**Universidade de Aveiro**
**2013**
Departamento de Eletrónica, Telecomunicações e Informática

**Tiago Filipe Santos Coelho**

# Algoritmos Inteligentes de Baixa Complexidade

**Universidade de Aveiro**
**2013**
Departamento de Eletrónica, Telecomunicações e Informática

**Tiago Filipe Santos Coelho**

# Algoritmos Inteligentes de Baixa Complexidade

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor Armando J. Pinho e da Professora Doutora Ana Maria Tomé, Professores Associados do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

**o júri**

presidente

Professor Doutor Luís Filipe de Seabra Lopes

Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais

Professor Doutor Armando José Formoso de Pinho

Professor Associado com Agregação do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)

Professora Doutora Dulce Helena Carvalho Coelho

Professora Adjunta do Departamento de Engenharia Eletrotécnica do Instituto Superior de Engenharia de Coimbra

**agradecimentos**

Os meus mais sinceros agradecimentos ao Professor Doutor Armando Pinho e Professora Doutora Ana Maria Tomé, por toda a motivação, disponibilidade e orientação prestada. Prezo a valiosa experiência transmitida e todos os conhecimentos e partilhados, assim como as críticas e sugestões efetuadas ao longo do desenvolvimento deste trabalho de dissertação.

Expresso igualmente a minha gratidão para com o Engenheiro Nelson Ferreira, representante da Bosch Termotecnologia SA, não só pela oportunidade de trabalhar nesta dissertação e de visitar as instalações da Vulcano em Aveiro, mas também pelas contribuições oportunas para o progresso deste trabalho.

Aos meus amigos e colegas, com quem tive a chance de partilhar opiniões, experiências e boa disposição, agradeço-vos o companheirismo, companhia e amizade demonstrada, e desejo-vos sucesso para o futuro.
Aos professores e a todos os que, de forma direta ou indireta, ao longo de todo o percurso universitário, contribuíram para a minha realização académica e pessoal, um sentido Obrigado.

Por fim e acima de tudo, estou profundamente grato a toda a minha família, em especial ao meu pai José Coelho, à minha mãe Etelvina, à minha irmã Catarina, ao meu padrinho António Coelho, à minha Tia Carla e aos meus avós (Ana, José e João) pelo amor, compreensão e valores transmitidos.
Espero que com o desfecho desta etapa na minha vida, possa de alguma forma retribuir e compensar todo o carinho, apoio e dedicação que constantemente me dão.

**palavras-chave**

Algoritmos inteligentes, microcontrolador de 8 bits, domótica, sistema doméstico de aquecimento de água elétrico, simulação, eficiência energética.

**resumo**

A domótica é uma área com grande interesse e margem de exploração, que pretende alcançar a gestão automática e autónoma de recursos habitacionais, proporcionando um maior conforto aos utilizadores. Para além disso, cada vez mais se procuram incluir benefícios económicos e ambientais neste conceito, por forma a garantir um futuro sustentável.
O aquecimento de água (por meios elétricos) é um dos fatores que mais contribui para o consumo de energia total de uma residência.

Neste enquadramento surge o tema "algoritmos inteligentes de baixa complexidade", com origem numa parceria entre o Departamento de Eletrónica, Telecomunicações e Informática (DETI) da Universidade de Aveiro e a Bosch Termotecnologia SA, que visa o desenvolvimento de algoritmos ditos "inteligentes", isto é, com alguma capacidade de aprendizagem e funcionamento autónomo.
Os algoritmos devem ser adaptados a unidades de processamento de 8 bits para equipar pequenos aparelhos domésticos, mais propriamente tanques de aquecimento elétrico de água. Uma porção do desafio está, por isso, relacionada com as restrições computacionais de microcontroladores de 8 bits.

No caso específico deste trabalho, foi determinada a existência de sensores de temperatura da água no tanque como a única fonte de informação externa aos algoritmos, juntamente com parâmetros pré-definidos pelo utilizador que estabelecem os limiares de temperatura máxima e mínima da água.
Partindo deste princípio, os algoritmos desenvolvidos baseiam-se no perfil de consumo de água quente, observado ao longo de cada semana, para tentar prever futuras tiragens de água e, consequentemente, agir de forma adequada, adiantando ou adiando o aquecimento da água do tanque. O objetivo é alcançar uma gestão vantajosa entre a economia de energia e o conforto do utilizador (água quente), isto sem que exista necessidade de intervenção direta por parte do utilizador final.
A solução prevista inclui também o desenvolvimento de um simulador que permite observar, avaliar e comparar o desempenho dos algoritmos desenvolvidos.

**keywords**

Intelligent algorithms, 8-bit microcontroller, home automation, domestic electric water heater, simulation, energy efficiency.

**abstract**

Home automation is an area of great interest and growth margin, which aims to achieve autonomous control of housing resources, providing greater comfort to the users. In addition, this concept is increasingly covering economic and environmental concerns, in order to ensure a sustainable future.
Water heating (using electric resources) is one of the factors that contribute the most to the total energy consumption of a residence.

In this context arises the subject “intelligent algorithms with low complexity”, originated in a partnership between the Department of Electronics, Telecommunications and Informatics (DETI) of the University of Aveiro and Bosch Thermotechnology SA, which targets the development of algorithms entitled "intelligent", in other words, with some capability for learning and autonomous functioning.
The algorithms must be adapted to 8-bit processing units in order to equip small appliances, more precisely domestic electric water heaters. A portion of the challenge is therefore related to the computational constraints of 8-bit microcontrollers.

In the specific case of this work, it was determined the existence of temperature sensors in the water tank as the only source of information external to the algorithms, along with pre-set user-defined thresholds which establish the maximum and minimum water temperature values.
On this basis, in order to predict future drawings of water and thus take appropriate action, anticipating or delaying the heating of the tank’s water, the developed algorithms use information from the hot water consumption profile, observed throughout each week. The goal is to achieve an advantageous management between energy savings and user comfort (hot water), with no need for direct intervention by the end user.
The solution conceived also includes the development of a simulator that allows us to observe, evaluate and compare the performance of the algorithms created.

# Conteúdo

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

Esta dissertação, com origem numa parceria entre o Departamento de Eletrónica, Telecomunicações e Informática (DETI) da Universidade de Aveiro e a Bosch Termotecnologia SA, visa o desenvolvimento de algoritmos ditos "inteligentes", isto é, com alguma capacidade de aprendizagem e funcionamento autónomo. Os algoritmos devem ser adaptados a unidades de processamento de 8 bits para equipar pequenos aparelhos domésticos, mais propriamente tanques de aquecimento elétrico de água. Uma porção do desafio está, por isso, relacionada com as restrições computacionais de microcontroladores de 8 bits. No âmbito deste trabalho, foi designada a plataforma *Arduino* Mega 2560 como referência ao tipo de processadores alvo.

## 1.1 Motivação e Objetivos

Algoritmos de aprendizagem automática são um tema recorrente no campo da inteligência artificial, existindo vários métodos para extrair modelos ou detetar padrões, que muitas vezes nem são aparentes para um ser humano, através do relacionamento de informação proveniente de um aglomerado de dados. Isto permite que, programaticamente, algoritmos sejam capazes de ajustar e aperfeiçoar o seu comportamento para melhor desempenhar a função a que se destinam. Conseguir o mesmo efeito através de algoritmos com complexidade reduzida, aplicados em *hardware* com recursos minimalistas, é desde logo um desafio interessante.

Ter em conta que estes algoritmos podem ser postos ao serviço do progresso da automação juntamente com a consciencialização da necessidade de economizar energia, não só por razões económicas mas também por razões ambientais, contribui para o futuro nesta área de desenvolvimento. Este tema mostra-se assim relevante sob o ponto de vista da domótica, com a potencialidade para integração em casas inteligentes, e também para aplicação em sistemas domésticos comuns, uma vez que envolve o melhoramento de sistemas de aquecimento de água já existentes, mas com a finalidade de oferecer um melhor compromisso entre o conforto e a economia de energia.

O principal objetivo proposto prevê então o desenvolvimento de algoritmos que sejam capazes de tomar a iniciativa de ligar ou desligar a resistência de aquecimento da água do tanque, sem que seja necessária uma intervenção direta do utilizador final, com vista a uma gestão vantajosa entre a poupança de energia e o conforto do utilizador (água quente). A tomada de decisão dos algoritmos pode ter em conta várias fontes de informação, tais como a tarifa energética, a estação do ano, a previsão meteorológica e o perfil de consumo de água

quente. No caso específico do trabalho desenvolvido, foi determinada a existência de sensores de temperatura da água no tanque como a única fonte de informação externa aos algoritmos, juntamente com parâmetros pré-definidos pelo utilizador que estabelecem os limiares de temperatura máxima e mínima da água.

Contudo, para a concretização deste objetivo primário, está pressuposta a criação de ferramentas de auxílio ao desenvolvimento e teste dos algoritmos. O plano de trabalho foi então dividido no seguinte conjunto de tarefas:

- Estudo de um modelo matemático que represente a temperatura da água num tanque de aquecimento elétrico, ao longo do tempo, tendo em conta as suas perdas térmicas e o consumo de água;
- Criação de uma figura de mérito para avaliação dos algoritmos;
- Desenvolvimento de uma ferramenta visual para simulação, que incorpore a implementação do modelo e a figura de mérito;
- Desenvolvimento de algoritmos "inteligentes":
  - Algoritmos de decisão (ligar/desligar a resistência de aquecimento);
  - Algoritmos de deteção de consumo de água;
- Migração para a plataforma *Arduino*.

## 1.2 Sistemas de Aquecimento de Água

Quando se aborda o aquecimento de água ao nível doméstico, existe uma grande variedade de equipamentos com características distintas. Por um lado, existem os sistemas de aquecimento de água instantâneos, isto é, sem tanque/repositório de água, em que o fluxo de água é aquecido no momento de tiragem. Em geral, este tipo de sistemas usa combustíveis fosseis para aquecer a água, tal como acontece com os sistemas a gás natural comuns. Por outro lado, temos os sistemas equipados com um tanque onde é armazenada e aquecida a água antes de ser usada. É a esta categoria que pertencem os sistemas de aquecimento solar e a maioria dos sistemas de aquecimento elétrico [1] [2].

O sistema ideal depende das necessidades e características de utilização de cada residência. Com um uso esporádico e reduzido de água quente, os sistemas a gás tendem a ser a opção mais económica, enquanto que quando se trata de um consumo mais intensivo de água quente, os sistemas elétricos passam a ser opções viáveis [2]. No âmbito desta dissertação, o alvo de interesse são os sistemas elétricos (com tanque).

Uma questão importante que não pode ser ignorada nos sistemas providos de tanque está relacionada com potências riscos para a saúde humana. Estudos revelam que, em certas circunstâncias, a água destes tanques está sujeita a problemas de contaminação bacterial, sendo a bactéria mais comum neste tipo de contaminação a *Legionella pneumophila*. Água estagnada e certos materiais podem promover o seu desenvolvimento, mas o fator mais relevante para a sobrevivência da bactéria advém da temperatura da água [3].

O intervalo de temperatura mais favorável à proliferação desta bactéria varia entre os $25°C$ e os $47°C$, sendo o auge do seu crescimento a $37°C$. Uma das razões que leva a que os tanques

tenham um limiar mínimo de temperatura, normalmente superior a $50°C$, deve-se ao facto de a partir dessa temperatura a bactéria não conseguir sobreviver, eliminando/reduzindo assim os potenciais problemas de contaminação bacterial [3].

### 1.2.1 Sistemas Tradicionais vs. Sistemas Inteligentes

Os sistemas considerados tradicionais são caracterizados por ter um comportamento binário de ligar ou desligar a resistência, recorrendo simplesmente a um termostato para manter a temperatura da água num dado intervalo de temperatura. O procedimento é por isso puramente reativo, consistindo em ligar a resistência se a temperatura atingir o limite mínimo estabelecido e desligar a mesma se a temperatura alcançar o valor máximo do intervalo estabelecido.

O mesmo efeito é facilmente reproduzido com recurso a sensores de temperatura e uma unidade de controlo programável que implemente a logica de decisão. Esta alternativa torna-se bastante interessante devido ao baixo custo do *hardware* e à potencialidade de inovação que oferece.

Mesmo com o crescimento da adoção de energias renováveis, o impacto ambiental devido ao consumo de energia ainda é bastante significativo. Foi estimado que, nas residências equipadas com estes sistemas elétricos, o consumo resultante do aquecimento de água pode chegar aos 30% do total de energia consumida numa residência [4]. O efeito negativo torna-se ainda mais evidente se for tido em conta que a distribuição do consumo de energia não é uniforme ao longo do dia, originando picos de consumo durante determinadas horas. A contribuição por parte dos tanques elétricos para estes picos de consumo manifesta-se com mais intensidade nas estações frias do ano [4] [5].

Existem vários estudos nesta área que visam minimizar este problema, através de estratégias de *Demand-Side Management* (DSM) denominadas *peak clipping/shaving* e *valley filling* [6]. Estas estratégias baseiam-se na capacidade dos tanques armazenarem energia na forma de calor (com perdas lentas), para assim desviar temporalmente os consumos de eletricidade destes, dos instantes onde se observam maiores picos de consumo, para os intervalos menos concorridos [4] [5] [6]. Os sistemas ditos inteligentes são caracterizados por implementar algoritmos desta natureza.

Para um algoritmo ser considerado "inteligente" tem que possuir capacidade autónoma para recolha de informação, aprendizagem e tomada de decisão, deixando esta última de ser meramente reativa. A finalidade dos algoritmos é providenciar conforto, ou seja, água quente, e ao mesmo tempo realizar uma gestão economizadora de energia.

As estratégias a explorar nesta situação não se limitam apenas a DSM. Enquanto estas implicam o conhecimento sobre a tarifa energética ou o consumo energético total da residência (ou da população em geral), o uso de informação de outras origens pode ser também relevante, nomeadamente o conhecimento sobre o dia do ano, a temperatura do ar exterior e da água do tanque, o perfil de consumo habitual de água quente, a previsão meteorológica, etc. No contexto da domótica, pode ainda haver a contribuição de informação relevante por parte de outros aparelhos inteligentes (e.g., consumos agendados) [7] [8].

Nesta dissertação, a abordagem ao desenvolvimento dos algoritmos "inteligentes" é focada na limitação da existência apenas de sensores de temperatura, decorrendo daí que a observação

e aprendizagem do perfil de consumo de água e consequentes decisões autónomas sobre o controlo da resistência resultam da variação da temperatura da água do tanque (e dos limites de temperatura pré-estabelecidos).

## 1.3 Estrutura do Documento

Este primeiro capítulo destina-se ao enquadramento do tema desta dissertação, onde é justificada a motivação que levou a sua execução e onde são apresentados os objetivos propostos no contexto da mesma. É ainda realizada uma breve análise aos sistemas domésticos de aquecimento elétrico de água tradicionais e é exposto o interesse e a potencialidade de sistemas equivalentes que, neste âmbito, possam ser considerados "inteligentes".

No segundo capítulo é apresentado o modelo matemático usado para descrever a evolução da temperatura da água num tanque de água com aquecimento elétrico. Adicionalmente, é explicada a figura de mérito desenvolvida para ajudar na avaliação e comparação da eficácia dos vários algoritmos criados.

Os detalhes sobre a implementação e funcionamento dos algoritmos desenvolvidos no contexto deste trabalho são apresentados no Capítulo 3, onde são abordados algoritmos de decisão, responsáveis por deliberar quando aquecer a água, e o algoritmo de deteção de utilização de água quente, concebido como componente dos algoritmos de decisão.

O Capítulo 4 é dedicado à implementação do simulador do tanque de aquecimento elétrico de água e todas as suas funcionalidades, inclusive a integração da figura de mérito. É igualmente apresentada a interface gráfica do simulador, é exposta a formatação dos vários ficheiros (perfis) necessários ao programa de simulação, e é explicado o funcionamento em modo "servidor" e o respetivo protocolo de comunicação.

As particularidades envolvidas na implementação dos algoritmos na plataforma de desenvolvimento são explanadas no Capítulo 5, onde são também apresentadas as limitações do *hardware* em questão, seguindo-se os detalhes relativos à migração do código dos algoritmos desenvolvidos.

No último capítulo, é demonstrada a validade dos algoritmos criados e são revelados os resultados ao nível de desempenho e consumo de energia destes, para várias situações simuladas. Para finalizar, são expostas as conclusões do trabalho desenvolvido, abordando os seus pontos fortes assim como as suas lacunas, deixando ainda apontamentos para futuros progressos nesta área.

# Capítulo 2

# Desenvolvimento de Métodos de Teste e Avaliação

Inicialmente, foi necessário conceber uma estrutura de suporte ao desenvolvimento dos algoritmos "inteligentes" que permita tirar conclusões sobre a sua eficácia. Neste capítulo são apresentados os métodos utilizados para a modelação dos sistemas em causa e para avaliação dos algoritmos implementados. Estes métodos serão posteriormente aplicados na construção da ferramenta da simulação.

## 2.1 Modelo do Tanque Elétrico de Aquecimento de Água

Devido à importância de testar, validar e comparar os possíveis algoritmos em tempo útil, é essencial a construção de um simulador de um sistema de aquecimento de água elétrico, que permita prever a evolução da temperatura da água num tanque.

Surge então a necessidade de encontrar um modelo coerente e realista que tome em consideração as características do tanque (e.g. volume), a perda de energia (calor) ao longo do tempo, a entrada de água fria, e o perfil de utilização de água (ditado pela saída de água quente do tanque).

O que se pretende obter do modelo é a representação, por meio de equações matemáticas, da temperatura da água no tanque ao longo do tempo, para posteriormente realizar programaticamente simulações dos vários algoritmos face às mais variadas situações.

### 2.1.1 Modelo Estudado - *Domestic Electric Water Heater*

O sistema doméstico de aquecimento de água elétrico, referido na literatura como *Domestic Electric Water Heater* (DEWH), caracteriza com precisão o tipo de sistema alvo desta dissertação. Observando a Figura 2.1, pode ver-se que este sistema contempla a entrada de água fria e a saída de água quente, respetivamente pela parte inferior e superior do contentor de água, e o aquecimento da água por via elétrica com resistências de aquecimento. É ainda assinalada a transferência de energia que ocorre entre a água quente presente no tanque e a água fria que entra no tanque $[B(T-T_{in})]$, e a dissipação de calor para o exterior $[G(T-T_{out})]$ [9][10] [11].

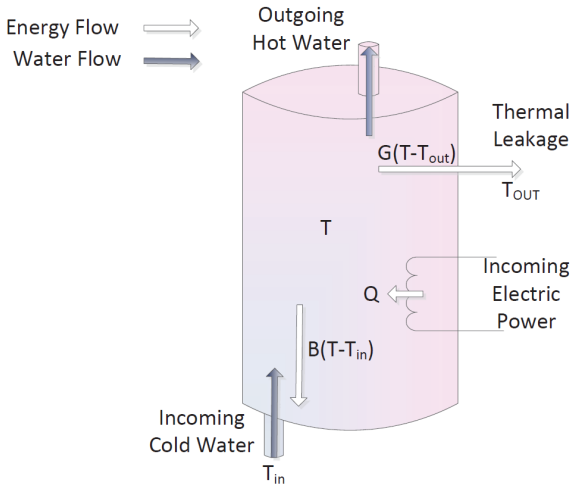


Figura 2.1: Esquema do fluxo de água e de energia num *Domestic Electric Water Heater* (figura adaptada de [9]).

O modelo em si é baseado na análise do fluxo de energia e permite prever a temperatura da água ao longo do tempo, descrevendo as características de transferência de calor num tanque DEWH [10] [11].

A restrição mais destacada do modelo é que este assume que a temperatura da água é uniforme em todo o tanque, significando isto que a transferência de calor na água é tida como instantânea [9]. Tomando como exemplo a situação de entrada de água fria no tanque, por interpretação do modelo, a descida de temperatura da água seria imediatamente sentida em todo o tanque quando na realidade, devido às características da transferência de calor entre a água, este seria um processo gradual.

### 2.1.2 Caracterização do Modelo Matemático

Recorrendo à mesma notação apresentada na referência bibliográfica, sendo $t$ uma variável representativa do tempo, tem-se que $T(t)$ é a temperatura da água ($^{\circ}C$), $T_{in}$ é a temperatura da água de entrada, $T_{out}$ é a temperatura do ambiente exterior, $Q(t)$ é a taxa de entrada de energia ($W$) e $W_D(t)$ representa a utilização de água ($m^3/s$). Relativamente às constantes que caracterizam o tanque, tem-se que $A$ é a área da sua superfície ($m^2$), $V$ é o seu volume ($m^3$) e $R$ é a resistência térmica intrínseca ($m^2\,^{\circ}C/W$). Por fim, $\rho$ é a densidade da água ($kg/m^3$) e $c_p$ é o calor específico da água ($J/^{\circ}C\,kg$) [9].

É de notar que, embora na especificação do modelo as variáveis $T_{in}$ e $T_{out}$ sejam tidas como constantes, na implementação do modelo no simulador estas podem ser consideradas variáveis com um período diário, para assim oferecer maior controlo sobre as situações simuladas.

A equação que descreve a temperatura da água no tanque ao longo do tempo é

$$\begin{gathered} CT(t) = Q(t) + G(T_{out} - T(t)) + HW_D(t)(T_{in} - T(t)) \\ C = \rho c_p V, H = \rho c_p, G = A/R \end{gathered} . \tag{2.1}$$

Assumindo $Q(t) = Q$ e $W_D(t) = W_D$ constantes no intervalo de tempo $t \in [t_0, t_f]$, a solução de (2.1) é dada por [9]

$$\begin{gathered} T(t) = T(t_0)e^{-(t-t_0)/\tau} + K(1 - e^{-(t-t_0)/\tau}) \\ \tau = \frac{C}{G + HW_D}, K = \frac{GT_{out} + HW_D T_{in} + Q}{G + HW_D} \end{gathered} . \tag{2.2}$$

Quando a resistência de aquecimento do tanque se considera ligada, a variável $Q$ deve tomar o valor da potência nominal do tanque, e quando se considera desligada, $Q$ deve ser 0. Sempre que houver mudanças nos valores de $Q$ ou $W_D$, o intervalo definido por $t_0$ e $t_f$ deve ser atualizado [9].

A solução (2.2) inclui todos os parâmetros relevantes, mencionados ao longo da descrição do modelo, e por isso será o recurso base usado para a implementação do simulador do tanque, assunto que será abordado no capítulo dedicado a essa temática.

### 2.1.3 Exemplo de Aplicação

Em condições normais, em que a temperatura da água de entrada é consideravelmente inferior à temperatura da água no tanque, e na situação em que não existe consumo de água quente ($W_D(t) = 0$), o modelo prevê uma subida exponencial da temperatura quando a resistência está ligada e um decaimento lento da temperatura quando a resistência está desligada. Este comportamento pode ser observado na Figura 2.2, onde está representada a temperatura da água, dada pelo modelo, ao longo de 7 dias.

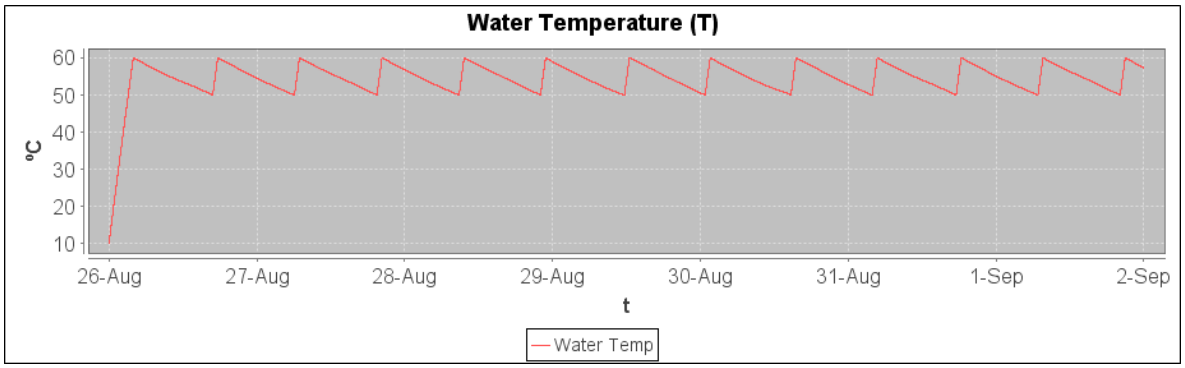


Figura 2.2: Temperatura da água no tanque com consumo de água inexistente (imagem obtida a partir do simulador desenvolvido).

Para este exemplo, foram usados os seguintes valores:

$$T_{in} = 10\,^{\circ}C, T_{out} = 14\,^{\circ}C, W_D(t) = 0\,m^3/s, A = 4.49\,m^2, R = 1\,m^2\,^{\circ}C/W,$$
$$\rho = 985\,kg/m^3, c_p = 4184\,J/^{\circ}C\,kg, V = 0.2\,m^3$$
$$Q(t) = \begin{cases} 3000W & \text{se } T(t) < 50 \\ 0W & \text{se } T(t) > 60 \\ Q(t - \Delta t) & \text{nos restantes casos} \end{cases} . \tag{2.3}$$

A resistência de aquecimento é ligada quando a temperatura é inferior a $50°C$ e é desligada quando a temperatura ultrapassa os $60°C$. A notação $Q(t - \Delta t)$ denota o valor de $Q(t)$ anterior.

## 2.2 Figura de Mérito

Para ajudar na avaliação e comparação da eficácia dos algoritmos desenvolvidos, foi idealizada uma figura de mérito que toma em consideração as duas componentes relevantes para avaliar os algoritmos: o conforto e o consumo de energia. Contudo, estas duas componentes são incompatíveis uma com a outra, isto porque o conforto é dado pela temperatura da água, que implica existência de consumo de energia para a aquecer. A preferência do utilizador para dar prioridade ao conforto ou à poupança de energia também deve ser contabilizada.

Por estes motivos, a criação de uma figura de mérito não é uma tarefa trivial, já que considerar um algoritmo melhor que outro, em sentido lato, nem sempre é possível. A solução pensada tem em consideração os pontos-chave mencionados:

- $f_C(t)$: mede o conforto em função da temperatura da água $T(t)$;
- $f_E(t)$: avalia a poupança de energia através do consumo de energia $Q(t)$;
- $f_P(t)$: representa o peso atribuído a cada componente, tendo em conta a utilização de água $W_D(t)$.

Seja $M(t)$ a figura de mérito em ordem ao tempo, definida por:

$$M(t) = f_P(t) \times f_E(t) + \big(1 - f_P(t)\big) \times f_C(t) \ .$$

Como esta função tem o intuito de ser aplicada em cada instante de tempo simulado, para realizar a comparação entre vários algoritmos deve ser usada a média da pontuação obtida ao longo do tempo (nos instantes discretos simulados). Há que ter em atenção que a comparação destes valores só é válida quando os algoritmos sob avaliação estão sujeitos à mesma situação de teste.

Na Figura 2.3 está exposto um exemplo do resultado da função de mérito ao longo de um dia, para uma situação em que não existe consumo de água. A função a vermelho corresponde à figura de mérito $M(t)$, enquanto que a função a azul representa a média do mérito ao longo do tempo. Neste exemplo foi dado mais peso à componente da poupança de energia e por isso pode identificar-se facilmente no gráfico os intervalos de tempo onde a resistência está ligada, já que o mérito atribuído nesses intervalos vai ser muito baixo (inferior a 0.25).

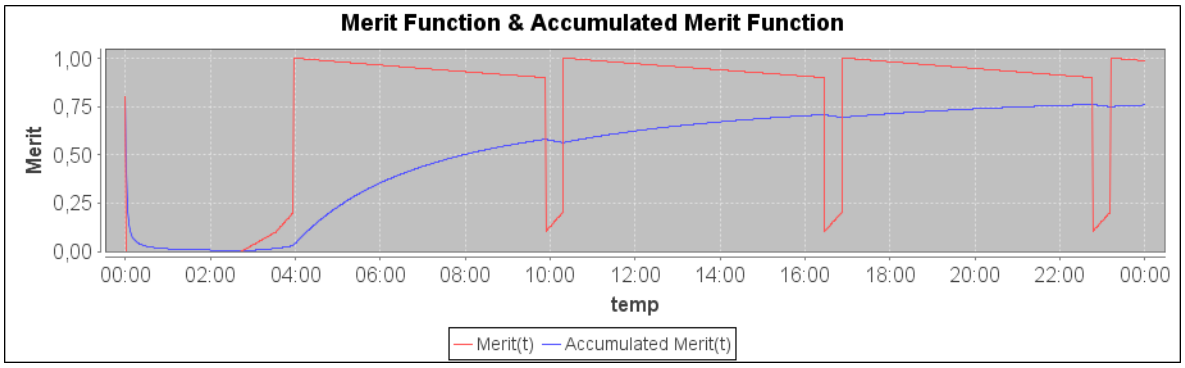


Figura 2.3: Exemplo da função da figura de mérito e da respetiva média (imagem retirada do simulador).

Ambas as funções de conforto e poupança de energia, assim como o elemento de peso, tomam valores entre 0 e 1. Tendo em conta a expressão da função de mérito, isto significa que a pontuação final por ela dada em cada instante varia também entre 0 e 1, respetivamente a pontuação mínima e máxima. A configuração dos parâmetros relativos à função de mérito idealizada poderá ser feita na interface de utilizador do simulador.

### 2.2.1 Função de Peso das Componentes

O parâmetro mais importante da função de mérito tem a ver com a prioridade que é atribuída a cada uma das componentes, visto que isto se trata de um *tradeoff* que depende da preferência do utilizador final. É então preciso decidir qual a componente que merece mais valorização na função de mérito, através de um elemento de peso.

Este elemento de peso foi pensado de modo a poder ser variável ao longo do tempo conforme exista ou não utilização de água quente $[W_D(t)]$, para que se possa valorizar mais o conforto quando realmente importa, ou seja, quando há consumo de água. Denominando de $\alpha_1$ e $\alpha_2$ os pesos escolhidos, a função que traduz o elemento de peso, $f_P(t)$, é definida da seguinte forma:

$$f_P(t) = \begin{cases} \alpha_1, & \text{se } W_D(t) > 0 \\ \alpha_2, & \text{se } W_D(t) = 0 \end{cases} \quad \text{com } \alpha_1, \alpha_2 \in [0,1] \ .$$

### 2.2.2 Função de Poupança de Energia

A função que caracteriza a poupança de energia, $f_E(t)$, é uma função binária dada pelo consumo de energia, $Q(t)$. Se a resistência estiver ligada, a função de poupança de energia tem o valor 0, caso contrário tem o valor 1, ou seja,

$$f_E(t) = \begin{cases} 0, & \text{se } Q(t) > 0 \\ 1, & \text{se } Q(t) = 0 \end{cases} \ .$$

### 2.2.3 Função de Conforto

A função de conforto, $f_C(t)$, está diretamente ligada à temperatura da água $T(t)$, sendo que quanto maior a temperatura, maior é o conforto. Esta correspondência é feita entre dois

intervalos de temperatura contínuos e exclusivos, nos quais a relação entre a temperatura e o conforto é linear.

Desta forma pode ser criada a aproximação de uma curva através de dois segmentos de reta, para que a relação não seja totalmente linear. Esta característica permite descrever 3 tipos de função, apresentados na Figura 2.4.

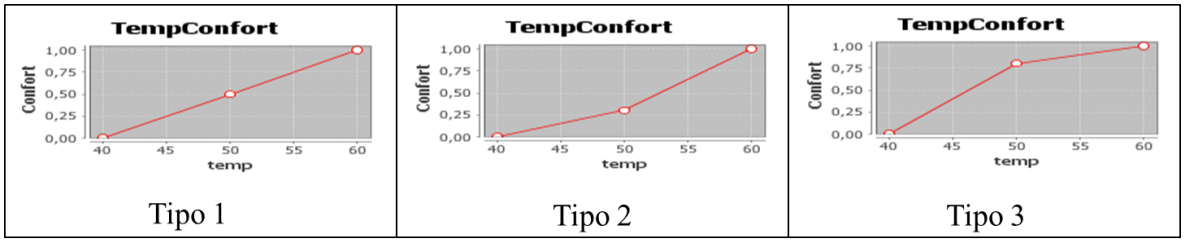


Figura 2.4: Exemplo dos 3 tipos de função de conforto (imagem retirada do simulador). O eixo da ordenada *Confort* corresponde à pontuação atribuida à temperatura correspondente no eixo das abscissas *temp*.

- **Tipo 1**: Relação completamente linear, usando a mesma equação da reta para os dois intervalos.

- **Tipo 2**: Crescimento lento do conforto no primeiro intervalo de temperatura e um crescimento acentuado no segundo intervalo.

- **Tipo 3**: Situação contrária à do tipo 2. Crescimento rápido seguido de um crescimento lento.

Seja $T(t)$ a temperatura da água no instante $t$, $T_{max}$ a temperatura necessária para ter o máximo de conforto, $T_{min}$ a temperatura que corresponde à pontuação mínima da função de conforto e $T_{ideal}$ a temperatura necessária para um conforto aceitável. Os dois intervalos de temperatura mencionados anteriormente vão ser definidos por $[T_{min}, T_{ideal}]$ e $]T_{ideal}, T_{max}]$. Considerando ainda que $C_{ideal}$ toma um valor entre $]0, 1[$ para representar a pontuação atribuída à temperatura $T_{ideal}$, tem-se que a representação matemática da função de conforto é a seguinte:

$$f_C(t) = \begin{cases} 0, & \text{se } T(t) < T_{min} \\ 1, & \text{se } T(t) > T_{max} \\ C_{ideal}/(T_{ideal} - T_{min}) \times [T(t) - T_{ideal}] + C_{ideal}, & \text{se } T_{min} \leq T(t) \leq T_{ideal} \\ (1 - C_{ideal})/(T_{max} - T_{ideal}) \times [T(t) - T_{ideal}] + C_{ideal}, & \text{se } T_{ideal} < T(t) \leq T_{max} \end{cases}.$$

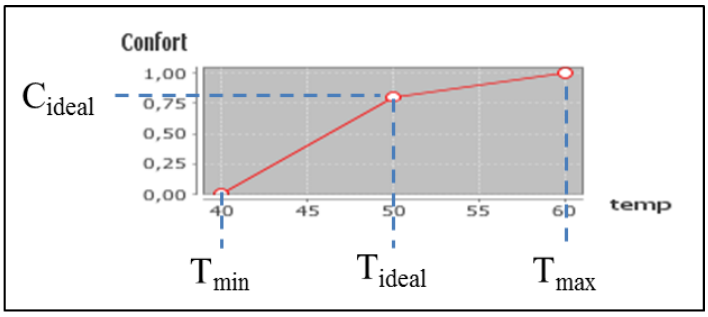


Figura 2.5: Representação dos parâmetros na função de conforto (imagem retirada do simulador).

Para a avaliação dos algoritmos, chegou-se à conclusão que uma função de conforto do tipo 3 é mais adequada, isto porque com temperaturas entre $T_{ideal}$ e $T_{max}$, o impacto no conforto do utilizador será sempre positivo sem que se sinta grande diferença na realidade. Por oposição, quando a temperatura da água disponibilizada vai diminuindo e transpõe $T_{ideal}$, o conforto do utilizador desce acentuadamente. Por estes motivos, $T_{max}$ e $T_{min}$ devem ser estrategicamente escolhidos com base nos *setpoints* máximo e mínimo.

Conclui-se assim desta forma a abordagem ao modelo DEWH e à figura de mérito idealizada. Os detalhes relativos à implementação do modelo como base para o simulador e a incorporação da figura de mérito na mesma ferramenta, serão abordados no Capítulo 4.

# Capítulo 3

# Desenvolvimento de Algoritmos Inteligentes

Nesta secção serão apresentados os algoritmos desenvolvidos no âmbito desta dissertação. Estes algoritmos deverão compreender alguma forma de decisão "inteligente", através da recolha da pouca informação disponível do exterior (mais propriamente a temperatura da água do tanque através de sensores de temperatura). Em primeiro lugar, será abordado o algoritmo de decisão, que é o algoritmo com a responsabilidade de gerir o estado da resistência (ligada ou desligada). Seguidamente, será apresentado um algoritmo que visa detetar ocorrências de tiragem de água que, a falta de um sensor de fluxo, estima esta variável através da variação da temperatura da água no tanque. Este último algoritmo vai ser usado pelo algoritmo de decisão "inteligente" para alcançar o objetivo pretendido.

## 3.1 Algoritmos "Inteligentes" de Decisão

Os algoritmos responsáveis pela decisão de ligar/desligar a resistência, tendo em conta a temperatura da água no tanque, são o objetivo final deste trabalho. Os algoritmos "inteligentes" desenvolvidos devem prever a utilização de água através do seu historial de utilização, para assim tentar otimizar o compromisso entre o conforto do utilizador e o consumo de energia.

### 3.1.1 Algoritmo Típico Baseado nos *Setpoints*

O primeiro algoritmo desenvolvido equivale ao sistema tradicional dos DEWH não inteligentes. A relevância da implementação deste sistema na forma de um algoritmo prende-se com a necessidade de ter um ponto de referência para comparação com o algoritmo inteligente idealizado.

Este sistema baseia-se apenas nos *setpoints* e utiliza uma lógica de decisão muito simples. O algoritmo limita-se a desligar a resistência quando a temperatura da água ultrapassa o *setpoint* máximo ($S_{max}$) e a ligar a resistência quando a temperatura da água passa abaixo do *setpoint* ($S_{min}$). Com uma temperatura entre $S_{min}$ e $S_{max}$, a resistência mantém-se no estado em que estava anteriormente. Este comportamento é representado pela máquina de estados da Figura 3.1.

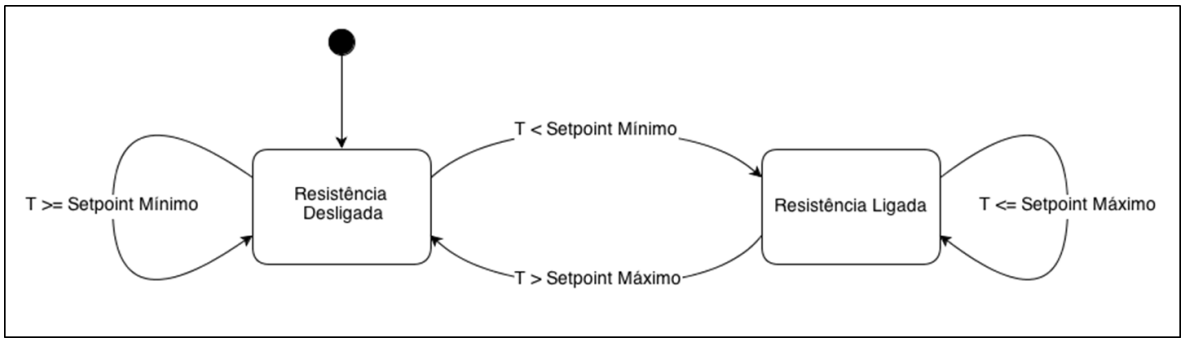


Figura 3.1: Diagrama de estados do algoritmo baseado nos *setpoints*.

Denominando de $T(t)$ a temperatura da água ao longo do tempo, o resultado produzido pelo algoritmo é descrito pela função de consumo de energia $Q(t)$:

$$Q(t) = \begin{cases} 3000\text{w} & \text{se } T(t-1) < S_{min} \\ 0 & \text{se } T(t-1) > S_{max} \\ Q(t-1) & \text{nos restantes casos} \end{cases} .$$

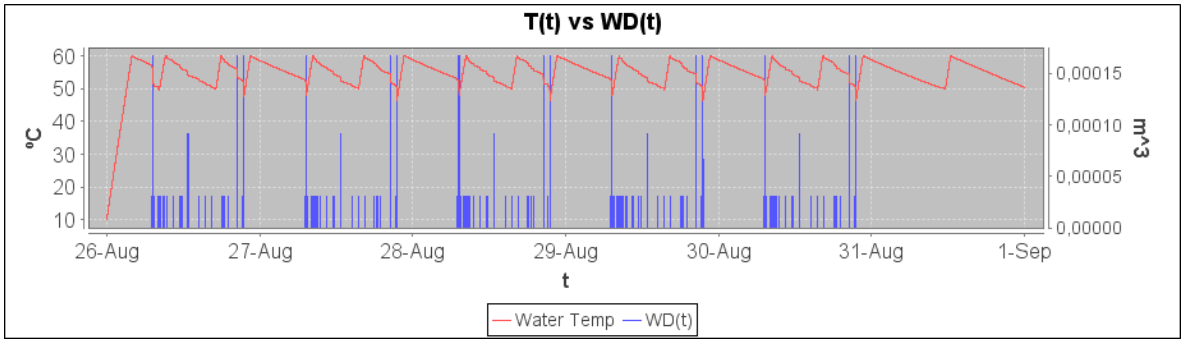


Figura 3.2: Representação gráfica da temperatura face ao perfil de utilização de água.

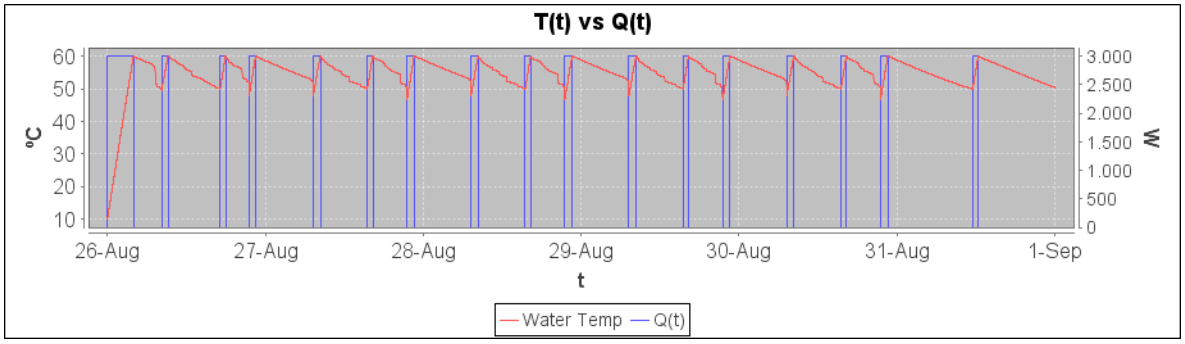


Figura 3.3: Representação gráfica da temperatura face ao consumo de energia para o algoritmo tradicional.

Pode ver-se nas Figuras 3.2 e 3.3, retiradas do simulador, o funcionamento deste algoritmo

numa simulação de 6 dias. O perfil de utilização de água juntamente com a temperatura da água, são visíveis no gráfico da Figura 3.2. O gráfico da Figura 3.3 demonstra o resultado produzido pelo algoritmo em simultâneo com a variação temperatura da água.

Este algoritmo foi designado de "*Basic Setpoint Based Algorithm*", na ferramenta de simulação.

### 3.1.2 Algoritmo "Inteligente" com Histórico Semanal

O algoritmo inteligente idealizado é caracterizado por manter um histórico de utilização de água com período semanal e tentar pré-aquecer a água do tanque nos momentos de grandes tiragens de água quente. O objetivo é conseguir ter água quente quanto realmente interessa e poupar energia nas situações oportunas.

A única informação do exterior a que o algoritmo tem acesso compreende o valor dos *setpoints* (*input*) e a temperatura da água do tanque (através de sensores de temperatura). Para além disso, o algoritmo possui a informação inerente sobre o estado da resistência e o conhecimento que vai formando ao longo do tempo a partir da informação disponível.

Funcionalmente, o algoritmo pode ser dividido em duas fases. Na primeira fase, a fase de decisão, pode tomar a decisão de ligar, desligar ou manter o estado da resistência. A segunda fase, denominada fase de deteção, resume-se à deteção do consumo de água quente e a construção do respetivo histórico de consumo. Estas duas fases são executadas sequencialmente em cada iteração do algoritmo. De notar que cada iteração corresponde a um instante de amostragem.

Durante a primeira semana de execução, o algoritmo ainda não tem qualquer informação sobre a rotina de consumo de água quente por parte dos utilizadores. Por essa razão, na primeira semana, o algoritmo (na fase de decisão) tem um comportamento idêntico ao algoritmo anterior (Figura 3.1), desligando a resistência quando a temperatura atinge o *setpoint* máximo ($S_{max}$) e ligando quando alcança o *setpoint* mínimo ($S_{min}$).

Relativamente à fase de deteção, o comportamento do algoritmo é sempre o mesmo independentemente de se tratar da primeira semana ou não. Nesta fase vão ser usados internamente dois detetores de consumo de água independentes, um para quando a resistência está ligada e outro para a situação oposta, que implementam um algoritmo específico para a deteção de consumo de água quente a partir da variação da temperatura da água. Detalhes sobre o funcionamento do algoritmo de deteção e a motivação para o seu desenvolvimento serão apresentados na Secção 3.2.

Na sequência da deteção, é registado no histórico semanal se no instante atual houve ou não consumo de água. No diagrama da Figura 3.4 está representado o fluxo de uma iteração durante a fase de deteção.

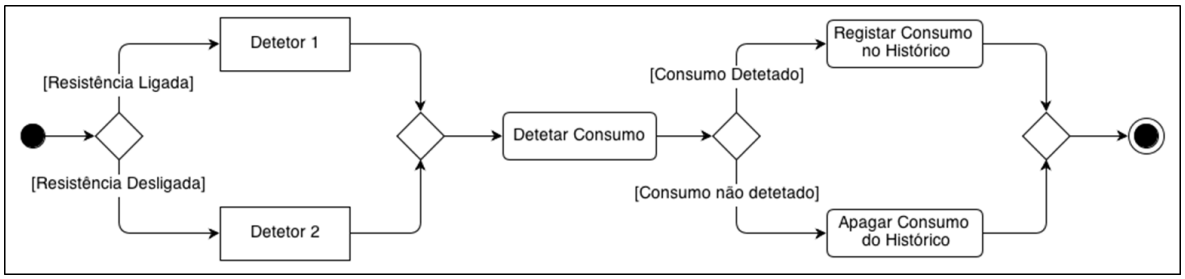


Figura 3.4: Diagrama de atividade da fase de deteção.

No fim da execução das duas fases da iteração, o algoritmo vai ainda atualizar constantemente uma variável que vai indicar a taxa de aquecimento da água quando a resistência está ligada e não existe consumo de água. Este valor será importante para estimar o tempo que demora a aquecer a água desde a temperatura $S_{min}$ até à temperatura $S_{max}$, e assim saber quanta antecedência é necessário prever o consumo de água para oferecer o maior conforto possível ao utilizador.

Em relação ao histórico semanal, a sua estrutura permite armazenar informação do consumo de água diário relativo a uma semana inteira. Isto traduz-se em 7 perfis de utilização de água independentes, um para cada dia da semana, podendo assim diferenciar-se a utilização de água em cada um deles.

Já a pensar nas restrições de memória impostas pelo *Arduino*, os registos foram organizados de forma a compactar a informação o mais possível. O esquema sobre a organização de memória está disposto na Figura 3.5. Cada registo corresponde a um instante de amostragem diário, ou seja, o índice dos registos está diretamente relacionado com o instante temporal diário. Cada registo armazena a informação sobre existência de consumo de água para cada um dos dias da semana nesse instante de tempo.

A sinalização de existência de consumo é feita apenas com um *bit* (*bit flag*) para cada dia da semana, significando isto que cada registo ocupa apenas um *byte*. O número total de registos depende exclusivamente do intervalo de amostragem definido.

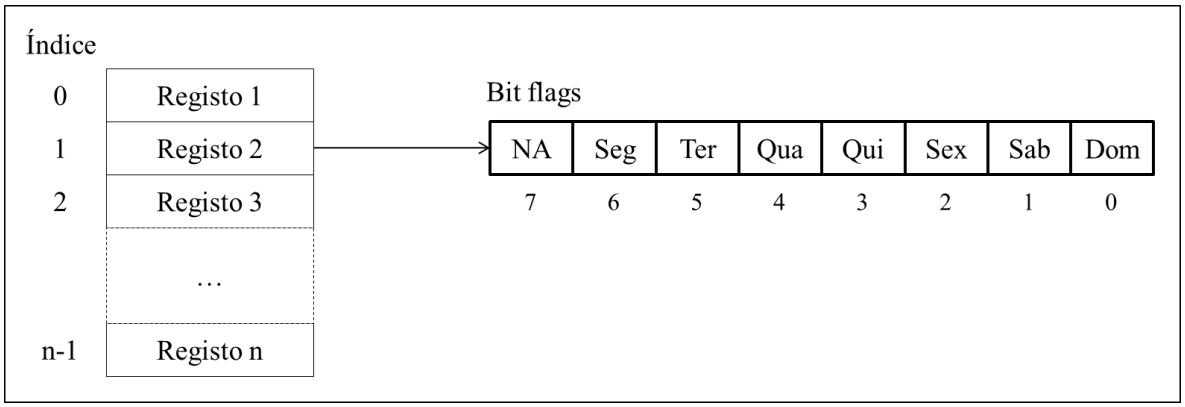


Figura 3.5: Representação da estrutura dos registos do histório semanal.

Retomando a fase de decisão, depois da primeira semana, o algoritmo já adquiriu informação sobre o consumo de água quente durante a semana passada. A decisão tomada na

semana atual está sempre dependente do consumo detetado da semana imediatamente anterior. A decisão em si, descrita pelo diagrama de estados da Figura 3.6, depende de 3 fatores essenciais:

- **Temperatura atual vs. *setpoints***: Conjunto de condições que forçam que a temperatura da água se mantenha sensivelmente dentro dos limites estabelecidos pelos *setpoints* dando prioridade à poupança de energia, ou seja, mantendo sempre que possível a resistência desligada;
- **Previsão de consumo no futuro próximo**: A previsão da existência ou inexistência de consumo no futuro próximo, dada pela observação do historial da semana anterior, é o ponto-chave deste algoritmo. O registo analisado na operação de previsão corresponde a um instante futuro, calculado a partir da estimativa do tempo que demora a aquecer a água desde o *setpoint* mínimo até ao máximo.

  A previsão de consumo implica uma tentativa de aquecimento da água até ao máximo permitido, antes da possível tiragem de água se efetivar. Para além disso, é inicializada uma contagem decrescente através do contador CD, que contabiliza quanto tempo falta até à ocorrência prevista de tiragem de água quente;
- **Valor do contador CD**: Na sequência da previsão de consumo de água, o contador CD é inicializado com o tempo estimado até à tiragem prevista (*offset*). O contador, que é decrementado em cada iteração do algoritmo, vai influenciar que a resistência se mantenha ligada até este atingir o valor 0.

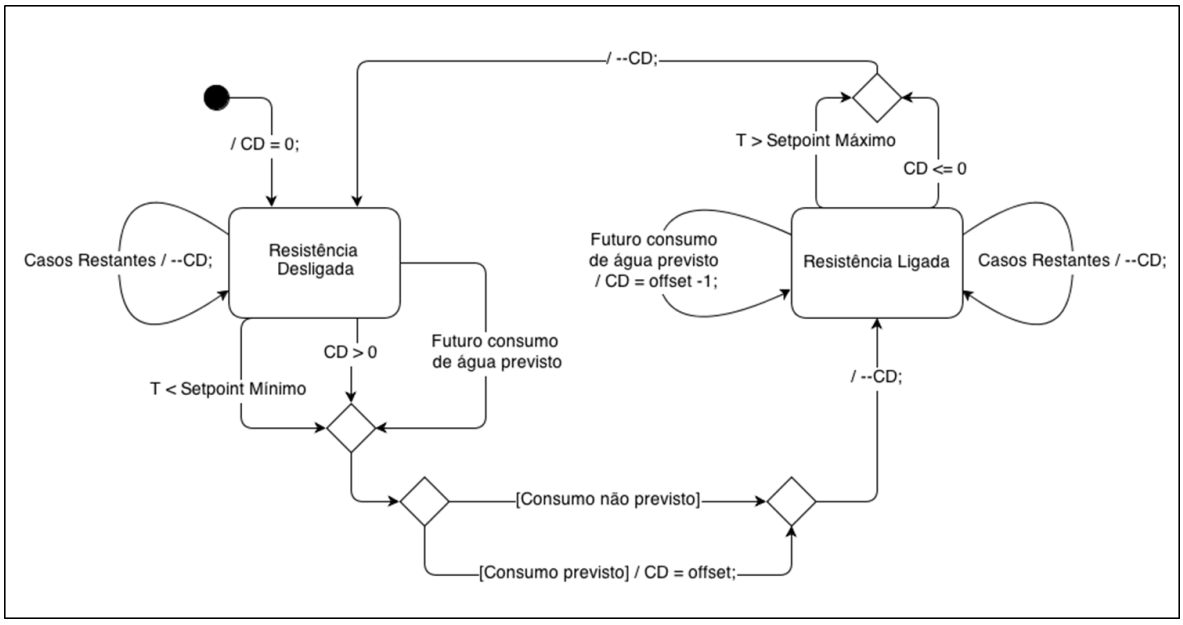


Figura 3.6: Diagrama de estados do algoritmo com histórico semanal.

Nas Figuras 3.7 e 3.8, recolhidas com recurso ao simulador, pode observar-se o comportamento do algoritmo ao longo de duas semanas. A evolução da temperatura face ao consumo de água durante a primeira semana (Figura 3.7) mantém um comportamento semelhante ao

algoritmo "não inteligente" apresentado anteriormente. Na segunda semana (Figura 3.8), com um perfil de utilização idêntico ao da primeira semana, pode constatar-se o pré-aquecimento da água nas grandes tiragens de água quente. De notar que, nesta situação, as tiragens de água curtas e pouco intensivas podem considerar-se desprezáveis, pois quase não se refletem na variação da temperatura e como tal não foram detetadas pelos detetores de consumo.

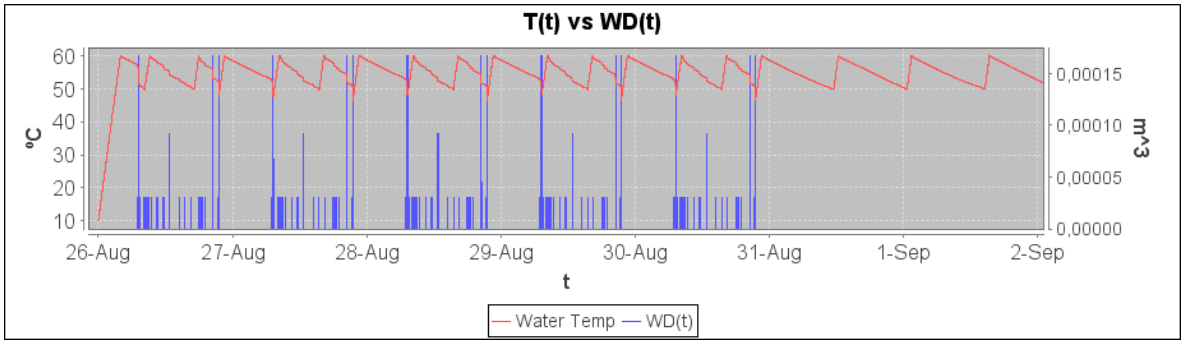


Figura 3.7: Representação gráfica da temperatura face ao perfil de utilização de água, durante a primeira semana.

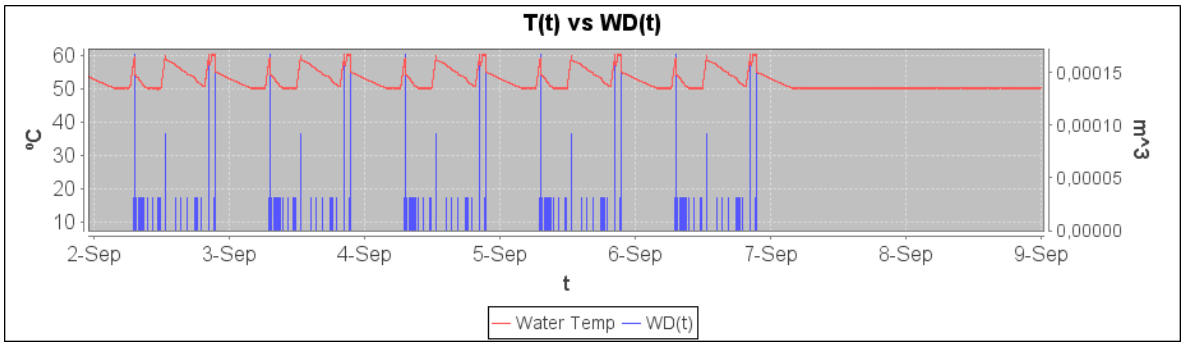


Figura 3.8: Representação gráfica da temperatura face ao perfil de utilização de água, durante a segunda semana.

Este algoritmo está presente no simulador com o nome "History Based Algorithm v2", juntamente com a versão anterior "History Based Algorithm v1" que não faz uso do algoritmo de deteção de utilização de água e os registos estão organizados de forma menos compacta para o caso em que surgisse a necessidade de armazenar mais informação.

## 3.2 Algoritmo de Deteção de Consumo de Água

O algoritmo de deteção do consumo de água quente é apenas uma pequena parte do algoritmo principal de decisão, porém, é de grande importância. Ter informação sobre a saída de água quente do tanque é essencial para conseguir traçar o perfil de consumo de água dos utilizadores ao longo dos dias, para assim otimizar a eficácia do algoritmo principal.

A informação adquirida do exterior está dependente dos sensores, que neste caso são apenas os sensores de temperatura da água que está no tanque, não existindo no âmbito deste

trabalho qualquer sensor que permita diretamente detetar quando existe um fluxo de saída de água. Esta deteção tem que ser então prevista através das variações da temperatura da água do tanque, ao longo do tempo.

Existem dois comportamentos distintos a contemplar na variação da temperatura, consoante se está ou não a usar água. Daqui em diante, a situação em que existe utilização de água será referida como a situação de consumo (C) e por oposição, a situação da não utilização de água será referida como a situação de não consumo (NC). Estes são portanto os dois casos que este algoritmo pretende classificar.

Em condições normais de utilização, a água do tanque deverá estar com uma temperatura superior à água que entra no tanque e à temperatura do ar exterior. É de prever também que, quando existe um consumo de água, a água do tanque tende a arrefecer, devido à saída de água quente e respetiva entrada de água fria.

Perante o caso de a resistência estar ligada ou desligada, o comportamento da variação da temperatura é relativamente diferente. Na situação NC, quando a resistência está ligada, espera-se que a água aqueça com o passar do tempo, enquanto que com a resistência desligada, a água irá arrefecer lentamente.

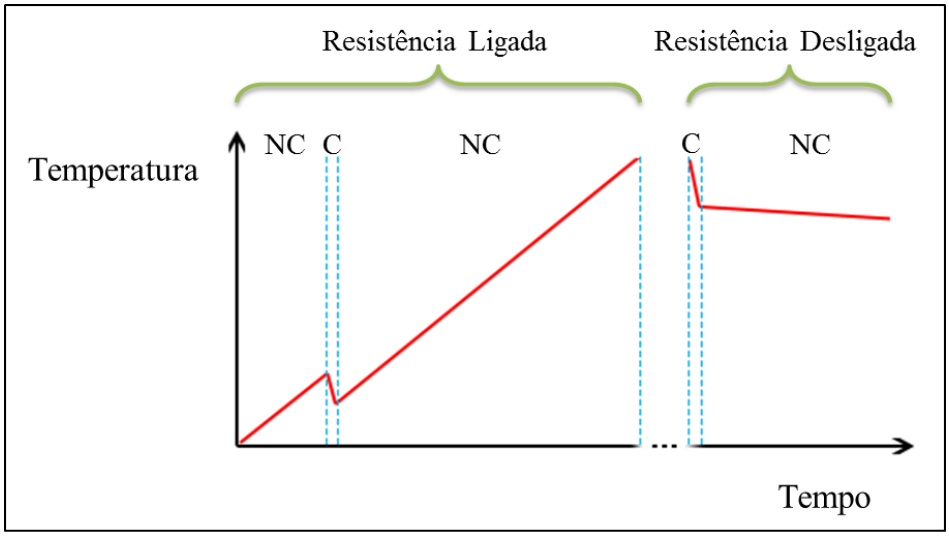


Figura 3.9: Possível representação da variação da temperatura da água no tanque ao longo do tempo.

Quando há consumo de água (C) é intuitivo que, no caso de a resistência estar desligada, a água tende a arrefecer mais rapidamente. No caso de a resistência estar ligada, o comportamento pode ser mais imprevisível. Conforme o fluxo de saída de água quente seja maior ou menor, a água do tanque pode, respetivamente, ir arrefecendo (como representado na Figura 3.9) ou até continuar a aquecer (mas muito lentamente).

Seja a temperatura da água no instante $nT_a$ dada por $T(n) \equiv T(nT_a)$, onde $T_a$ representa

o período de amostragem. A variação da temperatura entre dois instantes consecutivos será denotada por

$$\Delta T(n) = T(n) - T(n-1) \ .$$

É de prever que o valor de $\Delta T(n)$ seja tendencialmente menor nas situações de consumo relativamente às situações de não consumo.

### 3.2.1 Abordagem Inicial

O algoritmo tem como objetivo classificar a situação, para cada amostragem da temperatura da água, como sendo C ou NC, apenas com recurso a $\Delta T(n)$.

Numa primeira análise, assumindo que o modelo de aquecimento do tanque implementado é realista e o intervalo de amostragem é invariável, pode-se estimar empiricamente uma constante que separe os valores de $\Delta T(n)$ que correspondem a consumo dos que correspondem a não consumo. Com as restrições expostas, esta solução simplista permite descriminar corretamente entre as duas situações, comparando apenas se para um dado $\Delta T(n)$ este é maior ou menor que a constante. Contudo, se for tido em conta um intervalo de amostragem diferente, tanques com características diferentes (e.g., volume total, área, etc.), ou ainda variações muito grandes da temperatura do ar exterior e da água que entra no tanque (e.g., nas mudanças de estações), o valor da constante teria que ser ajustado, ou seja, passar a ser uma variável dependente de todos os parâmetros mencionados.

Modelar esta variável iria ser uma tarefa complicada e a sua eficácia difícil de validar num ambiente real, para todos os casos.

### 3.2.2 Deteção

O algoritmo de deteção é feito com recurso a dois pontos unidimensionais em que cada ponto representa o centro de um *cluster*. Os *clusters* são designados por $* = \{C, CN\}$, onde C representa consumo e NC representa não consumo. Cada *cluster* é então caracterizado pelas seguintes variáveis:

- Centro do *cluster* $[M^{(*)}]$: representa a localização do *cluster* e está diretamente relacionado com os valores de $\Delta T(n)$;
- Limite inferior $[L_1^{(*)}]$ e superior $[L_2^{(*)}]$: valores que delimitam o alcance do *cluster* e servem apenas para ajustar o centro deste;
- Peso do limite inferior $[P_1^{(*)}]$ e superior $[P_2^{(*)}]$: quantidades que permitem calcular o ponto de equilíbrio para reposicionar o centro do *cluster*;
- Contador de classificações "ganhas" $[W^{(*)}]$: conta o número de vezes que o *cluster* "ganhou" na classificação do $\Delta T(n)$.

A deteção em si é dada apenas pelo centro dos *clusters*, isto é, para cada valor de $\Delta T(n)$ é atribuída a classificação conforme o centro do *cluster* que está mais próximo. Isto significa que, caso $\Delta T(n)$ tenha como vizinho mais próximo o centro do *cluster* $C$ $[M^{(C)}]$, o algoritmo reporta a existência de consumo. Analogamente, para a situação contrária, caso $\Delta T(n)$ tenha como vizinho mais próximo o centro do *cluster* $NC$ $[M^{(NC)}]$, o algoritmo diz estar perante um caso NC. De notar que, pelo que já foi referido anteriormente, sabe-se que $M^{(C)}$ será sempre menor que $M^{(NC)}$.

Os limites e o seu peso têm como função ajustar o detetor, já que é com essa informação que o centro dos *clusters* é recalculado em cada iteração. Os contadores são valores cruciais para a deteção de mau posicionamento dos *clusters* que dão origem a classificações erradas.

### 3.2.3 Detalhes da Implementação do Algoritmo

O algoritmo é inicializado com dois parâmetros que representam os valores iniciais dos centros dos *clusters*, $M^{(C)}$ e $M^{(NC)}$, sendo que o valor de $M^{(C)}$ tem que ser inferior ao de $M^{(NC)}$.

Todos os pesos e contadores são inicializados com o valor "1" e ambos os limites de cada *cluster* são colocados no centro do seu respetivo *cluster*, ou seja, são inicializados com o valor do centro do *cluster* a que correspondem. Sejam $L_1^{(C)}$ e $L_2^{(C)}$ os limites inferior e superior do *cluster* de consumo e $L_1^{(NC)}$ e $L_2^{(NC)}$ os limites inferior e superior do *cluster* de não consumo, Temos então

$$\begin{aligned} L_1^{(C)} = L_2^{(C)} = M^{(C)} \\ L_1^{(NC)} = L_2^{(NC)} = M^{(NC)} \end{aligned} .$$

Em cada iteração (instante $n$), o algoritmo recebe o valor de $\Delta T(n)$ e verifica qual dos centros dos *clusters* é o vizinho mais próximo deste, classificando assim como havendo consumo no caso de estar mais próximo do *cluster* do consumo e como não havendo consumo caso contrário, isto é

$$\arg\min_{*=\{C,NC\}} |M^{(*)} - \Delta T(n)| .$$

O passo seguinte passa por atualizar os parâmetros do *cluster* que prevaleceu na classificação (*cluster* com centro mais próximo do valor de $\Delta T(n)$). O contador desse *cluster* $[W^{(*)}]$ é incrementado uma unidade.

Os limites do *cluster* que prevaleceu na classificação também são alvos da atualização. Denominando respetivamente por $L_1^{(*)}$ e $L_2^{(*)}$ o seu limite inferior e superior, temos

$$\begin{aligned} L_1^{(*new)} = \max(\Delta T(n), L_1^{(*)}) \\ L_2^{(*new)} = \min(\Delta T(n), L_2^{(*)}) \end{aligned} .$$

De notar que apenas uma das equações anteriores pode produzir uma mudança no valor dos limites, isto porque $L_1^{(*)} \leq M^{(*)} \leq L_2^{(*)}$.

No seguimento desta atualização, os pesos dos limites também estão sujeitos a sofrer alterações. O peso de $L_1^{(*)}$, denominado de $P_1^{(*)}$, será incrementado em uma unidade no caso de $\Delta T(n)$ ser menor que $M^{(*)}$. Por outro lado, o peso de $L_2^{(*)}$, denominado de $P_2^{(*)}$, será incrementado em uma unidade no caso de $\Delta T(n)$ ser maior que $M^{(*)}$:

$$\begin{array}{l} P_1^{(*)} = P_1^{(*)} + 1 \quad se\ L_1^{(*new)} \neq L_1^{(*)} \\ P_2^{(*)} = P_2^{(*)} + 1 \quad se\ L_2^{(*new)} \neq L_2^{(*)} \end{array} .$$

Já com os limites e pesos atualizados, $M^{(*)}$ é agora recalculado da seguinte forma:

$$M^{(*)} = \frac{L_1^{(*new)} P_1^{(*)} + L_2^{(*new)} P_2^{(*)}}{P_1^{(*)} + P_2^{(*)}} .$$

Como medida de prevenção de erros provocados por *overflow*, é verificado se algum dos contadores atingiu o limiar estabelecido de $2^{16} - 1$ [1]. Se isso acontecer, todos os parâmetros dos *clusters* são reinicializados com exceção dos centros, ou seja, os contadores de ambos os *clusters* e os seus limites são repostos a "1", e os limites em si tomam o valor do centro do seu *cluster*, ou seja,

$$\begin{array}{c} L_1^{(C)} = L_2^{(C)} = M^{(C)} \\ L_1^{(NC)} = L_2^{(NC)} = M^{(NC)} \end{array} .$$

### 3.2.4 Tratamento de Situações Excecionais

Opcionalmente, pode ser efetuado um último passo extra na iteração, que permite detetar situações em que os centros dos *clusters* foram mal posicionados, resultando em classificações erradas. Esta situação pode acontecer quando os valores iniciais de $M^{(C)}$ e $M^{(NC)}$ são pouco realistas e proporcionam que os valores de $\Delta T(n)$, quando há ou não há consumo, estejam sempre mais próximos do mesmo *cluster*, dando origem a permanentes falsos positivos ou falsos negativos. Quando assim é, o contador do *cluster* que está a prevalecer erradamente em todas as classificações vai continuar a ser incrementado, provocando um balanço extremamente irregular nos contadores dos *clusters*. Com este passo é possível detetar esta situação através de um valor de entrada que indica um limiar $TH$, para a razão $R$ entre os contadores dos dois *clusters* dada por,

$$R = \frac{\max(W^{(C)}, W^{(NC)})}{\min(W^{(C)}, W^{(NC)})} .$$

Se $R$ for maior que $TH$, então é dado início a uma rotina de rebalanceamento e reajuste dos centros que consiste em reposicionar o centro do *cluster* cujo contador é menor, ou seja,

$$\begin{array}{l} M^{(NC)} = L_2^{(C)} \quad se\ W^{(C)} > W^{(NC)} \\ M^{(C)} = L_1^{(NC)} \quad se\ W^{(C)} < W^{(NC)} \end{array} .$$

Para concluir este procedimento, os contadores dos *clusters* e os pesos de todos os limites são reinicializados com o valor "1", e os limites de cada *cluster* são colocados no centro do seu respetivo *cluster*, tal como acontece no início do algoritmo e na situação de prevenção de *overflow*.

[1] Valor máximo de uma variável de 16-bits sem sinal.

### 3.2.5 Exemplos

No algoritmo principal vão existir duas instâncias do algoritmo de classificação, uma para a situação em que a resistência está ligada e outra para quando está desligada, denominadas respetivamente de instância $ON$ e instância $OFF$. Significa isto que vão existir efetivamente quatro *clusters* em simultâneo, cada um representando um caso da Figura 3.9. Porém, só uma dessas instâncias é usada em cada instante conforme a informação (inerente ao programa principal) de a resistência estar ligada ou desligada nesse momento.

Os passos importantes do algoritmo de classificação são exibidos com recurso a um eixo horizontal que representa a gama de valores de $\Delta T(n)$, onde são dispostos os *clusters* (o seu centro e os limites respetivos) apenas de uma das instâncias, como ilustrado na Figura 3.10. Nos exemplos que são expostos nesta secção, todos os valores representados por números reais foram truncados a 3 casas decimais.

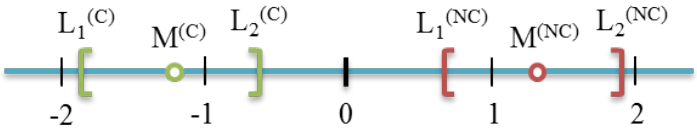


Figura 3.10: Exemplo da representação dos parâmetros dos *clusters* de uma instância.

### 3.2.6 Exemplo A - Parâmetros de Entrada Adequados

Neste exemplo foi testado o progresso do algoritmo para a instância $ON$, simulando um período de 30 dias com um intervalo de amostragem de 1 minuto, com os seguintes valores de entrada:

- $M^{(C)} = -0.1$
- $M^{(NC)} = 0.1$

No estado inicial, os limites tomam o valor do centro do respetivo *cluster* (Figura 3.11).


Figura 3.11: Estado inicial do algoritmo no exemplo A. $M^{(C)} = -0.1$; $M^{(NC)} = 0.1$; $L_1^{(C)} = -0.1$; $P_1^{(C)} = 1$; $L_2^{(C)} = -0.1$; $P_2^{(C)} = 1$; $L_1^{(NC)} = 0.1$; $P_1^{(NC)} = 1$; $L_2^{(NC)} = 0.1$; $P_2^{(NC)} = 1$; $W^{(C)} = 1$; $W^{(NC)} = 1$.

Na primeira iteração, $\Delta T(n) = 0.219$ (corresponde a uma situação de não consumo). Por este valor estar mais próximo de $M^{(NC)}$, a situação é corretamente classificada pelo algoritmo, resultando na atualização dos parâmetros $L_2^{(NC)}$, $P_2^{(NC)}$, $W^{(C)}$ e $M^{(NC)}$, como mostra a Figura 3.12.


Figura 3.12: Estado do algoritmo no exemplo A depois da primeira iteração. $M^{(C)} = -0.1$; $M^{(NC)} = 0.179$; $L_1^{(C)} = -0.1$; $P_1^{(C)} = 1$; $L_2^{(C)} = -0.1$; $P_2^{(C)} = 1$; $L_1^{(NC)} = 0.1$; $P_1^{(NC)} = 1$; $L_2^{(NC)} = 0.219$; $P_2^{(NC)} = 2$; $W^{(C)} = 1$; $W^{(NC)} = 2$.

Todas as 237 iterações que se seguiram resultaram em situações de não consumo classificadas corretamente, com $\Delta T(n)$ a tomar valores entre 0.2 e 0.219. Isto implica que o *cluster* $C$ continua com os seus parâmetros inalterados, assim como os limites do *cluster* $NC$. As únicas alterações decorrem da atualização do contador, dos pesos e do centro do *cluster* $NC$. Pode ver-se na Figura 3.13 que $M^{(NC)}$ se aproximou mais de $L_2^{(NC)}$ por este ter um peso maior (ver Figura 3.13).


Figura 3.13: Estado do algoritmo no exemplo A após 238 iterações. $M^{(C)} = -0.1$; $M^{(NC)} = 0.203$; $L_1^{(C)} = -0.1$; $P_1^{(C)} = 1$; $L_2^{(C)} = -0.1$; $P_2^{(C)} = 1$; $L_1^{(NC)} = 0.1$; $P_1^{(NC)} = 32$; $L_2^{(NC)} = 0.219$; $P_2^{(NC)} = 208$; $W^{(C)} = 1$; $W^{(NC)} = 239$.

Na iteração imediatamente a seguir é recebido um $\Delta T(n) = -1.924$ que representa uma situação de consumo que é corretamente classificada. Neste caso, o *cluster* $NC$ mantém os seus valores e o *cluster* $C$ tem $L_1^{(C)}$, $P_1^{(C)}$, $W^{(C)}$ e $M^{(C)}$ atualizados de acordo com a Figura 3.14.


Figura 3.14: Estado do algoritmo no exemplo A após a primeira situação de consumo. $M^{(C)} = -1.316$; $M^{(NC)} = 0.203$; $L_1^{(C)} = -1.924$; $P_1^{(C)} = 2$; $L_2^{(C)} = -0.1$; $P_2^{(C)} = 1$; $L_1^{(NC)} = 0.1$; $P_1^{(NC)} = 32$; $L_2^{(NC)} = 0.219$; $P_2^{(NC)} = 208$; $W^{(C)} = 2$; $W^{(NC)} = 239$.

Com o decorrer do tempo, o centro de cada *cluster* tende a convergir para um valor próximo dos $\Delta T(n)$ que correspondem à situação de cada um. No final da simulação o estado do algoritmo por ser visto na Figura 3.15.


Figura 3.15: Estado do algoritmo no exemplo A no final da simulação. $M^{(C)} = -1.639$; $M^{(NC)} = 0.203$; $L_1^{(C)} = -2.072$; $P_1^{(C)} = 160$; $L_2^{(C)} = -0.1$; $P_2^{(C)} = 45$; $L_1^{(NC)} = 0.1$; $P_1^{(NC)} = 973$; $L_2^{(NC)} = 0.219$; $P_2^{(NC)} = 6388$; $W^{(C)} = 204$; $W^{(NC)} = 7360$.

De notar que neste exemplo o algoritmo não produziu qualquer classificação errada, isto porque os parâmetros iniciais foram favoráveis ao ajuste natural dos *clusters* à sua posição ideal.

### 3.2.7 Exemplo B - Deteção de *Cluster* Mal Posicionado

Neste exemplo, o objetivo é demonstrar a utilidade da deteção de mau posicionamento dos *clusters* e o funcionamento da rotina de rebalanceamento. Foi testado o progresso do algoritmo para a instância $OFF$, simulando um período de 30 dias com um intervalo de amostragem de 1 minuto, com os seguintes valores de entrada intencionalmente desadequados:

- $M^{(C)} = -2$
- $M^{(NC)} = 2$

No estado inicial, os limites tomam o valor do centro do respetivo *cluster* (Figura 3.16).


Figura 3.16: Estado inicial do algoritmo no exemplo B. $M^{(C)} = -2$; $M^{(NC)} = 2$; $L_1^{(C)} = -2$; $P_1^{(C)} = 1$; $L_2^{(C)} = -2$; $P_2^{(C)} = 1$; $L_1^{(NC)} = 2$; $P_1^{(NC)} = 1$; $L_2^{(NC)} = 2$; $P_2^{(NC)} = 1$; $W^{(C)} = 1$; $W^{(NC)} = 1$.

Ao fim de várias iterações, foram sendo recebidos valores de $\Delta T(n)$ a rondar $-0.015$ (quando não houve consumo) e por vezes a atingir valores na ordem dos $-2.4$ (quando houve consumo). É fácil de compreender que que ambas as situações $M^{(C)}$ vai estar sempre mais próximo destes valores de $\Delta T(n)$, originando constantes classificações de consumo mesmo quando esse não é o caso. Na iteração número 249, o algoritmo encontra-se no estado ilustrado na Figura 3.17 (antes do rebalanceamento).


Figura 3.17: Estado do algoritmo no exemplo B no decorrer da iteração nº 249 (pré balanceamento). $M^{(C)} = -0.101$; $M^{(NC)} = 2$; $L_1^{(C)} = -2.442$; $P_1^{(C)} = 9$; $L_2^{(C)} = -0.014$; $P_2^{(C)} = 242$; $L_1^{(NC)} = 2$; $P_1^{(NC)} = 1$; $L_2^{(NC)} = 2$; $P_2^{(NC)} = 1$; $W^{(C)} = 250$; $W^{(NC)} = 1$.

Utilizando um limiar $TH = 250$, o algoritmo quando atinge a situação da Figura 3.16 vai perceber que o rácio das classificações ganhas ($R = \frac{250}{1}$) alcançou $TH$, dando assim início à rotina de rebalanceamento. No fim da iteração, já com o rebalanceamento executado, os parâmetros do algoritmo ficam como é mostrado na Figura 3.18.


Figura 3.18: Estado do algoritmo no exemplo B no fim da iteração nº 249 (pós balanceamento). $M^{(C)} = -0.101$; $M^{(NC)} = -0.014$; $L_1^{(C)} = -0.101$; $P_1^{(C)} = 1$; $L_2^{(C)} = -0.101$; $P_2^{(C)} = 1$; $L_1^{(NC)} = -0.014$; $P_1^{(NC)} = 1$; $L_2^{(NC)} = -0.014$; $P_2^{(NC)} = 1$; $W^{(C)} = 1$; $W^{(NC)} = 1$.

Como se pode observar, os pesos e os contadores foram reinicializados, $M^{(C)}$ ficou intacto, $M^{(NC)}$ moveu-se para onde se situava $L_2^{(C)}$, e ambos os limites de cada *cluster* colapsaram sobre o seu centro.

A partir deste momento, valores de $\Delta T(n)$ que não correspondem a consumo (aproximadamente $-0.015$) e valores de $\Delta T(n)$ que correspondem a consumo (inferiores a $-2$) são classificados devidamente, levando a que os centros dos *clusters* afluam para a posição apropriada, como se pode ver na Figura 3.19.

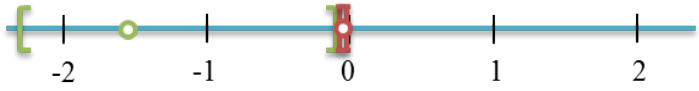


Figura 3.19: Estado final do algoritmo no exemplo B. $M^{(C)} = -1.622$; $M^{(NC)} = -0.014$; $L_1^{(C)} = -2.382$; $P_1^{(C)} = 2$; $L_2^{(C)} = -0.101$; $P_2^{(C)} = 1$; $L_1^{(NC)} = -0.015$; $P_1^{(NC)} = 12$; $L_2^{(NC)} = -0.014$; $P_2^{(NC)} = 33$; $W^{(C)} = 2$; $W^{(NC)} = 44$.

De notar que, perante situações improváveis em que o $TH$ é alcançado quando os *clusters* estão bem posicionados (provocando rebalanceamentos quando não era suposto), não haverá grande impacto na assertividade das deteções já que o centro de um dos *clusters* vai manter-se numa posição correta e o outro, mesmo que se afaste da sua posição ideal, vai estar sempre mais perto desta que o *cluster* oposto.

### 3.2.8 Resultado Final

Como foi referido anteriormente, vão existir duas instâncias do algoritmo de deteção em simultâneo (instância $ON$ e instância $OFF$) no programa principal. O exemplo A foi demonstrado com recurso à instância $ON$ e o exemplo B com recurso à instância $OFF$. Porém, ambos os exemplos pertencem à mesma execução. O que acontece é que o resultado das deteções e consequentes ajustes nos valores dos *clusters* são independentes entre as duas instâncias, sendo assim possível separar os exemplos. Isto implica também que a deteção de consumo ao longo do tempo é dada pelo conjunto de resultados das duas instâncias.

Pode ver-se no gráfico da Figura 3.20 o resultado dado pelas duas instâncias, confrontado com o consumo de água "real", durante a simulação do primeiro dia. A azul e verde estão representados os resultados das instâncias $ON$ e $OFF$ respetivamente. Exibido a vermelho está o consumo de água efectivo.

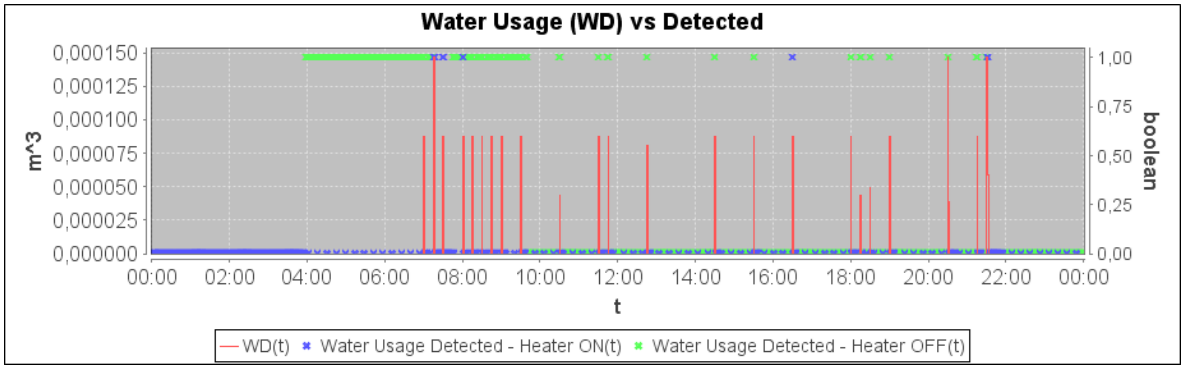


Figura 3.20: Consumo de água efectivo *versus* consumo detectado. 1 - "consumo" detetado; 0 - "não consumo" detetado.

Tal como explicado no exemplo A e pode ser observado na Figura 3.20, todas as situações foram corretamente detetadas no caso da instância $ON$. Por oposição, no exemplo B, no caso da instância $OFF$, todas as deteções iniciais indicam situações de consumo quando na realidade a maioria são falsos positivos. No entanto, após o rebalanceamento dos *clusters* (perto das 10 horas), o algoritmo passa a ter deteções acertadas até ao fim da simulação.

# Capítulo 4

# Implementação do Simulador

Embora o simulador em si não seja o objetivo desta dissertação, este mereceu um grande foco de atenção devido a sua importância para o estudo e validação do trabalho realizado. Este é essencial para conseguir obter *feedback*, em tempo útil, sobre o desempenho dos algoritmos que foram implementados.

O objetivo principal do simulador é retratar a evolução da temperatura da água do interior do tanque ao longo do tempo, com recurso ao modelo especificado anteriormente. Mas como será abordado de seguida, aplicação de simulação vai para além da implementação do modelo do tanque. Esta permite a parametrização das constantes do modelo do tanque, oferece uma representação intuitiva dos resultados importantes em formato gráfico, fornece funcionalidades de exportação de resultados e ainda algumas ferramentas para comparação do desempenho dos algoritmos. O simulador inclui também a implementação do sistema de pontuação idealizado para o desempenho dos algoritmos, referido como figura de mérito.

## 4.1 Solução Implementada

O programa de simulação foi desenvolvido em *Java (JDK 7)*, recorrendo a componentes *Swing* para criação da interface de utilizador. Foram também utilizadas algumas bibliotecas de terceiros, nomeadamente a *JFreeChart 1.0.14* para a manipulação e representação da informação por via de gráficos 2D, a *RXTX 2.2 (64-bit)* para a comunicação USB com a plataforma de desenvolvimento *Arduino*, e a *OpenCSV 2.3* para manipulação de ficheiros CSV (*comma-separated values*). Como referência foi usada a documentação disponível online ([12][13][14][15]) assim como o manual técnico da API *JFreeChart* ([16]).

Como nota prévia para esta secção, quando utilizados os termos "classe", "objeto", "método", "herança" e "interface", estes devem ser interpretados no contexto da Programação Orientada a Objetos ([17]).

Na Figura 4.1 é apresentada uma panorâmica da estrutura do simulador. Os pormenores relativos às entradas, saídas e componentes do simulador, assim como o seu funcionamento base, serão mencionados de seguida. Relativamente ao "Modo Normal" e "Modo Servidor" referidos no diagrama, estes referem-se à capacidade de o simulador correr o algoritmo de decisão localmente ou remotamente. Informação mais detalhada sobre o servidor será exposta na secção devida.

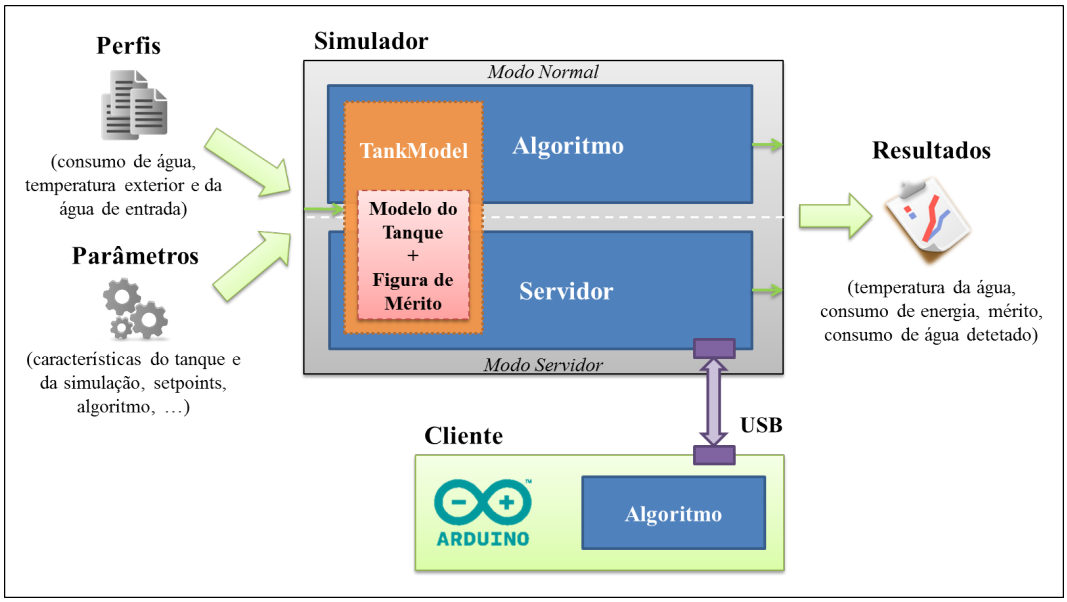


Figura 4.1: Diagrama da estrutura do simulador.

Tendo em conta que o modelo está dependente de uma variedade de parâmetros relacionados com as características inerentes ao mesmo, o simulador recebe como valores de entrada o volume e a área da superfície do tanque, a resistência térmica do tanque, a densidade e o calor específico da água, e também os perfis de temperatura do ar exterior e da água de entrada no tanque. Para além dos parâmetros relacionados com o modelo, é ainda necessário estabelecer qual a temperatura inicial da água no tanque, o número total de dias que se pretende simular, o intervalo de amostragem (*time step*), o *setpoint* máximo e mínimo, e o perfil te utilização de água.

O intervalo de amostragem é usado para definir quanto tempo (simulado), em segundos, passa entre cada cálculo de temperatura, resultando na aplicação discreta no tempo da função do modelo, algo que pode também ser interpretado como a resolução temporal do simulador. Na implementação real do sistema, este valor corresponde fundamentalmente ao intervalo de tempo definido entre cada iteração do algoritmo, onde se efetua a amostragem da temperatura da água do tanque e se toma a decisão de ligar ou desligar a resistência.

Os *setpoints* máximo e mínimo são valores que indicam os limites absolutos para a temperatura máxima e mínima da água do tanque e que têm que ser incondicionalmente respeitados pelos algoritmos. Isto significa que um qualquer algoritmo não pode tomar a decisão de manter a resistência ligada quando a temperatura do *setpoint* máximo é alcançada, e é obrigado a ligar a resistência quando a temperatura é inferior ao *setpoint* mínimo. Estes deverão ser dois valores configuráveis pelo utilizador final (do tanque de água quente).

Relativamente aos resultados produzidos pelo simulador, este retém a evolução da temperatura da água no tanque ao longo do tempo, a respetiva representação do perfil resultante do consumo de energia, e o valor de mérito em cada instante da simulação.

Os parâmetros de entrada vão possibilitar criar um objeto do tipo *TankModel*, que implementa o modelo do tanque e permite manipular e armazenar os resultados. Para além disso, fornece o método *calcNewValue()*, necessário para calcular a temperatura da água ao longo do tempo.

O método *calcNewValue()* calcula e retorna o valor de temperatura para o instante seguinte da simulação, por isso este é invocado repetidamente em ciclo, até ser atingida a totalidade de dias da simulação. Este método recebe como parâmetro de entrada a indicação do estado da resistência (ligada/desligada).

Como foi referido no capítulo dedicado aos algoritmos implementados, cada algoritmo, valendo-se da lógica adicional que lhe é característica, dos possíveis valores armazenados, da potencial previsão de consumo ou não consumo de água, e da informação de temperatura da água no instante atual, é que toma a decisão de ligar ou desligar a resistência. Assim sendo, esta decisão é fundamentada na altura da invocação do método *calcNewValue()*, já que o parâmetro de entrada deste é precisamente a indicação de a resistência estar ligada ou desligada. Estas afirmações implicam que é sobre os algoritmos de decisão que recai a responsabilidade de manter a instância do *TankModel* e controlar o ciclo de invocação do método *calcNewValue()*.

No fluxo de execução do método *calcNewValue()*, em primeiro lugar, o valor de entrada (estado da resistência) é imediatamente adicionado à devida estrutura, com a finalidade de ir construindo o perfil de consumo de energia.

No passo seguinte é aplicada a expressão do modelo para calcular a temperatura da água no instante atual. O valor da temperatura vai sendo armazenado na respetiva estrutura de dados para que no fim se possa reconstituir a evolução da temperatura ao longo de todo o tempo de simulação. É porém tido em atenção que este passo só é efetivado após assegurar que os valores de consumo de água e de energia estão em conformidade com as restrições do modelo. Como referido na apresentação do modelo, a expressão da solução do modelo do tanque é válida apenas para intervalos de tempo onde os valores de consumo de água ($W_D$) e de energia ($Q$) são constantes. Portanto, quando um destes valores varia, o instante inicial do intervalo é atualizado com o valor mais recente, antes de proceder à aplicação da expressão do modelo.

Por conseguinte, no último passo, já com o valor da temperatura da água e a indicação do estado da resistência para o instante da iteração em curso, juntamente com o perfil de utilização de água, a pontuação atribuída pela função de mérito e a sua média é determinada e armazenada pelo simulador para o instante em questão. No fim da simulação, estes dados oferecem a informação sobre o mérito do algoritmo ao longo do tempo de simulação.

É importante referir que, de cada vez que o método *calcNewValue()* é invocado, a data/relógio da simulação é automaticamente incrementada(o) pela quantidade estabelecida no parâmetro *time step* (intervalo de amostragem).

Terminada a simulação, ou seja, após calcular a temperatura da água para todo o tempo especificado, o programa utiliza os resultados armazenados em memória para os representar graficamente na interface de utilizador. O programa calcula ainda algumas estatísticas como o total de vezes em que a resistência foi ligada, valor que permite perceber o desempenho do algoritmo relativamente à energia consumida, e o valor de mérito final dado pela média da

pontuação de mérito ao longo de toda a simulação.

Devido à necessidade de comparação do desempenho entre os vários algoritmos implementados é importante que o simulador ofereça a capacidade de alternar facilmente entre eles, sem intervenções no código fonte. Para esse efeito, os algoritmos de decisão foram desenvolvidos com uma estrutura modular em mente. Todos eles partilham uma interface comum (através de herança da classe abstrata "Algorithm"). Para adicionar um novo algoritmo ao simulador, basta então expandir (estender) a classe "Algorithm" e implementar o método requerido, onde é exposta toda a lógica do algoritmo e a construção dos resultados. Para finalizar, é necessário referenciar e acrescentar a instanciação do novo algoritmo na classe principal.

### 4.1.1 Formato dos Perfis

Os perfis são ficheiros de texto em que consta informação relativa a um determinado perfil de temperatura ou utilização de água. O texto dos ficheiros apresenta uma estrutura tabular (formato CSV) em que cada linha representa um registo. Estes registos têm que ser dispostos por ordem crescente do tempo. Os campos (colunas) variam conforme se trate de um perfil de temperatura ou um perfil de consumo de água.

Um perfil de consumo de água é composto por três campos por registo. Cada registo contém a informação relativa a um consumo expressa pelos seguintes campos (ordenadamente):

- **Dia e Hora**: O instante da semana onde teve início o consumo de água, no formato "DDD:HH:mm:ss". Neste caso específico, o campo do dia [DDD] apenas deve tomar valores entre [000] e [006], sendo que o primeiro correspondente à segunda-feira seguindo-se depois os outros dias da semana pela ordem natural;
- **Duração**: Tempo total durante o qual foi utilizada a água, no formato "HH:mm:ss";
- **Água consumida por segundo**: Quantidade de água quente consumida (água que sai do tanque) em $m^3/s$

É importante referir que este perfil tem cariz semanal, isto é, o mesmo perfil de consumo de água será usado repetidamente para todas as semana da simulação.

Os perfis de utilização de água criados para as simulações foram adaptados da tabela "*Tapping Cycle M*" que dita os ciclos de tiragem de água ao longo de um dia. Esta foi extraída da norma europeia "EN 50440", que define requisitos e testes para classificação de unidades DEWH. A versão disponibilizada desta tabela encontra-se no Apêndice A.

O fluxo de água considerado nas tiragens foi de $5\,l/min$ para tiragens do tipo"*Small*" e de $10\,l/min$ para os restantes tipos. Assim sendo, tomando como exemplo a primeira tiragem do tipo "*Small*" da tabela, que ocorre às 7:00 horas e gasta $1.8\,l$ de água a uma taxa de $5\,l/min$, a sua transposição para o formato do perfil descrito, terá o seguinte aspeto: "000:07:00:00, 00:00:22, 0.000083333334" em que

$$\begin{gathered} \left\lceil \frac{1.8\ (l)}{5\ (l/min)}\ 60\ (s) \right\rceil = 22\ (s) \\ 5\ (l/min) \approx 0.000083333334\ (m^3/s) \end{gathered} .$$

Desta forma foram construídos vários perfis com algumas variações entre eles, que permitem simular múltiplas situações relevantes para a avaliação dos resultados dos algoritmos.

Quanto aos perfis de temperatura, estes permitem representar a variação da temperatura ao longo de um dia. Neste caso serão utilizados para caracterizar a temperatura do ar exterior e a temperatura da água que dá entrada no tanque. Cada registo representa a temperatura num dado instante do dia através dos seguintes campos (ordenadamente):

- **Hora do dia**: O instante do dia representado no formato "HH:mm:ss";
- **Temperatura**: A temperatura em graus Celsius para o respetivo instante.

Nos intervalos de tempo entre dois registos, a temperatura em cada um desses instantes é dada por interpolação linear, para suavizar as mudanças de temperatura e assim tornar o perfil mais realista.

Cada perfil de temperatura tem que ter obrigatoriamente pelo menos uma linha que deve corresponder à temperatura no instante inicial. Se esse for o único registo que consta no perfil, então a temperatura será considerada como constante durante todo o dia. Da mesma forma, se não for especificada a temperatura para o instante final do dia, a temperatura do último registo será usada até a conclusão do dia.

De notar que este tipo de perfil é apenas diário, significando isto que o mesmo perfil será usado repetidamente para todos os dias da simulação. Contudo estes já conferem alguma flexibilidade ao modelo implementado para prover mais realismo em situações em que a diferença entre temperaturas máxima e mínima (diárias) seja significativamente grande para que possa causar impacto visível nos resultados dos algoritmos.

Os perfis de temperatura devem ser construídos com base numa aproximação realista do que será a temperatura ambiente diária, observada durante uma época do ano específica (adequada a situação simulada).

### 4.1.2 Servidor e Protocolo de Comunicação

O simulador foi dotado de uma funcionalidade que permite que sejam utilizados na simulação algoritmos implementados fora da aplicação, como alternativa aos implementados no ambiente *Java*. Esta característica foi pensada especialmente para ser possível testar os algoritmos depois de estes serem portados para a plataforma de desenvolvimento *Arduino*.

Esta funcionalidade baseia-se no modelo cliente-servidor em que o servidor aguarda comandos do cliente e atua adequadamente. Nesta situação, o simulador interpreta o papel de servidor, e a plataforma de desenvolvimento comporta-se como cliente.

O servidor, no ponto de vista da simulação, vai substituir funcionalmente a tarefa dos algoritmos de decisão incorporados no simulador. A decisão passa então a ser tomada pelo cliente, que deverá correr um algoritmo de decisão análogo aos implementados, tendo em conta as suas restrições de *hardware*.

O servidor vai manipular uma instância do *TankModel* para fazer a "ponte" entre esta e o algoritmo executado no cliente. Como meio de comunicação é utilizada a interface USB (*Universal Serial Bus*) com recurso a um protocolo simples (apresentado de seguida) mas que permite realizar todas as operações necessárias para realizar a simulação.

Convém deixar clarificado que a velocidade real da simulação, no modo servidor, é ditada pela frequência a que o cliente realiza os pedidos, e para além disso está restringida pelos

atrasos impostos pelo meio de comunicação. É de antever que seja mais lento que a simulação com recurso aos algoritmos locais.

O servidor, assim que é iniciado, fica a aguardar que o cliente envie pedidos na forma de pacotes de 2 *bytes*, onde é indicado o número de sequência e o(s) comando(s) a executar.

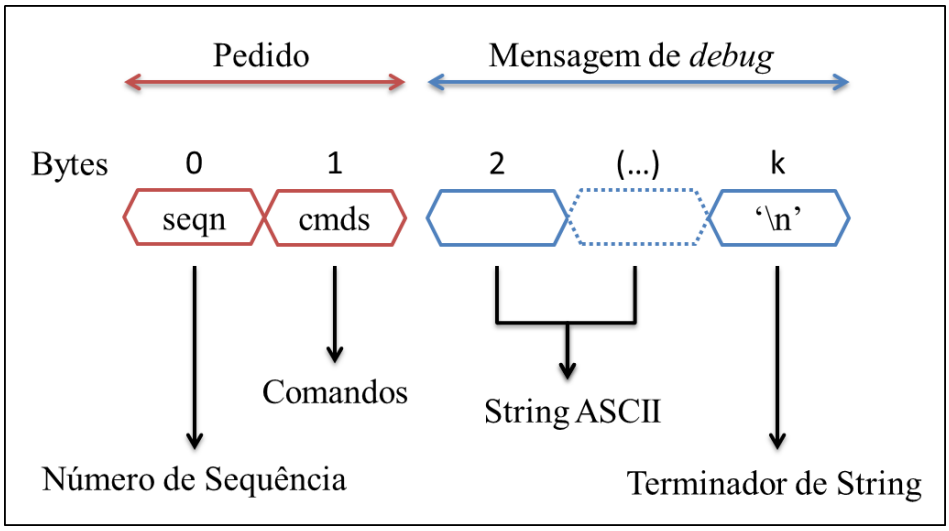


Figura 4.2: Representação da estrutura de um pedido, no protocolo implementado.

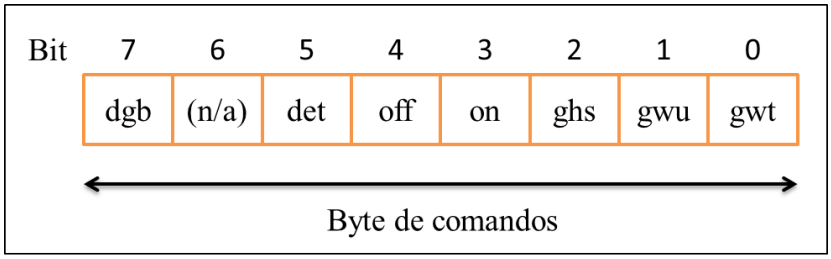


Figura 4.3: Representação detalhada do *byte* de comandos.

O primeiro *byte* contém o número de sequência e tem como função indicar o número da amostra que é pedida ao servidor. Este número é circular, isto é, quando o número chega ao seu máximo (255) o próximo número da sequência é 0. Podem no entanto ser envidas mensagens consecutivas com o mesmo número de sequência, para efectuar diferentes pedidos e operações no mesmo instante de simulação.

Quando se pretende avançar para o instante seguinte (do lado do servidor), o número de sequência deve ser incrementado em 1 de forma a sinalizar o servidor para calcular o próximo valor. Pedidos em que o número de sequência avançou mais do que uma unidade indicam ao servidor para calcular todos os valores até ao número de sequência atual, descartando os que ficaram para trás e não foram usados. Sendo este campo de um *byte*, esta particularidade permite avançar 255 valores de uma só vez em cada pedido. Por este motivo, o número de sequência do pedido inicial deve ser sempre 0.

No segundo *byte*, detalhado na Figura 4.3, vão indicados os comandos a realizar em cada pedido. Esses comandos vão mapeados em *bit flags* (lógica positiva), em que cada *bit* indica um determinado comando.Os comandos podem ser divididos em 3 categorias:

- *Request* (pedido de valores)
  - ***gwt (get water temperature)***: pedido da temperatura da água do tanque;
  - ***gwu (get water usage)***: pedido de informação sobre o consumo de água (usado apenas para *debug*);
  - ***ghs (get heater status)***: pedido de informação sobre o estado da resistência (usado apenas para *debug*).
- *Set* (definir estado da resistência)
  - ***on***: Liga a resistência;
  - ***off***: Desliga a resistência.
- *Debug*
  - ***det***: Indica que foi detetado consumo de água no intervalo anterior. Este parâmetro não afeta o funcionamento do algoritmo pois serve apenas para construir e visualizar no simulador o perfil de consumo de água detetado pelo algoritmo;
  - ***dgb***: Indica ao servidor que foi acoplada uma mensagem de *debug* (textual) no fim do pacote.

Em cada pedido só pode ser executado um comando de cada categoria, ou seja, o servidor processa até 3 comandos por pedido, desde que seja um de cada categoria diferente. No caso de serem estabelecidos múltiplos comandos da mesma categoria no mesmo pedido, o servidor apenas executa um deles conforme a ordem de atendimento estabelecida no servidor.

Os comandos para ligar (*on*) ou desligar (*off*) a resistência, assim como o comando para indicar deteção de consumo de água (*det*), só são processados para um novo número de sequência, sendo estes ignorados em todos os pedidos consecutivos com o mesmo número de sequência.

Quando o *bit* de *debug* é assinalado, o servidor interpreta todos os *bytes* posteriores ao pedido como caracteres *ASCII*, parando quando encontrar o terminador '\n'.

Relativamente à resposta do servidor ao pedido de um determinado valor, este é adaptado e enviado num pacote de 2 *bytes*. Como disposto na Figura 4.4, o primeiro *byte* contém os 8 *bits* mais significativos e o segundo *byte* os 8 *bits* menos significativos.

O valor da temperatura da água é codificado no servidor em vírgula flutuante, mas para facilitar o envio, ele é multiplicado por 100 e convertido para um inteiro de 16 *bits*. Desta forma é possível representar uma temperatura positiva até aos 655 graus, com uma resolução fixa de duas casas decimais e num formato adequado à plataforma *Arduino*.

Os valores de consumo de água e do estado da resistência estão representados por variáveis booleanas. É enviado 0 quando a variável em causa for "false" ou enviado 1 quando for "true". Estes valores deverão ser usados apenas na comparação com a informação ditada pelos algoritmos para facilitar as tarefas de *debug*, não podendo ser usados diretamente nas tomadas de decisão dos algoritmos.

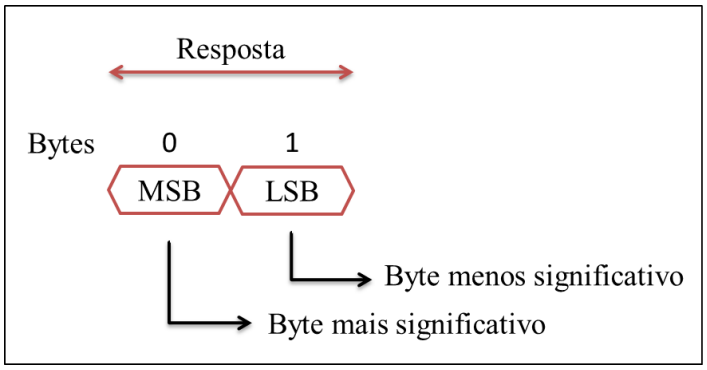


Figura 4.4: Representação da estrutura de uma resposta, no protocolo implementado.

## 4.2 Interface Gráfica

Para facilitar a utilização do simulador, a definição dos seus parâmetros e também fornecer meios para uma visualização intuitiva dos resultados em forma de gráficos, o simulador foi desenvolvido juntamente com uma interface gráfica interativa. A janela principal do simulador aparece representada na Figura 4.5.

Seguidamente, serão abordadas as várias funcionalidades que são suportadas pela interface do utilizador e o respetivo método de utilização.

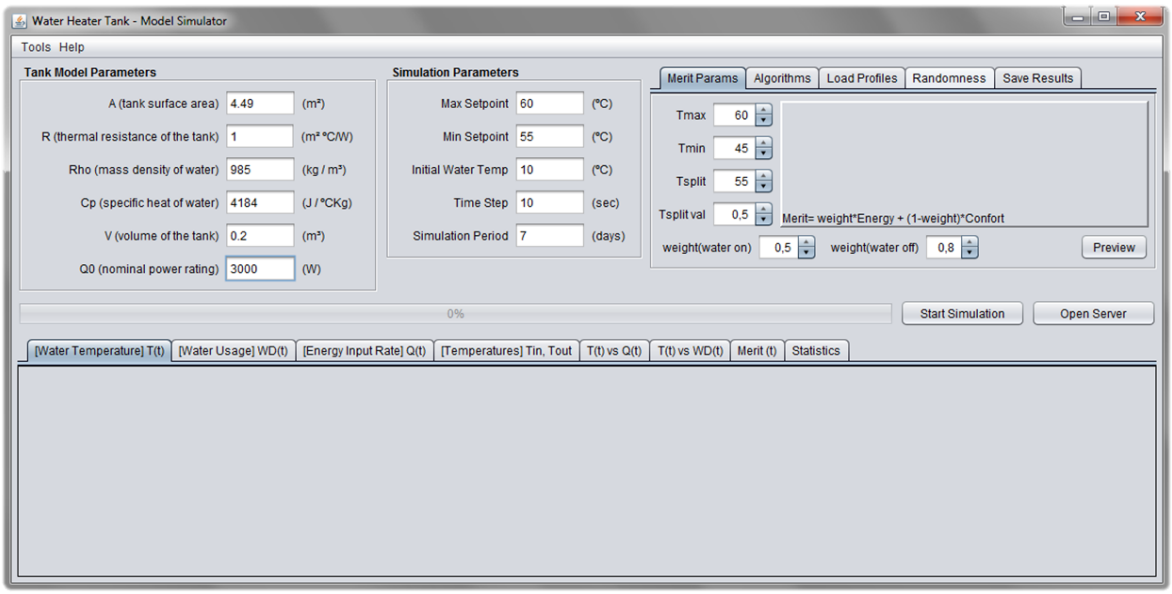


Figura 4.5: Imagem da janela do simulador.

O grupo "*Tank Model Parameters*" contém as caixas de texto onde se estabelecem as constantes do modelo em conformidade com as unidades indicadas. Estas constantes englobam as características do tanque (área da superfície, resistência térmica, volume e potência nominal) e aproximações de algumas propriedades da água necessárias ao modelo (densidade e calor específico).

O grupo denominado "*Simulation Parameters*" contém as caixas de onde são definidos os parâmetros da simulação: os *setpoints* máximo e mínimo, a temperatura inicial da água, o *time step* (ou intervalo de amostragem) e duração da simulação.

A aplicação encarrega-se de povoar automaticamente todos estes campos com valores pré-definidos, no arranque.

O separador "*Merit Params*" mostrado na Figura 4.6 permite definir os valores parametrizáveis da função de mérito. O botão "*Preview*" exibe o gráfico que representa a "função de conforto" caracterizada pelos parâmetros estabelecidos.

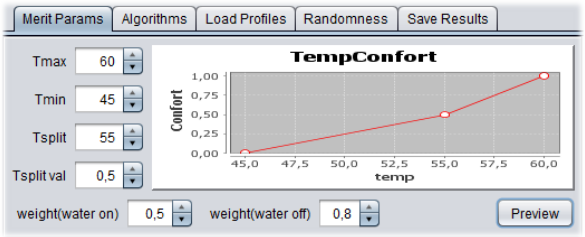


Figura 4.6: Imagem do separador "*Merit Params*".

- **"*Tmax*"**: Valor da temperatura máxima, a partir do qual a função de conforto atribui a pontuação 1.
- **"*Tmin*"**: Por contraste com o anterior, este valor estabelece o valor da temperatura mínima aceitável, indicando o limite inferior para a pontuação 0.
- **"*Tsplit*" e "*Tsplit val*"**: Estes dois valores juntos formam um ponto de interceção na função de conforto que permite dividir a função em duas retas, sendo assim possível ajustar a evolução da pontuação em função da temperatura para que esta não seja completamente linear.
- **"*Weight (water on)*"**: Valor que representa a contribuição (peso) da função de poupança de energia na função de mérito, quando existe consumo de água.
- **"*Weight (water off)*"**: Analogamente, este valor representa o peso da função de poupança de energia na função de mérito, mas agora para quando não existe consumo de água.

O separador "*Algorithms*" visível na Figura 4.7, é o local que permite escolher um algoritmo dos vários que foram implementados, para ser usado na simulação. É ao algoritmo selecionado que cabe a decisão de ligar ou desligar a resistência em cada instante da simulação.

Da lista de algoritmos disponíveis constam:

- **"Basic Setpoint Based Algorithm"**: Implementação do algoritmo de decisão típico que usa apenas os *setpoints* na tomada de decisão (Secção 3.1.1);
- **"Reactive (Water Usage Detection) Algorithm"**: Algoritmo usado para testar e observar o funcionamento do algoritmo de deteção de consumo de água (Secção 3.2). A tomada de decisão é semelhante ao algoritmo anterior, com a particularidade de este ligar também a resistência nos instantes onde é detetado consumo de água;

- **"History Based Algorithm v2"**: Versão final do algoritmo de decisão com histórico semanal, tal como apresentado na Secção 3.1.2;

- **"History Based Algorithm v1"**: Visão inicial do algoritmo de decisão com histórico semanal. Esta versão não utiliza o algoritmo detetor de consumo de água (usa apenas um valor de limiar para realizar a deteção) e para além disso utiliza uma estrutura de dados, para armazenamento do histórico, não otimizada. Esta versão foi mantida e disponibilizada apenas para demonstrar a evolução do algoritmo comparativamente à versão final.


Figura 4.7: Imagem do separador "*Algorithms*".

Para além dos parâmetros do modelo do tanque e da simulação, é necessário ainda definir os perfis para o consumo de água ("*Water Usage*"), temperatura da água que entra no tanque ("*Inlet Water Temp*") e a temperatura do ar exterior ("*Exterior Air Temp*"). O carregamento destes perfis é efetuado através do separador "*Load Profiles*" como apresentado pela Figura 4.8.

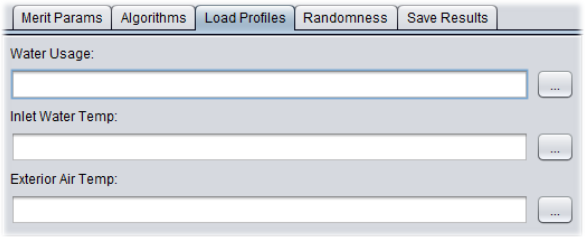


Figura 4.8: Imagem do separador "*Load Profiles*".

No separador "*Randomness*" é oferecida a possibilidade de impor alguma aleatoriedade ao perfil de consumo de água escolhido. Ativando esta funcionalidade, faz com que se sucedam desvios temporais aleatórios em cada ocorrência de utilização de água presente no perfil de consumo. A aleatoriedade depende da distribuição selecionada, uniforme ou Gaussiana. Na distribuição uniforme (Figura 4.9) é necessário definir o intervalo relativo para a geração do novo instante aleatório.

No caso da distribuição Gaussiana ou normal (Figura 4.10), é necessário estabelecer o desvio padrão e o intervalo de *cutoff* relativo. O intervalo de *cutoff* serve para forçar limites aos valores gerados, ou no caso de ser selecionada a opção "Remove Occurrences Outside Cutoff Limits", as ocorrências de consumo de água cujo novo instante aleatório calhe fora do intervalo serão removidas do perfil (nessa semana).

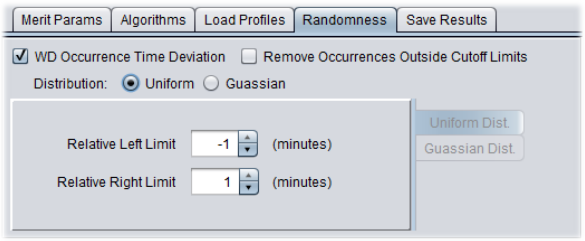


Figura 4.9: Imagem do separador "*Randomness*" - distribuição uniforme.

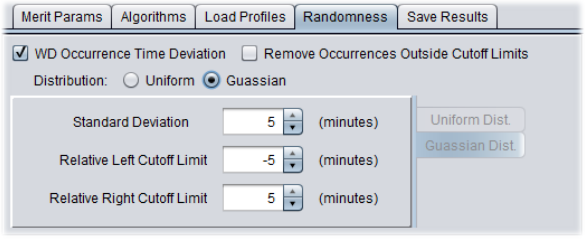


Figura 4.10: Imagem do separador "*Randomness*" - distribuição Gaussiana.

No separador "*Save Results*" existem várias opções para exportar os resultados mais importantes para serem usados como referência futura. Todos os dados são guardados num formato CSV para poderem ser facilmente interpretados por ferramentas externas e assim permitir análises mais detalhadas. Neles são representados os instantes de tempo de toda a simulação (em segundos) e os respetivos resultados em questão. As estatísticas são guardadas em formato HTML.

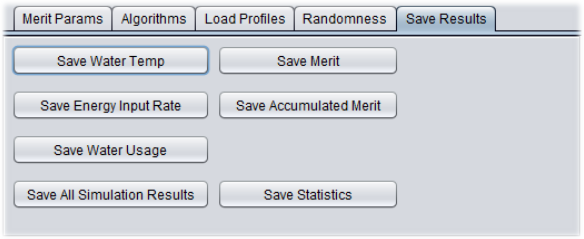


Figura 4.11: Imagem do separador "*Save Results*".

- **"Save Water Temp"**: Guarda o registo da temperatura da água para cada instante da simulação.
- **"Save Energy Input Rate"**: Guarda o registo da energia consumida ao longo do tempo.
- **"Save Water Usage"**: Guarda o registo de água consumida em função do tempo (formato alternativo ao perfil de consumo de água).
- **"Save Merit"**: Guarda o valor de mérito (instantâneo) ao longo da simulação.

- **"Save Accumulated Merit"**: Guarda a média do valor de mérito adquirido até cada um dos instantes da simulação.
- **"Save All Simulation Results"**: Guarda todos os resultados mencionados até agora, num único ficheiro.
- **"Save Statistics"**: Guarda as estatísticas resultantes da simulação.

Relativamente ao conjunto de separadores na região inferior da janela do simulador (Figura 4.12), estes estão reservados para a visualização dos resultados da simulação, dos perfis carregados e também para expor algumas estatísticas resultantes da simulação. As estatísticas são apresentadas em formato textual enquanto os que restantes resultados e perfis, são representados por forma de gráficos em ordem ao tempo.

Nos gráficos existem funcionalidades de *zoom* que podem ser executadas acedendo ao menu de contexto, utilizando o botão de *scroll* do rato ou ainda selecionando uma zona do gráfico para ampliar.

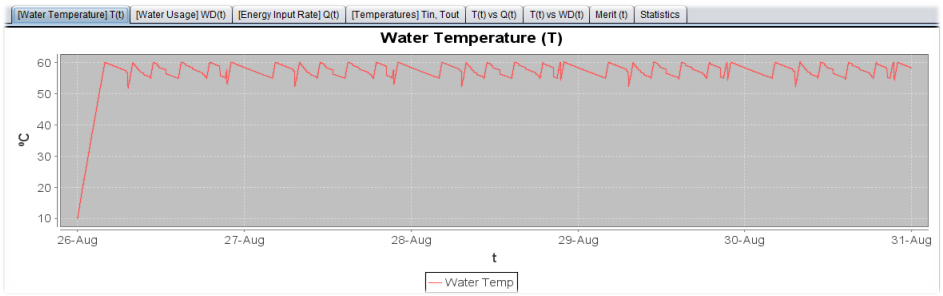


Figura 4.12: Exemplo de um gráfico do separador "*Water Temperature*".

- **T(t)**: Gráfico da temperatura da água no tanque ao longo do tempo;
- **WD(t)**: Gráfico do perfil de consumo de água efetivo e o respetivo resultado dado pelo algoritmo de deteção de consumo de água (quando aplicável);
- **Q(t)**: Gráfico da energia consumida ao longo do tempo;
- **Tin e Tout**: Gráficos dos perfis de temperatura da água de entrada no tanque e de temperatura do ar exterior, respetivamente;
- **Merit(t)**: Gráficos relativos à função de mérito.

Há que ter em conta que, por questões de facilidade da utilização da API *JFreeChart*, o eixo temporal dos gráficos é representado com uma data do calendário a começar sempre no início da semana da data da execução da simulação. Esta informação de calendário não tem no entanto qualquer impacto no desempenho dos algoritmos e serve essencialmente para a visualização gráfica, não sendo sequer exportada ao guardar os resultados.

Após todos os parâmetros estarem definidos e os perfis selecionados, pode dar-se início à simulação através do botão "*Run Simulation*". O progresso da simulação até à sua conclusão será indicado na barra de progresso disposta no centro da janela principal (ver Figura 4.5).

A funcionalidade de servidor que foi apresentada previamente está disponível através do botão "*Open Server*". A janela correspondente a esta funcionalidade está visivel na Figura 4.13. Nela pode ser selecionado o porto onde está ligado o dispositivo cliente e configurar a respetiva taxa de *bits* ("*Serial Data Rate*").

Na área de texto de "*Log*" são registados os eventos entre o cliente e o servidor, mensagens de estado e outros elementos relevantes para debug. O texto no "*Log*" pode ser limpo através do botão "*Clear Log*".

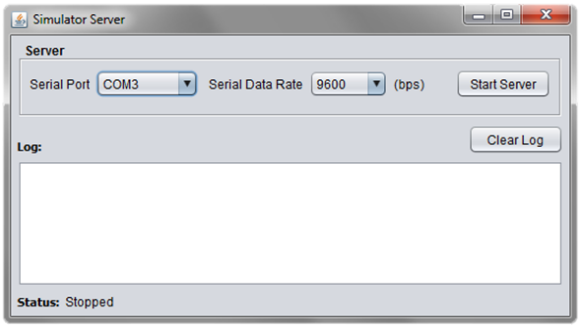


Figura 4.13: Imagem da janela do servidor.

O botão "Start Server"inicializa o servidor, sendo que a simulação pode ser interrompida a qualquer momento, mesmo antes do tempo total de simulação estabelecido ser alcançado, parando o servidor ou fechando a janela. No canto inferior esquerdo da janela é apresentado o progresso no decorrer da simulação.

Após o fecho desta, os resultados da simulação produzidos pelo servidor até ao momento são mostrados na janela principal tal como acontece com a simulação normal.

A ferramenta de cálculo de energia é acessível através da barra de menus, pelo menu "*Tools*" e selecionando a opção "*Energy Calculator*". Esta ferramenta permite calcular o total de energia gasta (mais propriamente o número de vezes que a resistência se encontra ligada) num dado intervalo de tempo especificado na interface de utilizador (Figura 4.14).

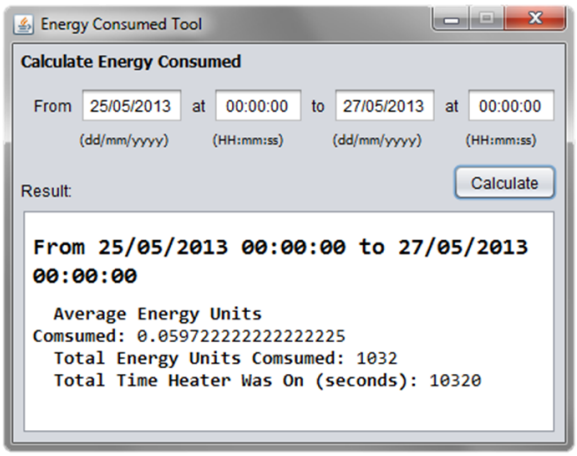


Figura 4.14: Imagem da janela da ferramenta de cálculo de energia.

Desta forma dá-se por terminada a apresentação da interface gráfica do simulador e o seu modo de utilização. Conclui-se assim também este capítulo dedicado ao simulador, onde foi mostrada a sua importância e detalhado o seu processo de funcionamento, assim como todas as suas funcionalidades. Foi também abordada a capacidade de executar a simulação, correndo algoritmos numa plataforma de desenvolvimento ligada por USB, com recurso ao modo servidor, o seu protocolo de comunicação simples, e a sintaxe dos ficheiros dos perfis.

# Capítulo 5

# Implementação dos Algoritmos Inteligentes

Com os algoritmos já desenvolvidos e testados no ambiente de simulação com os recursos computacionais de um computador pessoal, foi então necessário avaliar a eficácia dos mesmos numa plataforma de desenvolvimento com restrições de recursos.

Como o alvo dos algoritmos serão microcontroladores de 8-bits, no âmbito deste trabalho foi disponibilizada uma plataforma de desenvolvimento *Arduino* (Mega 2560), dotada de um microcontrolador ATmega2560 da linha de microcontroladores de 8-bits de baixo consumo da Atmel©, com uma arquitetura *RISC* (*Reduced Instruction Set Computing*).

Nesta secção serão apresentadas as características e limitações do *hardware* em questão, assim como os detalhes tidos em conta na adaptação do código dos algoritmos de *Java* para *C++*, linguagem suportada pelos compiladores do *Arduino*. Para finalizar, será validado o correto comportamento dos algoritmos através do simulador e da sua funcionalidade de servidor previamente apresentada.

## 5.1 Restrições do *Hardware*

Para analisar os recursos e as possíveis restrições impostas pelo *hardware* alvo, foi tida como referência as necessidades do algoritmo "History Based Algorithm v2". O microcontrolador ATmega2560 trabalha a uma frequência máxima de 16 MHz, mas oferece vários modos de poupança de energia, denominados "Sleep Modes", para ser possível economizar energia nos momentos livres (entre cada instante de amostragem). Estão ainda disponíveis vários *timers*/contadores para gerir a passagem do tempo e gerar as interrupções necessárias para proceder à execução do algoritmo nos instantes de amostragem [18].

Em termos de memória, este microcontrolador tem vários modos de armazenamento, cada um com a sua função. Relativamente à memória *flash* reprogramável, estão disponíveis 256 KB, dos quais 8 KB estão reservados para o *bootloader*. Esta memória vai servir de armazenamento para o código dos programas carregados no microcontrolador [18].

A quantidade disponibilizada é mais que suficiente para os algoritmos desenvolvidos, visto que após a compilação do programa com o algoritmo mais complexo ("History Based Algorithm v2"), o ficheiro binário apresenta aproximadamente 10500 *bytes*, isto já com a inclusão da biblioteca base do *Arduino* e das bibliotecas de acesso à EEPROM e comunicação série,

usando apenas 4% da memória disponível.

Como memória persistente, este microcontrolador disponibiliza 4 KB de EEPROM [18]. Esta vai ser o principal recurso para armazenamento do conhecimento adquirido pelo algoritmo, ou seja, o histórico semanal. Sabe-se que cada registo do histórico ocupa 1 byte e que o número de registos necessários (para abranger todos os instantes diários) depende do intervalo de amostragem. Na Tabela 5.1 estão apresentados os requisitos de memória para vários intervalos de amostragem. Pode comprovar-se que mesmo com um intervalo pequeno de 1 minuto, apenas são necessários 1440 bytes, deixando ainda mais de metade da EEPROM livre.

| **Intervalo de amostragem (minutos)** | **Espaço necessário (*bytes*)** |
|---|---|
| 15 | 96 |
| 10 | 144 |
| 5 | 288 |
| 2 | 720 |
| 1 | 1440 |

Tabela 5.1: Memória necessária para vários intervalos de amostragem.

A memória SRAM, também conhecida como memória principal ou de trabalho, tem capacidade de 8 KB e é usada para manter as variáveis e estruturas de dados durante a execução do programa [18].

No algoritmo, o maior foco de ocupação de memória prende-se com a estrutura de armazenamento do histórico de consumo de água semanal. Essa estrutura, como referido anteriormente, foi pensada de forma a ser armazenada na memória persistente do microcontrolador, libertando assim a SRAM desse encargo. Sendo que esta memória tem o dobro da capacidade da EEPROM, isto permite manter cópias do histórico semanal com valores temporários ou até estruturas maiores e mais complexas. Embora os algoritmos desenvolvidos não tirem proveito deste facto, esta informação abre possibilidades a futuros melhoramentos nos algoritmos.

Retomando o tópico da memória persistente (EEPROM) e abordando as velocidades de acesso da mesma, o fabricante especifica que o tempo de acesso típico a uma posição de memória (1 byte) é de $3.3ms$. Como em cada iteração do algoritmo vai existir apenas uma leitura e uma escrita (no pior caso), os tempos de acesso podem ser considerados desprezáveis.

Outra característica a ter em conta é o número total de escritas que esta memória suporta, sem perda da sua integridade. Nas especificações do microcontrolador é garantida uma duração mínima de $100,000$ ciclos de escrita [18]. Isto significa que para alcançar esse número de escritas, sabendo que cada registo só será escrito no máximo uma vez por dia, era necessário correr o algoritmo durante mais de 200 anos.

Para finalizar, a limitação óbvia dos microcontroladores de 8-bits é o facto do barramento de dados ser de 8-bits. Esta limitação implica que operações com variáveis de 16 ou 32-bits são divididas pelo compilador num conjunto de operações equivalentes. Esta particularidade é tida em conta na adaptação dos algoritmos para esta plataforma, reduzindo ao máximo o tamanho das variáveis usadas e o recurso a operações de vírgula flutuante.

## 5.2 Migração para a Plataforma de Desenvolvimento

Continuando o raciocínio anterior, a maior parte das alterações feitas nos algoritmos prende-se com a adaptação das variáveis a tamanhos o mais compactos possível. Como já foi referido na Secção 4.1.2, dedicada ao protocolo de comunicação do simulador, a temperatura da água antes de ser transmitida ao cliente é multiplicada por 100 e codificada num inteiro de 2 bytes. Para manter o rigor, as variáveis da temperatura e todas as variáveis relacionadas (e.g., diferença de temperatura, centro e limites dos *clusters*) foram alteradas em conformidade.

Sempre que é possível são usadas variáveis de 8-bits, como é o caso de valores booleanos e pequenos contadores. O único local onde foi necessário recorrer a operações de 32-bits foi na função de cálculo do centro dos *clusters* do algoritmo de deteção.

Outra modificação relevante que merece destaque foi a necessidade de adaptar a gestão do tempo que no *Java* era feita com recurso a classe *Calendar* e agora, no código produzido para o *Arduino*, foi simplificada por forma a manter apenas a informação relevante. Por essa razão, são usadas apenas duas variáveis para o efeito, uma para armazenar o instante temporal diário e outra para guardar o dia da semana. De notar que não é necessário concordância entre a informação temporal do programa e o tempo real, no sentido de que não é relevante que o algoritmo não saiba qual a hora real ou o dia correto, sendo apenas importante que a passagem do tempo seja autêntica.

Relativamente à tradução de *Java* para *C++*, em termos de sintaxe as duas são bastante idênticas e visto tratarem-se de linguagens com metodologia de programação orientada a objetos, apenas é necessário adaptar as classes *Java* à estrutura equivalente do *C++*, com um ficheiro de interface ou *header* (.h) e um ficheiro de implementação (.cpp).

O desenvolvimento do código nesta fase foi feito com recurso ao IDE "MariaMole", um ambiente para programação e gestão de projetos que oferece mais flexibilidade do que o *software* base do *Arduino*, mantendo a comodidade e facilidade de utilização deste último.

O projeto foi dividido nos seguintes ficheiros de código:

- "**simplealgorithm.***": contêm o código relativo ao algoritmo "Basic Setpoint Based Algorithm";
- "**intelalgorithm.***": versão adaptada do algoritmo "History Based Algorithm v2";
- "**waterusagedetector.***': adaptação do algoritmo de deteção de consumo de água;
- "**weightedlimit.***": classe auxiliar do algoritmo de deteção de consumo de água, para armazenar os valores dos limites dos *clusters*;
- "**histstorage.***": classe construída para a gestão da estrutura do histórico semanal e a interação com a memória EEPROM;
- "**simulclient.***": implementação do cliente para comunicação série com o simulador, com recurso ao protocolo explanado anteriormente. É essencial para pedir as temperaturas ao simulador e é também usado para transmitir ao servidor as decisões dos algoritmos;
- "**main.***": ficheiros do programa principal onde são inicializados os vários componentes do programa e é implementada a rotina de amostragem da temperatura. Esta rotina

resume-se à leitura da temperatura da água no tanque, aplicação do algoritmo e tomada de decisão (ligar/desligar a resistência). De notar que, antes de carregar o código para a placa, é necessário estabelecer neste conjunto de ficheiros o intervalo de amostragem, tal como definido no simulador, e selecionar qual o algoritmo a executar.

No programa principal existem duas funções fundamentais que são interpretadas de forma especial pelo *Arduino*: as funções "setup()" e "loop()". A função de *setup* é executada sempre que a placa é (re)inicializada. A rotina *loop* é invocada posteriormente à conclusão da rotina de *setup*, e como o próprio nome indica, é executada em ciclo infinito daí em diante.

Neste projeto, em vez de programar uma rotina de serviço à interrupção que seria chamada em intervalos de tempo constantes (dado pelo intervalo de amostragem), optou-se por colocar esse código diretamente na função "loop()" e simular a passagem do tempo através de variáveis de contagem (para o instante diário e o dia da semana) para assim maximizar a frequência de execução das iterações e concluir a simulação o mais rápido possível.

Para comprovar se a adaptação foi efetuada corretamente, foi testado o algoritmo com histórico semanal numa simulação de 14 dias, através do modo servidor do simulador, com um intervalo de amostragem de 5 minutos e um perfil de utilização de água idêntico nas duas semanas. A duração real da simulação foi de aproximadamente de 10 minutos e o resultado observado foi o esperado. Como tinha sido demonstrado anteriormente na secção relativa ao algoritmo com histórico semanal, a decisão do algoritmo durante a primeira semana é apenas baseada nos valores dos *setpoints*, enquanto que na segunda semana, já com um historial construído, o algoritmo prevê corretamente os instantes de maior consumo de água e pré-aquece a água em concordância. Tal comportamento pode ser observado na Figura 5.1 retirada do simulador.

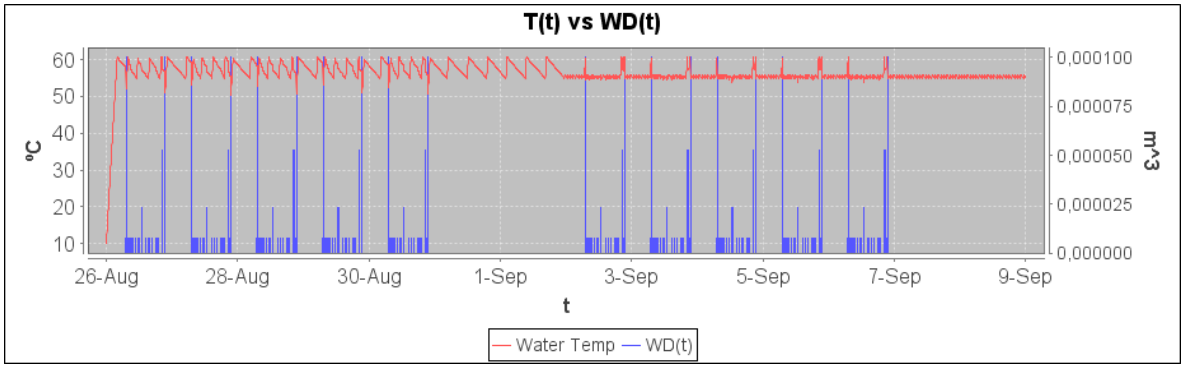


Figura 5.1: Representação da temperatura face ao perfil de utilização de água, ao longo de duas semanas, executando o algoritmo pelo *Arduino*.

Termina assim o capítulo dedicado à implementação dos algoritmos na plataforma de desenvolvimento onde foram esclarecidas as limitações do *hardware* e o impacto que estas tiveram na adaptação do código. Concluiu-se que as limitações ainda deixam uma margem de manobra significativa para o desenvolvimento de estruturas de dados mais complexas e algoritmos mais preponderantes.

# Capítulo 6

# Resultados e Conclusões

Neste último capítulo apresenta-se uma breve discussão sobre a implementação dos algoritmos, seguida da comparação dos seus resultados, finalizando com notas sobre os objetivos cumpridos e desenvolvimentos futuros.

## 6.1 Discussão da Implementação

O desenvolvimento do algoritmo "inteligente" tem origem na tentativa de melhorar a resposta do algoritmo tradicional em situações em que é possível prever o consumo de água antes de este acontecer.

Relembrando, o algoritmo tradicional apenas se propunha a cumprir o requisito básico de manter a temperatura da água dentro do intervalo de temperatura estabelecido pelo utilizador. Contudo, mesmo assim, não é garantido que a temperatura não desça abaixo do limiar inferior em casos limite. Por exemplo, se ocorrer uma tiragem intensa de água quente quando a temperatura desta já está próxima do limiar inferior e a taxa de aquecimento do tanque não é suficientemente alta para conseguir acompanhar o consumo de água, é de esperar que a água atinja temperaturas inferiores ao limite mínimo de conforto desejado.

Outra grande desvantagem óbvia deste algoritmo, dada pelo facto de a água ser constantemente e incondicionalmente aquecida até ao limar máximo, resulta na perda de energia (desperdício do calor armazenado) nas horas em que não há qualquer consumo de água quente.

O algoritmo "inteligente" surgiu então com a potencialidade de colmatar as falhas do algoritmo tradicional. Este algoritmo observa o perfil de consumo de água ao longo da semana, constrói o respetivo histórico, e posteriormente usa-o como base para antecipar a utilização de água da semana seguinte.

Devido à forma como o histórico é gerido, o algoritmo só armazena a informação relativa à semana transata. Isto implica que, se o perfil de consumo de água quente semanal do utilizador for semelhante de semana para semana, as desvantagens do algoritmo anterior são minoradas. Em contrapartida, perante um consumo semanal diferente todas as semanas, isto é, com poucos ou nenhuns intervalos de tiragem de água quente próximos ou sobrepostos entre semanas consecutivas, leva a que o algoritmo produza os seus piores resultados.

Relativamente ao desempenho dos algoritmos executados no simulador (implementação *Java*) comparativamente aos algoritmos executados no *Arduino*, tal como foi mencionado no

Capítulo 5, apresenta resultados semelhantes. As pequenas discrepâncias que podem existir entre os dois resultados decorrem apenas da perda de precisão dos valores da temperatura, quando estes são transmitidos ao *Arduino*.

Outra diferença percetível prende-se com a velocidade da simulação entre os dois métodos. A execução dos algoritmos no modo servidor é evidentemente mais lenta que a execução dos algoritmos implementados em *Java*. Na Tabela 6.1 pode observar-se o tempo decorrido na execução dos algoritmos nas duas plataformas diferentes (*Java* e *Arduino*). Os valores apresentados resultam da média de 5 execuções consecutivas do algoritmo tradicional com um intervalo de amostragem de 5 minutos num período de simulação de 7 dias. A taxa de transmissão de *bits* usada no modo servidor foi de 9600 bps.

| **Plataforma** | **Tempo decorrido (ms)** |
|---|---|
| *Java* | 135 |
| *Arduino* | 79626 |

Tabela 6.1: Exemplo do tempo de execução do algoritmo tradicional nas diferentes plataformas.

Por estes motivos, a simulação com recurso ao *Arduino* foi usada exclusivamente para a validação e teste da adaptação dos algoritmos a esta plataforma. Para a comparação da eficácia dos diferentes algoritmos foram usadas as implementações em *Java* incorporados no simulador.

## 6.2 Resultados e Comparações

De seguida são apresentados os resultados do desempenho dos algoritmos, perante situações de teste com carácter realista e também algumas situações especialmente escolhidas para acentuar as características de cada algoritmo. Consequentemente, é apresentada uma análise comparativa com o intuito de revelar em que situações é que o algoritmo "inteligente" é uma mais-valia. Os gráficos presentes nesta secção foram obtidos a partir do simulador desenvolvido.

Todas as simulações apresentadas nos exemplos seguintes têm em comum os seguintes valores dos parâmetros do modelo do tanque:

- $A$ (área da superfície do tanque) $= 4.49\ (m^2)$;
- $R$ (resistência térmica do tanque) $= 1\ (m^2\,^\circ C/W)$;
- $\rho$ (densidade da água) $= 985\ (kg/m^3)$;
- $c_p$ (calor específico da água) $= 4184\ (J/^\circ C\,kg)$;
- $V$ (volume do tanque) $= 0.2\ (m^3)$;
- $Q_0$ (potência nominal) $= 3000\ (W)$.

Para os vários testes foram usados dois perfis diferentes de consumo semanal de água, num ambiente doméstico, mas ambos foram inspirados na tabela disponibilizada no Apêndice A, referente à norma europeia "EN 50440". O primeiro perfil (Figura 6.1) tem como objetivo caracterizar uma situação habitual de consumo de água durante uma estação quente. Por oposição, o segundo perfil (Figura 6.2) descreve o consumo de água durante uma estação fria, que diverge do primeiro perfil na medida em que apresenta tiragens de água quente mais frequentes e intensas. De notar que, nestes perfis, apenas são tidas em conta as tiragens de água relevantes, sendo que pequenos consumos nem são detetados pelo algoritmo (como se pode observar nas Figuras 6.1, 6.2 e 6.3).

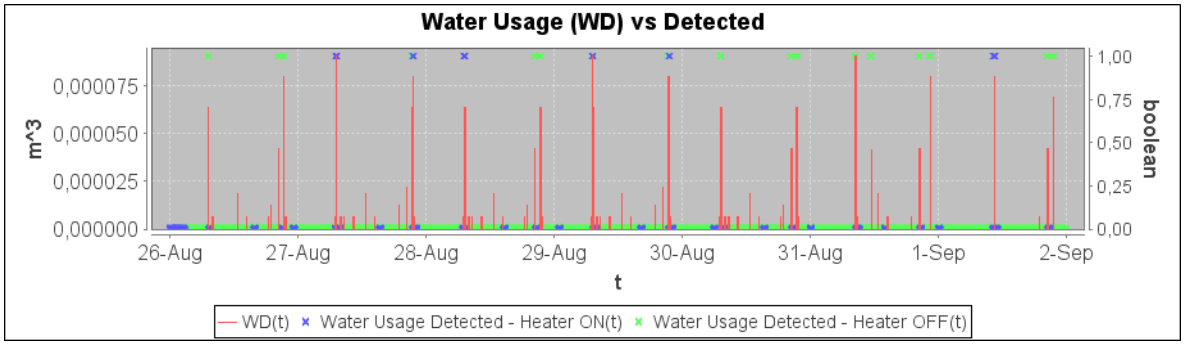


Figura 6.1: Gráfico do perfil de consumo de verão.

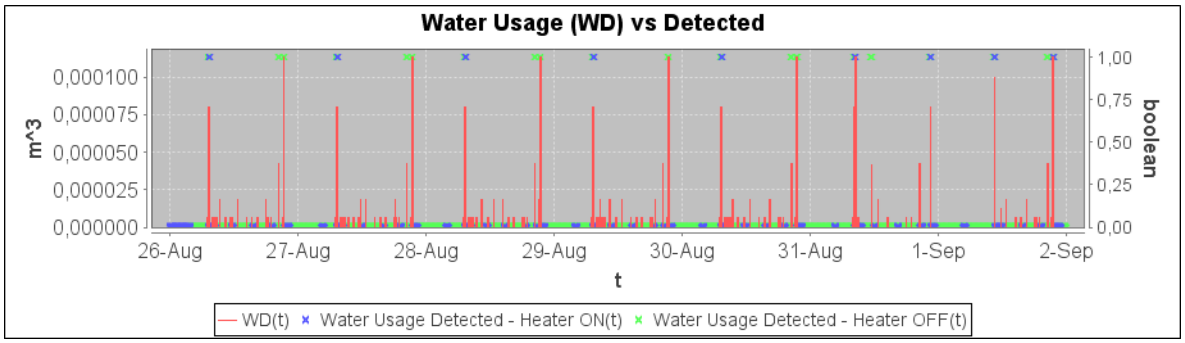


Figura 6.2: Gráfico do perfil de consumo de inverno.

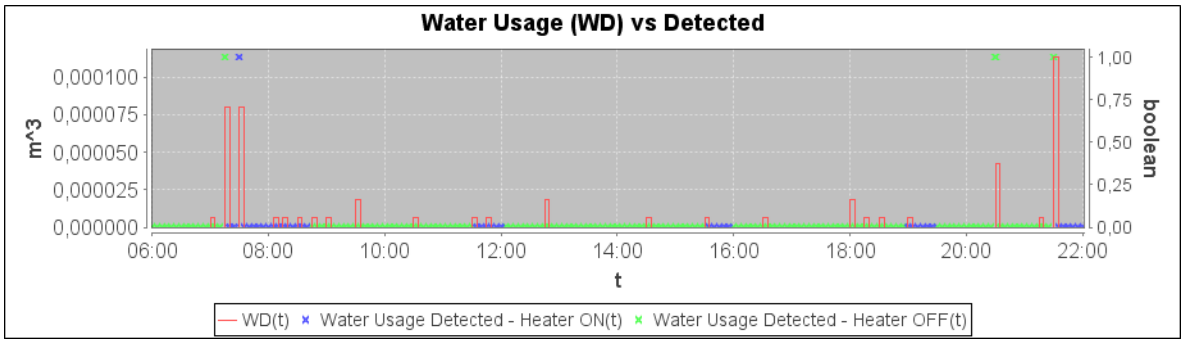


Figura 6.3: Gráfico detalhado do primeiro dia do perfil de consumo de inverno.

Para cada um destes perfis de consumo de água foram usados perfis de temperatura (ambiente e da água fria) adequados. O perfil de temperatura de verão, presente na Figura 6.4, é utilizado com o perfil de consumo de verão, enquanto que o perfil de temperatura de inverno, exposto na Figura 6.5, é utilizado com o perfil de consumo de inverno.

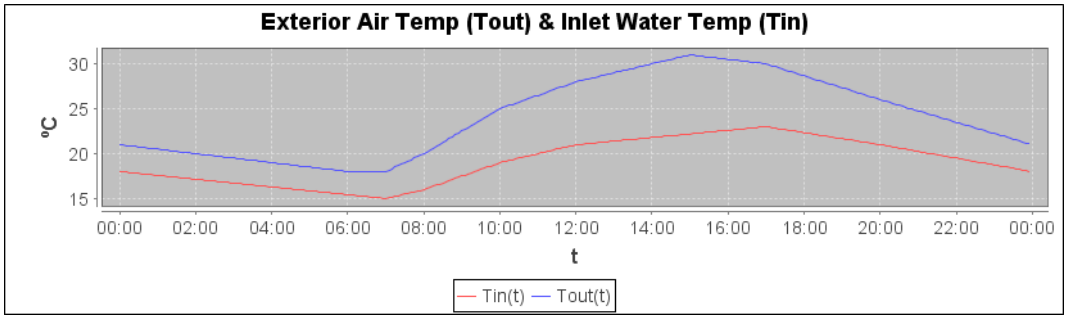


Figura 6.4: Gráfico dos perfis de temperatura de verão.

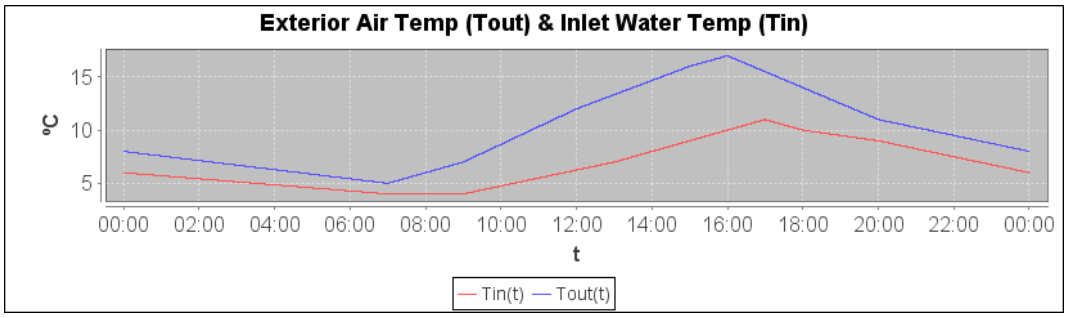


Figura 6.5: Gráfico dos perfis de temperatura de inverno.

Os limiares de temperatura empregues na realização das simulações foram de $50°C$ para o mínimo e $60°C$ para o máximo. Para estabelecer estes valores dos *setpoints* foi tida em conta a necessidade do intervalo, dado por estes limites, ser significativamente grande para permitir evidenciar as potencialidades dos algoritmos. Em simultâneo, o *setpoint* mínimo escolhido respeita o critério de segurança referente à contaminação bacterial e o *setpoint* máximo foi colocado numa temperatura alta o suficiente para oferecer conforto, até em condições de maior consumo, sem que seja perigosamente alta.

Nos testes foi usada uma figura de mérito que pondera igualmente a componente do conforto e da economia de energia, isto é, atribui 50% a cada uma delas. Porém, quando não há consumo de água, a componente de economia de energia passa a ter 100% do peso, visto que nessa situação o conforto é irrelevante. Relativamente aos restantes parâmetros da função de mérito que caracterizam a função de conforto, os seus valores são os enumerados:

- $T_{min} = 45°C$;
- $T_{max} = 60°C$;
- $T_{ideal} = 50°C$;
- $C_{ideal} = 50\%$.

Estabelecidos os parâmetros essenciais, executou-se um conjunto de simulações cujos resultados são apresentados de seguida.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 429.9 | 0.8992 |
| 5 | 429.5 | 0.8980 |
| 10 | 434.7 | 0.8944 |

Tabela 6.2: Resultados da simulação com o algoritmo tradicional num período de 180 dias, com os perfis de verão.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 426.4 | 0.9005 |
| 5 | 435.6 | 0.8964 |
| 10 | 439.8 | 0.8945 |

Tabela 6.3: Resultados da simulação com o algoritmo "inteligente" num período de 180 dias, com os perfis de verão.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 720.9 | 0.8318 |
| 5 | 734.5 | 0.8267 |
| 10 | 740.8 | 0.8224 |

Tabela 6.4: Resultados da simulação com o algoritmo tradicional num período de 180 dias, com os perfis de inverno.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 730.6 | 0.8281 |
| 5 | 752.3 | 0.8147 |
| 10 | 744.8 | 0.8173 |

Tabela 6.5: Resultados da simulação com o algoritmo "inteligente" num período de 180 dias, com os perfis de inverno.

Constatando os resultados destas simulações (Tabelas 6.2, 6.3, 6.4, 6.5), pode concluir-se que a diferença entre os dois algoritmos é quase nula com os perfis de verão, enquanto que com os perfis de inverno, o tradicional revela-se um pouco melhor. Isto deve-se em grande parte ao facto de que quando é prevista uma tiragem de água no futuro, o algoritmo "inteligente" aquece-a até ao seu limar máximo, mesmo que daí resulte uma tiragem que não exija água

tão quente. Assim sendo, como no perfil de consumo de inverno existem tiragens de água mais frequentes, a água é constantemente aquecida até ao máximo tornando este efeito mais evidente.

Contudo, como se pode comprovar pelos resultados seguintes (Tabelas 6.6, 6.7, 6.8, 6.9), reduzindo o *setpoint* máximo para $55°C$, os resultados tornam-se significativamente mais favoráveis em termos gerais. Desta feita, o algoritmo "inteligente" revela-se superior, tanto ao nível de consumo de energia como no mérito obtido.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 392.0 | 0.9065 |
| 5 | 396.9 | 0.9032 |
| 10 | 401.7 | 0.8991 |

Tabela 6.6: Resultados da simulação com o algoritmo tradicional num período de 180 dias, com os perfis de verão e o *setpoint* máximo reduzido para $55°C$.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 386.1 | 0.9085 |
| 5 | 394.8 | 0.9033 |
| 10 | 402.3 | 0.8965 |

Tabela 6.7: Resultados da simulação com o algoritmo "inteligente" num período de 180 dias, com os perfis de verão e o *setpoint* máximo reduzido para $55°C$.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 695.9 | 0.8352 |
| 5 | 710.5 | 0.8256 |
| 10 | 723.3 | 0.8168 |

Tabela 6.8: Resultados da simulação com o algoritmo tradicional num período de 180 dias, com os perfis de inverno e o *setpoint* máximo reduzido para $55°C$.

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 693.5 | 0.8359 |
| 5 | 706.3 | 0.8241 |
| 10 | 712.8 | 0.8146 |

Tabela 6.9: Resultados da simulação com o algoritmo "inteligente" num período de 180 dias, com os perfis de inverno e o *setpoint* máximo reduzido para $55°C$.

Contemplando todos estes resultados pode afirmar-se que com intervalos de amostragem maiores, a tendência mostra haver um consumo energético ligeiramente superior. Isto advém do facto de estes algoritmos só receberem informação de que foi atingido o limar de temperatura máximo, nos instantes de amostragem, o que leva a que a resistência fique ligada para além do tempo necessário. Este problema pode ser minimizado fazendo uma monitorização mais frequente da temperatura da água, ou seja, usando um intervalo de amostragem mais curto.

Comparando as situações de melhor desempenho de ambos os algoritmos, conclui-se que o algoritmo "inteligente", com uma configuração ponderada, tem vantagens em relação ao tradicional. Tratando-se este algoritmo "inteligente" de uma primeira abordagem à exploração destas técnicas, os resultados indiciam que a aposta neste tipo de sistemas poderá revelar benefícios expressivos, principalmente quando exploradas a longo prazo.

É importante realçar que os resultados têm origem em simulações. Numa situação real, o desempenho é mais subjetivo. Também se deve ter em conta que a figura de mérito formulada pode não transparecer uma avaliação precisa dos casos simulados, devido à natureza e subjetividade do compromisso entre conforto e economia de energia. Por este motivo, a indicação do consumo energético tem mais destaque na comparação dos resultados.

Para comprovar se o algoritmo "inteligente" consegue manter os mesmos resultados quando o perfil de consumo sofre pequenas variações de semana para semana, foram feitas simulações com a função de aleatoriedade ativa. Recorreu-se a uma distribuição Gaussiana com desvio padrão de 15 minutos e com limites relativos de *cutoff* de $\pm 30$ minutos. Isto significa que o instante temporal da ocorrência de cada tiragem de água presente nos perfis tem cerca de 68% de probabilidade de permanecer contida num intervalo de 30 minutos centrado no instante original. Os limites de *cutoff* impedem que os valores gerados se estendam para além destes limites.

De notar que os resultados da Tabela 6.10, referentes a esta situação, decorrem da média de 5 execuções consecutivas. Estes demonstram que mesmo com alguma incerteza no perfil de consumo, o algoritmo "inteligente" tem competência para manter um desempenho regular, visto que até foi capaz de superar o melhor desempenho conseguido anteriormente (com o perfil fixo).

| **Intervalo de amostragem** (*minutos*) | **Tempo total de consumo energético** (*horas*) | **Mérito** |
|---|---|---|
| 2 | 383.3 | 0.9094 |
| 5 | 392.8 | 0.9039 |
| 10 | 402.5 | 0.8975 |

Tabela 6.10: Resultados da simulação com o algoritmo "inteligente" num período de 180 dias, com os perfis de verão, função de aleatoriedade ativa e um *setpoint* máximo de $55°C$.

Na Figura 6.6 podemos observar o comportamento do algoritmo numa das simulações realizadas, perante a antecipação ou procrastinação das tiragens de água.

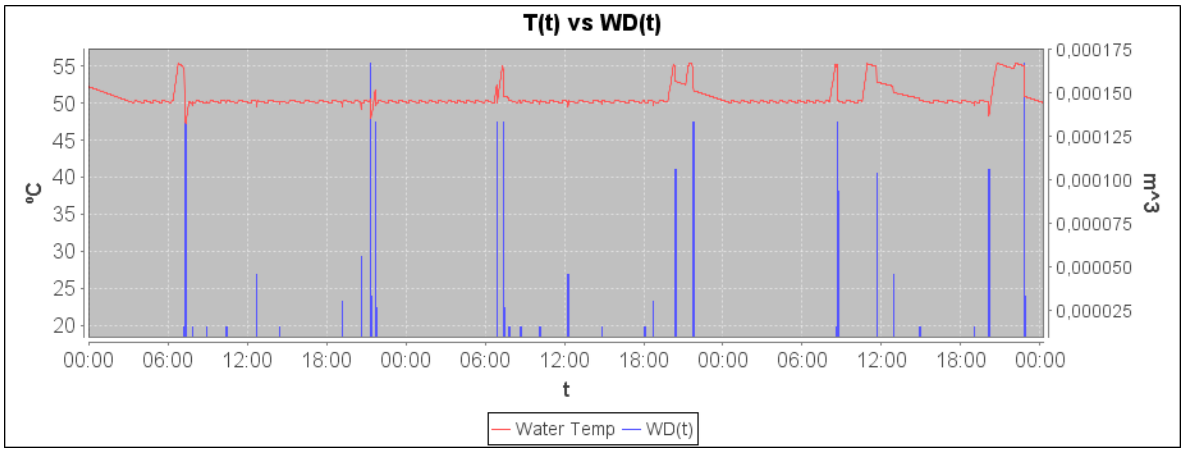


Figura 6.6: Gráfico da evolução da temperatura da água face a tiragens da mesma. Excerto de três dias de simulação do algoritmo "inteligente" com os perfis de verão e aleatoriedade ativa.

## 6.3 Objetivos Cumpridos e Desenvolvimentos Futuros

Analisando os objetivos inicialmente propostos face ao trabalho desenvolvido, pode-se afirmar que, de um ponto de vista geral, estes foram conseguidos.

Foi concebida toda uma plataforma de desenvolvimento e avaliação de algoritmos para tanques de aquecimento elétrico de água, através da construção de um simulador (recorrendo ao modelo apresentado no Capítulo 2) capaz de reproduzir a evolução da temperatura da água num desses tanques, tendo em consideração vários parâmetros e situações de consumo personalizáveis. Os resultados da simulação podem ser visualizados graficamente na própria ferramenta, ou alternativamente podem ser guardados para futura referência.

Também incluída na ferramenta de simulação está a figura de mérito projetada. Devido à natureza contraditória das componentes (conforto e poupança de energia), a subjetividade presente na função de mérito é necessariamente um aspeto sensível, e por essa razão nem sempre se pode recorrer ao resultado da figura de mérito para atribuir melhor desempenho a um algoritmo relativamente a outro. Mesmo não sendo esta figura de mérito ideal como meio exclusivo de avaliação e comparação entre algoritmos, permite todavia ter uma noção imediata sobre o desempenho destes.

Para além disto, o simulador facilita a adição de novos algoritmos à ferramenta, através do provisionamento de uma interface extensível (por meio de uma classe abstrata no *Java*).

Relativamente ao objetivo primário desta dissertação, foi criado um algoritmo adaptativo de deteção de consumo de água quente, a partir das variações de temperatura da água do tanque ao longo do tempo. Daí seguiu-se a implementação de um algoritmo que simula o comportamento dos sistemas tradicionais e de um algoritmo "inteligente" que antecipa o aquecimento da água baseando-se no histórico de consumo detetado.

Idealmente, deveria ter sido criado mais um algoritmo baseado numa estratégia diferente, ou alternativamente, deveriam ter sido exploradas e implementadas estratégias de *Demand-Side Management* (com perfis diários fixos que estimem o consumo energético habitual da

população) para ter assim à disposição mais opções para o confronto de resultados.

A adaptação dos algoritmos para a plataforma *Arduino*, tendo em conta as suas limitações e recursos, também foi um objetivo conseguido sem alterações de maior na lógica de decisão dos algoritmos.

Por fim, foram feitas comparações entre os algoritmos disponíveis, perante várias situações de teste, e tiradas as conclusões presentes neste capítulo, rematando desta forma os objetivos pré-estabelecidos.

As propostas para futuros desenvolvimentos na sequência deste trabalho cingem-se essencialmente à criação de novos algoritmos ou ao melhoramento dos algoritmos apresentados, tirando proveito da ferramenta de simulação concebida. Como ficou provado que o algoritmo "inteligente" implementado ainda deixava a plataforma de desenvolvimento com bastantes recursos de memória disponíveis, seria interessante dotar o algoritmo com um modelo de decisão probabilístico, em que cada instante de amostragem registado na memória se faz acompanhar com um valor que indica a probabilidade da ocorrência de consumo de água.

Continuando este raciocínio, seria relevante uma resposta mais complexa por parte do algoritmo para quando este é reiniciado, por exemplo, numa situação de falha de energia. No caso de não existir um relógio no sistema (sincronizado pelo utilizador para manter o tempo real), e para evitar desperdiçar o histórico armazenado previamente, poderiam ser exploradas estratégias de sincronização com base na correspondência entre perfil de consumo observado durante os primeiros dias de execução e a informação de consumo contida no histórico.

Tal como mencionado na introdução, no contexto da domótica pode ainda ser explorado o facto de se ter acesso a muito mais informação sobre o mundo exterior (e.g., tarifa energética, calendário, previsão meteorológica, consumo energético da casa, intenção do utilizador), que pode ser ponderada na decisão do algoritmo.

Ao longo deste documento apresentou-se o modelo matemático de suporte ao simulador, o método idealizado para a avaliação dos algoritmos e a implementação detalhada dos algoritmos e ferramentas. Foram também expostas considerações, resultados, análises, comparações e conclusões sobre o trabalho proposto.

# Apêndice A

# "Tapping Cycle M" (Adaptação)

Tabela que dita os ciclos de tiragem de água ao longo de um dia (utilizada como base para criação dos perfis de consumo de água). Versão adaptada da norma europeia "EN 50440" (*Efficiency of Domestic Electric Storage Water Heaters*), que define requisitos e testes adequados para classificação de unidades *Domestic Electric Water Heater* (DEWH).

| # | Hora | Potência | Perfil | deltaT | m3 | Litros |
|---|---|---|---|---|---|---|
| 1 | 7:00 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 2 | 7:15 | 1.400 | Shower | 30 | 0.024 | 24 |
| 3 | 7:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 4 | 8:01 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 5 | 8:15 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 6 | 8:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 7 | 8:45 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 8 | 9:00 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 9 | 9:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 10 | 10:30 | 0.105 | Floor cleaning | 30 | 0.0018 | 1.8 |
| 11 | 11:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 12 | 11:45 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 13 | 12:45 | 0.315 | Dish washing | 45 | 0.0054 | 5.4 |
| 14 | 14:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 15 | 15:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 16 | 16:30 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 17 | 18:00 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 18 | 18:15 | 0.105 | Household cleaning | 30 | 0.0018 | 1.8 |
| 19 | 18:30 | 0.105 | Household cleaning | 30 | 0.0018 | 1.8 |
| 20 | 19:00 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 21 | 20:30 | 0.735 | Dish washing | 45 | 0.0126 | 12.6 |
| 22 | 21:15 | 0.105 | Small | 15 | 0.0018 | 1.8 |
| 23 | 21:30 | 1.400 | Shower | 30 | 0.024 | 24 |
| | | 5.845 | | 0.1002 | m3 | 100.2 |

# Bibliografia


[1] U.S. Department of Energy. *Tankless or Demand-Type Water Heaters*. 2012. URL: http://energy.gov/energysaver/articles/tankless-or-demand-type-water-heaters (acedido em 07/2013).

[2] U.S. Department of Energy. *Selecting a New Water Heater*. 2012. URL: http://energy.gov/energysaver/articles/selecting-new-water-heater (acedido em 07/2013).

[3] Marcel Lacroix. «Electric water heater designs for load shifting and control of bacterial contamination». Em: *Energy Conversion and Management* 40.12 (1999), pp. 1313 – 1340. ISSN: 0196-8904. DOI: http://dx.doi.org/10.1016/S0196-8904(99)00013-8. URL: http://www.sciencedirect.com/science/article/pii/S0196890499000138.

[4] M.H Nehrir e B.J LaMeres. «A multiple-block fuzzy logic-based electric water heater demand-side management strategy for leveling distribution feeder demand profile». Em: *Electric Power Systems Research* 56.3 (2000), pp. 225 –230. ISSN: 0378-7796. DOI: http://dx.doi.org/10.1016/S0378-7796(00)00124-3. URL: http://www.sciencedirect.com/science/article/pii/S0378779600001243.

[5] Alain Moreau. «Control Strategy for Domestic Water Heaters during Peak Periods and its Impact on the Demand for Electricity». Em: *Energy Procedia* 12.0 (2011), pp. 1074 –1082. ISSN: 1876-6102. DOI: http://dx.doi.org/10.1016/j.egypro.2011.10.140. URL: http://www.sciencedirect.com/science/article/pii/S1876610211019667.

[6] B.J LaMeres, M.H Nehrir e V Gerez. «Controlling the average residential electric water heater power demand using fuzzy logic». Em: *Electric Power Systems Research* 52.3 (1999), pp. 267 –271. ISSN: 0378-7796. DOI: http://dx.doi.org/10.1016/S0378-7796(99)00022-X. URL: http://www.sciencedirect.com/science/article/pii/S037877969900022X.

[7] Dario Bonino, Fulvio Corno e Faisal Razzak. «Enabling machine understandable exchange of energy consumption information in intelligent domotic environments». Em: *Energy and Buildings* 43.6 (2011), pp. 1392 –1402. ISSN: 0378-7788. DOI: http://dx.doi.org/10.1016/j.enbuild.2011.01.013. URL: http://www.sciencedirect.com/science/article/pii/S0378778811000223.

[8] Hussein Joumaa et al. «A {MAS} integrated into Home Automation system, for the resolution of power management problem in smart homes». Em: *Energy Procedia* 6.0 (2011). Impact of Integrated Clean Energy on the Future of the Mediterranean Environment, pp. 786 –794. ISSN: 1876-6102. DOI: http://dx.doi.org/10.1016/j.egypro.2011.05.089. URL: http://www.sciencedirect.com/science/article/pii/S1876610211015001.

[9] M. Shaad et al. ≪Parameter identification of thermal models for domestic electric water heaters in a direct load control program≫. Em: *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*. 2012, pp. 1–5. DOI: 10.1109/CCECE.2012.6334885.

[10] Liam Paull, Howard Li e Liuchen Chang. ≪A novel domestic electric water heater model for a multi-objective demand side management program≫. Em: *Electric Power Systems Research* 80.12 (2010), pp. 1446 –1451. ISSN: 0378-7796. DOI: 10.1016/j.epsr.2010.06.013. URL: http://www.sciencedirect.com/science/article/pii/S0378779610001434.

[11] P.S. Dolan, M.H. Nehrir e V. Gerez. ≪Development of a Monte Carlo based aggregate model for residential electric water heater loads≫. Em: *Electric Power Systems Research* 36.1 (1996), pp. 29 –35. ISSN: 0378-7796. DOI: http://dx.doi.org/10.1016/0378-7796(95)01011-4. URL: http://www.sciencedirect.com/science/article/pii/0378779695010114.

[12] Oracle. *Java™Platform, Standard Edition 7. API Specification*. Versão 7. URL: http://docs.oracle.com/javase/7/docs/api/overview-summary.html (acedido em 05/2013).

[13] Object Refinery Limited. *JFreeChart Javadoc. API Documentation*. Versão 1.0.14. URL: http://www.jfree.org/jfreechart/api/javadoc/ (acedido em 05/2013).

[14] Trent Jarvi. *RXTX Javadoc. API Documentation*. Versão 2.1. URL: http://users.frii.com/jarvi/rxtx/doc/index.html (acedido em 05/2013).

[15] Glen Smith, Sean Sullivan e Scott Conway. *Opencsv Javadoc. API Documentation*. Versão 2.4. URL: http://opencsv.sourceforge.net/apidocs/ (acedido em 05/2013).

[16] David Gilbert. *The JFreeChart Class Library. Developer Guide*. Versão 1.0.9. 2008.

[17] Bruce Eckel. *Thinking in Java*. 4. Upper Saddle River, NJ: Prentice Hall, 2006, pp. 23–38, 72, 311–320, 1303–1306. ISBN: 978-0-13-187248-6.

[18] Atmel Corporation. *ATmega640/1280/1281/2560/2561 Complete Datasheet*. Versão revision P. 2012, pp. 1, 12–13, 21–36.