

Novas Tecnologias Aplicadas a Sistemas de Produção

Autor: João Ricardo Cardoso

Orientador: Prof. Dr. Marcelo Götz

Sumário

Agradecimentos	iii
Resumo / Abstract	iv
Lista de Figuras	v
Lista de Abreviaturas e Siglas	vi
1 Introdução	1
1.1 Objetivos	2
1.2 Estrutura do trabalho	2
2 Revisão Bibliográfica	3
2.1 Indústria 4.0	3
2.2 Internet das Coisas	5
2.3 Sistemas Multiagentes	7
2.4 Arquitetura Orientada a Serviços	9
3 Materiais e Métodos	11
3.1 Protocolo de comunicação	11
3.2 Node.js	12
3.3 Programação Web	13
3.4 Sistema embarcado	14
3.5 Software supervisor	15
3.6 Filosofia organizacional	15
4 Desenvolvimento da plataforma	16
4.1 Arquitetura	16
4.2 Virtualização	17
4.3 Serviço de computação em nuvem	19
4.4 Agente de Comunicação	20
4.5 Agentes de produção	21
4.6 Agente de Vendas	22
4.7 Agente supervisor	23
5 Resultados	24
6 Conclusões e Trabalhos Futuros	35
Referências	36
Anexos	39
A.1 General Electric	39
A.2 ThyssenKrupp	40
Apêndices	41

Agradecimentos

Esta é apenas uma tentativa de demonstrar minha gratidão por todos aqueles que estiveram ao meu lado durante todos esses anos de graduação. Este trabalho não seria possível sem o apoio e amor incondicional dos meus pais Osni e Neiva, dos meus irmãos Lázaro e Manuel e da minha namorada, Manu. Não poderia deixar de agradecer aos meus colegas de apartamento pelos momentos de companheirismo Gui, Jeff e Nico e à minha equipe de trabalho pelas lições dadas e colaboração autêntica para esta obra ABC, Moa, Posser e Zani. Por último e não menos importante, ao meu orientador Dr. Marcelo Götz por ter guiado este trabalho através de seu vasto conhecimento científico.

Acredito muito no poder que as circunstâncias têm na vida, e sou muito privilegiado em afirmar que estas pessoas proporcionaram um ambiente confortável, instigante e, sobretudo, inspirador, para que eu pudesse desenvolver este trabalho no extremo da minha lucidez, e finalizar minha trajetória acadêmica com ganas de contribuir para o bem-estar da sociedade através do avanço tecnológico.

“What we usually consider as impossible are simply engineering problems...

There is no law of physics preventing them.”

Michio Kaku

Resumo

O conceito de Indústria 4.0 foi recentemente criado e refere-se a uma suposta quarta revolução industrial, que surge, sobretudo, a partir das recentes mudanças na dinâmica de consumo. Estas mudanças exigem mais diversidade e flexibilidade na lista de produtos oferecidos pelas empresas, o que demanda mais eficiência em seus processos produtivos. Um dos elementos centrais na Indústria 4.0 é o paradigma da Internet das Coisas, que vem sendo intensamente discutido devido, entre outros motivos, à massiva difusão de dispositivos com capacidade de conexão com a internet, e as diversas, e ainda não exploradas, possibilidades de aplicação, inclusive na área industrial. A proposta deste trabalho é desenvolver uma plataforma capaz de integrar equipamentos, operadores, e demandas de produção em um sistema flexível de manufatura. As regras que regem o sistema são baseadas nos conceitos de Internet das Coisas, de Sistemas Multiagentes e de Arquitetura Orientada a Serviços. A implementação da plataforma proposta é desenvolvida com as linguagens de programação HTML e JavaScript e com a ferramenta Node, e irá utilizar o protocolo de comunicação MQTT. Além disso, utiliza-se uma aplicação do *software* supervisorio Eclipse E3 para controlar um PLC Dexter modelo μ Dx100, representando a interação de equipamentos industriais ao sistema. Adicionalmente, para representar a interação de dispositivos embarcados utiliza-se a plataforma de prototipagem Arduino UNO equipada com um *WiFi Shield*. Ao final, discute-se a respeito das funcionalidades obtidas com o sistema, das suas limitações e dos aspectos que podem ser aprimorados em trabalhos futuros.

Palavras-chave: Indústria 4.0, Internet das coisas, Sistemas Multiagentes, Sistemas Orientados à Serviço, MQTT, programação Web.

Abstract

The concept of Industry 4.0 was recently introduced as a reference of the fourth industrial revolution, which has its origin mainly at the recent changes in the dynamics of consumption. These changes demand more efficiency on the production's processes, as well as more diversity and flexibility on the list of products offered by companies. One of the central elements in Industry 4.0 is the Internet of Things paradigm, which has been intensively discussed due to, among other reasons, the mass diffusion of devices capable of connecting to the Internet, and the many, and still not explored, possibilities of application, including in the industrial area. The purpose of this work is to develop a platform capable of integrating devices, operators and production demands into a flexible manufacturing system. The rules governing the system are based on the concepts of Internet of Things, Multi-Agent Systems and Service Oriented Architecture. The platform is developed based on programming languages HTML and JavaScript, uses a tool called Node, and uses the MQTT as communication protocol. In addition, is used an application of the supervisory software Eclipse E3 controlling a Dexter μ Dx100 PLC, representing the interaction of industrial equipment with the system; To represent the interaction of embedded devices it is used the prototyping platform Arduino UNO equipped with a WiFi Shield. At the end, the functionalities obtained with this system, its limitations and the aspects that could be improved in future works are discussed.

Keywords: Industry 4.0, Internet of Things, Multi-Agent Systems, Service Oriented Systems, MQTT, Web programming.

Lista de Figuras

Figura 2.1 – Quatro estágios da revolução industrial.....	3
Figura 2.2 – Número de dispositivos conectados à internet.....	6
Figura 2.3 – Os três paradigmas em Internet das Coisas	7
Figura 2.4 – Negociação de serviços em um Sistema Multiagentes.	8
Figura 2.5 – Estrutura de uma SOA.	10
Figura 3.1 - Diagrama de funcionamento do protocolo MQTT	12
Figura 3.2 – Notação de um objeto na linguagem JavaScrip.	14
Figura 4.1 – Representação da arquitetura projetada.	16
Figura 4.2 – Estrutura de dados que representa um agente de produção.	18
Figura 4.3 – Estrutura de dados que representa um produto.	18
Figura 4.4 – Estrutura de dados que representa uma ordem de produção.	19
Figura 4.5 – Endereçamento de mensagens em um agente de produção.	21
Figura 5.1 – Agente de produção desenvolvido em Arduino UNO.	24
Figura 5.2 – Agente de Produção desenvolvido no Eclipse E3.....	25
Figura 5.3 – Interface do Agente de Produção desenvolvida em JavaScript.	26
Figura 5.4 – Janela de rotina de serviço para operadores.	26
Figura 5.5 – Interface do Agente de Vendas.	27
Figura 5.6 – Janela de estruturas de produtos.....	27
Figura 5.7 – Janela de adição de ordens de produção.	28
Figura 5.8 – Janela de histórico de ordens de produção.....	29
Figura 5.9 – Interface do Agente Supervisorio.....	30
Figura 5.10 – Janela de histórico de valores de um sensor.....	31
Figura 5.11 – Janela de análise de produtividade.	31
Figura 5.12 – Equipamentos utilizados no ensaio de capacidade.....	32
Figura A.1 – Arquitetura da plataforma Predix da General Eletric	39
Figura A.2 – Folheto de apresentação da plataforma MAX	40

Lista de Abreviaturas e Siglas

CoAP – Constrained Application Protocol

CPS – Cyber-Physical Systems

CRM - Customer Relationship Management

ERP – Enterprise Resource Planning

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IoT – Internet of Things

IVACE – Instituto Valenciano de Competitividad Empresarial

JSON – JavaScript Object Notation

MAS – Multi-Agent System

MES - Manufacturing Execution System

MIT – Massachusetts Institute of Technology

MQTT – Message Queue Telemetry Transport

MRP – Manufacturing Resource Planning

M2M – Machine to Machine

OEE – Overall Equipment Effectiveness

OPC – Object Linking and Embedding for Process Control

PLC – Programmable Logic Controller

RAM – Random access memory

RFID – Radio-Frequency Identification

ROM – Read only memory

SOA – Service Oriented Architecture

TCP – Transmission Control Protocol

URI – Uniform Resource Identifier

XML – Extensible Markup Language

XMPP – Extensible Messaging and Presence Protocol

1 Introdução

A Sociedade presenciou recentemente a transição entre a época em que a computação, de forma geral, era executada quase que exclusivamente por máquinas do tipo *desktop*, para a época em que a grande maioria dos computadores existentes no planeta encontra-se disponível em versões que cabem na palma da mão das pessoas. Em projeção a esta constatação é possível inferir que em um futuro próximo haverá microcomputadores integrados com muitas das coisas que fazem parte de nosso cotidiano.

Essas mudanças na dinâmica da utilização de computadores implicam na geração de uma enorme quantidade de dados que devem ser armazenados, processados e apresentados para usuários. É neste ponto que se origina um dos grandes desafios: executar todas essas tarefas de forma eficiente e apresentar os resultados de forma concisa e de fácil compreensão.

Observando as últimas estratégias e o destino dos investimentos das grandes empresas do ramo da computação, a infraestrutura computacional que dará suporte a essas mudanças será fornecida por servidores remotos – computação em nuvem. Segundo uma pesquisa realizada pelo Professor Otávio Próspero Sanchez da Fundação Getúlio Vargas, a adoção da computação em nuvem é vista como uma oportunidade para a redução dos investimentos em tecnologia da informação (SANCHEZ, 2012). Através da infraestrutura em nuvem será possível conectar dispositivos de monitoramento, dispositivos de armazenamento, sistemas de produção, ferramentas analíticas, e plataformas de comando e visualização. Esse modelo consistirá em uma variada gama de serviços disponibilizados pelos agentes virtuais do sistema, de forma similar à que ocorre com o mercado de serviços tradicional, em que cada serviço possui atributos como custo e prazo de entrega (GUBBI, 2013). Estes serviços serão negociados através de comunicação M2M (*machine-to-machine*) e possibilitarão que usuários acessem aplicações sob demanda de qualquer lugar.

De acordo com KAGERMANN et al., 2013, no futuro empresas irão estabelecer redes globais incorporando seu maquinário, almoxarifado e instalação de produção na forma de CPS (*Cyber-Physical Systems*). O conceito de CPS foi definido pela primeira vez em 2006 pelo Dr. James Truchard, CEO da empresa *National Instruments*, baseado na representação virtual de um processo de manufatura. No ambiente de produção, esses CPS compreendem máquinas, estoques e sistemas de produção capazes de trocar informação entre si de forma autônoma, possibilitando que a produção seja controlada de forma independente. Os sistemas centralizados rígidos de controle das fábricas cedem agora seu lugar para inteligência descentralizada, com a comunicação M2M no chão de fábrica (BRAGA, 2014).

Estas novas tecnologias enquadram-se no conceito de Indústria 4.0, que têm o objetivo de trazer mais flexibilidade de fabricação para customização em massa, além de garantir maior qualidade e produtividade para os sistemas de manufatura. No entanto, para atingir esses atributos, conforme indica DAVIES, 2015, as empresas precisarão investir em equipamentos, tecnologias da informação e da comunicação, bem como em ferramentas de análise de dados e integração dos fluxos de dados em toda a cadeia de valor.

Um dos indícios de que empresas no mundo todo, inclusive no Brasil, já estão realizando investimentos em direção ao futuro previsto por essas novas abordagens é a iniciativa da gaúcha Elipse Software, que recentemente deu início a um projeto com objetivo de desenvolver um sistema capaz de fornecer à indústria os benefícios trazidos com os métodos e as tecnologias presentes no conceito de Indústria 4.0. Este projeto tem como um dos seus colaboradores o autor deste trabalho.

Analisando o conteúdo exposto até o momento, é natural pensar que a indústria necessite de aporte teórico, e é nesse contexto que se encaixa um dos objetivos da produção acadêmica, que é o de prover conhecimento para gerar benefícios à sociedade através do avanço tecnológico.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver uma plataforma para gerenciamento parcial da produção de um sistema de manufatura flexível em um ambiente simulado, valendo-se de alguns dos conceitos usualmente presentes nas perspectivas de Indústria 4.0, como Internet das Coisas, Sistemas Multiagentes e Arquitetura Orientada a Serviços. Complementando o escopo deste trabalho está a tarefa de explorar os desafios e os desdobramentos práticos referentes à modelagem e ao desenvolvimento desta plataforma, avaliar as funcionalidades práticas dos conceitos utilizados e identificar as limitações do sistema desenvolvido.

Os objetivos específicos deste trabalho são: desenvolver um *software* que forneça serviço de comunicação entre os demais elementos do sistema; desenvolver um *software* que seja capaz de realizar pedidos de produção; desenvolver um *firmware* para sistemas embarcados que representem os equipamentos de um sistema de manufatura; desenvolver um *software* para que usuários possam visualizar e controlar as funcionalidades dos equipamentos; e, por fim, avaliar o desempenho do sistema, suas limitações, e identificar os desafios durante seu desenvolvimento.

1.2 Estrutura do trabalho

O Capítulo 2 apresenta a revisão bibliográfica sobre os tópicos em que o projeto se baseia, sendo eles: Indústria 4.0; Internet das Coisas; Sistemas Multiagentes; e Sistemas Orientados a Serviços. O Capítulo 3 apresenta os métodos e as ferramentas utilizadas para a implementação da plataforma proposta neste trabalho. O Capítulo 4 descreve a arquitetura projetada e o desenvolvimento de cada um dos componentes do sistema. O Capítulo 5 contém a apresentação da plataforma desenvolvida e os resultados obtidos com o projeto. Por fim, o Capítulo 6 contém uma avaliação em relação ao cumprimento dos objetivos, as conclusões sobre o trabalho desenvolvido e as propostas para trabalhos futuros.

2 Revisão Bibliográfica

2.1 Indústria 4.0

O termo “Indústria 4.0” foi criado por um grupo de trabalho convocado em 2012 pelo Ministério Federal da Educação e Pesquisa da Alemanha para desenvolver recomendações estratégicas visando inovação tecnológica à indústria alemã. O grupo foi liderado pelo Prof. Dr. Henning Kagermann, atual diretor da *National Academy of Science and Engineering* (acatech) e pelo Dr. Siegfried Dais, da empresa Bosch, e contava ainda com diversos pesquisadores de instituições alemãs, como o Instituto Fraunhofer, a Universidade Técnica de Munique e diversos membros de empresas como BMW, ThyssenKrupp, e Festo. O resultado do grupo de trabalho foi um documento intitulado “*Recommendations for implementing the strategic initiative INDUSTRIE 4.0*”. De acordo com DAIS, 2014, Indústria 4.0 é atualmente um dos tópicos mais comentados entre acadêmicos da Alemanha.

Poderosos microcontroladores autônomos (sistemas embarcados) vêm sendo crescentemente conectados entre si e com a internet. Este é o resultado da convergência entre o mundo físico e o mundo virtual na forma de CPS. No território de sistemas de manufatura, essa evolução tecnológica pode ser descrita como o quarto estágio da industrialização, ou indústria 4.0. A Figura 2.1 ilustra as quatro etapas da industrialização segundo KAGERMANN et al., 2013.

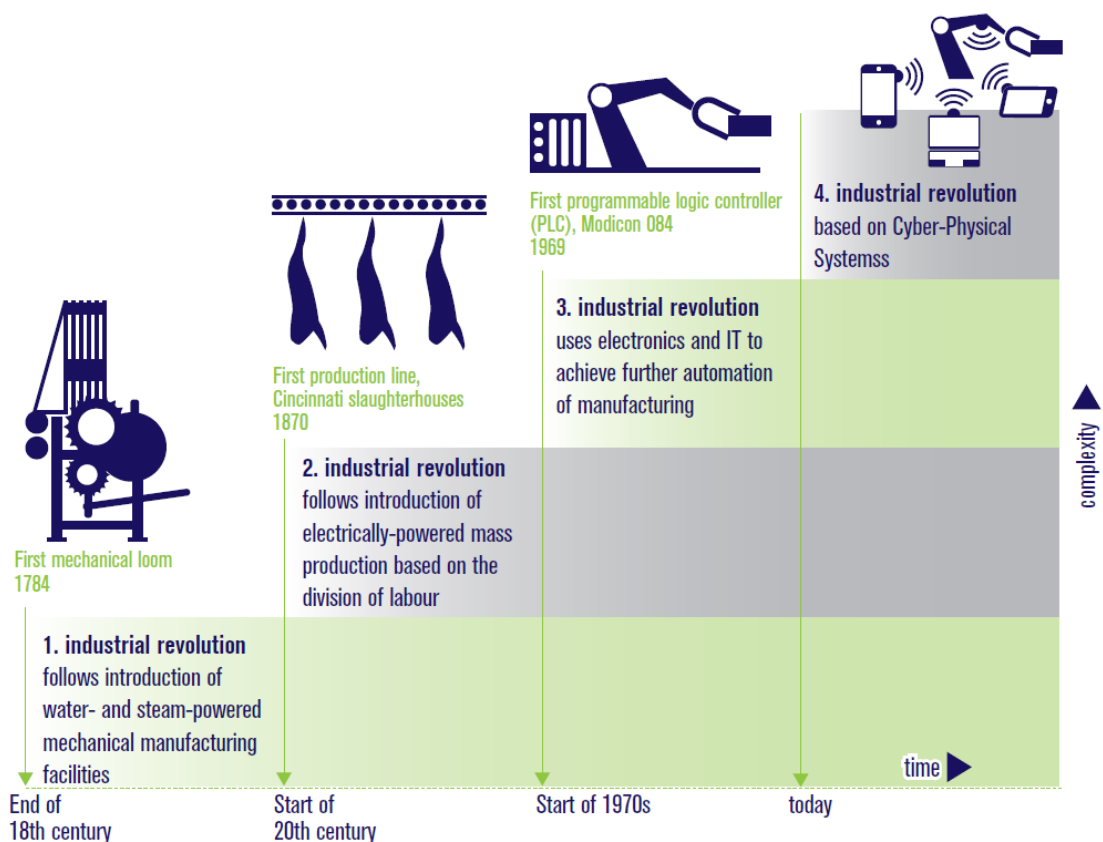


Figura 2.1 – Quatro estágios da revolução industrial – fonte: KAGERMANN et al., 2013.

Conforme HERMANN et al., 2015, Indústria 4.0 é um termo coletivo para tecnologias e conceitos para organizações de cadeia de valor. Esse mesmo trabalho cita os quatro elementos que se repetem com maior frequência nas definições de Indústria 4.0 encontradas atualmente na literatura, são eles: *Cyber-Physical Systems*; Internet das Coisas; Internet dos Serviços; e Fábricas Inteligentes.

Nas fábricas inteligentes, de estrutura modular, CPS monitoram processos físicos e realizam decisões descentralizadas. Via Internet das coisas, CPS comunicam-se e cooperam uns com os outros e com seres humanos em tempo real. Através de sua arquitetura orientada a serviços, processos organizacionais e processos de produção são oferecidos e utilizados por participantes da cadeia de valor.

As fábricas inteligentes constituem uma abordagem completamente nova para sistemas de produção, onde produtos inteligentes são universalmente identificáveis, e podem ser acessados a qualquer momento, assim como seu próprio histórico de processos, seu estado atual e as rotas alternativas para atingir o estado desejado. Os sistemas de manufatura embarcados são conectados verticalmente com os processos de produção dentro das fábricas e das empresas, e podem ser gerenciados em tempo real, desde a realização do pedido à saída do produto na área de logística (KAGERMANN et al., 2013).

De acordo com HERMANN et al., 2015, são seis os princípios de projeto na Indústria 4.0, os quais representam os atributos que se deseja adquirir ao adotar tal filosofia. Esses princípios orientam as empresas a identificar e implementar os cenários previstos na Indústria 4.0. São eles:

- Interoperabilidade: a habilidade dos CPS (suporte de peças, estações de montagem e produtos), dos humanos e das Fábricas Inteligentes de se conectarem e se comunicarem entre si através da Internet das Coisas e da Computação em Nuvem.
- Virtualização: uma cópia virtual das Fábricas Inteligentes é criada por sensores de dados interconectados (que monitoram processos físicos) com modelos de plantas virtuais e modelos de simulação.
- Descentralização: a habilidade dos CPS das Fábricas Inteligentes de tomarem decisões sem intervenção humana.
- Capacidade em tempo real: a capacidade de coletar e analisar dados e entregar conhecimento derivado dessas análises durante a operação.
- Orientação a Serviço: oferecimento dos serviços (dos CPS, humanos ou das Indústrias Inteligentes) através da computação em nuvem.
- Modularidade: adaptação flexível das Fábricas Inteligentes para requisitos mutáveis através da reposição ou expansão de módulos individuais.

Um caso claro de aplicação da exploração de dados com o objetivo de melhorar internamente um processo produtivo seria a manutenção preditiva de uma linha de produção. Nesse caso os dados seriam coletados de distintos sensores para monitorar determinados atributos dos equipamentos, especialmente aqueles com alta carga de

funcionamento. Isso possibilitaria a aprendizagem dos padrões que levam os equipamentos a estados conhecidos e fazer com que se disparassem alarmes com suficiente antecedência para aplicar uma ação de manutenção planejada, no lugar de aplicar uma solução reativa. Dessa forma seria possível aumentar a vida útil dos componentes dos equipamentos, ajustar o estoque de peças de reposição ao estritamente necessário e reduzir de forma significativa a porcentagem de paradas não planejadas na linha de produção. Com isso é possível migrar da abordagem estraga/conserta para a abordagem prediz/previne.

O SAIN4 é um projeto financiado pelo *Instituto Valenciano de Competitividad Empresarial* e a União Européia através do *Fondo Europeo de Desarrollo Regional*. O documento intitulado “*Informe sobre el Estado del Arte de la Industria 4.0*” (IVACE, 2016), mostra exemplos práticos de aplicação de alguns conceitos da Indústria 4.0 em todos os níveis, incluindo também o relativo às pessoas como um dos eixos principais da indústria do futuro. Alguns destes exemplos de aplicação podem ser visto no anexo deste trabalho.

Na análise realizada por IVACE, 2016, a chave para estes novos modelos de negócios é a inclusão do cliente como parte da cadeia de valor. Dessa forma é estabelecido um ciclo em que a informação de utilização dos produtos realimenta a cadeia produtiva, possibilitando que ambos os grupos - consumidores e produtores - se beneficiem. O consumidor irá melhorar a experiência de usuário, tendo acesso a produtos de baixo custo com maior valor agregado, e o produtor terá informações confiáveis para melhorar o projeto dos produtos existentes, e criar novas soluções para atingir novos mercados. Contudo, antes de implementar qualquer tipo de mudança, especialmente aquelas que envolvem investimentos significativos, é necessário realizar uma análise rigorosa em relação aos benefícios que se deseja obter e às soluções que melhor atendem o cenário previsto. É de extrema importância ter cautela, pois, como menciona DAVIES, 2015, nem todos os observadores estão convencidos do valor que a Indústria 4.0 acrescentará aos meios de produção. Alguns acham que a Indústria 4.0 como um conceito é mal definido e sofre de expectativas exageradas. Outros, porém, acreditam que os produtos totalmente digitalizados e as cadeias de valor ainda são um "sonho".

2.2 Internet das Coisas

O conceito de Internet das Coisas - IoT na sigla em inglês - emergiu dos avanços de várias áreas como a de sistemas embarcados, microeletrônica, comunicação e tecnologia da informação (SANTOS et al., 2015). Esse conceito é considerado como a terceira onda da tecnologia da informação, após a Internet e a rede de comunicação móvel (ZHU et al., 2010).

A expressão Internet das Coisas foi cunhada em 1999 por Kevin Ashton, um britânico que fora um dos fundadores do Auto-ID Center no MIT, laboratório este que desenvolveu projetos de pesquisa pioneiros na área de semântica e identificação para objetos em rede, em especial com tecnologias de identificação por rádio frequência - RFID na sigla em inglês.

De acordo com um estudo realizado pela Gartner (GARTNER, 2013), empresa especializada em pesquisas na área de tecnologia da informação, a Internet das Coisas figura como uma das dez maiores tendências em estratégias tecnológicas. O Cisco *Internet Business Solutions Group* projeta que até 2020 haverá cerca de 50 bilhões de dispositivos conectados à internet (EVANS, 2011), conforme ilustrado na Figura 2.2.

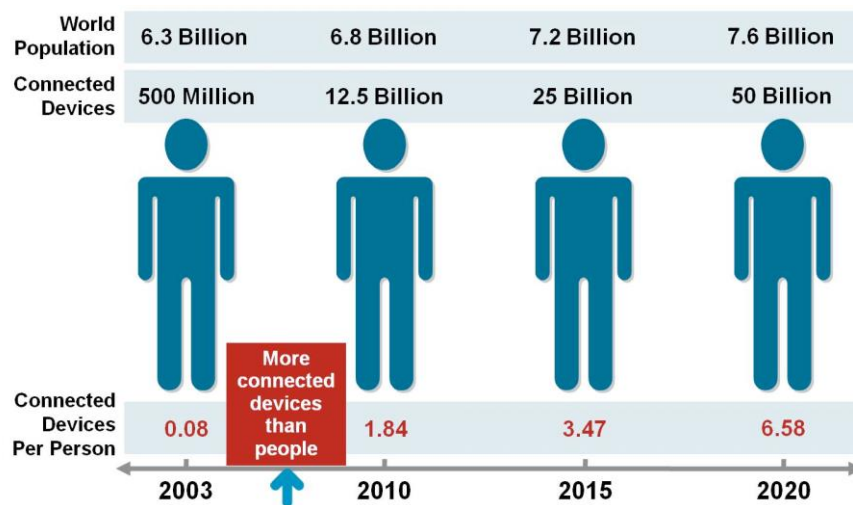


Figura 2.2 – Número de dispositivos conectados à internet – fonte: EVANS, 2011.

Com o lançamento do protocolo IPv6 em 2012 – ou seja, a versão atual do Protocolo de Internet - há suficientes endereços disponíveis para suportar comunicação direta entre dispositivos via internet. Isso significa que, pela primeira vez na história, será possível conectar recursos, informação, objetos e pessoas, criando, assim, a Internet das Coisas. Esse fenômeno também será sentido fortemente pela indústria (KAGERMANN, 2013).

É interessante notar que qualquer entidade do mundo real pode ser representada no domínio da Internet das Coisas, contanto que seja capaz de desempenhar alguma reação à estímulos, transmitir algum tipo de informação ou possuir algum atributo que possa ser medido e reportado através de um dispositivo conectado à internet. Nesse contexto, um elemento na Internet das Coisas não é necessariamente o dispositivo que se conecta a internet, mas a entidade que este representa. Por exemplo, é possível representar uma frota de automóveis através de seus respectivos dados de localização, utilizando em cada veículo um dispositivo capaz de coletar as coordenadas geográficas do veículo e reportá-las na internet. Do ponto de vista da representação do automóvel, não faz diferença se o dispositivo que executa as tarefas é o computador de bordo, um pequeno sistema embarcado instalado no automóvel, ou mesmo o aparelho celular de seus ocupantes - em qualquer um dos três cenários a representação do automóvel compreende apenas o conjunto de dados de sua localização.

Dentre os trabalhos mais citados quando se aborda a definição de IoT encontra-se ATZORI et al., 2010, que apresenta uma definição originária do cruzamento de três paradigmas: orientação para internet; orientação para coisas; e orientação semântica. Assume-se que este tipo de definição surge como necessária devido à natureza interdisciplinar do assunto, todavia a utilidade da Internet das Coisas pode ser desencadeada apenas em um domínio onde os três paradigmas se cruzam (GUBBI et al., 2013). Uma representação dessa abordagem pode ser observada na Figura 2.3.

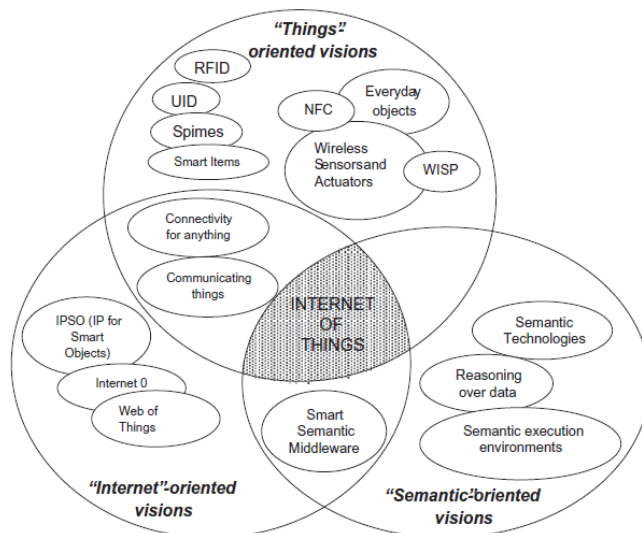


Figura 2.3 – Os três paradigmas em Internet das Coisas – fonte: ATZORI et al., 2010.

Na definição de GUBBI et al., 2013, IoT assume-se como algo mais centrado no utilizador e não restrito a protocolos de comunicação. Dessa forma, para este autor, IoT é “A interconexão de dispositivos possuidores de sensores e atuadores com a habilidade de compartilhar informação de diferentes plataformas através de uma rede unificada, desenvolvendo uma visão operacional comum e assim possibilitar aplicações inovadoras. Isso é atingido pela integração de computação ubíqua, sistemas analíticos e representação de informação, utilizando a computação em nuvem como elo.”

2.3 Sistemas Multiagentes

Sistemas Multiagentes, segundo FERBER, 1999, é uma tecnologia de *software* que é capaz de modelar e implementar comportamentos individuais e sociais em sistemas distribuídos. De acordo com PECHOUCEK et al., 2008, os subsistemas, agora chamados de agentes, são ativos, não apenas podem postar seus serviços e se submetem à solicitações de suas funcionalidades, mas assumem papel proativo para iniciar comunicação com outros agentes, propor negociação e alocar serviços.

A teoria dos agentes busca compreender o que é um agente, suas principais propriedades e como formalizá-los. Em RUSSEL, 2010, encontra-se a seguinte definição: “Um agente é tudo que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por meio de atuadores.”

Implementações de Sistemas Multiagentes possibilitam arranjos na forma de sistemas auto-organizáveis, em que agentes interagem e formam coalizões para gerar os fluxos produtivos, determinam suas restrições, realizam seu próprio monitoramento e tomam decisões de forma autônoma em meio ao processo produtivo. Os agentes adotam esse comportamento a fim de buscar, em conjunto, a melhor solução para atender a demanda de produção. Isto ocorre de forma dinâmica, a cada produto a ser processado, a cada etapa da produção, tornando as decisões e arranjos produtivos mais assertivos.

Esta abordagem visa atender a uma necessidade de rápidas respostas a novas demandas de mercado, onde sistemas com hierarquia normalmente não são capazes de atender (ONORI et al., 2011). A Figura 2.4 apresenta uma abstração da negociação de serviços em um Sistema Multiagentes.

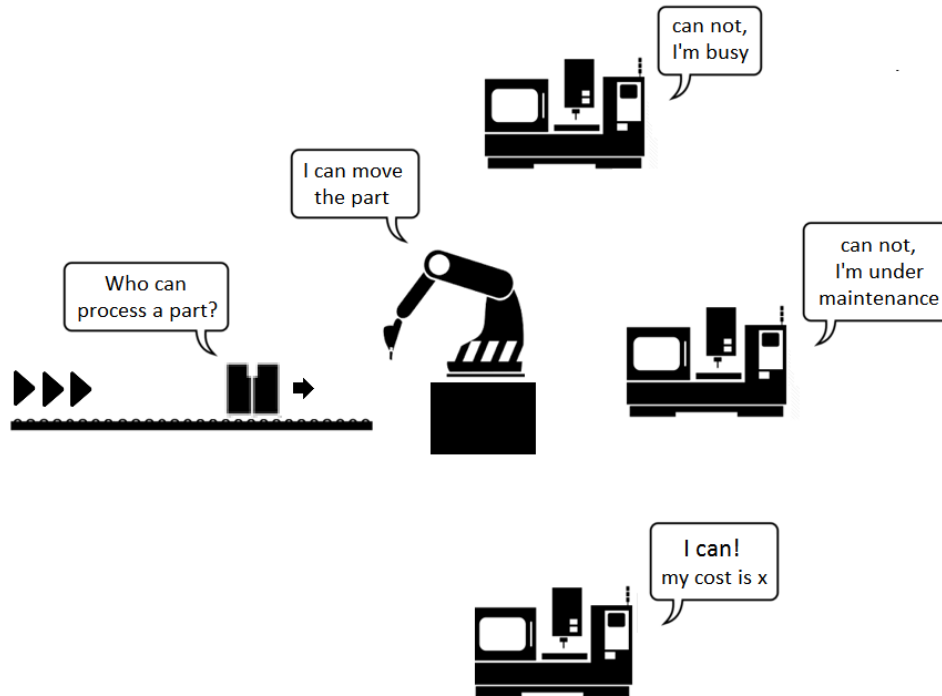


Figura 2.4 – Negociação de serviços em um Sistema Multiagentes.

É interessante notar que esse tipo de sistema configura-se como uma abordagem alternativa em relação às estruturas hierarquizadas tradicionalmente encontradas na indústria, compostas por sistemas do tipo ERP (Enterprise Resource Planning), MRP (Manufacturing Resource Planning) e MES (Manufacturing Execution System), que por vezes não interagem de forma prática. Estas alternativas buscam, entre outras coisas, preencher essa lacuna de conectividade entre as diversas plataformas de gerenciamento implantadas atualmente na indústria.

Sistemas auto-organizados não se limitam a atender fluxos de produção estáticos da forma mais eficiente, mas sim atendem a necessidade do produto e organizam o sistema produtivo em sua forma ótima, sem ter a preocupação de que se trate de um elemento já produzido anteriormente. Em síntese, a auto-organização atende a diversidade de produção com a pretensão de atender, no limite, à necessidade de produtos com lote único.

Conforme citado por PEIXOTO, 2016, os agentes são dotados de habilidade social, ou seja, tem a capacidade de se comunicar com outros agentes. Essa interação é realizada por meio de uma linguagem de comunicação entre agentes. Na literatura, essas linguagens são conhecidas como *Agent Communication Languages* e dentre as instituições que mais se destacam em trabalhos com a teoria dos agentes está a *Foundation for Intelligent Physical Agents* (FIPA). A FIPA foi criada em 1996 por uma associação internacional sem fins lucrativos para desenvolver uma coleção de normas

relativas à tecnologia de agentes de *software*. A iniciativa partiu de um grupo de acadêmicos e organizações industriais, que elaboraram um conjunto de estatutos para orientar a criação de um conjunto de especificações padrão para tecnologias de agentes de *software*. Naquela época os agentes de *software* já estavam muito bem conhecidos na comunidade acadêmica, mas até a data só recebiam atenção limitada de empresas comerciais, além de uma perspectiva exploratória. O consórcio concordou em produzir normas que formariam a base de uma nova indústria, as quais deveriam ser utilizáveis em um vasto número de possíveis aplicações.

A partir da leitura de algumas obras literárias sobre o tópico, é possível concluir que um dos *softwares* mais utilizados para a implementação da lógica de Sistemas Multiagentes a partir das orientações da FIPA é chamado Jade (JADE, 2017), que é um sistema nativo do ambiente Java de programação. Vale destacar que a plataforma desenvolvida neste trabalho utiliza algumas das funcionalidades presentes no Jade como referência, o qual possui entre outros mecanismos:

- a) Serviço de identificação de agentes: reconhece cada agente que se insere no domínio e o identifica com um nome único, disponibilizando esta lista de nomes para outros agentes que desejarem consultar quais estão atuando na plataforma;
- b) Serviço de páginas amarelas: área em que os agentes podem postar seus serviços e também consultar quais agentes realizam um determinado serviço desejado;
- c) Transporte de mensagens: permite que se interfira na troca de mensagens entre os agentes, servindo como um bom recurso para identificação de problemas na implementação;
- d) Serviço de análise da comunicação: é um verificador da troca de mensagens entre os agentes que estão na plataforma;
- e) Biblioteca de protocolos de interação FIPA: disponibiliza classes com requisitos de interação, facilitando a implementação dos agentes.

2.4 Arquitetura Orientada a Serviços

A pergunta “O que é Arquitetura Orientada a Serviços?” é respondida em SCOVINE, 2006, da seguinte forma: é uma arquitetura que permite que novas aplicações sejam criadas a partir da combinação de funcionalidades chamadas de serviços. Esses serviços podem ser novos desenvolvimentos ou até serem criados a partir da exposição de funcionalidades de aplicações existentes. Uma vez criado, um serviço pode ser reutilizado ou recombinado na construção de diferentes sistemas. A cada novo sistema implementado, o número de componentes disponíveis tende a aumentar. Com a reutilização dos serviços, o esforço para desenvolver os próximos sistemas diminui. Uma boa analogia são os blocos de montar. É possível combinar blocos de diferentes formas e cores para criar castelos, carros, barcos, entre outros. Os serviços comparam-se aos blocos e as aplicações às combinações geradas ao juntá-los. Assim como os mesmos blocos criam as mais variadas formas, os mesmos serviços criam as mais diversas aplicações (PEIXOTO, 2016).

Arquitetura Orientada a Serviços – SOA na sigla em inglês - é um conceito em que um sistema de manufatura pode ser visto como um conjunto de sistemas que dispõem serviços que se complementam para formar um processo produtivo (MENDES et al., 2008). Cada sistema dentro de uma SOA presta serviços independentemente dos demais e, ao serem conectados em conjunto, dispõem seus serviços uns aos outros (PEIXOTO, 2016). A Figura 2.5 apresenta um dos princípios de SOA, em que há uma publicação e uma oferta de serviço, bem como uma solicitação deste mesmo serviço.

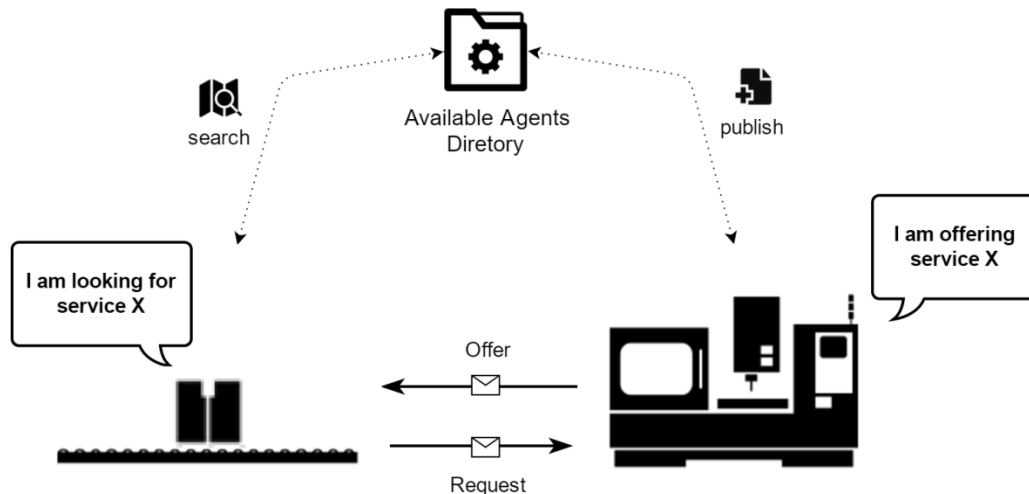


Figura 2.5 – Estrutura de uma SOA.

Um elemento bastante importante em sistemas SOA é o mecanismo de descoberta de serviços (*Available Agents Directory*) cuja representação pode ser observada na Figura 2.5. Existem diversos métodos para implementar este mecanismo, dos quais se destaca o Serviço de Páginas Amarelas, que permite aos agentes registrarem-se e publicarem as descrições de um ou mais serviços. Dessa forma os outros agentes do sistema podem facilmente descobri-los e os requisitar, dando assim continuidade ao fluxo de produção de forma autônoma, de acordo com informações coletadas em tempo real.

Dessa forma, em uma SOA cada subsistema é autônomo e o gerenciamento do fluxo de processo só depende da disponibilidade dos serviços, podendo-se agregar ou retirar equipamentos em função da necessidade de funcionalidades. Porém, o sincronismo deste gerenciamento não é trivial. MENDES et al., 2008, prevê um grande desafio ao descrever os processos que regulamentam o comportamento do sistema, sincronizar e coordenar a execução dos serviços oferecidos pelas entidades distribuídas a fim de atingir o comportamento desejado.

3 Materiais e Métodos

3.1 Protocolo de comunicação

Um dos aspectos mais importantes para a integração em rede de uma variada gama de dispositivos é a padronização da comunicação. Nos meios industriais, assim como na *Web*, há uma enorme variedade de protocolos de comunicação, cada um deles projetado para atender os requisitos específicos da aplicação a que se propõe atender.

Em projetos voltados para a internet o protocolo IP tem sido o mais utilizado para endereçamento de máquinas. Entretanto, é necessário definir um protocolo de nível de aplicação para ser utilizado sobre a camada de endereçamento. Entre os diversos protocolos de aplicação existentes destaca-se o HTTP (HyperText Transfer Protocol), o CoAP (Constrained Application Protocol), o XMPP (Extensible Messaging and Presence Protocol) e o MQTT (Message Queue Telemetry Transport), como protocolos que atendem especialmente as necessidades de aplicações via internet.

O HTTP é o protocolo de aplicação mais utilizado na internet. Possui uma vasta lista de componentes e um poderoso sistema de autenticação, possibilitando assim uma enorme quantidade de aplicações. Todo este aparato fornece a padronização necessária para gerenciar o gigantesco fluxo de informações que trafegam atualmente pela internet. Contudo, sua estrutura básica por vezes é demasiadamente extensa para aplicações envolvendo dispositivos com baixa capacidade computacional.

O CoAP é um protocolo de transferência especializado em aplicações na internet que utilizam dispositivos e redes com capacidade limitadas. Os dispositivos mencionados frequentemente possuem microcontroladores de oito bits com pequenas quantidades de ROM (Read Only Memory) e RAM (Random Access Memory), enquanto que as redes são executadas em dispositivos de baixa potência, possuindo altas taxas de erro e taxa de transferência bastante limitada. Nesse contexto o CoAP foi projetado para aplicações que utilizam comunicação M2M, como *smart energy* e automação de prédios. Este protocolo fornece um modelo de interação pedido/resposta entre os *endpoints* do sistema, oferece suporte à descoberta de serviços e inclui conceitos-chave da *Web*, tais como URIs (Uniform Resource Identifiers) e alguns tipos de transferência de mídia. O CoAP foi concebido para interagir facilmente com a internet através do protocolo HTTP, enquanto mantém atributos importantes como suporte *multicast*, sobrecarga baixa e simplicidade para ambientes restritos (SHELBY, 2016).

O XMPP foi inicialmente desenvolvido como um protocolo para mensagens para conectar pessoas através de mensagens de texto. Esse protocolo utiliza pacotes XML para formatação de texto, e sua principal característica é a utilização de uma estrutura do tipo "*name@domain.com*" como esquema de endereçamento, oferecendo assim um jeito fácil de identificar dispositivos em uma rede IoT.

O MQTT nasceu como um projeto interno da empresa norte-americana IBM (HIVEMQ, 2017), e foi projetado para aplicações de telemetria, ou monitoramento remoto. Normalmente, esse protocolo é utilizado para conectar grandes redes de pequenos dispositivos que serão monitorados ou controlados por um serviço em nuvem (Bauer et al. 2010). A Figura 3.1 mostra um diagrama que representa a dinâmica de troca de mensagens em um ambiente MQTT.

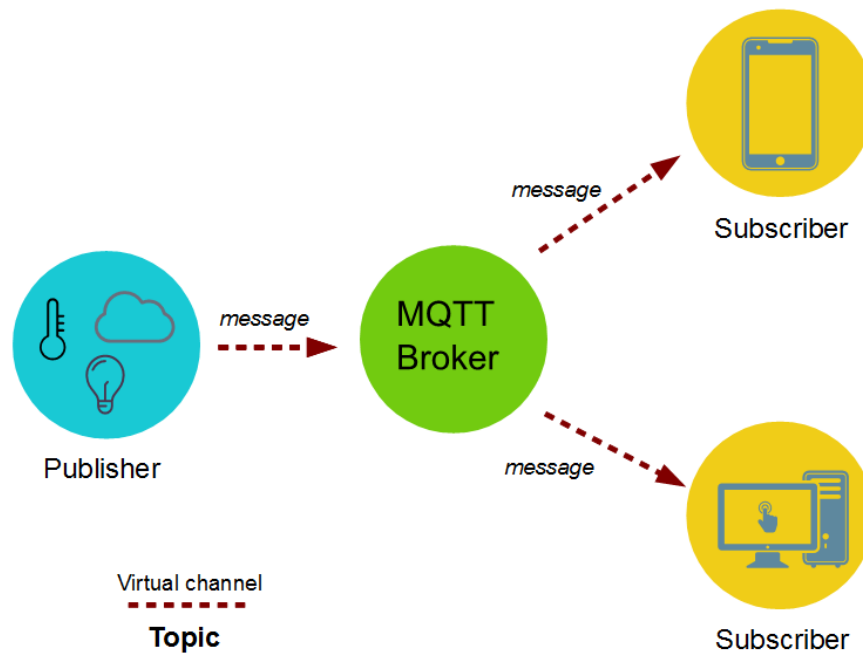


Figura 3.1 - Diagrama de funcionamento do protocolo MQTT – fonte: (SWA, 2017).

Como pode ser observado na Figura 3.1, os dispositivos interessados em enviar alguma informação (*Publishers*) devem publicá-la em um tópico – que funciona como o endereço da mensagem. Os dispositivos interessados em receber alguma informação (*Subscribers*) devem inscrever-se no tópico em que a mesma será publicada. A entidade que realiza o roteamento das mensagens é o servidor (*Broker*), por onde trafega todo o fluxo de dados.

Visto que a integridade dos dados é um ponto essencial para sistemas de telemetria, o mecanismo opera sobre o protocolo TCP, que oferece um mecanismo de troca de mensagens simples e confiável. Dentro do contexto desse protocolo, os *Publishers* são clientes de uma conexão TCP com o *Broker*, o qual mantém, também, conexões com os *Subscribers*.

Através da comparação dos atributos de cada um dos protocolos considerados nesta seção, decidiu-se optar pela utilização dos protocolos HTTP e MQTT, cuja união supre de forma bastante adequada o escopo deste projeto.

3.2 Node.js

O Node é um ambiente de desenvolvimento orientado a eventos assíncronos, que foi projetado para criar aplicações de rede com grande fluxo de informação. Este sistema é executado pelo sistema operacional, e apresenta um ciclo de eventos em *runtime*. O projeto é de código aberto, e sua arquitetura é semelhante a sistemas como o Event Machine da linguagem Ruby (EVENT MACHINE, 2017) e o Twisted da linguagem Python (TWISTED, 2017), que implementam uma máquina virtual que fornece o ambiente para a execução de aplicações, no caso do Node, em linguagem JavaScript.

Em outros sistemas, há sempre uma chamada de bloqueio para iniciar o ciclo de eventos, e normalmente seu comportamento é controlado através de *call-backs*. No ambiente Node não há uma chamada de início para o *loop* de eventos. As aplicações simplesmente entram no *loop* de eventos após a execução do *script* de entrada, e sai do *loop* quando não há mais retornos de chamada para serem executados (NODE, 2017).

A lógica encontrada no Node contrasta com a tradicionalmente encontrada em sistemas operacionais. Ao invés de criar uma nova *thread* a cada conexão (e alocar a memória anexa a ela), cada conexão dispara um evento que é executado na pilha de processos do Node. Neste tipo de comportamento o *loop* de eventos fica escondido do usuário.

A biblioteca escolhida para desenvolver o servidor MQTT chama-se Mosca. Este projeto é de código aberto, foi desenvolvido por Matteo Collina, e pode ser encontrada em seu *site* (COLLINA, 2017). A biblioteca Mosca implementa um *broker* MQTT, ou seja, fornece serviço de comunicação para qualquer dispositivo conectado a este, e permite ao programador desenvolver uma estrutura para gerenciar o fluxo de mensagens através do monitoramento de eventos.

Além disso, será utilizada a biblioteca Express (EXPRESS, 2017), também de código aberto, que implementa um servidor para páginas *Web* (requisições HTTP).

Para armazenar os dados será utilizado o MongoDB (MONGODB, 2017), que é um *software* de código aberto que implementa uma estrutura de banco de dados orientada a documentos. Em adição a esta estrutura será utilizada a biblioteca Mongoose (MONGOOSE, 2017), que funciona como *driver* entre aplicações Node e o MongoDB.

3.3 Programação Web

O HTML (*HyperText Markup Language*) é uma linguagem de marcação utilizada para a construção do *layout* de páginas da *Web*.

JavaScript é uma linguagem de programação interpretada, que é utilizada para executar *scripts* na máquina do cliente, realiza comunicação assíncrona, e altera o conteúdo do documento HTML exibido no navegador do usuário.

Embora haja outras linguagens de programação voltadas para o mesmo propósito, juntos os padrões HTML e JavaScript constituem a grande maioria do conteúdo *Web* executado nos navegadores da atualidade. Por apresentar um ambiente favorável para aplicações na internet, este par de linguagens será utilizado para o desenvolvimento de algumas das peças de *software* que compõe a plataforma proposta neste trabalho.

JSON (JavaScript Object Notation) é a notação para objetos da linguagem JavaScript. Uma estrutura JSON é constituída por um conjunto de propriedades, sendo que cada uma delas é representada de forma que à esquerda encontra-se seu nome e à direita seu conteúdo. Este conteúdo pode ser uma variável simples, do tipo *integer* ou *string*, por exemplo; um *array* de variáveis (delimitado por “ [] ” e separado por “ , ”); ou outro objeto (delimitado por “ { } ” e separado por “ , ”). A Figura 3.2 mostra um exemplo de uma estrutura do tipo JSON.

```
person = {
  firstName: "john",
  lastName: "richard",
  age: 27,
  favoriteFruits: [ "watermelon" , "mango" ],
  friends: [
    {
      firstName: "nick",
    },
    {
      firstName: "jeff",
    },
    ...
  ]
};
```

Figura 3.2 – Notação de um objeto na linguagem JavaScript.

Como pode ser observado na figura 3.2, o objeto *person* contém cinco propriedades, sendo que as duas primeiras (*firstName* e *lastName*) são do tipo *string*; a terceira (*age*) é do tipo *integer*; a quarta (*favoriteFruits*) é um *array* de *strings*; e a quinta (*friends*) é um *array* de outros objetos.

As estruturas JSON serão utilizadas neste trabalho para representar as entidades virtuais dos agentes.

Para fornecer ao navegador a capacidade de comunicação com os demais agentes do sistema, será utilizada a biblioteca MQTT.js (MQTT, 2017), a qual é um projeto de código aberto que implementa o ambiente *client-side* para comunicação MQTT.

3.4 Sistema embarcado

Dispositivos eletrônicos por vezes contêm microcontroladores que possibilitam a execução de tarefas através de um algoritmo programável, sendo estes muitas vezes chamados de sistemas embarcados. Este termo geralmente é utilizado para se referir a sistemas com baixa capacidade computacional, dedicados a tarefas específicas.

Para representar a interação de sistemas embarcados com a plataforma desenvolvida neste trabalho será utilizada a placa de prototipagem Arduino UNO. Esta placa é composta, entre outros componentes, por um microcontrolador Atmel modelo Atmega328p de oito *bits*, uma porta de entrada para alimentação de energia, uma porta USB por onde é inserido o algoritmo desejado através de um computador, além de entradas e saídas (digitais e analógicas) para realizar medições e executar comandos elétricos de baixa potência. Estes componentes fazem do Arduino UNO um sistema simples, porém completo, e adequado para prototipagem de sistemas eletrônicos. Mais informações podem ser encontradas no site oficial do projeto (ARDUINO, 2017).

Será desenvolvido um *firmware* para o Arduino UNO, tornando-o capaz de integrar o sistema na forma de um agente de produção.

3.5 Software supervisório

Softwares supervisórios são largamente utilizados como forma de controlar e monitorar equipamentos e processos em plantas industriais. Em muitos dos casos isso se dá por meio dos PLC (*Programmable Logic Controllers*), os quais são dispositivos eletrônicos robustos que, entre outras funções, coletam sinais elétricos (posteriormente traduzidos para grandezas físicas) e realizam tarefas programáveis através de comandos elétricos. Os *softwares* supervisórios, portanto, oferecem uma interface para que operadores possam controlar e monitorar equipamentos embarcados e outros processos por intermédio de PLC de forma facilitada em um ambiente industrial.

A escolha feita para este trabalho foi o supervisório E3 da empresa Elipse Software, que é um produto desenvolvido no Brasil, mais precisamente em Porto Alegre, e utilizado em todo o mundo. O Elipse E3 destaca-se como uma poderosa ferramenta no ambiente industrial, sendo capaz de interagir com inúmeros equipamentos, como por exemplo, robôs, estações de usinagem e especialmente com PLC. A comunicação com esses equipamentos se dá graças a uma extensa lista de *drives* disponível, possibilitando interação através de diversos protocolos de comunicação voltados à área industrial, como o ModBus e o padrão OPC (Object Linking and Embedding for Process Control). Mais informações sobre o E3 podem ser encontradas no *site* da empresa desenvolvedora (ELIPSE, 2017).

Adicionalmente ao *software* supervisório, utiliza-se um PLC da marca Dexter, modelo μ Dx100, e uma placa didática que contém, entre outros componentes, um arranjo de sensores de temperatura e de lâmpadas conectadas às portas do PLC.

Será desenvolvida uma aplicação em linguagem VBScript no ambiente do Elipse E3, que controla algumas das funcionalidades do PLC Dexter, e implementa nele a lógica de um agente de produção, representando a possibilidade de interação entre equipamentos de um sistema de manufatura real e a plataforma proposta neste trabalho.

3.6 Filosofia organizacional

Adotar uma filosofia ou metodologia organizacional é uma atitude que pode facilitar de forma significativa a execução de um projeto, além de possivelmente economizar tempo dos membros da equipe e assim utilizar os recursos disponíveis de forma eficaz. Ao se tratar de projetos de *software* contemporâneos, uma das abordagens que mais vem ganhando destaque é chamada *Agile*. A metodologia *Agile* apresenta-se como uma alternativa em relações aos métodos tradicionalmente empregados no gerenciamento de projetos, e será utilizada como forma de organizar o processo de desenvolvimento das peças de *software* que irão compor a plataforma proposta neste trabalho.

Na perspectiva *Agile*, divide-se o projeto em componentes com baixa complexibilidade, que são implementados em ciclos focados de desenvolvimento chamados de *sprints*. Dessa forma o projeto pode ser planejado, executado e monitorado a partir dos *sprints*, fazendo com que seu gerenciamento se resuma ao controle de poucas variáveis. Contudo, este é apenas um dos aspectos do método *Agile*, sendo que mais informações podem ser encontradas no *site* da iniciativa (AGILE, 2017).

4 Desenvolvimento da plataforma

4.1 Arquitetura

A plataforma proposta neste trabalho é composta por um conjunto de componentes independentes que juntos formam um ambiente para executar e gerenciar demandas de produção e outras tarefas operacionais em um sistema de manufatura simulado.

A integração dos componentes mencionados no parágrafo anterior é realizada através de uma arquitetura própria, que foi projetada para atender os objetivos específicos deste trabalho. Os componentes que formam o sistema são descritos a seguir:

- Serviço de computação em nuvem (*Cloud*): incumbido de receber, armazenar e disponibilizar os dados da operação;
- Agente de Comunicação (*Communication Agent*): responsável por fornecer a infraestrutura de comunicação MQTT para os demais agentes do sistema;
- Agente de Vendas (*Sales Agent*): responsável por gerar e monitorar as demandas de produção;
- Agente Supervisório (*Supervisory Agent*): responsável por gerenciar a operação dos equipamentos e dos operadores no ambiente do chão de fábrica;
- Arranjo de agentes de produção (*Production Agents*): responsáveis por executar as tarefas de fabricação de forma auto-organizada.

A Figura 4.1 mostra um diagrama da arquitetura projetada para a plataforma.

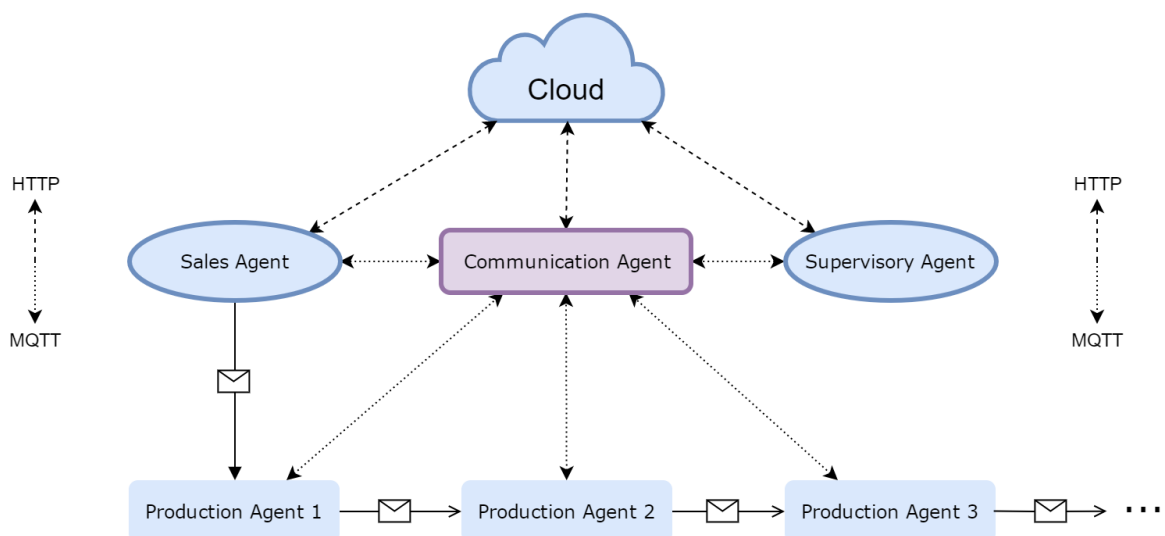


Figura 4.1 – Representação da arquitetura projetada.

Através do protocolo HTTP o Agente de Vendas, Agente de Comunicação e o Agente Supervisório podem interagir com o sistema de computação em nuvem (cuja conexão é simbolizada pela linha tracejada na Figura 4.1), publicando novos dados ou buscando os anteriormente armazenados.

Uma vez que um agente conecta-se ao Agente de Comunicação através do protocolo MQTT (simbolizado pela linha pontilhada na Figura 4.1), este adquire a capacidade de enviar mensagens para os demais agentes do sistema através de tópicos específicos. Os agentes de produção possibilitam a interação do sistema tratado neste trabalho com os equipamentos e os operadores de um sistema de manufatura. Como têm a capacidade de comunicação, estes agentes trocam mensagens entre si, formando coalizões e, assim, executam as ordens de produção inseridas no sistema de forma autônoma.

Outro encargo importante do Agente de Comunicação é o de receber os dados operacionais (mudança de estado dos serviços e de valores dos sensores) dos agentes de produção e repassá-los para o serviço de computação em nuvem, para então serem armazenados.

Na *interface* do Agente Supervisório é possível monitor a lista de agentes de produção conectados ao sistema. Também é possível acessar o histórico dos dados reportados por estes agentes, como mudanças de estado de serviços e valores de sensores, assim como visualizar uma análise de produtividade para cada serviço de cada agente de produção.

Na *interface* do Agente de Vendas é possível acessar o portfólio de produtos, criar novas estruturas de produto, realizar pedidos de produção, e acessar o histórico de fabricação de cada produto em cada ordem de produção efetuada.

O fluxo de uma ordem de produção inicia-se no Agente de Vendas, que a envia para o agente de produção (*Production Agent*) mais adequado. Este agente, ao finalizar sua tarefa, repassa a ordem (simbolizada pelo ícone de envelope na Figura 4.1) para o próximo agente de produção, e assim sucessivamente, até que todas as tarefas de fabricação daquela ordem sejam finalizadas.

4.2 Virtualização

Como já abordado em capítulos anteriores, a virtualização é um processo importante para obter os benefícios propostos pelo conceito de Indústria 4.0. Neste trabalho será realizada a virtualização de um sistema de produção genérico através de estruturas de dados persistentes, ou seja, cada entidade a ser virtualizada será representada por um JSON que a identifica e representa seu estado atual. Esses dados são armazenados em um banco de dados no serviço de computação em nuvem, e atualizados a cada mudança de estado. É importante salientar que as estruturas foram projetadas para conter apenas os dados estritamente necessários para a realização de fluxos de produção simples.

Uma abordagem semelhante a esta é utilizada pela Azure, que é o serviço de computação em nuvem da Microsoft, o qual oferece uma plataforma especializada para aplicações Internet das Coisas chamada IoT Hub. Na nomenclatura da Azure, a cópia virtual dos dispositivos conectados ao sistema é chamada de *device twin* e tem formato JSON. Os detalhes de implementação do projeto IoT Hub da Azure podem ser encontrados na documentação oficial do projeto (AZURE, 2017).

As três estruturas de dados que compõem a virtualização do sistema de produção abordado neste trabalho são: agente de produção; produto; e ordem de produção. A Figura 4.2 mostra a estrutura de dados que representa um agente de produção.

```
agent = {
  agentId: Number
  agentName: String,
  Services: [
    {
      serviceId: Number,
      serviceName: String
      serviceStatus: Number
    },
    ...
    {
      sensorId: Number,
      sensorValue: Number
    },
    ...
  ]
};
```

Figura 4.2 – Estrutura de dados que representa um agente de produção.

Como pode ser observado na Figura 4.2, cada agente de produção é representado por duas propriedades de identificação (*agentId* e *agentName*), e por um conjunto de serviços e de sensores, que por sua vez possuem duas propriedades de identificação (*serviceId* ou *sensorId* e *serviceName* no caso de um serviço), além de uma propriedade que representa o estado atual do serviço ou o valor lido no sensor (*serviceStatus* ou *sensorValue*).

A Figura 4.3 mostra a estrutura de dados que representa um produto.

```
product = {
  productId: Number,
  productName: String,
  Services: [
    {
      serviceId: String,
      serviceName: String
    },
    {
    },
    ...
  ]
};
```

Figura 4.3 – Estrutura de dados que representa um produto.

Como pode ser observado na Figura 4.3, cada produto é representado por duas propriedades de identificação (*productId* e *productName*), e por um conjunto de serviços, que por sua vez são representados cada um por duas propriedades de identificação (*serviceId* e *serviceName*).

Por fim, a Figura 4.4 mostra a estrutura de dados que representa uma ordem de produção.

```
order = {
  orderId: Number,
  orderStatus: String,
  customerId: String,
  customerName: String,
  orderItems: [
    {
      productId: Number,
      productName: String,
      serviceOrderId: Number,
      serviceOrderStatus: Number,
      Services: [
        {
          serviceId: Number,
          serviceName: String
        },
        {
          serviceId: Number,
          serviceName: String
        },
        ...
      ]
    },
    ...
  ]
};
```

Figura 4.4 – Estrutura de dados que representa uma ordem de produção.

Como pode ser observado na Figura 4.4, cada ordem de produção é representada por duas propriedades de identificação da ordem (*orderId* e *orderStatus*), duas variáveis de identificação do cliente (*customerId* e *customerName*) e por um conjunto de ordens serviços. Estas, que por sua vez, são representados por duas propriedades de identificação do produto (*productId* e *productName*), duas variáveis de identificação da ordem de serviço (*serviceOrderId* e *serviceOrderStatus*) e um conjunto de serviços representados cada um por duas propriedades de identificação (*serviceId* e *ServiceName*).

4.3 Serviço de computação em nuvem

O serviço de computação em nuvem foi desenvolvido no ambiente de programação Node utilizando a biblioteca Express que implementa um servidor para páginas *Web* (requisições HTTP). Além disso este serviço utiliza o *software* MongoDB para armazenar os dados, o qual é manipulado através da biblioteca Mongoose.

Esta Estrutura é executada em um computador utilizado estritamente com esse propósito, e os dados disponibilizados por esse servidor podem ser acessados de qualquer navegador conectado à internet. Este formato de implementação é comumente encontrado na arquitetura *backend* de sistemas *Web* contemporâneos.

A função do serviço de computação em nuvem é hospedar as páginas *Web* contendo a lógica do Agente Supervisor, do Agente de Vendas, e do Agente de Produção desenvolvido para operadores. Outra incumbência do serviço de computação em nuvem é a de receber, armazenar e atualizar as entidades virtuais dos agentes de produção que se conectam à plataforma, assim como as novas estruturas de produto criadas por usuários e transmitidas através do Agente de Vendas. Por fim, este serviço disponibiliza todas as informações da operação que foram armazenadas para qualquer agente capaz de realizar requisições HTTP, fornecendo, por exemplo, a lista dos produtos anteriormente salvos no sistema, o conteúdo histórico dos estados dos serviços ou dos sensores de um agente de produção.

4.4 Agente de Comunicação

O Agente de Comunicação foi desenvolvido no ambiente Node, utilizando a biblioteca Mosca, que implementa um *broker* MQTT. Sua função é prover serviço de comunicação para os demais agentes do sistema. Além disso, este agente é incumbido de gerenciar o serviço de Páginas Amarelas, que é a lista de dispositivos disponíveis para operar os serviços necessários para a fabricação dos produtos.

Ao iniciar, o Agente de Comunicação monitora novas conexões para registrar os agentes de produção. Uma vez conectado, os serviços de um agente são disponibilizados nas Páginas Amarelas. Esse procedimento configura-se como o *checkin* dos equipamentos e dos operadores na plataforma.

Uma vez que um agente de produção é registrado, o Agente de Comunicação cria uma entidade virtual deste, que é atualizada a cada mudança de estado e reportada para o serviço de computação em nuvem, onde é armazenada. Cada agente de produção pode oferecer um ou mais serviços, os quais devem encontrar-se em uma das quatro categorias de estados de serviço: “*available*”, “*not available*”, “*working*” e “*waiting*”.

O serviço de Páginas Amarelas registra em uma lista os agentes que estão disponíveis para efetuar as tarefas de fabricação. Quando um agente necessita repassar uma ordem de produção, ele verifica qual a próxima tarefa de fabricação (serviço) a ser realizado na ordem e envia o *serviceld* deste serviço para o serviço de Páginas Amarelas. Este serviço retorna a solicitação com uma mensagem contendo o *agentld* do agente de produção mais adequado para realizar tal serviço. O critério adotado para determinar o agente de produção mais adequado para um serviço é o tempo de registro do serviço. Ou seja, o agente mais adequado será aquele cujo serviço apresenta o *status* “*available*” a mais tempo registrado nas Páginas Amarelas. Esse algoritmo visa evitar que agentes de produção permaneçam com longos períodos ociosos, porém é um método simples e pouco rebuscado. Contudo, está fora do escopo deste trabalho desenvolver algoritmos avançados para agendamento de tarefas, embora seja um assunto de altíssima relevância no contexto em que se encontra. Uma análise altamente rica e aprofundada sobre teorias de agendamento de tarefas em ambientes de produção industriais pode ser encontrada na obra de PINEDO et al., 1995.

Como já mencionado anteriormente, o Agente de Comunicação não comanda a operação, apenas fornece o suporte para comunicação. Portanto, são os agentes de produção que criam os fluxos de produção de forma autônoma, gerando, assim, uma rede auto-organizada.

4.5 Agentes de produção

Os agentes de produção são os componentes que de fato executam os serviços de fabricação, ou seja, representam os equipamentos e os operadores de um ambiente de produção.

Para organizar a comunicação entre agentes com o modelo *publish/subscribe* do protocolo MQTT, foi criado um padrão para os tópicos em que as mensagens trafegam. Neste padrão cada agente envia e recebe mensagens através de tópicos criados especificamente para cada tarefa a ser executada. Dessa forma, o endereço da mensagem (tópico) contém em si a natureza da tarefa, e o conteúdo da mensagem carrega os parâmetros desta tarefa. A Figura 4.5 mostra a lista dos endereços (tópicos) das mensagens que são enviadas e recebidas por um agente de produção.

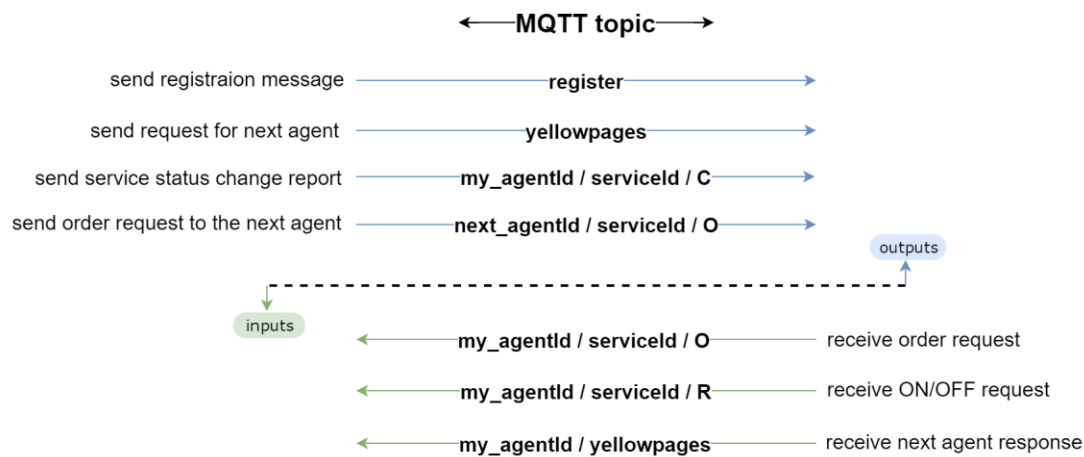


Figura 4.5 – Endereçamento de mensagens em um agente de produção.

O ciclo de vida de um agente de produção inicia-se quando este se registra no sistema, publicando no tópico “*register*” uma mensagem com suas informações de identificação, semelhante ao formato ilustrado na Figura 4.2. Após o registro, o agente passa a monitorar os tópicos representados por flechas verdes na Figura 4.5, e envia mensagens através dos tópicos representados por flechas azuis na Figura 4.5.

Para ilustrar o padrão criado para a troca de mensagens através de tópicos, considere um agente de produção cujo *agentId* é “*machine1*”, que fornece um serviço com *serviceId* igual a “*service1*”, e possui um sensor com *serviceId* igual a “*sensor1*”.

Depois de registrado, o agente passa a monitorar o tópico “*machine1/service1/O*”, em que eventualmente receberá ordens de produção em formato semelhante ao ilustrado na Figura 4.4. Da mesma forma, o agente monitora o tópico “*machine1/service1/R*” em que eventualmente receberá requisições para ativar ou desativar o serviço. Cada mudança de estado do “*service1*” ou “*sensor1*” é reportada para o Agente de Comunicação através do tópico “*machine1/service1/C*” ou “*machine1/sensor1/C*”, a qual será encaminhada para ser armazenada no serviço de computação em nuvem.

Para ilustrar a dinâmica da troca de mensagens durante o ciclo de fabricação de um produto, considere agora uma ordem de produção contendo apenas um produto, e que este produto necessite apenas de dois serviços para ser fabricado, os quais são identificados pelos códigos “service1” e “service2”, e que devem ser executados na ordem em que foram mostrados. Considere também mais um agente de produção cujo *agentId* é “machine2”, que fornece um serviço com *serviceId* igual a “service2”.

No momento em que a “machine1” recebe uma ordem de produção, ela efetua sua tarefa de fabricação (service1) por um determinado tempo, e em seguida busca na estrutura da ordem o *serviceId* do próximo serviço que deve ser executado no produto, no caso do nosso exemplo o “service2”. Obtida esta informação, a “machine1” a envia para o serviço de Páginas Amarelas através do tópico “yellowpages”. Logo depois de recebida a requisição, o serviço de Páginas Amarelas enviará uma mensagem para a “machine1” através do tópico “machine1/yellowpages” contendo o *agentId* do agente disponível, em nosso exemplo o conteúdo desta mensagem seria “machine2”. Por fim, a “machine1” envia a estrutura da ordem de produção para a “machine2” através do tópico “machine2/service2/O”, dando continuidade ao fluxo de produção.

A lógica apresentada nesta seção foi aplicada em três entidades de *software* distintas, as quais são descritas a seguir:

- Arduino UNO: implementação realizada em linguagens C e C++, representando sistemas embarcados;
- Eclipse E3: implementação em linguagem VBScript, representando equipamentos industriais e PLC controlados por *softwares* supervisórios na indústria;
- Página *Web*: implementação realizada em linguagens HTML e JavaScript, representando a interface para operadores.

Ao executar os *softwares* descritos acima, cria-se o ambiente necessário para que diferentes equipamentos e operadores de um sistema de manufatura interajam na forma de agentes de produção na plataforma desenvolvida neste trabalho.

4.6 Agente de Vendas

A implementação do Agente de Vendas foi realizada em uma página *Web*, entre outros motivos, pela facilidade de interação com usuários, com o serviço de computação em nuvem e com os demais agentes do sistema. Este agente foi desenvolvido com o objetivo de criar e gerenciar as demandas de produção, e possui três funcionalidades: consulta ao portfólio de produtos e inserção de novas estruturas de produto; consulta ao histórico das ordens de produção; e adição de ordens de produção no sistema.

Conforme mencionado anteriormente, é no Agente de Vendas que se inicia o ciclo de fabricação dos produtos. Ao ser realizada uma nova ordem de produção por um usuário (cliente), cria-se uma estrutura com as instruções de fabricação de cada produto, semelhante à ilustrada na Figura 4.4. Para encaminhar essa ordem, da mesma forma que os agentes de produção, o Agente de Vendas solicita ao serviço de Páginas Amarelas o endereço do agente mais adequado para a realização da primeira etapa de fabricação, e assim que recebe a resposta encaminha a ordem para este agente.

4.7 Agente supervisorio

O Agente supervisorio também foi implementado em uma página *Web*, sendo desenvolvido com o propósito de monitorar a operação dos agentes de produção. Este agente possui três funcionalidades: visualização da lista de agentes de produção conectados ao sistema – agentes *online* – assim como dos estados de seus serviços e sensores; consulta e *plot* do histórico dos valores dos sensores; consulta e *plot* de uma análise de produtividade básica dos serviços.

Ao conectar-se ao sistema, o Agente Supervisorio realiza uma requisição HTTP para o serviço de computação em nuvem, que o envia uma estrutura de dados contendo as entidades virtuais dos agentes de produção *online*. Com esta informação o Agente Supervisorio tem acesso a uma visão dos estados passados e do estado atual da rede de agentes. Logo em seguida ele passa a monitorar os tópicos referentes aos reportes de mudança de estado dos agentes de produção *online*. Dessa forma, o Agente Supervisorio mantém sua interface sempre atualizada em relação ao estado atual do sistema, possibilitando a visualização em tempo real de tais variáveis, assim como o histórico de operações.

Ao ser solicitada por um usuário uma consulta ao histórico de determinado serviço, o Agente Supervisorio realiza novamente uma requisição HTTP ao serviço de computação em nuvem, que o envia como resposta uma estrutura de dados contendo a informação desejada. Dessa forma, é possível imprimir o gráfico dos valores dos sensores assim como realizar a análise de produtividade e imprimi-la em sua *interface*.

5 Resultados

A plataforma descrita no capítulo anterior foi desenvolvida conforme o planejado, gerando a infraestrutura necessária para realizar o monitoramento de uma parcela de informações em um sistema de manufatura simulado. Neste capítulo serão ilustradas as *interfaces* dos agentes que compõe o sistema, e ao final comenta-se a respeito das funcionalidades adquiridas, das limitações, e dos resultados obtidos através de um ensaio realizado com a plataforma.

O primeiro componente a ser ilustrado é o que representa dispositivos embarcados. O código desenvolvido para implementar a lógica de agente de produção na plataforma Arduino UNO utiliza linguagens de programação C e C++, em conjunto com bibliotecas específicas para este propósito. A conexão deste agente com a internet foi realizada através de um *WiFi Shield*.

O *hardware* utiliza um conjunto de baterias como fonte de energia, e em umas de suas portas digitais foi instalado um *led* para representar os quatro possíveis estados de um serviço hipotético, o qual se comporta da seguinte forma: desligado para representar o estado “*not available*”; ligado continuamente para representar o estado “*available*”; piscando com alta frequência para representar o estado “*working*”; e piscando com baixa frequência para representar o estado “*waiting*”. A Figura 5.1 mostra uma foto do agente de produção desenvolvido em uma plataforma de prototipagem Arduino UNO.

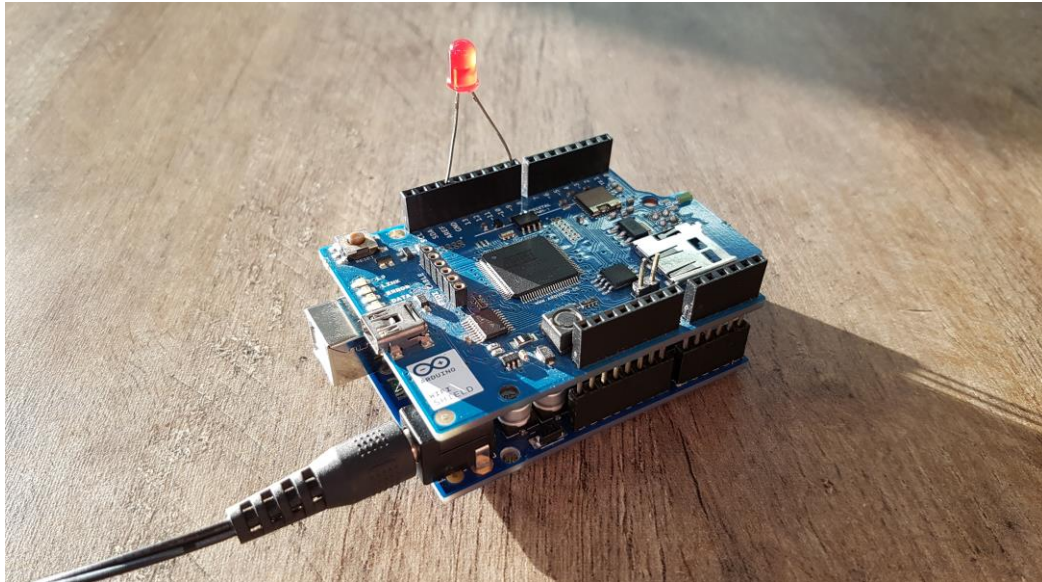


Figura 5.1 – Agente de produção desenvolvido em Arduino UNO.

Ao receber uma ordem de produção, o agente altera o estado de seu serviço para “*working*”, e seu *led* começa a piscar. Após um período programado, a ordem é encaminhada para o próximo agente de produção, e o serviço volta ao estado “*available*”. Caso nenhum agente esteja disponível, o serviço entra em estado “*waiting*” e o agente passa a enviar periodicamente uma requisição ao serviço de Páginas Amarelas, até que haja um agente de produção disponível para executar o próximo serviço.

A implementação do agente de produção no *software* supervisor foi feita de forma a oferecer: uma variável interconectada com uma lâmpada da placa do PLC Dexter representando um serviço; e outra variável interconectada a um sensor de temperatura da placa do PLC. A Figura 5.2 mostra uma imagem da interface do agente de produção desenvolvida com o Eclipse E3 junto ao PLC Dexter conectado à placa de componentes.

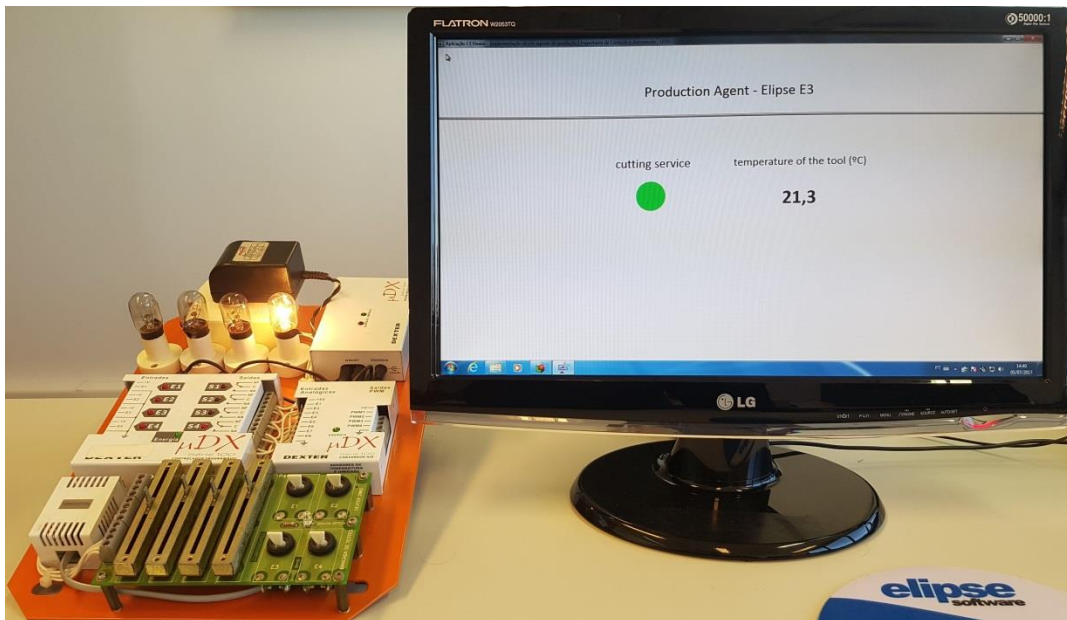


Figura 5.2 – Agente de Produção desenvolvido no Eclipse E3.

Nesta interface o valor do sensor de temperatura é mostrado logo após ser coletado, e o serviço de corte é representado por um círculo, como pode ser observado na Figura 5.2. O círculo funciona como botão de ativação, sendo que a cor vermelha representa que o serviço não está disponível (estado “*not available*”) e que a lâmpada está desligada. Clicando no círculo este torna-se verde, e a lâmpada se acende, demonstrando que o serviço agora está disponível (estado “*available*”). Ao receber uma ordem de produção, inicia-se automaticamente o estado “*working*”, e a lâmpada comporta-se de forma semelhante ao padrão do *led* na implementação com o Arduino UNO. Uma vez iniciada, esta aplicação conecta-se ao Agente de Comunicação, e passa a monitorar seus respectivos tópicos (conforme ilustrado na Figura 4.5) a fim de receber requisições, assim como a reportar as mudanças de *status* do serviço e os valores do sensor.

De forma semelhante, porém programado em linguagens de programação *Web*, desenvolveu-se o agente de produção orientado a operadores, o qual pode ser acessado de qualquer navegador conectado à rede (assim como o Agente de Vendas e Agente Supervisor mostrados a seguir). A Figura 5.3 mostra uma imagem da interface do agente de produção orientado a operadores.

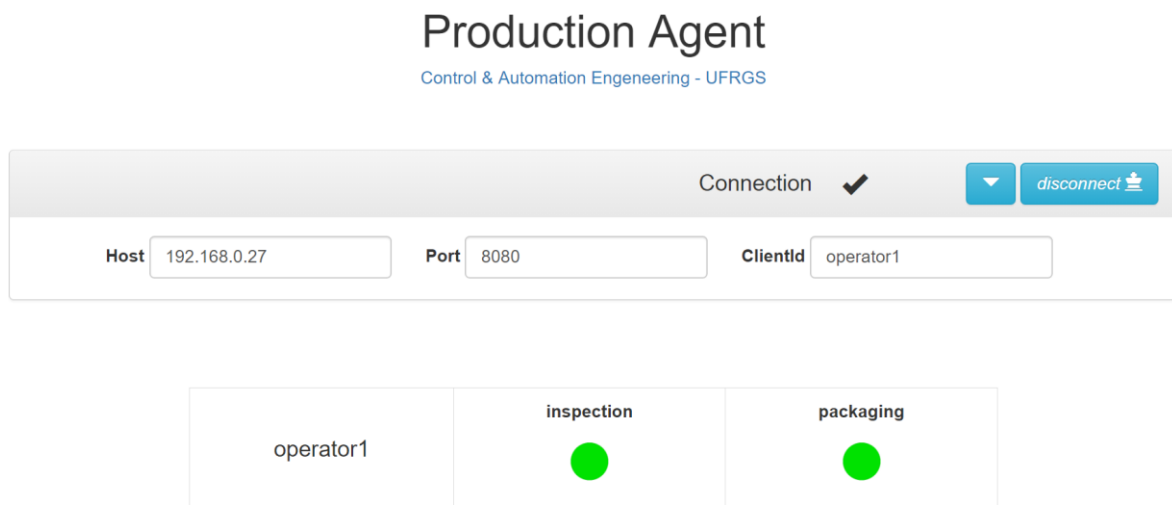


Figura 5.3 – Interface do Agente de Produção desenvolvida em JavaScript.

Nesta implementação foi adicionada uma barra de conexão, exigindo que os operadores informem seu código de identificação para conectar-se ao sistema. Esta mesma funcionalidade foi adicionada no Agente de Vendas e no Agente Supervisório, porém em suas ilustrações a seguir, a área contendo as informações de conexão está escondida - o que pode ser feito clicando-se no botão contendo uma seta que se encontra na barra de conexão. Como pode ser observado na Figura 5.3, o agente de produção ilustrado representa um operador identificado como “*operator1*”, que fornece dois serviços: “*inspection*” e “*packaging*”, ambos apresentando estado “*available*”. Também é possível observar que há um ícone na barra de conexão indicando que o programa está conectado ao sistema, ou seja, está pronto para receber ordens de produção.

Ao receber uma ordem, abre-se uma janela na interface do agente de produção. A Figura 5.4 mostra as duas variações da janela de rotina de serviço do agente de produção.

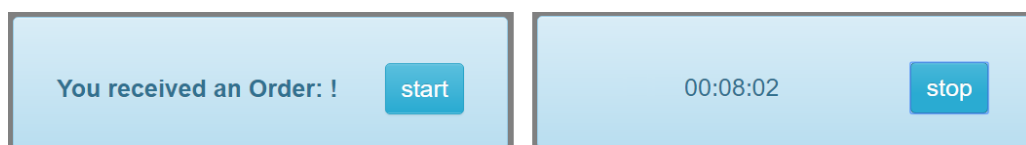


Figura 5.4 – Janela de rotina de serviço para operadores.

Assim que a ordem chega ao agente de produção, a janela de rotina de serviço possui um botão para que o operador possa confirmar o início da operação, conforme o quadro da esquerda na Figura 5.4. Ao clicar no botão “*start*” o estado do serviço altera-se para “*working*”, e a janela de rotina de serviço muda de aparência: a mensagem de aviso é substituída por um mostrador de tempo; e o botão “*start*” tem seu título alterado para “*stop*”, conforme o quadro da direita na Figura 5.4. Ao clicar no botão “*stop*” a janela de rotina de serviço se fecha. Em seguida o agente de produção encaminha a ordem para um próximo agente, e o estado do serviço retorna para “*available*”.

O próximo componente a ser ilustrado é o Agente de Vendas, o qual foi projetado para adicionar e monitorar as demandas de produção, além de gerenciar as estruturas de produtos. A Figura 5.5 mostra uma imagem da interface do Agente de Vendas.

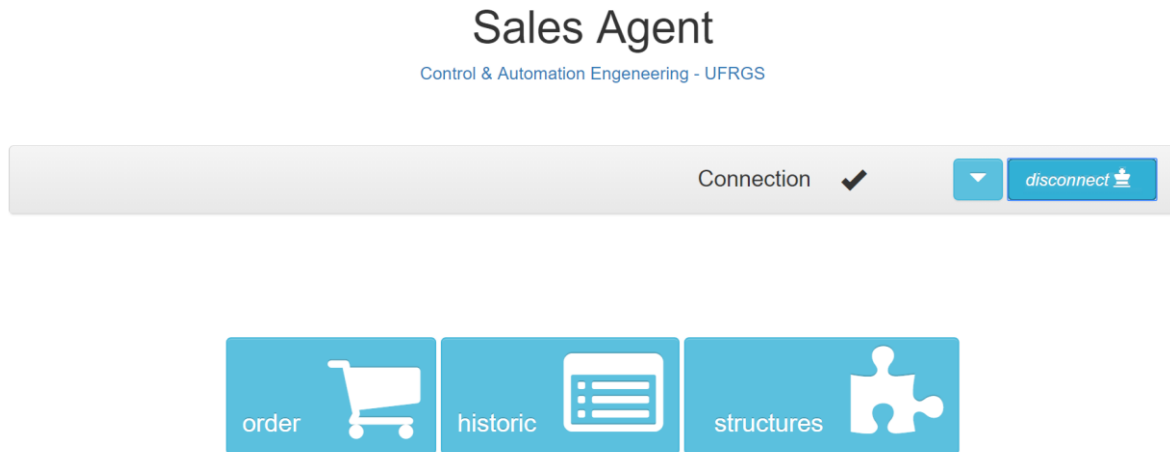


Figura 5.5 – Interface do Agente de Vendas.

Como pode ser observado na Figura 5.5, o Agente de Vendas ilustrado encontra-se conectado ao sistema, e possui três botões para desempenhar suas funcionalidades. Ao clicar no botão “structures” abre-se a janela de estruturas de produtos. A Figura 5.6 mostra uma imagem da janela de estruturas de produtos do Agente de Vendas.

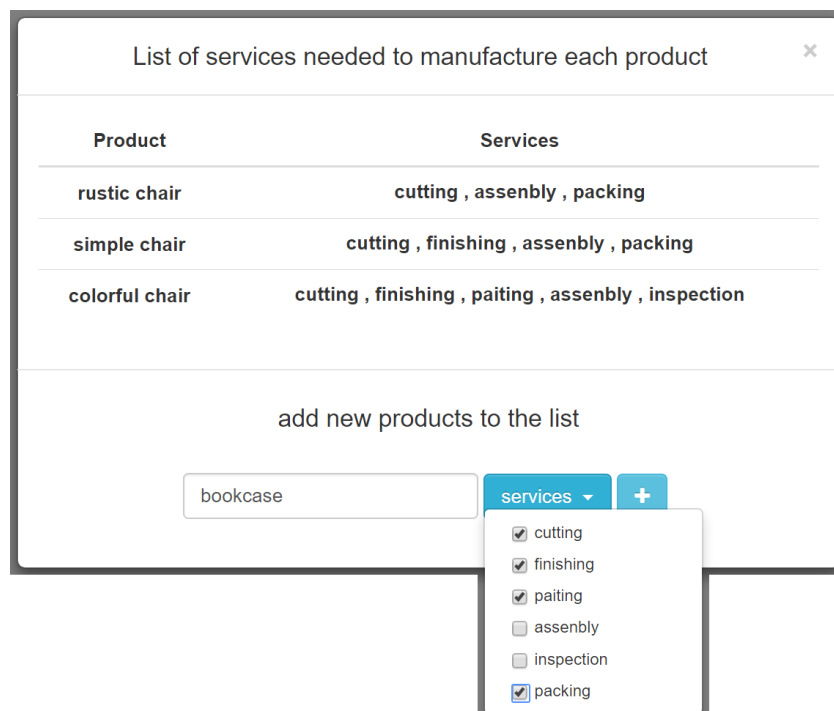
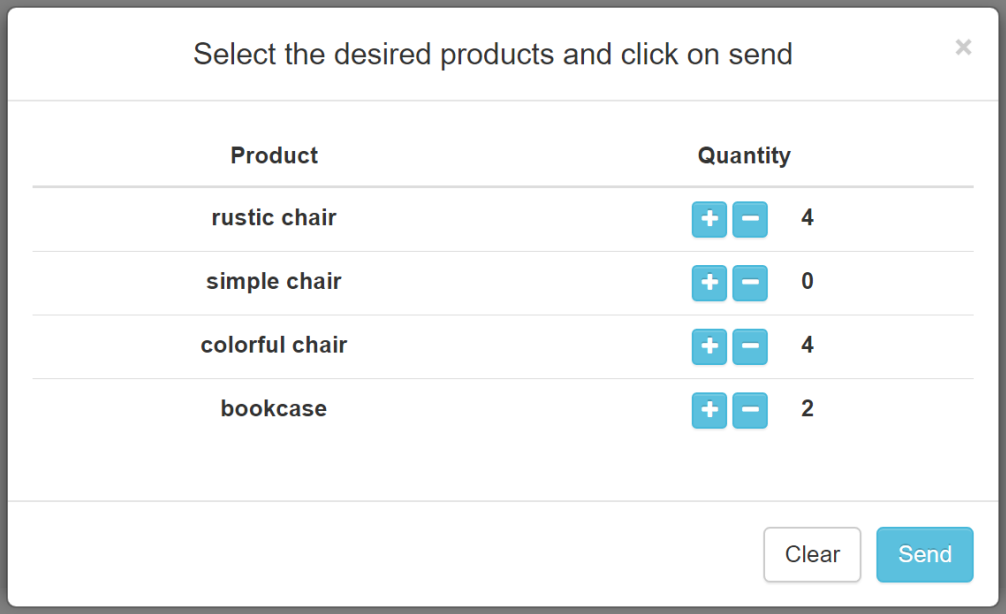


Figura 5.6 – Janela de estruturas de produtos.

Na janela de estruturas de produtos é possível observar a lista das estruturas anteriormente armazenadas, assim como adicionar novas estruturas de produto. Conforme pode ser observado na Figura 5.6, o sistema possui três estruturas de produto já cadastradas: “*rustic chair*”, “*simple chair*” e “*colorful chair*”.

A área inferior da janela de estruturas de produtos é dedicada para adição de novas estruturas. No caso ilustrado na Figura 5.6, ao clicar no botão “+” será adicionado uma nova estrutura de produto chamada “*bookcase*”, que requer os serviços “*cutting*”, “*finishing*”, “*painting*” e “*packing*”. Em seguida abre-se uma caixa de diálogo para confirmar o armazenamento da nova estrutura no serviço de computação em nuvem, e então a janela de estruturas de produtos também é atualizada com a nova estrutura que acaba de ser adicionada.

Ao clicar no botão “*order*” do Agente de Vendas abre-se a janela de pedidos, em que é possível adicionar ordens de produção ao sistema. A Figura 5.7 mostra uma imagem da janela de adição de ordens de produção do Agente de Vendas.



Product	Quantity
rustic chair	4
simple chair	0
colorful chair	4
bookcase	2

Figura 5.7 – Janela de adição de ordens de produção.

Conforme pode ser observado na Figura 5.7, a janela de adição de ordens contém a lista de opções de produtos que podem ser solicitados. Para adicionar produtos em uma ordem basta clicar no botão “+” na linha correspondente ao produto desejado. Para remover os produtos da ordem basta clicar no botão “-” na linha correspondente ao produto alvo. Também é possível limpar o número de produtos na ordem clicando no botão “*clear*”. Assim que a ordem estiver de acordo com o desejado, o usuário deve clicar no botão “*send*” no canto inferior direito da janela. Feito isso o Agente de Vendas reporta a nova ordem de produção para o serviço de computação em nuvem e busca por agentes de produção disponíveis para realizar o primeiro serviço da ordem.

Ao clicar no botão “*historic*” do Agente de Vendas abre-se a janela de histórico de ordens de produção, em que é possível visualizar detalhes da operação. A Figura 5.8 mostra uma imagem da janela de histórico de ordens de produção do Agente de Vendas.

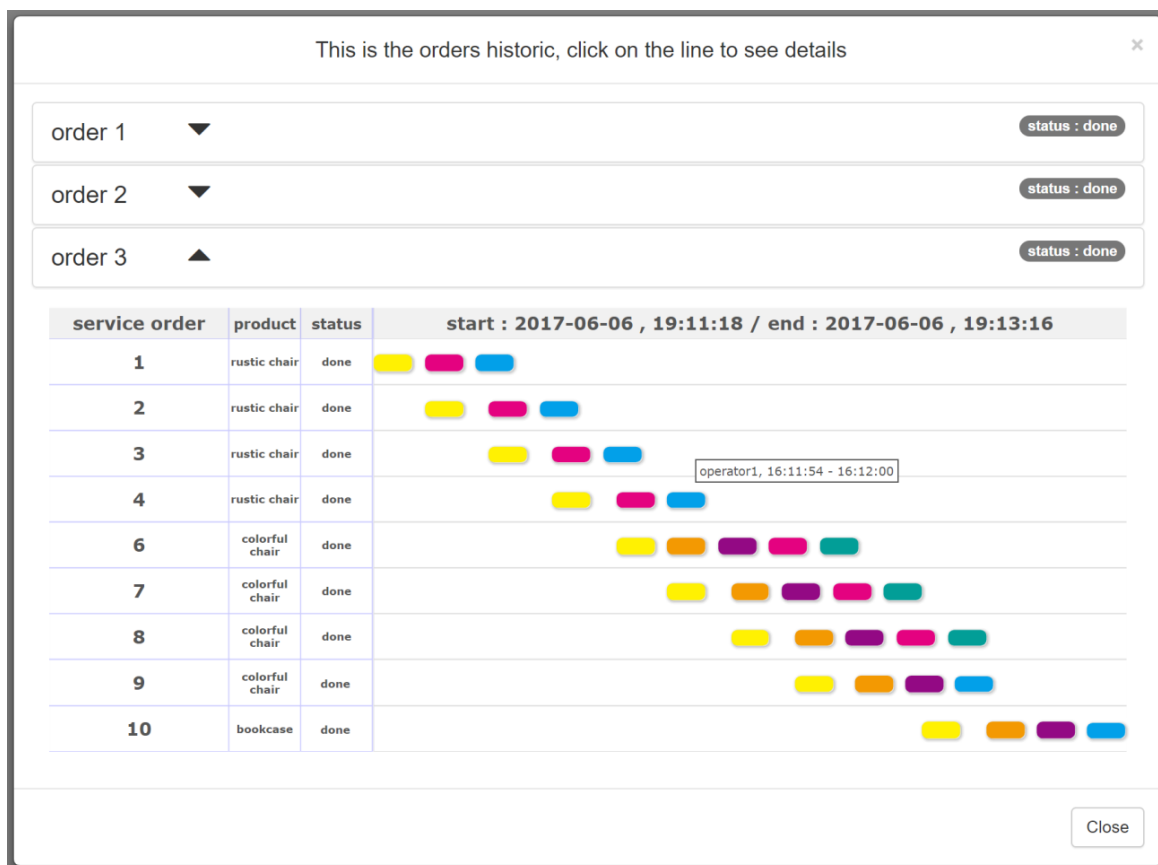


Figura 5.8 – Janela de histórico de ordens de produção.

Como pode ser observado na Figura 5.8, nesta janela há uma barra dedicada para cada ordem. No canto direito da barra é possível observar o *status* da ordem, sendo que no caso ilustrado na Figura 5.8 todas as ordens apresentam *status* “*done*” indicando que já foram concluídas com sucesso.

Ao clicar na barra correspondente a uma ordem de produção abre-se um relatório detalhado da mesma, conforme pode ser observado na Figura 5.8. Neste relatório cada linha representa uma ordem de serviço, e em suas três primeiras colunas é possível visualizar seus respectivos dados de “*serviceOrderId*”, nome do produto e *status* de fabricação correspondentes. As barras coloridas representam as contribuições dos agentes de produção na fabricação do produto correspondente à linha em que se encontra, sendo que cada cor representa um agente de produção distinto, o qual pode ter contribuído para a fabricação de um ou mais produtos naquela ordem. Ao posicionar o *mouse* sobre uma das barras coloridas é apresentada na tela uma legenda contendo a identificação do agente de produção e os horários de início e fim de operação na ordem de serviço correspondente à linha em que se encontra.

A última *interface* a ser ilustrada é a do Agente Supervisório, que segue a mesma linha de *design* das demais *interfaces*, e possui características de conexão semelhantes aos demais agentes de produção. A Figura 5.9 mostra a interface do Agente Supervisório.

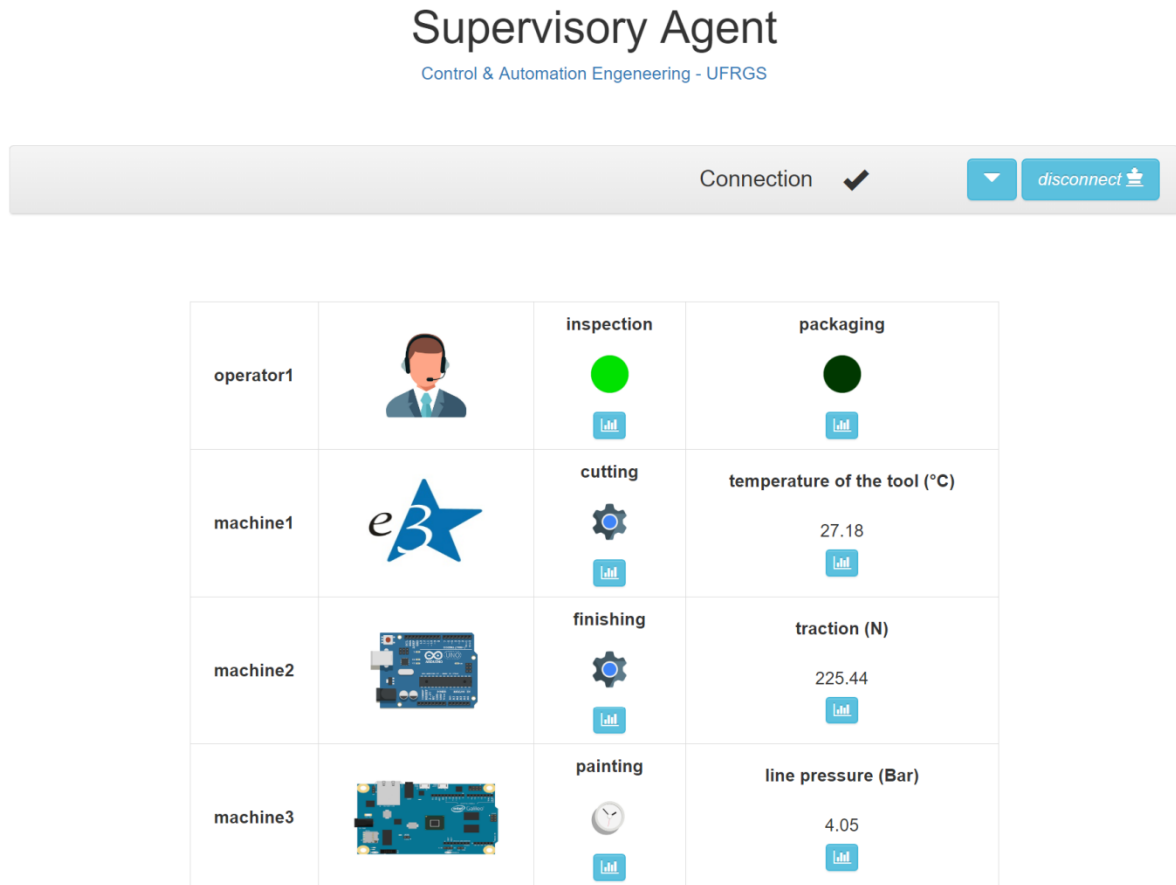


Figura 5.9 – Interface do Agente Supervisório.

A *interface* do Agente Supervisório disponibiliza em seu painel principal a lista de agentes de produção conectados ao sistema, bem como o estado de seus serviços e os valores de seus sensores. Conforme o caso ilustrado na Figura 5.9, o estado atual do sistema apresenta quatro agentes de produção: “*operator1*”, oferecendo dois serviços; “*machine1*”, fornecendo um serviço com status “*working*” (representado pelo ícone de engrenagem), e um sensor; “*machine2*”, com as mesmas características do agente anterior; e, por fim, “*machine3*”, fornecendo o serviço de “*painting*” no estado “*aguardando*” (representado pelo ícone de relógio).

Ao clicar no botão contendo o ícone de gráfico situado logo abaixo do mostrador de um sensor na interface do Agente Supervisório, abre-se a janela de histórico de valores do sensor. A Figura 5.10 mostra uma imagem da janela de histórico de valores de um dos sensores de um agente de produção.

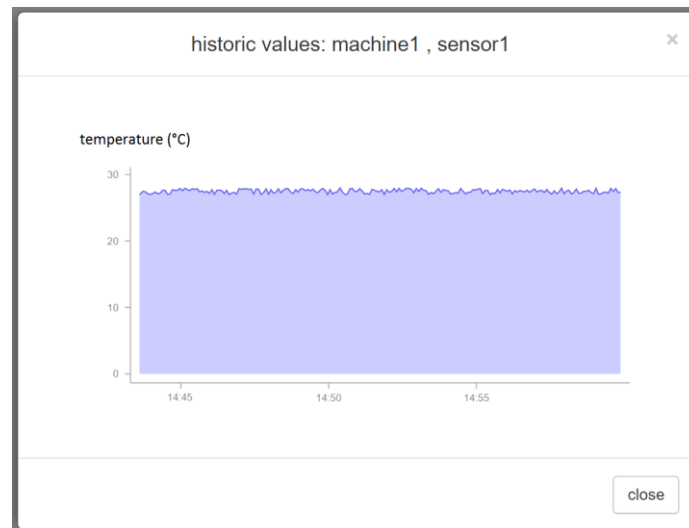


Figura 5.10 – Janela de histórico de valores de um sensor.

Como pode ser observado na Figura 5.10, nesta janela é possível visualizar o gráfico dos valores de um sensor durante seu período de operação.

Ao clicar no botão contendo o ícone de gráfico situado logo abaixo de um serviço na interface do Agente Supervisório, abre-se a janela de análise de produtividade do serviço. A Figura 5.11 mostra uma imagem da janela de visualização da análise de produtividade de um serviço de um dos agentes de produção.

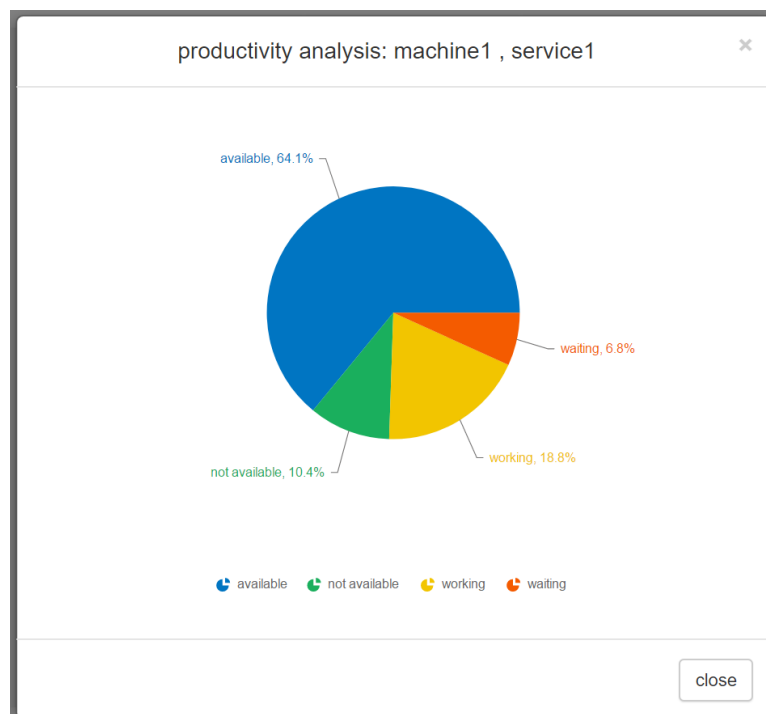


Figura 5.11 – Janela de análise de produtividade.

Como pode ser observado na Figura 5.11, esta janela permite que se visualize a análise de produtividade de um serviço, contendo a porcentagem de tempo que o serviço apresentou em cada uma das quatro categorias de estado (“available”, “not available”, “working” e “waiting”) durante seu período de operação.

Uma vez apresentadas todas as suas *interfaces*, passa-se a discorrer sobre os atributos de desempenho obtidos com a plataforma desenvolvida neste trabalho.

O código contendo o algoritmo do serviço de computação em nuvem foi executado em um servidor de uma empresa que oferece serviços de computação em nuvem localizada nos Estados Unidos. O algoritmo projetado se mostrou bastante adequado à aplicação, tanto no armazenamento dos dados de operação, quanto no processando das requisições dos agentes, executando ambas as tarefas com bastante agilidade. Os serviços oferecidos pelo Agente de Comunicação também se mostraram bastante adequados ao projeto, e não foi constatada nenhuma limitação de caráter crítico em relação às funcionalidades propostas.

A fim de avaliar o funcionamento da lógica de auto-organização dos agentes, determinar limites de capacidade populacional, e, por fim, validar alguns conceitos, realizou-se um ensaio com a plataforma desenvolvida neste trabalho. No ensaio, duzentos agentes de produção foram conectados simultaneamente ao sistema, sendo que: um deles foi executado em um Arduino UNO; quatro outros agentes foram executados individualmente em diferentes computadores utilizando a aplicação do Eclipse E3 junto aos PLC Dexter modelo μ Dx100; e o restante dos agentes de produção foi executado em páginas *Web* contendo a implementação do agente de produção orientado a operadores. Para facilitar o ensaio foi desenvolvida uma versão automática do agente de produção orientado a operadores, ou seja, uma versão que não exige confirmação de início e fim da operação de um serviço – que passa a iniciar automaticamente assim que chegam as ordens, permanecendo no estado “*working*” durante um intervalo fixo de cinco segundos. As páginas *Web* contendo os agentes de produção foram abertas em cinco computadores diferentes e todos os agentes presentes no ensaio foram conectados a uma rede *wireless* local. Vale ressaltar que a escolha por uma rede local é comum, visto que sistemas dessa natureza não podem depender inteiramente da internet para o andamento de sua rotina.

Durante o ensaio foram realizados diversos pedidos, sendo que um deles continha 500 produtos, que foram fabricados (virtualmente) durante um período de aproximadamente oito minutos. A Figura 5.12 mostra uma imagem de alguns dos equipamentos utilizados no ensaio de validação da plataforma.



Figura 5.12 – Equipamentos utilizados no ensaio de capacidade.

O ensaio foi realizado na cidade de Porto Alegre, no centro de treinamento da empresa Elipse Software, que gentilmente cedeu um de seus laboratórios. O ensaio foi executado com aparente sucesso, e com ele foi possível concluir, em primeiro lugar, que o Agente de Comunicação suporta um número razoavelmente grande de conexões simultâneas sem mostrar nenhuma anomalia. Também foi possível concluir que o arranjo de agentes de produção foi capaz de se auto-organizar, e assim instituir os fluxos de produção de forma eficiente e autônoma: Eficiente, pois durante a simulação os agentes de produção não apresentaram intervalos de tempo ocioso significativo - porém foi possível observar lacunas entre as tarefas de fabricação de uma mesma ordem devido aos atrasos de transporte das mensagens; Autônoma, pois em nenhum momento foi necessário indicar qual agente deveria trabalhar em qual produto, ou nenhuma outra informação que predefinissem a rota por onde o produto deveria seguir a fim de ser fabricado, mesmo se tratando de ordens de produção com um número elevado de produtos. Uma das limitações do sistema, constatada com o ensaio, foi que a janela de histórico de ordens do Agente de Vendas perde a clareza ao apresentar relatórios com ordens de produção muito extensas, sendo que uma melhoria estratégica de *layout* neste ponto seria interessante para trabalhos futuros.

Observando suas características, é possível constatar que a plataforma desenvolvida neste trabalho: apresentou interoperabilidade ao conectar diferentes tipos de dispositivos; implementou uma estrutura de virtualização para abstração de entidades; mostrou ser de natureza descentralizada ao deixar no encargo dos agentes a criação dos fluxos de produção; apresentou capacidade em tempo real para a visualização da lista de agentes conectados, assim como outras informações disponibilizadas ao longo da produção; possui arquitetura orientada a serviços; e, por fim, mostrou modularidade ao permitir a rotatividade de agentes para formar coalisões, mesmo com entradas e saídas de agentes de produção durante o ciclo de produção.

Conforme o exposto até o presente, pode-se dizer que este trabalho teve êxito ao aplicar os conceitos gerais da perspectiva de Indústria 4.0 em um sistema de manufatura simulado. A partir da aplicação destes conceitos, o sistema adquiriu certos atributos que, conforme discutido no Capítulo 2, são considerados altamente promissores para organizações que queiram atingir melhores resultados em eficiência, agilidade e flexibilidade em seus sistemas de produção. Ao adquirir estes atributos, empresas passam a usufruir de vantagens competitivas importantes na atual conjectura de consumo de produtos, e ganham a possibilidade de explorar novos modelos de negócios, sendo alguns deles bastante disruptivos em relação aos atualmente adotados na indústria.

Contudo, vale destacar que há muito a se explorar em várias questões relacionadas à plataforma de gerenciamento proposta neste trabalho, sendo que um dos pontos mais importantes é o desenvolvimento de algoritmos para otimização dos fluxos de produção, visando: diminuir o *lead time* dos produtos; diminuir o intervalo entre a fabricação de produtos em uma mesma ordem, e assim diminuir o risco de estoques intermediários; e, por fim, diminuir o tempo ocioso dos equipamentos e dos operadores durante o turno de trabalho. Essas melhorias podem ser parametrizadas por indicadores como o OEE (*Overall Equipment Effectiveness*), o qual é largamente utilizado na indústria para medir a eficiência local de um dado recurso. Contudo, O indicador OEE é apenas uma das ferramentas encontradas no TPM (Total Productive Maintenance), que se caracteriza por uma metodologia de gestão industrial (CHIARADIA, 2004), a qual poderia ser aplicada na plataforma desenvolvida neste trabalho para aumentar a eficiência da produção.

Outra limitação que vale ser destacada é a ausência de aspectos importantes de um sistema de manufatura real, como o controle de insumos para as máquinas e operadores, o controle de estoque de produtos finalizados, e a integração com o setor de engenharia da empresa, que são fundamentais para o gerenciamento efetivo de um sistema dessa natureza. Sistemas de controle como os mencionados são utilizados pela totalidade das organizações industriais atuais, que são de altíssima importância para o bem estar de suas operações, e que por vezes possuem estruturas virtuais gigantescas. Dito isso, configura-se como um grande desafio a tarefa de integrar a plataforma desenvolvida neste trabalho com sistemas do tipo ERP, MRP e MES atualmente instalados nas empresas.

Também foi possível constatar a necessidade de tratar os produtos como um conjunto de outros produtos, ou subprodutos, que também podem ser comercializados. Assim, um produto seria constituído por diversos subprodutos, que podem ser fabricados localmente ou disponibilizados por fornecedores. Esta abordagem é muito utilizada na indústria, e poderia, inclusive, viabilizar a integração de fábricas e fornecedores na plataforma desenvolvida neste trabalho.

Outro tópico importante a ser explorado é conhecido como *data-driven analytics*, que tem por objetivo extrair informações estratégicas em tempo real a partir do grande amontoado de dados que são coletados continuamente das rotinas operacionais dos sistemas de produção. Essas análises têm grande potencial de auxiliar gestores, desde o planejamento até as decisões importantes que devem ser tomadas durante a rotina de trabalho no chão de fábrica. Atualmente já existem empresas especializadas em serviços de *data-driven analytics* para a área industrial, das quais vale destacar a Mitek Analytics localizada em Palo Alto, no Vale do Silício. Mais informações sobre a Mitek podem ser encontradas no site oficial da empresa (MITEK, 2017). Segundo o Dr. Dimitry Gorinevsky, professor de engenharia da Stanford University e CEO da Mitek Analytics, a *Industrial Internet of Things* irá criar valor para organizações a partir das informações obtidas através de *data-driven analytics*, não só no domínio dos sistemas de manufatura, mas nas operações de produção de modo geral (GORINEVSKY, 2016).

6 Conclusões e Trabalhos Futuros

As mudanças na dinâmica de consumo impõem aos meios de produção constantes melhorias de modo que estes continuem atendendo os desejos dos consumidores de forma competitiva. O número de novas tecnologias sendo desenvolvidas, projetadas para as mais diversas aplicações, tem sido muito grande, e embora seja relativamente delicado aplicá-las em certos ambientes conservadores, não há dúvidas de que várias destas podem trazer benefícios reais às cadeias de produção das organizações.

A plataforma desenvolvida neste trabalho atingiu os objetivos de gerenciar uma parcela das etapas de um sistema de produção. Sua estrutura é constituída por um conjunto de *softwares* projetados para diferentes categorias de agentes presentes no sistema. Através de sua interface é possível visualizar algumas informações e relatórios que são atualizados em tempo real, conforme o andamento da produção. Além disso, o sistema apresenta um arranjo auto-organizado de máquinas e operadores, que executam a fabricação de produtos de forma autônoma, a partir de instruções básicas.

O sistema foi capaz de integrar, de forma simples e transparente, diferentes equipamentos, operadores e demandas de produção, em um ambiente de manufatura flexível simulado. De acordo com alguns testes realizados, a rede implementada mostrou ser robusta o bastante para suportar uma quantidade razoavelmente grande de agentes conectados simultaneamente.

Apesar de fornecer informações importantes sobre o andamento da produção, são necessárias mais informações, assim como ferramentas de análise para controlar efetivamente um sistema de manufatura real. Entretanto, o presente trabalho mostra de forma bastante ilustrativa um exemplo de aplicação dos conceitos de Indústria 4.0, e aponta algumas funcionalidades bastante atrativas que se pode obter com tais conceitos.

As maiores dificuldades encontradas ao desenvolver este sistema, sem dúvida, foram relacionadas à modelagem da arquitetura do sistema. Contudo, foi de grande valia a observação realizada em modelos utilizados por empresas do ramo da computação, com destaque para o projeto Azure da Microsoft, que recentemente passou a oferecer serviços especializados em aplicações Internet das Coisas, mostrando-se na vanguarda desta área, e cuja metodologia de virtualização serviu de referência para este projeto.

Em trabalhos futuros seria interessante adicionar à plataforma alguns outros aspectos encontrados na cadeia de valor de sistemas de produção, como o controle dos insumos para as máquinas e operadores, o gerenciamento do estoque de produtos finalizados, a integração com o setor de engenharia, entre outros. Nesse contexto destaca-se também a necessidade de alinhar-se com algumas abordagens tipicamente utilizadas nos ambientes industriais, e explorar os conceitos de *data-driven analytics*, a fim de obter dados estratégicos da produção em tempo real. Essas melhorias ajudariam no amadurecimento deste projeto, tornando-o mais próximo de uma ferramenta viável que possa agregar valor real a organizações industriais.

Referências

ARDUINO; Projeto Arduino. Disponível em <https://www.arduino.cc/>. Acessado em Abril de 2017.

AGILE; Agile Methodology. Disponível em <http://agilemethodology.org/>. Acessado em Março de 2017.

AZURE; Serviços de computação em nuvem da Microsoft. Disponível em <https://docs.microsoft.com/en-us/azure/iot-hub>. Acessado em Junho de 2017.

ATZORI, L.; IERA, A.; MORABITO, G.; The internet of things: A survey. Computer networks, Elsevier. 2010.

BAUER, E.; WILEY, J.; Design for Reliability: Information and Computer-Based Systems. 2010.

BASSI, A.; BAUER, M.; FIEDLER, M.; THORSTEN, K.; VAN KRANENBURG, R.; LANGE, S.; MEISSNER, S.; Enabling Things to Talk. 2013.

BRAGA, N.; Indústria 4.0 – O que é isso? 2014. Disponível em: <http://www.newtoncbraga.com.br/index.php/electronica/52-artigos-diversos/7571-industria-4-0-o-que-e-isso-art1350>. Acessado em Abril 2017.

CHIARADIA, A.; Utilização do indicador de eficiência global de equipamentos na gestão e melhoria contínua dos equipamentos: um estudo de caso da indústria automobilística. 2004.

COLLINA, M.; Biblioteca Mosca. Disponível em: <http://www.mosca.io>. Acessado em Junho de 2017.

DAIS, S.; Industrie 4.0 – Anstoß, Vision, Vorgehen. In: Bauernhansl. 2014.

DAVIES, R.; Industry 4.0 Digitalisation for productivity and growth. 2015.

ELIPSE; Elipse Software. Disponível em: <https://www.elipse.com.br>. Acessado em Junho de 2017.

EVANS, D.; The Internet of Things: How the Next Evolution of the Internet is Changing Everything. 2011.

EVENT MACHINE; Ambiente de desenvolvimento Ruby. Disponível em <https://github.com/eventmachine>. Acessado em Junho de 2017.

EXPRESS; Biblioteca Express. Disponível em <http://expressjs.com>. Acessado em Maio de 2017.

FERBER, J.; Multi-Agent System: An Introduction to Distributed Artificial Intelligence. 1999.

GARTNER; Gartner Identifies the Top 10 Strategic Technology Trends for 2013. Disponível em: <http://www.gartner.com/newsroom/id/2209615>. Acessado em Maio de 2017.

GORINEVSKY, D.; Analytics for Industrial Internet of Things. 2016.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M.; Internet of Things (IoT): A vision, architectural elements, and future directions. Elsevier. 2013.

HERMANN, M.; PENTEK, T.; OTTO, B.; Design Principles for Industrie 4.0 Scenarios: A Literature Review. 2015.

HIVEMQ; MQTT Essential: Part 1 – Introducing MQTT. Disponível em <http://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt>. Acessado em Julho de 2017.

IVACE; Informe sobre el Estado del Arte de la Industria 4.0. 2016.

JADE; Ambiente para desenvolvimento de Sistemas Multiagentes. Disponível em <http://jade.tilab.com/>. Acessado em Junho de 2017.

KAGERMANN, H., WAHLSTER, W.; HELBIG J.; Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group. 2013.

MENDES, J.; LEITÃO, P.; COLOMBO, A.; Service-Oriented Control Architecture for Reconfigurable Production Systems. 2008.

MITEK; Advanced Analytics for the Industrial Internet of Things. Disponível em <http://www.mitekan.com/>. Acessado em Junho de 2017.

MONGODB; Software de banco de dados. Disponível em <https://www.mongodb.com>. Acessado em Junho de 2017.

MONGOOSE; Driver de comunicação. Disponível em <http://mongoosejs.com>. Acessado em Junho de 2017.

MQTT; Biblioteca MQTT.j. Disponível em <https://github.com/mqttjs/MQTT.js>. Acessado em Junho de 2017.

NODE; Ambiente de desenvolvimento JavaScript. Disponível em: <https://nodejs.org/en/about>. Acessado em Maio de 2017.

ONORI, M.; SEMERE, D.; LINDBERG, B.; Evolvable systems: an approach to self-X production. 2011.

PECHOUCEK, M.; MARÍK, V.; Autonomous Agents and Multi-Agent Systems. 2008.

PEIXOTO, J.; Sistema Minimamente Invasivo Baseado em Agentes Aplicado em Controladores Lógicos Programáveis. 2016.

PINEDO, M.; Scheduling: Theory, Algorithms, and Systems. 1995.

RUSSEL, S.; NORVIG, P.; Artificial Intelligence: A Modern Approach. 2010.

SANCHEZ, O.; Antecedentes da Adoção da Computação em Nuvem: Efeitos da Infraestrutura, Investimento e Porte. 2012. Disponível em:

<http://gvpesquisa.fgv.br/publicacoes/gvp/computacao-em-nuvem>. Acessado em Maio de 2017.

SANTOS, B.; SILVA, L.; CELES, C.; BORGES, J.; PERES, B.; VIEIRA, A.; VIEIRA, L.; GOUSSEVSKAIA, O.; LOUREIRO, A.; Internet das Coisas: da Teoria à Prática. 2015.

SCOVINE, C.; Arquitetura Orientada a Serviços. 2006. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcb/entry/soa?lang=en>. Acessado em Abril de 2017.

SWA; MQTT Protocol Tutorial: Step by step guide. Disponível em: <https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html>. Acessado em Junho de 2017.

SHELBY, Z.; HARTKE, K.; BORMANN, C.; The Constrained Application Protocol (CoAP). 2014.

TWISTED; Ambiente de desenvolvimento Phyton. Disponível em <https://twistedmatrix.com/trac/wiki>. Acessado em Junho de 2017.

VERTEN, J.; ThyssenKrupp launches MAX: Maximum efficiency in cities with IoT technologies from Microsoft. 2015. Disponível em: <https://news.microsoft.com/de-de/thyssenkrupp-startet-max-maximale-effizienz-in-staedten-mit-iot-technologien-von-microsoft/#sm.001977ezp17tve3msbd2as8oaqyy6#hQE1kdgO1brCRtZh.97>. Acessado em Abril de 2017.

ZHU, Q.; WANG, R.; CHEN, Q.; LIU, Y.; QIN, W.; IoT gateway: Bridging wireless sensor networks into internet of things. In: IEEE. Embedded and Ubiquitous Computing (EUC), IEEE/IFIP 8th International Conference on. 2010.

Anexos

Anexo A – Estado da Arte

Atualmente já existem algumas ferramentas focadas em aplicar os conceitos da Indústria 4.0. Contudo, é interessante notar que há um histórico de grandes empresas de manufatura que desenvolveram ferramentas para atender seus próprios sistemas de produção. Estas ferramentas, antes de serem comercializadas, foram implementadas em alguma ou todas as fases do processo de produção das empresas que as desenvolveram, indicando que são reais as necessidades por ferramentas como estas. Na obra IVACE, 2016, como já mencionado, constam alguns exemplos de aplicação dos conceitos da Indústria 4.0, os quais são descritos a seguir.

A.1 General Electric

A empresa norte-americana General Electric foi uma das primeiras a utilizar a filosofia da Indústria 4.0 em suas plantas. Recentemente a empresa criou a plataforma Predix, a qual é baseada em computação em nuvem e destina-se a realizar análises em escala industrial. Esta ferramenta possibilita o gerenciamento dos ativos e a otimização das operações, fornecendo uma maneira padrão de conectar máquinas, dados e pessoas. Na Figura A.1 é possível observar uma representação da arquitetura da plataforma Predix.

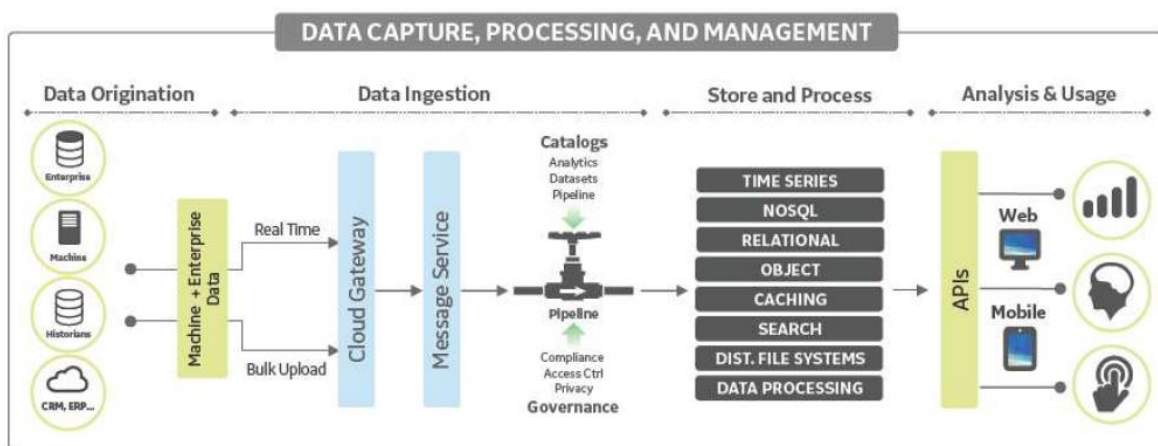


Figura A.1 – Arquitetura da plataforma Predix da General Eletric - fonte: (IVACE, 2016)

A seguir há uma breve descrição das quatro camadas que compõem a arquitetura da plataforma Predix.

1. *Data Origination* – é o processo de captura dos dados, cujos componentes estão diretamente ligados ao processo de produção no chão de fábrica, ou à ferramentas estratégicas e operacionais como sistemas do tipo ERP, CRM e MES.
2. *Data Ingestion* – é o processo de tratamento dos dados de entrada, que pode ser feito em tempo real ou baseado em dados históricos.
3. *Store and Process* – é o processo de armazenamento e pré-processamento dos dados, o qual utiliza diferentes ferramentas analíticas e de armazenamento.
4. *Analysis & Usage* – é o processo de exposição aos usuários dos resultados das análises realizadas com os dados.

A.2 ThyssenKrupp

A empresa alemã ThyssenKrupp apresentou recentemente o primeiro sistema de manutenção preditiva de elevadores na indústria. O projeto chamado MAX possui como principais objetivos aumentar a disponibilidade de elevadores e reduzir o tempo de resolução de problemas através de diagnósticos em tempo real. O sistema prevê eventos de falha antes que eles ocorram. Isso evita paradas não programadas de elevadores, reduzindo a necessidade de substituição de alguns componentes antes do final de seu ciclo de vida. As razões que levaram ao desenvolvimento do MAX são as grandes diferenças entre procedimentos tradicionais de manutenção de elevadores e as necessidades diárias do ambiente urbano moderno. O monitoramento remoto de elevadores surgiu no final de 1980, porém mesmo que estes sistemas alertem a empresa quando os elevadores param de funcionar isso não reduz o número de paradas. É justamente esse o problema que a plataforma MAX se propõe a resolver, utilizando computação em nuvem com o suporte da Microsoft Azure. A Figura A.2 apresenta uma imagem do folheto de apresentação da plataforma MAX da ThyssenKrupp.

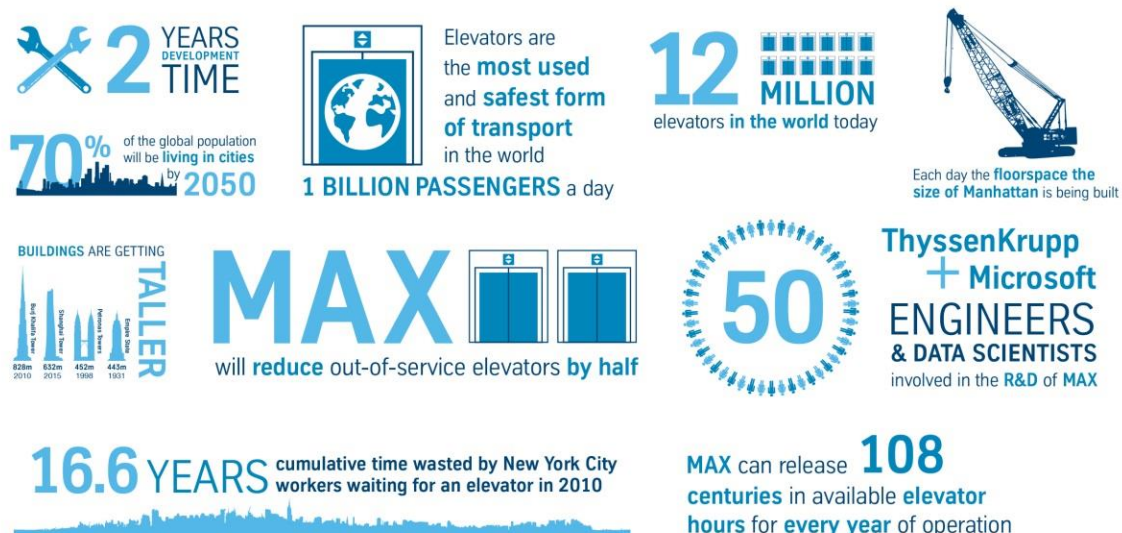


Figura A.2 – Folheto de apresentação da plataforma MAX – fonte: (VERTEN, 2015).

Este sistema baseia-se em relatórios gerados a partir dos dados do monitoramento remoto dos elevadores. Estes dados são enviados para a plataforma MAX e analisados por algoritmos do tipo *machine learning*. A ideia é analisar e estimar a duração de componentes dos elevadores para que os operadores possam prever falhas, podendo assim reduzir o número de avarias e possibilitar o agendamento estratégico de paradas para manutenção dos elevadores.

Apêndices

Apêndice A – Ilustração da estrutura de código do serviço de computação em nuvem.

```

1  /*****
2      Cloud service
3
4      author: João Ricardo Cardoso
5      *****/
6
7  //////////////////////////////////////
8  // Express requests and initiation
9  //////////////////////////////////////
10 + var express = require('express') ...
18  //////////////////////////////////////
19  // MongoDB requests and initiation
20  //////////////////////////////////////
21 + var mongoose = require('mongoose') ...
24  //////////////////////////////////////
25  // API infrastructure
26  //////////////////////////////////////
27
28  // http server configuration
29  app.use(express.static('www'),bodyParser.json());
30  // provides the historical data of agents
31 + app.get('/graphics', function (req, res) { ...
47  // provides the products list
48 + app.get('/products', function (req, res) { ...
63  // provides the service orders list
64 + app.get('/orders', function (req, res) { ...
79  // provides the production orders list
80 + app.get('/order', function (req, res) { ...
96  // http server inicialization
97 + server.listen(80, function (err) { ...
102  //////////////////////////////////////
103  // data base operations
104  //////////////////////////////////////
105
106  // MongoDB connection through mongoose
107 + mongoose.connect('mongodb://localhost/devices', function(err, res) { ...
117  // modelling of a production agent
118  var deviceModel = mongoose.model('devices', deviceSchema);
119
120  // modelling of a service
121  var serviceModel = mongoose.model('service', serviceSchema);
122
123  // modelling of a product
124  var productsModel = mongoose.model('product', productsSchema);
125
126  // modelling of a production order
127  var orderModel = mongoose.model('order', orderSchema);
128
129  // modelling of a production order item
130  var orderItemModel = mongoose.model('orderItem', orderItemSchema);

```

Apêndice B – Ilustração da estrutura de código do Agente de Comunicação.

```

1  /*****
2      Communication Agent
3
4      author: João Ricardo Cardoso
5      *****/
6  // Mosca requests and initiation
7  // Mosca requests and initiation
8  // Mosca requests and initiation
9  var mosca = require('mosca')...
15 // Mosca requests and initiation
16 // Global variables
17 // Mosca requests and initiation
18 var online_agents = {}...
23 // Mosca requests and initiation
24 // Mosca events
25 // Mosca requests and initiation
26 // fired when the broker is ready and running
27 broker.on('ready', function ( err , res ) {...
35 // fired when a agent connects on the broker
36 broker.on('clientConnected', function(client) {...
38 // fired when a agent disconnects from the broker
39 broker.on('clientDisconnected', function (client) {...
44 // fired when some agent subscribes on a topic
45 broker.on('subscribed', function ( topic , client ) {...
50 // fired when some agent unsubscribes on a topic
51 broker.on('unsubscribed', function( topic , client ) {...
56 // fired when a message is published on the broker
57 broker.on('published', function(packet, client) {...
80 // fired when a message acknowledge is received for the broker
81 broker.on('delivered', function(packet, client) {...
87 // Mosca requests and initiation
88 // communication infrastructure
89 // Mosca requests and initiation
90 // fired when some arrives on broker
91 function broker_processMessage( deviceId , topic , message ) {...
140 // fired when an agent send its properties on topic "register"
141 function broker_registerAgent( device ) {...
185 // fired when an agent disconencts from the broker
186 function broker_unregisterAgent( device ) {...
205 // fired when some agent reports a status change
206 function broker_switchReport( deviceId , tagId , value , timeStamp ) {...
303 // fired when some new production order arrives from the ERP Agent
304 function broker_processOrder( order ) {...
322 // used to forward an order to some agent when it comes from ERP agent
323 function broker_forwardOrder ( nOrder , orderItem , fromLine ) {...
382 // fired when some agent reposts an order's status change
383 function broker_updateOrder ( deviceId , orderInfo ) {...
425 // fired when some agent post a new product structure
426 function broker_addProduct( product ) {...
435 // used to publish agents activities on the broker
436 function broker_logPublish ( method , deviceId , tagId , value ) {...
459 // used to generate broker's time stamp
460 function broker_TimeStamp() {...
466 // Mosca requests and initiation
467 // yellow pages infrastructure
468 // Mosca requests and initiation
469 // fired when some service is turned on
470 function yellowpages_Add( nService , deviceId ) {...
477 // fired when some service is turned off
482 function yellowpages_Remove( nService , deviceId ){...
494 // used to find a service on yellow pages, returns an agentId
495 function yellowpages_Find( nService ) {...
503 // fired when some agent requests a yellow pages service
504 function yellowpages_Report( deviceId , nService ){...

```


Apêndice C – Ilustração da estrutura de código do Agente de Produção (JavaScript).

```

1  /*****
2      Production Agent
3
4      author: João Ricardo Cardoso
5      *****/
6
7  //////////////////////////////////////////////////
8  // Global variables
9  //////////////////////////////////////////////////
10 + var mqtt_client...
22  //////////////////////////////////////////////////
23  // Multiagent algorithm through MQTT
24  //////////////////////////////////////////////////
25
26  // used when the agent connects on MQTT broker
27 + function agent_Connect() {...
235 // used when the agent disconnects from MQTT broker
236 + function agent_Disconnect() {...
247 // used to send the registration message to MQTT broker
248 + function agent_Register() {...
266 // used to process incoming messages
267 + function agent_processMessage( topic , payload ) {...
335 // used when the agent receives a service ON/OFF request from another agent
336 + function agent_switchCommand( serviceId , value ) {...
355 // used when agent receives a order service request from another agent
356 + function agent_serviceRequest( serviceId , sOrder ) {...
396 // used to send a request of the next agent for the yellow pages
397 + function agent_getNextAgent() {...
408 // used to forward the order to the next agent
409 + function agent_orderForward( agentId ) {...
427 // used to publish on MQTT broker the status change of a service
428 + function agent_publishStatus( serviceId , value ) {...
507 // used to publish on MQTT broker random data to simulate sensor values
508 + function agent_generateRandomData ( ) {...
532 //////////////////////////////////////////////////
533 // screen operations
534 //////////////////////////////////////////////////
535
536 // used to create a row on the screen to illustrate the agent's services
537 + function screen_createRow( tags ) {...
600 // used to process ON/OFF commands sent by users through screen buttons
601 + function screen_switchCommand( serviceId , value ) {...
649 // used to animate the connection status icon on connection bar
650 + function screen_connectionAnimation( online ) {...

```

Apêndice D - Ilustração da estrutura de código do Agente de Vendas.

```

1  /*****
2  |           Sales Agent
3
4           author: João Ricardo Cardoso
5  *****/
6
7  //////////////////////////////////////////////////
8  // Global variables
9  //////////////////////////////////////////////////
10 ⊕ var mqtt_client...
99  //////////////////////////////////////////////////
100 // Multiagent algorithm through MQTT
101 //////////////////////////////////////////////////
102
103 // fired when the agent connects to the MQTT broker
104 ⊕ function agent_Connect() {...
164 // fired when the agent disconnects from the MQTT broker
165 ⊕ function agent_Disconnect() {...
178 // used to send an order to production agent
179 ⊕ function agent_sendOrder() {...
197 //////////////////////////////////////////////////
198 // Multiagent algorithm through http
199 //////////////////////////////////////////////////
200
201 // used to get the product list from the cloud service
202 ⊕ function agent_getProducts() {...
290 // used to get the order list from the cloud service
291 ⊕ function agent_getOrders() {...
374 //////////////////////////////////////////////////
375 // screen Operations
376 //////////////////////////////////////////////////
377
378 // used to send a new product structure to the cloud service
379 ⊕ function agent_addProduct() {...
424 // used to add a product to the order
425 ⊕ function srceen_addProduct ( nProduct ) {...
464 // used to remove a product from the order
465 ⊕ function screen_removeProduct ( nProduct ) {...
491 // used to clear the order
492 ⊕ function screen_clearOrder() {...
499 // used to animate the connection status icon on connection bar
500 ⊕ function screen_popConnectionStatus() {...
513 // used to plot the gantt graphic of an order
514 ⊕ function screen_ganttPlot ( nOrder , div ) {...

```

Apêndice E - Ilustração da estrutura de código do Agente Supervisório.

```

1  /*****
2      Supervisory Agent
3
4      author: João Ricardo Cardoso
5      *****/
6
7  ///////////////////////////////////////////////////////////////////
8  // Global variables
9  ///////////////////////////////////////////////////////////////////
10 |+ var mqtt_client...
14  ///////////////////////////////////////////////////////////////////
15  // MQTT Operations
16  ///////////////////////////////////////////////////////////////////
17
18  // fired when the agent connects to the MQTT broker
19 |+ function agent_Connect() {...
74  // fired when the agent disconnects from the MQTT broker
75 |+ function agent_Disconnect() {...
87  // fired when some agent connects on the MQTT broker
88 |+ function agent_Register ( device ) {...
94  // fired when some agent disconnects on the MQTT broker
95 |+ function agent_Unregister ( deviceId ) {...
102 // fired when some agent reports a status change
103 |+ function agent_switchReport( deviceId , tagId , value ) {...
149 // fired when some agent starts a working routine
150 |+ function agent_workingRoutine ( img ) {...
157 ///////////////////////////////////////////////////////////////////
158 // screen operations
159 ///////////////////////////////////////////////////////////////////
160
161 // used to create a row on the screen to illustrate the agent's services
162 |+ function screen_createRow( deviceId , tags ) {...
304 // used to remove a row of an agent from the table
305 |+ function screen_removeRow(row) {...
310 // used to process ON/OFF commands sent by users through screen buttons
311 |+ function screen_switchCommand( deviceId , tagId ) {...
321 // used to plot a graphic of the historical activity of a sensor
322 |+ function screen_plotHistoric( deviceId , tagId ) {...
362 // used to plot a graphic of the production analysis of a service
363 |+ function screen_plotAnalysis( deviceId , tagId ) {...
527 // used to animate the connection status icon on connection bar
528 |+ function screen_popConnectionstatus() {...

```