André Lagarto Ramião
Licenciatura em Ciências da Engenharia Eletrotécnica e de Computadores

Sistema Ciber-Físico de Produção Modular usando Raspberry Pi

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Doutor, FCT-UNL

Co-orientador: André Dionísio Bettencourt da Silva Parreira Rocha, Mestre, UNINOVA/CTS

Júri:

Presidente: Doutor José Manuel Matos Ribeiro da Fonseca, FCT/UNL

Vogais: Doutor José António Barata de Oliveira, FCT/UNL

Doutora Anikó Katalin Horváth da Costa, FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 2017

Sistema Ciber-Físico de Produção Modular usando Raspberry Pi

A todos os que me permitiram chegar a este ponto…

Agradecimentos

As palavras que se seguem são dedicadas a todos os que, sem os quais, a realização desta dissertação não teria sido possível.

Ao meu orientador, Professor Doutor José Barata, um obrigado por me facultar um tema que foi ao encontro dos meus interesses. Graças a si, todo o processo de pesquisa e realização da tese ficou assim muito mais fácil e interessante.

Ao meu coorientador, André Rocha, gostaria de agradecer por todo o apoio, paciência e dedicação durante este processo. Sei que não sou das pessoas mais fáceis de lidar, mas, no fim, sempre estiveste ali para me amparar.

A todos os diariamente presentes no local de trabalho, nomeadamente Pedro Monteiro, Mafalda Rocha, Ricardo Peres, Duarte Alemão, Paulo Ilhéu e os restantes presentes diariamente que sempre me ajudaram no que podiam e me proporcionaram um ótimo ambiente de trabalho, um sincero obrigado.

A todos os meus amigos, nomeadamente Andreia Bastos, Bernardo esteves, João Feliciano, Jéssica Simões, Ana Pereira, Bruna Bruno, Catarina Vera, Fábio Pires, Gisela Seixas, Nuno prata, Pedro Correia, Ricardo Barata, Tomás Deus, João Gomes, Cátia Botas, Inês Fernandes, Fábio Fernandes, Daniel Almeida, Lino Estevão, Mafalda Simões, Tânia Alves, Tiago Silva, Miguel Silvestre, Catarina Silva, Laura Santos, Raquel Faquinha, Adriana Miguel, Cátia Fernandes, Miguel Fernandes, João Rosário, Ricardo Guerreiro, Vítor Rodrigues, Rui Pacheco, Flávio Silva, João Mendes, Tiago Miranda, Ana Coutinho, entre outros que têm-me vindo a apoiar e aturar desde sempre. São a família que escolhi e não me arrependo um único dia! Um grande obrigado a todos vocês pelo apoio incondicional e pela motivação que me proporcionaram não só na duração da dissertação, mas em todo o tempo que nos conhecemos.

À minha família da faculdade, nomeadamente Pedro Corista, Eva Mendes, Tiago Nunes, Nídia Heleno, Bruno Rêga, Nuno Carvalhão e Manuel Faustino. Dizem que a família não se escolhe, mas neste caso era incapaz de recusar qualquer um de vocês! Obrigado por serem o tipo de pessoas que são. Tornaram toda esta experiencia muito mais fácil e divertida e são amigos que levo comigo para sempre.

Gostaria de deixar ainda um obrigado especial a duas pessoas que me têm vindo a acompanhar em todo o meu percurso e, para além de amigos, têm sido como mentores para mim, Carlos Fortunato e José Elloy. Quero agradecer-vos por todo o apoio e ajuda que me facultaram, sem vocês não estaria onde estou hoje!

Por fim e não menos importante, gostaria de agradecer a toda a minha família. Posso considerar-me sortudo ter nascido e ter sido criado nesta família que sempre me apoiou e repreendeu

sempre que necessário. Todas as lições recebidas não foram em vão e não era possível ter chegado onde cheguei sem todos vocês. Um obrigado do fundo do meu coração por tudo.

Resumo

A procura por produtos personalizados cresceu significativamente nos últimos anos o que exigiu profundas alterações nos processos de produção da indústria tradicional, de forma a que passassem a existir linhas de produção adaptadas, dinâmicas e flexíveis, por outras palavras, linhas capazes de produzir produtos personalizados em larga escala.

A adaptação dos formatos de produção foi um desafio para os produtores cujo os custos elevados e soluções limitadas tornaram-se num problema que vários estudos tentaram resolver. O avanço tecnológico permitiu explorar soluções viáveis mais simples e eficazes.

O objetivo deste projeto é desenhar um módulo de produção a um custo reduzido capaz de abstrair um componente de produção e oferecer uma solução de controlo flexível. Isto consegue-se através da integração de tecnologias da informação com a engenharia de controlo nos processos de produção, assentando em IoT e sistemas Ciber-Físicos. Propõem-se a utilização de uma arquitetura modular para gerir e controlar descentralizadamente os sistemas de produção com tecnologia *Plug&Produce*.

Concluindo, elegeu-se um Raspeberry Pi 3 (Modelo B) como módulo viável para a implementação de uma arquitetura onde um sistema multiagente é capaz de controlar vários recursos. A solução final foi testada num kit de demonstração que representa uma linha de produção a 24V.

Palavras-Chave: Internet das coisas; Sistemas Ciber-fisicos; Sistemas Multiagente; Plug&Produce; Sistemas de Produção; Raspeberry Pi.

Abstract

The search for personalized products has grown significantly in the last years, demanding profound changes in the way production processes exist in the traditional industry, going towards the existence of adaptable, dynamic and flexible production lines, in other words, lines able to produce personalized products in large scale.

The adaptation of production formats was a challenge for producers. Elevated costs and limited solutions soon become a problem that several studies attempted to solve. Technological advances allowed the exploration of simpler, more viable efficient solutions.

The objective of this project is to design a production module at a reduced cost that can abstract a production component and offer a flexible control solution. This is achieved through the integration of information technologies with the controlling engineering in the production processes, basing on IoT and Cyber-Physical systems. Proposing the use of Modular architecture to non-centrally manage and control the production systems with technology of Plug&Produce.

In conclusion, a Raspberry PI 3 (Model B) was elected as a viable module for the implementation of a multiagent system architecture with multi resource control. The final solution was tested in a demonstration kit that represents a production line of 24 volts.

Keywords: Internet of Things; Cyber Physic Systems; Multi Agent Systems; Plug&Produce; Production Systems; Raspeberry Pi.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

ADACOR	*Adaptive Holonic Control Architecture*
BMS	*Bionic Manufacturing Systems*
CLA	*Coalition Leader Agent*
CPPS	*Cyber -Physical Production Systems*
CPS	*Cyber -Physical Systems*
DA	*Deployment Agent*
DF	*Directory Facilitator*
DPWS	*Device Profile for Web Services*
EPS	*Evolvable Production Systems*
FIPA	*Foundation for Intelligent Physical Agents*
HA	Habilidade Atómica
HC	Habilidade Complexa
HIoT	*Human IoT*
HMS	*Holonic Manufacturing System*
HUA	*Handover Unit Agent*
IADE	*IDEAS Agent Development Environment*
IDEAS	*Instantly Deployable Evolvable Assembly Systems*
IIoT	*Industrial IoT*

IMS	*Intelligent Manufacturing Systems*
IoT	*Internet of Things*
JADE	*Java Agent Development Framework*
MAS	*Multi-Agent System*
OLE	*Object Linking and Embedding*
OPC	*OLE for Process Control*
PASOA	*Product Agent*
PSiA	*Product Sink Agent*
PSoA	*Product Source Agent*
RA	*Resource Agent*
RMS	*Reconfigurable Manufacturing System*
SOA	*Service-Oriented Architecture*
TEA	*Transport Entity Agent*
UA	*Unified Architecture*
UPnP	*Universal Plug and Play*
WSD	*Web Services on Devices*
YPA	*Yellow Pages Agent*

1 Introdução

O conceito de produção em massa permitiu solucionar o excesso de procura no mercado, sendo agora possível produzir grandes quantidades homogéneas do mesmo produto e reduzir os seus custos de produção. Porém, esta realidade tem como consequência a diminuição de produtos personalizados e únicos, ao contrário do que acontecia antigamente onde a relação do cliente com o produto era muito mais próxima, pois cada produto era feito individualmente segundo as especificações do cliente.

Esta "revolução industrial" começou com Henry Ford, um empreendedor e empresário americano que deixou a sua marca na história do mundo por ter sido pioneiro a aplicar métodos de montagem em série nas suas fábricas de automóveis. Henry ambicionava que o automóvel fosse um bem acessível a todos os indivíduos independentemente do seu estatuto social e, como tal, tinha de arranjar forma de produzir rapidamente mais automóveis e automóveis mais baratos. No entanto, Henry manteve a experiência de comprar um automóvel bastante impessoal. O mesmo referia que os clientes poderiam ter um carro de qualquer cor que eles quisessem, desde que fosse a cor preta. Como tal, a relação de proximidade entre produto e cliente estava a ser parcialmente valorizada uma vez que os clientes não tinham grande opção de escolha, podendo escolher o produto, mas não podendo personalizá-lo (Wilson, 2014).

No entanto esta realidade é contraditória porque para possibilitar a diversidade e personalização do produto, é necessário alterar constantemente a linha de produção encarecendo assim o custo de produção do produto e consequentemente o seu preço. Portanto, de modo a evitar este problema, é plausível concluir que é necessária uma linha de produção que seja flexível e dinâmica, permitindo atualizar os seus recursos de acordo com o produto a produzir.

A evolução da tecnologia e a sua adaptação na indústria tem sido fulcral na modelação deste tipo de problemas, permitindo a que nos dias que correm existam várias alternativas para estes casos, como a adaptação das plataformas existentes ou abordagens através de arquiteturas orientadas a serviços ou ainda modulação através de agentes onde existe uma rede de cooperação e corelação entre os recursos dos sistemas da linha de montagem.

As implementações multiagente são bastante comuns na implementação destes modelos tendo em conta que permitem ter aproveitamento ideal do conceito de agente (entidade reativa e proativa com inteligência, perceção parcial e capacidades sociais, autónomas e adaptativas). No entanto, estas ainda contêm algumas limitações referentes ao processamento em tempo real impedindo assim o seu crescimento no meio industrial. Para mais, as que existem exigem ao produtor poder económico para suportar os atuais custos associados à implementação destas novas soluções.

1.1 Questões e Hipóteses

Previamente é transmitida a ideia de que os atuais sistemas de manufatura não são perfeitos, havendo ainda muitos caminhos por estudar e muito espaço para melhorar, tendo sempre em conta que tanto o custo de produção, bem como o preço ao consumidor são prioridades. Mas refletindo também sobre as atuais soluções apresentadas a cima, é possível elaborar a seguinte questão:

- Será possível desenhar um módulo de produção a um custo reduzido capaz de abstrair um componente de produção e oferecer uma solução de controlo flexível?

É então colocada a hipótese de um sistema com base numa arquitetura multiagente, ou seja, que beneficia do comportamento proactivo de um agente, permitindo cooperação, adaptabilidade e autonomia, e aplica-o a uma plataforma acessível, mas com poder de processamento suficiente para o suportar.

1.2 Delineamento da Hipótese

Uma vez Identificados alguns dos problemas associados aos sistemas industriais atuais, é possível delinear uma hipótese. E tal como referido anteriormente, esta hipótese tem base numa plataforma poderosa e barata com uma arquitetura baseada num sistema multiagente, do inglês *Multi-Agent System* (MAS).

O *hardware* correspondente a esta plataforma deverá corresponder às seguintes especificações técnicas: ter várias portas de comunicação para que seja possível comunicar com outras

partes do sistema, e ter um microprocessador com relação qualidade-preço adequada a cumprir todos os requisitos do projeto já mencionados. Assim para este projeto foi elegida para plataforma física o microcomputador *Raspeberry Pi* auxiliado de uma pequena placa capaz de converter sinais da linha de produção de 24 Volts para 3.3 Volts (valor de tensão comum numa linha de montagem e valor de tensão do *Raspeberry Pi* respetivamente) e vice-versa.

Por cima do *hardware* estará uma plataforma baseada em multiagente tirando proveito das vantagens do agente referidas previamente neste capítulo. O principal problema da implementação dos sistemas multiagentes num contexto real são as suas limitações ao nível de poder de processamento o que, neste caso, é compensado pelo *Raspeberry Pi.* A arquitetura proposta tem em conta que os sistemas industriais atuais sofrem de algumas protuberâncias e, como tal, prevê situações que são recorrentes como falhas no sistema e a adição ou remoção de novos recursos ao longo do processo de produção. Estas são as vantagens de uma arquitetura multiagente.

O sistema será ainda testado num kit de demonstração. Este irá servir de simulador de um ambiente industrial dando assim uma aproximação mais exata dos resultados face a um sistema real.

1.3 Principais Contribuições

A solução proposta neste projeto oferece um novo conjunto de opções para controladores industriais de baixo custo permitindo a pequenas e médias empresas entrar no mercado com um investimento inicial menor.

A solução apresentada passa apenas por uma configuração possível de aplicar sobre o *harware* escolhido deixando uma grande margem para modificações de acordo com a necessidade do produtor abrangendo assim um grupo maior de produtores.

1.4 Delineamento do documento

A tese é composta por oito capítulos: Introdução, Estado de Arte, Conceitos de Suporte, Arquitetura, Implementação da Solução, Validações e Testes, Conclusões e Trabalho Futuro e Referências.

O primeiro e presente capitulo expõe uma breve descrição do problema e da pesquisa associada ao problema, bem como um delineamento da hipótese apresentada e as suas vantagens.

No Estado de Arte é resumido o contexto da tese passando por conceitos, soluções e projetos existentes. Tópicos como Internet das Coisas, Sistemas Ciber-Físicos e Sistemas Inteligentes de Manufatura serão o foco deste capítulo.

No terceiro capítulo, Conceitos de Suporte, à semelhança do anterior e como sugere o seu nome, irá descrever alguns conceitos importantes no contexto do trabalho realizado como modularidade ou *Plug and Produce*.

No capitulo de Arquitetura, é apresentada em detalhe a arquitetura da solução bem como o seu comportamento, ou seja, é descrita uma solução com bases tanto físicas como computacionais para o problema apresentado na Introdução.

Na Implementação da Solução, é apresentada a implementação da arquitetura detalhada no capítulo anterior. As entidades que compõem o sistema, as interações entre elas, as tecnologias e os dispositivos utilizados também são detalhados.

Em Validações e Testes são apresentados dois tipos distintos de teste: o primeiro, onde é utilizado um kit de simulação de uma linha de produção, e o segundo, onde é realizado um conjunto de testes de performance. São apresentados e comentados os resultados obtidos.

Por fim, o capítulo Conclusões e Trabalho Futuro é uma visão crítica do trabalho desenvolvido para esta tese e possíveis diretrizes de pesquisa futuras.

2 Estado de Arte

Os sistemas de manufatura têm sofrido alterações ao longo do tempo com o intuito de seguir as tendências de mercado. Estes têm agora um novo conjunto de requerimentos para conseguirem acompanhar estas mudanças, nomeadamente a redução dos ciclos de vida dos produtos e do tempo de colocação dos produtos no mercado, assim como uma maior flexibilidade nos números de produção e a redução dos custos de produção (Schäfer, 2007).

No entanto, não foi unicamente a procura do mercado que mudou. Com o aparecimento de novas tecnologias é agora possível criar novas e mais vantajosas soluções para a forma como todo o sistema de manufatura opera , principalmente na área de controlo (Schäfer, 2007).

É então fundamental que as tecnologias de informação sejam integradas com a engenharia de controlo nos processos de produção, assim para esse efeito é imprescindível que os sistemas sejam capazes de comunicar e operar em conjunto (Xu, Fang and Li, 2016) .

Assim conceitos como *Internet of Things*, onde o foco se encontra na ligação de milhões de objetos físicos à Internet, e *Industry* 4.0 que pressupõe uma comunicação e cooperação seguras de todos os participantes em tempo real ao longo de todo o processo, ganham terreno e estão cada vez mais presentes na sociedade. (Han *et al.*, 2015; Xu, Fang and Li, 2016).

Contudo os estudos existentes são ainda muito divergentes, oferecendo enumeras opções teoricamente viáveis, e da realidade, onde toda a complexidade existente na teoria ainda não é totalmente possível aplicar na pratica (Schäfer, 2007).

2.1 Internet das Coisas

A Internet das coisas, da expressão inglesa *Internet of Things (IoT)*, é o conceito que procura otimizar os meios de comunicação utilizados na sociedade, e que procura desenhar um futuro próximo onde o domínio das comunicações interpessoais será substituído na sua totalidade pela comunicação exclusiva entre dispositivos e para dispositivos, isto é, sem que exista intervenção ou interação humana (Aloi *et al.*, 2016).

A menção de IoT surgiu pela primeira vez em 1999, cuja aplicabilidade aos vários sectores de atividade (primário, secundário e terciário), foi imediatamente reconhecida e como tal surgiu também um forte investimento no desenvolvimento destas mesmas soluções que se mantém atualmente. Porém, apesar do forte desenvolvimento e de já existirem aplicações IoT funcionais no mercado não existem ainda protocolos de comunicação standard que uniformizem o desenvolvimento desta tecnologia (Zhong, Zhu and Huang, 2016).

A IoT liga biliões de dispositivos como eletrodomésticos, câmaras de vigilância, sensores de monotorização, veículos, máquinas elétricas, etc., e é esperado que ligue muitos mais. IoT é um conceito versátil que teoricamente pode ser aplicado a qualquer coisa, sempre com o objetivo de tornar cada processo mais inteligente e em constante comunicação com a rede – Internet, a tendência é que o fluxo de informação disponível é proporcional ao número de dispositivos IoT (Aloi *et al.*, 2016; Razzaque *et al.*, 2016).

A definição de IoT é simples de compreender, mas a sua aplicabilidade e capacidade poderá não ser tão intuitiva, pelo que pode ser linearizada sendo então dispositivos IoT percecionados como recetores de informação. Estes recetores utilizam protocolos de comunicação existentes e que estão acordados (3G, LTE, 5G, Wi-Fi, LTN), para alcançarem sempre que solicitado, em tempo real e independentemente da localização, qualquer troca de dados para qualquer finalidade e sempre em prol do gerenciamento independente de uma rede (Aloi *et al.*, 2016; Zhong, Zhu and Huang, 2016).

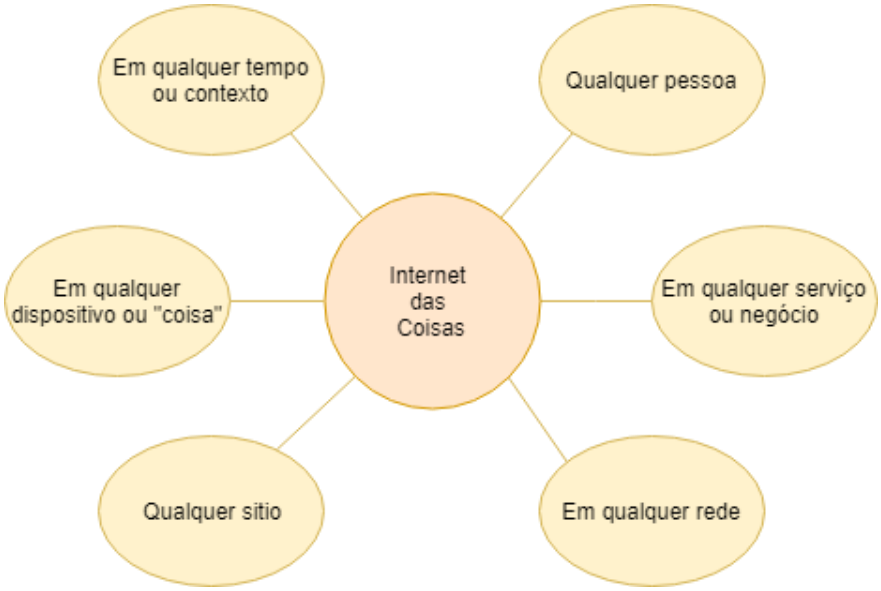

Figura 2.1 - Definição de IoT

A figura 2.1 descreve sucintamente o funcionamento por de trás do conceito IoT, que como referido anteriormente neste capítulo tem e terá um impacto ainda mais forte em vastas áreas associadas à tecnologia, à saúde e ao negócio, sejam considerados exemplos as atividades de domótica, de automação industrial, de auxiliares médicos, de unidades de cuidados de saúde móveis, assistência a idosos, gestão inteligente do trânsito nas cidades, entre muitos outros (Fig. 2.2). Todas estas aplicações utilizam uma enorme quantidade de dados que são gerados por estes dispositivos permitindo oferecer aos cidadãos, às empresas e às administrações públicas novos serviços (Razzaque *et al.*, 2016).

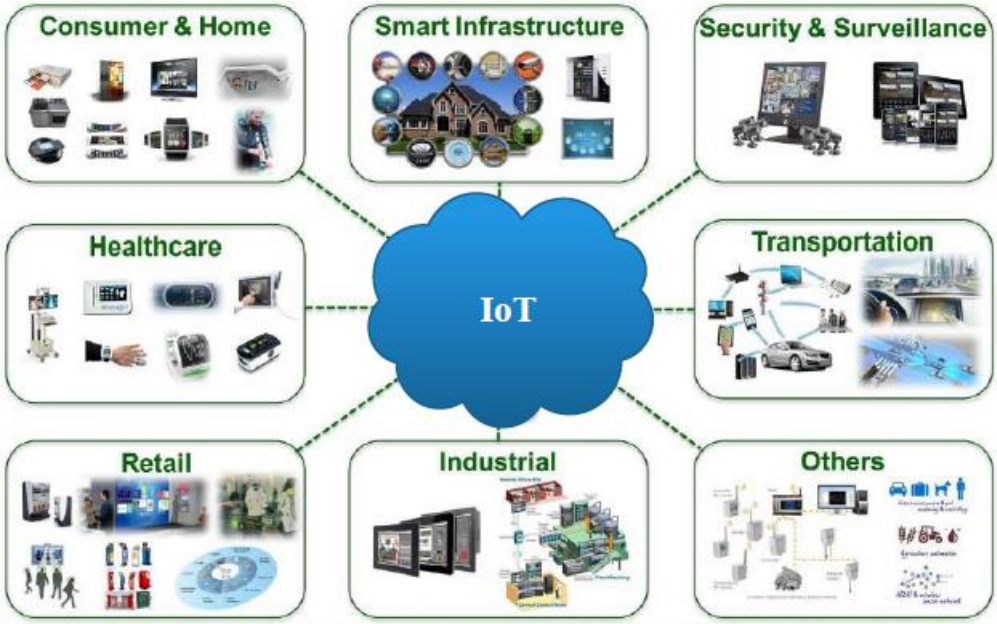

Figura 2.2 - Possíveis aplicações de IoT (Razzaque *et al.*, 2016)

É possível olhar para IoT através de três pontos de vista: 1. IoT orientada para a rede, 2. IoT orientada para as coisas (dispositivos, sensores, etc.), e 3. IoT orientada para o conhecimento ou aprendizagem. Poderia ainda ser considerado um quarto ponto de vista se olhássemos para IoT com orientação para o suporte de aplicações dos consumidores (*Human IoT* - HIoT) ou para o suporte de aplicações industriais (*Industrial IoT* - IIoT), no entanto esta perspetiva é muitas vezes negligenciada porque HIoT e IIoT são categorias que se desenvolvem de forma separada (Razzaque *et al.*, 2016).

A maior parte dos artigos disponíveis que referenciam a definição de IoT não destacam explicitamente a visão industrial do IoT, mas é algo muito presente e cada vez mais importante. Empresas líderes a nível mundial investem fortemente na procura de soluções industriais que apliquem arquiteturas IoT, e embora utilizem termos diferentes para as suas descobertas: IBM – "Planeta Inteligente", Cisco – "*Internet of Everything*", ou GE – "*Industrial Internet*", o seu objetivo é sempre utilizar IoT para melhorar os seus processos industriais (reduzir os períodos de inatividade das máquinas e reduzir o custo do consumo energético) (Razzaque *et al.*, 2016).

2.2 Sistemas Ciber-Físicos

Sistemas Ciber-Físicos, na literatura apresentados como *Cyber-Physical Systems* (CPS), são uma nova geração de sistemas dotados de capacidades computacionais e físicas capazes de interagir com humanos. Esta capacidade de expandir os recursos dos sistemas para o mundo físico através da computação, comunicação e controlo constituí uma chave para desenvolvimentos futuros da tecnologia. Atualmente são objetos de pesquisa e investigação aviões de última geração, veículos espaciais, veículos elétricos ou híbridos que conduzem de forma autónoma e próteses que comunicam com o cérebro dos seus utilizadores (Baheti and Gill, 2011). Existe, portanto, para CPS tanta aplicabilidade como existia para IoT na secção anterior, porém devido à existência de elevada confidencialidade à volta destes sistemas muitas vezes estes acabam por não ser do conhecimento público (ZHANG *et al.*, 2013).

Sistemas de Produção Ciber-Físicos, do inglês *Cyber-Physical Production Systems* (CPPS), são no fundo a aplicação dos CPS no contexto da produção industrial, a Industria 4.0 na Alemanha e o US Industrial Consortium são exemplos de projetos conhecidos que trabalham ou desenvolvem atualmente com a tecnologia CPPS. O CPPS oferece uma grande vantagem no sentido em que também é um conceito flexível capaz de se adaptar rapidamente e eficientemente a novas topologias de plantas ou novos produtos (Otto, Vogel-Heuser and Niggemann, 2017).

Os CPPS integram e constroem uma variedade de tecnologias e componentes existentes, como robótica, automação e controlo industrial, IoT, dados importantes e computação em nuvem. (Cloud) e, a integração desta tecnologia, pode ser observada por três perspetivas: horizontal, vertical e de ponta-a-ponta. Colocado de forma simples os CPPS interligam sistemas industriais com CPS para otimizar a capacidade de produção e automação de uma fábrica - A Industrie 4.0 procura um sistema de operação autónomo com a personalização em massa de produtos, processos de fabrico colaborativos e integração digital de ponta-a-ponta (Ma *et al.*, 2017).

Outra vantagem, ou ideia por de trás do CPPS é o CPPS adaptativo cuja inovação assenta sobre a modularidade e reutilização do sistema, isto é, as plantas de produção têm de ser construídas sobre topologia *plug-and-work* e através da agregação de módulos predefinidos e componentes de maquinaria, como robôs, transportadoras, compressores, etc. Sobre um ponto de vista de automação, as unidades de produção modulares devem ser baseadas em software de automação modular, ou seja, o software utilizado deve ser agregado a partir de elementos de software que correspondem a módulos físicos predefinidos (Otto, Vogel-Heuser and Niggemann, 2017).

Por outro lado, isto significa também que estes softwares de automação modular necessitam de parâmetros bem delineados e um conjunto de soluções alternativas que permitam atender novos requisitos e novos cenários de produção sempre que solicitado, é necessário fornecer softwares modulares com maior grau de liberdade. A Figura 2.3 mostra a ideia principal da adaptação deste tipo de software, onde se pode observar que os componentes de software são agregados de acordo com a topologia modular da planta (Otto, Vogel-Heuser and Niggemann, 2017).

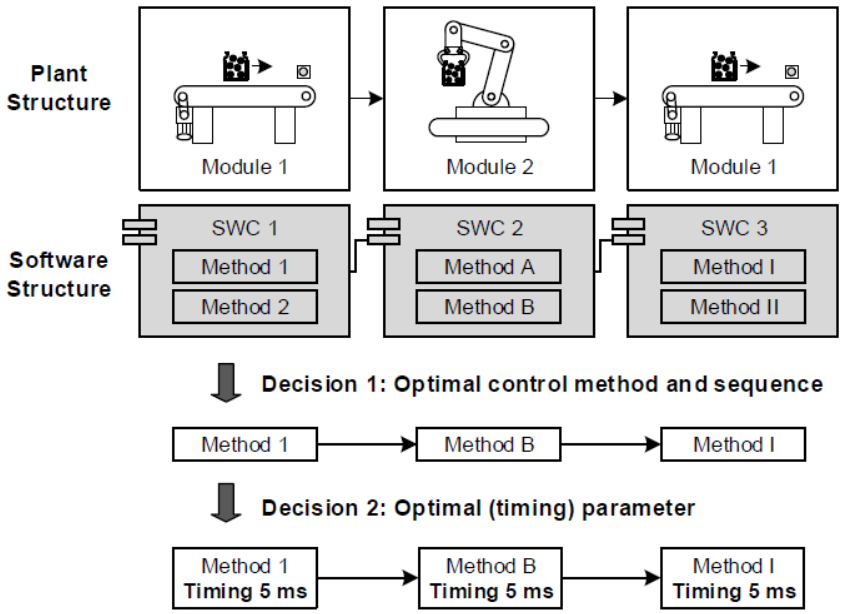

Figura 2.3 - Exemplo de CPPS (Otto, Vogel-Heuser and Niggemann, 2017)

Por exemplo, um robô pode utilizar um comando de controlo x para executar movimentos simples e utilizar um controlo y para executar movimentos mais complexos, ou um reator químico pode usar uma fórmula z para produzir uma certa matéria-prima e um método m totalmente diferente para produzir outro material. É fácil ver que todos os comandos de controlo que pertencem a um elemento de software convergem, geralmente, resultados semelhantes. Assim, a primeira decisão que tem de ser tomada é qual o método de controlo escolhido para aquela situação específica, e a segunda decisão a ser tomada deverá ser colocar todos os métodos escolhidos na sequência certa que resulta no produto requerido. E como já referido, cada método de controlo deverá ter parâmetros adaptáveis (tempo, velocidade ou temperatura) que devem ser regulados para cada método de controlo e para cada meta de otimização (Otto, Vogel-Heuser and Niggemann, 2017).

2.3 Sistemas Inteligentes de Manufatura

A introdução da informática no mundo industrial veio redefinir o processo de produção e a grande demanda de produtos personalizados veio exigir sistemas capazes de se adaptarem constantemente mantendo a eficiência de produção.

O planejamento do processo de manufatura determina a forma de fabrico de um produto. É um processo de seleção e sucessão de processos e parâmetros de manufatura, de modo a que uma ou mais metas sejam alcançadas, tais como menor custo de produção, menor tempo de processamento, etc, enquanto satisfaz um conjunto de restrições (Ligong, Zude and Quan, 2008).

A fim de atingir os objetivos anteriormente mencionados, os sistemas de manufatura devem ser mais precisos, ágeis, interoperáveis, reconfiguráveis e geralmente inteligentes através da inovação contínua (Huang, Solvang and Yu, 2016). O uso de sistemas convencionais com base em arquiteturas hieráticas com decisão centralizada, insuficiência em dinamismo e flexibilidade é agora obsoleto dando lugar a novos modelos que partem da adaptação dos presentes à criação de novos sistemas inteligentes de manufatura, do inglês *Intelligent Manufacturing Systems* (IMS).

Vários projetos com base nestes paradigmas usam ainda um MAS, ou seja, uma rede de agentes que interagem entre si dentro de um sistema com o objetivo de distribuir as tarefas entre si, resolvendo problemas que um sistema monolítico teria dificuldade em realizar ou não conseguiria de todo.

As subsecções seguintes irão abordar quatro destes paradigmas emergentes no ambiente de manufatura.

2.3.1 Sistemas Biónicos de Manufatura

A definição dos sistemas biónicos é comparável à definição dos sistemas biológicos que se definem como sistemas naturais complexos capazes de exibir comportamentos autónomos, espontâneos e que obedecem a uma estrutura hierárquica bem definida (Tharumarajah, Wells and Nemes, 1996). Portanto, os sistemas biónicos de manufatura - *Bionic manufacturing Systems* (BMS), são capazes de aproveitar todas as características mencionadas para os sistemas biológicos e adaptá-las aos ambientes fabris onde a reconfiguração de sistemas é de supra importância (Li, Ying and Qing, 2006).

O facto de estes sistemas deterem uma estrutura hierárquica bem definida significa que, tal como os seres vivos são definidos por estruturas ascendentes de complexidade (células, órgãos, vida e sociedade), também estes sistemas são estruturados semelhantemente, onde as células biológicas correspondem às unidades de produção mais simples ou mais pequenas, os órgãos a grupos de unidades e assim por adiante. Cada nível hierárquico suporta o nível superior, o que por sua vez significa que a ação espontânea de uma unidade isolada não ativa nenhuma função do sistema, todo o funcionamento do sistema é sempre feito em cadeia, passando por cada camada (Tharumarajah, Wells and Nemes, 1996).

A diferença dos sistemas biónicos para o tipo de sistemas anteriormente referidos reside na adaptação dos processos de produção de produtos através de funcionalidades de autorreconhecimento, auto-crescimento, autorrecuperação e evolução (Lazinica and Katalinic, 2005)(Ueda, Vaario and Ohkura, 1997).

Assim, é notório o enorme potencial dos sistemas bio-inspirados na resolução de problemas complexos de engenharia aliados aos MAS, resultam em sistemas altamente uteis no domínio da manufatura, a fim de enfrentar os desafios atuais.

2.3.2 Sistemas Reconfiguráveis de Manufatura

Respostas financeiras às mudanças do mercado requerem sempre uma avaliação dos processos de fabricos, e se for o caso da implementação de novas abordagens que permitam reagir a essas mudanças de forma rápida e eficiente. Por norma estas abordagens seguem os seguintes pontos (Koren *et al.*, 1999):

- Novo design para o sistema e respetivas máquinas que permita o seu ajuste consoante a demanda do mercado e adaptabilidade do sistema a novos produtos. Este ajuste pode ser feito quer a nível do sistema, quer a nível das máquinas onde é alterado o hardware das máquinas e o seu software de controlo;
- Novo design do sistema de produção com foco na produção do portfólio de produtos, com espaço para incluir os processos de personalização necessários para a produção do portfólio, o que consequentemente reduz o custo total dos sistemas.

Esta adaptação de elevado nível pode ser atingida pelo desenvolvimento e implementação de Sistemas de Manufatura Reconfigurável (RMS), estes sistemas são baseados em equipamentos

modulares que facilmente podem ser integrados, ajustados e convertidos em termos de funcionalidades e cenários (Koren *et al.*, 1999).

O RMS começou como um conceito de produção intermediária que funcionava como ponte entre as linhas de produção tradicionais e as linhas de produção dinâmicas ou flexíveis. A diferença do RMS quando comparado aos sistemas tradicionais reside no facto de o primeiro poder ser dimensionado em termos de capacidade e parâmetros varáveis, e de ser constituído por módulos que se complementam sempre visando o baixo custo de implementação e de produção (Andersen, Nielsen and Brunoe, 2016). Os RMS deverão ser desde início projetados para serem reconfiguráveis (com máquinas e software igualmente reconfiguráveis), rápidos e económicos.

Uma vez que os RMS permitem o seu dimensionamento em função da capacidade, faz com que estes sistemas sejam muito úteis tanto para as linhas de produção de grandes volumes, como as linhas de produção nas indústrias automóvel, como para as linhas de produção de menor escala que utilizam um pequeno conjunto de máquinas ou apenas uma única máquina (Koren, 2005). Esta versatilidade impõe algumas restrições aos designers dos produtos que têm de criar cada produto tendo sempre em mente a estrutura e capacidades do sistema de produção, que tem de ser alterada para cada produto. A este processo é chamado projeto de produtos orientados a processos (Koren, 2005).

Novamente a ideia por de trás do conceito base destes sistemas permanece a mesma, isto é, independentemente do equipamento que seja adquirido é sempre possível manipulá-lo de forma a corresponder as necessidades do momento. Estas necessidades ou alterações podem estar como já referido na capacidade da produção ou nas funcionalidades dos equipamentos, mas também residirem na sequencialidade de tarefas realizadas entre máquinas, ou seja, as necessidades podem corresponder a níveis de hardware ou software (Koren, 2005).

Para garantir a modularidade, a mudança e a comunicação entre os vários recursos do processo de fabrico é importante ter um hardware e sistema de controlo fisicamente capazes de alcançarem os objetivos. A transição da pirâmide hierárquica tradicional para sistemas de controlo descentralizados é vital nos sistemas RMS, entre as tecnologias de habilitação mais importantes para a configuração ad hoc dos recursos de produção estão as IoT e os CPS (Lesi, Jakovljevic and Pajic, 2016).

De uma forma geral, é possível caracterizar um RMS em seis características (Makinde, Mpofu and Popoola, 2014):

- Convertibilidade: é o poder de transformar facilmente as funções dos sistemas e máquinas para atender os novos requisitos de produção;

- Escala: é a capacidade de modificar a capacidade de produção adicionando ou subtraindo processos/recursos de produção;
- Modularidade: é o encapsulamento de funcionalidades operacionais em unidades que podem ser manipuladas em qualquer tipo de esquemas de operação alternativos;
- Integrabilidade: é a capacidade de integrar módulos de forma rápida e precisa através de um conjunto de interfaces mecânicas, informativas e de controlo que visam otimizar a comunicação e coesão do todo;
- Personalização: é a competência de produzir produtos com características específicas segundo pedidos ou requisitos por parte dos clientes;
- Diagnóstico: estes sistemas têm a habilidade de se autodiagnosticarem, porque leem automaticamente o estado atual do sistema por forma a detetar as causas responsáveis pelo mau funcionamento das linhas de produção, como defeitos em produtos ou defeitos operacionais.

2.3.3 Sistemas Holónicos de Manufatura

O Sistema holónico de manufatura, do inglês *Holonic Manufacturing Systems* (HMS), é um paradigma que adapta para o mundo industrial os conceitos sobre organismos vivos e organizações sociais desenvolvidos por Arthur Koestler. A introdução do paradigma holónico permite uma nova abordagem aos problemas de manufatura, trazendo as vantagens de modularidade, descentralização, autonomia e escalabilidade. A produção holónica é caracterizada por holarquias, entidades autônomas e cooperativas (hólons), que representam toda a gama de entidades industriais. Por sua vez, tal como descrito por Koestler, um hólon é simultaneamente um todo e também uma parte do sistema (Leitão and Restivo, 2006).

Estes sistemas estão presentes na industria desde a década de 1980, quando investigadores japoneses desenharam e implementaram pela primeira vez um sistema holónico num controlador. Este apresentava como principais vantagens um design mais rustico, devido ao decréscimo na complexidade de ligações por cabo, e um aumento na segurança do dispositivo (Ceanu and Chen, 2006).

No inicio da década de 1990, as mudanças na procura eram já mais evidentes, por isso os sistemas holónicos surgiram como uma solução viável ao problema que estava a afetar todo o

setor. Estabeleceu-se assim um consórcio com investigadores da Austrália, Canadá, Europa, Japão e Estados Unidos dentro do programa IMS com o objetivo de desenvolver ferramentas e implementar o conceito holónico, até então praticamente teórico, na indústria de manufatura. Para orientar a pesquisa na área, os participantes do consórcio HMS estabeleceram uma série de definições de trabalho para as entidades integrantes dos sistemas holónicos (Ceanu and Chen, 2006):

- Hólon: um bloco construtivo autónomo e cooperativo de um sistema de fabrico para transformar, transportar, armazenar e /ou validar informações e objetos físicos. O hólon consiste numa parte de informação e muitas vezes numa parte de processamento físico, e um hólon pode fazer parte de outro hólon;
- Autonomia: a capacidade de uma entidade criar e controlar a execução de seus planos e / ou estratégias;
- Cooperação: um processo pelo qual um conjunto de entidades desenvolve planos mutuamente aceites e executa esses planos;
- Holarquia: um sistema de hólons que podem cooperar para atingir uma meta ou objetivo;
- HMS: uma holarquia que integra toda a gama de atividades de fabricação, desde a reserva de encomendas até o design, produção e marketing para realizar a empresa de fabricação ágil.

Já existem atualmente algumas abordagens referentes a este paradigma. Uma das mais recentes é a *ADAptive holonic COntrol aRchitecture* (ADACOR) aborda as alterações frequentes do sistema. Aqui são definidas quatro classes de hólons - *Product Holon, Task Holon, Supervisor Holon e Operational Holon* - onde existe uma auto-organização de cada um. É de destacar a existência do elemento supervisor com capacidade de coordenação descentralizada e responsabilidade pela formação de grupos hólons (Jorge *et al.*, 2004; Leitão and Restivo, 2006).

2.3.4 Sistemas Evolutivos de Produção

Tal como o HMS, o sistema evolutivo de produção ou *Evolvable Production Systems* (EPS) é um paradigma criado com o intuito de atender às contantes alterações de mercado de forma a facilitar o fabrico de produtos altamente personalizados, assim como também lidar com quaisquer irregularidades na linha de produção.

Este paradigma foca-se na adaptabilidade dos componentes do sistema tendo um controlo distribuído. É baseado numa arquitetura de referência, aberta e com modularização de nível baixo d'onde resulta uma granularidade fina. O comportamento inteligente é resultado das interações entre os distintos módulos (Frei, Barata and Onori, 2006; Rocha *et al.*, 2014).

Para criar um sistema EPS é necessário ter em conta alguns fatores, tais como(Frei, Barata and Onori, 2006):

- Modularidade: é necessária a existência de módulos distintos. Estes representam qualquer recurso ou ferramenta que possa executar uma operação;
- Granularidade: é essencial que se defina o nível de granularidade de cada módulo. Podem existir diferentes níveis e, neste caso, pode dar origem a todo um novo conjunto de operações possíveis;
- Conectividade: o sistema deve ter a capacidade de reorganizar e integrar componentes do sistema dentro da estrutura duma arquitetura do sistema. Esta ideia vai ao encontro do conceito de *Plug & Produce* retratado em 2.5.;
- Reconfigurabilidade: o sistema deve ser capaz de reagir perante mudanças no sistema de modo a realizar novas operações pré-definidas. Esta funcionalidade tira ainda proveito da conectividade do sistema como parte da possível solução face a mudança;
- Evolutividade: deve ser uma plataforma que contem um sistema mecatrónico totalmente "reconfigurável" que apresenta um comportamento emergente introduzindo assim funcionalidades de alto nível. Por forma a aplicar este conceito corretamente é necessária a existência de uma arquitetura de referencia.

Os benefícios do EPS são reconhecidos, e como tal foram desenvolvidos vários projetos que tentam tirar vantagem deste paradigma, como a abordagem FP7 IDEAS. Esta abordagem implementou o EPS com recurso a um MAS onde cada agente mecatrónico está aplicado a um módulo. Estes agentes são orientados a *skills*, os serviços que definem as ações. O sistema cria ainda uma representação virtual de um módulo - *Coalition Leader*, que representa um conjunto de agentes mecatrónicos, sendo este responsável pelos seus subordinados permitindo realizar ações mais complexas (Ribeiro and Barata, 2011).

2.4 Conclusões

Ao longo deste capitulo é possível verificar que as alterações na procura vieram afetar, em grande escala, os processos de produção. Pelo que surgiu a necessidade de estudar vários projetos que refletiam sobre viáveis soluções para os impasses.

Juntamente com estes estudos e o crescimento continuo da tecnologia foi possível adaptar muitas destas soluções teóricas ao mundo real, principalmente com o aparecimento de conceitos como os EPS, RMS, BMS e HMS que permitiram abolir o sistema de arquitetura centralizada tornando o processo de manufatura mais flexível e dinâmico.

Contudo existem ainda alguns elementos que demostram um certo défice de desenvolvimento, impedindo que algumas das novas soluções propostas não cheguem ao mercado ou que apresentem preços elevados impossibilitando que alguns produtores se interessem ou tenham acesso às mesmas.

O presente documento apresenta uma proposta de baixo custo que utiliza os conceitos a cima mencionados com o objetivo de fazer com o acesso a estas tecnologias seja facilitado a todos os interessados.

3 Conceitos de Suporte

3.1 Modularidade

A modularidade no conceito de sistemas de manufatura é a divisão de um sistema em subsistemas que podem interagir entre si. Cada subsistema corresponde a um módulo que pode representar elementos mecânicos, elétricos ou de controlo, e pode ainda conter outro módulo, por exemplo um robô pode ter uma garra e um braço.

A capacidade de subdividir um sistema é muito mais importante quando aliada à capacidade de um sistema se auto reconfigurar pois irá permitir que o sistema se torne mais flexível a qualquer mudança ou imprevisto.

Os módulos permitem ainda a diminuição do investimento e dos tempos de desenvolvimento uma vez que podem ser reutilizados. Além disto, em situações de avaria d'um sistema, os módulos avariados podem ser substituídos por suplentes diminuindo assim os custos de reparo bem como o tempo de inatividade do sistema global. Este tempo de inatividade pode chegar a ser nulo no caso de o sistema ser inteligente o suficiente para procurar alternativas por via de outros módulos instalados, citando um caso análogo da avaria de um robô o sistema procurará a alternativa mais viável dentro do seu conjunto de subsistemas para realizar a mesma função

Em suma e citando (Schäfer, 2007) , um módulo porção controlável, programável, reutilizável e reconfigurável de um sistema, com funcionalidades orientadas ao processo de interfaces especificas que executam funções específicas e podem ser utilizadas individualmente ou em combinado com outros módulos do sistema.

A modularidade em conjunto com a possibilidade de reconfigurar um sistema é algo cada vez mais presente nos sistemas modernos colocando quatro grandes tendências no mercado:

- Substituir o controlo centralizado por distribuído;

- Interfaces estandardizadas tomarem o lugar de interfaces próprias;
- A engenharia conduzida pelo processo de fabrico sobrepor-se à engenharia conduzida por elementos mecânicos;
- Os dispositivos eletrónicos individuais darem lugar aos dispositivos eletrónicos integrados em módulos.

3.2 Plug and Produce

Nos sistemas de montagem é frequente a necessidade de modificar a configuração dos seus dispositivos devido à inserção de novas máquinas, robôs, tapetes e alimentadores de peças, seja por estes sofrerem avaria ou simplesmente porque as necessidades se alteram (Arai *et al.*, 2001). Todavia, existem casos em que é necessário a reinicialização do sistema ou a substituição de dispositivos, de produtos e de processos. (Rocha *et al.*, 2014). Nestes casos, a reconfiguração propõe um longo tempo de inicialização em sistemas comuns, devido parcialmente ao fato de apenas um dispositivo poder afetar todo o sistema de controle. Como tal, para diminuir esse tempo, a reconfiguração de todo o sistema deve ser separada da do dispositivo e a interface dos dispositivos para o sistema de controlo deve ser o mais amigável e simples possível (Arai *et al.*, 2001).

A este conceito de fácil e rápida integração e remoção de recursos de produção num sistema de manufatura que não interrompe a linha de produção é chamado *Plug and Produce* e é adaptado do *Plug and Play* usado em computadores pessoais (Rocha *et al.*, 2014).

A arquitetura *Plug and Produce* divide o sistema em dois subsistemas: hardware de montagem e sistema de gestão. Mas é importante aludir que para a implementação desta arquitetura é essencial ter em consideração alguns pontos relativos a cada sistema(Arai *et al.*, 2001).

Cada dispositivo de montagem do *Plug and Produce* precisa de informar o sistema de gerenciamento da sua capacidade, nome, tipo e protocolos. Por fim, o dispositivo deve ainda informar a sua posição no sistema de coordenadas do mundo, isto é o tamanho do espaço de trabalho onde se insere e outras características físicas e geométricas (Arai *et al.*, 2001).

No subsistema de gestão considerar que o sistema deve funcionar continuamente durante a remoção / adição de um dispositivo de montagem. O sistema deve ser capaz de reconhecer automaticamente o dispositivo recém-adicionado. Novamente, estas características são semelhantes ao *Plug and Play*. Além disso, a atribuição de tarefas, isto é, o agendamento da tarefa, é importante no caso de sistemas de montagem (Arai *et al.*, 2001).

3.3 Device Profile for Web Services

O *Device Profile for Web Services* (DPWS), por vezes conhecido como *Web Services on Devices* (WSD), foi lançado em 2004 por um grupo liderado pela empresa Microsoft com o intuito inicial de servir de base à atualização do *Universal Plug and Play* (UPnP). Contudo, a transição de UPnP para DPWS não ocorreu devido à sua falta de compatibilidade com versões de produtos anteriores. (Sleman and Moeller, 2008; Unger and Timmermann, 2015)

Como tal, em vez de sucessor do UPnP como previsto, o DPWS tornou-se uma tecnologia independente, e foi assim lançado um pedido de estandardização à OASIS em 2008, uma organização não lucrativa que promove consenso industrial e produz estandardizações a nível mundial, tornando se *standard* desde julho de 2009 (Sleman and Moeller, 2008).

O DPWS define um conjunto mínimo de restrições de implementação para permitir mensagens de serviços da Web em dispositivos com recursos limitados e, de modo a ser mais abrangente, este é baseado em padrões existentes na arquitetura de *Web Services*, como é possível ver na figura 3.1, fazendo com que os dispositivos sejam dinamicamente descobertos e descritos e que ainda troquem mensagens, enviem e recebam dados e eventos (Fysarakis *et al.*, 2015; Xu, Fang and Li, 2016).

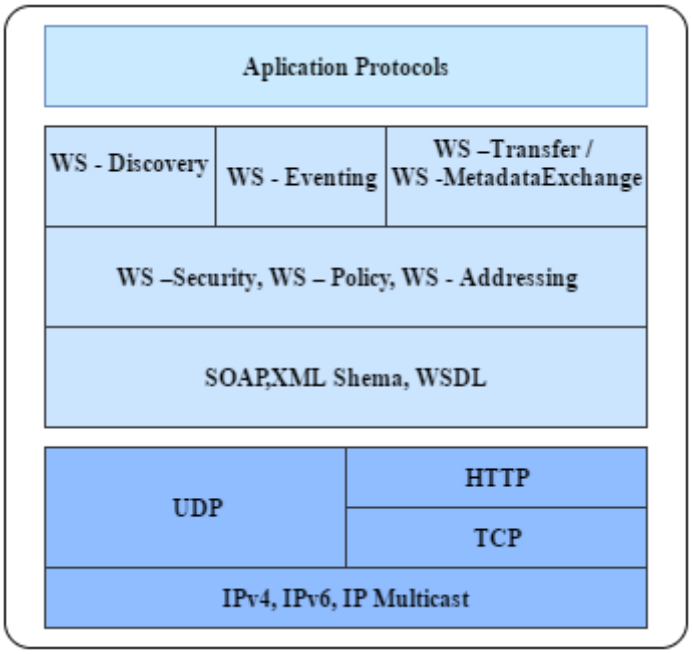

Figura 3.1 - Pilha de protocolos referente ao DPWS

A arquitetura do DPWS inclui serviços de *'hosting and hosted'*, isto é, permite criar uma ponte entre o software existente nos dispositivos e os serviços web (Fysarakis *et al.*, 2015).

Ao ser permitida a migração de informação de baixo nível, por exemplo a informação sensorial, em contexto de alto nível, como a extração de conhecimento ou operações comerciais, é possibilitada a existência de novos tipos de serviços. E é esperado que estes sejam vitais para o futuro dos utilizadores assim como para implantações empresariais, num vasto número de domínios da indústria. O uso e os benefícios do DPWS já foram amplamente estudados por investigadores no contexto de vários campos como sistemas ferroviários, automação industrial, eHealth e cidades e casas inteligentes (Fysarakis *et al.*, 2015).

Atualmente a sua maior desvantagem reside na existência de numerosas especificações (protocolos, ligações etc.) que ainda não foram consolidadas. No entanto, continua a ser uma tecnologia muito promissora, tanto que é ativamente impulsionado pelas partes interessadas da indústria para implantações em larga escala onde é bastante comum nesta era da *Internet of Things* (IoT) que tem uma grande influencia no mercado (Fysarakis *et al.*, 2015).

3.4 OPC Arquitetura Unificada

Num mercado onde as arquiteturas orientadas a serviços, do inglês *Service-Oriented Architecture* (SOA), estão cada vez mais presentes surgem novos desafios associados à segurança e à modelação de dados. Assim em 2008, a fundação OPC desenvolveu a OPC Arquitetura Unificada, do inglês OPC *Unified Architecture* (UA), uma plataforma independente de arquitetura orientada a serviços que integra as vantagens e habilidades dos seus predecessores como OPC *Data Access*, OPC *Alarms & Events* ou OPC *Historical Data Access* num único *standard.* (Renjie, Feng and Dongbo, 2010; Wassilew *et al.*, 2016).

Além de ter sido desenhada para substituir os seus antecessores oferecendo um fornecedor independente de arquitetura aberta, esta foi ainda concebida para oferecer uma tecnologia com futuro viável e uma arquitetura robusta, estável e extensível (Garcia *et al.*, 2016; *Unified Architecture - OPC Foundation*, 2017). Como tal, é considerada uma tecnologia "chave" para a arquitetura de referencia *Industry 4.0* devido à sua capacidade de descrição semântica e integração online por modelagem de informações (Wassilew *et al.*, 2016).

Tal como os seus precedentes, a OPC-UA segue o paradigma de cliente / servidor. Referente à portabilidade, em que é o servidor que usa uma pilha de comunicação portátil que pode ser diretamente utilizada em sistemas de automação. Por sua vez a OPC UA fornece meios de comunicação confiáveis, robustos e de alto desempenho, adequados para aplicações e dispositivos de automação industrial (Garcia *et al.*, 2016; Grüner, Pfrommer and Palm, 2016).

É ainda importante realçar que a OPC UA é uma das mais promissoras tecnologias de comunicação para arquiteturas *Plug and Produce* tal como especificado no seu standard IEC 62541 (Wassilew *et al.*, 2016)

.

3.5 Projeto IDEAS

O projeto IDEAS (Instantly deployable evolvable assembly systems) consistia em desenvolver um ambiente totalmente distribuído e conectável capaz de se auto-organizar e controlar no nível de "*shop floor*" inspirado nos EPS e com recurso da tecnologia de agentes.

Os sistemas de montagem baseados no projeto IDEAS funcionam com uma configuração de controlo multiagente e podem ser reconfigurados sem interromper o sistema assegurando a

integração de diferentes módulos autoconfigurados no "*shop floor*" em tempo de execução (*Plug & Produce*). Além disto, cada módulo permite ter um diagnóstico distribuído onde todo o sistema é capaz de verificar a propagação de problemas e adaptar-se sempre que um componente (módulo) é conectado sem necessidade de um esforço maior na programação referente à gestão de comportamentos imprevistos (Ribeiro and Barata, 2011).

A arquitetura multiagente *IDEAS Agent Development Environment* (IADE) tem um núcleo composto por 9 entidades principais de funções especificas agrupadas em três grupos consoante o seu propósito: Suporte, Execução, Transporte (Fig. 3.2).

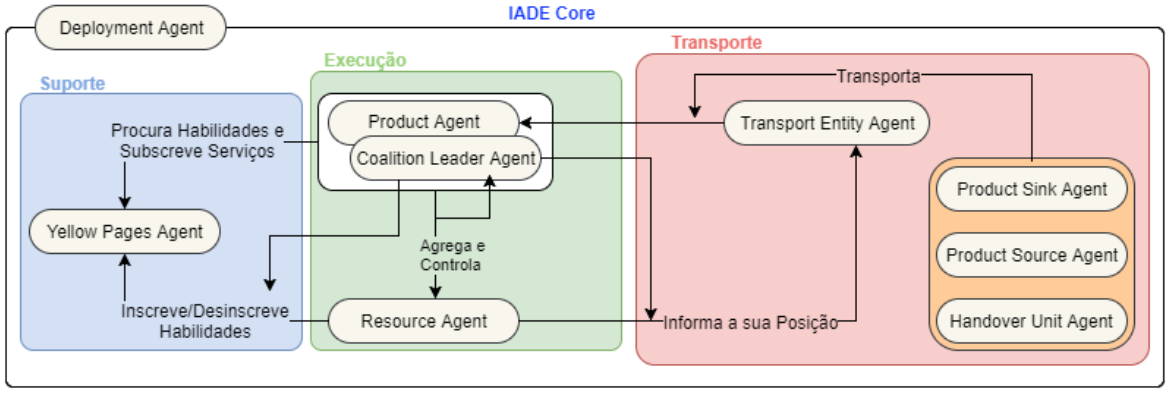

Figura 3.2 - Núcleo da arquitetura IADE

O sistema é administrado principalmente por Product Agents (PA) que representam instâncias de produtos em processamento onde o objetivo de cada PA é cumprir o seu plano de processo. Estes dependem assim da ação conjunta de outros agentes como os Coalition Leader Agents (CLA) que permitem a execução de planos mais complexos, e de Resource Agents (RA), onde oferecem ações simples. Dito isto, a tabela seguinte (Tabela 3.1) apresenta uma breve descrição de cada agente envolvido no núcleo da arquitetura IADE (Ribeiro and Barata, 2013; Ribeiro *et al.*, 2013).

Tabela 3.1 - Agentes presentes no núcleo da arquitetura IADE (Ribeiro *et al.*, 2013)

Grupo	Classe	Função
Suporte	DeploymentAgent (DA)	Serve de intermediário entre a plataforma agente e os controladores da linha. Possibilita ao utilizador lançar qualquer agente processos de execução e transporte para um controlador especifico.
	YellowPagesAgent (YPA)	Acompanha os serviços/habilidades que cada um dos agentes disponibiliza no sistema. Os agentes podem consultar o YPA para localizar outros agentes e as suas habilidades bem como assinar um serviço de notificação que os informa se um agente específico deixou a plataforma.
Execução	ResourceAgent (RA)	São a entidade de menor abstração possível na pilha IADE. Interagem diretamente com os controladores da linha e têm como função principal traduzir habilidades para código nativo e garantir a sua execução e sincronização com a plataforma de agentes. Os RA's informam ainda o sistema de transporte sobre a localização física onde as suas habilidades podem ser executadas.
	CoalitionLeaderAgent (CLA)	Os CLA's são utilizados para oferecer habilidades mais complexas recorrendo a uma agregação de habilidades de agentes já existentes. Embora sejam capazes da execução e coordenação de CLA e RA, estes não interferem com qualquer controlador da linha. À semelhança do RA, o CLA informa o sistema de transporte sobre a localização física onde as habilidades estão disponíveis.
	ProductAgent (PA)	O PA representa o nível mais alto da abstração do sistema e é uma representação virtual de um objeto a ser produzido. Cada PA tem a capacidade de controlar o seu próprio plano de execução e de tomar decisões sobre o local onde cada etapa do processo é realizada. A suas decisões têm ainda por base a ação coletiva existente entre si e os agentes do sistema de transporte.

Transporte	ProductSourceAgent (PSoA)	O PSoA administra a entrada de paletes no sistema e associa-lhe um PA específico após estas estarem no sistema. Isto é fundamental tendo em conta a possibilidade de haver mais PA's do que paletes e neste contexto, o PSoA gere a entrada de cada PA em espera.
	ProductSinkAgent (PSiA)	O PSiA tem a função oposta ao PSoA, ou seja, este liberta as paletes do PA, sempre que um certo PA termina o seu plano de execução.
	TransportEntityAgent (TEA)	Abstrai um transportador específico do sistema sendo responsável pelo deslocamento de paletes entre encaminhadores. Cada TEA disponibiliza vários pontos de acoplamento, onde as estações, constituídas por RA's e CLA's, podem ser acopladas e desacopladas durante a execução e gere o tráfego das paletes no transportador, informando o PA sobre o custo do encaminhamento e a posição atual aquando da chegada ao destino.
	HandoverUnitAgent (HUA)	Cada HUA controla um encaminhador do sistema e calculam a árvore de custos mínimos para chegar a cada um dos destinos possíveis na linha. Esta informação é posteriormente partilhada com o TEA associado que, quando necessário, transmitem para os PA's.

4 Arquitetura

Tal como referido no capitulo 2, existem no mercado enumeras soluções que permitem criar um sistema descentralizado tornando o processo de manufatura mais flexível e dinâmico. Contudo, estas focam-se principalmente a nível de *software* tornando todo o sistema inacessível ou dificilmente adaptável para plataformas físicas de baixo custo.

A arquitetura proposta toma em conta tanto *software* como *hardware* tendo em vista um sistema dinâmico e acessível. Foca-se num sistema adaptável com o objetivo de fornecer uma reconfiguração intuitiva nas linhas de produção em massa e a robustez desta abordagem permite o uso de diferentes tecnologias e permite aplicações em vários propósitos diferentes, sem muita adaptação de *hardware*.

A arquitetura segue uma abordagem multiagente de modo a maximizar a capacidade do sistema. Estes tipos de abordagem oferecem autonomia às entidades para que estas reajam a mudanças durante a sua execução permitindo assim características como o *Plug & Produce* e alta tolerância diante de distúrbios e mudanças inesperadas.

A nível de hardware há que ter em conta o custo total associado ao mesmo e ainda a sua capacidade de processamento bem como a capacidade de se conectar em rede.

4.1 Arquitetura da Solução

A arquitetura proposta segue uma abordagem focada na entidade de mais alto nível do sistema, o produto a ser produzido. Este é que vai ditar a disposição do sistema conforme as suas necessidades (plano de habilidades necessárias para a sua produção). Toda a gestão do sistema é realizada pelo MAS e os elementos do Sistema de produção encontram-se todos virtualizados

nele, desde o produto aos recursos. Parte deste sistema irá correr nos dispositivos IoT uma vez que estes serão os únicos com capacidade de interagir diretamente com o *hardware.*

Para reduzir o processamento nos dispositivos IoT e permitir tirar maior vantagem do uso deste tipo de dispositivos, é frequente que a parte de maior toma de decisão de aplicativos baseados em IoT estejam alocados na nuvem (Wang, Bi and Xu, 2014).

A computação realizada na nuvem (*Cloud Computing*) permite um acesso conveniente, escalável e passivo a um conjunto de recursos computacionais configuráveis e tem um poder de processamento e capacidade de armazenamento praticamente ilimitados ao contrario da IoT. Assim permite que a IoT abstraia as suas limitações, heterogeneidade, conectividade, identificação e segurança dos dispositivos envolvidos (Díaz, Martín and Rubio, 2016).

A integração da IoT na nuvem, conhecida como *Cloud of Things*, permite resolver as limitações do IoT, acesso a dados, computação, análise de dados e pode criar novas oportunidades, como *Smart Things, Things as a Service e SenaaS* (*Sensor as a Service*). Além disso, oferecendo uma plataforma como serviço onde os utilizadores podem criar aplicativos para interagirem com o sistema (Díaz, Martín and Rubio, 2016).

Devido às vantagens a cima descritas, existem diversos projetos com base nestas tecnologias e ainda uma grande investigação cientifica das mesmas. Estas tecnologias oferecem assim uma plataforma que oferece alta escalabilidade, armazenamento e processamento uniforme em tempo real (Díaz, Martín and Rubio, 2016).

De forma simplificada, o utilizador introduz os produtos desejados no sistema e o sistema reproduz as necessárias ações sem afetar mais recursos do que os necessários através do MAS e dos dispositivos IoT. Estes últimos são a única parte do sistema capaz de interagir com os recursos físicos (Figura 4.1). Os recursos físicos podem ser impressoras 3D, braços mecânicos, mecanismos de perfuração e qualquer outro dispositivo ou ferramenta.

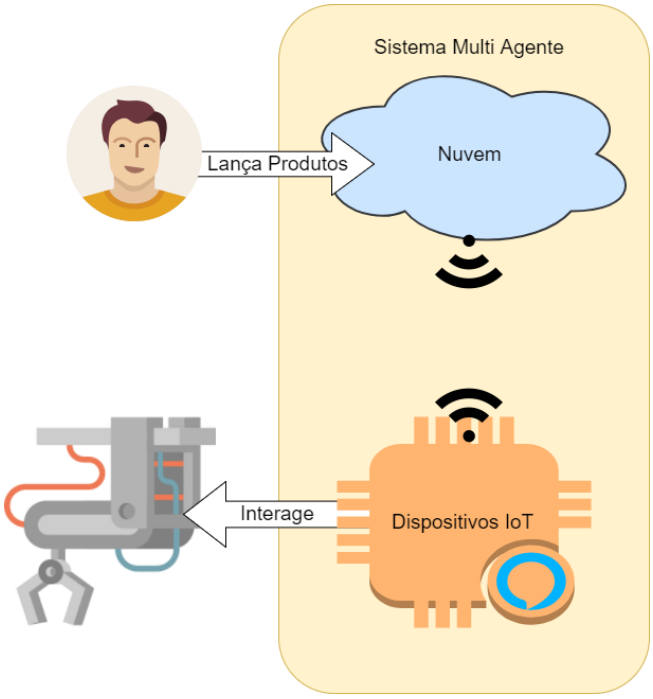

Figura 4.1 - Esquema simples da arquitetura proposta

Por vezes existe mais do que um recurso a oferecer a mesma habilidade. Este ponto pode ser uma vantagem em caso de produção em série em que um recurso esteja ocupado ou em caso de avaria onde o produto tem que ser redirecionado. Isto só é possível devido à modularidade da arquitetura proposta onde cada componente do sistema é um módulo independente permitindo que a gestão realizada pelo MAS redirecione o produto para outro módulo de produção capaz de realizar a habilidade pretendida.

Ao ser composta por módulos independentes, a presente arquitetura oferece ainda a vantagem de uma fácil e rápida integração e remoção de recursos de produção sem interromper o sistema de produção (*Plug & Produce*). Contudo, é necessário que cada módulo informe o sistema da sua habilidade e da sua posição no mesmo bem como o sistema tem de ser capaz de detetar automaticamente a integração ou remoção do módulo.

É importante realçar que para uma arquitetura modular ser possível é necessário que as interfaces dos componentes sejam genéricas. Na solução proposta, a importância da interface recai na ligação do MAS através dos dispositivos IoT com os recursos físicos.

O produto a ser produzido nem sempre será o mesmo, ou seja, o conjunto de habilidades necessárias para a produção do produto pode variar. Para tal é recomendado a existência de diversos tipos de recursos de modo a oferecer ao sistema global um leque de habilidades maiores.

A Figura 4.2 resume o mencionado anteriormente. Um utilizador qualificado pode lançar produtos variados para o sistema, cada produto tendo uma lista interna de operações necessárias (por exemplo: uma cadeira pode necessitar de ser aparafusada), o sistema organiza os produtos de acordo com as habilidades necessárias e as disponíveis fisicamente. Os dispositivos IoT usam as suas interfaces para se ligarem a um ou mais recursos físicos que terão habilidades diferentes até que consigam garantir a construção esperada. Vários dispositivos IoT podem existir ao mesmo tempo, garantindo que os mesmos dispositivos podem estar a fabricar vários tipos de produtos e usando o MAS para garantir que os recursos físicos são distribuídos entre as varias tarefas a serem feitas.

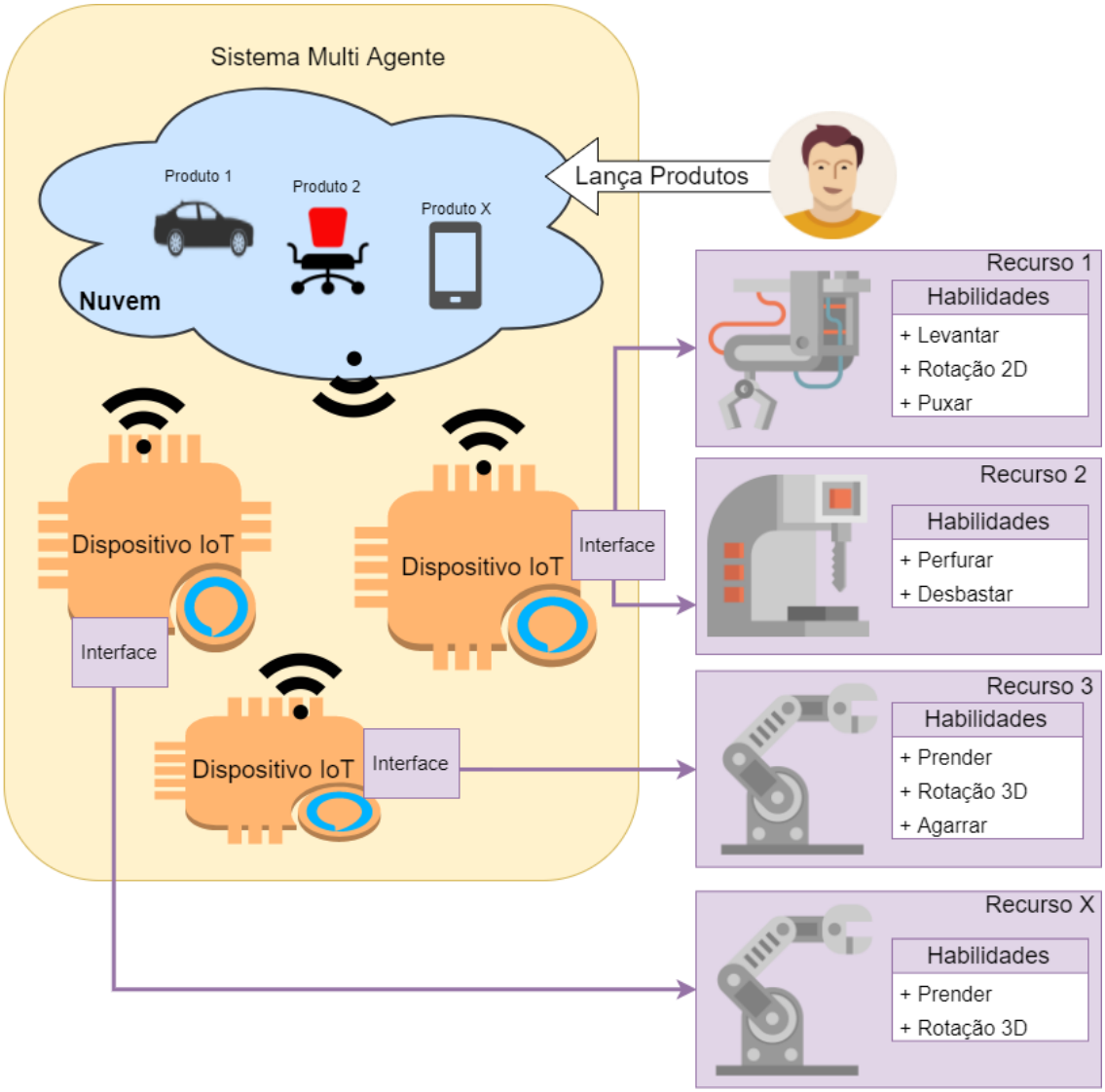

Figura 4.2 - Esquema geral da arquitetura proposta

Pode-se assim concluir que a arquitetura proposta apresenta solução onde a modularização do sistema permite recorrer ao *Plug & Produce* e a um controlo distribuído, a componente IoT permite a escalabilidade do sistema bem como uma conectividade simples e eficaz entre os diferentes módulos de interfaces standard. Por sua vez, o MAS completa a evolutividade e reconfigurabilidade através da virtualização e gestão dos diversos módulos que o constituem o sistema. Como resultado final temos uma arquitetura robusta, flexível e dinâmica.

A arquitetura proposta tem ainda por base o projeto IDEAS e a arquitetura IADE (Ribeiro et al., 2013) . No sistema em estudo, a implementação do sistema de transporte não foi tida em conta uma vez que este sistema pode ser abordado, de forma ampla, como uma ação simples e já existem algumas implementações para tal. Um exemplo de uma implementação mais rigorosa para o sistema de transporte pode ser visto em (Rocha, 2013).

Devido a fatores anteriormente referidos, a solução apresentada terá de ter foco na componente IoT dado que, como demonstrado, esta é a base de toda a conexão com o sistema físico e ainda parte do processamento do MAS.

4.1.1 Definição de Habilidades

Anteriormente foi referido que o PA tem como objetivo cumprir um plano de processos associado a um produto. Este plano é composto por um conjunto de habilidades necessárias para a fabricação de um dado produto.

Os recursos, por sua vez, oferecem um conjunto de habilidades diferentes que representam as suas capacidades e que podem ser usadas, de forma individual ou sob a forma de um conjunto, por um produto.

Neste sentido, um sistema pode conter uma quantidade elevada de habilidades sendo assim importante a existência de um conjunto de dados mínimo que especifiquem cada habilidade, nomeadamente um ID, um nome, uma breve descrição e uma lista de parâmetros para controlar a sua execução.

Um produto pode ainda precisar de habilidades constituídas por um conjunto de habilidades mais simples. Como tal é importante a existência de pelo menos dois tipos de habilidades:

- Habilidade Atómica (HA) / Atomic Skill: Representam capacidades atómicas provenientes de uma entidade que podem ou não corresponder a um processo. Estas

associam a informação disponibilizada por uma habilidade à execução de um dado mecanismo num controlador especifico. No contexto deste sistema, este tipo de habilidades é o único que representa diretamente um processo do mundo real;

- Habilidade Complexa (HC) / Complex Skill: Representam um conjunto de habilidades, podendo estas ser tanto HA como HC. Estas devem ser criadas quando existe a necessidade de um processo onde as habilidades existentes não o conseguem satisfazer. Estas são criadas combinando HA e HC disponíveis no sistema seguindo um conjunto pré-definido de regras. Ao contrario das HA, estas nunca interagem diretamente com o mundo real.

4.2 Componente IoT

No capitulo 2 foi referido o uso de IoT no contexto industrial onde empresas lideres mundiais estão a investir consideravelmente nesta tecnologia. Segundo (Wang, Bi and Xu, 2014), a adoção de sistemas IoT e de computação em Cloud pode superar as ferramentas existentes de software assistido por computador ao lidar com a complexidade, dinâmica e incertezas nas aplicações empresariais.

Num sistema de produção, a quantidade de recursos existentes por norma é elevada e, a informação derivada destes deve ser tida em conta para as decisões empresarias. Isto cria novos desafios numa empresa onde descentralização, modularidade e capacidade de expansão se tornam requisitos. Para além disto, os novos produtos tendem a ser mais inteligentes, versáteis e sofisticados, ou por outras palavras, mais complexos criando novos desafios nos processos de fabricação. O desempenho do sistema (lucro, tempo de execução, qualidade e custo) irá assim depender consideravelmente da eficácia do modelo de montagem (Wang, Bi and Xu, 2014).

Tal como mencionado no capitulo 2, para atender a estes tipos de produtos é importante a existência dum sistema flexível e dinâmico capaz de comunicar e operar em conjunto. A adoção do IoT num sistema terá um impacto significativo na disponibilidade de dados para o planeamento da fabricação do produto bem como no nível de dificuldade ao modelar um sistema de montagem. Por um lado, pode-se estabelecer uma nuvem privada ou uma nuvem híbrida para que os dados possam ser acedidos por qualquer utilizador necessário, independentemente de como e onde os recursos estão distribuídos geograficamente.

Quanto à dinâmica dos dados, o IoT faz a ligação entre todos os objetos e possibilita a monitorização e a recolha de dados em tempo real dos mesmos permitindo que, por exemplo, incertezas passam ser identificadas e usadas para suportar decisões de otimização. Por outro, o IoT usa arquitetura orientada a serviços. As ferramentas distribuídas acessíveis no IoT são modularizadas e interoperáveis. Eles podem ser associados para cumprir algumas decisões mais complexas, conforme necessário (Wang, Bi and Xu, 2014).

Podemos assim concluir que um dispositivo IoT tem como principal foco a recolha de dados ou a atuação no mundo físico e a comunicação com um sistema virtual como, por exemplo, um sistema disponível numa nuvem. Segundo (Wang, Bi and Xu, 2014), um aplicativo IoT bem-sucedido deve conseguir tomar decisões em objetos complexos. A Figura 4.3 indica assim as principais capacidades que um dispositivo IoT deve oferecer para a arquitetura proposta. É de notar que o dispositivo não deve oferecer sensores e atuadores mas sim ter a capacidade de interagir com os mesmos.

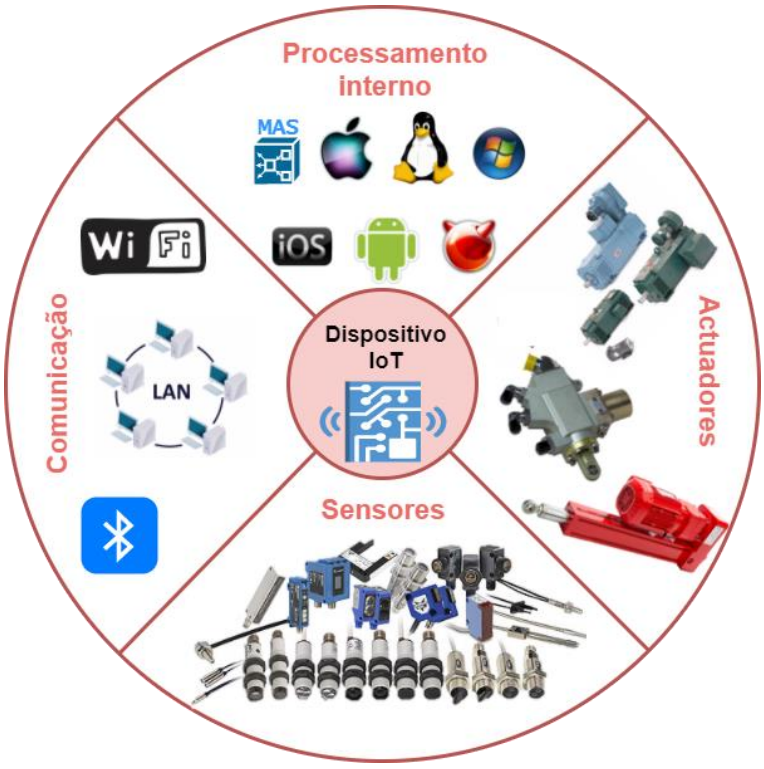

Figura 4.3 - Capacidades do Dispositivo IoT para a solução proposta

Na arquitetura proposta, os dispositivos IoT têm assim diversas “responsabilidades”. Entre as quais, os dispositivos devem ser capazes de comunicar com o sistema global, devem conseguir interagir com o Hardware (sensores e atuadores) e por fim, devem ter a capacidade de

correr agentes, uma vez que têm autoridade para realizar algumas ações predispostas face alguns fatores, ou seja, os agentes darão o componente inteligente aos dispositivos.

De modo a correr agentes, os dispositivos necessitam de capacidade de processamento e armazenamento diretamente dependentes do software adotado para a implementação dos agentes. A escolha dos dispositivos bem como do software deve ter ainda em consideração o custo associado à combinação de ambos.

Pode-se assim definir que, nesta arquitetura, a componente IoT corresponde a um CPS onde os agentes que correm nos dispositivos IoT são responsáveis por parte ou total processamento de dados do sistema global, considerando-se assim como uma componente virtual, e são capazes de interagir diretamente com o Hardware, formando a parte física do sistema. No caso de estudo (Figura 4.4), o dispositivo IoT irá conter 3 tipos de agentes, DA, RA e CLA. Uma vez que estes comunicam diretamente com o hardware, irão ter um RA por cada recurso físico que possuam. Irão ter CLA no caso dos RA fornecerem habilidades suficientes para a criação de alguma habilidade composta pré-definida. Por fim, os DA irão lançar os RA caso tenham recursos ligados ao dispositivo IoT e CLAs no caso mencionado a cima.

Figura 4.4 - Componente Ciber-Físico

Os restantes agentes do MAS irão correr em nuvem de modo a facilitar o processamento nestes dispositivos como mencionado previamente.

4.2.1 Sistema de Produção

No capitulo 2 chegou-se à conclusão que os sistemas de produção tradicionais são incapazes de, por si só, acompanhar a procura de produtos no mercado atual. Como tal, estes sistemas têm de ser adaptados ou substituídos por abordagens de arquitetura descentralizada e reconfigurável.

A arquitetura presente propõe a divisão do sistema de produção em subsistemas com capacidade de interação entre si (módulos) tirando partido das diversas vantagens associadas ao conceito de modularidade referido em 3.

Pode-se assim considerar um recurso ou ferramenta do sistema de produção como um módulo. No entanto, os componentes ciber-físicos também são considerados módulos, ou seja, um componente ciber-físico é um módulo composto por diversos módulos, nomeadamente o dispositivo IoT e os recursos ou ferramentas ligadas a este (Figura 4.5).

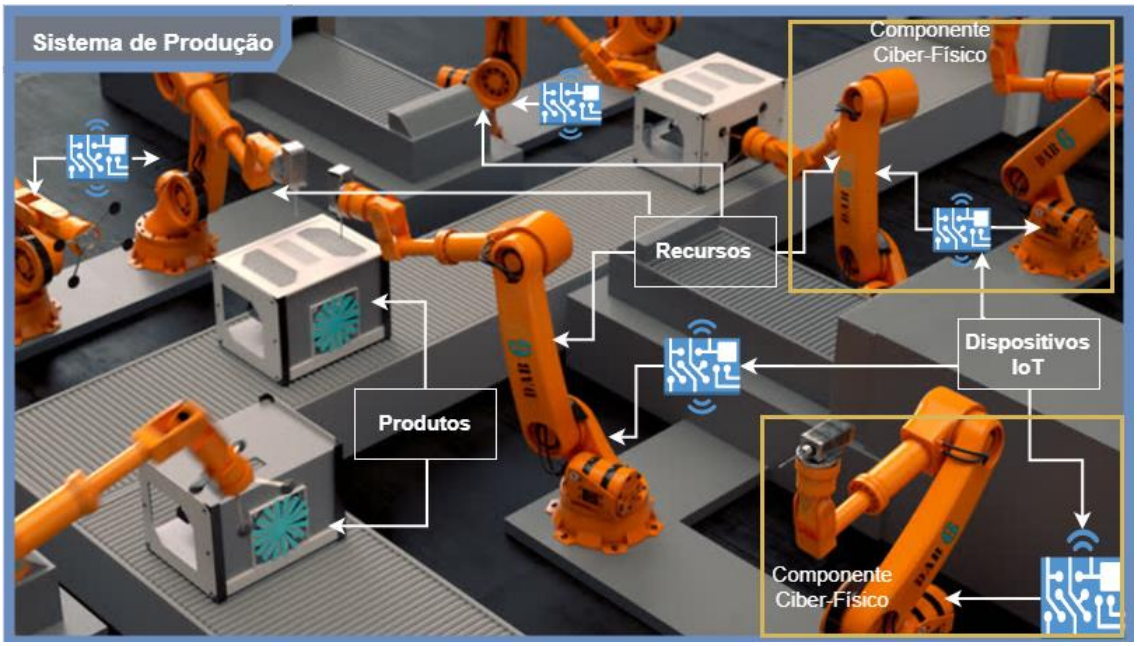

Figura 4.5 - Sistema de Produção

Os dispositivos IoT interagem com o hardware através de interfaces. Estas são criadas de acordo com o recurso ou ferramenta especifica. Após a implementação da devida interface, o DA reconhece quando o recurso está ou não ligado e lança ou desliga o RA respetivo no sistema.

Semelhante ao anterior, sempre que um dispositivo IoT se liga, automaticamente conecta-se ao restante sistema.

Para melhor compreensão do acoplamento de módulos, vamos considerar o seguinte caso de exemplo:

- Inicia-se o sistema numa nuvem (Figura 4.6).

Figura 4.6 - Caso de exemplo, estado 1

- Liga-se um Dispositivo IoT. Este automaticamente irá conectar-se ao sistema principal (Figura 4.7).

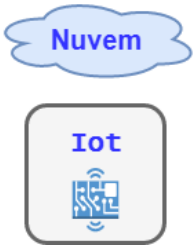

Figura 4.7 - Caso de exemplo, estado 2

- Após se conectar ao sistema principal reconheceu que outros dois módulos se ligaram a ele. Neste caso especifico, 2 recursos (R). Estes dois recursos e as suas habilidades passam assim a serem reconhecidos pelo sistema através do dispositivo IoT (Figura 4.8)

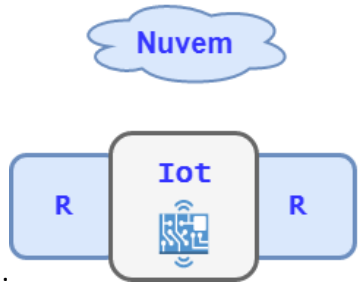

Figura 4.8 - Caso de exemplo, estado 3

- Dois novos dispositivos IoT foram ligados. Novamente, os mesmos conectam-se ao sistema (Figura 4.9).

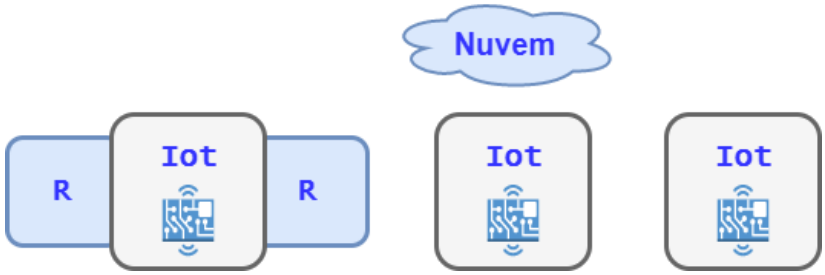

Figura 4.9 - Caso de exemplo, estado 4

• Um dos novos dispositivos verificou que não tinha nenhum módulo ligado a si enquanto que o outro reconheceu 3 recursos e integrou-os no sistema disponibilizando as suas habilidades no mesmo (Figura 4.10).

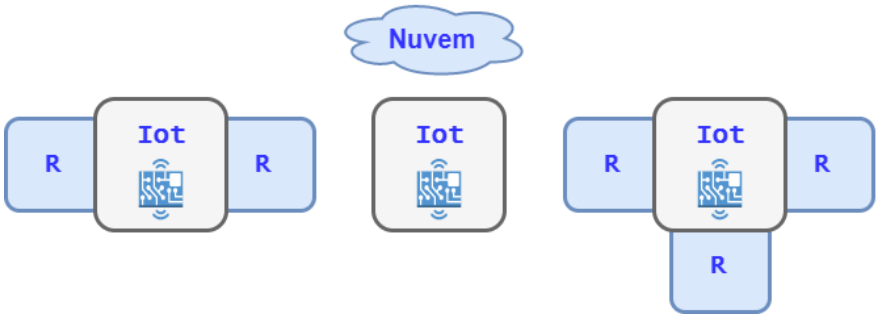

Figura 4.10 - Caso de exemplo, estado 5

• Por sua vez, um dos dispositivos notou que um dos módulos que tinha ligado a si desligou-se e, como tal, desconectou-o do sistema (Figura 4.11). Por consequencia, as habilidades disponibilizadas por esse módulo deixaram de estar disponiveis no sistema. No entanto, estas habilidades podem ser oferecidas por outro ou outros módulos que tejam conectados e que, no seu conjunto, tenham as mesmas habilidades.

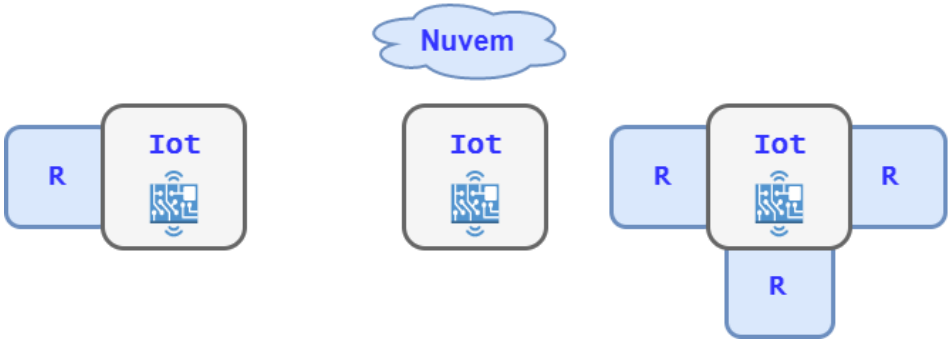

Figura 4.11 - Caso de exemplo, estado 6

• Por fim, um dos dispositivos IoT desliga-se. O dispositivose todos os módulos ligados a ele, deixam de fazer parte do sistema (Figura 4.12)

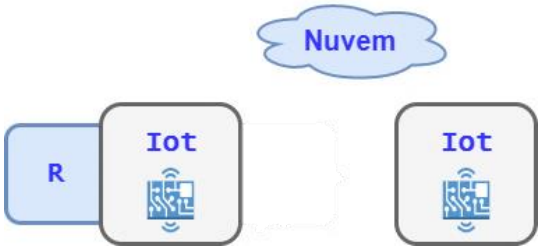

Figura 4.12 - Caso de exemplo, estado 7

Em suma, qualquer módulo composto por um dispositivo IoT existente no sistema tem a capacidade de se connectar automaticamente ao sistema e o sistema reorganiza-se de acordo com as necessidades do produto a produzir e as funcionalidades que cada módulo oferece. O mesmo se sucessede sempre que algum modolo se disconecta do sistema, este irá reorganizar-se de acordo com os módulos que têm disponivel. Os recursos são apenas reconhecidos no sistema através dos dispositivos IoT que por sua vez usam interfaces para com os recursos.

4.3 Arquitetura Multiagente

Em 4.1 foi referido que cada componente do sistema físico teria uma representação virtual no MAS, ou seja, o produto a produzir dá origem a um PA, um recurso/ferramenta do sistema dá origem a um RA. Para alem destes temos ainda outros agentes com funcionalidades distintas à semelhança de 3.5.

Este sistema gere assim todo o sistema de produção, desde qual produto deve ir para um determinado recurso a qual recurso deve estar a ser executado num determinado produto num momento especifico. Toda esta gestão é focada no PA. Este tem um plano de produção a cumprir e, como base neste plano o MAS decide para onde deve encaminhar o produto.

Todo este processamento pode ser realizado num local único como ser distribuído, ou seja, um dispositivo IoT pode ter o PA, e 2 RA e os restantes agentes necessários ao sistema ou então o PA e um RA estar num dispositivo e o outro RA noutro. De modo a melhorar o processamento, na solução proposta, os dispositivos IoT alojam agentes do tipo RA, DA e CLA enquanto que os restantes se encontram numa nuvem de agentes.

A figura 4.13 mostra um exemplo possível de implementar esta arquitetura onde os PA são lançados na nuvem enquanto que os RA estão alocados aos dispositivos IoT.

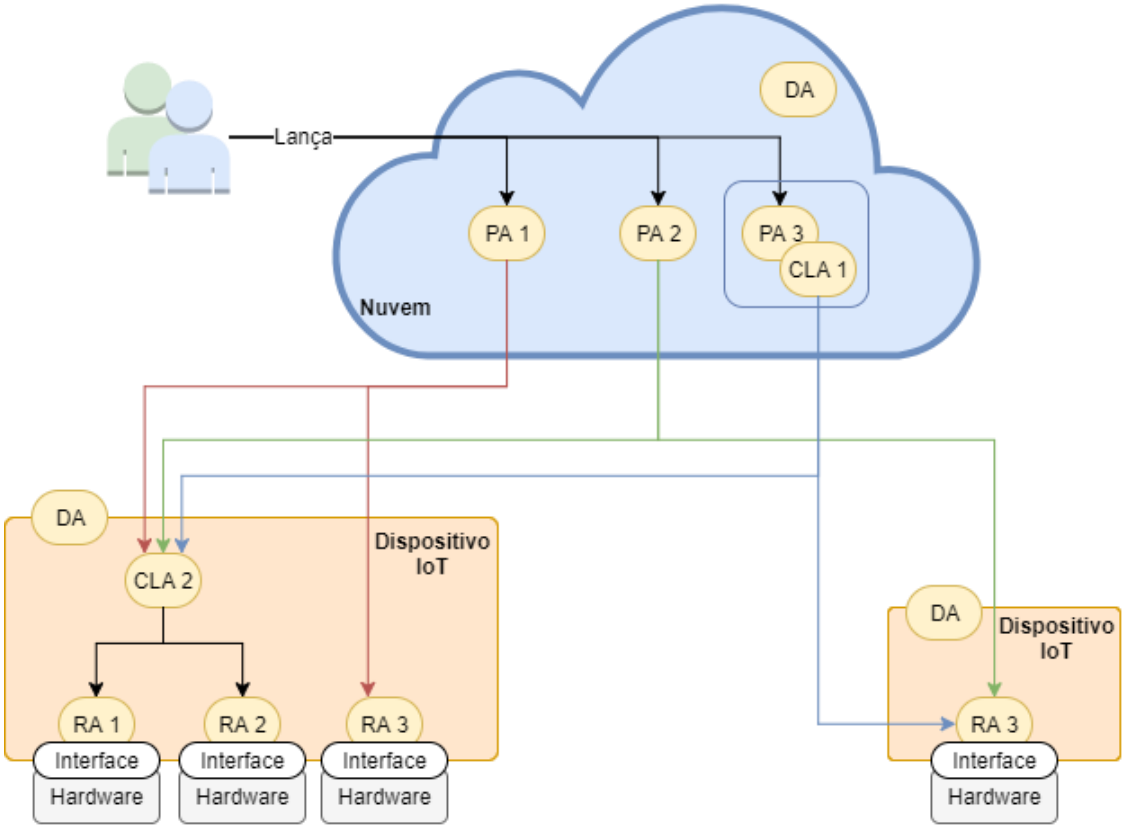

Figura 4.13 - Exemplo de implementação da arquitetura proposta.

No caso desta arquitetura, os dispositivos IoT não necessitam de um processamento mais elevado para correr um numero maior de agentes uma vez que parte destes agentes estão alocados na nuvem.

Num outro exemplo para a implementação da arquitetura proposta, poderíamos ter o dispositivo IoT como simples prestadores de serviços, ou seja, os agentes estariam todos na nuvem e o dispositivo serviria de interface com o sistema físico através da execução de serviços.

Os diversos agentes podem assim ser representados por breves fluxogramas onde se pode observar o comportamento individual de cada um deles bem como a ligação existente entre os agentes. As subsecções seguintes irão apresentar uma vista de alto nível dos principais agentes do sistema.

4.3.1 Product Agent

Uma vez que o sistema funciona todo em prol da produção de um dado produto, este é o que irá ditar o processo de produção. O produto tem uma lista de habilidades (plano de execução) que necessita para a sua produção e irá verificar se o sistema permite a realização do seu plano de execução e, caso permita, irá iniciar a produção. A figura 4.14 apresenta um fluxograma ilustrativo deste processo.

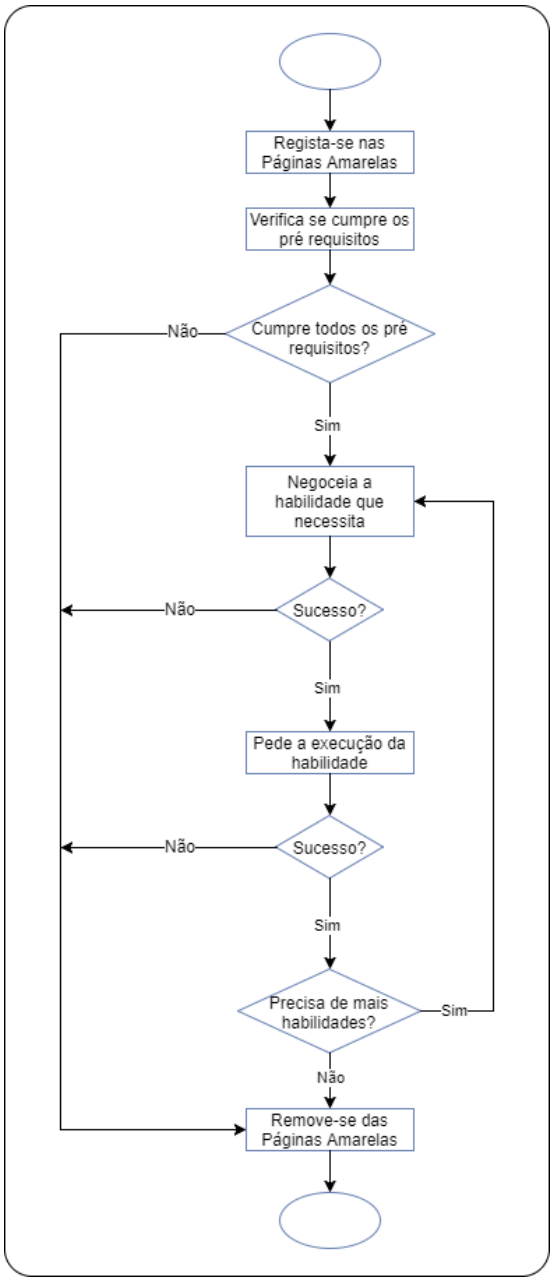

Figura 4.14 - Fluxograma do agente Produto

Ao analisar o fluxograma notamos que o agente que representa o produto, ao ser criado, regista-se no serviço de paginas amarelas e verifica se, no sistema, existem recursos capazes de oferecer todas as habilidades necessárias para cumprir a execução do sem plano.

Caso não existam este remove-se das paginas amarelas e é terminado com a mensagem de impossibilidade de produção devido à falta de habilidades para a realização do plano. Caso contrario, este inicia o processo de produção.

O processo de produção consiste em ir ao seu plano e, habilidade a habilidade necessária, negociar a mesma, ou seja, ver qual recurso é o mais indicado para a produção da habilidade em questão e, em seguida, executá-la.

A negociação da habilidade é um processo que passa por procurar nas paginas amarelas todos os RA e CLA que disponibilizem habilidade dada e verificar qual é o mais favorável para a execução da habilidade tendo em conta fatores como o estado do recurso, ou seja, quantos produtos já estão em lista de espera para aquele recurso e o valor associado à produção do mesmo.

Após a negociação podem existir dois casos possíveis: a negociação falhar por algum motivo ou ter sucesso. Caso falhe, o agente remove-se das paginas amarelas, caso não ele segue o seu processo para o estado de execução.

Na execução, o produto pede ao recurso que ganhou a negociação para executar a habilidade em questão. À semelhança da negociação, caso este processo falhe o agente remove-se das paginas amarelas e caso seja sucedido o agente procede para a verificação do plano, ou seja, verifica se existem mais habilidades no plano de produção quem ainda não foram negociadas e executadas e, caso existam, volta ao estado de negociação e repete o processo para as habilidades que faltam.

Quando todas as habilidades tiverem sido realizadas com sucesso, o agente da por terminado o processo de produção e remove-se das paginas amarelas.

É importante realçar que, caso não tivesse sido ignorado o sistema de transporte, este tomaria um estado entre a negociação e a execução onde o produto, após a negociação terminar, iria pedir que fosse transportado para a posição do recurso que ganhou a negociação para poder iniciar a execução da habilidade.

Outro aspeto importante a destacar aponta para o facto de, nesta implementação, sempre que existe um erro o produto é dado como defeituoso e removido. Numa implementação mais elaborada, caso exista erros na execução ou na negociação, o produto poderia voltar ao estado de

negociação e verificar se já conseguiria resolver o problema. Este processo poderia ser realizado, por exemplo, três vezes antes de se considerar o produto defeituoso.

4.3.2 Resource Agent

O RA é o agente de mais baixo nível deste sistema. Este simula um recurso/ferramenta do sistema físico e, como tal, é responsável pela interação com o Hardware. Na figura 4.15 está representado o diagrama de atividades associada a este agente.

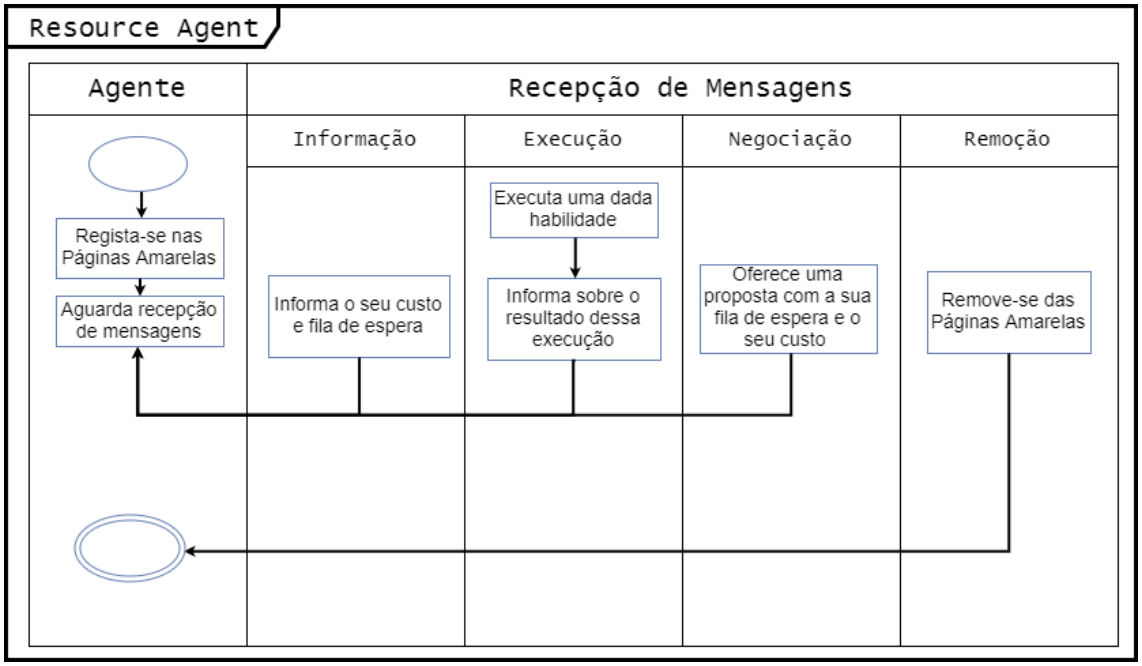

Figura 4.15 – Diagrama de atividades do agente Recurso

Na figura 4.15 pode-se observar que este agente após nascer e registar-se no serviço de paginas amarelas, onde regista as habilidades capazes de realizar, ele apenas responde a mensagens (pedidos ou negociações) de outros agentes, nomeadamente do PA, CLA e DA.

Ao receber uma comunicação de negociação este dá uma proposta com o valor associado a produção de uma habilidade e o tamanho da sua lista de espera. Caso a sua proposta seja aceite, o agente que inicio esta negociação irá ser colocado na lista de espera de execução.

Ao receber uma comunicação de execução, ele verifica se o iniciador da comunicação é o próximo na sua lista de execução e, caso seja, inicia a execução da habilidade. Neste caso, ele comunica com o hardware para a execução da habilidade e aguarda a resposta com o resultado. Esta resposta é depois passada ao que iniciou a comunicação, seja uma mensagem de sucesso ou não. Caso não seja o próximo na lista, o RA faz com que o iniciador fique à espera da sua vez.

A comunicação de informação é semelhante à da negociação, contudo, esta só informa o valor associado ao recurso e o tamanho da sua lista de espera.

Ao receber uma cominação de remoção, este remove-se das paginas amarelas e conclui o seu ciclo.

4.3.3 Coalition Leader Agent

O CLA tem como funcionalidade oferecer habilidades complexas para o produto. Para isso ele coordena RA e CLA. A sua estrutura assemelha-se ao do RA contudo, este ao receber pedidos do PA irá comunicar com os CLA e RA que coordena. As figura 4.16 e 4.17 demonstra o funcionamento de um CLA.

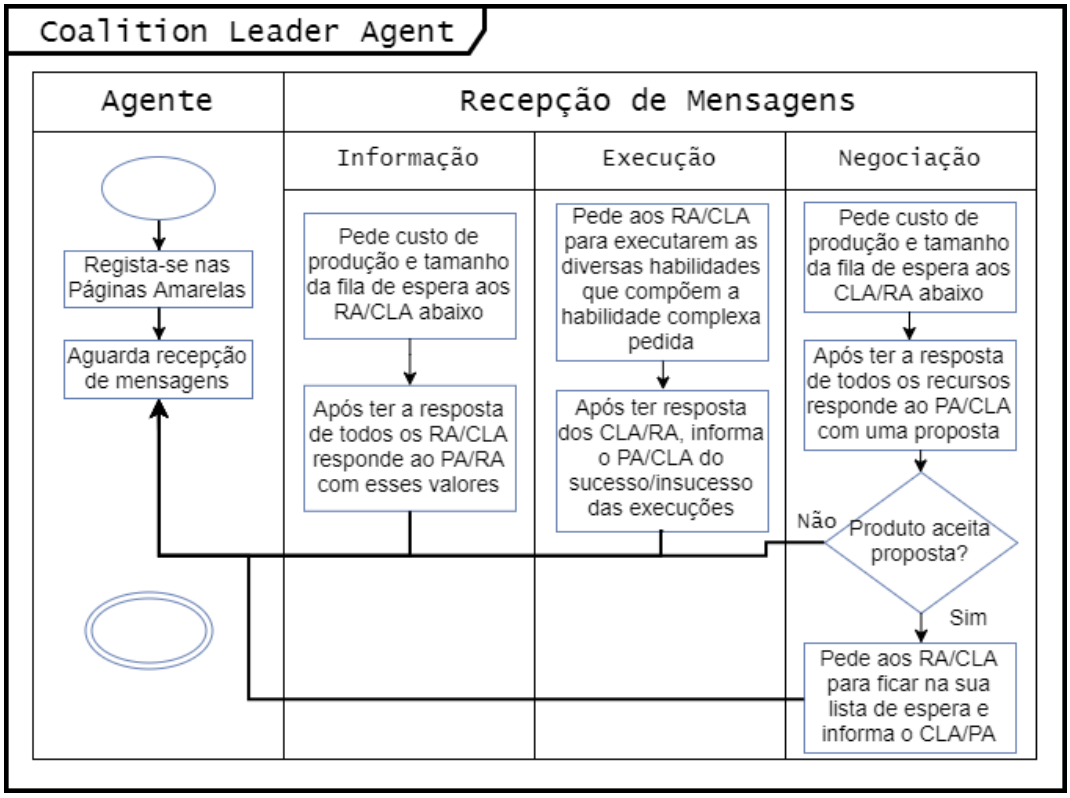

Figura 4.16 - Diagrama de atividade de um CLA (Parte 1)

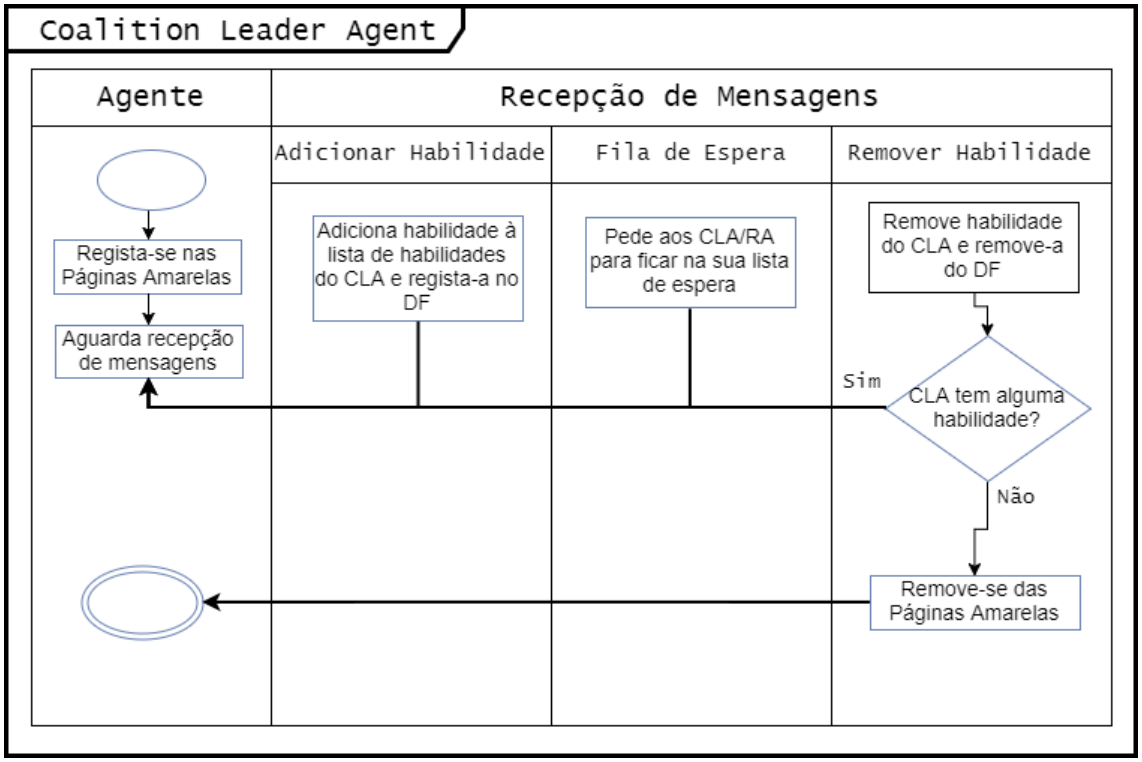

Figura 4.17 - Diagrama de atividade de um CLA (Parte 2)

O CLA, após ser iniciado, regista-se nas paginas amarelas e, em seguida, fica a aguardar quatro tipos de mensagem correspondentes a quatro ações distintas referentes à negociação, execução, adição e remoção de habilidades.

A negociação é semelhante à do RA, no entanto o CLA ao receber uma negociação irá pedir o valor de produção e o tamanho da fila de espera aos CLA e RA controlados por ele de modo a fazer uma proposta à habilidade complexa pretendida juntando assim os valores das diversas habilidades simples que a compõem. Caso a proposta seja aceite, este ainda informa os CLA e RA que rege de modo a esses poderem colocá-lo na lista de espera.

A execução de habilidade consiste em pedir a todos os RA e CLA que este CLA rege que executem uma determinada habilidade e, após receber a confirmação de todos, informa o produto do resultado.

Ao receber uma mensagem de que uma habilidade foi adicionada, o CLA regista esse serviço nas paginas amarelas.

Caso a mensagem seja de remoção de habilidade este remove a habilidade das paginas amarelas e caso não tenha mais nenhuma habilidade dá por terminado o seu ciclo.

Por fim, temos duas mensagens referentes à comunicação entre CLA – CLA: a mensagem de valores, onde um CLA recebe uma mensagem a pedir o valor de produção de uma dada habilidade e o tamanho da fila de espera dessa habilidade e pede esses valores aos CLA/RA que coordena, junta-os e envia a resposta; a mensagem de fila de espera, onde o CLA recebe uma mensagem para ser colocado na fila de espera e diz aos CLA/RA que rege para fazerem o mesmo.

4.3.4 Deployment Agent

Este agente tem como função lançar RA e CLA. O seu funcionamento é simples, após iniciar ele carrega a informação referente aos recursos e às habilidades complexas e, com base nessa informação vai correndo um conjunto de condições definidas de mono a lançar ou remover RA, lançar CLA e adicionar novas habilidades a um CLA ou remover uma que deixou de ser possível devido a algum recurso se ter desconectado. Este agente só termina quando o sistema for desligado. A figura 4.18 mostra, de uma forma geral, o funcionamento deste agente.

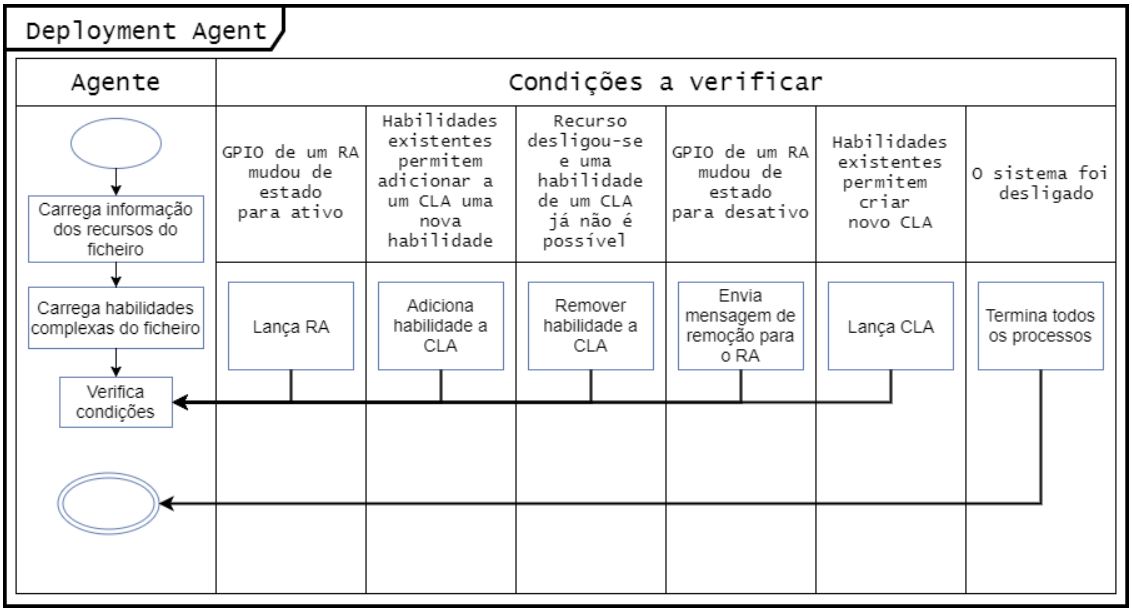

Figura 4.18 – Diagrama de atividades do DA

4.3.5 Comunicação entre Agentes

Uma parte fundamental do MAS é permitir comunicação entre os diversos agentes. Nesta solução pode-se destacar as comunicações existentes entre os agentes PA, CLA e RA. Estas são a base do sistema permitindo assim que os diversos agentes comuniquem entre si para realizar as diversas operações necessárias. A tabela 4.1 permite observar os agentes iniciantes e os participantes nas diversas comunicações existentes entre eles.

Tabela 4.1 - Comunicação entre agentes

Participantes / Iniciantes	PA	RA	CLA
PA	-	**X**	**X**
RA	-	-	-
CLA	-	**X**	**X**

5 Implementação da Solução

Este capitulo apresenta uma implementação possível para arquitetura ciber-física descrita no capitulo anterior.

A parte cibernética, composta pelo MAS, foi desenvolvida em Java com recurso a uma plataforma de desenvolvimento e execução de MAS nesta tecnologia denominada Java Agent Development Framework (JADE). Esta plataforma é amplamente utilizada em muitas aplicações devido à sua flexibilidade, boa documentação e sua capacidade de atender aos padrões de comunicação entre os agentes tendo ainda outras vantagens descritas em (Carrasco *et al.*, 2014), nomeadamente:

- Fornece uma camada estável para desenvolver aplicativos distribuídos tornando a complexidade da camada inferior transparente para o programador;
- Facilita questões como coordenação de agentes, segurança, comunicação, mobilidade, redundância e outros serviços básicos em uma arquitetura distribuída;
- É open-source. Assim, muitas pessoas contribuem para desenvolvê-lo e mantê-lo;
- Foi desenvolvido em Java logo beneficia das propriedades dessa tecnologia;
- É compatível com comunicação "Foundation for Intelligent Physical Agents" (FIPA), como tal, permite comunicar com agentes desenvolvidos em outros ambientes que também implementem FIPA;
- A sua API é intuitiva, fácil de aprender e simples de usar, reduzindo assim o tempo de desenvolvimento;
- Permite uma integração fácil de outras bibliotecas de raciocínio lógico, e PROTÉGÉ (*software*) para desenvolver ontologias.

Na parte física foi tido em conta os diversos requisitos descritos no capitulo anterior tendo sempre visão no preço final da solução. Esta contem um sistema operativo Linux de modo a correr Java SE Development Kit e o MAS.

Assim, este capitulo irá seguir a seguinte estrutura: a secção 5.1 oferece uma breve explicação dos protocolos de comunicação entre os agentes, a secção 5.2 descreve a implementação do sistema multiagente, desde a implementação de cada agente como as comunicações existentes entre eles e ainda onde estarão alocados na arquitetura proposta e, por fim, a secção 5.3 aponta as razões pela escolha de um Raspberry Pi 3 Modelo B para a implementação da solução e ainda refere a necessidade de alguns sistemas complementares bem como uma solução para os mesmos.

5.1 Comunicação FIPA

A Foundation for Intelligent Physical Agents (FIPA) é um órgão de desenvolvimento e estabelecimento de padrões (protocolos) de software para sistemas baseados em agentes.

Existem dois principais protocolos de comunicação entre agentes FIPA. Estes são o FIPA Request e o FIPA ContractNet.

5.1.1 FIPA Request

O FIPA Request oferece a possibilidade de um agente solicitar a realização de uma certa ação a outro agente (FIPA, 2002b). Para dar inicio a este protocolo, o agente iniciante deve enviar um pedido ao participante (Fig. 5.1).

Após o pedido, o participante pode optar por aceitar ou recusar enviando uma mensagem a concordar ou recusar. Caso aceite, executa o pedido e envia uma resposta face ao mesmo, ou seja, se este foi bem-sucedido ou não e pode ainda enviar o resultado no caso de ser.

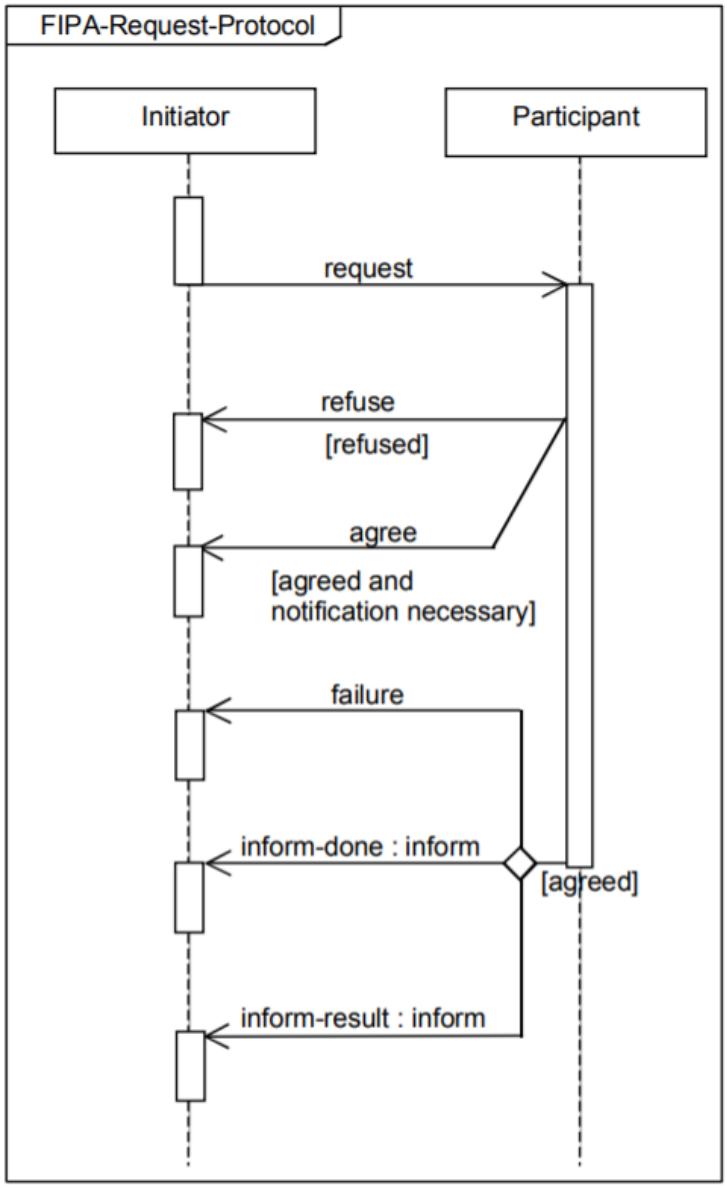

Figura 5.1 - Protocolo FIPA Request (FIPA, 2002b)

5.1.2 FIPA Contract Net

O FIPA Contract Net oferece a possibilidade de negociação entre o agente iniciante e outros agentes onde o iniciante pode avaliar diferentes propostas e escolher a que melhor se adapte às suas necessidades (FIPA, 2002a). A figura 5.2 descreve a interação entre o iniciante e m participantes.

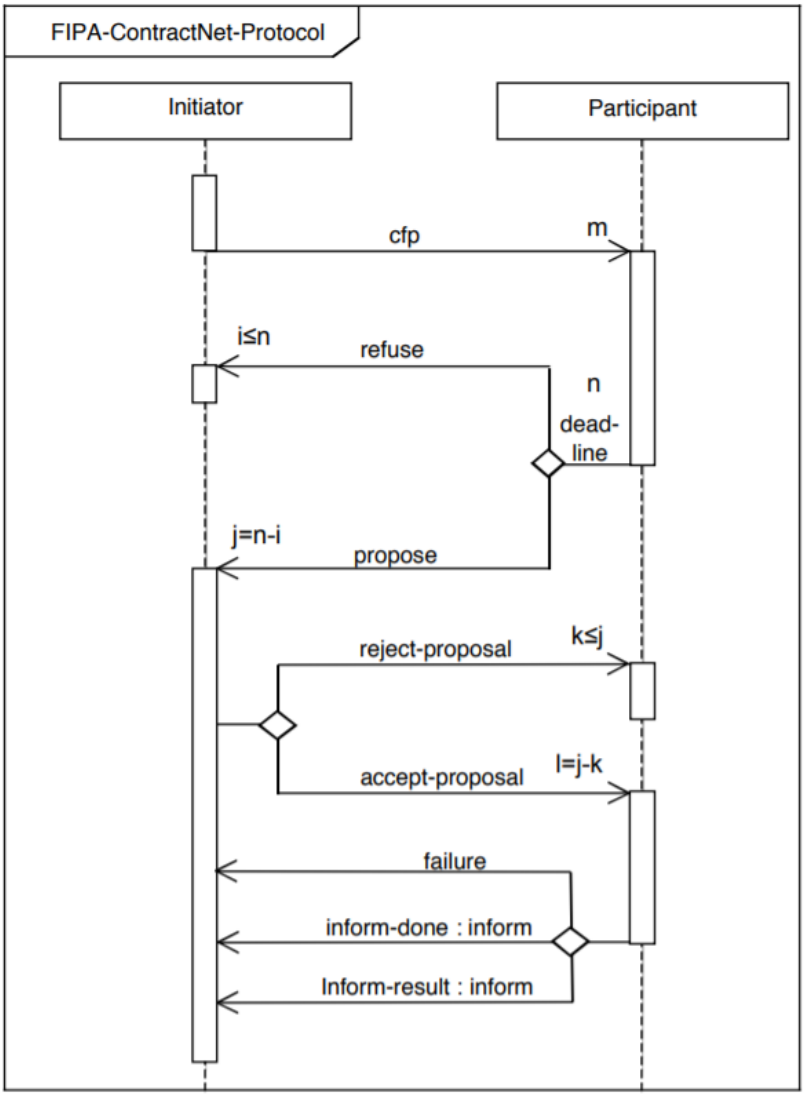

Figura 5.2 - Protocolo FIPA Contract Net (FIPA, 2002a)

Este protocolo tem inicio com um pedido de propostas aos diversos participantes. Estes por sua vez podem recusar ou responder com uma proposta.

Após o iniciador receber todas as mensagens, sejam rejeições ou propostas, este irá avaliar as mesmas de modo a escolher a ou as propostas que lhe forem mais favoráveis.

Cada proposta é assim respondida de acordo com a avaliação feita, ou seja, é enviada uma mensagem para aceitar ou rejeitar a proposta efetuada. Por sua vez, os participantes efetuam uma dada ação e respondem com uma mensagem a informar que a ação está concluída ou o resultado da mesma ou ainda a dizer que a ação não foi bem-sucedida.

5.2 Sistema Multiagente

Como já foi referido previamente, o MAS foi implementado em Java com recurso a JADE. As subsecções seguintes irão demonstrar o processo logico da implementação dos principais agentes e ainda as comunicações existentes entre eles.

5.2.1 Serviço de Páginas Amarelas

O serviço de páginas amarelas foi implementado através *Directory Facilitator* (DF) em vez do YPA. Este pertence à arquitetura do JADE.

O DF é um registro usado para armazenar e pesquisar os serviços fornecidos pelos agentes ativados na plataforma e permite aos agentes descobrir dinamicamente as descrições de um ou mais serviços disponíveis em um determinado momento.

No MAS, descobrir um serviço é uma tarefa crítica para encontrar uma instância de serviço particular, neste caso um RA ou CLA que o contem. O DF localiza os serviços aderindo a um conjunto de requisitos relativos à consulta (Kim *et al.*, 2009). Como tal, um agente que deseja procurar por serviços deve solicitar ao DF fornecendo uma descrição do modelo do serviço pretendido. O resultado da pesquisa é retornado para todas as descrições que correspondem ao modelo fornecido.

5.2.2 Classes Auxiliares

Usou-se classes auxiliares para evitar a repetição de código e a facilitar a manutenção do mesmo. Entre as quais encontram-se as classes Constants, DFInteraction e Serialize.

A classe Constants (Fig. 5.3) define constantes que irão ser usadas por todas as restantes classes sempre que necessário. Esta facilita a alteraçao de temporizadores ou ontologias entre outros. Esta classe não tem assim nenhuma função associada.

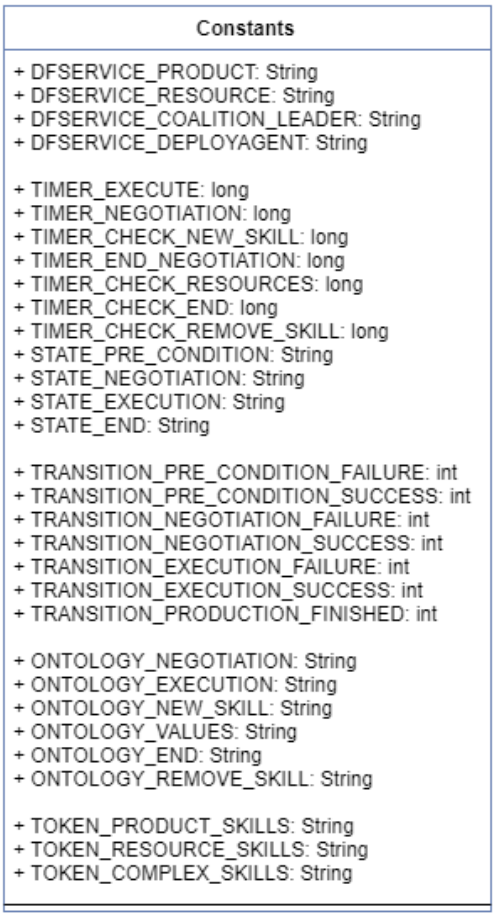

Figura 5.3 - Classe Constants

A classe DFInteraction define algumas funções para a interação com o DF (Fig. 5.4).

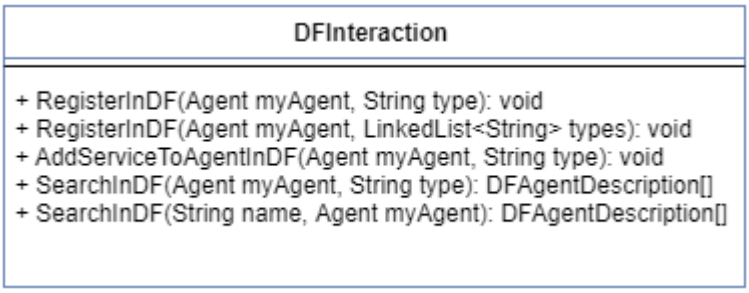

Figura 5.4 - Classe DFInteraction

Uma vez que, em JADE, a troca de conteúdo entre mensagens é realizada através de texto (*String*), são necessárias funções capazes de realizar a troca entre objeto para texto e vice-versa. Essa troca é facilitada com recurso à classe Serialize (Fig. 5.5).

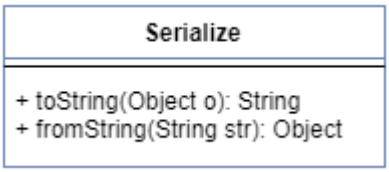

Figura 5.5 - Classe Serialize

5.2.3 Product Agent

O Agente responsável por um produto tem uma implementação relativamente simples (Fig. 5.6). Esta classe prolonga a classe Agent do JADE.

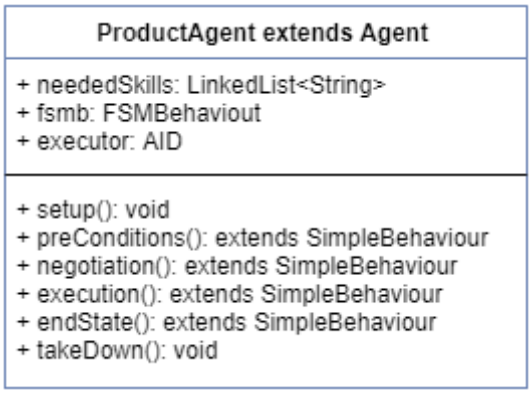

Figura 5.6 - Classe ProductAgent

Este tem uma função setup responsável por se registar no DF, contruir a máquina de estados e executar a máquina de estados.

A máquina de estados é composta por quatro comportamentos simples, nomeadamente representados por preConditions, negotiation, execution e endState.

No preConditions, são verificadas as pré-condições para a produção, ou seja, é verificado se no sistema existem recursos a oferecer cada uma das habilidades necessárias ao produto. Esta verificação é feita com recurso a uma pesquisa no DF.

No negotiation, é efetuado a negociação da primeira habilidade necessária. Para tal, é efetuada uma pesquiza no DF pelos recursos que oferecem a mesma habilidade e é iniciado uma negociação com os mesmos. Após todos responderem, são comparadas as propostas e é aceite a que tiver melhor relação custo/fila de espera.

Com base na proposta aceite da negociação, é enviando um pedido de execução ao recurso "vencedor". Após a confirmação de sucesso da execução da habilidade, esta é retirada da lista de habilidades necessárias e, caso a lista esteja vazia a produção acaba, passando assim ao estado final (endState), caso contrario, volta ao estado de negociação.

No endState é transmitida uma mensagem ao utilizador sobre o resultado do processo de produção. Todos os estados anteriores podem vir diretamente a este caso haja um erro e, como tal, este estado tanto pode enviar uma mensagem de produção com sucesso como pode enviar uma mensagem em que a produção terminou devido a um erro em algum dos estados, por exemplo um erro na execução de uma habilidade.

Por fim tempos a classe takeDown que é executada a quanto a conclusão do agente, ou seja, quando este termina a máquina de estados. Neste estado é efetuada a sua remoção do DF.

5.2.4 Resource Agent

Um recurso do sistema é implantando prolongando a classe Agent. Este é ainda composto por um *hardware* cuja implementação é efetuada através de uma interface (ResourceHardwareInterface). A figura 5.7 representa assim o diagrama de classes referente ao Resource Agent e à sua interface com o *hardware*.

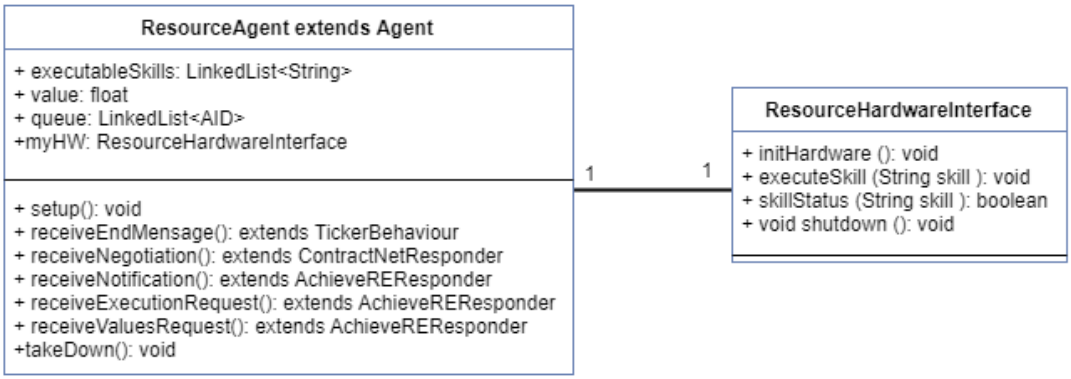

Figura 5.7 - Diagrama de Classes associado às classes ResourceAgent e ResourceHardwareInterface

À semelhança do PA, a classe que representa o RA contem uma função de setup e uma de takeDown. As diferenças encontram se principalmente no setup onde o RA, em vez de uma máquina de estados, cria uma nova classe referente ao hardware e regista as suas habilidades no DF. No setup, é ainda feita a atribuição das variáveis referentes ao preço de utilização e habilidades

que é capaz de executar cujos valores provêm da passagem de parâmetros para dentro da classe ResourceAgent. É ainda iniciada a fila de espera como vazia.

O receiveEndMensage é um comportamento cíclico que ocorre a uma frequência, previamente definida nas constantes, que verifica a chegada de uma mensagem especifica com a ontologia ONTOLOGY_END. Esta mensagem é proveniente do Deployment Agent e tem o objetivo de terminar o agente a quanto o seu hardware seja desconectado do sistema.

O receiveNegotiation corresponde ao comportamento de resposta de um FIPA ContractNet iniciado no PA. Este, caso tenha a habilidade em questão envia o seu custo e a sua fila de espera e, caso seja aceite, coloca o produto na sua fila de espera.

O receiveValuesRequest é semelhante ao receiveNegotiation, no entanto, este trata-se apenas de um pedido de valores referentes ao custo de produção e fila de espera caso tenha a habilidade. É implementado de acordo com um FIPA Request.

O receiveNotification é implementado seguindo o protocolo FIPA Request e, ao receber a mensagem, coloca o agente que a enviou na fila de espera.

Por fim, o receiveExecutionRequest é responsável por executar uma dada habilidade requerida. Implementado por um FIPA Request, este comportamento recebe o pedido de execução de uma habilidade, verifica se pode executá-la e confirma se o agente que pediu a execução é o próximo da fila. Caso o agente não seja o próximo na fila, a mensagem é colocada em espera ate esta situação se verificar. Caso exista algum erro é enviado uma mensagem de erro, caso contrario é enviado uma mensagem a informar que a execução teve sucesso.

5.2.5 Coalition Leader Agent

A classe CoalitionLeaderAgent prolonga a classe agente e serve como “descodificador” de habilidades compostas logo, ao receber qualquer tipo de pedido, esta classe tem que simplificar a habilidade e reencaminhar o pedido para as classes que representam os agentes que este coordena. Esta tem uma subclasse (Proposal) que representa uma proposta, ou seja, o preço, a filia de espera, a habilidade que está a tratar e os que a vão executar. A implementação destas classes pode ser analisada em mais pormenor na figura que se segue (Fig.5.8).

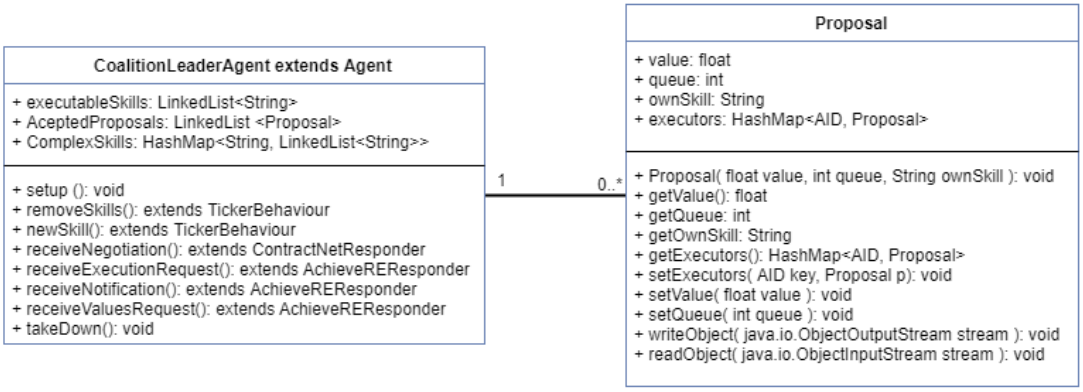

Figura 5.8 - Diagrama de Classes associado às classes CoalitionLeaderAgent e Proposal

Dentro da classe CoalitionLeaderAgent temos um setup onde o agente regista as suas habilidades no DF e adiciona os diversos comportamentos.

Os comportamentos removeSkills e newSkills são utilizados para adicionar ou remover habilidades à lista de habilidades que este agente é capaz de realizar. Em caso de se remover uma habilidade e a lista ficar vazia, este agente é terminado recorrendo ao takeDown, onde o agente se remove do DF.

À semelhança do RA, o CoalitionLeader também tem os comportamentos receiveNegotiation, receiveExecution, receiveNotification e receiveValuesRequest. A diferença nestes comportamentos é a criação de novos comportamentos para a comunicação com os CLA/RA que este coordena, ficando a aguardar a resposta dos mesmos para retomar a comunicação com o CLA ou PA que a iniciou.

A subclasse Proposal é utilizada apenas dentro de CLA e na comunicação entre agentes do tipo CLA. Esta cria propostas ou junta à já existente à medida que se sobe na hierarquia. Como tal, esta é criada ao receber propostas de RA e vai sendo combinada no CLA até que o próximo da hierarquia seja o PA. Nesse caso são retirados os valores necessários para o produto e enviados ao mesmo.

Para tal, esta classe é composta por quatro variáveis, nomeadamente o valor associado à produção da habilidade, a fila de espera, a habilidade que a faz, e os recursos que o fazem. É composta ainda de funções que permitem preencher/retirar os valores destas variáveis e ainda duas funções auxiliares para a conversão do objeto em texto e vice-versa.

5.2.6 Deployment Agent

O DA é implementado na classe DeploymentAgent, prolongando a classe Agent, e é ainda composto por 3 subclasses auxiliares (Fig. 5.9).

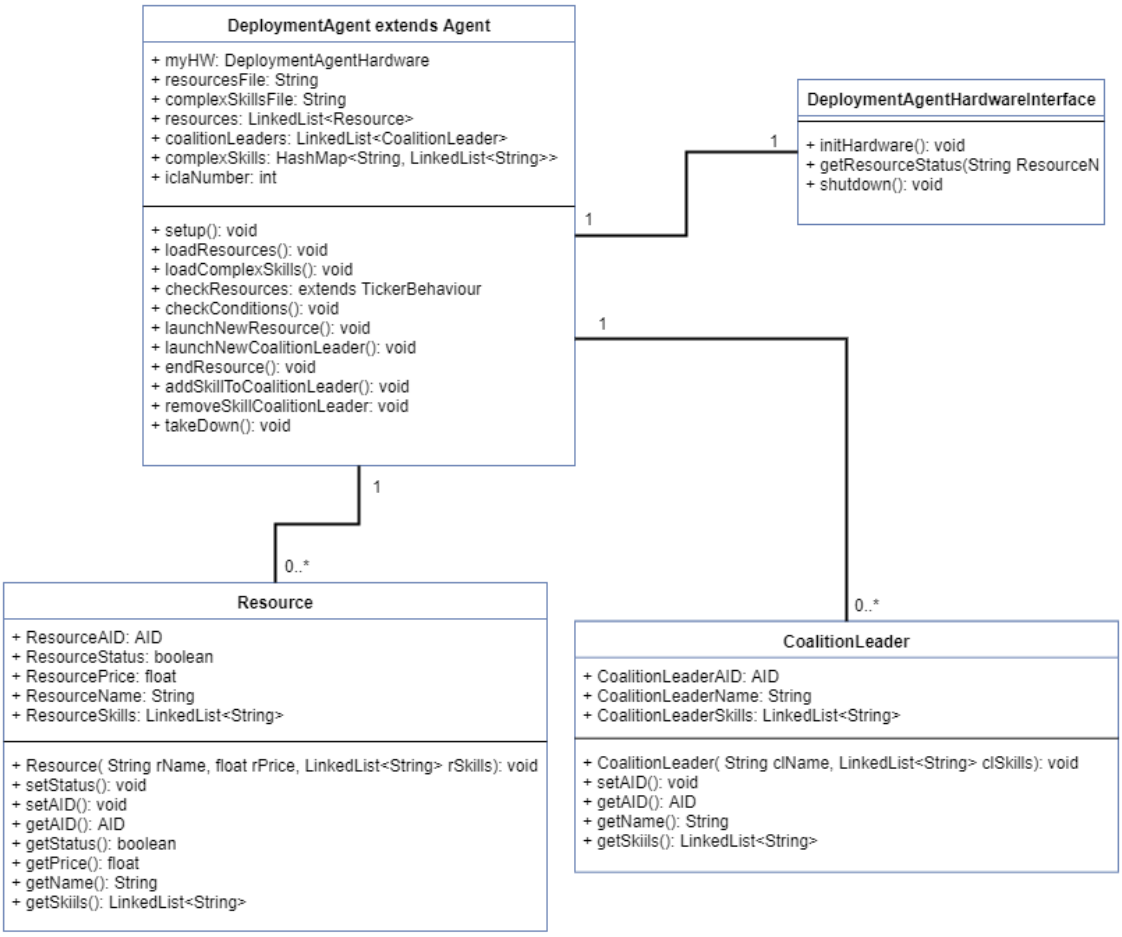

Figura 5.9 - Diagrama de Classes associado às classes DeploymentAgent, Resource, CoalitionLeader e DeploymentHardwareInterface

Dentro das subclasses encontra-se a classe DeploymentHardwareInterface responsável pela comunicação com *hardware*. Esta comunicação é efetuada para verificar quais recursos se encontram ligados ao sistema.

Por outro lado, as classes Resource e CoalitionLeader são classes auxiliares. Estas são utilizadas para guardar a lista de recursos lidos do ficheiro e guardar a lista de CoalitionLeaders que se vão criando.

A classe DeploymentAgent é responsável por toda a gestão de recursos existentes, ou seja, esta utiliza ficheiros para definir os recursos que podem ser utilizados e a lista de habilidades complexas que os seus CLA irão reconhecer.

Como esta classe é a responsável de lançar agentes do tipo RA ou CLA, a classe tem de analisar a lista de recursos possíveis e verificar se estes se encontram ligados ou não para os lançar ou pedir aos RA para terminarem a sua execução.

Com base nos RA ativos e na lista de habilidades complexas, esta classe verifica ainda que habilidades dessa mesma lista consegue realizar e, seguindo essa analise, adiciona ou remove habilidades complexas a um determinado CLA ou inicia novos CLA.

5.2.7 Comunicação entre Agentes

Em 4.3.5 referiu-se as comunicações existentes entre os principais agentes, nomeadamente os PA, os CLA e os RA. Estas comunicações foram implementadas com recurso a dois protocolos FIPA, o FIPA Request e o FIPA ContractNet.

Este subcapítulo irá apresentar as diversas ontologias associadas a cada comunicação e as variáveis envolventes.

- **Comunicação PA-RA:**

A comunicação mais simples é a comunicação entre o PA e o RA. Estes dois agentes comunicam em duas situações distintas: uma para negociar uma habilidade e a outra para que o PA solicite a execução de uma habilidade ao RA.

A negociação entre estes dois agentes consiste num FIPA Contract NET onde o agente produto pede o preço de execução bem como a fila de espera aos diversos recursos com a habilidade necessária. Neste caso, os recursos oferecem uma proposta com estes dados e, após o produto ter a resposta de todos os RA com a habilidade necessária, este recusa as que não lhe interessam e aceita a que lhe é mais favorável.

Do lado do RA, caso a proposta seja recusada, a comunicação acaba naquele momento, caso contrario, o RA coloca o PA na sua lista de espera e informa o produto de que a negociação terminou com sucesso.

A figura 5.10 mostra assim esta comunicação tendo em conta os diversos casos possíveis a ocorrer.

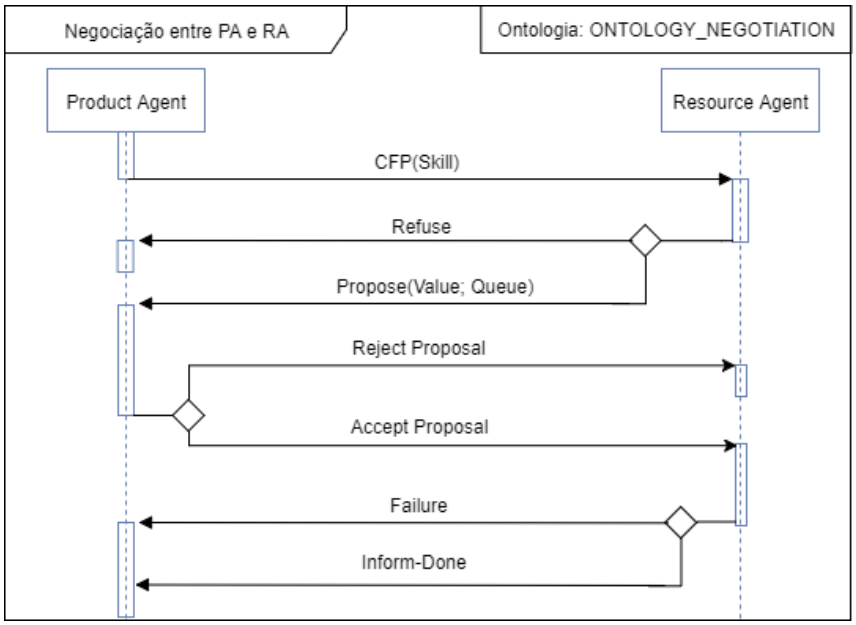

Figura 5.10 - Diagrama de sequência referente à negociação entre PA e RA

Por sua vez, a execução de uma habilidade trata-se de um FIPA Request onde um PA requere a execução de uma dada habilidade ao RA e o RA recusa ou aceita o pedido de execução e, caso aceite, informa posteriormente sobre o resultado da ação (Figura 5.11).

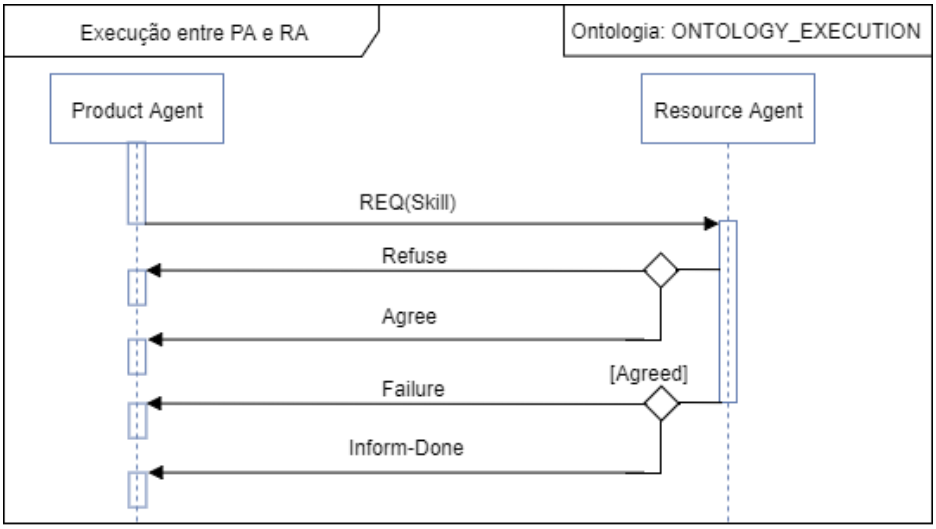

Figura 5.11 - Diagrama de sequência da comunicação referente à execução de habilidades entre PA e RA

- **Comunicação PA-CLA-RA:**

A comunicação PA-CLA-RA permite ao sistema a execução de processos mais complexos. Para tal, quando um produto pretende realizar uma habilidade complexa, este comunica com os CLA que tenham essa habilidade e os CLA por sua vez é que irão comunicar com os diversos RA ou CLA que tenham o conjunto de habilidades que formem a anterior. O Caso de um CLA comunicar com outro CLA será analisado posteriormente.

Do pondo de vista do produto, comunicar com um CLA ou um RA é o mesmo e, como tal, a comunicação é semelhante à descrita anteriormente. O que realmente muda é que, no meio da negociação ou execução, estas são colocadas em espera e existe uma comunicação com os RA.

Na negociação, a comunicação entre o PA e o CLA é interrompida duas vezes, a primeira quando é pedido uma proposta e a segunda quando uma proposta é aceite. Na primeira, o CLA ao receber um CFP, verifica quais, dos agentes que coordena, realizam as habilidades simples necessárias para a habilidade pedida e inicia um FIPA Request com os RA a pedir os dados necessários e, só após ter a resposta de todos é que o CLA conjuga o resultado e envia uma proposta ao produto. Na segunda, após a aceitação de uma dada proposta, o CLA inicia outro FIPA Request para pedir aos RA para o colocarem na fila de espera para execução. Mais uma vez, só depois de ter a resposta de todos é que informa o PA.

A figura 5.12 mostra assim o diagrama de sequencias da negociação envolvendo PA, CLA e RA. É de notar que a ultima comunicação efetuada entre o CLA e o RA é utilizada uma ontologia diferente da principal, ou seja, é usada a ontologia ONTOLOGY_NEGOTIATION com exceção daquela comunicação que é efetuada com a ontologia ONTOLOGY_NOTIFICATION.

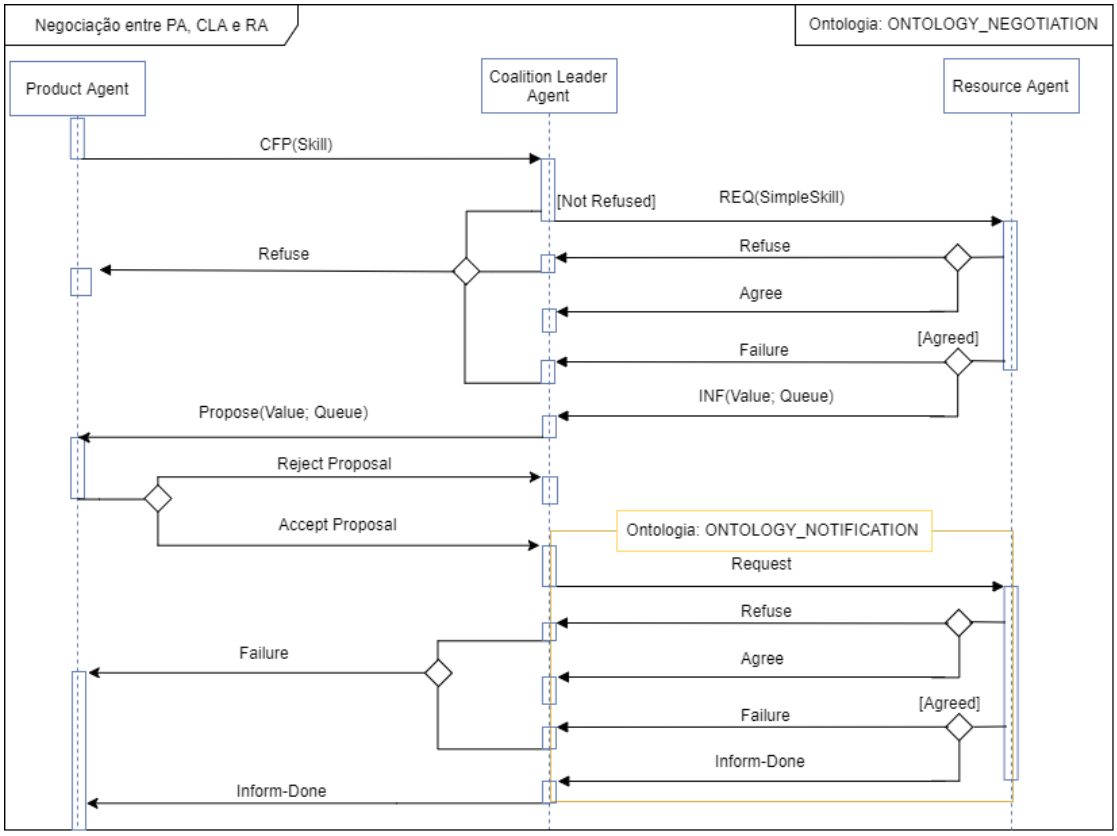

Figura 5.12 - Diagrama de sequência referente à negociação entre PA, CLA e RA

Na execução, a comunicação entre o PA e o CLA é apenas interrompida uma vez onde, posteriormente a um pedido de execução de uma certa habilidade do PA ao CLA, o CLA vê os recursos responsáveis pela execução das habilidades que compõem a pedida e solicita aos mesmos a execução das habilidades. O CLA e os RA podem rejeitar caso não reconheçam a habilidade e o resultado pode ser informativo em caso da habilidade ser executada com sucesso ou de falha caso contrario (Figura 5.13).

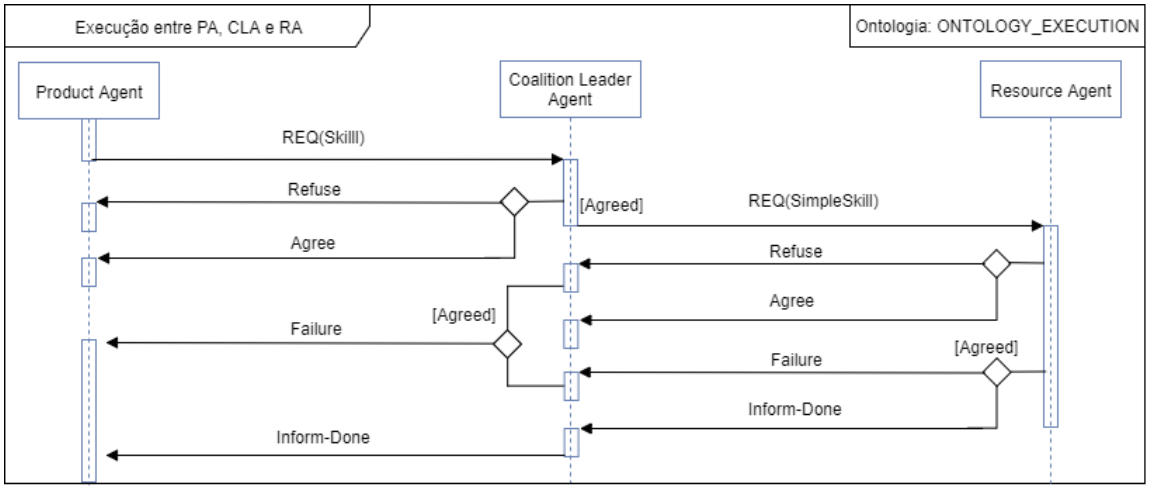

Figura 5.13 - Diagrama de sequência referente à execução de uma habilidade entre PA, CLA e RA

- **Comunicação CLA-CLA-RA:**

Em casos mais concretos, quando uma dada habilidade complexa solicitada pelo PA é composta por habilidades complexas, é necessário a comunicação de CLA com CLA. Nestes casos, a comunicação PA-CLA permanece a mesma e a CLA-CLA é semelhante à de CLA-RA, contudo um CLA ao receber um pedido de execução de uma dada habilidade, ou um pedido do valor de execução e tamanho da fila de espera ou ainda um pedido para ser colocado na lista de espera, este, identicamente à comunicação CLA-RA, comunica os RA/CLA que coordena e, posteriormente a ter uma resposta dos mesmos informa o CLA que iniciou ou pedidos. As figuras 5.14, 5.15 e 5.16 mostram assim um exemplo possível destas 3 comunicações.

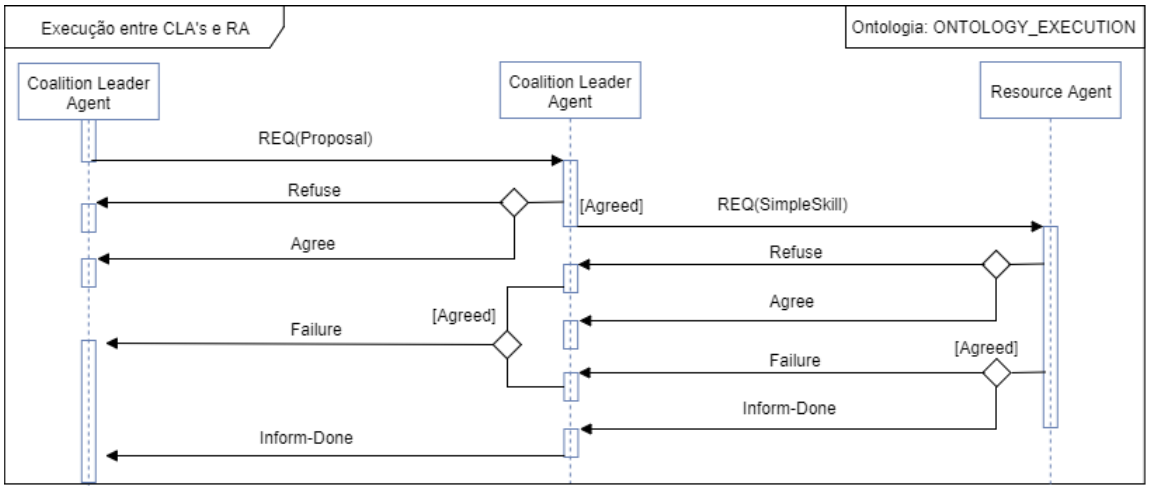

Figura 5.14 - Diagrama de sequência referente à execução de habilidades entre CLA's e RA

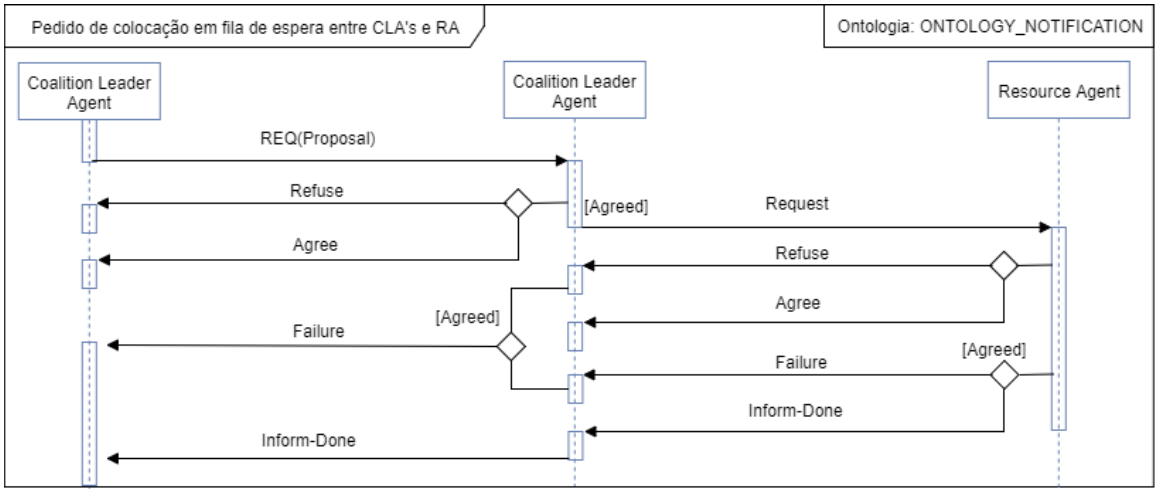

Figura 5.15 - Diagrama de sequência da comunicação entre CLA's e RA referente ao pedido de colocação em fila de espera

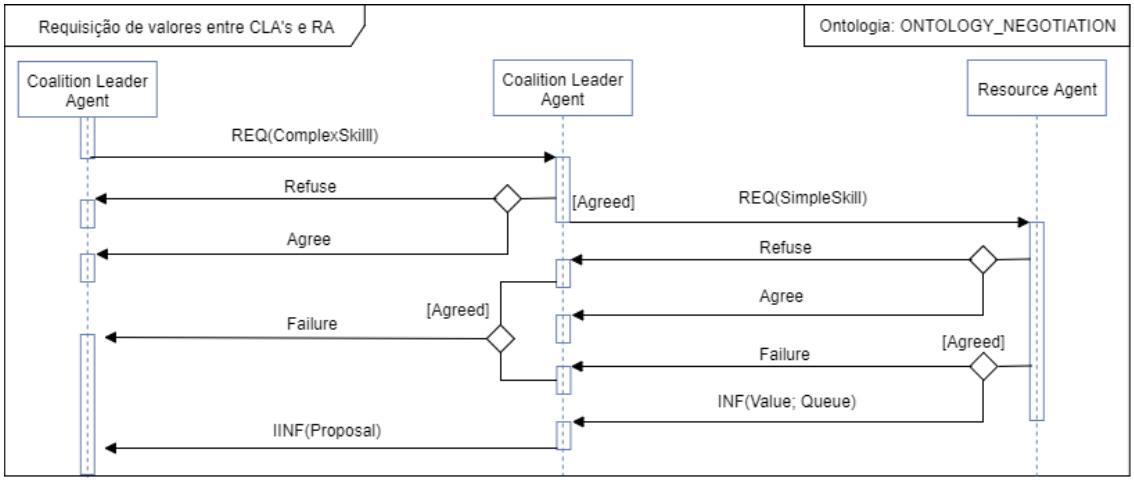

Figura 5.16 - Diagrama de sequência referente à requisição de valores entre CLA's e RA

5.3 Módulos de Produção

Os MAS apresentados anteriormente correm em parte nos dispositivos IoT. Estes por sua vez necessitam de alguns sistemas auxiliares para a copulação com o Hardware. Para tal, fez-se uma comparação entre alguns dispositivos IoT, elegendo um para implementação da solução, e apresentou-se um exemplo de sistema auxiliar para o dispositivo escolhido tendo em conta que o sistema de produção iria ser testado tendo em conta um demonstrador físico (apesentado no capitulo seguinte).

5.3.1 Dispositivo IoT

Previamente foi mencionado que o dispositivo IoT deve corresponder a uma plataforma capaz de interagir com o Hardware, comunicar com outros dispositivos/sistemas e processamento e armazenamento suficientes para correr o software referido em 5.2. De forma a cumprir os requisitos a cima mencionados e obter a melhor relação performance-preço foi realizado um estudo sobre alguns dispositivos existentes onde se podem destacar 3: Raspberry Pi 3 Model B, Arduino YÚN e BeagleBone Black (Fig 5.17).

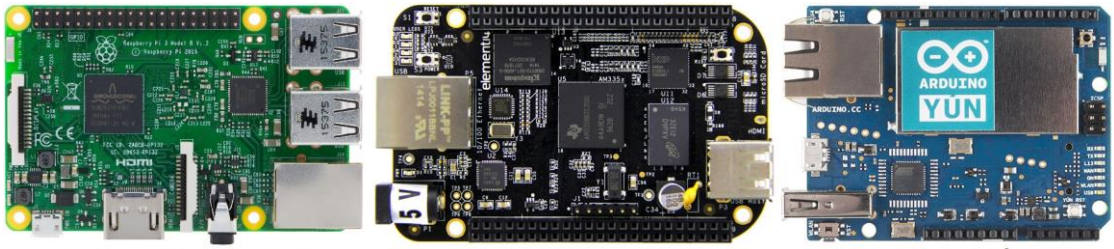

Figura 5.17 - Raspberry Pi 3 Modelo B, Arduino YÚN e BeagleBone Black

O Raspberry PI é um computador de dimensões reduzidas que permite realizar diversas coisas que um computador normal permite (por exemplo, jogar, navegar na internet, processamento de texto, etc) bem como realizar projetos eletrónicos. Em especifico, o Rasberry Pi 3 Modelo B apresenta um processador de 4 núcleos ARMv8 1.2 GHz de arquitetura 64 bits e uma memoria RAM de 1 GB. Incorpora ainda conectividade wireless LAN 802.11b/g/n e Bluetooth Clássico e Low Energy (BLE), um slot para cartão de memória Micro SD, um slot para câmara e ainda 40 pinos de GPIO. Este permite ainda correr sistemas operativos como Linux, Windows e outros como Raspbian.

O BeagleBoard é também um computador de dimensões reduzidas desenvolvido pela Texas Instruments. É open-source tanto a nível de Hardware como Software e foi desenvolvido para fins educacionais. O Modelo BeagleBoard Black é aconselhado para computação física e pequenas aplicações embutidas. Oferece assim um processador AM335x 1GHz ARM® Cortex-A8 de arquitetura 32 bit e 512MB de RAM. Tem ainda 4GB de armazenamento interno e slot para cartão de memoria. Tem opção de escolha entre o modelo com porta LAN ou o modelo onde esta porta é substituída por uma placa de comunicação wireless oferecendo assim WiFi e Bluetooth. Para além de software personalizado, este suporta ainda Debian, Android, Ubuntu, Cloud9 IDE em Node.js entre outros.

O Arduino, embora seja considerado um computador de software e hardware open-source, foi desenvolvido para ser um microcontrolador, ao contrario do Raspberry e do BeagleBone que foram desenvolvidos como microprocessadores. O Arduino YÚN é assim constituído por um microprocessador Atheros AR9331 MIPS @400MHz, um microcontrolador ATmega32u4 e uma memoria RAM de 64 MB. Suporta ainda Ethernet e WiFi e tem 20 pinos GPIO, uma porta USB 2.0 e um slot para cartão de memoria Micro SD.

A tabela 5.1 apresenta assim uma comparação entre estes 3 dispositivos em alguns fatores como processador, RAM, armazenamento, conectividade e preço para uma escolha mais assertiva da plataforma física para a solução proposta.

Tabela 5.1 - Comparação entre dispositivos físicos para integração de IoT

	Raspberry Pi 3 Modelo B	**BeagleBone Black**	**Arduino YÚN**
Processador	Broadcom BCM2837 ARMv8 Quad-Core a 1.2 GHz 64-bit	AM335x ARM® Cortex-A8 Dual-Core a 1 GHz 32.bit	Atheros AR9331 MIPS a 400MHz
RAM	1GB SDRAM	512MB DDR3 RAM	64 MB DDR2
Armazenamento	Micro SD	4 GB interno + Micro SD	Micro SD
Conectividade	Ethernet:1 x 10/100Mbps Sem Fios: Placa onboard Wi-Fi BCM43438 + Bluetooth 4.1 4 x USB2.0 1 x HDMI 1.4 1 x Jack 3.5mm 1 x CSI 1 x DSO 1 x MicroSD 40 Pinos GPIO	Sem Fios: 802.11b/g/n + Bluetooth 4.1 plus BLE 1 x Micro HDMI 1 x Micro SD 2 x 46 pinos GPIO 1 x USB 2.0	Ethernet:802.3 10/100Mbit/s WiFi: 802.11b/g/n 2.4 GHz Pinos Digitais I/O: 20 Pinos Analógicos I/O: 12 PWM: 7 1 x USB 2.0 1 x Micro SD
Preço	40 € a 50 €	60 € a 70 €	80 € a 90 €

Após esta breve analise, podemos excluir o Arduino YÚN não só pelo preço elevado face aos restantes, mas também pelo facto do mesmo ser de microcontrolador em vez de microprocessadores. Ao limitar o poder de processamento compromete a arquitetura proposta. Este também não permite correr sistemas operativos nem a execução de multitarefas.

Ao compararmos o Raspberry Pi 3 Modelo B e o BeagleBone Black notamos algumas semelhanças. Ambos são microprocessadores com capacidade de conexão a rede e com pinos GPIO.

A grande vantagem do BeagleBone Black é o numero de pinos GPIO, 92 face às 40 do Raspberry Pi Modelo B. Já a ligação à rede tem a desvantagem de ter-se que optar por uma placa com porta Ethernet ou uma placa com comunicação wireless.

Comparando o processador de ambos, o poder de processamento do Raspberry Pi 3 Modelo B é maior e tem ainda o dobro da memoria RAM. A nível de armazenamento o BeagleBone Black tem memória interna e capacidade de expansão recorrendo a um Micro SD ao contrario do Raspberry Pi Modelo B onde o armazenamento é exclusivamente dependente do Micro SD utilizado.

Tendo em vista os requisitos previamente apresentados, o Raspberry Pi 3 Modelo B destaca-se face aos outros não só em poder de processamento como também a nível de custo e ainda devido à grande variedade de conectividades disponíveis. Como tal, este foi o dispositivo escolhido para implementar esta arquitetura.

5.3.2 Sistemas Complementares

Uma vez que, no dispositivo escolhido em 5.3.1, os pinos GPIO operam a uma tensão de 3.3 V e que o Kit Staudinger (descrito no capítulo seguinte) usado para demonstração opera a 24 V, existiu a necessidade de um sistema complementar que fizesse esta conversão de tensão de forma segura visto que os pinos do Raspberry Pi 3 Modelo B não apresentam qualquer proteção contra picos de corrente/tensão. Este sistema teria ainda de ser de baixo custo de modo a garantir um orçamento total reduzido.

Existem inúmeras soluções para o problema apresentado, contudo um *optocoupler* é a escolha mais popular devido a diversas vantagens como: tamanho e preço reduzido, extensa vida útil, capacidade de isolamento e resistente a interferências externas (Cheng *et al.*, 2014; Kang, Kim and Cho, 2015).

Um *optocoupler* é constituído por um díodo emissor de luz (LED) na sua entrada e um foto-transístor de silício à saída tal como indica o esquemático na figura 5.18 (Takahashi, Nishihara and Koyama, 1984).

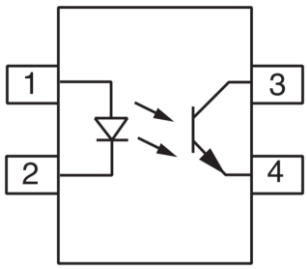

Figura 5.18 - Esquemático de um *Optocoupler*

Optou-se assim por usar placas prefabricadas composta por 4 *optocoupler* cada no valor aprox imado de 5 euros (Fig. 5.19). Estas são responsáveis por converter os sinais de tensão. É importante realçar que a placa escolhida só permite realizar uma conversão de tensão por placa, ou seja, se queremos converter sinal de 3.3 V para 24 V e o inverso teremos de usar 2 placas.

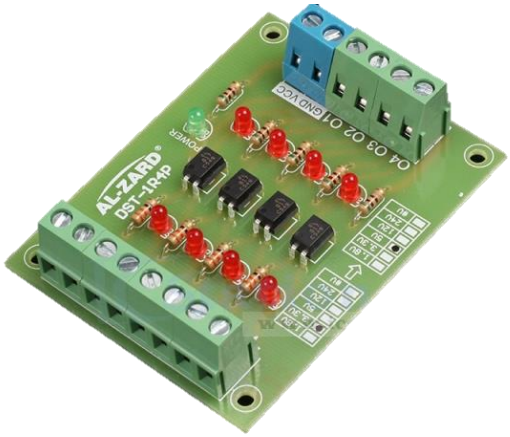

Figura 5.19 - Placa Optocoupler de 4 bits

Contudo, devido à deterioração da superfície e à geração de defeitos de cristal, o o*ptocoupler* faz com que o resultado de saída sofra uma pequena deterioração resultando na diminuição

no rácio de transferência de corrente. Este fenómeno impedia assim a ativação dos relés no sistema de teste (Shi *et al.*, 2014).

Este problema foi resolvido recorrendo a implementação de um simples componente. Foram adicionados transístores NPN a cada saída do *optocoupler* compensando assim a falta de corrente previamente mencionada. A saída do optocoupler foi assim ligada à base do transístor, o coletor do transístor foi ligado à fonte de tensão do sistema de teste e o emissor do transístor foi ligado ao demonstrador (Figura 5.20).

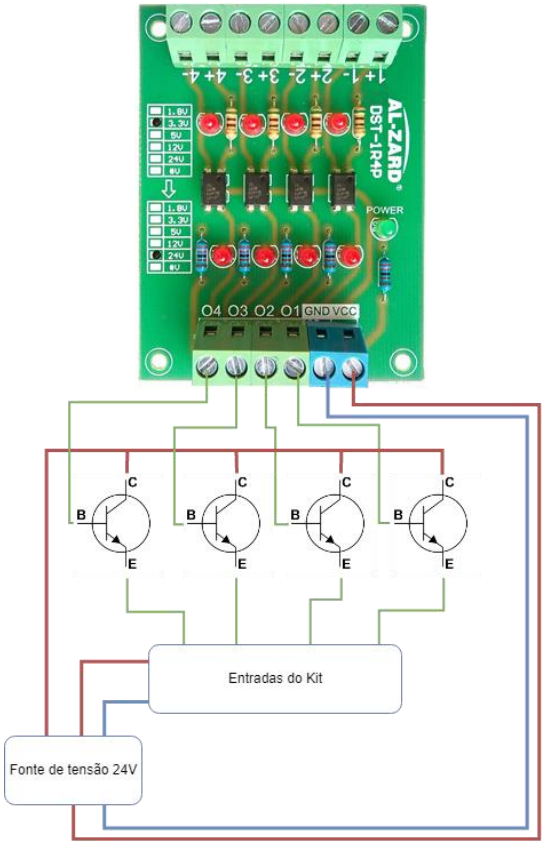

Figura 5.20 - Ligação efetuada entre a placa de *optocoupler*, os transístores e o demonstrador físico

A figura 5.21 demonstra uma montagem do sistema, desde a nuvem até ao demonstrador físico.

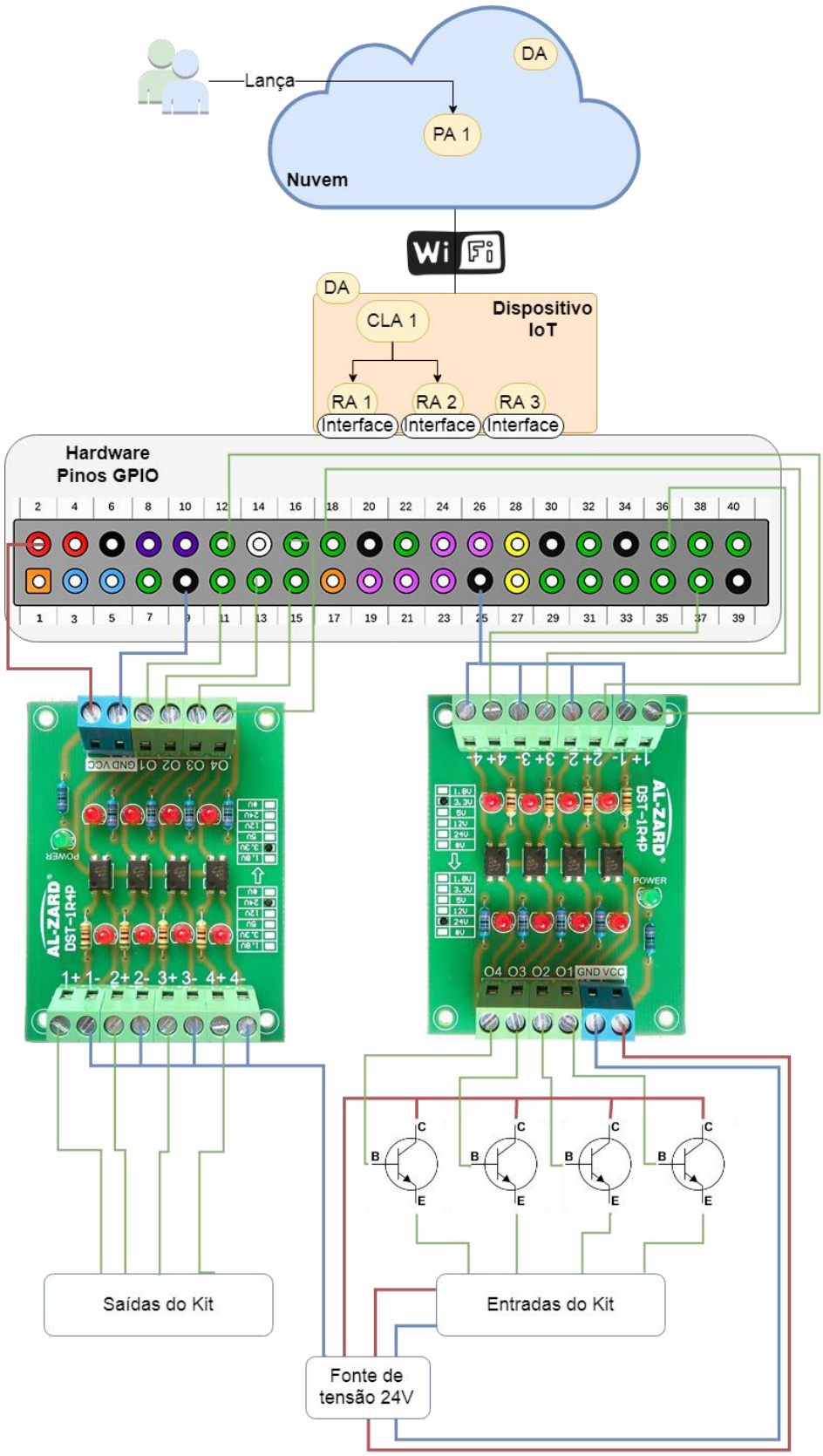

Figura 5.21 - Esquemático referente à implementação descrevendo os diversos componentes

6 Validações e Testes

Para se por a prova a arquitetura proposta teve-se em conta dois tipos de teste: um teste onde se testou o sistema num demonstrador físico e um teste onde o objetivo passou por verificar o desempenho do sistema no dispositivo IoT escolhido.

O MAS foi alojado numa rede local da qual estavam conectados os dispositivos IoT e um computador pessoal.

Os PA e alguns CLA eram lançados num computador pessoal e os RA e os restantes CLA eram alojados nos dispositivos IoT.

Cada dispositivo IoT, após ser ligado à alimentação, iniciava automaticamente o seu sistema operativo (Linux) e conectava-se automaticamente ao computador pessoal. Este ligava ainda os seus pinos GPIO e verifica o estado dos mesmos. Consoante este estado, os recursos virtuais associados eram ligados e, caso formassem um conjunto de habilidades simples pré-definidas alguns CLA também eram iniciados.

Ao ligarem-se ao computador pessoal, era vista a sua localização e, caso existissem 2 dispositivos na mesma posição, eram vistas as suas habilidades e, à semelhança dos dispositivos, eram testadas um certo conjunto de predefinições para o lançamento de CLA no computador pessoal.

6.1 Teste recorrendo a um demonstrador físico

Para este teste usou-se o demonstrador Correia Transportador Compacta, artigo numero 226005 da marca Staudinger GMBH (Fig. 6.1).

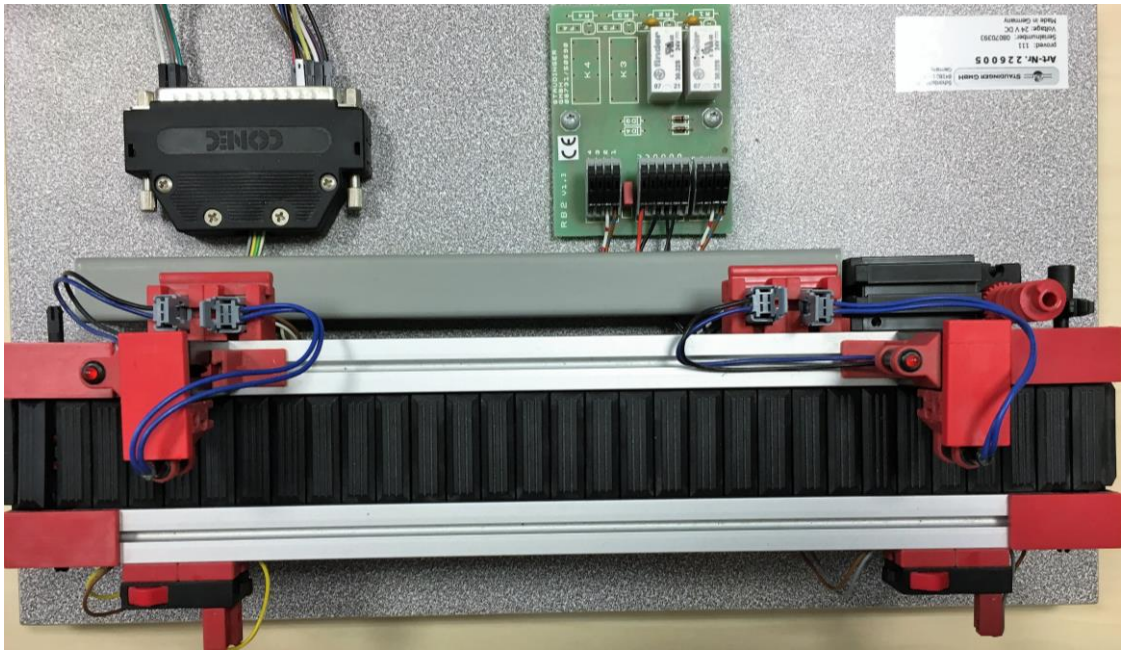

Figura 6.1 - Kit de Demonstração

Este modelo simula uma unidade móvel com posições iniciais e finais definidas com precisão semelhante aos de produção de produtos em série. É composto por 4 sensores, dos quais dois são mecânicos e por 3 atuadores, compostos por 2 LED's e 1 motor de duas direções. Todo o sistema trabalha ainda a uma tensão de alimentação de 24 V DC.

Para o teste teve-se assim em conta que a ação de mover para a esquerda, mover para a direita, acender um LED e pressionar um botão seriam habilidades simples e que cada LED e sensor mecânico correspondiam a habilidades diferentes. Considerou-se ainda a possibilidade de parar o tapete rolante qualquer um dos sensores, mecânicos ou não. O exemplo de um LED e dos dois tipos de sensor podem ser vistos na figura 6.2.

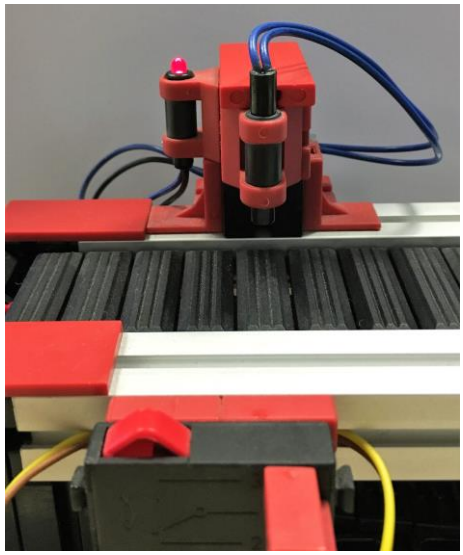

Figura 6.2 - Exemplo de LED e Sensores do Demonstrador

Considerou-se então o uso de dois dispositivos IoT, cada um com dois RA e um deles com um CLA. O Produto foi ainda lançado num computador pessoal (Fig 6.3). O CLA seria assim responsável por uma habilidade complexa formada por um sensor mecânico e um LED.

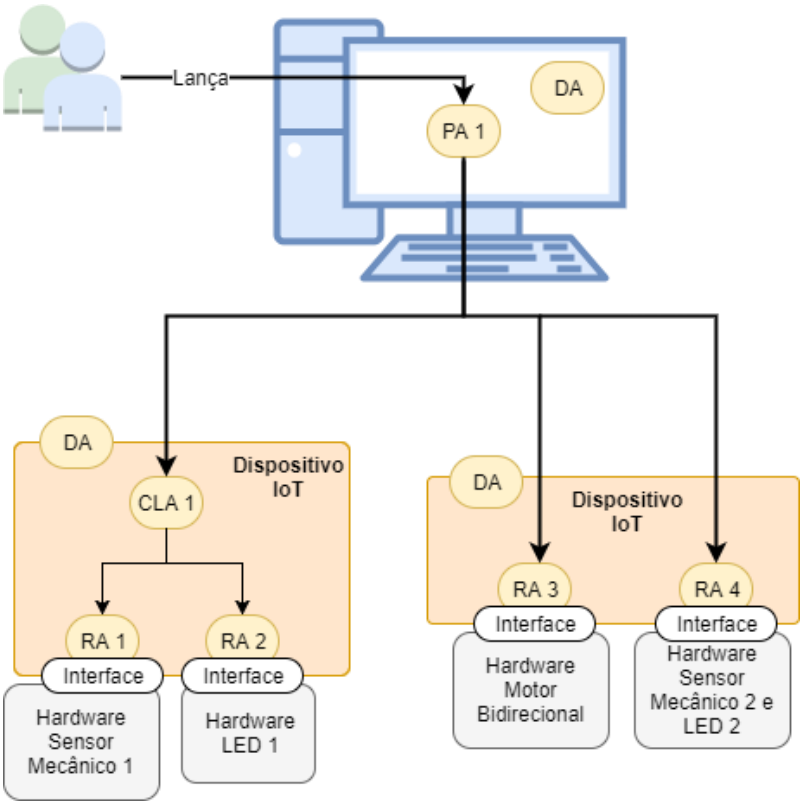

Figura 6.3 - Teste efetuado com auxilio do Demonstrador

Neste teste o produto foi colocado a meio do tapete rolante e lançado com o plano de execução equivalente a: Move_Direita, Habilidade_Composta_Sensor_Mecânico1_E_LED1, Move_Esquerda, Sensor_Mecânico2, Move_Direita.

Como se pode observar, são executadas quatro habilidades atómicas e uma habilidade composta por duas atómicas. O teste foi assim efetuado com sucesso.

6.2 Teste de Desempenho

Saber como um sistema se comporta perante diversos cenários possibilita não só ver o seu desempenho como também encontrar limites no seu processamento. Para tal, realizaram-se diversos testes com o intuito de observar o desempenho do dispositivo IoT escolhido na arquitetura proposta.

Utilizou-se assim um computador pessoal MSI 2QE Ghost Pro 4k de 4ª geração a alocar a parte principal do MAS, ou seja, os PA e todos os agentes nativos do JADE como a DF. Utilizou-se ainda um Raspberry Pi 3 Modelo B para correr então todos os RA e CLA. O Raspberry Pi automaticamente ligava-se ao computador via WiFi e registava assim os seus recursos. A rede WiFi utilizada teve por base uma rede privada com velocidades de download de 200 MB/s e de upload de 100 MB/s na qual a velocidade máxima WiFi proveniente do router consta em ¼ da principal. Por sua vez o router usado foi o modelo tg784n v3 da technicolor.

Para a realização dos testes seguintes teve-se em conta um único dispositivo IoT e limitou-se o mesmo para que cada recurso só permitisse a execução de uma dada habilidade atómica, ou seja, supondo que essa habilidade é a habilidade A então o recurso em questão só pode fazer a A, se for necessário a habilidade B então é necessário um segundo recurso.

Procedeu-se assim aos testes que se seguem.

6.2.1 Habilidades Atómicas versus Habilidades Complexas

Procedeu-se à produção de um produto cujo plano de ação passa pela verificação, negociação e execução de uma única habilidade atómica. Foram retiradas 30 amostras do tempo de execução do plano dando origem à tabela 6.1:

Tabela 6.1 - Tempo produção de uma habilidade atómica

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	0,125	0,093	0,172	4,184E-04	0,020

À semelhança do anterior, repetiu-se o processo, mas desta vez para a produção de duas habilidades atómicas e, tal como referido anteriormente, este plano requere dois recursos. A tabela 6.2 apresenta o resultado do teste efetuado.

Tabela 6.2 - Tempo de produção de duas habilidades atómicas

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	0,288	0,203	0,516	0,003	0,056

Em seguida procedeu se à execução de duas habilidades atómicas através da execução de uma complexa, equivalente assim a um CLA e dois RA (Tabela 6.3).

Tabela 6.3 - Tempo produção de uma habilidade composta por duas atómicas

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	0,251	0,218	0,328	0,001	0,032

6.2.2 Escalamento no numero de CLA e RA

De modo a testar a escalabilidade do sistema, executou-se apenas habilidades complexas compostas por uma atómica e a complexa anterior, ou seja, partindo da primeira complexa, composta por duas atómicas, a seguinte irá ser composta pela anterior e uma atómica e assim sucessivamente.

Isto traduz-se num CLA a coordenar 1 RA e 1 outro CLA com exceção do primeiro CLA que coordena dois RA.

Para facilitar a visualização da diferença de tempos entre cada etapa da execução voltou-se a efetuar os testes anteriores e, em seguida, o escalamento.

No entanto, para a realização destes testes, procedeu-se à criação de planos de 50 habilidades iguais em cada amostra, ou seja, para testar o tempo do processo de execução de uma habilidade A, considera-se o plano de execução de 50 habilidades A e retira-se a soma dos tempos das 50 habilidades. Repete-se o processo de modo a obter 30 amostras. No caso de uma dada habilidade composta C, constituída por duas habilidades atómicas A e B, será criado um plano de execução constituído por 50 Habilidades C. Este plano ira assim executar 50 vezes a habilidade A e 50 vezes a B.

A implementação referente a este teste pode ser analisada na figura 6.4.

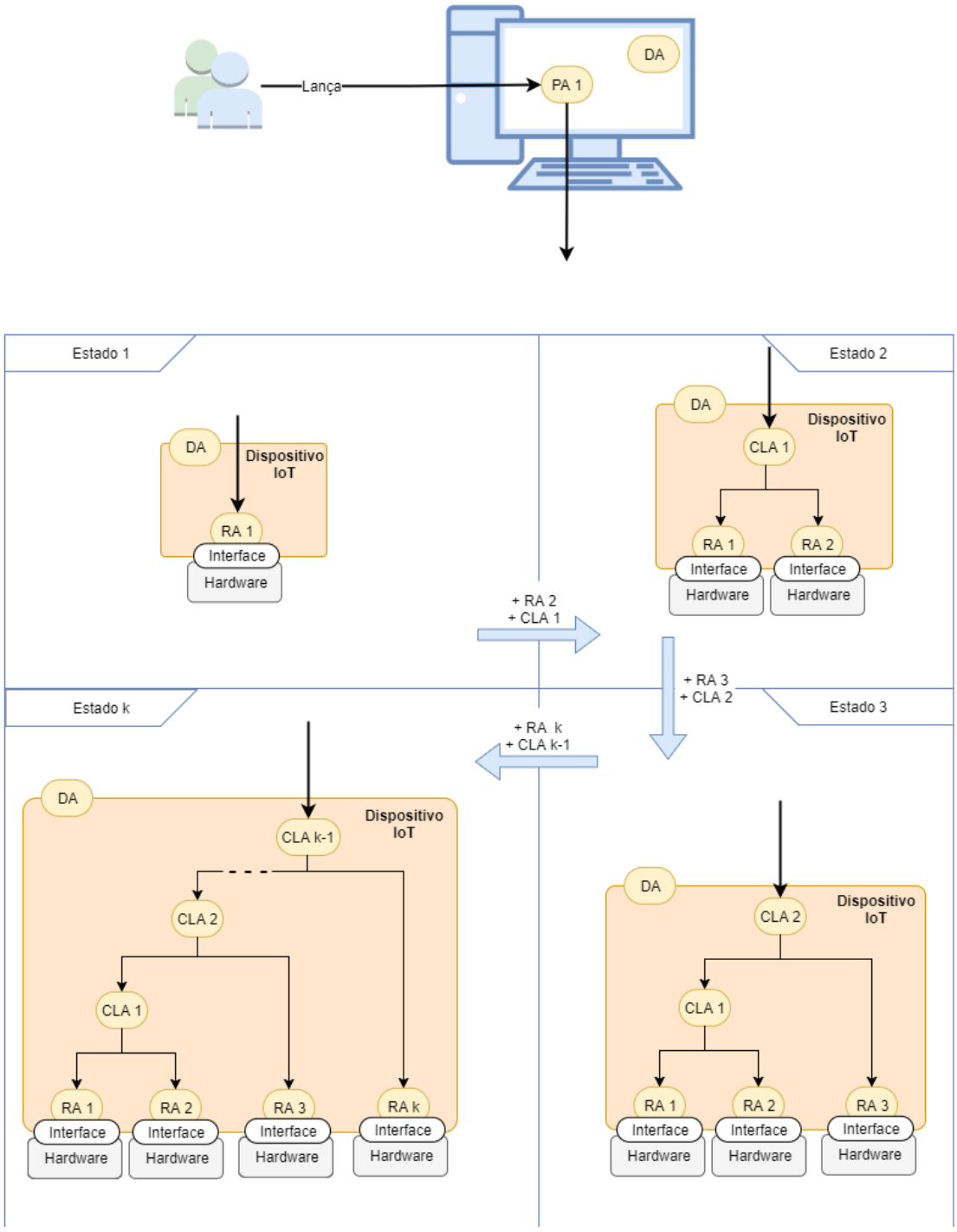

Figura 6.4 - Sistema utilizado no teste de estabilidade em diferentes estados.

- **Uma habilidade atómica:**

Tabela 6.4 - Escalamento de uma habilidade atómica

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	5,432567	4,955	6,143	0,101	0,317

- **Duas Habilidade atómicas:**

Neste caso especifico será considerado um plano de produção com 50 habilidades de um tipo e 50 de outro, obtendo um total de 100 habilidades.

Tabela 6.5 - Escalamento de duas habilidades atómicas

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	12,193	11,043	16,185	0,986	0,993

- **Primeira Habilidade Composta:**

Tabela 6.6 - Escalamento referente à primeira habilidade complexa

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	11,6347	11,171	12,394	0,073	0,270

Comparando os resultados da tabela 6.5 com os da tabela 6.6 pode-se verificar que a execução de habilidades compostas em vez de simples reduz o tempo de execução. Quando um produto necessita de duas ou mais habilidades atómicas no mesmo local físico que possam ser executadas em simultâneo, é aconselhado executar uma complexa composta pelas duas habilidades atómicas caso exista.

- **Segunda Habilidade Composta:**

Tabela 6.7 - Escalamento referente à segunda habilidade complexa

Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
30	22,46693	22,030	22,939	0,042	0,205

- **Resultado após as 10 Habilidades Compostas:**

Após realizar os testes anteriores para 10 habilidades compostas obtemos o resultado representado na tabela 6.8.

Tabela 6.8 - Escalamento de habilidades complexas

Número da Habilidade Composta	Número de Amostras	Média (Segundos)	Mínimo (Segundos)	Máximo (Segundos)	Variância (Segundos)	Desvio Padrão (Segundos)
1	30	11,6347	11,171	12,394	0,073	0,270
2	30	22,46693	22,030	22,939	0,042	0,205
3	30	35,22597	34,626	35,639	0,046	0,215
4	30	49,60683	48,840	50,104	0,064	0,253
5	30	65,48837	64,683	66,100	0,105	0,325
6	30	83,4124	82,938	84,166	0,077	0,278
7	30	103,5425	101,550	110,896	7,127	2,670
8	30	123,6654	122,918	124,299	0,109	0,270
9	30	147,6482	146,573	148,368	0,164	0,406
10	30	173,2813	172,523	174,536	0,231	0,480

Para uma visão melhor sobre o efeito de carga sobre a velocidade de execução do plano do PA construiu-se um gráfico (Fig. 6.5) onde é representado um conjunto de habilidades e o seu tempo de execução. As habilidades “a” e “b” correspondem a habilidades simples e as representadas por números a compostas.

É importante realçar que, para cada habilidade representada, o tempo de execução do plano corresponde à execução de 50 vezes a mesma habilidade, facilitando assim verificar as diferenças à medida que se aumenta a carga no sistema.

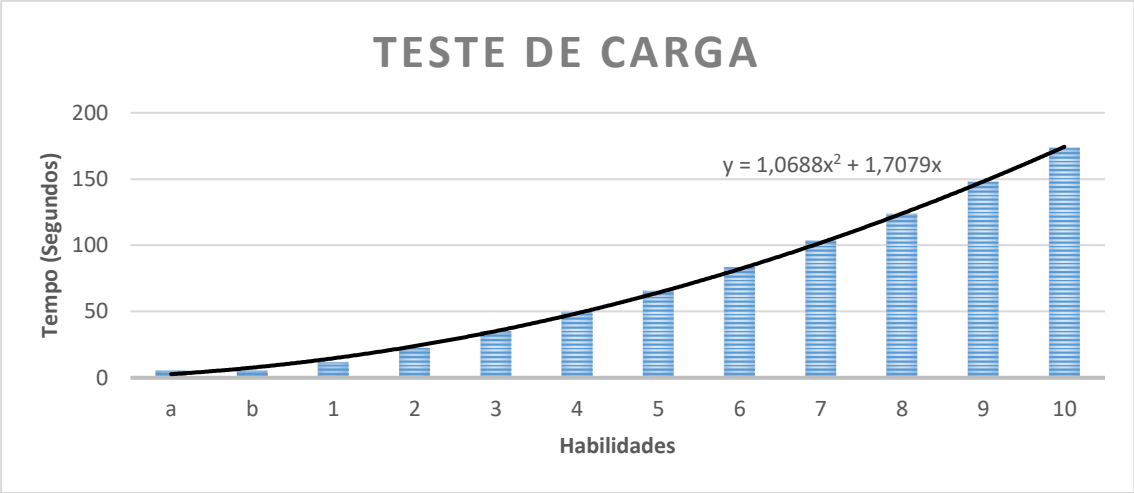

Figura 6.5 - Teste de carga num Raspberry Pi 3 Modelo B

Ao observar o gráfico é possível verificar que o tempo de execução vai aumentando de habilidade em habilidade como era previsto visto que corresponde ao aumento da execução de mais 50 habilidades. Este segue uma tendência de tempo $y = 1{,}0688x^2 + 1{,}7079x$ segundos onde x representa a complexidade da habilidade em que x = 1 representa a habilidade "a" e x = 12 representa a habilidade "10".

Este tempo por habilidade está diretamente ligado a dois grandes fatores principais, a velocidade de processamento e a velocidade de transmissão de dados entre os agentes, uma vez que o atraso na conexão cria grande atraso quando associado a um numero elevado de execuções. Neste caso, a executar 50 habilidades de cada tipo, este atraso nota-se consideravelmente no Raspberry Pi.

No entanto, devido ao baixo custo do dispositivo IoT escolhido, é ainda possível melhorar este problema dividindo a carga por vários dispositivos.

Caso o sistema seja realizado todo no mesmo dispositivo (Fig. 6.6), o atraso referente ao processamento ainda é revelado e mostra que um fator importante é a conexão entre os dispositivos. É de notar que este gráfico foi obtido com o sistema a correr totalmente no computador pessoal a cima referido.

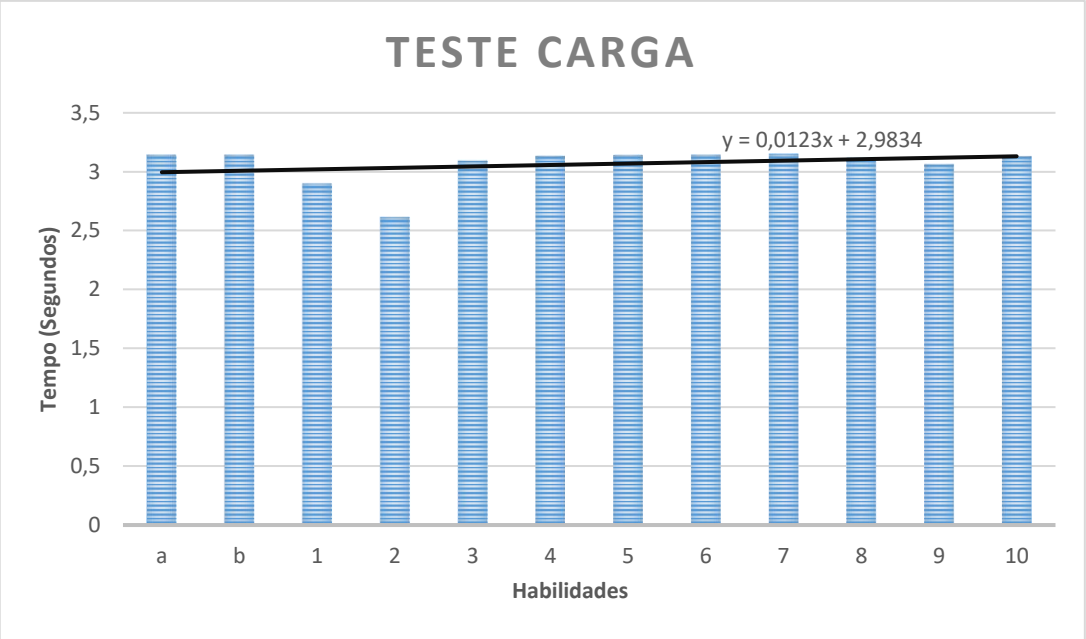

Figura 6.6 - Teste de carga do sistema num único dispositivo

Conclui-se assim que a ligação entre dispositivos se torna um fator fundamentar a ser melhorado na arquitetura para a realização de habilidades muito complexas. Nas habilidades menos complexas esta demora nem é notada para uma utilização comum deste tipo de sistemas.

É notório que o poder de processamento dos dois dispositivos é diferente e, como tal, a diferença no tempo de execução ainda é mais elevada.

7 Conclusões e Trabalho Futuro

7.1 Conclusões

Conclui-se que a utilização de dispositivos IoT e sistemas multiagentes permitem um controlo dinâmico e flexível de linhas de montagem mantendo um custo relativamente baixo. Conclui-se ainda que a arquitetura apresentada responde afirmativamente à pergunta inicial: é possível desenhar um módulo de produção a um custo reduzido capaz de abstrair um componente de produção e oferecer uma solução de controlo flexível.

As características desta arquitetura, tais como a modularidade, a capacidade de *Plug&Produce*, um sistema descentralizado e hierárquico, permitem que um sistema possa ser reorganizado de acordo com a necessidade de produção. Parte disto só é possível devido à utilização de uma arquitetura ciber-física de produção modular onde cada bloco do sistema (módulo) pode ser realocado ou substituído de acordo com as necessidades correntes.

Os testes realizados com recurso ao kit de demonstração permitem validar a utilização da arquitetura proposta em sistemas industriais. Esta abordagem concorre diretamente com sistemas tradicionais de controlo industrial que normalmente têm um preço mais elevado e características dos sistemas centralizados dificultando profundamente a produção de produtos altamente personalizáveis.

Os testes realizados à performance do sistema comprovam que o sistema é eficaz em habilidades de baixo grau de complexidade e vai perdendo a sua eficiência a medida que a complexidade aumenta. Como tal, as melhorias devem passar pela eficácia e rapidez da comunicação entre cada elemento do sistema bem como do processamento do sistema em geral. Isto torna-se um fator chave para que o sistema possa ser aplicado a grande escala de complexidade mantendo uma eficiência satisfatória.

7.2 Trabalho Futuro

Trabalhos futuros neste projeto devem refletir-se principalmente no melhoramento da comunicação entre agentes, permitindo assim realizar operações mais complexas mantendo a eficiência do sistema.

Pode-se ainda ter em conta uma placa auxiliar mais flexível de modo a oferecer ao dispositivo IoT uma melhor e maior adaptação aos sistemas de produção, como por exemplo permitir que o dispositivo controle um maior numero de recursos ou recursos mais complexos (que necessitem de maior numero de inputs/outputs) ou ainda recursos com características analógicas efetuando algumas modificações na placa auxiliar com a adição de conversores analógico-digital e digital-analógico permitindo assim um controlo tanto digital como analógico.

Deve-se também passar a ter em conta o sistema de transporte na arquitetura de modo a que a esta se torne mais robusta e eficiente. Uma correta implementação do sistema de transporte aumenta a rapidez da execução do plano com base na localização física de cada estação.

8 Referências

Aloi, G. *et al.* (2016) ‘A mobile multi-technology gateway to enable IoT interoperability’, *Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pp. 259–264. doi: 10.1109/IoTDI.2015.29.

Andersen, A. L., Nielsen, K. and Brunoe, T. D. (2016) ‘Prerequisites and Barriers for the Development of Reconfigurable Manufacturing Systems for High Speed Ramp-up’, *Procedia CIRP*. The Author(s), 51, pp. 7–12. doi: 10.1016/j.procir.2016.05.043.

Arai, T. *et al.* (2001) ‘Holonic assembly system with Plug and Produce’, 46, pp. 289–299.

Baheti, R. and Gill, H. (2011) ‘Cyber-physical Systems’, *The Impact of Control Technology*, (1), pp. 161--166. doi: 10.1145/1795194.1795205.

Carrasco, A. *et al.* (2014) ‘PeMMAS: A tool for studying the performance of multiagent systems developed in JADE’, *IEEE Transactions on Human-Machine Systems*, 44(2), pp. 180–189. doi: 10.1109/THMS.2014.2302993.

Ceanu, R. F. B. and Chen, F. F. (2006) ‘Development and applications of holonic manufacturing systems : a survey’, pp. 111–131.

Cheng, G. *et al.* (2014) ‘Comparative Analysis between CTR and Low- Frequency Noiseto Characterize the Optocoupler Reliability’, (61201028), pp. 36–40.

Díaz, M., Martín, C. and Rubio, B. (2016) ‘State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing’, *Journal of Network and Computer Applications*. Elsevier, 67, pp. 99–117. doi: 10.1016/j.jnca.2016.01.010.

FIPA, (Fundation for Intelligent Physical Agents) (2002a) ‘FIPA Contract Net Interaction Protocol Specification’, *Architecture*, (SC00029H), p. 9. Available at: http://www.mit.bme.hu/projects/intcom99/9106vimm/fipa/XC00029E.pdf.

FIPA, (Fundation for Intelligent Physical Agents) (2002b) ‘FIPA Request Interaction Protocol Specification’, *Online*. Available at: http://www.fipa.org/specs/fipa00026/SC00026H.pdf.

Frei, R., Barata, J. and Onori, M. (2006) ‘EVOLVABLE ASSEMBLY SYSTEMS’.

Fysarakis, K. *et al.* (2015) ‘Node . DPWS : Efficient Web Services for the IoT Node . DPWS : Efficient Web Services for the IoT’, (JANUARY), pp. 1–11.

Garcia, M. V. *et al.* (2016) ‘OPC-UA communications integration using a CPPS architecture’, *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–6. doi:

10.1109/ETCM.2016.7750838.

Grüner, S., Pfrommer, J. and Palm, F. (2016) 'RESTful Industrial Communication with OPC UA', *IEEE Transactions on Industrial Informatics*, 12(5), pp. 1832–1841. doi: 10.1109/TII.2016.2530404.

Han, S. N. *et al.* (2015) 'DPWSim: A Devices Profile for Web Services (DPWS) Simulator', *Internet of Things Journal, IEEE*, 2(3), pp. 221–229. doi: 10.1109/JIOT.2014.2388131.

Huang, T., Solvang, W. D. and Yu, H. (2016) 'An Introduction of Small-scale Intelligent Manufacturing System Regarding its design objectives and relevant tools and technologies', (June), pp. 31–39.

Jorge, P. *et al.* (2004) 'An Agile and Adaptive Holonic Architecture for Manufacturing', (January).

Kang, S. W., Kim, H. J. and Cho, B. H. (2015) 'A dual-loop voltage control for optocoupler feedback in capacitive output filter converter', *2015 IEEE Energy Conversion Congress and Exposition, ECCE 2015*, 2, pp. 6825–6831. doi: 10.1109/ECCE.2015.7310615.

Kim, B. S. *et al.* (2009) 'Agent platform-based directory facilitator for efficient service discovery', *CyberC 2009 - International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 101–107. doi: 10.1109/CYBERC.2009.5342177.

Koren, Y. *et al.* (1999) 'Reconfigurable manufacturing systems', *Annals of the CIRP*, 48(2), pp. 527–540.

Koren, Y. (2005) 'Reconfigurable Manufacturing and Beyond', (May), pp. 1–7.

Lazinica, A. and Katalinic, B. (2005) 'BEHAVIOUR OF TRANSPORT MOBILE ROBOT IN BIONIC ASSEMBLY SYSTEM Aleksandar', in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1547–1552. doi: 10.1109/AIM.2005.1511231.

Leitão, P. and Restivo, F. (2006) 'ADACOR : A holonic architecture for agile and adaptive manufacturing control', 57, pp. 121–130. doi: 10.1016/j.compind.2005.05.005.

Lesi, V., Jakovljevic, Z. and Pajic, M. (2016) 'Towards Plug-n-Play numerical control for Reconfigurable Manufacturing Systems', *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. doi: 10.1109/ETFA.2016.7733524.

Li, W., Ying, C. and Qing, X. U. E. (2006) 'Research on Bionic Self-Organizing Theory of Reconfigurable Manufacturing System', *Technology and Innovation Conference*, pp. 1075–1079. doi: 10.1049/cp:20060921.

Ligong, X. U. E., Zude, Z. and Quan, L. I. U. (2008) 'Application of Multi-agent in Intelligent Manufacturing System', pp. 501–504. doi: 10.1109/CSSE.2008.549.

Ma, Z. *et al.* (2017) 'Security Viewpoint in a Reference Architecture Model for Cyber-Physical Production Systems'. doi: 10.1109/EuroSPW.2017.65.

Makinde, O. A., Mpofu, K. and Popoola, A. P. I. (2014) 'Review of the status of reconfigurable manufacturing systems (rms) application in South Africa mining machinery industries', *Procedia CIRP*. Elsevier B.V., 17, pp. 136–141. doi: 10.1016/j.procir.2014.02.035.

Otto, J., Vogel-Heuser, B. and Niggemann, O. (2017) 'Automatic Parameter Estimation for Reusable Software Components of Modular and Reconfigurable Cyber-Physical Production Systems in the Domain of Discrete Manufacturing', *IEEE Transactions on Industrial Informatics*, 3203(c), pp. 1–1. doi: 10.1109/TII.2017.2718729.

Razzaque, M. A. *et al.* (2016) ‘Middleware for internet of things: A survey’, *IEEE Internet of Things Journal*, 3(1), pp. 70–95. doi: 10.1109/JIOT.2015.2498900.

Renjie, H., Feng, L. and Dongbo, P. (2010) ‘Research on OPC UA security’, *Proceedings of the 2010 5th IEEE Conference on Industrial Electronics and Applications, ICIEA 2010*, pp. 1439–1444. doi: 10.1109/ICIEA.2010.5514836.

Ribeiro, L. *et al.* (2013) ‘Multiagent mechatronic systems with simulation on the loop’, *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 3842–3847. doi: 10.1109/smc.2013.656.

Ribeiro, L. and Barata, J. (2011) ‘Self-organization in Automation - The IDEAS’, pp. 2752–2757.

Ribeiro, L. and Barata, J. (2013) ‘Self-organizing multiagent mechatronic systems in perspective’, *IEEE International Conference on Industrial Informatics (INDIN)*, pp. 392–397. doi: 10.1109/INDIN.2013.6622916.

Rocha, A. (2013) ‘An agent based architecture for material handling systems’. Available at: http://run.unl.pt/handle/10362/10504.

Rocha, A. *et al.* (2014) ‘An Agent Based Framework to Support Plug And Produce’, pp. 504–510.

Schäfer, C. (2007) ‘On the Modularity of ANUFACTURING SYSTEMS CHANGED TOWARDS HIGHER’, (I), pp. 20–27.

Shi, Z. *et al.* (2014) ‘The real-time fault diagnosis of optocoupler in switching mode power supply’, *ICRMS 2014 - Proceedings of 2014 10th International Conference on Reliability, Maintainability and Safety: More Reliable Products, More Secure Life*, pp. 263–266. doi: 10.1109/ICRMS.2014.7107183.

Sleman, A. and Moeller, R. (2008) ‘Integration of Wireless Sensor Network Services into other Home and Industrial networks’, *Icictfta*, pp. 1–5. doi: 10.1109/ICTTA.2008.4530160.

Takahashi, H., Nishihara, H. and Koyama, J. (1984) ‘Light-Detectable Function of the Negative Resistance Characteristic Presented by an Optocoupler’, *IEEE Transactions on Electron Devices*, 31(7), pp. 951–954. doi: 10.1109/T-ED.1984.21635.

Tharumarajah, A., Wells, A. and Nemes, L. (1996) ‘Comparison of the bionic, fractal and holonic manufacturing system concepts’, *International Journal of Computer Integrated Manufacturing*, 9(3), pp. 217–226. doi: 10.1080/095119296131670.

Ueda, K., Vaario, J. and Ohkura, K. (1997) ‘Modelling of Biological Manufacturing Systems for Dynamic Reconfiguration’, *CIRP Annals - Manufacturing Technology*, 46(1), pp. 343–346. doi: 10.1016/S0007-8506(07)60839-7.

Unger, S. and Timmermann, D. (2015) ‘DPWSec: Devices profile for Web Services Security’, *2015 IEEE 10th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015*, (April), pp. 7–9. doi: 10.1109/ISSNIP.2015.7106961.

Unified Architecture - OPC Foundation (2017). Available at: https://opcfoundation.org/about/opc-technologies/opc-ua/ (Accessed: 24 February 2017).

Wang, C., Bi, Z. and Xu, L. Da (2014) ‘IoT and cloud computing in automation of assembly modeling systems’, *IEEE Transactions on Industrial Informatics*, 10(2), pp. 1426–1434. doi: 10.1109/TII.2014.2300346.

Wassilew, S. *et al.* (2016) ‘Transformation of the NAMUR MTP to OPC UA to allow Plug

and Produce for Modular Process Automation', *Emerging Technologies and Factory Automation (ETFA)*.

Wilson, J. M. (2014) 'Henry Ford vs. assembly line balancing', *International Journal of Production Research*, 52(3), pp. 757–765. doi: 10.1080/00207543.2013.836616.

Xu, D., Fang, B. and Li, H. (2016) 'An Abstract Communication Service Interface of DPWS on Embedded Device in Industrial Automation', pp. 4695–4699.

ZHANG, Y. *et al.* (2013) 'High Fidelity Virtualization of Cyber-Physical Systems', *International Journal of Modeling, Simulation, and Scientific Computing*, 4(2), p. 1340005. doi: 10.1142/S1793962313400059.

Zhong, C. Le, Zhu, Z. and Huang, R. G. (2016) 'Study on the IOT architecture and gateway technology', *Proceedings - 14th International Symposium on Distributed Computing and Applications for Business, Engineering and Science, DCABES 2015*, pp. 196–199. doi: 10.1109/DCABES.2015.56.