). Assim, verifica-se que o objeto de maior massa causa maior erro de leitura da posição do centro de massa.

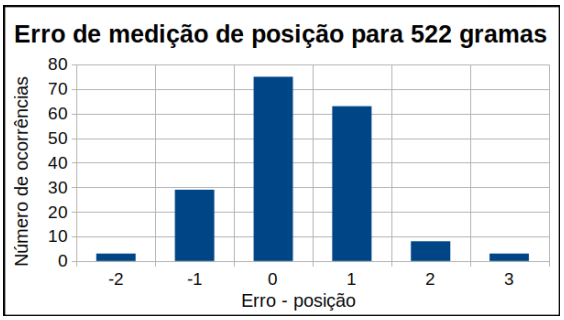



Figura 4.19: Distribuição do erro das medições da posição com um objeto de 522 gramas.

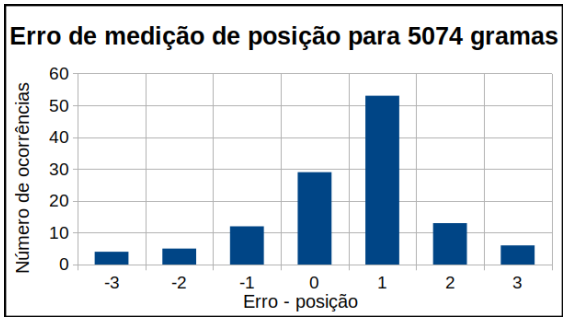



Figura 4.20: Distribuição do erro das medições da posição com um objeto de 5074 gramas.

### 4.3.3 Gestão de stock

Para esta secção foi considerado que a prateleira continha 3 produtos diferentes. Sendo eles, garrafas de água com gás de 250 mililitros, garrafas de água de 500 mililitros, e garrafas de água de 330 mililitros dispostos na prateleira segundo a representação da figura 4.21.

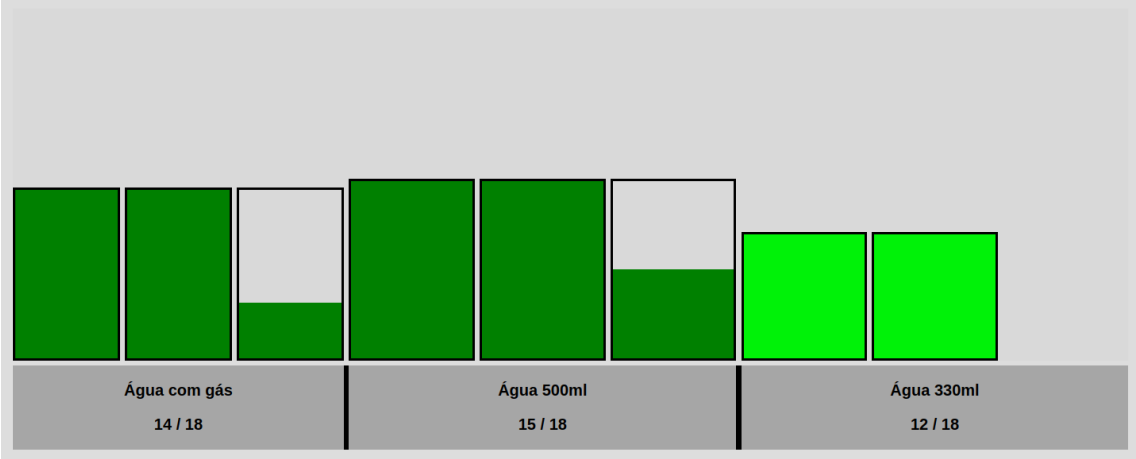



Figura 4.21: Planograma de objetos dispostos na prateleira.

**Reposição de produtos**

O reabastecimento da prateleira teve início com ela completamente vazia. O reabastecimento consistiu em 14 garrafas de água com gás, 15 garrafas de água de 500 ml e 12 garrafas de água de 330 ml. As garrafas de água com gás e as garrafas de água de 330 ml existiam em pacotes de 4 ou individualmente. As garrafas de água de 500 ml existiam em pacotes de 6 ou de forma individual. Fazendo uso da interface de reposição de produtos, o sistema determinou que o peso médio de cada produto era 426 gramas para a água com gás, 520 gramas para a água de 500 ml e 350 gramas para a água de 330 ml. No final das operações de reposição a representação da prateleira era como a da figura 4.21.

**Deteção de entrada e saída de produtos**

De forma a avaliar a capacidade de o sistema inferir corretamente os produtos associados a cada evento detetado, foram realizados 100 movimentos de produtos - especificamente foram

realizados movimentos de retirar e recolocar produtos nas posições designadas para cada um deles.

Os movimentos realizados consistiram, numa primeira fase, em retirar todos os objetos de um tipo de produto e de seguida colocá-los novamente no lugar designado. Foram realizados 42 movimentos. Numa segunda fase, os movimentos incluíam produtos de tipos diferentes e foram realizados um total de 58 movimentos. Destes, o algoritmo estimou com sucesso os produtos envolvidos em 95% dos movimentos. Verificou-se que os 5% dos casos que não foram identificados com sucesso estavam associados a movimentos que incluíam grandes quantidades de produtos, os pacotes de 4 e 6 existentes para cada tipo de produto. Dado que os eventos que apresentam maior peso, têm maior probabilidade de gerarem múltiplas combinações de produtos que resultam nas mesmas condições de peso e posição, no entanto, não correspondem à situação ocorrida na prateleira.

# Capítulo 5

# Conclusão

## 5.1 Conclusões

A presente dissertação teve como objetivo desenvolver um sistema capaz de detetar movimentações de produtos, peso e localização, na superfície de uma prateleira de supermercado de forma a permitir o controlo e gestão de stock na mesma. Para tal foi necessário, em primeiro lugar, estudar as características físicas de uma superfície apoiada em apenas dois pontos. Posteriormente, procurou-se aperfeiçoar o método de aquisição da grandeza de carga nas células de carga através de um filtro de média deslizante alterado. Seguidamente, obteve-se uma forma de detetar movimentos na superfície da prateleira e classificá-los em quatro categorias: colocar, retirar, deslocar ou indefinido, determinando o peso e a localização do movimento. De seguida, foi desenvolvido um concentrador em *software* capaz de agregar várias prateleiras, conectadas ao barramento RS485, e a aplicação *web*, esta desenvolvida na última fase.

A gestão do stock passa por inferir as quantidades de produtos que são colocados na prateleira em cada evento que a mesma deteta. Desta forma, foi desenvolvido um algoritmo capaz de obter as combinações de produtos que satisfazem as condições de peso, as condições de centro de massa e por fim, se o stock existente na prateleira ou o espaço vazio existente para cada produto permite que o evento ocorra. A última etapa passou por desenvolver uma série de métodos associados à API que permitissem a um operador visualizar o stock existente e receber alertas quando a quantidade de um dado produto fosse reduzida. Tal tem como objetivo permitir uma reposição de um dado produto de forma rápida e eficaz para minimizar o tempo em que o produto não está exposto na prateleira - colmatando assim possíveis custos por escassez de produtos. Ademais, a interface permite configurar as zonas de produtos das prateleiras. Por fim, as situações de erro são sinalizadas na interface e são disponibilizadas ações que o funcionário pode tomar de forma a corrigir o erro.

Assim sendo, verificou-se que após a escolha do hardware adequado para proteção do ADC e o processamento digital do sinal, é possível obter medições de peso com baixo nível de ruído. Em relação ao algoritmo de deteção de eventos na prateleira, verificou-se ser possível detetar movimentações de objetos com elevada precisão de peso e posição. Nomeadamente, a medição de

peso apresenta um erro máximo de $\pm 28$ gramas e a medição do centro de massa $\pm 5$ cm. Ainda, constatou-se que quando o algoritmo não era capaz de detetar dois movimentos separadamente, estes eram combinados num só, sendo o peso total do evento a soma do peso associado a cada movimento e o centro de massa resultante representa de forma correta a sobreposição dos eventos. 100% dos eventos foram classificados corretamente, sendo os eventos que envolviam a troca de produtos no centro da prateleira os mais difíceis de detetar.

O algoritmo de inferência de produtos, apresentou resultados favoráveis. Inferindo o tipo e as quantidades de produtos corretamente em 95% dos eventos detetados pela prateleira de teste. No entanto, verificou-se que a prateleira apresenta dificuldades em inferir as quantidades de produto corretas quando o peso do evento corresponde a diversas combinações de produtos. Uma outra situação em que o algoritmo mostra dificuldades em determinar o tipo e quantidade é quando a um só evento está associada uma troca de produtos.

Em suma, após o trabalho realizado é possível afirmar que efetivamente o sistema desenvolvido permite fazer a gestão de stock de prateleiras de supermercado. Apesar de, poderem ser atribuídas certas limitações ao projeto apresentado, que serão apresentadas de seguida.

**Limitações**

Na presente secção procura-se enunciar algumas limitações presentes no projeto apresentado. Primeiramente, verifica-se uma dificuldade em detetar a deslocação de objetos através da prateleira. De seguida, averigua-se uma certa falha no que diz respeito à deteção de trocas de objetos quando essas acontecem perto do centro da prateleira. Por último, é possível verificar um erro na deteção da ocorrência de eventos que envolvam simultâneamente produtos associados à prateleira e objetos estranhos.

Contudo, é possível colmatar essas mesmas limitações. No que diz respeito à primeira, essa pode ser resolvida pela introdução de divisórias de acrílico (ou outro material), entre cada tipo de produto. Desta forma não é possível deslocar um produto para a posição de outro sem que o cliente tenha de o levantar, o que seria detetado pelo sistema e acabando por gerar um alerta de produto mal colocado.

Os efeitos da segunda limitação poderão ser atenuados se a disposição dos produtos for feita de forma a que o centro da prateleira seja ocupado apenas por um tipo de produto. Assim, a ocorrência de movimentos simultâneos de colocar e retirar diferentes objetos no centro da prateleira é reduzida, pois este é composto por apenas um tipo de produto.

A terceira limitação, poderá ser revertida com pequenas alterações, nomeadamente a inclusão dos objetos estranhos no algoritmo de determinação de produtos detetados.

## 5.2 Trabalho Futuro e Perspetivas de Evolução

O sistema desenvolvido efetivamente será alvo de futuras melhorias. Uma vez que, o objetivo é torná-lo o mais eficaz possível e à prova de falhas para que possa ser implementado em superfícies comerciais. Assim, uma melhoria fundamental seria conjugar o presente projeto com a utilização de um sistema baseado em visão máquina com inteligência artificial, fazendo com que o sistema

possuísse um maior grau de robustez e confiança. No entanto, aumentando o custo global do sistema.

Para mais, poderia ser melhorada a interface de utilizador, de forma a torná-la mais responsiva. Para tal, deveria ser utilizado *WebSockets*, uma vez que, esses permitiriam ter uma melhor comunicação *browser*/servidor.

# Referências


[1] Roland Helm, Thomas Hegenbart, e Herbert Endres. Explaining costumer reactions to real stockouts. *Review of Managerial Science*, 7(3):223–246, 2013. doi:10.1007/s11846-012-0079-8.

[2] Aug 2019. URL: https://www.distribuicaoxxi.pt/planograma/.

[3] Smartshelf weighing pad, Aug 2020. URL: https://www.mt.com/au/en/home/products/Industrial_Weighing_Solutions/bench-scales/smart-shelf.html.

[4] Shelfx home. https://www.shelfx.com/. (Acedido a 02/10/2021).

[5] Wiseshelf | retail iot smart shelf solution. https://www.wiseshelf.com/. (Acedido a 02/10/2021).

[6] Amazon.com: Amazon go. https://www.amazon.com/b?ie=UTF8&node=16008589011. (Acedido a 02/10/2021).

[7] Continente labs. URL: https://labs.continente.pt/#/.

[8] Understanding the difference between cat6a cable types - warren & brown networks. https://wbnetworks.com.au/blog/understanding-the-difference-between-cat6a-cable-types. (Acedido a 02/13/2021).

[9] Omega Engineering. Load cells &; force sensors, Oct 2020. URL: https://www.omega.com/en-us/resources/load-cells.

[10] Ivan Muller, Renato Machado De Brito, Carlos Eduardo Pereira, e Valner Brusamarello. Theory and a Novel Application. *Technology*, (February), 2010.

[11] Sparkfun load cell amplifier - hx711 - sen-13879 - sparkfun electronics. https://www.sparkfun.com/products/13879. (Acedido a 02/10/2021).

[12] 12.1 conditions for static equilibrium – university physics volume 1. https://opentextbc.ca/universityphysicsv1openstax/chapter/12-1-conditions-for-static-equilibrium/. (Acedido a 02/13/2021).

[13] Signal smoothing - matlab & simulink example. https://www.mathworks.com/help/signal/ug/signal-smoothing.html. (Acedido a 02/09/2021).

[14] Yong Zhang, Yikang Gu, Vlatko Vlatkovic, e Xiaojuan Wang. Progress of smart sensor and smart sensor networks. *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, 4:3600–3606, 2004. doi:10.1109/wcica.2004.1343265.

[15] Arduino. Arduino - arduino board nano. https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano. (Acedido a 02/04/2021).

[16] Espressif Systems. Esp32 wi-fi & bluetooth mcu | espressif systems. https://www.espressif.com/en/products/socs/esp32. (Acedido a 02/04/2021).

[17] Raspberry pi 3 model b – raspberry pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. (Acedido a 02/04/2021).

[18] Line Drivers, Differential Line Drivers, e Related Parts. How Far and How Fast Can You Go with RS-485 ? páginas 1–10, 2018.

[19] Steve Corrigan. Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN). (May):1–17, 2002. URL: www.ti.com.

[20] An overview of http - http | mdn. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#client_the_user-agent. (Acedido a 02/05/2021).

[21] Http request message - http requests documentation. https://documentation.help/DogeTool-HTTP-Requests-vt/http_request.htm. (Acedido a 02/04/2021).

[22] What is an application programming interface (api)? | ibm. https://www.ibm.com/cloud/learn/api. (Acedido a 02/05/2021).

[23] Fielding dissertation: Chapter 5: Representational state transfer (rest). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. (Acedido a 02/05/2021).

[24] Slim framework, 2021. URL: https://www.slimframework.com/.

[25] Postman, 2021. URL: https://www.postman.com/.

[26] Petr Cintula, Christian G. Fermüller, e Carles Noguera. Fuzzy logic, Jul 2017. URL: https://plato.stanford.edu/entries/logic-fuzzy/.

# Anexo A

# Código Controlador

## A.1 Filtro de Média Deslizante

```
void filtroMediaDeslizante(uint8_t sensor)
{
    // Obter diferenca entre o valor medio e o novo valor lido
    int32_t diff = scale.HX711[sensor].valor_raw - scale.HX711[sensor].peso_raw;
    if (diff > threashold || diff < threashold_neg){
        scale.HX711[sensor].flag_change = scale.HX711[sensor].flag_change + 1;
    }else if (scale.HX711[sensor].flag_change != 0){
        scale.HX711[sensor].flag_change = scale.HX711[sensor].flag_change - 1;
    }
    if (scale.HX711[sensor].flag_change == max_change){
        scale.HX711[sensor].count_replace = change_count;
        scale.HX711[sensor].flag_change = 0;
    }
    else if (scale.HX711[(sensor + 1) % 2].count_replace == change_count_one){
        scale.HX711[sensor].count_replace = change_count - 1;
        scale.HX711[sensor].flag_change = 0;
    }
    if (scale.HX711[sensor].count_replace > 0){
        for (uint8_t i = 0; i < MOVING_AVERAGE_SIZE; i++){
            scale.HX711[sensor].movAvg.values[i] = scale.HX711[sensor].valor_raw;
        }
        scale.HX711[sensor].count_replace = scale.HX711[sensor].count_replace - 1;
        scale.HX711[sensor].movAvg.nextPos = 0;
        scale.HX711[sensor].movAvg.sum = scale.HX711[sensor].valor_raw *
    MOVING_AVERAGE_SIZE;
        scale.HX711[sensor].peso_raw = scale.HX711[sensor].valor_raw;
    }else{
        if (scale.HX711[sensor].movAvg.numberOfReadings < MOVING_AVERAGE_SIZE){
            scale.HX711[sensor].movAvg.sum = scale.HX711[sensor].movAvg.sum + scale
    .HX711[sensor].valor_raw;

            scale.HX711[sensor].movAvg.numberOfReadings++;
```

```
        }else{
            scale.HX711[sensor].movAvg.sum = scale.HX711[sensor].movAvg.sum + scale.HX711[sensor].valor_raw - scale.HX711[sensor].movAvg.values[scale.HX711[sensor].movAvg.nextPos];
        }
        scale.HX711[sensor].movAvg.values[scale.HX711[sensor].movAvg.nextPos] = scale.HX711[sensor].valor_raw;
        scale.HX711[sensor].movAvg.nextPos++;
        if (scale.HX711[sensor].movAvg.nextPos >= MOVING_AVERAGE_SIZE){
            scale.HX711[sensor].movAvg.nextPos = 0;
        }
        scale.HX711[sensor].peso_raw = (scale.HX711[sensor].movAvg.sum) / (scale.HX711[sensor].movAvg.numberOfReadings);
    }
    if (scale.HX711[sensor].peso_raw >= scale.HX711[sensor].peso_raw_anterior)
        scale.HX711[sensor].peso_raw_diff1 = scale.HX711[sensor].peso_raw - scale.HX711[sensor].peso_raw_anterior;
    else
        scale.HX711[sensor].peso_raw_diff1 = scale.HX711[sensor].peso_raw_anterior - scale.HX711[sensor].peso_raw;

    scale.HX711[sensor].peso_raw_anterior = scale.HX711[sensor].peso_raw;
}
```

## A.2 Deteção de Eventos

```
void detetarEvento(){
    uint8_t eventoNovo = 0;
    uint8_t eventoIntermedioNovo = 0;
    if (scale.event_f.estado == INIT){  // Estado inicial
        if (scale.characteristc == Estavel){
            scale.event_f.estado = 0;
        }
    }else if (scale.event_f.estado == 0){  // Estado 0
        if (scale.characteristc == Estavel){
            scale.event_f.novoEvento = 0;
            scale.event_f.ultimoPesoEstavel = scale.peso_total;
            scale.event_f.ultimoPesoEstavelA = scale.HX711[sensor_A].peso;
            scale.event_f.ultimoPesoEstavelB = scale.HX711[sensor_B].peso;
            scale.event_f.ultimaPosicaoEstavel = scale.position;
            scale.event_f.flagNovoEvento = 0;
            scale.event_f.duration = 0;
        }else if (scale.characteristc == PoucoEstavel){
            scale.event_f.estado = 1;
            scale.event_f.score = 0;
            scale.event_f.flagNovoEvento = 1;
            scale.event_f.countEstavel = 0;
            scale.event_f.duration++;
```

```
23          }else if (scale.characteristc == Instavel){
24              scale.event_f.estado = 2;
25              scale.event_f.score = 0;
26              scale.event_f.flagNovoEvento = 1;
27              scale.event_f.countEstavel = 0;
28              scale.event_f.duration++;
29          }
30      }else if (scale.event_f.estado == 1){  // Estado 1
31          if (scale.characteristc == Estavel){
32              scale.event_f.countEstavel++;
33              scale.event_f.duration++;
34              if (scale.event_f.countEstavel >= MIN_EVENT_STABLE){
35                  eventoNovo = 1;
36                  scale.event_f.estado = 0;
37              }
38          }else if (scale.characteristc == Instavel){
39              scale.event_f.estado = 2;
40              scale.event_f.countEstavel = 0;
41              scale.event_f.duration++;
42          }else if (scale.characteristc == PoucoEstavel){
43              scale.event_f.duration++;
44              scale.event_f.countEstavel = 0;
45          }
46      }else if (scale.event_f.estado == 2){   // Estado 2
47          if (scale.characteristc == Estavel){
48              scale.event_f.countEstavel++;
49              scale.event_f.duration++;
50              if (scale.event_f.countEstavel >= MIN_EVENT_STABLE){
51                  eventoNovo = 1;
52                  scale.event_f.estado = 0;
53              }
54          }else if (scale.characteristc == PoucoEstavel){
55              scale.event_f.estado = 3;
56              scale.event_f.score++;
57              scale.event_f.countEstavel = 0;
58              scale.event_f.duration++;
59          }else if (scale.characteristc == Instavel){
60              scale.event_f.duration++;
61          }
62      }else if (scale.event_f.estado == 3){    // Estado 3
63          if (scale.characteristc == Estavel){
64              scale.event_f.countEstavel++;
65              scale.event_f.duration++;
66              if (scale.event_f.countEstavel >= MIN_EVENT_STABLE){
67                  eventoNovo = 1;
68                  scale.event_f.estado = 0;
69              }
70          }else if (scale.characteristc == PoucoEstavel){
71              scale.event_f.score++;
```

```
72            scale.event_f.countEstavel = 0;
73            scale.event_f.duration++;
74        }else if (scale.characteristc == Instavel){
75            scale.event_f.estado = 2;
76            scale.event_f.countEstavel = 0;
77            scale.event_f.duration++;
78
79            if (scale.event_f.countEstavel > 0 || scale.event_f.score >=
    MIN_EVENT_SCORE){
80                eventoNovo = 1;
81                eventoIntermedioNovo = 1;
82            }
83        }
84    }
85
86    if (eventoNovo == 1 || eventoIntermedioNovo == 1){  // Obter dados do evento
87
88        scale.event_f.novoEvento = 1;
89
90        if ((scale.HX711[sensor_A].peso - scale.event_f.ultimoPesoEstavelA) * (
    scale.HX711[sensor_B].peso - scale.event_f.ultimoPesoEstavelB) < 0){
91            scale.event_f.event.alavanca = 1;
92        }else{
93            scale.event_f.event.alavanca = 0;
94        }
95        scale.event_f.event.duration = scale.event_f.duration;
96
97        if (eventoNovo == 1 && eventoIntermedioNovo == 0)
98            scale.event_f.event.certeza = 255;
99        else
100           scale.event_f.event.certeza = scale.event_f.score;
101
102       scale.event_f.event.posicaoInicial = scale.event_f.ultimaPosicaoEstavel;
103       scale.event_f.event.pesoInicial = scale.event_f.ultimoPesoEstavel;
104
105       if (eventoIntermedioNovo == 0){
106           scale.event_f.event.pesoFinal = scale.peso_total;
107           scale.event_f.event.posicaoFinal = scale.position;
108           scale.event_f.event.peso = scale.event_f.event.pesoFinal - scale.
    event_f.event.pesoInicial;
109           scale.event_f.event.posicao = ((scale.HX711[sensor_B].peso - scale.
    event_f.ultimoPesoEstavelB) * 100) / scale.event_f.event.peso;
110
111           scale.event_f.ultimoPesoEstavel = scale.peso_total;
112           scale.event_f.ultimoPesoEstavelA = scale.HX711[sensor_A].peso;
113           scale.event_f.ultimoPesoEstavelB = scale.HX711[sensor_B].peso;
114           scale.event_f.ultimaPosicaoEstavel = scale.position;
115       }else if (eventoIntermedioNovo == 1){
116           scale.event_f.event.pesoFinal = scale.event_f.ultimoPeso;
```

```
117             scale.event_f.event.posicaoFinal = scale.position;
118             scale.event_f.event.peso = scale.event_f.event.pesoFinal - scale.
    event_f.event.pesoInicial;
119             scale.event_f.event.posicao = ((scale.event_f.ultimoPesoB - scale.
    event_f.ultimoPesoEstavelB) * 100) / scale.event_f.event.peso;
120 
121             scale.event_f.ultimoPesoEstavel = scale.event_f.ultimoPeso;
122             scale.event_f.ultimoPesoEstavelA = scale.event_f.ultimoPesoA;
123             scale.event_f.ultimoPesoEstavelB = scale.event_f.ultimoPesoB;
124             scale.event_f.ultimaPosicaoEstavel = scale.event_f.ultimaPosicao;
125         }
126         if (scale.event_f.event.peso >= 0)
127             scale.event_f.event.evento = 1;
128         else{
129             scale.event_f.event.evento = 2;
130             scale.event_f.event.peso = scale.event_f.event.peso * -1;
131         }
132 
133         if (scale.event_f.event.peso <= 15){
134             if (scale.event_f.event.posicaoFinal != scale.event_f.event.
    posicaoInicial){
135                 scale.event_f.event.evento = 3;
136             }else{
137                 scale.event_f.novoEvento = 0;
138             }
139         }
140     }
141     // Atualizar as variaveis dos valores anteriores
142     scale.event_f.ultimoPeso = scale.peso_total;
143     scale.event_f.ultimoPesoA = scale.HX711[sensor_A].peso;
144     scale.event_f.ultimoPesoB = scale.HX711[sensor_B].peso;
145     scale.event_f.ultimaPosicao = scale.position;
146 }
```

# Anexo B

# Algoritmo de Inferência de Produtos

## B.1 Determinar Combinações

```
/**
 * @description Obter as combinacoes que satisfazem a condicao de peso
 *
 * @param  array $items contem a combinacao inicial e os dados associados a cada
    produto
 *
 * @return array
 */
private function obterCombinacoesComBaseNoPeso($items){
    $data = $items['data'];
    unset($items['data']);

    $combinations = array();
    $total_combinations = 0;
    $cycle_count = 5000000;
    $total_cycles = 0;
    $overweight = 0;
    $underweight = 0;

    while ($cycle_count > 0) {
        $total_cycles++;
        $combination_weight_l = -1 * $this->data->scale_weight_var;
        $combination_weight = 0;
        $combination_weight_h = $this->data->scale_weight_var;
        $aux_stock = array();
        $first = true;
        $change = 0;
        $actualSignal = 0;
        $lastSignal = 0;
        foreach ($items['combination'] as $key => $combination) {
            $combination_weight_l += $combination * $items[$key]['min_weight'];
            $combination_weight += $items[$key]['weight'] * $combination;
```

```
32            $combination_weight_h += $combination * $items[$key]['max_weight'];
33
34            if (isset($aux_stock[$items[$key]['planogram_key']]) == false) {
35                $aux_stock[$items[$key]['planogram_key']] = 0;
36            }
37            $aux_stock[$items[$key]['planogram_key']] += $combination;
38            if ($first == true) {
39                $lastSignal = $combination >= 0;
40                $first = false;
41            } else {
42                $actualSignal = $combination >= 0;
43                if ($actualSignal != $lastSignal) {
44                    $change++;
45                }
46                $lastSignal = $actualSignal;
47            }
48        }
49        $combination_weight_error = abs($this->data->weight - $combination_weight);
50
51        if ($this->data->weight >= $combination_weight_l && $this->data->weight <=
    $combination_weight_h) {
52            // o peso da combinacao aproxima-se do peso do evento - guardar a
    combinacao
53            $combinations[$total_combinations] = $items['combination'];
54            $combinations[$total_combinations]['combination_weight_l'] =
    $combination_weight_l;
55            $combinations[$total_combinations]['combination_weight'] =
    $combination_weight;
56            $combinations[$total_combinations]['combination_weight_h'] =
    $combination_weight_h;
57            $combinations[$total_combinations]['weight_error'] =
    $combination_weight_error;
58            $combinations[$total_combinations]['signal_change'] = $change;
59            $total_combinations++;
60
61            $overweight = 0;
62            $underweight = 0;
63        } else if ($combination_weight_l > $this->data->weight) {
64            // overweight - obter a proxima quantidade que se aproxima ao peso do
    evento
65            $overweight++;
66            $underweight = 0;
67            for ($j = count($items['combination']) - 1; $j >= count($items['
    combination']) - $overweight; $j--) {
68                $items['combination'][$j] = $items[$j]['max_capacity'];
69            }
70        } else if ($combination_weight_h < $this->data->weight) {
71            // underweight - obter a proxima quantidade que se aproxima ao peso do
    evento
```

```
72              $underweight++;
73              $overweight = 0;
74              $weight_difference = $this->data->weight - $combination_weight;
75              $j = $data['subplanogram_count'] - 1;
76              if ($items[$j]['weight'] != 0) {
77                  $aux_products = round($weight_difference / $items[$j]['weight']) -
    1;
78                  if ($items['combination'][$j] < 0) {
79                      $aux_products += $items['combination'][$j];
80                  }
81              } else {
82                  $aux_products = 0;
83              }
84              if ($aux_products <= $items[$j]['max_capacity'] && $aux_products >=
    $items['combination'][$j]) {
85                  $items['combination'][$j] = $aux_products;
86              } else {
87                  $items['combination'][$j] = $items[$j]['max_capacity'];
88              }
89          } else {
90              $overweight = 0;
91              $underweight = 0;
92          }
93          // Determinar a proxima combinacao
94          $j = count($items['combination']) - 1;
95          $items['combination'][$j]++;
96          for ($k = $j; $k > 0; $k--) {
97              if ($items['combination'][$k] > $items[$k]['max_capacity']) {
98                  $items['combination'][$k] = $items[$k]['min_capacity'];
99                  $kj = $k - 1;
100                 $items['combination'][$kj]++;
101             }
102         }
103         $k = 0;
104         if ($items['combination'][$k] > $items[$k]['max_capacity']) {
105             $cycle_count = 0;
106         }
107         $cycle_count--;
108     }
109     $combinations['data'] = $data;
110     return $combinations;
111 }
```