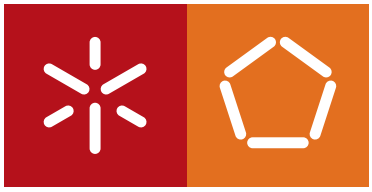
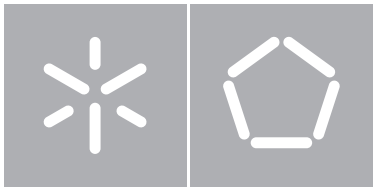


Universidade do Minho
Escola de Engenharia

João Pedro Almeida Brandão

Plataforma integradora de serviços em tempo real baseada em tecnologia OSGi

Julho de 2013



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Pedro Almeida Brandão

Plataforma integradora de serviços em tempo real baseada em tecnologia OSGi

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor Paulo Novais
Doutor Ângelo Costa

Julho de 2013

Agradecimentos

Gostaria de agradecer a todos que direta e indiretamente influenciaram positivamente a minha vida e percurso académico:

Ao Professor Paulo Novais e ao Doutor Ângelo Costa pela partilha de experiencia, dedicação e pelo apoio constante.

Aos meus colegas de casa, André Pimenta e Miguel Gomes, pelo companheirismo e força que me transmitiram.

A todos que partilharam comigo o trajeto académico principalmente aos que o iniciaram comigo.

Aos meus velhos amigos, Honoré Tinoco, David Gomes, Jorge Ferreira e André Gonçalves pela amizade ao longo dos anos.

À minha família que sempre me apoiou.

E por fim um especial agradecimento aos meus pais Mavilde Almeida e José Brandão por toda a dedicação e apoio, e ao meu irmão Jorge Brandão por todos os concelhos e incentivo.

Este trabalho foi desenvolvido no contexto do projeto CAMCoF - Context-aware Multimodal Communication Framework financiado por Fundos FEDER através do Programa Operacional Fatores de Competitividade - COMPETE e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no âmbito do projeto FCOMP-01-0124-FEDER-028980.

Resumo

Este projeto tem o objetivo principal de melhorar a qualidade de vida de pessoas idosas, com deficiências físicas ou psicológicas. Quando ficam em casa sozinhas são privadas de uma vida autónoma e ativa o que leva à necessidade de uma monitorização continua. Tendo em consideração a crescente disponibilidade de dispositivos interativos num ambiente doméstico abre-se uma porta para necessidade de sistemas de integração e fusão de sensores. Uma plataforma inteligente de monitorização capaz de comunicar com os vários dispositivos prova a sua utilidade ao ser capaz de tornar as pessoas mais autónomas proporcionando aos familiares e amigos mecanismos de monitorização ajustados conforme o perfil do utilizador em caso de ausência.

Pretende-se desenvolver um sistema que possa ser implementado num dispositivo central que estabeleça a comunicação entre os diferentes dispositivos e sistemas electrónicos, seja capaz de integrar serviços em tempo real e registe o seu estado em determinado momento. Tem o intuito de aproveitar todos os aparelhos e serviços que as pessoas já possuem, evitando assim um gasto monetário exagerado tendo em conta o estado económico existente e a capacidade monetária dos utilizadores. Estes sistemas electrónicos ou dispositivos podem ser, por exemplo, um sistema de ar condicionado, capaz de adequar a temperatura à preferível pelo utilizador, ou um sistema de iluminação com capacidade de regular a intensidade da luz e assim reduzir nas despesas.

Estes sistemas implicam uma melhoria em termos de qualidade de vida do utilizador ao providenciar automatismos inteligentes no seu dia-a-dia.

Abstract

This project has the main goal of improving the quality of live of elderly people with physical or psychological disabilities. When they stay at home alone they are deprived of an independent and active life and which leads to the need for continuous monitoring. Taking into account the increasing availability of interactive devices in a home environment, opens a door for the need of integration systems and sensor fusion. An intelligent monitoring platform capable of communicating with multiple devices proves its utility in being able to make people more autonomous, providing to the family and friends monitoring mechanisms adjusted as the user's profile in case of absence.

The aim is to develop a system that can be implemented in a central device to establish communication between different electronic devices and systems, capable of integrating services in real-time and record their status. Aims to take advantage of all the devices and services that people already have, thus avoiding spending money exaggeratedly, given the existing economic and monetary capacity of users. These electronic systems or devices can be, for example, an air conditioning system capable of adjusting the temperature preferred by the user and lighting system capable of regulating the light intensity and thus reduce the costs.

These systems imply an improvement in quality of life of the user by providing intelligent automation in their day-to-day.

Índice

Agradecimentos iii

Resumo v

Abstract vii

Índice ix

Índice de Figuras xiii

Índice de Tabelas xvii

Glossário xix

1. Introdução 1

1.1. Desafios 2

1.1.1. Sociais 2

1.1.2. Tecnológicos 4

1.2. Objectivos 5

1.3. Metodologia de Investigação 5

1.4. Estrutura do documento 6

2. Ambient Intelligence 7

2.1. Arquitetura de sistemas 8

2.2. Protocolos de comunicação 10

2.2.1. Com fios 10

2.2.2. Sem fios 11

2.3. Ambient Assisted Living 13

3. Tecnologias 15
3.1. Projetos 15
3.2. Framework OSGi 18
3.2.1. Bundles 18
3.2.2. Serviços 20
3.2.3. Plataformas 20
3.2.4. Bnd 21
3.3. Sensores 21
3.3.1. Características 23
3.3.2. Aplicação em Inteligência Ambiente 23
3.3.3. Phidgets 24
3.4. Jetty 26
3.5. Java 27
3.6. Síntese 30
4. BSense 31
4.1. Arquitetura 31
4.2. Camada Física 33
4.3. Camada bundles de sensores 34
4.4. Camada de serviços 37
4.5. Camada API de serviços 38
4.6. Camada de aplicação 42
4.7. Estrutura de dados 45
4.8. Síntese 47
5. Casos de teste 49
5.1. Cenários 50
5.1.1. Perigo 50

5.1.2. Conforto 51
5.1.3. Monitorização 52
5.1.4. Segurança 53
5.2. Implementação dos bundles para Sensores 53
5.2.1. Bundle Interface Sensores 54
5.2.2. Bundle RFID 56
5.2.3. Bundle Advanced Servo 59
5.3. Síntese 61
6. Conclusões e Trabalho Futuro 63
6.1. Trabalho Futuro 64
7. Bibliografia 67
Anexo I 71
Características de Phidgets 71
Anexo II 75
Área de configuração da plataforma 75

Índice de Figuras

Figura 1 - Proporção de pessoas acima dos 60 anos 2

Figura 2 - Indicadores de envelhecimento segundo os Censos em Portugal 3

Figura 3 - Camas em hospitais por 100 mil habitantes 3

Figura 4 - Evolução da IA 8

Figura 5 - Protocolos de comunicação 10

Figura 6 - Arquitetura OSGi 13

Figura 7 – Componentes de um *Ambient Assisted Living* 14

Figura 8 - Cisco TES301 IP 16

Figura 9 - Dispositivos da AlerteMe 16

Figura 10 - Arquitetura AALuis 17

Figura 11 - Arquitetura Hearing@Home 17

Figura 12 - Conteúdo de um *bundle* 19

Figura 13 - Transições de estados de um *bundle* 19

Figura 14 - Capacidade sensorial de um *smartphone* usual 22

Figura 15 - Interação com sensores 22

Figura 16 - Interface Kit 8/8/8 25

Figura 17 - RFID Phidget 26

Figura 18 - Advanced Servo Phidget 26

Figura 19 - Carregamento de classes em Java 28

Figura 20 - Primeiro acesso a página JSP 29

Figura 21 - Arquitetura externa da plataforma BSense 31

Figura 22 - Arquitetura interna da plataforma BSense 32

Figura 23 - Camada física do sistema BSense 33

Figura 24 - Camada bundle de sensores do sistema BSense 34

Figura 25 - Fluxograma de carregamento de classes em tempo real 35

Figura 26 - Manifest de RFID Phidget para o sistema BSense 36

Figura 27 - Camada de serviços do sistema BSense 37

Figura 28 - Diagrama de classes da camada de serviços do OSGi 38

Figura 29 - Camada API de serviços do OSGi 38

Figura 30 - Diagrama de classes da camada API de serviços 39

Figura 31 – Interface para Serviço Analog Senses que implementa a API SenseService 40

Figura 32 – Interface do atributo DataRate Presence do serviço Sensitivity que implementa a API SliderSensor 40

Figura 33 – Interface do atributo Antenna do serviço Switchs que implementa a API SwitchSenseor 40

Figura 34 - Exemplo de implementação de um serviço Actuador 41

Figura 35 – Diagrama de sequencia do evento de selecionar controlador para um actuador 41

Figura 36 - Exemplo de comunicação entre controlador e actuador 42

Figura 37 - Camada de aplicação do sistema BSense 42

Figura 38 - Diagrama de Use Case para a interação entre o utilizador e a plataforma 43

Figura 39 - Página inicial da aplicação Web 44

Figura 40 - Página de instalação e desinstalação de *bundles* 45

Figura 41 – Estrutura de sensores do sistema BSense 45

Figura 42 - Ficheiro exemplo de dados guardados em JSON 46

Figura 43 - Material utilizado para testes 49

Figura 44 - Distribuição de sensores para cenário de perigo 50

Figura 45 - Distribuição de sensores para cenário de conforto 51

Figura 46 - Distribuição de sensores para cenário de monitorização 52

Figura 47 - Posicionamento de leitor RFID em cenário de segurança 53

Figura 48 - Diagrama de classes para *bundle* Interface Kit 8/8/8 54

Figura 49 - Diagrama de sequencia para a conexão de dispositivo 55

Figura 50 - Valores dos sensores analógicos disponibilizados pela plataforma 55

Figura 51 - Diagrama de sequencia quando sensor muda o seu estado 56

Figura 52 - Atributos do Serviço SensitivityService 56

Figura 53 - Diagrama de classes de *bundle* para *Phidget* RFID 57

Figura 54 - Diagrama de sequencia para quando Phidget RFID é conectado 58

Figura 55 - Diagrama de sequancia para quando *tag* é identificada 58

Figura 56 - Diagrama de classes de *bundle* para *Phidget* Advanced Servo 59

Figura 57 - Diagrama de sequencia para a conexão de *Phidget* AdvancedServo 60

Figura 58 - Diagrama de sequencia para mudança de estado do motor RC 60

Figura 59 - Serviço de Log..... 75

Figura 60 - Informação do sistema..... 76

Figura 61 - Serviços no sistema..... 76

Índice de Tabelas

Tabela 1 - Placa Interface 8/8/8 Phidget 71

Tabela 2 - *Inputs* Analógicos 71

Tabela 3 - *Inputs* Digitais da Interface 71

Tabela 4 - *Outputs* digitais da Interface 72

Tabela 5 - Leitor RFID 72

Tabela 6 - Propriadades elétcias 72

Tabela 7 - Propriadades físicas 72

Tabela 8 - *Outputs* digitais RFID 72

Tabela 9 - *Advanced Servo Phidget* 73

Glossário

AAL *Ambient Assisted Living*

AmI Inteligência Ambiente

AP *Access Point*

ASP *Active Server Pages*

CGI *Common Gateway Interface*

HTML *HyperText Markup Language*

HTTP *HyperText Transfer Protocol*

I/O *Input/Output*

ISTAG *Information Society Technologies Advisory Group*

J2EE *Java 2 Platform Enterprise Edition*

IA Inteligência Artificial

JASPI *Java Authentication Service Provider Interface*

JMX *Java Management Extensions*

JNDI *Java Naming and Directory Interface*

JSON *JavaScript Object Notation*

JSP *JavaServer Pages*

LAN *Local Area Network*

ML *Machine Learning*

OSGi *Open Services Gateway Initiative*

PHP *PHP:Hypertext Preprocessor*

RDF *Resource Description Framework*

SOA *Service-oriented Architecture*

SPDY *Speedy*

STA *Station*

UPnP *Universal Plug and Play*

USB *Universal Serial Bus*

VoIP *Voice over Internet Protocol*

WAN *Wide Area Network*

WLAN *Wireless Local Area Network*

XML *eXtensible Markup Language*

YAML *Yet Another Markup Language*

1. Introdução

A autonomia, assim como a componente social, são atributos essenciais para a felicidade humana principalmente para pessoas com limitações. De acordo com o Gandhi e o Truman, um dos índices de avaliação da civilização, da sociedade é a forma que as pessoas idosas são tratadas. Este projeto tem o objectivo de contribuir para um melhoramento da qualidade de vida de pessoas com limitações físicas ou mentais.

Esta dissertação insere-se num contexto de *Ambient Assisted Living*(AAL), estas plataformas tem o objetivo de poder garantir aos doentes várias vantagens e serviços que permitem melhorar a qualidade de vida.

O projeto trata-se de uma plataforma, de nome BSense, capaz de integrar serviços em tempo real, com o objectivo que o utilizador final seja capaz de instalar e desinstalar dispositivos sem necessitar de reiniciar o sistema.

Surge no seguimento de outros projetos realizados na mesma área pelo grupo *Intelligent Systems Lab* (ISLab) associado à Universidade do Minho. BSense nasce com o desafio lançado por outro projeto de nome iGenda, que consiste em uma agenda com agentes inteligentes num sistema Android, capaz de autonomamente reorganizar a agenda pessoal de um utilizador, relembrar de medicamentos, cancelar reuniões em conjunto com outros utilizadores, entre outras funcionalidades. Pretende-se uma futura integração entre os dois projetos de forma a colaborarem para poder facilitar a monitorização e oferecer uma maior numero de serviços oferecidos pelo dispositivo Android.

1.1. Desafios

Esta dissertação tem como motivação enfrentar desafios atuais relativamente a questões sociais como o envelhecimento da população e o aumento da qualidade de vida, assim como questões tecnológicas como a capacidade de integrar serviços em tempo real.

1.1.1. Sociais

O envelhecimento da população é um dos grandes problemas globais mas principalmente em Portugal e na Europa(Nações Unidas, 2009), e é acompanhado do decréscimo da natalidade, tem por consequência o aumento de pessoas dependentes e um aumento da importância das pessoas mais idosas para a sociedade.

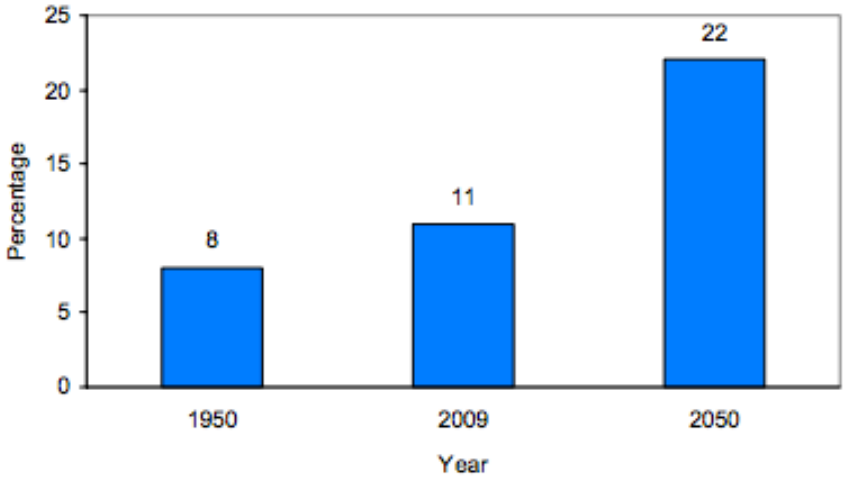


Figura 1 - Proporção de pessoas acima dos 60 anos

Como se pode ver na figura 1(Nações Unidas, 2009) é um problema global e em Portugal está comprovado que para além de o índice de envelhecimento esteja a aumentar cada vez mas a dependência de idosos aumenta como mostra a figura 2.

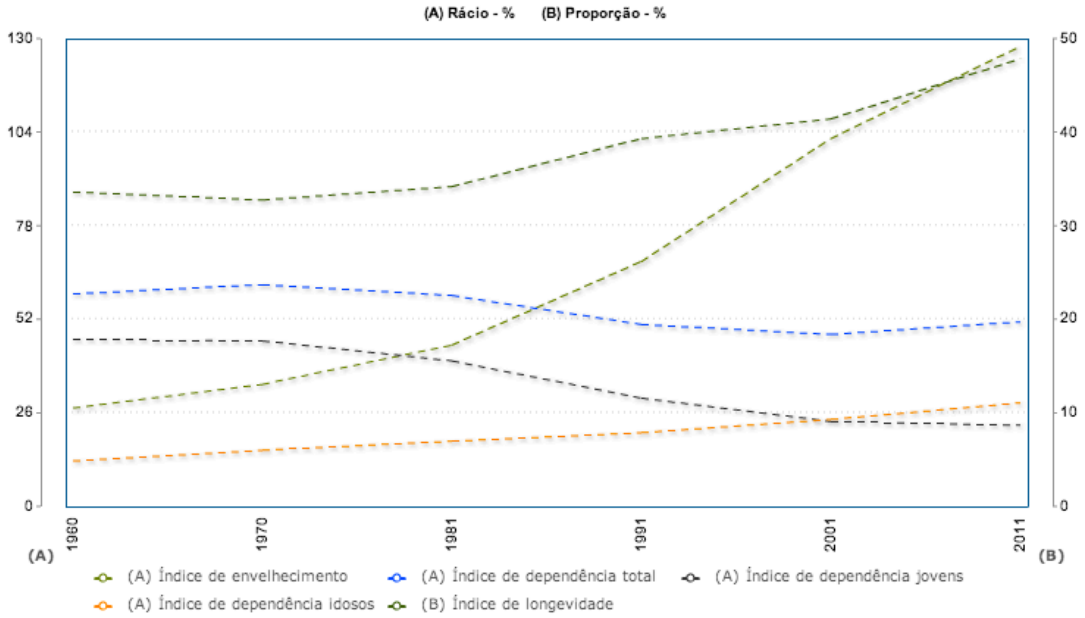


Figura 2 - Indicadores de envelhecimento segundo os Censos em Portugal

No panorama médico em Portugal no ano de 2010, foram registadas apenas 335,7 camas por 100000 pessoas(Pordata, 2011), por vezes os doentes são enviados para casa mas com determinados cuidados, e necessitam de monitorização.

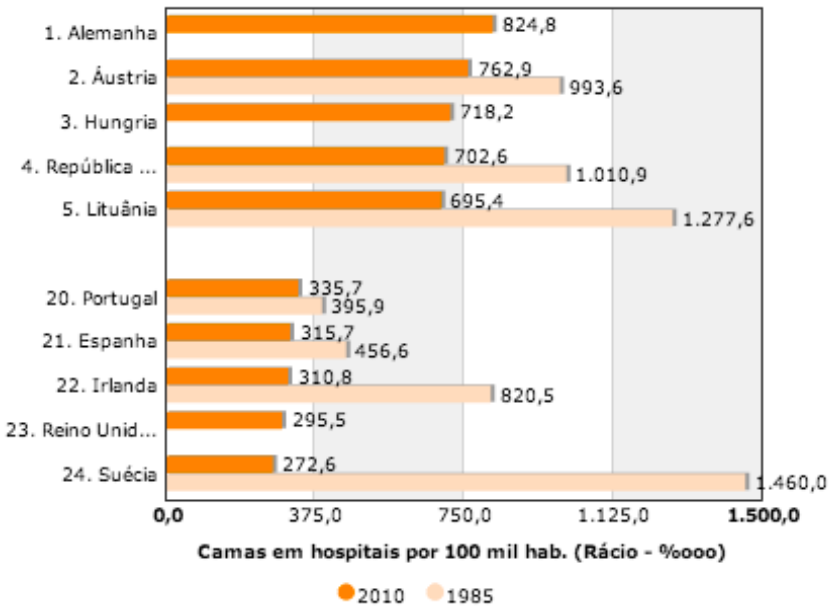


Figura 3 - Camas em hospitais por 100 mil habitantes

O estudo [1] comprova que a maior parte das pessoas preferem ficar em casa do que numa unidade médica. A implementação de uma plataforma AAL, pode fornecer um sistema de moni-

torização, e uma facilidade de comunicação com a família/amigos, mesmo estando em casa. A monitorização de pacientes remotamente através destas plataformas permite o suporte a tratamentos eficazes e detecção precoce de situações anormais. Para além disso também está provado que que contribui para uma redução significativa de hospitalizações e um aumento do sucesso em tratamentos de longo prazo [20] [21] [22].

Um sistema AAL é naturalmente atrativo para famílias, pois estes são a entidade em que recai a responsabilidade de cuidar e aumentar a qualidade de vida dos seus parentes mais envelhecidos. Para além do sistema proporcionar autonomia ao doente, oferece também uma maior independência ao resto da família pois o sistema AAL pode supervisionar e cuidar com segurança os seus ente-queridos. Assim como ajudar a entender as necessidades do idoso, como revelado no estudo [10], muitos dos responsáveis não conhecem os desejos e necessidades reais do idoso.

Lares e organizações de prestações de cuidados são também um dos principais interessados neste tipo de tecnologia devido ao grande numero de pessoas à sua responsabilidade que poderão estar em estado muito débil e requerem uma monitorização e tratamento continuo e eficaz.

1.1.2. Tecnológicos

Cada vez mais as pessoas adquirem dispositivos para a sua habitação, muitos dos dispositivos electrónicos estão a diminuir o tamanho e a aumentar a precisão dos sensores, tornando-se mais eficientes e ao mesmo tempo pouco intrusivos [1].

Este projeto é elaborado com o intuito de possibilitar interligar dispositivos domiciliários através de um *home gateway*, de fácil configuração, providenciando ao utilizador um maior conforto e segurança.

Este tem a importante característica de utilizar uma tecnologia emergente de nome *Open Services Gateway initiative* (OSGi). Possibilitando a instalação de módulos e integração de serviços em tempo real.

É desenvolvida uma plataforma capaz de detectar dispositivos ligados e disponibilizar uma interface dinâmica ao utilizador através da internet.

1.2. Objectivos

Como referido anteriormente o foco deste projeto passa por desenvolver um sistema que no futuro seja capaz de proceder à monitorização de pessoas. Para isso pretende-se desenvolver uma plataforma de integração consistido num *home gateway* com a tecnologia OSGi. A plataforma deverá ser capaz de carregar módulos que iniciem a execução em tempo real.

O sistema deve integrar uma aplicação Web dinâmica que permite aos utilizadores interagirem com o ambiente e monitorizarem as entidades presentes através de dispositivos com acesso à internet podendo deste modo a família mesmo longe estar atenta a possíveis inconvenientes. Tendo em conta os requisitos deste projeto podem ser definidos os seguintes objetivos principais a atingir neste projeto:

- Desenvolvimento da arquitetura usando tecnologia OSGi
- Desenvolvimento da plataforma.
- Integrar Aplicação Web dinâmica
- Integrar sistema de comunicação em um *homegatway*
- Desenvolvimento e implementação de casos de teste

1.3. Metodologia de Investigação

Este projeto foi desenvolvido segundo uma metodologia ação-investigação, em que perante a presença de um dado desafio é estipulada uma hipótese de solução e posteriormente tiradas conclusões.

Procedeu-se então a uma compilação e organização de informação relevante para o problema. Inserindo-se este projeto em uma área em constante evolução é necessária uma observação continua sobre a comunidade científica. Depois de identificado o problema foi formulada uma hipótese de solução para o problema identificado. Procedeu-se então ao seu planeamento e posteriormente ao desenvolvimento. Na etapa final, foram formulados casos de destes e efectuadas as respetivas conclusões que permitam avaliar os resultados obtidos. Em suma o projeto passou pelas fases de:

- Diagnosticar ou descobrir uma preocupação temática, identificando o problema.
- Estudo sobre o problema e projetos inseridos na área.
- Construção de um plano de ação
- Proposta prática do plano e observação do seu funcionamento.
- Formulação de casos de teste.
- Reflexão, interpretação e avaliação dos resultados.

1.4. Estrutura do documento

No capítulo 2 é apresentada uma aproximação ao conceito de Inteligência Ambiente (AmI). Dentro deste será abordada a arquitetura tradicional de sistemas de comunicação uma análise sobre os desafios de uma arquitetura para AmI e posteriormente é apresentada a arquitetura orientada ao serviço que é utilizada no projeto e na tecnologia OSGi, que pretende resolver alguns desses desafios. Posteriormente serão abordados alguns dos protocolos de comunicação mais utilizados que podem ser encontrados em AmI, com e sem fios e a sua comparação. No fim desse mesmo capítulo é abordado o conceito de *Ambient Assisted Living*, uma área dentro da AmI em que o projeto se insere.

Durante o capítulo 3 são abordadas as diferentes tecnologias utilizadas para o desenvolvimento da plataforma. Com um grande foco na tecnologia OSGi explicando os seus conceitos necessários para a compreensão da dissertação. São abordados os diversos tipos de sensores com foco nos utilizados para os casos de teste. Finalmente são expostas algumas das características mais relevantes das tecnologias para o desenvolvimento ao nível de software.

No capítulo 4 é apresentada a plataforma BSense, explicando a sua arquitetura assim como cada uma das suas camadas. No final é feita uma síntese sobre o desenvolvimento da plataforma.

O capítulo 5 aborda os casos de teste elaborados utilizando os sensores disponíveis. São então definidos 4 cenários possíveis, de perigo, conforto, monitorização e segurança. Posteriormente é explicada toda a concepção de *bundles*, desenvolvida para integrar na plataforma BSense.

Por fim no capítulo 6 é feita uma síntese de todo o trabalho realizado, tiradas algumas conclusões e especificados possíveis caminhos futuros que terá este projeto.

2. Ambient Intelligence

Ambient Intelligence[1] é um termo falado pela primeira vez numa investigação da comissão *Information Society Technologies Advisory Group* (ISTAG) em 2001 para descrever a criação de um ambiente com tecnologia avançada para criar um espaço envolvente para os ocupantes, proporcionando um maior conforto, segurança e prazer. Capaz de ajudar em tarefas diárias ou mesmo profissionais.

Um sistema AmI pode ser constituído por sensores e actuadores distribuídos de modo a criar uma camada de tecnologia extensível, capaz de comunicar com o utilizador de um modo transparente. De um modo passivo deve observar e tentar compreender as ações do utilizador assim como as suas intenções, de um modo ativo deve aprender as preferências do utilizador e adaptar os parâmetros do sistema, melhorando assim a qualidade de vida e de trabalho do utilizador. O conceito pode ser aplicado a diversos tipos de ambiente, seja um espaço fechado, aberto, grande ou pequeno. Desde que permita uma rede difundida capaz de perceber as preferências do utilizador através do espaço e do tempo e capaz de proporcionar uma relação interativa Homem-Maquina.

Diane J. Cook, Juan C. Augusto e Vikramaditya R. Jakkula [11] identificam cinco características chave de um AmI: Integração, Sensibilidade ao Contexto, Personalização, Adaptabilidade e Antecipação.

Na opinião de Carlos Ramos, Juan C. Augusto e Daniel Shapiro [15] para o funcionamento de um AmI é necessário uma simbiose com o campo de Inteligência Artificial(IA), assumindo que AmI é o passo seguinte da evolução da IA. Na Figura 4 é descrito, segundo [15], como pode ser visto um AmI, pela AI.

[1] Inteligência Ambiente, neste documento usaremos a abreviatura AmI para simplificar a sua designação.

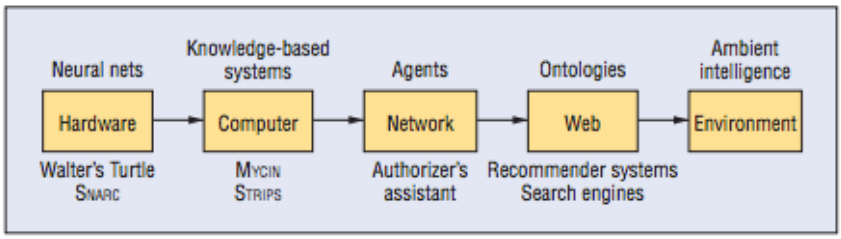


Figura 4 - Evolução da IA

Uma área de Sistemas Inteligentes, é AAL, que se foca mais no utilizador e é destinado para pessoas com necessidades especiais, idosos e pessoas com deficiências motoras ou cognitivas.

2.1. Arquitetura de sistemas

Na arquitetura tradicional, todos os componentes de uma rede são controlados por um *home gateway* que fornece serviços aos utilizadores [8]. Os utilizadores controlam outros sistemas através do mesmo servidor, permitindo definir cenários sobre como os dispositivos domiciliários devem funcionar em conjunto. A partir dos diferentes cenários o servidor controla os diferentes dispositivos. O servidor deve ser capaz de trabalhar com diferentes tecnologias e protocolos.

Em alguns casos, as aplicações podem trabalhar numa rede ponto-a-ponto, no entanto esta característica é destinada para aplicações que utilizam o mesmo protocolo. O problema está em não ser possível integrar outros dispositivos que utilizam protocolos diferentes. E é por essa razão que um *home gateway* deve funcionar como *service gateway*, traduzindo os diferentes protocolos possibilitando assim a comunicação entre eles.

Desafios

Um dos problemas com as aplicações atuais trata-se da interoperabilidade. Muitas das soluções estão restritas a uma só tecnologia de comunicação [8]. Como já foi referido pode ser utilizado então um *service gateway* para controlar e interligar os diferentes sistemas, no entanto esta opção traz algumas dificuldades na sua utilização e configuração.

Outro problema reside sobre a escalabilidade do sistema, com a utilização de um *service gateway* de forma a permitir a comunicação entre diferentes tecnologias existe o problema de quando um novo dispositivo é adicionado e a sua tecnologia é desconhecida para o *gateway*. O utilizador para poder utilizar o novo dispositivo, terá de atualizar o servidor, que é um processo complicado. O mesmo acontece quando se efetua mudanças num dado dispositivo, os outros podem não conseguir adaptar-se a essas mudanças.

Com as dificuldades apresentadas, é necessário surgirem arquiteturas capazes de colmatar estas falhas. De seguida apresentamos Arquitetura Orientada ao Serviço, que se pretende a isso mesmo.

Arquitetura Orientada ao Serviço

Service Oriented Architeture(SOA) [8], é uma arquitetura de software que não é uma revolução mas sim uma evolução da Arquitetura Baseada em Componentes, Arquitetura Orientada a Objetos e Sistemas Distribuídos.

A característica fundamental de um SOA é de que existem serviços com interfaces definidas que podem ser executadas sem conhecer quem as executa. Cada componente no sistema dá a conhecer os serviços que disponibiliza ao mesmo tempo que encontra serviços que outros componentes disponibilizam na mesma forma. Os componentes negoceiam entre si para poderem utilizar os serviços alheios sem nenhum controlador central.

SOA pode assim resolver as dificuldades mencionadas anteriormente. Em SOA os dispositivos dão a conhecer os seus serviços com a sua descrição e num formato normalizado. A finalidade é de que se dois dispositivos utilizam tecnologias diferentes, eles continuam a entender como executar os serviços um do outro. Como na arquitetura os componentes são autónomos, torna o sistema escalável e mais flexível. Pois podem trabalhar independentemente com pouca ou nenhuma configuração do utilizador. Mudança na rede deixa assim de ser problema.

Para a implementação em inteligência ambiente pode ser usada qualquer implementação de SOA (*Web Services*, .NET, J2EE) como *framework* para construir aplicações a na camada de serviços. No entanto a *framework* deve ser suportada por um middleware. A nosso ver a plataforma OSGi, é uma boa escolha para utilizar em AmI. A razão é que o objectivo original de OSGi é providenciar uma *framework* para construir um *gateway* de serviços central. No capítulo 3 a plataforma OSGi será abordada em mais detalhe, pois foi a plataforma escolhida para o desenvolvimento do projeto.

2.2. Protocolos de comunicação

Nas últimas décadas, a automação tem evoluído numa área de pesquisa muito atraente incorporando diferentes disciplinas modernas, incluindo comunicação, computador, controle, sensor, actuador e engenharia de forma integrada, levando a novas soluções, melhor desempenho e sistemas completos. Um dos componentes cada vez mais importantes na automação é a industria da comunicação.

O número de protocolos de comunicação utilizada para interligar dispositivos domiciliarios é uma lista considerável, como se pode ver pela figura 5, existe o Bluetooth, HAVi, IEEE 802.11 ®, IEEE 802.15, IEEE 1394, a arquitetura UPnP ™, X10, HomePlug, e HomePNA, entre outros.

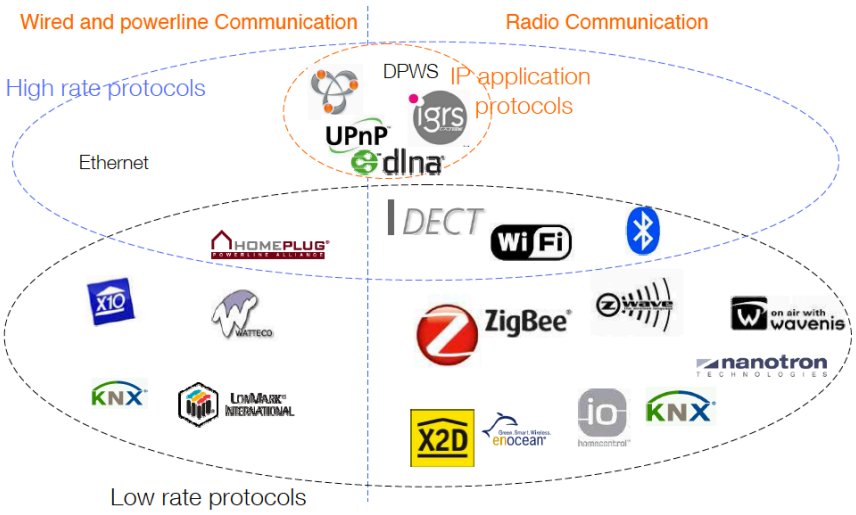


Figura 5 - Protocolos de comunicação

Nesta secção serão apresentadas algumas das tecnologias mais usadas, estas podem ser divididas em protocolos sem fios e com fios.

2.2.1. Com fios

- As redes **Ethernet** têm sido usadas amplamente como norma na maioria das redes locais(LANs) e redes amplas(WANs). A maioria dos fabricantes de computadores equipam os seus produtos com entradas Ethernet que podem ser posteriormente conectadas diretamente a redes de 10 Mb/s ou 100 Mb/s. Tornando assim muito fácil criar um novo nodo numa rede Ethernet. Outro beneficio da Ethernet é a capacidade de detetar coli-

sões na rede, uma colisão acontece quando dois ou mais nodos tentam enviar dados ao mesmo tempo. Não precisa de software e hardware muito especifico, providencia redes flexíveis e com uma qualidade de serviço suficiente para redes domésticas e de pequenas empresas [13].

- A tecnologia **X10** utiliza a instalação elétrica como meio de comunicação com pouca banda larga. Os dispositivos X10 enviam cerca de um comando por segundo e os comandos são bastante simples como por exemplo "Dispositivo A1: Ligar". Como uma rede *broadcast*, cada comando é enviado para todos os dispositivos da rede, cabendo a estes decidir se necessitam de responder ao comando. Para identificar os dispositivos individualmente, X10 usa um esquema que providencia até 256 endereços únicos. Para quem quer automatizar a sua residência a tecnologia X10 é conhecida por ser uma das mais fáceis de configurar, com preços bastante acessíveis e que aproveita a instalação elétrica já existente. No entanto esta tecnologia tem algumas deficiências, sendo uma das principais a interferência com entre dispositivos da rede. Outra das maiores fraquezas é a de não confirmar se um comando foi realmente executado. Para colmatar algumas dessas falhas foram desenvolvidas novas tecnologias como INSTEON em que para além da rede elétrica suporta comunicação RF. *Universal Protocol Bus* é outra que adiciona a possibilidade de comunicação bidirecional e com velocidade superior. Em [12] é deito uma analise comparativa de algumas tecnologias que usam rede elétrica.
- A tecnologia **Universal Plug and Play**(UPnP) [12] define uma arquitetura de conexão ponto-a-ponto para aplicações inteligentes, aparelhos sem fios e computadores. Foi desenhada para ser fácil de usar e flexível.

  A arquitetura de dispositivos UPnP é mais do que uma extensão do modelo Plug and Play(PnP). Foi desenhada para ser de configuração automática e permitir a descoberta de dispositivos automaticamente. Capacitando um dispositivo de se conectar dinamicamente à rede, obtendo o seu endereço IP, transmitir as suas características, e aprender sobre a presença e as características de outros dispositivos na rede.

2.2.2. Sem fios

- **Bluetooth**(http://www.bluetooth.com/) é uma tecnologia sem fios especifica para conexões de curta distancia que é simples de utilizar, segura e de baixo custo que pode

ser portátil e incluída em diversos dispositivos do dia a dia como telemóveis. Uma das características mais importantes para o seu sucesso é a capacidade de tratar simultaneamente da transferência de dados e voz. Isto permitiu uma nova variedade de soluções, uma das mais usadas é o dispositivo de mãos livres para chamadas telefónicas. O seu alcance mais usual é 10 metros mas pode variar até um limite de cerca de 100 metros.

- **Wi-Fi**, fisicamente é transmissão de sinais de rádio, no então é mais de que isso, provem do padrão IEEE 802.11 a/b/c e é destinada para redes locais sem fios (WLAN). Cada componente de uma WLAN requer um receptor de rádio e antena. Estes componentes podem ser AP ou Estações(STA). Uma STA são clientes de uma WLAN e podem corresponder a um computador, PDA, dispositivo USB, entre outros. Um AP forma uma ponte entre redes locais sem fio e redes locais com fio. Um *Basic Service Set* (BSS) é o termo que se utiliza para designar duas ou mais STA conectadas. Essa conexão pode ser configurada de duas maneiras principais, Ponto-a-Ponto ou Cliente/Servidor.
- O protocolo **ZigBee** [14] foi especificado com por completo em Dezembro de 2004 [3] com a colaboração de cerca de 70 empresas. O objetivo foi construir uma tecnologia de comunicação sem fios de baixo custo e baixo consumo com o uma velocidade transferência média. Para usar em electrónica, computadores, periféricos, controlo de ilumina-ção em residências, controlo industrial, sistemas de construção automáticos e monitorização de variáveis médicas. O seu design oferece uma grande durabilidade em termos de bateria, que é uma das características mais importantes numa rede *wireless*, a distancia de recepção do sinal é cerca de 100 metros, dependendo das condições do ambiente. Uma das características mais importantes do é a possibilidade de poder usar uma topologia em árvore ou em malha. Pois dá uma grande liberdade ao designer ao nível da aplicação, podendo assim ajustar melhor a solução ao problema, tirando partido das vantagens de cada tipo de topologia.

Comparação

Existem vários estudos que comparam os protocolos de comunicação [6], neste projeto apenas são referenciados alguns dos protocolos que pertencem à lista de mais usados em Inteligência Ambiente.

Um dos maiores problemas num sistema de comunicação é sem duvida a capacidade de fazer com que diferentes protocolos comuniquem entre si. A figura 6 [4] mostra a arquitetura de um

AmI como as diferentes redes de protocolos que podem ser combinadas através da tecnologia OSGi.

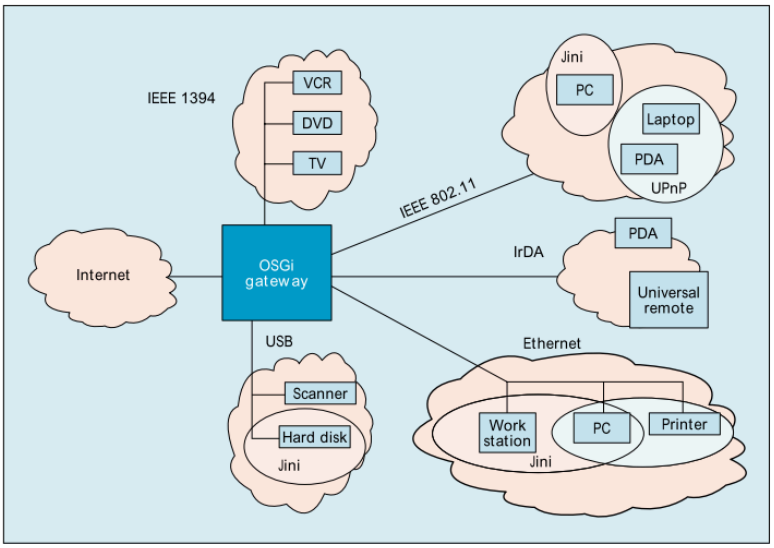


Figura 6 - Arquitetura OSGi

Como já foi referenciado neste projeto vão ser usadas as especificações OSGi que permite esta capacidade. Já foram desenvolvidos vários projetos que permitem construir pontes de comunicação com tecnologias como Bluetooth e UPnP [6].

2.3. Ambient Assisted Living

Por consequência do envelhecimento, doença ou acidente as pessoas perdem mobilidade, necessitando de alguém que supervisione ou ajude o que faz com que percam a sua autonomia. As rotinas físicas e mentais tornam-se complicadas o que influência negativamente as suas vidas.

O conceito principal de um *Ambient Assisted Living* (AAL) é implementar dispositivos e sistemas de baixo custo de operação. No entanto estes sistemas não vieram substituir o ser humano, servem apenas como apoio suplementar à sua assistência, facilitando e melhorando a qualidade das suas tarefas.

Com plataformas AAL podemos obter a supervisão e assistência a doentes nas suas atividades diárias, conseguindo assim melhorar a segurança, assegurar a saúde e o bem estar. Podemos dar às pessoas mas independência, autonomia, mobilidade e até mesmo dignidade e confiança, combatendo também o isolamento social e aumentar a sua segurança(Figura 7, *Austrian Institute of Technology*).

Figura 7 – Componentes de um *Ambient Assisted Living*

Os sistemas AAL oferecem também um imenso potencial melhorando a supervisão dos residentes, especialmente à luz de um aumento da importância de detecção precoce e cuidados preventivos [22].

O objetivo é conseguir a integração de dispositivos como aparelhos de ar condicionado, aquecedores, sistemas de iluminação, para combater todos os problemas referidos. O potencial destes sistemas e da tecnologia que apoia os cuidados de saúde em casa foi confirmado por uma variedade de projetos incluindo auto-monitorização e educação alimentar dos pacientes diabéticos [23], monitorização remota de pacientes, aconselhamento remoto para pacientes com insuficiência cardíaca congestiva [24] ou mesmo monitorização remota de arritmia cardíaca [25].

3. Tecnologias

Neste capítulo são abordados vários aspectos sobre as tecnologias para o desenvolvimento do projeto. Será abordado primeiramente projetos avançados dentro da área que podem comprovar que OSGi tem sido uma tecnologia de referencia para AmI. De seguida é especificado em mais detalhe o OSGi desde a seu objectivo, principais conceitos e funcionamento. São apresentados alguns conceitos sobre sensores e as suas características, mais especificamente serão apresentados os *Phidgets* que foram usados no caso de teste. Por fim são também expostas as tecnologias de desenvolvimento assim como as diversas ferramentas tecnológicas para a necessária concepção do projeto

3.1. Projetos

Na atualidade grandes representantes do mundo empresarial estão a adoptar OSGi reduzindo assim custos operacionais e manutenção. Empresas como Adobe, IBM, Oracle, VMWare e outras, têm feito investimentos significativos na tecnologia como membros da OSGi Alliance.

Diversos mercados estão a implementar e a usar a tecnologia OSGi, os principais mercados são: Empresarial, *Open Source*, Móvel , Telemática, Casa Inteligente e E-Health. O âmbito deste projeto abrange estes dois últimos mercados.

No mercado de casas inteligentes, são evidenciados de seguida dois projetos sustentáveis:

- O gestor de serviços *home gateway* da Cisco Modelo TES301 IP, é dum *home gateway* avançado, de alta performance, que combina as tecnologias de Ethernet, Voice over IP(VoIP), *router* e *Acess Point* wireless num único dispositivo. Fornece serviços de segurança e controlo de energia. Corre na plataforma OSGi ProSyst mBS Smart Home.

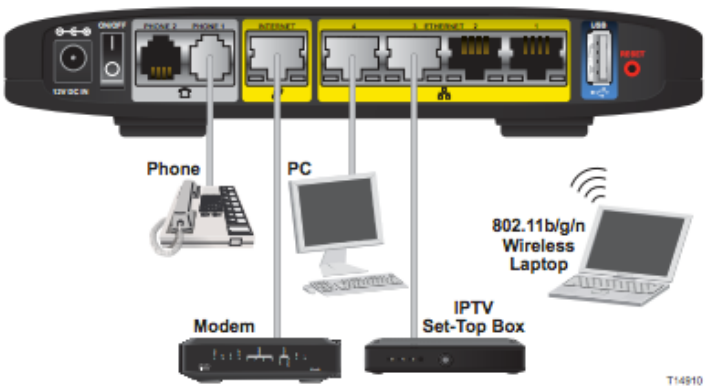


Figura 8 - Cisco TES301 IP

- AlerteMe(www.alertme.com) é um sistema distribuído que parte está em *cloud*(on-line), e outra parte está em casa, criando assim uma rede doméstica mais segura que conecta o utilizador à sua casa, eletricidade e dispositivos. Dando uma visibilidade em tempo real e controlo a qualquer hora. Em 2011 aderiu à tecnologia OSGi e é implementado na plataforma OSGi ProSyst mBS Smart Home.

Figura 9 - Dispositivos da AlerteMe

No mercado da E-Health existem também vários projetos, temos o exemplo de mais dois casos:

- O projeto AALuis(http://www.aaluis.eu/) é cofinanciado pela AAL *Join Programme*, BMVIT, *the Programm benefit*, *the* BMBF e ZonMw. Este é um projeto que traz interfaces inovadoras para o utilizador poder usufruir dos serviços num domínio AAL usando os benefícios de OSGi. A camada AALuis baseada em OSGi permite aos fornecedores de serviços e aos implementadores de interfaces facilmente integrar so-

luções novas e existentes de uma maneira normalizada. O objetivo do projeto é construir serviços, interfaces para utilizadores e uma camada de conectividade, para posteriormente integrar em plataformas *middleware* de AAL, baseadas em OSGi.

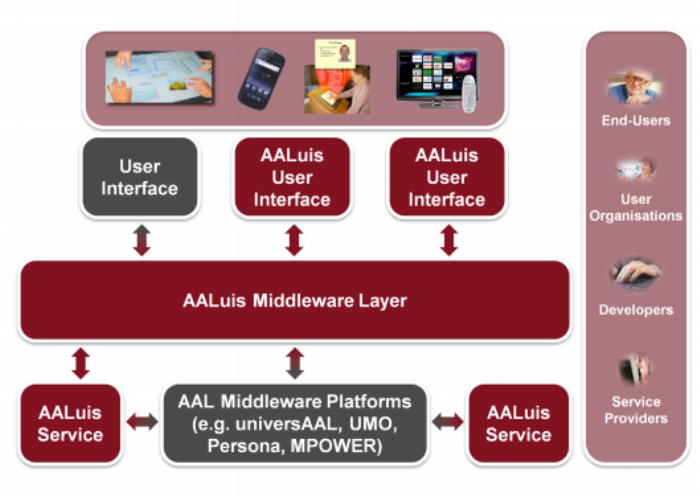


Figura 10 - Arquitetura AALuis

- Hearing@Home(http://www.hearing-at-home.eu/) é financiado pela comissão europeia no 6*th framework program*, focado nas necessidades dos deficientes auditivos em suas casas. Baseado na tecnologia OSGi dispositivos como computadores pessoais, sistemas hi-fi, câmaras digitais, telefones, faxes, serviço de internet, VoIP, entre outros tornam se acessíveis através da TV que está conectada a um PC ou a uma *power box* que implementa interfaces para os *gateways* da rede. Tornando assim a TV uma plataforma central de informação e comunicação de casa.

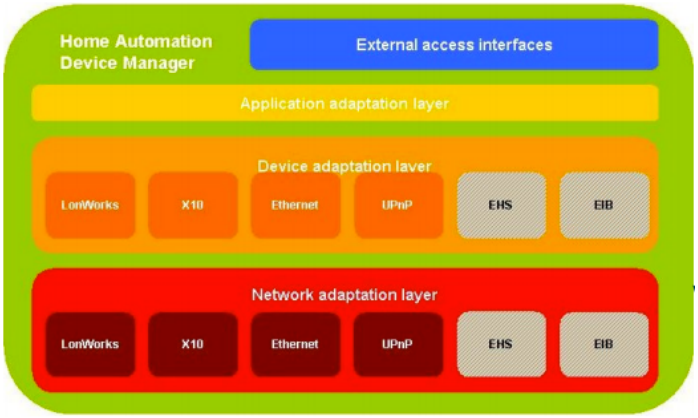


Figura 11 - Arquitetura Hearing@Home

3.2. Framework OSGi

A *framework* OSGi executa sobre Java e tem uma arquitetura orientada a serviços para *home gateways*, na *framework* é proposto a utilização de componentes de execução (*bundles*) que podem ser instalados, desinstalados e atualizados de um modo dinâmico e escalonável.

Um *bundle*, para além de disponibilizar serviços pode obter serviços de outro *bundle*, através do registo de serviços oferecido pela *framework*. Da mesma forma pode também exportar e importar classes Java. Existe um controlo por parte da *framework* que gere dependências entre *bundles* e serviços, proporcionando aos *bundles* um correto funcionamento. A *framework* separa a implementação de serviços da sua especificação, permitindo assim aos programadores uma maior independência no desenvolvimento dos serviços.

Uma das características mais importantes da *framework* é a de permitir a execução de *bundles* em tempo real através da *framework* de registo de serviços. Os *bundles* registam novos serviços, recebem notificações sobre o estado dos serviços e procuram serviços existentes para se adaptar às atuais capacidades do dispositivo. Esta característica, torna a torna a rede dinâmica e expansível facilmente. Bastante vantajoso pois o utilizador não necessita de reiniciar nem configurar o sistema após instalar novos *bundles*, novos serviços e atualizar *bundles* já existentes.

3.2.1. Bundles

Um *bundle* é uma unidade funcional com um ciclo de vida e com capacidade de carregar classes. Para além disso contém bem definidos os métodos que lhe permitem ser ligado à *framework*. Trata-se de um arquivo JAR que contem as classes e os recursos que podem ser imagens, páginas HTML, entre outros. Como já foi referenciado, um *bundle* contem um conjunto de serviços. Pode conter, por exemplo, o serviço “DigitalPlayer” assim como ficheiros MP3 para reproduzir. Outro que controla a temperatura pode conter também um conjunto com páginas HTML e imagens que servem de interface. A figura 12 [35] apresenta um exemplo de ficheiros que podem constituir um *bundle*.

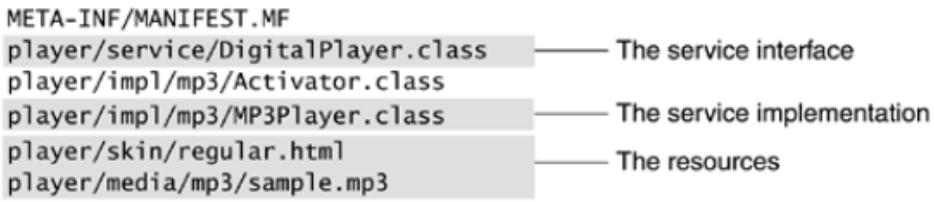


Figura 12 - Conteúdo de um *bundle*

Um *bundle* no seu ciclo de vida pode assumir o seu estado como instalado, iniciado, atualizado, parado e desinstalado. A figura 13 [34] mostra as possíveis transições entre estados de um *bundle*.

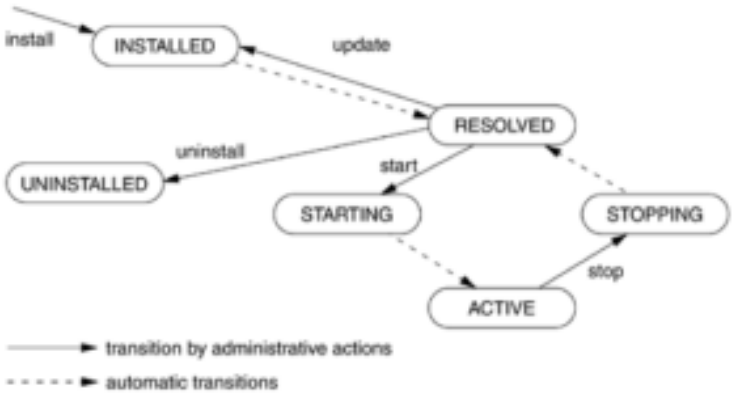


Figura 13 - Transições de estados de um *bundle*

Cada *bundle* contem um padrão de interface já definido, de forma a possibilitar a *framework* de entender e instalar o *bundle*. O "manifesto" é um ficheiro padrão de entrada no arquivo JAR. Este inclui meta-informações sobre o ficheiro, de forma padronizada, com o objetivo da *framework* ser capaz de processar o *bundle* corretamente, por exemplo, o *bundle* exporta as classes do pacote *player.service* e necessita de importar o pacote *http.service* que deve ser exportado por outro *bundle*.

O "ativador do *bundle*" é executado apenas quando o *bundle* é iniciado ou parado, e implementa o método start e stop. Quando o *bundle* é inicializado o *framework* analisa o cabeçalho do manifesto . Se o *bundle* necessita de importar pacotes, a *framework* procura o pacote em todos os *bundles* com o estado de resolvido e de seguida exporta os seus pacotes para o *framework*. Esta por si cria um objeto "BundleContext" para o *bundle* e executa o método START do "ativador do

bundle" e passa o objecto criado como argumento. E é no método *start* que o *bundle* deve registar todos os serviços que presta. O *bundle* entra no estado de iniciado e em caso de sucesso da ativação o *bundle* passa para o estado de ativo e um outro evento notifica os observadores interessados que o *bundle* foi iniciado.

3.2.2. Serviços

Para implementar um serviço, é necessário definir uma interface que diz o que faz o serviço, e as correspondentes classes que definem como o serviço é realmente executado. Existe um separação da interface e da implementação, garantindo assim que a interface do serviço se mantenha estável, mesmo que a implementação sofra alterações. Por exemplo, uma interface do serviço "DigitalPlayer" poderá ser usada por uma classe que implemente um mp3, e continuar a funcionar se for trocada por uma classe que implemente uma aparelhagem.

Os serviços em OSGi são especificados por uma interface Java juntamente com um documento detalhado sobre a semântica da interface. Nas diversas plataformas são oferecidos serviços dos mais importantes destacam-se:

- Serviço de Log: tem como objetivo registar erros, informações, eventos, entre outras informações para posterior analise.
- Serviço HTTP: oferece a possibilidade de serviços serem acedidos através da Internet com diversos com standards como HTML, XML e Servlets.
- *Device Acess*: Modelo para gerir dispositivos e *drivers* externas.
- Gestor de configurações: serve para gerir *bundles* remotos e locais.
- *User Management*: gere a autenticação de utilizadores e atividades sobre a plataforma de serviços.

3.2.3. Plataformas

Diversas implementações estão presentes no mercado, algumas open-source. As serão apresentadas apenas as mais importantes.

- Apache Felix(http://felix.apache.org), é open-source e implementa a as especificações da plataforma OSGi R4 e outras tecnologias relacionadas, sobre a licença da Apache. Esta plataforma está organizada em subprojectos, cada um centrado no desenvolvimento de um serviço ou especificação em concreto.

- Equinox(http://eclipse.org/equinox) é um subprojecto da Eclipse implementa as especificações OSGi R4. É open-source e disponibiliza diversos *bundles*.
- Knoplerfish(https://www.knopflerfish.org) é open-source no entanto existe uma versão paga de nome Knoplerfish Pro. É desenvolvido pela Makewave e a última versão é o Knoplerfish 3 que implementa especificações OSGi R4 v4.2. Esta versão fornece um extenso de *bundles* e componentes.
- ProSyst(http://www.prosyst.com) é uma plataforma não gratuita, mas que oferece produtos e serviços para os mais variados mercados da tecnologia OSGi, como *smartphones*, Casa Inteligente e aplicações industriais.

3.2.4. Bnd

Bnd é intitula pelo seu criador, Peter Kriens, o "canivete suíço" do OSGi. É usado para criar e trabalhar com *bundles* OSGi, o seu principal objetivo é facilitar o desenvolvimento desses mesmos *bundles*.

Em OSGi é obrigatório ter em cada *bundle* um ficheiro *manifest* para verificar a consistência do "*class path*". O bnd facilita o processo de criação deste *manifest*, e elimina possíveis redundâncias automaticamente. O seu funcionamento centra-se em analisar os ficheiros de classe existentes e descobrir as suas dependências. Estas dependências são depois fundidas com outras instruções fornecidas pelo utilizador.

3.3. Sensores

A primeira década do século XXI foi etiquetada com o nome "Década de Sensores". Com o aumento das investigações e aplicações depois da década de 90 surgiu uma revolução similar à experienciada com os microprocessadores na década de 80(Jon S. Wilson X). No nosso dia a dia vivemos rodeados de sensores sem por vezes nos apercebermos, um *smartphone* usual contem cerca de 14 sensores como podemos verificar na figura 14 [36].

Figura 14 - Capacidade sensorial de um *smartphone* usual

Sensores são dispositivos sofisticados que são usados frequentemente para detectar e responder a sinais elétricos. Convertem um fenómeno físico(temperatura, pressão sanguínea, humidade, velocidade, etc) num sinal elétrico. Como tal, os sensores representam parte da interface entre o mundo físico e o mundo dos dispositivos electrónicos. Um exemplo pode ser o da temperatura, o mercúrio no vidro do termómetro expande e contrai o liquido para converter a temperatura medida e posteriormente poder ser lida num tubo de vidro calibrado. A outra parte da interface é representada através dos actuadores, que convertem sinais elétricos para um fenómeno físico. (Dr. Tom Kenny, Stanford). O fenómeno físico pode ser executado por nós humanos ou detectado através do ambiente como visto na figura 15.

Figura 15 - Interação com sensores

Nos sensores é importante ter em atenção diversas características de forma a sabermos se é fiável e se corresponde às necessidades. Algumas das principais são a precisão, a condição ambiental(pois existem valores limite como por exemplo para a temperatura e humidade), o alcance, a calibração(essencial para a maioria dos dispositivos de medição), resolução(a mudança mais pequena detectada) e a repetição.

3.3.1. Características

Os sensores podem ser caracterizados de diversas formas, como industriais ou não industriais, como ativos ou passivos e pelas suas diversas propriedades. A caracterização por propriedade é talvez a mais interessante para se poder entender os as suas funcionalidades. Podem ser caracterizados como sensores de:

- **Presença:** trabalham com a emissão e recepção de luz, detectando qualquer tipo de material sem que haja contacto entre eles.
- **Posição:** Conseguem detectar a posição física de um objecto.
- **Velocidade:** São usados em dispositivos como leitores de CD-ROM, DVD *Players*, robótica, maquinas automáticas de soldagem, entre outros.
- **Aceleração:** Este tipo de sensores(acelerómetros e giroscópios) fornecem sinal elétrico proporcional à aceleração do sistema. São componentes do tipo inercial e indicam o movimento do sistema através de uma variável de eixo inercial.
- **Temperatura:** Mais utilizados em industrias, veículos, eletrodomésticos, e edifícios.
- **Vazão:** Representam a quantidade de líquidos, gases ou vapores que passam em um determinado ponto, durante um determinado período de tempo.
- **Tensão, Corrente e Potencia:** Utilizados para medir grandezas elétricas.
- **Humidade, Gases e pH:** Medição de humidade, gases, detectores de incendio e pH.

Entre estes existem outros tipos de sensores tais como de pressão , ópticos, de nível, tensão , corrente e potencia.

3.3.2. Aplicação em Inteligência Ambiente

A camada se sensores é uma peça da estrutura essencial para o funcionamento da AmI. Os sensores podem ser usados para disponibilizar diversos serviços capazes de fazer parte de dis-

positivos inteligentes que se adequam a necessidade das pessoas. O projeto [6] apresenta alguns desses dispositivos tais como:

- **Cama inteligente:** A cama tem um equipamento especial capaz de monitorizar o sono dos ocupantes, definir um padrão e manter o registo de noites mal dormidas.
- **Chão inteligente:** Chão com sensores de pressão capazes de determinar a posição dos seus ocupantes.
- **Porta da frente Inteligente:** Inclui uma *tag* RFID que é identificada pelo leitor para residentes autorizados e sem chave. Contem também um sistema com microfone, câmara , colunas, trinque elétrico e abertura automática.
- **Persianas Inteligentes:** Podem ser configuradas e ajustadas por um dispositivo remotamente para providenciar privacidade e quantidade de luz ambiente desejada.
- **Casa de banho inteligente:** Inclui um sensor para o papel higiénico, um detector de descarga e chuveiro que regula a temperatura da água para evitar escaldões.
- **Detector de fugas inteligente:** Sensor que detecta fugas de água de aparelhos como a maquina de lavar loiça e maquina de lavar roupa.

Entre estes muitos outros dispositivos podem usufruir das diversas propriedades dos sensores, dispositivos estes que podem ser adaptáveis quer no contexto de Inteligência Ambiente, quer em *Ambient Assisted Living*.

3.3.3. Phidgets

Para os nossos casos de teste usamos *Phidgets. Phidgets* são um conjunto de "*building-blocks*" de baixo custo para monitorização e controlo a partir de um dispositivo ligado por USB e suporta diversas linguagens de programação.

Um dos *Phidgets* usados no caso de teste é o Interface Kit 8/8/8, um *Phidget* flexível com I/O(input/output) e capacidade de ler e controlar sensores analógicos(figura 16).

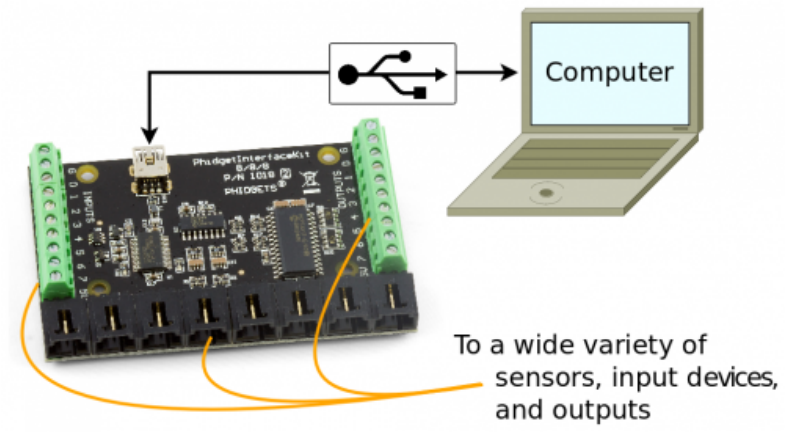


Figura 16 - Interface Kit 8/8/8

Os sensores analógicos usados foram:

- **Precisão de temperatura:** Varia entre -30°C até 80°C, com erro de 0.75°C em temperaturas positivas.
- **Precisão de luz:** Mede o nível de luz perceptível por um ser humano em lux. Varia entre 1 lux até 1000 lux (Intensidade de luz de um estúdio de televisão).
- **Vibração:** Mede a intensidade de vibração e seu valor varia entre 0 e 1000.
- **Rotação:** Pode ser rodado 300 graus e o seu valor varia entre 0 1000, baseando se na posição do eixo.
- ***Slider*:** É um *slider* de 60mm e o seu valor varia de 0 a 1000.
- **Toque:** Consegue detectar toque através do plástico, vidro ou papel, com espessura recomendável de 1/8".
- **Presença :** Detecta presença de um objecto altamente reflectido a 10 cm, para objetos menos refletíeis, como mãos, a distancia é cerca de 5 cm.

Outro dos *Phidgets* usados é um leitor de *tags* RFID. O *Phidget* usa o protocolo de comunicação EM4102 e consegue ler todas as *tags* que usem o mesmo protocolo. Para ser possível a leitura a distancia mínima necessária entre a *tag* e o leitor varia muito dependendo do tipo de *tag*, mas nas utilizadas varia entre 3"- 4"(figura 17).

Figura 17 - RFID Phidget

Por último o usamos também o *Phidget* AdvancedServo 8-Motor que permite controlar a posição, velocidade e aceleração de oito pequenos motores RC servo. Este *Phidget* requer uma fonte de energia externa de 6-15VDC e liga diretamente por USB a um computador(figura 18).

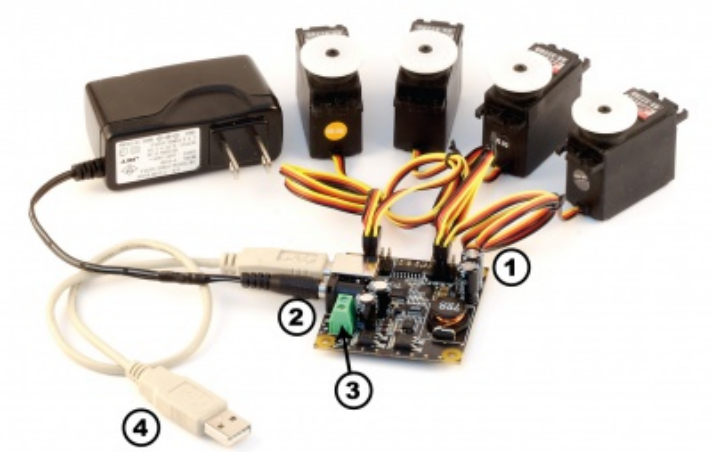


Figura 18 - Advanced Servo Phidget

Para mais informações relativas aos *Phidgets* usados como o seu alcance, consumo, resolução e outras características mais especificas, podem ser consultadas no Anexo I.

3.4. Jetty

No projeto é utilizado o servidor Web Jetty que pode ser designado como *Server-side* Java. Esta é uma tecnologia que utiliza maquina virtual Java para responder a pedidos da Web. É um subcon-

junto da tecnologia Java 2 Enterprise Edition(J2EE). Os servidores que correm Java são conhecidos como "*servlet containers*" ou "servidores Java".

Jetty providencia um servidor HTTP e um *servlet container* desenvolvido em Java capaz de dar resposta a conteúdo dinâmico e estático.
É um projeto *open source* e é considerado um dos servidores Java mais rápidos [19]. A partir da versão 7 o servidor passou a pertencer à Eclipse Foundations. O servidor oferece características como [http://www.eclipse.org/jetty]:

- Servidor HTTP assíncrono
- Padrões baseados em *servlet containers*
- Servidor de WebSockets
- Servidor SPDY(*pronounced speedy*)
- Cliente HTTP assíncrono
- OSGi, JNDI, JMX, JASPI.

É usado por vários projectos importantes, grandes *clusters* como o Yahoo Hadoop Cluster, *cloud computing* como a Google App Engine, SaaS(*Software as a Service*) como o Yahoo!Zimbra e servidores como o Apache Geronimo.

3.5. Java

Java é uma linguagem orientada a objetos baseada em classes. Desenhada para ser simples o suficiente para que muitos programadores a consigam perceber rapidamente [26].

Class loading é uma característica de referencia da plataforma Java. Permite que processos Java carreguem código da aplicação e do sistema em tempo de execução das mais variadas fontes. Existem dois tipos de *class loaders,* os definidos pelo utilizador e o *bootstrap class loader* da Java Virtual Machine(JVM). Todos que são definidos pelo utilizador são subclasses de `java.lang.ClassLoader`, classe que contem métodos que permitem encontrar, executar e disponibilizar as ligação para as classes necessárias. Os *class loaders* organizam-se de forma hierárquica, em que o *bootstrap class loader* é a raiz da árvore, como apresentado na figura 19 [27].

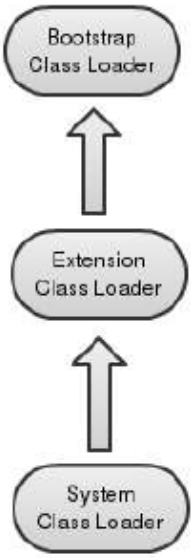


Figura 19 - Carregamento de classes em Java

As especificações OSGi tentam tornar todo o processo de carregamento de classes mais dinâmico usando uma estrutura abordada nas secções 3.2.1 e 4.3.

Para o desenvolvimento Java fui utilizado o IDE Eclipse, para ajudar o desenvolvimento de *bundles* foi utilizada a ferramenta do eclipse bndtools baseada no conceito de bnd e para o desenvolvimento Web foi usado *JavaServer Pages*.

Bndtools

Bndtools é um ambiente de desenvolvimento para eclipse que oferece funcionalidades muito úteis para o desenvolvimento de *bundles*, é baseado no conceito bnd descrito na secção 3.2.4 e foi criada por Peter Kriens. Têm funcionalidades como:

- **Análise de importações:** É usada uma analise *bytecode* para calcular com precisão as dependências de cada *bundle*.
- **Repositórios:** Contem repositórios para *bundles* que podem ser referenciados em tempo de compilação e também usados para resolver dependências em tempo de execução.
- **Versão Semântica:** Durante o lançamento os bundles são comprados com a sua anterior base procurando mudanças, calculando assim automaticamente as suas verões.
- **Compilador instantâneo:** Ao contrario do eclipse que compila o código Java quando este é salvo, bndtools reúne automaticamente os *bundles* quando o seu *input* se modifica. Assim os *bundles* estão sempre na sua última versão e prontos para correr.

- **Testes integrados:** É utilizada uma *framework* integrada que contem a seleção de *bundles*, executa os casos testes declarados em cada *bundle*. Todo esse processo em tempo bastante rápido, os resultados são apresentados na janela JUnit do Eclipse.

JavaServer Pages

Servlets é uma tecnologia Java desenvolvida para dar resposta à programação para *Common Gateway Interface*(CGI), programas que executam num servidor Web que pertencem a uma cada entre os pedidos vindos do browser ou outro cliente HTTP e base de dados ou aplicações no servidor HTTP.

A tecnologia JavaServer Pages(JSP) permite juntar HTML estático com conteúdo dinâmico dos *servlets* Java. A abordagem usada em JavaServer Pages oferece várias vantagens em relação às tecnologias concorrentes tais como ASP, PHP ou Cold Fusion. Estas podem ser consultadas mais em detalhe em [28], mas andam em torno de dois factos, que JSP é extremamente apoiado e portanto não causa entraves para determinados sistemas operativos ou servidores Web e que JSP dá acesso total a *servlets* e à tecnologia Java para a parte dinâmica, excluindo a obrigação de ter de usar uma tecnologia diferente e por vezes mais fraca para um propósito especial.

O processo de disponibilizar as páginas JSP na Web é um processo mais simples do que para os *servlets.* Embora o código pareça mais HTML do que *servlet*, no servidor o que acontece é que as páginas JSP são automaticamente convertidas para *servlets* com o HTML estático imprimido para o *output stream* associado ao método `service` do *servlet,* o processo de conversão está representado na figura 20.

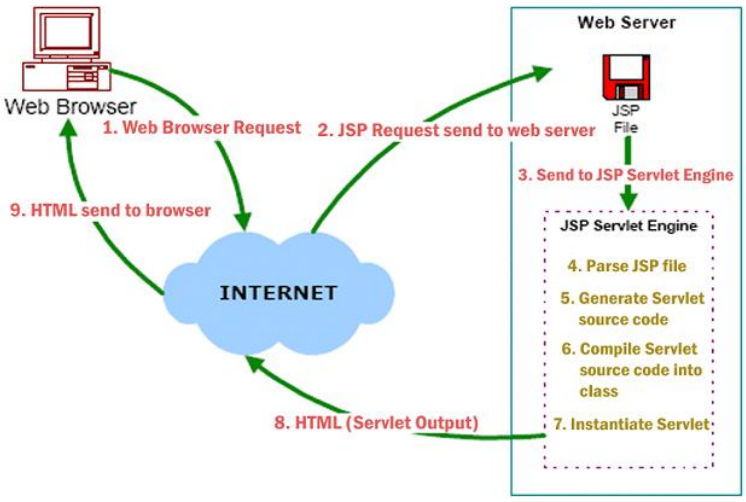


Figura 20 - Primeiro acesso a página JSP

Esta conversão é normalmente executada apenas da primeira vez que a página é acedida, para assegurar que o primeiro utilizador real não tem de esperar demasiado, basta que os programadores acedam uma primeira vez antes dos utilizadores reais para que possa ser instanciada.

3.6. Síntese

Neste projeto são utilizadas tecnologias diversificadas no entanto a mais importante de abordar é OSGi. Trata-se de especificações capazes de instalar e desinstalar módulos de modo dinâmico. Estes módulos são conhecidos como *bundles* que têm a possibilidade de registar serviços que posteriormente poderão ser usados por outros *bundles*.

Sensores estão cada vez mais ao nosso redor e podem ser caracterizados pelo tipo de informação que obtêm assim como pelas suas características mais técnicas. No projeto foram utilizados *Phidgets,* pequenos sensores que ligados por USB que fornecem informação sobre o ambiente.

Para a implementação do projeto foi usada a tecnologia Java pois é a marca registada segundo as especificações OSGi. Para todo o desenvolvimento de *bundles* e testes na plataforma foi usada a ferramenta bndtools que agiliza todo o processo pois tem base a tecnologia bnd. A parte Web foi desenvolvida usando JavaServer Pages que possibilita o desenvolvimento do páginas dinâmicas juntando a linguagem Java com HTLM estático. A parte Web está inserida em um servidor Jetty que suporta *sevlets*. Este é referenciado como bastante rápido e um forte concorrente ao Tomcat.

4. BSense

Para responder aos desafios sociais e tecnológicos descritos na secção 1.2 , foi elaborado o projeto de nome "BSense". Uma plataforma integradora de serviços em tempo real que utiliza como base OSGi, uma tecnologia emergente em AmI.

O projeto irá apresentar uma arquitetura externa com base na estrutura da figura 21.

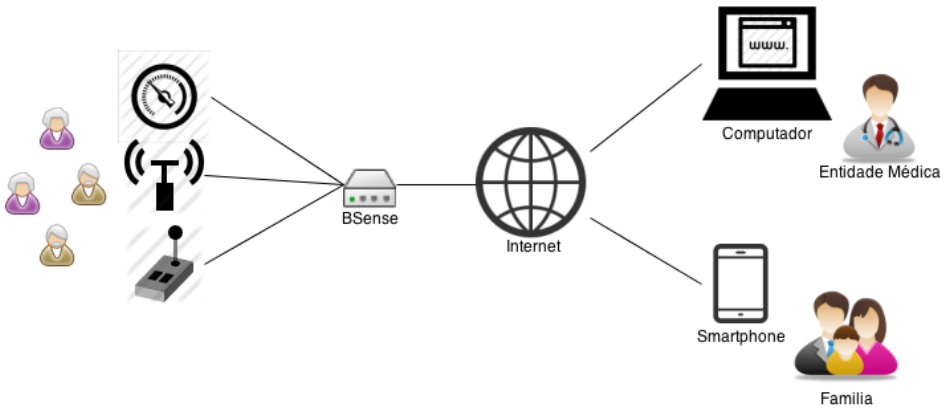


Figura 21 - Arquitetura externa da plataforma BSense

Como se pode verificar no esquema a plataforma é inserida num *home gatway* com ligação à internet, esse sistema é desenvolvido segundo as especificações OSGi. São instalados *bundles* capazes de integrar os diversos sensores no sistema. A plataforma tem integrada uma aplicação Web capaz de providenciar uma interface que pode ser acedida através da internet.

Neste capítulo é explicada toda a estrutura do projeto começando por mostrar a sua arquitetura interna e depois explicando todas as camadas que a constituem.

4.1. Arquitetura

Para o desenvolvimento da arquitetura do projeto BSense foi tido em conta todas as características da tecnologia OSGi faladas na secção 3.2, e houve a preocupação te tornar o sistema modular para futuras alterações. Assim sendo a arquitetura do projeto pode ser divida em 5 camadas principais como está representado na imagem 22.

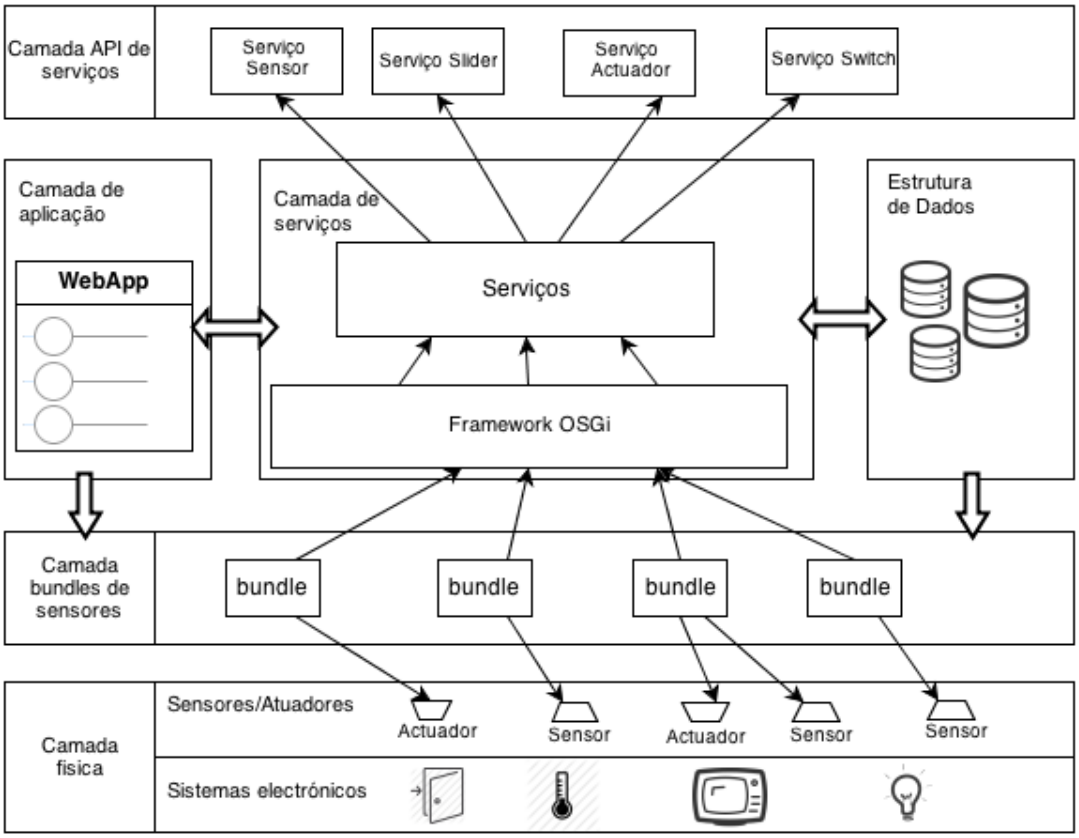


Figura 22 - Arquitetura interna da plataforma BSense

A Camada física é mais ligada ao hardware do que ao desenvolvimento no entanto o desenvolvimento na camada de *bundles* é dependente da camada física. A camada *bundles* Sensores é instalada na *framework* OSGi onde os *bundles* são ativados e registam os seus serviços. Os serviços devem-se registar usando uma das APIs de Serviço disponibilizadas pela plataforma. Deste modo camada de aplicação é capaz de saber quando são registados os serviços da Camada API e assim criar automaticamente uma interface dinâmica para o utilizador. A estrutura de dados representa o registo dos estados dos sensores em determinado momento. Todas as camadas serão explicadas mais em detalhe nas secções seguintes.

4.2. Camada Física

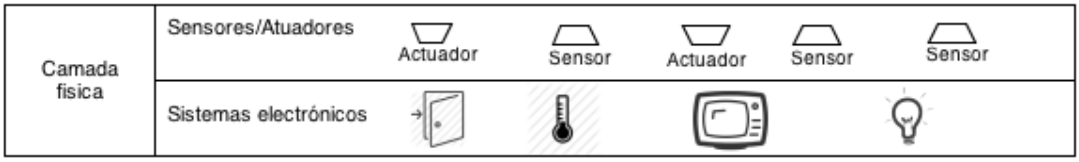


Figura 23 - Camada física do sistema BSense

Esta camada diz respeito aos componentes de hardware do projeto, e pode ser dividida em duas partes Dispositivos e Sensores/Actuadores.

A parte relacionada com sistemas electrónicos descreve tecnologia avançada nos dispositivos do dia a dia que são equipados com sensores, actuadores, memoria e capacidade de comunicação. Portanto são capazes de capturar informação à sua volta, comunicando uns com os outros e reagindo de acordo com regras previamente definidas [22]. Através da sua capacidade de interagir com pessoas diretamente

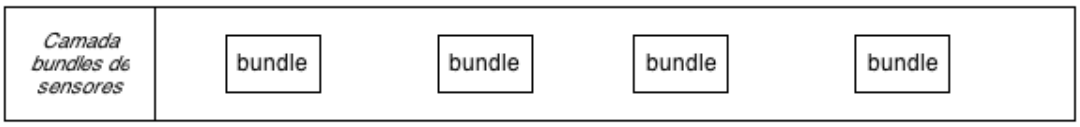
A Almofada Inteligente [29], por exemplo, é um sistema de monitorização na forma de uma almofada tradicional, que verifica os parâmetros vitais do utilizador assim como a sua respiração, pulso e temperatura corporal e em caso de emergência ou adoecimento pode notificar automaticamente uma entidade medica. Outro exemplo interessante é o Sofá Inteligente [30] é um sistema de monitorização avançado capaz de identificar a pessoa que está sentada e oferece serviços personalizados com base nesta informação.

Num contexto mais díspar podemos encontrar Loiça Inteligente [31] que é um conjunto de dispositivos de cozinha desenvolvidos pelo instituto de tecnologia de Massachusetts, inclui por exemplo uma panela inteligente capaz de determinar se está demasiado quente para ser tocada, uma colher capaz de medir a temperatura e a viscosidade da comida, uma chaleira que determina o tempo restante de espera para o chá ficar pronto [32].

A parte superior desta camada é referente aos sensores/actuadores desses sistemas electrónicos estão representados na parte superior da camada física. Estes são integrados nos dispositi-

vos da camada anterior de modo aos dispositivos conseguirem perceber e atuar no ambiente que se inserem. Pode ser dado o exemplo de um sistema de abrir porta automaticamente que, por exemplo, deve ter um sensor de presença de detecta quando o utilizador chega assim como deve ter um actuador capaz de abrir a porta.

Esta camada não exige desenvolvimento especifico para integrar a plataforma mas no entanto a camada superior, *bundles* de sensores, é desenvolvida de acordo com as especificações de cada sistema electrónico.



4.3. Camada bundles de sensores

Figura 24 - Camada bundle de sensores do sistema BSense

Para todos os sistemas electrónicos devem ser desenvolvidos *bundles* capazes de tirar partido de todas as funcionalidades do sistema e expor essas funcionalidades na plataforma através do registo de serviços. Pegando no exemplo de uma *Smart* TV, os *bundles* implementados podem disponibilizar serviços como o de aumentar o volume, escolher canal e ligar/desligar. Num exemplo do sistema de abrir porta automaticamente, deve ser disponibilizado o serviço de "presença de individuo" e serviço um serviço "abrir porta".

Um dos maiores benefícios da plataforma é que o *bundle* pode ser instalado sem ser preciso reinicializar o sistema. Para isso é necessário recorrer ao carregamento de classes em tempo real do OSGi. Cada bundle instalado na plataforma não pode ter associado um *class loader* até passar para o estado de resolvido. Depois de entrar no estado de resolvido a framework OSGi cria um class loader para cada *bundle*.

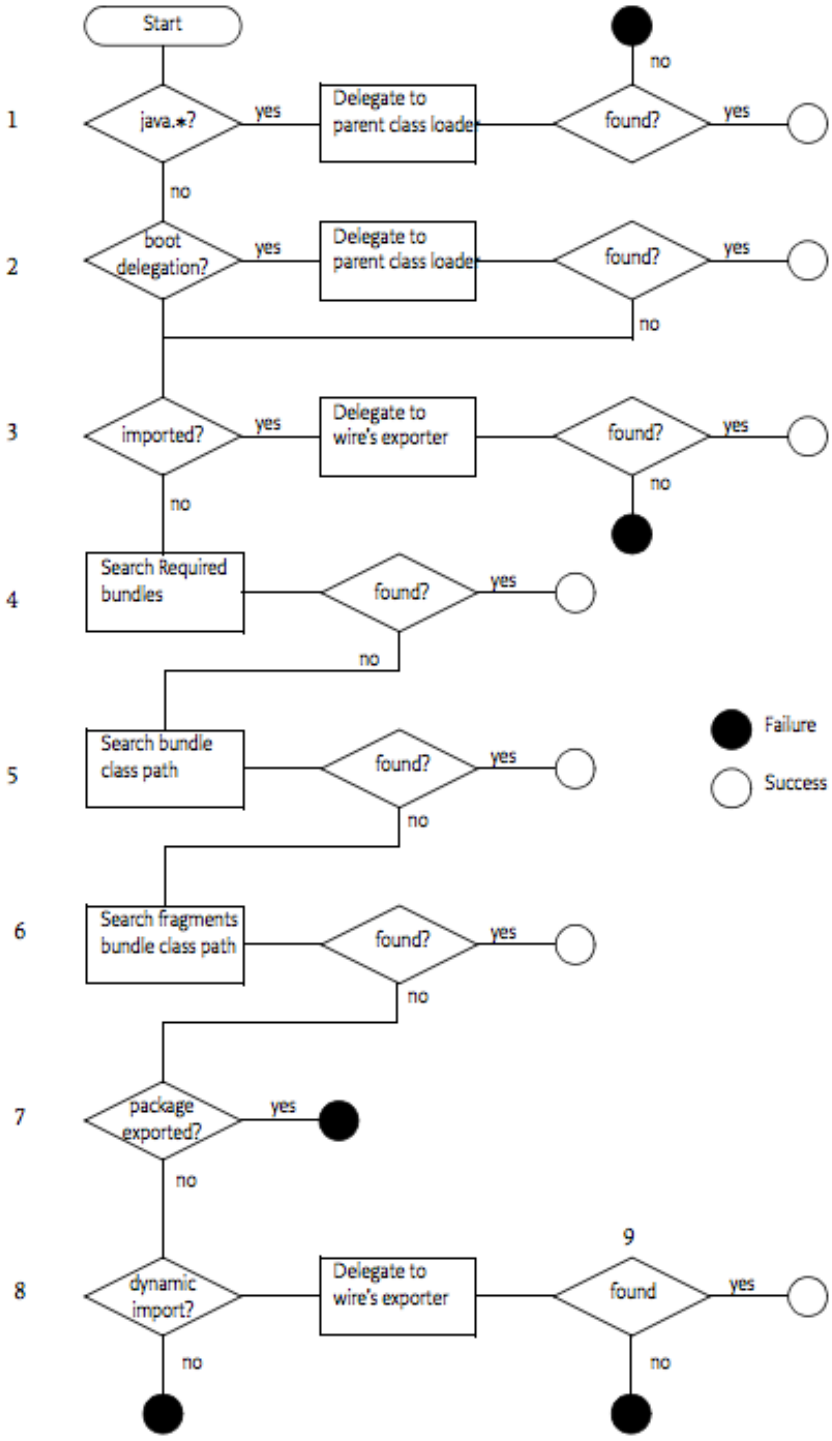


Figura 25 - Fluxograma de carregamento de classes em tempo real

Na figura 25 pode ser visualizado o fluxograma executado quando um *bundle* é iniciado. Através da procura dinâmica de classes consegue carregar módulos em tempo real assim como integrar serviços.

O desenvolvimento de cada *bundle* é totalmente independente em relação à sua integração na plataforma, no entanto o desenvolvimento dos seus serviços devem ter em conta a camada API de serviços se desejar que a plataforma Web ser capaz de gerar interface gráfica dinamicamente. O seu *manifest* que, entre outras funcionalidades, serve para dar a conhecer todas as dependências e exportações, deve conter informação adicional para que a camada de aplicação consiga distinguir os *bundles* referentes a sistemas electrónicos/dispositivos dos *bundles* referentes a configurações, resolução de dependências, entre outros. Essa distinção é feita adicionando apenas uma linha como está representado na figura 26.

```
-buildpath: osgi.core,\
    phidget21bnd,\
    org.device.switch.api;version=latest,\
    osgi.cmpn,\
    org.action.api;version=latest
Bundle-Activator: org.RFIDPh.impl.Activator
Private-Package: org.RFIDPh.services,\
    org.RFIDPh.impl,\
    org.RFIDPh.listeners
Device: RFIDPhidget
```

Figura 26 - Manifest de RFID Phidget para o sistema BSense

Como se pode ver no *manifest* apresentado para além das normais definições é adicionada a linha "Device: Nome do Sistema/dispositivo", permitindo assim a sua integração na camada de aplicação. Informação em detalhe sobre o funcionamento de *bundles*, pode ser encontrada na secção 3.2.1.

É muito importante também que o programador do *bundle* usufruía da capacidade da plataforma integrar serviços em tempo de execução. Neste contexto se existir a capacidade de detectar através de *listners* o evento de ligar e de desligar, é importante manter registados os serviços disponibilizados pelo *bundle*, apenas se o dispositivo estiver ligado.

4.4. Camada de serviços

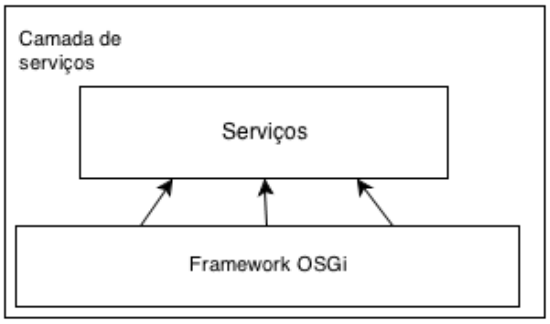


Figura 27 - Camada de serviços do sistema BSense

A camada de serviços é o core do sistema OSGi onde os *bundles* são instalados e os seus serviços registados.

Aqui não estão presentes só os *bundles* referentes aos sensores mas também muitos outros que permitem o correto funcionamento de todo o sistema. Podem ser instalados os mais diversos *bundles* incluindo algumas bibliotecas para o bom funcionamento de outros *bundles* capazes de resolver falta de dependências.

Na figura 28 [34] é representado o diagrama de classes referente à camada de serviço do OSGi. Para registar os seus serviços o `BundleContex` serve de ponte entre o *bundle* desenvolvido e a framework. O *bundle* deve comunicar através do `BundleContext` que serviços quer registar. O `BundleContex` trata de interagir com as classes responsáveis da plataforma para registar o serviço pedido.

A *framework* OSGi escolhida foi Apache Felix, pois para além de providenciar muitos dos componentes necessários para o desenvolvimento disponibiliza uma boa diversidade de projetos *bundle* capazes de permitir a ponte entre tecnologias diferentes e serviços bastante úteis. É uma *framework* estável que oferece um bom suporte tecnológico. Além disso possibilita uma fácil integração com o IDE Eclipse e com a ferramenta bndtools explicada no capítulo 3.

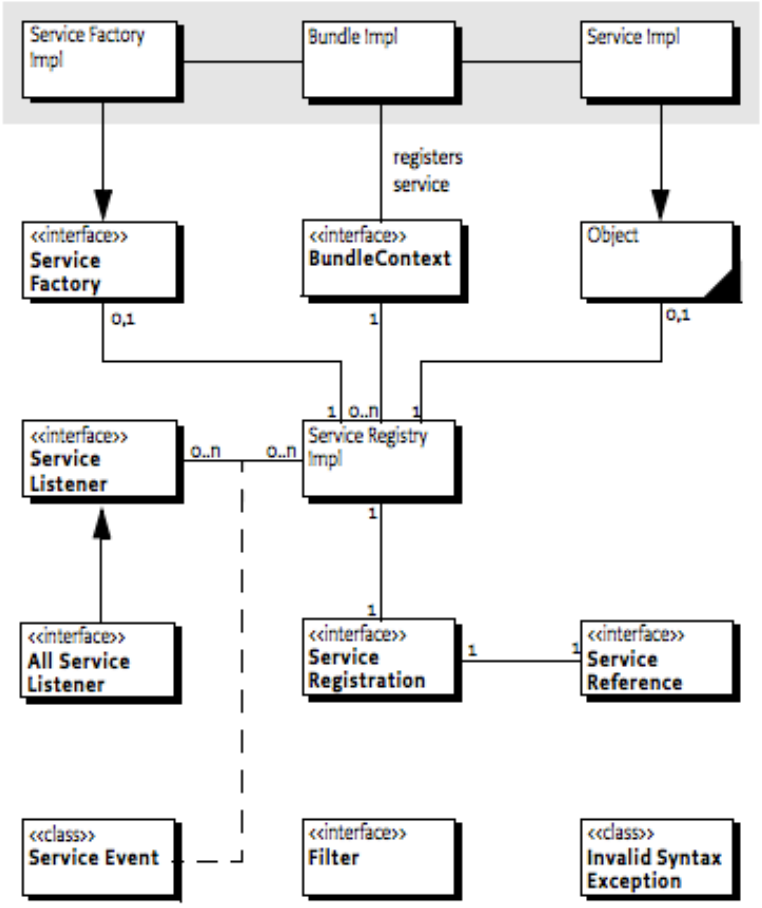


Figura 28 - Diagrama de classes da camada de serviços do OSGi

No entanto o serviço HTTP que a framework Apache Felix disponibilizava foi substituído por uma extensão com o nome Pax Web que disponibilizou *bundles* importantes para a plataforma desenvolvida, como o servidor aplicacional Jetty e suporte JSP. Pax Web está a crescer cada vez mais tendo apenas lançado a versão 0.2.2 em 2009 em 2012 já lançaram a 3.

Para os serviços registados na *framework* poderem ser usados pela camada de aplicação terão obrigatoriamente que implementar uma interface correspondente da camada API de serviços.

4.5. Camada API de serviços

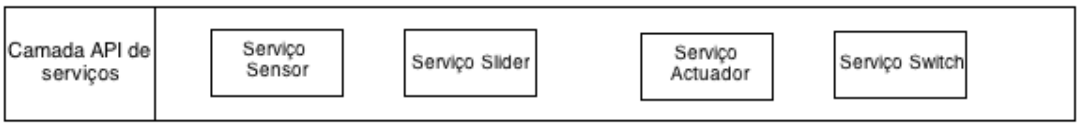


Figura 29 - Camada API de serviços do OSGi

Esta camada é responsável por servir como um guia para a camada de aplicação perceber qual o tipo de serviço e que ele realmente faz. Cada uma das APIs é um *bundle* especial que fornece uma interface para os serviços poderem implementar. O serviço que implementar determinada interface terá obrigatoriamente de conter os métodos forçados pela mesma.

A aplicação Web reconhece os tipos de serviços e dependendo do seu tipo gera a sua interface Web dinamicamente, permitindo assim uma plataforma homogénea mesmo para os diferentes tipos de dispositivos.

Entre estas APIs podemos ter dois tipos, aquelas que são implementadas por serviços em que o estado do sensor apenas muda com a interação na plataforma Web(ou no próprio dispositivo), e aquelas que mudam de estado quando a interação é feita não pela plataforma mas por outro serviço. Para uma melhor explicação podemos olhar para o diagrama da figura 30.

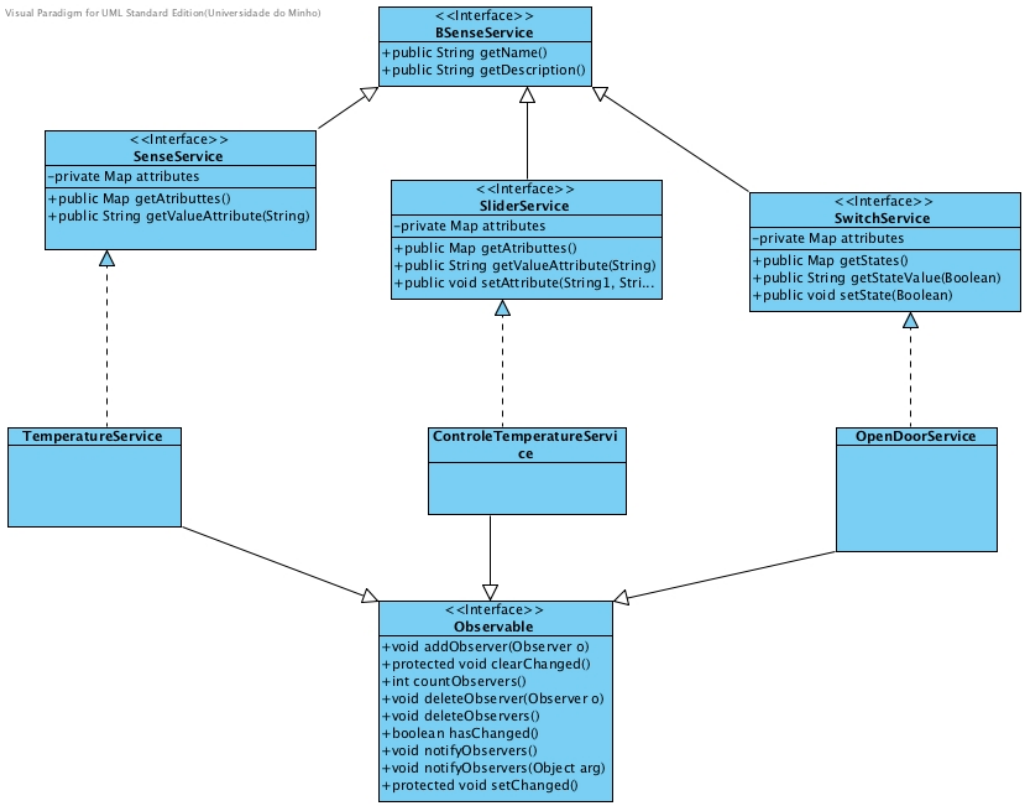


Figura 30 - Diagrama de classes da camada API de serviços

Estas interfaces representam três diferentes APIs de sensores que a interface vai reconhecer e gerar a sua interface. Na API SenseService a plataforma vai gerar uma tabela que mostra os valores dos atributos dos sensores.

Service	Attribute	Value
Analog Senses		
	Vibration	0
	Temperature	28.0ºC

Figura 31 – Interface para Serviço Analog Senses que implementa a API SenseService

Esta API é direcionada para serviços que não querem que a interface Web permita mudar o seu estado, mas sim apenas deixar o utilizador ver o seu estado. Na API `SliderSensor` é gerado um *slider* para cada atributo.

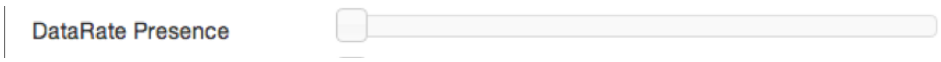


Figura 32 – Interface do atributo DataRate Presence do serviço Sensitivity que implementa a API SliderSensor

Este tipo de API permite ao utilizador regular o valor correspondente ao estado de um determinado atributo do sensor através da plataforma Web. Por último é apresentado a API `SwitchSensor` que gera um botão capaz de mudar o estado de um sensor. Na Figura 33 é apresentado o *switch* referente à antena de um leitor RFID, que pode ser ligada ou desligada através da plataforma.

Figura 33 – Interface do atributo Antenna do serviço Switchs que implementa a API SwitchSenseor

Esta API é direcionada para serviços que desejam que a plataforma permita o utilizador mudar o estado de atributos booleanos do dispositivo.

Para além destes foi também desenvolvida uma API para serviços com objectivo diferente. Não o de permitir que a interface Web seja capaz de oferecer ao utilizador a capacidade de interagir, mas sim dar a possibilidade de outro serviço interagir com este.

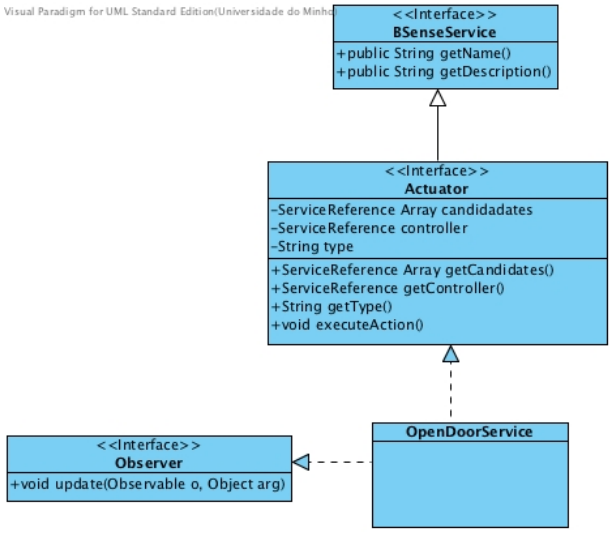


Figura 34 - Exemplo de implementação de um serviço Actuador

Como podemos ver pelo diagrama da figura 34 a interface tem a variável *candidates*, esta representa todos os possíveis serviços que poderão interagir com este. Entre esses candidatos é selecionado pelo utilizador qual dos candidatos ocupa o lugar de *controller*, guardado na variável *controller*. Quando isto acontece a plataforma utiliza o padrão *observer*, conhecido em Java, para adicionar o serviço actuador como *observer* do serviço referenciado como o seu controlador. Assim quando o seu controlador muda o seu estado o observador é notificado e age de acordo com o sua funcionalidade.

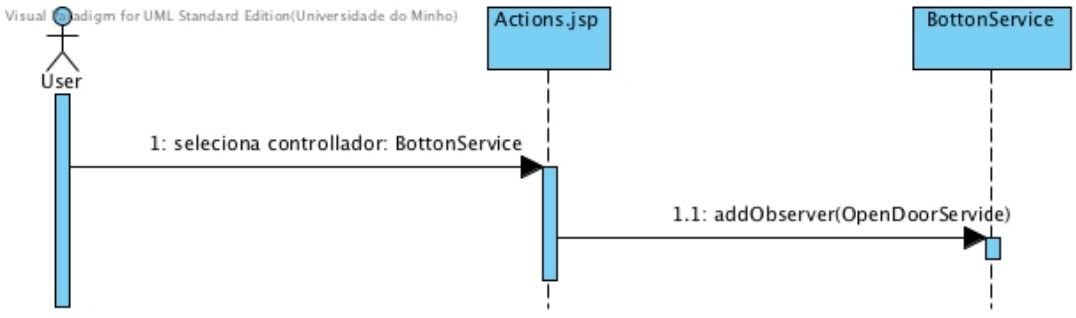


Figura 35 – Diagrama de sequencia do evento de selecionar controlador para um actuador

Podemos então ter um serviço que implementa esta API em que a sua função é a de abrir uma porta. Este serviço identifica na plataforma os seus candidatos. Posteriormente o utilizador seleciona um dos candidatos, por exemplo um serviço "botão" que implemente a API `SwitchService`. Quando o método `setStateValue(String)`, for invocado este consequentemente invoca o método `notifyObservers()` que notifica o serviço "abrir porta" e então que envia sinais elétricos para poder abrir realmente a porta.

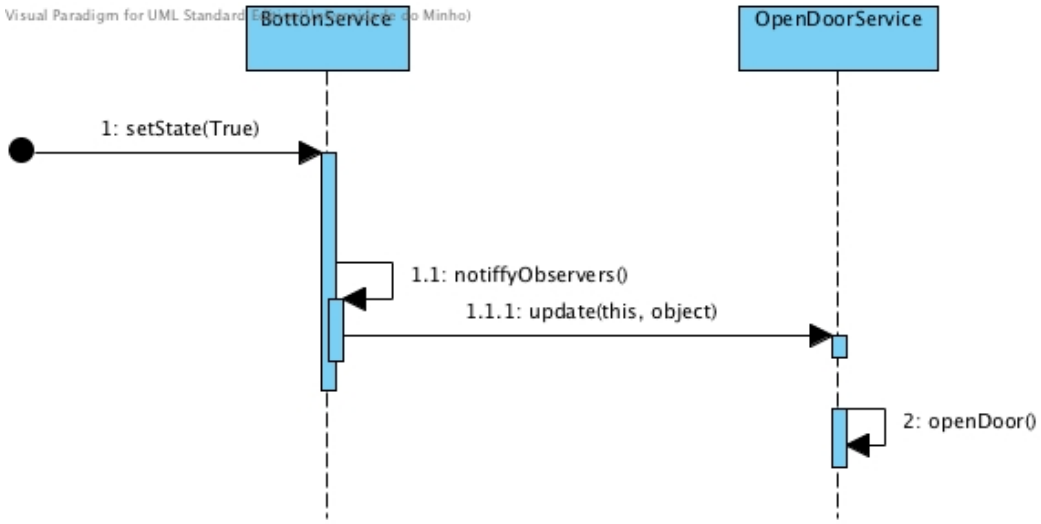


Figura 36 - Exemplo de comunicação entre controlador e actuador

4.6. Camada de aplicação

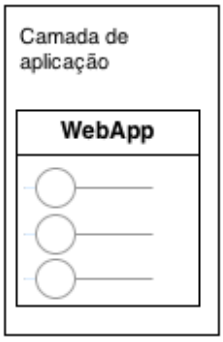


Figura 37 - Camada de aplicação do sistema BSense

A camada de aplicação é responsável pela interação com utilizador, consiste numa plataforma Web num servidor Jetty. Esta plataforma permite mostrar ao utilizador o estado dos sensores e

oferece também a possibilidade de interagir com o sistema modificando o estado de alguns deles. Possibilita também o administrador de instalar/desinstalar *bundles*.

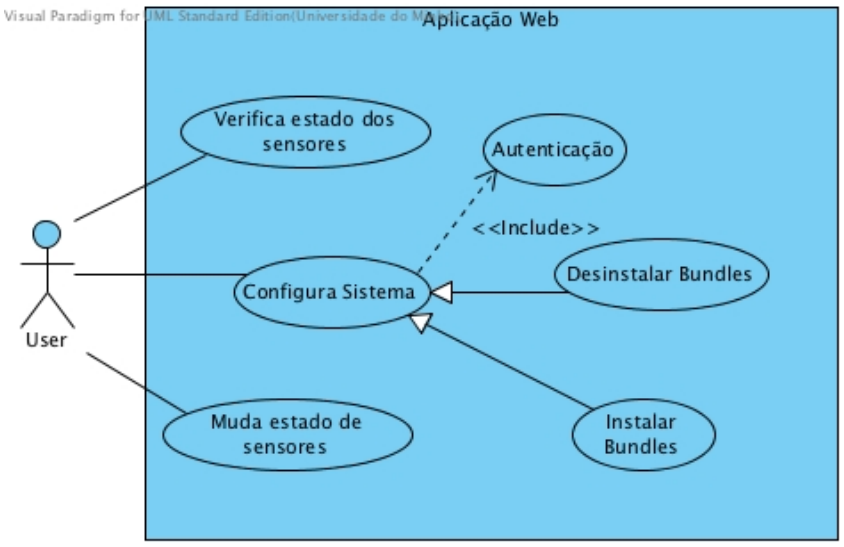


Figura 38 - Diagrama de Use Case para a interação entre o utilizador e a plataforma

No diagrama de Use Case apresentado na figura 38, podemos verificar todas as ações que estão disponíveis para o utilizador na aplicação Web.

A aplicação foi desenvolvida utilizando a tecnologia Java, para a programação Web foi usada a linguagem JavaServer Pages. Para o *front-end*, da aplicação foi usada a *framework* de design Bootstrap e a biblioteca de Javascript jQuery.

O resultado final da página inicial pode ser visto na figura 39.

Temos assim na página inicial a capacidade de ver os valores dos sensores, e poder alterar o estado de alguns. A página inicial está dividida por seções que são os diferentes serviços de API existentes da camada API de serviços falada na secção 4.5.

A plataforma capta as mudanças feitas pelo utilizador através de jQuery que posteriormente envia determinada mudança em um determinado sensor para um *servlet* através de HTTP *request*. Este *servlet* comunica com a classe que tem o registo dos serviços e envia um pedido para mudar o estado antigo para novo.

A plataforma permite também associar um serviço a outro serviço de modo a um controlar o outro se forem compatíveis.

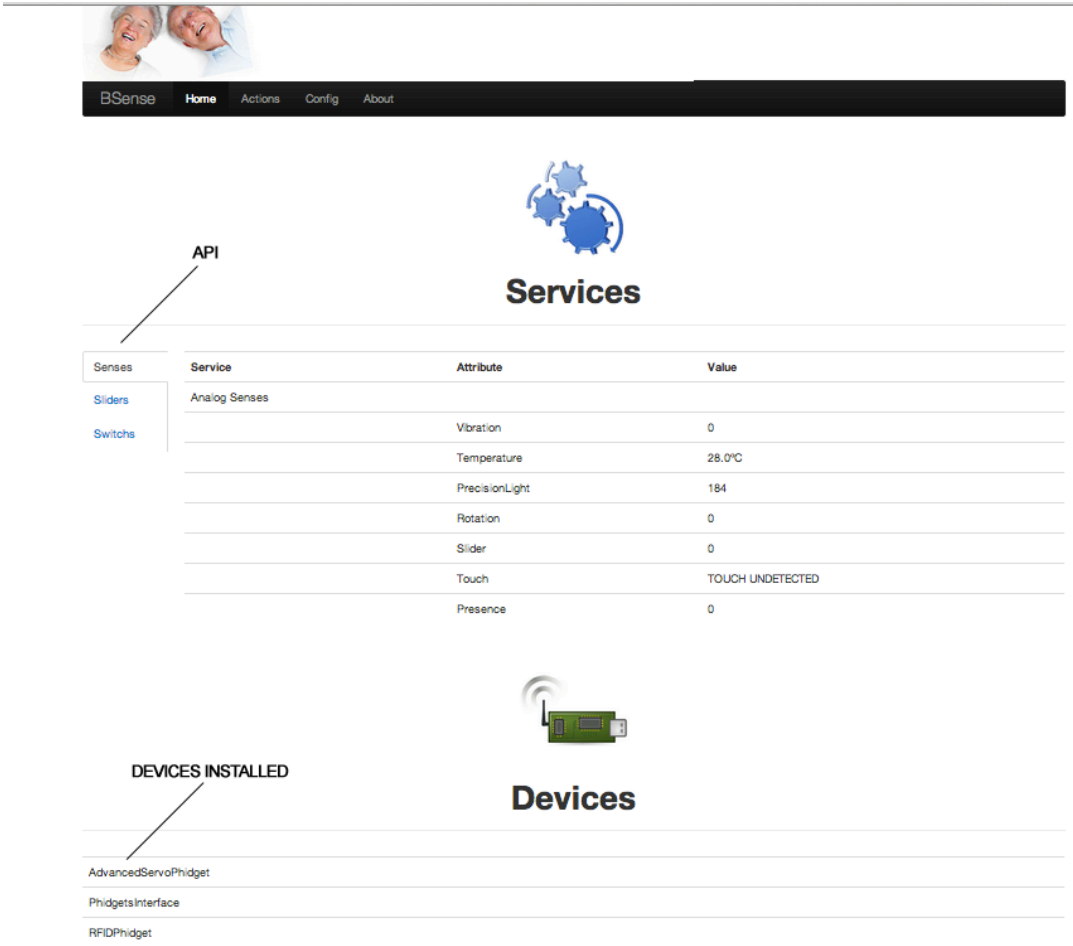


Figura 39 - Página inicial da aplicação Web

É então possível definir serviços disponibilizados por outros dispositivos capazes de controlar os seus actuadores. Permitindo assim uma maior dinâmica e interoperabilidade entre diferentes dispositivos. Para isto poder acontecer o serviço obrigatoriamente tem de ser programado de acordo com as especificações explicadas na secção 4.5.

Existe também uma área mais restrita em que é necessária a introdução de nome de utilizador e password. Esta área dá permissão ao utilizador para poder executar funções de configuração como instalação, atualização e remoção de *bundles* e descrição avançada das suas características.

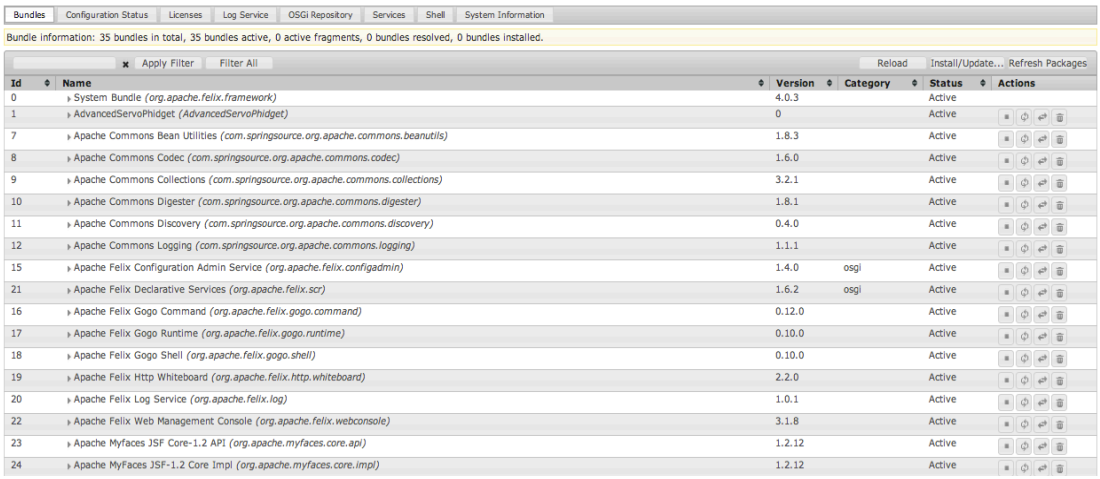


Figura 40 - Página de instalação e desinstalação de *bundles*

Como pode ser visto na figura 40 é oferecido ao utilizador uma interface capaz de configurar o sistema de modo a instalar *bundles* referentes a dispositivos ou não. Na parte de configurações da aplicação Web é também possível aceder a informações importantes como ver todos os serviços instalados assim como informação do sistema, a interface gráfica pode ser vista no Anexo II.

4.7. Estrutura de dados

Figura 41 – Estrutura de sensores do sistema BSense

A estrutura de dados é responsável pelos registos dos estados dos sensores num determinado momento. Isto para que assim no futuro seja possível fazer uma análise sobre os dados regista-

dos e poder extrair conhecimento a partir dessa informação através de algoritmos de *Machine Learning*(ML).

Para tal objectivo foi decidido que os dados deveriam ser guardados de forma a serem facilmente utilizados por *parsers*, que possam posteriormente gerar *datasets* que servirão de input em algoritmos de ML para detectar padrões e inferir resultados. Para esse efeito foi considerado guardar os dados em XML(*eXtensible Markup Language*) ou em JSON(*JavaScript Object Notation*), as duas são consideradas *data interchange formats*, um estudo comparativo em relação à velocidade está representado em [33]. Depois de analisadas as duas opções tem as suas vantagens e desvantagens no entanto foi escolhido o JSON.

JSON é considerado um formato *data exchange* assim como XML, RDF ou YAML. É constituído por uma estrutura de fácil compreensão para um ser humano assim como para fazer *parse* através de programação. O seu conteúdo é constituído por duas estruturas, um conjunto de pares de nome/valor e uma lista ordenada de valores.

Figura 42 - Ficheiro exemplo de dados guardados em JSON

Na figura 42 podemos ver um exemplo de um ficheiro JSON, neste caso podemos verificar que foi um registo realizado às dezassete horas, do dia quatro de Maio de 2013.

Regista dados como, o período do dia, dia da semana e estação do ano para no futuro poder ser mais fácil determinar padrões. Isto porque é recomendado em determinados algoritmos de *Data Mining* utilizar valores discretos para mais facilmente inferir resultados mais significativos.

Devido à sua estrutura, os dados podem vir a ser alterados caso no futuro seja necessário recolher mais ou menos informação dependendo da necessidade do caso de estudo.

4.8. Síntese

A arquitetura do projeto é constituída por 5 camadas em que cada uma tem os seus objectivos específicos. A primeira camada representa os diversos dispositivos e sistemas electrónicos que poderão existir no ambiente, assim como os seus sensores que detectam alterações no ambiente. A segunda camada representa os *bundles* instalados em dos diversos dispositivos que podem ser instalados em tempo de execução. Temos também a camada de serviços que representa o core do OSGi, onde *bundles* são instalados e os serviços são registados. A camada de API de Serviços é uma camada importante pois permite à camada de aplicação reconhecer os dispositivos e gerar a sua interface dinamicamente. A camada de aplicação representa uma aplicação Web que tem a responsabilidade de criar a ponte entre o utilizador e todo o sistema. Por último os dados podem ser registados com uma estrutura em JSON para em uma fase futura ser fácil de aplicar *parsers* para obter *datasets*.

BSense tem a característica de que as camadas são desenvolvidas em módulos diferentes permitindo assim a possibilidade de modificação módulos sem implicação geral. Podendo ser assim extensível a novas camadas e melhoramentos. Permite a instalação de *bundles* em tempo real assim como o registo de serviços e a camada de aplicação consegue gerar código para a manipulação dos dispositivos instalados.

A plataforma permite assim a monitorização de habitantes em um ambiente através de dispositivos, integrando uma aplicação Web ser acedida pela internet em que os utilizadores podem ver o estado dos vários sensores assim como modificar o seu estado.

No entanto para que os dispositivos sejam reconhecidos a plataforma os seus *bundles* necessitam de implementar certas especificações explicadas ao longo deste capitulo. Deixando o traba-

lho todo de configuração do sistema para os programadores de *bundles* e não para os habitantes da casa.

5. Casos de teste

Depois de a elaboração de toda arquitetura do sistema e a sua implementação é necessária a fase de casos de teste de forma a percebermos se de facto a arquitetura funciona como desejado.

Um dos objectivos seria integrar o sistema desenvolvido em um *home gateway*, no entanto e devido a limitações em termos de hardware, o sistema foi emulado num desktop. A plataforma corre num servidor com sistema operativo OS X *Snow Leopard*, processador 2.4 GHz Intel Core i5 e memória RAM de 4GB 1067 MHz DD3.

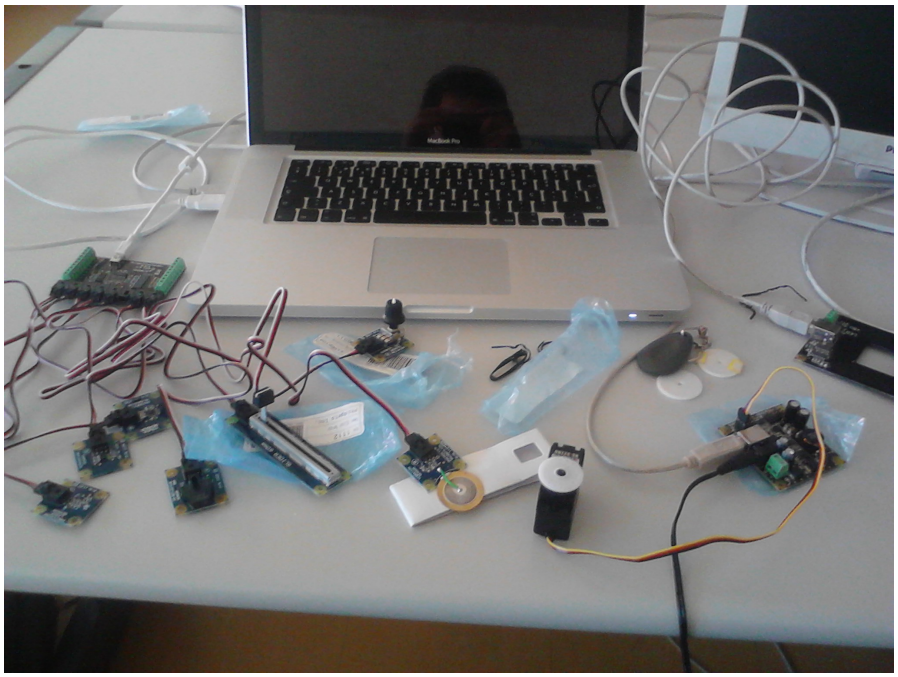


Figura 43 - Material utilizado para testes

5.1. Cenários

Para os casos de teste foram usados 3 sensores *Phidgets* abordados na secção 3.3.3. O primeiro trata-se de um com ligação a 7 sensores analógicos, temperatura, intensidade de luz, vibração, botão *slider*, botão de rotação, presença e toque. O segundo trata-se de um leitor de *tags* RFID. Por último temos um com um *Advanced Servo* que controla um motor RC.

Pegando nestes sensores testados na plataforma é possível criar vários cenários de implementação. De seguida serão abordados quatro cenários em diferentes contextos, no contexto do perigo, conforto, monitorização e segurança.

5.1.1. Perigo

Este cenário passa-se em um ambiente fechado, mais precisamente em uma cozinha. Representa um cenário de caso de perigo pois tem o fogão que pode incendiar algo que se pode propagar à restante mobília.

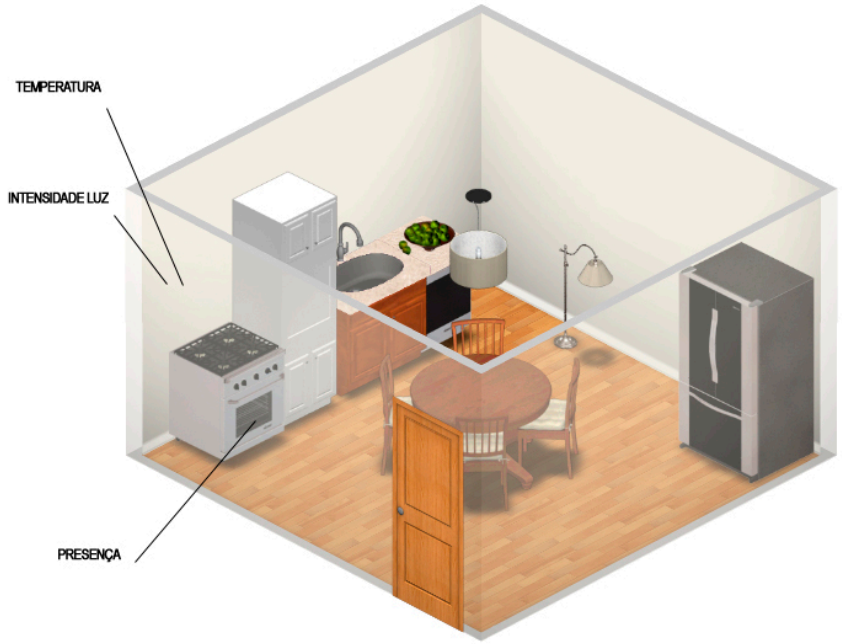


Figura 44 - Distribuição de sensores para cenário de perigo

A figura 44 mostra de que forma podem ser utilizados alguns dos sensores testados no sistema, ao colocar um detector de temperatura para conseguir saber se está demasiado quente, podendo ser um alerta. Este sensor deve estar próximo da zona de perigo para detectar a anormalidade o mais rápido possível.

O sensor de intensidade de luz é um índice de alerta pois se estiver perto da zona de perigo quando o fogo se alastrar a intensidade de luz aumenta. Se aumentar para um nível de alerta juntamente com o sensor temperatura torna o a informação mais viável.

Pode ser também usado o sensor de presença para detectar se está alguém perto do fogão, isto indicia que está alguém a controlar a situação e o aumento de temperatura e luminosidade seja justificável.

5.1.2. Conforto

Este é um cenário que passa em uma sala de estar, onde normalmente os habitantes descaçam e relaxam. Este é um cenário que tem o objetivo de proporcionar conforto a todos os presentes no ambiente.

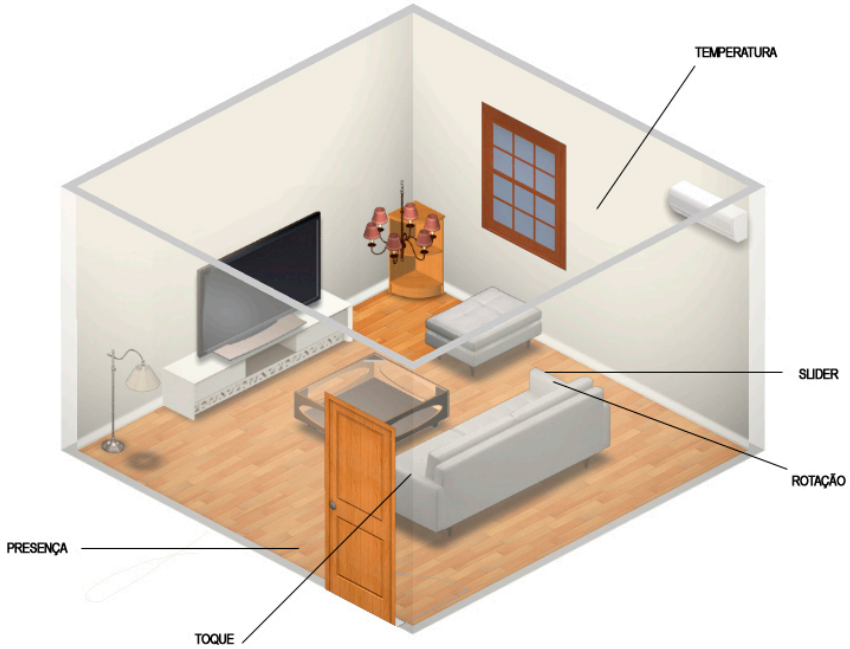


Figura 45 - Distribuição de sensores para cenário de conforto

Alguns dos sensores usados para teste podem ser distribuídos como segundo a figura 45. Utilizando o sensor de toque no apoio do sofá é possível detectar se alguém põem o braço. Este sensor de toque comunicaria com a televisão de modo a ligar-se automaticamente evitando o esforço do habitante.

O sensor temperatura pode estar localizado na sala permitindo que o utilizador utilize a *Smart Tv* para aceder à plataforma através da internet e possa aceder à informação relativa à temperatura da sala. Se achar que a temperatura é desconfortável tem a possibilidade de utilizar o sensor relativo ao *slider* integrado no sofá para regular a temperatura da sala através do ar condicionado.

O sensor de presença deve estar localizado perto da porta isto para que quando alguém entrar as luzes se acendam automaticamente.

Por último, é possível o sensor de rotação também estar integrado no sofá, e assim regular a abertura da janela. Para abrir a janela pode ser usado um motor com funcionamento equivalente ao motor RC testado na plataforma.

5.1.3. Monitorização

Os sensores utilizados também podem ser usados em cenários de monitorização. O cenário apresentado toma lugar em um quarto com uma cama onde normalmente os habitantes dormem. Este ambiente também é propicio a quedas, seja durante o sono ou mesmo quando um ocupante é fisicamente debilitado e tem dificuldades em se deitar na cama.

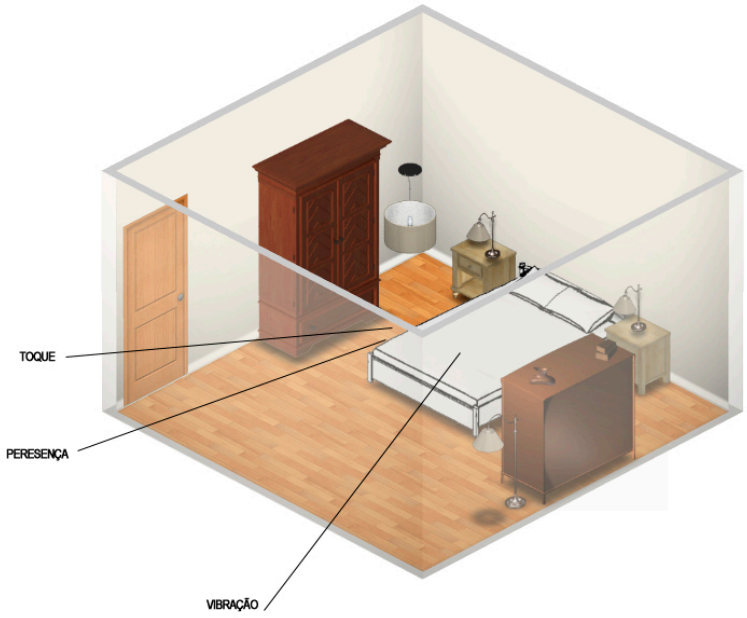


Figura 46 - Distribuição de sensores para cenário de monitorização

O sensor de toque colocado no chão perto da cama deteta possíveis quedas. O sensor de presença na parte lateral da cama tem a capacidade de confirmar que existe um obstáculo no chão perto da cama, aumentando assim fiabilidade do alerta.

5.1.4. Segurança

O sensor RFID *Phidget* utilizado nos casos de teste e pode ser adaptado a um cenário de segurança. O contexto apresentado é bastante simples trata-se apenas de uma entrada principal de acesso ao corredor de uma determinada casa. Esta entrada é constituída por um a porta elétrica de difícil arrombamento.

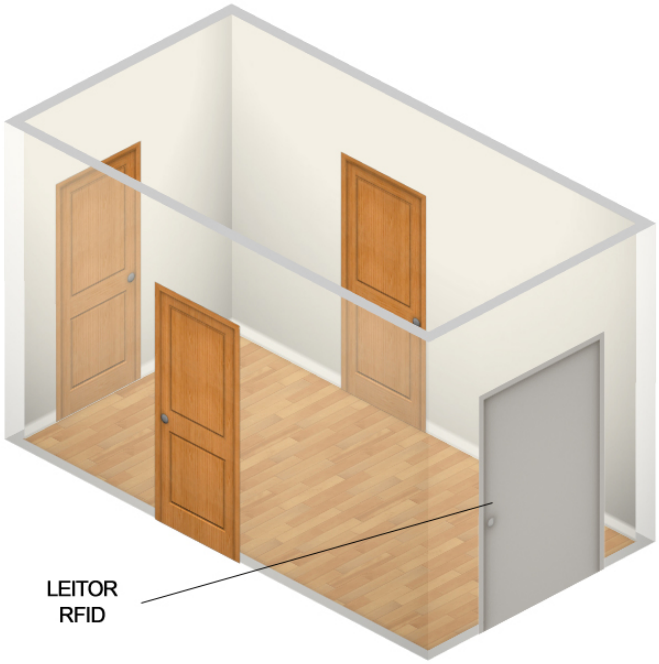


Figura 47 - Posicionamento de leitor RFID em cenário de segurança

Como se pode ver na figura 47 o leitor RFID deve estar frente à porta no lugar da fechadura. O habitante da casa para abrir a porta tem de aproximar a *tag* RFID do leitor. Após a leitura da *tag* o sistema verifica se a *tag* identificada pertence à lista das quais permitem abrir esta porta. Se pertencer a porta abre para a entrada do utilizador e fecha posteriormente.

5.2. Implementação dos bundles para Sensores

Para a integração dos sensores na plataforma foi necessário proceder ao desenvolvimento dos seus *bundles*. De seguida é explicado o processo de desenvolvimento assim como o funcionamento de cada um dos sensores de acordo com as especificações requeridas pela arquitetura abordadas no capítulo 4.

5.2.1. Bundle Interface Sensores

Este *bundle* diz respeito ao *Phidget Interface Kit 8/8/8* e é instalado na plataforma em tempo de execução sem ser necessário a sua reinicialização. Este *Phidget* possibilitou a iteração com 7 sensores, são eles sensores de temperatura, presença, intensidade de luz, vibração, botão *slider*, botão rotativo e toque. Além disso disponibiliza 2 serviços diferentes, um relacionado com a sensibilidade em relação à observação do ambiente e outro para dar a conhecer os dados do ambiente ao utilizador.

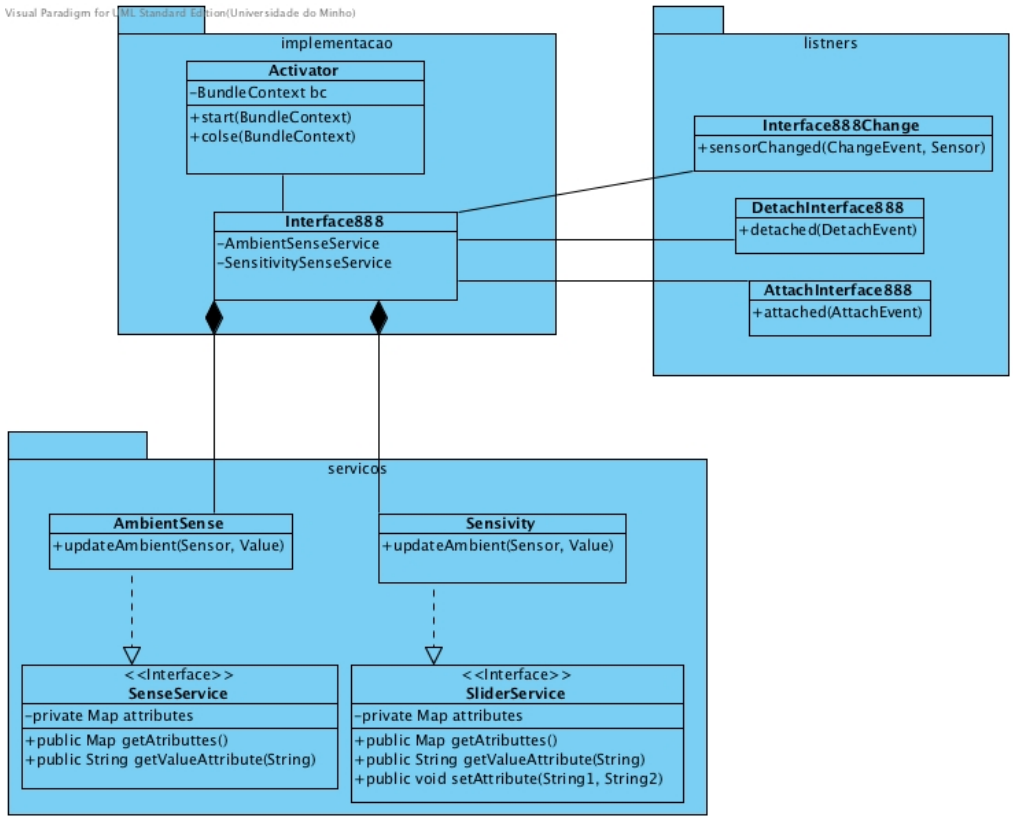


Figura 48 - Diagrama de classes para *bundle* Interface Kit 8/8/8

É importante que quando o dispositivo é ligado sejam registados todos os serviços disponibilizados, assim como é importante quando o dispositivo for desligado tirar do registo os serviços de forma a eles não estarem disponíveis no sistema. Temos assim a possibilidade ter serviços disponíveis ou indisponíveis automaticamente sem ser preciso necessário parar a plataforma.

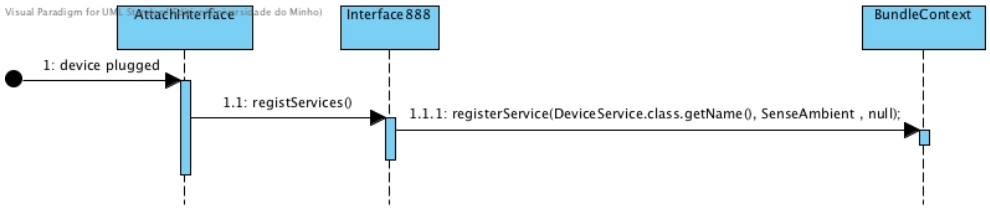


Figura 49 - Diagrama de sequencia para a conexão de dispositivo

Para isso são usados *listners* capazes de avisar quando o dispositivo foi ligado/desligado como podemos ver na figura 49. A classe `AttachInterface` comunica com a `Interface888` que trata do registo dos serviços na plataforma através da classe `BundleContext`.

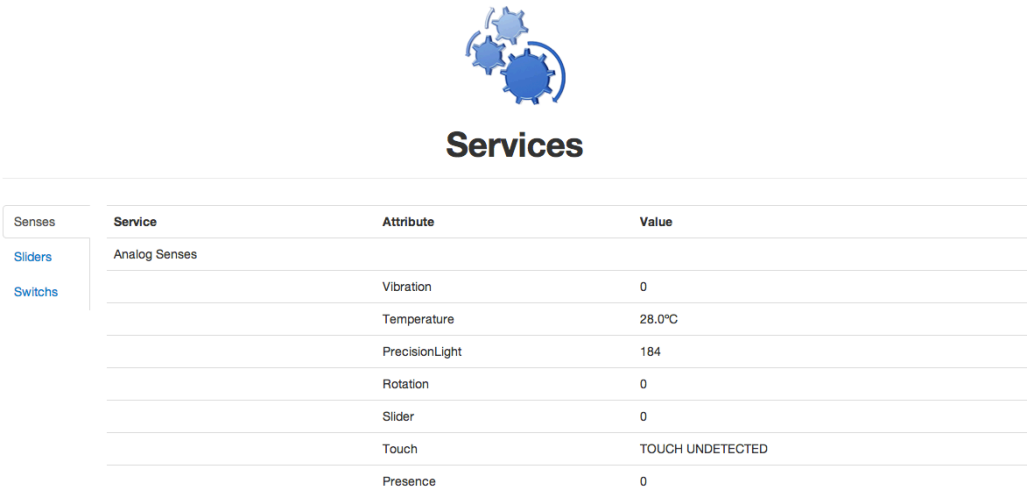


Figura 50 - Valores dos sensores analógicos disponibilizados pela plataforma

Na figura 50 pode-se visualizar o estado dos diversos sensores, no cenário de monitorização apresentado se a plataforma apresenta-se o valor do sensor de toque como "DETECTED", o de presença por volta dos 800 e vibração a 0, seria necessário lançar um alerta pois existiria a possibilidade de alguém ter caído da cama.

Cada um dos sensores analógicos regista um determinado valor com informação sobre o ambiente, que não deve ser mudados pela plataforma, apenas o ambiente tem o papel de fazer mudar o seu estado. Isto quer dizer que todos implementam um serviço que usam a API `SenseService` falada na secção 4.5.

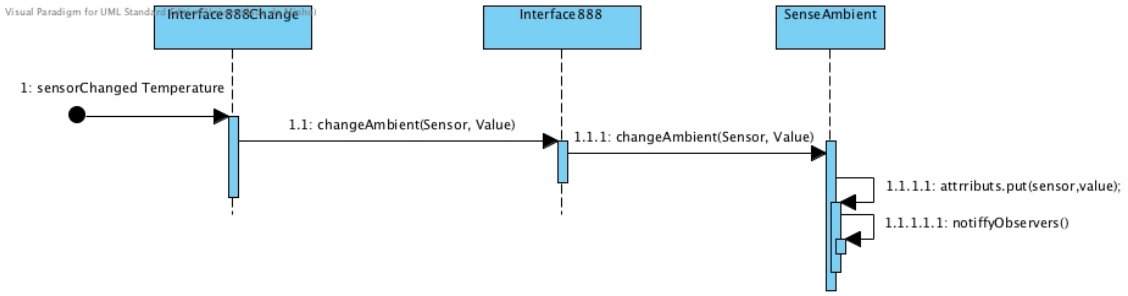


Figura 51 - Diagrama de sequencia quando sensor muda o seu estado

A mudança é identificada pela classe `Interface88Change` como se pode ver na figura 51, este comunica com a `Interface888` que posteriormente comunica com classe do serviço que avisa todos os seus observadores.

Este *bundle* também regista outro serviço de nome `SensitivityService` que implementa a API `SliderService`. Permitindo assim o utilizador de mudar o estado dos seus atributos através da plataforma Web. Os atributos dizem respeito a sensibilidade de cada sensor analógico relativamente ao ambiente em que estão inseridos.

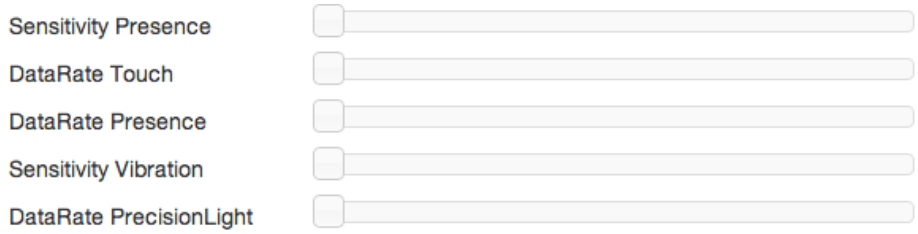


Figura 52 - Atributos do Serviço SensitivityService

No cenário de perigo apresentado, se fossem detectados vários falsos alertas com base nos valores dos sensores de temperatura, intensidade de luz e presença, seria necessário reajustar a sensibilidade aumentando assim a fiabilidade dos alertas.

5.2.2. Bundle RFID

Este *bundle* é implementado para o RFID *Phidget* e também pode ser instalado em tempo de execução. Este dispositivo é constituído por um leitor de *tags* RFID. No projeto foi usado como um leitor de *tags* que abre a porta principal de casa num cenário de segurança, ou seja, um habitante da casa contendo uma das *tags*, poderia abrir a porta.

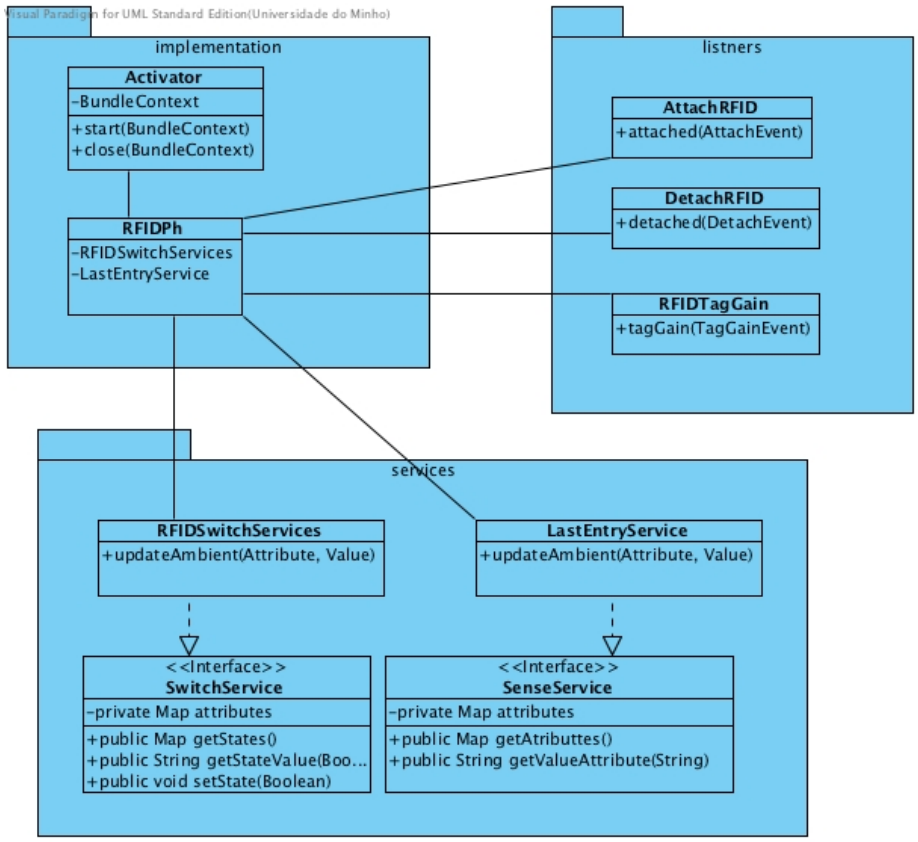


Figura 53 - Diagrama de classes de *bundle* para *Phidget* RFID

Este *bundle* Este regista o serviço `RFIDSwitchService`, que tem dois atributos um que definem o estado da antena e do led como ligados/desligados, este serviço implementa a API `Switch-Service` capaz de dar ao utilizador a possibilidade de mudar o estado através da plataforma Web. Regista também o serviço `LastEntryService` que tem dois atributos que representam a última vez que alguém registou uma *tag* pelo leitor RFID. Este serviço implementa a API `Sense-Service` que possibilita mostrar ao utilizador através da plataforma Web qual foi a última *tag* a ser lida e quando.

Quando o dispositivo é ligado regista os seus serviços, assim como que quando é desligado re-gista também todos os seus serviços de modo a estarem disponíveis/indisponíveis em tempo real.

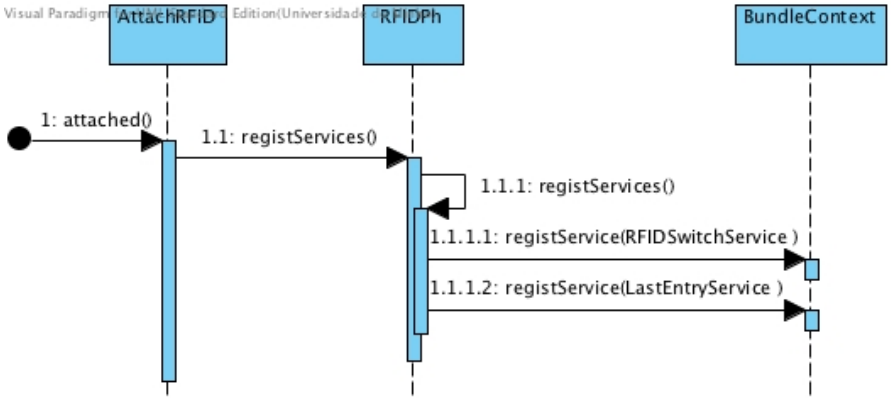


Figura 54 - Diagrama de sequencia para quando Phidget RFID é conectado.

Como se pode verificar na figura 54 este processo é feito através da classe `AttachRFID` que comunica ao `RFIDPh`, este último regista os seus dois serviços através do `BundleContext`.

É necessário também detectar quando uma *tag* é identificada para atualizar o serviço registado. Para isso foi implementado o *listner* `RFIDTagGain`, o diagrama da imagem X representa a comunicação entre casses quando o leitor lê uma *tag*.

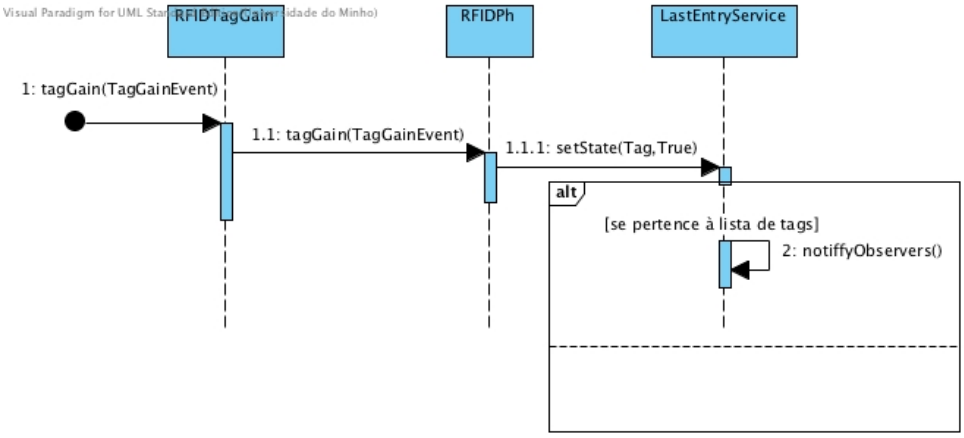


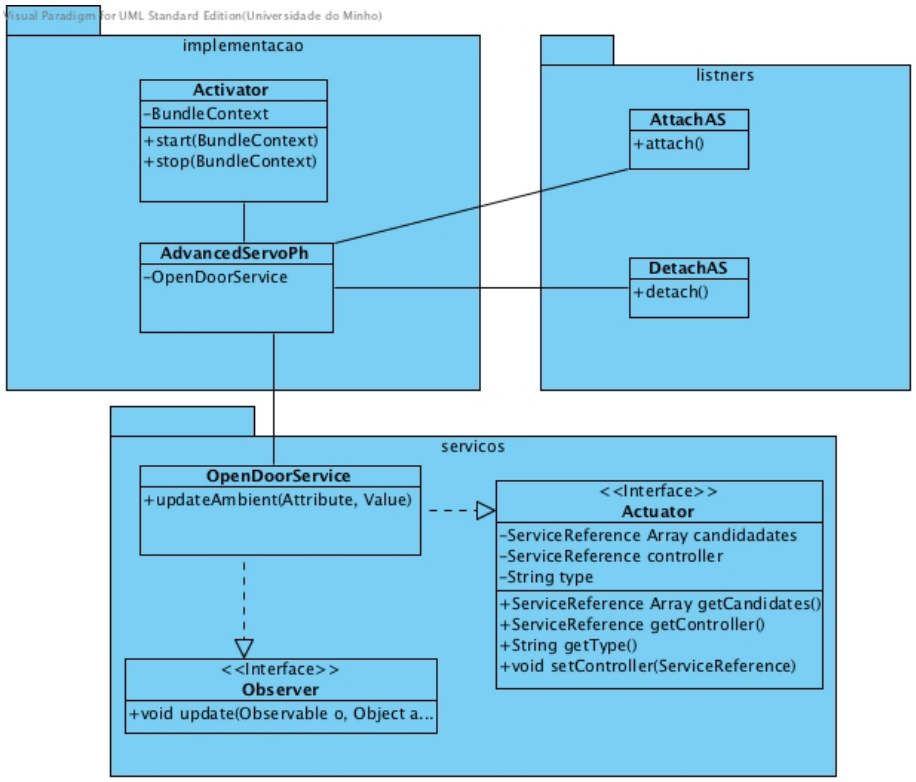
Figura 55 - Diagrama de sequancia para quando *tag* é identificada

Quando uma *tag* é identificada a classe `RFIDTagGain` é avisada, esta envia o evento para a `RFIDPh` que comunica com o serviço `LastEnteryService` que posteriormente verifica se a *tag* registada pertence à lista de *tags*, se pertencer muda o estado do atributo e avisa todos os seus observadores.

No cenário de segurança apresentado o habitante da casa teria de passar a *tag* pelo RFID de modo a abrir a porta, esta porta só abriria se pertencesse à lista de *tags* autorizadas pelo sistema.

5.2.3. Bundle Advanced Servo

O *Phidget Advanced Servo* permite controlar até 8 pequenos motores RC, no entanto nos testes apenas foi necessário utilizar um. Este motor nos testes realizados trabalhava apenas quando uma *tag* RFID era passada pelo leitor através do *Phidget* RFID. Este *bundle* apenas regista um



serviço que simula o abrir de uma porta referido no cenário de segurança.

Figura 56 - Diagrama de classes de *bundle* para *Phidget* Advanced Servo

Assim como todos os *bundles* devem fazer, este regista todos os serviços quando o dispositivo é ligado e tira-os do registo quando são desligados. Neste caso é apenas o OpenDoorService. Este serviço implementa a API Actuador não permitindo assim o utilizador controlar o dispositivo pela

plataforma mas sim por outro serviço chamado de *controller*. Mas antes de escolher o *controller* é necessário preencher uma lista de candidatos.

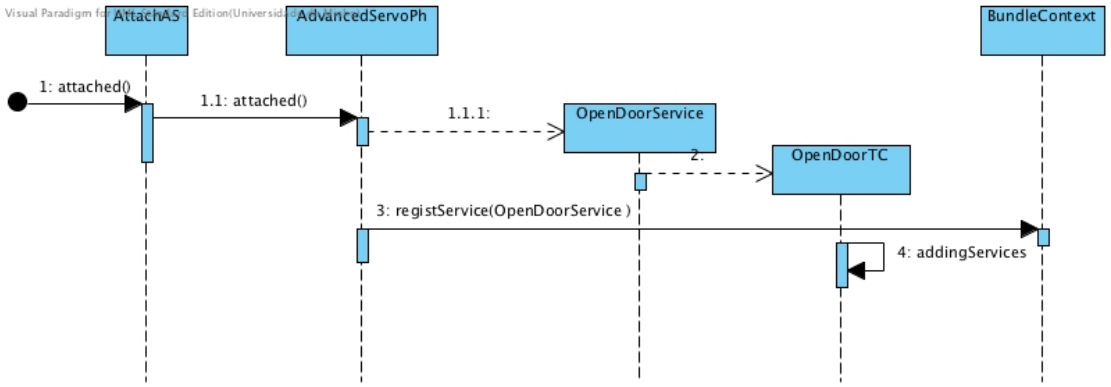


Figura 57 - Diagrama de sequencia para a conexão de *Phidget* AdvancedServo

A lista de candidatos é preenchida quando o serviço é registado. No digrama de sequencia da figura 57 é mostrado o evento de conectar o dispositivo. Como se pode ver o *listner* `AttachAS` é notificado, este notifica a class `AdvancedServo` que cria uma instancia do serviço `OpenDoorService` registando-o de seguida. `OpenDoorService` instancia um `ServiceTracker` com os candidatos, este `ServiceTracker` adiciona serviços à sua lista de acordo com o método `Object addingService(Service Reference sr)` definido na interface `OpenDoorTC`. O *controller* é por predefinição o primeiro elemento da lista, no entanto este pode ser mudado pelo utilizador na plataforma Web na página *Actions*. O Serviço OpenDoorService torna-se observador do controller.

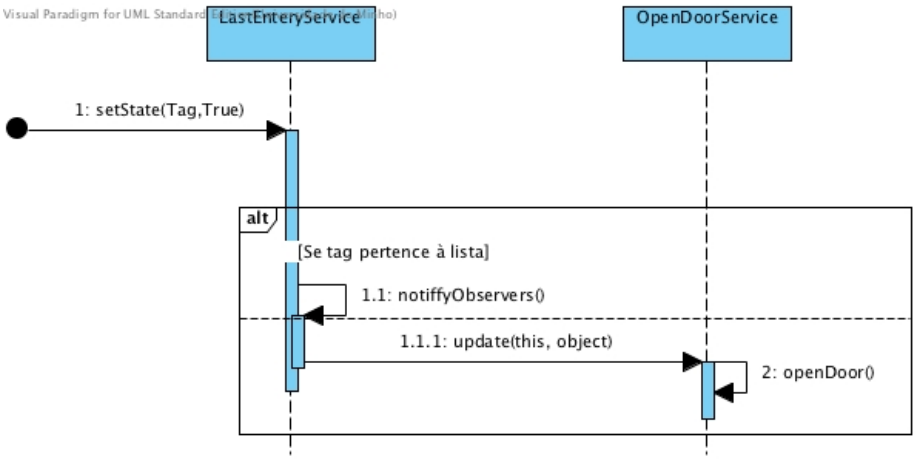


Figura 58 - Diagrama de sequencia para mudança de estado do motor RC

Depois da *tag* RFID ser lida pelo Phidget RFID o serviço LastEntryService notifica todos os seus observadores. Se a *tag* for válida, isto vai fazer com que o Phidget Advanced Servo envie sinais

elétricos para o motor rodar 100º, simulando a abertura da porta, e rodar 100º no sentido contrario 5 segundos depois, simulando o fecho da porta automática.

Assim como consegue abrir e fechar a porta no cenário de segurança, um motor idêntico consegue abrir a janela no cenário de conforto de acordo com o valor do sensor de o *slider*. Para isso é apenas necessário desenvolver *bundle* que transforme o valor do *slider* em proporção de janela aberta.

5.3. Síntese

Para os testes foram usados vários *Phidgets*. O tipo de Phidgets usado pode ser útil para diversos cenários em AmI, seja no cenário de perigo quando um o fogo do fogão se pode alastrar, seja em cenário de conforto numa sala em que o utilizador não necessita de fazer esforços para determinadas ações, em cenário de monitorização em que se pode monitorizar o sono ou detectar quedas da cama ou mesmo num cenário de segurança em que apenas quem tem determinadas *tags* consegue abrir determinada porta.

Para realizar testes à plataforma foi necessário desenvolver três *bundles* referentes aos três diferentes dispositivos utilizados. Os dispositivos são sensores *Phidgets* já abordados na secção 3.2.3. Um dos dispositivos utilizados foi o *Phidget Interface Kit 8/8/8* em conjunto com sensores analógicos de temperatura, vibração, presença, toque, intensidade da luz, switch e slider. Foi usado também um *RFID Phidget* e as suas diversas *tags*. Por último foi usado o *Advanced Servo Phidget* com um motor RC. Todas as especificações dos dispositivos podem ser encontrados no Anexo I.

6. Conclusões e Trabalho Futuro

O envelhecimento da população em Portugal e na europa está a tornar-se em um problema cada vez mais grave. É necessário a construção de sistemas e infraestruturas capazes de dar resposta a este problema. Inteligência Ambiente e mais especificamente a área de *Ambient Assisted Living* podem dar resposta a estas necessidades. Com a modernização do estilo de vida cada vez mais as pessoas usam dispositivos e sistemas elétricos, seria então uma mais valia poder usufruir destes para uma melhoria da qualidade de vida.

OSGi são especificações com foco na tecnologia Java que permitem a instalação de módulos em tempo real. Associando estas especificações a AmI, se cada um dos módulos disser respeito à instalação de dispositivos os utilizadores não necessitam de reinicializar o sistema depois da instalação.

Um problema grave é que os dispositivos utilizam normalmente protocolos de comunicação diferentes e por serem de diferentes fabricantes não é possível a sua comunicação. Na comunidade OSGi existem também vários projetos que fazem a ponte entre diferentes protocolos de comunicação, podendo assim diferentes tipos de dispositivos comunicarem entre si.

Este é um projeto que tendo como motivação todos estes fatores tem o objectivo inovador de juntar as especificações OSGi em *home* um *gatway*, para poder controlar diversos dispositivos em um contexto de AAL. Embora já existam alguns projetos na área estes trabalham com diferentes objectivos. Em resumo pode ser dito que durante este projeto e tendo em conta os objectivos estipulados foram elaborados as seguintes trabalhos:

- Estudo sobre projetos inseridos na mesma área, analisados projetos no âmbito de AAL, AmI e OSGi.
- Desenvolvimento de uma arquitetura modular usando a tecnologia OSGi.
- Sistema capaz de fazer de carregar módulos e integrar serviços em tempo real.
- Sistema capaz de permitir a comunicação entre diferentes dispositivos.

- Integração de uma aplicação Web dinâmica.
- Desenvolvimento e implementação de casos de teste em ambiente controlado.

Este sistema pode assim responder aos diversos desafios sociais e tecnológicos identificados no inicio desta dissertação. Permitindo a possibilidade de ser um sistema de fácil utilização e capaz de utilizar diversos sensores de forma a monitorizar o ambiente e assim como responder a diferentes cenários.

A nível tecnológico este é um trabalho que para além de pretender inovar no campo de Plataformas de comunicação para AmI e AAL é uma plataforma que utiliza uma tecnologia emergente OSGi. A plataforma permite a integração serviços em tempo real não necessitando assim de reinicializar a plataforma quando um novo dispositivo é adicionado ou removido. Disponibiliza também uma aplicação Web que gera interface para o utilizador poder controlar os diversos dispositivos através da internet.

6.1. Trabalho Futuro

A tecnologia OSGi oferece a importante característica de possibilitar fundir diferentes tecnologias. Existem vários projetos a fazer *bundles* capazes de fazer essa ponte, um próximo passo seria tentar juntar diferentes tecnologias na mesma plataforma, permitindo assim uma maior diversidade de aplicações possíveis a integrar no sistema. Este trabalho passará por não só por integrar projetos já elaborados capazes de providenciar comunicação entre algumas tecnologias [4], mas também a criação de *bundles* inexistentes que permitam a comunicação entre outras tecnologias.

Outro dos objectivos principais seria no futuro integrar o projeto iGenda que deu origem ao BSense e assim de forma colaborativa poder aumentar a diversidade de serviços e a monitorização.

Com a capacidade de poder integrar sensores e guardar a seu estado num determinado espaço de tempo, um futuro passo passará também por gerar grande quantidade de dados para posteriormente se avançar no campo de Context-Awerenss e perceber através de Machine Learning, algoritmos de *Data Mining* e extração de conhecimento, reconhecer padrões e posteriormente adicionar uma camada capaz de tornar BSense autónomo ao nível de conseguir prever determi-

nada ação. Por exemplo se a partir das dez da manhã o utilizador costuma fechar as persianas da sala durante o verão, o sistema poderá detectar este padrão e executa-lo autonomamente facilitando a vida do utilizador.

Na camada APIs de serviços foram desenvolvidas APIs o mais genéricas possíveis e pelo menos uma das disponíveis cobrem a maior parte dos serviços possíveis no entanto esta camada devem ser um caso de estudo para o futuro podendo vir a sofrer alterações, como por exemplo, os *bundles* correspondentes terem de oferecer *servlets* com interface Web, permitindo assim o um desenvolvimento mais genérico da plataforma, no entanto existiria uma perca de uniformidade a nível visual em relação à aplicação Web.

7. Bibliografia

1. Costa, Â., & Novais, P. (2012). Mobile Sensor Systems on Outpatients. *International Journal of Artificial Intelligence*. Retrieved from http://ceser.in/ceserp/index.php/ijai/article/view/1295

2. Costa, Â., & Castillo, J. (2012). Sensor-driven agenda for intelligent home care of the elderly. *Expert Systems with ...*. Retrieved from http://www.sciencedirect.com/science/article/pii/S0957417412006550

3. Carneiro, D. (n.d.). *Simulating and Monitoring Ambient Assisted Living*. Universidade do Minho.

4. Dobrev, P., & Famolari, D. (2002). Device and service discovery in home networks with OSGi. *Communications ...*, (August), 86–92. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1024420

5. Fuchsberger, V. (2008). Ambient assisted living: elderly people's needs and how to face them. *... 1st ACM international workshop on Semantic ambient ...*, 21–24. Retrieved from http://dl.acm.org/citation.cfm?id=1461917

6. Helal, S., Mann, W., & El-Zabadani, H. (2005). The Gator Tech Smart House : A Programmable Pervasive Space. *Computer*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1413118

7. Lopes, F. S. (2010). *Smart Home Device Application*. FEUP.

8. Ngo, L. (n.d.). Service-oriented architecture for home networks, 3–8.

9. O'Grady, M., Muldoon, C., & Dragone, M. (2010). Towards evolutionary ambient assisted living systems. *Journal of Ambient ...*. Retrieved from http://www.springerlink.com/index/GG535327010NHQL2.pdf

10. Ramos, J., Anacleto, R., Novais, P., Figueiredo, L., & Almeida, A. (n.d.). Orientation System for People with Cognitive Disabilities.

11. Cook, D., Augusto, J., & Jakkula, V. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 1–38. Retrieved from http://www.sciencedirect.com/science/article/pii/S157411920900025X

12. RODZI, M. R. B. M. (n.d.). *Home automation using x-10 technology mohamad ridhwan bin mohamed rodzi university malaysia pahang*. UNIVERSITY MALAYSIA PAHANG.

13. Lee, J., Su, Y., & Shen, C. (2007). A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. *Industrial Electronics Society, 2007. ...*, 46–51. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4460126

14. Raut, A., & Malik, L. (2011). ZigBee: The Emerging Technology in Building Automation. *International Journal on Computer Science ...*, *3*(4), 1479–1484. Retrieved from http://www.enggjournals.com/ijcse/doc/IJCSE11-03-04-013.pdf

15. Ramos, C., Augusto, J. C., & Shapiro, D. (2008). Ambient Intelligence—the Next Step for Artificial Intelligence. *IEEE Intelligent Systems*, *23*(2), 15–18. doi:10.1109/MIS.2008.19

16. Lansford, J., Stephens, A., & Nevo, R. (2001). Wi-Fi (802.11 b) and Bluetooth: enabling coexistence. *Network, IEEE*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=953230

17. Understanding Wi-Fi ™. (2002). HP Invent.

18. Hollenbeck, R. (2001). The IEEE 802.3 Standard (Ethernet): An Overview of the Technology. Retrieved from http://rionhollenbeck.com/GradPortfolio/Papers/620-Ethernet/Ethernet.pdf

19. Titchkosky, L., Arlitt, M., & Williamson, C. (2003). A performance comparison of dynamic Web technologies. *ACM SIGMETRICS Performance Evaluation* Retrieved from http://dl.acm.org/citation.cfm?id=974037

20. Röcker, C. (2011). Designing Ambient Assisted Living Applications: An Overview over State-of-the-Art Implementation Concepts. *International Conference on Modeling and Simulation 10*, 167–172. Retrieved from http://www.ipcsit.net/vol10/30-ICMSC2011D001.pdf

21. Great Britain Dept Of Health. (2001). *National Service Framework for Older People*. *National Service Framework for Older People* (Vol. 97, p. 202). Department of Health. doi:10.1192/pb.27.4.121

22. Wannamethee, S. G., Shaper, A. G., Walker, M., & Ebrahim, S. (1998). Lifestyle and 15-year survival free of heart attack, stroke, and diabetes in middle-aged British men. *Archives of Internal Medicine*, *158*(22), 2433–2440. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/9855381

23. Turnin, M. C., Bolzonella-Pene, C., Dumoulin, S., Cerf, I., Charpentier, G., Sandre-Banon, D., Valensi, P., et al. (1995). *Multicenter evaluation of the Nutri-Expert Telematic System in diabetic patients*. *Diabete metabolisme* (Vol. 21, pp. 26–33).

24. Jerant, A. F., Azari, R., & Nesbitt, T. S. (2001). *Reducing the cost of frequent hospital admissions for congestive heart failure: a randomized trial of a home telecare intervention. Medical Care* (Vol. 39, pp. 1234–1245). Lippincott Williams \& Wilkins. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11606877

25. Wu, J., Kessler, D., Chakko, S., Kessler, K. (1995). A Cost-Effectiveness Strategy for Transtelephonic Arrhythmia Monitoring. In: American Journal of Cardiology, 75, pp. 184-185.

26. Gosling, J., Joy, B., Steele, G., & Bracha, G. (2005). *Java(TM) Language Specification, The (3rd Edition)*. *AddisonWesley* (p. 688). Addison Wesley. Retrieved from http://www.amazon.de/exec/obidos/ASIN/0321246780

27. Lindholm, T., & Yellin, F. (1999). *The Java Virtual Machine Specification*. *Managing* (pp. 1–84). Addison-Wesley. Retrieved from http://java.sun.com/docs/books/vmspec/

28. Hall, M. (2000). *Core Servlets and JavaServer Pages*. *Muscle & nerve* (Vol. 48). doi:10.1002/mus.23576

29. Park, S. H., Won, S. H., Lee, J. B., & Kim, S. W. (2003). Smart home - digitally engineered domestic life. *Personal and Ubiquitous Computing*, *7*(3-4), 189–196. doi:10.1007/s00779-003-0228-9

30. Legon, J. (2003). 'Smart Sofa' Aimed at Couch Potatoes. CNN, September 23, 2003

31. MIT (2006). Things That Think. Massachusetts Institute of Technology, Cambridge, MA, USA

32. Cook, D. J., & Das, S. K. (2007). How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, *3*(2), 53–73. doi:10.1016/j.pmcj.2006.12.001

33. Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. *Scenario*, *59715*, 157–162. Retrieved from http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf

34. Alliance, Osg. (2007). OSGi Service Platform Core Specification Release 4. *OSGi Specification*. IOS Press, Inc. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:OSGi+Service+Platform,+Core+Specification+Release+4#0

35. Gong L., C. Kirk(2002). Programming open services gateways with Java embedded server technology. Boston, Montréal Addison-Wesley.

36. O. Ian(2012). Sensor technologies leading a convergence of human interface and product differentiation. IEEE SCV ComSoc Monthly Meeting Presentation.

Anexo I

Características de Phidgets

API Object Name	InterfaceKit
USB Voltage Min	4.6 V DC
USB Voltage Max	5.5 V DC
Current Consumption Min	13 mA
Current Consumption Max	500 mA
Available External Current	487 mA
Recommended Wire Size	16 - 26 AWG
USB Speed	Full Speed
Operating Temperature Min	0 °C
Operating Temperature Max	70 °C

Tabela 1 - Placa Interface 8/8/8 Phidget

Number of Analog Inputs	8
Analog Input Resolution	10 bit
Input Impedance	900 kΩ
Analog Input Voltage Min	0 V DC
Analog Input Voltage Max	5 V DC
5V Reference Error Max	0.5 %
Analog Input Update Rate Min	1 samples/s
Analog Input Update Rate Max (4 Channels)	1000 samples/s
Analog Input Update Rate Max (8 Channels)	500 samples/s
Analog Input Update Rate Max (WebService)	62.5 samples/s

Tabela 2 - *Inputs* Analógicos

Number of Digital Inputs	8
Pull-up Resistance	15 kΩ
Low Voltage Max (True)	900 mV DC
High Voltage Min (False)	4.2 V DC
Digital Input Voltage Max	± 15 V DC
Digital Input Update Rate	125 samples/s
Trigger Length Min	3 ms

Tabela 3 - *Inputs* Digitais da Interface

Number of Digital Outputs	8
Series Resistance	300 Ω

Digital Output Current Max	16 mA
Digital Output Voltage Min	0 V DC
Digital Output Voltage Max	5 V DC

Tabela 4 - *Outputs* digitais da Interface

RFID Phidget

API Object Name	RFID
Antenna Output Power Max	10 µW
Antenna Resonant Frequency Min	125 kHz
Antenna Resonant Frequency Max	140 kHz
Protocol	EM4102
Read Rate	33 ms
USB Speed	Low Speed

Tabela 5 - Leitor RFID

Available External Voltage (+5V)	5 V DC
Available External Voltage (LED)	5 V DC
Available External Current (+5V)	400 mA
Available External Current (LED)	16 mA
Output Impedance (LED)	250 Ω
Current Consumption Min	16 mA
Current Consumption Max	100 mA

Tabela 6 - Propriadades elétcias

Recommended Wire Size	16 - 26 AWG
Operating Temperature Min	0 °C
Operating Temperature Max	70 °C

Tabela 7 - Propriadades físicas

Number of Digital Outputs	2
Digital Output Voltage Min	0 V DC
Digital Output Voltage Max	5 V DC

Tabela 8 - *Outputs* digitais RFID

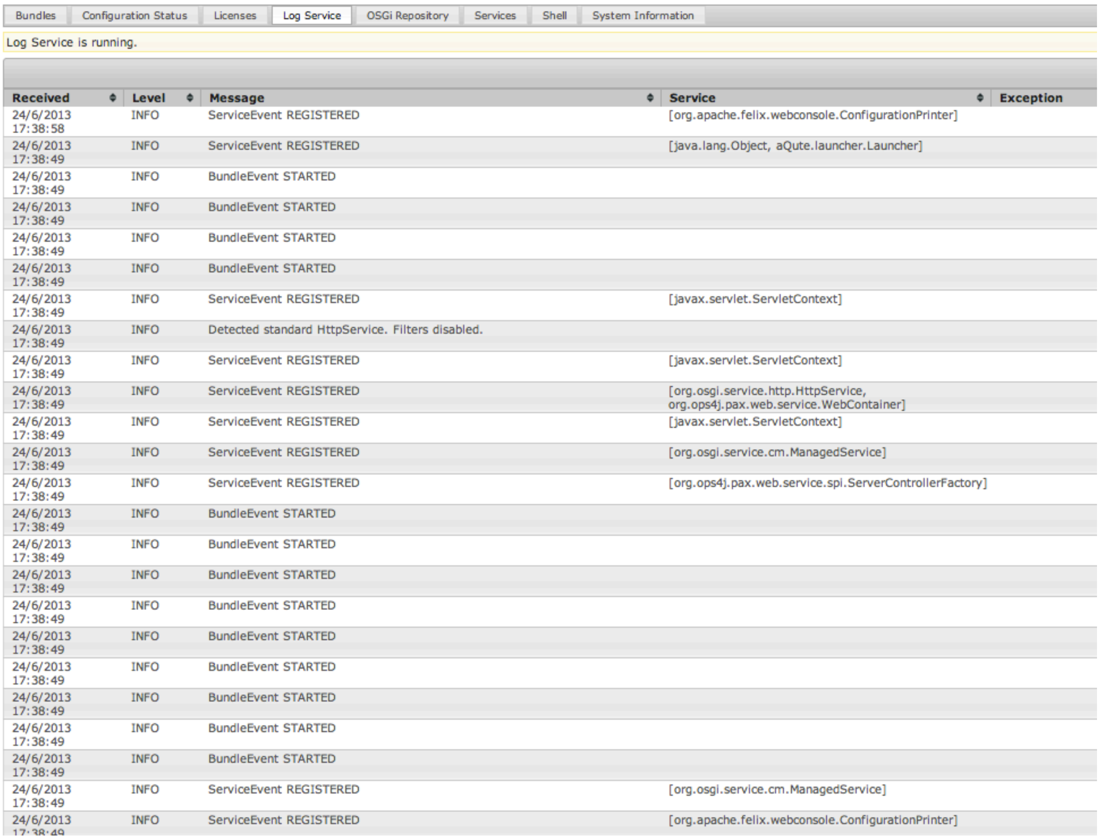
Pulse Code Period	Typical: 20ms - Maximum: 25ms
Minimum Pulse Width	83.3ns
Maximum Pulse Width	2.7307ms
Output Controller Update Rate	Typical: 31 updates/second
Output Impedance (control)	600 Ohms
Position Resolution	0.0078125º (15-bit)
Lower Position Limit	-22.9921875º
Upper Position Limit	233º
Velocity Resolution	0.390625º/s (14-bit)

Velocity Limit	6400º/s
Acceleration Resolution	19.53125º/s2 (14-bit)
Acceleration Limit	320000º/s2
Time Resolution	83.3ns
Minimum Power Supply Voltage	6V
Maximum Power Supply Voltage	15V
Power Jack Dimensions	5.5x2.1mm
Power Jack Polarity	Center Positive
Max Motor Current Continuous(individual)	1.6A
Max Motor Current (Surge)	3 A
Motor Overcurrent Trigger (combined)	12A
Operating Motor Voltage	5.0V
Device Current Consumption	26mA max
Operating Temperature	0 - 70°C

Tabela 9 - *Advanced Servo Phidget*

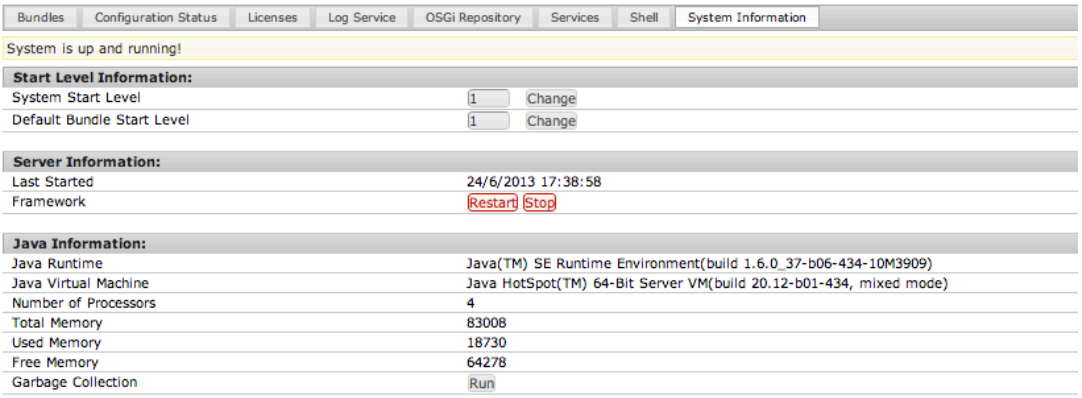
Anexo II

Área de configuração da plataforma



Received	Level	Message	Service	Exception
24/6/2013 17:38:58	INFO	ServiceEvent REGISTERED	[org.apache.felix.webconsole.ConfigurationPrinter]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[java.lang.Object, aQute.launcher.Launcher]	
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[javax.servlet.ServletContext]	
24/6/2013 17:38:49	INFO	Detected standard HttpService. Filters disabled.		
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[javax.servlet.ServletContext]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[org.osgi.service.http.HttpService, org.ops4j.pax.web.service.WebContainer]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[javax.servlet.ServletContext]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[org.osgi.service.cm.ManagedService]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[org.ops4j.pax.web.service.spi.ServerControllerFactory]	
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	BundleEvent STARTED		
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[org.osgi.service.cm.ManagedService]	
24/6/2013 17:38:49	INFO	ServiceEvent REGISTERED	[org.apache.felix.webconsole.ConfigurationPrinter]	

Figura 59 - Serviço de Log



Bundles | Configuration Status | Licenses | Log Service | OSGi Repository | Services | Shell | System Information

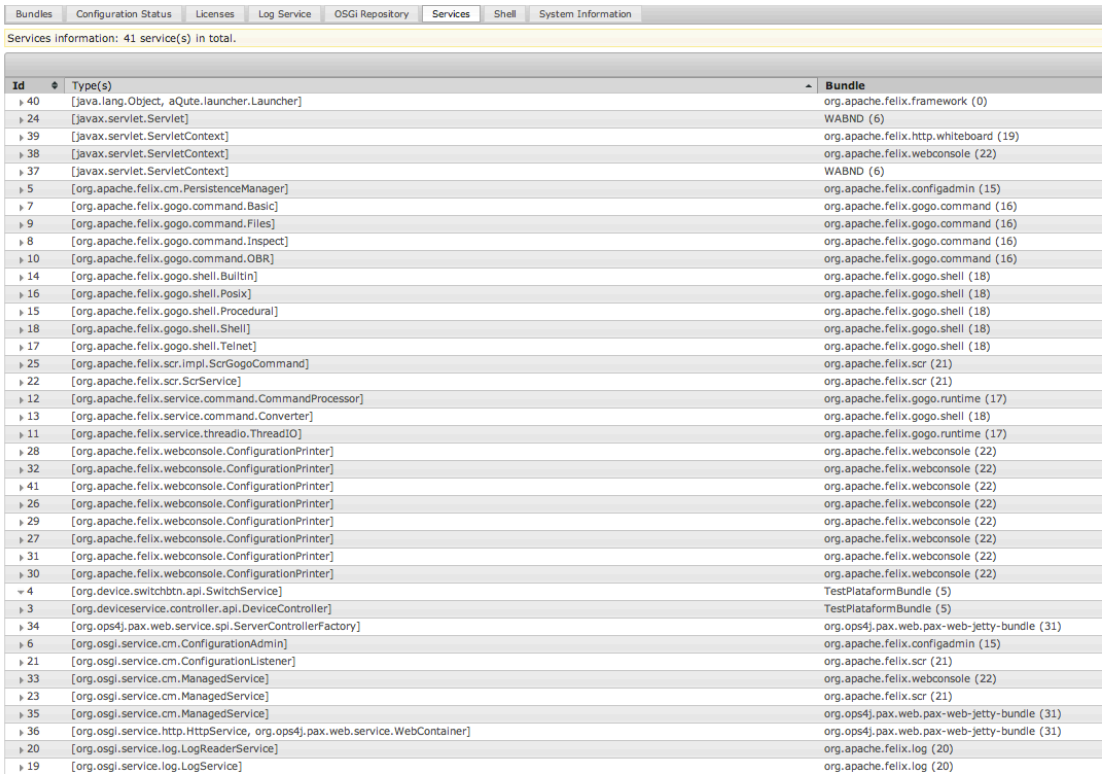
System is up and running!

Start Level Information:	
System Start Level	1 Change
Default Bundle Start Level	1 Change

Server Information:	
Last Started	24/6/2013 17:38:58
Framework	Restart Stop

Java Information:	
Java Runtime	Java(TM) SE Runtime Environment(build 1.6.0_37-b06-434-10M3909)
Java Virtual Machine	Java HotSpot(TM) 64-Bit Server VM(build 20.12-b01-434, mixed mode)
Number of Processors	4
Total Memory	83008
Used Memory	18730
Free Memory	64278
Garbage Collection	Run

Figura 60 - Informação do sistema



Bundles | Configuration Status | Licenses | Log Service | OSGi Repository | Services | Shell | System Information

Services information: 41 service(s) in total.

Id	Type(s)	Bundle
40	[java.lang.Object, aQute.launcher.Launcher]	org.apache.felix.framework (0)
24	[javax.servlet.Servlet]	WABND (6)
39	[javax.servlet.ServletContext]	org.apache.felix.http.whiteboard (19)
38	[javax.servlet.ServletContext]	org.apache.felix.webconsole (22)
37	[javax.servlet.ServletContext]	WABND (6)
5	[org.apache.felix.cm.PersistenceManager]	org.apache.felix.configadmin (15)
7	[org.apache.felix.gogo.command.Basic]	org.apache.felix.gogo.command (16)
9	[org.apache.felix.gogo.command.Files]	org.apache.felix.gogo.command (16)
8	[org.apache.felix.gogo.command.Inspect]	org.apache.felix.gogo.command (16)
10	[org.apache.felix.gogo.command.OBR]	org.apache.felix.gogo.command (16)
14	[org.apache.felix.gogo.shell.Builtin]	org.apache.felix.gogo.shell (18)
16	[org.apache.felix.gogo.shell.Posix]	org.apache.felix.gogo.shell (18)
15	[org.apache.felix.gogo.shell.Procedural]	org.apache.felix.gogo.shell (18)
18	[org.apache.felix.gogo.shell.Shell]	org.apache.felix.gogo.shell (18)
17	[org.apache.felix.gogo.shell.Telnet]	org.apache.felix.gogo.shell (18)
25	[org.apache.felix.scr.impl.ScrGogoCommand]	org.apache.felix.scr (21)
22	[org.apache.felix.scr.ScrService]	org.apache.felix.scr (21)
12	[org.apache.felix.service.command.CommandProcessor]	org.apache.felix.gogo.runtime (17)
13	[org.apache.felix.service.command.Converter]	org.apache.felix.gogo.shell (18)
11	[org.apache.felix.service.threadio.ThreadIO]	org.apache.felix.gogo.runtime (17)
28	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
32	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
41	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
26	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
29	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
27	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
31	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
30	[org.apache.felix.webconsole.ConfigurationPrinter]	org.apache.felix.webconsole (22)
4	[org.device.switchbtn.api.SwitchService]	TestPlataformBundle (5)
3	[org.deviceservice.controller.api.DeviceController]	TestPlataformBundle (5)
34	[org.ops4j.pax.web.service.spi.ServerControllerFactory]	org.ops4j.pax.web.pax-web-jetty-bundle (31)
6	[org.osgi.service.cm.ConfigurationAdmin]	org.apache.felix.configadmin (15)
21	[org.osgi.service.cm.ConfigurationListener]	org.apache.felix.scr (21)
33	[org.osgi.service.cm.ManagedService]	org.apache.felix.webconsole (22)
23	[org.osgi.service.cm.ManagedService]	org.apache.felix.scr (21)
35	[org.osgi.service.cm.ManagedService]	org.ops4j.pax.web.pax-web-jetty-bundle (31)
36	[org.osgi.service.http.HttpService, org.ops4j.pax.web.service.WebContainer]	org.ops4j.pax.web.pax-web-jetty-bundle (31)
20	[org.osgi.service.log.LogReaderService]	org.apache.felix.log (20)
19	[org.osgi.service.log.LogService]	org.apache.felix.log (20)

Figura 61 - Serviços no sistema