escola superior de tecnologia e gestão
instituto politécnico de leiria

Indústria 4.0:

Monitorização das condições de funcionamento de uma máquina de injeção com OPC UA

Mestrado em Engenharia Mecânica – Produção Industrial

Anatoliy Kaduk

Leiria, março de 2024

escola superior de tecnologia e gestão
instituto politécnico de leiria

Indústria 4.0:

Monitorização das condições de funcionamento de uma máquina de injeção com OPC UA

Mestrado em Engenharia Mecânica – Produção Industrial

Anatoliy Kaduk

Trabalho de Projeto realizado sob a orientação do Professor Doutor Carlos Fernando Couceiro de Sousa Neves e do Professor Doutor Lino Miguel Moreira Ferreira

Leiria, março de 2024

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Engenharia Mecânica – Produção Industrial, no ano letivo 2023/2024, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Agradecimentos

Gostaria de expressar minha profunda gratidão ao Professor Doutor Carlos Fernando Couceiro de Sousa Neves e ao Professor Doutor Lino Miguel Moreira Ferreira por sua orientação, apoio e sabedoria ao longo deste trabalho. Seus insights e conselhos foram inestimáveis e contribuíram significativamente para o desenvolvimento deste projeto.

Também gostaria de estender meus sinceros agradecimentos ao Engenheiro Miguel Ritto, Diretor Geral da Empresa PSAplast – Plásticos Santo António, Lda, por sua generosidade em compartilhar seu conhecimento e experiência nesta área. Sua colaboração foi fundamental para a realização deste trabalho e para o enriquecimento do meu aprendizado.

Além disso, gostaria de dedicar um agradecimento especial à minha esposa, Valentyna Lashkiba, pela paciência, apoio inabalável e compreensão ao longo deste processo. Seu amor e encorajamento foram fundamentais para enfrentar os desafios e alcançar os objetivos deste projeto.

Por último, mas não menos importante, gostaria de agradecer à minha querida filha, Margarida Kaduk, por ser uma fonte constante de alegria, inspiração e motivação. Seu sorriso e entusiasmo trouxeram leveza aos momentos mais desafiadores e tornaram esta jornada ainda mais significativa.

Resumo

O principal objetivo do presente trabalho consiste no desenvolvimento de um sistema de monitorização de uma máquina de injeção baseado no protocolo OPC UA (*Open Platform Communications Unified Architecture*) e, assim, na adaptação da máquina IMI1300S (construída no ano de 1973) para a era da Indústria 4.0. A modernização da máquina permite monitorizar e analisar algumas das respetivas condições e parâmetros de funcionamento. Isso permite, em caso de necessidade, agir de acordo com os resultados obtidos para manter a estabilidade de funcionamento e aumentar a eficiência operacional de equipamento. Assim, consegue-se aumentar a qualidade e quantidade das peças produzidas. Também está previsto que informação do servidor fique disponível em todos os dispositivos com cliente OPC UA ou navegador WEB instalados e ligados à mesma rede do servidor OPC UA. Além de monitorização da máquina em tempo real, o servidor cria uma base de dados e armazena as variáveis da produção no disco do servidor OPC UA, com a possibilidade de extrair essa base de dados para dispositivos externos.

Os parâmetros da máquina de injeção monitorizados neste projeto compreendem a temperatura de água nos circuitos de arrefecimento do molde, a temperatura de óleo hidráulico da máquina, a pressão de injeção, a energia elétrica consumida durante um ciclo e durante a produção total, a quantidade de ciclos e o tempo de ciclo. A cada ciclo de injeção é realizado o registo de todos os valores atuais numa base de dados em formato SQLITE (*Structured Query Language Database Engine*).

A análise de dados obtidos ajuda a avaliar eficiência atual na produção e elaborar as estratégias e técnicas de funcionamento da máquina de injeção para melhorar e otimizar o desempenho, reduzir o tempo de ciclo e diminuir o custo de produção, aumentando a competitividade da empresa no mercado. Além disso, a análise de dados permanente auxilia constantemente a controlar e avaliar a viabilidade económica das soluções propostas e melhorias implementadas e, com base de novos dados, a desenvolver novas recomendações práticas para a otimização dos processos de injeção. No final, contribui para avaliar o impacto dos princípios de Indústria 4.0 na produção de plásticos e como estes princípios podem ser aplicados e integrados nos processos de funcionamento para transformar a produção de plásticos.

Palavras-chave: Indústria 4.0, OPC UA, Máquina de Injeção, IoT

Abstract

The current work objective consists of developing a monitoring system for an injection machine based on the OPC UA protocol thus adapting the machine IMI1300S (built in 1973) to the Industry 4.0 era. Modernizing the machine allows monitoring and analyzing some its operating conditions and parameters. This enables, in case of need, acting according to the results obtained to maintain operational stability or increase operational efficiency. Thus, it is possible to maintain or increase the quality of the produced parts. Also, it is planned that the server's data will be available on all devices with OPC UA client or WEB browser installed and connected to the same network as the OPC UA server. In addition to real-time machine monitoring, the server creates a database and stores production variables on the OPC UA server disk with the possibility of extracting this database to external devices.

The parameters of the injection machine monitored in this development are: water temperature in the mold cooling circuits, hydraulic oil temperature, injection pressure, electric energy consumed during a cycle and during total production, number of cycles, and cycle time. At each cycle, all current values are recorded in a database in SQLITE format.

The analysis of the obtained data helps to evaluate current production efficiency and to develop strategies and techniques for operating the injection machine to improve and optimize performance, reduce cycle time, and reduce production costs, increasing the company's competitiveness in the market. Additionally, continuous data analysis helps to constantly control and evaluate the economic feasibility of the proposed solutions and implemented improvements, and based on new data, develop new practical recommendations for optimizing injection processes. Ultimately, it also helps to evaluate the impact of Industry 4.0 principles on plastics production and how these principles can be applied and integrated into operational processes to transform plastics production.

Keywords: Industry 4.0, OPC UA, Injection Molding Machine, IoT

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

2D	Two Dimensional
AC	Alternating Current
ACID	Atomicity, Consistency, Isolation, Durability
ADC	Analog-to-Digital Converter
API	Application Programming Interface
ANN	Artificial Neural Network
CBR	Case-Based Reasoning
COM	Component Object Model
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DC	Direct Current
DCOM	Distributed Component Object Model
DHCP	Dynamic Host Configuration Protocol
ERP/MES	Enterprise Resource Planning/Manufacturing Execution System
ESP-IDF	Espressif IoT Development Framework
EUROMAP	European Association of Plastics and Rubber Machinery Manufacturers
FTDI	Future Technology Devices International
FREERTOS	Free Real-Time Operating System
GA	Genetic Algorithm
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HDMI	High Definition Multimedia Interface
HC	Hill Climbing
HMI	Human-Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
ID	Identification
IDE	Integrated Development Environment

IEC	International Electrotechnical Commission
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IQ	Intelligence Quotient
ISA	International Society of Automation
I2C	Inter-Integrated Circuit
LAN	Local Area Network
lwIP	Light Weight IP
MAC	Media Access Control
M2M	Machine-to-Machine
MES	Manufacturing Execution System
microSD	Micro Secure Digital
MQTT	Message Queuing Telemetry Transport
MPC	Model Predictive Control
NVS	Non-Volatile Storage
OLE	Object Linking and Embedding
OPC Clássico	OLE for Process Control
OPC	Open Platform Communications
OPC A&E	OPC Alarm & Events
OPC DA	OPC Data Access
OPC HDA	OPC Historical Data Access
OPC ROUTER	OPC Router
OPC UA	OPC Unified Architecture
OPC UA APL	OPC UA Advanced Physical Layer
OPC UA FX	OPC UA Field eXchange
OPC UA TSN	OPC UA over Time-Sensitive Networking
PC	Personal Computer
PID	Proportional-Integral-Derivative
PIM	Product Information Management
POSIX	Portable Operating System Interface
PWM	Pulse Width Modulation
RAM	Random Access Memory

RAMI 4.0	Reference Architecture Model Industrie 4.0
RMS	Root Mean Square
RTOS	Real-Time Operating System
SA	Simulated Annealing
SBC	Single Board Computer
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
SNTP	Simple Network Time Protocol
SoC	System on a Chip
SO	Sistema Operativo
SOA	Service-Oriented Architecture
SQL	Structured Query Language
SQLITE	SQL Database Engine
SSID	Service Set Identifier
SPC	Stored Program Control
SSH	Secure Shell
SPI	Serial Peripheral Interface
SVR	Support Vector Regression
TCP/UDP	Transmission Control Protocol/User Datagram Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UART	Universal Asynchronous Receiver-Transmitter
UNIX	Operating System
USB	Universal Serial Bus
USB-C	USB Type-C
VNC	Virtual Network Computing
Wi-Fi	Wireless Fidelity
XML	Extensible Markup Language

1. Introdução

A injeção de plásticos é um processo de fabricação padrão que normalmente é caracterizado por volumes de produção elevados e é um dos métodos de processamento de materiais mais importantes para a produção em massa de produtos plásticos. São amplamente utilizados em diversos setores da indústria e os seus produtos são onipresentes no nosso dia a dia. Em 2022, só na União Europeia, a importância da área da indústria do plástico foi representada por 53150 empresas, mais de 1.5 milhões postos de trabalho, com um volume de negócios de mais de 400 mil milhões e lucros de 9.2 mil milhões de euros [1]. O aumento de eficiência dos processos de produção significa o aumento da sustentabilidade e, como consequência, uma redução do impacto ambiental da indústria de transformação de plásticos. Ao mesmo tempo, isso leva à diminuição do custo de produção, aumenta competitividade e lucro obtido pelas empresas. Assim, a utilização da tecnologia e do paradigma Indústria 4.0 traz uma vasta gama de benefícios para a indústria em geral e para cada empresa em particular.

Por ser um processo altamente complexo, o processo de produção por injeção de plástico pode ser dividido em três etapas principais: enchimento, compactação e arrefecimento. Ao longo de todo o processo, o plástico está sujeito a significativas e dinâmicas alterações de pressão e temperatura. Todo o processo é complicado porque as suas variáveis estão fortemente acopladas e são difíceis de analisar com precisão. Por enquanto, a fabricação de produtos plásticos por injeção ainda depende principalmente de operação manual e de métodos de tentativa e erro [2]. Obviamente, esta abordagem tradicional tem desvantagens de baixa eficiência de produção, baixa confiabilidade e repetibilidade, sendo fortemente dependente da experiência anterior do operador. Portanto, é imperativo e crucial desenvolver um método avançado de injeção que seja baseado na ciência e orientado pela tecnologia.

As configurações e otimização do processo de injeção ditam a precisão geométrica e as propriedades mecânicas dos produtos finais. Portanto, a deteção, otimização e controlo do processo de produção por injeção têm uma influência crucial na qualidade do produto e tornaram-se num campo de investigação ativo. Além de benefícios económicos, a adoção de práticas sustentáveis pode trazer uma série de benefícios ambientais, incluindo a redução das emissões de carbono e de outros impactos negativos no meio ambiente.

1.1. Motivação

A motivação deste trabalho reside na necessidade premente da indústria de transformação de plásticos de modernizar e otimizar os seus processos de produção. Com o crescente foco na eficiência operacional, sustentabilidade e qualidade do produto, há uma necessidade crescente por soluções inovadoras que possam lidar com os desafios complexos enfrentados por esta indústria. Além disso, a crescente pressão ambiental e regulatória exigem que as empresas adotem práticas mais sustentáveis e reduzam o seu impacto ambiental. Assim, surgiu a ideia de modernizar a máquina de injeção hidráulica tradicional IMI1300S fabricada em 1973 equipando-a com um sistema digital de monitorização de alguns dos seus parâmetros de funcionamento. A análise dos dados adquiridos permitirá identificar oportunidades de otimização dos processos de produção, contribuindo para aumentar a eficiência e a competitividade da empresa, além de reduzir seu impacto ambiental.

1.2. Objetivos de trabalho

Os objetivos deste trabalho são: desenvolver e implementar soluções tecnológicas avançadas que melhorem a eficiência, a qualidade e a sustentabilidade dos processos de produção de plásticos por injeção. Isso inclui a integração de tecnologias como o OPC UA para comunicação e controlo eficientes, a monitorização avançada dos circuitos de arrefecimento do molde, a análise da pressão de injeção do plástico e de consumo de energia elétrica. O principal objetivo é desenvolver um sistema de recolha e disponibilização daqueles dados que permitam, no futuro, desenvolver métodos de injeção baseados na ciência e orientados pela tecnologia, que reduzam a dependência de métodos tradicionais de operação manual e tentativa e erro.

1.3. Estrutura de tese

Esta tese está estruturada de forma a abordar os principais aspetos envolvidos na digitalização de uma máquina de injeção de plástico tradicional, integrando-a na Indústria 4.0 através da tecnologia IIoT4.0. Após este primeiro capítulo introdutório, no Capítulo 2 é apresentado o estado da arte sobre as principais práticas e tecnologias utilizadas na indústria de injeção de plásticos, nomeadamente na área de Indústria 4.0, monitorização de circuitos de arrefecimento, monitorização de temperatura de óleo hidráulico e da tremonha, monitorização de pressão de injeção do plástico, monitorização de consumo de energia elétrica e especificação OPC UA. O Capítulo 3 descreve em detalhe a metodologia utilizada

neste trabalho e o desenvolvimento das soluções aqui propostas. São descritas as etapas envolvidas no desenvolvimento dos sistemas de monitorização com servidor OPC UA em ESP32 e Raspberry Pi 4, bem como os módulos de monitorização. As conclusões e assim como algumas direções para o trabalho futuro são apresentados no Capítulo 4. Por fim, são fornecidas referências bibliográficas.

2. Tecnologia de base e estado da arte

Como todas as áreas industriais, a indústria de transformação de plásticos continua a evoluir e adaptar-se às novas tecnologias e inovações. Nos últimos anos, várias novidades têm surgido no setor, impulsionando melhorias significativas na eficiência, qualidade e sustentabilidade dos processos de fabricação de produtos plásticos. Uma das tendências mais notáveis é a adoção crescente de tecnologias da Indústria 4.0, também conhecida como produção inteligente. Neste contexto, destacam-se tecnologias como o OPC UA, que desempenha um papel crucial na integração e comunicação eficiente entre diferentes sistemas e dispositivos numa instalação industrial.

Um dos equipamentos essenciais na indústria de transformação de plásticos é a máquina de injeção, responsável por moldar peças plásticas mediante a injeção de material fundido num molde. A monitorização dos circuitos de arrefecimento do molde é uma prática comum para garantir que a temperatura do molde seja mantida dentro dos limites ideais, evitando defeitos e garantindo a qualidade das peças produzidas.

Além disso, a monitorização e análise da pressão de injeção do plástico são aspetos cruciais para garantir a consistência do processo de injeção e a qualidade das peças finais. O controlo preciso da pressão de injeção ajuda a evitar problemas como falhas de preenchimento, retrações, rebarbas, descoloração e deformações, resultando em peças com dimensões precisas e características mecânicas adequadas.

Outra área de interesse crescente é a análise do consumo de energia elétrica nas instalações industriais. Um analisador de energia elétrica pode fornecer informações detalhadas sobre a qualidade da energia (formas de ondas, ausências de interrupções, frequência elétrica constante, etc.) e respetivo consumo em diferentes partes do processo de fabrico, permitindo identificar oportunidades de economia de energia, reduzir custos de operação e minimizar o impacto ambiental.

Neste contexto, esta revisão do estado da arte visa fornecer uma visão abrangente das tecnologias e práticas mais recentes relacionadas com a monitorização e controlo de processos na indústria de transformação de plásticos. Serão analisados e discutidos os avanços mais recentes em termos de hardware, software e metodologias de análise de dados,

bem como os benefícios e desafios associados à implementação dessas tecnologias nas operações industriais.

2.1. Conceitos fundamentais da Indústria 4.0

O termo Indústria 4.0 refere-se à quarta revolução industrial, marcada pela integração avançada entre tecnologias digitais e físicas nos processos de produção. Essa transformação representa uma mudança fundamental na forma como as empresas fabricam, armazenam e entregam os seus produtos. Assim, a indústria 4.0 é caracterizada pela interconexão de máquinas, sistemas e dispositivos por meio da *Internet* das Coisas – IoT (*Internet of Things*) e outras tecnologias de comunicação. Isso possibilita a aquisição de dados em tempo real e a comunicação entre as partes diferentes do processo de produção. A aquisição massiva de dados (*Big Data*) proveniente de várias fontes é fundamental. A análise avançada desses dados fornece informações valiosas para a otimização dos processos, a previsão de falhas e a tomada de decisões informadas. Com a crescente conectividade, a cibersegurança torna-se crucial. Assim, a proteção contra ameaças cibernéticas é uma prioridade para garantir a integridade dos dados e a operação segura dos sistemas. A convergência de sistemas cibernéticos e físicos é central na Indústria 4.0. Isso envolve a integração de sensores, atuadores e sistemas de controlo para criar ambientes de produção altamente flexíveis e eficientes. Além disso, a Indústria 4.0 permite otimizar o uso de recursos, incluindo energia, e promover práticas sustentáveis na produção. A integração de tecnologias não substitui, mas complementa as capacidades humanas, por isso, a colaboração entre os operadores humanos e os sistemas autónomos é uma característica central. Mas a implementação bem-sucedida dos conceitos da Indústria 4.0 exige investimentos significativos em tecnologia, formação de pessoal e adaptação de processos. No entanto, ela oferece oportunidades substanciais para melhorar a eficiência, qualidade e inovação nos vários setores produtivos.

2.2. Especificação OPC UA

Nos últimos anos, o uso da especificação OPC UA tem-se tornado cada vez mais comum em aplicações industriais, incluindo a monitorização das operações das máquinas de injeção de plástico e do estado dos moldes. À medida que as tecnologias de automação industrial avançam, a monitorização das operações dessas máquinas torna-se cada vez mais importante. O servidor OPC UA é uma tecnologia de comunicação industrial que permite a

aquisição de dados de várias fontes em tempo real, tornando-o ideal para a monitorização de máquinas de injeção de plástico e canais de arrefecimento dos moldes.

Com a expansão de variedade de dispositivos inteligentes e interligados entre si na Indústria, uma das chaves principais que está por detrás da IIoT (*Industrial Internet of Things*) é a capacidade de ter acesso à *Internet* a nível dos sensores e passar, nos sistemas automáticos, de uma arquitetura tradicional – Hierarquia Piramidal, para a nova arquitetura baseada nas redes com acesso a nuvens *on-line*, definida na parte 3 da norma ISA-95 (*International Society of Automation*) [3], [4] e [5]. É previsível que os sistemas automatizados venham a crescer e a incluir grande quantidade de *stakeholders* que hoje operam em separado. A Figura 1 ilustra a evolução da arquitetura ISA-95 tradicional para a plataforma baseada em redes e nuvens [6].

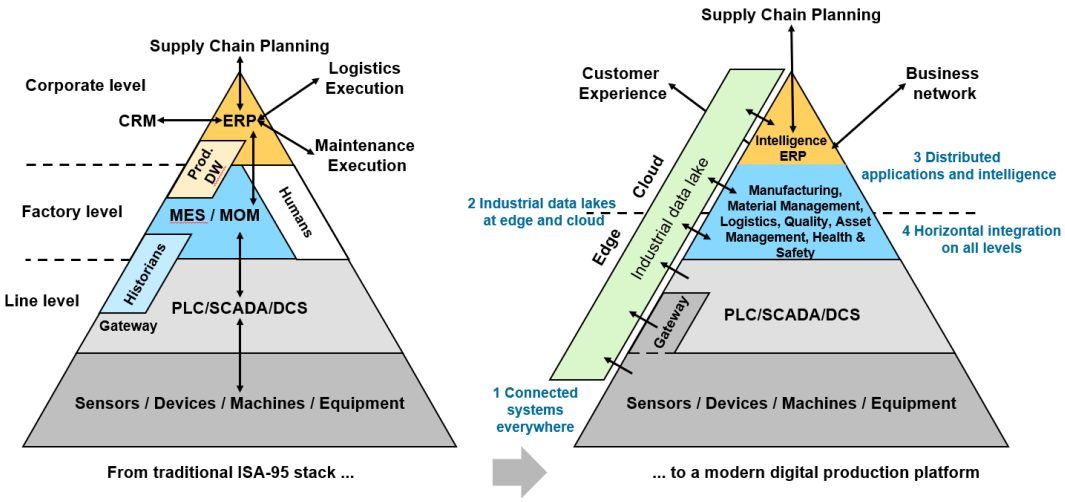

Figura 1 – Evolução da arquitetura ISA-95 tradicional para a plataforma baseada em redes e nuvens[1]

Antes de OPC UA, na década de 1990, surgiu o OPC (*OLE for Process Control*) Clássico como uma solução para integração de sistemas de controlo em ambientes industriais. Originalmente desenvolvido pela Microsoft, o OPC era baseado na tecnologia OLE (*Object Linking and Embedding*) e DCOM (*Distributed Component Object Model*), permitindo a comunicação entre dispositivos de diferentes fabricantes em uma rede industrial. Ao longo do tempo, o OPC Clássico evoluiu para incluir diferentes especificações e tipos, como:

- OPC DA (*Data Access*) – acesso aos dados atuais em tempo real. É o grupo de normas de comunicação entre cliente e servidor que inclui HMI (*Human - Machine Interfaces*), sistemas SCADA (*Supervisory Control and Data*

[1] https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/industry-4-0-and-the-standard-isa-95-the-pathway-to-revolutionizing-the/ba-p/13563444 consultado em março de 2023

Acquisition) e sistemas ERP/MES (*Enterprise Resource Planning/Manufacturing Execution System*);

- OPC AE (*Alarms and Events*) – alarmes e eventos. Trata de notificações dos alarmes, obtém a sequência das notificações, realiza estratégia de auditorias dos alarmes e otimiza estas operações;
- OPC HDA (*Historical Data Access*) – dados históricos. Estabelece acesso às bases de dados, ficheiros de dados ou unidade terminal remoto RTU (*Remote Terminal Unit*) e permite retirar e analisar dados históricos de um processo;
- OPC XML-DA (*Extensible Markup Language – Data Access*) – a primeira plataforma independente de serviços baseados na interação WEB – comunicação em XML.

Cada uma dessas especificações projetada para atender aos requisitos específicos de comunicação e integração de dados em sistemas industriais. Essas várias especificações do OPC Clássico possibilitaram a troca de dados em tempo real, históricos e de eventos entre dispositivos de automação, sistemas de controlo e software de supervisão em diferentes plataformas e ambientes industriais.

No entanto, pelo OPC Clássico foram enfrentadas algumas limitações:

- Interoperabilidade limitada – o OPC Clássico dependia fortemente da tecnologia DCOM, que era específica para o ambiente Windows e apresentava desafios de interoperabilidade em redes heterogêneas;
- Segurança – o OPC Clássico tinha preocupações com segurança devido à dependência do DCOM, que exigia configurações complexas e podia ser vulnerável a ataques cibernéticos;
- Escalabilidade – o modelo de comunicação do OPC Clássico tinha algumas limitações em termos de escalabilidade para ambientes industriais grandes e complexos;
- Integração com tecnologias modernas – com o avanço da tecnologia e a adoção de normas de comunicação mais abertas e seguras, havia uma necessidade crescente de uma arquitetura mais flexível e compatível com as tecnologias modernas.

O OPC UA foi desenvolvido para superar essas limitações, oferecendo arquitetura mais robusta, segura e interoperável para comunicação e integração de sistemas industriais. Desde o lançamento da sua especificação em 2010, o OPC UA obteve um crescimento estável no apoio na comunidade e sucessivas melhorias e adaptações melhorando os seus contextos de aplicação. O protocolo OPC UA é gerido pela OPC Foundation, uma organização global sem fins lucrativos, dedicada ao avanço da interoperabilidade entre sistemas industriais. Padronizado como parte da série IEC 62541 (*International Electrotechnical Commission*) e, é mais recente que não se baseia na tecnologia Microsoft COM (*Component Object Model*) e, portanto, permite compatibilidade entre plataformas. Ao abordar a IIoT, o OPC UA fornece uma ampla base de recursos para acesso, construção, monitorização e controlo de processos industriais. Isto inclui uma arquitetura cliente-servidor que define e normaliza modelo de interface de acesso aos recursos. Consequentemente, todos os sistemas implementados com nós OPC UA podem compartilhar informações com qualquer dispositivo compatível com a norma OPC UA. Essa capacidade de interoperabilidade entre diferentes sistemas, independentemente do fabricante ou plataforma, é uma das principais vantagens do OPC UA. Na Figura 2 pode ver a comparação das normas de OPC Clássico e OPC UA.

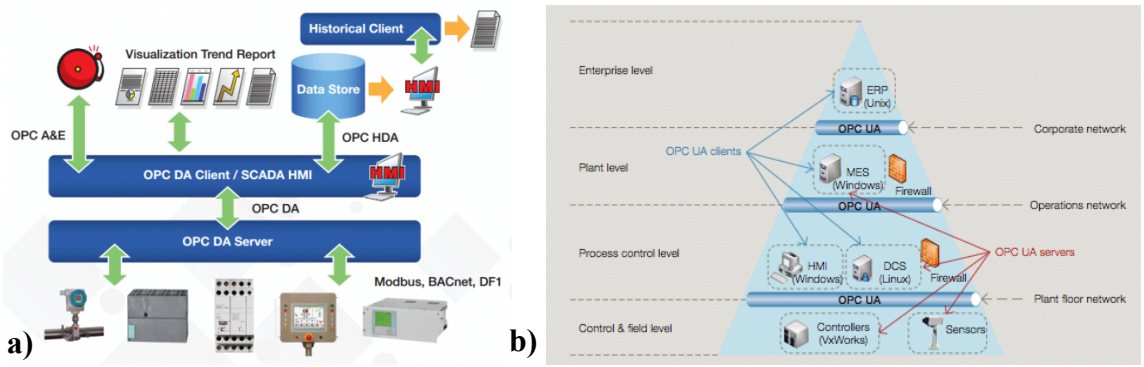

Figura 2 – Comparação das normas de OPC Clássico (a) e OPC UA (b)[2]

A nova norma OPC UA une três normas realizadas feitos para troca de dados e comunicação *online* entre sistemas automatizados e leva estes para a tecnologia de estado da arte, utilizando uma arquitetura orientada para serviços SOA (*Service-Oriented Architecture*). Estas normas são: OPC DA, OPC AE e OPC HDA [7]. Além disso existem extensões adicionais para estes blocos de construção:

[2] https://embarcados.com.br/introducao-ao-opc-ua/ consultado em março de 2023

- OPC Complex Data – interprete e transforme os tipos de dados *non-standard* ou tipos específicos de um servidor em standard da OPC UA [8];
- OPC Batch – especificações desenvolvidas para indústria de processos por lotes [9];
- OPC Data Exchange – descreve comportamento de comunicação entre os servidores de OPC UA (comunicação entre as máquinas) [10] e [11];
- O modelo independente de plataforma PIM (*Platform-Independent Model*) permite utilizar aplicações de OPC UA não só nos sistemas baseados em Windows, mas também nos sistemas baseados em LINUX / UNIX (*Operating System*).

A Plataforma da Indústria 4.0 examinou minuciosamente o OPC UA e validou-o como uma tecnologia chave para a implementação da "Camada de Comunicação" do RAMI4.0 (*Reference Architecture Model Industry* 4.0). A Plataforma Indústria 4.0, dentro da RAMI4.0, é um modelo arquitetónico de referência que define a estrutura para integrar sistemas de produção e tecnologias digitais, promovendo a digitalização e a interoperabilidade na indústria. Assim, a especificação OPC UA continua a evoluir, aparecem novos ramos: OPC UA TSN (*OPC UA over Time-Sensitive Networking*), OPC ROUTER (*OPC Router*), OPC UA APL (*Advanced Physical Layer*), OPC UA FX (*Field eXchange*, inclui APL e TSN), interligação de OPC UA e M2M (*Machine*-2-*Machine*). Além disso, destaca-se a adição da capacidade de comunicação utilizando o paradigma *publish/subscribe*, proporcionando uma forma mais eficiente e escalável de troca de dados entre clientes e servidores OPC UA. Este paradigma permite que os clientes se inscrevam para receber atualizações apenas quando necessário, reduzindo assim a sobrecarga de comunicação e melhorando a eficiência do sistema como um todo.

2.3. OPC UA na indústria de transformação de plásticos

Em novembro de 2016, a Plataforma da Indústria 4.0 publicou uma lista de controlo para classificar e promover produtos como *Basic*, *Ready* ou *Full* para a Indústria 4.0. Para cumprir o critério de comunicação da Indústria 4.0, mesmo a categoria mais baixa requer que o produto seja acessível pela rede via TCP/UDP (*Transmission Control Protocol/User Datagram Protocol*) ou IP (*Internet Protocol*) e integre, pelo menos, o modelo de

informação OPC UA [12].Um exemplo de uma rede típica de comunicação com OPC UA é apresentada na Figura 3.

Em 2019, a EUROMAP (*European Association of Plastics and Rubber Machinery Manufacturers*) e a OPC Foundation estabeleceram o Grupo de Trabalho "OPC UA para Máquinas de Plástico e Borracha". O objetivo deste grupo internacional é desenvolver interfaces normalizadas para máquinas de plástico e borracha com base na tecnologia OPC UA. As recomendações existentes da EUROMAP compreendem:

- EUROMAP 77 – transferência de dados entre máquinas de injeção e MES [13];
- EUROMAP 82.1 – dispositivos de controlo de temperatura;
- EUROMAP 83 – definições gerais [14], foram publicadas pela OPC Foundation como OPC 40077, 40082-1 e 40083.

Figura 3 - Rede de comunicação com o protocolo OPC UA[3]

A mudança mais importante é que o chamado *namespace*, que define a identificação a nível mundial de um padrão de interface, foi transferido da **euromap.org** para **opcfoundation.org**. Isso abriu caminho para a adoção do padrão em outras regiões, como Ásia e América. Mesmo que o padrão seja traduzido para normas nacionais e, se necessário, receba um número local diferente, o *namespace* uniforme garante que os dados a serem trocados sejam uniformes e reconhecidos pelos dispositivos conectados.

[3] Aminabadi, S.S.; Tabatabai, P.; Steiner, A.; Gruber, D.P.; Friesenbichler, W.; Habersohn, C.; Berger-Weber, G. Industry 4.0 In-Line AI Quality Control of Plastic Injection Molded Parts. *Polymers* **2022**, *14*, 3551. https://doi.org/10.3390/polym14173551 consultado em abril de 2023

2.4.Funcionamento de uma máquina de injeção

Uma máquina de injeção é um equipamento industrial utilizado para fabricar peças de plástico por meio do processo de moldação por injeção. Como se pode ver na Figura 4, em geral, a máquina de injeção é constituída por duas unidades principais – a unidade de fecho e a unidade de injeção. A unidade de fecho é responsável por manter o molde fechado durante o processo de injeção. Ela consiste numa estrutura robusta que inclui o prato móvel e o prato fixo, nos quais se montam as duas partes do molde (bucha e a cavidade). A unidade de injeção é responsável pela fusão do material plástico e pela sua injeção no molde. Ela inclui um cilindro onde o plástico é derretido, um fuso de injeção que transporta o plástico derretido e um bico de injeção que injeta o material fundido no molde. Após a injeção do material, o molde permanece fechado enquanto o plástico dentro do molde arrefece e solidifica, tomando a forma da peça desejada. Após a solidificação, o molde é aberto, e a peça moldada é ejetada ou retirada (manualmente, por meio de um robô ou automatismo dedicado) e o ciclo repete-se para produzir mais peças. Durante todo o processo, vários parâmetros, como a temperatura, a pressão e o tempo, são monitorizados e ajustados pelo sistema de controlo para garantir a qualidade da peça e a eficiência do processo [15].

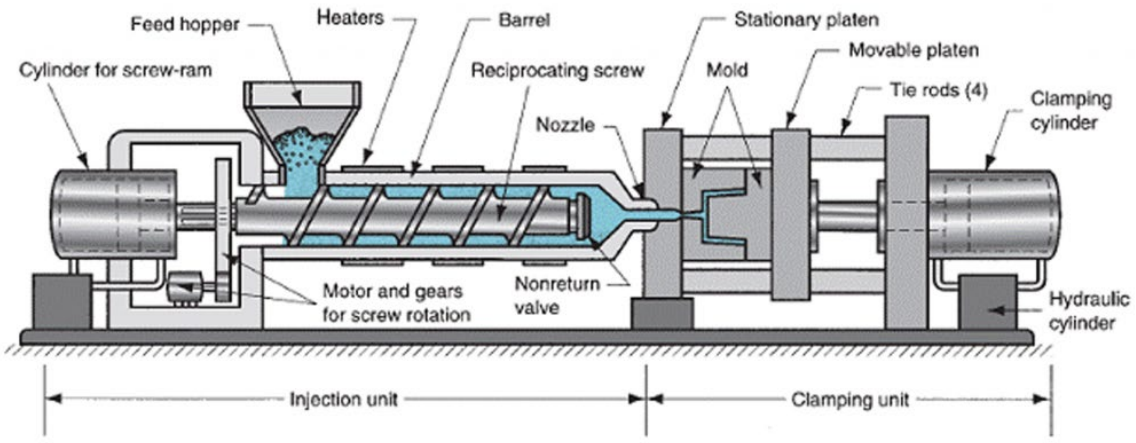

Figura 4 – Estrutura de uma máquina de injeção[4]

A temperatura desempenha um papel crítico na produção de peças plásticas e, por isso, tem de ser monitorizada e controlada com precisão para garantir alta qualidade das peças produzidas. Entre as temperaturas mais importantes nesse processo contam-se a temperatura do cilindro de injeção, a temperatura do plástico fundido, se for o caso, temperatura do sistema dos canais quentes do molde, a temperatura de água de arrefecimento do molde, e

[4] https://xcentricmold.com/injection-molding-process/ consultado em abril de 2023

temperatura do molde, temperatura da zona inferior da tremonha, finalmente, a temperatura do óleo hidráulico da máquina [16].

Outro fator bastante critico na produção de peças plásticas é a pressão, que também é alvo de monitorização e controlo numa máquina de injeção. A pressão é usada para forçar o material plástico derretido a fluir para dentro do molde, preenchendo todos os detalhes da cavidade do molde. Logo depois de injeção, aplicada uma pressão de compactação para garantir que o material permaneça pressionado contra as paredes do molde enquanto solidifica. Isso ajuda a evitar vazamentos ou deformações antes que o plástico esteja completamente solidificado e remove qualquer excesso de ar ou gás e garante que o plástico preenche completamente todos os detalhes do molde [17].

Uma outra informação importante no funcionamento de uma máquina de injeção é o tempo. Analisar e ajustar o tempo de operações permite otimizar a eficiência de produção e garante que a máquina funcione no menor tempo possível para produzir as peças desejadas. A redução de tempo de ciclo e a identificação e melhoria das operações ineficientes economiza energia, material e mão de obra. Menos tempo de funcionamento significa menor consumo de recursos e, portanto, custos mais baixos de produção. Isso pode ser crucial para atender à procura do mercado e aumentar a capacidade de produção sem a necessidade de investir em mais máquinas. A análise do tempo de operações pode ser usada para prever problemas de manutenção, já que alterações nos tempos de operação podem indicar desgaste ou a necessidade de manutenção em componentes específicos da máquina. Além disso, a análise dos tempos de operação ajuda a padronizar os processos de produção o que é especialmente importante na fabricação de peças de alta precisão, onde a consistência é fundamental [17].

O consumo de energia elétrica durante o funcionamento de uma máquina de injeção também é um fator importante para monitorização e análise e é crucial para identificar oportunidades de economia de energia e redução de custos de operação. A monitorização e análise constantes do consumo de energia elétrica ajuda a identificar ineficiências no processo de produção, pode revelar áreas onde há desperdício de energia devido a configurações inadequadas, sistemas obsoletos ou outras falhas. Alterações no padrão de consumo de energia podem indicar desgaste ou necessidade de substituição de componentes específicos da máquina, podendo também ser usada para prever problemas de manutenção. Também, com dados de consumo de energia, é possível otimizar o processo de produção, orientando

os processos de ajuste de parâmetros de operação, programar ciclos de produção de forma mais eficiente e minimizar o desperdício de energia [18].

2.5. Monitorização de circuitos de arrefecimento do molde

Atualmente, estão disponíveis na literatura diversos estudos na área da monitorização e do controlo e da temperatura em moldes para a produção de peças plásticas. As principais áreas de investigação incluem:

- Otimização do projeto do sistema de arrefecimento;
- Tecnologias de arrefecimento ativo;
- Sensores inteligentes de temperatura;
- Materiais avançados de transferência de calor;
- Estratégias de controlo do fluxo de água;
- Integração de energias renováveis;
- Simulação computacional do fluxo de fluidos;
- Moldes com canais conformados;
- Monitorização remoto do arrefecimento.

Das áreas de investigação acima mencionadas, as mais próximas do presente trabalho são os sensores inteligentes de temperatura e a monitorização remota do arrefecimento. Um dos trabalhos que demonstra a importância da monitorização e análise dos parâmetros como temperatura e pressão para determinar a eficiência dos moldes com o intuito de melhorar a qualidade dos resultados de produção, é o realizado por Zhao et al. [19]. Nele, os autores indicam a importância de novos métodos e técnicas de monitorização e controlo de processo de injeção de plástico. Além dos métodos convencionais como medição de pressão e temperatura, Zhao et al. destacam métodos emergentes como realização de observação sobre plástico fundido utilizando ultrassons. Outro método consiste em usar moldes com uma chapa transparente e câmara de filmar ultrarrápida. Também pode ser realizada a tomografia computorizada de raios-x, utilização de sensores capacitivos e até o efeito de levitação magnética [19]. Os dados obtidos são utilizados na otimização dos parâmetros dos processos. Para isso, são utilizados vários métodos de otimização não iterativos – método de Taguchi, CBR (*Case-based reasoning*), *Expert systems*, *Fuzzy systems* e *Surrogate Models* que incluem RSM (*Response Surface Method*), ANN (*Artificial Neural Network*), SVR (*Support Vector Regression*), Modelo de Kriging e Processo Gaussiano. Além disso, Zhao et al.

abordam métodos iterativos, tais como GA (*Genetic Algorithm*), PSO (*Particle Swarm Optimization*), SA (*Simulated Annealing*) e HC (*Hill Climbing*). O passo seguinte é utilização de vários tipos e métodos de controlo. Aqui Zhao et al. indicam PID (*Proportional-Integral-Derivative*) *Controller*, MPC (*Model Predictive Control*), *Learning Control*, *Conventional Iterative Learning Control*, 2D (*Two-Dimensional*) *Control* e *Reinforcement Learning Control.* Tudo isso separado ou em combinações variadas permite realizar o controlo de produção em tempo real e alcançar resultados positivos em aumento de eficiência do processo e de qualidade do produto final.

Neste trabalho são também apresentados métodos e estratégias sobre a deteção, otimização e controlo inteligente da moldação por injeção e são resumidos estudos recentes nessas três áreas. No que diz respeito à deteção de processos, métodos convencionais como sensores de pressão e temperatura ainda são indispensáveis, enquanto métodos emergentes como o uso de ultrassons podem obter outras variáveis físicas importantes para enriquecer a informação extraída do processo. Assim, faz sentido de criar sistemas de monitorização não invasivos como, por exemplo, o sistema de monitorização de temperatura de várias zonas do molde através de temperatura da água nas saídas dos circuitos de arrefecimento do molde.

O arrefecimento do molde é uma etapa crucial no processo de fabricação por injeção de plástico. Após a injeção do material plástico no molde, é essencial que o plástico arrefeça e solidifique adequadamente para garantir a qualidade e precisão das peças produzidas. O processo de arrefecimento do molde é projetado para controlar a temperatura do molde de maneira eficiente, garantindo que o plástico arrefece uniformemente em todas as áreas do molde.

Geralmente, o arrefecimento é realizado por meio de canais de circulação de água que são projetados dentro do molde. A água circula através desses canais para absorver o calor do molde e do plástico, ajudando a reduzir a temperatura do sistema. O controlo preciso da temperatura do molde é essencial para evitar deformações, defeitos e tensões residuais nas peças plásticas produzidas. Além disso, o design dos canais de arrefecimento e o fluxo de água são cuidadosamente otimizados para garantir uma distribuição uniforme da temperatura ao longo do molde. Isso ajuda a minimizar os tempos de ciclo de produção e a aumentar a eficiência geral do processo de fabricação.

Atualmente, estão disponíveis no mercado alguns modelos de equipamento de monitorização dos circuitos de arrefecimento dos moldes, como por exemplo os Flosense 1.0, Flosense 2.0, Flosense 3.0 and Flosense 4.0 da marca DME.

A Flosense 1.0 mede fluxo, temperatura e a pressão em um único canal de fluxo usando um único sensor combinado. Quando usado no circuito de arrefecimento do molde, o aparelho também pode ser conectado a um segundo sensor de temperatura e pressão de água. À medida que a água de arrefecimento passa pelo molde, ela transfere calor do aço para a água de arrefecimento. Quanto mais turbulento for o fluxo, mais eficiente será esse processo de arrefecimento. A diferença entre a temperatura de entrada e a temperatura de retorno é medida como ΔT. Variações repentinas no ΔT podem ser causadas por um aquecedor/arrefecedor defeituoso, canal bloqueado, acumulação de incrustações, etc. Assim, o Flosense 1.0 calcula e exibe a diferença de temperatura e pressão conhecida como ΔT e ΔP com um sensor na entrada e outro na saída. Usando esses valores, o Flosense 1.0 fornece uma indicação da estabilidade do processo e verifica a eficiência, identificando energia desperdiçada e variações na pressão que podem indicar fugas ou canais de água bloqueados. O Flosense 1.0 é projetado para poder ser instalado em vários locais no circuito de arrefecimento, incluindo no fornecimento principal de água, no aquecedor do molde, em canais críticos de arrefecimento ou nos coletores de distribuição. Instalação rápida e facilidade de configuração tornam o Flosense 1.0 num componente indispensável a qualquer configuração de arrefecimento do molde e que deve fazer parte de qualquer instalação em que o controlo de custos e a qualidade sejam considerações-chave [20]. Na Figura 5 é ilustrado o equipamento de monitorização dos circuitos de arrefecimento dos moldes Flosense 1.0 e um circuito de arrefecimento típico.

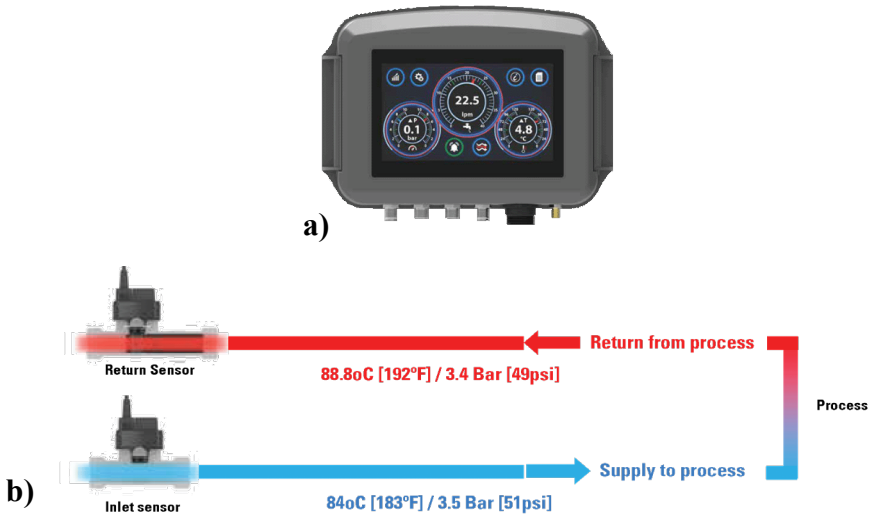

Figura 5 – Flosense 1.0 terminal (a) e circuito de arrefecimento do molde (b)[5]

O Flosense 2.0, apresentado na Figura 6, tem a capacidade de ligar até 4 coletores para monitorizar até 48 circuitos de arrefecimento separados. Este equipamento, permite configurar alarmes de limites de fluxo e temperatura para controlar a estabilidade do processo e a qualidade da peça produzida.

Figura 6 – Flosense 2.0 com um distribuidor[6]

O Flosense 3.0, apresentado na Figura 7, tem um sensor de pressão instalado na entrada principal do circuito e um regulador digital de fluxo. Em comparação com o fluxómetro tradicional analógico, o Flosense 3.0 dispõe de uma monitorização digital do fluxo, alarmes, maior caudal do fluxo, gama de temperatura mais ampla, capacidade de armazenamento e transferência de dados, mudança rápida do molde e interface OPC-UA/EUROMAP.

[5] https://www.dme.net/flosense/ consultado em abril de 2023
[6] https://www.dme.net/flosense/ consultado em abril de 2023

Figura 7 – Flosense 3.0[7]

O Flosense 4.0 apresentado na Figura 8 é a versão mais recente do fluxómetro da marca Flosense. Este apresenta um display com 4 entradas dos coletores, além de um suporte magnético que permite uma instalação fácil e rápida. Está ainda equipado com conectores de alimentação, alarme, USB (*Universal Serial Bus*) e Ethernet.

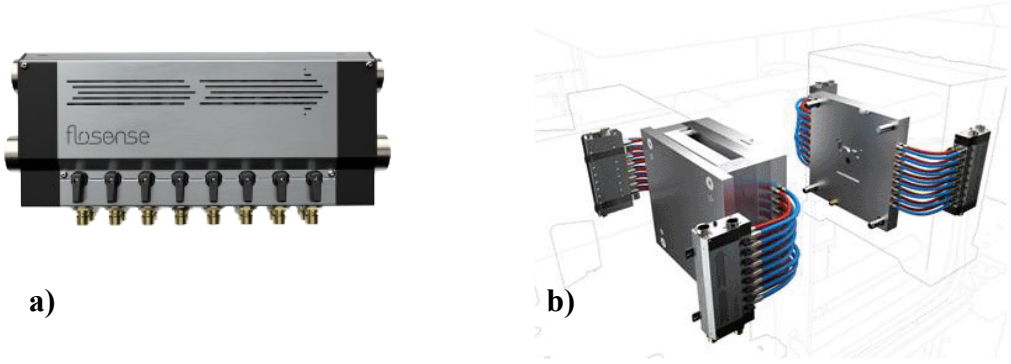

Figura 8 – Flosense 4.0 (a) e montagem no molde (b)[8]

Outro equipamento de controlo e monitorização dos circuitos de arrefecimento do molde é o IQ *flow control* da ENGEL. Este conjunto está ilustrado na Figura 9 e é constituído pelos módulos E-flomo, E-temp e CC300. O módulo E-flomo é um fluxómetro digital que monitoriza fluxo até 40l/min, pressão até 10bar, temperatura até 120ºC e diferença das temperaturas da entrada e da saída. Disponível em vários modelos para monitorizar 2, 4, 6 ou 8 circuitos, é completamente integrável com o CC300 [21]. O módulo E-temp é um equipamento de controlo de temperatura e de fluxo com interface OPC-UA, projetado para aumentar eficiência energética na produção, também totalmente integrado com o E-flomo e CC300. Finalmente, o painel de operador CC300 é um sistema HMI projetado para aplicações industriais exigentes e é um sistema de estação de trabalho com ecrã tátil integrado.

[7] https://www.dme.net/flosense/ consultado em abril de 2023
[8] https://www.dme.net/flosense/ consultado em abril de 2023

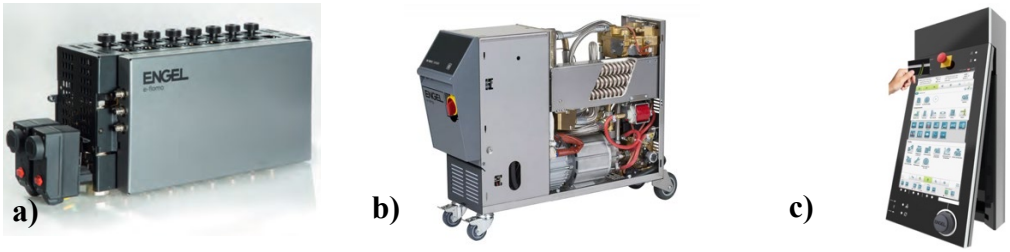

Figura 9 – E-flomo (a), E-temp (b) e CC300 (c)[9]

O painel do operador CC300 é baseado numa placa com arquitetura de processador INTEL que tem instalado o sistema operativo LINUX 64bit. Todos os componentes, hardware e drivers de hardware são projetados para suportar a 100% o sistema operativo ENGEL LINUX x86 64bit. O painel do operador CC300 oferece duas portas LAN (*Local Area Network*) externas. Portanto, ele pode ser conectado simultaneamente, por exemplo, à rede corporativa e a um programa controlo armazenado SPC (*Stored Program Control*). O painel do operador CC300 foi projetado para operar a 24V DC (*Direct Current*) [22].

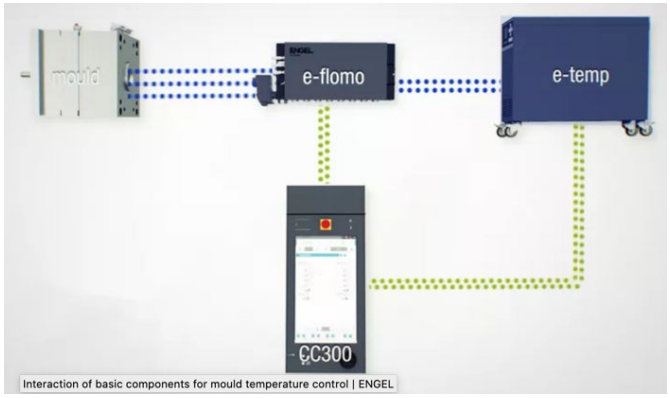

Figura 10 – Sistema de controlo dos circuitos de arrefecimento ENGEL flomo[10]

O CC300 suporta completamente a interface OPC UA e pode comunicar com uma ampla gama de dispositivos periféricos: unidades de controlo de temperatura, sistemas de canais quentes, sistemas de dosagem, para os quais também existem as recomendações da EUROMAP. O conjunto completo do sistema de controlo de circuitos de arrefecimento ENGEL flomo ilustrado na Figura 10 [23].

[9] https://www.dme.net/flosense/ consultado em abril de 2023
[10] https://www.dme.net/flosense/ consultado em abril 2023

2.6. Monitorização de temperatura do óleo hidráulico e da tremonha de alimentação de plástico

As máquinas de injeção modernas estão equipadas com sistemas do controlo que garantem a temperatura ideal do óleo hidráulico e a eficiência do funcionamento. O sistema de controlo da temperatura do óleo hidráulico inclui o sensor de temperatura, o controlador e a válvula no circuito de água do permutador de calor. Isto permite que a máquina atinja rapidamente a temperatura ideal de funcionamento numa gama de 45-50ºC e a mantenha durante todo o processo. Já o sistema que aumenta a eficiência da máquina inclui o variador de velocidade que controla o motor da bomba hidráulica, válvulas hidráulicas proporcionais e os sensores respetivos.

A importância da temperatura correta do óleo hidráulico é crucial porque a viscosidade do óleo está diretamente ligada à sua temperatura. Temperaturas baixas do óleo podem levar a um consumo de energia excessivo e à diminuição da velocidade de funcionamento, impactando negativamente a produtividade da máquina. Por outro lado, temperaturas elevadas do óleo reduzem significativamente sua vida útil e provocam o endurecimento das vedações, o que pode resultar em avarias graves na máquina [24]. Chao Wu et al. salientam a importância da monitorização e do controlo da temperatura do óleo hidráulico nos sistemas hidráulicos, o que permite prolongar a vida útil do óleo e do sistema, além de reduzir as possíveis falhas de funcionamento [25].

Outro ponto importante de monitorização é a temperatura na secção de alimentação do fuso, localizada abaixo da tremonha. Nesta zona, um circuito de arrefecimento é essencial para evitar o sobreaquecimento dos grânulos da matéria-prima e prevenir a sua fusão. O sobreaquecimento pode resultar no entupimento da câmara e na interrupção da produção até que o problema seja resolvido.

2.7. Monitorização e análise da pressão de injeção do plástico

Diversos estudos comprovam a importância da pressão de injeção na cavidade do molde. Durante o processo de injeção, a válvula do bico da máquina é o principal fator que afeta a pressão de injeção [26]. Hernández et al. destacam as características não lineares da pressão de injeção na cavidade do molde e a necessidade de desenvolver sistemas com estratégias avançadas de controlo de injeção. Plant e Maher em [27] salientam a importância da monitorização do perfil de pressão na cavidade do molde para garantir que o tempo de

arrefecimento do jito permanece consistente e, ao mesmo tempo, alcançar o menor tempo de ciclo de injeção possível. Criens et al. [28] propõem que o sentido e grau de orientação das partículas afetam a resistência dos termoplásticos, dependendo da pressão de injeção na cavidade do molde. Esta opinião foi apoiada por Rawabdeh e Petersen, que concluem que o controlo de pressão da injeção e da temperatura de fusão podem ser utilizados para alcançar o comportamento mecânico desejado e a qualidade da superfície da peça produzida [29].

Outros investigadores concentram-se em determinar os parâmetros que afetam a pressão injeção na cavidade. Dontula et al. [30] relatam que o aumento da velocidade de rotação do fuso e a contrapressão durante a dosagem podem aumentar a temperatura do plástico, o que influência indiretamente o perfil de pressão da injeção na cavidade. Macfarlane e Dubay em [31] investigam os efeitos de diversas condições na pressão de injeção na cavidade e nas variações correspondentes na massa da peça. O volume específico é considerado o fator mais importante que afeta o pico de pressão de injeção na cavidade. Essa importância intensifica-se com o aumento da velocidade de injeção, especialmente em peças de espessura baixa [32]. Estes estudos demostram a necessidade de instalar sensores de pressão em diversos pontos do circuito de injeção da máquina e do molde.

Os sensores de pressão podem ser instalados no bico da máquina, no sistema dos canais quentes, no sistema dos canais frios e na cavidade do molde. Este sistema dos sensores pode medir diretamente a pressão do plástico nos pontos exatos durante os processos de injeção, enchimento e arrefecimento da peça. Os dados dos sensores podem ser gravados numa base de dados e utilizados para resolver as anomalias e os problemas possíveis em modo de tempo real ajustando a pressão de injeção.

Atualmente existem dois tipos de sensores usados na cavidade do molde: sensores de montagem direta e de montagem indireta. Exemplo de sensores de pressão de montagem direta (tipo botão) e de montagem indireta (tipo deslizante) bem como as respetivas formas de instalação podem ser vistos na Figura 11. Uma vantagem dos sensores da montagem direta é que não são afetados durante o processo da extração da peça. Por outro lado, uma desvantagem é que podem ser facilmente danificados em condições de funcionamento com temperaturas elevadas. O sensor indireto do tipo deslizante é instalado no elemento de extração ou atras do outro pino especialmente designado para esse fim. Assim, a pressão do plástico através do pino de extração transferido para o sensor de pressão. Sensor do tipo de botão é instalado em uma cavidade especialmente projetada para isso no molde. Como o

sensor do tipo de botão é fixo, as suas leituras são fiáveis em comparação com o sensor de tipo deslizante [33].

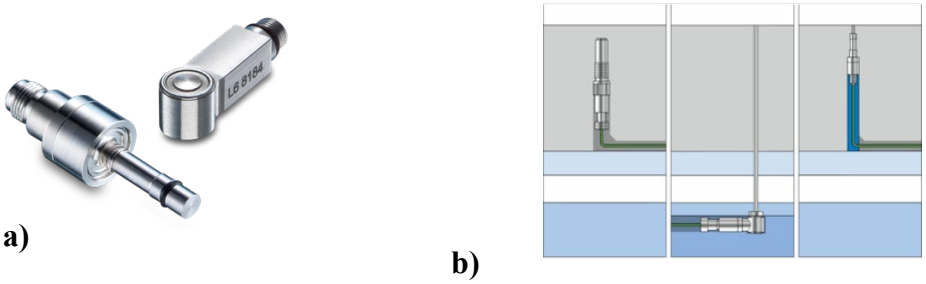

Figura 11 – Os sensores de pressão direto (tipo botão) e indireto (tipo deslizante) (a) e formas de instalação (b)[11]

A utilização destes sensores permite analisar o processo de injeção e descobrir as causas de falhas e defeitos nas peças injetadas relacionados com a pressão de injeção. Como alternativa aos sensores de pressão instalados na cavidade do molde, podem ser utilizados o sensor de pressão de plástico instalado na câmara de aquecimento da máquina ou sensor de pressão de óleo hidráulico instalado no circuito do cilindro hidráulico de injeção. O sensor instalado no circuito hidráulico do cilindro de injeção mede a pressão. A partir de um gráfico com a evolução desse valor durante o ciclo de injeção, o operador pode analisar o processo, identificar anomalias que podem causar defeitos na peça e ajustar as condições de injeção para solucionar os problemas. A Figura 12 ilustra um gráfico típico da pressão de injeção. O gráfico pode variar e depender do molde que está em funcionamento, mas desvios significativos podem indicar para a máquina desafinada, avariada ou outros problemas da máquina ou do molde [34].

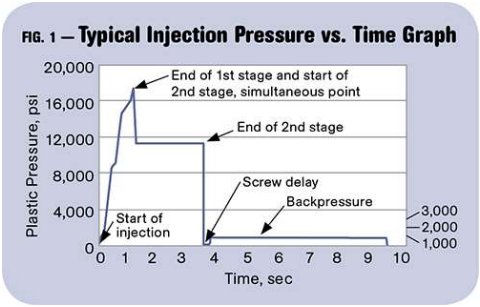

Figura 12 – Gráfico típico da pressão de injeção[12]

As máquinas modernas possuem os sensores de pressão instalados no circuito hidráulico do cilindro de injeção e exibem os valores de pressão hidráulica, pressão do plástico na câmara de aquecimento e velocidade de injeção, permitindo assim a geração dos gráficos correspondentes.

[11] https://gudmould.wordpress.com/ consultado em abril de 2023
[12] https://www.ptonline.com/articles/improve-profits-by-graphing-injection-pressure consultado em julho de 2023

2.8. Analisador de energia elétrica

O objetivo principal da produção é a transformação da matéria-prima e energia num produto. Durante esse processo a energia é consumida e a matéria-prima é transformada. Cada um desses recursos pode ter um impacto significativo no ambiente e na sustentabilidade. Essa influência também pode afetar o custo de produto final, especialmente quando o processo se realiza em grande escala [35], [36]. Assim, aparece o conceito da eficiência energética que consiste em utilizar menos energia para atingir as mesmas metas [37].

O consumo de energia elétrica pela máquina é dinâmico e depende do estado da máquina e etapa de produção. Segundo os autores [38], os diferentes estados da máquina podem ser definidos como:

- Desligada – sem consumo de energia;
- Em aquecimento – designada por picos de consumo devido ao acionamento de vários componentes e etapas de aquecimento;
- Inativa – o consumo de energia elétrica relativamente constante após a fase de aquecimento e a máquina está pronta para iniciar a produção;
- Produção – a energia consumida usada para o processo de produção de peças.

O consumo de energia na produção de peças por injeção é distribuído entre os diferentes componentes do processo. Existem estudos feitos por Spiering et al. que investigaram o consumo de energia na produção de peças numa máquina de injeção hidráulica. Os autores determinaram que cerca 50% da energia consumida estava associada aos movimentos da máquina, enquanto 20% foram consumidos pelo arrefecimento do molde, 10% pelo fornecimento de ar comprimido, 10% pelo aquecimento da câmara e outros 10% pelo transporte e secagem de matéria-prima [39]. Meekers et al. concluíram que cerca de 10% de energia elétrica é consumida durante a injeção, 26% durante a compactação, outros 26% durante a plastificação, 32% durante o arrefecimento e 6% durante a etapa extração de peça produzida [40].

A indústria de injeção de plástico enfrenta diversos desafios, entre os quais a necessidade de otimizar o consumo de energia elétrica. A monitorização e análise do consumo energético das máquinas de injeção é uma ferramenta crucial para alcançar esse objetivo, permitindo identificar as etapas de produção com maior consumo e implementar medidas para reduzi-lo sem comprometer o tempo de produção e a qualidade do produto. Para isso, são utilizados

analisadores de energia elétrica externos como o ETCR5000 ilustrado na Figura 13 ou o HBM eDrive Testing System ou analisadores incorporados na máquina de injeção como nas máquinas da série EC-SXIII da Shiabura-Machine (Japão) [41], [42] e [43].

Figura 13 – Analisador de energia elétrica ETCR 5000[13]

O analisador de energia elétrica ETCR5000 pode medir simultaneamente a corrente de 4 canais (corrente das fases R, S, T e do neutro), a tensão de 4 canais (tensão das fases R, S, T e tensão do neutro em relação a terra), os valores de pico da corrente e da tensão, o valor máximo/mínimo ao longo de um período, fator de desequilíbrio de três fases, cintilação de tensão de curto prazo, fator K do transformador, potência ativa, potência reativa, potência aparente, fator de potência e fator de potência de deslocamento, distorção harmónica total e harmónicos, etc. O analisador disponibiliza as formas de onda em tempo real, gráficos de barras de taxa harmónico de corrente e tensão, captura dinamicamente as mudanças instantâneas de corrente e tensão, monitoriza a corrente inicial e os parâmetros de potência. O analisador permite ainda gerar uma lista de alarmes e gráficos de tendência para dados de teste de longo prazo. Assim, o equipamento ETCR 5000 apresenta o exemplo típico de um analisador de energia elétrica de entre os presentes no mercado comercial [41].

Neste capítulo, foi realizada uma revisão bibliográfica sobre os processos envolvidos na monitorização dos vários sistemas que compõem uma máquina de injeção, tais como os circuitos de arrefecimento do molde, a monitorização da temperatura do óleo hidráulico, e a monitorização da pressão de injeção, entre outros. No Capítulo 3 será apresentado o sistema desenvolvido neste trabalho.

[13] https://www.etcrmeter.com/etcr5000-3-phase-energy-power-quality-analyzer-logger_p41.html consultado em julho de 2023.

3. Desenvolvimento do sistema de monitorização

O objetivo principal deste trabalho é criar um sistema de monitorização para uma máquina de injeção, sistema esse com capacidade de comunicação utilizando o protocolo OPC UA. A máquina de injeção a ser digitalizada é um equipamento com o modelo IMI1300S, fabricada em 1973, que, portanto, não dispõe de sistemas atuais de sensores e monitorização de parâmetros de funcionamento. Os dados e características técnicas principais da máquina são:

- Comprimento máximo da máquina – 11580mm;
- Comprimento mínimo da máquina – 10200mm;
- Curso de abertura e fecho máximo – 700mm;
- Curso de extração central máximo – 200mm;
- Curso de grupo de injeção máximo – 350mm;
- Espessura do molde – 250-750mm;
- Alta pressão no circuito hidráulico – 125bar;
- Baixa pressão no circuito hidráulico – 60bar;
- Força de fecho – 450t;
- Força de injeção de plástico – 74t;
- Capacidade de plastificação – 185kg/h;
- Depósito de óleo hidráulico – 900l;
- Depósito de azoto – 245l @95bar;
- Modelo da máquina – IMI1300S;
- Marca – Industria Macchine Idrauliche S.p.A, Italia;
- Data de fabríco – setembro de 1973.

Na Figura 14 pode se ver a máquina de injeção hidráulica a digitalizar – IMI1300S e o quadro geral da máquina.

Figura 14 - Máquina de injeção hidráulica IMI1300S (a) e quadro geral (b)

Na parte de monitorização da temperatura do óleo hidráulico a máquina a modificar tem apenas o termómetro mecânico para visualizar a temperatura do óleo hidráulico e o termostato mecânico para evitar sobreaquecimento do óleo hidráulico. No caso de sobreaquecimento do óleo o termostato simplesmente para a máquina no final do ciclo e desliga a bomba. Então foi decidido monitorizar temperatura do óleo e, com os alarmes daí resultantes avisar o operador sobre alguma anomalia no circuito do arrefecimento da máquina e evitar uma interrupção indesejada.

Outra parte que se pretende monitorizar é a temperatura dos circuitos de arrefecimento do molde. O caudalímetro utilizado na distribuição de água e controlo de fluxo para circuitos de arrefecimento do molde está equipado com os termómetros mecânicos, ou seja, também aqui não há nenhum tipo de alarme ou aviso no caso de entupimento do circuito de arrefecimento e sobreaquecimento do molde. Foi decidido criar um sistema de monitorização das temperaturas dos circuitos de arrefecimento do molde equipando esses circuitos com sensores digitais.

No circuito hidráulico de injeção a máquina está equipada com um manómetro mecânico que permite visualizar apenas a pressão atual no circuito. Assim, foi decidido criar um sistema de monitorização de pressão de injeção no circuito hidráulico utilizando sensor digital. Isso permite monitorizar pressão de injeção atual e apresentar gráfico de injeção para facilitar análise de processo.

Por fim, a máquina também não está equipada com qualquer sistema de monitorização de energia elétrica utilizada durante funcionamento. A implementação desse sistema permite monitorizar e analisar consumo de energia elétrica durante o funcionamento, identificar

oportunidades de economia de energia, reduzir custos operacionais e minimizar o impacto ambiental.

Para poder realizar uma análise posterior dos dados obtidos durante o funcionamento da máquina e a comparar produção atual com produções anteriores, foi decidido criar um base de dados. Para isso o servidor a construir foi equipado com um dispositivo de memória local.

Neste capítulo serão apresentados os dois sistemas de monitorização com servidores OPC UA um no microcontrolador da família do ESP32 e outro num SBC (*Single Board Computer*).

3.1. Monitorização com servidor OPC UA em ESP32

O desenvolvimento do sistema de monitorização de parâmetros da máquina é dividido em quatro etapas. Cada etapa consiste no desenvolvimento de uma parte do sistema: monitorização das temperaturas, da pressão de injeção hidráulica, da energia elétrica utilizada e desenvolvimento da base de dados. A primeira etapa na realização do presente trabalho consiste no desenvolvimento do sistema de monitorização das temperaturas do óleo hidráulico e dos circuitos de arrefecimento do molde. O diagrama de blocos do servidor OPC UA baseado na ESP32 desenvolvido na primeira etapa está ilustrado na Figura 15.

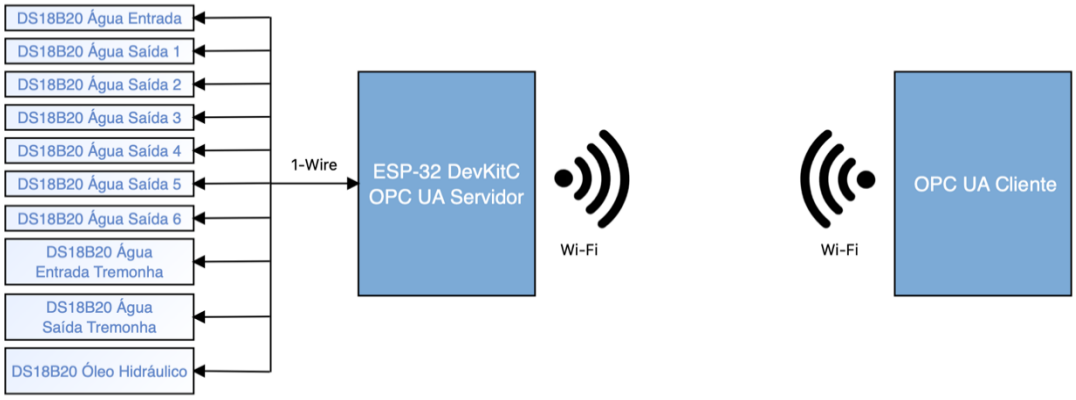

Figura 15 – Diagrama de blocos de servidor OPC UA baseado na ESP32

A segunda etapa consiste no desenvolvimento do sistema de monitorização de pressão de injeção no circuito hidráulico. A terceira etapa é focada no desenvolvimento do sistema de monitorização de energia elétrica utilizada pela máquina durante o funcionamento. E, no final, na quarta etapa, será desenvolvida a base de dados.

3.2. Hardware

A placa de desenvolvimento ESP32-DevKitC da Espressif Systems foi escolhida para este trabalho, ver Figura 16. Esta placa possui um microcontrolador ESP32 com CPU Xtensa® Dual-Core de 32bit e a frequência de operação de 240MHz, uma memória RAM (*Random Access Memory*) de 520kBytes, com os módulos de comunicação sem fios Wi-Fi (*Wireless Fidelity* norma IEEE802.11) e Bluetooth integrados e equipado com uma série de entradas e saídas digitais e entradas analógicas [44]. Assim, ao microcontrolador ESP32 podem ser ligados vários dispositivos digitais e analógicos externos, tais como os sensores, módulo de cartão de memória, etc. Esse módulo, em conjunto com a biblioteca open62541 (de código aberto e desenvolvida em C), permite implementar um servidor OPC UA tamanho reduzido com cerca de 100kBytes [45].

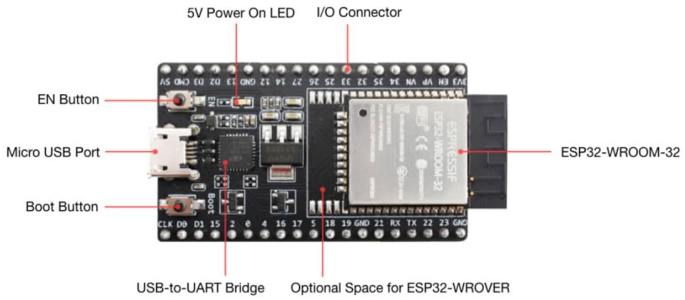

Figura 16 - ESP32 DevKitC[14]

Um dos parâmetros da máquina a monitorizar é a temperatura de água na entrada e na saída dos circuitos de arrefecimento do molde. Para isso, foram utilizados sensores de temperatura digitais DS18B20 da Maxim Integrated Products, Inc. (apresentados na Figura 17). As características principais desse sensor são:

- Sensor digital, na saída retorna um endereço único e o valor de temperatura em formato digital;
- Consumo:
  - 3.3mW, com 1mA em modo ativo;
  - 2.475µW, com 750nA em modo *standby*;

[14] https://pt.mouser.com/new/espressif/espressif-esp32-devkitc-da-development-board/ consultado em agosto de 2023

- Cada sensor dispõe um único número de série de 64bit, também chamado endereço;
- OneWire interface, que permite ligar vários sensores a um único condutor de comunicação;
- Gama das temperaturas medidas de -55ºC a +125ºC com resolução programável de 9 a 12bit;
- Precisão +/-0.5ºC (entre a gama de -10ºC a +85ºC);
- Alimentação de 3.0V a 5.5V DC.

Assim, este sensor é bastante versátil e aplicável em vasta gama de situações industriais, considerando-se adequado para o trabalho a realizar. No total, o sistema de monitorização foi equipado com 10 sensores de temperatura em versão à prova de água e aplicados em vários pontos da máquina com vários comprimentos do cabo, o mais comprido dos quais com cerca de 15m entre o módulo ESP32 e os 7 sensores do caudalímetro. Para estabelecer comunicação estável e robusta, adequada para o comprimento de cabo e o ambiente de ruído eletromagnético industrial, foi utilizado o cabo blindado e entre o pino de alimentação e o pino de comunicação inserida uma resistência de 4.7kΩ recomendada pelo datasheet [46].

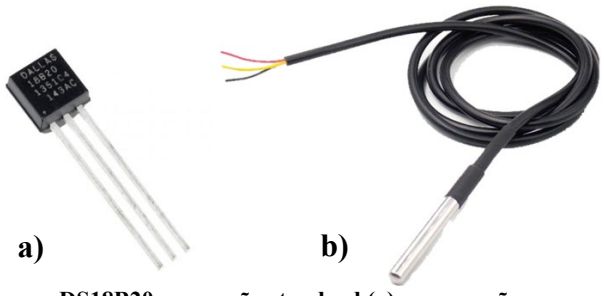

Figura 17 – Sensor DS18B20 em versão standard (a) e em versão a prova de água (b)[15]

O desenvolvimento de software simples para módulos baseados no microcontrolador ESP32 pode ser feito utilizando sistema de desenvolvimento Arduino IDE (*Integrated Development Environment*)[16] ou, para elaboração de software mais complexo, o ESP-IDF (*Espressif IoT Development Framework*)[17] da Espressif, o ambiente de desenvolvimento do fabricante para

[15] https://www.circuits-diy.com/ds18b20-temperature-sensor-2/ consultado em agosto de 2023
[16]https://www.arduino.cc/en/software consultado em agosto de 2023
[17] https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html consultado em agosto de 2023

o hardware baseado no ESP32. Com a complexidade do servidor OPC UA a construir aqui, com uma série de sensores e integrando uma base de dados foi considerado um projeto suficientemente complexo, pelo que foi escolhido a ESP-IDF. Para o desenvolvimento de um servidor OPC UA na ESP32, optou-se pelo uso da biblioteca do projeto open62541[18], desenvolvida em C, com código aberto e sem custos de utilização, uma vez que é compatível com o ESP-IDF e com o microcontrolador ESP32. Também, existe uma biblioteca em C para os sensores DS18B20, compatível com ESP-IDF e microcontrolador ESP32.

3.3. Desenvolvimento do servidor OPC UA no ESP32

A construção do servidor OPC UA foi iniciada com a montagem do hardware seguindo o esquema apresentado na Figura 18. O módulo ESP32 DevKitC em paralelo com os 10 sensores DS18B20 está conectado à fonte de alimentação de 5V DC nos respetivos pinos de alimentação. Como o sensor DS18B20 utiliza tecnologia OneWire na transferência de dados, os pinos de dados de todos os sensores estão conectados em paralelo entre si e ao pino GPIO 26 (*General Purpose Input/Output*) do módulo ESP32 DevKitC [46].

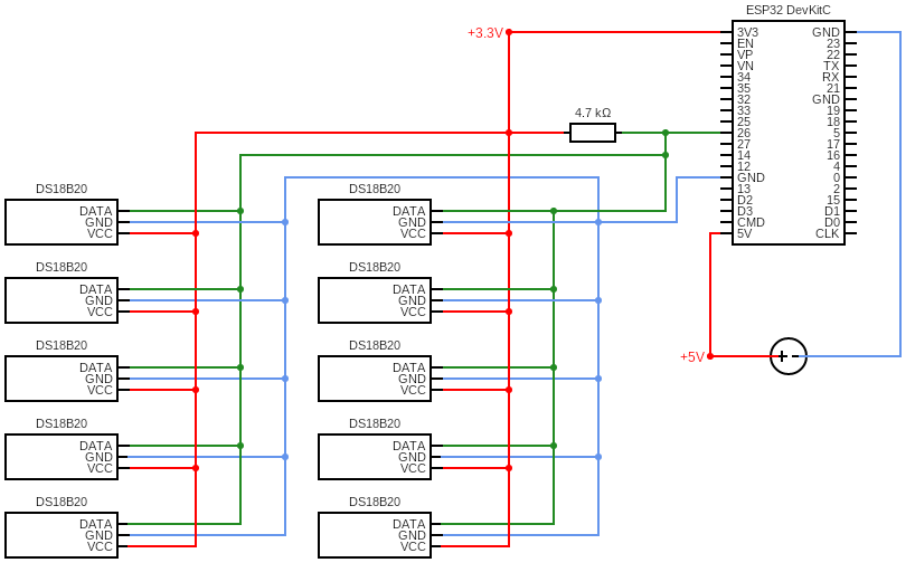

Figura 18 - Esquema do servidor OPC UA baseado no ESP32

Na primeira etapa de desenvolvimento, a estrutura geral do servidor foi construída com duas partes principais: a parte de medição de temperatura e a parte do servidor em si. Basicamente, o funcionamento do servidor consiste na medição de temperatura pelos sensores e na comunicação entre os sensores e o módulo ESP32 DevKitC. Posteriormente, o servidor

[18] https://github.com/Pro/open62541-esp32 consultado em outubro de 2023

atribui os valores de temperaturas às variáveis em formato OPC UA de acordo com as normas do protocolo. A seguir, disponibiliza os dados através da rede Wi-Fi para os clientes OPC UA que se ligarem ao servidor.

A primeira parte de código do servidor consiste do ficheiro `main.c` e inclui várias funções entre as quais estão a inclusão do ficheiro de protótipos da biblioteca, `opcua_esp32.h` e a definição das constantes e variáveis necessárias. A configuração dos *buffers* de envio e receção para o servidor é feita para que tenham um tamanho de 8192bytes. Segue-se a configuração do nome da aplicação appUri (`open62541.esp32.server`) e do *hostname* (opcua-esp32). Depois é feita a inicialização do servidor OPC UA, e do protocolo SNTP (*Simple Network Time Protocol*) a que se segue a criação e configuração do novo objeto servidor OPC UA com o porto padrão de acesso ao servidor (4840). Neste caso, o parâmetro que indica que é o servidor que irá escolher automaticamente uma porta disponível está definido como 0. Os parâmetros `sendBufferSize` e `recvBufferSize` são os tamanhos dos *buffers* de envio e recebimento configurados anteriormente. Também se inclui a adição e configuração de variáveis do servidor OPC UA e inicia-se o ciclo de funcionamento do servidor OPC UA, responsável pelo tratamento eventos de conexão e desconexão. É ainda executada a função que inicializa o contador de reinicialização e as configurações da NVS (*Non-Volatile Storage*) e inicializa a verificação de conexão e o *loop* de eventos.

A biblioteca `opcua_esp32.h` inclui algumas bibliotecas necessárias para funcionamento do servidor, entre os quais se destacam:

- `esp_netif.h` – oferece APIs (*Application Programming Interface*) para configuração e manipulação de interfaces de rede no ESP32;
- `esp_flash_encrypt.h` – fornece funcionalidades relacionadas com a criptografia de dados na flash da ESP32;
- `esp_task_wdt.h` – oferece funcionalidades para a monitorização do *watchdog timer* específico da tarefa no ESP32;
- `esp_sntp.h` – implementa o SNTP para sincronização de tempo;
- `esp_event.h` – relacionado com o sistema de eventos no ESP32 e permite que as partes diferentes do código comuniquem eventos entre si;
- `esp_system.h` – fornece acesso às funções e informações gerais do sistema, como reinicialização e encerramento;

- `nvs_flash.h` – oferece uma API para trabalhar com o NVS no ESP32;
- `lwip/ip_addr.h` – contém definições relacionadas com o endereço IP no contexto da *stack* TCP/IP (*Transmission Control Protocol/Internet Protocol*) leve (lwIP (*Light Weight IP*));
- `sdkconfig.h` – contém as configurações do SDK (*Software Development Kit*) do ESP32, incluindo definições condicionais com base nas opções selecionadas durante a compilação;
- `open62541.h` – esta biblioteca está relacionada com a realização das funcionalidades de open62541, que é uma implementação do OPC UA para sistemas embebidos.

Além disso, é feita a configuração da ligação Wi-Fi – definindo SSID (*Service Set Identifier*) e palavra-passe de acesso a rede Wi-Fi da fábrica. Essa configuração é feita no ficheiro das configurações do ESP32 `sdkconfig`. A Figura 19 apresenta a parte do ficheiro `sdkconfig` com a configuração da rede Wi-Fi do módulo ESP32 DevKitC.

```
# Connection Configuration
#
CONFIG_EXAMPLE_CONNECT_WIFI=y
# CONFIG_EXAMPLE_CONNECT_ETHERNET is not set
CONFIG_WIFI_SSID="TP-LINK_1544"
CONFIG_WIFI_PASSWORD="*********"
CONFIG_USE_STATIC_IP=y
CONFIG_ETHERNET_HELPER_STATIC_IP4_ADDRESS="192.168.0.201"
CONFIG_ETHERNET_HELPER_STATIC_GATEWAY="192.168.0.254"
CONFIG_ETHERNET_HELPER_STATIC_NETMASK="255.255.255.0"
CONFIG_DNS_ADDRESS="8.8.8.8"
# end of Connection Configuration
```

Figura 19 – Configuração da rede Wi-Fi

O router da fábrica tem o servidor DHCP (*Dynamic Host Configuration Protocol*) ativado e tem a gama de endereços IP dinâmicos de 192.168.0.2 a 192.168.0.199. Isso significa é que o router automaticamente atribui os endereços IP a todos os dispositivos novos conectados ao router e, por definição, reserva esse endereço durante 12h. Por isso há necessidade de alterar as definições do router para atribuir ao módulo ESP32 DevKitC um endereço IP estático. Para isso, basta conectar o ESP32 DevKitC ao router através da Wi-Fi, e, através de um browser, entrar na página do router que normalmente se encontra no endereço 192.168.0.1, entrar na secção Wireless Network do router e encontrar o módulo ESP32 DevKitC na lista dos dispositivos conectados. Nessa lista o router apresenta alguma

informação sobre os dispositivos conectados e, o que interessa, o endereço MAC (*Media Access Control*) do dispositivo. A seguir, na página de reserva de endereços IP estáticos, é necessário atribuir ao endereço MAC do ESP32 DevKitC endereço IP estático que pertence a gama dos endereços IP estáticos.

A segunda parte de código consiste no ficheiro `ds18b20.c` e inclui uma série de bibliotecas e funções para garantir o funcionamento do conjunto de 10 sensores de temperatura ligados ao ESP32-DevKitC. Algumas das bibliotecas necessárias são relacionadas com o funcionamento do microcontrolador ESP32 e outras com o funcionamento dos sensores DS18B20. A biblioteca `ds18b20.h` está relacionada com o sensor de temperatura DS18B20 e geralmente fornece um conjunto de funções e definições que facilitam a interação com o sensor DS18B20. Algumas das funcionalidades típicas que podem ser encontradas nesta biblioteca incluem:

- Inicialização do sensor – funções para inicializar e configurar o sensor DS18B20 antes de iniciar a leitura de temperatura;
- Leitura de temperatura – rotinas para iniciar a conversão de temperatura no sensor e ler os dados de temperatura obtidos;
- Configuração de resolução – o DS18B20 suporta diferentes resoluções de leitura de temperatura e a biblioteca inclui funções para configurar a resolução desejada de 9 a 12bit;
- Manipulação de endereços – como os dispositivos OneWire têm endereços únicos, a biblioteca fornece funções para trabalhar com os endereços dos sensores DS18B20;
- Verificação de conexão – funções para verificar se o sensor DS18B20 está corretamente conectado e responde.

A biblioteca de sistema operacional de tempo real `FreeRTOS.h` (*Real-Time Operating System*) fornece as funcionalidades seguintes:

- Multitarefa – permite a execução concorrente de várias tarefas, cada tarefa tem o seu próprio *stack* e contexto e o *FreeRTOS* (*Free Real-Time Operating System*) gere o escalonamento dessas tarefas com base em prioridades e determina qual tarefa deve ser executada em um determinado momento, com base em suas prioridades;

- Semáforos, Mutexes e Filas – o *FreeRTOS* fornece mecanismos de sincronização, como semáforos, mutexes e filas, que são essenciais para a comunicação e coordenação entre diferentes tarefas num sistema multitarefa;
- Temporizadores – a biblioteca possui recursos de temporização que permitem que as tarefas sejam agendadas para execução em tempos específicos ou após intervalos específicos.

A biblioteca `gpio.h` está relacionada com o controlo e manipulação de GPIOs em sistemas embebidos. O termo GPIO refere-se aos pinos de propósito geral que podem ser configurados como entradas ou saídas digitais, permitindo a comunicação com periféricos, sensores e outros dispositivos. As funcionalidades principais da biblioteca são seguintes:

- Configuração de pinos – a biblioteca fornece funções para configurar pinos GPIO para diferentes modos de operação, como entrada, saída, *pull-up*, *pull-down*, etc;
- Controlo de pinos – permite controlar o estado (alto ou baixo nível lógico) de pinos de saída digital e ler o estado de pinos de entrada digital;
- Interrupções GPIO – oferece suporte às interrupções associadas aos pinos GPIO. Isso permite é que o sistema seja notificado quando ocorre uma mudança no estado de nível lógico de um pino específico;
- Manipulação de registos – em sistemas embebidos, muitas vezes é vantajoso operar diretamente com os registos de hardware para configurar e controlar GPIOs. A biblioteca `gpio.h` fornece uma interface de alto nível para simplificar o acesso a esses registos.

A biblioteca `ets_sys.h` faz parte do ESP-IDF que é um conjunto de ferramentas de desenvolvimento para microcontroladores da família ESP32 e fornece algumas funcionalidades de baixo nível relacionadas com o sistema e com hardware do ESP32:

- Temporização e contagem de ciclos de CPU – a biblioteca pode fornecer funções para medir o tempo ou contar o número de ciclos de CPU. Isso pode ser útil para otimização de código ou para medir a eficiência de determinadas operações;
- Controlo de interrupções – as funções relacionadas com o controlo de interrupções, como habilitar ou desabilitar interrupções em níveis específicos;
- Gestão de memória – funções para alocar ou libertar memória;

- Inicialização e configuração do sistema – funções relacionadas com a inicialização e configuração do sistema, configuração de registos e outros aspetos de baixo nível durante a inicialização do dispositivo;
- Temporizadores e temporizações – funções relacionadas com a gestão de temporizadores e eventos temporizados;
- Controlo de energia – funções relacionadas com o controlo de energia, como, por exemplo, colocar o microcontrolador no estado de baixo consumo;

A biblioteca `esp_timer.h` está relacionada com o ESP-IDF e fornece funcionalidades para trabalhar com os temporizadores (*timers*) no ambiente ESP-IDF. A principal funcionalidade oferecida por essa biblioteca é a capacidade de criar e gerir os temporizadores internos no sistema. Isso é bastante útil em sistemas embebidos, onde o tempo é crítico para o funcionamento adequado dos dispositivos. Alguns dos conceitos e funcionalidades associados ao `esp_timer.h` são:

- Temporizadores de hardware – a biblioteca está envolvida na configuração e manipulação dos temporizadores de hardware disponíveis no ESP32. Esses temporizadores de hardware podem ser usados para gerar interrupções em intervalos específicos;
- Precisão de tempo – o ESP32 é conhecido pela sua capacidade de oferecer alta precisão em termos de medição de tempo. A biblioteca permite que os programadores aproveitem essa precisão para aplicações sensíveis ao tempo;
- Temporizadores únicos e repetitivos – oferece suporte à criação de temporizadores únicos, que disparam uma vez após um determinado período, e temporizadores repetitivos, que disparam a intervalos regulares;
- *Callbacks* de temporizador – a capacidade de associar *callbacks* (funções de retorno) aos temporizadores para que o programador especifique a ação a ser executada quando um temporizador expira;
- Configuração e controlo – são funções para configurar e controlar diferentes parâmetros associados aos temporizadores, como intervalos de tempo, contagem de repetições, etc;
- Temporizadores de software – fornece as funcionalidades relacionadas aos temporizadores baseados em software, que podem ser úteis quando temporizadores de hardware específicos não estão disponíveis ou são limitados.

A programação da parte dos sensores inclui inicialização das variáveis, a configuração de resolução dos sensores (12bit no caso presente), a definição de pino do módulo ESP32 que efetua leitura de dados dos sensores e uma série de funções que garantem o funcionamento coreto dos sensores, ou seja, a leitura de endereços dos sensores e valores de temperatura de cada sensor . Na Figura 20 são apresentados os endereços dos sensores com os valores de temperatura respetivos. Como cada sensor tem o endereço estático, esses sensores podem ser colocados em vários locais do sistema e sempre vão corresponder ao esse mesmo sensor nesse mesmo local. Por fim, esses valores de temperatura são atribuídos às variáveis do servidor OPC UA e tornam-se disponíveis para apresentação no cliente OPC UA.

```
Device Address: 28A05449F6EB3CAC Temp C: 17.50 Temp F: 63.50
Device Address: 287C0849F6DC3CE2 Temp C: 17.50 Temp F: 63.50
Device Address: 28222B49F6C53C37 Temp C: 17.50 Temp F: 63.50
Device Address: 28BA6649F68B3C65 Temp C: 18.12 Temp F: 64.62
Device Address: 28222B49F6C53C37 Temp C: 17.50 Temp F: 63.50
Device Address: 2871F796F0013CFA Temp C: 18.50 Temp F: 65.30
Device Address: 28D3B349F62E3CAD Temp C: 17.50 Temp F: 63.50
Device Address: 282B8149F6C33C85 Temp C: 19.87 Temp F: 67.77
Device Address: 2867AA49F6143C33 Temp C: 19.25 Temp F: 66.65
Device Address: 28574896F0013CDE Temp C: 31.12 Temp F: 88.02
```

Figura 20 – Endereços dos sensores e valores de temperatura respetivos

Após a realização da programação e das configurações necessárias, todos os ficheiros incluídos na pasta do projeto foram compilados pelo ESP-IDF e transferidos para o módulo ESP32-DevKitC. Para iniciar o funcionamento do servidor OPC UA, basta reiniciar o módulo e iniciar a monitorização dos sensores de temperatura através do cliente OPC UA.

3.4. Testes realizados

Para realizar primeiros testes e avaliar o funcionamento do servidor OPC UA é necessário instalar um cliente OPC UA num dispositivo e conectar esse dispositivo à mesma rede do OPC UA servidor. No caso particular do trabalho presente, no computador foi instalado o cliente genérico UaExpert da Unified Automation[19]. Já os primeiros testes demonstraram que o módulo ESP32 DevKitC, apesar de tempo de um tempo de inicialização bastante elevado (alguns minutos), suporta o protocolo OPC UA. No entanto, nos testes ficou demonstrado que não é capaz de garantir o funcionamento estável do servidor com os 10 sensores de temperatura. O módulo ESP32 DevKitC apresentou consistentemente o erro `BadOutOfMemory` depois de alguns minutos de funcionamento. Isso significa que não há

[19] https://www.unified-automation.com/products/development-tools/uaexpert.html consultado em outubro de 2023

memória suficiente disponível para realizar a operação solicitada. Na Figura 21 apresentada visualização de temperatura proveniente de sensores de temperatura e servidor OPC UA antes de gerar erro de falta de memória.

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	urce Timestar	rver Timestar	Statuscode
1	opcua-esp32	NS1|String|temperature1	Temperature1	26.75	Float	23:42:52.937	23:42:52.937	Good
2	opcua-esp32	NS1|String|temperature2	Temperature2	29.38	Float	23:41:19.550	23:41:19.550	Good
3	opcua-esp32	NS1|String|temperature3	Temperature3	27.5	Float	23:42:27.521	23:42:27.521	Good
4	opcua-esp32	NS1|String|temperature4	Temperature4	29.81	Float	23:42:51.191	23:42:51.191	Good
5	opcua-esp32	NS1|String|temperature5	Temperature5	28.75	Float	23:42:28.458	23:42:28.458	Good
6	opcua-esp32	NS1|String|temperature6	Temperature6	28.75	Float	23:42:28.772	23:42:28.772	Good
7	opcua-esp32	NS1|String|temperature7	Temperature7	28.13	Float	23:42:24.779	23:42:24.779	Good
8	opcua-esp32	NS1|String|temperature8	Temperature8	28.69	Float	23:42:49.560	23:42:49.560	Good
9	opcua-esp32	NS1|String|temperature9	Temperature9	27.81	Float	23:42:29.592	23:42:29.592	Good
10	opcua-esp32	NS1|String|temperature10	Temperature10	29.31	Float	23:41:53.339	23:41:53.339	Good

Figura 21 – Visualização de temperatura com OPC UA cliente UaExpert

Nos objetivos de projeto foi previsto, além de temperatura, também monitorizar energia elétrica consumida e pressão de injeção do óleo hidráulico. Isso significa a utilização de mais 7 sensores no servidor OPC UA – 3 sensores de tensão, 3 sensores de corrente e 1 sensor de pressão. Todos esses sensores são sensores analógicos, portanto, têm que ser ligados aos pinos ADC (*Analog-to-Digital Converter*) do ESP32 DevKitC. O módulo ESP32 DevKitC dispõe de 16 pinos ADC – 6 pinos pertencem ao ADC1 e 10 pinos ao ADC2. No entanto, a limitação de utilização de pinos ADC na ESP32 resulta na impossibilidade de usar o ADC2 durante a utilização de Wi-Fi. Como a utilização de Wi-Fi é essencial para o projeto, significa que só vão estar disponíveis os 6 pinos do ADC1. Essa limitação obrigaria a recorrer a uma solução de utilizar um módulo auxiliar com canais ADC. Os dois problemas, a falta de entradas ADC disponíveis e o problema de falta de memória, levaram á procura de uma solução alternativa para o hardware do servidor.

Assim, surgiu a ideia de utilizar módulos separados para leitura de temperatura, para medição da corrente e tensão da energia elétrica e para leitura da pressão de injeção no óleo hidráulico. Para realizar essas tarefas podem ser utilizados módulos baseados no microcontrolador ATmega328P, tendo a escolha recaído no módulo Arduino Pro Mini. O módulo Arduino Pro Mini apresenta duas versões: uma de 3.3V DC de alimentação e 8MHz de frequência de relógio e de 5V DC de alimentação e 16MHz de frequência de relógio. Ambas as versões dispõem de 8 entradas analógicas, o que é importante para o módulo de monitorização de energia elétrica já que este necessita de 6 entradas analógicas. Para manter o funcionamento dos módulos o mais estável e robusto possível, foi decidido de utilizar os módulos de 5V DC de alimentação e 16MHz de frequência de relógio. Como na parte de comunicação entre módulos Arduino e ESP32 existe alguma incompatibilidade na tensão

operacional, tem que ser utilizado um *Level Shifter* de 5V para 3.3V DC. Na Figura 22 ilustrados Arduino Pro Mini e adaptador FTDI (*Future Technology Devices International*).

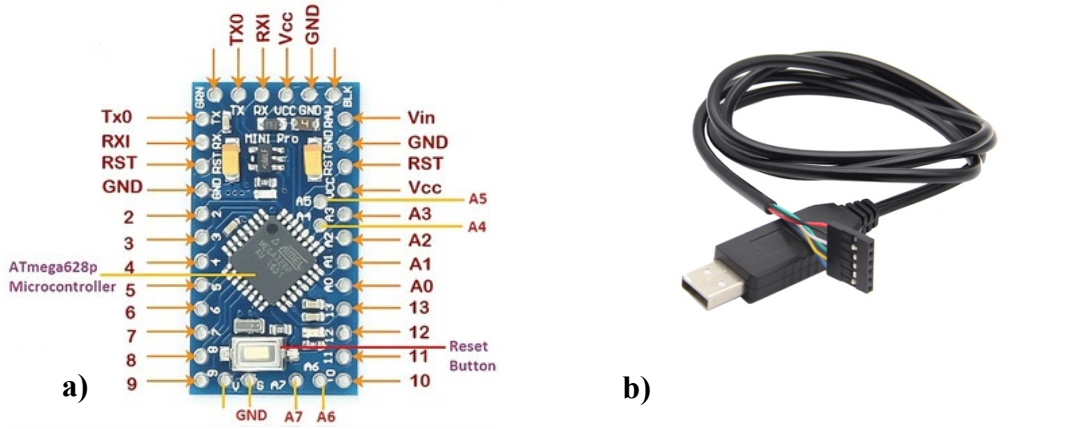

Figura 22 - Arduino Pro Mini (a)[20] e adaptador FTDI (b)[21]

Para desenvolver o software para os módulos Arduino Pro Mini foi utilizado o Arduino IDE. O módulo Arduino Pro Mini não tem conector USB integrado como alguns dos outros modelos da família Arduino. Em vez disso, ele é projetado para ser programado usando uma conexão série. O adaptador FTDI converte os sinais USB do computador em sinais série que podem ser compreendidos pelo Arduino Pro Mini. Assim, o adaptador FTDI fornece uma interface de comunicação entre o ambiente de desenvolvimento e o módulo sendo usado para enviar o código compilado para o Arduino Pro Mini e depurar o código.

A parte de hardware do módulo de leitura de temperatura e pressão hidráulica de injeção consiste em 10 sensores DS18B20, 1 sensor de pressão HEIM3335 182 010 e duas resistências de 10kΩ ligadas em divisor de tensão. O sensor HEIM3335 182 010 é um sensor de pressão hidráulico de grau industrial, com gama de medição de 0 a 160bar, com tensão de alimentação de 12 a 30V DC e tensão de saída de 0 a 10V, onde 0V corresponde a 0bar e 10V corresponde a 160bar. Na Figura 23 apresentado um sensor de pressão hidráulico HEIM3335.

[20] https://robocraze.com/blogs/post/complete-guide-to-arduino-pro-mini consultado em agosto de 2023
[21] https://superacpov.live/product_details/7368539.html consultado em agosto de 2023

Figura 23 - Sensor de pressão hidráulico HEIM3335[22]

A parte de software é um *sketch* de Arduino IDE que combina a leitura digital dos sensores de temperatura e a leitura analógica da tensão proveniente do sensor de pressão. Na leitura dos sensores DS18B20 o procedimento, basicamente, é o mesmo descrito acima para a implementação em ESP32: primeiro o microcontrolador deteta a presença, a quantidade e os endereços dos sensores, depois efetua a leitura dos valores de temperatura de cada sensor sequencialmente. Para realizar essa tarefa são utilizadas as bibliotecas `OneWire.h` e `DallasTemperature.h`. A biblioteca `OneWire.h` fornece as funcionalidades seguintes:

- Protocolo OneWire – permite realizar a comunicação entre os dispositivos usando apenas um fio de dados. A comunicação é feita por meio de pulsos preciosamente cronometrados;
- Inicialização do barramento OneWire;
- Deteção de dispositivos conectados ao barramento OneWire;
- Leitura e escrita de dados nos dispositivos conectados;
- CRC (*Cyclic Redundancy Check*) – efetua verificação de redundância cíclica para garantir a integridade de dados.

A biblioteca `DallasTemperature.h` é uma extensão da biblioteca `OneWire.h` e é projetada para funcionar em conjunto com os sensores de temperatura da família DS18B20 da Maxim Integrated. A biblioteca fornece uma interface de programação para facilitar a leitura de dados desses sensores.

A leitura de sensor de pressão hidráulica é realizada por meio da função integrada no Arduino IDE – `analogRead()`. Como o Arduino Pro Mini opera a 5V DC e tem um ADC

[22] https://suku.de/en/product/580-type-3335-heim-pressure-transmitter-0-10-v-dc-with-stainless-steel-diaphragm.html consultado em agosto de 2023

de 10bit, isso significa que pode representar valores em 2^{10} = 1024 níveis diferentes. Em termos de tensão, isto é, o mínimo 0V corresponde ao 0 e 5V DC corresponde ao 1023 com a precisão de 0.0049V. Em termos de pressão hidráulica, isto traduz-se em: 0bar corresponde ao 0 e 160bar corresponde ao 1023 com a precisão de 0.1564bar. Para converter o valor lido de sensor basta dividir esse valor em 1023/160=6.39375, ou seja, `pressão = analogRead(A0)/6.39375`, onde A0 é o pino de entrada do sinal analógico do módulo Arduino Pro Mini. Na Figura 24 é ilustrado diagrama de blocos de servidor OPC UA alterado.

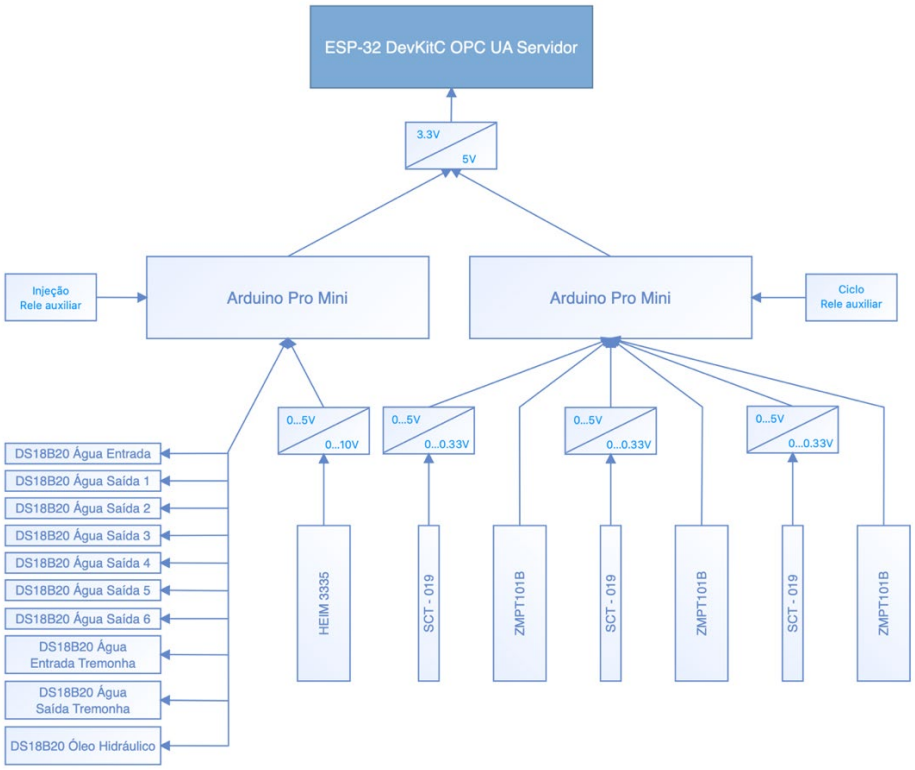

Figura 24 -Diagrama de blocos de servidor com módulos Arduino Pro Mini

Lógica de funcionamento do módulo é a seguinte:

1) O microcontrolador Arduino comunica com os sensores conectados ao barramento OneWire e recebe resposta com quantidade de sensores e os endereços únicos de cada sensor;
2) O microcontrolador Arduino comunica com cada sensor um por um e solicita o valor de temperatura com a precisão definida anteriormente pelo código de programação;
3) Os sensores devolvem os valores de temperatura atuais, o microcontrolador adiciona aos valores recebidos o prefixo de número de sensor e guarda esses

valores em variáveis correspondentes. Por exemplo, valor de temperatura do sensor número 7 tem o aspeto seguinte 1615.63, onde 16 é o número do sensor 7 e 15.63 é a temperatura atual. Isto feito para facilitar distinção de sensores durante a comunicação com o módulo ESP32 DevKitC;

4) Ao mesmo tempo, o microcontrolador monitoriza o pino digital conectado ao relé auxiliar de injeção da máquina;
5) Quando o relé auxiliar estiver ativado, isto significa que a máquina está na fase de injeção de plástico no molde. Nessa altura, o microcontrolador efetua a leitura do sensor de pressão hidráulica de injeção durante 9s a cada 0.075s e guarda todos os valores numa variável do tipo *array* de 120 valores. O tempo de 9s obtido durante a monitorização de funcionamento da máquina, resulta da adição do tempo de injeção máximo e o tempo de compactação máximo, e nunca excede os 9s, independentemente do molde montado nesta máquina. Os 120 valores do *array* correspondem aos valores de pressão hidráulica no instante de tempo respetivo durante a injeção e servem para desenhar o gráfico de pressão de injeção e guardar numa base de dados. Na Figura 25 é apresentado o formato de comunicação entre o módulo Arduino Pro Mini e o ESP32 DevKitC;
6) Quando o relé auxiliar de injeção não está ativado e o microcontrolador não comunica com os sensores de temperatura, ele entra em modo de *broadcast* e comunica com o módulo ESP32 DevKitC enviando os valores de temperatura e valor de pressão de injeção lidos. Antes de enviar o *array* de valores de pressão de injeção, o microcontrolador adiciona o prefixo 99 ao *array* como o número de sensor, para distinguir sensor de pressão hidráulica de outros sensores;
7) Repete o ciclo em modo *loop*.

Figura 25 – Formato de comunicação entre Arduino Pro Mini e ESP32 DevKitC

Antes de desenvolver a parte de analisador de energia elétrica foi realizado outro teste de funcionamento de servidor OPC UA com o hardware modificado. O teste demonstrou que a parte de comunicação entre Arduino Pro Mini e o ESP32 DevKitC consome bastante memória RAM da ESP32. Assim, a nova configuração de servidor OPC UA suporta o

funcionamento estável com apenas 5 sensores de temperatura. A utilização de mais de 5 sensores de temperatura provoca de novo erro `BadOutOfMemory` durante o funcionamento do servidor. Para fazer a depuração deste problema, usou-se a função interna, disponível na ESP-IDF, `esp_get_free_heap_size()`, que retorna a quantidade de memória RAM livre disponível para execução da tarefa. Quando o servidor opera com mais de que 5 sensores de temperatura, esta função retorna a mensagem `I (11540) OPCUA_ESP32: Heap Left: 20520`. Isso significa que o microcontrolador dispõe 20520bytes de memória dinâmica livre e, já essa quantidade é insuficiente para o funcionamento normal de microcontrolador. Assim, o microcontrolador gera erro `BadOutOfMemory` e deixa de funcionar.

3.5. Conclusões

O microcontrolador ESP32 de 240MHz em várias modificações dispõe de 520kB de memória RAM e é bastante poderoso e versátil para vasta gama de projetos IIoT e não só. Essa memória RAM é compartilhada entre várias tarefas do microcontrolador – Wi-Fi, TCP/IP, cache, FREERTOS, comunicação e outras tarefas de baixo e alto nível necessárias para funcionamento correto do microcontrolador. O resto da memória está disponível para a aplicação desenvolvida que inclui a tarefa de servidor OPC UA, a tarefa de comunicação com Arduino Pro Mini, a comunicação com servidor SNTP, etc. O aumento de complexidade funcional do projeto realizado na ESP32 implica o aumento de consumo de recursos computacionais e de memória RAM do microcontrolador. Assim, chega a uma certa altura que torna impossível a realizar projeto pretendido usando este tipo de hardware. Como consequência, houve a necessidade de construir o servidor num equipamento com outras capacidades, optando-se pelo SBC Raspberry Pi 4 Modelo B.

3.6. OPC UA servidor avançado baseado na Raspberry Pi 4

Para a etapa seguinte no desenvolvimento de trabalho foi decidido utilizar um SBC Raspberry Pi 4 Modelo B. Esse SBC dispõe de um processador BCM2711, SoC (*System-on-Chip*) quad-core Cortex-A72 de 64bit e frequência de relógio de 1.5GHz, memória RAM de 4GB, duas portas HDMI (*High Definition Multimedia Interface*), portas USB 3.0 e USB 2.0, Gigabit Ethernet e Wi-Fi 802.11ac de 2.4GHz e 5GHz, slot para cartões microSD (*Micro Secure Digital*) e 40 pinos GPIO que suportam interfaces UART (*Universal Asynchronous Receiver-Transmitter*), SPI (*Serial Peripheral Interface*), I2C (*Inter-Integrated Circuit*) e

PWM (*Pulse Width Modulation)* [47]. Na Figura 26 é apresentado o SBC Raspberry Pi 4 Modelo B.

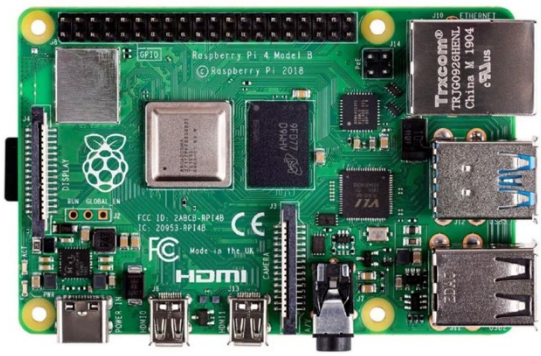

Figura 26 – *Single Board Computer* Raspberry Pi 4 Modelo B[23]

Com essas características técnicas o Raspberry Pi 4 é uma melhor solução para continuar o desenvolvimento do projeto. A estrutura do servidor basicamente é mesma, mas em vez de ESP32 DevKitC está utilizado Raspberry Pi 4. Na Figura 27 é apresentado o diagrama de blocos do servidor OPC UA com base no Raspberry Pi 4.

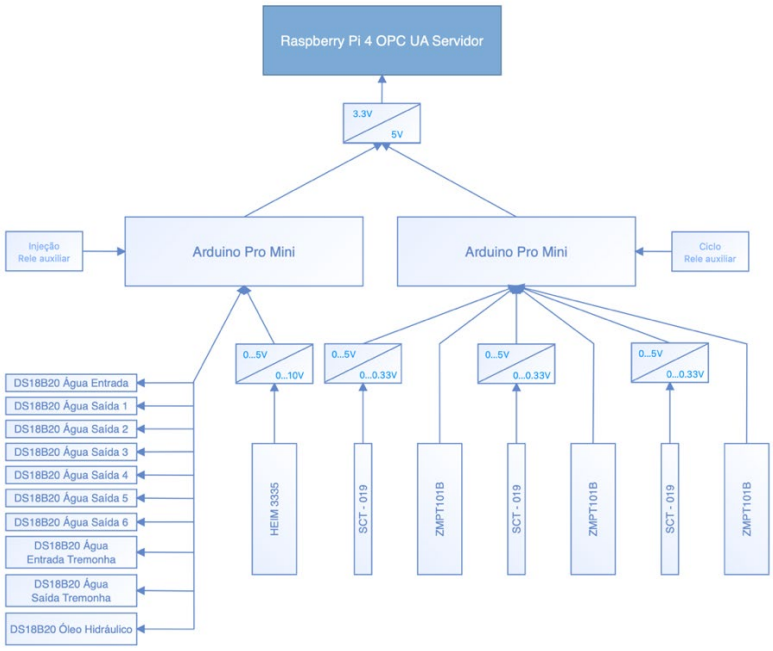

Figura 27 – Diagrama de blocos de servidor OPC UA a base de Raspberry Pi 4

A alteração de hardware do servidor OPC UA implicou a alteração de parte de software e a elaboração de código do servidor OPC UA para Raspberry Pi 4. A parte de software do módulo de temperatura e pressão hidráulico baseado no Arduino Pro Mini não sofreu

[23] https://robocraze.com/products/raspberry-pi-4-model-b-4gb-ram consultado em agosto de 2023

alterações, uma vez que se manteve o formato de comunicação entre Arduino Pro Mini e o servidor anterior.

Para iniciar o desenvolvimento do projeto no Raspberry Pi 4, foi instalada a última versão do SO (Sistema Operativo) Debian Bullseye no cartão microSD através da aplicação Raspberry Pi Imager[24]. A seguir, esse cartão foi instalado no slot para cartões microSD no Raspberry Pi 4. Quando o SBC é ligado pela primeira vez, ele tem que ser configurado, o que inclui, entre outras configurações, também a conexão à rede Wi-Fi da fábrica, o que implica a configuração do router conforme já foi explicado anteriormente. Além disso, foi ativada a utilização de SSH (*Secure Shell*), VNC (*Virtual Network Computing*), Serial Port e Serial Console para a comunicação com o sistema operativo do SBC. Na Figura 28 é apresentada a janela de configuração de interface no Raspberry Pi 4.

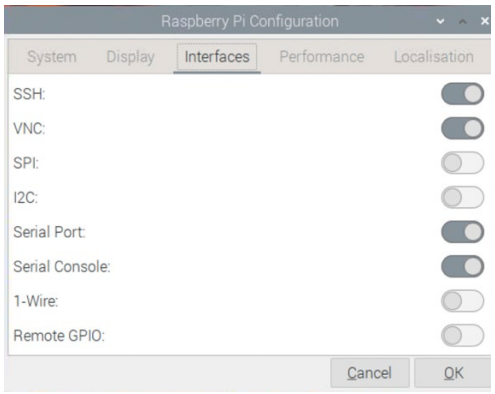

Figura 28 – Configuração de interface na Raspberry Pi 4 Modelo B

O SSH é um protocolo que permite estabelecer acesso e controlar dispositivos remotos de forma segura. Isto pressupõe acesso à linha de comando de um dispositivo a partir de outro computador numa rede local ou pela *Internet*. Uso de SSH permite executar comandos, transferir arquivos e configurar o dispositivo remotamente, sem a necessidade de um monitor, teclado ou rato diretamente conectados ao Raspberry Pi 4.

Já o VNC é um sistema de partilha de desktop que permite visualizar e controlar a interface gráfica de um dispositivo remotamente. Com o VNC é possível estabelecer acesso à área de trabalho do Raspberry Pi 4 a partir de outro computador, tablet ou smartphone numa rede local ou pela *Internet*. O uso do VNC permite interagir com a interface gráfica do Raspberry Pi 4, executar aplicações, navegar na WEB e realizar outras tarefas remotamente, como se estivesse sentado em frente ao dispositivo.

[24] https://www.raspberrypi.com/software/ consultado em agosto de 2023

Raspberry Pi 4 possui até 6 portos de comunicação série que podem ser usados para comunicação de dados com outros dispositivos. Esses portos podem ser usados para conectar dispositivos como módulos GPS (*Global Positioning System*), sensores, módulos de radio, etc. Assim, os portos série são especialmente úteis para comunicação de baixo nível entre dispositivos eletronicos e são amplamente utilizadas em projetos de hardware, IoT e IIoT. Na Figura 29 é apresentada a lista de portos série disponíveis no Raspberry Pi 4 Modelo B.

All UART are **3.3V only** (caveat emptor). The available UARTs and their possible pin assignments are:

Name	Type	Models	Enabled	GPIO (TX)	GPIO (RX)	CTS/RTS
UART0	PL011	All	Yes	14 32 36	15 33 37	-
UART1	mini UART	All	No	14 32 40	15 33 41	-
UART2	PL011	Pi4 Only	No	0	3	2 3
UART3	PL011	Pi4 Only	No	4	7	6 7
UART4	PL011	Pi4 Only	No	8	11	10 11
UART5	PL011	Pi4 Only	No	12	15	14 15

Figura 29 – Portos Serie disponíveis no Raspberry Pi 4 Modelo B

O porto *Console Serial* é uma interface de comunicação que permite estabelecer acesso à linha de comando do Raspberry Pi 4 por meio de uma conexão série. Isso é útil quando o Raspberry Pi 4 não está inicializado corretamente ou quando há necessidade de acesso direto à linha de comando do sistema operativo. Por isso, o *Console Serial* é uma ferramenta poderosa para diagnóstico e solução de problemas e permite visualizar mensagens de inicialização, interagir com o sistema operativo e executar comandos mesmo quando o Raspberry Pi 4 não está a responder por outros meios. Para realizar essa conexão, basta conectar um cabo adaptador FTDI à porta GPIO da Raspberry Pi 4 e estabelecer acesso ao *Console Serial* usando um programa de terminal, como PuTTY no Windows ou o Minicom no LINUX.

Como o módulo de monitorização de temperatura e pressão hidráulica está feito e encontra-se em estado funcional da experiência anterior, para continuar o projeto é necessário desenvolver módulo de monitorização de energia elétrica. Pretende-se que este módulo permita, em tempo real, monitorizar o consumo de energia elétrica da máquina durante um ciclo, o consumo medio durante um ciclo e o consumo total durante a produção atual. Também é necessário desenvolver o software de servidor OPC UA o e a comunicação série com os módulos Arduino Pro Mini mas agora para a arquitetura Raspberry Pi 4. Para facilitar o desenvolvimento do servidor OPC UA no computador portátil foi instalada aplicação VNC Viewer que permite trabalhar no Raspberry Pi 4 remotamente desde que ambos estejam

ligados à mesma rede Wi-Fi. Também, para facilitar desenvolvimento e modificação dos módulos baseados no Arduino Pro Mini diretamente do Raspberry Pi 4 foi aí instalado também o Arduino IDE. Assim, os módulos Arduino Pro Mini conectados diretamente ao Raspberry Pi 4 no canal de programação e no canal de comunicação série e, é mais fácil testar o software desenvolvido.

3.7. Desenvolvimento do servidor avançado

O desenvolvimento do servidor OPC UA para a base Raspberry Pi 4 foi feito a partir de um computador PC por meio do sistema VNC. O desenvolvimento de software no Raspberry Pi 4 foi feito no Geany, um IDE leve, rápido e fácil de usar, com uma interface de utilização intuitiva. O Geany IDE suporta várias linguagens de programação – C, C++, Python, Java, HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheets*), JavaScript, etc. O Geany realça a sintaxe do código, o que facilita a sua leitura e a compreensão. Ele exibe palavras-chave, variáveis, strings e comentários em cores diferentes, o que ajuda a identificar elementos importantes do código. Geany também compila e executa o código diretamente no IDE, o que é conveniente para testar e depurar rapidamente os programas.

Para iniciar a elaboração do servidor OPC UA no Raspberry Pi 4, foi decidido desenvolver a parte de software de servidor e conectar ao Raspberry Pi 4 o módulo de monitorização de temperatura e de pressão de injeção hidráulico baseado no Arduino Pro Mini, reutilizado o desenvolvimento da fase anterior. No Raspberry Pi 4 foi criada uma pasta opcua-pi e dentro dessa pasta um ficheiro de código do servidor `main.c`. O código de servidor foi, de novo, baseado na biblioteca do projeto `open62541.h`, o que facilitou a sua a criação. Além da biblioteca `open62541.h`, o código de servidor inclui uma série de bibliotecas necessárias para funcionamento do servidor entre os quais `math.h`, `time.h`, `sqlite3.h`, `wiringPi.h` e `wiringSerial.h`.

A biblioteca `math.h` é uma biblioteca padrão em C e C++ que fornece funções matemáticas para realizar uma variedade de cálculos numéricos. Ela contém funções para operações matemáticas básicas, funções trigonométricas, funções exponenciais e logarítmicas, funções de arredondamento e muito mais. A biblioteca `time.h` é uma biblioteca padrão em C e C++ que fornece funções e estruturas de dados para lidar com informações de tempo e data. Ela permite que os programas obtenham informações sobre a hora atual, manipulem valores de tempo e data e realizem operações relacionadas com o calendário. A biblioteca

`sqlite3.h` é a interface em C para o SQLite (*Structured Query Language Database Engine*), um sistema de gestão de bases de dados relacional, leve e de alta fiabilidade. O SQLite é amplamente utilizado em aplicações que exigem armazenamento de dados local, como aplicações móveis, navegadores da WEB, sistemas embebidos e aplicações de desktop. A biblioteca `sqlite3.h` fornece todas as funções e estruturas necessárias para interagir com uma base de dados SQLite a partir de um programa em C. A biblioteca `wiringPi.h` é uma biblioteca de código aberto escrita em C para permitir o controlo de pinos GPIO em placas Raspberry Pi. Ela fornece funções para estabelecer acesso e controlar os pinos GPIO de maneira simples e facilita o desenvolvimento de projetos e aplicações que interajam com os periféricos e dispositivos externos. A biblioteca `wiringSerial.h` é uma extensão da biblioteca `wiringPi.h` para comunicação série em placas Raspberry Pi. Ela fornece funções para configurar e usar portas série para comunicação com dispositivos externos, como sensores, módulos GPS, módulos Bluetooth, etc.

No programa, começa por definir-se primeiro um conjunto de parâmetros de funcionamento do hardware como a especificação de portas de comunicação série e os respetivos valores de baud rate, como ilustrado na Figura 30.

```
75  #define DEVICE2 "/dev/ttyAMA1"
76  #define DEVICE "/dev/ttyAMA2"
77
78  #define BAUD_RATE 9600
79  #define BAUD_RATE2 9600
80
```

Figura 30 – Especificação de portos série e baud rate respetivos

Uma vez que a ligação com os módulos de sensores é feita usando comunicação série, é necessário definir um protocolo de formação das mensagens de dados. No caso, o módulo de aquisição de temperatura e de pressão hidráulica envia dados em modo *broadcast* com a forma 1615.63 (apenas um exemplo), onde 16 é o número do sensor 7 e 15.63 é a sua temperatura atual. Para a pressão hidráulica de injeção a forma da mensagem é (de novo exemplificando) 99023024025... onde 99 é o número do sensor de pressão hidráulica e os valores de cada ponto de pressão hidráulica lida pelo sensor (em bar): 23bar, 24bar, 25bar. Assim, o servidor no Raspberry Pi 4 tem que receber esses dados e distinguir todos os sensores pelo número que serve de prefixo em cada mensagem. O programa do servidor monitoriza em modo de *loop* o porto série e guarda os dados no *buffer* de entrada até receber o caracter da fim de linha '`\n`'. Nessa altura, o programa extrai os primeiros dois símbolos

e copia esse valor para o *buffer* de número de sensor, através da função `atoi()` converte em número inteiro do tipo `int` e guarda na variável `sensor`, se o número de sensor estiver entre 10 e 19 o valor que vem no resto da mensagem é guarda na variável `temp`. A seguir, programa atribui o valor de temperatura à variável `temper7` (no caso do sensor número 7). Se o número do sensor é 99 o programa guarda o resto da mensagem na variável *array* do tipo `int` de 120 valores `pressao[120]`. Neste caso o programa copia três caracteres do *buffer* para a variável *array* do tipo `char numStr[4]`, adiciona carater de *Null-terminate* de string '`\0`', converte os caracteres em número através da função `atoi()` e guarda esse número na variável `pressao[i]`, repetindo o processo 120 vezes até preencher a variável `pressao[120]`. Esta parte do código está apresentada na Figura 31.

```
673     if (sensor == 99){
674             for (int i = 0; i < sizeof(pressao) / sizeof(pressao[0]); i++) {
675                     // Convert the three-character substrings to integers
676                     char numStr[4];
677                     strncpy(numStr, rxBuffer2 + 2 + i * 3, 3);
678                     numStr[3] = '\0';
679                     pressao[i] = atoi(numStr);
680             }
681     pressao[120] = '\0'; // Null-terminate the string
682             }
```

Figura 31 – Processo de conversão de valor de pressão hidráulica

Assim, todas as variáveis recebem os valores das grandezas medidas. A seguir, o programa insere esses valores nas variáveis especificas do *address space* do servidor OPC UA, ficando disponíveis para serem partilhadas com os clientes a ele conectados. Na Figura 32 é apresentada a função de atribuição de valor de temperatura à variável correspondente do servidor.

```
1340    UA_StatusCode readCurrentTemperature7(UA_Server *server,
1341                    const UA_NodeId *sessionId, void *sessionContext,
1342                    const UA_NodeId *nodeId, void *nodeContext,
1343                    UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
1344                    UA_DataValue *dataValue) {
1345        UA_Float temperature7 = temper7;
1346        UA_Variant_setScalarCopy(&dataValue->value, &temperature7,
1347                                 &UA_TYPES[UA_TYPES_FLOAT]);
1348        dataValue->hasValue = true;
1349        return UA_STATUSCODE_GOOD;
1350    }
```

Figura 32 – Atribuição de valor de temperatura a variável de OPC UA servidor

A função recebe vários parâmetros , incluindo um ponteiro para o servidor OPC UA (`UA_Server *server`), um ponteiro para o contexto de sessão (`void *sessionContext`), um ponteiro para o ID do nó (`const UA_NodeId *nodeId`),

um ponteiro para o contexto do nó (`void *nodeContext`), um booleano que indica que deve usar o tempo da fonte (`UA_Boolean sourceTimeStamp`), uma estrutura que representa um intervalo numérico (`const UA_NumericRange *range`) e um ponteiro para o valor de dados (`UA_DataValue *dataValue`) que será preenchido com a temperatura lida.

Dentro da função o valor da temperatura é atribuído ao campo `value` da estrutura `UA_DataValue` usando a função `UA_Variant_setScalarCopy`. O tipo de dados é definido como `UA_TYPES_FLOAT` para indicar que se trata de um valor de vírgula flutuante. O campo `hasValue` da estrutura `UA_DataValue` é definido como `true` para indicar que o valor está disponível. Finalmente, a função retorna `UA_STATUSCODE_GOOD` para indicar que a leitura da temperatura foi bem-sucedida.

Depois do valor de temperatura armazenado na variável padrão de OPC UA, esse valor pode ser adicionado ao servidor OPC UA e, assim, tornar visível para um cliente. Na Figura 33 apresentada função que adiciona variável OPC UA ao servidor OPC UA.

```
1352  void addCurrentTemperature7DataSourceVariable(UA_Server *server) {
1353      UA_VariableAttributes attr = UA_VariableAttributes_default;
1354      attr.displayName = UA_LOCALIZEDTEXT("en-US", "Temperature7");
1355      attr.dataType = UA_TYPES[UA_TYPES_FLOAT].typeId;
1356      attr.accessLevel = UA_ACCESSLEVELMASK_READ;
1357      UA_NodeId currentNodeId = UA_NODEID_STRING(1, "temperature7");
1358      UA_QualifiedName currentName = UA_QUALIFIEDNAME(1, "Temperature7");
1359      UA_NodeId parentNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER);
1360      UA_NodeId parentReferenceNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_ORGANIZES);
1361      UA_NodeId variableTypeNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_BASEDATAVARIABLETYPE);
1362      UA_DataSource timeDataSource;
1363      timeDataSource.read = readCurrentTemperature7;
1364      UA_Server_addDataSourceVariableNode(server, currentNodeId, parentNodeId,
1365                                          parentReferenceNodeId, currentName,
1366                                          variableTypeNodeId, attr,
1367                                          timeDataSource, NULL, NULL);
1368  }
```

Figura 33 – Função de atribuição de variável ao servidor OPC UA

Na primeira linha dessa função são definidos os atributos padrão da variável que depois podem ser modificados conforme necessário. Nas linhas seguintes os atributos da variável são configurados:

- O nome de exibição é definido como “Temperature7”;
- O tipo de dados é configurado como `UA_TYPES_FLOAT`, que representa números de vírgula flutuante;
- O nível de acesso é configurado para leitura apenas.

Na seguinte parte de código são criados identificadores de nós que serão usados ao adicionar a variável ao servidor OPC UA. Por exemplo, `currentNodeId` representa o ID da variável, `parentNodeId` representa o nó superior onde a variável será adicionada, etc. A seguir, uma fonte de dados é configurada para a variável do servidor. Neste caso, a função `readCurrentTemperature7` será usada para ler o valor da temperatura. Finalmente, a variável com o valor de temperatura armazenado é adicionada ao servidor OPC UA utilizando a função `UA_Server_addDataSourceVariableNode`. Os argumentos fornecidos incluem os identificadores de nós, os atributos da variável, a fonte de dados e outros parâmetros necessários.

3.8. Testes intermédios

Antes de elaborar o módulo de monitorização de energia elétrica foi decidido realizar os testes intermédios para avaliar o funcionamento de servidor OPC UA com base em Raspberry Pi 4. Para que os testes a realizar fossem completos foi decidido experimentar o funcionamento em três plataformas independentes. Assim, foram instalados três tipos de cliente OPC UA: um no mesmo Raspberry Pi 4 com SO Debian, no PC (*Personal Computer*) com SO Windows e no smartphone com SO Android.

O cliente usado para o teste no Raspberry Pi 4 foi o que está disponível em https://github.com/FreeOpcUa/opcua-client-gui e está escrito em língua de programação Python. Por isso, antes de o instalar é necessário instalar o ambiente de programação Python, o que inclui o interpretador Python, as bibliotecas padrão do Python e outras bibliotecas de terceiros que podem ser facilmente instaladas usando o gestor de pacotes `pip`. Antes de instalar o Python é necessário instalar o Git package, uma ferramenta de controlo de versões de software distribuído. Para isso, no Raspberry Pi 4 basta abrir uma janela de terminal e introduzir o comando `sudo apt-get install git`. A seguir, no mesmo terminal introduzir comando `sudo apt-get install python3-pip`, que faz a instalação da versão 3 do Python. O passo seguinte é copiar repositório do cliente OPC UA a partir do GitHub. Para isso, no mesmo terminal, é necessário introduzir comando:

`git clone https://github.com/FreeOpcUa/python-opcua-client.git`.

A seguir, utilizando comando `cd python-opcua-client`, entrar na pasta python-opcua-client. Finalmente, utilizando o comando `sudo python3 setup.py install` instalar o cliente OPC UA no Raspberry Pi 4.

No smartphone com SO Android foi instalado o cliente OPC UA da PROSYS OPC Ltd[25]. Quanto ao PC, foi usado o cliente UaExpert da UnifiedAutomation que já tinha sido referido atrás neste relatório. Com os três clientes OPC UA instalados nas plataformas diferentes é possível realizar testes de funcionamento e estabilidade de funcionamento de servidor OPC UA baseado na Raspberry Pi 4. Primeiro, servidor OPC UA foi testado com o cliente OPC UA local, ou seja, instalado no mesmo dispositivo que o servidor. Na Figura 34 é apresentado a sua interface. Como servidor e cliente estão no mesmo dispositivo, no campo de endereço do servidor está introduzido `opc.tcp://localhost:4840`. Onde `opc.tcp` especifica o protocolo de comunicação utilizado.

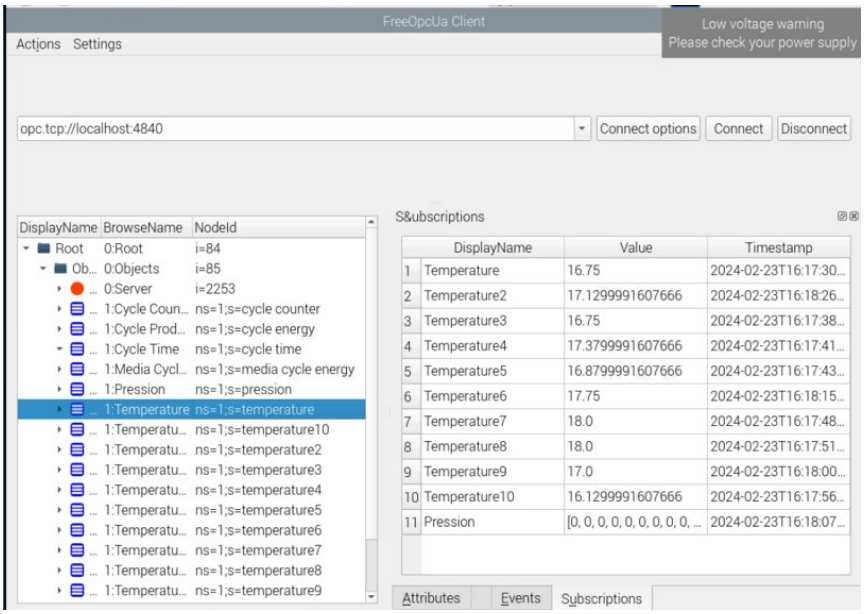

Figura 34 – OPC UA cliente no Raspberry Pi 4

O funcionamento do servidor a par do cliente local mostrou-se estável, sem atrasos de resposta e sem falhas de qualquer tipo. O arranque do servidor demora alguns segundos e a aquisição dos primeiros dados dos sensores demora mais alguns segundos. Os dados dos sensores todos estão visíveis no cliente OPC UA em menos de um minuto.

Com o cliente UaExpert da Unified Automation instalado no PC e conectado ao servidor por meio da rede Wi-Fi, para adicionar novo servidor OPC UA ao cliente, basta introduzir o respetivo endereço IP e atribuir o nome ao servidor na página correspondente das configurações. O cliente UaExpert demonstrou as mesmas características de funcionamento, ou seja, altamente robusto e estável, sem atrasos de resposta e sem falhas de qualquer tipo. Os tempos de arranque do servidor e aquisição dos primeiros dados dos sensores

[25] https://prosysopc.com/products/opc-ua-client-for-android/ consultado em outubro de 2023

demonstraram um comportamento semelhante ao anterior. Na Figura 35 mostra-se a interface do cliente OPC UA UaExpert instalado no PC com SO Windows.

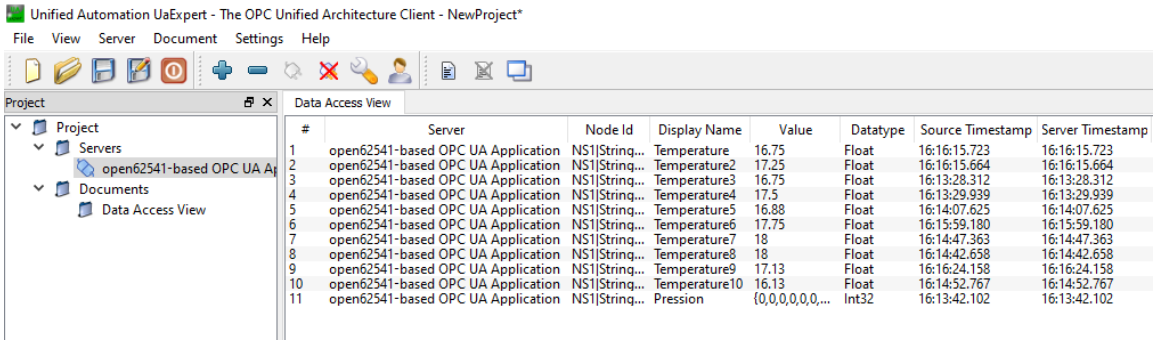

Figura 35 - OPC UA cliente UaExpert da Unified Automation

Finalmente, testou-se o acesso com o cliente da PROSYS OPC desenvolvido para dispositivos com SO Android. Para conectar o servidor OPC UA ao cliente é necessário conectar o smartphone à mesma rede Wi-Fi a que está ligado o servidor. Na aplicação, é necessário realizar o mesmo procedimento: introduzir o endereço IP do servidor e definir o nome. Como nos casos anteriores o funcionamento foi robusto e estável, de novo sem atrasos de resposta e sem falhas de qualquer tipo e com um comportamento temporal semelhante. Na Figura 36 apresenta-se a interface do cliente OPC UA da PROSYS OPC instalado no smartphone com SO Android.

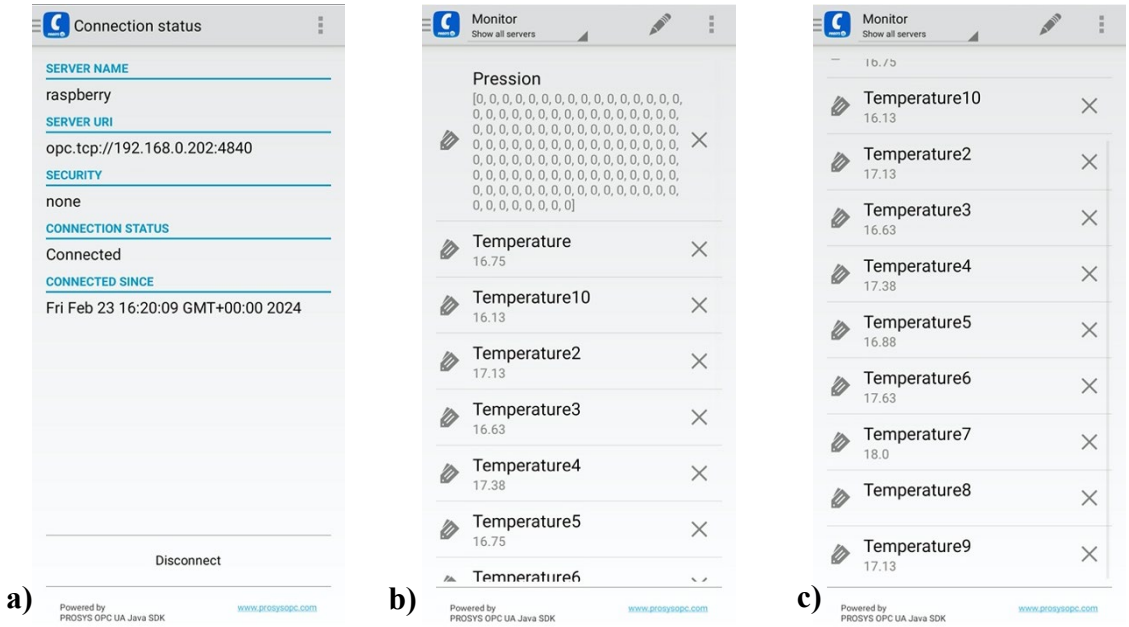

Figura 36 – OPC UA cliente da PROSYS OPC

Esses três testes validaram a solução, que se expandiu para o desenvolvimento do módulo de monitorização de energia elétrica.

3.9. Desenvolvimento do módulo de monitorização de energia elétrica

Um dos aspetos de funcionamento do módulo de monitorização de energia elétrica envolve a medição de corrente e tensão em tempo real. Isso é feito por meio dos sensores de corrente e tensão que estão conectados ao circuito elétrico que está monitorizado. Como o circuito da máquina é um circuito trifásico de corrente alternada, para a realização dessa parte do trabalho é necessário usar três sensores de corrente e três sensores de tensão. O consumo máximo total de máquina com fator de simultaneidade 1, é cerca de 81.4kW, isso é cerca de 151.35A por fase. Por isso como sensor de corrente foi escolhido o sensor não invasivo SCT-019 com uma corrente de medição nominal de 200A.

Tabela 1 – Consumo de energia elétrica pela máquina de injeção

	Potência total nominal[kW]	cos φ	Corrente por fase [A]
Motor 1	22.00	0.87	43.00
Motor 2	30.00	0.87	57.00
Aquecimento Câmara	26.40	1.00	38.26
Aquecimento Bico	1.00	1.00	4.35
Aquecimento Molde	2.00	1.00	8.70
Corrente total por fase [A]			151.35

Na Tabela 1 são apresentados os valores de consumo de energia elétrica pelas várias partes da máquina. Valores dos motores foram retirados das respetivas chapas de características técnicas. O aquecimento da câmara inclui 12 resistências de 2.2kW @230V AC (*Alternating Current*), cuja ligação está distribuída pelas três fases, resultando em 8.8kW ou 38.26A por fase. Também, está incluída a resistência do bico da máquina e o aquecimento máximo possível do molde. Não está incluído o consumo da parte de comando, feito a partir de um transformador 230/110V, compreendendo contactores, válvulas hidráulicas e pneumáticas. Também, não está incluído o equipamento auxiliar ocasionalmente ligado nas tomadas trifásicas e monofásicas da máquina.

Figura 37 – Sensor de corrente SCT-019[26]

Na Figura 37 está apresentado o sensor de corrente SCT-019. De acordo com o *datasheet* [48] do sensor, a corrente de 200A registada pelo sensor corresponde à tensão de 0.333V na saída de sensor. Como o módulo Arduino Pro Mini utilizado para elaborar o módulo de monitorização de energia elétrica opera com tensão de 5V DC há necessidade de adaptar tensão presente na saída do sensor à tensão adequada para Arduino Pro Mini. Para isso pode ser utilizado o módulo amplificador baseado no circuito integrado de amplificador operacional duplo LM358. Esse módulo pode ser afinado para amplificação até 100 vezes.

O circuito elétrico da máquina é de corrente alterna com a frequência da rede (50Hz). Por isso, na saída do sensor aparece uma onda sinusoidal de 50Hz com picos de -0.333V a +0.333V, ou seja, com tensão entre os picos de 0.666V. Amplificação de 0.666V para 5V significa uma amplificação de 5/0.666=7.51 vezes. Para atingir esse valor de amplificação é necessário afinar o respetivo potenciómetro que faz parte do circuito do módulo. A afinação dos três módulos amplificadores foi feita no laboratório da escola por meio de um gerador de sinal sinusoidal com frequência de 50Hz e osciloscópio. Na Figura 38 apresentado o módulo amplificador baseado no amplificador operacional duplo LM358. Como a alimentação do módulo é 0 e 5V DC, ou seja, não tem tensão negativa na alimentação, na saída do amplificador o sinal tem um offset de 2.5V. Isso significa que, na entrada do amplificador está presente o sinal sinusoidal com frequência de 50Hz e amplitude de -0.333V a +0.333V @200A e, na saída obtém-se um sinal sinusoidal com a mesma frequência mas com uma amplitude de 0 a 5V e com o ponto de referência a 2.5V que pode ser lida pelo conversor analógico-digital do Arduíno Pro Mini e interpretado como valor instantâneo da corrente de fase.

[26] https://robolabor.ee/en/sensors-and-accessories/1016-ac-current-sensor-sct019-split-core-current-transformer-0-200a-33ma.html consultado em outubro de 2023

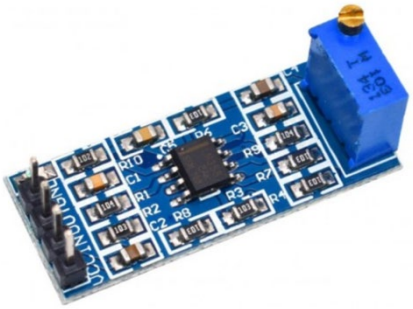

Figura 38 – Módulo amplificador baseado no LM358[27]

Outra grandeza física a ser monitorizada para poder obter o valor da energia elétrica consumida pela máquina é a tensão. Como sensor de tensão utilizou-se um módulo para medição de tensão alternada nominal de 230V e frequência 50Hz baseado no transformador ZMPT101B e no amplificador operacional LM358. O sensor de tensão utilizado é apresentado na Figura 39.

Figura 39 – Sensor de tensão ZMPT101B[28]

De acordo com o *datasheet* [49] do transformador, a corrente nominal na entrada e na saída do transformador é 2mA, em conjunto com a resistência de amostragem de 100Ω, obtém-se uma tensão alternada com os picos de:

$$V_{pico} = \frac{V_{RMS}}{0.707} = \frac{\pm 0.2}{0.707} = \pm 0.283\text{V} \quad (1)$$

Aqui, V_{pico} é o valor de pico da tensão e V_{RMS} é o valor RMS (*Root Mean Square*) da tensão [50]. O amplificador LM358 efetua amplificação de tensão na saída do transformador e impõe-lhe um offset, conformando o sinal para a ligação ao conversor do módulo Arduino Pro Mini e interpretada como a tensão atual no condutor.

[27] https://www.botnroll.com/pt/conversores-isoladores/3925-m-dulo-lm358-amplificador-at-100x.html consultado em outubro de 2023
[28] https://www.botnroll.com/pt/outros/4301-sensor-de-tens-o-230vac-para-arduino.html consultado em outubro de 2023

Na utilização do conjunto dos sensores de corrente e de tensão para calcular a potência, há que ter o cuidado de sentido correto de posição de sensor de corrente para obter resultados fiáveis.

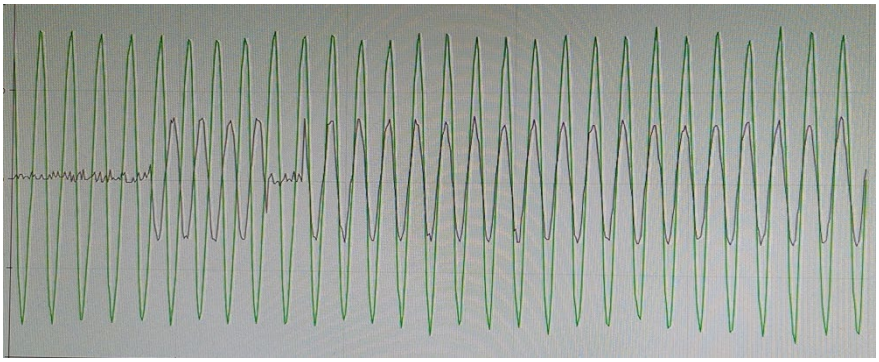

Figura 40 – Sentido de sensor de corrente invertido e correto

Na Figura 40 são apresentados os gráficos de tensão e corrente, desenhados pela ferramenta interna do Arduino IDE (Arduino *Serial Plotter*). Este exemplo demonstra a montagem do sensor de corrente na posição invertida e na posição correta. Na posição invertida a onda de corrente (a preto) tem um desfasamento de 180º em relação a onda de tensão (a verde). Na posição correta a onda de corrente está em fase com a onda de tensão. Isso corresponde a um fator de potência igual a 1, o que é verdadeiro, pois, trata-se de um teste com carga puramente resistiva.

A componente de software do módulo de monitorização de energia elétrica engloba um conjunto de funções destinadas ao tratamento dos dados recolhidos pelos sensores de corrente e tensão. As funções empregues neste módulo são:

- `int findPeakValue (float* samples, int numSamples)` – esta função é utilizada para encontrar o valor de pico num *array* de amostras. A função inicializa a variável `peakValue` com o primeiro elemento do *array* `samples`. A seguir, a função percorre o *array* de amostras a partir do segundo até ao último elemento. Para cada elemento no *array*, a função compara o valor da amostra com o valor atual de `peakValue` e, se o valor de amostra atual for maior do que o valor atual de `peakValue`, atualiza `peakValue` com o valor da amostra atual. Após percorrer todas as amostras, a função retorna o valor máximo encontrado no *array*, ou seja, o valor de pico.

$$samples = [s0, s1, s2, \dots, s_{n-1}] \quad (2)$$

$$peakValue = \max(s0, s1, s2, \dots, s_{n-1})$$

Onde $samples$ é uma série temporal de amostras, sN é o número de amostra e o $peakValue$ é o valor de pico encontrado na série temporal de amostras;

- `int findPeakIndex(float* samples, int numSamples, int peakValue)` – esta função tem o objetivo de encontrar o índice em que ocorre o valor de pico num *array* de amostras. A função percorre o *array* de amostras elemento por elemento usando um loop `for`. Para cada elemento ela verifica se o valor da amostra é igual ao `peakValue`. Se encontrar um valor igual, significa que encontrou o pico, então retorna o índice atual `i`. Se o pico não foi encontrado, a função retorna `-1` para indicar que o pico não foi encontrado;
- `float calculatePhaseAngle(float* voltageSamples, float* currenteSamples, int numSamples, int samplingPeriod, float frequency)` – esta função calcula o ângulo de fase em radianos. O primeiro passo é encontrar os valores de picos de tensão e corrente nos *arrays* correspondentes usando a função `findPeakValue`. A seguir, encontrar os índices dos valores de pico para tensão e corrente usando função `findPeakIndex`. A seguir, calcula o deslocamento de tempo entre os picos subtraindo o índice do pico da corrente do índice do pico da tensão e multiplicar pelo período de amostragem. A seguir, calcula o ângulo da fase em radianos, multiplicando o deslocamento de tempo pela frequência de sinal de entrada e pelo valor de 2π. Na Figura 41 mostra-se o desfasamento entre tensão e corrente de 0º e 45º, o que corresponde a carga totalmente resistiva e fator de potência 1 e carga indutiva e fator de potência 0.707. A Expressão matemática de cálculo do ângulo de desfasamento entre a onda sinusoidal de tensão e de corrente é apresentada na (3). Aqui, t_{shift} é o tempo entre o pico de tensão e o pico de corrente, $t_{index\_}Vpeak$ é o número da amostra em que ocorre o pico de tensão, $t_{index\_}Ipeak$ é o número da amostra onde ocorre o pico de corrente, $samplingPeriod$ é o tempo entre as amostras em µs, $\phi_{radianos}$ é o ângulo em radianos e f é a frequência de onda em Hz.

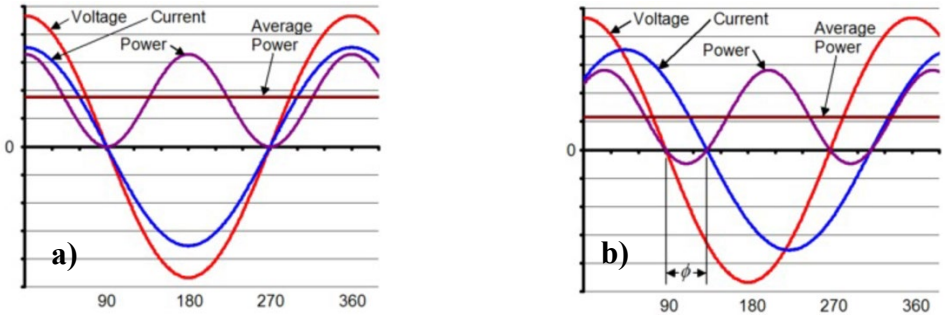

Figura 41 - Desfasamento entre tensão e corrente de 0º (a) e 45º (b)[29]

$$t_{shift} = (t_{index\_}Vpeak - t_{index\_}Ipeak\ )\frac{samplingPeriod}{1000000} \quad (3)$$

$$\phi_{radianos} = 2\pi f t_{shift}$$

O cálculo da energia elétrica consumida pela máquina é realizado em várias etapas. A primeira etapa é a aquisição das amostras de tensão e corrente das três fases. Para cada fase o programa guarda 200 valores de tensão e de corrente, efetuando leituras a cada 100µs. Para 200 valores da aquisição isso dá 0,0001x200=0,02s, o que, para frequência da rede de 50Hz, corresponde a 1 ciclo da onda sinusoidal. A quantidade de amostras por aquisição influencia a precisão de medição e o valor final de calculo e é limitada pela memória interna disponível no módulo Arduino Pro Mini. Na Figura 42 apresenta-se a aquisição de amostras de corrente e tensão das três fases.

```
279  for (int i = 0; i < numSamples; i++) {
280    voltageSamplesR[i] =  analogRead(TensRpin);
281    currentSamplesR[i] =  analogRead(CorRpin);
282    voltageSamplesS[i] =  analogRead(TensSpin);
283    currentSamplesS[i] =  analogRead(CorSpin);
284    voltageSamplesT[i] =  analogRead(TensTpin);
285    currentSamplesT[i] =  analogRead(CorTpin);
286    delayMicroseconds(samplingPeriod);
287  }
```

Figura 42 – Aquisição de amostras de tensão e corrente

A seguir, o programa percorre cada *array* e encontra os valores dos picos e posições dos picos respetivos. Com estes valores calcula o ângulo de desfasamento entre tensão e corrente em cada fase. Com valores dos picos de tensão e corrente calcula valores RMS de tensão e corrente de cada fase. A expressão matemática de calculo de RMS para a onda sinusoidal é apresentada na (4), onde $rmsValue$ é o valor RMS calculado e $peakValue$ é o valor de pico encontrado numa série temporal das amostras [50].

[29] U.S. Department of Energy, "Better Plants. Power Factor."

$$rmsValue = peakValue \times 0.7071 \quad (4)$$

Com os valores de RMS de tensão e corrente e ângulo de desfasamento entre tensão e corrente calcula-se a potência de cada fase. Com a potência de cada fase e tempo de ciclo calcula-se a energia consumida pela máquina durante o ciclo de aquisição de amostras. A soma dos valores de energia consumidos no final de cada ciclo permite obter o valor de energia total consumida durante o funcionamento da máquina em tempo real. Além disso, o módulo de monitorização de energia elétrica efetua a contagem de ciclos de produção e tempo de um ciclo de produção de uma peça. Na Figura 43 apresentado o diagrama de cálculo de energia elétrica.

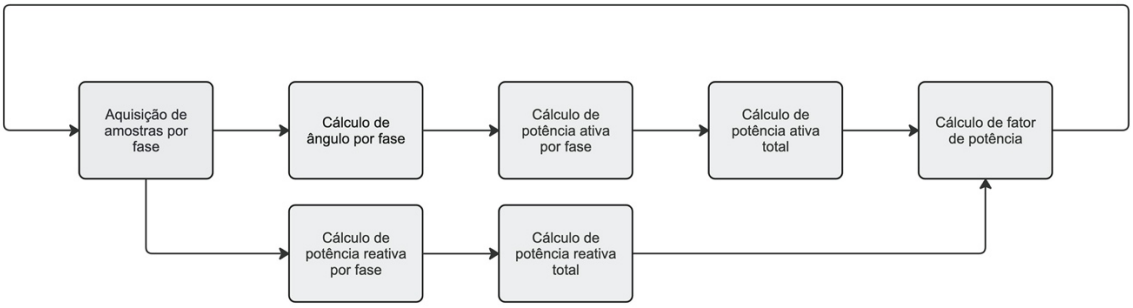

Figura 43 – Diagrama do cálculo da energia elétrica

No processo de comunicação com o servidor, módulo Arduino Pro Mini adiciona aos valores comunicados prefixos para distinguir os dados. Assim, o módulo envia para o Raspberry Pi 4 os dados seguintes:

- número de ciclo com o prefixo 98. Exemplo: 98153;
- energia elétrica consumida durante um ciclo com o prefixo 97. Exemplo: 970.357;
- energia elétrica consumida durante a produção com o prefixo 96. Exemplo: 9612.423;
- tempo de ciclo com o prefixo 95. Exemplo: 9523.15.

A partir do número de ciclos e da energia elétrica consumida durante a produção, o programa no servidor calcula o consumo médio de energia elétrica durante um ciclo. À semelhança dos valores de temperatura, o servidor OPC UA armazena os valores relacionados com o cálculo de energia elétrica nas variáveis OPC UA e adiciona ao servidor.

3.10. Montagem de analisador de energia elétrica na máquina

A montagem de sensores na máquina foi feita da seguinte maneira: os sensores de corrente foram colocados nas linhas de alimentação geral da máquina, os sensores de tensão foram conectados às respetivas linhas de alimentação geral. As saídas dos sensores de corrente são ligadas às entradas dos módulos de amplificação de sinal e as saídas desses módulos conectadas aos pinos ADC do Arduino Pro Mini. As saídas dos sensores de tensão também estão ligadas aos pinos ADC respetivos do Arduino Pro Mini. Os pinos analógicos de Arduino Pro Mini usados na monitorização de energia elétrica são A0 e A1 – sensor de tensão e sensor de corrente para fase R, A2 e A3 – sensor de tensão e sensor de corrente para fase S e A4 e A5 – sensor de tensão e sensor de corrente para fase T. O esquema elétrico da montagem dos sensores na máquina de injeção está apresentado na Figura 44. Aqui, o contacto ligado ao pino 2 do Arduino Pro Mini pertence ao relé auxiliar que regista inicio e fim de ciclo de produção de uma peça.

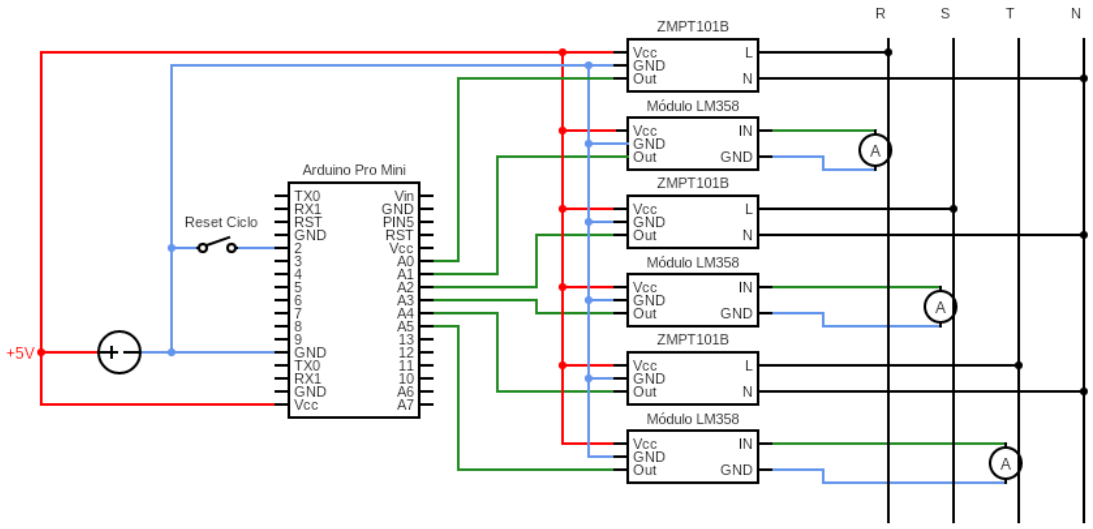

Figura 44 - Montagem dos sensores de analisador de energia elétrica na máquina

Testes feitos durante a desenvolvimento de módulo demonstraram a precisão e sensibilidade de módulo na medição de consumos baixos e elevados. Um dos testes foi feito na máquina, na medição de corrente, potência e fator de potência de uma resistência de bico da máquina. De acordo com a Tabela 1, a potência nominal dessa resistência é 1kW e a corrente nominal @230V AC é cerca de 4.35A.

```
Tensao R : 227
Corrente R : 4.37
Angle R: 0.00
Tensao S : 231
Corrente S : 0.00
Angle S: 0.00
Tensao T : 231
Corrente T : 0.73
Angle T: 0.74
Apparent Power : 1161.60
Power : 1117.49
Power Factor: 0.96
```

Figura 45 - Resultados de teste com resistência de aquecimento

Na Figura 45 são apresentados os dados obtidos durante o teste de módulo de monitorização de energia elétrica e visualizados através de *Serial Monitor* – ferramenta interna de Arduino IDE. Esse teste demonstrou que os dados obtidos de módulo são muito próximos dos esperados. Dos dados obtidos pelo módulo de monitorização de energia elétrica podemos ver que a resistência conectada a fase R consome 4.37A, o que @227V AC dá:

$$I * V * \cos(\phi) = 4.37 * 227 * 1 = 992W \quad (5)$$

Também, podemos ver na fase T conectado mais um consumidor de energia elétrica que corresponde ao transformador geral de comando. Este transformador consome cerca de 0.73A com um ângulo de desfasamento entre a tensão e corrente de cerca de 0.74 radianos o que significa fator de potência cerca de 0.738 e a potência ativa:

$$I * V * \cos(\phi) = 0.73 * 231 * 0.738 = 124W \quad (6)$$

A utilização total de energia elétrica pela máquina é cerca de 1117.5W com fator de potência cerca de 0.96 [51].

O teste seguinte consiste na medição de parâmetros de utilização de energia elétrica pela máquina com dois motores das bombas hidráulicas ligados.

Tabela 2 - Características técnicas dos motores

Motor	Tensão nominal [V]	Corrente nominal [A]	Potência nominal [kW]	cos(ϕ)
1	380	43	22	0.87
2	380	57	30	0.87

Na Tabela 2 são apresentadas as características técnicas dos motores elétricos das bombas hidráulicas. Dessas características, podemos ver que a potência nominal de um motor é de 22kW e do outro de 30kW, e ambos com o cos(ϕ) é de 0.87. Na Figura 46 são apresentados resultados do teste com dois motores elétricos de alta potência visualizados através de *Serial Monitor* de Arduino IDE. O fluxograma de cálculo para esse teste é apresentado na Figura 43. Dos resultados de teste podemos observar que a distribuição de corrente nas três fases da máquina está ligeiramente desequilibrado. O desfasamento entre tensão e corrente é cerca de 0.60 radianos em cada fase. A potência aparente da máquina é de 43.162kVA e a potência ativa de 35.278kW, de onde vem o fator de potência da máquina de 0.82.

```
Tensao R : 227
Corrente R : 65.72
Angle R: 0.63
Tensao S : 231
Corrente S : 62.11
Angle S: 0.60
Tensao T : 231
Corrente T : 60.16
Angle T: 0.61
Apparent Power : 43162.07
Power : 35277.85
Power Factor: 0.82
```

Figura 46 – Resultados do teste com dois motores elétricos da máquina

Assim, podemos concluir que a potência utilizada pelos dois motores está abaixo da potência nominal, uma vez que o ensaio foi feito com os motores a trabalhar em vazio. Além disso, o fator de potência também está mais baixo de que o nominal. Em carga nula, um motor consome uma corrente de magnetização grande e uma pequena componente ativa para suportar as perdas de funcionamento em vazio. Isso significa é que o motor de indução consome corrente em vazio, atrasada em relação à tensão aplicada por um ângulo grande. O rendimento dos motores durante o funcionamento em vazio é ligeiramente mais baixo de que durante o funcionamento à carga nominal [52].

3.11. Base de dados SQLITE

Para análise posterior dos dados de produção foi decidido criar a base de dados SQLITE[30] e armazenar nela todos os dados obtidos no sistema de aquisição. Para implementar a base de dados SQLITE a partir do servidor OPC UA no Raspberry Pi 4 foi utilizada biblioteca `sqlite3.h`. A biblioteca `sqlite3.h` é a interface de programação de aplicações (API) C usada para efetuar acesso e manipular bases de dados SQLITE em aplicações C e C++.

[30] https://www.sqlite.org/ consultado em novembro de 2023

Esta biblioteca fornece uma variedade de funções e estruturas de dados que permitem criar, abrir, consultar e modificar bases de dados SQLITE a nível de programação.

As funcionalidades principais fornecidas pela biblioteca `sqlite3.h` são:

- Criação e abertura de bases de dados – a função `sqlite3_open()` é usada para abrir ou criar uma base de dados SQLITE. Se a base de dados já existe, ela será aberta, se não existe – será criada;
- Executa consultas SQL (*Structured Query Language*) – a função `sqlite3_exec()` permite executar instruções SQL, como SELECT, INSERT, UPDATE e DELETE, diretamente do código C ou C++;
- Preparação de consultas SQL – a função `sqlite_prepare_v2()` é usada para preparar consultas SQL para execução posterior. Ela compila a consulta SQL em formato intermediário que pode ser executado repetidamente com diferentes parâmetros;
- Leitura e gravação de dados – as funções `sqlite_step()` e `sqlite_column_*()` são usadas para executar consultas preparadas e ler os resultados. Elas permitem iterar sobre os resultados de uma consulta e ter acesso aos valores retornados pelas colunas;
- Manipulações de transações – a biblioteca `sqlite3.h` oferece suporte a transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade), e permite iniciar, confirmar ou reverter transações na base de dados;
- Gestão de erros – a API `sqlite3.h` fornece funções para lidar com erros e mensagens de status retornadas pelas operações SQLITE e permite que os programadores identifiquem e resolvam problemas de forma adequada.

A primeira função a ser utilizada na manipulação com a base de dados é a função de inicialização de base de dados `int initializeDatabase()`. Esta função cria ou abre a base de dados com o nome IMI1300 utilizando a função `sqlite3_open("IMI1300.db",&db)`. O passo seguinte é a criação, se não existem, das tabelas onde vão ser escritos e armazenados os dados das variáveis. Na Figura 47 é apresentada a criação de uma tabela, onde as variáveis de temperatura, tempo de ciclo, energia de ciclo e energia total são armazenadas como número real de vírgula flutuante, a

variável de contador de ciclos como número inteiro, a pressão como *string* e o *timestamp* em formato de data/hora.

```
150     // Create tables if they don't exist
151     const char *createTableSQL = "CREATE TABLE IF NOT EXISTS operating_data ("
152                                  "id INTEGER PRIMARY KEY AUTOINCREMENT,"
153                                  "temperature1 REAL,"
154                                  "temperature2 REAL,"
155                                  "temperature3 REAL,"
156                                  "temperature4 REAL,"
157                                  "temperature5 REAL,"
158                                  "temperature6 REAL,"
159                                  "temperature7 REAL,"
160                                  "temperature8 REAL,"
161                                  "temperature9 REAL,"
162                                  "temperature10 REAL,"
163                                  "cycle_counter INTEGER,"
164                                  "cycle_time REAL,"
165                                  "cycle_energy REAL,"
166                                  "total_energy REAL,"
167                                  "pression TEXT,"
168                                  "timestamp DATETIME DEFAULT CURRENT_TIMESTAMP"
169                                  ");";
```

Figura 47 – Criação de tabelas de base de dados SQLITE

A função `int insertData(sqlite3 *db)` serve para inserir dados em bases de dados. Um dos primeiros passos nessa função é a declaração de SQL para inserir dados em tabela – `const char *sql = "INSERT INTO operating_data (temperature1, temperature2, temperature3, temperature4, temperature5, temperature6, temperature7, temperature8, temperature9, temperature10, cycle_counter, cycle_time, cycle_energy, total_energy, pression) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);`. Outro passo necessário é a converter o *array* de valores de variável `pression` em *string* de texto, essa operação está apresentada na Figura 48.

```
206     for (int i = 0; i < 120; ++i) {
207         char numStr[10];  // Adjust the size as needed
208         sprintf(numStr, "%d", pressao[i]);  // Convert int to string
209         strcat(buffer, numStr);             // Concatenate the string
210         if (i < 119) {
211             strcat(buffer, ",");  // Add a comma between values
212         }
213     }
```

Figura 48 – Conversação de *array* em a *string*

O passo seguinte é a inserção de dados na tabela de base de dados SQLITE, apresentada na Figura 49.

```
215     sqlite3_bind_double(stmt, 1, temper1);
216     sqlite3_bind_double(stmt, 2, temper2);
217     sqlite3_bind_double(stmt, 3, temper3);
218     sqlite3_bind_double(stmt, 4, temper4);
219     sqlite3_bind_double(stmt, 5, temper5);
220     sqlite3_bind_double(stmt, 6, temper6);
221     sqlite3_bind_double(stmt, 7, temper7);
222     sqlite3_bind_double(stmt, 8, temper8);
223     sqlite3_bind_double(stmt, 9, temper9);
224     sqlite3_bind_double(stmt, 10, temper10);
225     sqlite3_bind_double(stmt, 11, counter);
226     sqlite3_bind_double(stmt, 12, cicloTempo);
227     sqlite3_bind_double(stmt, 13, cicloEnergy);
228     sqlite3_bind_double(stmt, 14, totalEnergy);
229     sqlite3_bind_text(stmt, 15, buffer, -1, SQLITE_STATIC);
```

Figura 49 - Inserção de dados na tabela

Nesse passo a função `sqlite3_bind_double` é usada para associar um valor double a um parâmetro em uma instrução SQL preparada. Os parâmetros são identificados por números, começando em 1 para o primeiro parâmetro, 2 para o segundo e assim por diante. Por exemplo, `sqlite3_bind_double(stmt, 1, temper1)` associa o valor da variável `temper1` ao primeiro parâmetro na instrução SQL preparada representada por `stmt`. Da mesma forma, `sqlite3_bind_text` é usada para associar um valor de texto (ou string) a um parâmetro numa instrução SQL preparada. A função `sqlite3_bind_text` tem a mesma semântica que `sqlite3_bind_double`, exceto que ela é usada para valores de texto. A função `sqlite3_bind_text` neste caso associa o valor contido no *buffer* de caracteres `buffer` ao 15º parâmetro na instrução SQL preparada representada por `stmt`. Agora, quando a instrução SQL preparada é executada, os parâmetros associados serão substituídos pelos valores correspondentes durante a execução da instrução. Isto permite a execução da mesma instrução SQL várias vezes com diferentes valores de parâmetros, melhorando a eficiência e segurança do seu código SQL.

Finalmente, o último passo desta função é a chamada de duas funções `sqlite3_finalize(stmt)` e `sqlite3_close(db)`, para finalizar e fechar a base de dados SQL utilizada. A função `sqlite3_finalize(stmt)` é usada para finalizar uma declaração preparada anteriormente. Após finalizar a declaração, ela já não pode voltar a ser usada para executar consultas. Isso é importante para libertar recursos e evitar fugas de memória. A função `sqlite3_close(db)` é usada para fechar a conexão com a base de dados SQLITE, libertando todos os recursos associados à conexão. O armazenamento de dados na base de dados SQLITE é realizado em todos os ciclos, sempre que houver alteração no valor do contador de ciclos – variável `counter`.

3.12. Cliente OPC UA personalizado baseado no Node-RED

Os clientes OPC UA utilizados para testar o funcionamento de servidor OPC UA elaborado, são ferramentas que dispõem da funcionalidade completa de interação entre servidor e cliente OPC UA. Por exemplo, o cliente OPC UA UaExpert, dispõe das funcionalidades seguintes:

- Explora os servidores OPC UA disponíveis na rede e visualiza a estrutura dos dados e serviços oferecidos por esses servidores;
- Efetua leitura e escrita de dados dos servidores OPC UA conectados, permitindo a interação com os dispositivos e sistemas controlados por esses servidores;
- Oferece recursos de monitorização em tempo real e permite visualizar e acompanhar as mudanças nos dados dos servidores OPC UA;
- Pode ser usado para testar e depurar sistemas OPC UA e ajuda a identificar e resolver problemas de comunicação ou configuração;
- Suporta recursos avançados do OPC UA, como segurança, autenticação, descoberta de servidores, etc.

Com todas as vantagens desses clientes, eles têm uma desvantagem o aspeto visual da interface do utilizador é pouco intuitivo, não dispõe de gráficos, diagramas, etc.

Para melhorar a apresentação visual dos dados disponibilizados pelo servidor OPC UA foi decidido criar um cliente OPC UA personalizado. Como solução pode ser utilizada uma plataforma de código aberto para programação visual de fluxo de dados, o Node-RED[31]. Ele fornece uma interface baseada na WEB para conectar dispositivos de hardware, APIs e serviços *online* de maneira simples e intuitiva, usando uma abordagem de arrastar e soltar blocos e ligações para criar fluxos de trabalho automatizados. Node-RED é amplamente utilizado em projetos de IoT, integração de sistemas e desenvolvimento de aplicações de *backend*. Ele é construído com base em `Node.js` e oferece uma grande variedade de *nodes* predefinidos que podem ser facilmente estendidos com a criação de *nodes* personalizados. Entre as funcionalidades pré-existentes encontra-se um cliente OPC-UA, que foi usado neste caso.

[31] https://nodered.org/ consultado em novembro de 2023

Para instalar o Node-RED no Raspberry Pi 4, primeiro, é necessário instalar o `Node.js` que é essencial para o seu funcionamento. Para instalar o `Node.js` basta abrir terminal e executar os comandos seguintes:

`curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -` e `sudo apt-get install -y nodejs`.

Após a instalação no mesmo terminal executar comando `sudo npm install -g --unsafe-perm node-red`,

Este comando instala o Node-RED no Raspberry Pi 4. Quando estiver tudo instalado, no mesmo terminal, executar o comando `node-red`, que inicia a execução do Node-RED no dispositivo. O acesso ao editor Node-RED pode ser feito através de um navegador WEB, inserindo o endereço do dispositivo e porta reservados pelo Node-RED (1880). Por exemplo, `http://localhost:1880`, se se aceder no mesmo dispositivo ou `http://<ip-do-Raspberry-Pi>:1880`, se se aceder de um dispositivo externo. Já nessa página de WEB, é possível construir a interface do cliente OPC UA customizado e garantir funcionamento desejado.

Para criar nodes personalizados é necessário instalar as paletas correspondentes. As paletas no Node-RED são os conjuntos de nós ou blocos pré-construídos que executam funções especificas. Cada nó numa paleta realiza uma função ou ação particular e pode ser arrastado e solto na interface gráfica do Node-RED para criar fluxos de dados e lógica de programação.

No caso presente, além de paletas básicas pré-instaladas e utilizadas na realização do trabalho como *Input*, *Function* e *Dashboard*, foram instaladas paletas OPC UA e ChartJs. As funcionalidades dessas paletas são:

- *Input* – oferece nós para entrada de dados, como HTTP (*Hypertext Transfer Protocol*), MQTT (*Message Queuing Telemetry Transport*), WebSocket, etc;
- *Function* – contém nós para realizar operações de processamento de dados usando JavaScript;
- *Dashboard* – contém nós para criar painéis de controlo e interfaces de utilizador personalizados;
- OPC UA – oferece nós que permitem a integração com sistemas que utilizam a norma OPC UA;

- ChartJs – é uma coleção de nós que permitem criar e exibir gráficos usando a biblioteca `Chart.js`.

Como foi referido anteriormente, a utilização de nós de Node-RED para construir a interface é realizada na página WEB. Na Figura 50 é apresentado o conjunto de nós que constituem o *dashboard* de interface produzida.

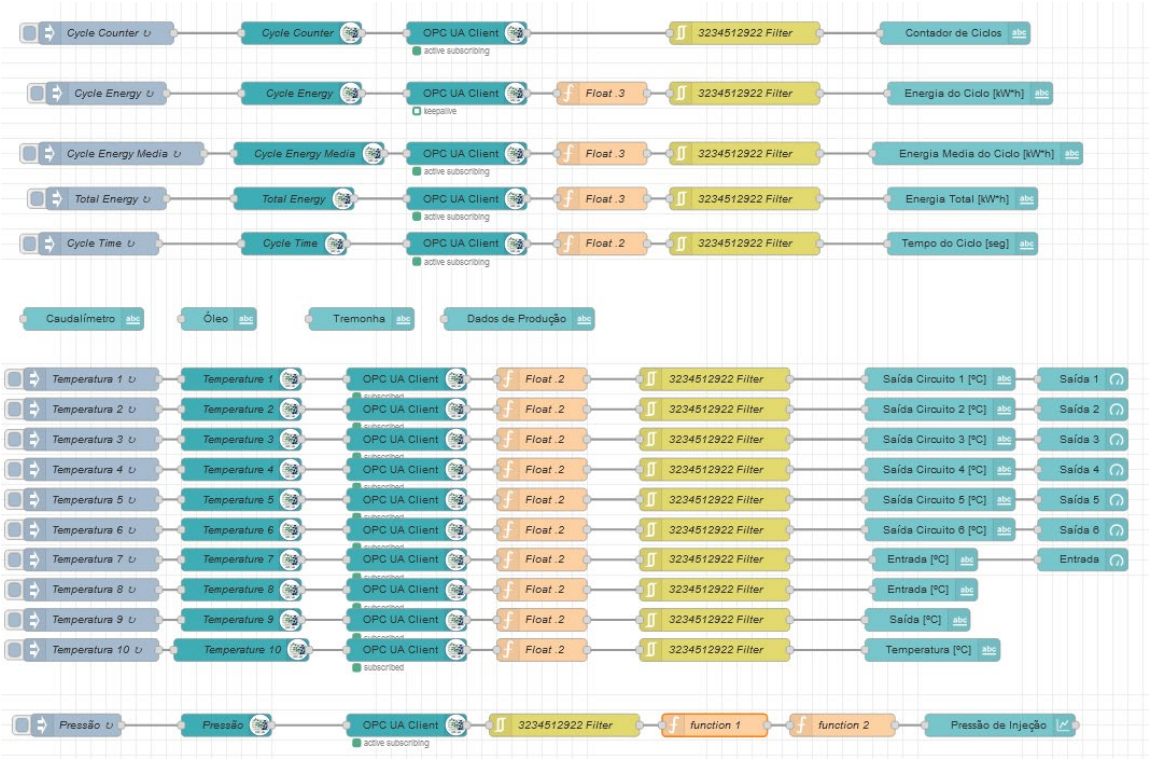

Figura 50 - Utilização de nós de Node-RED

Cada fluxo é constituído por vários nós conectados: nó *inject* da paleta *Input*, nós OpcUa-item e OpcUa-Client da paleta OPC UA, nós *function* e *filter* da paleta *Function*, nó *text* é o nó padrão que está disponível na interface de Node-RED, nó *gauge* da paleta *Dashboard* e o nó `Chart.js` da paleta `Chart.js`. Cada nó de fluxo executa uma função especifica e a combinação desses nós num fluxo permite realizar a automação de tarefas complexas.

O nó *inject* é usado para injetar mensagens num fluxo em momentos específicos ou a intervalos regulares. Ele fornece opções para configurar o tipo de mensagem a serem injetadas e o momento em que a mensagem é enviada para o fluxo.

O nó OpcUa-item é usado para comunicar com servidores OPC-UA. Ele permite definir um item específico num servidor OPC-UA para leitura ou escrita. Também permite configurar o endereço do servidor OPC-UA, o nome do item, o tipo de dados, etc. O nó OpcUa-Client

permite ler e escrever dados de variáveis de um servidor OPC UA, bem como inscrever-se para receber notificações de alterações nesses dados. Assim, essas funcionalidades são essenciais para integrar sistemas baseados em Node-RED com ambientes industriais que utilizam padrão OPC-UA para comunicação de dados em tempo real.

O nó *function* pode ser usado para executar operações de processamento de dados, como cálculos, transformações de dados e formatação de saída. Aqui, foi utilizado para formatar números de vírgula flutuante para uma precisão especifica de duas ou três casas decimais.

O nó *text* é usado para exibir texto estático ou dinâmico de mensagens num fluxo. Pode ser configurado para exibir texto estático para mensagens simples ou informações fixas ou texto dinâmico para visualizar dados que lhe estão a ser passados nas mensagens de fluxo. Assim, o nó *text* pode ser usado para visualizar dados de sensores, status de sistema, mensagens de erro e qualquer outro tipo de informação textual relevante para a aplicação.

O nó *gauge* é usado para criar e exibir medidores ou indicadores gráficos de valores numéricos num fluxo. Ele permite visualizar rapidamente a magnitude de um valor em relação a um intervalo específico. O medidor é atualizado automaticamente conforme os novos dados são recebidos em tempo real. Isso permite ver as mudanças à medida que ocorrem e facilita a compreensão e a interpretação de dados.

O nó `Chart.js` é usado para criar gráficos interativos usando a biblioteca JavaScript `Chart.js`. Ele permite visualizar dados numéricos em diferentes tipos de gráficos, como gráficos de linha, barras, radar, pizza, etc.

No sistema de fluxo completo, os nós de injeção de temperatura estão definidos para injetar as mensagens no fluxo de 20 em 20s. Os nós de Opcua-Cliente relacionados com os sensores de temperatura também, estão definidos para executar a subscrição e receber notificações de alterações nesses dados de 20 em 20s. Os nós de injeção e nós de Opcua-Cliente relacionados com os outros parâmetros, foram definidos para injetar as mensagens no fluxo de 2 em 2s.

Na secção de *Dashboard Layout Editor* foi realizado posicionamento dos elementos gráficos do *dashboard*. Na Figura 51 é apresentado o *Dashboard Layout Editor* e a visualização dos elementos respetivos no *dashboard* no navegador WEB.

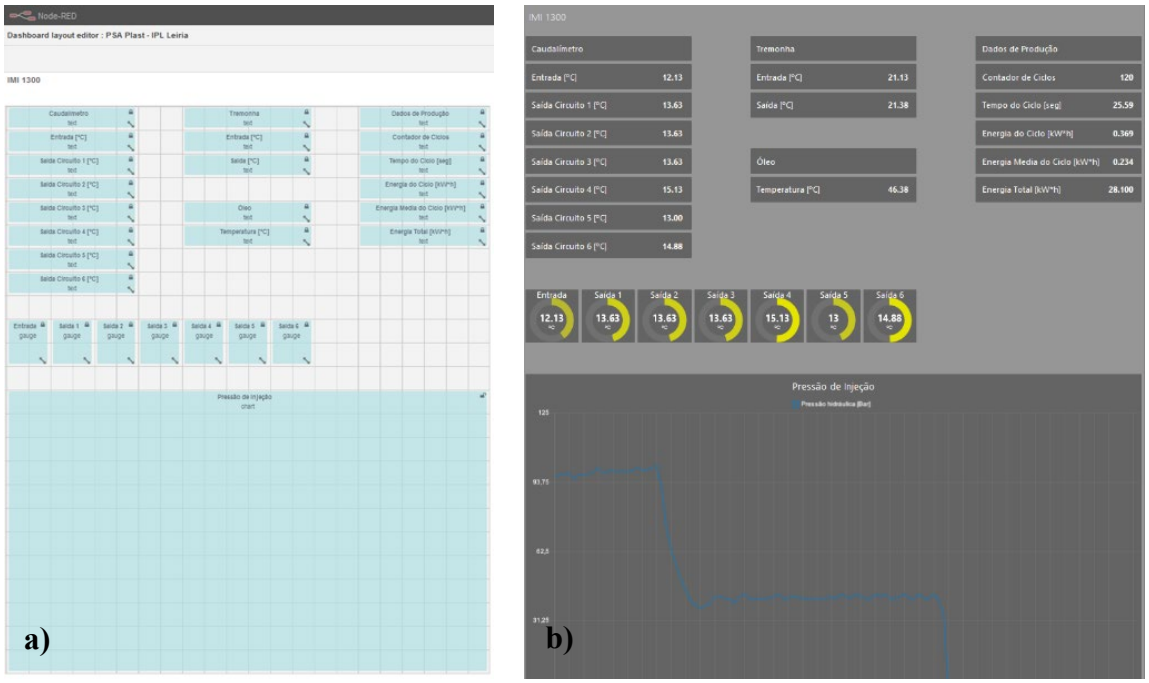

Figura 51 - *Dashboard Layout Editor* (a) e *Dashboard* no navegador WEB (b)

3.13. Testes realizados

Para realizar os testes finais todo o hardware desenvolvido foi colocado na máquina. Os sensores de temperatura foram colocados nos respetivos sítios da máquina (na entrada geral do circuito de arrefecimento do molde e nas 6 saídas de caudalímetro), na entrada e saída do circuito de arrefecimento da tremonha e no depósito de óleo hidráulico. Os sensores de corrente e tensão foram devidamente colocados nas linhas elétricas, na entrada geral de máquina. O sensor de pressão de injeção foi colocado no circuito do cilindro hidráulico de injeção. O relé auxiliar de injeção foi colocado dentro de quadro geral da máquina e ligado em paralelo com a bobine da válvula hidráulica de injeção. O relé auxiliar de contador de ciclos foi colocado também dentro de quadro geral de máquina e ligado em paralelo com o contador de ciclos interno responsável pela lubrificação de máquina de 60 em 60 ciclos. Por sua vez, esse circuito está ligado ao circuito de "Reset" de ciclo de máquina – reinício de ciclo que ocorre todos os ciclos durante a funcionamento de máquina em modo automático.

Figura 52 - Hardware de servidor OPC UA sem conexões

Quadro do servidor foi colocado dentro de quadro geral de máquina. Dentro de quadro do servidor foram colocados um *Single Board Computer* Raspberry Pi 4, dois módulos Arduino Pro Mini, três amplificadores dos sensores de corrente, três sensores de tensão, conversor de tensão de alimentação de 24V DC para 5V DC e o barramento necessário. O quadro do servidor OPC UA, ainda sem as ligações externas, é apresentado na Figura 52.

Para evitar sobreaquecimento de dispositivos e garantir a circulação de ar adequada dentro do quadro, foram instalados dois ventiladores – um na entrada de ar e outro na saída. Para minimizar a interferência eletromagnética o quadro foi ligado à terra. Todas as ligações entre o quadro e os sensores foram feitas usado o cabo blindado com fio flexível, devidamente conectado à terra.

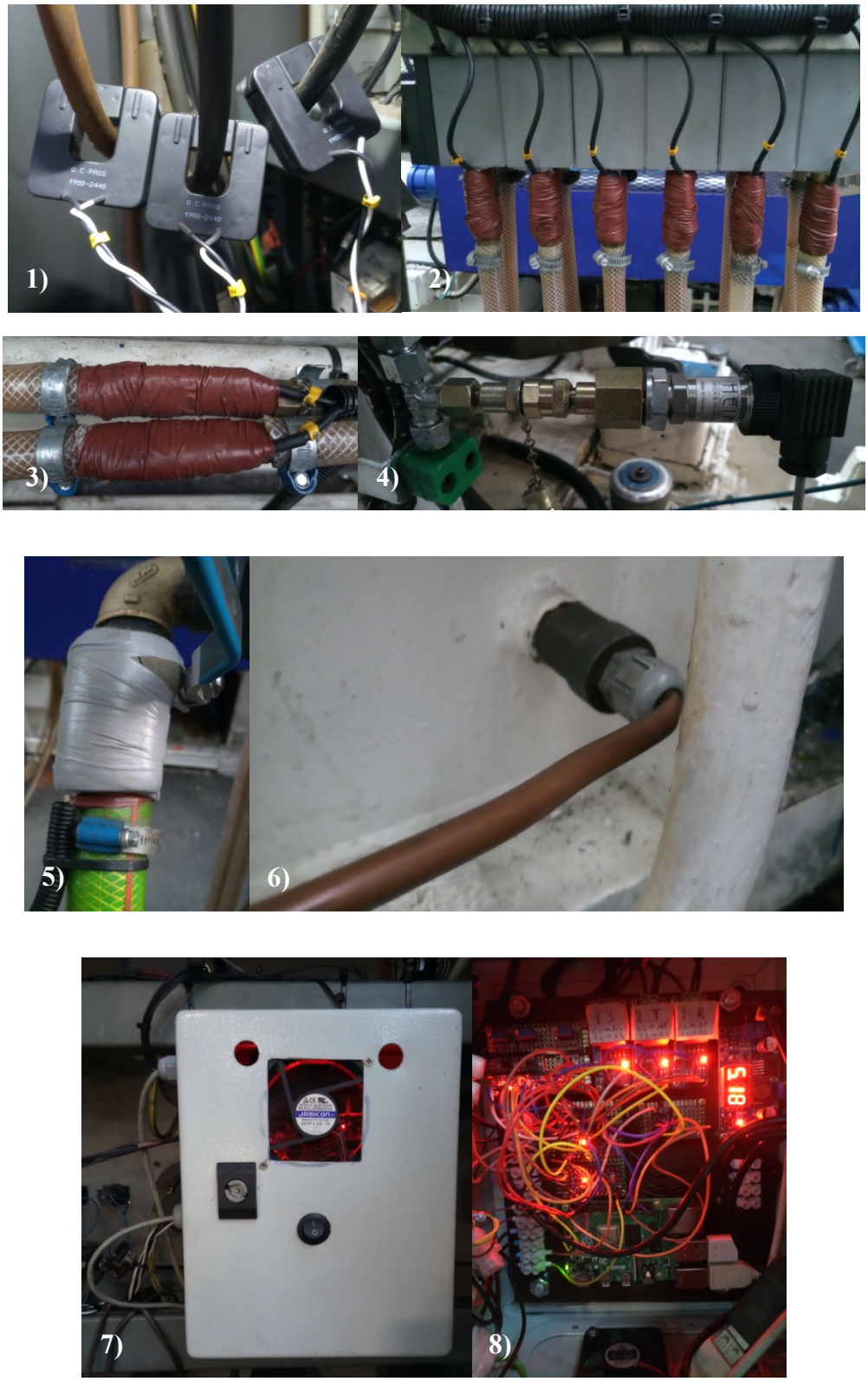

Figura 53 - Hardware instalado na máquina

Na Figura 53 é apresentado o hardware instalado na máquina, o que inclui:

1) Sensores de corrente SCT-019, montados na linha de alimentação geral de máquina;
2) Sensores de temperatura DS18B20, montados no caudalímetro, na saída de circuitos de arrefecimento do molde;
3) Sensores de temperatura DS18B20, montados no circuito de arrefecimento da tremonha;
4) Sensor de pressão do óleo hidráulico HEIM3335, montado no circuito hidráulico de injeção;
5) Sensor de temperatura DS18B20, montado na entrada principal do caudalímetro;
6) Sensor de temperatura DS18B20, montado no deposito de óleo hidráulico;
7) Quadro do servidor OPC UA, montado dentro do quadro geral da máquina;
8) Quadro do servidor OPC UA (mostrado aberto).

Para garantir o acesso estável e robusto do servidor OPC UA à rede Wi-Fi da fábrica em ambiente industrial (sujeito a altos níveis de interferência eletromagnética), perto do quadro de máquina foi colocado e ligado um repetidor de sinal Wi-Fi MWE300RE da Mercusys, apresentado na Figura 54.

Figura 54 - Repetidor Wi-Fi Mercusys MWE300RE[32]

Antes de realizar os testes finais do servidor OPC UA e dos módulos dos sensores baseados no Arduino Pro Mini, o sistema operativo do Raspberry Pi 4 foi modificado para iniciar automaticamente no arranque o Node-RED e a aplicação de servidor OPC UA. Isso garante que ambos os sistemas estejam operacionais imediatamente após a inicialização do

[32] https://manuals.plus/mercury/mw300re-300mbps-wi-fi-range-extender-manual#axzz8VRjE6R4F consultado em dezembro de 2023

dispositivo. Para isso no ficheiro `/etc/rc.local` foram adicionadas duas linhas de texto `sudo systemctl start node-red` e `/opcua-pi/opcua-pi`.

Os primeiros testes de conjunto do sistema construído demonstraram algum atraso na passagem de valores das variáveis internos para as variáveis em formato OPC UA e apresentação no *dashboard* do servidor. Isso pode ser causado pela logica de funcionamento do programa interno de servidor, já que todo o programa está a correr sequencialmente num ciclo e num fluxo de execução único. Para melhorar o funcionamento de servidor foi decidido criar fluxos de execução paralelos utilizando biblioteca `pthread.h`.

"Pthreads" é uma abreviação de "POSIX (*Portable Operating System Interface*)Threads", que se refere à biblioteca de *threads* POSIX em sistemas operativos compatíveis com POSIX, como o LINUX e o UNIX. As *pthreads* permitem que um programa crie e gerencie vários fluxos de execução simultaneamente. Com *pthreads*, é possível criar fluxos de execução separados que podem ser executados concorrentemente ou em paralelo, realizando tarefas diferentes ao mesmo tempo. Como o Raspberry Pi 4 é equipado com um processador quad-core Cortex-A72, *pthreads* executa fluxos em paralelo. O sistema operativo é responsável por agendar a execução desses fluxos nos núcleos do microprocessador físico disponíveis. O sistema operativo gere este agendamento de forma automática, distribuindo os fluxos entre os núcleos da CPU para otimizar o desempenho e a utilização dos recursos. Ainda por cima, cada núcleo pode executar múltiplos fluxos concorrentemente usando o escalonamento de fluxos implementado pelo sistema operativo.

Assim, foi decidido executar em fluxo paralelo o processo de comunicação do módulo Raspberry Pi 4 com os módulos Arduino Pro Mini e o resto de programa que, basicamente, contém a execução do servidor OPC UA. O teste do servidor OPC UA realizado depois da implementação de *pthreads* demonstrou uma melhoria significativa na taxa de atualização de dados recebidos dos módulos Arduino Pro Mini e da visualização posterior no *dashboard* Node-RED.

Durante o funcionamento da máquina e com a última versão do servidor OPC UA, tudo correu como esperado. A alteração de temperatura nos circuitos de arrefecimento de molde ocorreu como foi previsto, seja, mantendo-se uma temperatura próxima dos 12ºC – na entrada de caudalímentro e, nas saídas dos circuitos temperatura, subiu, conforme os circuitos de arrefecimento, entre 13 e 18ºC. À luz dessa alteração de temperatura podemos analisar e tirar conclusões sobre estado dos circuitos de arrefecimento do molde e verificar

a influência na qualidade da peça produzida. A temperatura do óleo hidráulico subiu gradualmente até chegar aos valores de 45-49ºC, o que é dentro da gama das temperaturas normais de funcionamento desta máquina. A temperatura de água no circuito de arrefecimento de tremonha também se comportou como foi previsto, mantendo-se, na entrada um valor próximo dos 20ºC, verificando-se, na saída, uma subida gradual até cerca de 30ºC.

A parte de medição da pressão de injeção no circuito de óleo hidráulico demonstrou também o funcionamento pretendido. Devido ao serviço de interrupções implementado no módulo Arduino Pro Mini, sempre que se inicia o processo de injeção e o relé auxiliar é ativado, o módulo deixa de executar outras tarefas e passa a registar os valores de pressão de injeção no circuito de óleo hidráulico.

O módulo de medição de energia elétrica demonstrou também o funcionamento esperado. O sistema regista valores de tensão e corrente e calcula a energia elétrica utilizada durante um ciclo e a energia elétrica utilizada durante a produção inteira. Também é calculada a média da energia elétrica utilizada no ciclo, e efetuada a contagem de tempo de um ciclo e a quantidade total de ciclos. Como esperado, a energia de um ciclo para outro varia conforme o estado de funcionamento das resistências de aquecimento da câmara da máquina, enquanto que o consumo de energia elétrica pelos motores elétricos deve ser mais estável devido à uniformidade dos movimentos da máquina durante os ciclos. Assim, a utilização de energia elétrica pela máquina durante um ciclo varia dentro da gama de 0.23-0.33kW*h para tempo de ciclo cerca de 23-25s.

A parte de visualização de parâmetros de produção no *dashboard* de Node-RED funciona de forma bastante estável, com um tempo de resposta da alteração de dados de cerca de 1-2s. Ou seja, em todos os ciclos o servidor efetua a escrita de dados do ciclo na base de dados e, ao mesmo tempo, apresenta esses dados no *dashboard* do Node-RED. Portanto, entre o instante em que o módulo Raspberry Pi 4 recebe a mensagem de alteração de número de ciclo e o instante de apresentação de dados do ciclo no *dashboard*, incluindo no gráfico de pressão de injeção, passam cerca de 1-2s.

A base de dados SQLITE com todos os valores registados encontra-se na pasta de aplicação de servidor no módulo Raspberry Pi 4 e está disponível para transferência para outros dispositivos através de Wi-Fi utilizando serviço VNC. O funcionamento da escrita de dados

foi estável, não houve erros relacionados com a abertura e escrita de dados, tudo funcionou como foi pretendido.

Em geral, a última versão do servidor OPC UA com os seus módulos de medição, apesar de operarem num ambiente industrial com elevada taxa de interferências eletromagnéticas, demonstrou a sua funcionalidade de forma bastante estável e fiável. Foi conseguido monitorizar parâmetros de funcionamento da máquina de injeção e guardar os valores desses parâmetros na base de dados com a precisão e fiabilidade pretendida. No Capítulo 4, serão apresentados conclusões e possível trabalho futuro a realizar.

4. Conclusões e trabalho futuro

Inicialmente, para criar o servidor OPC UA, foi planeado utilizar um módulo ESP32 DevKitC. Mas, devido à quantidade de sensores e complexidade do projeto, com esse módulo não foi possível atingir as metas pretendidas. No entanto, o módulo ESP32 demonstrou funcionalidade e poder computacional suficiente para criar dispositivos para IoT e IIoT de tamanho e funcionalidade reduzidos, incluindo servidores OPC UA com funcionalidades limitadas.

O *Single Board Computer* Raspberry Pi 4 é um dispositivo, também de tamanho bastante reduzido, mas mais poderoso, tanto em termos da capacidade de processamento quanto da quantidade de memória, do que o módulo ESP32. Esse módulo serve perfeitamente para este tipo de utilizações, verificando-se até, durante os testes finais do servidor, que os recursos mobilizados para o funcionamento do servidor OPC UA, em média, não consumiram mais de que 10% de recursos computacionais do CPU. Em conjunto com o Node-RED e outros serviços necessários para o funcionamento correto do *Single Board Computer* e servidor OPC UA, o sistema, em média, não consome mais de que 30% dos recursos computacionais de CPU e menos de que 10% de memória RAM. Isso significa que o Raspberry Pi 4 pode servir para desenvolvimento de sistemas ainda mais complexos com as funcionalidades mais avançadas.

O sistema de monitorização de parâmetros de máquina de injeção elaborado durante o trabalho, permite obter dados sobre o funcionamento da máquina em modo de tempo real e guardar esses dados para análise posterior. A observação de dados em modo de tempo real permite identificar e reagir a anomalias ou falhas de funcionamento da máquina durante a produção. Por seu lado, a análise posterior dos dados obtidos durante a produção permite encontrar soluções para melhorar a qualidade e eficiência na produção de peças plásticas nas produções seguintes. Também a análise dos dados de operação da máquina pode revelar oportunidades para otimizar o consumo de energia elétrica, o que pode resultar em economias significativas no longo prazo e redução do impacto ambiental.

A base de dados SQLITE provou ser um excelente recurso para armazenamento de grandes volumes de dados para análise posterior do funcionamento da máquina utilizando serviços externos para analises massivas de dados (desenhar gráficos, visualizar tendências, etc.). A

utilização dos dados armazenados na base de dados SQLITE permite realizar análises detalhadas do desempenho da máquina ao longo de tempo, identificar padrões de operação, detetar anomalias e tomar decisões para otimizar a eficiência e a produtividade da produção.

O *dashboard* desenvolvido com recurso ao Node-RED proporciona uma interface gráfica intuitiva e de fácil utilização para a ligação de dispositivos e serviços IoT. Com a quantidade significativa de bibliotecas de nós pré-construídos, o Node-RED permite criar protótipos rapidamente, acelerando o processo de desenvolvimento de soluções IoT. O Node-RED é altamente flexível e pode ser estendido com a criação de novos nós personalizados, permitindo a adaptação para uma ampla variedade de casos de uso. Além disso, esta plataforma é escalável e pode ser utilizada em projetos pequenos e grandes. O Node-RED suporta grande variedade de protocolos e APIs, incluindo OPC UA, o que permite integração de ampla gama de dispositivos e serviços, facilitando a criação de soluções abrangentes e interconectadas. Com as ferramentas integradas de monitorização e depuração, o Node-RED facilita a identificação e resolução de problemas durante o desenvolvimento e funcionamento das aplicações IoT.

Uma possível evolução deste trabalho pode consistir na construção de sistema de controlo manual e automático de algumas funcionalidades do servidor e da máquina, tais como:

- Controlo automático de temperatura de óleo hidráulico por meio de aplicação de uma válvula no circuito de arrefecimento, controlada pelo servidor OPC UA ou por um módulo auxiliar;
- Controlo automático de temperatura de água nos circuitos de arrefecimento do molde por meio de aplicação de válvulas proporcionais nos circuitos, controladas pelo servidor OPC UA ou por um módulo auxiliar;
- Elaborar e implementar o processo de classificação e separação de peças aprovadas e rejeitadas com o registo respetivo por meio de desenvolvimento de uma interface respetiva no servidor OPC UA.

A adição da capacidade de guardar dados históricos pelo servidor, implementando a funcionalidade OPC UA HDA, poderia permitir o acesso a esta informação por outras ferramentas de análise e decisão sem a necessidade de uma base de dados intermédia.

Outro ponto de melhoria, seria a implementação das características de OPC UA *Alarms and Conditions* para a criação de mecanismos para a gestão de alarmes e condições no funcionamento da máquina e enviar alarmes e relatórios de condição aos clientes OPC UA.

No que respeita ao *dashboard* do Node-RED, seria importante fazer refletir cada uma das novas funcionalidades na interface, para melhor gestão do equipamento e da sua operação.

Referências bibliográficas

[1] Plastics Europe, "Plastics – the fast Facts 2023," https://plasticseurope.org/knowledge-hub/plastics-the-fast-facts-2023/.

[2] D. Li, H. Zhou, P. Zhao, and Y. Li, "A real-time process optimization system for injection molding," *Polym Eng Sci*, vol. 49, no. 10, pp. 2031–2040, Oct. 2009, doi: 10.1002/pen.21444.

[3] C. J. D. Brandl and C. Johnsson, "Beyond the pyramid: Using isa95 for industry 4.0 and smart manufacturing." 2022.

[4] J. Delsing, J. Eliasson, J. van Deventer, H. Derhamy, and P. Varga, "Enabling IoT automation using local clouds," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, Dec. 2016, pp. 502–507. doi: 10.1109/WF-IoT.2016.7845474.

[5] O. Carlsson, C. Hegedus, J. Delsing, and P. Varga, "Organizing IoT Systems-of-Systems from standardized engineering data," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Oct. 2016, pp. 5277–5282. doi: 10.1109/IECON.2016.7792932.

[6] Hisham GOUDA, "Industry 4.0 and the Standard ISA-95: The Pathway to Revolutionizing the Energy Industry," *SAP Community*, Apr. 2023.

[7] "AutomationML Whitepaper: OPC Unified Architecture Information Model for Automation ML," p. 15, Mar. 2016.

[8] M. H. Schwarz and J. Borcsok, "A survey on OPC and OPC-UA: About the standard, developments and investigations," in *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, IEEE, Oct. 2013, pp. 1–6. doi: 10.1109/ICAT.2013.6684065.

[9] N. T. T. Tu, N. D. Cuong, V. Van Tan, and H. Q. Thang, "Research and development of OPC client-server architectures for manufacturing and process automation," in *Proceedings of the 1st Symposium on Information and Communication Technology*, 2010, pp. 163–170.

[10] Control Engineering Staff, "OPC Data eXchange spec, sample code released," *Control Engineering Daily News*, Jun. 2003.

[11] S. Chilingaryan and W. Eppler, "High speed data exchange protocol for modern distributed data acquisition systems based on OPC XML-DA," in *14th IEEE-NPSS Real Time Conference, 2005.*, IEEE, 2005, p. 5 pp. doi: 10.1109/RTC.2005.1547472.

[12] S. Hoppe, "There is no Industrie 4.0 without OPC UA," *Opcfoundation: Scottsdale, AZ, USA*, 2017.

[13] B. Müllers, "Euromap 77: OPC UA Interfaces for Plastics and Rubber Machinery—Data Exchange between Injection Moulding Machines and MES.," May 2018.

[14] B. Müllers, " Euromap83: OPC UA for Plastics and Rubber Machinery—General Type Definitions. ," Jun. 2021.

[15] Friedrich Johannaber, *Injection Molding Machines. A User's Guide. Edition: 4th* . 2008.

[16] D. V Rosato and M. G. Rosato, *Injection Molding Handbook*. Springer US, 2012. [Online]. Available: https://books.google.pt/books?id=4VHxBwAAQBAJ

[17] D. V. ; R. M. G. Rosato, Rosato, and Marlene G., *Injection Molding Handbook*. 2001.

[18] Robin Kent, *Energy Management in Plastics Processing*. Elsevier, 2018. doi: 10.1016/C2017-0-02035-9.
[19] P. Zhao *et al.*, "Intelligent Injection Molding on Sensing, Optimization, and Control," *Advances in Polymer Technology*, vol. 2020, pp. 1–22, Mar. 2020, doi: 10.1155/2020/7023616.
[20] DME, "Flosense - Flexible Flow Monitoring." 2023.
[21] TECHSPAN NEW ZEALAND and Engel, "Smart machine iQ weight control | iQ clamp control | iQ flow control," Sep. 2019.
[22] Kontron Europe GmbH, "CC300 Operator Panel User Guide (Preliminary Version 0.2) P/N: XXXX-XXXX... ." 2013.
[23] Mathias Schläger, Christoph Balka, and Josef Giessauf, "Integrated Temperature Control Solution for Reduced Rejects and Greater Energy Efficiency," *INJECTION MOLDING*, 2017.
[24] "What is the better temperature of the hydraulic oil of the injection molding machine?," Feb. 2021.
[25] C. Wu, C. Xu, X. Mao, B. Li, J. Hu, and Y. Liu, "Heating Analysis in Constant-pressure Hydraulic System based on Energy Analysis," *IOP Conf Ser Earth Environ Sci*, vol. 100, p. 012147, Dec. 2017, doi: 10.1088/1755-1315/100/1/012147.
[26] J. M. Hernández, M. Abu-Ayyad, and R. Dubay, "Cavity pressure control during injection in an injection molding machine," vol. 4, pp. 2172–2176, Jan. 2008.
[27] R.T. Maher and H.T. Plant, "Proceedings of SPE ANTEC ," 1975.
[28] R.M. Criens, M. Handler, and H.G. Mosle, *Injection molding of plastics*. 1985.
[29] I. A. Rawabdeh and P. F. Petersen, "In-line monitoring of injection molding operations: A literature review," *Journal of Injection Molding Technology*, vol. 3, no. 2, p. 47, 1999.
[30] N. Dontula, P. C. Sukanek, H. Devanathan, and G. A. Campbell, "An experimental and theoretical investigation of transient melt temperature during injection molding," *Polym Eng Sci*, vol. 31, no. 23, pp. 1674–1683, Dec. 1991, doi: 10.1002/pen.760312308.
[31] S. Macfarlane and R. Dubay, "The Effect of Varying Injection Molding Conditions on Cavity Pressure," *SPE ANTEC Tech. Papers*, pp. 653–657, 2000.
[32] G. Xu, L. Yu, L. J. Lee, and K. W. Koelling, "Experimental and numerical studies of injection molding with microfeatures," *Polym Eng Sci*, vol. 45, no. 6, pp. 866–875, Jun. 2005, doi: 10.1002/pen.20341.
[33] Gud Mould Industry Limited, "Selection and installation of injection mold pressure sensor," Nov. 2021.
[34] John Bozzelli, "Improve Profits by Graphing Injection Pressure," *Plastics Technology*, Feb. 2012.
[35] A. Elduque, D. Elduque, C. Pina, I. Clavería, and C. Javierre, "Electricity Consumption Estimation of the Polymer Material Injection-Molding Manufacturing Process: Empirical Model and Application," *Materials*, vol. 11, no. 9, 2018, doi: 10.3390/ma11091740.
[36] T. G. Gutowski, M. S. Branham, J. B. Dahmus, A. J. Jones, A. Thiriez, and D. P. Sekulic, "Thermodynamic Analysis of Resources Used in Manufacturing Processes," *Environ Sci Technol*, vol. 43, no. 5, pp. 1584–1590, Mar. 2009, doi: 10.1021/es8016655.
[37] Letschert and Virginie E, "Energy Efficiency Appliance Standards: Where do we stand, how far can we go and how do we get there? An analysis across several economies," Jun. 2013.

[38] S. Thiede, G. Bogdanski, and C. Herrmann, "A Systematic Method for Increasing the Energy and Resource Efficiency in Manufacturing Companies," *Procedia CIRP*, vol. 2, pp. 28–33, 2012, doi: 10.1016/j.procir.2012.05.034.
[39] T. Spiering, S. Kohlitz, H. Sundmaeker, and C. Herrmann, "Energy efficiency benchmarking for injection moulding processes," *Robot Comput Integr Manuf*, vol. 36, pp. 45–59, Dec. 2015, doi: 10.1016/j.rcim.2014.12.010.
[40] I. Meekers, P. Refalo, and A. Rochman, "Analysis of Process Parameters affecting Energy Consumption in Plastic Injection Moulding," *Procedia CIRP*, vol. 69, pp. 342–347, 2018, doi: 10.1016/j.procir.2017.11.042.
[41] "ETCR5000 3-phase Energy Power Quality Analyzer Logger," https://www.etcrmeter.com/etcr5000-3-phase-energy-power-quality-analyzer-logger_p41.html.
[42] "HBM eDrive Testing: Power Analyzer and Data Acquisition System for Testing Electrical Drives," https://www.hbm.com/en/8945/power-analyzer-and-daq-system-for-testing-electrical-driv/?product_type_no=Power%20Analyzer.
[43] "Monitor Injection Molding Energy Consumption Easily on the SXIII Series," https://shibaura-machine.com/articles/im-2022-02-16-monitor-injection-molding-energy-consumption-easily-on-the-sxiii-series/.
[44] Espressif Systems, "ESP32-WROOM-32 Datasheet Version 3.4." 2023.
[45] open62541, "Open source implementation of OPC UA," https://github.com/open62541/open62541.
[46] Inc. Maxim Integrated Products, "DS18B20 Programmable Resolution 1-Wire® Digital Thermometer." 2019.
[47] S. Monk, *Raspberry pi cookbook*. " O'Reilly Media, Inc.," 2022.
[48] Ltd. Beijing Yaohuadechang Electronic Co., "SCT019 Split core current transformer Datasheet."
[49] Qingxian Zeming Langxi Electronic, "ZMPT101B Current-type Voltage Transformer."
[50] A. D. ;Ochs, T. L. ; Hartman and United States. Bureau of Mines., "Accurate power monitoring for electric arc furnaces," 1994.
[51] M. K. Nalbant, "Power factor calculations and measurements," in *Fifth Annual Proceedings on Applied Power Electronics Conference and Exposition*, IEEE, 1990, pp. 543–552. doi: 10.1109/APEC.1990.66453.
[52] S. D. Umans, "AC induction motor efficiency," in *Proceedings of the 19th Electrical Electronics Insulation Conference*, IEEE, pp. 99–107. doi: 10.1109/EEIC.1989.208201.