Universidade do Minho
Escola de Engenharia

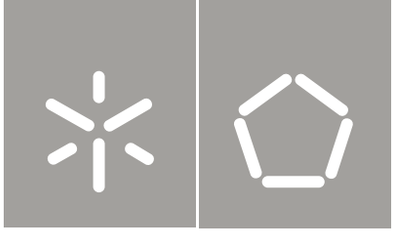
José Manuel da Rocha Ferreira

Middleware para Ambient Assisted Living: casos de estudo

dezembro de 2013

José Manuel da Rocha Ferreira Middleware para Ambient Assisted Living: casos de estudo

UMinho | 2013



Universidade do Minho
Escola de Engenharia

José Manuel da Rocha Ferreira

Middleware para Ambient Assisted Living: casos de estudo

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Doutor Paulo Francisco Silva Cardoso

dezembro de 2013

DECLARAÇÃO

Nome: José Manuel da Rocha Ferreira

Endereço eletrónico: a54016@alunos.uminho.pt **Telefone:** 916842294

Número do Bilhete de Identidade: 13596745

Título da tese: *Middleware* para *Ambient Assisted Living*: casos de estudo

Orientador: Doutor Paulo Francisco Silva Cardoso

Ano de conclusão: 2013

Designação do Mestrado:

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Universidade do Minho, 26/12/2013

Assinatura: _____

Agradecimentos

Este trabalho é o culminar de cinco anos de árduo esforço e dedicação. Cinco anos que foram compartilhados com diversas pessoas, às quais dirijo o meu sincero agradecimento.

Ao Professor Paulo Cardoso e ao Professor Adriano Tavares por todas as recomendações, disponibilidade e empenho que demonstraram na orientação deste trabalho, e pelo incentivo no sentido de cumprir com as metas definidas para a dissertação.

Aos meus colegas de Mestrado pelo companheirismo, entreajuda, trabalho em equipa e amizade que fomos desenvolvendo ao longo destes anos.

A todos os docentes com quem me fui cruzando e que contribuíram para a minha formação e aos técnicos do Departamento de Eletrónica Industrial pelo conhecimento partilhado.

Por fim, um agradecimento especial à minha família por me ter dado a oportunidade de frequentar este Mestrado, por todo o apoio, dedicação e motivação que me deram, e por toda a compreensão e amparo que sempre estiveram presentes.

Título da dissertação

Middleware para *Ambient Assisted Living*: casos de estudo

Resumo

O tema de *Ambient Assited Living* surge como uma possível solução para a crescente necessidade de proporcionar uma melhor qualidade de vida à faixa etária mais idosa da nossa sociedade. Esta solução procura integrar os conceitos de *Ambient Intelligence* para criar sistemas que permitam auxiliar os idosos a executar as suas tarefas diárias. Além desta vertente de auxílio, também se procura uma monitorização constante dos parâmetros de saúde, possibilitando o acompanhamento médico à distância destas pessoas resultando num maior conforto para estas. Para que estes objetivos sejam cumpridos é necessário que exista uma rede de sensores que permita a aquisição dos diversos dados associados à habitação e aos seus habitantes. Contudo, as diversas tecnologias utilizadas por estes dispositivos podem levar a uma maior complexidade na aquisição dos dados, criando problemas problemas de interoperabilidade. Para resolver estes problemas, é apresentada uma camada de abstração denominada *Middleware* que permite lidar com as diversas tecnologias e protocolos associados aos dispositivos sensores.

Palavras-chave

Middleware, *Ambient Assisted Living*, *Ambient Intelligence*, Ontologia, Interoperabilidade, Camada de abstração

Dissertation Title

Middleware for Ambient Assisted Living: study cases

Abstract

The Ambient Assisted Living theme is presented as a possible solution to the demand of providing a better quality of life to elderly people of our society. This solution integrates the concepts of Ambient Intelligence to build systems that can assist the elder in their daily life activities. Besides this objective of assistance, these systems should also provide means to constantly monitor the health parameters, enabling remote medical care. To achieve these objectives, there must be present a sensor network that can cope with the needed data acquisition regarding the housing and its inhabitants. However, the different technologies used by these sensor networks may pose a problem of interoperability and greater complexity in data acquisition. To solve this problem is presented an abstraction layer named Middleware that allows dealing with the various technologies and protocols associated with the devices used in sensor networks.

Keywords

Middleware, Ambient Assisted Living, Ambient Intelligence, Ontologies, Interoperability, Abstraction Layer

Índice

Índice de figuras

Lista de Acrónimos

AAL	Ambient Assisted Living
AmI	Ambient Intelligence
API	Application Programming Interface
BAN	Body Area Network
CAN	Controlled Area Network
COTS	Commercial Off-The-Shelf
ECG	Eletrocardiograma
EDGE	Enhanced Data rates for GSM Evolution
EHS	European Home Systems
EIB	European Installation Bus
GPRS	General packet radio service
KIF	Knowledge Interchange Format
LAN	Local Area Network
MAN	Metropolitan Area Network
OKBC	Open Knowledge Base Connectivity
OSGi	Open Services Gateway Initiative
OWL	Web Ontology Language
PAN	Personal Area Network
RDF	Resource Description Framework
RF	Radiofrequência
RFID	Radio Frequency Identification
SOA	Service Oriented Architecture
UMTS	Universal Mobile Telecommunication System
UPnP	Universal Plug and Play
USB	Universal Serial Bus
UWB	Ultra-wide-band
W3C	World Wide Web Consortium
WAN	Wide Area Network
WIMAX	Worldwide Interoperability for Microwave Access
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

Capítulo 1 – Introdução

Este capítulo pretende fazer uma breve apresentação e enquadramento do tema desta dissertação. Serão também apresentadas as motivações e os objetivos a atingir.

1.1 Enquadramento

A introdução de dispositivos de domótica nas habitações surgiu como resposta à necessidade de aumentar o conforto e segurança das pessoas nas suas casas. Inicialmente, as funcionalidades elementares que estes dispositivos concediam eram suficientes para auxiliar em algumas tarefas mais básicas como o controlo de temperatura e iluminação numa habitação. Contudo, a massificação dos dispositivos levou a que fossem procuradas novas soluções para aumentar ainda mais o conforto e segurança das pessoas nas suas habitações. Assim, começaram a surgir os dispositivos com capacidades de comunicação e processamento que cooperavam entre si para automatizar algumas tarefas nos ambientes habitacionais onde estavam integrados.

Foi a partir destes dispositivos que se começaram a idealizar as soluções de casas inteligentes (*Smart Homes*), onde se pretendia que tarefas do dia-a-dia fossem executadas sem a intervenção das pessoas. Para tal, foram integrados nestes sistemas alguns conceitos associados à área de inteligência artificial e *Ambient Intelligence* de modo a que fosse possível contextualizar os dados obtidos dos diversos dispositivos para que a atuação se tornasse mais autónoma. A concretização destes sistemas proporcionou as condições necessárias para o desenvolvimento de soluções que centram o seu foco de estudo numa faixa etária que está em crescimento e necessita de ter um auxílio e supervisão mais eficaz. Surgiu assim a área de *Ambient Assisted Living* que pretende dar resposta às necessidades das pessoas idosas e de pessoas com doenças crónicas. Esta área de investigação e desenvolvimento pretende auxiliar estas pessoas nas suas tarefas diárias, utilizando proactivamente os diversos dispositivos que podem ser integrados numa habitação.

Além da automatização de processos, área de investigação em *Ambient Assisted Living* procura também a monitorização constante das pessoas de modo a prevenir e/ou alertar para situações anómalas ou de perigo. Assim, os sistemas de *AAL* devem dispor de alguma inteligência para que consigam distinguir as situações referidas. Para tal, os conhecimentos adquiridos pela área de investigação em Inteligência Artificial e *Ambient Intelligence* são utilizados em conjunto para conferir a "inteligência" necessária ao sistema, sendo que uma das formas para alcançar este objetivo é através da modelação e implementação ontológica dos sistemas representados.

O conceito da modelação e implementação ontológicas tem vindo a ser explorado nas mais diversas áreas, com o intuito de proporcionar sistemas dotados de uma maior autonomia de decisão. Para tal, é crucial a interpretação dos dados adquiridos e a sua contextualização no modelo que descreve o ambiente envolvente. Neste caso específico, os dados adquiridos pelos dispositivos devem ser interpretados à luz de um modelo que descreve os diversos elementos dos sistemas de *AAL* como o ambiente habitacional, os utilizadores, os serviços prestados, entre outros. Também se utilizam estes dados para uma interpretação do estado clínico dos habitantes monitorizados, sendo esta também uma das finalidades deste tipo de soluções.

Idealmente, a comunicação com os dispositivos deveria seguir uma norma *standard* que permitisse efetuar a troca de dados de forma homogénea. Contudo devido à diversidade de fabricantes e da liberdade de escolha protocolar, esta situação não acontece levantando problemas de interoperabilidade. Por este motivo, existe a necessidade de criar uma camada de abstração que permita lidar com os diversos dispositivos, protocolos de comunicação e tecnologias, de modo a tornar a interação menos complexa. Esta camada de abstração é denominada *middleware* e tem por objetivo fazer o interface entre as aplicações que necessitam dos dados dos diversos dispositivos e o *hardware* existente nas habitações que providencia esses dados.

Para atingir os objetivos pretendidos pelo *middleware*, são disponibilizadas aos programadores das aplicações *APIs* de alto nível que ocultam a complexidade de comunicação com os dispositivos geradores de dados. Além desta funcionalidade, o *middleware* também pode ser responsável pela uniformização na formatação e apresentação dos dados, caso esta

característica se revele importante para a compreensão dos dados quer por parte dos utilizadores, quer por parte do sistema.

1.2 Motivação

A procura de soluções para o aumento do conforto e segurança tem vindo a ser objeto de grande atenção por parte da sociedade. Esta procura torna-se ainda mais significativa quando se pretende aplicar estas soluções em contextos onde as pessoas necessitam de um maior acompanhamento e de cuidados especiais. É sobre estas soluções que se insere a área de investigação de *Ambient Assisted Living (AAL)*, sendo que esta área tem vindo a receber cada vez mais atenção no que respeita a projetos de investigação e desenvolvimento. O investimento nestas soluções por parte de equipas de investigação e desenvolvimento de diversos países tem sido muito significativo, destacando-se assim a sua importância.

1.3 Objetivos

Esta dissertação está inserida no âmbito dos sistemas de *Ambient Assisted Living* e tem por objetivo expor um estudo sobre este tema e as abordagens de *middleware* que podem ser utilizadas neste âmbito. Pretende-se ainda abordar outros conceitos associados ao tema de *Ambient Assisted Living* como os conceitos de ontologia e também redes de sensores.

1.4 Organização e estrutura da dissertação

De modo a promover uma melhor compreensão do tema em estudo e da metodologia adotada, este documento foi organizado em seis capítulos principais. O conteúdo desses capítulos é apresentado de seguida.

No capítulo 1 (Introdução) é apresentado o tema da dissertação e é feito um breve enquadramento do mesmo. São também apresentadas as motivações e os objetivos a atingir.

No capítulo 2 (Domótica, Casas Inteligentes e AAL) pretende-se fazer uma introdução aos sistemas de domótica, casas inteligentes e *Ambient Assisted Living*, sendo que se procura apresentar também as relações existentes entre eles.

O capítulo 3 (AAL e as ontologias) pretende demonstrar a relação existente entre os conceitos de *Ambient Intelligence*, ontologias e os sistemas de *AAL*. Ao longo do capítulo o tema de *AAL* será abordado e será apresentado um exemplo de modelação destes sistemas.

Ao longo do capítulo 4 (AAL e o Middleware) pretende-se demonstrar a relação entre o tema de *AAL* e o *middleware*. Serão apresentados diferentes tipos de *middleware* bem como a sua implementação.

Por fim, no capítulo 5 (Conclusões) serão apresentadas as conclusões retiradas do trabalho efetuado e apresentadas algumas ideias para trabalho a desenvolver futuramente.

Capítulo 2 – Domótica, Casas Inteligentes e AAL

Neste capítulo pretende-se fazer o enquadramento dos conceitos de domótica e casas inteligentes associados ao tema de *Ambient Assited Living*.

2.1 Introdução

O conceito de domótica surgiu como resposta à necessidade de introduzir no ambiente habitacional, recursos que permitissem facilitar a execução de algumas tarefas diárias. Para tal, podem ser incorporadas nas residências redes de sensores e atuadores que possibilitam a automatização de algumas tarefas na habitação. Esta automatização é feita através da verificação das alterações no estado dos diversos sensores que despertam uma reação. Um exemplo desta automatização pode ser a ligação automática da iluminação quando uma pessoa entra numa divisão.

Inicialmente, as ações de controlo dos sistemas de iluminação, temperatura, energia e segurança eram as mais comuns nos sistemas de domótica. No entanto, com a expansão e integração de sensores e atuadores nas habitações, começaram a procurar-se soluções que fossem além da automatização de processos. Para alcançar estas soluções, foram introduzidas nas habitações as funcionalidades disponibilizadas pela inteligência artificial para criar sistemas capazes de compreender o ambiente envolvente, interpretar as suas alterações e reagir autonomamente de acordo com o mesmo. Além destas funcionalidades, estes sistemas devem ser capazes de fazer a monitorização dos intervenientes na habitação e reagir de acordo com as suas ações. Surge assim o conceito de casas inteligentes (*Smart Homes* [1], [2]).

É através da conjugação destes conceitos e funcionalidades disponibilizadas que se torna possível alcançar os objetivos propostos pelo tema de *Ambient Assisted Living*, cujo foco se centra na procura de soluções que respondam às necessidades da faixa etária mais idosa da população.

2.2 As redes de sensores e atuadores

A aquisição de dados tem um papel fundamental para o correto funcionamento dos sistemas de domótica, casas inteligentes e de *AAL*. São esses dados que podem fornecer as informações necessárias aos sistemas computacionais de modo a que possam ser tomadas decisões e executadas ações autonomamente.

A evolução dos dispositivos de aquisição de dados veio preencher um conjunto de condições necessárias para uma integração cada vez maior desses dispositivos no quotidiano das pessoas. Essa evolução pautou-se pela diminuição do tamanho e consumo energético destes dispositivos e também pelo aumento da sua capacidade de processamento. Estas características permitiram incorporar num só dispositivo diversos sensores para fazer leituras de diferentes grandezas físicas. Além da possibilidade de medir as diversas grandezas físicas, estes dispositivos foram dotados de tecnologias para armazenamento dos dados recolhidos e também para comunicação em rede. A integração destas tecnologias nos dispositivos veio possibilitar a dispersão destes numa área mantendo a comunicação entre eles, dando assim origem ao conceito de redes de sensores [3].

A investigação sobre as redes de sensores iniciou-se no âmbito militar, durante a Guerra Fria, em que foram implantados sensores acústicos no fundo do oceano. Este sistema (*SOSUS – Sound Surveillance System*) desenvolvido pela marinha dos Estado Unidos da América tinha como objetivo detetar e rastrear submarinos soviéticos. Com o passar dos anos, o sistema foi substituído por outros sistemas mais atuais e eficazes e, hoje em dia, é utilizado pela Administração Oceânica e Atmosférica Nacional (*NOAA*) para monitorizar eventos no oceano como atividade sísmica ou animal [4].

Além desta iniciativa, foram criados programas para investigação sobre redes de sensores distribuídas que tiveram grande impacto nas redes de sensores atuais. Nesses projetos foram propostas as ligações das redes de sensores à internet e a incorporação de inteligência artificial nos dispositivos de modo a compreender os sinais recebidos e avaliar as situações. Atualmente, as redes de sensores já possuem algumas das características idealizadas aquando do início da sua investigação. No entanto, a investigação nesta área não

estagnou sendo que agora se procuram desenvolver novas técnicas para ligação *ad hoc* destes dispositivos às redes e também formas de adquirir a informação proveniente das redes de forma fiável e num período de tempo útil.

As redes de sensores baseiam-se em dispositivos que dependem de três grandes áreas de investigação: área de sensores, área de comunicação e a área de computação. Todas elas contribuem para a criação de dispositivos que podem ser introduzidos nos mais diversos cenários para controlar sistemas como habitações, sistemas de gestão de tráfego ou até cidades completas. As áreas de segurança física, militar, controlo aéreo, automação fabril e industrial, monitorização de edifícios, de estruturas e de ambientes são alguns dos exemplos de potenciais áreas onde estas redes de sensores podem ser aplicadas.

2.2.1 Aquisição de dados

Uma vez que as redes de sensores e atuadores afetos aos sistemas de domótica, casas inteligentes e *AAL*, necessitam de algum meio de comunicação entre os diversos intervenientes no sistema, foram introduzidos nos dispositivos elementos capazes de permitir a aquisição de dados e a sua comunicação [5]. A introdução desta capacidade de comunicação levou a que fosse também preciso estabelecer o alcance destas redes de comunicação, sendo para isso usados os conceitos de *WAN*, *MAN*, *LAN*, *PAN* e *BAN*. Na Figura 1 está representado o alcance destas redes, tendo como exemplo as tecnologias sem fios. As redes *WAN* são as de maior alcance e as redes *BAN* de menor alcance.

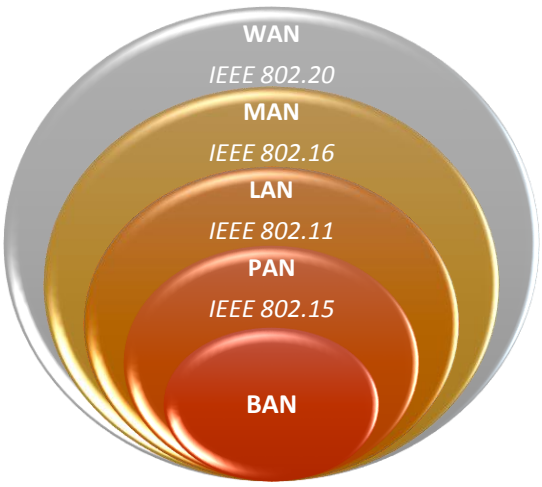


Figura 1 – Tipos de rede sem fios [5]

As redes *WAN* (*Wide Area Network*) e *MAN* (*Metropolitan Area Network*) são utilizadas para transmissão de dados em ambientes exteriores. No caso das redes *WAN* são utilizadas tecnologias como *UMTS*, *GPRS* e *EDGE* para fazer a comunicação de dados. Já nas redes *MAN* utiliza-se o *WIMAX* para atingir este mesmo objetivo.

Quanto às redes *LAN* (*Local Area Network*), *PAN* (*Personal Area Network*) e *BAN* (*Body Area Network*), estas são utilizadas para transmissão de dados em ambientes interiores. No caso das redes *LAN*, as ligações *Wifi* e *Ethernet* são as tecnologias mais adotadas. As redes *PAN* têm uma maior variedade de tecnologias associadas como a tecnologia *Bluetooth*, *RFID*, *Zigbee*, *UWB*, *CEBus*, *HAVi*, *HomePNA*, *HomePlug*, *HomeRF*, *UPnP*, *USB*. As redes *BAN* estão ainda numa fase inicial de investigação pelo que apresentam poucas soluções, podendo ainda assim referir-se a tecnologia *BodyLAN* que pretende transmitir dados através da pele do utilizador.

Além das tecnologias mencionadas, pode ainda referir-se outras tecnologias normalizadas associadas aos sistemas de casas inteligentes como *KNX* (normalização dos protocolos *BatiBus*, *EIB* e *EHS*), *X10*, *CEBus* e *HBS*. Estes *standards* estão associados às redes *PAN* e utilizam ligações através de meios físicos para comunicação de dados (barramentos de dados).

Num contexto de *AAL*, tomando como exemplo de aplicação o sistema de monitorização de saúde, torna-se então possível adquirir dados sobre parâmetros fisiológicos dos habitantes (pressão arterial, frequência cardíaca, peso, etc.) e armazenar esses dados num computador afeto à casa ou enviar os dados a entidades que possam analisá-los [6]. Também pode ser interessante neste cenário obter os dados sobre a mobilidade apresentada pelos habitantes. Para tal são utilizados sensores capazes de detetar a postura do corpo, vibrações e quedas dos utilizadores.

No exemplo citado em [6], existe uma divisão dos sensores em três tipos: genéricos, integridade vegetativa e específicos. No caso dos sensores genéricos, é utilizado um acelerómetro que permite detetar movimentações e quedas dos pacientes. São também utilizados sensores magnéticos nas portas que permitem monitorizar a localização do paciente e alertar para possíveis situações anormais. Estas situações podem ser detetadas pela frequência ou

pelo tempo que o paciente despende numa divisão, como por exemplo a casa de banho. No caso dos sensores de integridade vegetativa, estes estão destinados a monitorizar constantemente os sinais vitais dos pacientes em causa. Assim, procura-se monitorizar o ritmo respiratório, a frequência cardíaca e a pulsação, parâmetros estes de elevada importância nestes pacientes. Por fim, os sensores específicos estão associados a problemas de saúde que exijam uma monitorização específica de um determinado parâmetro. Como exemplo, é referido o edema pulmonar que exige uma monitorização da permeabilidade da membrana alvéolo-capilar. Os sensores referidos neste exemplo partilham da mesma necessidade de comunicação com um sistema central para armazenamento e análise dos dados.

Através das diversas tecnologias mencionadas é possível fazer a comunicação de dados entre dispositivos e fornecer aos sistemas de domótica, casas inteligentes e *AAL* os dados imprescindíveis ao seu funcionamento. Contudo é necessário ter em conta alguns dos desafios que são inerentes às redes de sensores.

2.2.2 Desafios

Alguns dos desafios associados às redes de sensores estão relacionados com o processamento dos dados, a comunicação e a gestão dos sensores. Estes problemas estão associados às características dos ambientes onde estes dispositivos são introduzidos, sendo que as restrições provocadas pela utilização de bateria e a largura de banda da rede podem influenciar bastante estes dispositivos. Assim, é necessário resolver problemas como a descoberta da rede, controlo da rede, processamento colaborativo de dados na rede, atribuição de tarefas e consultas na rede, e a segurança.

i. Descoberta de rede

A descoberta e conhecimento da rede de sensores que envolve um dispositivo são obviamente dois fatores muito importantes nestes sistemas. Cada dispositivo é considerado um nodo da rede e deve ter um conhecimento dos dispositivos que o rodeiam para que possa cooperar com estes. Em alguns sistemas onde a rede é projetada *a priori*, torna-se mais fácil saber a topologia da rede e a distribuição dos dispositivos sensoriais. Contudo, quando se utilizam redes *ad hoc* a topologia da rede é alterada a cada introdução de um novo dispositivo e essa atualização na rede deve ser apresentada aos dispositivos da vizinhança na rede. Além do reconhecimento dos dispositivos que o rodeiam, cada nodo da rede deve ter a informação da sua localização no sistema.

ii. Controlo de rede

O controlo de rede tem como principal preocupação a fiabilidade da mesma, uma vez que os diversos recursos (energia, largura de banda e processamento) que a compõem podem alterar-se de forma dinâmica. Por este motivo, o sistema deve ser projetado de modo a ser capaz de operar autonomamente no sentido de alterar as configurações para responder convenientemente a esta alteração dos recursos. Para tal, é necessário garantir que são implantados os nodos suficientes para promover alguma redundância na dispersão da informação, assegurando assim a sobrevivência da rede e a sua adaptação ao ambiente envolvente.

iii. Processamento colaborativo na rede

Os nodos de uma rede de sensores *ad hoc* colaboram entre si no sentido de adquirir dados e processar dados que possam dar origem a informação útil num sistema. Para tal, quando um nodo recebe alguma informação proveniente de outro nodo seu vizinho, essa informação deve ser combinada com a sua informação local. Contudo é necessário estabelecer uma relação de compromisso (*trade-off*) entre o processamento dos dados provenientes dos outros sensores e a capacidade de comunicação da rede. Isto porque a utilização de informação de vários sensores geralmente resulta num melhor desempenho

do sistema, mas obriga a que sejam despendidos mais recursos de comunicação da rede. Por sua vez, se não se utilizarem tantos sensores, os recursos de comunicação serão menos utilizados mas poderá por em causa o desempenho do sistema.

iv. Consultas dos dados na rede

Devido à dispersão dos diversos nodos das redes, a consulta dos dados adquiridos por parte dos sistemas pode torna-se um desafio. Por este motivo, é importante desenvolver interfaces simples e interativos para fazer a consulta desses dados. Um exemplo de um interface é a consulta de dados através de comandos de voz. Neste caso é necessário que exista um componente que aceite como comando de entrada a voz e faça o interface com a rede, dando acesso aos dados existentes na rede mas, por outro lado, escondendo os detalhes individuais sobre os sensores. Outros desafios passam pela busca de mecanismos distribuídos mais eficientes para consulta de dados, para organização de dados e para aquisição e armazenamento dos dados.

v. Segurança

A segurança nas redes de sensores é de grande importância uma vez que o funcionamento dos sistemas é dependente das redes. Por este motivo, além da segurança na transmissão dos dados, é também necessário garantir a disponibilidade da rede. É também importante que as redes de sensores sejam projetadas de forma a tentar impedir intrusões ou ataques por *spoofing* à mesma.

Existem ainda outros desafios associados às redes de sensores para além dos mencionados acima. Contudo, não é objetivo desta dissertação abordar com maior detalhe esses desafios, sendo que se remete o leitor para o artigo de Ian F. Akyildiz et al. [3] para obter mais informação.

2.2.3 Aplicações

No que respeita a aplicações, as redes de sensores podem ser introduzidas nas mais diversas áreas como a área da indústria. Aqui procuram-se soluções que permitam automatizar cada vez mais os processos das unidades fabris. Além da automatização, também são utilizadas as capacidades dos dispositivos para baixar custos na manutenção, aumentar o desempenho das máquinas industriais. Para tal, o estado de conservação da máquina é monitorizado por diversos sensores e essa informação é posteriormente analisada.

Uma outra aplicação das redes de sensores pode ser a aplicação de segurança de infraestruturas contra ameaças externas. Através dos dados recolhidos através de imagens, som, movimento entre outros, é possível detetar ameaças antecipadamente, aumentando assim o tempo de reação. Podem ser exemplos de ameaças os ataques a edifícios de grande relevância como centrais de energia, e as entradas forçadas em edifícios ou habitações.

As aplicações nas áreas de controlo de tráfego também têm vindo a ser bastante exploradas. Neste caso, podem ser mencionados os controlos de tráfego de veículos terrestres, nomeadamente o controlo de tráfego em interseções através de sinais luminosos. Este controlo de tráfego pode ser feito com recurso a sensores para deteção de veículos ou até mesmo através de imagens adquiridas com câmaras de vídeo e enviadas a operadores humanos que possam controlar o estado dos sinais.

Uma solução mais radical e que poderá não abranger todos os veículos, nomeadamente os mais antigos, está associada à utilização das redes de sensores existentes nos veículos. Neste caso, as redes de sensores poderão partilhar informação sobre a densidade do tráfego, a velocidade de circulação e a localização de engarrafamentos quando os veículos se cruzam. Estes dados podem ser utilizados para apresentar informações ao condutor sobre o estado do tráfego e apresentar rotas alternativas para evitar congestionamentos.

Uma outra área de grande potencial para aplicação das redes de sensores é a área de monitorização do ambiente envolvente. Esta área vale-se da implantação dos diversos sensores para aquisição de diferentes dados que caracterizam o ambiente envolvente. Assim, é possível fornecer esses dados ao

utilizador de modo a que este possa tomar decisões que alterem o ambiente em seu favor, como por exemplo alterar as condições de temperatura de uma divisão ou até a deteção de padrões de atividade das pessoas num ambiente. As possibilidades de aplicação das redes de sensores podem ainda passar por servir para estudar a resposta da vegetação à alteração do clima, ou para identificar, rastrear e quantificar a população de aves ou outras espécies num ambiente.

Estas aplicações são possíveis graças à evolução dos dispositivos que constituem as redes de sensores. Essa evolução caracterizada pela disponibilização de equipamentos com maior capacidade de processamento e armazenamento, também permitiu a introdução de componentes de comunicação nos próprios dispositivos levando à criação dos dispositivos capazes de cooperar entre si. Com a expansão destes dispositivos e o aumento das suas funcionalidades, rapidamente se compreendeu que as soluções de domótica não se ficariam apenas pela automatização de tarefas. O espectro de utilização nesta área tornou-se bastante mais alargado, e começaram a ser idealizadas soluções de casas inteligentes (*Smart House*) que possibilitaram auxiliar os habitantes nas suas tarefas diárias de forma mais autónoma e contextualizada. Estas soluções têm por base o tema de *Ambient Intelligence* que procura aplicar ao ambiente envolvente os diversos conceitos afetos ao tema de inteligência artificial. Assim, procura-se contextualizar os dados adquiridos pelo sistema de modo a que seja possível detetar e responder às variações do ambiente de forma mais autónoma.

Estas funcionalidades disponibilizadas pelo *Ambient Intelligence* e os conhecimentos adquiridos sobre os temas de domótica e casas inteligentes permitem criar os sistemas de *Ambient Assisted Living*.

2.3 *Ambient Assisted Living*

A crescente escalada do número de habitantes a nível mundial tem vindo a criar novos desafios nas mais variadas áreas. Tendo sido atingido o número de 7 mil milhões de habitantes a 31 de Outubro de 2011, as previsões feitas pela Organização das Nações Unidas sugerem que em 2050 a população mundial possa atingir os 9,3 mil milhões de pessoas [7]. Este estudo alerta ainda para o facto da população idosa estar a crescer rapidamente, passando dos 748 milhões de pessoas em 2011 para mais de 2 mil milhões de pessoas em 2050 (Figura 2).

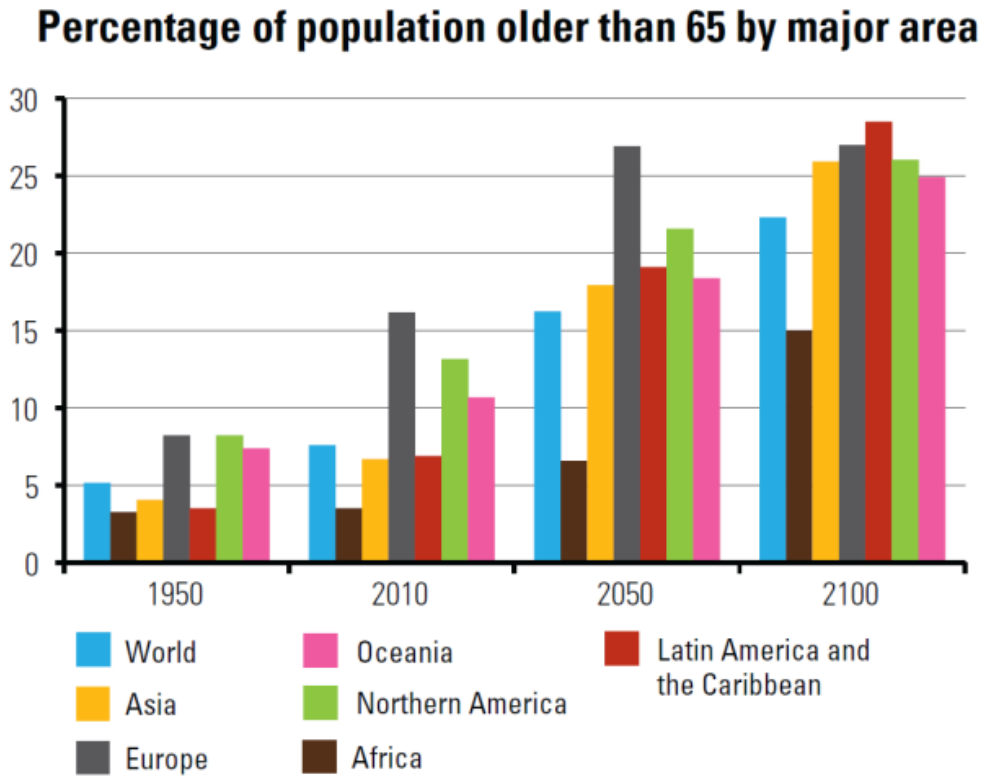


Figura 2 – Distribuição demográfica (dados da ONU) [7]

Este aumento da população idosa está relacionado com as melhorias alcançadas na prestação de cuidados e assistência médica, que se traduzem num aumento da esperança média de vida. Apesar de ser um dado positivo, esta mudança traz consigo também novos desafios e responsabilidades para com esta faixa etária. Assim, torna-se uma preocupação social garantir que os cuidados de saúde e o acompanhamento médico chegam a todos os idosos.

Além desta preocupação, é também necessário garantir que esta faixa etária mais idosa possa manter a sua autonomia, sendo para isso utilizadas as capacidades das casas inteligentes para auxiliar na execução das tarefas diárias. É sobre estas preocupações que incide o tema de *Ambient Assisted Living* (*AAL*).

Mencionando o *workshop* sobre o tema de *AAL* que ocorreu a 10 de Novembro de 2007 em Darmstadt [8], considera-se que estes sistemas devem focar-se nas tarefas de segurança de bens e pessoas, bem como nas tarefas de deteção e prevenção de emergências. A questão da segurança de pessoas neste tipo de sistemas é evidente, uma vez que estes atuam diretamente na vida das pessoas que os vão utilizar e depender deles, tornando-se necessário garantir não só a segurança como também a confiabilidade do sistema. É também importante garantir a segurança dos dados recolhidos através das monitorizações de modo a manter a privacidade dos utilizadores. Assim, é consensual que estes sistemas deverão ter uma taxa de confiabilidade muito elevada e uma segurança ainda maior que os sistemas computacionais mais comuns. No que respeita à deteção e prevenção de emergências, os utilizadores destes sistemas deverão estar providos de equipamentos que permitam fazer a monitorização dos seus sinais vitais. Este é certamente o objetivo mais importante, já que permitirá detetar situações de perigo e alertar outras pessoas desse mesmo perigo. A monitorização e aviso sobre a medicação diária a tomar por parte dos utilizadores poderá também mostrar-se uma tarefa relevante nestes sistemas, enquanto outras tarefas mais complexas ainda não foram implementadas. Uma dessas tarefas passa pela idealização de algoritmos que permitam detetar com antecedência doenças que possivelmente se venham a desenvolver nas pessoas monitorizadas.

É também consensual que os sistemas de *AAL* poderão servir não só os idosos, como também uma vasta gama de utilizadores de outras faixas etárias. Desta forma, dados como lembretes, calendários automáticos e monitorizações de atividades poderiam ser de grande importância e uma mais-valia na monitorização de qualquer tipo de utilizador.

De acordo com Kleinberger *et al.* [9], as soluções de *AAL* devem compreender o ambiente envolvente de modo a poderem adaptar-se e responder proactivamente às necessidades dos utilizadores. Deste modo será possível auxiliá-los nas suas tarefas diárias sem, contudo, limitar a sua autonomia. Outra vantagem desta compreensão do ambiente está na possibilidade de o sistema assimilar as rotinas dos utilizadores, alertando para qualquer alteração significativa, o que poderá representar uma situação de perigo. Mantém-se assim uma maior segurança e vigilância dos mesmos.

A compreensão do ambiente envolvente está associada a uma área de estudo designada *Ambient Intelligence* (*AmI*). Esta área procura valer-se das diversas tecnologias existentes para conseguir obter informações importantes sobre o ambiente envolvente e contextualizar essa mesma informação. Para tal, tira-se partido dos diversos equipamentos tecnológicos existentes atualmente que, devido às suas características como tamanho, permitem ser integrados de forma ubíqua e não intrusiva no quotidiano das pessoas. Cumpre-se deste modo o requisito de uma integração não intrusiva, facilitando assim uma maior adesão a este tipo de sistemas. Além desta característica, é também necessário que os diversos equipamentos permitam a comunicação entre si e também permitam o acesso aos seus serviços de forma externa. Isto é, um dispositivo utilizado num sistema de *AAL* deve conseguir comunicar com outros dispositivos que se encontram à sua volta (reconhecer o ambiente envolvente) e também permitir que os serviços que disponibiliza sejam acedidos por outros dispositivos. É este outro dos pontos consensuais discutidos em [8], onde se enuncia que os dispositivos deverão ser de baixo custo e universais de modo a que todas as aplicações de *AAL* possam utilizar qualquer dispositivo no mercado. Além deste fator, o interface com esses dispositivos deveria ser aberto de modo a permitir uma maior interoperabilidade.

No entanto, é necessário garantir que a integração dos dispositivos na habitação é feita de forma razoável, de modo a que as pessoas não fiquem dependentes apenas da interação com a tecnologia para realizar as suas tarefas. Esta dependência poderá levar a um isolamento destas pessoas, pelo que se procura também introduzir o auxílio providenciado por outras pessoas e o incentivo das atividades sociais. Estas funcionalidades de auxílio são mencionadas por Hong Sun *et al.* [10] como assistência mútua comunitária, onde

se pretende integrar nas casas inteligentes (*Smart House*) serviços providenciados pela sociedade como acompanhamento social, requisição de serviços de assistência, atendimento médico, entre outros. Na Figura 3 está representado o conceito de assistência mútua comunitária que se pretende implementar através da expansão dos sistemas de *AAL*.

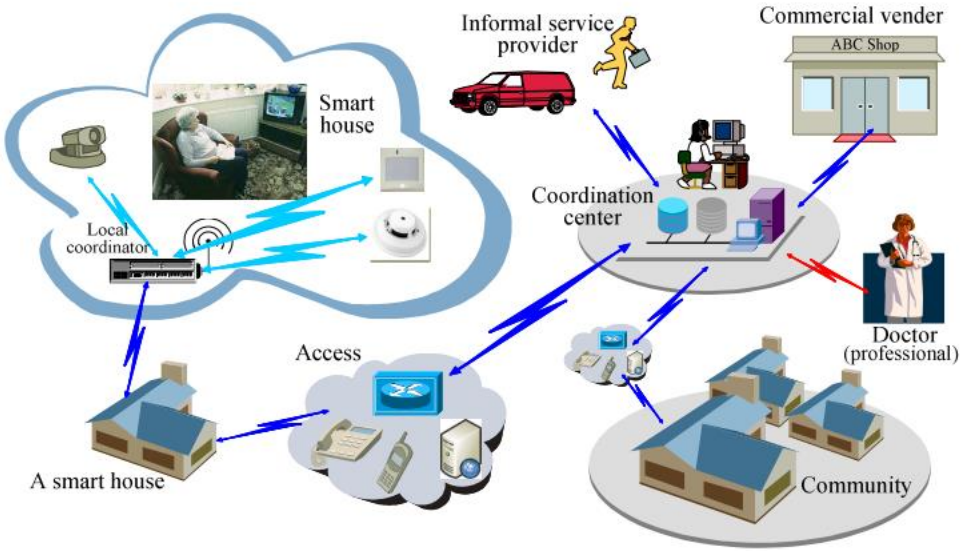


Figura 3 – Sistema de AAL – Assistência Mútua Comunitária [10]

Esta integração da comunidade nos sistemas de *AAL* poderá também vir a ser a resposta para o combate ao isolamento das pessoas que necessitam destes serviços. No entanto, esta integração necessita que as aplicações existente na habitação consigam conciliar os diferentes serviços. Para isso, as aplicações devem estar sempre em contacto com um centro de coordenação (assinalado na imagem) que por sua vez estará em contacto com os provedores dos serviços.

2.3.1 Desafios

No projeto dos sistemas de *AAL* devem ainda ser tidos em conta alguns desafios de implementação, não só relacionados com a vertente tecnológica do projeto, mas também com a vertente humana. Assim, na vertente humana, pode ver-se como um desafio a aceitação das tecnologias por parte da população mais idosa. No que respeita à vertente tecnológica, desafios como o mapeamento de

serviços e a disponibilização dinâmica desses serviços devem ser tidos em conta. Estes desafios são descritos com mais detalhe de seguida.

i. Aceitação das tecnologias

A introdução dos dispositivos tecnológicos no quotidiano das pessoas deve ser feita com alguma cautela. Este desafio deve ser tido em conta uma vez que poderá haver alguma desconfiança por parte das pessoas submetidas a estes sistemas no que respeita à sua privacidade. Além deste fator, é também necessário ter em conta a facilidade de utilização deste tipo de sistemas. São relatados dois fatores importantes na decisão destas pessoas utilizarem o sistema de *AAL*: fatores psicológicos e fatores tecnológicos.

> Fatores psicológicos

Os fatores psicológicos estão associados ao facto de algumas pessoas não aceitarem que estão a perder as suas capacidades motoras para realizar algumas tarefas, levando assim a uma frustração psicológica. Esta frustração acentua-se quando percebem que se estão a tornar dependentes da sociedade em vez de serem membros ativos desta. Para aliviar esta frustração, os sistemas de *AAL* devem dar aos idosos alguma autonomia no desempenho das tarefas e garantir que haja uma participação da sociedade na vida destas pessoas.

> Fatores tecnológicos

Os fatores tecnológicos estão relacionados com a aptidão para lidar com as tecnologias inseridas na habitação. Este problema é provocado em parte por alguma desconfiança dos idosos na aplicação das novas tecnologias. Assim, é necessário criar interfaces que se adaptem ao utilizador e sejam de fácil utilização, bem como dar a formação necessária para a utilização dos mesmos. Além dos interfaces de fácil utilização, é também necessário introduzir antecipadamente os dispositivos que permitem auxiliar as pessoas, num cenário em que as pessoas ainda não estejam dependentes destes. Essa introdução antecipada destes dispositivos na vida das pessoas levará a que haja uma maior aceitação quando forem realmente necessários.

ii. Disponibilização dinâmica de serviços

Os serviços integrados nos sistemas de AAL estão dependentes da disponibilidade quer dos diversos dispositivos, quer dos sistemas prestadores de serviços (atendimento ao domicílio). Esta dinâmica na disponibilização de serviços pode criar um grande desafio na gestão dos diversos serviços providenciados por estes sistemas. Uma das soluções apresentadas para fazer face a este desafio passa por criar o sistema com uma arquitetura orientada a serviços (*SOA* [11]). Esta arquitetura assenta sobre um paradigma cliente-servidor, onde o servidor disponibiliza serviços que podem ser consumidos pelo cliente. Cada serviço corresponde a uma funcionalidade específica que pode ser requisitada por um ou mais clientes, sendo que cabe ao sistema a gestão desses serviços. Uma das aplicações deste tipo de arquitetura é apresentada pela plataforma *OSGi*.

Além da disponibilização dos serviços numa habitação, este mesmo conceito tem sido aplicado aos serviços fornecidos pelas pessoas em forma de atividades. Dois exemplos desta aplicação são os *frameworks BPEL4People* e *WS-Human Task*. Nestes *frameworks* são disponibilizadas notações, diagramas de estados e *APIs* específicas para as tarefas executadas pelas pessoas, assim como protocolos de coordenação que permitem encarar as tarefas humanas como serviços que podem ser prestados. Estas duas abordagens podem servir de modelo para complementar os sistemas de *AAL* e integrar os serviços da comunidade no sistema.

iii. Mapeamento de serviços

Outro dos grandes desafios com que os sistemas de *AAL* se deparam é a forma como o sistema consegue descobrir os serviços que são providenciados/requisitados. Este mapeamento de serviços deve ser projetado de tal forma a que seja um processo automático executado por parte do computador central. Sendo a descrição do serviço fundamental para o mapeamento do mesmo, é necessário que haja um suporte semântico que permita fazer essa descrição. Para tal, podem ser utilizadas ontologias que descrevem o domínio dos ambientes de casas inteligentes, sendo a abordagem

mais utilizada designada por *OWL-S*. Esta permite que sejam descritos os serviços semanticamente, na perspetiva das instâncias, na requisição, invocação ou até na composição dos serviços. Contudo, a utilização destas ferramentas revelam a desvantagem de despenderem bastante tempo para executar o processo de correspondência do serviço.

Conforme mencionado, as aplicações nos sistemas de *AAL* deverão estar cientes do contexto em que se encontram, deverão reagir às alterações externas do ambiente e adaptar o seu comportamento de acordo com os objetivos, necessidades e situação em que os utilizadores se encontram [12]. Isto é, estas aplicações necessitam compreender os serviços disponibilizados e utilizá-los da forma mais adequada ao auxílio das pessoas dependentes destes sistemas. Para tal, existem algumas *frameworks* que permitem atingir este objetivo, como *Context Toolkit*, *Context Broker Architecture*, *Context Management Framework*, *Gaia*, *CASS*, *CORTEX*, *Hydrogen*, *SOCAM*, entre outras. Estas *frameworks* providenciam *APIs* de alto nível que permitem aos programadores uma abstração da camada de recolha de dados e posterior contextualização.

Uma das formas de possibilitar a contextualização de dados é através das ontologias como mencionado em [13]. Este tema será abordado com maior detalhe no próximo capítulo.

2.3.2 Aplicações

Dentre os exemplos da implementação dos sistemas de *AAL* podem ser referidos os projetos *Gator Tech Smart House* [14], [15], *Personal Assistent Unit for Living* [16], [17], *Health Smart Home* [6], [18], [19], *Welfare Techno House* [20], [21], *MavHome* [22], [23], *Center for Advanced Studies on Adaptative Systems* [24]–[26], *I-Living* [27], *Aware Home* [28], [29], e *Amigo* [30]–[32], entre outros.

i. Gator Tech Smart House

O projeto *Gator Tech Smart House* (*GTSH*) [14], [15] localizado na Florida, EUA, é o resultado de um projeto anterior denominado *Matilda Smart House* [33]. Neste projeto optou-se por direcionar a atenção às necessidades dos idosos e das pessoas com algum tipo de incapacidade. Assim, foram criados ambientes "assistidos" que permitiam, através da disponibilização de serviços, auxiliar os habitantes nas suas tarefas diárias. Serviços como caixas de correio inteligentes que notificam os habitantes da chegada de novo correio, serviços de monitorização dos padrões de sono e até serviços que permitem fazer ligações de áudio e vídeo entre habitações diferentes para combater a solidão foram testados neste projeto. No que respeita à segurança, foi desenvolvido um sistema que permitia monitorizar continuamente todas as portas e janelas da habitação e informa o utilizador se estão abertas ou fechadas. Já no campo da deteção e prevenção de situações de emergência, está a ser desenvolvido um sistema que permitirá antever potenciais situações de emergência (como quedas, alterações de parâmetros de saúde, etc.) e alertar tanto o utilizador como pessoas não residentes na habitação (médicos e/ou familiares). Por fim, é também destacado um serviço que está também em desenvolvimento e permitirá relembrar os residentes através de sinais sonoros e visuais sobre eventos que estão marcados nos seus calendários. Também deverá relembrar os utilizadores sobre a necessidade de tomar a medicação prescrita.

Neste projeto, os investigadores descobriram que uma das tarefas mais morosas e complexas é a de possibilitar a interoperabilidade entre os diversos tipos de dispositivos. Isto porque este tipo de sistemas baseia-se em dispositivos que adotam protocolos e modos de controlo diferentes. Além deste problema, é necessário que haja uma utilização colaborativa entre os dispositivos existentes de modo a conseguir fornecer os serviços requeridos neste tipo de habitações. Outra tarefa que também demonstrou ser bastante morosa foi a de ligação dos dispositivos escolhidos para o sistema. Também se notou que esta tarefa demonstra uma grande suscetibilidade para a ocorrência de erros devido à quantidade de portos I/O a serem ligados.

De modo a conseguir atingir a interoperabilidade entre todos os dispositivos, foi criado neste projeto um *middleware* denominado Atlas que é composto por seis camadas: Camada física; Camada de sensores; Camada de serviços; Camada de Conhecimento; Camada de Gestão de Contexto; e a Camada de Aplicação. Este *middleware* tem como base os serviços disponibilizados pelo *OSGi* e será abordado numa secção posterior.

Por fim, é também de assinalar a necessidade de contextualização dos dados obtidos dos diversos sensores. Esta contextualização permite compreender os dados recolhidos pelos diversos sensores e utilizá-los como uma fonte de informação. Por este motivo, é utilizado neste projeto uma camada que permite fazer essa contextualização, recorrendo a ontologias. Este tema será discutido também numa secção posterior.

ii. Personal Assistent Unit for Living

O projeto *Personal Assistent Unit for Living* (*PAUL*) [16], [17] teve início em Dezembro de 2007 e foi desenvolvido por uma equipa de investigadores da Universidade de Kaiserlautern. Neste projeto vinte habitações foram equipadas com diversos sensores e atuadores que permitem manter um registo das atividades dos seus ocupantes e providenciar auxílio para desempenhar algumas das tarefas dos mesmos. É referido também que apenas foram utilizados sensores e atuadores cujo interface é feito através do protocolo *KNX*. Como se pode compreender, esta opção limita a liberdade de escolha sobre novos equipamentos que possam surgir futuramente e que sejam uma mais-valia para as atividades monitorizadas na habitação.

Apesar do sistema estar distribuído por toda a habitação, o núcleo do sistema *PAUL* é um *tablet-PC* onde é apresentada uma interface ao utilizador e que permite que este decida qual a tarefa a desempenhar. É também nesse *tablet-PC* que são processados e guardados os dados provenientes dos sensores, aumentando assim a segurança dos dados e a privacidade dos utilizadores. Esses dados, após serem analisados, permitem criar padrões de atividade e ajustar o sistema para cooperar autonomamente nas diversas atividades. Também são esses mesmos dados que permitem detetar situações de emergência e lançar alertas, à semelhança do sistema anterior.

No que diz respeito ao aspeto técnico do sistema *PAUL*, este é composto por uma componente de interface com o utilizador conseguida através do *tablet-PC*, uma componente de processamento e uniformização dos dados recebidos dos sensores presentes na habitação e uma componente para interpretação dos dados uniformizados e consequente ativação dos atuadores para auxiliar o utilizador nas suas tarefas. De acordo com [16], o sistema está modelado para reagir aos eventos gerados pelos sensores existentes na casa. Deste modo, quando um sensor lança um evento, é ativada a unidade de processamento e uniformização dos dados e seguidamente a unidade que faz a interpretação desses dados. Através destes passos é possível automatizar algumas operações de acordo com os sensores ativados e detetar possíveis situações de emergência.

iii. Health Smart Home

O projeto *Health Smart Home (HSH)* [6], [18], [19] foi desenvolvido em França, também como uma possível resposta ao crescente aumento da população idosa. Neste projeto foi tida em conta principalmente a vertente de monitorização clínica dos idosos e as respostas às necessidades desta faixa etária. Esta opção está relacionada com o facto de ser necessário permitir a esta faixa etária regressar ao seu lar e manter a monitorização das condições de saúde devido a uma situação de doença crónica ou até mesmos após uma intervenção clínica. Assim, são monitorizados dados referentes a pressão arterial, frequência cardíaca, peso, atividade física entre outros. Também são monitorizados dados referentes à postura dos habitantes e de possíveis quedas.

Todos os dados monitorizados são enviados para um sistema central que analisa os seus valores e responde por comparação com um valor de referência. Essa resposta pode ser o envio de um sinal de alarme para uma unidade hospitalar, envio de informação para médicos ou paramédicos, ou ainda fornecer informações aos serviços de cuidados de saúde que possam existir em casa.

Neste projeto as habitações alguns sensores foram ligados através do barramento *CAN*, enquanto outros foram ligados através de uma ligação *RF*. É assim previsível a existência de uma camada de abstração que permita gerir os

dados provenientes de diferentes fontes e que possa uniformizar esses dados de modo a serem processados.

Os dados recebidos dos sensores e posteriormente processados servem o propósito da automatização de algumas tarefas. Para tal, foram criadas regras e idealizados cenários de atuação de acordo com os dados recebidos dos sensores e os valores de referência.

Além de servir o propósito da automatização de tarefas, os dados recolhidos pelos sensores permitem também uma monitorização constante e mais precisa dos padrões de movimento das pessoas no interior da casa. Desta forma o requisito de deteção de anomalias nas rotinas dos idosos poderá ser cumprido e poderá ainda possibilitar a deteção antecipada de possíveis problemas de saúde que afetem o habitante a ser monitorizado.

iv. Welfare Techno House

O projeto *Welfare Techno House* (*WTH*) [20], [21] tem vindo a ser desenvolvido no Japão, desde 1995. À semelhança dos projetos referenciados anteriormente, este tem também como principal objetivo providenciar auxílio a idosos e pessoas com incapacidades nas suas tarefas, de modo a que possam permanecer nas suas habitações mantendo dos cuidados de saúde que lhes são requeridos.

Neste projeto são apresentadas diversas formas de monitorização, sendo uma delas a monitorização com recurso ao exame de ECG. Esta monitorização pode ser feita através de têxteis com capacidades condutivas ou mesmo através da água. Para tal foram instalados na cama do paciente têxteis com capacidades condutivas que permitem manter uma monitorização constante e não intrusiva quando este se encontra a dormir. Por sua vez, a monitorização através da água é conseguida através de um sistema instalado na banheira do paciente. Além do exame de ECG, também são monitorizados dados como peso e volume de urina a fim de despistar qualquer tipo de doença.

No que respeita à recolha dos dados, esta é feita automaticamente quando há alteração de um estado conhecido. São dados como exemplo a alteração do nível da água no caso da banheira ou o aumento da temperatura

detetada na cama do paciente. Estas alterações provocam a resposta de monitorização automática dos pacientes.

À semelhança dos sistemas até agora discutidos, é também usado um sistema central para armazenar os dados provenientes dos diversos sensores espalhados pela habitação. Esses dados são posteriormente guardados num servidor baseado no sistema operativo Linux e podem ser acedidos através de uma ligação de acesso remoto.

Uma das características deste sistema é que todos os dados estão devidamente associados a cada paciente, sendo que esta distinção entre pacientes foi um dos desafios que se revelou de grande importância ao longo do projeto. Foram propostas abordagens para resolver este desafio que se baseavam na identificação da forma de onda específica do exame de ECG ou no reconhecimento facial dos pacientes, dependendo do local onde estes métodos seriam utilizados.

v. MavHome

O projeto *MavHome* (*Managing an Intelligent Versatile Home*) [22], [23] desenvolvido pela Universidade do Texas tem um espectro mais abrangente do tema de *AAL*. Neste projeto não são apenas consideradas soluções para os idosos e incapacitados, mas são também apresentadas soluções para todas as pessoas que habitem uma casa independentemente da sua faixa etária. Apesar desta ligeira diferença, os objetivos de monitorização, aprendizagem, adaptação às rotinas dos habitantes e automatização de processos continua a ser um ponto em comum com os projetos referidos anteriormente.

No que respeita à arquitetura do sistema, este está dividido em quatro camadas: uma camada Física que é composta por todos os sensores/atuadores existentes na habitação; uma camada de Comunicação que faz a partilha da informação entre os diferentes agentes do sistema; uma camada de Informação que recolhe a informação e gera inferências utilizadas para a tomada de decisões; e por fim uma camada de Decisão que seleciona uma ação a executar. São estas quatro camadas que, trabalhando em conjunto, fazem com que esta habitação possa interpretar as ações dos seus habitantes e aprender as suas

rotinas. Com essa aprendizagem, passará posteriormente a executar as tarefas automaticamente proporcionando um maior conforto aos habitantes.

Além da monitorização e automatização, é também de salientar a incorporação de algoritmos que permitem prever quais as ações futuras que o sujeito vai executar. Desta forma, a resposta a determinados eventos gerados pelo sujeito torna-se mais rápida e acertada.

A ligação entre os diversos sensores/atuadores que estão instalados numa habitação e o sistema informático que realiza a monitorização e decisão dos processos a executar é feito através do *middleware CORBA* [34]. A opção de utilizar este *middleware* possibilita a integração de novos componentes comerciais de uso genérico (*COTS – commercial of-the-shelf*) que podem ser instalados na habitação independentemente do protocolo que utilizam para comunicar com o sistema. Após a sua integração, o sistema reconhece o novo componente e disponibiliza um conjunto de serviços que permitem a interação entre este e os restantes componentes.

vi. Center for Advanced Studies on Adaptative Systems

O projeto *Center for Advanced Studies on Adaptative Systems* (*CASAS*) [24]–[26], desenvolvido pela Universidade de Washington, tem como principal objetivo reconhecer padrões de atividades que os habitantes de uma casa efetuam e automatizar processos que são repetidos pelos mesmos. Esta solução apresenta-se como uma forma de aumentar o conforto e bem-estar dos habitantes através da execução autónoma de tarefas.

À semelhança do projeto mencionado anteriormente, é grande a aposta no reconhecimento e aprendizagem das rotinas dos habitantes, associando-as corretamente a cada utilizador. Esta aprendizagem é resultado da aplicação de algoritmos de reconhecimento aos dados adquiridos pelos sensores e traduzem-se posteriormente em modelos que representam essas rotinas.

Relativamente aos dados adquiridos pelos sensores, estes são guardados numa base de dados *SQL* acessível pelo sistema através do protocolo *Jabber* [35]. É então possível inferir sobre as ações a serem executadas aquando da alteração de estado dos sensores e/ou interpretar essa alteração como uma nova rotina que deverá ser aprendida.

vii. I-Living

O projeto *I-Living* [27] utiliza os diversos sensores, atuadores, ecrãs e dispositivos para conseguir fornecer serviços aos idosos que ocupam uma habitação. Nestas habitações procura-se garantir também que os diversos interfaces sem fios com os sensores são suportados pelo sistema, levando a uma utilização colaborativa.

No que respeita aos requisitos deste projeto, são mencionados a confiabilidade do sistema, a redução de custos e flexibilidade de integração, a segurança e privacidade dos dados recolhidos dos habitantes, a qualidade dos serviços providenciados, a mitigação dos problemas de interoperabilidade entre diferentes interfaces sem fios, a simplicidade dos interfaces Homem-máquina. Estes requisitos são normalmente transversais aos sistemas de AAL, sendo que as abordagens aos problemas podem diferir.

Relativamente aos serviços idealizados para este projeto, destacam-se a monitorização dos sinais vitais dos habitantes e a geração de perfis comportamentais dos habitantes. Estes serviços permitem a recolha de dados importantes para o despiste de possíveis doenças das quais o habitante possa padecer. Assim, dados sobre os níveis de glicose, pressão arterial, ritmo cardíaco, entre outros, são recolhidos e analisados por médicos para despiste de doenças. No caso dos perfis comportamentais, é possível através das técnicas de localização detetar padrões de movimentação que indiquem problemas como depressão, doença de Parkinson, doença de Alzheimer, entre outros.

Um outro serviço mencionado é o "*Activity Reminder*". Este atua em diferentes componentes do sistema, sendo responsável por avisar os utilizadores sobre tarefas que devem executar num determinado momento. Para tal, o sistema serve-se dos equipamentos que se encontram perto do utilizador para lançar alertas, quer sejam eles visuais ou auditivos. Como exemplo pode ser referido o horário para a toma de medicamentos prescritos. Neste caso, o sistema é responsável por avisar o utilizador para tomar a medicação, independentemente do local onde este se encontra. Além dos serviços já referidos, também foram incluídos neste projeto os serviços de auxílio de

localização de bens e os serviços de deteção de emergência citados no projeto anterior.

viii. Aware Home

O projeto *Aware Home* [28], [29] pretende pôr em prática os conceitos de *AAL* e criar sistemas inteligentes de auxílio aos idosos. Estes sistemas devem estar sempre cientes do contexto como já referido, sendo para isso necessário introduzir no sistema técnicas de reconhecimento. Assim, numa primeira abordagem foram idealizadas soluções para reconhecimento da localização e orientação das pessoas no interior da habitação com o auxílio de sensores *RFID*. Contudo, esta abordagem não permitia saber a localização exata das pessoas na habitação, pelo que se recorreu à utilização de câmaras e algoritmos de reconhecimento de imagem para resolver este problema.

De acordo com [36], o reconhecimento através do comportamento local é também uma característica interessante a incluir nestes projetos. Isto é, pretende-se contextualizar melhor os dados recolhidos através do reconhecimento de tarefas que estão a ser executadas num local. Por exemplo, o sistema de localização e posicionamento consegue detetar que o utilizador está sentado num sofá. No entanto, a atividade que este desempenha poderá ser a leitura de um jornal ou pode apenas estar a ver televisão. Através da análise do comportamento do utilizador, é possível ao sistema inferir qual destas tarefas o utilizador está a executar.

Outra abordagem ao reconhecimento está relacionada com a interação com os objetos num determinado período de tempo. Este reconhecimento pretende utilizar os conceitos de tempo e tempo-espaço para contextualizar a informação, tornando o sistema mais inteligente. Por exemplo, a interação com o frigorífico, a máquina da loiça, o fogão e os produtos alimentares podem indicar a preparação de uma refeição. Esta contextualização da ação pode permitir ao sistema ativar serviços que auxiliem o utilizador nessa tarefa. Este tipo de reconhecimento é uma solução que ainda está a ser analisada e mostra-se como um grande desafio nestes sistemas.

A introdução de técnicas de reconhecimento para contextualização dos sinais no ambiente dá a possibilidade de criar serviços como o "*Technology*

Coach". Este serviço pretende dar suporte às pessoas que utilizam equipamentos médicos disponíveis em casa, proporcionando-lhes um guia para utilização correta dos mesmos. Outros serviços como o "*Memory Mirror*" e o "*Cook's Collage*" podem ser citados. O primeiro pretende mostrar a utilização de um objeto escolhido pelo utilizador num determinado período de tempo. É também capaz de ajudar a encontrar objetos que o utilizador não se lembra onde os deixou pela última vez. Já o serviço "*Cook's Collage*" foi projetado para auxiliar a pessoa que se encontra a preparar alguma refeição e que por qualquer motivo seja interrompida. Assim, o sistema pretende relembrar a pessoa que foi interrompida sobre qual o próximo passo a seguir para continuar a preparação da refeição.

ix. Amigo

O projeto *Amigo* [30]–[32], apesar de não estar diretamente relacionado com o auxílio aos idosos, tem como objeto de investigação o tema de *Ambient Intelligence* associado à disponibilização de serviços numa rede de domótica. Este projeto de investigação combina os princípios da automação, a eletrónica de consumo, a comunicação móvel e as tecnologias associadas à computação para providenciar serviços de forma personalizada. A sua aplicação não se restringe apenas ao contexto habitacional mas usa os dispositivos móveis para acompanhar os utilizadores nas suas saídas da habitação. Assim, é possível manter alguns serviços mesmo quando os utilizadores se encontram no trabalho ou noutras atividades fora da habitação. Esta poderá ser também uma característica importante nos sistemas de *AAL*, uma vez que desta forma será possível manter um acompanhamento dos idosos mesmo nas suas atividades sociais fora das habitações.

Além dos serviços já referidos, o projeto *Amigo* tem a particularidade de centralizar as informações dos diversos sensores e criar um serviço de gestão de contexto. Esta arquitetura permite às aplicações obter a informação já contextualizada das diversas fontes, não se preocupando com esse processo. As fontes deste serviço de gestão de contexto são também, por sua vez, serviços referentes a notificações, histórico de contexto, gestão de localização, e modelação e criação dos perfis dos utilizadores. Na Figura 4 está representada

a arquitetura utilizada no projeto *Amigo* para a disponibilização do serviço de gestão de conteúdo.

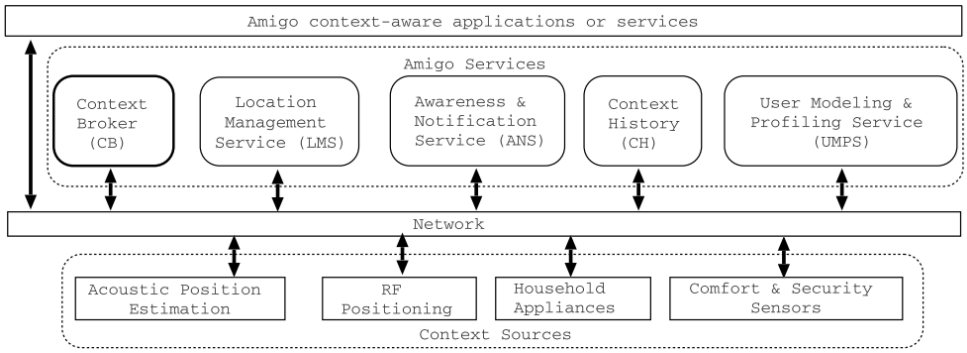


Figura 4 – Serviço de Gestão de Contexto [32]

Esta opção de arquitetura vem simplificar o desenvolvimento de aplicações "*context-aware*". Essas aplicações podem assim fazer consultas ao componente "*Context Broker*" através da linguagem *SPARQL,* para obter informações relevantes para o seu funcionamento. As informações de contexto são disponibilizadas sob a forma de descrição em *RDF*/*XML*. Este assunto será revisitado no próximo capítulo.

De referir ainda que neste projeto também foi apresentada uma técnica de localização baseada em sinais sonoros. Esta técnica utiliza diversos microfones espalhados pela habitação e, aplicando técnicas de análise dos sinais sonoros recebidos, pode fazer a localização dos utilizadores. Um exemplo genérico de aplicação desta técnica é pode ser um cenário em que o utilizador está a falar e se pretende que as camaras sejam controladas autonomamente na direção deste. No âmbito do *AAL*, esta técnica poderá ser aplicada quando se pretende que os idosos contactem os seus familiares através de videochamada.

Os projetos referidos até agora apresentam alguns pontos em comum, nomeadamente a necessidade de uma instalação prévia dos sensores/atuadores no domicílio a receber este tipo de sistemas, a monitorização constante dos habitantes e a preocupação com as possíveis situações de emergência, e também a necessidade de disponibilizar serviços aos utilizadores que permitam aumentar a sua segurança e conforto. Além destes pontos em comum, uma característica importante neste tipo de projetos na área de *Ambient Assisted Living* referenciada em todos os casos é a necessidade da contextualização e interpretação dos dados recebidos dos sensores. É através desta contextualização que se poderá atingir uma representação do ambiente envolvente referenciada em [8]. Uma das formas de fazer essa contextualização passa pela representação do ambiente envolvente recorrendo às ontologias. Este tema será discutido com mais detalhe no próximo capítulo.

2.4 Sumário

Ao longo deste capítulo foi apresentada a evolução dos sistemas de domótica que levaram à criação do conceito de casas inteligentes. As investigações que deram origem às redes de sensores e a introdução dos conceitos de inteligência artificial nas habitações foram catalisadores desta evolução. É assim possível idealizar sistemas inteligentes que permitem compreender os dados recebidos do ambiente envolvente, interpretá-los a fim de extrair informação deles, e atuar de forma autónoma. Um dos exemplos destes sistemas são os sistemas de *Ambient Assisted Living*, que concentram a sua pesquisa na resolução dos problemas das faixas etárias mais idosas e das pessoas com necessidades de cuidados e monitorização constantes.

No próximo capítulo proceder-se-á à apresentação dos conceitos de *Ambient Intelligence*, ontologias e as suas relações com os sistemas de *Ambient Assisted Living*.

Capítulo 3 – AAL e as ontologias

Este capítulo tem como propósito abordar com maior detalhe os conceitos de *Ambient Intelligence*, ontologias e *AAL*. Pretende-se também demonstrar os passos necessários para criar uma ontologia que possa figurar num cenário de *AAL*.

3.1 Introdução

O tema de *Ambient Assisted Living* (*AAL*) surge da aplicação dos conceitos de domótica e casas inteligentes na resolução dos problemas associados ao aumento da faixa etária mais idosa. Com auxílio das redes de sensores e atuadores, a domótica permite automatizar tarefas numa habitação, aumentando assim o conforto e segurança dos seus habitantes. Após esta automatização, procurou-se introduzir alguma autonomia nos sistemas de modo a que estes pudessem desempenhar as tarefas sem a intervenção dos habitantes. Para tal, foi integrado o tema de *Ambient Intelligence (AmI)* que permite associar os conceitos da área de inteligência artificial às habitações. Desta forma foi possível criar sistemas "inteligentes" que, a partir da contextualização dos dados referentes ao ambiente envolvente, executam ações de forma autónoma no sentido de automatizar ou auxiliar na execução de tarefas. É através da integração do conceito de *AmI* que são criadas as casas inteligentes (*Smart House*) onde, além da automatização de tarefas já referida, também se procura fazer uma monitorização constante dos habitantes de modo a alterar o ambiente de acordo com as suas preferências e das ações que estes desempenham.

A representação do conhecimento sobre um domínio está associada aos conceitos de AmI e inteligência artificial. A representação do conhecimento sobre o ambiente, serviços disponibilizados, rotinas e preferências das pessoas monitorizadas vai permitir ao sistema contextualizar os dados adquiridos pelas redes de sensores. Esta representação de conhecimento pode ser conseguida através das ontologias.

3.2 Ambient Intelligence

A área de *Ambient Intelligence* (*AmI*) [37] procura integrar o conceito de inteligência com as preferências e configurações que caracterizam os ambientes que habitamos. Esta área está normalmente associada a sistemas que conseguem detetar e ser responsivos à presença das pessoas num determinado ambiente. Quando se refere o tema de *AmI* é comum fazer-se referência aos sensores que se encontram integrados forma ubíqua e não intrusiva no quotidiano das pessoas para compreender o ambiente envolvente. Assim, dispositivos eletrónicos embebidos nas roupas, dispositivos eletrónicos posicionados direta ou indiretamente no corpo dos pacientes, eletrodomésticos, equipamentos portáteis ou outro tipo de sistemas embebidos podem servir o propósito de aquisição de dados para análise e compreensão por parte do sistema.

O conceito de *Ambient Intelligence* foi introduzido pelo *ISTAG* [38] (*IST Advisory Group* [39], organização da Comunidade Europeia) com a visão de que o tema de *AmI* não deveria apenas focar-se nas tecnologias mas sim no processo de inovação que parte da investigação até à entrega da solução ao utilizador final. Devido à sua abrangência, esta área pode sobrepor-se a áreas como a computação ubíqua e difusa, compreensão de contexto e sistemas embebidos. No caso da computação ubíqua e difusa, é explorado o conceito de acesso à computação em qualquer lugar e a qualquer altura, conceito esse também explorado pelo *AmI*. A diferença entre estes dois conceitos encontra-se na área de aplicação, uma vez que o conceito de *AmI* pode ser aplicado apenas localmente. A sobreposição com a área de compreensão do contexto é por si só autoexplicativa, já que esta compreensão do contexto em que o sistema se encontra a desempenhar uma determinada tarefa é uma característica também partilhada com o conceito de *AmI*. Quanto à área de sistemas embebidos, o conceito de *AmI* partilha das funcionalidades de computação e integração dos diversos dispositivos eletrónicos utilizados nos sistemas embebidos.

Conforme mencionado no Capítulo 2, o tema de *AmI* está associado à área de investigação de inteligência artificial. Os avanços na área de inteligência artificial têm vindo a ter uma integração de cada vez mais áreas de estudo como a automação, computação gráfica, comunicação, entre outras. Esta integração

em conjunto com os conceitos associados à inteligência artificial (aprendizagem, inteligência computacional, planeamento, representação de conhecimento, reconhecimento da linguagem, robótica inteligente, raciocínio incompleto ou incerto) permitem criar os sistemas de *AmI*.

3.2.1 Aprendizagem

A aprendizagem feita pelos sistemas computacionais é sem dúvida um aspeto importante. Esta começou a ser idealizada na década de 70 com a introdução das redes neurais que foram aplicadas na resolução de problemas do mundo real e permitiam obter resultados satisfatórios. Outras técnicas como a aprendizagem indutiva, a compreensão baseada em casos e as árvores de decisão foram também exploradas apresentado resultados positivos.

Na década de 80 começou a ser utilizado o termo *Data Mining* para se referir às técnicas de aprendizagem dos sistemas e posteriormente, já no final da década de 90, o conceito de *business intelligence* começou a ser amplamente utilizado pelos sistemas de informação, abrangendo todos os métodos ligados à aprendizagem.

Sendo a aprendizagem por parte dos sistemas computacionais um componente essencial nos sistemas de inteligência artificial, este deverá ser também integrado nos sistemas de *AmI*. Deste modo, o sistema deverá aprender através da observação dos dados referentes aos comportamentos dos utilizadores do sistema. Esta característica poderá também fazer com que os sistemas de *AmI* sejam melhor aceites pelos utilizadores.

3.2.2 Inteligência computacional

A inteligência computacional envolve a compreensão de padrões e métodos orientado à otimização como redes neurais, algoritmos genéricos, "*Fuzzy Logic*", entre outros. Estes métodos são projetados para a resolução de problemas específicos pelo que o ajustamento de parâmetros associados aos mesmos se revela crucial para o seu sucesso.

Esta capacidade de compreensão de padrões deverá ser incluída nos sistemas de *AmI* para permitir a compreensão das diversas escolhas do utilizador do sistema.

3.2.3 Planeamento

O planeamento é uma atividade que permite a resolução de um problema, através da realização de ações que formam um plano e conduzem a uma solução. Estes planos podem ser estabelecidos antes da sua execução ou durante a execução. Além disso, estes podem também ser de carácter deliberativo (o plano é seguido sem considerar possíveis eventos não esperados), podem ser de carácter reativo (reagem a eventos e adaptam-se à situação) ou combinar os dois tipos e criar um plano de carácter híbrido. Esta capacidade é também procurada pelos sistemas de inteligência artificial para resolução de problemas reais que o sistema possa enfrentar.

Sendo este um dos temas mais relacionados com a inteligência artificial, deve também ser incorporado nos sistemas de *AmI*. Será assim possível disponibilizar orientações e ações mais inteligentes ao utilizador.

3.2.4 Representação de conhecimento

A representação do conhecimento é uma das temáticas mais importantes associadas à Inteligência Artificial. É através desta representação que se poderão criar sistemas capazes de compreender e relacionar as diversas variáveis existentes no sistema e apresentar soluções inteligentes. Esta representação de conhecimento dá origem a bases de conhecimento que podem ser utilizadas pelos diversos sistemas e complementadas. A partilha dessas bases de conhecimento potenciada pelo desenvolvimento da internet e da *World Wide Web* permitiu que o conhecimento representado fosse aumentado. No entanto, é necessário fazer a ligação entre a informação disponibilizada e o conhecimento que a mesma representa. Uma das formas de fazer essa ligação passa por recorrer à utilização das ontologias e à Web Semântica.

Sendo uma parte tão importante na inteligência artificial, a representação do conhecimento também deve ser introduzida nos sistemas de *AmI*.

3.2.5 Raciocínio incompleto ou incerto

Os problemas do mundo real são caracterizados por vezes pela falta de dados ou incertezas quanto aos mesmos. Por este motivo, os sistemas de Inteligência Artificial devem estar preparados para apresentar decisões baseando-se em dados incompletos ou incertos. Existem alguns métodos (*Fuzzy Logic*, *Rough Sets*, etc.) que permitem lidar com esta falha nos dados recolhidos e, ainda assim, conseguir realizar o raciocínio para apresentar as soluções ao utilizador.

Uma vez que os sistemas de *AmI* estão a lidar com dados reais, é mais que provável que venham a ocorrer situações em que os dados estão incompletos ou poderão ser incertos. Por este motivo é necessário que os sistemas de *AmI* tenham suporte a esta característica.

3.2.6 Reconhecimento da linguagem natural

A forma mais comum de as pessoas interagirem é através da linguagem, seja ela voz, escrita ou gestual. Por este motivo, é expectável que os sistemas de *AmI* forneçam suporte à interação através da linguagem natural dos utilizadores. Para tal, é necessário introduzir capacidades de reconhecimento de voz no sistema e também reconhecimento da linguagem. O reconhecimento de voz trata de utilizar o sinal elétrico obtido para identificar fonemas que permitam formar uma frase. Já o reconhecimento da linguagem trata da análise sintática e semântica da mesma, de modo a descobrir qual o significado da frase adquirida. Estas duas capacidades têm vindo a ser objeto de estudo por parte de diversos grupos ligados ao tema de inteligência artificial.

3.2.7 Robótica inteligente

A utilização de robôs que permitem desempenhar tarefas é uma realidade cada vez mais comum. Desde os ambientes fabris até às habitações, é possível verificar uma crescente introdução destes dispositivos com o objetivo de facilitar a execução de tarefas. Os sistemas de *AmI* podem também beneficiar com esta introdução de robôs inteligentes, por exemplo nas habitações. Nesse caso será

então possível controlar esses robôs e desempenhar as tarefas sem intervenção dos utilizadores.

Os conceitos apresentados estão diretamente associados à Inteligência Artificial. Estes são apenas alguns dos conceitos que os sistemas de *Ambient Intelligence* devem suportar para conseguir cumprir com os objetivos a que se propõe. Por outro lado, também é necessário que o interface com o utilizador seja feito de forma amigável e que se adapte ao próprio utilizador como é mencionado em [40]. Este conceito de adaptação ao utilizador é também um dos objetivos deste tipo de sistemas. A estrutura dos sistemas de *AmI* pode ser representada pela Figura 5.

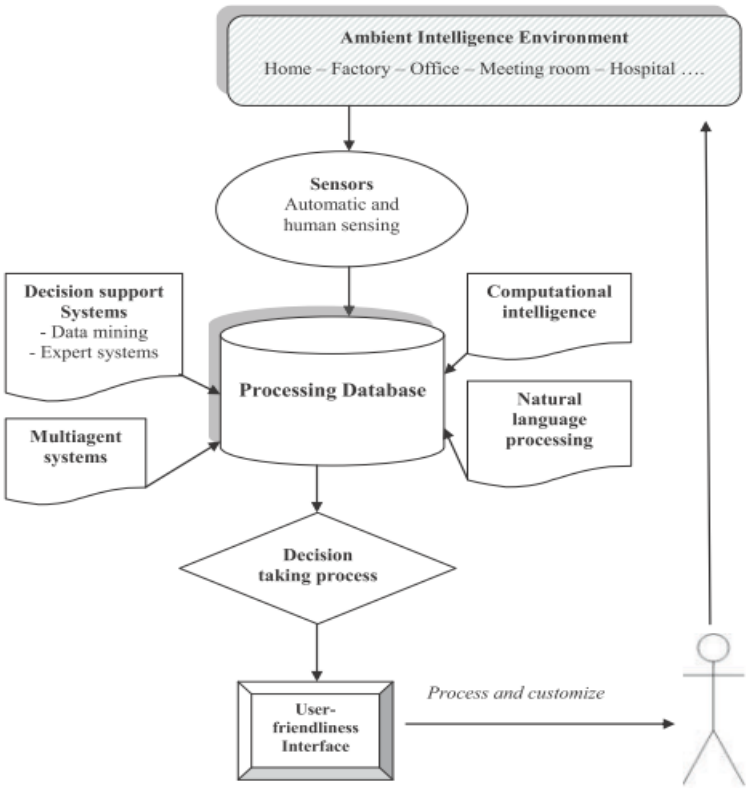


Figura 5 – Estrutura de um sistema AmI [40]

Nesta figura é possível verificar a integração dos conceitos associados à Inteligência Artificial nos sistemas de *AmI*, bem como a interação dos utilizadores com o sistema. Esta interação é tida como muito importante nestes sistemas, uma vez que é condição necessária para despoletar as respostas do sistema às ações do utilizador.

3.2.8 Aplicações

No que respeita a aplicações, o conceito de *Ambient Intelligence* pode ser explorado nas mais diversas áreas. Nas habitações são utilizadas as competências disponibilizadas pelos sistemas de *AmI* de modo a introduzir o conceito de inteligência nas habitações por meio da compreensão das ações dos habitantes e consequente aprendizagem e auxílio na execução de tarefas.

A área de veículos e transportes tem vindo também a explorar este conceito para criar sistemas capazes de conduzir autonomamente ou auxiliar os condutores na execução de manobras. Ainda na área de transportes, o conceito de *AmI* tem vindo a ser utilizado de modo a criar soluções que permitam a gestão de tráfego, gestão de situações de emergência e gestão de situações de acidentes.

Nos ambientes laborais também são aproveitadas as competências disponibilizadas pelo *AmI*. Exemplo disso são os sistemas que permitem auxiliar na tomada de decisões em grupo sobre projetos. Estes sistemas têm em conta diversos dados como contexto, argumentos e suporte a decisões considerando aspetos emocionais e racionais.

Os conceitos de *AmI* são também utilizados na área de desporto. Assim, pretende-se criar sistemas inteligentes capazes de auxiliar na prática do desporto como demonstrado pelo sistema *FlyMaster NAV*+ da *Avionics* [41].

No que respeita à integração de *AmI* nas habitações de modo a criar ambientes inteligentes, destaca-se ainda a área de cuidados de saúde e assistência aos idosos. Procura-se nesta área utilizar os conceitos de *AmI* para desenvolver sistemas inteligentes capazes de monitorizar os pacientes e, ao compreender os dados da monitorização médica, sejam capazes de lançar alertas e atuar autonomamente face a situações de perigo. Para tal, estas casas são projetadas para fornecer serviços de emergência, serviços de assistência à autonomia e serviços de conforto.

De acordo com [42], a disponibilização dos serviços mencionados potencia uma melhor qualidade de vida para os habitantes, tendo em conta que a tecnologia associada à monitorização está dispersa e "escondida" dos mesmos. Deste modo, não existe uma sensação de intrusão por parte da pessoa a ser monitorizada, levando a uma melhor aceitação do sistema. Além da melhor

qualidade de vida, também é constantemente referenciada a possibilidade de monitorizar as pessoas nas suas habitações. Esta característica permite dar um maior conforto às pessoas, sem contudo descurar os seus cuidados de saúde e o seu bem-estar.

A possibilidade de monitorização apresenta-se também como uma solução para os hospitais e centros de cuidados, uma vez que as pessoas que necessitam de cuidados médicos podem ser monitorizadas à distância. São também referenciadas as tarefas de assistência ao habitante que permitem evitar situações de emergência ou então apresentar soluções para uma situação de emergência que esteja a decorrer. Todos estes serviços são sempre acompanhados de um interface natural com o sistema e que se adapta aos indivíduos que vão usufruir dele. As capacidades disponibilizadas por estes sistemas são conseguidas através da contextualização dos dados recolhidos, sendo as ontologias uma das formas que permite fazer essa contextualização.

3.3 Ontologias

A origem etimológica do termo ontologia deriva das palavras gregas "*ontos*" que significa ser e "*logos*" que significa linguagem e razão. Este termo é proveniente da área de estudo filosófica que infere sobre a existência do ser. Apesar da sua origem completamente deslocada da área das ciências da computação, este termo veio a ser adotado mais tarde nesta área com o intuito de descrever um artefacto que permitisse modelar uma base de conhecimento sobre um domínio ou sistema.

Inicialmente foi introduzido pelos investigadores da área de Inteligência Artificial na década de 80 [43]. Estes recorriam a ontologias para definir a modelação do mundo, bem como para definir a base de conhecimento usada pelos sistemas. Desta forma poderiam obter modelos computacionais que permitiam certos tipos de "raciocínio automatizado".

Posteriormente, na década de 90, o termo ontologia passou então a fazer parte da área das ciências da computação e a sua designação foi proposta inicialmente por Tom Gruber: "*An ontology is an explicit specification of a conceptualization*" [44]. Com esta definição, o autor pretendia dar a entender que

as ontologias forneciam um meio de especificar as relações que interligam os diversos objetos, conceitos e entidades existentes num determinado contexto. Esta definição gerou bastante controvérsia devido à aplicação das ontologias em diversas áreas. No entanto, as diferentes definições podem ser resumidas através da seguinte afirmação: "*an ontology is a shared description of concepts and relationships of a domain expressed in a computer readable language*" [45].

Mais importante que a definição de ontologia é saber qual a sua utilidade. O grande objetivo das ontologias é capturar e representar o conhecimento sobre um determinado ambiente, conceptualizando os seus conceitos e estabelecendo relações entre os mesmos. Desta forma será possível a partilha desse conhecimento entre sistemas computacionais levando a uma compreensão mútua do contexto por parte dos intervenientes. Assim, será possível tornar mais intuitiva a comunicação entre pessoas e organizações, e entre organizações com estruturas diferentes. Esta representação do conhecimento poderá ser facilmente reutilizada por diversos sistemas para contextualização de dados num ambiente específico.

Um dos exemplos da aplicação de ontologias é conhecido por "*Semantic Web*" ou "*Web 3.0*" e traduz-se na representação dos dados existentes na "*World Wide Web*". Este conceito foi apresentado pela organização *W3C* [46] e pretende-se com esta abordagem utilizar de forma mais eficiente os dados disponibilizados. Ou seja, através de uma análise semântica dos dados existentes na web torna-se possível integrar, contextualizar e produzir resultados mais inteligentes.

Como já foi referido anteriormente, as ontologias estão relacionadas com o tema de Inteligência Artificial. Esta relação surge da necessidade de contextualização de dados requerida pelos sistemas que incorporam os conhecimentos da inteligência artificial. Para suprir esta necessidade, utiliza-se a modelação ontológica para representar o conhecimento sobre o ambiente desejado. No entanto, para que estes sistemas funcionem autonomamente, é necessário passar da modelação ontológica dos sistemas para uma implementação, recorrendo a linguagens que possam ser compreendidas pelos sistemas computacionais.

No que diz respeito a linguagens ontológicas para implementação de modelos, existe uma extensa lista de diversos tipos de linguagens. Estas podem ser geralmente distinguidas entre linguagens de representação de conhecimento e linguagens de base lógica.

As linguagens de base lógica (lógica de primeira ordem e lógica descritiva) são as que servem os objetivos das ontologias de forma mais abrangente. Isto porque este tipo de linguagens vale-se de uma sintaxe clara e concisa e é auxiliada por uma semântica formal bem definida. Além disso, estas linguagens estão associadas ao domínio da linguística computacional, o que torna o processo de comunicação da base de conhecimento mais fácil. Podem ser referidas como exemplo deste tipo de linguagens a *Common Logic* [47], *CycL* [48]e *KIF* [49](linguagens de lógica de primeira ordem) e *KL-ONE* [50], *Gellish* [51] e *OWL* [52] (linguagens de lógica descritiva).

Por sua vez, as linguagens de representação do conhecimento estão mais associadas à inteligência artificial. Este tipo de linguagens disponibiliza as primitivas mais usuais referentes à modelação ontológica. Isto é, as classes (ou conceitos) são definidas por atributos imputados às mesmas e estruturadas em forma de subclasses onde serão posteriormente identificadas as relações que as interligam. Como exemplo de linguagens de representação do conhecimento podem ser referidas a *F-Logic* [53] e *OKBC* [54].

Apesar das diferentes linguagens e das suas particularidades, é consensual que a representação das mesmas deverá permitir a sua leitura e análise pelo Homem, apesar de serem linguagens que servirão de base aos sistemas computacionais. Por outro lado, a semântica que é utilizada pelas linguagens deve ser bem definida para permitir fazer inferências à mesma.

A modelação e implementação ontológicas têm vindo a ser utilizadas nos mais variados contextos. Prova disso são os diversos repositórios existentes na *internet*, onde são mantidas várias ontologias sobre diversas temáticas. Esses repositórios são mantidos quer por organizações públicas, quer por organizações privadas, e permitem que sejam utilizadas algumas das ontologias que disponibilizam. Uma das formas mais rápidas de encontrar uma ontologia sobre um tema em específico é através do motor de busca criado exatamente para esse efeito: *Swoogle* [55].

Como exemplos de ontologias que podem ser encontradas na internet temos as ontologias para descrição de um ambiente empresarial (*The Enterprise Ontology* [56]), ontologias para descrição de terminologia associada ao ambiente e à Terra (*SWEET* [57]), ontologias para descrição de conhecimento biológico e biomédico (*OBO Foundry* [58]), entre outros.

Já no âmbito dos sistemas de *AAL*, a modelação através das ontologias fornece as ferramentas necessárias para a contextualização dos dados adquiridos e para a sua interpretação e análise mais inteligente. Será assim possível criar os modelos de rotinas e tarefas referenciados nos projetos apresentados em 2.3, como também automatizar e auxiliar os residentes nas suas tarefas.

3.3.1 Exemplos de ontologias

Como exemplos da modelação ontológica associadas às habitações, podem ser referidas as ontologias *DogOnt* [59], [60], *DomoML* [61], [62] e *IntelliDomo* [63], [64], entre outros.

i. DogOnt

A ontologia *DogOnt* [59], [60] foi desenvolvida no contexto da Web Semântica e pretende dar suporte aos diversos serviços e redes que se encontram nas habitações. Esta ontologia está dividida em duas partes: uma parte referente à descrição dos diversos elementos constituintes de uma habitação, quer sejam eles controláveis ou estruturais; e uma parte referente às regras que permitem modelar automaticamente os estados e funcionalidades de um dispositivo de domótica.

Esta ontologia foi desenvolvida para atuar em conjunto com o sistema DOG (*Domotic OSGi Gateway*), uma iniciativa de código livre (*open source*) que pretende criar um sistema de domótica de baixo custo. Para isso utiliza a *framework OSGi* para disponibilizar serviços associados aos dispositivos de domótica. Por este motivo, a ontologia criada foca o seu âmbito de estudo na interoperabilidade entre diferentes sistemas de domótica.

No que respeita à modelação dos elementos da habitação, estão identificadas cinco hierarquias principais: Objetos presentes na habitação; Ambiente envolvente; Estado dos dispositivos; Funcionalidade dos dispositivos; e a rede de domótica presente na habitação. São estes componentes que, em conjunto, permitem localizar os dispositivos de domótica na habitação, saber o seu estado e responder segundo uma funcionalidade presente no espaço onde se encontram.

ii. DomoML

A ontologia *DomoML* [61], [62] foi desenvolvida para atuar sobre a base do projeto *NICHE* (*Natural Interaction in Computerized Home Environment*). Este projeto pretende utilizar a linguagem humana como meio de interação com o ambiente habitacional, permitindo ao utilizador controlar, saber o estado e programar os equipamentos de domótica existentes na habitação.

À semelhança da ontologia mencionada anteriormente, esta ontologia tem como objetivo operar como protocolo de comunicação entre diferentes dispositivos de domótica existentes na habitação, sendo também seu objetivo a descrição dos dispositivos de domótica e das funcionalidades disponibilizadas por estes. Esta ontologia está dividida em três níveis: um nível para a descrição do ambiente; um nível para a descrição das funcionalidades; e um nível para a descrição dos métodos de comunicação. Estes três níveis que compõem a ontologia são utilizados em conjunto pela aplicação principal que faz a monitorização e automatização das tarefas.

iii. IntelliDomo

A ontologia *IntelliDomo* [63], [64] é baseada numa outra ontologia denominada *OntoDomo* que contém a descrição dos diversos equipamentos de domótica, dos seus atributos e das relações entre eles. Esta ontologia permite controlar de maneira autónoma e em tempo real os dispositivos de domótica presentes na habitação, reagindo às alterações do sistema envolvente.

Além da reação à alteração do estado e consequente ação por parte do sistema, também está incluído nesta ontologia um módulo de aprendizagem que

permite ao sistema detetar as tarefas executadas com maior frequência, bem como as preferências do utilizador. Essa aprendizagem é conseguida através da análise dos dados adquiridos dos sensores, e que são armazenados na base de dados do sistema.

3.4 Modelação ontológica

A modelação ontológica surge como resposta à necessidade de representar o conhecimento sobre um determinado domínio, levando à conceptualização desse domínio [65]. Assim, os conceitos que caracterizam um domínio e as relações que se estabelecem entre esses conceitos podem ser capturados e descritos numa linguagem que permita a compreensão quer por parte das pessoas, quer por parte dos sistemas computacionais. Esta capacidade de compreender os conceitos de um domínio, em conjunto com a interpretação de dados provenientes desse mesmo domínio permitem atingir o objetivo dos sistemas de *Ambient Intelligence*: a contextualização.

Através desta contextualização de dados torna-se possível desenvolver aplicações cientes do contexto em que estão a operar e fornecer serviços personalizados que respondam às necessidades do utilizador num determinado contexto [66]. No que respeita aos sistemas de *AAL*, os objetos de modelação incidem, obviamente, sobre o ambiente habitacional e os seus diversos componentes, sendo que essa modelação deverá abranger os diversos aspetos associados às habitações e aos serviços disponibilizados, bem como os utilizadores destes sistemas. Após a modelação, é possível exportar a descrição do ambiente para ficheiro, o que possibilita a interpretação das ontologias por outras máquinas e a sua partilha.

Porém, a modelação por si só não fornece todos os meios para criar os sistemas autónomos desejados pelo tema de *AAL*, sendo então necessário implementar aplicações que permitam utilizar o conhecimento modelado. Estas aplicações criadas sobre uma base ontológica são então denominadas de "*Context-aware applications*", e baseiam a sua atuação nos conceitos definidos pela ontologia.

A modelação do sistema desenvolvido foi feita com recurso ao *software Protégé* [67] desenvolvido pela universidade de Stanford. Este *software* apresenta uma interface simples e um vasto conjunto de documentação de apoio à utilização deste *software*. Além destes dois fatores, também existem diversas ontologias que servem de exemplificação e permitem uma melhor compreensão dos processos desta ferramenta.

Assim, foram tomadas como exemplo as ontologias *DomoML* [61], *DogOnt* [59] e *ELDeR* [68], e procedeu-se ao desenvolvimento de uma ontologia para um contexto de *AAL*. Este desenvolvimento seguiu os processos aconselhados por [69], sendo que se podem encontrar os cenários de automatização de algumas tarefas descritos pelo Anexo 1. Seguidamente procedeu-se à modelação do ambiente, utilizando uma abordagem "*top-down*" para descrever o mesmo. Na Figura 6 estão representados os conceitos principais afetos a um ambiente habitacional.

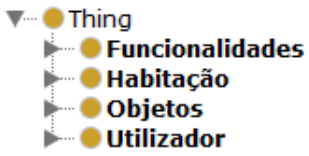


Figura 6 – Classes principais

Os conceitos apresentados na figura estão associados aos sistemas de *AAL*, sendo a classe Habitação referente ao ambiente habitacional, a classe Objetos refere-se aos diversos componentes encontrados na habitação, a classe Funcionalidades referente aos serviços disponibilizados na habitação e, por fim, a classe Utilizador que permite modelar os utilizadores do sistema.

i. Habitação

Um conceito primordial nos sistemas de *AAL* é o conceito de Habitação. Este refere-se obviamente à habitação onde se encontra o sistema implementado e sem a qual não fariam sentido estes sistemas.

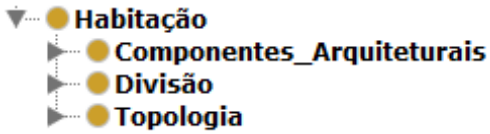


Figura 7 – Classe Habitação

Nesta primeira classe são ainda apresentados conceitos que fazem parte da habitação como Topologia, Divisão e Componentes Arquiteturais (Figura 7). Cada um destes conceitos será explicado de seguida.

ii. Topologia

Uma habitação pode adotar a topologia de Apartamento ou a topologia de Moradia (Figura 8).

Figura 8 – Classe Topologia

Neste caso, os dois conceitos deverão possuir a propriedade disjuntiva para que não se possa atribuir à mesma habitação as duas topologias simultaneamente.

iii. Divisão

O conceito de divisão pretende representar as divisões existentes numa habitação como o quarto, sala, cozinha, etc. (Figura 9).

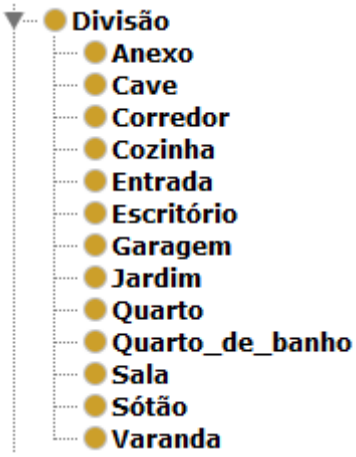


Figura 9 – Classe Divisão

Esta distinção é particularmente importante já que a cada divisão vão estar associados serviços diferentes e perfis de monitorização.

iv. Componentes Arquiteturais

O conceito de Componentes Arquiteturais foi definido com o intuito de representar os componentes característicos e teoricamente inalteráveis de qualquer habitação (Figura 10).

Figura 10 – Classe Componentes Arquiteturais

Estes conceitos representam o conhecimento mais básico de uma habitação, sendo que estes só poderão existir se estiverem associados a uma divisão (iii).

Nesta primeira análise aos constituintes deste sistema foi possível identificar alguns conceitos transversais a qualquer sistema de *AAL*. No entanto, esta representação do sistema necessita ainda que sejam definidas relações entre os diversos conceitos. Esta definição de relações é feita através das propriedades de objeto (*Object Properties*) e permitem ligar duas instâncias por uma ou mais relações. Alguns exemplos dessas relações podem ser:

- Uma Habitação tem apenas uma Topologia;
- Uma Habitação é constituída por uma ou mais Divisões;
- Uma Divisão contém pelo menos uma Porta;
- Uma Divisão contém pelo menos quatro Paredes;
- Uma Divisão contém apenas um Chão;
- Uma Divisão contém apenas um Tecto;

Estes são alguns dos exemplos de relações que se podem estabelecer entre os diversos conceitos. Estas relações servirão para modelar o sistema de forma mais completa e, consequentemente, providenciar um maior conhecimento do ambiente às aplicações que utilizam a ontologia.

v. Objetos

Numa habitação existe um conjunto de objetos que podem ou não ser controláveis (Figura 11).

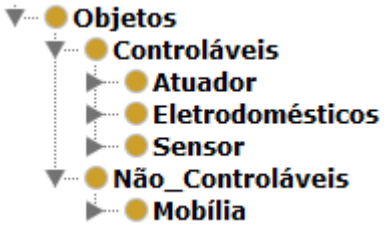


Figura 11 – Classe Objetos

Os objetos não controláveis referem-se a objetos como a mobília (Figura 12) que não podem ser controlados por este sistema. No entanto, esta informação deve estar presente na ontologia para que haja uma correta representação do conhecimento sobre o ambiente.

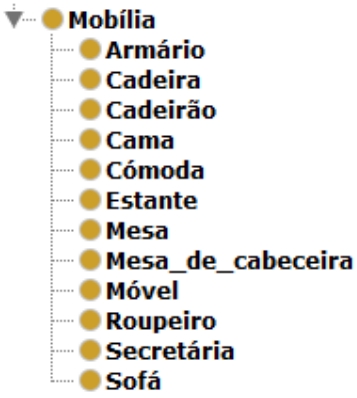


Figura 12 – Classe Mobília

Já os objetos que podem ser controlados pelo sistema estão agrupados na classe "Controláveis", sendo que esta possui três subclasses: Eletrodomésticos, Sensores e Atuadores. A classe de eletrodomésticos é dividida em duas subclasses: grandes eletrodomésticos (Figura 13) e pequenos eletrodomésticos (Figura 14). Esta divisão foi feita tendo como fator de comparação o gasto dos equipamentos.

Figura 13 – Classe Grandes Eletrodomésticos

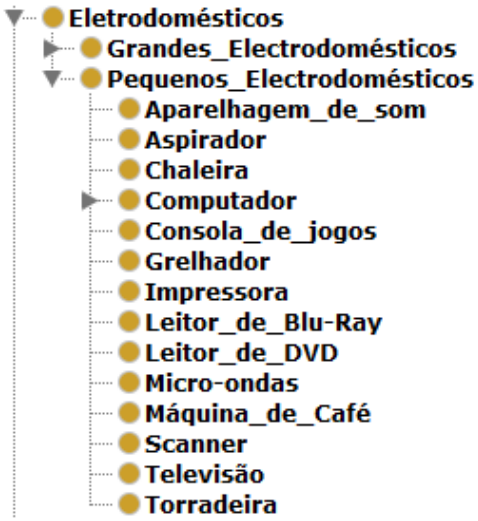


Figura 14 – Classe Pequenos Electrodomésticos

As classes “Sensor” e “Atuador” são subclasses da classe “Controláveis” e pretendem descrever os diversos sensores e atuadores presentes numa habitação (Figura 15).

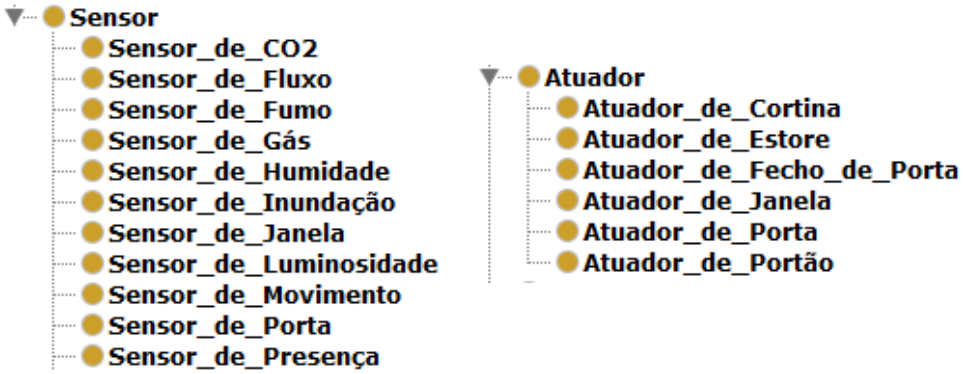


Figura 15 – Classes Sensor e Atuador

vi. Funcionalidades

Além dos componentes existentes numa habitação, é também necessário modelar o conjunto de funcionalidades que serão disponibilizadas. Estas funcionalidades podem ser elementares como ligar e desligar um eletrodoméstico, ou podem ser mais complexas como fazer o pedido para saber o estado dos sistemas de segurança existentes na habitação. Na Figura 16 estão representadas algumas dessas funcionalidades idealizadas.

Figura 16 – Classe Funcionalidades

vii. Utilizador

A classe utilizador (Figura 17) pretende representar os habitantes para os quais estes sistemas de AAL são projetados. Esta classe guarda dados sobre o utilizador para futura referência por parte das aplicações "*context-aware*".

Figura 17 – Classe Utilizador

Também é definida uma subclasse denominada Preferência para que no sistema de *AAL* as preferências possam também estar associadas às mudanças no ambiente envolvente. Desta forma, ao contextualizar a informação do ambiente com a informação proveniente destas preferências, poderá ser possível alterar o ambiente de forma a aumentar a sensação de conforto do utilizador.

No entanto, as ontologias e as suas implementações não têm capacidade por si só de interagir com o meio envolvente. É necessária uma camada de abstração que faça a ligação entre a implementação das ontologias, os dados gerados pelos diversos dispositivos e o sistema onde as ontologias vão atuar. Além desta funcionalidade, esta camada de abstração deverá ser capaz de ligar com os diversos protocolos utilizados por diferentes dispositivos, garantindo assim a interoperabilidade entre os mesmos e o sistema. Essa camada de abstração é comummente conhecida por *middleware* e será abordada com mais detalhe no próximo capítulo.

3.5 Implementação

Após a modelação do ambiente habitacional e das suas funcionalidades, é necessário passar à implementação da ontologia de modo a criar um sistema autónomo. É através desta implementação que são criadas as aplicações "inteligentes" e cientes do contexto. Para tal, existe um conjunto de ferramentas que fornecem métodos de interação com a ontologia modelada. Como exemplo pode ser referida a *framework JENA* [70].

Esta *framework* foi desenvolvida em linguagem Java e tem como objetivo permitir a criação de aplicações "*context-aware*" para a Web Semântica. Para tal, fornece diversas *APIs* de alto nível que permitem lidar com a informação dos dados em formato *RDF* (*Resource Description Framework* [71]). Este formato de dados encontra-se normalizado pela organização *W3C* e é um dos formatos utilizados para definir os modelos criados pelas ontologias.

A *framework JENA* tem um conjunto de *APIs* especificamente dedicados a lidar com as ontologias [72] e que permitem criar ontologias, fazer consultas, atualizar campos, navegar e procurar conceitos na ontologia.

i. Aplicação

A Figura 18 apresenta o exemplo de uma aplicação desenvolvida em Java e que utiliza a *framework JENA* para interagir com a ontologia criada. Nesta figura está apresentado o ecrã inicial de seleção da ontologia a utilizar.

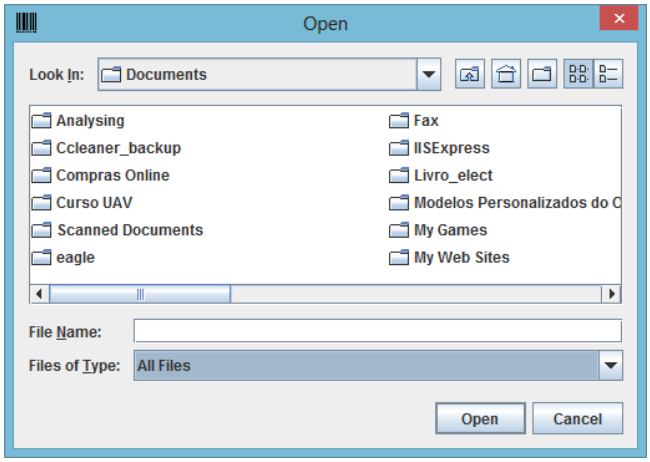


Figura 18 – Aplicação - Ecrã inicial

Após a seleção da ontologia, são inicializados os objetos que permitem lidar com a informação contida nesta. Para tal, são utilizadas as rotinas *createOntoModel*() e *getOntoClasses()* como é apresentado na Figura 19.

```
public void setupForm()
{
    this.setIconImage(iconImage.getImage());
    this.setTitle("OntoManager - Simple View");
    esrgImage = new ImageIcon(getClass().getResource("logo.jpg"));
    logojLabel.setIcon(esrgImage);

    oa.setOntoPath(filePath);
    ont = oa.createOntoModel();

    // -------------- Mostrar informações gerais  -------------- //
    URIjLabel.setText(oa.getURI());
    locjLabel.setText(oa.getOntoPath(true));

    rootClassesjTable.setModel(hf.populateModel(oa.getOntoClasses(ont),"New"));
    viewjTable.setModel(hf.populateModel(oa.getOntObjectProperties(ont),"Object View"));
}
```

Figura 19 – Aplicação - Código de inicialização

Estas rotinas utilizam as *APIs* disponibilizadas pela *framework JENA* para adquirir as informações associadas à ontologia selecionada. Na Figura 20 e Figura 21 são apresentados detalhes sobre o código destas rotinas.

```
public OntModel createOntoModel()
{
    OntModel domainModel = ModelFactory.createOntologyModel();  //objeto ontmodel para

    try{
        domainModel.read(getOntoPath(false));   //leitura do ficheiro
    }
    catch(JenaException je){    //erro no formato do ficheiro. deve ser gravado no form
        JOptionPane.showMessageDialog(null, "Wrong OWL format.\nPlease save as RDF/XML"
        System.exit(0);
    }
    setURI(domainModel.getNsPrefixURI(""));     //aquisição do prefixo da ontologia

    return domainModel;
}
```

Figura 20 – Aplicação - createOntoModel()

Esta rotina permite inicializar um objeto que possibilita a interação com a ontologia.

```
public ArrayList<String> getOntoClasses(OntModel ontModel)
{
    ArrayList<String> rows = new ArrayList<String> ();

    //iterator para percorrer o conjunto de classes existentes na ontologia
    Iterator<OntClass> i = ontModel.listNamedClasses()
                .filterDrop( new Filter<OntClass>() {
                        @Override
                        public boolean accept( OntClass r ) {
                            return r.isAnon();
                        }} );
    while (i.hasNext()) {
        testFunction(i.next(), rows, 0);
    }
    return rows;
}
```

Figura 21 – Aplicação - getOntoClasses()

Nesta rotina é utilizado o objeto criado anteriormente para adquirir as classes definidas na ontologia.

A aquisição dos dados sobre as propriedades de objetos que definem as relações entre as diferentes classes é feita através da rotina apresentada na Figura 22. Esta utiliza um objeto (*Iterator*) para percorrer uma lista de propriedades de objetos existentes na ontologia.

```
public ArrayList<String> getOntObjectProperties (OntModel ontModel)
{
    ArrayList<String> al = new ArrayList<String>();

    //Iterator para percorrer a lista de propriedades de objetos
    ExtendedIterator<ObjectProperty> listObjectProperties = ontModel.listObjectProperties();
    while (listObjectProperties.hasNext())
    {
        ObjectProperty op = listObjectProperties.next();
        al.add(op.getLocalName());
    }
    return al;
}
```

Figura 22 – Aplicação - getOntObjectProperties()

Da mesma forma, as propriedades de dados associadas a cada classe são adquiridas através da rotina apresentada pela Figura 23.

```
public ArrayList<String> getOntDataProperties (OntModel ontModel)
{
    ArrayList<String> al = new ArrayList<String>();

    ExtendedIterator<DatatypeProperty> listDatatypeProperties = ontModel.listDatatypeProperties();
    while (listDatatypeProperties.hasNext())
    {
        DatatypeProperty dp = listDatatypeProperties.next();
        al.add(dp.getLocalName());
    }
    return al;
}
```

Figura 23 – Aplicação - getOntDataProperties()

Por fim, na Figura 24 está representada a rotina que permite adquirir todas as instâncias presentes na ontologia.

```
public ArrayList<String> getOntInstances (OntModel ontModel)
{
    ArrayList<String> al = new ArrayList<String>();

    ExtendedIterator<Individual> listIndividuals = ontModel.listIndividuals();
    while (listIndividuals.hasNext())
    {
        Individual id = listIndividuals.next();
        al.add(id.getLocalName());
    }
    return al;
}
```

Figura 24 – Aplicação - getOntInstances()

Os dados adquiridos são apresentados ao utilizador através de um interface representado na Figura 25.

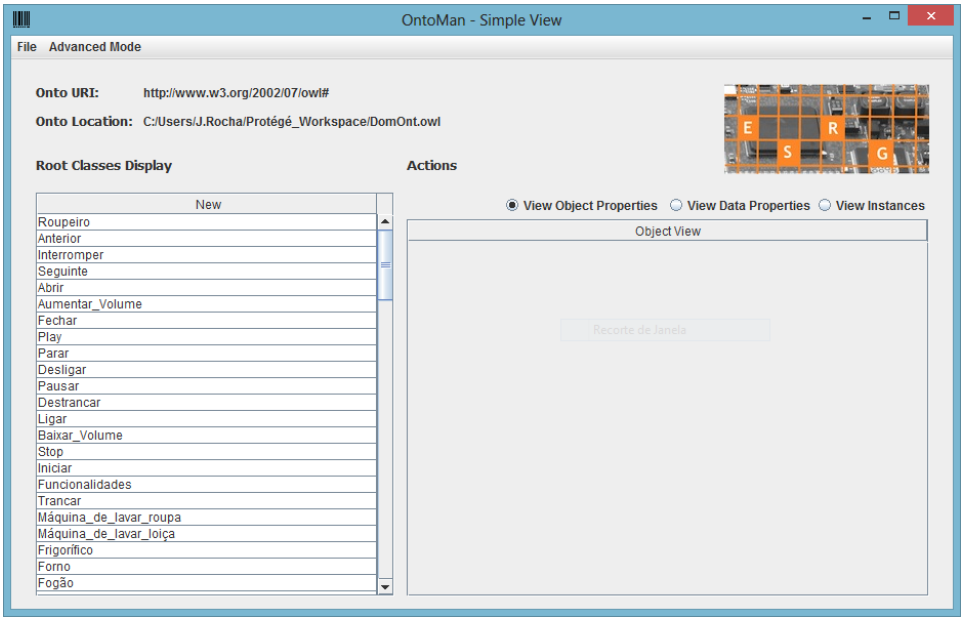


Figura 25 – Aplicação

Através desta aplicação verificou-se que é possível capturar o conhecimento representado pela ontologia e transpô-lo para uma aplicação. Recorrendo às *APIs* fornecidas pelo *JENA* torna-se mais simples a implementação destas aplicações como era esperado.

3.6 Sumário

Ao longo deste capítulo foram apresentadas as relações existentes entre os temas de *Ambient Intelligence*, ontologias e *AAL*. Além destas relações, foi ainda demonstrada a modelação de um ambiente habitacional. Por fim, foi testada a implementação através da criação de uma aplicação que permite compreender os conceitos definidos na ontologia criada.

No próximo capítulo o tema de *AAL* será novamente referido com o intuito de o relacionar com o tema de *middleware*. Serão então referidas as dependências entre estes dois temas e apresentado com mais detalhe o tema *middleware*.

Capítulo 4 – AAL e o Middleware

Após a compreensão do tema de *Ambient Assisted Living*, torna-se importante compreender a relação entre o tema de *AAL* e o tema de *middleware*. É sobre este propósito que se foca o conteúdo deste capítulo.

4.1 Introdução

Como tem sido discutido ao longo desta dissertação, os sistemas de *AAL* utilizam as funcionalidades disponibilizadas pelas redes de sensores para recolher dados sobre o ambiente, e pelas redes de atuadores para alterar esse ambiente de forma a aumentar o conforto e segurança dos habitantes. Estas redes de sensores e atuadores são constituídas por diversos dispositivos, cada qual com os seus interfaces específicos e, por vezes, com protocolos de comunicação distintos. Esta diversidade de interfaces e protocolos de comunicação levanta um problema de interoperabilidade quando é necessário fazer a integração de todos os dados disponíveis num ambiente, como é o caso dos sistemas de *AAL*.

Para lidar com este problema, foi idealizada uma abordagem que permitisse gerir os diversos interfaces e protocolos de comunicação, providenciando um acesso aos dados de forma homogénea. Essa abordagem é conhecida como *middleware* e tem como objetivo criar uma camada de abstração entre as aplicações que necessitam dos dados e as diversas fontes de dados, escondendo a complexidade da comunicação. Para tal, são disponibilizadas diversas *APIs* que podem ser utilizadas pelas aplicações. Estas *APIs* são por sua vez responsáveis por fazer a interação com os dispositivos e os demais intervenientes existentes no sistema.

4.2 Middleware

A grande diversidade de dispositivos, tecnologias e protocolos de comunicação utilizados nos sistemas de *AAL* colocam um problema de interoperabilidade aquando da interação entre os diversos intervenientes destes sistemas. Com vista a solucionar este problema, foi idealizada uma camada de abstração com o objetivo de gerir o acesso aos dados provenientes dos dispositivos de forma homogénea. Assim, esta camada de abstração permite que as equipas de desenvolvimento de *software* utilizem os dados sem se preocuparem com a complexidade da comunicação entre os intervenientes no sistema. A esta camada de abstração dá-se o nome de *middleware*, cuja definição mais abrangente pode ser encontrada em [73]:

"*Middleware is the software that assists an application to interact or communicate with other applications, networks, hardware, and/or operating systems. This software assists programmers by relieving them of complex connections needed in a distributed system. It provides tools for improving quality of service (QoS), security, message passing, directory services, file services, etc. that can be invisible to the user.*"

De acordo com T. Bishop e R. Karne [73] verifica-se que é possível fazer uma distinção entre os diversos tipos de *middleware* e o seu propósito. Assim, o *middleware* pode ser dividido em duas classificações principais: Integração e Aplicação. No caso da Integração, são referidos os *middleware* que têm uma maneira específica de ser integrados num sistema, sendo que cada um destes middleware tem diferentes protocolos de comunicação e formas de operar. Já no caso da Aplicação, o *middleware* criado é integrado para interagir especificamente com determinados tipos de funções das aplicações, sendo por isso desenvolvidos para interagir com uma aplicação específica. Além desta divisão, é ainda possível subdividir estas duas categorias principais como apresentado na Figura 26.

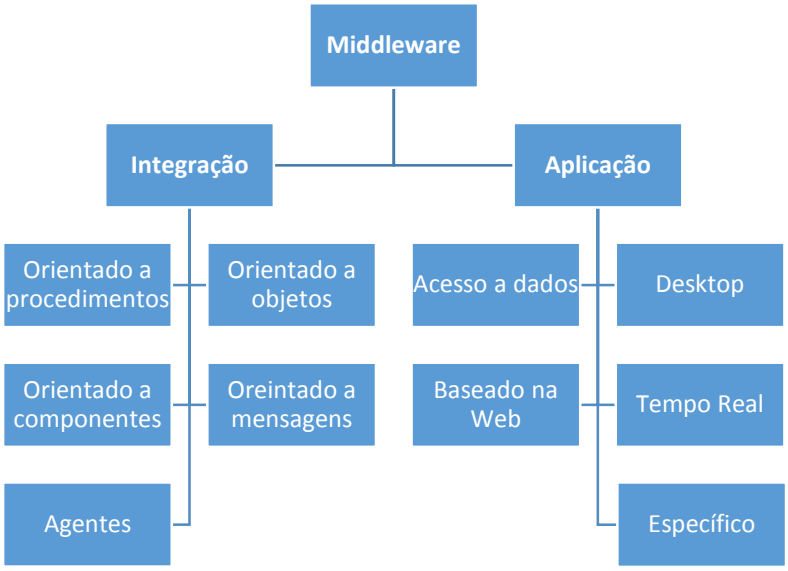


Figura 26 – Classificação de middleware [73]

4.2.1 Integração

A componente associada à Integração pode ser subdividida em *middleware* orientado aos procedimentos, orientado a objetos, orientado a mensagens, baseado em componentes, e agente.

i. Orientada a Procedimentos

O *middleware* orientado a procedimentos utiliza os conceitos de "*stubs*" e "*skeletons*" associados aos sistemas distribuídos. Nesta abordagem, os dados são convertidos em mensagens pelos *stubs* do lado do cliente e enviados para o servidor através de comunicação síncrona. Já do lado do servidor, a mensagem é recebida e convertida novamente em dados através dos *skeletons*, sendo depois enviados esses dados para a aplicação de servidor para processamento. Quando este processo está completo, ocorre o processo inverso para dar resposta ao pedido.

As vantagens deste tipo de abordagem estão ligadas aos serviços prestados, sendo que estes são conhecidos antecipadamente, têm designações fixas, o processamento dos pedidos é feito pelo servidor e pode transacionar múltiplos tipos de formatos de dados. Já as desvantagens estão na fraca escalabilidade do sistema e ser um processo rígido que está intrinsecamente ligado e dependente do procedimento que se pretende executar.

ii. Orientada a Objetos

A abordagem orientada a objetos tem também associado os conceitos de sistemas distribuídos, sendo que dá suporte a pedidos através de objetos distribuídos. A comunicação entre objetos pode ser feita de forma síncrona ou assíncrona, e suporta múltiplas transações através do agrupamento de vários pedidos similares dos diversos clientes. O *middleware* orientado a objetos utiliza inicialmente um objeto do lado do cliente para fazer uma chamada a um método remoto que implementa esse objeto. À semelhança da abordagem anterior, existe também do lado do cliente um componente (*stub*) para fazer a conversão dos dados em mensagens que possam ser enviadas através do componente denominado *Broker*. Este componente atua como um intermediário na comunicação que pode selecionar as diversas fontes de dados e organizar os dados recolhidos. Do lado do servidor a mensagem enviada é convertida através dos *skeletons* e os dados obtidos são submetidos para desempenhar o processo requerido.

Esta abordagem permite uma maior escalabilidade do sistema através dos objetos do servidor e suporta a gestão de carga de trabalho no servidor. Também pode operar sobre diferentes tipos de dados, formatos e pode fazer uma gestão interna dos objetos distribuídos. Já as desvantagens estão na necessidade de existir uma ligação prévia ao servidor antes da execução e a necessidade de formatar os dados para serem recebidos pelo servidor.

iii. Orientada a Mensagens

A abordagem orientada a mensagens pode ser subdividida em dois tipos: *Message Passing/Queuing* e *Message Publish/Subscribe*. No primeiro tipo, as aplicações enviam as mensagens através do *middleware* do lado do cliente; por sua vez, o *middleware* do lado do servidor capta as mensagens e insere-as numa pilha numa ordem pré-determinada. Neste caso o *middleware* do servidor atua como um *router* para as mensagens e normalmente não interage com estas.

No segundo tipo, o *middleware* está projetado para responder a eventos. Para que o cliente possa enviar e receber mensagens, é necessário que este se ligue a um barramento de informação. Dependendo do seu objetivo (publicação

de dados ou subscrição) é registado no barramento um novo evento. Esse evento é utilizado pelo barramento de dados quando o servidor envia o sinal ao barramento a informar que já tem os dados disponíveis.

iv. Baseada em Componentes

Esta abordagem de *middleware* captura a definição de componente e tenta introduzir essa mesma definição na sua implementação. A definição de componente é tida como um programa que executa uma função específica e é desenvolvida de forma a operar com outros componentes e aplicações. No caso deste *middleware*, tenta fazer-se com que existam diversos componentes que podem ser implementados quer aquando da sua compilação ou mesmo durante a sua execução. Para tal, são necessárias grandes bibliotecas com diversos componentes que suportam a construção de sistemas que podem atuar em múltiplas plataformas.

A grande vantagem desta abordagem é a sua configuração e reconfiguração. Esta reconfiguração pode também ser feita durante a execução, o que proporciona uma grande flexibilidade ao sistema e um maior suporte a diversas aplicações.

v. Agentes

Os agentes são considerados um *middleware* que consiste em diversos componentes como entidades (objetos), meios de comunicação (entre diferentes agentes) e regras (utilizadas para coordenação da comunicação entre agentes). Estes agentes têm capacidade de atuar autonomamente de modo a atingir os objetivos para os quais foram projetados. Estas ações podem ser conseguidas através da partilha de informação entre diferentes agentes sobre o ambiente envolvente.

4.2.2 Aplicação

No que respeita à Aplicação, podemos encontrar as categorias de *middleware* para acesso a dados, *middleware* para *Desktop*, *middleware* baseado na Web, *middleware* para tempo real, e *middleware* específico.

i. Acesso a Dados

O *middleware* de acesso a dados é caracterizado pela interação entre a aplicação e as bases de dados locais e/ou remotas, centros de dados ou outros ficheiros de dados. Este tipo de *middleware* inclui monitorização de transações de dados, acessos às bases de dados e processamento de transações distribuídas, e é ainda capaz de dar suporte a requisitos específicos das bases de dados como segurança e proteção de dados. Como os dados podem estar distribuídos por diversas bases de dados, este *middleware* monitoriza o progresso de cada transação e pode fazer o pedido de "*rollback*" quando a transação não é efetuada com sucesso. Quando a transação é feita corretamente, a aplicação é informada do estado do pedido feito ao *middleware* e recebe os dados requisitados. Existem alguns *middleware* que implementam esta abordagem que podem alterar a apresentação dos dados para se tornar mais compreensível para o utilizador.

ii. Desktop

Este *middleware* permite fazer alterações à forma de apresentação dos dados requeridos pelo utilizador através da monitorização e assistência fornecidos à aplicação, pode também fazer a gestão dos serviços de transporte de dados e providenciar cópias de proteção e outras funções operacionais. Além destas funcionalidades, também estão disponíveis serviços de gestão de gráficos, gestão de ficheiros, eventos de notificação e controlos de acesso, entre outros.

iii. Baseado na Web

O *middleware* baseado na Web está associado à utilização da internet e auxilia o utilizador em tarefas como busca, navegação e permite uma previsão de páginas de interesse através da visualização do histórico do utilizador. Este *middleware* fornece serviços de autenticação para várias aplicações, e processos de comunicação que são independentes do sistema operativo, protocolos de rede e plataformas de *hardware*. Este *middleware* pode ligar-se diretamente às aplicações caso haja ganho na comunicação entre o servidor e o cliente. Alguns dos serviços principais disponibilizados por este *middleware* são associados a serviços de correio eletrónico, serviços de faturação, serviços de gestão de provisões em grande escala, serviços de acesso remoto a dados e acesso remoto a aplicações.

Uma das aplicações mais utilizadas para este *middleware* passa pela mediação na área de comércio eletrónico. Assim, quando ocorre alguma comunicação entre duas ou mais entidades empresariais, o *middleware* permite controlar o acesso à informação das entidades. Além destas funcionalidades, também são fornecidos serviços para concluir com sucesso as operações de negócio como compra e venda de itens.

Uma outra área de aplicação desta abordagem de *middleware* está associada à computação móvel, onde se procura a integração das aplicações distribuídas para acesso aos serviços de correio eletrónico, calendário, contactos, entre outros, de forma segura e sem fios.

iv. Tempo Real

Esta abordagem centra o seu princípio na entrega dos dados em tempo útil. Esse tempo é determinante para a utilização ou rejeição dos dados já que existem aplicações onde a componente temporal desempenha um papel muito importante.

Um exemplo onde este *middleware* está inserido é a área de multimédia onde se lida com dados como imagem, som, linguagem natural, música e vídeo. Como este tipo de dados pode estar em constante mudança, as aplicações que

necessitam destes dados têm de os receber em tempo real para que possam ser corretamente avaliados.

v. Específico

Algumas das abordagens de *middleware* tentam incorporar diversas características dos demais, fornecendo às suas aplicações serviços específicos. Assim sendo, todas essas abordagens que são projetadas para uma determinada aplicação são categorizadas como específicas. Como exemplos podem ser referidos os *middleware* que suportam as aplicações de sistemas médicos.

4.2.3 Requisitos

Sendo que o tema de *AAL* procura integrar as redes de sensores das habitações num sistema autónomo e responsivo às necessidades dos habitantes, os requisitos estão associados aos recursos de *hardware* presentes, à topologia e escalabilidade da rede, à heterogeneidade, à organização da rede, à integração no mundo real, à qualidade de serviços e à segurança [74], [75].

i. Recursos de Hardware

Devido às limitações de energia e recursos associadas aos dispositivos das redes de sensores, é necessária uma maior gestão destes parâmetros. Para tal, o *middleware* deve ter em conta estas limitações e utilizar eficazmente os recursos de processamento e memória dos dispositivos e promover mecanismos que permitam uma comunicação com o menor gasto de potência possível. Além disso, cada nodo da rede de sensores deve ser capaz de obter os dados, processá-los e comunicá-los sem esgotar os seus recursos. Algumas implementações de *middleware* que têm em atenção a utilização de energia permitem desligar o módulo de comunicação para permitir uma maior poupança de energia.

ii. Topologia e Escalabilidade da Rede

A topologia da rede está associada às mudanças frequentes que a rede sofre. Essas mudanças podem ser devido à falha dos dispositivos na rede, devido a interferências, obstáculos e devido à mobilidade. Por este motivo, o *middleware* deve garantir o correto funcionamento em rede independentemente desta alteração dinâmica da topologia de rede, bem como suportar mecanismos de tolerância à falha, configuração e manutenção autónoma dos nodos da rede.

Já a escalabilidade está associada ao crescimento do sistema. Se o sistema e as suas aplicações crescem, então a rede que suporta esse sistema deve também ter a capacidade de crescer sem afetar o desempenho. Assim, o *middleware* deve garantir suporte à expansão da rede e manter a qualidade de serviços prestados.

iii. Heterogeneidade

O *middleware* deve providenciar modelos de programação de baixo nível que permitam fazer a integração entre o *hardware* e as diversas atividades necessárias às aplicações. Deve ainda estabelecer mecanismos que permitam o interface entre os diversos tipos de *hardware* e as redes de comunicação.

iv. Organização da Rede

A alteração dinâmica da rede apresenta também um desafio de organização da rede. Sendo que cada nodo da rede de sensores deve saber a sua localização e a topologia da rede a que pertence, é essencial que haja suporte a estes serviços. Além desta informação, também é essencial numa rede de sensores a descoberta de parâmetros como o tamanho da rede, densidade, pois estes vão afetar a latência, segurança e energia. Assim é necessário que o *middleware* providencie métodos de descoberta de recursos.

v. Integração no Mundo Real

Muitas das aplicações que utilizam os dados provenientes da rede de sensores estão dependentes que estes sejam entregues em tempo real. Assim, o *middleware* deve fornecer serviços em tempo real que permitam ao sistema adaptar-se às mudanças através da entrega de dados em tempo real.

vi. Qualidade dos serviços

A qualidade de serviços pode ser vista de duas perspetivas diferentes: do ponto de vista da aplicação e do ponto de vista da rede. No caso da aplicação, a qualidade dos serviços referem-se a parâmetros específicos para a aplicação como o número de sensores ativos, o seu raio de cobertura e os dados que o nodo mede. Do ponto de vista da rede, a qualidade de serviço refere-se à capacidade de suportar a comunicação da rede de modo a servir as aplicações, utilizando os recursos eficazmente.

Assim, o *middleware* deve ser projetado de modo a ter em conta os parâmetros de capacidade de comunicação, atrasos na entrega de dados e consumo de energia da rede.

vii. Segurança

Devido à crescente utilização das redes de sensores nas habitações para aplicações de prestação de cuidados de saúde e bem-estar, a segurança nestes sistemas deve ser um ponto de elevada importância. Assim, é necessário não só garantir a segurança das redes como também dos dados que são transmitidos. É ainda necessário garantir a disponibilidade do sistema já que este é aplicado em áreas de grande impacto na vida dos utilizadores. Assim o *middleware* deve ter em conta estes cuidados, introduzindo segurança desde as fases iniciais da sua implementação para que sejam cumpridos requisitos como confidencialidade, autenticação, integridade e disponibilidade.

Além dos requisitos mencionados, é também necessário ter em conta as especificidades do *middleware* quando introduzido num contexto de *AAL*. Neste tipo de cenários, o próprio *middleware* poderá ter associado uma camada de contextualização ou ser apenas uma camada típica de *middleware* que lida com os detalhes de baixo nível associados às redes como a comunicação, controlo de concorrência ou gestão de transações [76]. No caso do *middleware* que integra as capacidades de contextualização, essa contextualização envolve a aquisição de informação do contexto, a inferência sobre essa informação e a adaptação do comportamento ao contexto. A Figura 27 apresenta a estrutura genérica deste tipo de *middleware*.

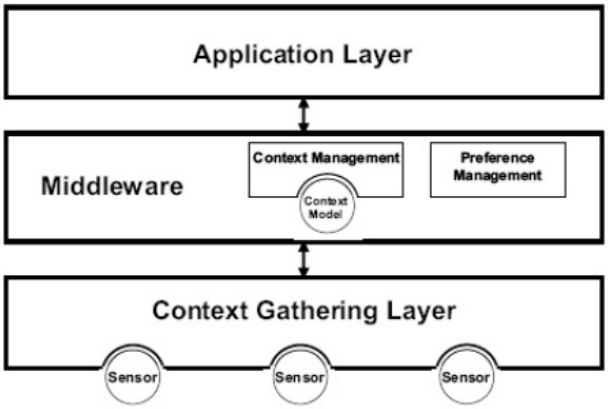


Figura 27 – Arquitetura de middleware context-aware [76]

A opção de incluir uma camada de contextualização no próprio *middleware* permite que os dados sejam diretamente contextualizados, retirando essa tarefa à aplicação e tornando a sua implementação mais simples. Além deste fator, os mecanismos de adaptação do sistema ao ambiente que são desenvolvidos ao nível do *middleware* permitem a sua reutilização por diversas aplicações, ao passo que os mesmos mecanismos desenvolvidos ao nível da aplicação não permitem a reutilização.

4.2.4 Desafios

A introdução de um componente de contextualização no *middleware* trouxe um novo conjunto de desafios. Alguns dos desafios associados a este tipo de *middleware* estão relacionados com a segurança, o compromisso controlo/automatização, e a resolução de conflitos que se podem gerar devido a ambiguidades, contradições e inconsistências lógicas.

i. Compromisso controlo/automatização

De modo a alcançar uma maior autonomia, as aplicações dependem da informação do contexto para adaptarem o seu comportamento ao ambiente e às preferências do utilizador. No entanto, a adaptação do comportamento ao contexto pode nem sempre corresponder ao esperado pelo utilizador, levando a que este sinta que está a perder o controlo sobre a aplicação. Por este motivo é necessário estabelecer um compromisso entre a autonomia e o controlo efetuado pelo utilizador, sendo que nestes sistemas esse compromisso deve ser feito de forma dinâmica. Para tal, as aplicações cientes do contexto necessitam que o *middleware* forneça acesso aos componentes de forma transparente e também forneça suporte à personalização e adaptação baseada no contexto.

ii. Resolução de conflitos

Os recursos monitorizados pelo *middleware* podem ser alterados dinamicamente de acordo com as necessidades apresentadas pelas aplicações. Ao realizar estas alterações, as aplicações podem introduzir situações de ambiguidade, contradições e inconsistências lógicas. Estas situações são consideradas conflitos e devem ser resolvidos pelo *middleware* com recurso a mecanismos de resolução de conflitos. Esses mecanismos devem considerar como requisitos o dinamismo, simplicidade e personalização dessas soluções.

- **Dinamismo:** um conflito nunca deverá ser detetado e resolvido de forma estática devido à natureza do contexto em que se encontra. A análise estática do conflito levaria a um consumo elevado de

recursos para verificar a informação sobre o contexto em que se deu esse conflito;

> **Simplicidade:** o mecanismo de resolução de um conflito deverá ser simples de modo a consumir o mínimo de recursos possíveis;

> **Personalização:** a resolução de um conflito deve ser feita pelo *middleware* da forma automática. No entanto, a aplicação poderá intervir nesta resolução através da escolha de políticas que o *middleware* poderá adotar para resolver o conflito.

A resolução de conflitos poderá ser aplicada a situações que ocorram entre uma aplicação e um dispositivo isolado, sendo considerado pelo *middleware* um conflito local. Também pode ocorrer uma situação de conflito entre diversas aplicações que envolvam um conjunto de dispositivos, sendo que neste caso o *middleware* considera que existe uma situação de conflito distribuído.

4.2.5 Acesso a dados

Além das diferentes categorias de *middleware* mencionadas anteriormente, este também pode ser dividido de acordo com a sua abordagem para acesso a dados. Tomando como objeto de aplicação os sistemas de *AAL* e as redes de sensores existentes nestes sistemas, o *middleware* poderá ser dividido em quatro abordagens principais enunciadas em [77]:

- *database-inspired approaches;*
- *tuple space approaches;*
- *event-based approaches;*
- *service discovery based approaches.*

A primeira abordagem é das mais comuns e tem como princípio efetuar pedidos à rede de dispositivos sensores seguindo o mesmo método adotado para os pedidos efetuados às bases de dados. São exemplos desta abordagem o *COUGAR* [78], SINA [79] e *TinyDB* [80].

A segunda abordagem tem um princípio similar à abordagem anterior, diferindo na organização dos dispositivos. Neste caso é formada uma lista de dispositivos que pode ser acedida através de um número fixo (identificação) que lhe é atribuído. São exemplos desta abordagem o *TinyLIME* [81] e *LINDA* [82].

Seguidamente, a abordagem baseada em eventos tem como base a publicação de eventos por parte do *middleware* aquando da chegada de novos dados dos dispositivos. Estes eventos deverão ser subscritos pela aplicação de modo a obter os respetivos dados. Um exemplo que utiliza esta abordagem é o *Mires* [83].

Por fim, surge a abordagem baseada na descoberta de serviços. Com esta abordagem pretende-se fornecer *API's* de alto nível que interajam com as redes de sensores já implementadas, de modo a adquirir os dados provenientes dos diversos dispositivos afetos a essas redes. São exemplos desta abordagem o *MiLAN* [84] e *OSGi*.

4.2.6 Exemplos de middleware

Como exemplos de abordagens de *middleware* existentes podem ser referidos os *middleware COUGAR* [78], SINA [79], *TinyDB* [80], *TinyLIME* [81], *Mires* [83], *MiLAN* [84], *CORBA* [85], *OSGi* [86], entre outros.

i. COUGAR

O *middleware COUGAR* [78] é uma camada de *software* que opera sobre as bases de dados onde estão guardados os dados provenientes dos sensores existentes na habitação. Para que os dados sejam válidos, é necessário que tenham associados o tempo em que foram recolhidos. Este é um dos problemas que é discutido na abordagem a este *middleware*, sendo apresentada uma solução que envolve a representação dos dados através de um tipo especial de dados: *ADT* (*Abstract Data Type*). Para aquisição dos dados, a interface é feita através de *queries* aos sensores, utilizando as funções representadas por *ADT*.

ii. SINA

O *middleware SINA* [79] pretende possibilitar a execução de comandos e consultas, obter resultados e monitorizar as redes de sensores de uma habitação. A utilização deste *middleware* permite facilitar estes processos já que é executado em cada nodo da rede de sensores. Para tal, a cada nodo de sensores está associado um conjunto de atributos que os caracteriza e define as funcionalidades que estes disponibilizam.

No que respeita à interoperabilidade entre sensores na mesma rede, por se encontrar a correr em cada nodo da rede, este *middleware* permite que sejam feitas consultas entre diferentes nodos. Para isso é utilizado um cabeçalho específico na mensagem distribuída pela rede que identifica o nodo a ser inferido.

A linguagem *SQTL* (*Sensor Query and Tasking Language*) é utilizada para fazer passar as mensagens entre os vários nodos e fazer pedidos de novos dados aos diversos sensores. No entanto, para as aplicações que pretendam utilizar os dados dos sensores é disponibilizada a linguagem *SQL* para interagir com os mesmos. Os pedidos em *SQL* são processados pelo *middleware* e convertidos na linguagem *SQTL* que trata de fazer as inferências aos sensores.

iii. TinyDB

No *middleware TinyDB* [80], os diversos sensores existentes numa rede estão descritos numa tabela e podem apenas apresentar um tipo de atributo por cada instante de tempo.

Para obter os dados dos diversos sensores, são utilizadas *queries* semelhantes às utilizadas nas bases de dados definidas pela linguagem *SQL*. Estas inferências feitas aos estados dos sensores podem ser combinadas de diversas formas de modo a cumprir os requisitos da aplicação. É possível fazer *queries* de modo a agregar dados de diferentes sensores, *queries* com detalhes temporais, baseadas em eventos e ainda baseadas em tempo de vida útil.

Além das combinações referidas, há ainda outras inferências mais específicas que podem ser feitas aos diversos sensores. São elas as *queries* de

monitorização, de exploração do ambiente, de atuação e *queries* que permitem registar eventos mais rápidos que a capacidade de transmissão dos dados.

iv. TinyLIME

O *middleware TinyLIME* [81] foi desenvolvido sobre um outro *middleware* já existente denominado *LIME*. Neste *middleware* existe um conceito de memória partilhada onde os diversos sensores podem armazenar os dados das suas leituras do ambiente.

Para fazer a interação entre as aplicações e os dados adquiridos, são disponibilizadas diversas *API's* (*Application Programming Interface*) aos programadores para adquirir os dados dos sensores.

Neste *middleware* existe uma centralização dos dados, sendo que toda a rede de sensores transmite os valores adquiridos das leituras do ambiente a uma estação central.

v. Mires

O *middleware Mires* [83] foi desenvolvido para uma arquitetura de redes de sensores sem fios (*WSN – Wireless Sensor Network*). Este *middleware* incorpora características de um *middleware* orientado a mensagens, uma vez que permite às aplicações comunicar com a rede de sensores através de métodos de publicação/subscrição de dados. Desta forma, as aplicações apenas recebem os dados referentes aos sensores que pretendem analisar.

A forma de subscrição de mensagens por parte de uma aplicação inicia-se nos nodos da rede de sensores. Estes têm de dar a conhecer o tipo de dados que recolhem a toda a rede. Essa informação há-se chegar a um nodo específico onde as aplicações estão conectadas. É então que se dá a subscrição das mensagens por parte da aplicação e a comunicação fica definida.

vi. MiLAN

A abordagem adotada pelo *middleware MiLAN* (*Middleware Linking Applications and Networks*) [84] tem como base a utilização dos protocolos já existentes e utilizados pelas redes de sensores. Com esta abordagem pretende-se usar os protocolos padronizados que estão associados às redes de sensores e criar uma camada de abstração que permita às aplicações aceder aos dados independentemente do protocolo utilizado. Para tal os comandos executados pelo *MiLAN* são convertidos para os protocolos específicos de cada sensor a ser inferido.

Além desta característica, este *middleware* também tem a capacidade de gerir a rede de sensores e controlar esses mesmos sensores de modo a garantir uma melhor qualidade de serviço na rede, eficiência energética e prolongamento da atividade da aplicação.

vii. CORBA

O *middleware CORBA* (*Common Object Request Broker Architecture*) [85] foi desenvolvido pelo grupo OMG (*Object Management Group*) e é amplamente utilizado na programação de sistemas distribuídos. Este *middleware* baseia a sua arquitetura num paradigma orientado a objetos e assume as operações remotas como sendo um interface. Por este motivo, as instanciações dos interfaces neste *middleware* correspondem a um objeto *CORBA* que será utilizado para comunicação. A interface com os objetos *CORBA* é definida através de uma linguagem de definição de interface (*IDL*).

Este *middleware* foi projetado para dar suporte a diversas linguagens e protocolos, reduzindo assim os problemas de interoperabilidade.

viii. OSGi

O *middleware OSGi* (*Open Service Gateway Initiative*) [86] é um *middleware* desenvolvido com recurso à linguagem *Java* que se baseia numa arquitetura orientada à disponibilização de serviços (*SOA*). Este *middleware* permite integrar, atualizar e desintegrar os diversos serviços (*bundles*) que se mostrem relevantes para o sistema. Além desta característica, estas alterações de serviços no *middleware* podem ser efetuadas sem haver necessidade de reiniciar o sistema.

Estes serviços (*bundles*) podem ser desenvolvidos de modo a dar suporte aos diversos protocolos existentes nos sistemas de automação. Esta modularização permite que o sistema seja altamente configurável e leva a uma maior distribuição dos serviços. Esta abordagem tem vindo a ser amplamente utilizada no contexto em que esta dissertação se insere, sendo o *OSGi* um dos *middleware* mais utilizados [87].

Por fim, destaca-se a utilização de uma variante deste *middleware* (*R-OSGi* [88]) que é utilizado no projeto *PERSONA*. Este poderá vir a ser um bom candidato a ser implementado nos cenários de *AAL*, caso consiga ultrapassar as dificuldades de interoperabilidade, segurança e qualidade dos serviços prestados como é referido em [89].

Após a caracterização deste tema e apresentados os seus requisitos e desafios associados, surge a necessidade de implementar o *middleware* para responder aos desafios criados pelo tema de *AAL*.

4.2.7 Implementação

As diversas funcionalidades disponibilizadas pelo *middleware* têm vindo a ser integradas em diversos sistemas de casas inteligentes e também em contexto de *Ambient Assisted Living*. Esta integração leva a que existam diversos projetos associados a estes temas, sendo que cada projeto utiliza diferentes abordagens ao tema. No entanto, tem vindo a verificar-se uma maior adesão às arquiteturas orientadas a serviços, sendo que as especificações do *middleware OSGi* (*Open Service Gateway Initiave* [90]) são as mais utilizadas nestes sistemas de casas inteligentes e *AAL*.

Dos projetos de *middleware* existentes serão descritas três implementações de código aberto (*open source*), baseadas no *middleware OSGi*. São esses projetos o *openHAB*, *openAAL* e *DOG*.

i. openHAB

O *middleware openHAB* [91] é uma iniciativa de código aberto (*open source*) que pretende dar suporte às aplicações de domótica. Este *middleware* é desenvolvido com recurso à linguagem *Java* e tem por base o *middleware OSGi*. O *middleware openHAB* não contempla uma camada de contextualização pelo que a mesma fica a cargo da aplicação que utiliza os serviços disponibilizados por este *middleware*. Na Figura 28 está representada a arquitetura deste *middleware*, onde é possível verificar a dependência dos serviços disponibilizados pelo *OSGi*.

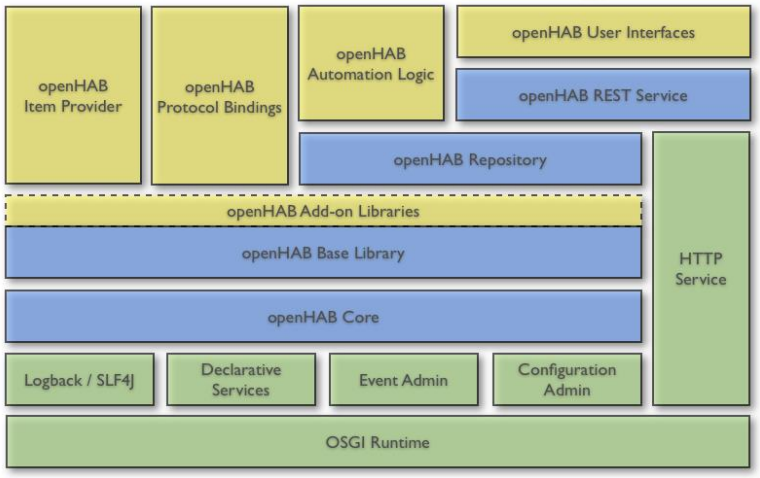


Figura 28 – Arquitetura do middleware openHAB [91]

A comunicação neste *middleware* é feita através de um barramento de eventos (*Event Bus*), sendo este um serviço de extrema importância deste *middleware*. Isto porque os serviços que gerem as especificidades das ligações associadas aos diversos protocolos devem estar diretamente conectados a este barramento de modo a receber e enviar eventos associados aos dispositivos.

Os eventos existentes podem ser de dois tipos: comandos que desencadeiam uma ação ou mudança de estado de algum dispositivo; ou atualizações que alertam para a alteração do estado dos dispositivos. Este serviço é baseado no serviço *EventAdmin* providenciado pelo *middleware OSGi*, cuja abordagem de publicação/subscrição de serviços implementada serve os propósitos da plataforma *openHAB*. Na Figura 29 é ilustrada a forma de comunicação desta plataforma.

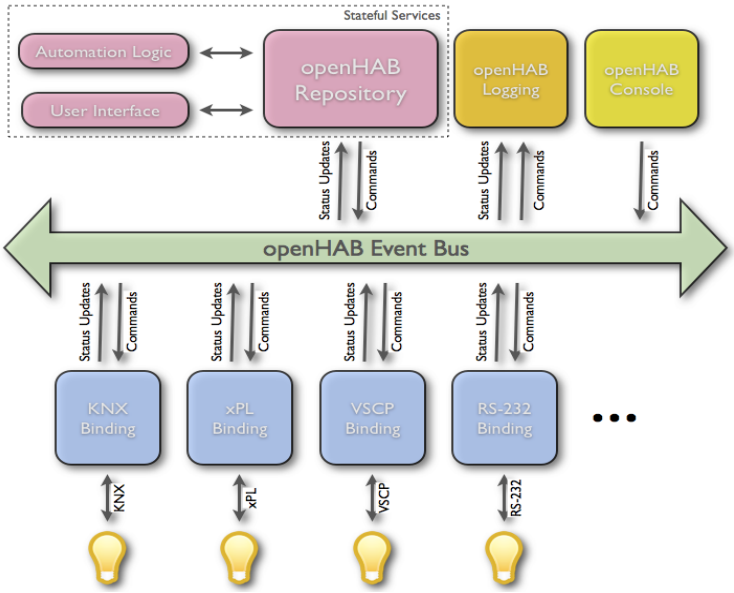


Figura 29 – Diagrama de comunicação no middleware openHAB [91]

É ainda de referenciar que neste *middleware* os diversos dispositivos são descritos como “Item” e lhes são associadas funcionalidades (serviços) que estes podem desempenhar e os estados que podem tomar. Para que seja possível manter o registo dos estados e funcionalidades este *middleware* utiliza um componente denominado repositório de itens (*Item Repository*).

ii. openAAL

O *middleware openAAL* [92], [93] é uma iniciativa de código aberto (*open source*) que tem por base as pesquisas desenvolvidas em diversos projetos, entre os quais o projeto *SOPRANO*. Este *middleware* foi desenvolvido com recurso à *framework OSGi* para permitir a integração e comunicação entre serviços, tendo sido também integrada uma plataforma genérica de serviços para gestão do contexto de modo a recolher e fazer a abstração dos dados sobre o ambiente, especificações do comportamento do sistema e descoberta semântica de serviços.

A arquitetura deste *middleware* pode ser representada pela Figura 30, onde é possível destacar o ambiente *OSGi* que serve de base a este *middleware* e a ontologia para contextualização de dados. Estes primeiros componentes servem como primeira camada para outros três componentes de grande importância neste *middleware*: o gestor de contexto (*context manager*), o gestor de procedimentos (*procedural manager*) e o compositor (*composer*).

Figura 30 – Arquitetura do middleware openAAL [93]

- ***Context Manager*:** este componente permite guardar a informação capturada pelos sensores e dados introduzidos pelo utilizador com recurso às ontologias. Desta forma é possível aplicar algoritmos sobre dados guardados de modo a encontrar situações de interesse para o sistema que possam ser relevantes para a adaptação ao contexto. Este *middleware* suporta também a

inferência sobre o modelo de baixo nível baseado em sensores e o modelo de alto nível baseado nos serviços. Estes modelos constituem a ontologia utilizada pelo *middleware openAAL*, facilitando a integração entre os diversos sensores e os serviços associados ao *AAL*.

- ***Procedural Manager*:** este componente gere e executa diferentes procedimentos que permitem reagir a situações de interesse que ocorram no sistema. Esses procedimentos definem a sequência de ações (*workflow*) necessárias para reagir autonomamente a uma alteração de contexto identificada pelo componente de gestão de contexto. De modo a garantir a independência destes procedimentos quer do ponto de vista do *hardware* disponível, quer do ponto de vista do *software* e serviços disponibilizados, os serviços requeridos pelos procedimentos são definidos de forma abstrata. Esta característica permite separar a deteção das situações de interesse e a concretização dos serviços a executar, proporcionando uma maior configuração de cada uma destas tarefas. A sequência de ações definida para reagir a uma determinada situação deve resultar num conjunto de objetivos a cumprir por parte do sistema para se adaptar a essa situação.
- ***Composer*:** este componente analisa, seleciona e combina os serviços disponíveis de modo a alcançar os objetivos delineados pelo gestor de procedimentos. Esta agregação de serviços resulta num serviço abstrato (apenas existe pela agregação de outros serviços) mas com objetivos concretos. Para que seja feita a ligação entre estes serviços abstratos requisitados e os objetivos concretos para adaptação ao contexto, é utilizada a noção de serviço virtual. Estes serviços virtuais fornecem métodos abstratos que aquando da sua realização conseguem executar os objetivos concretos delineados.

iii. DOG

O *middleware DOG* [60] (*Domotic OSGi Gateway* [94]) foi desenvolvido por equipas de investigação do Politécnico de Turim, e explora as funcionalidades de conexão e computação para estabelecer uma ponte de ligação que permita integrar e coordenar diferentes redes de domótica. Esta abordagem utiliza o *middleware OSGi* de modo a permitir a ativação dinâmica de módulos, integração de novos dispositivos em *runtime* e reação às falhas dos módulos. O *middleware DOG* também integra um modelo ontológico sobre os sistemas de domótica e o ambiente envolvente.

A arquitetura deste *middleware* está representada na Figura 31, onde se pode verificar a existência de quatro divisões principais (anéis) que assentam sobre o *middleware OSGi*. Cada uma destas divisões lida com diferentes tarefas cujos objetivos vão desde as interligações de baixo nível até à modelação e interface de alto nível. Cada uma das divisões integra serviços providenciados pelo *OSGi* que correspondem aos módulos funcionais da plataforma.

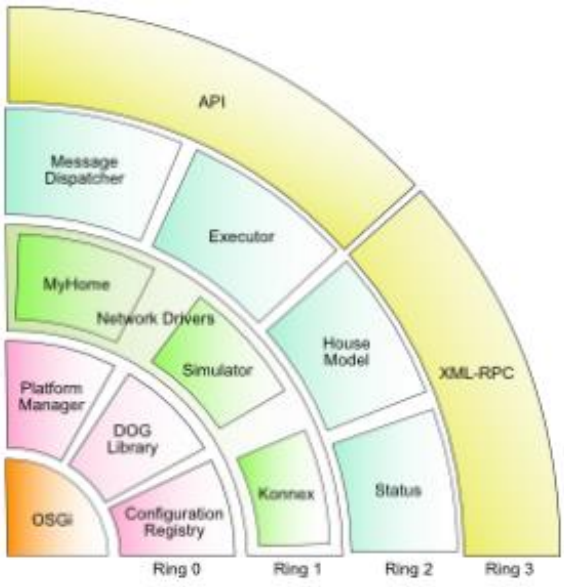


Figura 31 – Arquitetura do middleware DOG [60]

A primeira divisão (***Ring 0***) inclui as bibliotecas comuns ao sistema e os serviços necessários para controlar e gerir as interações entre o *OSGi* e os serviços do *middleware DOG*. É neste nível que podem ser gerados eventos do sistema relacionados com configurações, erros ou falhas que serão posteriormente enviados para a plataforma *DOG*.

- ***DOG Library*:** este serviço é utilizado como repositório de bibliotecas para todos os serviços associados à plataforma *DOG*.
- ***Platform manager*:** este serviço lida com o processo de inicialização do sistema e gere o ciclo de vida dos serviços da plataforma *DOG*, fazendo também a coordenação e gestão dos erros associados a estes.
- ***Configuration Registry*:** este serviço implementa o interface de configuração através da manutenção e exportação dos parâmetros que lhe estão associados.

A segunda divisão (***Ring 1***) engloba os diversos serviços (*bundles*) que fornecem o interface às diversas redes às quais a plataforma *DOG* pode ser ligada. Cada tecnologia de rede é gerida por um *driver* dedicado que permite fazer a abstração das especificidades de comunicação associadas à tecnologia e utilizar uma representação de alto nível para fazer o interface com os dispositivos.

- ***Network drivers*:** de modo a conseguir fazer a interligação entre diferentes interfaces de rede, o *middleware DOG* fornece um conjunto de *drivers* de rede, existindo um para cada tecnologia (*KNX*, *X10*, *Z-Wave*, etc.). Cada *driver* implementa um "configurador" que interage com o modelo ontológico da habitação (*House Model*) de modo a receber a lista de dispositivos que deve gerir e as suas funcionalidades.

A terceira divisão (***Ring 2***) fornece as infraestruturas para encaminhamento de mensagens ao longo da rede de *drivers* e que são dirigidas aos serviços da plataforma *DOG*. Além desta funcionalidade, esta divisão também inclui o modelo ontológico associado ao ambiente de domótica e que é implementado pelo serviço *House Model*.

- ***Message Dispatcher*:** este serviço é responsável pela entrega das mensagens aos destinatários corretos, sejam eles os *drivers* de rede ou outros serviços da plataforma.
- ***Executor*:** este serviço valida os comandos recebidos pelo serviço *API*. Esses comandos são analisados sintaticamente através da verificação da relação entre o tipo da mensagem e o seu conteúdo, sendo também analisados semanticamente através da comparação com os comandos modelados pela ontologia. No caso de as verificações terem um resultado positivo, a mensagem é encaminhada para o *Message Dispatcher*.
- ***Status*:** este serviço mantém um registo dos estados de todos os dispositivos controlados pelo *middleware DOG*, sendo que utiliza as notificações provenientes dos *drivers* de rede para saber esses estados. Estes estados são usados pelo *middleware DOG* para reduzir o tráfego nos barramentos das redes de domótica e para filtrar comandos desnecessários.
- ***House Model*:** o modelo da habitação é considerado o núcleo da inteligência desta plataforma. Este baseia-se num modelo formal de uma habitação definido através da instanciação de uma ontologia (*DogOnt*) associada a este *middleware*. Esta ontologia descreve o ambiente de domótica e os diversos dispositivos e dados associados a estes. São também modeladas as funcionalidades disponibilizadas pelos dispositivos e as possíveis configurações que estes podem suportar. Esta ontologia é utilizada pelo *middleware DOG* para implementar diversas funcionalidades que abrangem a validação de comandos, abstração dos dispositivos e operações com registo de estados (*stateful operation*).

Por fim, a quarta divisão (***Ring 3***) aloja os serviços que permitem o acesso à plataforma *DOG* por parte de aplicações externas, seja através de *APIs* específicos que podem ser usados por aplicações *OSGi*, ou através de métodos *XML-RPC* [95] utilizado por aplicações baseadas noutras tecnologias.

> ***API*:** este serviço permite o acesso por parte das aplicações baseadas em *OSGi* às configurações da habitação, ao envio de comandos para os dispositivos, à receção de eventos, e outros serviços prestados pela plataforma *DOG*.

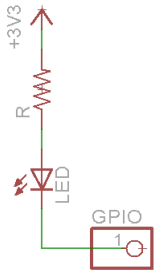
> ***XML-RPC*:** este serviço permite a integração dos serviços disponibilizados pela plataforma *DOG* por parte de aplicações que não são baseadas em *OSGi*.

4.2.8 Resultados

Um dos fatores importantes aquando da integração de uma tecnologia no dia-a-dia das pessoas é sem dúvida o custo que esta representa. No caso da integração das soluções de *middleware* em contexto de casas inteligentes e *AAL*, esse fator deve também ser tido em conta. Assim, as plataformas de desenvolvimento a integrar nestes sistemas deverão ser de baixo custo, sem contudo por em causa a capacidade de dar suporte ao *middleware* e aos diversos componentes do sistema. Tendo em vista este fator, recorreu-se à utilização da plataforma de desenvolvimento *Raspberry Pi* [96] ao longo deste trabalho para testar os *middleware* mencionados anteriormente. Esta é uma plataforma baseada na arquitetura *ARM* caracterizada pelo seu baixo custo e pela possibilidade de integração de módulos que permitem dar suporte às diversas tecnologias associadas às redes de sensores.

Esta plataforma de desenvolvimento foi escolhida não só pelo fator custo, como também pela facilidade de integração e desenvolvimento que esta disponibiliza e pela capacidade de suportar aplicações em linguagem Java. Esta última característica é essencial uma vez que é através desta linguagem que são disponibilizados os serviços do *middleware OSGi* sobre o qual se baseiam as implementações apresentadas.

De modo a testar as implementações mencionadas, foi idealizado um cenário de monitorização de dados relativos à temperatura, luminosidade, estado de um sensor de presença e estado de um sensor de janela afetos a uma divisão. De igual modo foi também idealizado o controlo do sistema de iluminação e do sistema de aquecimento afetos à mesma, sendo estes sistemas simulados através de um conjunto de *LED*. Para tal foi desenvolvido o circuito representado na Figura 32, bem como o respetivo código para alteração do estado do *LED*, simulando assim o comportamento de ligar/desligar dos sistemas mencionados.




```
import RPi.GPIO as GPIO
import sys

def altEstado(canal, estado):
        if(estado == "ON"):
                print "Liga"
                GPIO.output(canal, False)        #Alterar saida
        elif(estado == "OFF"):
                print "Desliga"
                GPIO.output(canal, True)         #Alterar saida

if(len(sys.argv)<3):
        print "Erro: comando incompleto"
        print "Comando esperado: IO.py <canal> <estado>"
else:
        canal = int(sys.argv[1])
        estado = str(sys.argv[2])

        #print canal
        #print estado

        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(canal, GPIO.OUT)      #Configurar como saida
        altEstado(canal, estado)         #alterar o estado

        print "------------Estado alterado----------------"
```

Figura 32 – Circuito e código para simulação de controlo dos sistemas

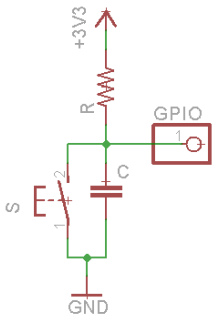
O código apresentado na figura permite alterar o estado dos pinos de saída que estão associados ao controlo dos sistemas de aquecimento e iluminação. Este código foi desenvolvido com recurso à linguagem *Python*, tendo sido utilizada uma biblioteca específica (*RPi.GPIO*) para interação com os diversos pinos existentes na placa de desenvolvimento. Inicialmente cada pino é definido como saída através da função *GPIO.setup(canal, GPIO.OUT)*, sendo depois o estado alterado através da função *GPIO.output(canal, nível)* de acordo com o estado pretendido. Devido à configuração do circuito, o comportamento de ligar é conseguido quando a saída é definida com nível lógico baixo (0V), sendo por sua vez o comportamento de desligar conseguido quando se mantém a saída num nível lógico alto (3.3V).

No que à aquisição de dados diz respeito, o código criado permite o envio dos dados através do protocolo *TPC/IP*, sendo que este protocolo é suportado pelos serviços (*bundles*) integrados nas implementações de *middleware* testadas. Na Figura 33 está representado o código para criação do servidor onde as *bundles* do *middleware* que lidam com este protocolo se vão ligar.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

conn, addr = s.accept()
print 'Connection address:', addr

while True:
        time.sleep(30)
        Temp()
        time.sleep(30)
        Lux()

conn.close()
```

Figura 33 – Servidor TCP/IP para envio de dados

Os sensores de presença e estado da janela foram simulados através de um botão de pressão. Na Figura 34 está representado o circuito desenvolvido para o efeito e o código associado.



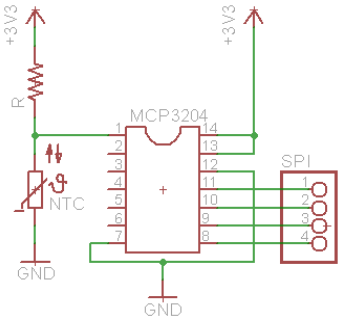

```
import RPi.GPIO as GPIO
import sys

def Janela(canal):
        estado_jan = "erro"
        print('Evento detetado no canal %s'%canal)
        if GPIO.input(18) == GPIO.LOW:
                print('CLOSED')
                estado_jan = "CLOSED"
        elif GPIO.input(18) == GPIO.HIGH:
                print('OPEN')
                estado_jan = "OPEN"
        try:
                conn.send("J " + estado_jan)
        except socket.error as e:
                if e.errno in (errno.EPIPE, errno.E
                        restartConn()

GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.IN)
GPIO.setup(5, GPIO.IN)

GPIO.add_event_detect(18, GPIO.BOTH)
GPIO.add_event_callback(18, Janela)
```

Figura 34 – Circuito e código para simulação dos sensores

Através do código apresentado na imagem é possível detetar o estado dos botões que simulam o comportamento do sensor de presença e o comportamento de estado da janela. Para tal, é utilizada a função *GPIO.setup(canal, GPIO.IN)* para definir os pinos como entrada e a função *GPIO.add_event_detect(canal, GPIO.BOTH)* para ativar a monitorização do

estado dos pinos. Neste cenário pretende-se detetar as transições ascendentes e descendentes, sendo para tal utilizado o argumento *GPIO.BOTH* para ativar a deteção de ambas as transições. Posteriormente é associada uma rotina que será executada aquando da alteração de estado da entrada através da função *GPIO.add_event_callback(canal, rotina)*. Aquando da transição de estado, é verificado o estado atual do pino através da função *GPIO.input(canal)* e o estado correspondente é enviado.

A aquisição de dados sobre temperatura e luminosidade foi conseguida através da utilização de um sensor de luminosidade (*LDR*) e um sensor de temperatura (*NTC*), acoplados a um conversor analógico-digital (*MCP3204*). Na Figura 35 e Figura 36 são apresentados os circuitos e respetivos códigos para aquisição e interpretação dos dados mencionados.




```
import math
from webiopi.devices.analog import MCP3004, MCP3008, MCP3204, MCP3208

def Temp():
        mcp0=MCP3204()
        Vo = mcp0.analogReadVolt(1)
        Resistance = (10000*Vo)/(3.3-Vo)
        Temp = math.log(Resistance)
        Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp))
        Temp = round((Temp - 273.15), 1)
        try:
                conn.send("T " + repr(Temp))
                print Temp
        except socket.error as e:
                if e.errno in (errno.EPIPE, errno.ESHUTDOWN, errno.ECONNABORTED, errno.ECONNREFU
                        restartConn()

while True:
        time.sleep(30)
        Temp()
        time.sleep(30)
        Lux()
```

Figura 35 – Circuito e código para aquisição de temperatura

O circuito apresentado permite adquirir os valores de temperatura na divisão, sendo para tal utilizado o *ADC MCP3204*. O interface com este é feito através do protocolo *SPI*, sendo por este motivo utilizada a biblioteca *webiopi*

onde estão definidas as configurações e rotinas a executar para aquisição dos dados. A função *analogReadVolt(canal)* permite adquirir o valor da tensão elétrica, sendo esse valor utilizado de seguida para obter o valor de resistência do sensor através da equação de divisor de tensão (1):

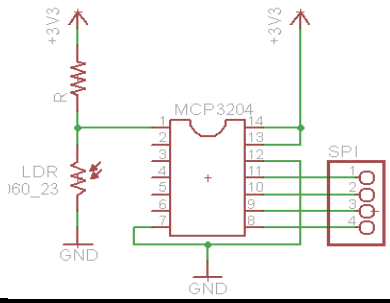
$$Vo = Vi * \frac{R_{\text{ntc}}}{R_{\text{ntc}}+R} \qquad (1)$$

$$R_{\text{ntc}} = \frac{10000*Vo}{3.3-V0} \qquad (2)$$

Este valor é de seguida utilizado na equação de *Steinhart-Hart* (3):

$$\frac{1}{T} = A + B * \ln(R) + C * (\ln(R))^3 \qquad (3)$$

Sendo A, B e C os coeficientes de *Steinhart-Hart*, T a temperatura em graus Kelvin e R a resistência apresentada pelo sensor. É através desta equação que se torna possível relacionar a resistência obtida pelo sensor com a temperatura (em graus Kelvin) detetada na divisão. De seguida a temperatura obtida é convertida em graus Celsius e enviada.




```
import math
from webiopi.devices.analog import MCP3004, MCP3008, MCP3204, MCP3208


def Lux():
        mcp0=MCP3204()
        Vo = mcp0.analogReadVolt(0)
        if Vo == 0:
                Lux = 0
        else:
                Lux = 5*(10*Vo+33)/Vo
        Lux = round(Lux,1)
        try:
                conn.send("L " + repr(Lux))
                print Lux
        except socket.error as e:
                if e.errno in (errno.EPIPE, errno.ESHUTDOWN, errno.ECONNABO
                        restartConn()

while True:
        time.sleep(30)
        Temp()
        time.sleep(30)
        Lux()
```

Figura 36 – Circuito e código para aquisição de luminosidade

O circuito que permite adquirir os valores de luminosidade presentes na divisão (Figura 36) é semelhante ao anterior, sendo que para se obter o valor da luminosidade (em lux) é necessário saber o coeficiente de variação Ω/lx. O *LDR* utilizado apresentava um coeficiente de 5kΩ/lx. Após a aquisição dos dados através da função *analogReadVolt(canal)*, é calculada a luminosidade de acordo com o valor da resistência obtida pelo sensor e os dados serão enviados.

Apresentados os circuitos que permitem simular alguns sensores e atuadores numa habitação, serão apresentados de seguida os resultados obtidos com as implementações de *middleware*.

i. openHAB

De modo a conseguir utilizar este *middleware*, é necessário fazer primeiramente algumas configurações no sistema. Essas configurações passam pela descrição da habitação (*Casa.items*) no que respeita a divisões, dispositivos, protocolos e serviços disponibilizados, e também definição de regras (*Casa.rules*). Com o propósito de demonstrar o funcionamento do *middleware*, foi idealizado o cenário de controlo do sistema de iluminação e do sistema de aquecimento numa divisão, como já mencionado anteriormente. Assim, procedeu-se à descrição do ambiente como está representado pela Figura 37.

Casa.items

File Edit Search Options Help

```
z/* Divisões */
Group   Quartos

Group   Q_Principal     "Quarto Principal"      <bedroom>       (Quartos)
Group   Q_Hospedes      "Quarto Hospedes"       <bedroom>       (Quartos)
Group   Q_Criancas      "Quarto Crianças"       <boy1>          (Quartos)

/* Grupos */
Group Estados
Group:Switch:OR(ON, OFF)                Luzes                   "Todas as luzes [(%d)]"                         (Estados)
Group:Contact:OR(OPEN, CLOSED)          Janelas                 "Janelas abertas [(%d)]"        <contact>       (Estados)
Group:Number:AVG                        Temperatura             "Temperatura média [%.1f °C]"   <temperature>   (Estados)

/* Luzes */
Switch  Luz_QPrinc_Tecto        "Tecto"                 (Q_Principal, Luzes)
Dimmer  Luz_QPrinc_MC1          "Mesa cabeceira 1"      (Q_Principal, Luzes)
Dimmer  Luz_QPrinc_MC2          "Mesa cabeceira 2"      (Q_Principal, Luzes)
Switch  Luz_QH_Tecto            "Tecto"                 (Q_Hospedes, Luzes)
Dimmer  Luz_QH_MC               "Mesa cabeceira"        (Q_Hospedes, Luzes)
Switch  Luz_QC_Tecto            "Tecto"                 (Q_Criancas, Luzes)
Dimmer  Luz_QC_MC               "Mesa cabeceira"        (Q_Criancas, Luzes)

/* Janelas */
Contact Janela_QPrinc           "Janela quarto [MAP(en.map):%s]"        (Q_Principal, Janelas)
Contact Janela_QH               "Janela quarto [MAP(en.map):%s]"        (Q_Hospedes, Janelas)
Contact Janela_QC               "Janela quarto [MAP(en.map):%s]"        (Q_Criancas, Janelas)

/* Temperatura */

Number Temp_QPrinc              "Temperatura [%.1f °C]" (Q_Principal, Temperatura)      <temperature>
Number Temp_QH                  "Temperatura [%.1f °C]" (Q_Hospedes, Temperatura)       <temperature>
Number Temp_QC                  "Temperatura [%.1f °C]" (Q_Criancas, Temperatura)       <temperature>
```

Figura 37 – openHAB - Descrição do ambiente

Seguidamente é necessário especificar o serviço que dá suporte ao protocolo de comunicação que cada um dos dispositivos utiliza para troca de informação com o *middleware*. É através destes serviços que se torna possível o envio de comandos e receção de atualizações de estado dos dispositivos com diferentes protocolos (*KNX*, *X10*, *Z-Wave*, *Bluetooth*, etc.).

A especificação dos serviços a utilizar é feita com recurso ao ficheiro de configuração do *middleware* (*openHAB.cfg*) e o ficheiro de descrição já mencionado (*Casa.items*). Neste exemplo de implementação foi utilizado um serviço específico (*ExecBinding*) que permite ao *middleware* utilizar a linha de comandos do sistema operativo para executar diretamente comandos. Esta opção foi tomada de modo a conseguir simular através de um conjunto de *LED*, o efeito pretendido pela tarefa de ligar/desligar as luzes da habitação. Essa especificação está representada pela Figura 38.

```
/* Luzes */
Switch  Luz_QPrinc_Tecto        "Tecto"                 (Q_Principal, Luzes)    {exec=">[ON:sudo python /home/pi/Desktop/Liga.py]"}
```

Figura 38 – openHAB - Serviço ExecBinding

Com recurso a esta configuração, é então possível executar um programa desenvolvido em linguagem *Python* que altera o estado de saída dos pinos da placa de desenvolvimento, simulando assim a ação requerida. De referir que este serviço, contrariamente aos outros disponibilizados por este *middleware*, só precisa ser configurado no ficheiro de descrição (*Casa.items*).

Para que fosse possível integrar neste *middleware* os dados dos sensores de temperatura, luminosidade, de estado de janela e de presença, foi necessário recorrer ao serviço *"TCP/UDP Binding"* de modo a utilizar o protocolo *TCP/IP* para comunicação dos dados. A configuração deste serviço é feita no ficheiro de configurações (*openhab.cfg*) e está representada na Figura 39.

```
tcp:refreshinterval = 250
tcp:buffersize=1024
tcp:port=5005
tcp:retryinterval=5
tcp:timeout=3000
tcp:itemsharedconnections=true
tcp:adressmask=true
tcp:updatewithresponse=true
```

Figura 39 – openHAB - Configuração TCP/UDP Binding

Além desta configuração, é ainda necessário definir no ficheiro de descrição (*Casa.items*) qual o item que irá receber os dados, bem como o endereço de *IP* e a porta esperados para receção desses dados. Neste cenário de teste, foi definido o endereço de *loopback* (*127.0.0.1*) e a porta 5005 para receção dos dados, como pode ser visualizado na Figura 40.

```
String data     "[%s]"      {tcp="<[127.0.0.1:5005]"}
```

Figura 40 – openHAB - Configuração TCP Binding

Para permitir a automatização de algumas tarefas, esta solução permite a criação de um ficheiro com regras (*Casa.rules*) que é verificado a cada alteração do estado dos componentes do sistema. Deste modo é possível definir comportamentos autónomos que reagem à alteração do estado do ambiente envolvente. Assim, definiu-se uma regra que permite executar ações aquando da alteração do item "*data*" que recebe os dados relativos aos sensores mencionados anteriormente. Essas ações estão dependentes dos dados recebidos e estão representadas na Figura 41.

```
rule    "Novos Dados"
    when
        Item data changed
    then
        buffer = data.state.toString.split(" ")

        if(buffer.get(0) == "L") {
            Lux = new Double(buffer.get(1))
            postUpdate(Lux_QPrinc, Lux)
        }
        else if(buffer.get(0) == "T") {
            Temp = new Double(buffer.get(1))
            postUpdate(Temp_QPrinc, Temp)
        }
        else if(buffer.get(0) == "P") {
            if(buffer.get(1).contains("ON")) postUpdate(Pres_QPrinc, ON)
            else if(buffer.get(1).contains("OFF")) postUpdate(Pres_QPrinc, OFF)
        }
        else if(buffer.get(0) == "J") {
            if(buffer.get(1).contains("OPEN")) postUpdate(Janela_QPrinc, OPEN)
            else if(buffer.get(1).contains("CLOSED")) postUpdate(Janela_QPrinc, CLOSED)
        }
end
```

Figura 41 – openHAB - Regra para receção de dados

Após a definição desta regra e das respetivas ações a desempenhar de acordo com os dados recebidos, tornou-se possível atualizar o sistema com esses dados. Na Figura 42 está representado o resultado obtido com a utilização desta regra, que se caracteriza pela atualização dos valores de temperatura,

luminosidade, e dos estados de presença e estado da janela, conforme idealizado.

```
16:15:30.301 INFO  runtime.busevents[:46] - data state updated to T 18.5
16:15:32.562 INFO  runtime.busevents[:46] - Temp_QPrinc state updated to 18.5

16:17:13.619 INFO  runtime.busevents[:46] - data state updated to J CLOSED
16:17:16.714 INFO  runtime.busevents[:46] - Janela_QPrinc state updated to CLOSE
D
16:17:17.490 INFO  runtime.busevents[:46] - data state updated to J OPEN
16:17:18.358 INFO  runtime.busevents[:46] - Janela_QPrinc state updated to OPEN

16:16:00.353 INFO  runtime.busevents[:46] - data state updated to L 100.1
16:16:00.610 INFO  o.o.model.script.Casa.rules[:73] - Lux updated to 100.1

16:18:00.691 INFO  runtime.busevents[:46] - data state updated to P ON
16:18:02.016 INFO  runtime.busevents[:46] - data state updated to P OFF
```

Figura 42 – openHAB - Atualização de estados

Outro exemplo da utilização destas regras é apresentado na Figura 43.

Casa.rules

File Edit Search Options Help

```
import org.openhab.core.library.types.*
import org.openhab.core.persistence.*
import org.openhab.model.script.actions.*

var Timer temporizador

rule "Desligar luzes automaticamente"
        when
                Sensor_Ocup_QPrinc changed from ON to OFF
        then
                if(Sensor_Ocup_QPrinc.state==OFF) {
                        temporizador=createTimer(now.plusSeconds(30)) [|sendCommand(Luz_QPrinc_Tecto, OFF]
                }
                else {
                        if(timer!=null) {
                                timer.cancel
                                timer=null
                        }
                }
end
```

Figura 43 – openHAB - Exemplo de regra

Neste exemplo pretende-se atuar sobre as luzes associadas ao quarto quando o sensor de ocupação deteta uma transição do estado ocupado (*ON*) para o estado desocupado (*OFF*). Contudo, a alteração do estado da iluminação não deve ser imediata já que pode originar situações em que a luz é desligada apesar do utilizador estar no quarto devido a uma leitura errada feita pelo sensor. Para evitar esta situação, é utilizada a funcionalidade de temporização que permite definir um intervalo de tempo a aguardar (trinta segundos neste exemplo) até ao envio do comando para desligar a luz.

Por fim, é ainda de referir que o interface entre o utilizador e este *middleware* é feito através de *APIs REST* (*Representational State Transfer*), sendo que o *middleware* é utilizado como servidor para resposta aos pedidos enviados pela aplicação (cliente). Através desta aplicação é possível obter o estado dos diferentes dispositivos bem como alterar esses estados remotamente. A Figura 44 apresenta o caso de teste idealizado para prova de conceito onde é alterado o estado da iluminação do quarto principal.

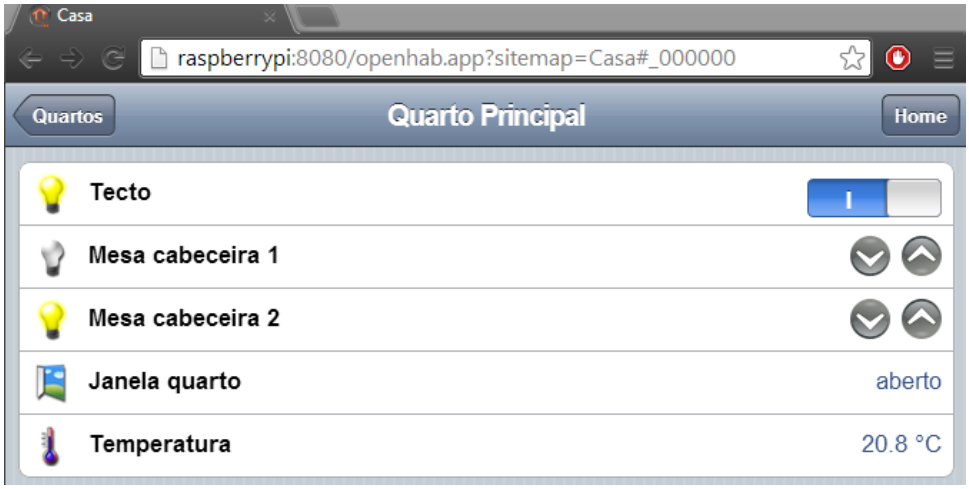


Figura 44 – openHAB - Aplicação

Quando o utilizador altera o estado da iluminação, o *middleware* recebe um novo evento no barramento de eventos (*Event Bus*) com o dispositivo a ser atuado e o comando a executar. De seguida, é utilizado o serviço associado a esse dispositivo para fazer a comunicação com o mesmo. Esta interação pode ser visualizada na Figura 45.

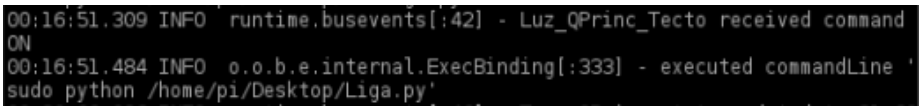


Figura 45 – openHAB - Ligar luz

ii. openAAL

À semelhança do *middleware* anterior, é também necessário executar algumas configurações prévias neste *middleware*, sendo que estas se referem à declaração dos dispositivos existentes na habitação, dos seus atributos e protocolos de comunicação utilizados. Estas configurações são feitas através da utilização de comandos em linguagem *perl*, uma vez que a aplicação *web* que interage com os diferentes componentes da habitação é também implementada com recurso a esta linguagem.

Para efeitos de prova de conceito, foi idealizado novamente o cenário de controlo da iluminação de um quarto. Assim, procedeu-se à descrição do componente de iluminação que se pretendia utilizar através do comando:

define Luz_QPrinc_Tecto dummy

Desta forma foi introduzido no sistema um componente com o nome ***Luz_QPrinc_Tecto*** do tipo ***dummy***, sendo este tipo apenas utilizado para efeitos de programação e teste do sistema. Seguidamente, foi necessário definir os atributos associados a este componente, sendo para tal utilizados os comandos:

attr Luz_QPrinc_Tecto alias Luz Tecto

attr Luz_QPrinc_Tecto webCmd on: off

O primeiro comando é utilizado para atribuir ao componente uma descrição mais compreensível para o utilizador e de modo a facilitar a tarefa de localização do mesmo na aplicação. Já o segundo comando associa duas funcionalidades ao componente declarado, sendo elas a função de ligar (***on***) e desligar (***off***). Através destas funcionalidades será possível alterar o estado da iluminação, quer através da interação efetuada pelo utilizador, quer através dos comportamentos/regras definidos pelo sistema.

Após esta configuração inicial é possível verificar a integração do componente no sistema, como está representado na Figura 46.

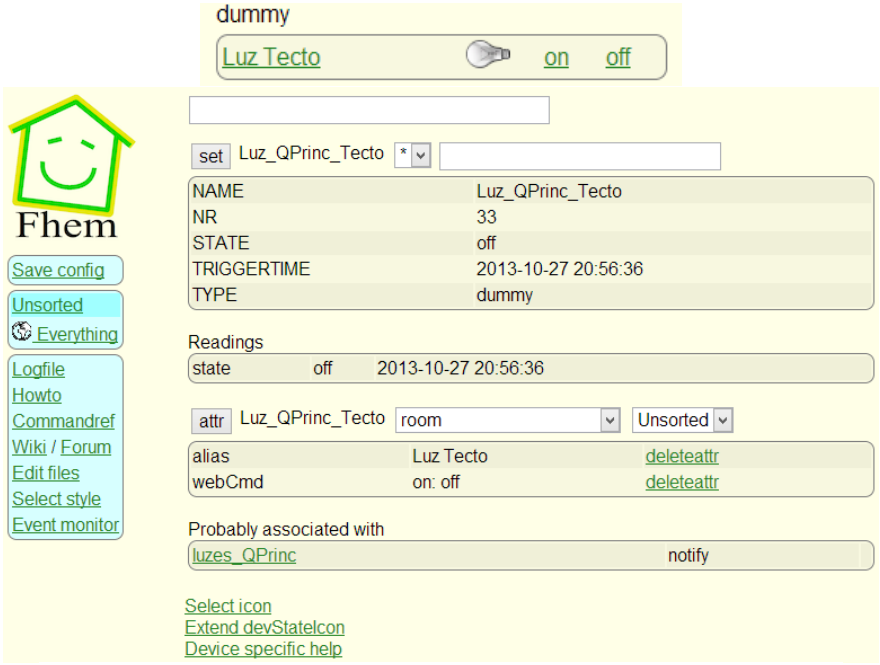


Figura 46 – openAAL - Novo componente

De notar que nesta figura é ainda apresentada a informação de uma associação deste componente a um evento de notificação (***luzes_QPrinc***). Esta associação surge da necessidade de informar o sistema e enviar os comandos necessários aquando da mudança de estado do componente definido. Para definir este evento foi utilizado o comando:

define luzes_QPrinc notify Luz_QPrinc_Tecto "sudo python /home/pi/Desktop/luz.py %"

Este comando permite definir o comportamento executado aquando da alteração de estado do componente ***Luz_QPrinc_Tecto***, sendo que neste caso de teste se pretende simular a tarefa de ligar/desligar a luz de um quarto através da alteração de estado de um conjunto de LED. Este objetivo é conseguido com recurso ao serviço disponibilizado por este *middleware* que permite aceder à linha de comandos do sistema para executar os comandos necessários ao desempenho da tarefa. Neste exemplo foi implementada uma aplicação em linguagem *Python* para alterar o estado da saída onde está ligado o conjunto de LED de acordo com o comando recebido do *middleware*.

Todas as definições dos componentes e respetivos eventos são guardadas num ficheiro de configuração do sistema (*fhem.cfg*), como pode ser visualizado na Figura 47.

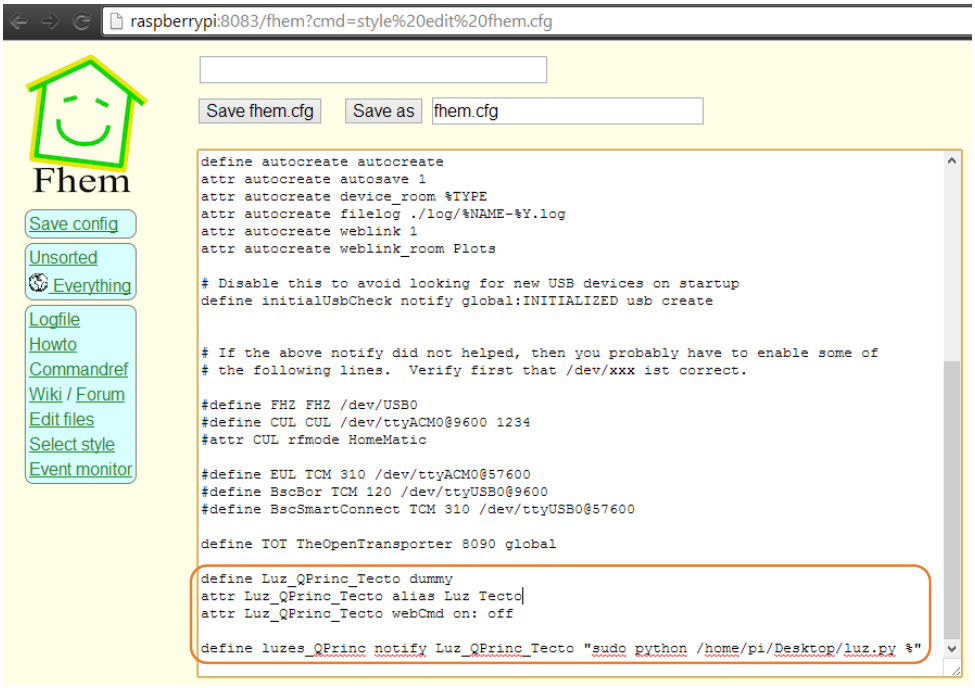


Figura 47 – openAAL - Ficheiro de configuração (fhem.cfg)

Além destas funcionalidades, este *middleware* fornece ainda a possibilidade de integrar um componente de gestão de contexto que permite definir os diversos serviços de *Ambient Assisted Living*. Este componente de gestão de contexto é suportado por uma ontologia definida para o sistema. Na Figura 48 é apresentado o interface que permite a integração e gestão dos serviços de *Ambient Assisted Living*, bem como a gestão da plataforma e dos componentes existentes na habitação.

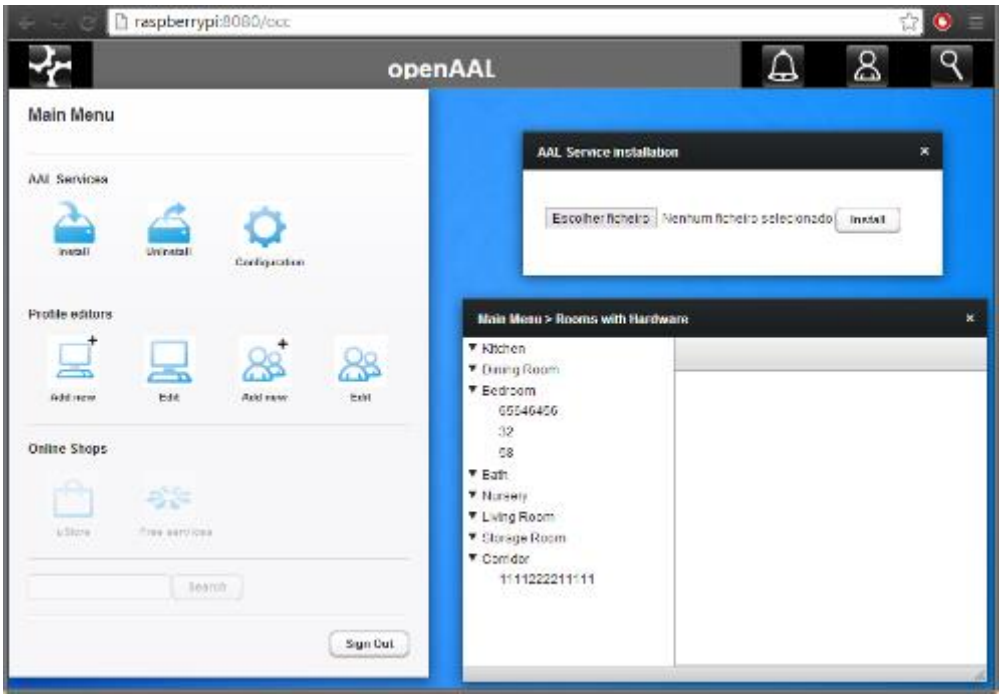


Figura 48 – openAAL - Serviços AAL

4.3 Sumário

Ao longo deste capítulo foi apresentada a relação existente entre o tema de *Ambient Assisted Living* e o *middleware*. Foram também estudadas com mais detalhe as diferentes abordagens de *middleware*, os requisitos que lhe são característicos, os desafios associados e também algumas das implementações existentes. Por fim as implementações mencionadas foram integradas numa plataforma de desenvolvimento de modo a comprovar o seu funcionamento e provar o conceito associado.

Capítulo 5 – Conclusões

Ao longo desta dissertação foi abordado o tema de *Ambient Assisted Living*, tendo sido apresentado o seu propósito, os desafios que se lhe apresentam e as abordagens e aplicações já desenvolvidas nesta área. Este tema é considerado de grande importância já que poderá dar as respostas necessárias aos problemas que advêm do envelhecimento da sociedade. Para tal, o tema de *AAL* tem associado um vasto conjunto de conceitos e áreas de investigação que, apesar de lhe conferirem uma maior complexidade, permitem visar os diversos problemas que lhe estão associados e implementar soluções mais completas, mais adaptáveis e por isso de maior abrangência. Alguns desses conceitos e áreas de investigação foram também explorados nesta dissertação como os conceitos de ontologias e *middleware* ou as áreas de investigação em inteligência artificial e *Ambient Intelligence*.

A integração das áreas de investigação de inteligência artificial e *Ambient Intelligence* nos sistemas de *AAL* surgiu como resposta à necessidade de compreensão do ambiente em que este tipo de sistemas se encontra a atuar. Esta é uma característica relevante nestes sistemas já que é necessário que o sistema compreenda as rotinas, tarefas e preferências dos utilizadores para que possa atuar de acordo com as mesmas. Assim, estas duas áreas de investigação permitem compreender essas rotinas e preferências através da contextualização dos dados que são adquiridos pelos diversos dispositivos sensores existentes na habitação.

A contextualização dos dados está dependente de uma representação do conhecimento associado ao ambiente e que nesta dissertação foi atingida recorrendo à modelação ontológica. O modelo ontológico desenvolvido permitiu descrever os diversos conceitos que são representativos de uma habitação e das possíveis funcionalidades associadas à mesma, além de permitir também estabelecer as diversas relações existentes entre esses conceitos. Este modelo permite ainda a introdução de mais conceitos e relações de modo a expandir o conhecimento sobre a habitação, o que poderá vir a ser vantajoso para dar suporte a mais funcionalidades que se pretendam integrar na habitação. Por sua

vez, a implementação do modelo ontológico criado permitiu compreender o funcionamento das aplicações cientes do contexto que fazem uso deste tipo de modelos.

No que ao *middleware* diz respeito, a sua integração nos sistemas de *AAL* está relacionada com a necessidade de lidar com os diferentes dispositivos e protocolos de comunicação utilizados nas habitações. Esta diversidade de tecnologias e protocolos de comunicação levantam problemas de interoperabilidade que podem ser resolvidos recorrendo às funcionalidades disponibilizadas pelo *middleware*. Assim, as soluções de *middleware* providenciam métodos eficazes de abstração que passam pela disponibilização de serviços, de modo a que a implementação das aplicações que utilizam os dados dos dispositivos não necessite ter em conta as especificidades de comunicação com os diversos dispositivos, focando-se apenas na disponibilização das funcionalidades requeridas. O *middleware* pode ser visto como uma camada de abstração que permite fazer a comunicação entre dispositivos e que pode ter associado a si um componente de contextualização que permite disponibilizar serviços de forma mais elaborada e "inteligente". Nesta dissertação foram abordadas estas duas variantes de *middleware*, sendo que a primeira leva a uma maior complexidade de implementação da aplicação que utiliza o *middleware* pois necessita da contextualização ao nível da aplicação. Já no caso da segunda abordagem, a implementação da aplicação que utiliza o *middleware* tornou-se menos complexa já que pode recorrer aos serviços contextualizados disponibilizados pelo *middleware*.

Ainda sobre o conceito de *middleware*, foi encontrada uma grande necessidade de configuração deste componente dos sistemas de *AAL* para o seu correto funcionamento. Estas necessidades de configuração variam de acordo com o ambiente onde o sistema é implementado, criando assim a necessidade de possuir conhecimentos técnicos sobre estes sistemas para que se possa proceder à sua correta implementação. Tendo em vista o ambiente em que estas soluções serão implementadas, esta necessidade de configuração leva a que seja mais complexo dar suporte às soluções *COTS* (*Commercial Off-The-Shelf*), uma vez que será certamente mais difícil fazer com que a população mais idosa, que vai beneficiar deste sistema, possa fazer as configurações necessárias.

5.1 Trabalho futuro

Como trabalho futuro pode ser apontada a procura de soluções que permitam a integração de novos dispositivos e funcionalidades nas habitações sem que seja necessário o utilizador proceder a configurações. Essas soluções poderão passar pela introdução de mecanismos nos próprios dispositivos que permitam aos sistemas reconhecer os serviços por eles disponibilizados e fazerem uma configuração automática do sistema de modo a suportar esses mesmos dispositivos.

Outro ponto que deverá ser tido em conta para trabalho futuro está relacionado com a modelação das funcionalidades a suportar. Conforme foi exposto ao longo da dissertação, a modelação das funcionalidades disponibilizadas é também de grande importância para os sistemas de *AAL*, levando assim a uma necessidade de integrar mecanismos de modelação na própria base de conhecimento que permitam a assimilação das novas funcionalidades de forma automática. Este conceito poderá ser integrado nestes sistemas através da exploração do conceito de inteligência artificial e os respetivos avanços que tem vindo a ser obtidos na área de aprendizagem autónoma dos sistemas computacionais.

Referências

[1] M. Chan, E. Campo, D. Estève, and J.-Y. Fourniols, “Smart homes - current features and future perspectives.,” *Maturitas*, vol. 64, no. 2, pp. 90–7, Oct. 2009.

[2] L. C. De Silva, C. Morikawa, and I. M. Petra, “State of the art of smart homes,” *Eng. Appl. Artif. Intell.*, vol. 25, no. 7, pp. 1313–1321, Oct. 2012.

[3] I. Akyildiz, “A survey on sensor networks,” *IEEE Commun. Mag.*, pp. 102–114, 2002.

[4] C. Chong and S. Kumar, “Sensor networks: evolution, opportunities, and challenges,” *Proc. IEEE*, vol. 91, no. 8, 2003.

[5] V. Ricquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, “The Smart Home Concept : our immediate future,” *2006 1ST IEEE Int. Conf. E-Learning Ind. Electron.*, pp. 23–28, Dec. 2006.

[6] J. Demongeot, G. Virone, F. Duchêne, G. Benchetrit, T. Hervé, N. Noury, and V. Rialle, “Multi-sensors acquisition, data fusion, knowledge mining and alarm triggering in health smart homes for elderly people.,” *C. R. Biol.*, vol. 325, no. 6, pp. 673–82, Jun. 2002.

[7] U. Nations, “World Population Prospects: The 2010 Revision, Highlights and Advance Tables.,” *United Nations, Dep. Econ. Soc. Aff. Popul. Div.*, 2011.

[8] M. Becker, M. Floeck, and T. Fuhrmann, “Ambient Assisted Living Systems–Notes on a Plenary Discussion,” *Constr. Ambient Intell.*, pp. 194–196, 2008.

[9] T. Kleinberger, M. Becker, E. Ras, and A. Holzinger, “Ambient Intelligence in Assisted Living : Enable Elderly People to Handle Future Interfaces,”

Access, vol. 4555, no. Universal Access in Human-Computer Interaction. Ambient Interaction, pp. 103–112, 2007.

[10] H. Sun, V. De Florio, N. Gui, and C. Blondia, “Promises and Challenges of Ambient Assisted Living Systems,” *2009 Sixth Int. Conf. Inf. Technol. New Gener.*, pp. 1201–1207, 2009.

[11] H. He, “What is service-oriented architecture,” *Publicação eletrônica em*, pp. 1–5, 2003.

[12] A. Hristova, A. M. Bernardos, and J. R. Casar, “Context-aware services for ambient assisted living: A case-study,” *2008 First Int. Symp. Appl. Sci. Biomed. Commun. Technol.*, pp. 1–5, Oct. 2008.

[13] B. Pogorelc, R.-D. Vatavu, A. Lugmayr, B. Stockleben, T. Risse, J. Kaario, E. C. Lomonaco, and M. Gams, “Semantic ambient media: From ambient advertising to ambient-assisted living,” *Multimed. Tools Appl.*, vol. 58, no. 2, pp. 399–425, 2012.

[14] S. Helal and C. Chen, “The Gator Tech Smart House : Enabling Technologies and Lessons Learned,” *Heal. San Fr.*, vol. 5, pp. 1–4, 2009.

[15] J. King and E. Jansen, “The Gator Tech Smart House: A Programmable Pervasive Space,” 2005.

[16] M. Brinkmann, M. Floeck, and L. Litz, “Concept and Design of an AAL home monitoring system based on a personal computerized assistive unit,” *Constr. Ambient Intell.*, pp. 218–227, 2008.

[17] M. Floeck and L. Litz, “Activity- and Inactivity-Based Approaches to Analyze an Assisted Living Environment,” *2008 Second Int. Conf. Emerg. Secur. Information, Syst. Technol.*, pp. 311–316, Aug. 2008.

[18] a Fleury, N. Noury, M. Vacher, H. Glasson, and J. F. Seri, “Sound and speech detection and classification in a Health Smart Home.,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2008, pp. 4644–7, Jan. 2008.

[19] N. Noury and G. Virone, “The Health Integrated Smart Home Information System (HIS2): rules based localization of a human,” *Proc. IEEE–MMB 2002*, IEEE, Madison, USA, 2002.

[20] T. Tamura, T. Togawa, M. Ogawa, and M. Yoda, “Fully automated health monitoring system in the home.,” *Med. Eng. Phys.*, vol. 20, no. 8, pp. 573–9, Nov. 1998.

[21] T. Tamura, A. Kawarada, M. Nambu, A. Tsukada, K. Sasaki, and K.-I. Yamakoshi, “E-healthcare at an experimental welfare techno house in Japan.,” *Open Med. Inform. J.*, vol. 1, pp. 1–7, Jan. 2007.

[22] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, “MavHome: an agent-based smart home,” in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications 2003 PerCom 2003*, 2003, no. January, pp. 521–524.

[23] S. Das and D. Cook, “Designing smart environments: A paradigm based on learning and prediction,” *Pattern Recognit. Mach. Intell.*, pp. 80–90, 2005.

[24] P. Rashidi and D. Cook, “Keeping the resident in the loop: Adapting the smart home to the user,” *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, vol. 39, no. 5, pp. 949–959, 2009.

[25] D. J. Cook, M. Schmitter-Edgecombe, A. Crandall, C. Sanders, and B. Thomas, “Collecting and disseminating smart home sensor data in the CASAS project,” *Proc. CHI Work. Dev. Shar. Home Behav. Datasets to Adv. HCI Ubiquitous Comput. Res.*, 2009.

[26] P. Rashidi and D. J. Cook, “Activity knowledge transfer in smart environments,” *Pervasive Mob. Comput.*, vol. 7, no. 3, pp. 331–343, Jun. 2011.

[27] Q. Wang, W. Shin, X. Liu, Z. Zeng, C. Oh, B. K. AlShebli, M. Caccamo, C. a. Gunter, E. Gunter, J. Hou, K. Karahalios, and L. Sha, “I-Living: An Open System Architecture for Assisted Living,” *2006 IEEE Int. Conf. Syst. Man Cybern.*, pp. 4268–4275, 2006.

[28] J. a. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd, “The Georgia Tech aware home,” *Proceeding twenty-sixth Annu. CHI Conf. Ext. Abstr. Hum. factors Comput. Syst. - CHI ’08*, p. 3675, 2008.

[29] C. Kidd, R. Orr, and G. Abowd, “The aware home: A living laboratory for ubiquitous computing research,” *Proc. 2nd Int’,l Work. Coop. Build.*, vol. 1670, pp. 191–198, 1999.

[30] “Amigo project.” [Online]. Available: http://www.hitech-projects.com/euprojects/amigo/.

[31] T. Danielsen and U. Pankoke-Babatz, “The AMIGO Project: advanced group communication model for computer-based communications environment,” *Proc. 1986 ACM Conf. Comput. Coop. Work*, pp. 115–142, 1986.

[32] J. Schmalenstroeer, V. Leutnant, and R. Haeb-Umbach, “Amigo context management service with applications in ambient communication scenarios,” *Commun. Comput. Inf. Sci. Constr. Ambient Intell.*, pp. 397–402, 2008.

[33] “Matilda Smart House.” [Online]. Available: http://www.icta.ufl.edu/gt.htm#2.

[34] “CORBA (Standard OMG).” [Online]. Available: http://www.corba.org/.

[35] “Protocolo Jabber/XMPP.” [Online]. Available: http://www.ejabberd.im/.

[36] G. Abowd, A. Bobick, and I. Essa, “The aware home: A living laboratory for technologies for successful aging,” *Proc. AAAI Work. Autom. as a Care Giver*, 2002.

[37] D. Cook and W. Song, “Ambient intelligence and wearable computing: Sensors on the body, in the home, and beyond,” *J. Ambient Intell. Smart Environ.*, vol. 1, no. 2, pp. 83–86, 2009.

[38] C. Ramos, “Ambient intelligence–A state of the art from artificial intelligence perspective,” *Prog. Artif. Intell.*, pp. 285–295, 2007.

[39] “ISTAG.” [Online]. Available: http://cordis.europa.eu/fp7/ict/istag/home_en.html.

[40] I. Veronica, G. Mirela, and B. Maria, “Modern Approaces In The Context Of Ambient Intelligence,” *Ann. Fac. …*, no. 6, pp. 963–967, 2009.

[41] “Flymaster Avionics.” [Online]. Available: http://www.flymaster-avionics.com/.

[42] M. Becker, E. Werkman, T. Kleinberger, and M. Anastasopoulos, “Approaching ambient intelligent home care systems,” *Proc. Pervasive Heal. Conf. Work.*, pp. 1–10, 2006.

[43] “Ontology Definition (Tom Gruber).” [Online]. Available: http://tomgruber.org/writing/ontology-definition-2007.htm.

[44] T. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?,” *Int. J. Hum. Comput. Stud.*, pp. 907–928, 1995.

[45] C. Roche, “Ontology: a survey,” *8th Symp. Autom. Syst. Based …*, 2003.

[46] “World Wide Web Consortium.” [Online]. Available: http://www.w3.org/.

[47] I. Standard, “INTERNATIONAL STANDARD ISO / IEC,” vol. 2007, no. CI, 2007.

[48] “OpenCyc.” [Online]. Available: http://www.opencyc.org/doc/.

[49] “KIF.” [Online]. Available: http://logic.stanford.edu/kif/dpans.html.

[50] R. Brachman and J. Schmolze, “An overview of the KL-ONE knowledge representation system,” *Cogn. Sci.*, vol. 216, pp. 171–216, 1985.

[51] A. Van Renssen, “Gellish: an information representation language, knowledge base and ontology,” *Stand. Innov. Inf. …*, 2003.

[52] D. L. McGuinness and F. Van Harmelen, “OWL Web Ontology Language Overview,” *W3C Recomm.*, vol. 10, no. February, pp. 1–22, 2004.

[53] M. Balaban, “The F-logic approach for description languages,” *Ann. Math. Artif. Intell.*, vol. 15, pp. 19–60, 1995.

[54] V. Chaudhri and A. Farquhar, “Open Knowledge Base Connectivity 2.0. 3 (Proposed),” *… Knowl. …*, 1998.

[55] “Swoogle.” [Online]. Available: http://swoogle.umbc.edu/.

[56] “The Enterprise Ontology.” [Online]. Available: http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html.

[57] “SWEET Ontology.” [Online]. Available: http://sweet.jpl.nasa.gov/ontology/.

[58] “OBO Foundry.” [Online]. Available: http://www.obofoundry.org/.

[59] D. Bonino and F. Corno, “Dogont-ontology modeling for intelligent domotic environments,” *Semant. Web-ISWC 2008*, 2008.

[60] D. Bonino, E. Castellina, and F. Corno, “DOG: An Ontology-Powered OSGi Domotic Gateway,” *2008 20th IEEE Int. Conf. Tools with Artif. Intell.*, pp. 157–160, Nov. 2008.

[61] L. Sommaruga, A. Perri, and F. Furfari, “DomoML-env: an ontology for Human Home Interaction.,” *SWAP*, pp. 1–7, 2005.

[62] F. Furfari, L. Sommaruga, C. Soria, and R. Fresco, “DomoML: the definition of a standard markup for interoperability of human home

interactions," *Proc. 2nd Eur. Union Symp. Ambient Intell.*, pp. 41–44, 2004.

[63] P. Valiente-Rocha and A. Lozano-Tello, "Ontology-based expert system for home automation controlling," *23th Int. Conf. Ind. Eng. Other Appl. Appl. Intell. Syst.*, 2010.

[64] P. Valiente-Rocha, J. L. Redondo-García, and A. Lozano-Tello, "Ambient intelligence system for controlling home automation installations," *Fifth Iber. Conf. Inf. Syst. Technol.*, 2010.

[65] E. Meshkova and J. Riihijarvi, "Modeling the home environment using ontology with applications in software configuration management," *… , 2008. ICT 2008. …*, 2008.

[66] N. Roy, G. Pallapa, and S. Das, "An ontology-driven ambiguous contexts mediation framework for smart healthcare applications," *… 1st Int. Conf. …*, 2008.

[67] "Protégé." [Online]. Available: http://protege.stanford.edu/.

[68] D. Saldaña-jimenez and M. D. Rodríguez, "ELDeR : An Ontology for Enabling Living inDependently of Risks," pp. 622–627, 2009.

[69] N. F. Noy and D. L. Mcguinness, "Ontology Development 101 : A Guide to Creating Your First Ontology," pp. 1–22, 2000.

[70] "JENA." [Online]. Available: http://jena.apache.org/.

[71] "RDF." [Online]. Available: http://www.w3.org/RDF/.

[72] "JENA Ontology API." [Online]. Available: http://jena.apache.org/documentation/javadoc/jena/.

[73] T. A. T. Bishop and R. Karne, "A survey of middleware," 2002.

[74] S. Hadim and N. Mohamed, "Middleware for wireless sensor networks: A survey," *… Syst. Softw. Middleware, …*, 2006.

[75] S. Hadim and N. Mohamed, *Middleware: middleware challenges and approaches for wireless sensor networks*, vol. 7, no. 3. IEEE Computer Society, 2006, pp. 1–1.

[76] M. Bessi and L. Bruni, “A survey of context-aware middleware,” *Proc. 25th Conf. IASTED …*, 2007.

[77] K. Henricksen and R. Robinson, “A survey of middleware for sensor networks: state-of-the-art and future directions,” *Proc. Int. Work. Middlew. Sens. networks*, vol. 218, pp. 60–65, 2006.

[78] P. Bonnet, J. Gehrke, and P. Seshadri, “Towards Sensor Database Systems,” in *Mobile Data Management*, 2001, vol. 1987, pp. 3–14.

[79] C. C. Shen, C. Srisathapornphat, and C. Jaikaeo, “Sensor information networking architecture and applications,” *Ieee Pers. Commun.*, vol. 8, no. 4, pp. 52–59, 2001.

[80] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “TinyDB: an acquisitional query processing system for sensor networks,” *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[81] C. Curino, M. Giani, and M. Giorgetta, “Tinylime: Bridging mobile and sensor networks through middleware,” *Proc. Third IEEE Int. Conf. Pervasive Comput. Commun.*, no. PerCom, pp. 61–72, 2005.

[82] D. Gelernter, “Generative communication in Linda,” *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 1, pp. 80–112, Jan. 1985.

[83] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner, “Mires: a publish/subscribe middleware for sensor networks,” *Pers. Ubiquitous Comput.*, vol. 10, no. 1, pp. 37–44, Oct. 2005.

[84] W. Heinzelman and A. Murphy, “Middleware to support sensor network applications,” *IEEE Netw.*, vol. 18, no. February, pp. 6–14, 2004.

[85] S. Vinoski, “New features for CORBA 3.0,” *Commun. ACM*, 1998.

[86] C. Lee, D. Nordstedt, and S. Helal, “Enabling smart spaces with OSGi,” *Pervasive Comput. IEEE*, pp. 89–94, 2003.

[87] D. Valtchev, “Using OSGi for the Realization of Home Automation Systems OSGi Evolution,” in *OSGi Community Event 2010*, 2010.

[88] J. Rellermeyer, G. Alonso, and T. Roscoe, “R-OSGi: distributed applications through software modularization,” *Lect. Notes Comput. Sci.*, vol. 4834, pp. 1–20, 2007.

[89] C. Fabbricatore, M. Zucker, S. Ziganki, and A. P. Karduck, “Towards an unified architecture for smart home and Ambient Assisted Living solutions: A focus on elderly people,” in *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, 2011, vol. 5, no. June, pp. 305–311.

[90] “OSGi Alliance.” [Online]. Available: http://www.osgi.org/Main/HomePage.

[91] “openHAB.” [Online]. Available: http://www.openhab.org/index.php/start/.

[92] “openAAL.” [Online]. Available: http://openaal.org/.

[93] P. Wolf, A. Schmidt, and J. Otte, “openAAL-the open source middleware for ambient-assisted living (AAL),” *AALIANCE …*, no. March, pp. 1–5, 2010.

[94] “DOG.” [Online]. Available: http://elite.polito.it/dog-tools-72.

[95] M. Allman, “An evaluation of XML-RPC,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 2–11, Mar. 2003.

[96] “Raspberry Pi.” [Online]. Available: http://www.raspberrypi.org/.

Anexos

Anexo 1 – Cenários de automatização para a ontologia

Iluminação:

> Desligar luzes quando o utilizador adormece;
> Ligar/Desligar luzes quando existem pessoas na casa;
> Ligar/Desligar luzes quando existem pessoas na casa e está de noite;
> Ligar/Desligar luzes quando passam pessoas nas divisões;
> Ligar/Desligar luzes quando o utilizador se direciona para uma divisão;
> Ligar/Desligar luzes recorrendo à utilização de gestos;
> Alterar cor da iluminação de acordo com o utilizador na divisão;
> Alterar intensidade da iluminação de acordo com o utilizador na divisão;
> Alterar intensidade da iluminação quando o utilizador está a ver um filme (baixar a intensidade da luz);
> Alterar intensidade da iluminação quando o utilizador está a ler um livro (focar na iluminação perto do utilizador);
> Alterar intensidade da iluminação quando o utilizador está na cama (se estiver perto da hora de dormir, baixar intensidade);
> Alterar intensidade da iluminação quando o utilizador está no computador (fornecer boa iluminação na zona – saúde);
> Alterar intensidade da iluminação conforme o período do dia (manhã, tarde, noite) e época (estação do ano) e força da luz no exterior;
> Fornecer um modo de iluminação progressiva (intensidade incrementada) para quando o utilizador acorda;
> Alterar intensidade da iluminação no exterior conforme o período do dia (manhã, tarde, noite), época (estação do ano);
> Ligar/Desligar luz quando se toca à campainha (luz do exterior);
> Ligar/Desligar luzes remotamente (ex.: aplicação mobile);
> Abrir/Fechar estores conforme a razão de iluminação interior/exterior;
> Abrir/Fechar estores conforme o utilizador presente na divisão;

- Abrir/Fechar estores remotamente (ex.: aplicação *mobile*);
- Abrir/Fechar estores recorrendo à utilização de gestos;
- Controlar inclinação dos estores conforme razão de iluminação interior/exterior.

Som Ambiente:

- Ligar/Desligar o som ambiente do *home theater* (ex.: ao ver um filme);
- Ligar/Desligar o som ambiente conforme a pessoa presente na divisão;
- Ligar/Desligar o som ambiente, em modo rádio, conforme a pessoa presente na divisão e quando esta está a acordar;
- Alterar o som a tocar na divisão conforme a pessoa presente na divisão;
- Alterar o volume do som a tocar na divisão conforme a pessoa presente na divisão;
- Alterar o volume e tom do som a tocar pela campainha de entrada conforme a pessoa que está no exterior;
- Reproduzir o nome da pessoa no exterior quando esta toca à campainha (se for reconhecida);
- Alterar o volume da TV quando o telefone toca;
- Alterar o volume do som ambiente quando o telefone toca;
- Alterar o volume da TV quando a campainha toca;
- Alterar o volume do som ambiente quando a campainha toca;
- Reproduzir no som ambiente as tarefas que o utilizador tem de realizar no dia (ex.: quando acorda ou quando vai sair de casa);

Climatização:

- Ligar/Desligar climatização quando existem pessoas na casa;
- Ligar/Desligar climatização quando o utilizador adormece ou está prestes a acordar;
- Ligar/Desligar climatização recorrendo à utilização de gestos;
- Alterar o valor da climatização de acordo com o utilizador na divisão;

- Alterar o valor da climatização conforme o período do dia (manhã, tarde, noite) e época (estação do ano);
- Ligar/Desligar o aquecimento da cama conforme o período do dia (se o utilizador está prestes a adormecer ou acordar);
- Ligar/Desligar o aquecimento dos guarda-fatos conforme o período do dia (se o utilizador está prestes a adormecer ou acordar);
- Controlar inclinação dos estores conforme a temperatura exterior;
- Abrir/Fechar estores conforme o período do dia de modo a poupar no aquecimento/arrefecimento da casa;

Painéis de notificação/controlo:

- Fornecer integração com ambiente displays;
- Mostrar na TV a imagem capturada pelo videoporteiro;
- Alterar imagens presentes nos painéis conforme o utilizador presente na divisão;
- Fornecer uma lista de produtos presentes no frigorífico;
- Criar uma lista de produtos em falta no frigorífico;
- Fornecer uma lista de produtos presentes na despensa;
- Criar uma lista de produtos em falta na despensa;
- Fornecer uma lista de tarefas que o utilizador tem de completar para o dia;
- Fornecer uma lista de contas que o utilizador tem para pagar;
- Fornecer uma notificação sobre a existência de correspondência na caixa de correio;
- Fornecer uma lista de aniversários a decorrer num futuro próximo;
- Fornecer uma agenda virtual para questões profissionais;
- Fornecer uma lista de programas que irão decorrer na TV que poderão ser do agrado do utilizador;
- Fornecer uma lista de programas que irão decorrer na rádio que poderão ser do agrado do utilizador;
- Fornecer uma lista de programas que irão decorrer na internet que poderão ser do agrado do utilizador;

- Fornecer uma notificação visual sobre o clima que se fará sentir no dia e nos dias seguintes;
- Fornecer um aviso visual sobre a falta de peças de roupa adequadas ao clima;
- Fornecer uma notificação visual sobre o estado do trânsito. Enfâse nas localizações de interesse do utilizador;
- Fornecer uma lista de possíveis receitas num painel junto à cozinha;
- Fornecer método de transferir conteúdos de um monitor ligado numa divisão para outra (computador ligado a um monitor no escritório estar a dar imagem na sala);
- Fornecer uma notificação visual e sonora sobre o estado de bateria dos dispositivos que irão ser necessários para o dia;
- Fornecer um método de sincronização entre os aparelhos móveis e o core da casa para calendários, imagens, mensagens, etc.;
- Direcionar automaticamente as chamadas do telefone de casa para o telemóvel do utilizador, quando este não se encontra em casa;
- Habilitar/Desabilitar notificações (telefonemas, mensagens, emails, etc.) quando o utilizador não quer ser incomodado;
- Controlar manualmente através de ecrã táctil, a temperatura da casa;
- Controlar manualmente através de ecrã táctil, a iluminação da casa;
- Controlar manualmente através de ecrã táctil, os estores da casa;
- Controlar manualmente através de ecrã táctil, as janelas da casa;
- Controlar manualmente através de ecrã táctil, alguns eletrodomésticos da casa;
- Controlar manualmente através de ecrã táctil, o sistema de rega da casa;
- Controlar manualmente através de ecrã táctil, a temperatura da água da casa;
- Controlar manualmente através de ecrã táctil, as portas da casa;

Outros:

- Fechar os portões quando se deliga o carro;
- Monitorizar a qualidade do ar e abrir janelas ou ligar purificador. De preferência quando o utilizador não estiver em casa;
- Monitorizar os parâmetros básicos de saúde para percecionar o seu estado de saúde. Adaptar ambiente (luz, temperatura e som) conforme essa medição;
- Notificar o utilizador para tomar a medicação a tempo;
- Ligar/Desligar exaustores ou ventilações quando se detetam odores indesejados (cozinha, WC);

Segurança casa/bens:

- Ativar automaticamente o alarme quando o utilizador sai de casa;
- Fechar automaticamente as janelas de casa quando o utilizador sai de casa;
- Trancar automaticamente portas e garagem quando o utilizador sai de casa;
- Fechar automaticamente os estores quando o utilizador sai de casa;
- Trancar automaticamente as portas quando se deteta uma pessoa estranha à casa no seu exterior;
- Se for detetado movimente no exterior durante a noite, ligar as luzes onde o movimento é detetado;
- Trancar todas as portas e janelas à noite, quando o utilizador for dormir;
- Ativar o alarme automaticamente quando o utilizador for dormir;
- Permitir simular a presença de pessoas em casa (ligar/desligar luzes) quando o utilizador se ausenta para férias;
- Desligar automaticamente a água em situações de inundação;
- Desligar localmente o fornecimento de eletricidade em situações de inundação;
- Desligar automaticamente a água de uma torneira em caso de esquecimento;
- Desligar o fornecimento de gás em caso de fuga;

- Desligar o fornecimento de gás e eletricidade em situação de fumo;
- Notificar o utilizador para a presença de pessoas estranhas à casa;
- Enviar imagens das camaras de vigilância para o telemóvel/portátil do utilizador quando alguém se aproxima de casa (se não estiver ninguém lá);
- Detetar intrusão a partir de janelas e notificar o utilizador;
- Permitir o reconhecimento de impressões digitais para entrar em casa;
- Recolha automática do correio quando o dono se ausenta de casa durante algum tempo;
- Fornecer uma notificação quando existem intrusões na rede wi-fi da casa;
- Notificar quando os níveis de bateria dos detetores de fumo estão baixos;
- Notificar quando os níveis de bateria dos detetores de fugas de água estão baixos;
- Notificar quando os níveis de bateria dos detetores de inundações estão baixos;
- Notificar quando os níveis de bateria do alarme estão baixos;
- Notificar o utilizador em caso de falha dos dispositivos de corte de água;
- Notificar o utilizador em caso de falha dos dispositivos de corte de gás;
- Notificar o utilizador em caso de falha dos dispositivos de corte de energia elétrica;
- Notificar em caso de sobrecargas elétricas nas divisões;
- Em caso de sobrecargas elétricas, o sistema deve desligar alguns dos aparelhos;
- Avisar se houver elementos que possam impedir as portas e janelas de fechar;
- Impedir que as portas e janelas se fechem se existir algum elemento que as possa bloquear;
- Impedir que a porta e/ou janela afeta à divisão da varanda se feche quando o utilizador sai através dela;

- Em caso de situação de incêndio, destrancar todas as portas;
- Em caso de falha de energia elétrica, ligar automaticamente às fontes de energia renováveis;

Segurança pessoal:

- Impedir que a temperatura ambiente ultrapasse os 30ºC;
- Impedir que a temperatura da água no WC seja superior a 38ºC;
- Monitorizar o estado de saúde dos utilizadores e lançar alertas caso seja detetado algum problema de saúde (ligar a familiares, ligação com médico, etc.);
- Monitorizar utilizador e em caso de queda ligar para números de emergência;
- Monitorização e alerta em caso de crianças em zonas de risco (varandas, janelas abertas, etc.);
- Fechar cobertura da piscina quando não estiver a ser utilizada;
- Trancar armário com objetos perigosos (armas, bastões, etc.) na presença de crianças;
- Trancar armário dos produtos químicos na presença de crianças;
- Impedir ligar/desligar de eletrodomésticos por crianças;
- Ligar exaustores ou ventilações se forem detetados gases tóxicos;
- Quando for premido o botão de pânico, ligar para familiares;