Desenvolvimento de serviços *REST* para monitorização dos consumos de energia em chão de fábrica, baseado nas plataformas *Eclipse IoT*: *Bosch* e *SCoT*

Daniel Camarneiro[1], José Paulo Santos[1], Paulo Pedreiras[2], Rui L. Aguiar[2]

[1] Departamento de Engenharia Mecânica, Universidade de Aveiro, Aveiro, Portugal
[2] Instituto de Telecomunicações, Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, Aveiro, Portugal
Email: daniel.camarneiro@ua.pt

Resumo. Com o objetivo de simplificar a integração de equipamentos na Indústria 4.0, foi desenvolvido um protótipo de arquitetura para interligar diferentes equipamentos na instalação fabril, que utilizem protocolos de comunicação padrão, com os serviços alocados na *Cloud*. Os serviços alocados na *Cloud* permitem criar e configurar as imagens (*digital twins*) dos equipamentos físicos, assim como o armazenamento da informação numa Base de Dados e a sua visualização num *Dashboard*. Também foi desenvolvida uma REST API para o acesso à Base de Dados e o envio de mensagens de comando aos equipamentos em chão de fábrica, permitindo assim facilitar o desenvolvimento de novos serviços e interfaces pelo utilizador.

Palavras-Chave: Conectividade, Indústria 4.0, IoT, Automação, Controlo, Energia.

Acknowledgements. The present study was also developed in the scope of the Project *Augmented Humanity* [POCI-01-0247-FEDER-046103 and Lisboa-01-0247-FEDER-046103], financed by Portugal 2020, under the Competitiveness and Internationalization Operational Program, the Lisbon Regional Operational Program, and by the European Regional Development Fund.

1 Introdução

A implementação da Indústria 4.0 nas empresas é um processo complicado. Esta dificuldade provém, principalmente, dos equipamentos arcaicos que não possuem uma *interface* de fábrica para realizar esta ligação com a Internet.

Esta comunicação foi baseada num projeto cooperativo entre a Universidade de Aveiro e a Bosch Termotecnologia S.A., com o objetivo de integrar equipamentos industriais (nomeadamente analisadores de energia) numa plataforma *IoT* baseada no *Eclipse IoT*. Assim, neste projeto foi desenvolvida e implementada de uma arquitetura genérica, que permita integrar estes equipamentos com a Internet, oferecendo serviços

para a sua gestão, visualização e controlo. Os objetivos propostos para a realização deste projeto foram:

- Disponibilizar um conjunto de serviços WEB (REST/SOAP), reutilizáveis e genéricos, para um grande lote de equipamentos fabris que permitam monitorizar o seu funcionamento e os seus consumos de energia.
- Desenvolvimento dos módulos de interface com os sensores e atuadores locais;
- Desenvolvimento do código/*firmware* para que estes módulos possam ser acedidos a partir da internet.

Esta comunicação está estruturada da seguinte forma. Na Secção 2 realiza-se uma breve revisão do estado da arte atual. Na Secção 3 a apresentação de uma arquitetura genérica contendo todos os componentes a implementar para o funcionamento da plataforma protótipo. Na Secção 4 as etapas da implementação e modificações realizadas durante o desenvolvimento do protótipo. Na Secção 5 os resultados do funcionamento e desempenho do protótipo. Na Secção 6 a análise e conclusão dos resultados obtidos, assim como a exposição das próximas etapas do desenvolvimento.

2 Trabalhos Relacionados

A fase inicial do projeto na exploração e análise de arquiteturas e tecnologias que permitam idealizar uma possível solução para o problema em questão.

Xu e Liu, no artigo [1], desenvolveram uma arquitetura para integrar uma máquina ferramenta padrão na Indústria 4.0. Esta integração foi dividida em três níveis: o *Nível Físico*, onde se encontram todos os equipamentos físicos em chão de fábrica, o *cyberspace*, onde existem as imagens virtuais destes equipamentos, conhecidas como *digital twins*, assim como Bases de Dados, e os *Serviços na Cloud*, onde se encontram um conjunto de serviços destinados à interação, avaliação e gestão dos equipamentos nos níveis anteriores. Apesar de funcional, a arquitetura foi desenvolvida especificamente para uma máquina ferramenta, sendo necessário adaptá-la para outros equipamentos. A evolução da performance e gestão da arquitetura perante a adição de outros equipamentos é outro fator a ponderar, uma vez que não existem resultados que realizem esta avaliação.

Aydin, Hallac e Karakus apresentam no artigo [3] uma arquitetura escalável, dedicada à aquisição, armazenamento e análise de informação proveniente de sensores. Este desenvolvimento foi focalizado na componente *Cloud* da arquitetura, apresentando um exemplo de estrutura implementável neste nível. A utilização da Base de Dados *NoSQL MongoDB* permite armazenas ficheiros *media*, expandindo a flexibilidade da aplicação desta arquitetura, contudo, em aplicações no chão de fábrica, a monitorização é principalmente realizada a partir de registos em texto, adquiridos pelos sensores e atuadores. A aquisição contínua de informação dos equipamentos resulta no aumento da capacidade de armazenamento e processamento necessários, podendo exigir a necessidade de organizar e agregar os registos.

Os serviços baseados na *Eclipse IoT* [9] facilitam a implementação e gestão de dispositivos e as suas funcionalidades na estrutura digital da arquitetura. As plataformas exploradas que oferecem estes serviços foram *Bosch IoT Suite* [8] e SCoT [2]. Os principais componentes do *Bosch IoT Suite* são: *Bosch IoT Hub*, baseado (principalmente) no *Eclipse Hono*, que realiza a adaptação de mensagens de vários protocolos para *AMQP*, criando uma camada abstrata entre os equipamentos e a *Cloud*, o *Bosch IoT Things*, baseado no *Eclipse Ditto*, onde são criados e identificados os digital twins, o *Bosch IoT Rollouts*, baseado no *Eclipse hawkBit*, que permite atribuir automaticamente atualizações de software, e *o Bosch IoT Insights*, que permite o armazenamento (em *MongoDB*) e a visualização de informação. Uma vez que o *Insights* não pertence à família *Eclipse IoT*, a sua ligação com o *Hub* necessita de ser estabelecida manualmente, e a elaboração de *templates* na componente *dashboard* requere o conhecimento prévio da elaboração de *queries* em *MongoDB* para comunicar com a Base de Dados. Outra desvantagem desta plataforma consiste na falta de informação relativamente à implementação de uma comunicação por *WebSocket*, caso seja necessário desenvolver um serviço paralelo ao *Insights* localmente.

O *SCoT*, assim como o *Bosch IoT Suite*, possui o *Eclipse Hono* e *Ditto* na sua composição, além do *Eclipse Kapua*, serviço que permite realizar gestão dos dispositivos e utilizadores. Contrariamente ao *Bosch IoT Suite*, esta plataforma possui exemplos para interagir com a plataforma, nomeadamente para estabelecer a comunicação por *WebSocket* (facilitando o desenvolvimento do serviço paralelo), contudo o *Bosch IoT Suite* possui uma extensa documentação e um portal dedicado à plataforma, com os seus vários serviços.

Liew apresenta no artigo [4] a constituição e os processos na elaboração e compreensão de uma *REST API*, num formato simples e direto, nomeadamente os métodos utilizados e as respetivas ações num pedido realizado por *HTTP*, a estrutura das mensagens (pedidos e respostas), autenticação e os *status code* relativamente às respostas dos pedidos realizados.

3 Solução Proposta

Como fora explicado na introdução, este trabalho foi realizado com o objetivo de integrar equipamentos industriais numa arquitetura baseada no *Eclipse IoT*, permitindo a gestão, armazenamento e visualização dos dados recolhidos, assim como o controlo remoto destes equipamentos. O equipamento principal do desenvolvimento foi o analisador de energia *Janitza UMG 103*, equipamento utilizado na implantação fabril para a monitorização dos consumos energéticos (potências, frequência, correntes, entre outros) dos restantes equipamentos em chão de fábrica. Este analisador comunica por *Rs485 MODBUS RTU*. Também fora proposta a exploração do módulo *IBH Link UA*, que permite configurar um *OPC-UA Server*, realizando a ligação entre os autómatos conectados e *gateways* que possuam um *OPC-UA Client*. Esta comunicação foi estabelecia por *Ethernet*, utilizando um autómato *S7-1200* da *Siemens*.

O desenvolvimento do protótipo teve como referência a arquitetura proposta [**Fig. *1***], que ilustra a constituição dos componentes necessários implementar:

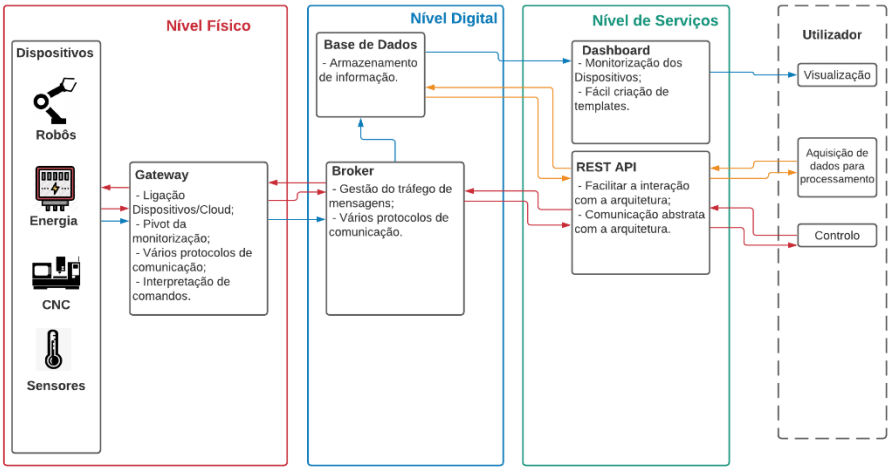

Fig. 1. Diagrama da Arquitetura proposta

1. *Dispositivos*: conjunto de equipamentos em chão de fábrica, como sensores ou *PLCs*, que permitem comunicar através de um protocolo genérico (*MODBUS* ou *OPC-UA* por exemplo).
2. *Gateway*: dispositivo de *middleware* que realiza a ligação entre os dispositivos e o *broker* (da plataforma *IoT*). Se permitir, este componente também pode fornecer serviços de Bases de Dados e *dashboard* locais.
3. *Broker*: serviço fornecido pela plataforma *IoT*, que realiza a gestão da troca de mensagens entre os dispositivos e a *Cloud*. Este serviço também pode realizar a implementação de *digital twins* para a identificação e configuração dos dispositivos (e os respetivos *gateways*).
4. *Base de Dados*: programa responsável pela gestão e armazenamento de informação, ao longo do tempo.
5. *Dashboard*: programa dedicado à elaboração de *templates* para a visualização da informação proveniente dos dispositivos. Caso este não possua uma *API* que interaja com a Base de Dados utilizada, requere a elaboração de um serviço intermédio para este fim.
6. *REST API*: serviço que facilita a interação do utilizador com a arquitetura desenvolvida, utilizando pedidos *HTTP* (*requests*) para interagir com a Base de Dados e os *gateways* (que comunicam com os dispositivos).

A segmentação da arquitetura em três níveis (Físico, Digital e de Serviços) permitiu organizar a disposição e o funcionamento de cada componente para a viabilidade da arquitetura. O *gateway* e a *REST API* instituem elementos de comunicação abstrata entre os restantes componentes da arquitetura (por exemplo, quem programa os equipamentos só necessita de documentar os parâmetros acessíveis por um agente externo), permitindo flexibilizar o seu desenvolvimento.

4 Implementação

A primeira abordagem de arquitetura teve como objetivo o envio de informação para a plataforma *Bosch IoT Suite*. Nesta abordagem utilizaram-se dois dispositivos: um sensor meteorológico *BME280*, que comunica por I^2C, e um analisador de energia *SDM120*, que comunica por *Rs485 MODBUS RTU*. O *gateway* utilizado foi um *ESP32* (microcontrolador pequeno e de baixo consumo, que possui um *shield Wi-Fi* por padrão), programado em *Arduino* (*IDE open-source*, baseado em C/C++, fácil de utilizar, que possui um conjunto variado de bibliotecas que facilitam a prototipagem de microcontroladores), que publicava (por *MQTT* [10]) esta informação para o *broker* da *Bosch IoT* (*Bosch IoT Hub*), responsável por atualizar o *digital twin* do dispositivo (*Bosch IoT Things*), assim como armazenar e visualizar a informação (*Bosch IoT Insights*). Assim como foi dito na secção 2, a ligação entre o *Bosh IoT Insights* e o *Bosch IoT* Hub foi configurada manualmente, e a elaboração de *tempates* requere o conhecimento de *queries* em *MongoDB*.

A segunda intervenção na arquitetura do protótipo foi a migração dos serviços do *Bosch IoT Insights* para uma Base de Dados e *Dashboard* locais. Na fase inicial utilizou-se a Base de Dados *MySQL* (Base de Dados *SQL* convencional) e o *Dashboard* do *NodeRed* [7] (ambiente de programação visual open-source, baseada em JavaScript, permitindo o rápido desenvolvimento de programas, promovido pela sua fácil utilização e extensa biblioteca de módulos). A aquisição de informação do *Bosch IoT Things* também foi realizada em *NodeRed* (por *HTTP*), através da *REST API* deste serviço, sendo necessária uma verificação para o seu acesso (credenciais ou *JWT* (*JSON Web Token*)).

A terceira intervenção baseou-se na procura e implementação de uma Base de Dados e *Dashboard* eficientes perante os requisitos propostos. Assim, alterou-se a Base de Dados de *MySQL* (dificuldade na agregação de elevadas quantidades de informação, necessitando o desenvolvimento de um gestor automático desta informação) para *InfluxDB* [11] (Base de Dados *timeseries*, desenvolvida para históricos de informação, permitindo agrupar esta informação perante a sua idade) e o *Dashboard* do *NodeRed* (*Dashboard* genérico que requere o desenvolvimento da aquisição e formatação da informação) para o *Grafana* [12] (*Dashboard* que possui duas *API*: uma que facilita a ligação a várias Bases de Dados diferentes (incluindo o *InfluxDB*), e outra que facilita a criação de *queries* para comunicar com as Bases de Dados).

Na quarta intervenção realizou-se o desenvolvimento da *REST API* em *Node.js* [5] (ambiente de execução, em *JavaScript*, destinado a aplicações *Web* em servidores) e *Express.js* [6] (*framework* que facilita o desenvolvimento de aplicações Web em Node.js). A *REST API* desenvolvida permite interagir com a Base de Dados, através da formação automática das *queries*, e do envio de mensagens de comando para os dispositivos, através da plataforma *IoT*. Esta implementação apenas apresenta funcionalidades simples, mas constitui uma base sólida para a sua expansão e desenvolvimento de outros serviços.

Na quinta intervenção realizou-se a migração do *Bosch IoT Suite* para o *SCoT*. Esta migração facilitou o processo de aquisição de dados da plataforma, uma vez que a plataforma *SCoT* possui exemplos relativamente à subscrição, por *WebSocket*, de

todas as mensagens proveniente dos *gateways*. Esta funcionalidade foi implementada num programa em *Node.js*, que realiza esta subscrição e injeta esta informação na Base de Dados automaticamente.

A última intervenção foi a implementação do protocolo *OPC-UA* na arquitetura, para comunicar com o módulo *IBH Link UA* e um autómato *S7-1200*. Esta implementação necessitou de um *gateway* mais poderoso, sendo selecionado o *Raspberry Pi 4 B* (pequeno computador de baixo consumo, que utiliza uma distribuição do *Linux* (*Raspbian*), permitindo executar programas desenvolvidos/compatíveis com outras distribuições) para a elaboração de testes. A programação do *gateway* foi realizada em *NodeRed*, recorrendo a uma biblioteca *OPC-UA*. A programação do autómato e do módulo foram realizadas em *TIA Portal* (*IDE* de autómatos da *Siemens*). Neste *gateway* também foi estabelecida uma ligação com o analisador de energia *Janitza UMG 103*, por *Rs485 MODBUS RTU*, recorrendo a um adaptador *USB-Rs485* e uma biblioteca *MODBUS* para o *NodeRed*. A utilização do *Raspberry Pi* também permitiu a implementação de uma Base de Dados e um *Dashboard* locais, como *backup*.

Na ***Fig. 2*** encontra-se a arquitetura do protótipo final, sendo possível observar os programas e equipamentos utilizados, assim como os fluxos de informação entre os componentes.

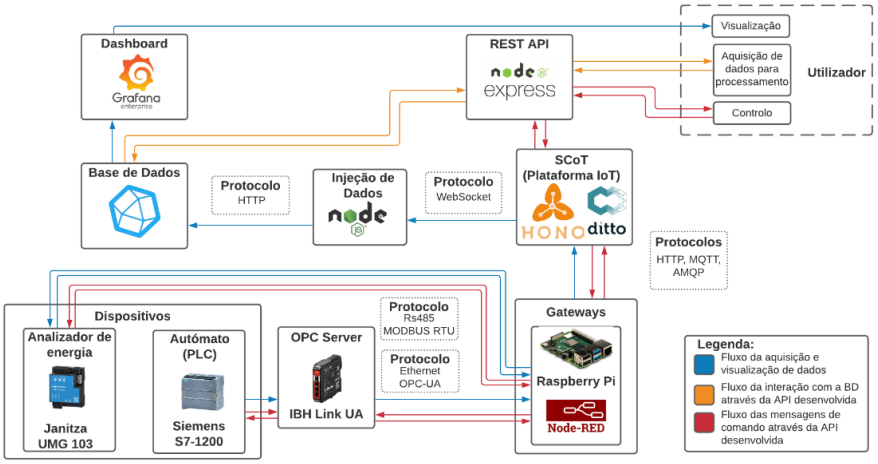

Fig. 2. Arquitetura do protótipo final

5 Resultados

Neste capítulo serão apresentados os ensaios e os respetivos resultados do protótipo desenvolvido. O *setup* utilizado para a elaboração dos ensaios encontra-se na **Fig. 3**.

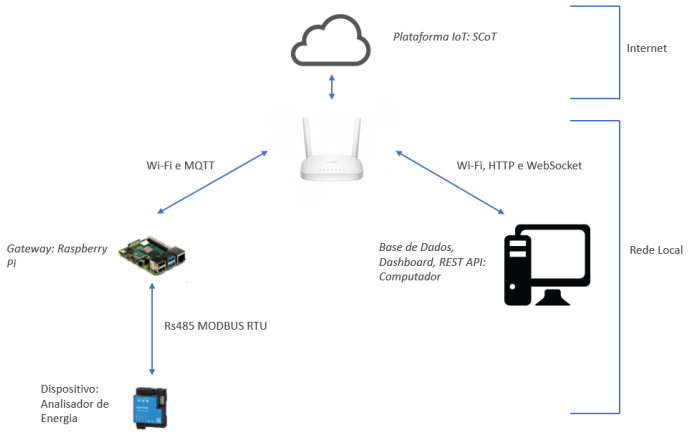

Fig. 3. Setup utilizado para a elaboração de ensaios

5.1 Visualização dos dados do dispositivo

Esta análise permite validar o funcionamento de todos os componentes do protótipo desenvolvido (exceto a *REST API*), desde a aquisição de informação do dispositivo (analisador de energia) até ao armazenamento de informação da Base de Dados, e a sua visualização no *dashboard*. Na **Fig. *4*** é possível observar um exemplo de *template* desenvolvido para visualizar a informação do analisador de energia utilizado.

Fig. 4. *Template* da visualização dos valores 'live' adquiridos do analisador de energia

5.2 Latência de mensagens telemetria entre o Gateway e a Base de Dados

Esta análise permite avaliar o intervalo de tempo entre a publicação de uma mensagem (por *MQTT*), pelo *gateway*, e a sua receção pelo programa responsável na injeção de informação na Base de dados (por *WebSocket*). Como a plataforma *IoT* não se encontra na mesma rede local que os outros componentes, estes ensaios foram repetidos com diferentes taxas de utilização de Internet (com recurso a outro computador, conectado na mesma rede doméstica, variando a velocidade de *download* utiliada),

uma vez que esta representa o maior gargalo atual da arquitetura (uma vez que o trabalho foi desenvolvido em casa). O programa utilizado para teste foi desenvolvido em *NodeRed*, realizando a publicação do *timestamp* atual, num intervalo de 10ms.

Tabela 1. Latência no envio de mensagens de telemetria entre o *gateway* e a Base de Dados

Wi-Fi Load (%)	Mean (ms)	Median (ms)	St. deviation (ms)
0	41	39	13
25	88	42	148
50	110	60	158
75	226	78	298
100	356	247	343

Nos resultados é possível observar o desempenho ideal (a 0%) da publicação de mensagens, sendo esta transação realizada num curto período de tempo e com uma variação baixa. Contudo esta variação aumenta entre 25-50%, podendo influencia o desempenho da monitorização de equipamentos delicados. A 75% e 100% a variação de tempo já compromete o desempenho de toda a arquitetura.

5.3 Latência de mensagens de comando

Esta análise permite avaliar o intervalo de tempo entre as duas etapas do envio de mensagens de comando: o pedido realizado pelo cliente e a resposta por parte do *gateway*. O programa utilizado para teste foi desenvolvido em *NodeRed*, realizando a diferença entre o *timestamp* atual com o do envio do *request* e o da resposta, num intervalo de 1s. O pedido destas mensagens foi realizado através da *REST API* desenvolvida. Estes resultados foram realizados nas mesmas condições dos ensaios anteriores.

Tabela 2. Latência no envio de mensagens de comando entre o cliente e o *gateway*

Wi-Fi Load (%)	Mean (ms)	Median (ms)	St. deviation (ms)
0	93	91	8
25	101	90	32
50	130	92	128
75	151	109	151
100	378	218	418

Tabela 3. Latência no envio de mensagens de comando entre o *gateway* e o cliente

Wi-Fi Load (%)	Mean (ms)	Median (ms)	St. deviation (ms)
0	37	36	3
25	46	36	39

50	75	36	146
75	130	44	272
100	469	269	551

A primeira análise obtida destes resultados é o sucesso da integração dos pedidos de comando à plataforma *IoT* (*SCoT*) na i desenvolvida. Relativamente aos intervalos de tempo, o aumento do intervalo de tempo entre pedidos afetou (e significativamente) a performance do tráfego de mensagens para 25% de uso, comparativamente com a análise realizada para as mensagens de telemetria. Para 50%, 75% e 100%, os resultados são semelhantes aos obtidos para as mensagens de telemetria.

6 Conclusão e Trabalhos futuros

O protótipo desenvolvido apresenta uma arquitetura simples, mas funcional na aquisição, armazenamento, visualização e comando de equipamentos (dispositivos) em chão de fábrica. Esta arquitetura simples e genérica permite flexibilizar a implementação de novos equipamentos e serviços, dependendo dos requisitos necessários para o bom funcionamento da infraestrutura. A utilização de protocolos genéricos (*MODBUS, OPC-UA e* I^2C) permite facilmente integrar um conjunto de equipamentos variados. O desenvolvimento da *REST API* facilita a integração e utilização das funcionalidades da arquitetura, assim como no desenvolvimento de novos serviços, criando uma camada de comunicação abstrata entre a arquitetura e o utilizador. O *InfluxDB* conjuga a familiaridade da linguagem *SQL* com uma gestão automática da sua informação perante as configurações implementadas, mas esta informação é limitada ao formato de texto, sendo necessário adicionar uma outra Base de Dados (*NoSQL*) em paralelo, caso seja necessário armazenar informação noutro formato. O *Grafana* permite facilmente comunicar e interagir com um amplo conjunto de Bases de Dados.

Outro fator importante na avaliação do protótipo desenvolvido é a latência no envio de mensagens (telemetria e comando), fator influenciado pela distância entre os dois componentes, assim como as configurações da rede onde comunicam. Das análises realizadas anteriormente, pondera-se estabelecer diferente intervalos para a publicação de mensagens, dependendo das características do equipamento em monitorização, permitindo assim gerir a utilização de tráfego da rede, otimizando a sua performance. A migração da plataforma *IoT* para uma rede local também permitiria diminuir a latência entre as mensagens.

Embora apresente resultados favoráveis na aplicação originalmente proposta, o protótipo requere algumas intervenções para melhorar a sua flexibilidade de utilização. A lista seguinte é possível observar um conjunto de propostas para melhorar o seu funcionamento:

- Utilizar as funcionalidades de configuração dos *digital twins* para configurar automaticamente o funcionamento dos *gateways*, flexibilizando a instalação e alteração dos equipamentos em chão de fábrica;

- Desenvolver um serviço 'virtual' *gateway* para redes previamente implementadas, permitindo diminuir os custos iniciais na implementação da arquitetura;
- Substituir os *gateways* de prototipagem utilizados por *gateways* industriais, uma vez que a poluição industrial (elétrica, magnética, mecânica, entre outras) pode comprometer o funcionamento destes equipamentos.
- Implementação de um sistema de segurança na *REST API* desenvolvida.
- Implementação do protótipo em chão de fábrica para avaliar a sua performance numa aplicação real.

Referências

1. Liu, C.; Xu, X.: Cyber-physical Machine Tool – The Era of Machine Tool 4.0. Procedia CIRP, 70-75 (2017)
2. Santiago, A.R.; Antunes, M.; Santiago, A. R.; Barraca, J. P.; Gomes, D.; Aguiar, R. L.: SCoTv2: Large Scale Data Acquisition, Processing and Visualization Platform. In: 7th International Conference on Future Internet of Things and Cloud, Istanbul, Turkey (2019). 10.1109/FiCloud.2019.00053
3. Aydin, G.; I Hallac, I. R.; Karakus, B.: Architecture and Implementation of a Scalable Sensor Data Storage and Analysis System Using Cloud Computing and Big Data Technologies, Journal of Sensors 11, (2015).
4. Liew, Z.: Understanding and Using REST APIs. Smashing Magazine. (2021)
5. Node.js, https://nodejs.org/en, last accessed 2021/07/09
6. Express.js, https://expressjs.com, last accessed 2021/07/09
7. NodeRed, https://nodered.org, last accessed 2021/07/09
8. Bosch IoT Suite, https://developer.bosch-iot-suite.com, last accessed 2021/07/12
9. Eclipse IoT, https://iot.eclipse.org, last accessed 2021/07/12
10. MQTT, https://mqtt.org, last accessed 2021/07/12
11. InfluxDB, https://www.influxdata.com, last accessed 2021/07/13
12. Grafana, https://grafana.com, last accessed 2021/07/13