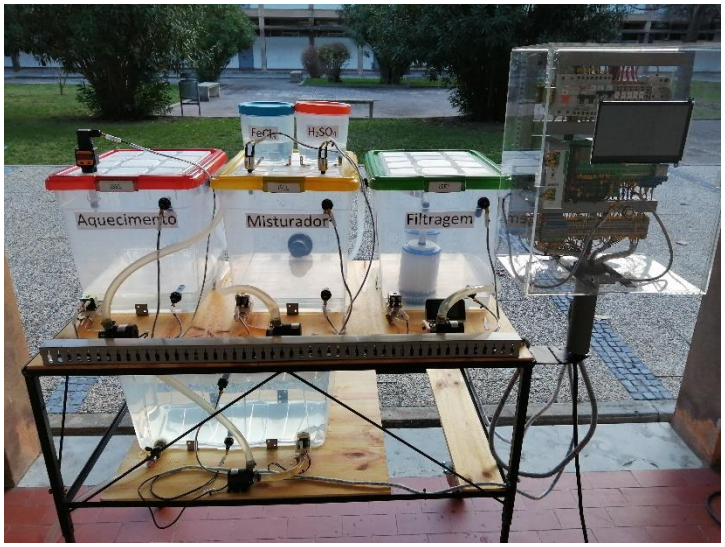
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrotécnica Energia e Automação



Desenvolvimento de um simulador para sistemas de tratamento de água

MIGUEL CORREIA LOPES DE ALMEIDA
(Licenciado em Engenharia Eletrotécnica)

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia Eletrotécnica – Ramo de Automação Industrial

Orientadores:
Prof. Doutor Armando José Leitão Cordeiro
Prof.ª Doutora Maria da Graça Vieira de Brito Almeida

Júri:
Presidente: Prof. Doutor Luís Manuel dos Santos Redondo
Vogais:
Prof.ª Doutora Mafalda Maria Morais Seixas

Abril de 2022

Dissertação realizada sob orientação de

Doutor Armando José Leitão Cordeiro

Professor Coordenador do Departamento de

Engenharia Eletrotécnica Energia e Automação

Instituto Superior de Engenharia de Lisboa

e

Doutora Maria da Graça Vieira de Brito Almeida

Professora Adjunta do Departamento de

Engenharia Eletrotécnica Energia e Automação

Instituto Superior de Engenharia de Lisboa

Resumo

Propõe-se, nesta dissertação, a implementação de um simulador didático de automação industrial apoiado num sistema de supervisão.

Como resultado deste trabalho, será desenvolvida uma solução baseada num sistema de seis reservatórios, juntamente com um sistema de supervisão que permite comandar e monitorizar o simulador através de um *Raspberry Pi* com um display com ecrã *touch* e uma aplicação *Android*.

Para tal, irão ser estudados temas relacionados com sistemas SCADA e protocolos de comunicação. Como os processos simulados estão presentes em estações de tratamento de água, será efetuado também um estudo do funcionamento das estações de tratamento de água e águas residuais, seus ciclos de funcionamento e automatismos de modo a obter-se a água com a qualidade desejada de acordo com a respetiva normalização.

Pretende-se com esta dissertação, desenvolver um sistema que possa ser usado por outros alunos de licenciatura e mestrado do curso de engenharia eletrotécnica do ISEL, de modo a estes aprofundarem os seus conhecimentos em automação. Pretende-se ainda que o simulador seja usado em feiras, exposições e dias abertos a alunos internos e externos ao ISEL, mostrando algum trabalho que tem vindo a ser desenvolvido no mestrado em engenharia eletrotécnica, especialmente no ramo de automação industrial.

Palavras-Chave: Automação; SCADA; Estações de Tratamento de Águas; Comunicação; Ensino em Automação

Abstract

This dissertation proposes the implementation of a didactic simulator for industrial automation supported by a supervision system.

As a result of this work, a solution based on a system of six reservoirs will be developed, together with a supervision system that allows the simulator to be controlled and monitored through a Raspberry Pi with a touch screen display and an Android application.

To this end, SCADA and communication protocols will be studied. Because the simulated processes are present in water treatment plants, a study on the operation of the water and wastewater treatment plants, their operating cycles and automation to obtain the water with the desired quality according to with the respective standardization, will also be carried out.

The aim of this dissertation is to develop a system that can be used by other undergraduate and master's students of the electrical engineering course at ISEL, for them to deepen their knowledge in automation. It is also intended that the simulator be used in fairs, exhibitions, and open days to internal and external students at ISEL, showing some of the work that has been developed in the master's degree in electrotechnical engineering, especially in the field of industrial automation.

Keywords: Automation; SCADA; Water Treatment Plants; Communication; Teaching in Automation

Agradecimentos

Pretendo dedicar esta página de agradecimentos a todas as pessoas que comigo percorreram o longo caminho da realização desta dissertação.

Desejo em especial agradecer à minha família e companheira por me terem acompanhado durante os anos de escrita da dissertação, por todos os sacrifícios que fizeram, pela minha ausência ou indisponibilidade e por nunca me terem deixado desistir do objetivo final.

Um agradecimento em particular aos meus orientadores, Professor Doutor Armando José Leitão Cordeiro e Professora Doutora Maria da Graça Vieira de Brito Almeida, por todo o esforço, trabalho e dedicação com que me apoiaram e me guiaram ao longo destes últimos anos.

Agradeço também ao Instituto Superior de Engenharia de Lisboa, aos seus docentes e não docentes pelos conhecimentos que me passaram ao longo de todo o meu percurso académico de mestrado que resultaram na elaboração desta dissertação para obtenção do grau de mestre.

Lista de Siglas

ADU – *Application Data Unit*

ANACOM – Autoridade Nacional de Comunicações

BA – *Bitwise Arbitration*

CD – *Collision Detection*

COM – *Component Object Model*

CPS – *Cyber-Physical Systems*

CPU – *Central Process Unit*

CSMA – *Carrier Sense Multiple Access*

CTS – *Clear to Send*

DC – *Direct Current*

DCOM – *Distributed Component Object Model*

DCS – *Decentralized Control System*

EHF – *Extremely High Frequency*

EIA – *Electronics Industries Association*

ETA – Estação de Tratamento de Água

ETAR – Estação de Tratamento de Águas Residuais

FBD – *Function Block Diagram*

HF – *High Frequency*

HMI – *Human Machine Interface*

I/O – *Input/Output*

IDE – *Integrated Development Environment*

IEC – *International Electrotechnical Commission*

IEEE – *Institute of Electrical and Electronics Engineers*

IIoT – *Industrial Internet of Things*

IL – *Instruction List*

IoT – *Internet of Things*

IP – *Internet Protocol*

ISA – *International Society of Automation*

ISO – *International Standards Organization*

ITED – Infraestruturas de Telecomunicações de Edifícios

LAN – *Local Area Network*

LD – *Ladder Diagram*

LF – *Low Frequency*

MBAP – *Modbus Application Protocol*

MIT – *Massachusetts Institute of Technology*

MTU – *Master Terminal Unit*

OPC – *Open Platform Communications*

OPC DA – *Open Platform Communications Data Access*

OPC UA – *Open Platform Communications Unified Architecture*

OSI – *Open Systems Interconnection*

PDU – *Protocol Data Unit*

PLCs – *Programmable Logic Controllers*

RTS – *Request to Send*

RTU – *Remote Terminal Unit*

SCADA – *Supervisory Control and Data Acquisition*

SFC – *Sequential Function Chart*

SHF – *Super High Frequency*

ST – *Structured Text*

TCP – *Transmission Control Protocol*

TTL – *Transistor-Transistor Logic*

UART – *Universal Asynchronous Receiver-Transmitter*

UHF – *Ultra High Frequency*

UPS – *Uninterrupted Power Supply*

VHF – *Very High Frequency*

VISIR – *Virtual Instrument System in Reality*

VLF – *Very Low Frequency*

VPN – *Virtual Private Network*

WAN – *Wide Area Network*

Índice

Índice de figuras

Índice de tabelas

Capítulo 1

Introdução

Resumo: O presente capítulo apresenta uma introdução ao tema da dissertação assim como o enquadramento e motivação científica para a elaboração da mesma, prosseguindo com os objetivos a concretizar e uma breve descrição da estrutura do trabalho.

1.1 Enquadramento e Motivação

Ao longo das últimas décadas temos assistido a um crescimento muito acelerado de novas soluções tecnológicas em muitos ramos de atividade. Estes avanços tecnológicos resultam muitas vezes de parcerias estratégicas compostas por equipas de investigação que envolvem centros de investigação, institutos, universidades e empresas com o objetivo de criar mais e melhores produtos ou soluções a custos mais baixos ou com elevado potencial de venda. Esses produtos, especialmente os dispositivos eletrónicos programáveis, são atualmente responsáveis por novas oportunidades de negócio, por promover o entretenimento e comunicação entre pessoas, o acesso quase ilimitado a fontes de informação em qualquer momento e em qualquer lugar bem como novas ou melhoradas ferramentas com fins didáticos, entre muitos outros aspetos positivos. É curioso verificar como grande parte das receitas provenientes da produção de novos equipamentos é investida na melhoria desses mesmos equipamentos ou de novos equipamentos criando uma cadeia de evolução tecnológica com dimensão exponencial.

O ramo da automação industrial tem também vindo a registar enormes progressos resultantes do surgimento de novas técnicas de produção com recurso a novos equipamentos e com consequente impacto positivo para a economia e para a qualidade final dos produtos. Estes avanços tecnológicos vieram também trazer ao longo das últimas décadas uma crescente flexibilidade e complexidade para os processos automatizados. Exemplo disso são, por exemplo, as fábricas de automóveis onde o nível de integração, a multiplicidade de sistemas e processos a decorrer em paralelo é de tal forma grande e complexa que é praticamente impossível a sua gestão sem a utilização de soluções computorizadas.

A automação industrial é atualmente uma área de natureza multidisciplinar que obriga aos engenheiros vastos conhecimentos em áreas afins à eletrotecnia, como a mecânica, a informática ou a eletrónica. Isto porque, quase todos os equipamentos industriais existentes atualmente possuem inúmeros circuitos de potência com regulação eletrónica (Ex.: retificadores controlados, inversores, conversores DC-DC) e com possibilidade de programação a diversos níveis. Estas mudanças registadas na automação industrial (e em muitos outros ramos) representam um elevado desafio para as diversas instituições de ensino pois é necessário facultar aos seus alunos as ferramentas necessárias para colocar em funcionamento, manter e operar estes modernos equipamentos sem esquecer os princípios e leis fundamentais da física e matemática pelos quais esses mesmos equipamentos se regem.

O trabalho que se propõe nesta dissertação vai de alguma forma ao encontro de algumas necessidades educativas em termos de automação industrial, nomeadamente no que diz respeito aos controladores programáveis, redes de comunicação em ambientes industriais e sistemas de supervisão. Sendo estes equipamentos alguns dos mais utilizados em ambientes industriais faz todo o sentido desenvolver soluções didáticas que façam uso destes mesmos equipamentos para promover o ensino e os conhecimentos deste ramo da engenharia eletrotécnica. O trabalho que se propõe nesta dissertação visa desenvolver uma solução que permita aos alunos de licenciatura e mestrado em engenharia

eletrotécnica controlar de forma simples uma representação de algumas etapas de uma estação de tratamento de água. Além disso, a utilização de soluções desta natureza pretende que os alunos se tornem conhecedores do funcionamento parcial ou integral de alguns processos industriais através da interação com esses sistemas através de um sistema de supervisão.

1.2 Objetivos

O principal objetivo da presente dissertação é desenvolver um simulador didático de automação industrial apoiado num sistema de supervisão. Este será constituído por um sistema de seis reservatórios, juntamente com um sistema de supervisão que permite comandar e monitorizar o simulador através de um display com ecrã táctil e uma aplicação *Android*.

O propósito desta solução é dar a oportunidade aos alunos de licenciatura e mestrado do curso de engenharia eletrotécnica, do Instituto Superior de Engenharia de Lisboa, de interagir e testar diversos processos automatizados baseados em estações de tratamento de água. Desta forma têm oportunidade de experienciar um desafio diferente e de aprofundarem os seus conhecimentos em automação. Durante o desenvolvimento desta dissertação o equipamento ficará em condições dos alunos poderem interagir com três processos automatizados relativos ao tratamento de água, podendo no futuro ser programados e alterados. Com esta solução, os processos poderão ser manipulados pelos alunos e/ou docentes das disciplinas de automação com ótica de melhorar o que se encontra implementado, sendo que os limites serão aqueles que forem impostos pelo hardware e/ou software que compõe a solução. De qualquer modo a solução pode sempre ser expandida com a utilização de equipamento mais recentes com mais ou melhores recursos.

1.3 Estrutura

A presente dissertação encontra-se dividida em quatro capítulos com os seguintes títulos: introdução, estado da arte, simulador didático e conclusão. O documento começa com uma introdução ao tema e objetivos propostos. É depois apresentada, no capítulo do estado da arte, uma visão geral da evolução da tecnologia até ao seu estado atual e do ensino em engenharia, bem como uma breve explicação do funcionamento das estações de tratamento de água e águas residuais. No terceiro capítulo – simulador didático – é apresentada a solução desenvolvida para dar a oportunidade aos alunos de interagir e testar diversos processos automatizados baseados em estações de tratamento de água. Neste são descritos todos os equipamentos utilizados e programação realizada, assim como os resultados experimentais do simulador. Finalmente, no último capítulo são apresentadas as conclusões.

Durante a escrita do documento foram adotadas algumas convenções que se apresentam de seguida:

- Alguns dos conceitos técnicos mantêm a sua designação em língua inglesa, colocada em itálico;
- Todas as siglas utilizadas de conceitos técnicos são escritas na língua inglesa;
- A numeração de todas as figuras e tabelas é sequencial ao longo de cada capítulo, sendo referenciadas por dois números separados por um ponto (o primeiro número refere-se ao capítulo e o segundo à sequência ordenada dentro do capítulo);
- As referências bibliográficas foram ordenadas pela ordem que surgem ao longo da dissertação.

Capítulo 2

Estado da Arte

Resumo: O capítulo do estado da arte apresenta uma revisão do estado da arte que se considera relevante para o tema e trabalho desenvolvido nesta dissertação. Inicia-se com aspetos teóricos relacionados com a automação industrial, sistemas SCADA e ensino de engenharia e culmina no tema das estações de tratamento de água e automatização das mesmas dado ser um aspeto central deste trabalho.

2.1 Automação Industrial

O termo automação surge associado ao vice-presidente da Ford, Delmer S. Harder, quando em 1948 proferiu o termo na frase: "*What we need is more automation*". No caso da Ford, a automação era necessária para a otimização dos recursos através da substituição do Homem pela máquina, libertando-o das tarefas repetitivas e perigosas sendo, ainda hoje, esse o seu principal propósito [1].

No entanto, apesar de o termo estar ligado a Harder, pode considerar-se que os sistemas automáticos já eram utilizados pelo Homem há muitos séculos atrás. Podem ser mencionados alguns exemplos, sendo que uma das primeiras referências da utilização de um sistema automático aparece associada aos relógios de água de Ktesibios, matemático e engenheiro Grego (270 AC) [2]. Contudo, o primeiro trabalho significativo a aparecer em automação foi o "dispositivo centrífugo" de James Watt que fazia o controlo de velocidade do motor a vapor, representado na figura 2.1 [4]. Este foi considerado o primeiro passo para o início da revolução industrial.

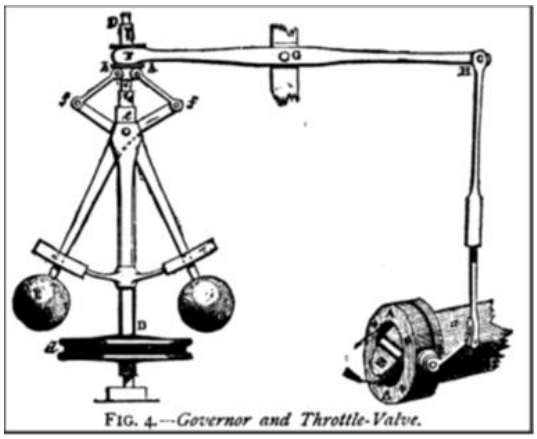


Figura 2.1 – Dispositivo centrífugo de James Watt [4]

Foi durante a revolução industrial que apareceram as primeiras máquinas. Com a evolução da tecnologia até aos dias de hoje, a automação sofreu várias mudanças e alterações, mas foi a segunda guerra mundial e a corrida ao espaço na segunda metade do século XX que impulsionaram o desenvolvimento tecnológico com o aperfeiçoamento dos computadores digitais e microcontroladores. Nesse período apareceu a lógica a relés, tendo sido bastante usada nos caminhos de ferro, tanto na sinalização como no controlo de direção dos carris. Esta foi criada para representar a lógica binária de *on-off* usada pelos relés, tendo sido desde então, associada aos controladores programáveis lógicos, do inglês *programmable logic controlers* (PLC). Estes são, atualmente, a base da automação industrial.

2.2 Controladores Programáveis (PLC's)

Com o aperfeiçoamento dos computadores digitais e microcontroladores, surgiram os controladores lógicos programáveis como uma solução industrial para o controlo de variáveis maioritariamente digitais, em tempo real.

O primeiro PLC a aparecer foi criado por Dick Morley, da General Motors, em 1968 [2]. Este primeiro exemplar tinha 125 *words* de memória, ficando bastante lento nos primeiros testes efetuados e, por isso mesmo, nunca chegou a ser entregue. O primeiro a ser realmente entregue foi o Modicon – Modular Digital Controler – tendo sido concebido para controlo de máquinas usadas na fabricação de peças. A visão que deu origem ao Modicon era a de existir um controlador capaz de emular a lógica implementada pelos relés com um fim industrial, que ainda permanece até aos dias de hoje. Para tal, o aparelho teria de ser robusto, mas flexível, usar a programação *ladder* já conhecida pelos técnicos e eletricistas e, ainda, ser adaptável aos circuitos já existentes no terreno através de ligações físicas. Hoje existem inúmeros PLC's, criados por diversas marcas, tal como é possível verificar na tabela 2.1, no entanto, a sua constituição é idêntica.

Tabela 2.1 – Lista de algumas das marcas mais populares de PLC's [5]

Nº	**Empresa**	**Nº**	**Empresa**
1	Siemens	12	Koyo Electronics Industries
2	Rockwell/Allen-Bradley	13	Delta
3	Mitsubishi Electric	14	Eaton
4	Schneider Electric	15	Keyence
5	ABB	16	LS Electric
6	Honeywell Process	17	Panasonic
7	Omron	18	Phoenix Contact
8	Hitachi Industrial	19	Pilz
9	IDEC	20	WAGO
10	B&R Industrial Automation	21	Yokogawa Electric
11	Bosch Rexroth	22	Toshiba

2.2.1 Constituição do PLC

Para que seja de fácil compreensão, é possível comparar o PLC a um computador doméstico na medida em que ambos têm memória, um CPU e motherboard. As suas diferenças prendem-se no contexto de operação. Sendo um equipamento a operar em ambiente industrial, o PLC é bastante

resistente a vapores corrosivos, óleos, sujidade, a valores extremos de temperatura e humidade e a ruídos eletromagnéticos. Para além de compacto, é versátil e flexível para que seja de fácil manuseamento e análise aos erros, através de alarmística luminosa, por qualquer operador. A sua estrutura é maioritariamente modular permitindo a fácil remoção e substituição de elementos que necessitem de reparação. Internamente, ao contrário dos computadores, o PLC não tem discos rígidos fixos nem removíveis, mas sim uma unidade de memória já integrada onde são guardados os seus programas. Estes programas são criados pelo operador e corridos no controlador, respeitando o seu ciclo de varrimento. O ciclo começa pela leitura do estado das variáveis de entrada, correndo de seguida o programa. Após execução, é efetuado um diagnóstico de erros pelo CPU terminando o ciclo, caso o diagnóstico tenha sido bem-sucedido, com a atualização dos estados das variáveis de saída. Por norma, todos os PLC's têm um temporizador – *watchdog* – que deteta falhas críticas no controlador, acionando um contacto externo para alertar o operador e desencadear medidas corretivas [6].

Os seus principais componentes focam-se na comunicação, seja ela com outros equipamentos ou com os sensores e atuadores no campo, tal como representado na figura 2.2.

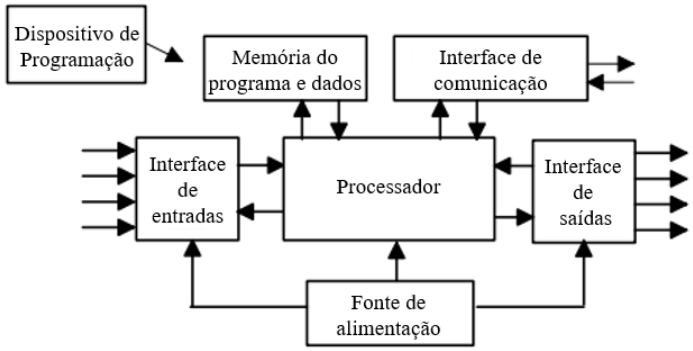


Figura 2.2 – Representação dos componentes do PLC [7]

A interface de entradas/saídas faz a ligação com todos os equipamentos no campo, podendo ser fixa ou modular. As entradas recebem os sinais enviados pelos equipamentos, convertendo-os em sinais compreendidos pelo processador do PLC e as saídas convertem os sinais enviados pelo PLC em sinais para controlar os equipamentos no campo [8].

Esta conversão, tanto para as entradas como para as saídas, tem de proteger o PLC internamente contra picos de corrente e tensão. O seu processador é bastante sensível a essas variações, não suportando muitos dos valores aos quais poderia estar sujeito. Para tal são usados circuitos com isolamento galvânico. Em circuitos eletrónicos, sem entrar ao detalhe, este tipo de isolamento consiste por norma num acoplamento ótico através de semicondutores apropriados, com o objetivo de separar dois circuitos que necessitem de comunicar. Estes ficam com as massas isoladas havendo na mesma a transmissão de sinais e, consequentemente, comunicação [8].

As entradas e saídas podem ser digitais (binárias) ou analógicas. As digitais são *bit* a *bit*, sendo usadas na maior parte dos equipamentos como por exemplo, seletores ou lâmpadas. As analógicas aparecem para permitir uma maior versatilidade ao PLC, sendo usadas para o controlo de variáveis como por exemplo, o nível de água, de pressão e de temperatura.

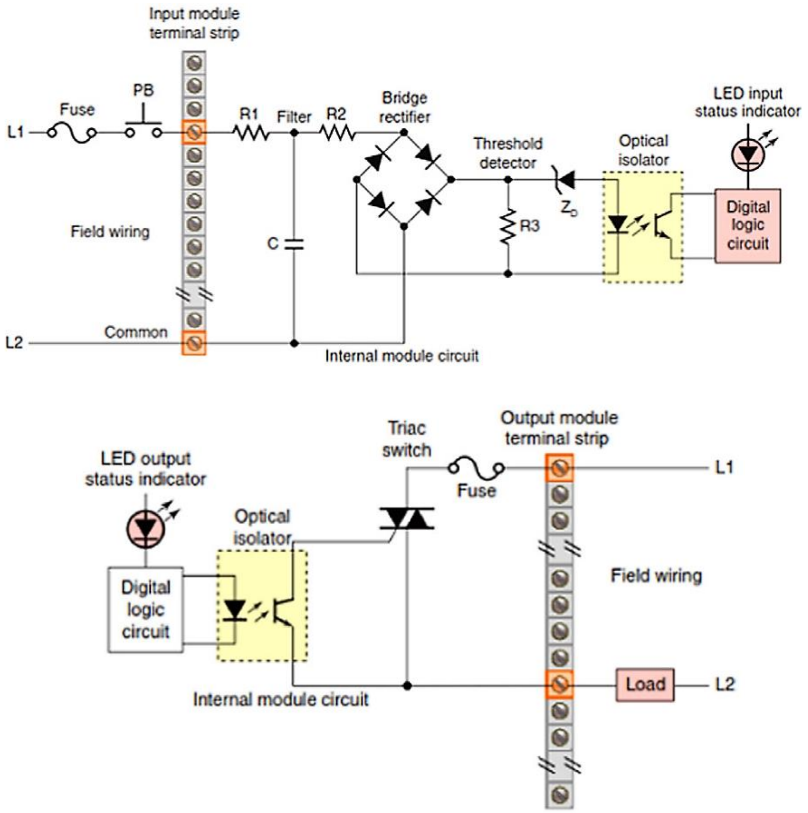


Figura 2.3 – Diagramas simplificados para uma entrada e saída digitais [8]

Uma parte importante de um automatismo é a sua versatilidade de programação. Para permitir uma uniformidade na linguagem de programação, a IEC (*International Electrotechnical Commission*) publicou a norma IEC 61131-3 que prevê a possibilidade de utilização de cinco linguagens de programação, três gráficas e duas baseadas em texto [6]. São elas, respetivamente:

- Diagrama funcional sequencial ou *grafcet* (SFC)
- Diagrama de blocos lógicos (FBD)
- Diagrama de contactos ou *ladder* (LD)
- Linguagem textual estruturada (ST)
- Lista de instruções (IL)

Todas se regem pelo mesmo princípio, estando estruturadas de forma semelhante, sendo a programação *ladder* a mais famosa e usada no mundo industrial por aproveitar os conhecimentos de eletricidade dos operadores, facilitando assim a sua manipulação. Para além de pretender facilitar a compreensão dos operadores dos controladores lógicos programáveis, a linguagem *ladder* surge como uma maneira de simular os esquemas dos circuitos elétricos, replicando o funcionamento lógico dos relés. É um tipo de programação gráfica em que a sua arquitetura é bastante linear, usando contatos e blocos com funções especiais como contadores e temporizadores, para energizar as saídas. Estes blocos são dispostos horizontalmente em ramos ou sub-ramos, interligados por uma linha que representa os cabos de ligação. Em *ladder*, a alimentação é representada por uma linha vertical à esquerda (fase) e outra à direita do esquema (neutro – podendo nem sempre ser representada). A sua leitura e execução é realizada horizontalmente da esquerda para a direita e verticalmente de cima para baixo [10].

Nos últimos anos têm aparecido diferentes soluções para facilitar o manuseio dos PLC's por parte dos técnicos. Uma das soluções existentes atualmente no mercado é da Phoenix Contact, com a gama de PLC's "Next". Esta permite usar linguagens de programação de alto nível para além das mencionadas na norma IEC 61131-3, como C, C++ e MATLAB/simulink. Desta forma é possível alargar o espectro de utilização, permitindo uma adaptação às tarefas a executar. É ainda possível integrar com uma plataforma IoT de *cloud* baseado na rede PROFINET, intitulado de PROFICLOUD. Outra solução é a usada na série de PLC's MELSEC-Q da Mitsubishi Electric. Nesta gama de PLC's modulares é possível adquirir um módulo que permite, em conjunto com o software VxWorks, programar o controlador através da linguagem C [9].

Inicialmente, o papel do PLC na automação industrial baseava-se essencialmente numa solução centralizada. Esta solução era problemática pois apresentava problemas de interferências eletromagnéticas, comprometendo desta forma os dados recebidos pelo PLC. Necessitava ainda de cablagens com uma grande extensão que constituíam custos mais elevados de instalação. Para os resolver, foram pensadas soluções descentralizadas através de DCS (*Decentralized Control System*). Desta forma seria possível diminuir a complexidade dos sistemas, pois são divididos em subsistemas mais simples, localizados mais perto do processo industrial e interligados entre si através de uma rede de comunicação, denominada de rede de campo [11]. A sua representação pode ser verificada na figura 2.4.

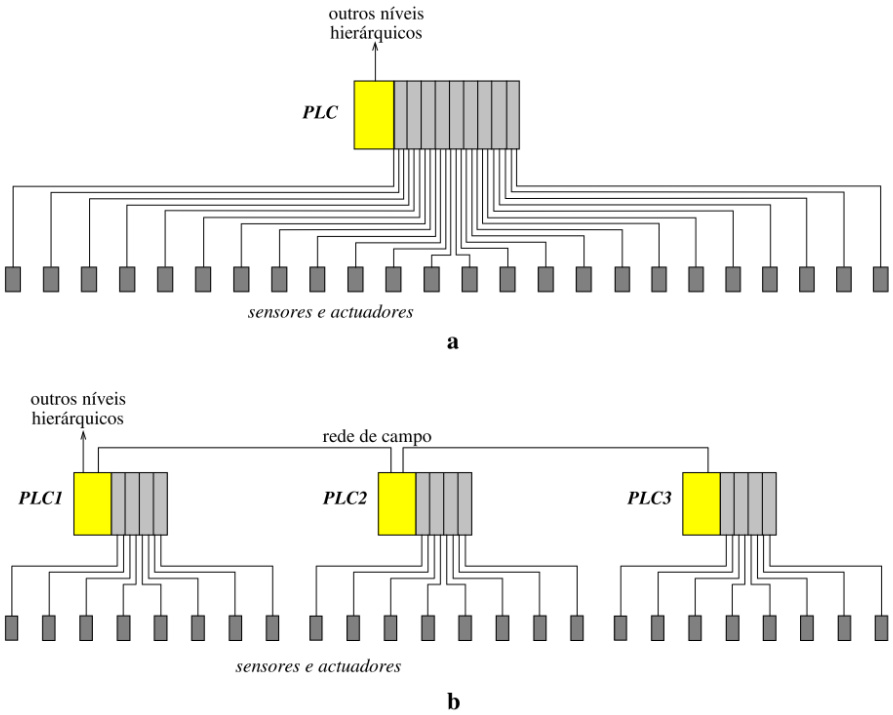


Figura 2.4 - Exemplos da utilização do PLC numa solução centralizada (a) e descentralizada (b) [11]

2.3 Comunicação na Automação Industrial

2.3.1 Redes e Métodos de Acesso

A descentralização do PLC na automação industrial implicou necessariamente constantes melhorias nas redes de campo. Assim como os PLC evoluíram dos computadores domésticos, também as redes em automação evoluíram das redes domésticas, por necessidade de algo mais robusto e fiável. As redes de campo são por norma redes LAN (*local area network*), podendo ser divididas em três níveis, de acordo com os equipamentos incluídos: superior (*fieldbus*), intermédio (*devicebus*) e de sensor (*sensorbus*).

As redes de campo *fieldbus* são redes de nível alto usadas para interligar os equipamentos mais sofisticados, normalmente usados para a supervisão das mesmas, com tempos nas transações de dados na ordem das centenas de ms. As redes de campo *devicebus* são redes de nível médio com transações de dados na ordem das dezenas de ms, normalmente usadas para interligar equipamentos de controlo. As redes de campo *sensorbus* são redes de nível baixo normalmente usadas ao nível do processo, para interligar sensores e atuadores. Estas têm tempos na ordem dos µs. As redes *sensorbus* têm a particularidade de poderem ser usadas para alimentar os dispositivos com consumos inferiores a algumas dezenas de mA [11]. Na figura 2.5 estão mencionados alguns exemplos dos diferentes níveis das redes de campo.

Redes de campo em automação industrial	nível superior	**MAP** (*General Motors*), **Ethernet** (*Xerox, Digital, Intel*), **Profibus FMS** (*Siemens*), **Fipway** (*Telemecanique, Celec*), **ControlNet** (*Allan-Bradley*), **HSE** (*Foundation Fieldbus*), **Profinet** (*PI*)
	nível intermédio	**Profibus DP** (*Siemens*), **WorldFip** (*Telemecanique, Cegelec*), **Interbus-S** (*Phoenix Contact*), **Genius I/O** (*General Electric*), **LONworks** (*Echelon*), **ArcNet** (*EIA*), **SDS** (*Microswitch-Honewell*), **Modbus plus** (*Modicon*), **H1** (*Foundation Fieldbus-ISA*), **SERCOS** (*VDW/ZVEI*)
	nível de sensor	**CAN** (*Bosh*), **AS-i** (*Siemens*), **Seriplex** (*APC/Square D*), **Sensoplex** (*Ford alemã*), **HART** (*Rosemount*), **DeviceNet** (*Allan-Bradley*), **H1** (*FF-ISA*), **WorldFip** (*Telemecanique, Cegelec*), **Profibus PA** (*Siemens*)

Figura 2.5 – Exemplos de redes de campo [11]

As redes podem ter diversas topologias, representadas na figura 2.6, como anel e estrela, sendo o barramento a mais usada [11]. Independentemente da topologia, para que a informação seja transmitida corretamente, é imprescindível que apenas um dos equipamentos envie mensagens de cada vez.

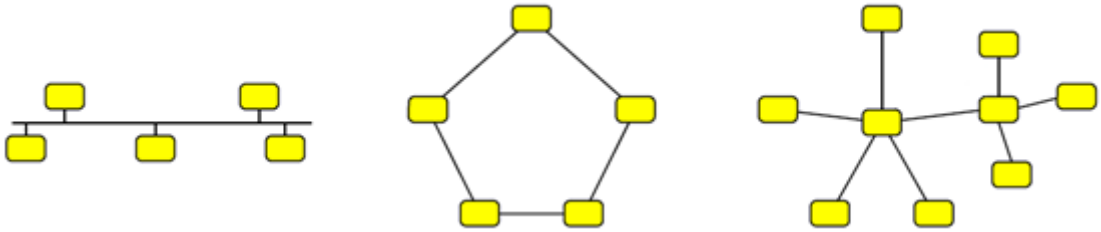


Figura 2.6 – Topologias de rede (da esquerda para a direita): barramento, anel e estrela [11]

Para tal, existem diversos modos de acesso à rede, de onde se destacam: *master-slave*, passagem de testemunho e espontâneo. No modo *master-slave*, apenas um participante – o *master* – pode realizar o acesso espontâneo à rede para enviar uma mensagem. A mensagem pode ser endereçada a todos os *slaves* ou apenas para um, ficando a aguardar por uma mensagem de resposta. O *master* é o único que pode iniciar a comunicação, pelo que os *slaves* apenas podem enviar mensagens de resposta ao *master* da rede. [11].

O modo de passagem de testemunho permite que cada elemento da rede tenha permissão cíclica, durante um período de tempo, de acesso à mesma. Sendo uma permissão cíclica, este modo de acesso é mais usual em topologias de rede em anel [11].

Finalmente, no modo de acesso espontâneo à rede cada participante apenas inicia a comunicação se não existir atividade na mesma. Dado que este modo de acesso é espontâneo poderão surgir colisões de dados. Para resolver este problema existem métodos de deteção e resolução de colisões, como o CSMA (*Carrier Sense Multiple Access*) tendo duas vertentes que explicam como é realizada a recuperação caso haja colisão de dados: CD (*Collision Detection*), normalmente usada em redes maiores

e BA (*Bitwise Arbitration*) normalmente usada em redes mais pequenas e com pouco fluxo de dados. Na vertente CSMA/CD, o emissor ao enviar uma mensagem continua a ler os dados presentes na rede. A eventualidade de existência de discrepâncias, indica que existiu um erro ou colisão. Nesse caso, o emissor suspende a comunicação e aguarda um intervalo de tempo aleatório até voltar a enviar novamente. Na vertente CSMA/BA esta verificação é realizada em cada *bit* onde, no caso de existência de colisão, é dada a prioridade ao *bit* 0. Nesta situação, o emissor que escreveu o *bit* 0 continua a sua transmissão e os restantes param de comunicar [11].

2.3.2 Modelo OSI

Numa rede, para que duas máquinas consigam comunicar é essencial que falem a mesma linguagem e, para tal, surge o conceito de protocolo. Este é o modo de formar mensagens, transmitindo-as com certeza de que são entregues corretamente [11].

Durante muitos anos os protocolos existentes eram exclusivamente propriedade de cada fabricante, não sendo permitido a integração de diferentes equipamentos. Foi em 1978 que começou a ser estruturado um modelo universal utilizado na comunicação digital em redes pela *International Standards Organization* (ISO), intitulado de modelo OSI (*open systems interconnection*) [11]. Este é constituído por 7 camadas, passando a constituir a norma ISO 7498:

1. Física;
2. De ligação;
3. De rede;
4. De transporte;
5. De sessão;
6. De apresentação;
7. De aplicação.

O objetivo era que este modelo criasse, de alguma forma, regras no desenvolvimento e na compatibilização de redes de comunicação o que não se veio a concretizar na íntegra. Cada camada tem a sua função e hierarquia, tal como se descreve na figura 2.7 [12].

Modelo OSI			
Camada		**PDU**	**Função**
7	Aplicação	Dados	API de alto nível, partilha de recursos, acesso remoto a ficheiros e terminais virtuais.
6	Apresentação		Transferência de dados entre a rede e a aplicação, inclui encriptação e compressão de dados.
5	Sessão		Gestão da comunicação entre dois nós, sincronização e deteção de falhas.
4	Transporte	TCP/UDP	Controlo de transmissão, reconhecimento, multiplexagem e controlo de congestionamentos.
3	Rede	Pacotes	Encaminhamento entre nós da mesma rede, controlo de trafego, *routing* e endereçamento.
2	Ligação	Tramas	Métodos de acesso, composição de tramas.
1	Meio Físico	*Bits*	Transmissão e receção de bits através de meio físico.

Figura 2.7 – Modelo OSI [13]

Os dados são processados em cada camada, que encaminha a mensagem para a seguinte, seja inferior ou superior. Nas três camadas mais baixas encontra-se especificado o modelo de rede e nas restantes superiores o modelo de aplicação.

A camada física define o meio físico que é usado para transmitir uma mensagem. É nesta camada que são especificadas todas as características de *hardware* (interface) e parâmetros funcionais, com codificação ao nível do *bit*. Na camada de ligação são definidos os métodos de acesso e realizado um controlo de erros, mas é na camada de rede que são definidos os diferentes caminhos a tomar pelos *packets* de dados entre nós, dentro de uma mesma rede ou de sub-redes diferentes. É realizado ainda o controlo de tráfego, *routing* e endereçamento. A camada de transporte é responsável pela transmissão de dados e pelo controlo de congestionamentos. Na camada de sessão é realizada a gestão e o controlo da comunicação entre dois nós. É na camada de apresentação que são transferidos os dados entre a rede e a aplicação, sendo a camada de aplicação que contém os serviços de informação, fornecendo a interface da aplicação [12].

Foi graças a este modelo que foi possível a uniformização da comunicação. Cada protocolo é baseado nas várias camadas do modelo OSI, não tendo necessariamente de usar todas, facilitando assim a integração de sistemas. Atualmente, diversas redes industriais usam protocolos do modelo TCP/IP.

2.3.3 Modelo TCP/IP

O modelo TCP/IP (*Transmission Control Protocol/Internet Protocol*) é um grupo de protocolos hierárquico, constituído por quatro camadas: 1- *host-to-network*, 2- rede, 3- transporte, 4 – aplicação, tendo sido desenvolvido antes do modelo OSI. Por vezes é feita uma comparação com as camadas do modelo OSI, onde a camada *host-to-network* engloba as camadas do meio físico e de ligação, as camadas de rede e de transporte acabam por ter uma ligação direta ao modelo OSI e a camada de aplicação engloba as últimas três camadas do modelo OSI [14]. Essa comparação encontra-se representada na figura 2.8:

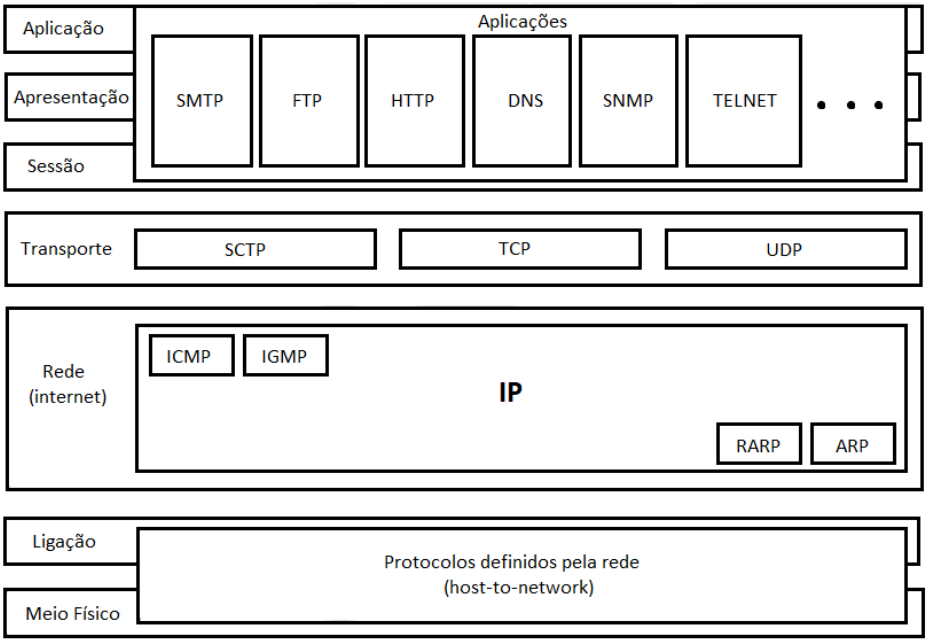


Figura 2.8 – Relação entre o modelo OSI e TCP/IP [14]

Contudo, enquanto o modelo OSI define uma funcionalidade específica do protocolo nas suas camadas, o TCP/IP define pequenos protocolos independentes entre si [14].

2.3.4 Interfaces Físicas

As interfaces físicas mais comuns nas redes de automação são baseadas nas normas RS-232 e RS-485. Estas são duas das três normas importantes, criadas pela associação americana EIA (*Electronics Industries Association*), para interfaces de comunicação série. Cada uma define o tipo de ligação física, meio de transmissão de dados entre equipamentos e correspondência entre níveis lógicos e sinais elétricos [11].

A norma RS-232 define ligações ponto-a-ponto, entre dois equipamentos, com transmissão realizada por sinais elétricos em tensão. Para os dados, ao nível lógico “0” corresponde o nível de tensão +3V a +15V e ao nível lógico “1” corresponde o nível de tensão -3V a -15V e, para os sinais de controlo

são usados sinais com a polarização oposta. Esta permite taxas de transmissão até 20 kbit/s e distâncias até 15m [11].

A norma RS-485, por sua vez, permite ligações até 32 equipamentos. Utiliza um sinal diferencial com valores de -1.5V a -6V para o nível lógico "0" e de +1.5V a +6V para o nível lógico "1" para os dados e a polarização oposta para os sinais de controlo, com taxas de transmissão até 35 Mbit/s e distâncias até 1000m [11].

2.3.5 Meios de Transmissão

Os meios de transmissão utilizados em redes de campo estão representados na figura 2.9 e podem ser por cabo ou pelo ar. Dentro dos meios por cabo existem os de par entrelaçado, coaxiais e de fibra ótica [14].

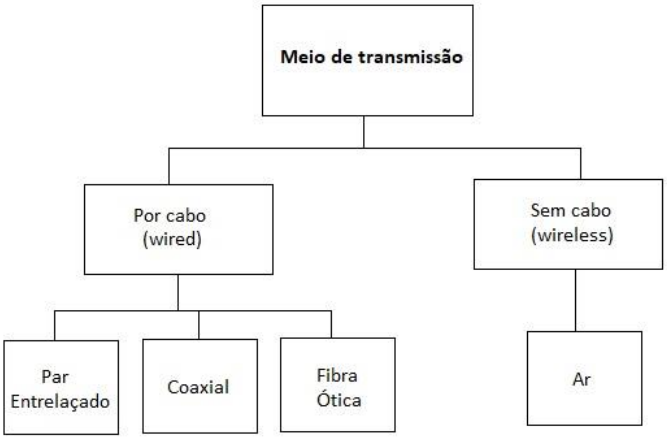


Figura 2.9 – Meios de transmissão utilizados em redes de campo

Os cabos de par entrelaçado, do inglês *twisted pair*, são constituídos por um ou diversos pares de condutores de cobre entrelaçados. O entrelaçar em pares é uma medida para reduzir as interferências eletromagnéticas normalmente causadas por circuitos de potência. Este tipo de cabo, para além de ter os condutores entrelaçados a pares, pode ou não ser blindado [14]. A blindagem pode ser individual em cada par ou geral no cabo, com malha metálica ou folha de alumínio, sendo classificados segundo a norma NP922 [15]:

- U/UTP – Sem blindagem
- F/UTP – Blindagem geral com folha de alumínio
- FF/UTP – Blindagem geral com duas folhas de alumínio
- SF/UTP – Blindagem geral com malha metálica e folha de alumínio
- U/FTP – Blindagem individual com folha de alumínio

- S/FTP – Blindagem geral com malha metálica e individual com folha de alumínio
- F/FTP – Blindagem geral e individual, ambas com folha de alumínio

São ainda categorizados de acordo com a sua frequência máxima de operação. Atualmente a norma TIA-568.2-D reconhece cinco categorias [16], tal como apresentado na tabela 2.2.

Tabela 2.2 – Categorias segundo TIA-568.2-D [16]

Categoria	**Frequência máxima de operação**
3	16 MHz
5e	100 MHz
6	250 MHz
6A	500 MHz
8	2000 MHz

Os cabos de par entrelaçado são os mais comuns em automação industrial pela sua versatilidade, ao contrário dos cabos coaxiais. Estes são bastante usados em telecomunicações pela sua prestação com frequências altas, atingindo taxas de transmissão de Mbit/s ou até Gbit/s em distâncias superiores a 100 metros [14]. Contudo, a sua utilização requer especial atenção na montagem, na terminação e derivação.

Os cabos de fibra ótica, por outro lado, são bastante versáteis relativamente à taxa de transmissão, distância e blindagem. Estes são compostos por um núcleo cilíndrico flexível de vidro ou plástico extrudido, envolvidos por um material com baixo nível de refração usado como isolamento e uma bainha exterior de plástico para proteção mecânica. Como usam um feixe de luz em vez de sinais em tensão, ao contrário dos tipos de cabos explicados anteriormente, estes não são suscetíveis a interferências eletromagnéticas podendo existir algum efeito de refração consoante o diâmetro dos cabos [14].

A transmissão de dados sem cabo (*wireless*) é realizada usando o ar como meio de transmissão, para a propagação de ondas eletromagnéticas. Este tipo de transmissão é bastante vantajoso pelas suas características estando, no entanto, sujeito a diferentes tipos de interferência, seja física, eletromagnética ou mesmo meteorológica. [11].

A transmissão *wireless* pode ser com ou sem linha de vista, dividindo-se em três grupos, de acordo com o tipo de onda: ondas rádios, micro-ondas e infravermelhos. O intervalo de frequências do espectro eletromagnético para ondas rádios e micro-ondas vai de 3kHz a 300GHz sendo classificadas em oito bandas de transmissão de acordo com a sua frequência. [14]. Na tabela 2.3 são apresentadas as características das bandas de transmissão.

Tabela 2.3 – Bandas de transmissão [14]

Banda	Intervalo de frequências	Propagação
VLF (*very low frequency*)	3 – 30 kHz	Sem linha de vista
LF (*low frequency*)	30 – 300 kHz	Sem linha de vista
MF (*middle frequency*)	300 kHz – 3 MHz	Sem linha de vista
HF (*high frequency*)	3 – 30 MHz	Sem linha de vista
VHF (*very high frequency*)	30 – 300 MHz	Com e sem linha de vista
UHF (*ultra high frequency*)	300 MHz – 3 GHz	Com linha de vista
SHF (*super high frequency*)	3 – 30 GHz	Com linha de vista
EHF (*extremely high frequency*)	30 – 300 GHz	Com linha de vista

No entanto, as bandas de transmissão mais usadas em automação são as VHF, UHF e SHF tal como está representado na figura 2.10. As ondas de infravermelhos funcionam com frequências superiores a 300GHz e são usadas para comunicação de curto alcance, num espaço fechado e com linha de vista [14].

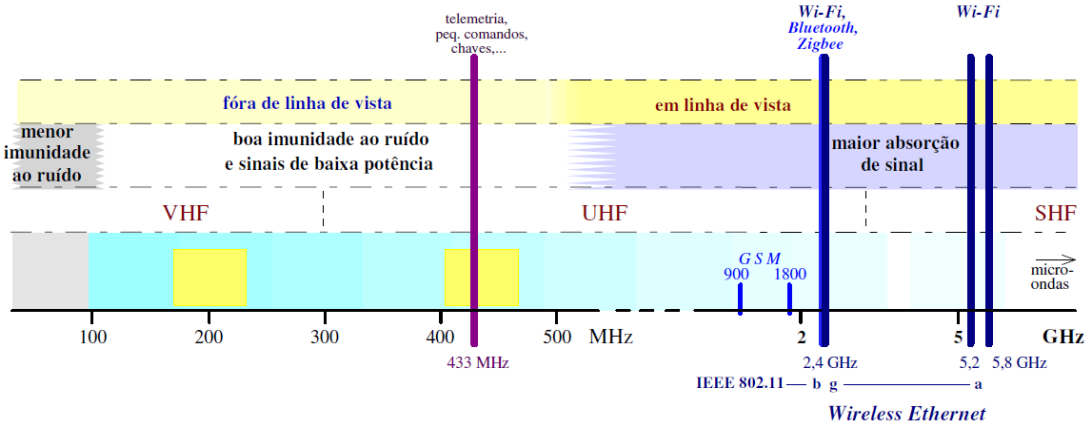


Figura 2.10 – Espetro de frequências [11]

Atualmente as tecnologias mais usadas na automação industrial, dentro da transmissão wireless, são o *bluetooth*, o *wifi* e *zigbee*. Encontram-se todas reguladas pela norma 802 desenvolvida pelo IEEE e estão representadas na tabela 2.4.

Tabela 2.4 – Lista de tecnologias wireless mais usadas [14]

Norma	Designação	Faixas de frequência	Débitos	Alcances típicos (interior/exterior)
IEEE 802.11. a	*Wi-Fi*	5.2; 5.8 GHz	6 – 54 Mbits/s	10 m / 140 m
IEEE 802.11. b	*Wi-Fi*	2.4 GHz	1 – 11 Mbits/s	10 m / 120 m
IEEE 802.11. g	*Wi-Fi*	2.4 GHz	6 – 54 Mbits/s	10 m / 140 m
IEEE 802.15.4	*Zigbee*	2.4 GHz	250 kbits/s	10 m / 100 m
IEEE 802.15.1	*Bluetooth*	2.4 GHz	723.3 kbits/s	1 m / 100 m

Em Portugal, a entidade que regula todos os parâmetros relativos à comunicação é a ANACOM. Para a elaboração de projetos de ITED em edifícios industriais é necessária a consulta das normas EN 50173-2 e EN 50173-3 [15].

2.3.6 Rede Ethernet

A arquitetura de rede Ethernet é atualmente a mais usada para a transmissão de dados em rede. A sua criação é atribuída ao Centro de Pesquisa Xerox Palo Alto, em meados de 1990. É normalmente dividida em 3 partes, baseadas no modelo OSI: Encontra-se regulada pela IEEE com as normas 802-2 e 802-3, abrangendo as camadas 1 e 2 do modelo OSI [14]. Usa o modelo TCP/IP já descrito anteriormente, nas camadas centrais e, por fim, a camada de aplicação [17].

Na norma 802-3 estão descritas as topologias de rede, meio físico para transmissão de dados e ritmos de transmissão. Atualmente a rede Ethernet usa o método de acesso CSMA/CD e pode usar cabo coaxial, pares entrelaçados ou fibra ótica como meio de transmissão, com taxas de transmissão de 1 Mbits/s a 400 Gbits/s [18].

2.3.7 Ethernet Industrial

As redes de Ethernet industrial usam como base a arquitetura da rede Ethernet, diferenciando-se pelo ambiente a que estão expostas. Por serem desenvolvidas para o meio industrial, as redes Ethernet industrial têm *hardware* específico, robusto e fiável, capaz de suportar as características adversas da indústria.

Os problemas principais da arquitetura da rede Ethernet prendem-se na velocidade, controlo e fiabilidade da comunicação. No meio industrial são necessárias operações em tempo real, sendo difícil de atingir com o método de acesso CSMA/CD pois este implica um tempo de espera entre mensagens. No entanto este método, por realizar um controlo de colisões, acrescenta a fiabilidade necessária às operações de controlo industriais. De forma a colmatar esta situação, foram migrados diversos protocolos para a arquitetura Ethernet, permitindo operações em tempo real e adicionando novos mecanismos de deteção de colisão [19].

Alguns exemplos de protocolos Ethernet industrial são: PROFINET, EtherCat e ModbusTCP.

2.3.8 Modbus

O protocolo Modbus foi desenvolvido pela firma Modicon Incorporated, em 1979, para os seus autómatos. As suas especificações estão bastante divulgadas e atualmente encontra-se aplicado em numerosos sistemas industriais de diversos fabricantes. É um protocolo aberto, implementado na

camada 7 do modelo OSI (aplicação), normalmente usado para transferência de dados discretos/analógicos entre controladores industriais e equipamentos a ser monitorizados, principalmente na comunicação entre aplicações SCADA/HMI e autómatos. Para já, encontra-se implementado em transmissão série assíncrona por cabo, TCP/IP via Ethernet, entre outros [20].

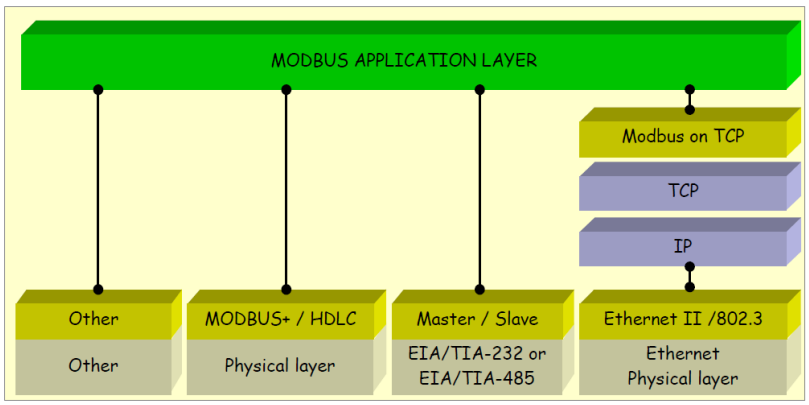


Figura 2.11 – Arquitetura Modbus [21]

Define um PDU (*Protocol Data Unit*) simples independente das várias camadas de comunicação, podendo ser adicionados diferentes campos à ADU (*Application Data Unit*), dependendo da rede usada. Utiliza o método de acesso *master*/*slave* (cliente/servidor) e, como tal, só o *master* pode tomar a iniciativa de enviar uma mensagem. A mensagem pode ser endereçada a um *slave* específico ou enviada em *broadcast* para todos. Os *slaves* apenas respondem à mensagem endereçada, enviando uma mensagem de volta ou executando o pedido iniciado, mas nunca respondem a mensagens em *broadcast*. A mensagem do *master* é composta por quatro campos: o endereço do *slave* ou de *broadcast*, um código funcional que define a ação a executar, alguma informação necessária e um campo de controlo de erros. A resposta do *slave* consiste em três campos: a confirmação da ação, qualquer informação que tenha de ser devolvida e o campo de controlo de erros [22].

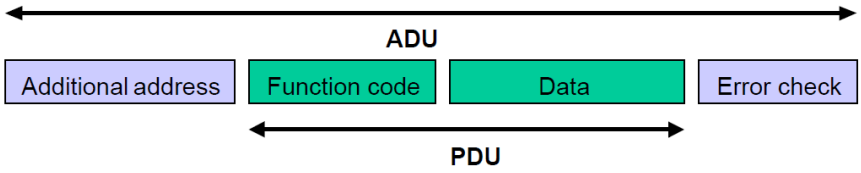


Figura 2.12 – Estrutura de uma mensagem Modbus [22]

As mensagens do protocolo Modbus estão limitadas a 256 bytes. Para transmissão em comunicação série, retirando o byte relativo ao endereço do *slave* e os 2 bytes do controlo de erros, concluímos que o PDU tem um máximo de 253 bytes [21].

2.3.8.1 Modbus TCP/IP

No caso do Modbus TCP/IP é usada uma interface TCP/IP para transmitir a estrutura de mensagem Modbus. Resumidamente, a camada de aplicação que define as regras para organização e interpretação dos dados a serem transmitidos é baseada na arquitetura do protocolo Modbus RTU, enquanto o meio de transmissão é assegurado pelo TCP/IP, já descrito no ponto 2.3.3 [20]. Nesse caso, a mensagem é composta pelo PDU do Modbus (253 bytes), precedida de um cabeçalho MBAP com 7 *bytes*, num total de 260 *bytes*, tal como representado na figura 2.13.

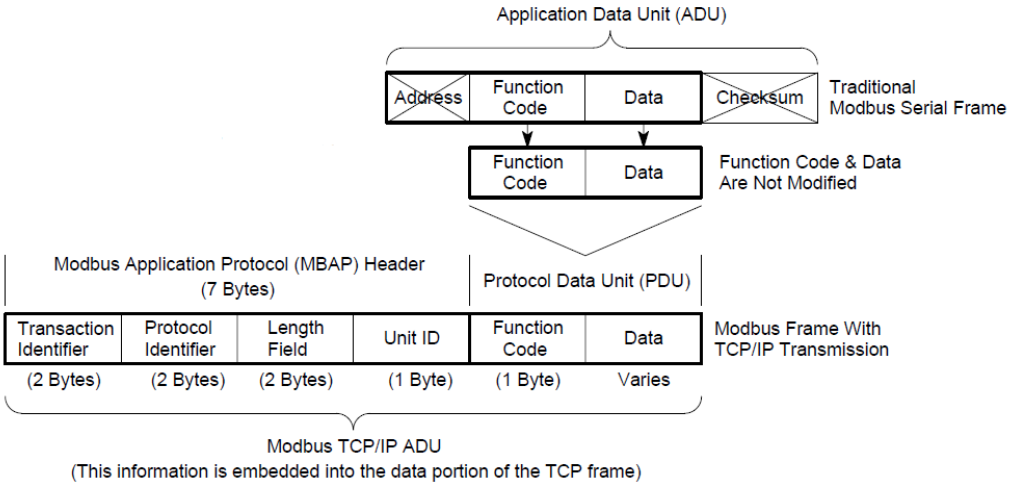


Figura 2.13 – Cabeçalho MBAP [20]

O cabeçalho MBAP define quatro campos – *transaction identifier*, *protocol identifier*, *length field* e *unit ID* – descritos na tabela 2.5.

Tabela 2.5 – Descrição dos campos do cabeçalho MBAP [22]

Nome	Tamanho	Descrição
Transaction Identifier	2 Bytes	Identifica o pedido/resposta da comunicação Modbus
Protocol Identifier	2 Bytes	É sempre 0 para serviços Modbus
Lenght Field	2 Bytes	Indica o tamanho dos restantes campos (Unit ID, Function Code e Data)
Unit ID	1 Byte	Identifica um servidor remoto numa rede que não seja TCP/IP. Por norma toma o valor 00 ou FF em aplicações Modbus TCP/IP, sendo ignorado pelo slave.

Os campos *transaction identifier*, *protocol identifier* e *unit ID* são inicializados pelo *master*, sendo copiados pelo *slave*. O campo *length field* é inicializado pelo *master* e pelo *slave*. Todos os ADU de Modbus TCP/IP são enviados para o porto 502 [22].

2.3.8.2 Mensagem Modbus

O protocolo Modbus define três tipos de PDU: de pedido, de resposta e de exceção. Aquando da iniciação da comunicação pelo *master*, é enviado um PDU de pedido ao qual se espera um PDU de resposta, indicando uma resposta por parte do *slave*, sem erros. Ambos os PDU são compostos por 1 *byte* com o código de função que indica o comando a executar pelo *slave* e n *bytes* de dados, dependendo do tipo de função pedida. No caso de, por qualquer motivo, a transmissão da mensagem não ser concluída com sucesso, o *slave* envia uma mensagem de resposta com código de exceção. Esta mensagem é composta por dois *bytes*: 1 *byte* com o código de função enviado pelo *master* e o *bit* mais significativo alterado para 1 e 1 *byte* com o código de exceção [21]. A lista de códigos de exceção pode ser encontrada na tabela 2.6.

Tabela 2.6 – Lista de códigos de exceção [11]

Código	Descrição
01	Código de função desconhecido
02	Endereço de variável com erro
03	Dado incorreto
04	Erro ao tentar executar o comando
05	O comando foi aceite, mas vai ser executado a seguir
06	Está ocupado e não pode receber o comando
07	Não aceita o comando

O modelo de mensagem deste protocolo define um endereço específico entre 0 e 65536 (endereço 0 = *bit* 1 do equipamento) para cada dado num PDU e permite aceder indiretamente aos dados de aplicação do servidor, através de quatro áreas de memória com características distintas. São elas: entradas digitais (*discretes input*), saídas digitais ou bobinas (*coils*), registos de entrada (*input registers*) e registos de saída ou retentivos (*output/holding registers*), tal como representado na tabela 2.7. No entanto deve ser realizada, localmente, um mapeamento entre estas áreas e os dados do servidor, normalmente disponibilizado pelo fabricante [21].

Tabela 2.7 – Descrição de áreas de memória [22]

Áreas	Tipo de objeto	Dados
Entradas Digitais	1 bit	Leitura
Saídas Digitais	1 bit	Leitura e escrita
Registos de Entrada	Word de 16 bit	Leitura
Registos Retentivos	Word de 16 bit	Leitura e escrita

2.4 Sistema de Supervisão (SCADA)

Com a descentralização do PLC na automação industrial surgiu a necessidade de ter um sistema interativo que receba, analise e guarde os dados, em tempo real. Esse sistema é a supervisão, do inglês SCADA (*Supervisory Control and Data Acquisition*).

O sistema SCADA é usado em praticamente todo o tipo de processos de diversas áreas: automação industrial, tratamento de águas, domótica, energia, transporte, entre outras. Com as suas funcionalidades de tratamento de dados em tempo real, tem como objetivos o controlo de qualidade e otimização de custos e produção, sendo o seu principal foco facilitar a manutenção em locais remotos. Por ser um sistema de controlo e visualização, este permite ao utilizador a monitorização e manipulação de um processo que esteja localizado num lugar remoto e de difícil acesso. Seja abrindo ou fechando válvulas ou contactos, monitorizando alarmes e estados dos sensores e atuadores ou recolhendo dados sobre o processo [23]. Desta forma não é necessário ter sempre uma pessoa nesse local, havendo uma melhor gestão de recursos.

É composto por um terminal *master* (MTU) e vários terminais remotos (RTU) ligados em rede. O controlo do processo é realizado localmente pelos RTU (*Remote Terminal Unit*) e a supervisão centralmente, através dos MTU (*Master Terminal Unit*).

2.4.1 Arquitetura do SCADA

A arquitetura moderna do sistema SCADA está dividida em quatro níveis hierárquicos: Sensores/atuadores, RTU's, redes de comunicação e MTU's. [24].

O MTU é, tal como o nome indica, o master do sistema de supervisão. Tem como componentes principais um servidor central, consolas de operador, ecrãs de visualização – normalmente um *videowall* – e uma HMI. Pode ter também uma impressora e servidores auxiliares.

Deve ser flexível, podendo receber qualquer tipo de entradas e saídas e ter a capacidade de filtrar alarmes. Dada a sua importância, deve ser redundante nos seus elementos de controlo e supervisão e de energia, a última através de UPS (*Uninterrupted Power Supply*). Deve ainda ser capaz de comunicar com qualquer tipo de protocolo, facilitando a integração de equipamentos para que não se torne obsoleto.

As unidades remotas ou RTU são compostas por um microprocessador ou controlador com capacidade de receber e enviar dados para o MTU. São elas que realizam o controlo do processo localmente, recebendo todos os dados do terreno recolhidos pelos sensores e dando ordens de ação aos atuadores. As unidades devem ser compactas e robustas, de forma a suportar ambientes adversos e ter redundância para comunicação, controlo e energia, à semelhança das MTU.

As duas unidades – MTU e RTU – comunicam através de uma rede de comunicações com características apropriadas à sua dimensão, seja quanto ao meio de transmissão ou ao protocolo usado. A figura 2.14 apresenta uma possível solução SCADA.

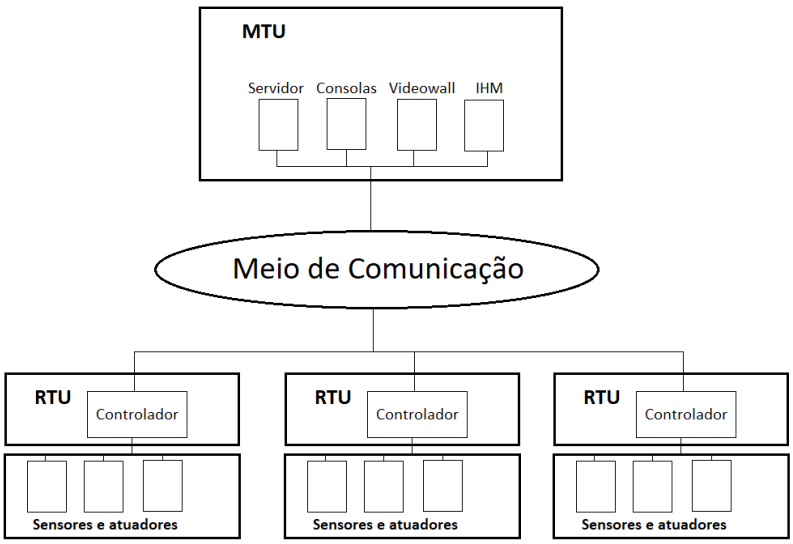


Figura 2.14 – Diagrama de blocos descritivo da arquitetura do SCADA

2.4.2 Software SCADA

Todo o sistema de supervisão assenta num software SCADA capaz de receber e processar a informação de campo, em formato digital. O software tem a capacidade de HMI, apresentando os dados em ecrãs sinópticos com a possibilidade de animações representativas dos processos. Recolhe e armazena os dados, sendo possível criar alarmes, relatórios, estatísticas e qualquer outro documento de controlo e processamento.

Por norma, o *software* SCADA divide-se em quatro módulos funcionais:

- Aquisição de dados (*Inputs/Outputs*)
- Base de dados (Armazenamento)
- Controlo e processamento (Alarmes, relatórios e estatísticas)
- Interface (*Displays*)

2.4.2.1 Funcionalidades do Software SCADA

A aquisição dos dados dos sensores é realizada pela interface de entradas e saídas das RTU's. A aquisição só por si não é suficiente, sendo necessário converter os sinais elétricos do mundo físico para o mundo digital. Feita a conversão, os sinais discretos e analógicos passam a ser representados com

valores binários e *words* ou *doublewords*. Os dados são depois armazenados em base de dados na memória interna do controlador e no sistema SCADA [25].

A conversão dos dados e a sua transmissão entre o MTU e as RTU é assegurada por um driver de comunicação ou pela tecnologia OPC, através da rede de comunicações, onde é especificado o protocolo a usar. Os drivers são responsáveis por transferir os dados recebidos da memória das RTU's e por enviar os comandos do MTU para os elementos de campo. Desta forma, a comunicação torna-se mais eficiente pois é possível a testagem imediata da ligação e cablagem, importação da base de dados interna do controlador e a ligação direta às entradas e saídas do controlador através de um sistema de *tags* [26].

Estas *tags*, para além de fazerem a ligação das variáveis da base de dados SCADA com as da memória do controlador, permitem ainda ao utilizador uma personalização da mesma com atributos como o nome, descrição e tipo de variável.

Estabelecida a relação entre a informação do MTU e a informação de campo das RTU's, é possível o controlo e processamento dos dados, para melhor visualização e interpretação dos utilizadores, através de relatórios, gráficos, alarmes, entre outros. Esta representação é apresentada ao utilizador através de uma interface gráfica, intitulada de Interface Humano – Máquina com o acrónimo HMI (*Human – Machine Interface*).

2.4.3 Human-Machine Interface (HMI)

A HMI é constituída por um ecrã ou painel, para visualização e controlo do sistema. A interface faz a ligação entre o sistema e o utilizador, apresentando-lhe de uma forma dinâmica os processos, alarmes, relatórios e estatísticas. Por norma, num sistema complexo de automação industrial, existe uma interface centralizada como elemento do MTU para representação geral do sistema e diversas HMI nos RTU's, oferecendo a possibilidade de supervisão e controlo local por parte do utilizador. As interfaces devem ser robustas e adaptáveis ao ambiente onde serão usadas.

2.4.4 Novas soluções SCADA

Com a evolução da tecnologia os fabricantes têm apostado cada vez mais em soluções de sistema SCADA poderosas com recurso da inteligência artificial, interfaces intuitivas e de uso fácil, com recursos de segurança e serviço *Web*, possibilitando uma supervisão a partir de qualquer lugar.

O sistema Movicon.Next 4.0 da Progea, por exemplo, é apresentado como *"the new industrial software platform"*, oferecendo diversas soluções de tecnologia orientadas para a indústria 4.0 [27]. O Movicon.Next 4.0 usa uma arquitetura aberta e modular, permitindo-o ser adaptável e não se tornar obsoleto com o passar dos anos. A sua comunicação é baseada no OPC UA, oferecendo também um

vasto número de drivers de comunicação para diversos protocolos. Está preparado para a integração de protocolos específicos IIoT, permitindo a aquisição e análise de dados na *cloud* (ver 2.5.2). Pode ser usado com variados ambientes industriais como tratamento de águas, energia, produção industrial e indústria automóvel, garantindo segurança na transmissão de dados pela norma IEC 62443-3-3. Com o Movicon.Next 4.0 é ainda possível aceder a um conjunto de opções no tratamento e monitorização de dados, remotamente, através de qualquer equipamento (computador, telemóvel ou tablet) e sistema operativo (figura 2.15) [27].

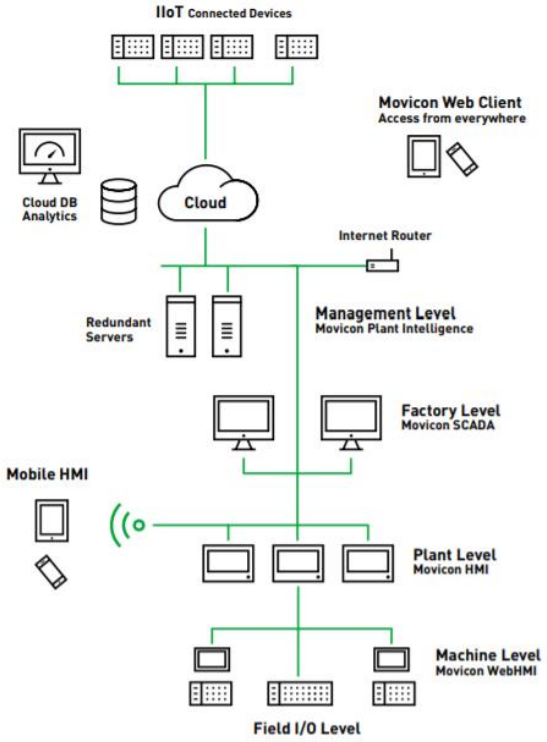


Figura 2.15 – Arquitetura Movicon.Next 4.0 [27]

2.4.5 Open Platform Communications (OPC)

A *Open Platform Communications* é um conjunto de especificações para a uniformização da transmissão de dados na automação industrial, desenvolvida pela OPC Foundation. Foi criada em 1996 com a intenção de ser um "intermediário" entre cliente e servidor, em sistemas de SCADA/HMI [28].

A interface OPC usa o método de cliente – servidor para a transmissão de dados. Para aceder aos dados, o cliente OPC cria uma ligação ao servidor OPC que tem os dados a transmitir guardados e acessíveis. A OPC Foundation desenvolveu várias especificações de interfaces, todas com a tecnologia COM e DCOM da Microsoft, sendo a OPC *Data Access* a mais relevante (figura 2.16) [29].

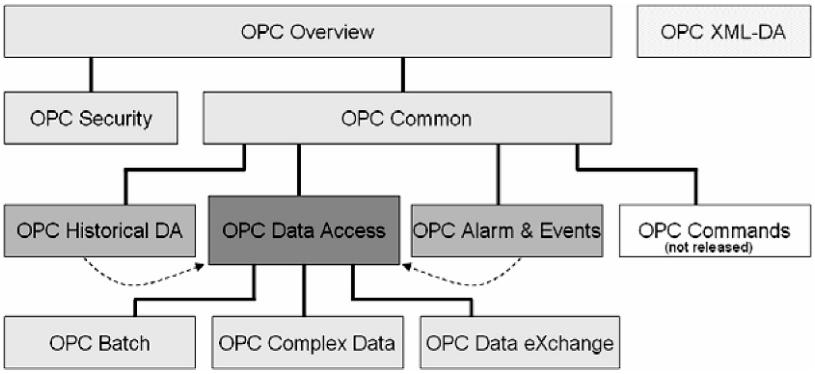


Figura 2.16 – Interfaces OPC [29]

A OPC *Data Access* (figura 2.17) permite a leitura/escrita e monitorização das variáveis de um determinado processo. É maioritariamente usada para transmissão de dados em tempo real, entre PLC e HMI, sendo a interface OPC a mais importante. Para aceder aos dados, o cliente OPC DA seleciona as variáveis que quer através dos OPC *items*. De forma a realizar uma conexão ao servidor cria um objeto (*OPCServer*) que lhe permite mover nos endereços de memória e encontrar os *items*. Esses *items* são agrupados num objeto denominado de *OPCGroup* por definições idênticas como, por exemplo, a data da última atualização dos dados. É definida uma taxa de atualização pelo cliente OPC, usada pelo servidor para realizar uma procura cíclica por alterações nos dados. Após cada ciclo, são enviados para o cliente OPC apenas os dados que sofreram alteração [29].

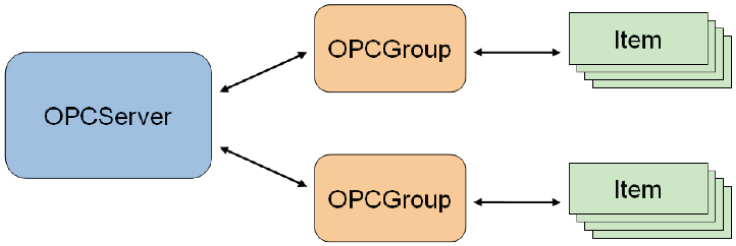


Figura 2.17 – OPC DA [29]

Desde a data da sua criação, as especificações OPC tornaram-se bastante famosas, sendo usadas em diferentes contextos de automação. No entanto, a sua expansão não é total devido à sua dependência com a tecnologia da Microsoft. Para além da falta de independência, junta-se a necessidade de robustez e fiabilidade com alta performance para substituir a comunicação proprietária dos diversos fabricantes. Estes foram os pontos que motivaram a criação de uma nova especificação de interface, a OPC *Unified Architecture* (figura 2.18).

Para a criação da OPC UA, a OPC Foundation juntou-se a diferentes organizações como a IEC e a ISA. Nesta parceria, a OPC define como descrever e transportar os dados, enquanto que as organizações tratam de definir que tipo de dados serão descritos e transportados [29]. A OPC UA (IEC

62541), engloba as especificações anteriores da OPC e permite a troca de dados de qualquer tipo de complexidade, tendo sido pensada para suportar uma vasta escolha de sistemas, seguindo um modelo por camadas [30].

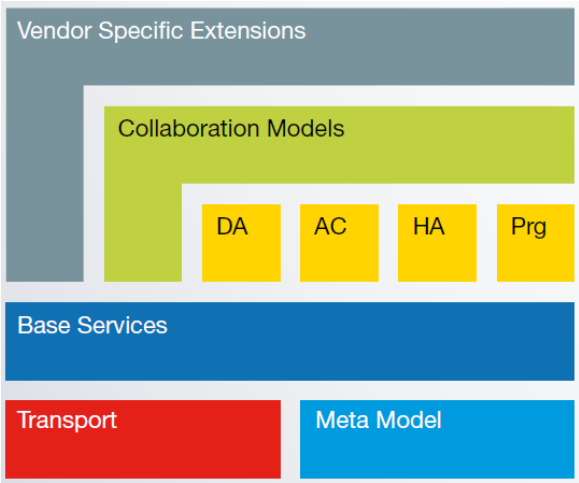


Figura 2.18 – Modelo OPC UA [30]

A camada de transporte define os mecanismos de transmissão de dados entre aplicações OPC-UA. A camada "*meta model*" define as regras de acesso aos dados do servidor pelo cliente e a camada "*services*" funciona como interface entre servidor e cliente [30].

Devido às suas características de segurança, fiabilidade e de plataforma independente da transmissão de dados, a OPC UA tornou-se numa das soluções mais importantes na 4ª revolução industrial.

2.5 Indústria 4.0

Em 2006, aproveitando o crescimento do movimento de IoT, o governo alemão apresentou o documento "*High Tech Strategy*" que representa o primeiro esforço e investimento coletivo nacional em tecnologias avançadas. Mais tarde, em 2010, o documento foi estendido a mais áreas de investimento passando a chamar-se "*High-Tech Strategy 2020*", apresentando 10 projetos do futuro. Entre outros, foi apresentado o projeto *Industrie 4.0* (Indústria 4.0), que tinha como objetivo uma visão de indústria integrada [13].

O termo indústria 4.0 define a evolução tecnológica para sistemas ciber-físicos, do inglês *Cyber-Physical Systems* (CPS), representando a 4ª revolução industrial. O conceito da 4ª revolução industrial passa por construir fábricas inteligentes (*smart factories*), descentralizadas e personalizadas, deixando o paradigma da produção em massa. Com a 4ª revolução industrial, a produção é personalizável e adaptável ao mundo exterior ao invés dos processos convencionais. Em suma, tal como é mencionado

no artigo "Industries 4.0: smart manufacturing for the future", pela German trade & invest: *"[…] this means that industrial production machinery no longer simply 'processes' the product, but that the product communicates with the machinery to tell it exactly what to do."* [31].

A grande vantagem da indústria 4.0 é a otimização das linhas de produção, por serem totalmente integradas e automatizadas na relação humano-máquina, misturando o físico e o digital, aumentando assim a sua eficiência [32]. Para tal, são definidos quatro conceitos: *Internet of Things* (IoT), *Industrial Internet of Things* (IIoT), *Cloud-based Manufacturing* e *Smart Manufacturing* [33].

2.5.1 Internet of Things (IoT) e Industrial Internet of Things (IIoT)

A maneira como as máquinas interagem com o mundo não é recente, sendo um assunto já debatido há alguns anos. Em 1999, durante uma apresentação de Kevin Ashton sobre o mesmo, foi proferido termo "*Internet of Things*" (IoT). Desde então a definição de "*Internet of Things*" tem evoluído podendo ser descrita como o conceito de interação através de sensores e atuadores ligados em rede [34].

Esta interação em rede permite a aquisição de um conjunto largo de dados em tempo real, abrindo possibilidades imensas com o tratamento dos mesmos. Seja na área comercial, domótica ou comercial, o acesso a um largo conjunto de dados em tempo real deu origem ao conceito de *big data*. Este conceito pode ser bastante vantajoso, especialmente quando aplicado à indústria, se o processamento dos dados for bem executado [35].

Juntamente com a *internet of things*, surge a IIoT – *Industrial Internet of Things*. A IIoT refere-se à aplicação da tecnologia IoT ao ambiente industrial. É, tal como mencionado por Hugh Boyes em "*The industrial internet of things (IIoT): An analysis Framework"*: *"A system comprising networked smart objects, cyber-physical assets, associated generic information Technologies and optional cloud or edge computing platforms, which enable real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, so as to optimise overall production value. This value may include; improving product or service delivery, boosting productivity, reducing labour costs, reducing energy consumption, and reducing the build to order cycle."* [36]

2.5.2 Cloud-based Manufacturing e Smart Manufacturing

Para além da IoT e IIoT, surgem os conceitos de *cloud-based manufacturing* e *smart manufacturing* na base da estrutura da indústria 4.0.

A *cloud-based manufacturing* usa o modelo de plataforma de *cloud* ou nuvem como base para o processamento da *big data* no contexto industrial. O modelo de *cloud* usa um conjunto de servidores,

que não têm de estar fisicamente perto, para oferecer ao cliente recursos, serviços e ferramentas de processamento que de outra forma não seriam alcançáveis [37].

A produção inteligente, do inglês *smart manufacturing*, complementa os conceitos anteriores com outras ferramentas, tornando-se mais eficaz. Esta tira partido da simulação 2D e 3D onde é possível esquematizar virtualmente um processo físico, da evolução dos robots autónomos e da inteligência artificial e realidade aumentada [32]. Com as tecnologias de inteligência artificial e realidade aumentada é possível diminuir os tempos de falha, podendo prever avarias ou ajudar um operador durante uma manutenção corretiva. Um exemplo de plataforma de realidade aumentada é a *EcoStruxure Augmented Operator Advisor* da Schneider. Esta plataforma permite ao operador efetuar o diagnóstico instantâneo de uma avaria, sem contacto, aumentando dessa forma a sua eficiência e diminuindo os custos. É possível diagnosticar um quadro elétrico, por exemplo, sem ter de o abrir nem parar a produção [38] (figura 2.19 e figura 2.20).

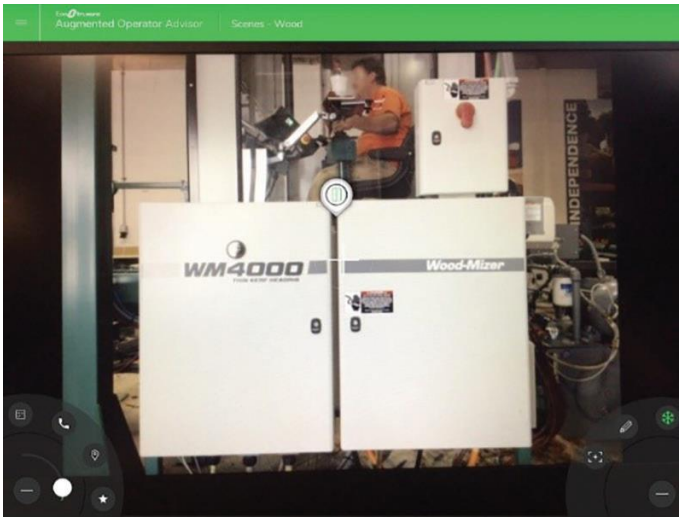


Figura 2.19 – Visualização do quadro elétrico físico fechado [38]

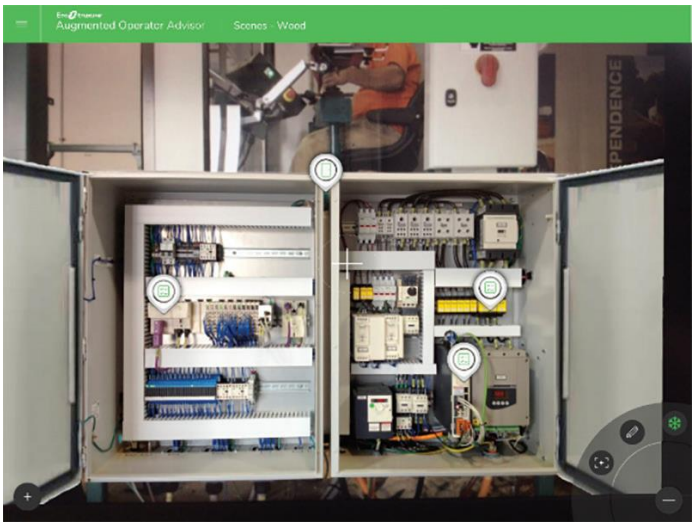


Figura 2.20 – Representação virtual do quadro elétrico aberto [38]

A 4º revolução industrial permitiu o desenvolvimento de ferramentas necessárias para a criação de ecossistemas industriais. Usando ainda o exemplo da empresa Schneider, esta tem disponível atualmente um ecossistema composto por diferentes elementos como o PLC *Modicon M262*, a HMI *Magelis STW6* e a plataforma de realidade aumentada *EcoStruxure Augmented Operator Advisor*. À semelhança do PLC next da Phoenix Contact já mencionado anteriormente, também o *Modicon M262* da Schneider é considerado como um PLC IoT, por estar preparado para ser integrado na plataforma *cloud* através de OPC UA.

No entanto, esta revolução tem mais implicações que apenas no meio industrial. O aumento e otimização da produção direcionada ao cliente têm implicações sociais e económicas no sentido em que muitos postos de trabalho irão desaparecer, dando lugar a novos. Os trabalhos rotineiros e de monitorização serão substituídos por máquinas, pelo que serão mais valorizadas num trabalhador as competências de criatividade, proatividade e comunicação. Consequentemente, a educação terá de sofrer alterações nos modelos de ensino para acompanhar a realidade da indústria 4.0, especialmente no ensino de engenharia. Deverá evoluir, tirando partido de diversas ferramentas e recursos como as *learning factories*. Estas aproximam o ensino teórico e prático do ambiente industrial real, permitindo a consolidação de diversos conceitos da indústria 4.0 como CPS, robótica e IoT [39]. Dois exemplos de *learning factories* são a *learning factory* na TU Wien na Áustria e a *learning factory* na unidade W. Booth School of Engineering Practice and Technology da Universidade McMaster no Canadá. A *learning factory* na TU Wien é a primeira deste tipo na Áustria, servindo como plataforma de investigação e de ensino, permitindo aos alunos uma aprendizagem prática das linhas de produção, desde o pedido ao produto final [33]. A *learning factory* na unidade W. Booth School of Engineering Practice and Technology (figura 2.21) foca-se nas três componentes de ensino: teoria, prática e investigação. É pretendido que esta ferramenta sirva de complemento ao ensino teórico e prático, disponibilizando diversas estações especializadas com foco na indústria 4.0, Iot e IIoT [40].

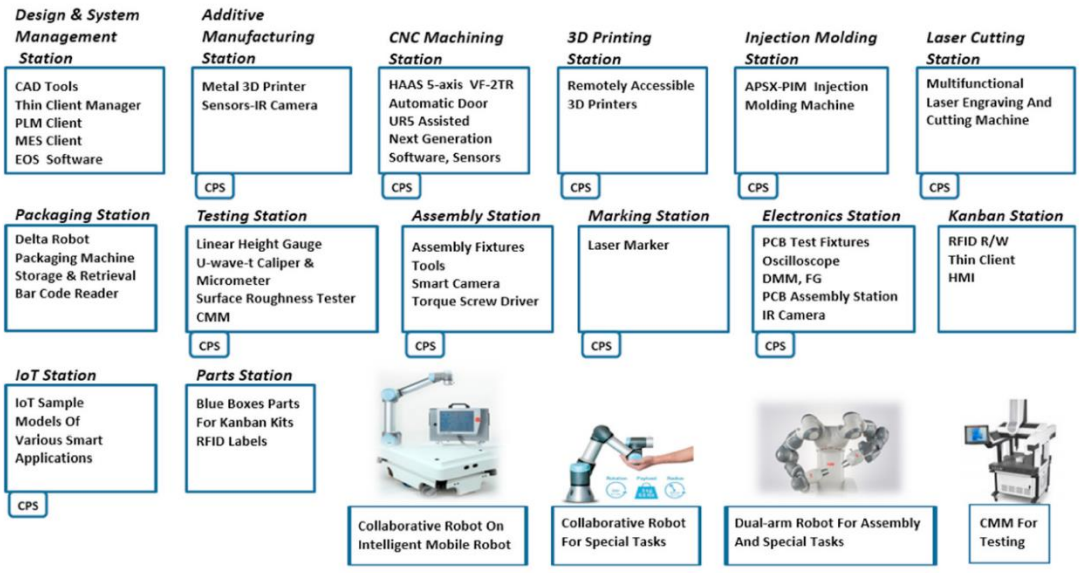


Figura 2.21 – Elementos principais da learning factory W. Booth [40]

2.6 Indústria 5.0

Com a crise causada pela doença de COVID-19 foram expostas diversas vulnerabilidades da indústria europeia, forçando a repensar os métodos de trabalho e abordagens atuais. Por isso, em julho de 2020 foi discutido em dois workshops virtuais organizados pela Direção-Geral de Pesquisa e Inovação da Comissão Europeia, o conceito de indústria 5.0 [41].

O conceito de indústria 5.0 não é um seguimento, continuação ou alternativa à indústria 4.0, mas sim um complemento da mesma. Esta enfatiza aspetos importantes do papel da indústria no futuro da sociedade europeia, não se focando apenas em fatores económicos ou tecnológicos, mas também em fatores ambientais e sociais. O termo é bastante recente, encontrando-se definido no documento "*Towards a sustainable, human-centric and resilient European industry"*, pela Direção-Geral de Pesquisa e Inovação: *"Industry 5.0 recognises the power of industry to achieve societal goals beyond jobs and growth to become a resilient provider of prosperity, by making production respect the boundaries of our planet and placing the wellbeing of the industry worker at the centre of the production process."* [41].

Nos workshops de julho de 2020 foram identificadas seis categorias com capacidade de explorar o potencial do conceito, trabalhando em conjunto: interação humano – máquina individualizada; biotecnologia e materiais inteligentes; simulação virtual e *digital twins*; tecnologia para análise, transmissão e armazenamento de dados; inteligência artificial; tecnologias para autonomia, armazenamento e eficiência energética e energias renováveis [42].

2.7 Ensino na Engenharia

A educação é dos pilares mais importantes da sociedade, devendo ser tratada como tal. Com o ensino em engenharia pretende-se dotar os alunos com uma combinação de competências técnicas e práticas, sensibilidade social e *soft skills* para que se formem bons profissionais de engenharia. Um engenheiro deve ser capaz de usar o seu conhecimento técnico e social para desenvolver e implementar sistemas que, de algum modo, são benéficos para a comunidade [43]. Citando Theodore Von Kármán, matemático, engenheiro aeroespacial e físico húngaro: *"Scientists discover the world that exists; engineers create the world that never was*." [44].

Os avanços tecnológicos das últimas décadas, especialmente os introduzidos pela indústria 4.0, permitiram uma série de oportunidades para o papel do engenheiro na sociedade. No entanto, para que seja possível manter atualizadas as competências técnicas dos futuros engenheiros, mantendo a formação de bons profissionais, é necessária uma sucessiva remodelação no ensino de engenharia [45]. Em 2002, na revista *Issues in Science and Technology*, WM. A. WULF e GEORGE M. C. FISHER escreveram que "*[…] students are being prepared to practice engineering for their parents' era, not for*

the 21st century." [46] Algum trabalho tem vindo a ser feito desde a data dessa publicação, no entanto continua a existir uma discrepância do que é procurado pelas empresas e promovido no ensino. Tal discrepância é apontada em [47], onde escrevem que as empresas que contratam os estudantes saídos da faculdade continuam a referir que estes não estão prontos para trabalhar, por falta de competências técnicas.

A geração atual e seguinte de alunos de engenharia tem acesso a uma enorme quantidade de informação. Qualquer resposta está a um clique de distância em qualquer motor de busca ou em plataformas de cursos gratuitos como o *Khan Academy*. Assim, o papel do professor tem também de acompanhar a mudança. Têm de ajudar a processar a informação relevante para desenvolver as competências necessárias [48]. Para colmatar a discrepância do ensino para o "mundo real", parece ser consensual na literatura, a importância dos laboratórios. Diversos artigos (ex. [49], [50], [51], [52], [53]) mencionam que os laboratórios, pela sua abordagem prática dos conteúdos teóricos lecionados, permitem uma melhor consolidação dos mesmos, mantêm a motivação dos alunos e preparam para o que é procurado pelas empresas num engenheiro. No entanto o seu planeamento e execução não são triviais, havendo limitações de recursos. Os materiais são caros e por vezes pouco versáteis tornando-os rapidamente obsoletos, existe escassez de recursos humanos qualificados e de espaços adequados.

2.7.1 Laboratórios em Automação

Grande parte do material didático de laboratório disponível é caro na sua aquisição e manutenção e pouco versátil. Existe escassez de recursos humanos qualificados e de tempo para a planificação dos materiais de forma a serem um bom complemento das competências teóricas lecionadas e, por vezes, falta de um espaço adequado para a prática dos mesmos.

Para contornar os problemas descritos, tirando partido da evolução da tecnologia, começaram a ser desenvolvidos diversos materiais para o ensino prático. Este desenvolvimento deu lugar a novos laboratórios que podem ser classificados pelo tipo de equipamento usado e pelo acesso ao mesmo. Dividem-se entre equipamentos físicos e/ou simulados, podendo ser de acesso local, remoto ou ambos [53], como é possível observar na tabela 2.8.

Tabela 2.8 – Classificação dos laboratórios [50]

Acesso	**Equipamento**	**Tipo de laboratório**
Local	Físico	*Hands-on*
	Simulado	Simulado
Remoto	Físico	Remoto
	Simulado	Virtual
Local e/ou remoto	Físico/simulado	*Interchangeable components*

Os laboratórios *hands-on* baseiam-se em processos de investigação físicos e reais, onde tanto os alunos como o material estão fisicamente presentes no laboratório [53]. Este tipo de laboratório tem vantagens pela proximidade das relações professor – aluno e aluno – aluno que permitem um melhor acompanhamento da evolução do aluno e o desenvolvimento de competências de trabalho de equipa. Por outro lado, como já foi mencionado, os materiais são caros e a limitação dos mesmos e do espaço obrigam, por vezes, à restrição no número de alunos que os usam.

Nos laboratórios simulados os alunos estão presencialmente no laboratório, no entanto a experiência é simulada por computadores. *Softwares* como o MatLab ou LabView têm a capacidade de simular diversos cenários reais. Neste caso é possível manter a proximidade das relações como nos laboratórios *hands-on* e diminuir o preço dos materiais. Os laboratórios simulados têm também a vantagem de acelerar processos que são demorados e de permitir ao aluno parar o processo para uma melhor análise e diagnóstico mas os dados experimentais são simulados e não reais [50].

Os laboratórios remotos e virtuais permitem uma descentralização do espaço físico no sentido em que os alunos não estão presentes fisicamente, podendo aceder aos mesmos de qualquer lugar. Perdem a relação de proximidade garantida pelos laboratórios anteriores, mas no caso dos laboratórios remotos, onde o processo é real (laboratório *hands-on* com acesso remoto), é potenciada a partilha de dados experimentais reais entre diversas universidades e indústria. Nos laboratórios virtuais o processo é simulado. Este tipo de laboratórios possibilita abranger um maior número de alunos de diversas nacionalidades, servindo de apoio às plataformas de *e-learning*. Como exemplos de laboratórios remotos existe o projeto VISIR (*Virtual Instrument System in Reality*) do Blekinge Institute of Technology, na Suécia, o *Weblabs* da University of Cambridge e a *Remote Farm* da TU Berlin.

O conceito de laboratórios *interchangeable components* é apresentado em [50] como a combinação dos equipamentos reais e virtuais (figura 2.22). Neste, o autor escreve que a linha que divide os quatro tipos de laboratório apresentados é ténue no sentido em que, por norma, os alunos usam HMI para aceder tanto aos equipamentos físicos como aos simulados. Como tal, surge a necessidade de definir um novo tipo de laboratórios, uma mistura entre realidade e simulação, como "*a live, direct or indirect, view of a handson lab whose physical devices are augmented by computergenerated sensory input such as sound, video, graphics or other information."* [50].

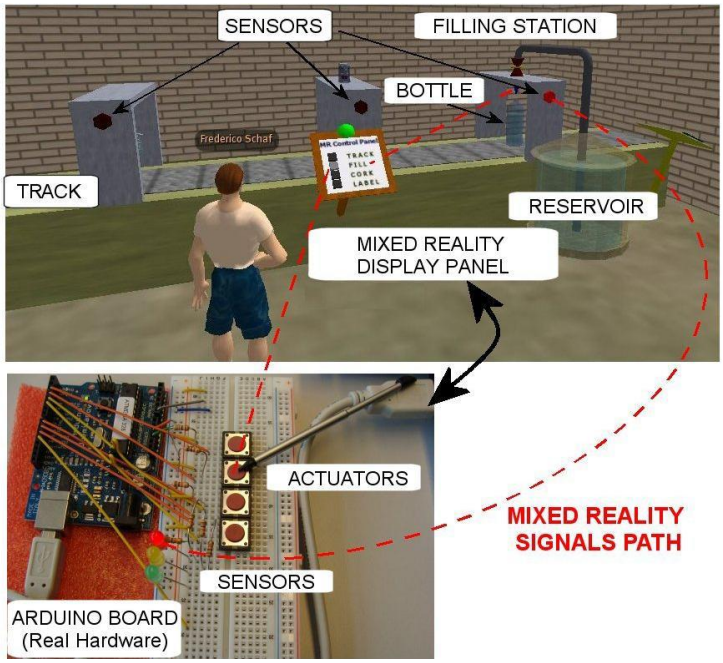


Figura 2.22 – Exemplo de um laboratório interchangeable components [50]

Cada laboratório tem a sua aplicação, devendo ser planeado e implementado com objetivos concretos, de forma a oferecer ao aluno ferramentas para a aquisição e consolidação de competências técnicas e teóricas. Com o trabalho prático realizado nesta dissertação foi desenvolvida uma solução didática *hands-on*, permitindo aos alunos contactar com diversos temas da automação industrial aplicado no contexto das estações de tratamento de água e águas residuais.

2.8 Estações de Tratamento de Água e Águas Residuais

A água é um elemento essencial à vida. Sem ela nenhum ser vivo poderia sobreviver. A sua qualidade é um fator extremamente importante, devendo apresentar certos parâmetros, seja para consumo ou higiene. Sem o devido tratamento, a propagação de doenças é inevitável, tornando-se num problema de saúde pública. Para além das consequências causadas no ser humano, se não existir um tratamento próprio da água bruta, também o ambiente sai prejudicado. Para que tal seja evitado, existem dois tipos de instalações: as Estações de Tratamento de Água (ETA) e as Estações de Tratamento de Águas Residuais (ETAR).

Enquanto as ETA tratam a água bruta, proveniente do ambiente, de maneira que esta esteja em condições para uso humano – seja doméstico, comercial ou industrial – as ETAR tratam a água residual para que esta esteja em condições de ser devolvida ao ambiente, sem o poluir. Ambas começam com a captação da água bruta, adequando depois as etapas e processos de tratamento às suas necessidades. No entanto, de uma forma geral, qualquer ETA é composta por gradagem, sedimentação, filtração/filtragem, desinfeção e estabilização. No caso das ETAR estão divididas em tratamento preliminar, primário,

secundário e terciário, tendo como base de tratamento parte dos processos existentes nas ETA. Ao longo destas etapas é feita a remoção de substâncias sólidas suspensas e de substâncias dissolvidas indesejáveis, a estabilização da água e a sua desinfeção [54].

Em ambas, a automação está presente sendo extremamente importante no controlo e otimização de todo o processo. Esta "reside nos automatismos de cada um dos diferentes tipos de estações, nas infraestruturas de controlo integrado de todos esses subsistemas, na interface com os operadores humanos e nos sistemas de registo dos dados de exploração." [54].

2.8.1 Funcionamento geral das ETA

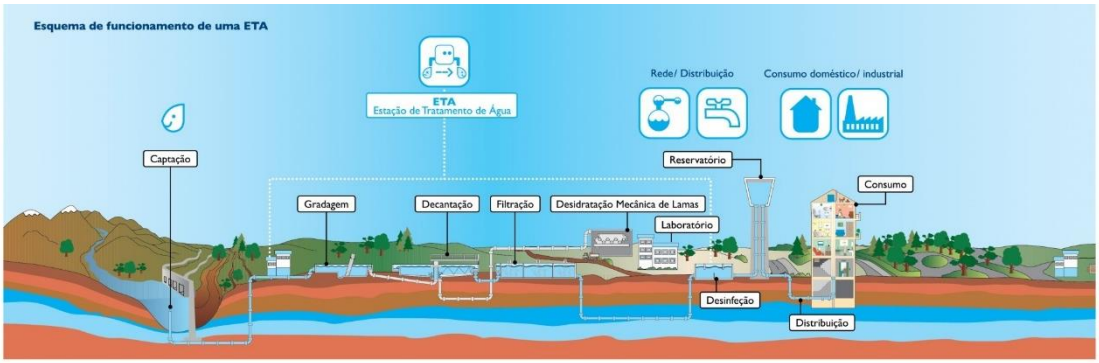


Figura 2.23 – Esquema de funcionamento de uma ETA [Retirado do site da Águas do Norte, S.A]

Nas ETA, depois de captada a água, é realizada a regulação do caudal da mesma antes de seguir para a etapa de gradagem, onde serão retirados da água os lixos de maior dimensão como folhas, trapos, plásticos, etc., ficando retidos em grades. Durante a regulação do caudal, é ainda adicionado coagulante iniciando o processo de coagulação, denominado de mistura rápida. Esta mistura, como o nome indica é rápida, servindo para desagregar as partículas sólidas que se encontram suspensas e criar pequenos flocos. As substâncias mais usadas para o efeito, como coagulantes, são o sulfato de alumínio, o cloreto de ferro e o sulfato de ferro, no entanto a sua escolha depende das características da água bruta, como por exemplo o pH [55]. Para esta escolha, por norma, usa-se um teste denominado de *Jar Test*, onde se ensaia em pequena escala o processo de coagulação.

Segue-se uma mistura lenta, denominada de floculação, para que os flocos não sejam destruídos, mas sim aglomerados, aumentando em tamanho para que sejam de rápida sedimentação. Para que os flocos tenham o tamanho ideal, é considerado o gradiente de velocidade G, um fator importantíssimo que determina a probabilidade da formação dos mesmos. Tanto a coagulação como a floculação têm como objetivo a remoção de microrganismos presentes à superfície, da cor e turvação da água, através das diferentes velocidades de mistura – rápida e lenta. As misturas podem ser conseguidas recorrendo a energia hidráulica, mecânica ou pneumática.

Após a floculação, segue-se a etapa de decantação onde ocorre um processo de sedimentação ou flotação dos coágulos formados na etapa anterior. Este processo é lento para evitar a perturbação da água, permitindo a sedimentação dos coágulos no fundo da câmara formando uma camada de lodo denominada de lamas ou à superfície (flotação) formando uma espuma. A eficácia da sedimentação e remoção das partículas está inerente à eficácia do processo coagulação-floculação. Caso contrário a agregação das partículas em flocos de dimensões maiores poderá não acontecer, não permitindo a sua sedimentação no fundo do tanque. Na figura 2.24 é possível encontrar a representação de um tanque de sedimentação.

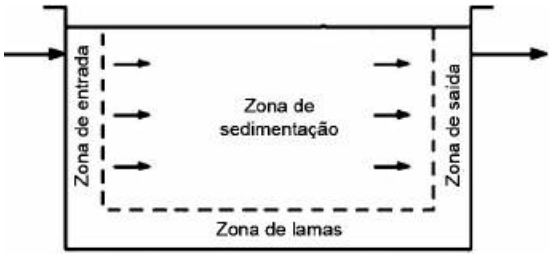


Figura 2.24 – Esquema de um tanque de sedimentação [56]

As lamas removidas dos tanques de sedimentação são o resíduo principal produzido nas ETA. Estas são compostas pelos sedimentos formados pelo coagulante e pelo floculante, tendo cheiro e um grande volume de água. Como tal, a sua composição está diretamente relacionada com o coagulante usado. Após a sua sedimentação, as lamas devem ser espessadas para reduzir o seu volume, estabilizadas e desidratadas, antes de serem descartadas e armazenadas. As lamas tratadas, dependendo da sua composição, são classificadas de diferentes formas, segundo o Decreto-Lei n.º73/2011 de 17 de junho: como um resíduo industrial banal, resíduo não perigoso ou resíduo industrial perigoso. Se as lamas satisfizerem determinados requisitos, a sua utilização mais usual é como fertilizante agrícola [56].

Paralelamente ao tratamento das lamas, também a água e odores resultantes têm o seu circuito próprio na ETA, para que possam ser tratados. Tanto na flotação como na sedimentação são removidos a maioria dos coágulos. A pequena quantidade de coágulos que não são removidos e permanecem na água a tratar, serão removidos por meio da etapa de filtragem que é a seguinte na cadeia de tratamento das ETA. A filtragem é a última etapa de remoção de partículas sólidas da cadeia de tratamento das ETA. Existem diversos tipos de tecnologia de filtragem, apresentados na figura 2.25: em leito granular, superficial e em membrana (microfiltração, ultrafiltração e nano filtração).

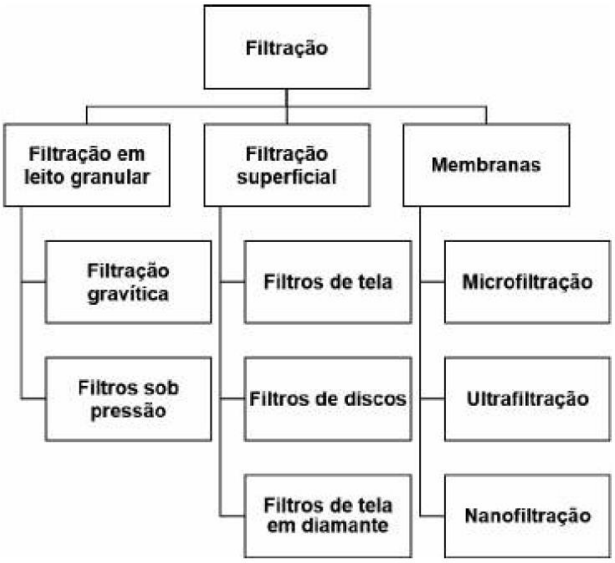


Figura 2.25 – Enumeração dos diversos tipos de filtragem [56]

A mais comum é a filtragem em meio granular. Nesta, a água atravessa o meio filtrante – que pode ser constituído por variados materiais – por ação da gravidade ou por pressurização, para remoção das partículas presentes. À medida que a água atravessa o meio filtrante, as diversas partículas presentes nela vão preenchendo os espaços vazios deste, colmatando-o. Com o filtro colmatado, a perda de carga é maior e a sua eficácia de filtragem menor, sendo necessária a sua lavagem. A lavagem de um filtro (figura 2.26) é realizada com recurso a injeção de água e de ar comprimido, cumprindo a seguinte sequência: esvaziamento do filtro até perto de 10cm de altura do material filtrante; injeção de ar comprimido para uma repartição de ar homogénea no filtro; realização de purga de ar para impedir que fique acumulado no leito filtrante; circulação de água no sentido inverso para lavagem do leito filtrante; enchimento do filtro purgando as primeiras águas para garantir a sua limpeza antes de avançar. A eficácia da filtragem depende essencialmente do caudal de água, das partículas a filtrar e das características do filtro.

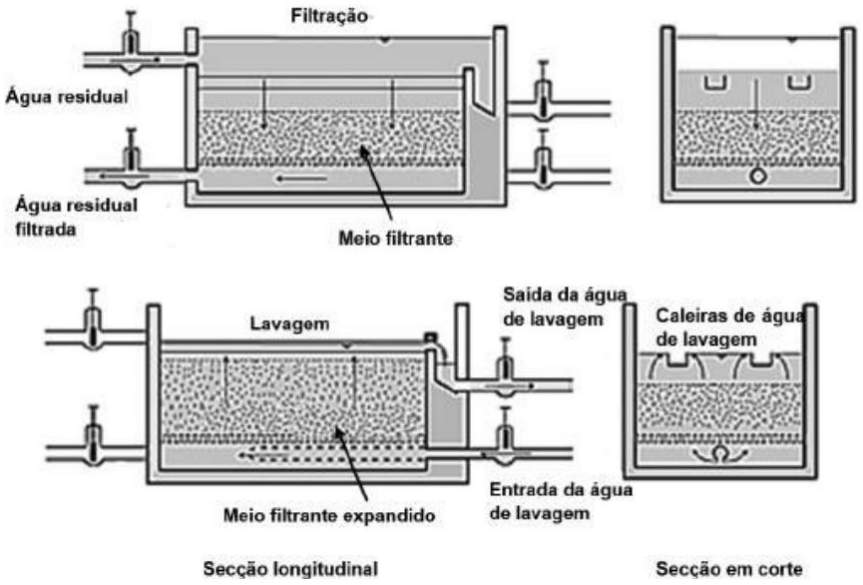


Figura 2.26 – Representação geral de um filtro em filtragem e em lavagem [56]

O meio filtrante pode ser constituído por uma ou várias camadas, normalmente de areia e/ou antracite. A velocidade de filtragem ou carga hidráulica é o quociente entre o caudal a tratar e a área de superfície do filtro, dividindo os filtros em leito granular por filtros rápidos e lentos [56].

Depois da filtragem segue-se a desinfeção, que é a etapa mais importante do tratamento da água. O seu objetivo visa eliminar ou reduzir até um nível significativo (compatível com valores que mantenham a proteção da saúde pública) as três categorias de organismos presentes na água e preocupantes para o ser humano: bactérias, vírus e parasitas. Os mecanismos para a desinfeção das águas são normalmente por agentes químicos, como a aplicação de cloro simples. Podem também ser realizados através de agentes físicos como a aplicação de temperatura, de raios ultravioletas e gama.

Por fim chega a etapa de estabilização. Esta é a última etapa antes da distribuição, sendo imprescindível na qualidade da água tratada. Saindo da desinfeção, são realizadas diferentes medições de controlo, de forma a perceber a sua composição. Consoante os resultados das medições, é realizada uma correção de pH através da adição de substâncias corretivas como carbonato de sódio e o dióxido de carbono.

Para além do tratamento líquido há ainda tratamento das lamas resultantes da sedimentação e uma reutilização das águas de lavagem. A água segue depois para as estações elevatórias, constituídas por eletrobombas, que irão bombeá-la até ao seu destino final para consumo, sempre que não é possível realizar um escoamento gravítico.

2.8.2 Etapas e Processos de uma ETAR

Nas ETAR, depois de captada a água, esta segue por diferentes tipos de tratamento (preliminar, primário, secundário e terciário) sendo exposta a diferentes processos em cada, podendo repeti-los com um grau de intensidade maior, até estar em condições para ser devolvida ao meio recetor (figura 2.27), repetindo o esquema de processamento de uma ETA. Para além da água residual são ainda tratadas as lamas e o ar.

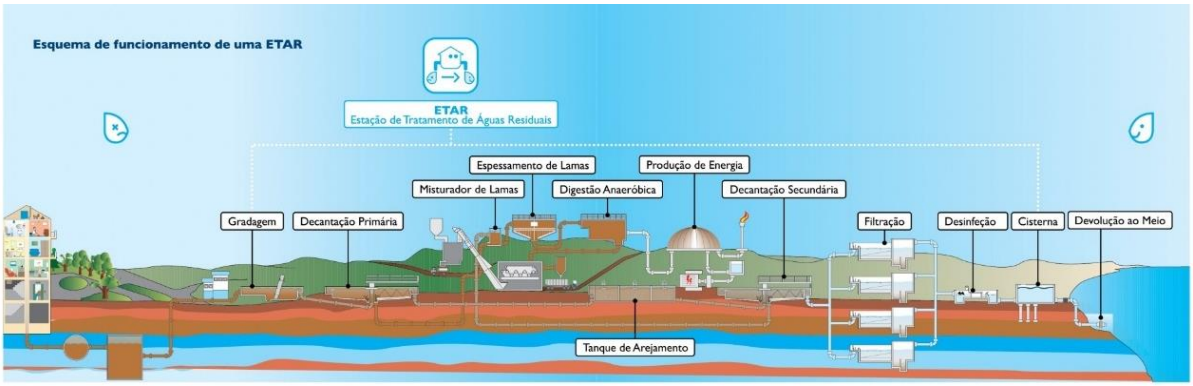


Figura 2.27 – Esquema de funcionamento de uma ETAR [Retirado do site da Águas do Norte, S.A]

No tratamento preliminar são removidas as partículas sólidas, areias e gorduras e, por vezes, é realizada uma equalização dos caudais e da carga poluente com o objetivo de proteger as etapas seguintes. Os processos usados são a gradagem, a equalização, a desarenação e a remoção de óleos e gorduras.

O tratamento primário, de acordo com o Decreto-Lei nº152/97, é definido como "o conjunto de operações e processos de tratamento que asseguram uma redução mínima de 20% da CBO_5 e de 50% das partículas sólidas em suspensão (SST) nas AR." [56]. Para tal, são por norma usados processos de sedimentação por decantação, sendo raro o recurso à flotação.

O tratamento secundário é realizado através de processos como a decantação e floculação, com o objetivo de reduzir a "matéria orgânica biodegradável dissolvida ou em suspensão coloidal que não foi removida até este nível de tratamento" [56].

O tratamento terciário é opcional e serve como complemento às etapas anteriores. O seu objetivo é a remoção de nutrientes e/ou microrganismos patogénicos para evitar o crescimento excessivo de plantas aquáticas (eutrofização), por motivo da proteção da saúde pública. Para além deste tratamento pode ainda ser usado o tratamento avançado na reutilização de águas residuais tratadas. Este foca-se na "remoção de poluentes dissolvidos presentes em concentrações residuais" [56].

2.8.3 Automação e Supervisão nas ETA e ETAR

Tal como acontece em qualquer contexto industrial, também as ETA e ETAR têm sistemas de automação e supervisão. Estes sistemas juntam áreas como instrumentação e controlo de potência para monitorizar e controlar os processos existentes para tratamento da água, seja bruta, seja residual. Juntam vários controladores, sensores e atuadores tendo como principais tarefas efetuar movimentação e agitação de água, manobra de válvulas e comportas, operação e limpeza de filtros, movimentação e compactação de lamas; existem também pontes e tapetes rolantes, ventilação, produção de ar comprimido, doseamento e aplicação de reagentes, etc. A nível de instrumentação, a pressão, o caudal, o nível e a qualidade da água são as grandezas fundamentais a medir por processo automático [54].

Existem ainda diversos equipamentos de potência, usados para diversos tipos de funções tais como: bombagem, transporte de sólidos, ventilação, fontes de alimentação (para órgãos de controlo, medição, comunicação, processamento), conversores estáticos, transformadores, unidades de alimentação de recurso, etc.

Para conectar todos os equipamentos, as estações assentam sobre uma rede LAN com computadores num centro de comando e controlo. Dependendo da complexidade da estação, poderão ser utilizadas redes de campo e *sensorbus*. A rede de campo faz a comunicação entre autómatos, enquanto as redes *sensorbus* realizam a comunicação dos sensores e os autómatos correspondentes. Na figura 2.28 encontra-se representado um exemplo de arquitetura de automação de uma ETA complexa [54].

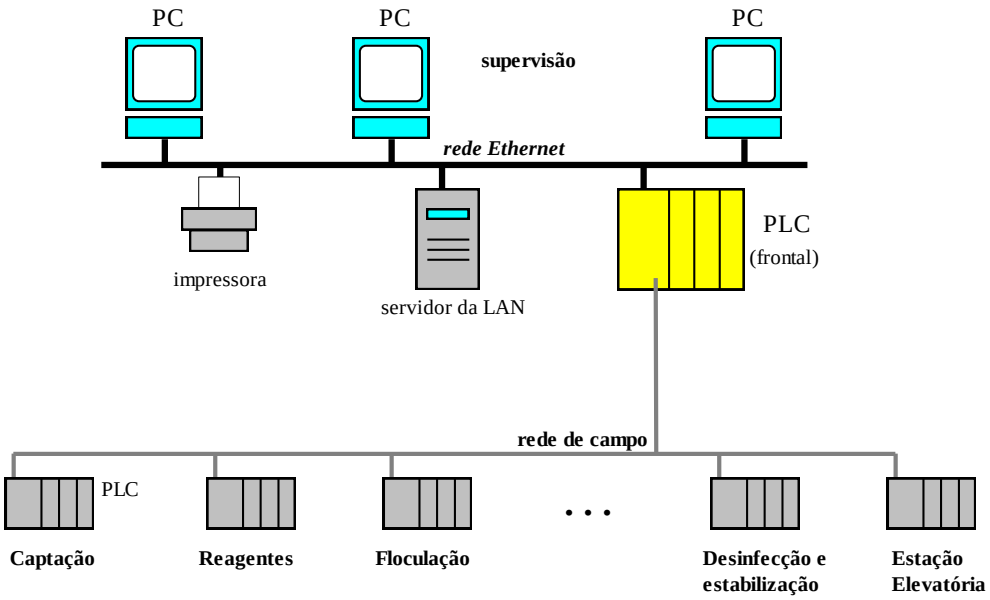


Figura 2.28 – Exemplo de arquitetura de automação de uma ETA complexa [54]

A arquitetura de automação existente consiste normalmente em diversos subsistemas de controlo e comando local de cada processo, todos interligados e funcionalmente integrados num centro

de comando e controlo. Cada subsistema é assegurado por controladores – por norma PLC's – que enviarão em tempo real os seus dados para o sistema de SCADA, presente no centro de comando e controlo. A figura 2.29 ilustra um exemplo de alguns ecrãs sinópticos de um sistema SCADA usado nas ETA e ETAR.

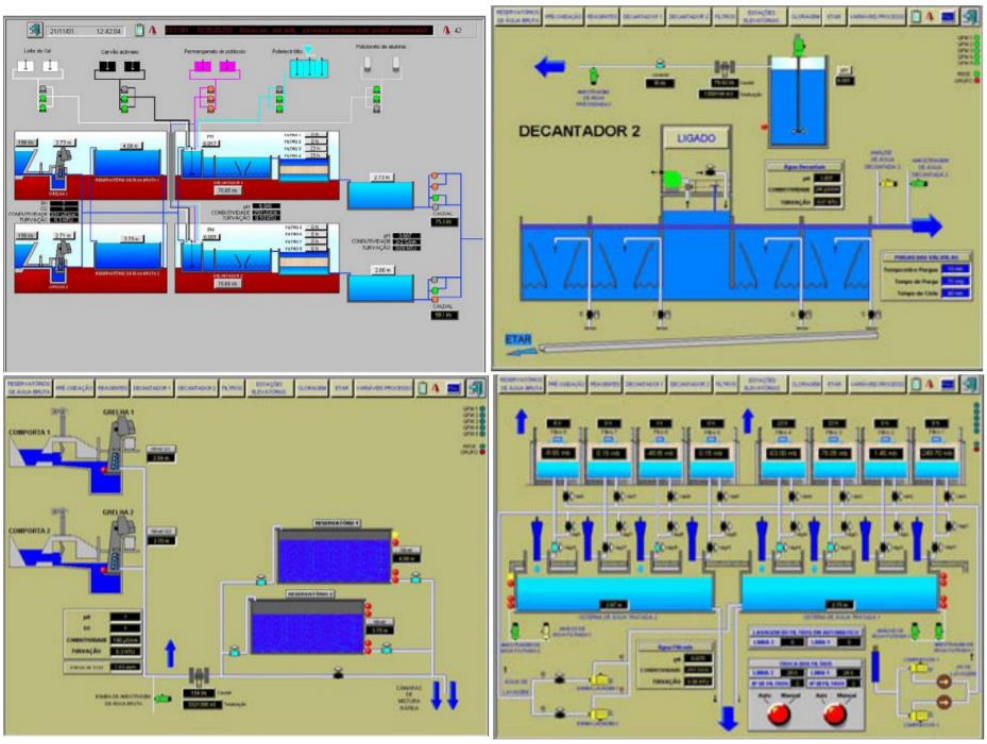


Figura 2.29 - Exemplos dos sinópticos do sistema de supervisão da ETA das Fontainhas [57]

As funções do sistema de SCADA passam por recolher e exibir o estado dos elementos da estação, emitir alarmes, recolher comandos dos operadores e enviá-los para os subsistemas, planeamento e otimização da estação, entre outros.

Capítulo 3

Simulador Didático

Resumo: No presente capítulo é apresentada a solução desenvolvida com a função de simulador didático para sistemas de tratamento de água. São apresentados todos os materiais utilizados (*hardware* e *software*), a sua estrutura e as fases de desenvolvimento. São ainda apresentados alguns resultados experimentais.

3.1 Descrição do Simulador

Pretendeu-se com o desenvolvimento da presente dissertação, a criação de uma solução de simulador que dê a oportunidade aos alunos de licenciatura e mestrado do curso de engenharia eletrotécnica, do Instituto Superior de Engenharia de Lisboa, de interagir e testar diversos processos automatizados com aspeto e semelhanças face a aplicações reais. Desenvolveu-se também este simulador com o objetivo de ser usado em feiras, exposições e dias abertos a alunos internos e externos ao ISEL, mostrando algum trabalho que tem vindo a ser desenvolvido no mestrado em engenharia eletrotécnica, especialmente no ramo de automação industrial. Deste modo, foi desenvolvido um simulador didático com o intuito de simular e controlar de forma simples alguns sistemas automatizados relacionados com estações de tratamento de água. A solução é composta por seis recipientes de três tamanhos diferentes para armazenamento de água interligados entre si pelas respetivas tubagens. O maior recipiente serve a função de reservatório e nos restantes serão simuladas três etapas diferentes: aquecimento, mistura e filtragem. A figura 3.1 ilustra o aspeto geral do simulador desenvolvido. A solução foi concebida com elevada portabilidade de modo a ser movida para diversos locais.

Figura 3.1 – Representação da solução desenvolvida

Com a solução desenvolvida foram aprofundados diversos temas relacionados com a automação industrial, nomeadamente a instrumentação, o controlo, a informática e as redes de comunicação e os seus respetivos protocolos.

3.2 Estrutura do Simulador

O simulador desenvolvido, tal como foi mencionado, é composto por seis recipientes para armazenamento de água. No seu desenvolvimento foram utilizados diversos sensores e atuadores e algum equipamento de telecomunicações e elétrico como cablagem e equipamentos de proteção. A figura 3.2 ilustra a estrutura do simulador, apresentando a disposição dos sensores, atuadores e dos 6 recipientes (reservatório, tanque 1, tanque 2, tanque 3, tanque 4 e tanque 5).

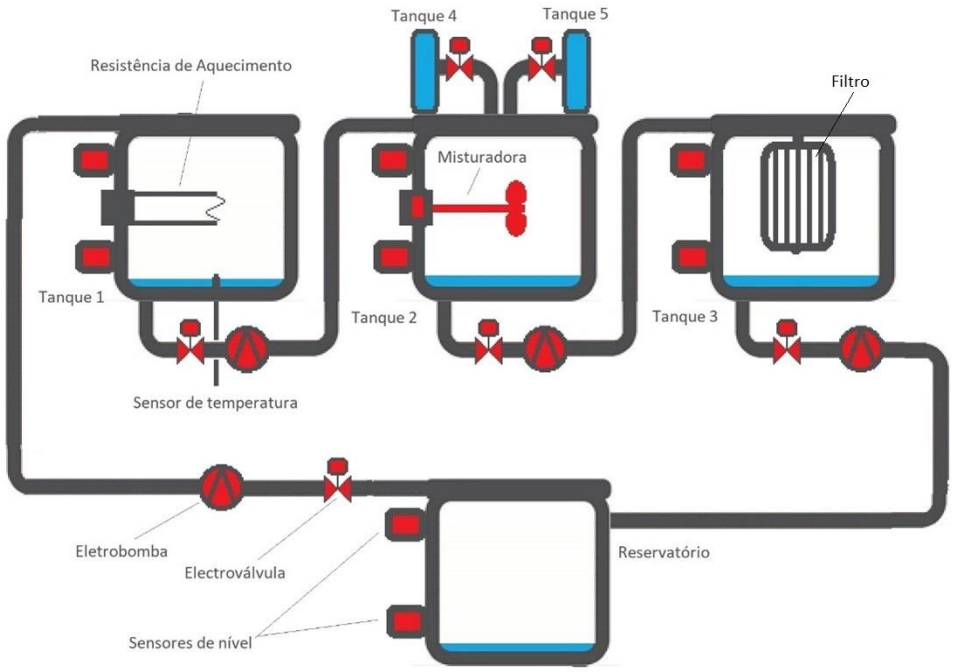


Figura 3.2 – Estrutura da solução desenvolvida

Para o controlo do simulador foi usado um controlador industrial (PLC) da Phoenix Contact, conectado a um *Raspberry Pi* 3 com *display touch panel* que proporcionam a supervisão e a respetiva interface Humano-Máquina (HMI). Foi ainda desenvolvida uma aplicação *Android* para adicionar alguma possibilidade de configuração e acompanhamento remoto do estado do simulador. A programação do controlador foi implementada usando diversas linguagens da norma IEC61131-3, nomeadamente a linguagem textual estruturada (ST), os blocos lógicos (FBD), os diagramas de contactos (LD) e o *grafcet* (SFC) com recurso ao software PCWORX da Phoenix Contact. A programação da aplicação a ser executada no *Raspberry Pi* foi desenvolvida em linguagem *python* com

recurso ao software ERIC e a aplicação *Android* foi criada através da plataforma online *MIT App Inventor* desenvolvida pelo *Massachusetts Institute of Technology* (MIT).

Para facilitar a sua explicação, o simulador pode ser dividido em três subgrupos: de instrumentação, de controlo e potência e de comunicação de acordo com o diagrama representado na figura 3.3.

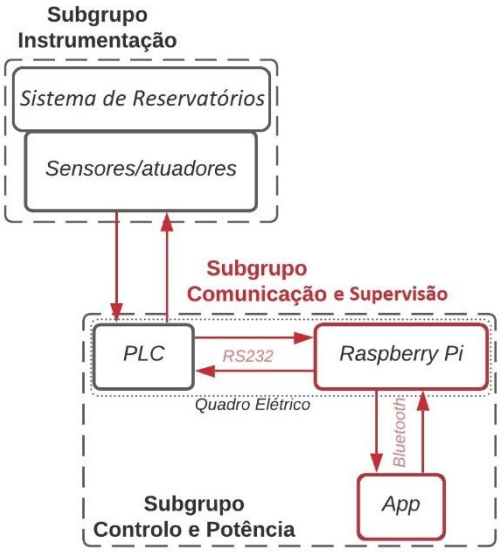


Figura 3.3 – Esquema da solução desenvolvida

3.2.1 Subgrupo de Instrumentação

O subgrupo de instrumentação é responsável pela aquisição dos dados a enviar para o subgrupo de controlo e potência. Este inclui todo o material que compõe a estrutura física do simulador e ainda os sensores e atuadores.

O simulador é composto por seis recipientes de três tamanhos diferentes para armazenamento de água, apoiados numa estrutura metálica móvel de dois andares. No primeiro andar encontra-se o maior recipiente, tendo a função de reservatório de armazenamento. Toda a água que irá circular pelo simulador encontra-se nele armazenada. No segundo andar estão os três recipientes de tamanho médio, com a função de tanque, onde serão simuladas as etapas de aquecimento, mistura e filtragem. No topo do tanque da etapa de mistura estão colocados os restantes recipientes, que consistem em dois recipientes pequenos, onde serão colocados reagentes diferentes para serem adicionados durante a simulação da etapa de mistura. Existe ainda, no segundo andar, um suporte para colocar o quadro elétrico do subgrupo de controlo e potência.

Para aproximar o simulador didático desenvolvido do funcionamento básico de uma ETA, foram utilizados diversos sensores e atuadores com atuação real: quatro eletrobombas centrífugas, seis eletroválvulas, oito sensores de nível, dois sensores de temperatura, uma resistência de aquecimento, uma bomba misturadora e duas válvulas manuais para purga do sistema. O reservatório e cada tanque

têm associados uma eletrobomba centrífuga, uma eletroválvula e dois sensores de nível (com indicação de máximo e de mínimo). Os tanques dos reagentes (tanques 4 e 5) são uma exceção, tendo apenas uma eletroválvula onde os reagentes escorrem por efeito da gravidade. Para além dos sensores e atuadores mencionados, foram introduzidos outros elementos: o reservatório tem uma válvula manual; o tanque de aquecimento (tanque 1) tem uma resistência de aquecimento e um sensor de temperatura; o tanque de mistura (tanque 2) tem uma bomba misturadora; o tanque de filtragem (tanque 3) tem uma válvula manual e um sensor de temperatura apenas para leitura e monitorização do valor final da temperatura.

Os sensores de nível foram colocados a pares no reservatório e nos tanques das etapas para identificar a presença de nível máximo e mínimo de água. São sensores de nível do tipo boia com dois estados (fechado ou aberto), representando a presença de água do seguinte modo: quando não existe água o sensor é um interruptor aberto, no entanto, devido ao efeito de boia, quando existe água o sensor fecha o circuito, passando um sinal de 24VDC ao controlador. A figura 3.4 ilustra os sensores de nível utilizados.

Figura 3.4 – Sensor de nível usado no simulador

Os sensores de temperatura analógicos variam a sua resistência interna em função da variação do valor de temperatura da água apresentando-o na forma de variação de um sinal elétrico. Neste simulador existem dois sensores analógicos, um deles colocado no início no tanque de aquecimento e outro no fim do circuito de água do simulador, no tanque de filtragem, no entanto apenas o sensor inicial está ligado ao PLC. Este é na realidade um módulo alimentado a 24VDC que contém o sensor de temperatura, um *display* e um transmissor que realiza a aquisição e conversão do sinal elétrico em corrente, enviando-o posteriormente para o PLC. A gama de valores de temperatura medidos pelo sensor é entre 0ºC e 150ºC e a gama do valor em corrente é normalizada entre 4mA e 20mA. O sensor final trabalha também com um sinal de corrente não normalizado e possui um *display* próprio, onde apresenta a tensão de alimentação do sistema bem como o valor de temperatura no tanque de filtragem. A figura 3.5 ilustra uma imagem de cada um desses sensores de temperatura. A figura da esquerda ilustra o sensor de temperatura do tanque de aquecimento e a da direita do tanque de filtragem.

Figura 3.5 – Sensores de temperatura inicial (esquerda) e final (direita) usados no simulador

As eletroválvulas são válvulas solenoides em plástico, normalmente fechadas. As eletrobombas centrífugas possuem motores DC sem escovas com uma capacidade de fluxo de 800 L/h. Quer as eletroválvulas, quer as eletrobombas são alimentadas a 24VDC e apenas têm dois estados: ligado ou desligado. A figura 3.6 ilustra as eletroválvulas e eletrobombas utilizadas.

Figura 3.6 –Eletroválvulas (esquerda e centro) e eletrobomba centrífuga (direita) usadas no simulador

A resistência de aquecimento e a bomba misturadora são alimentadas a 230VAC e são atuadores reaproveitados de eletrodomésticos, cada um com a sua função específica de aquecer e misturar, respetivamente. A resistência de aquecimento é de 850W e a bomba misturadora é de 150W.

3.2.2 Subgrupo de Controlo e Potência

O subgrupo de controlo e potência consiste num quadro elétrico em acrílico com material de controlo e potência.

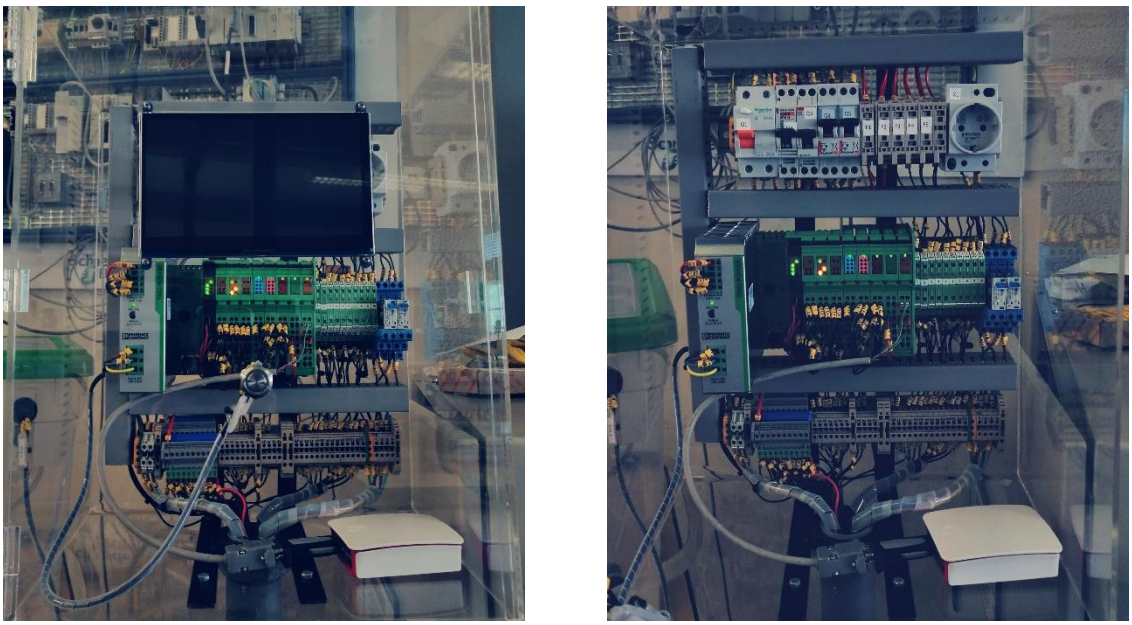


Figura 3.7 – Quadro elétrico fechado (fig. esquerda) e aberto (fig. direita)

O quadro tem a sua construção em acrílico para, de um modo didático, facilitar aos alunos e outros utilizadores a visualização do seu interior onde estão todas as ligações elétricas e respetivas proteções, um interruptor *on/off* com led, um *Raspberry Pi* 3 e o elemento central no controlo do simulador: o PLC. Será o PLC que realizará todas as operações de controlo do simulador como abertura e fecho de eletroválvulas, controlo das eletrobombas, controlo do aquecimento e mistura, leitura dos sinais dos sensores de níveis binários e do sensor de temperatura. Para tal, foi escolhido o PLC ILC 131 ETH da Phoenix Contact com 8 entradas e 4 saídas digitais, juntamente com 5 módulos de expansão: um de 8 entradas binárias, dois de saídas binárias (8+4), um de 2 entradas analógicas e ainda um para comunicação série em RS232-C. A figura 3.8 ilustra o aspeto do PLC ILC131 ETH com os módulos de expansão.

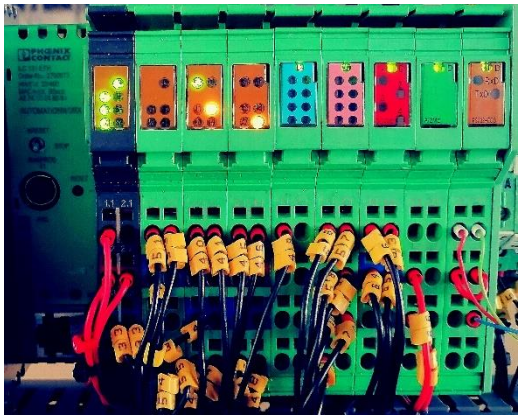


Figura 3.8 – PLC ILC 131 ETH e respetivos módulos

3.2.2.1 Módulos do PLC ILC 131 ETH

O PLC escolhido, tal como foi mencionado no ponto anterior, é constituído apenas por 8 entradas e 4 saídas binárias e é composto por cinco módulos de expansão:

- Um módulo IB IL 24 DI8 HD/PAC com 8 entradas binárias;
- Um módulo IB IL 24 DO8 HD/PAC com 8 saídas binárias;
- Um módulo IB IL 24 DO4/ME com 4 saídas binárias;
- Um módulo IB IL AI2 SF/ME com 2 entradas analógicas (selecionáveis em tensão ou em corrente);
- Um módulo IB IL RS232/ECO para comunicação série.

Os módulos de entradas e saídas binárias IB IL 24 DI8 HD/PAC, IB IL 24 DO8 HD/PAC e IB IL 24 DO4/ME trabalham a 24VDC, onde as saídas binárias são a transístores e o seu funcionamento já foi explicado anteriormente no ponto 2.2.1. O módulo IB IL AI2 SF/ME tem duas entradas, permitindo-lhe ler até dois sinais analógicos de tensão com gamas de 0V a 10V e -10V a +10V ou corrente com gamas de 0mA a 20mA, 4mA a 20mA e -20mA a +20mA. Cada entrada é composta por quatro terminais: tensão, corrente, *ground* e ligação de blindagens. A figura 3.9, ilustra os esquemas de ligações do módulo IB IL AI2 SF/ME.

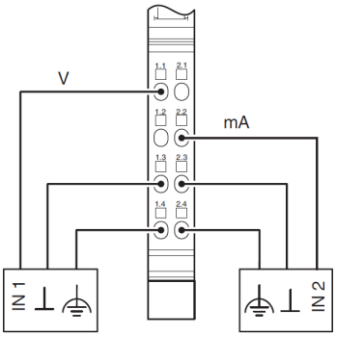
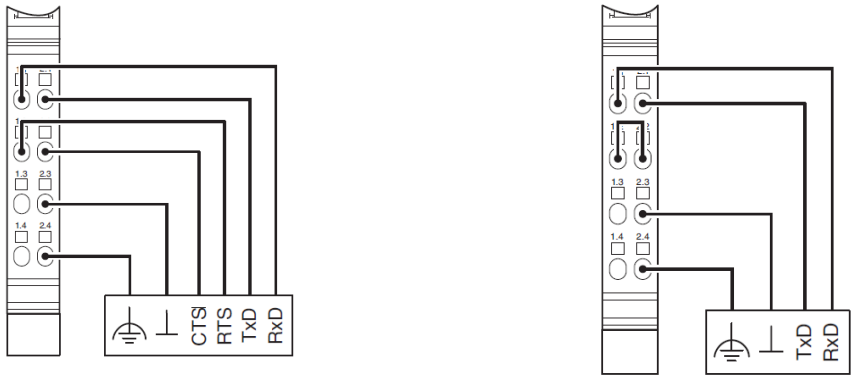


Figura 3.9 – Esquema de ligações do módulo IB IL AI2 SF/ME

O módulo IB IL RS232/ECO tem um canal de entrada e saída RS232, permitindo a transmissão série de dados. A velocidade de transmissão pode ir até 38400 bits/s e aceita os sinais de *handshake* (controlo de dados) RTS e CTS.



(a) (b)

Figura 3.10 – Esquema de ligações do módulo IB IL RS232/ECO com handshake (a) e sem handshake (b).

3.2.2.2 Quadro elétrico

O quadro elétrico foi dimensionado para alimentar todo o sistema, estando dividido em quatro circuitos com recurso a disjuntores magneto-térmicos:

1. Um barramento de 24VDC alimentado a partir de uma fonte de alimentação estabilizada de 120W e protegida com um disjuntor curva C de 2A;
2. A resistência de aquecimento do tanque de aquecimento com um disjuntor curva C de 10A;
3. A bomba misturadora do tanque de mistura com um disjuntor curva C de 10A;
4. Uma tomada monofásica para diversos fins incluído a alimentação do *Raspberry Pi* e respetivo *display*, com um disjuntor curva C de 16A.

A alimentação do quadro é realizada pela rede de baixa tensão, com proteção através de um interruptor diferencial de 25A. O barramento de 24VDC alimenta o PLC e respetivos módulos, os sensores e os atuadores (à exceção da resistência de aquecimento e bomba misturadora que são de 230VAC) com recurso a relés auxiliares. Foi usada uma fonte de alimentação Trio Power da Phoenix Contact para obter os 24VDC e seis seccionadores porta-fusíveis de 2A cada para realizar a proteção dos equipamentos a ele ligados. As ligações dos sensores/atuadores e PLC estão descritas da tabela 3.1 à tabela 3.4, no entanto os esquemas elétricos completos podem ser consultados nos anexos.

Tabela 3.1 – Ligações elétricas das entradas Binárias do PLC

Módulo	Terminal de ligação	Endereço	Função
PLC ILC 131 ETH	I1	1.1	Interruptor on/off
	I2	2.1	Sensor de nível mínimo do reservatório
	I3	1.4	Sensor de nível máximo do reservatório
	I4	2.4	Sensor de nível mínimo do tanque de aquecimento
	I5	3.1	Sensor de nível máximo do tanque de aquecimento
	I6	4.1	Sensor de nível mínimo do tanque de mistura
	I7	3.4	Sensor de nível máximo do tanque de mistura
	I8	4.4	Sensor de nível mínimo do tanque de filtragem
IB IL 24 DI8 HD/PAC	I9	1.1	Sensor de nível máximo do tanque de filtragem

Tabela 3.2 – Ligações elétricas das entradas analógicas do PLC

Módulo	Terminal de ligação	Endereço	Função
IB IL AI2 SF/ME	---	1.2	Sensor de temperatura
	---	1.3	Ligação em corrente

Tabela 3.3 – Ligações elétricas das saídas binárias do PLC

Módulo	Terminal de ligação	Endereço	Função
PLC ILC 131 ETH	Q1	1.1	Eletroválvula reservatório
	Q2	2.1	Eletrobomba reservatório
	Q3	1.4	Electroválvula tanque de aquecimento
	Q4	2.4	Eletrobomba tanque de aquecimento
IB IL 24 DO8 HD/PAC	Q5	1.1	Eletroválvula tanque de mistura
	Q6	2.1	Eletrobomba tanque de mistura
	Q7	1.2	Eletroválvula reagente tanque de mistura
	Q8	2.2	Eletroválvula reagente tanque de mistura
	Q9	1.3	Eletroválvula tanque de filtragem
	Q10	2.3	Eletrobomba tanque de filtragem
	Q11	1.4	Led interruptor on/off
IB IL 24 DO4/ME	---	1.1	Resistência de Aquecimento
	---	2.1	Bomba misturadora

Tabela 3.4 – Ligações elétricas do módulo de comunicação do PLC

Módulo	Terminal de ligação	Endereço	Função
IB IL RS232/ECO	---	1.1	RxD (Receção)
	---	2.1	TxD (Transmissão)
	---	1.3	Ground

3.2.3 Subgrupo de comunicação e supervisão

O subgrupo de comunicação e supervisão, tal como o nome indica, consiste nos diferentes meios de comunicação e supervisão usados no desenvolvimento do simulador. Com a função de supervisão, foi escolhido e colocado no quadro elétrico um *Raspberry Pi* 3 ligado a um *display* 1024x600 HD com funcionalidade *touch screen* e desenvolvida uma aplicação para *Android*. Com esta solução é possível visualizar e enviar comandos para o PLC.

Figura 3.11 – Raspberry Pi 3

Para que fosse possível cumprir os objetivos propostos, foi necessário idealizar uma rede de comunicação que interligasse o subgrupo de instrumentação, o PLC, o *Raspberry Pi* e o telemóvel com a aplicação *Android*.

A ligação dos sensores (binários ou analógicos) e atuadores está ligada diretamente (ou por intermédio de relés) com condutores às entradas físicas do PLC através dos terminais disponibilizados para esse efeito.

Dada a simplicidade da solução foi estabelecida uma rede de comunicação série de ligação ponto a ponto RS232-C com transmissão *half-duplex* para interligar o PLC e o *Raspberry Pi*. Foi também usada a comunicação sem fios *Bluetooth* para interligar o *Raspberry Pi* e um telemóvel com uma aplicação *android* desenvolvida especificamente para este simulador. Esta escolha foi tomada com base na arquitetura e objetivos pretendidos para o simulador. Por ser de acesso local, ponto a ponto, com curtas distâncias entre os equipamentos e por ser para demonstração existindo um baixo nível de risco relativamente à segurança, ambas as soluções para comunicação (tanto RS232-C, como *Bluetooth*) são fiáveis e de moderada dificuldade de implementação. A figura 3.12, ilustra de forma simplificada as ligações dos dispositivos.

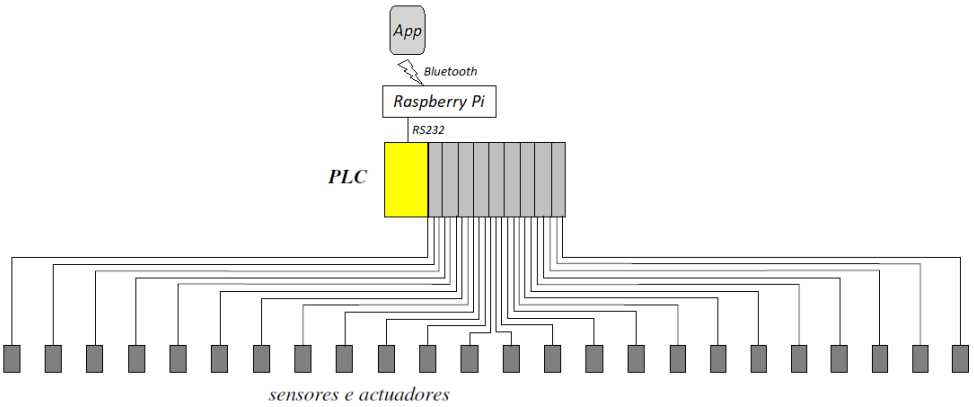


Figura 3.12 – Esquema das ligações dos sensores e atuadores ao PLC e a ligação do PLC aos restantes dispositivos de comunicação e supervisão.

3.2.3.1 Interface e comunicação série RS232-C

Um dos principais problemas para interligar o PLC e o *Raspberry Pi* através de uma interface RS232-C prende-se nas grandezas envolvidas na aquisição do sinal. O sinal RS232-C proveniente do PLC, tal como mencionado no ponto 2.3.4, varia o nível de tensão de +3V a +15V para o nível lógico "0" e -3V a -15V para o nível lógico "1", enquanto o sinal do *Raspberry Pi* é CMOS/TTL (*transistor-transistor logic*) com 0V para o nível lógico "0" e 3,3/5V para o nível lógico "1". Como tal, não é possível realizar uma ligação direta com risco de queimar o equipamento sendo necessário algum tipo de conversão de sinal. Foram testadas duas soluções: usando um conversor TTL ligado aos pinos UART do *Raspberry Pi* e usando um conversor série/USB ligado à porta USB do *Raspberry Pi*. Pela sua simplicidade e facilidade de utilização, foi implementado o conversor série/USB entre o PLC e o *Raspberry Pi*. Do lado do PLC foi usada uma ficha DB9 interligada ao módulo IB IL RS232/ECO (tabela 3.4) e do lado do *Raspberry Pi* foi ligado o conversor diretamente à sua porta USB.

Contudo, de uma forma geral ambas as soluções têm o mesmo funcionamento. Realizam a conversão dos sinais RS232-C provenientes do PLC para sinais com níveis de tensão seguros a receber pelo *Raspberry Pi* para interpretação dos dados transmitidos e vice-versa. A figura 3.13 ilustra a interface RS232-C utilizada.

Figura 3.13 – Interface RS232-C

Para a interpretação dos dados enviados/recebidos foi necessário definir a construção do corpo da mensagem a enviar. Esta é composta por 9 *bytes* (8 de dados e 1 auxiliar), onde estão os estados de

todos os sensores, atuadores e variáveis de supervisão, alguns *bits* auxiliares e lista de erros. A figura 3.14 ilustra um exemplo de mensagem enviado do PLC para o *Raspberry pi* usando a comunicação série RS232-C.

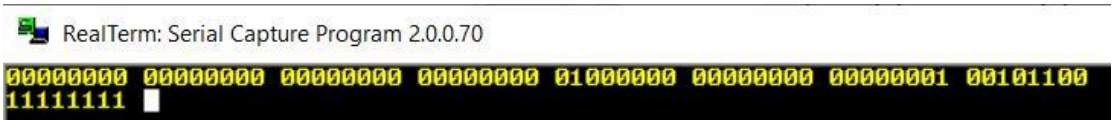


Figura 3.14 – Exemplo da mensagem PLC - Raspberry Pi em comunicação série RS232-C

Para facilitar a explicação e interpretação da mensagem através da tabela 3.5, a nomenclatura adotada será a atribuição de algarismos de 1 a 9 a cada *byte* e de 1 a 8 a cada *bit*, realizando sempre a leitura do *bit* mais ao menos significativo (i.e, da esquerda para a direita). No exemplo da figura 3.14, o *byte* 1 corresponde a *00000000* e o *byte* 9 a *11111111*. No *byte* 1 estão alguns *bits* auxiliares e lista de erros, nos *bytes* 2 e 3 está o valor real da temperatura, no *byte* 4 estão os estados dos sensores, no *byte* 5 estão ainda alguns estados de sensores e os estados dos atuadores, no *byte* 6 estão os estados dos atuadores e nos *bytes* 7 e 8 está o valor de referência da temperatura. O *byte* 9 é um *byte* auxiliar de fim de mensagem com os seus *bits* sempre a "1", para realizar uma verificação básica de erros no código do *Raspberry Pi*.

Tabela 3.5 – Descrição da mensagem PLC-Raspberry Pi

Byte	**Bit**	**Descrição**
1	1	*Bit* auxiliar de existência de dados novos
	2	*Bit* auxiliar de existência de dados novos de entrada
	3	Seis *bits* reservados à lista de erros, permitindo até 63 erros
	4	
	5	
	6	
	7	
	8	
2 e 3	---	Valor real da temperatura medido pelo sensor colocado no início do circuito
4	1	Interruptor on/off do subgrupo de controlo
	2	Sensor de nível máximo do reservatório
	3	Sensor de nível máximo do tanque de aquecimento
	4	Sensor de nível máximo do tanque de mistura
	5	Sensor de nível máximo do tanque de filtragem
	6	Sensor de nível mínimo do reservatório
	7	Sensor de nível mínimo do tanque de aquecimento
	8	Sensor de nível mínimo do tanque de mistura

Byte	Bit	Descrição
5	1	Sensor de nível mínimo do tanque de filtragem
	2	Interruptor virtual (supervisão)
	3	Seletor virtual do modo manual ou automático (supervisão)
	4	Bit auxiliar de existência de dados novos de saída
	5	Eletroválvula do reservatório
	6	Eletroválvula do tanque de aquecimento
	7	Eletroválvula do tanque de mistura
	8	Eletroválvula do tanque de filtragem
6	1	Eletroválvula do reagente do tanque de mistura
	2	Eletroválvula do reagente do tanque de mistura
	3	Eletrobomba do reservatório
	4	Eletrobomba do tanque de aquecimento
	5	Eletrobomba do tanque de mistura
	6	Eletrobomba do tanque de filtragem
	7	Resistência de aquecimento
	8	Bomba misturadora
7 e 8	---	Valor de referência para a temperatura (supervisão)
9	---	*Byte* de controlo com todos os *bits* a 1

3.2.3.2 Comunicação sem fios *Bluetooth*

Para interligar o *Raspberry Pi* e o telemóvel com a aplicação *Android* desenvolvida, foi necessário idealizar uma solução de comunicação diferente. Neste caso a solução implementada foi bastante direta, de forma a tirar partido do único modo de comunicação sem fios ponto a ponto existente em ambos os equipamentos: a comunicação *Bluetooth*. Para que a comunicação fosse possível teve de ser desenvolvido um servidor no *Raspberry Pi* e um cliente no telemóvel com a aplicação *Android*, devendo ser emparelhados manualmente na primeira conexão. O servidor foi desenvolvido em *python* e o cliente através da plataforma *MIT App Inventor*, explicados em detalhe mais à frente.

Na comunicação por *Bluetooth* foi alterada a construção do corpo das mensagens enviadas entre a aplicação *Android* e o *Raspberry Pi*, simplificando-as. Esta alteração prende-se com o facto de ser possível enviar mensagens no formato de texto com codificação UTF-8 e com o tipo de programação usado no desenvolvimento da aplicação *Android*. Na mensagem *Raspberry Pi -> App Android* foram convertidos os valores de cada variável para decimal, à exceção dos 24 *bits* que mantêm a sua representação binária de acordo com a tabela 3.5 (*bytes* 4, 5 e 6), sendo enviados no formato **##Erros,Temperatura,TemperaturaDeRef.,TemposDeEnchimento,24bitsFF**. Onde ## e **FF** representam o início e fim da mensagem e as variáveis estão divididas por vírgulas. No caso da mensagem *App Android -> Raspberry Pi* é mais simples e eficaz enviar o comando relativo ao estado

do equipamento que se quer alterar, ao invés de enviar sempre a mensagem completa com todos os estados de todos os equipamentos. Como tal, o corpo da mensagem a enviar é **##XXX00000000FF**, onde **##** e **FF** representam o início e fim da mensagem, **XXX** identifica o equipamento e **00000000** o estado ou valor do mesmo no caso da temperatura. Os equipamentos estão descritos na tabela 3.6 e os estados possíveis são 00000000 e 00000001 para desligado e ligado, respetivamente. Quanto aos valores de temperatura foram usados três algarismos para os representar com uma casa decimal, preenchendo os restantes a 0 (ex.: 00000344 = 34,4ºC). Nos tempos de enchimento dos tanques, em ambas as mensagens, cada algarismo representa um fator multiplicativo (0-0seg; 1-5seg; 2-10seg; 3-15seg; 4-20seg; 5-25seg; 6-30seg), com a ordem de tempo intermédio e tempo final dos três tanques, tempo tanque 4 e tempo tanque 5.

Tabela 3.6 – Descrição da mensagem App-Raspberry Pi

Equipamento	**Mensagem App (XXX)**	**Equipamento**	**Mensagem App (XXX)**
Interruptor on/off	STR	Eletrobomba do reservatório	EBR
Seletor modo (manual/auto)	MOD	Eletrobomba do tanque de aquecimento	EB1
Eletroválvula do reservatório	EVR	Eletrobomba do tanque de mistura	EB2
Eletroválvula do tanque de aquecimento	EV1	Eletrobomba do tanque de filtragem	EB3
Eletroválvula do tanque de mistura	EV2	Resistência de aquecimento	RES
Eletroválvula do tanque de filtragem	EV3	Bomba misturadora	MIS
Eletroválvula do reagente do tanque de mistura	EV4	Valor de referência para a temperatura	TMP
Eletroválvula do reagente do tanque de mistura	EV5	Tempos de enchimento	TIM

3.3 Funcionamento do Simulador

No ponto anterior foi descrita a componente de *hardware* do simulador, assim como a sua estrutura. Neste ponto irá ser apresentado com detalhe o *software* desenvolvido para o simulador didático.

Pela componente didática do simulador, foi pensada uma solução visual, prática e intuitiva relacionada com o tema das estações de tratamento de água. O processo consiste num ciclo de fluxo da água existente no sistema pelos quatro tanques, iniciado no reservatório e simulando três etapas:

aquecimento, mistura e filtragem. Antes de iniciar, toda a água a circular no sistema deve estar no reservatório, enchendo-o manualmente até ao nível máximo, para garantir que é suficiente para encher os tanques das etapas seguintes. Foi planeada a existência de dois interruptores para início e paragem do sistema, com encravamento para segurança do mesmo: um físico presente no painel da porta do quadro elétrico e um virtual através do programa de supervisão. Após iniciado, o simulador permite dois modos de funcionamento do processo – manual e automático – sendo as ordens de comando, em manual, realizadas através da supervisão existente no *Raspberry Pi* e/ou paralelamente pela aplicação *Android*.

No modo automático o processo de bombagem é realizado de forma sequencial e cíclica, correndo todos os tanques e regressando novamente ao reservatório de armazenamento, continuando ciclicamente enquanto todas as condições se verificarem. Foi pensada no decorrer do trabalho, uma melhoria no simulador que consiste numa solução interativa com base nos tempos de enchimento de cada tanque, no entanto não foi implementada na sua totalidade, sendo sugerida como trabalho futuro. Tentou-se que a programação desenvolvida fosse o mais flexível e intuitiva possível, podendo ser alterada no futuro por colegas ou alunos através da normal programação do autómato e dos parâmetros do corpo das mensagens já descritos.

3.3.1 Descrição do processo simulado

O processo simulado, tal como foi mencionado, consiste num ciclo de fluxo da água existente no sistema pelos quatro tanques, iniciado no reservatório, simulando as etapas de aquecimento, mistura e filtragem de uma ETA. Inicialmente, toda a água deve estar armazenada no reservatório antes do início do ciclo após ligar o simulador, para garantir que a água é suficiente para completar o ciclo de operações. Como o sistema permite um modo manual de funcionamento, é permitido, neste modo ao utilizador a manipulação de todas as eletroválvulas e eletrobombas, alterando os níveis dos tanques à sua vontade. No entanto isto cria uma dificuldade acrescida quando se transita do modo manual para automático dada a incerteza dos níveis dos tanques após cada utilização em modo manual. Por isso, foi necessário a criação de um algoritmo de inicialização que permita ao PLC garantir que as condições iniciais são cumpridas e que existe água suficiente para completar o ciclo em modo automático.

Com o comando manual do processo por parte do utilizador, após a manipulação das eletroválvulas e eletrobombas, existem diversos cenários possíveis na distribuição da água. Deste modo, o algoritmo de inicialização deve ser capaz de repor as condições iniciais para que o funcionamento cíclico em modo automático possa funcionar adequadamente. O algoritmo de inicialização deve começar por garantir que o reservatório tem o nível desejado de água. De seguida, o algoritmo verifica o estado dos sensores dos tanques. Pelo facto de cada tanque ter apenas dois sensores indicando o seu nível máximo e mínimo, não é possível obter informação relativa ao nível exato do seu enchimento. O que significa que com o nível mínimo e sem o nível máximo, o tanque tanto pode estar praticamente vazio como cheio. Como o sensor de nível mínimo não está colado à base do recipiente, existe ainda a

possibilidade de o tanque estar completamente vazio. Neste cenário, se o sistema iniciar sem água, existe o risco de estragar as eletrobombas por trabalharem a seco. Para incluir todos os cenários possíveis foram realizadas duas verificações: se pelo menos um dos três tanques não tem água (i.e. não tem nível mínimo) ou se os três tanques têm água (i.e. têm nível mínimo). A figura 3.15 ilustra a posição do nível máximo e mínimo do tanque de mistura e a diferença entre a base e o nível mínimo (a amarelo na figura), que representa a quantidade possível de água existente mesmo sem nível mínimo.

Figura 3.15 – Diferença entre a base dos tanques e o sensor de nível mínimo

Na primeira situação é despoletada uma sequência de eventos que permite a transferência da água existente entre tanques até que fiquem quase vazios (i.e. perder o estado de nível mínimo). Como os tanques não ficam realmente vazios é necessário garantir que a água existente no primeiro tanque é suficiente para cobrir o volume de água para os restantes. Pelo facto de não ser possível precisar o volume de água existente no tanque após o nível mínimo, pelo que já foi explicado, a solução desenvolvida obriga o enchimento e esvaziamento de todos os tanques até ao nível mínimo. Para isso, o sistema começa por bombear a água do reservatório para o tanque de aquecimento até o encher ou o próprio ficar vazio. De seguida bombeia a água do tanque de aquecimento para o tanque de mistura até ser atingido o nível mínimo do primeiro. O mesmo processo é repetido para os tanques 2 e 3 até todos os tanques estarem quase vazios e o reservatório cheio. Existe ainda o controlo do estado dos sensores de nível máximo dos tanques parando o processo, para que não sejam ultrapassados com risco de fazer trasbordar a água. Na segunda situação, onde os três tanques têm o nível mínimo, é realizado o bombeamento da água entre tanques, em simultâneo, até perderem o seu nível mínimo ficando quase vazios. Nesta fase é também realizado o controlo do estado dos sensores de nível máximo.

Realizado o algoritmo de inicialização, o processo pode começar em modo automático podendo, no entanto, passar para modo manual em qualquer momento. Sempre que seja manipulada alguma eletroválvula ou eletrobomba em modo manual, o simulador obriga o processo a realizar novamente o algoritmo de inicialização antes de voltar ao modo automático. No modo automático é realizado o processo de forma sequencial e cíclica, correndo todas as etapas e regressando ao reservatório novamente, continuando indefinidamente se todas as condições de verificarem. O modo automático começa com enchimento de metade do tanque de aquecimento, bombeando a água do reservatório, iniciando nesta etapa o aquecimento da água nele contida. Ao fim de alguns segundos é bombeado o restante até ao nível máximo e o processo aguarda até que a temperatura da água atinja a temperatura de referência definida pelo utilizador (a temperatura é medida por um sensor de temperatura analógico). Após atingir a temperatura de referência, a água é bombeada do tanque de aquecimento para o tanque de mistura até atingir novamente um nível médio (contabilizado por um tempo pré-definido), acima da bomba misturadora para que esta possa funcionar de forma adequada. São adicionados posteriormente dois reagentes e iniciada a mistura durante alguns segundos. Após realizada a mistura, a água é bombeada para o tanque de filtragem onde é filtrada. Antes de ser bombeada novamente para o reservatório de armazenamento é realizada uma pausa de alguns segundos no sistema para a estabilização da água e visualização do valor de temperatura final. Assim que seja atingido o nível máximo de água no reservatório de armazenamento, o processo está em condições de reiniciar o ciclo, bombeando a água para o tanque de aquecimento. Em qualquer momento o processo pode ser parado com o interruptor *on/off* do quadro ou pela supervisão, retomando do mesmo ponto assim que tenha novamente a instrução para iniciar. O processo termina quando é realizada a transição para o modo manual, permitindo ao utilizador os comandos através da supervisão (ecrã táctil na porta do quadro) ou através da aplicação no telemóvel. Quando é realizada a transição inversa, de manual para automático, sempre que seja acionada alguma eletroválvula ou eletrobomba, o processo realiza sempre o algoritmo de inicialização como já foi descrito. Caso contrário, ou seja, sem qualquer modificação e desde que tenha a instrução para iniciar em modo automático, o processo retoma a etapa onde tinha parado na sequência do modo automático anterior.

3.3.2 Programação do controlador

O controlo do simulador didático, como já foi referido, está a cargo de um controlador industrial. Todas as instruções e tarefas descritas serão realizadas por ele, assim como as operações de segurança. Como tal, para desenvolver o simulador e implementar o processo foi necessário configurar e programar o PLC ILC131 ETH da Phoenix Contact através do software PCWORX. A figura 3.16 ilustra o aspeto geral do software de programação da PCWORX.

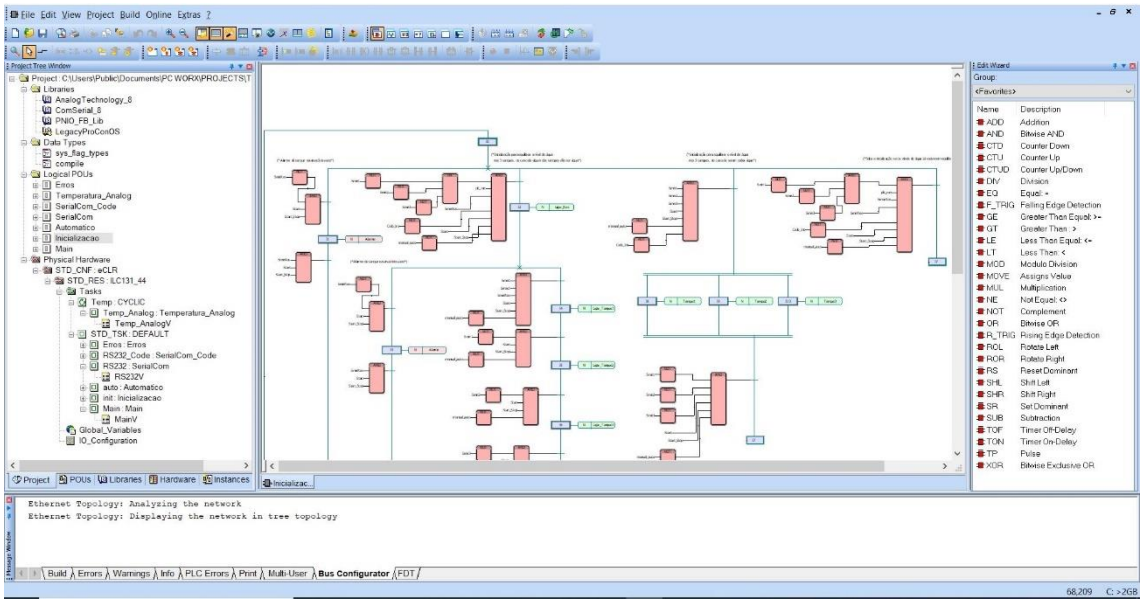


Figura 3.16 – Software de desenvolvimento PCWORX

Os primeiros passos a tomar foram a criação de um novo projeto selecionando o *template ILC 131 ETH Rev. > 00/4.40* correspondente ao PLC usado e configuração dos diversos módulos do mesmo. Para tal, é necessário entrar na janela "*Bus Configuration Workspace*", escolhendo os módulos desejados na lista de equipamentos (*device catalog*) e inserindo-os através da opção "*Insert Device into Bus Structure*".

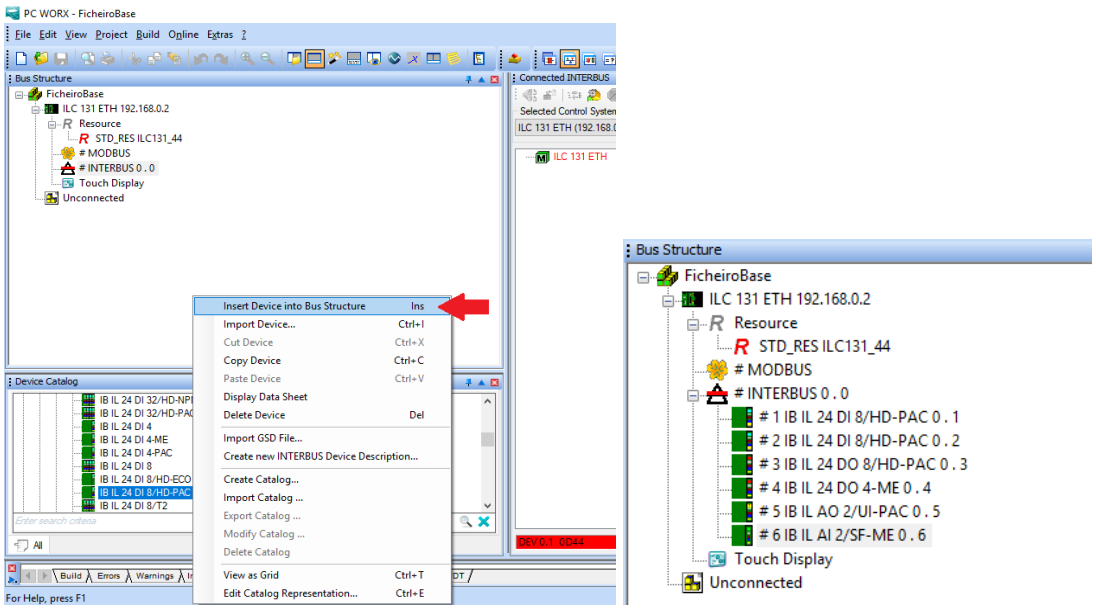


Figura 3.17 – Configuração dos módulos do PLC

A programação do PLC foi desenvolvida com recurso a quatro das cinco linguagens da norma IEC 61131-3: blocos lógicos, *ladder*, *grafcet* e textual estruturada. Está dividida em sete programas, implementando o algoritmo de inicialização, o modo automático, o modo manual (entre outros), a

listagem de erros, o tratamento da entrada analógica e da comunicação série. Por ser sequencial, optou-se por programar o algoritmo de inicialização e o modo automático recorrendo à linguagem de *grafcet*. O modo manual e tratamento da entrada analógica foram realizados em linguagem de blocos lógicos e *ladder* e a listagem de erros em linguagem textual estruturada. Para a comunicação série foram programados dois programas: um em linguagem de blocos lógicos para configurar o módulo RS232 e outro em linguagem textual estruturada para definir o modo de comunicação. Todos os programas estão a correr no ciclo interno do PLC, à exceção do tratamento da entrada analógica que corre num ciclo à parte, a cada 100ms.

3.3.2.1 Algoritmo de inicialização

O algoritmo já foi descrito no ponto anterior, pelo que apenas irá ser focada a sua implementação no software, no programa "Inicializacao". Para o seu desenvolvimento recorreu-se à linguagem de *grafcet*, por ser sequencial e, aproveitando algumas das características do PCWORX, foi utilizada a linguagem de blocos lógicos e *ladder* para as transições e ações. Os diagramas apresentados serão grafcets de nível 2 (especificações tecnológicas). Para otimizar a escrita, pela sua repetição, foi convencionada a representação da operação AND entre o interruptor *on/off* (start), o botão virtual (start_stop) e o seletor do modo (manual_auto) como ***INIT***. Desta forma, sempre que aparecer ***INIT*** nos *grafcets*, leia-se ***start . start_stop . $\overline{manual\_auto}$*** (programa iniciado fisicamente e pela supervisão em modo automático). Todas as ações estão condicionadas pela operação descrita, para que o sistema pare e retome no mesmo ponto, sempre que as condições iniciais não se verifiquem.

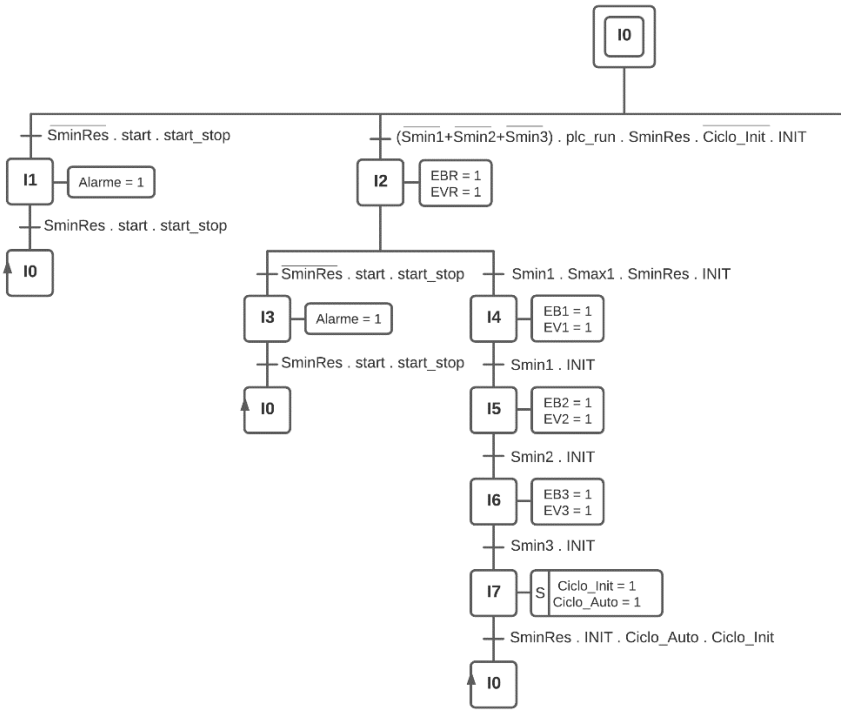


Figura 3.18 – Grafcet da situação inicial onde pelo menos um dos três tanques não tem água

Na figura 3.18 estão representados dois ramos em detalhe do *grafcet* do algoritmo, representando a primeira situação onde pelo menos um dos três tanques não tem água (i.e. não tem nível mínimo). Foram criadas duas variáveis auxiliares – *Ciclo_Init* e *Ciclo_Auto* – para controlo do estado do programa. A variável *Ciclo_Init* indica se o programa já realizou o algoritmo de inicialização, passando ao estado "1" quando o fizer. No caso da variável *Ciclo_Auto*, esta muda o seu estado para "1" quando o programa está apto a iniciar em modo automático. A variável *plc_run* indica quando o PLC passa ao estado RUN. Nas etapas I1 e I3 é realizada a verificação de reservatório vazio. Esta verificação é realizada antes do sistema iniciar e depois da primeira bombagem de água do reservatório para o tanque de aquecimento (etapa I2). Caso o reservatório esteja vazio, é acionada uma variável de alarme e o programa regressa à etapa inicial I0. Se tiver água, o programa continua bombeando sequencialmente a água entre tanques, enchendo e esvaziando-os até todos estarem abaixo do nível mínimo. No fim, na etapa I7, é feito um *set* às variáveis *Ciclo_Init* e *Ciclo_Auto*, indicando que o algoritmo de inicialização foi realizado e que o programa está pronto para iniciar em modo automático.

Na segunda situação, já descrita, onde os três tanques têm o nível mínimo, é realizado o bombeamento da água entre tanques, em simultâneo, até perderem o seu nível mínimo. O *grafcet*, tendo as condições iniciais verdadeiras (ter os níveis mínimos, o start, start_stop e estar em modo automático, mas não ter realizado o algoritmo de inicialização), entra numa ramificação AND, executando as etapas I8, I9 e I10 em simultâneo até serem atingidas as condições de saída. Nas ações das etapas é realizado o bombeamento da água para o tanque seguinte, ligando a respetiva eletroválvula e eletrobomba. As ações são condicionadas, sendo realizado o controlo do estado dos sensores de nível mínimo. Quando todos os tanques perdem a sinalização de nível mínimo dado pelo sensor, o *grafcet* está em condições de sair da ramificação AND, regressando à etapa I7 onde é finalizado o algoritmo de inicialização. A representação em detalhe do seu ramo no *grafcet* pode ser analisada na figura 3.19.

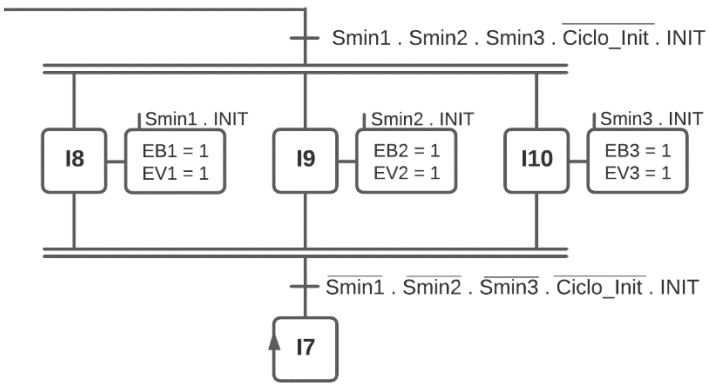


Figura 3.19 – Grafcet da situação inicial onde os três tanques têm água

3.3.2.2 Modo Automático

Finalizado o algoritmo de inicialização, o programa está apto a iniciar em modo automático, desde que tenha água no reservatório mesmo que não esteja totalmente cheio, tenha indicação de *start* (fisicamente e pela supervisão) e tenha o seletor em modo automático (*manual_auto* = 0). A representação do seu *grafcet* pode ser analisada na figura 3.20, correspondendo ao programa "Automatico" no PLC.

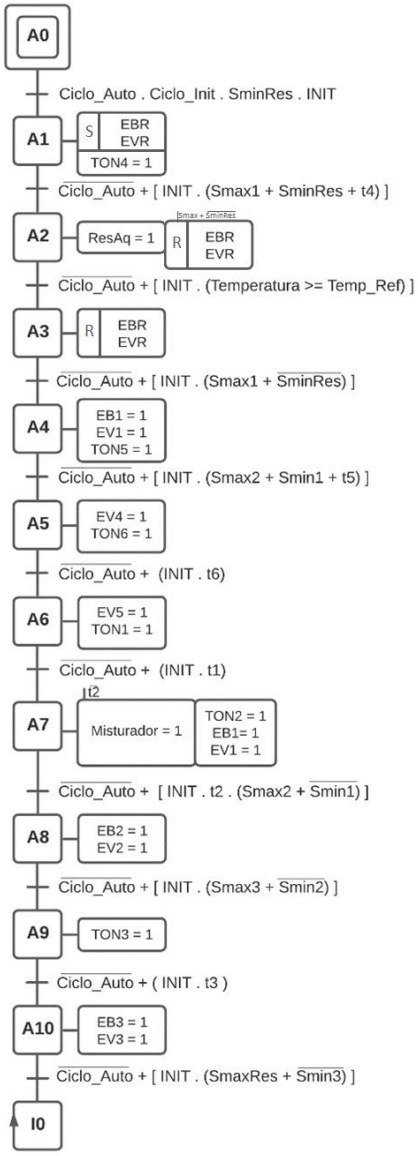


Figura 3.20 - Grafcet do programa em modo automático

Iniciado o modo automático, todas as ações são realizadas sequencialmente enquanto as respetivas transições o permitirem. A indicação de *start* (fisicamente e pela supervisão) e o seletor em modo automático têm de estar sempre presentes ou o programa termina. Caso o sistema seja alterado em modo manual (realizada qualquer mudança de estado em modo manual), a variável *Ciclo_Auto* passa ao estado "0" e o programa volta à etapa A0. Na etapa A1, é realizado o bombeamento da água do

reservatório para o tanque de aquecimento e acionado um temporizador com o tempo suficiente para encher metade do tanque. Na etapa A2, é ligada a resistência de aquecimento. O programa fica depois à espera de que a água aqueça até atingir o valor de referência, podendo terminar o enchimento do tanque de aquecimento na etapa A3 caso ainda não tenha terminado. Com o tanque de aquecimento cheio ou o reservatório vazio, é iniciado o enchimento do tanque de mistura com a bombagem da água na etapa A4 e iniciado outro temporizador, à semelhança do que foi feito na etapa A1. O tanque de mistura enche até metade controlado pelo temporizador TON5 e, à vez, liga as eletroválvulas dos reagentes EV4 e EV5 durante 6 segundos através dos temporizadores TON6 e TON1 nas etapas A5 e A6. Na etapa A7, é de seguida ligada a bomba misturadora durante o tempo definido no temporizador TON2, terminando o enchimento do tanque de mistura no fim do tempo para mistura. A transição para a etapa seguinte dá-se quando o temporizador chegar ao fim e o tanque de mistura estiver cheio ou o tanque de aquecimento vazio. Na etapa A8 é realizada a bombagem da água para o tanque de filtragem, parando quando estiver cheio ou o tanque de mistura vazio. Com o tanque de filtragem cheio, o sistema aguarda 10 segundos antes de continuar, através do temporizador TON3, iniciado na etapa A9. Finalmente, na etapa A10 é realizada a bombagem da água novamente para o reservatório até ao seu nível máximo ou até o nível mínimo do tanque de filtragem, regressando à etapa A0. Desde que tenha água no reservatório mesmo que não esteja totalmente cheio, tenha indicação de *start* (fisicamente e pela supervisão) e tenha o seletor em modo automático (*manual_auto* = 0), o programa está apto para repetir o ciclo.

3.3.2.3 Modo Manual (programa principal)

A programação do modo manual foi realizada no programa "Main", juntamente com a atribuição das entradas/saídas internas do PLC e os encravamentos de segurança, entre outros. O programa "Main" foi realizado em *ladder* com ajuda, por vezes, da linguagem de blocos lógicos. Pode ser dividido em cinco partes que serão apresentadas e explicadas de seguida.

Assim que o simulador é alimentado e iniciado, o PLC e o *Raspberry Pi* são ligados. Para garantir que a interface no *Raspberry Pi* inicia antes do programa do PLC começar, foi implementado um atraso de 20 segundos no programa (figura 3.21).

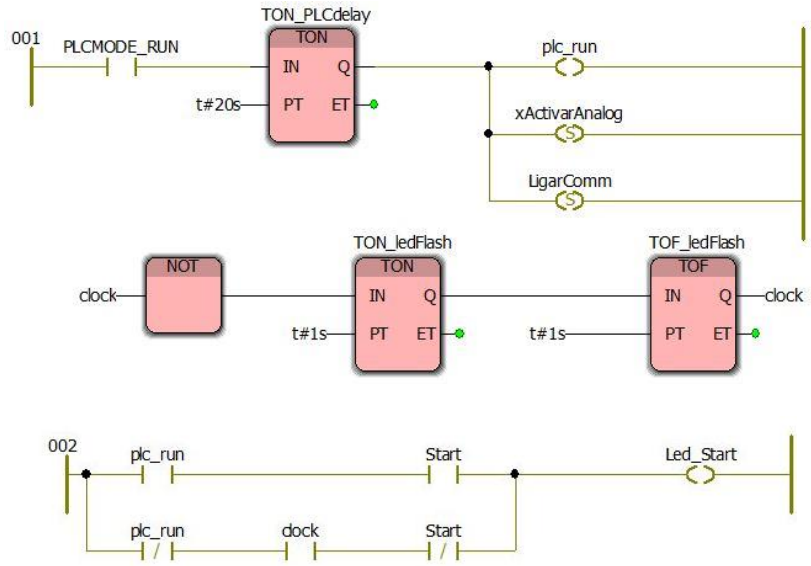


Figura 3.21 – Representação do atraso no início do programa no PLC e alerta visual ao utilizador

Analisando a figura anterior, quando o PLC entra no seu modo RUN, é acionado um temporizador TON que, terminando o tempo, ativa a variável *plc_run* já mencionada e as variáveis *xActivarAnalog* e *LigarComm*, explicadas mais à frente, que inicializam os programas de tratamento da entrada analógica e de comunicação série. Durante o tempo de atraso é colocado o led do interruptor *on/off* a piscar intermitente com cadência de um segundo, para informar o utilizador. Terminando o tempo de atraso o led apaga, acendendo apenas quando o interruptor é acionado.

Ao longo de todos os programas desenvolvidos vão sendo chamadas e manipuladas as mesmas variáveis correspondendo aos sensores/atuadores físicos. No entanto, se tiverem o mesmo nome vão sobrepor-se, sendo executadas independentemente do ponto ou etapa em que estão. Para ultrapassar essa limitação foram criadas variáveis auxiliares para cada utilização, sendo depois associadas à variável principal. Na figura 3.22 está representado um excerto da programação *ladder* explicada.

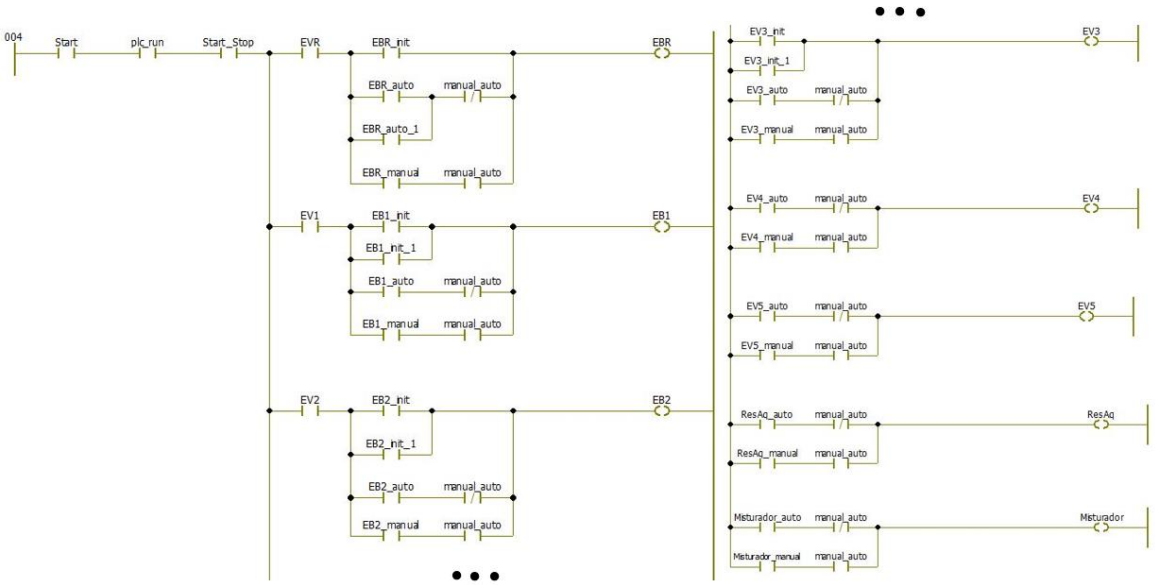


Figura 3.22 – Representação da associação de variáveis auxiliares às principais

Sempre que um elemento é manipulado, seja no algoritmo de inicialização, em modo automático ou modo manual, é ativada a variável auxiliar respetiva e associada à variável geral. É ainda realizado um encravamento de segurança, no caso das eletrobombas, para apenas funcionarem com a respetiva eletroválvula aberta. As variáveis gerais são internas do PLC, tendo de ser atribuídas às entradas e saídas físicas tal como está representado na figura 3.23. A atribuição das restantes variáveis aos respetivos módulos é realizada tal como apresentado na figura 3.35 e figura 3.36, de acordo com a tabela 3.1 à tabela 3.4.

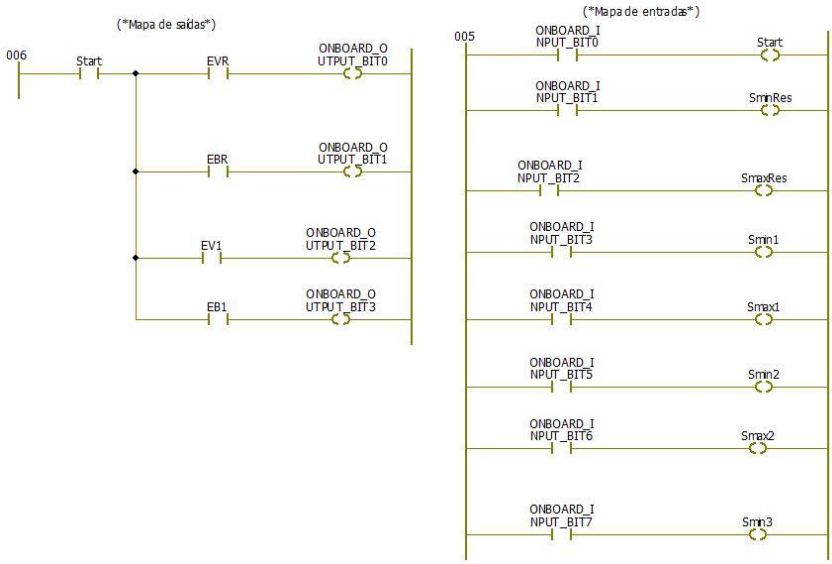


Figura 3.23 – Atribuição das variáveis gerais às entradas e saídas físicas do PLC

No programa “Main” são realizados mais dois encravamentos de segurança já explicados. Quando o simulador se encontra em modo manual, o utilizador pode manipular à sua vontade qualquer elemento do sistema. Para garantir que o simulador inicia o algoritmo de inicialização quando regressa ao modo automático, é realizado um *reset* às variáveis *Ciclo_Init* e *Ciclo_Auto* sempre que o estado de alguma eletrobomba ou eletroválvula seja alterado. No entanto, se o simulador entrar em modo manual, mas não for manipulado nenhum atuador, o simulador não é obrigado a realizar o algoritmo, com o propósito de o otimizar.

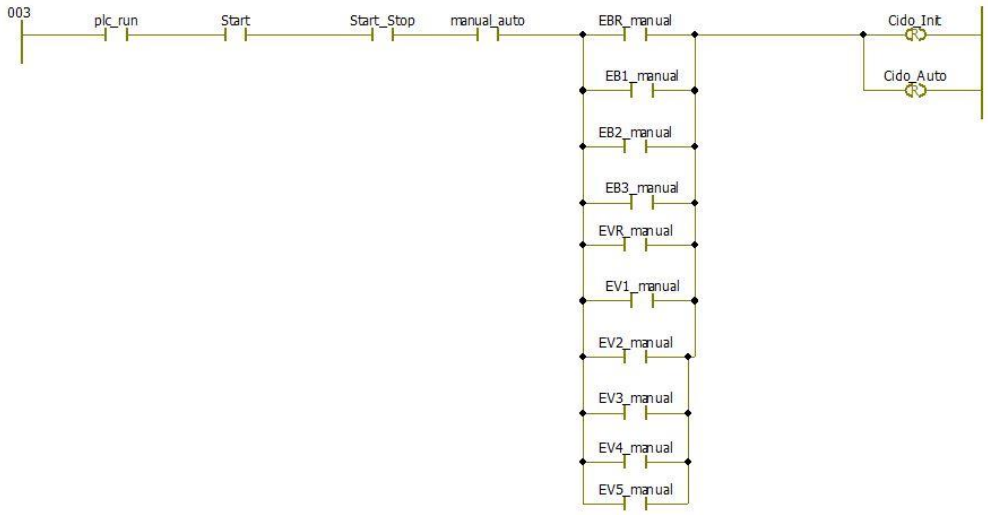


Figura 3.24 – Representação do encravamento de segurança para o modo manual

A qualquer momento o programa deve parar quando perde o estado do interruptor on/off. Esta paragem deve ser realizada mesmo numa situação extrema de bloqueio da interface ou perda de comunicação. Como tal, o estado do interruptor está presente como condição em todas as transições e ações do programa. Sendo realizado um *reset* a todos os atuadores do sistema quando é dada a ordem para parar (interruptor *off*).

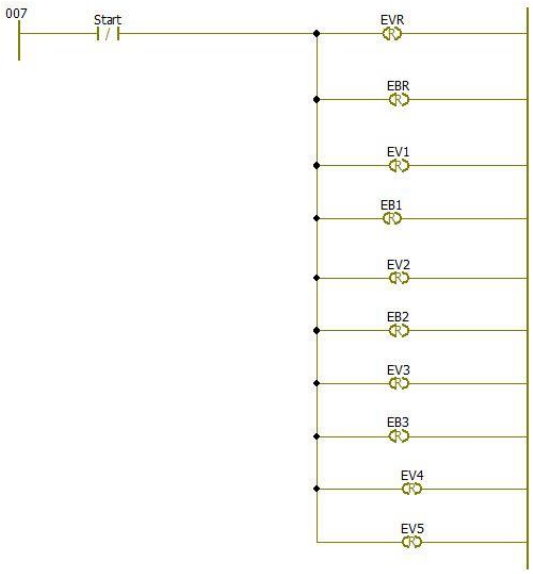


Figura 3.25 – Representação do encravamento de segurança para o interruptor on/off (start)

3.3.2.4 Listagem de erros

Para a listagem dos erros/mensagens do simulador didático, foi realizado um programa recorrendo à linguagem textual estruturada. Dependendo das condições que despoletam o erro/mensagem é escrito um inteiro na variável do tipo *byte Error_byte*. Esse valor é enviado para o *Raspberry Pi* sendo interpretado e representado na interface. A listagem é dinâmica, podendo ser alterada no PLC e no *Raspberry Pi* sempre que necessário (figura 3.26).

```
1   (* Listagem de Erros/Mensagens *)
2
3   (* Programa parado *)
4   IF (Start_Stop = FALSE OR Start = FALSE) THEN
5       Error_byte :=INT_TO_BYTE(1);
6   END_IF;
7
8   (* Modo Manual *)
9   IF (Start_Stop = TRUE AND Start = TRUE AND manual_auto = TRUE) THEN
10      Error_byte :=INT_TO_BYTE(2);
11  END_IF;
12
13  (* Modo Automatico *)
14  IF (Start_Stop = TRUE AND Start = TRUE AND manual_auto = FALSE) THEN
15      Error_byte :=INT_TO_BYTE(3);
16  END_IF;
17
18  (* Reservatorio vazio *)
19  IF Alarme = TRUE THEN
20      Error_byte :=INT_TO_BYTE(4);
21  END_IF;
```

Figura 3.26 – Excerto da listagem de erros/mensagens do PLC em ST

3.3.2.5 Tratamento da entrada analógica

O tratamento da entrada analógica foi realizado recorrendo à biblioteca *AnalogTechnology_8* do PCWORX através da linguagem de blocos lógicos. Para a parametrização da entrada analógica foi usado o bloco IL_AI_2_SF_ V1_02_1, representado na figura 3.27.

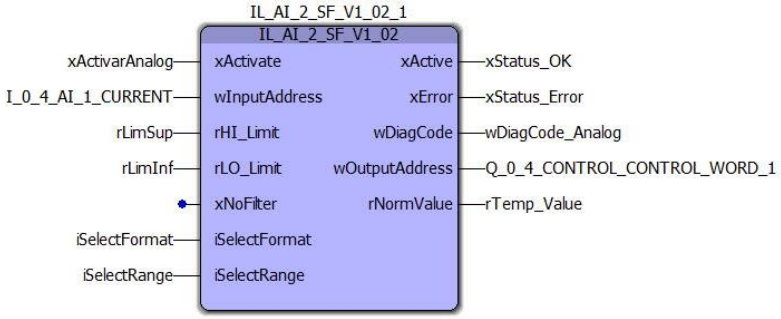


Figura 3.27 – Bloco para parametrização da entrada analógica

O bloco IL_AI_2_SF_ V1_02_1 converte o valor *word* – correspondente ao sinal em corrente que chega ao PLC – para formato real, de acordo com a escala definida, armazenando-a posteriormente numa variável. Este é iniciado com a variável booleana *xActivarAnalog* através do programa "Main". Do seu lado esquerdo encontram-se os parâmetros de entrada e à direita os de saída, explicados na tabela 3.7 e tabela 3.8.

Tabela 3.7 – Nome, tipo de dados e descrição dos parâmetros de entrada do bloco IL_AI_2_SF_ V1_02_1

Nome	Tipo de dados	Descrição
xActivate	Booleano	Ativa o bloco
wInputAddress	Word	Variável fixa de entrada
rHI_Lim	Real	Limite superior para o alcance escolhido
rLO_Lim	Real	Limite inferior para o alcance escolhido
xNoFilter	Booleano	Em modo padrão, é ativado um filtro que gera a medida dos últimos 16 valores. Esta entrada permite inativá-lo
iSelectFormat	Inteiro	Escolha do formato do módulo utilizado. Neste caso, como o módulo é da gama IL, a entrada é 0
iSelectRange	Inteiro	Modo e alcance do input. Neste caso é unipolar (4-20mA) e por isso a entrada é 2

Tabela 3.8 – Nome, tipo de dados e descrição dos parâmetros de saída do bloco IL_AI_2_SF_ V1_02_1

Nome	Tipo de dados	Descrição
xActive	Booleano	TRUE indica funcionamento sem problemas
xError	Booleano	TRUE indica um erro de parametrização
wDiagCode	Word	Mostra o código de erro
wOutputAddress	Word	Variável fixa de saída
rNormValue	Real	Valor REAL correspondente ao sinal de entrada de acordo com a escala definida pelo utilizador (rLO_Limit e rHI_Limit).

As variáveis *wInputAddress* e *wOutputAddress* fazem a interligação direta com o módulo usado IB IL AI2 SF/ME. A sua escolha é realizada na janela "*Process Data Assigment*" atribuindo "*Process Data Items*" específicos de acordo com a grandeza elétrica da entrada. No caso do presente projeto a entrada é em corrente, pelo que é escolhida a variável *I_0_6_A_1_CURRENT* para a *wInputAddress*. No caso da *wOutputAddress* é indiferente a sua grandeza pois a variável é sempre *Q_0_6_CONTROL_CONTROL_WORD_1*. No fim do tratamento da entrada analógica o seu valor é guardado na variável do tipo real *rNormValue*, que é depois movida para a variável global *Temperatura* para uso dos restantes programas.

3.3.2.6 Comunicação série

O programa de implementação da comunicação série "SerialCom" é um pouco mais complexo que o das entradas analógicas. Para a parametrização do módulo foi escolhida a biblioteca *ComSerial_8* do PCWORX e usados dois blocos lógicos: *IL_RS232_ECO_5_1* e *IL_RS232_ECO_Diag_1_1*. Foram ainda realizados o tratamento dos valores reais e de referência de temperatura e a interligação das variáveis globais do PLC com as variáveis de envio e receção do módulo.

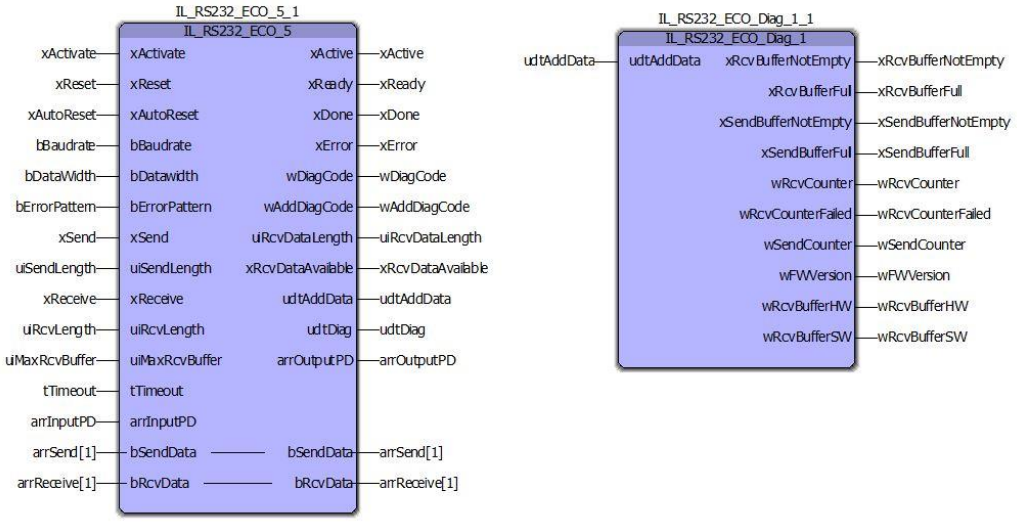


Figura 3.28 – Blocos para parametrização do módulo de comunicação série

É o bloco *IL_RS232_ECO_5_1* que parametriza o módulo *IB IL RS232/ECO*, usando o *IL_RS232_ECO_Diag_1_1* como bloco auxiliar para descodificar o parâmetro de saída *udtAddData*. Todos os parâmetros de entrada e saída estão explicados na tabela 3.9 à tabela 3.11.

Tabela 3.9 – Nome, tipo de dados e descrição dos parâmetros de entrada do bloco IL_RS232_ECO_5_1

Nome	Tipo de dados	Descrição
xActivate	Booleano	Ativa o bloco quando está no flanco ascendente e desativa-o se for FALSE
xReset	Booleano	Faz reset ao bloco quando está no flanco ascendente
xAutoReset	Booleano	Elimina os erros automaticamente, sendo indicados nos parâmetros de saída *xError*, *wDiagCode e wAddDiagCode* durante um ciclo. O bloco é depois reiniciado
bBaudrate	Byte	Indica o valor do baudrate. O valor escrito no parâmetro é hexadecimal (0 = 110; 1 = 300; 2 = 600; 3 = 1200; 4 = 1800; 5 = 2400; 6 = 4800; 7 = 9600; 8 = 15625; 9 = 19200; A = 38400; B-F = 7 bytes reservados)
bDataWidth	Byte	Indica o tamanho e construção da mensagem. Os valores são escritos em hexadecimal: 0 = 7 bits de dados, par, 1 stop bit 1 = 7 bits de dados, ímpar, 1 stop bit 2 = 8 bits de dados, par, 1 stop bit 3 = 8 bits de dados, ímpar, 1 stop bit 4 = 8 bits de dados, nenhum, 1 stop bit 5 = 7 bits de dados, nenhum, 1 stop bit 6 = 7 bits de dados, par, 2 stop bits 7 = 7 bits de dados, ímpar, 2 stop bits 8 = 8 bits de dados, par, 2 stop bits 9 = 8 bits de dados, ímpar, 2 stop bits A = 8 bits de dados, nenhum, 2 stop bits B = 7 bits de dados, nenhum, 2 stop bits C = 8 bits de dados, const. 0, 1 stop bit D = 8 data bits, const. 1, 1 stop bit E = 6 bits de dados, nenhum, 1 stop bit F = reservado
bErrorPattern	Byte	Define o padrão da mensagem de erro: 24 = $ xx = Qualquer caracter 00 = Se um caracter é recebido com erro, nenhum padrão é guardado FF = O caracter inválido é guardado em vez do padrão de erro
xSend	Booleano	Envia os dados no flanco ascendente e a sua transmissão é monitorizada com um timeout
uiSendLength	Inteiro (UINT)	Número de bytes a enviar
xReceive	Booleano	Faz a leitura dos dados no flanco ascendente, sendo monitorizada com um timeout
uiRcvLength	Inteiro (UINT)	Número de bytes a ler. Se o valor for 0, todos os dados serão lidos. Se os dados não estiverem acessíveis no momento da leitura, o bloco aguarda pelos restantes. O tempo de leitura é monitorizado com um timeout
uiMaxRcvBuffer	Inteiro (UINT)	Especificação do tamanho do *array* de bytes usado no parâmetro *bRcvData*
tTimeout	Tempo	Valor de timeout para monitorização de um comando para o modulo e respetiva resposta
arrInputPD	RSUNI_ARR_B_1_14	Dados de entrada do processo

Tabela 3.10 – Nome, tipo de dados e descrição dos parâmetros de saída do bloco IL_RS232_ECO_5_1

Nome	Tipo de dados	Descrição
xActive	Booleano	Indica se o bloco está ativo
xReady	Booleano	Indica se o bloco está pronto a executar serviços. Caso seja "0", o bloco está a executar serviços
xDone	Booleano	Indica se o pedido foi realizado e recebido pelo cliente
xError	Booleano	É "1" se existir algum erro. Os detalhes do erro encontram-se em *wDiagCode* and *wAddDiagCode*
wDiagCode	Word	Código de diagnóstico do erro
wAddDiagCode	Word	Código adicional de diagnóstico do erro
uiRcvDataLength	Inteiro (UINT)	Número de bytes recebidos
xRcvDataAvailable	Booleano	Indica se ainda existem bytes em memória para ler
udtAddData	RSUNI_UDT_DATA_V1	Estados adicionais das variáveis. Pode ser descodificado pelo bloco IL_RS232_ECO_Diag
udtDiag	COM_UDT_232ECO_DIAG	Informação de diagnóstico
diBaudrate	Inteiro (DINT)	Valor de baudrate usado
arrOutputPD	RSUNI_ARR_B_1_14	Dados de saída do processo

Tabela 3.11 – Nome, tipo de dados e descrição dos parâmetros de saída do bloco IL_RS232_ECO_Diag_1_1

Nome	Tipo de dados	Descrição
xRcvBufferNotEmpty	Booleano	Indica que o buffer de receção não está vazio e os dados podem ser lidos
xRcvBufferFull	Booleano	Indica que o buffer de receção está cheio
xSendBufferFull	Booleano	Indica que o buffer de transmissão está cheio
xSendBufferNotEmpty	Booleano	Indica que o buffer de transmissão não está vazio
wRcvCounter	Word	Número de caracteres válidos recebidos
wRcvCounterFailed	Word	Número de caracteres inválidos recebidos
wSendCounter	Word	Número de caracteres enviados
wFWVersion	Word	Versão do *firmware* do módulo
wRcvBufferHW	Word	Número de caracteres válidos recebidos (hardware)
wRcvBufferSW	Word	Número de caracteres válidos recebidos no buffer (software)

Tabela 3.12 – Nome, tipo de dados e descrição dos parâmetros de entrada do bloco IL_RS232_ECO_Diag_1_1

Nome	Tipo de dados	Descrição
udtAddData	RSUNI_UDT_DATA_V1	Estados adicionais das variáveis

Os parâmetros *bSendData* e *bRcvData* são ambos *array's* do tipo *byte* contendo os dados a enviar e a ler, respetivamente. Para garantir o envio e leitura corretos dos dados foi necessário criar três blocos lógicos para monitorização e *reset* do bloco IL_RS232_ECO_5_1, em conjunto com o programa "SerialCom_Code" desenvolvido em linguagem textual estruturada (figura 3.29).

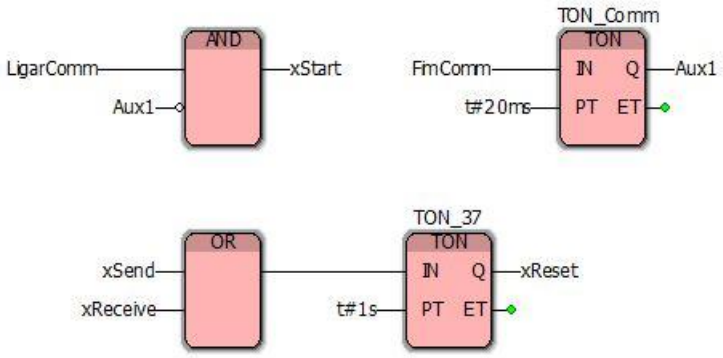


Figura 3.29 – Blocos lógicos para monitorização e reset do bloco IL_RS232_ECO_5_1

Os primeiros dois blocos lógicos monitorizam a variável booleana de inicialização do programa de comunicação série chamada quando o PLC inicia e uma variável booleana de fim de comunicação – *FimComm*. O início de um ciclo de comunicação é assinalado com a variável booleana *xStart* quando o PLC inicia e após 20 segundos do fim do último ciclo. Durante a comunicação é realizado o envio e receção dos dados intercalados a cada segundo, realizando um *reset* ao bloco no fim da ação.

No programa em linguagem textual estruturada "SerialCom_Code" já mencionado é parametrizada e implementada a estrutura e ordem do ciclo de comunicação. A sua programação foi organizada num *switch case* onde são identificados diferentes trechos de código por algarismos, sendo executados de acordo com as condições de transição (figura 3.30).

```
CASE iState OF
0: (*wait for start*)
    xActivate   := FALSE;
    xSend   := FALSE;
    xReceive    := FALSE;
    IF xStart = TRUE THEN
        iState  := 10;
    END_IF;
10: (*parametrization of the function block*)
    bBaudrate   := BYTE#16#07;
    bDataWidth  := BYTE#16#02;
    uiMaxRcvBuffer  := UINT#1024;
    uiMaxRcvBuffer := UINT#1024;
    tTimeout    := TIME#500ms;
    uiSendLength    := UINT#9;
    uiRcvLength := UINT#9;
    iState  := 20;
20: (*activate the function blocks*)
    xActivate   := TRUE;
    IF (xActive = TRUE AND xReady = TRUE) THEN
        iState  := 30;
    END_IF;
    IF (xReset = TRUE) THEN
        iState  := 75;
    END_IF;
```

Figura 3.30 – Excerto do código do programa "SerialCom_Code"

O início do ciclo de comunicação dá-se com a variável booleana *xStart* ativa, mas com o bloco IL_RS232_ECO_5_1 e respetivas variáveis de envio e receção desativas, avançando para a parametrização do mesmo. A comunicação série foi realizada com um *baudrate* de 9600 bits/s, 8 *bits* de dados, paridade par e 1 stop *bit*. O tamanho da mensagem a enviar e receber é de 9 *bytes* já explicada anteriormente. Realizada a parametrização do bloco, é ativado o mesmo e iniciada a comunicação. Esta começa com o envio da mensagem e posterior receção com um *timeout* de um segundo. Tanto o envio como a receção da mensagem são realizados através da verificação do estado dos respetivos *buffers* (se estão cheios, vazios e se existem dados). A comunicação termina após receber a mensagem ou atingir o tempo de *timeout*, ativando a variável *FimComm* e reiniciando o ciclo (figura 3.31 e figura 3.32).

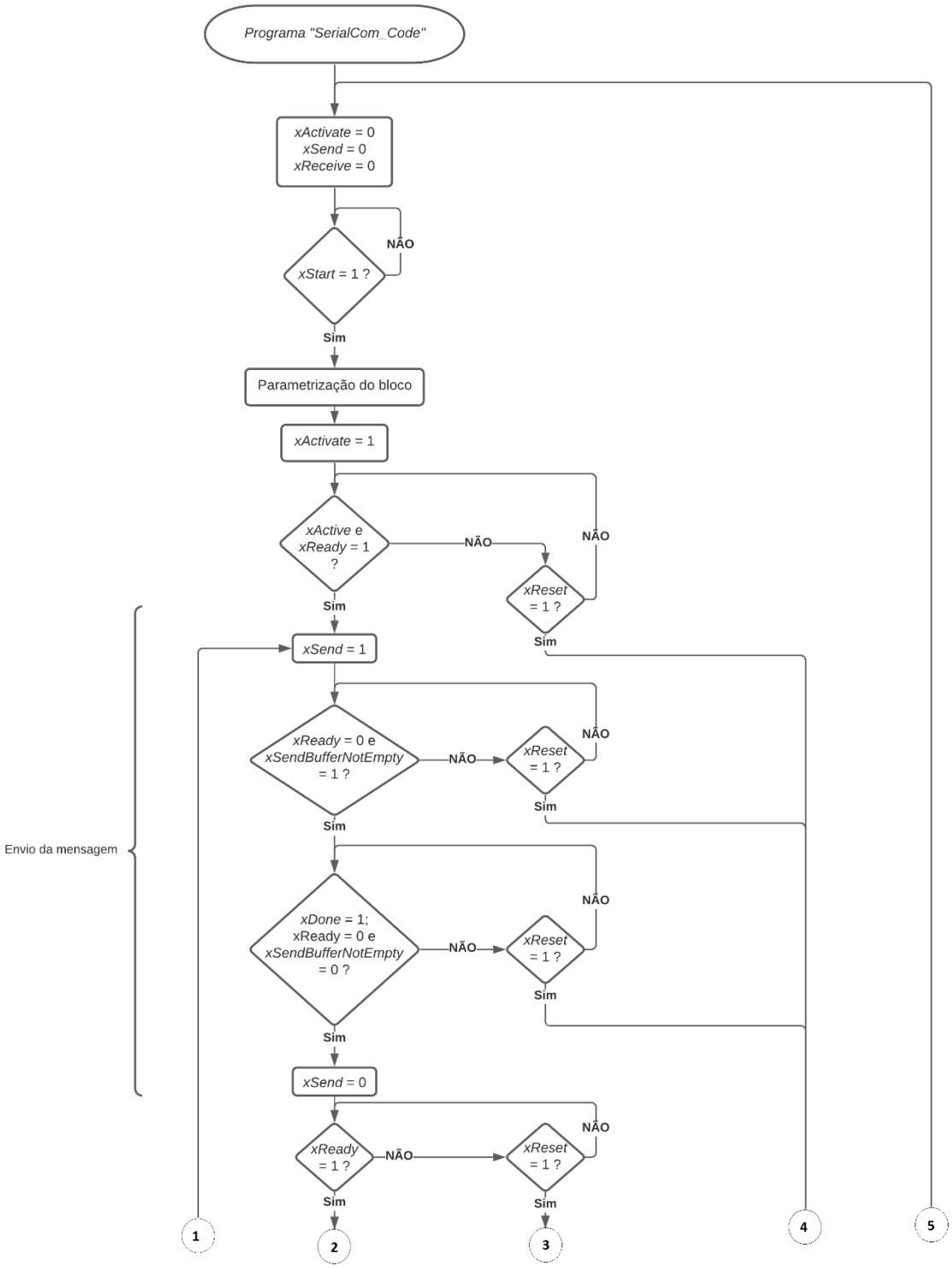


Figura 3.31 – Primeira parte do fluxograma representativo do programa de comunicação série "SerialCom_Code"

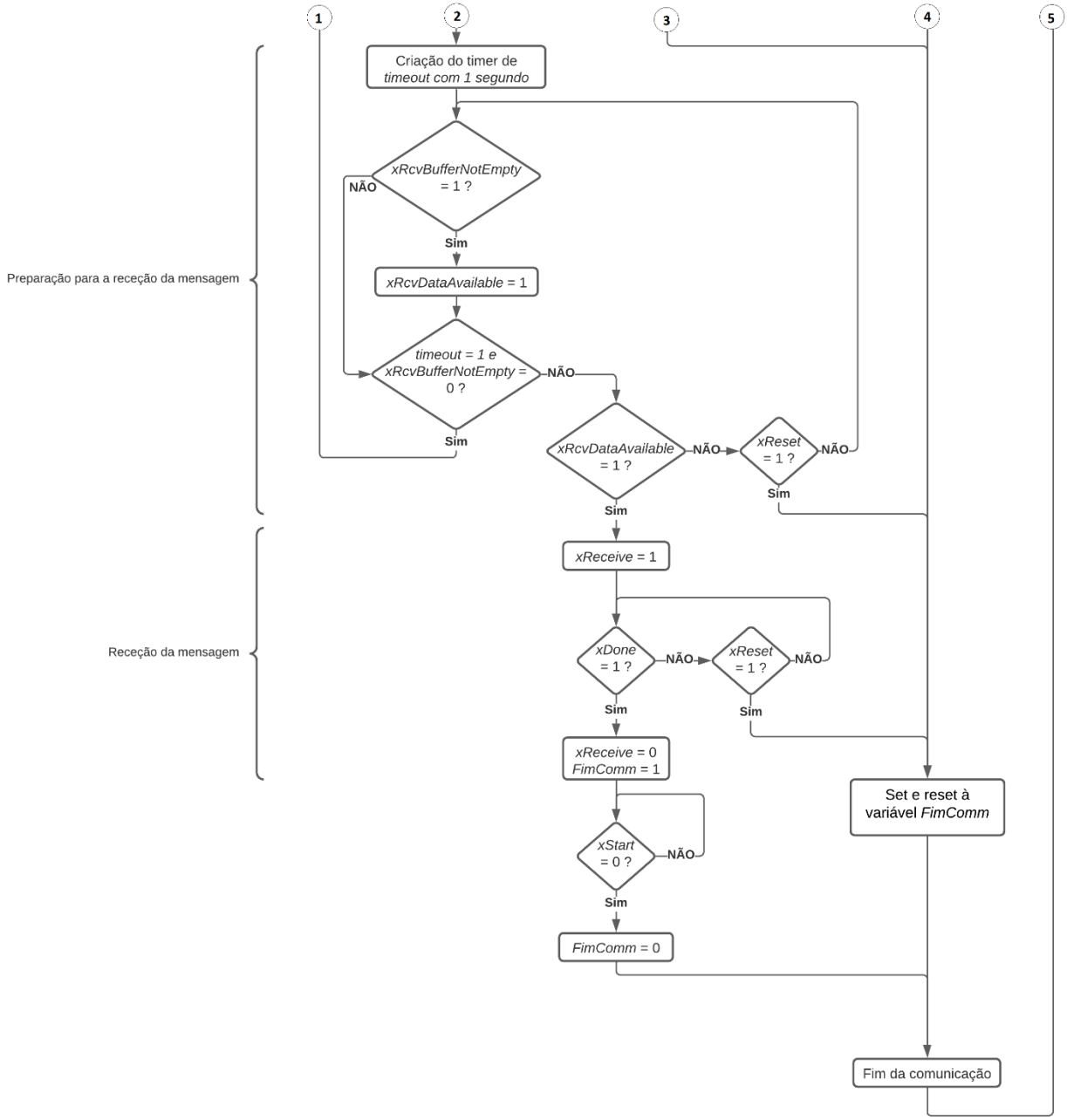


Figura 3.32 – Segunda parte do fluxograma representativo do programa de comunicação série "SerialCom_Code"

A mensagem é depois dividida e interligada com as variáveis globais do PLC no programa "SerialCom" (figura 3.33), de acordo com a estrutura já descrita na tabela 3.5.

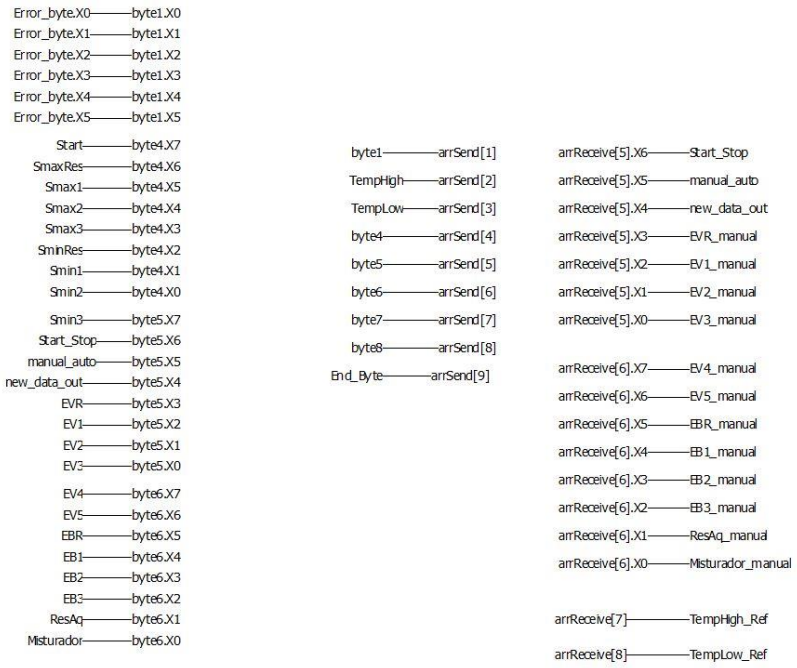


Figura 3.33 – Divisão da mensagem e interligação com variáveis globais do PLC

No programa "SerialCom" é ainda realizado o tratamento dos valores reais e de referência de temperatura (figura 3.34).

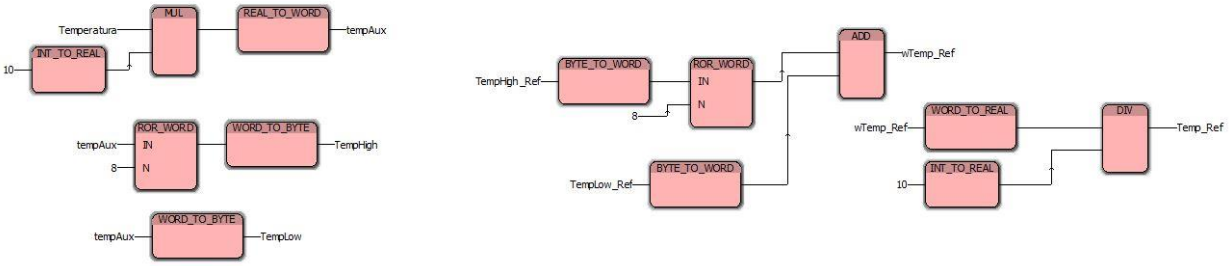


Figura 3.34 – Blocos lógicos para tratamento dos valores reais e de referência de temperatura

O seu tratamento é necessário pois a temperatura ocupa dois *bytes* separados na mensagem e o seu formato é de três algarismos significativos sem casa decimal. O valor de temperatura real obtido pelo sensor será enviado para a supervisão devendo ser multiplicado por 10 para corresponder ao formato esperado. É depois convertido numa *word* (2 *bytes*). Para o envio do valor, é necessário dividir a *word* em dois *bytes* – *TempHigh* e *TempLow* – que serão interligados com o respetivo elemento do *array* de envio. Na receção do valor de referência de temperatura é realizado o processo inverso. Os dois *bytes* recebidos são adicionados numa *word* que é convertida para um real e dividido por 10 para obter o valor em graus Celcius.

Após a criação e desenvolvimento dos programas, é necessário ligar as restantes variáveis aos módulos físicos do PLC. Para tal, na opção "*Process Data*" no menu "*View*" é possível escolher a opção "Default" para mostrar a lista de variáveis globais criadas. Arrastando a variável desejada para o campo do módulo correspondente é realizada a sua ligação. Na figura 3.35 e figura 3.36 está exemplificada a ligação entre uma variável global a uma variável analógica de entrada.

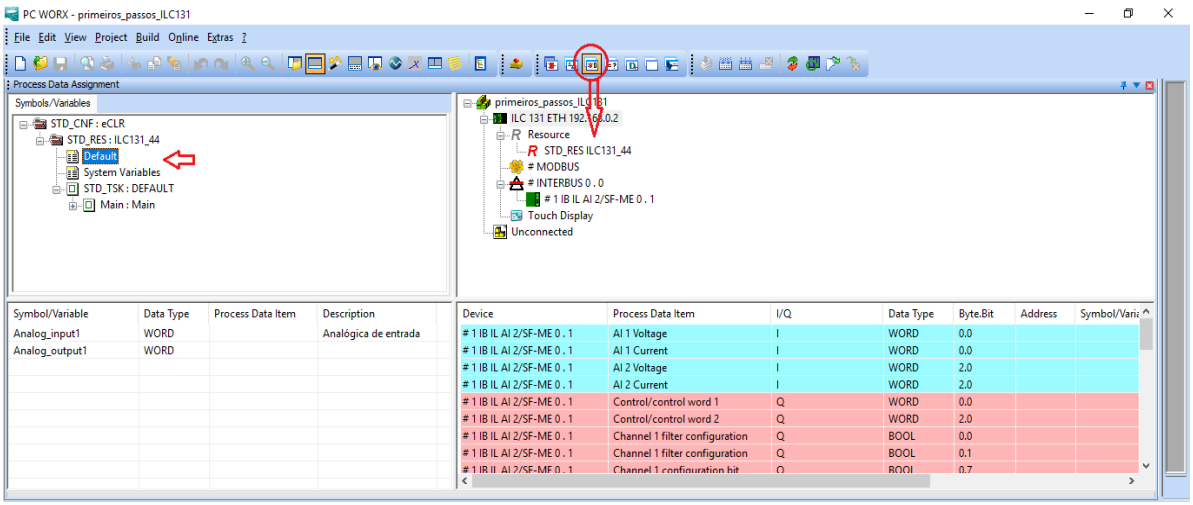


Figura 3.35 – Atribuição de uma variável global do programa a uma variável analógica de entrada

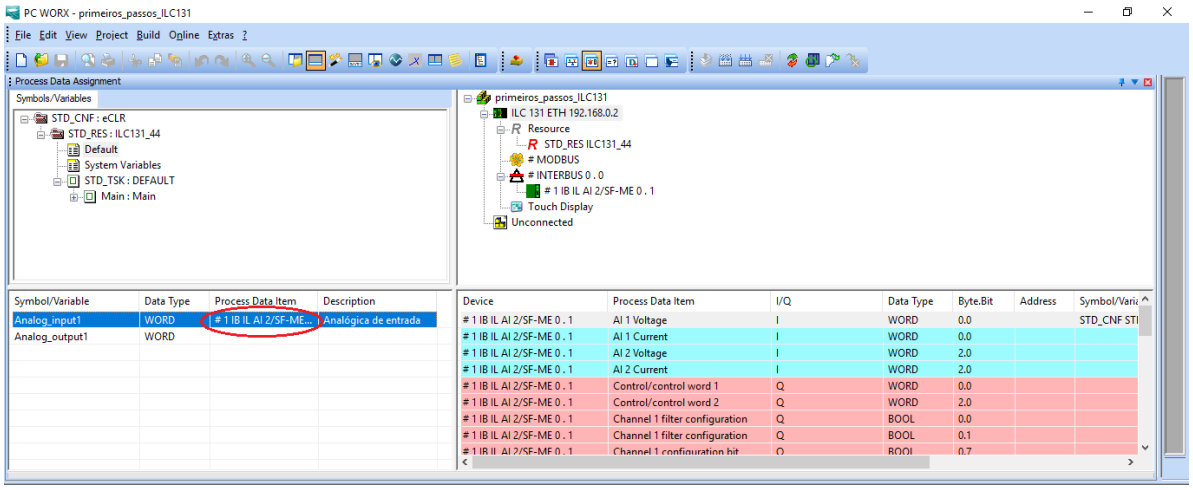


Figura 3.36 – Variável global do programa atribuída a uma variável analógica de entrada

Em conjunto com o PLC, foi ainda desenvolvido um sistema de supervisão composto por um *Raspberry Pi* e uma aplicação *Android*.

3.3.3 Desenvolvimento da supervisão no *Raspberry Pi*

No *Raspberry Pi* foi usado o *software* ERIC para desenvolver a interface gráfica de supervisão do simulador didático. ERIC é um ambiente de desenvolvimento integrado (IDE) para *python*, estando representado na figura 3.37.

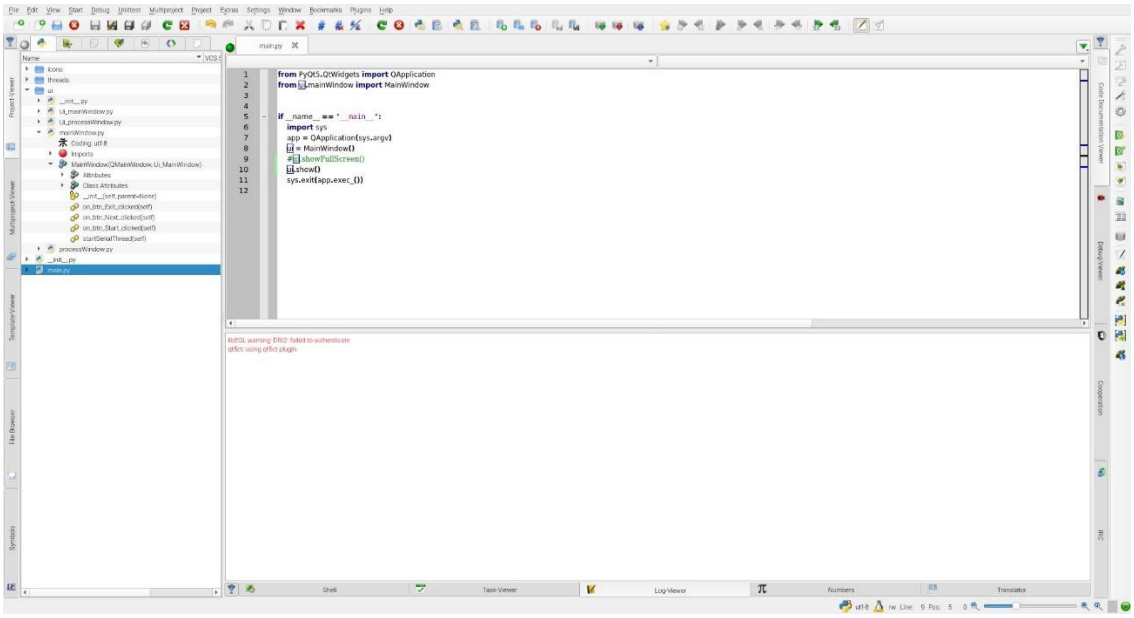


Figura 3.37 – Janela principal do software ERIC

Para o desenvolvimento da interface foram idealizadas quatro classes *python*: duas para a representação gráfica do simulador didático e outras duas para a comunicação com o PLC e com a aplicação *Android*, a correr em segundo plano. A partilha de variáveis entre as várias classes é realizada com recurso a *slots* e sinais. Esta é uma ferramenta usada pelo *PyQt* que permite a criação de sinais, uma espécie de *flag*, e a sua atribuição a um ou mais *slots* (normalmente funções). Para criar um sinal usa-se a igualdade *sinal=**pyqtSignal**(tipo de variável)* no início da classe e para o atribuir a um *slot* usa-se *sinal.**connect**(slot)*.

3.3.3.1 Interface gráfica da supervisão

Para a representação gráfica da supervisão do simulador tirou-se partido das ferramentas do *software* ERIC com o *Qt*, permitindo uma utilização por blocos intuitiva sem necessidade de escrever linhas de código para a sua criação, apenas para a sua funcionalidade. Os diversos elementos são arrastados (de modo semelhante a uma programação em Visual Basic) e arrumados na janela principal, sendo posteriormente gerado automaticamente o seu código em *python*. Com os elementos gráficos criados é apenas necessário escrever as linhas de código de função.

A interface está dividida graficamente em três janelas. A primeira serve de apresentação indicando o nome do projeto e logotipo do ISEL (figura 3.38). Avançando para a segunda janela (figura

3.39) é confirmado com o utilizador se existe água no reservatório, servindo de requisito para avançar para a terceira janela (figura 3.40). Caso essa condição não seja cumprida, é apresentada uma mensagem de erro. Na terceira janela é apresentado o processo, subdividido em separadores. Em todas as janelas existe um botão que permite ao utilizador sair da interface.

Figura 3.38 – Representação gráfica da primeira janela

Figura 3.39 – Representação gráfica da segunda janela (com e sem erro)

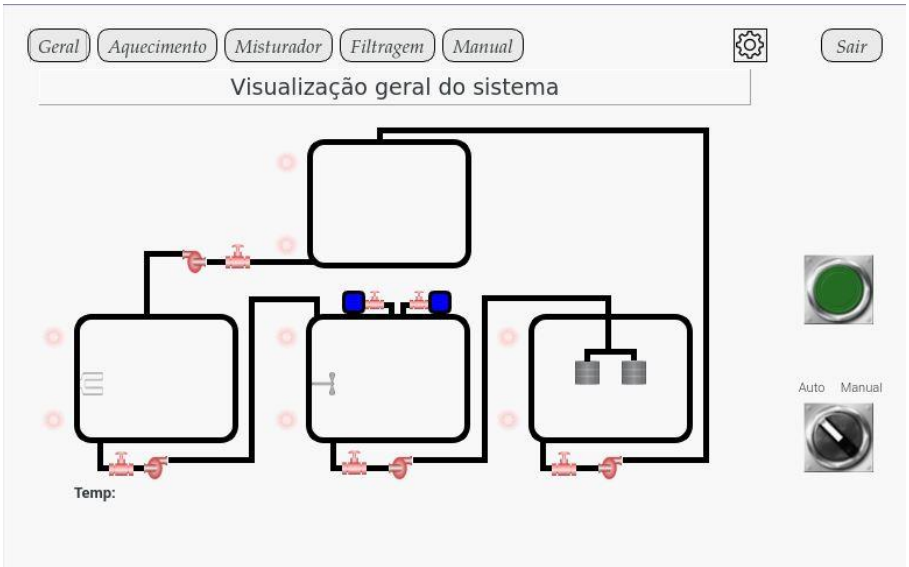


Figura 3.40 – Representação gráfica da terceira janela

A janela do processo está subdividida em cinco separadores – geral, aquecimento, mistura, filtragem, manual – representando o processo de diversos modos de operação. Em todos é possível ter acesso ao interruptor de início e paragem do processo e ao seletor do modo de funcionamento. No separador geral é possível ter uma representação global do processo. São apresentados todos os tanques e respetivas eletroválvulas, eletrobombas e níveis de água. É ainda apresentado o valor real da temperatura e os estados da resistência de aquecimento e bomba misturadora.

Nos restantes separadores "aquecimento", "mistura" e "filtragem" (figura 3.41 a figura 3.43) é apresentada uma representação da etapa correspondente ao seu nome, com o tanque e respetivas eletroválvulas, eletrobombas e níveis de água de acordo com os sensores. No separador de aquecimento é representado o estado da resistência de aquecimento e dada possibilidade ao utilizador de variar o valor de referência da temperatura. No separador "mistura" é apresentado também o estado da bomba misturadora.

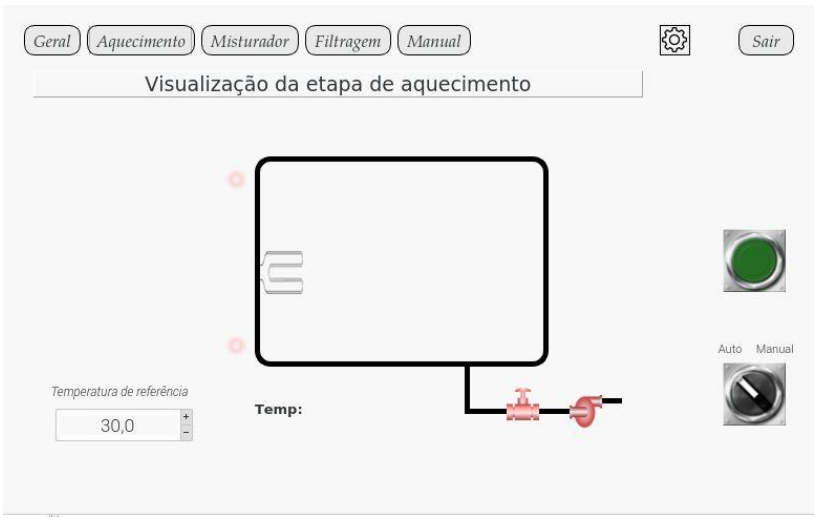


Figura 3.41 – Representação gráfica do separador "aquecimento"

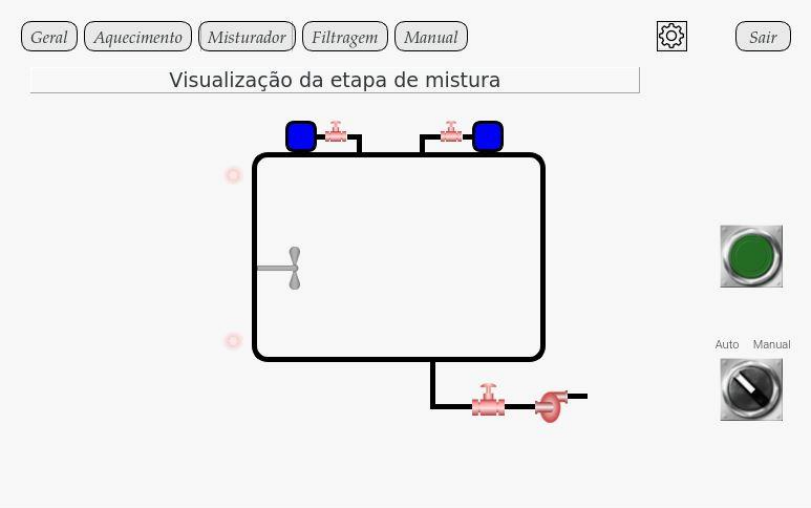


Figura 3.42 – Representação gráfica do separador "mistura"

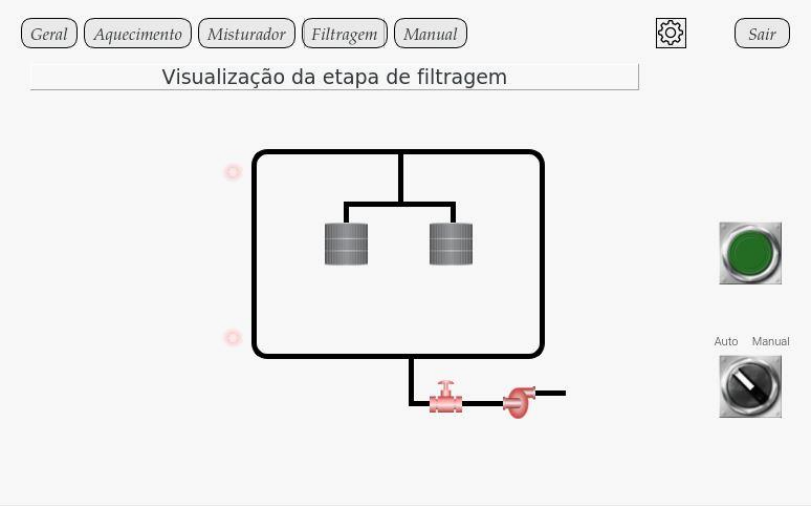


Figura 3.43 – Representação gráfica do separador "filtragem"

Finalmente, no separador "manual" (figura 3.44) estão representados botões para alteração manual dos estados dos atuadores do simulador: eletroválvulas, eletrobombas, resistência de aquecimento e bomba misturadora. Os botões só funcionam se o processo estiver a trabalhar em modo manual.

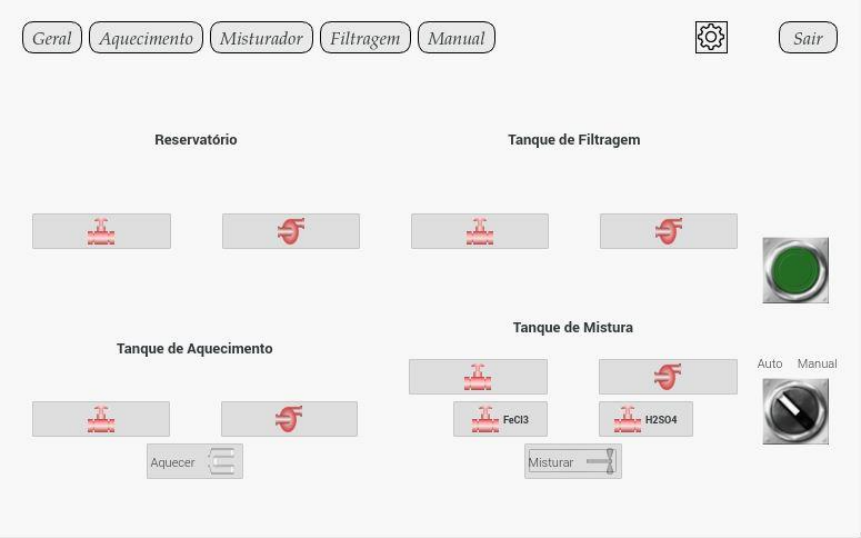


Figura 3.44 – Representação gráfica do separador "manual"

No canto superior direito, existe ainda um ícone de definições (⚙) onde, após selecionado, é possível ter acesso a um histórico de registos de eventos ou *logs* e algumas informações sobre a interface (figura 3.45).

Figura 3.45 – Representação gráfica dos logs da interface

3.3.3.2 Funcionalidade da interface de supervisão

O desenvolvimento da funcionalidade da interface está dividido em duas classes *python*: *MainWindow* e *ProcessWindow*. Na classe *MainWindow* serão desenvolvidas as primeiras duas janelas de apresentação e na *ProcessWindow* será desenvolvida a janela de representação do processo. Em ambas, foi tirado partido do *stacked widget* – um dos elementos gráficos *Qt* – que permite a criação de várias janelas num só elemento. Desta forma, é possível incluir diversas janelas num só espaço, diminuindo e otimizando o código funcional das mesmas em *python*. No decorrer da interação entre o

utilizador e a interface serão selecionados diversos botões que realizarão variadas funções. Quando um botão é selecionado despoleta uma função desenvolvida na respetiva classe.

Assim que a interface é inicializada, é corrido o código da classe *MainWindow*. Nesta são configuradas e iniciadas as *threads* de comunicação, explicadas no ponto seguinte, e atribuídos *slots* aos sinais criados. Após realizadas as configurações e inicializações, a interface fica à espera do sinal de cada botão, correndo o código da respetiva função assim que algum deles seja selecionado. Os botões associados a esta classe são três: (1) o de saída da interface, (2) para avanço para a segunda janela e (3) para avanço para a terceira janela (representação do processo). As funções cujo código é corrido aquando da seleção de cada botão estão descritas nos fluxogramas da figura 3.46.

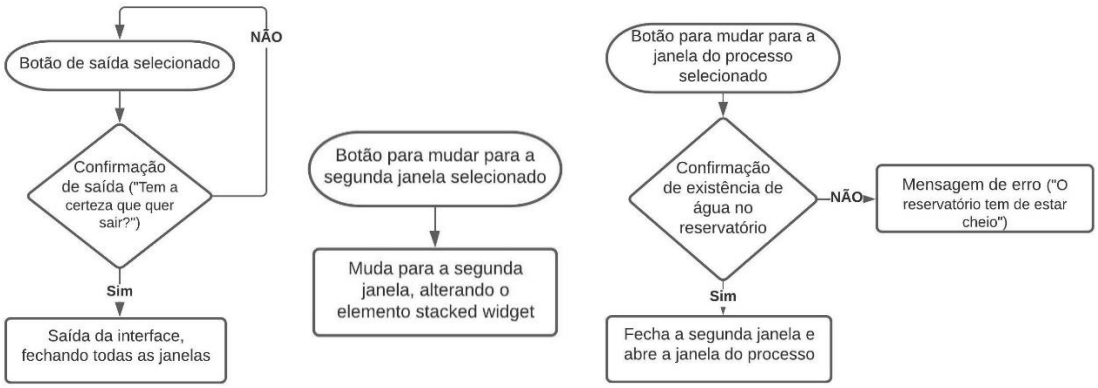


Figura 3.46 – Fluxogramas representativos das funções de cada botão

Quando o botão de saída da interface é selecionado, aparece uma pequena janela *pop-up* com a pergunta "Tem a certeza que quer sair?", para confirmar com o utilizador que realmente pretende sair da interface, dando-lhe a opção de confirmar ou cancelar. Se a resposta for afirmativa as janelas são fechadas, terminando e saindo da interface. Caso contrário, a interface retoma o seu estado anterior continuando o programa. Os botões de avanço funcionam de forma semelhante. Quando o botão para mudar para a segunda janela é selecionado, é simplesmente alternado o elemento do *stacked widget*, alterando a representação da primeira para a segunda janela. Quando o botão para mudar para a terceira janela (processo) é selecionado, é confirmada a existência de água no reservatório com o utilizador através de uma *checkbox*. Caso esta não esteja ativa é apresentada a mensagem de erro "O reservatório deve estar cheio" e a interface retoma o seu estado anterior, mantendo-se na segunda janela. Caso esteja ativa, a existência de água é confirmada, a janela atual é fechada e a janela do processo é aberta.

Quando a terceira janela (processo) é aberta, é corrido o código da classe *ProcessWindow*. Esta é encarregue de realizar o tratamento dos dados recebidos do PLC e de atualizar a representação gráfica do processo na interface, estando em constante contacto com a *thread* de comunicação série explicada mais à frente. Quando é inicializada, é lida uma primeira função que altera a cor dos elementos usados para a representação dos tanques. A função foi criada pois não era possível realizar esta alteração diretamente no elemento. De seguida são criadas duas variáveis globais, transversais a todas as funções

da classe, para a mensagem recebida e a enviar para o PLC, através da *thread* de comunicação série. Ambas são *array's* binários. A variável da mensagem recebida *process_data_updated* serve para atualizar a interface com os dados enviados pelo PLC e é inicializada com todos os elementos a 0. A variável *process_data_send* contém os dados alterados pela interface a enviar para o PLC e é inicializada com todos os elementos a "0", exceto o valor da temperatura de referência que tem o valor 300 (i.e. 30,0ºC). Realizada a criação das variáveis, o programa fica à espera dos respetivos sinais para chamar as restantes funções da classe. Foram desenvolvidas duas funções para tratamento dos dados do PLC e atualização da interface gráfica, uma para a listagem de erros, duas para o tratamento dos valores de temperatura real e de referência, duas para o tratamento dos *logs* do sistema, uma para o envio da mensagem alterada para o PLC e as restantes para resposta à seleção de botões. A descrição destas funções pode ser encontrada na tabela 3.13.

Tabela 3.13 – Descrição da tarefa de cada função da classe ProcessWindow e de como é chamada

Função	**Chamada**	**Tarefa**
widgets_initialParameters	Construtor da classe	Alteração da cor dos tanques
processDataUpdated	Quando existem dados do PLC recebidos pela SerialThread	Função que recebe os dados atualizados do PLC e chama a função uiUpdate, para atualizar a interface
uiUpdate	Função processDataUpdated	Função para atualizar a interface gráfica
erros	Função uiUpdate	Função com a listagem dos erros
temperatureValue	Função uiUpdate	Função para converter o valor do sensor de temperatura
on_btn_Geral_clicked	Botão "geral"	Função para mudar para o separador "geral"
on_btn_Aquecimento_clicked	Botão "aquecimento"	Função para mudar para o separador "aquecimento"
on_btn_Misturador_clicked	Botão "misturador"	Função para mudar para o separador "misturador"
on_btn_Filtragem_clicked	Botão "filtragem"	Função para mudar para o separador "filtragem"
on_btn_Start_clicked	Interruptor de start/stop	Função de Start/Stop pela interface
on_btn_Modo_clicked	Seletor do modo	Função para mudar o modo pela interface
on_btn_Sair_clicked	Botão "sair"	Função para sair do programa fechando a porta série e enviando uma mensagem com tudo a 0 para o PLC
on_btn_EVR_clicked	Botão da eletroválvula do reservatório	Função que liga a válvula do reservatório em modo manual
on_btn_EBR_clicked	Botão da eletrobomba do reservatório	Função que liga a bomba do reservatório em modo manual
on_btn_EV1_clicked	Botão da electroválvula do tanque de aquecimento	Função que liga a válvula do tanque de aquecimento em modo manual

Função	Chamada	Tarefa
on_btn_EB1_clicked	Botão da eletrobomba do tanque de aquecimento	Função que liga a bomba do tanque de aquecimento em modo manual
on_btn_EV2_clicked	Botão da eletroválvula do tanque de mistura	Função que liga a válvula do tanque de mistura em modo manual
on_btn_EB2_clicked	Botão da eletrobomba do tanque de mistura	Função que liga a bomba do tanque de mistura em modo manual
on_btn_EV3_clicked	Botão da eletroválvula do tanque de filtragem	Função que liga a válvula do tanque de filtragem em modo manual
on_btn_EB3_clicked	Botão da eletrobomba do tanque de filtragem	Função que liga a bomba do tanque de filtragem em modo manual
on_btn_EV5_clicked	Botão da eletroválvula de reagente do tanque de mistura	Função que liga a válvula dir. do tanque de mistura em modo manual
on_btn_EV4_clicked	Botão da eletroválvula de reagente do tanque de mistura	Função que liga a válvula esq. do tanque de mistura em modo manual
on_btn_ResAq_clicked	Botão da resistência de aquecimento	Função que liga a resistência de aquecimento em modo manual
on_btn_Mist_clicked	Botão da bomba misturadora	Função que liga a bomba misturadora em modo manual
on_tempRef_valueChanged	Variador do valor de temperatura (SpinBox)	Função que define o valor da temperatura de referência
processDataSend	Sempre que é alterado o estado de elementos do sistema pela interface	Função que envia a mensagem completa para a SerialThread para ser enviado para o PLC
on_btn_Manual_clicked	Botão “manual”	Função para mudar para o separador modo manual
on_btn_config_clicked	Botão com o ícone de definições	Função para mudar para o separador de configurações
on_btn_connectBt_clicked	Botão “connect”	Função para realizar a ligação Bluetooth
on_btn_cancelBt_clicked	Botão “cancel”	Função para cancelar a ligação Bluetooth
bluetoothLogs	Sempre que existe algum erro ou informação	Função que apresenta os logs relacionados com a ligação Bluetooth
programLogs	Sempre que existe algum erro ou informação	Função que apresenta os logs relacionados com o programa

A maior parte das funções é bastante direta, executando uma simples tarefa quando é chamada como é o caso das funções de alteração entre separadores. As restantes serão explicadas um pouco mais em detalhe para facilitar a sua compreensão. As funções de controlo do sistema pela interface são chamadas quando o respetivo botão/seletor/interruptor é selecionado, sendo realizada uma verificação do estado do mesmo, atualizado o *bit* correspondente na variável *process_data_send* e chamada a função

processDataSend para envio da mensagem para a *thread* de comunicação série *SerialThread* e posteriormente para o PLC, tal como mostra o fluxograma seguinte.

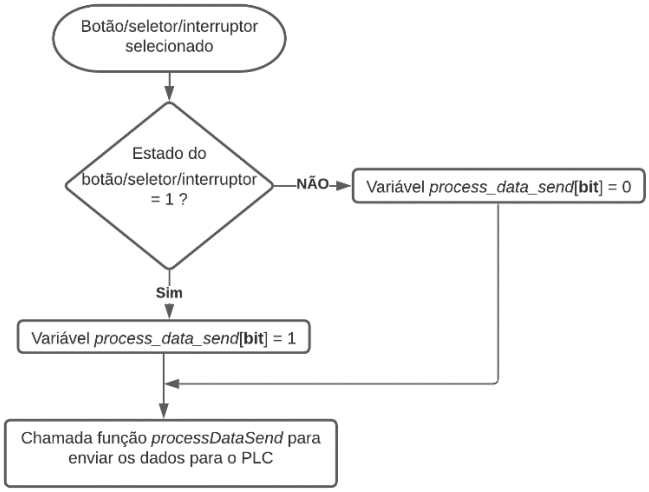


Figura 3.47 – Fluxograma representativo das funções de controlo do sistema pela interface

As funções *processDataUpdated* e *uiUpdate* foram criadas para atualização dos dados do PLC recebidos e da interface gráfica. A primeira é chamada sempre que os dados são recebidos via comunicação série e lidos pela *SerialThread*, sendo copiados e gravados da variável *process_data_updated*. É posteriormente emitido o sinal que despoleta a função *uiUpdate*. A função *uiUpdate*, por sua vez, realiza a divisão da mensagem binária nas diversas variáveis que representam cada elemento, tal como explicado na tabela 3.5, à exceção da listagem de erros e valores de temperatura que serão iniciados, mas tratados noutras funções. Após a divisão da mensagem, é realizada a verificação dos estados de cada sensor de nível, eletroválvulas, eletrobombas, resistência de aquecimento e bomba misturadora para representação na interface, alterando a sua cor e preenchimento. Foi assumido o verde para representar o estado ligado e o vermelho para desligado. No caso da resistência de aquecimento e bomba misturadora o estado desligado é representado a cinzento e o estado de ligado a laranja e verde, respetivamente.

Na função *erros*, é criada uma variável do tipo dicionário, onde são correspondidas duas colunas: uma com algarismos decimais a começar em zero e outra com mensagens de erro ou informação. A função começa por ler o valor binário dos bits correspondentes na mensagem e converte-o para um valor decimal. Esse valor é depois comparado com os algarismos do dicionário, representando a mensagem de erro ou informação correspondente. Essa mensagem é apresentada no canto inferior esquerdo da janela do processo, quando chamada pela função *uiUpdate*. Na tabela 3.14 está descrita a listagem de erros implementados, podendo ir até 63.

Tabela 3.14 – Descrição do dicionário com a listagem de mensagens de erro

Algarismo decimal	**Mensagem de erro/informação**
0	(Campo vazio, sem mensagem de erro)
1	Programa parado
2	Programa em modo manual
3	Programa em modo automático
4	Reservatório vazio
5	Avaria nos sensores do reservatório
6	Avaria nos sensores do tanque de aquecimento
7	Avaria nos sensores do tanque de mistura
8	Avaria nos sensores do tanque de filtragem

À semelhança da função anterior, a função *temperatureValue* lê o valor binário dos *bits* correspondentes na mensagem e converte-o para um valor decimal. Esse valor é representado por três algarismos, sendo necessário realizar a operação matemática de divisão por 10 para obter o valor de temperatura com uma casa decimal (ex.: 359 = 35,9ºC). O valor de temperatura é depois guardado na sua variável, quando chamado pela função *uiUpdate*.

No caso do valor de referência de temperatura é realizada a operação inversa. A função *on_tempRef_valueChanged* é chamada quando o valor é alterado, pelo utilizador, no elemento gráfico *SpinBox* e multiplicado por 10 para obter os três algarismos significativos sem casa decimal. São depois realizados um conjunto de operações para converter o valor decimal em binário. De notar que o valor de temperatura com três algarismos significativos e sem casa decimal, em binário, ocupa no máximo 12 dos 16 *bits* reservados. Como tal, foi necessário adicionar quatro bits a 0, sendo estes os mais significativos (i.e. colocados à esquerda). Estando completo, é adicionado o valor binário à variável *process_data_send* e chamada a função *processDataSend* para envio da mensagem para a *thread* de comunicação série *SerialThread* e posteriormente para o PLC.

As funções *bluetoothLogs* e *programLogs* recebem todos os *logs* do simulador didático e apresentam-nas na janela de definições, adicionando-lhes a data e hora do registo. No caso das funções *on_btn_connectBt_clicked* e *on_btn_cancelBt_clicked*, é enviado o mesmo sinal com o seu valor a "1" ou "0", para realizar ou cancelar a ligação *Bluetooth*, respetivamente. A ligação é realizada automaticamente quando a *thread* de comunicação é iniciada, no entanto são disponibilizados ao utilizador, no separador de *logs Bluetooth* na página de definições, dois botões para a realizar manualmente.

3.3.3.2 Comunicação com o PLC e aplicação *Android*

Para o desenvolvimento da plataforma de comunicação foram criadas duas classes do tipo objeto, em *python*. Um dos principais impedimentos no seu adequado desenvolvimento foi a capacidade do *Raspberry Pi* para responder à interação do utilizador com a interface e processar pedidos de comunicação com o PLC e a aplicação *Android*, ao mesmo tempo. Para ultrapassar esta dificuldade, foi utilizado o conceito de *thread* na implementação das classes. Com uma *thread*, é possível dividir o programa para que realize várias tarefas ao mesmo tempo. As classes do tipo objeto criadas são a *SerialThread* para tratar da comunicação série com o PLC e a *BluetoothThread* para tratar da comunicação *Bluetooth* com a aplicação *android*. Como já foi mencionado, ambas são adicionadas à sua *thread* na classe *MainWindow*, pelo que a partir deste ponto, serão sempre referidas pelo seu nome ou pelo termo *thread* e não classe.

As *threads* têm um funcionamento similar, divergindo apenas na sua implementação. Começam por abrir um *socket* ou porta para estabelecer a sua comunicação, ficando depois a aguardar pelos dados para enviar ou receber. A *SerialThread* está dividida em sete funções: uma para criar e abrir a porta série, uma para fechar, duas para enviar os dados para o PLC, duas para receber os dados do PLC e uma para realizar a conversão dos dados a enviar. No caso da *BluetoothTread*, esta está dividida em cinco funções: duas para criar e abrir o *socket* de comunicação, uma para fechar, uma para enviar e receber dados e uma para a interpretação dos dados recebidos. Como a interface e a aplicação trabalham em paralelo representando a supervisão, o fluxo dos dados foi centralizado na *SerialThread* para envio para o PLC. O seu diagrama pode ser consultado na figura 3.48.

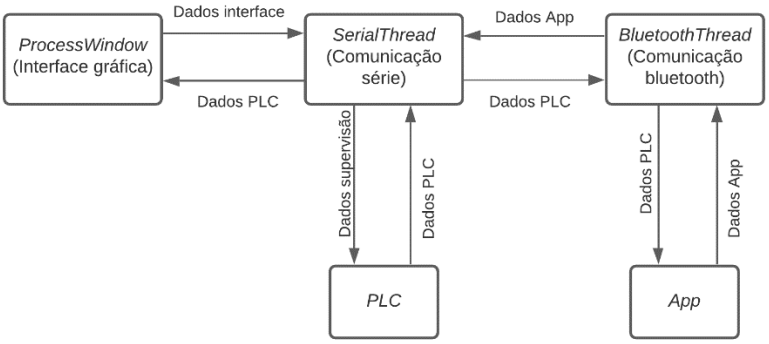


Figura 3.48 – Fluxo dos dados no simulador didático

Para a *thread* de comunicação série foi utilizada uma das várias bibliotecas disponibilizadas pelo *PyQt* para o seu desenvolvimento. Quando é iniciada, é criada uma variável global para armazenamento dos dados recebidos do PLC e atribuídos *slots* aos sinais do *buffer* e função de receção de dados. É ainda criada, parametrizada e aberta a porta série. No caso de não ser possível abrir a porta série é emitindo um erro e volta a tentar passados cinco segundos. Estabelecida a ligação, a *thread* fica

a aguardar os dados para enviar ou receber, chamando as funções *dataToSend* e *send_serialData* ou *receive_serialData* e *rcvDataBuffering*, respetivamente.

O envio dos dados da interface para o PLC é iniciado pela classe *ProcessWindow* quando é chamada a função *processDataSend*, que despoleta a *dataToSend*. Nesta, é realizada uma cópia da mensagem numa variável e enviada para a função *send_serialData* que a escreve na porta série com o comando *porta.**write**(dados)*. A mensagem, antes de ser escrita na porta série, tem de ser primeiramente convertida para que seja corretamente interpretada pelo PLC. A descrição geral do envio da mensagem está representada na figura 3.49.

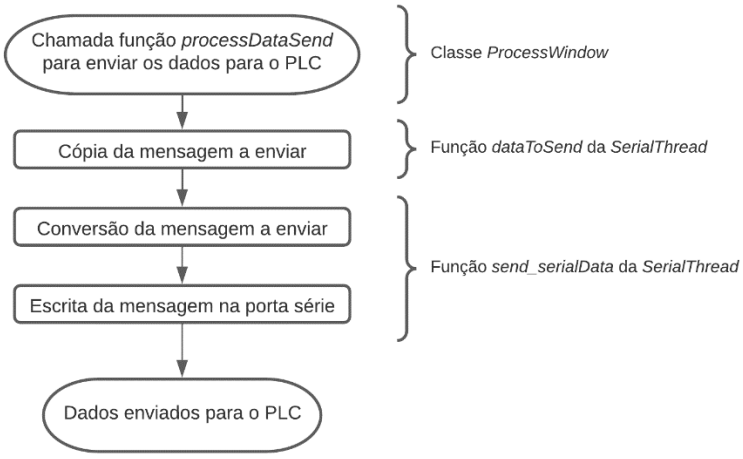


Figura 3.49 – Descrição do envio da mensagem para o PLC pela SerialThread

A receção dos dados é um pouco mais complexa pois a biblioteca *PyQt* para comunicação série usada não define o início e fim da mensagem, pelo que a *flag* de existência de dados na porta série por vezes é despoletada antes do fim da mensagem. Para confirmar a autenticidade da mesma, após confirmação da *flag* de existência de dados, é realizada a leitura da porta série na sua totalidade e gravada a mensagem no *buffer* de receção, até ter o tamanho correto. Os dados do *buffer* são depois tratados até obter um *array* binário para enviar para a *ProcessWindow* e atualizar a interface gráfica. No fim é realizada uma limpeza ao *buffer* para iniciar um novo ciclo. O fluxograma seguinte ilustra este processo.

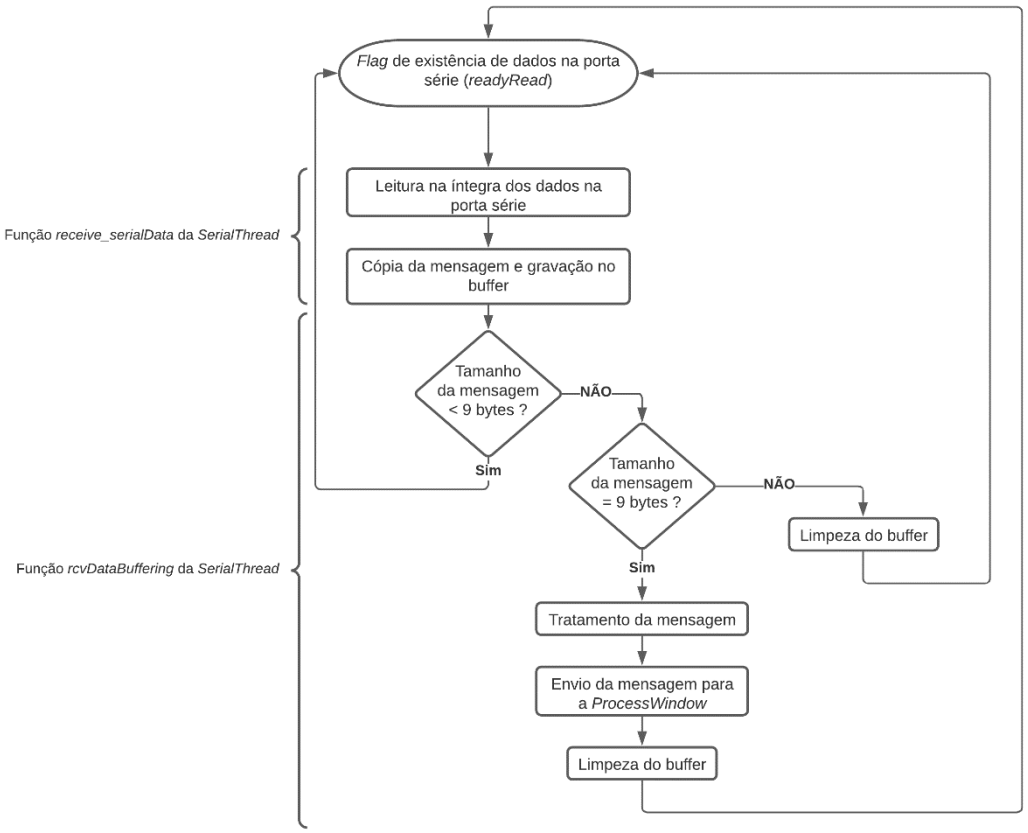


Figura 3.50 – Descrição da receção da mensagem do PLC pela SerialThread

Com a *thread* de comunicação *Bluetooth* a abordagem foi ligeiramente diferente, tendo sido usada uma biblioteca *python* para o efeito. Assim que é inicializada, é criado o *socket Bluetooth* e associado a uma porta, entrando depois num *loop* de sucessivas tentativas de envio e receção da mensagem. Após vários testes, foi verificado que a única porta à qual seria possível associar o *socket* para comunicação *Bluetooth* é a porta 1. Por isso, aquando da criação do *socket*, foi necessário realizar uma confirmação do número da porta e repetir o processo até ser a correta. Terminado o processo de criação do *socket* e associação à porta 1, a *BluetoothThread* fica a aguardar a ligação por parte do telemóvel com a aplicação. Recebido o pedido de ligação, esta é aceite e estabelecida, estando apto a receber e enviar mensagens. Sempre que algo não corre como esperado, ocorre algum erro ou a aplicação fecha, é encerrada a ligação *Bluetooth* e eliminado o *socket*, sendo necessário estabelecer nova ligação através dos botões explicados no fim do ponto anterior.

O envio e receção da mensagem *Bluetooth* é realizado na função *bluetoothDataToSend* que é chamada assim que a ligação *Bluetooth* é estabelecida e aceite. A função começa por enviar uma primeira mensagem com todos os elementos a "0" para a aplicação, ficando a aguardar a mensagem de resposta durante 500ms. Ao fim desse tempo, tenta enviar uma nova mensagem para a aplicação. O ciclo é repetido indefinidamente até que a ligação *Bluetooth* seja encerrada. A mensagem a enviar para a aplicação *android* é a mesma que atualiza a interface gráfica com os dados do PLC e a mensagem

recebida da aplicação é enviada para a *SerialThread* para envio para o PLC. No caso da comunicação *Bluetooth* teve de ser ainda desenvolvida uma função para interpretação da mensagem da aplicação. Nesta é atribuído o valor binário correspondente na mensagem, como descrito no ponto 3.2.3.2. O fluxograma seguinte ilustra a *thread* desenvolvida para comunicação *Bluetooth*.

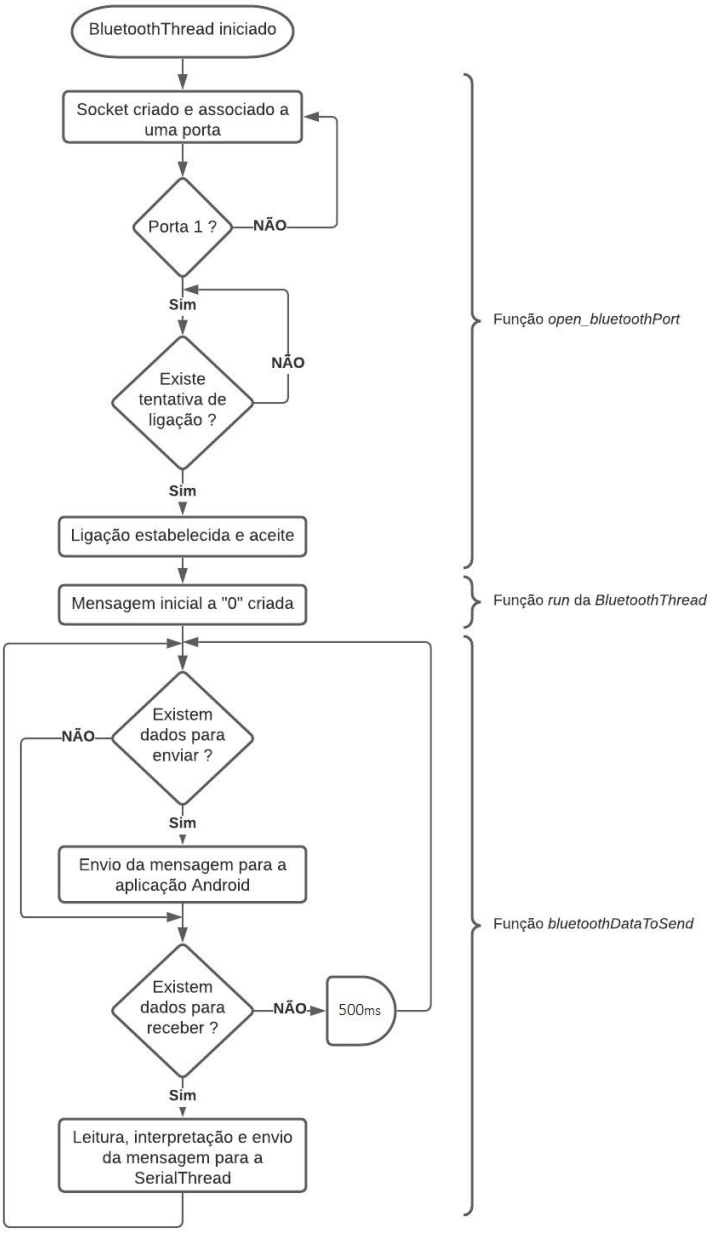


Figura 3.51 – Fluxograma representativo da thread desenvolvida para comunicação Bluetooth

3.3.4 Desenvolvimento da supervisão na aplicação *Android*

A aplicação *Android* foi desenvolvida através da plataforma online *MIT App Inventor* desenvolvida pelo *Massachusetts Institute of Technology* (MIT). A plataforma é um ambiente de programação visual intuitivo desenhado para que qualquer pessoa possa criar aplicações *Android* ou *iOS* para telemóveis e tablets. Nas figuras seguintes estão representadas as duas janelas de desenvolvimento

da aplicação. Na janela “designer” (figura 3.52) são inseridos diversos elementos gráficos como botões e caixas de texto ou blocos de conectividade e gravação como *Bluetooth* e base de dados. Na janela “blocks” (figura 3.53) é realizada a programação da aplicação, recorrendo aos blocos pré-definidos e disponibilizados pela plataforma.

Figura 3.52 – Janela de desenvolvimento gráfico da aplicação

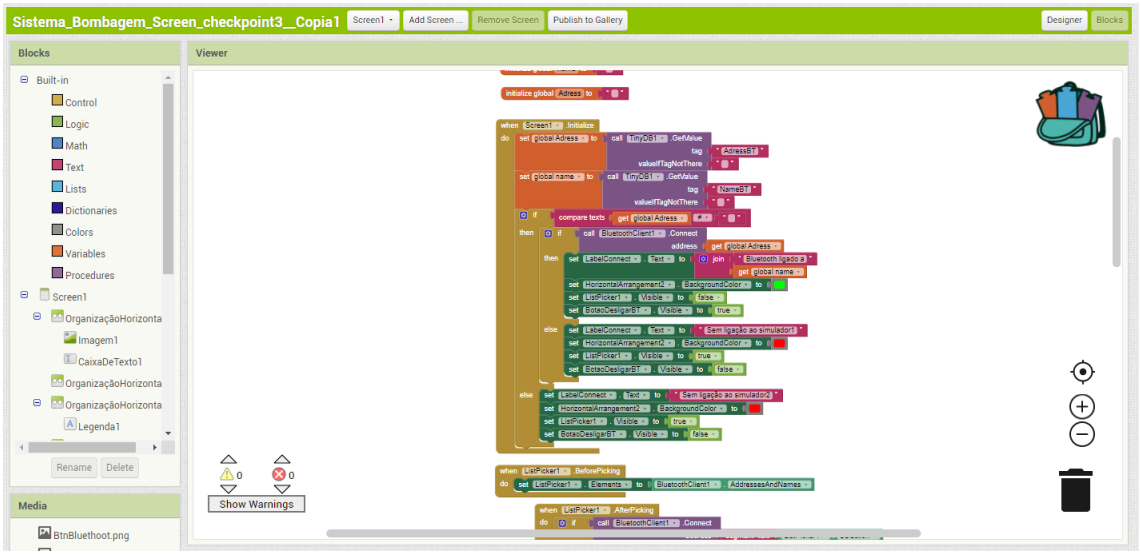


Figura 3.53 – Janela de desenvolvimento da programação da aplicação

3.3.4.1 Interface gráfica da aplicação *Android*

Para o desenvolvimento da aplicação foram desenhadas 5 janelas cada uma com a sua funcionalidade:

- Janela inicial;
- Menu;
- Modo Manual;
- Sinótico;
- Configuração.

A janela inicial é composta pelo logótipo do ISEL, juntamente com o nome do departamento e título da aplicação. Existem ainda uma caixa de texto indicativa do estado da ligação e dois botões: um para avançar para a página de menu e outro para realizar a ligação *Bluetooth*. A janela de menu é composta por seis botões que permitem enviar comandos para o *Raspberry Pi* e abrir novas janelas. O botão "auto", "start" e "stop" enviam os comandos de modo automático, de *start* e de *stop*, respetivamente. O botão "manual" envia o comando de modo manual e abre a respetiva janela e os botões "processo" e "conf" abrem as janelas de sinótico e configuração. É ainda possível voltar à janela inicial, atualizar a ligação *Bluetooth* e aceder a uma página com informações do simulador. Na figura 3.54 é possível pré-visualizar a representação das janelas principal e de menu da aplicação.

Figura 3.54 – Representação da janela principal e de menu da aplicação Android

A janela de modo manual é composta por diversos seletores que permitem ao utilizador ligar e desligar cada eletroválvula e eletrobomba do simulador, assim como a resistência de aquecimento e a bomba misturadora. Cada seleção corresponde a um comando que será enviado para o *Raspberry Pi*. A janela de configuração é composta por diversos seletores e três botões: para enviar os valores, para voltar à janela de menu e para atualizar a ligação *Bluetooth*. Nesta janela é possível selecionar o valor de referência da temperatura de aquecimento e os tempos de enchimento intermédio e final de cada tanque. Ambas as janelas têm uma caixa de texto indicativa do estado da ligação e estão representadas na figura 3.55.

Figura 3.55 – Representação da janela de modo manual e configuração da aplicação Android

Finalmente, a janela de sinótico é composta por uma caixa de texto indicativa do estado da ligação, um botão para voltar à janela de menu e outro para atualizar a ligação *Bluetooth* e um conjunto de imagens representativas da estrutura do simulador. Nela encontram-se os estados de todos os sensores e atuadores, à semelhança da interface gráfica do *Raspberry Pi*, sendo possível monitorizar todo o sistema em tempo real. A figura 3.56 representa a janela de sinótico.

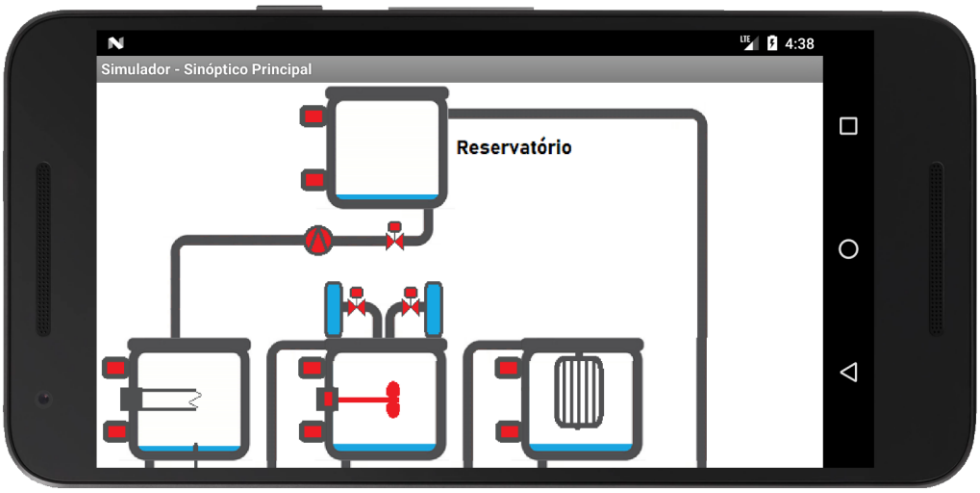


Figura 3.56 – Representação da janela de sinótico da aplicação Android

3.3.4.2 Programação da aplicação *Android*

Para que a aplicação seja funcional, cada janela tem associada uma página de programação desenvolvida com recurso a blocos tal como foi apresentado na figura 3.53. A programação de cada janela é bastante semelhante entre si e à programação realizada no *Raspberry Pi*, pelo que a explicação seguinte será breve. A comunicação e a estrutura da mensagem usada já foram explicadas nos pontos 3.2.3.2 e 2.1.1.1 pelo que não serão abordadas.

Para realizar a programação de cada janela, a plataforma online *MIT App Inventor* disponibiliza blocos pré-definidos para diversas funcionalidades. Podem ser gerais como ciclos *if* e *while* (figura 3.57) ou mais específicas como a comunicação *Bluetooth*. Os blocos são escolhidos do separador "*blocks*" e apresentados no separador "*viewer*", bastando arrastar o bloco pretendido para o espaço em branco.

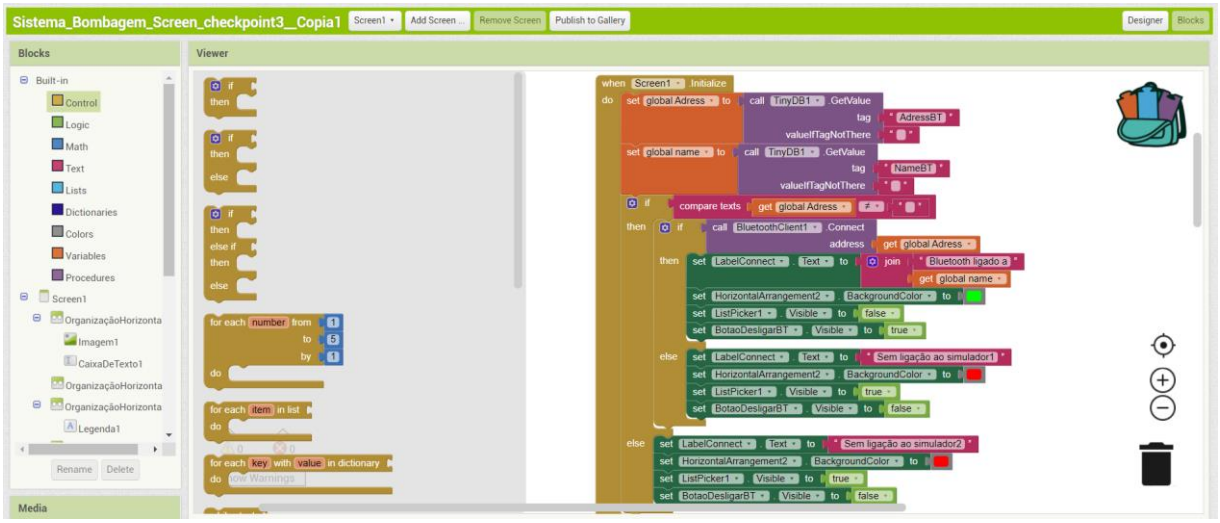


Figura 3.57 – Exemplo de blocos pré-definidos para a programação de cada janela

Sempre que uma janela é inicializada é realizada uma certa tarefa utilizando o bloco "*when screen.initialize*", tal como representado na figura 3.58, ficando depois a aguardar uma interação com a aplicação. Todas as funções criadas são despoletadas pela seleção de um botão ou seletor à exceção das funções da janela de sinótico – PreencheDados e RefreshSinoptico – onde é realizada a aquisição e divisão dos dados recebidos e atualizada a imagem do sinótico, respetivamente e das funções da janela configuração – LerDadosBt, SendData, DataRefTemp e DataTimes – onde são recebidos e enviados os dados e manipulados os valores da temperatura de referência e dos tempos de enchimento dos tanques.

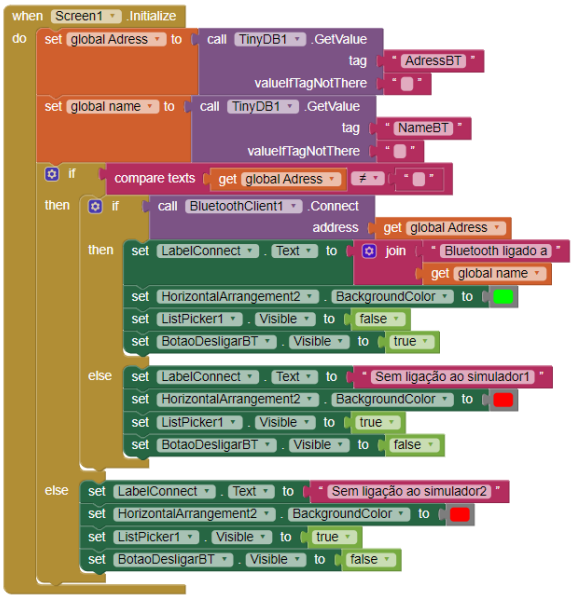


Figura 3.58 – Representação do bloco de janela inicializada

Aquando da inicialização da página principal é realizada uma verificação da conectividade *Bluetooth* e da lista dos equipamentos previamente emparelhados, permitindo ao utilizador a escolha do equipamento ao qual se quer ligar. Realizada a conexão *Bluetooth*, é gravado o nome e endereço do equipamento para, quando a aplicação voltar a ser inicializada, não ser necessário repetir o processo e é apresentada uma mensagem de sucesso no ecrã. Caso a conexão não seja realizada com sucesso, é apresentada uma mensagem de erro. Devido a uma limitação imposta pela plataforma online *MIT App Inventor*, não é possível manter a ligação *Bluetooth* ativa entre as janelas e, por isso, esta verificação é realizada sempre que uma janela é inicializada ou manualmente pelo utilizador através de um botão visível em cada janela. Terminada a verificação, a aplicação fica a aguardar a interação do utilizador com os botões ou seletores, realizando depois as tarefas já mencionadas em 3.3.4.1.

Capítulo 4

Resultados Experimentais

Resumo: No presente capítulo é apresentado um resumo do trabalho realizado e alguns resultados experimentais do funcionamento do simulador.

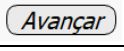
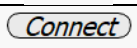
4.1 Resumo do trabalho realizado

O trabalho realizado consistiu na configuração e integração de um PLC, um *Raspberry Pi* e uma aplicação *Android*. O seu funcionamento e desenvolvimento encontra-se descrito em detalhe no capítulo anterior, no entanto, apresentam-se de seguida duas tabelas descritivas com o resumo de comandos do simulador (tabela 4.1) e a descrição dos diversos botões da supervisão do simulador (tabela 4.2).

Tabela 4.1 – Resumo de comandos do simulador

Ação	Comando
Ligar o simulador	Alimentar pela rede de baixa tensão e aguardar que o botão do painel apague o led e que a interface inicie
Iniciar o processo	Carregar no botão do painel em conjunto com o botão de *start* da interface ou enviando o comando pela aplicação *Android*
Parar o processo	Reverter os comandos realizados para iniciar o processo
Mudar o modo de funcionamento	Alterar o estado do seletor de modo da interface ou enviar o comando pela aplicação *Android* (figura 3.54)
Alterar o valor de referência da temperatura	Alterar o valor pela interface no separador "Aquecimento" (figura 3.41) ou pela aplicação *Android* (figura 3.55)
Controlar o processo em modo manual	Alterar o modo de funcionamento para manual e enviar comandos pela interface no separador "Manual" (figura 3.44) ou pela aplicação *Android* (figura 3.55)
Analisar funcionamento do simulador e/ou processo	Aceder à janela de configuração da interface e analisar o histórico apresentado nos separadores "Bluetooth" e "Logs"
Sair do processo	Parar o processo e carregar no botão "Sair" no canto superior direito da interface
Desligar simulador	Sair do processo e desligar a alimentação do simulador

Tabela 4.2 – Resumo dos botões da supervisão do simulador

Botão Interface	Botão app Android	Descrição
		Início e paragem do processo
		Modo de funcionamento
	---	Configurações
		Avançar
---		Voltar
	---	Sair
		Iniciar *Bluetooth*

4.2 Resultados do funcionamento do simulador

Neste ponto são apresentados alguns resultados experimentais do funcionamento do simulador. Para tal, serão apresentados exemplos de diversas situações representados em imagens e acompanhados de uma breve descrição. No entanto, o resultado experimental fundamental a retirar da solução desenvolvida é o bom funcionamento do simulador didático. O simulador será utilizado em

apresentações do departamento e estará disponível no laboratório de automação João Palma, localizado no edifício E, no ISEL.

Para demonstrar o bom funcionamento do simulador didático, foram escolhidos três exemplos diferentes:

- Mensagem entre o PLC e o *Raspberry Pi*, confirmado com recurso a um osciloscópio;
- Funcionamento geral com diferentes níveis de água nos tanques;
- Situação de falha nos sensores de nível.

4.2.1 Mensagem entre o PLC e o *Raspberry Pi*

Na figura 4.2 é possível observar a representação de um exemplo de mensagem do PLC para o *Raspberry Pi* no osciloscópio. Nela são enviados os seguintes comandos: mensagem de erro "1" indicando que o programa está parado; temperatura real de 16,3ºC; sensores máximo e mínimo do reservatório e sensor mínimo do tanque de aquecimento ativos e todos os outros desativados; modo automático; indicador de *start* físico e pela supervisão a "0"; temperatura de referência não selecionada (com o valor de 0). A mensagem pode ser visualizada na figura 4.1.

Figura 4.1 – Mensagem escolhida entre o PLC e o Raspberry Pi

Tal como já foi mencionado, a comunicação série foi realizada com um *baudrate* de 9600 bits/s, 8 *bits* de dados, paridade par e 1 stop *bit*. Cada *byte* de dados é enviado precedido de um start *bit* a "0" e seguido do *bit* de paridade e do stop *bit* a "1", começando do *bit* menos significativo. Com uma transmissão de 9600 *bits* por segundo, significa que cada *bit* é transmitido em 0,1ms o que corresponde a um décimo de cada quadricula do osciloscópio. Nesse caso, os dois primeiros *bytes* seriam: **0**1000000**110**0000000**01**, com cada *byte* de dados sublinhado (figura 4.3). A restante representação da mensagem é possível ser observada e comprovada na figura 4.2, à exceção do *byte* auxiliar de fim de mensagem com os seus *bits* sempre a "1".

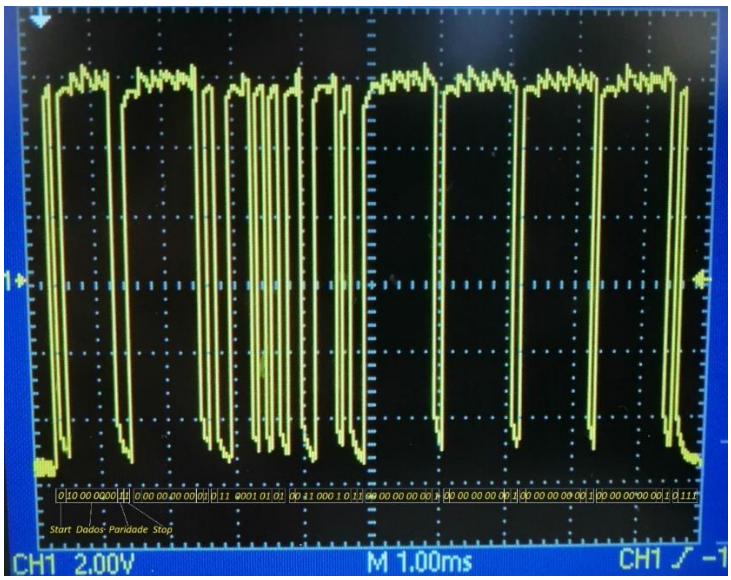


Figura 4.2 – Representação da mensagem do PLC para o Raspberry Pi no osciloscópio

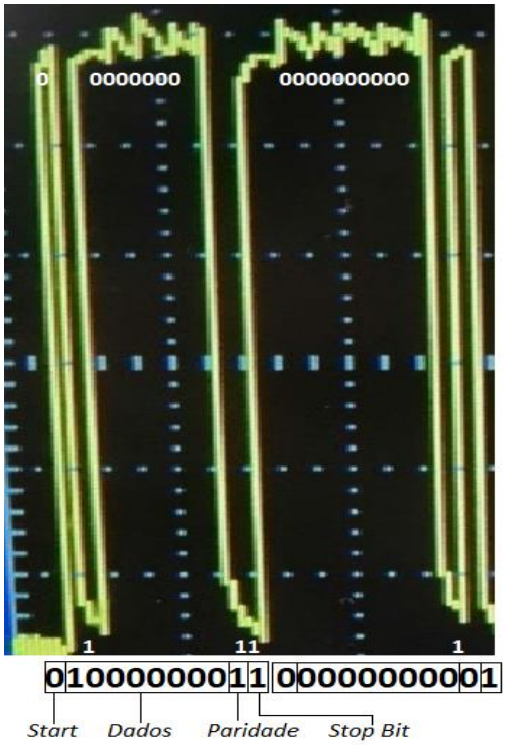


Figura 4.3 – Representação em detalhe da mensagem no osciloscópio

4.2.2 Funcionamento geral do simulador

Na figura 4.4 é possível observar o simulador parado, em modo automático e com o nível mínimo em todos os tanques exceto o tanque de mistura. É difícil captar o funcionamento do simulador em imagens, no entanto é interessante verificar um detalhe que foi explicado sobre algumas limitações do trabalho. Enquanto na supervisão são representados de igual modo os níveis mínimos dos tanques, é possível verificar que na realidade os níveis de água nos tanques são diferentes. O tanque de

aquecimento tem o seu nível praticamente a meio enquanto o tanque de filtragem está realmente no mínimo. Tal deve-se ao facto da existência de apenas dois sensores de nível, não sendo possível precisar o nível da água durante o enchimento.

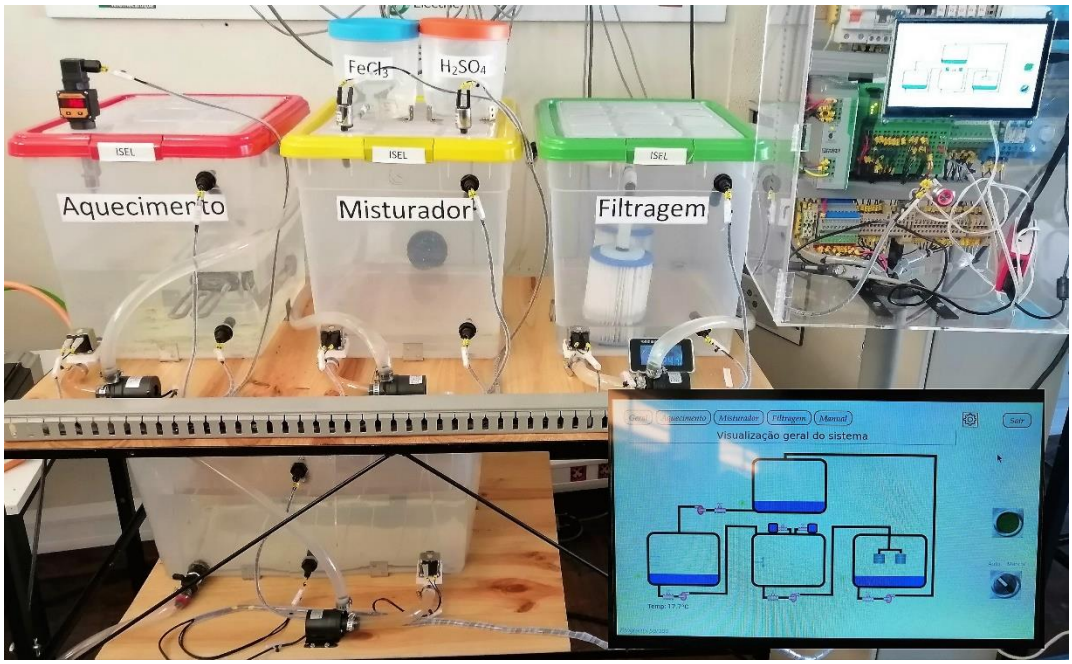


Figura 4.4 – Representação de um exemplo de funcionamento geral

4.2.3 Situação de falha nos sensores de nível

Para a simulação de falha nos sensores de nível foi necessário ativar manualmente o sensor máximo do tanque de filtragem. Nesta situação, o sensor de nível mínimo fica desativado enquanto o de nível máximo permanece ativo, despoletando uma mensagem de erro. Não sendo a representação ideal, é possível, no entanto, captar a ideia geral do funcionamento.

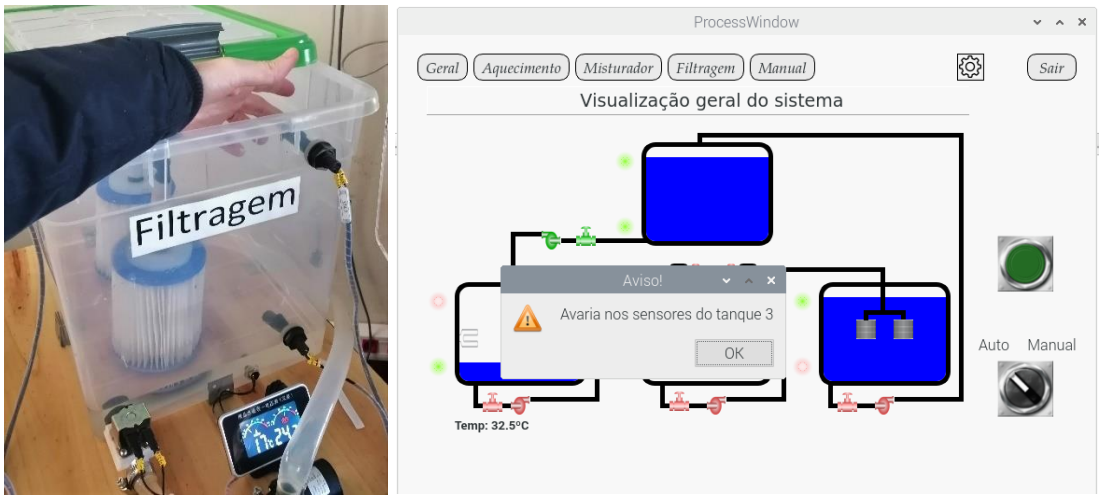


Figura 4.5 – Representação de uma situação de falha nos sensores de nível

Capítulo 5

Conclusões

Resumo: No presente capítulo apresentam-se as conclusões obtidas a partir da realização desta dissertação no âmbito da área de automação industrial e ainda algumas possibilidades de desenvolvimento futuro.

4.3 Conclusões

Era pretendido como objetivo da presente dissertação desenvolver um simulador didático de automação industrial apoiado num sistema de supervisão com o propósito de dar a oportunidade aos alunos de licenciatura e mestrado do curso de engenharia eletrotécnica, do Instituto Superior de Engenharia de Lisboa, de interagir e testar diversos processos automatizados baseados em estações de tratamento de água. Para tal, foi desenvolvida uma solução baseada num sistema de seis reservatórios, juntamente com um sistema de supervisão que permite comandar e monitorizar o simulador através de um *Raspberry Pi* com um *display* com ecrã *touch* e uma aplicação *Android*.

A automação industrial é atualmente uma área de natureza multidisciplinar. Com o simulador didático desenvolvido foi possível estudar e aplicar os diversos ramos da automação industrial como a instrumentação, o controlo, a potência, as redes de campo e os protocolos de comunicação. Foi ainda possível abordar alguns tipos de tecnologia mais recentes como é o caso do *Raspberry Pi* e do sistema *Android* e integrá-los com o equipamento mais usado no meio industrial, o PLC.

Pela complexidade do projeto, durante todo o seu desenvolvimento foram aparecendo diversos desafios. Muitos já estavam contabilizados no projeto, mas alguns foram surgindo inesperadamente, obrigando a uma adaptação do mesmo. Um dos desafios prendeu-se com o facto de não ter trabalhado, no passado, com nenhum dos equipamentos utilizados. Como tal, foi necessário estudar e familiarizar com a operação do PLC da Phoenix Contact, do *Raspberry Pi* e do sistema *Android*. No entanto, o maior desafio foi realizar a comunicação entre os três equipamentos numa linguagem de programação com a qual nunca tinha trabalhado, mantendo a interface gráfica da supervisão funcional e intuitiva. Para tal, houve uma curva de aprendizagem significativa desde a realização de exemplos de envio e receção de pequenas mensagens entre o PLC, o *Raspberry Pi* e a aplicação *Android*, até à interface final onde existe envio e receção de mensagens em tempo real.

Para tal, foi utilizada uma rede de comunicação série de ligação ponto a ponto RS232-C com transmissão *half-duplex*, um *baudrate* de 9600 bits/s, 8 *bits* de dados, paridade par e 1 stop *bit* para interligar o PLC e o *Raspberry Pi*. Foi também usada a comunicação sem fios *Bluetooth* para interligar o *Raspberry Pi* e um telemóvel com uma aplicação *android* desenvolvida especificamente para este simulador. Com este tipo de comunicação, a adoção de uma estrutura recorrendo a *threads* de comunicação foi essencial para manter a interface gráfica da supervisão funcional e intuitiva.

De uma forma geral o simulador didático foi concluído com sucesso cumprindo os objetivos propostos. É possível realizar o aquecimento, mistura e filtragem da água de forma automática e manual através de um *display* com ecrã *touch* e de uma aplicação *Android* instalada no telemóvel. Todos os estados dos sensores e atuadores são apresentados em tempo real na supervisão, assim como algumas mensagens de erro e diagnóstico. Contudo nenhum sistema é perfeito e existe sempre espaço para melhorias.

4.4 Perspetivas de desenvolvimento futuro

Para desenvolvimento futuro pode ser adicionada a opção de alteração dos tempos de enchimento dos tanques já iniciada. Para tal deve ser alterado o programa do PLC e a construção da sua mensagem com o *Raspberry Pi*. De forma a melhorar o simulador didático desenvolvido é também possível atuar nos três subgrupos da seguinte forma:

No subgrupo de instrumentação sugere-se:

- A substituição das eletroválvulas para facilitar o fluxo de água e aumentar a velocidade de enchimento dos tanques;
- A colocação de um sensor de nível intermédio nos três tanques e reservatório para aumentar a sensibilidade do simulador e as possibilidades de programação.

No subgrupo de controlo e potência é possível:

- A manipulação da programação do PLC, alterando os programas de cada etapa para a realização de outras tarefas;

No subgrupo de comunicação e supervisão sugere-se:

- A otimização da programação da aplicação *Android*, podendo ser usado outro ambiente de desenvolvimento;
- A adoção de uma rede Modbus TCP e de uma VPN, transformando o simulador num laboratório remoto, sendo possível controlá-lo fora das instalações do ISEL.

Bibliografia

[1] M. W. Marshall and R. W. Burchfield, "'Automation' Today and in 1669," *Am. Speech*, vol. 34, no. 3, p. 236, 1959.

[2] Bennett, S., "A brief history of automatic control.," *IEEE Control Syst.*, vol. 44, no. June 1996, pp. 17–25, 1996.

[3] O. Katsuhiko, "Modern Control Engineering (Ogata 3rd Edition)." 1996.

[4] R. Routledge, *Discoveries and Inventions of the Ninteenth Century*, vol. 14, no. 316. Routledge, 2018.

[5] S. Francis, "Top 20 programmable logic controller manufacturers," 2020. [Online]. Available: https://roboticsandautomationnews.com/2020/07/15/top-20-programmable-logic-controller-manufacturers/33153/.

[6] M. J. Walker, "The Programmable Logic Controller: its prehistory, emergence and application," Faculty of Mathematics, Computing and Technology, 2012.

[7] W. Bolton, *Programmable Logic Controllers*, 4th ed., vol. 53, no. 9. 2006.

[8] E. R. Alphonsus and M. O. Abdullah, "A review on the applications of programmable logic controllers (PLCs)," *Renewable and Sustainable Energy Reviews*, vol. 60. Elsevier, pp. 1185–1205, 2016.

[9] Mitsubishi Electric Automation, "Q Series Programmable Logic Controllers," 2019.

[10] S. Keane, "Chapter 1 PLC Ladder Diagram and the Coding Rules of Mnemonic," no. 1, pp. 1–15, 2011.

[11] J. Palma, "Introdução Às Redes De Campo De Automação," pp. 2004–2017, 2004.

[12] B. M. Wilamowski and J. D. Irwin, *Industrial communication systems*, 2nd ed., no. 2. CRC Press, 2016.

[13] P. Costa, "Integração e supervisão de múltiplas redes de automação," 2016.

[14] Behrouz A. Forouzan, "Data Communication and Networking, 4th Edition." pp. 1–1134, 2007.

[15] ANACOM, "Manual ITED 4ª Edição," 2019.

[16] "Standards reference," in *Practical Electrical Equipment and Installations in Hazardous Areas*, Elsevier, 2005, p. 361.

[17] S. Zhang, N. Marketing, and R. Automation, "EVALUATING INDUSTRIAL ETHERNET."

[18] IEEE Computer Society, *IEEE Standard for Ethernet*. 2018.

[19] N. I. Aristova, "Ethernet in industrial automation: Overcoming obstacles," *Autom. Remote Control*, vol. 77, no. 5, pp. 881–894, 2016.

[20] M. I. U. S. a, "Introduction To Modbus Tcp / Ip," *Instrumentation*, vol. 44, no. 248, 2005.

[21] Modbus - IDA, "Modbus Application Protocol Specification," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, pp. 1–51, 2006.

[22] IDA Modbus, "Modbus messaging on TCP." pp. 1–46, 2006.

[23] S. A. Boyer, *SCADA: Supervisory Control and Data Acquisition, Fourth Edition*. International Society of Automation, 2016.

[24] Gordon Clarke, D. Reynders, and E. Wright, *Practical Modern SCADA Protocols*. Elsevier, 2003.

[25] S. G. McCrady, *Designing SCADA Application Software*. Elsevier, 2013.

[26] Progea, "Movicon Programer Guide," no. 11.5, 2016.

[27] Progea, "Movicon.Next 4.0 - Innovative, powerful, simple, scalable for every automation need."

[28] OPC Foundation, "What is OPC?," *About OPC*, 2010. [Online]. Available:

https://opcfoundation.org/about/what-is-opc/.

[29] W. Mahnke, S. H. Leitner, and M. Damm, *OPC unified architecture*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[30] T. Burke, “OPC Unified Architecture - Interoperability for Industrie 4.0 and the Internet of Things,” 2016.

[31] Germany Trade & Invest, “SMART MANUFACTURING FOR THE FUTURE INDUSTRIE 4.0 Future Markets,” 2014.

[32] S. Vaidya, P. Ambad, and S. Bhosle, “Industry 4.0 - A Glimpse,” in *Procedia Manufacturing*, 2018, vol. 20, pp. 233–238.

[33] S. Erol, A. Jäger, P. Hold, K. Ott, and W. Sihn, “Tangible Industry 4.0: A Scenario-Based Approach to Learning for the Future of Production,” in *Procedia CIRP*, 2016, vol. 54, pp. 13–18.

[34] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[35] K. Kumar, D. Zindani, and J. P. Davim, “Industry 4.0 Developments towards the Fourth Industrial Revolution,” in *SpringerBriefs in Applied Sciences and Technology. Springer, Singapore*, 2019, p. 59.

[36] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “The industrial internet of things (IIoT): An analysis framework,” *Comput. Ind.*, vol. 101, no. April, pp. 1–12, 2018.

[37] O. Alzakholi, L. Haji, H. Shukur, R. Zebari, S. Abas, and M. Sadeeq, “Comparison Among Cloud Technologies and Cloud Performance,” *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 40–47, 2020.

[38] Schneider Electric, “EcoStruxure Augmented Operator Advisor,” no. April, 2019.

[39] R. M. E. D. Nafea and E. K. Toplu, “Future of Education in Industry 4.0,” no. February, 2019, pp. 267–287.

[40] M. Elbestawi, D. Centea, I. Singh, and T. Wanyama, “SEPT Learning Factory for Industry 4.0 Education and Applied Research,” *Procedia Manuf.*, vol. 23, no. 2017, pp. 249–254, 2018.

[41] L. De Nul, M. Breque, and A. Petridis, *Industry 5.0 - Towards a sustainable, human-centric and resilient European industry*. 2021.

[42] J. Müller, “Enabling Technologies for Industry 5.0 - Results of a workshop with Europe’s technology leaders,” 2020.

[43] E. F. Crawley, J. Malmqvist, S. Östlund, and D. R. Brodeur, *Rethinking engineering education: The CDIO approach*. Boston, MA: Springer US, 2007.

[44] A. L. Mackay, P. Medawar, A. L. Mackay, and P. Medawar, *A Dictionary of Scientific Quotations*, vol. 16, no. 1. 2019.

[45] L. J. Shuman *et al.*, “The future of engineering education,” vol. 50, no. 1, 2002.

[46] W. A. Wulf and G. M. C. Fisher, “A makeover for engineering education,” *Issues Sci. Technol.*, vol. 18, no. 3, p. 35, 2002.

[47] D. May and C. Terkowsky, “What should they learn? A short comparison between different areas of competence and accreditation boards’ criteria for engineering education,” *IEEE Glob. Eng. Educ. Conf.*, pp. 1046–1050, 2014.

[48] K. Moore, C. Jones, and R. S. Frazier, “Engineering Education For Generation Z,” *Am. J. Eng. Educ.*, vol. 8, no. 2, pp. 111–126, 2017.

[49] N. Lima *et al.*, “Do Students Really Understand the Difference Between Simulation and Remote Labs ?,” 2017.

[50] C. E. Pereira, S. Paladini, and F. M. Schaf, “Control and Automation Engineering Education : combining physical , remote and virtual labs,” 2012.

[51] M. S. Matijevi and N. D. Jovi, “Remote Labs and Problem Oriented Engineering Education,” no. April, pp. 1391–1396, 2017.

[52] F. Soares, C. P. Leão, J. Machado, and V. Carvalho, “Experiences in Automation and Control in Engineering Education with Real-world Based Educational Kits,” *Sensors & Transducers*, vol. 193, no. 10, pp. 135–144, 2015.

[53] J. V Nickerson, “Hands-on , simulated , and remote laboratories : A comparative literature review Hands-On , Simulated , and Remote Laboratories : A Comparative Literature Review,” no. April, 2015.

[54] J. Palma, “APLICAÇÕES DE AUTOMAÇÃO EM SISTEMAS DE ABASTECIMENTO DE ÁGUA.” 1999.

[55] Arianne Nunes Dualibi, “ESTUDO COMPARATIVO DA INFLUÊNCIA DO SULFATO DE ALUMÍNIO LÍQUIDO E SULFATO DE ALUMÍNIO GRANULADO NA TURBIDEZ, COR E NO VOLUME DE RESÍDUO GERADO NO TRATAMENTO DE ÁGUAS.,” UNIVERSIDADE FEDERAL DE MATO GROSSO, 2010.

[56] H. M. Monte, M. T. Santos, A. M. Barreiros, and A. Albuquerque, *Tratamento de Águas Residuais - Operações e Processos de Tratamento Físico e Químico*, Entidade R. 2016.

[57] C. E. R. da Cunha, “Telegestão de uma rede de abastecimento de água e drenagem de águas residuais,” p. 150, 2007.

Anexos

Resumo: Apresentam-se como anexos, os esquemas das ligações elétricas.

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

SIMULADOR DE SISTEMA DE TRATAMENTO DE ÁGUA

QUADRO ELÉCTRICO DE COMANDO E POTÊNCIA

		Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
	Simulador de Sistema de Tratamento de Água Quadro Elétrico de Potência e Comando		Desenho nº 0.0 Ano Letivo 2020/2021 Substitui Substituido

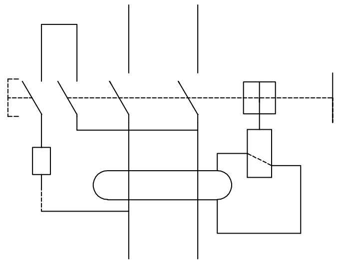
PRODUCED BY AN AUTODESK STUDENT VERSION

INDICIE:

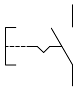
	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
ÍNDICE		Desenho nº 0.1
		Ano Letivo 2020/2021
		Substitui / Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION

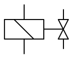
SIMBOLOGIA:



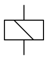
– Interruptor Diferencial;



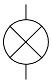
– Interruptor com Encravamento;



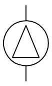
– Eletroválvula;



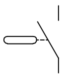
– Contactor Auxiliar;



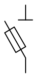
– Sinalizador;



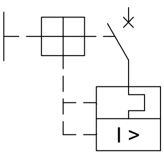
– Electrobomba;



– Sensor Nível;



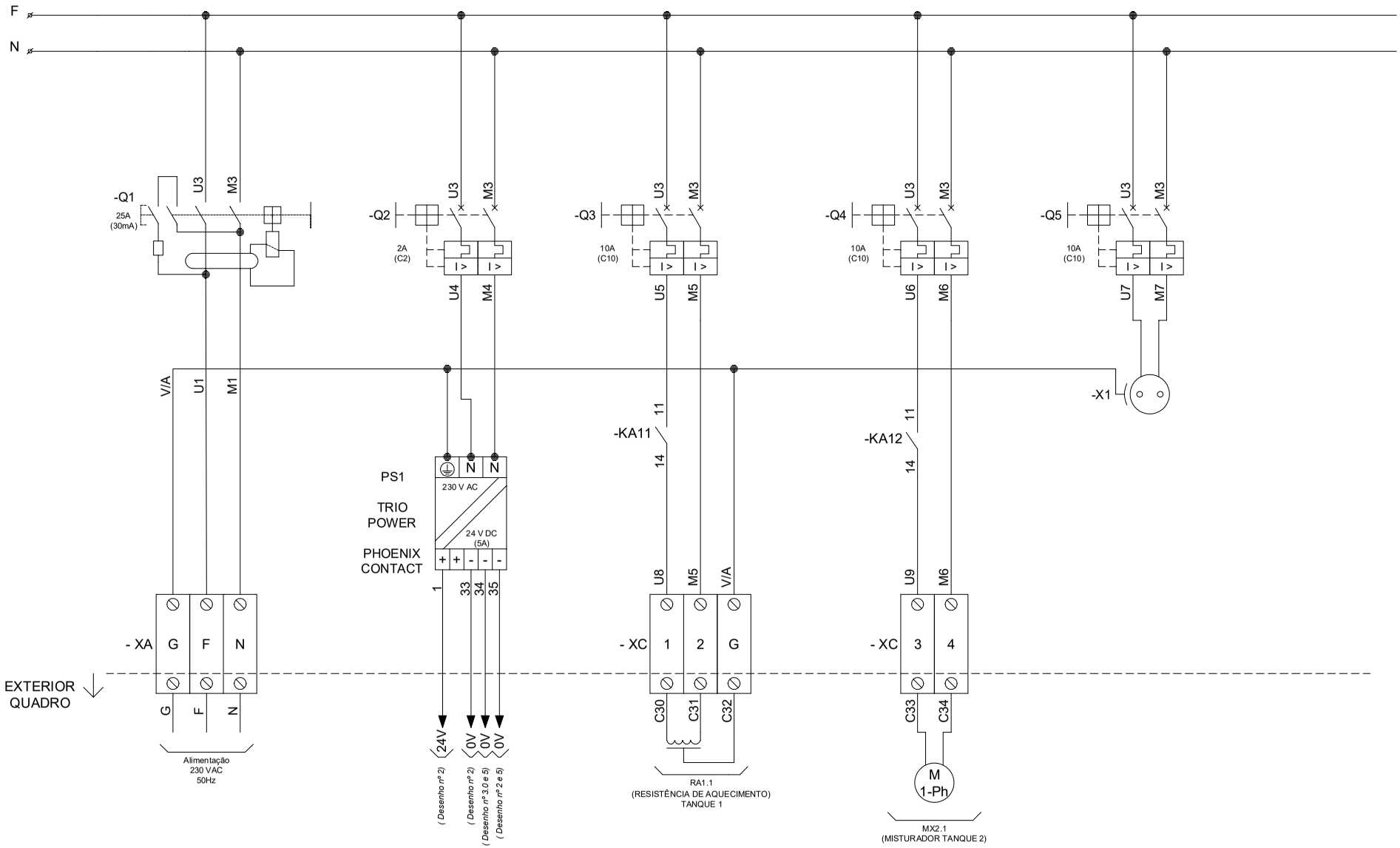
– Secionador porta-fusíveis;



– Disjuntor magnetotermico;

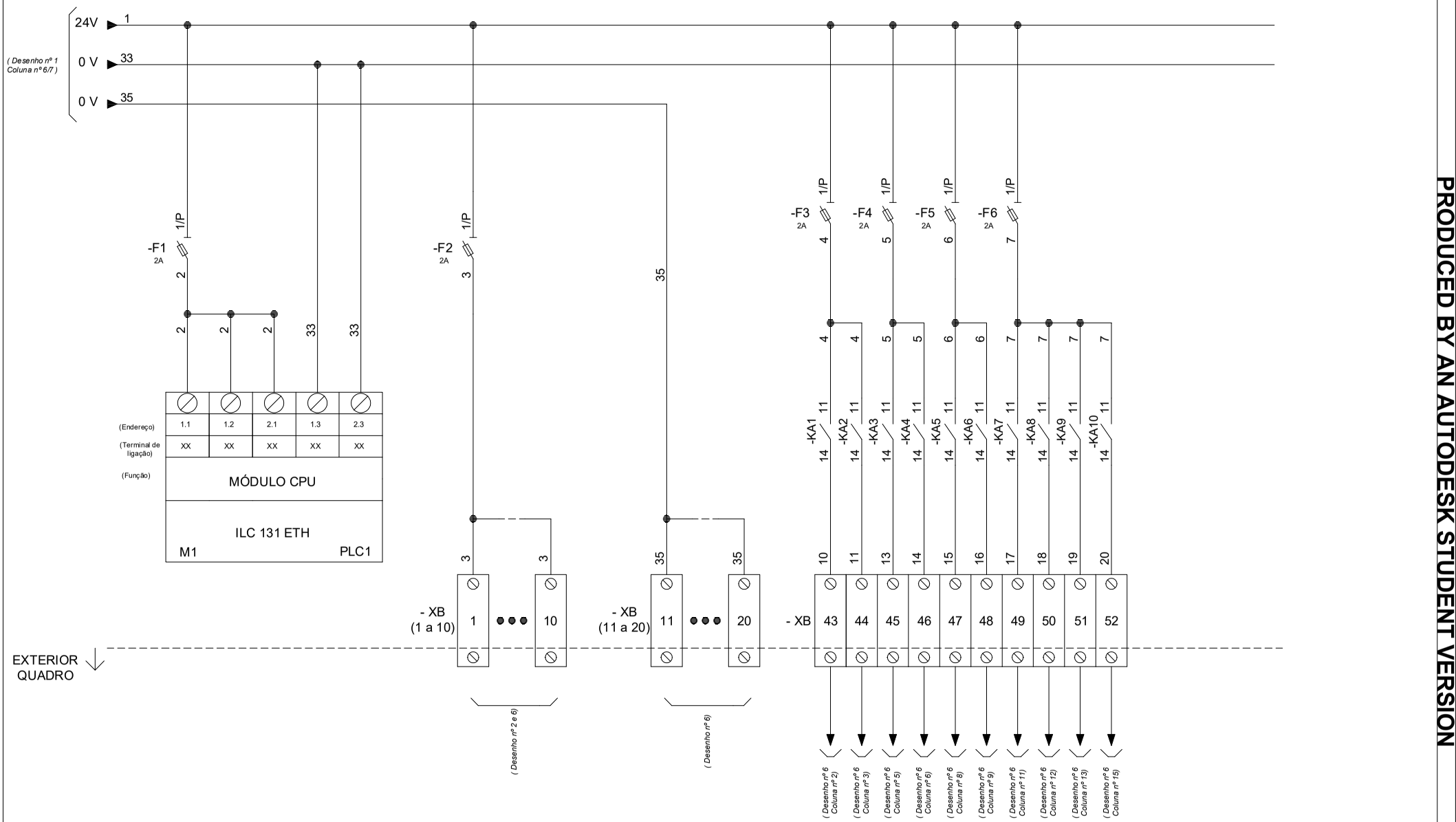
	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
	SIMBOLOGIA	Desenho nº 0.2
		Ano Letivo 2020/2021
		Substitui
		Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION



	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica Energia e Automação
	ESQUEMA LIGAÇÕES MONOFÁSICAS	Desenho nº 1 Ano Letivo 2020/2021 Substitui Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION



Secção de Automação e Robótica
ISEL

ISEL
Área Departamental de Engenharia Electrotécnica de Energia e Automação

PROTEÇÕES 24V

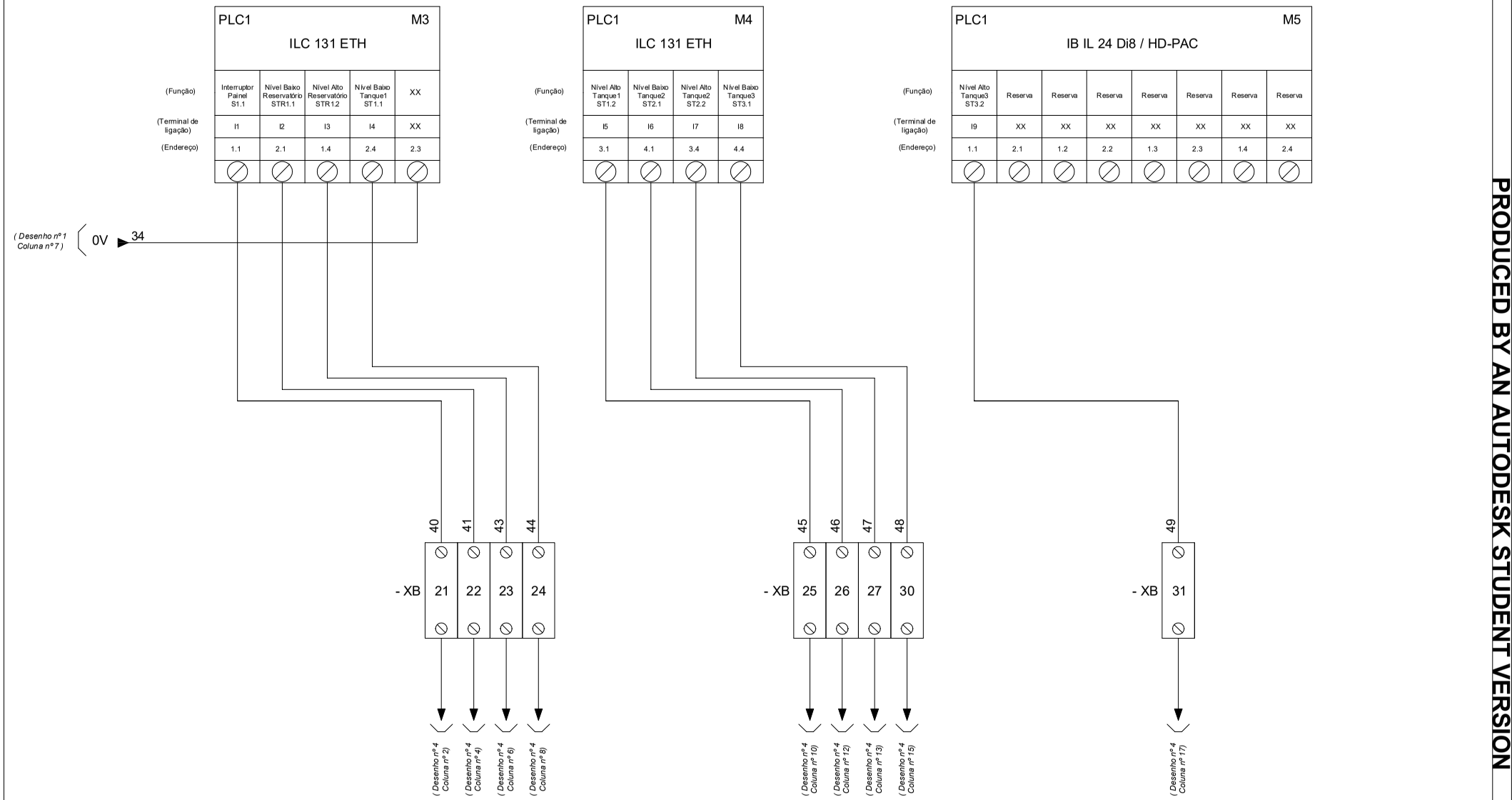
Desenho nº 2

Ano Letivo 2020/2021

Substitui

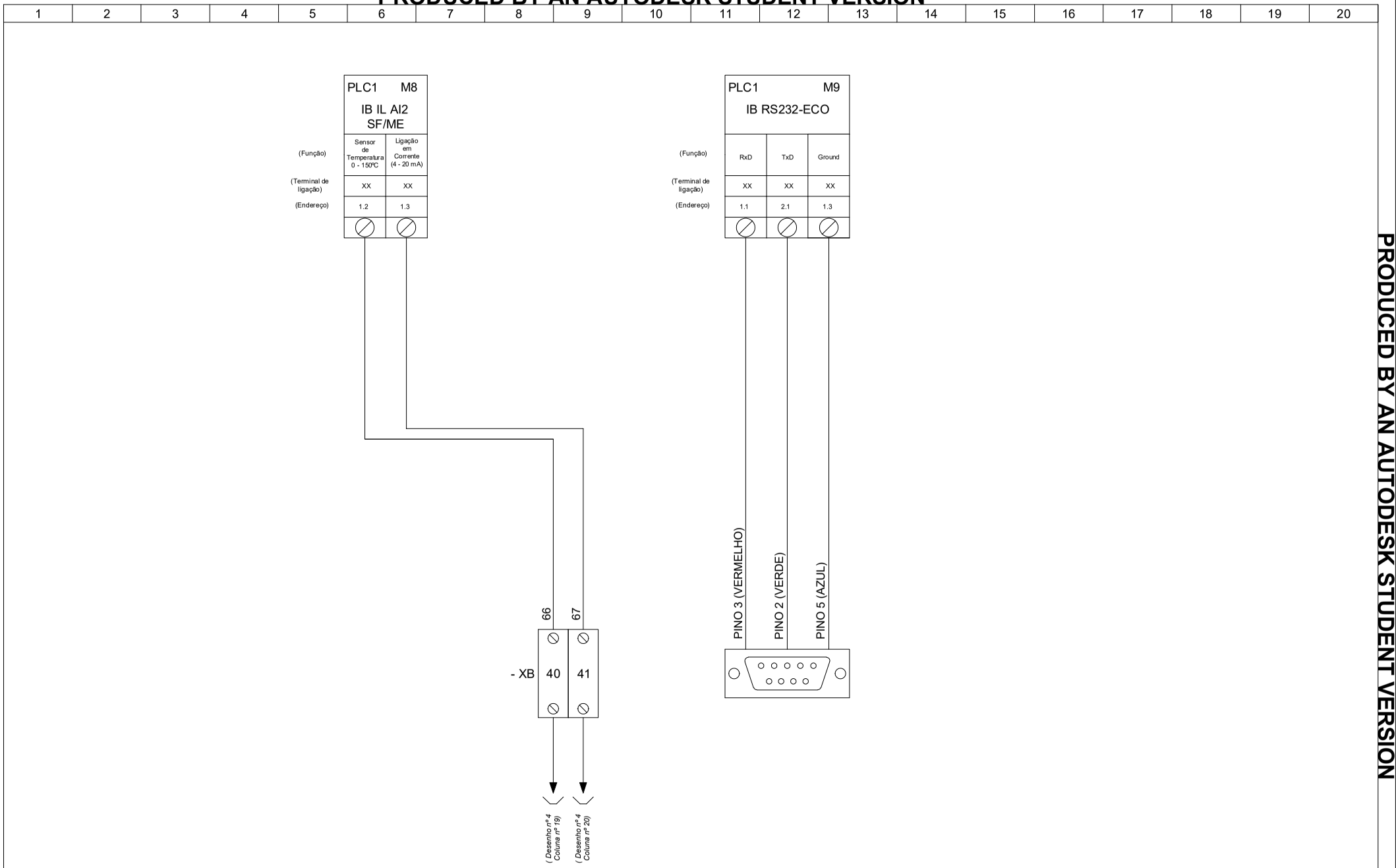
Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION



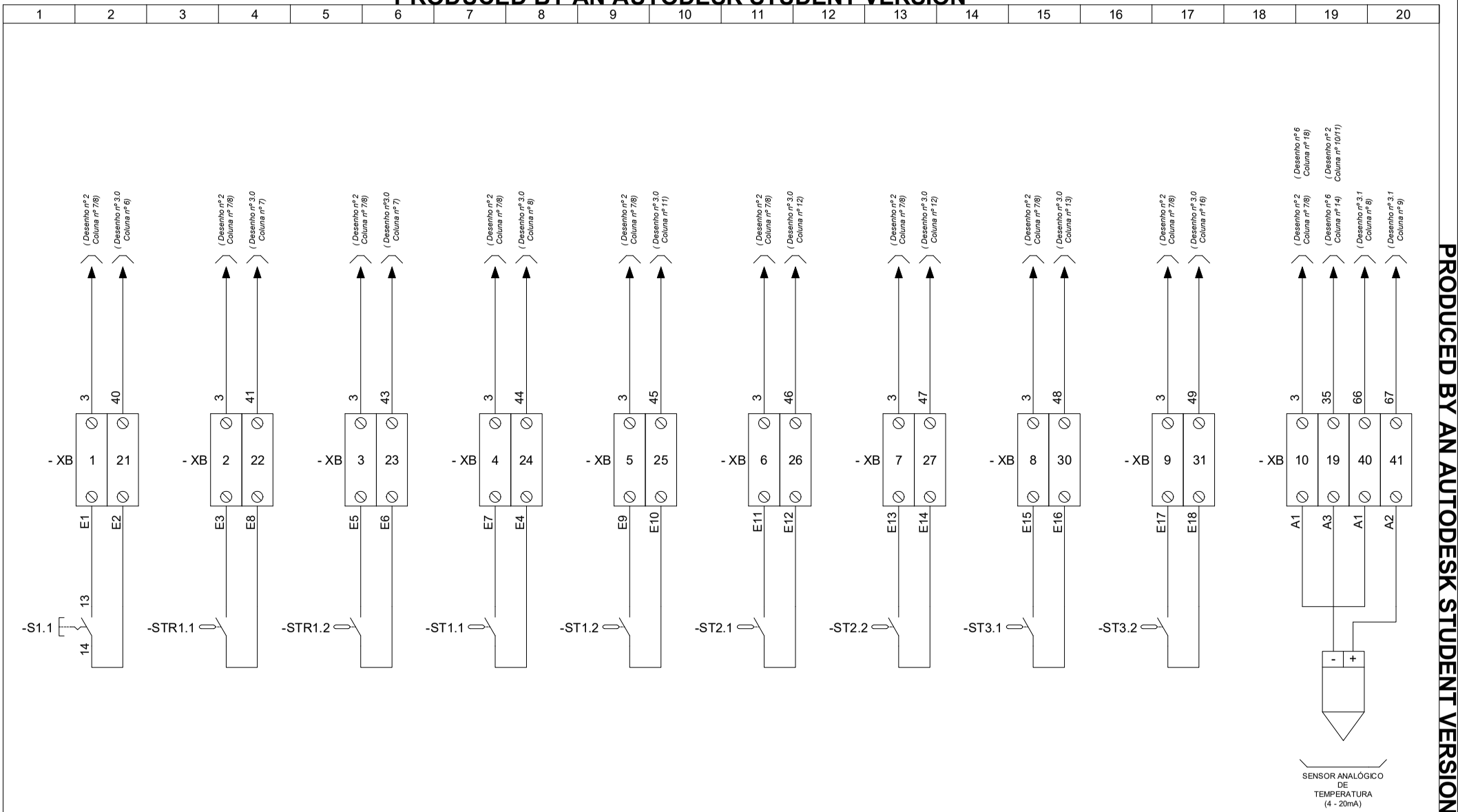
	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
	BORNES DE ENTRADAS	Desenho nº 3.0 Ano Letivo 2020/2021 Substitui Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION

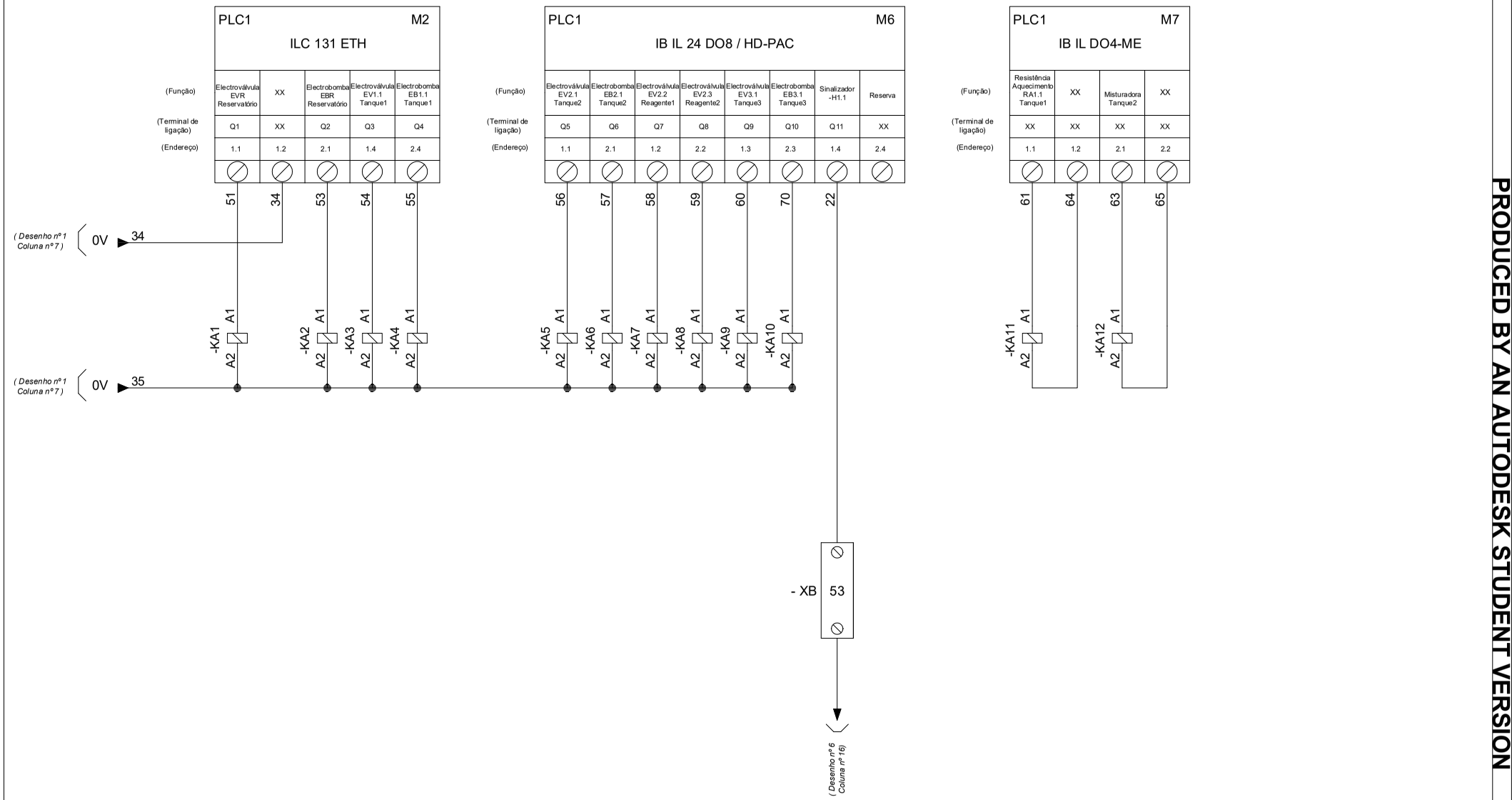


	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
	BORNES DE ENTRADAS	Desenho nº 3.1 Ano Letivo 2020/2021 Substitui Substituido

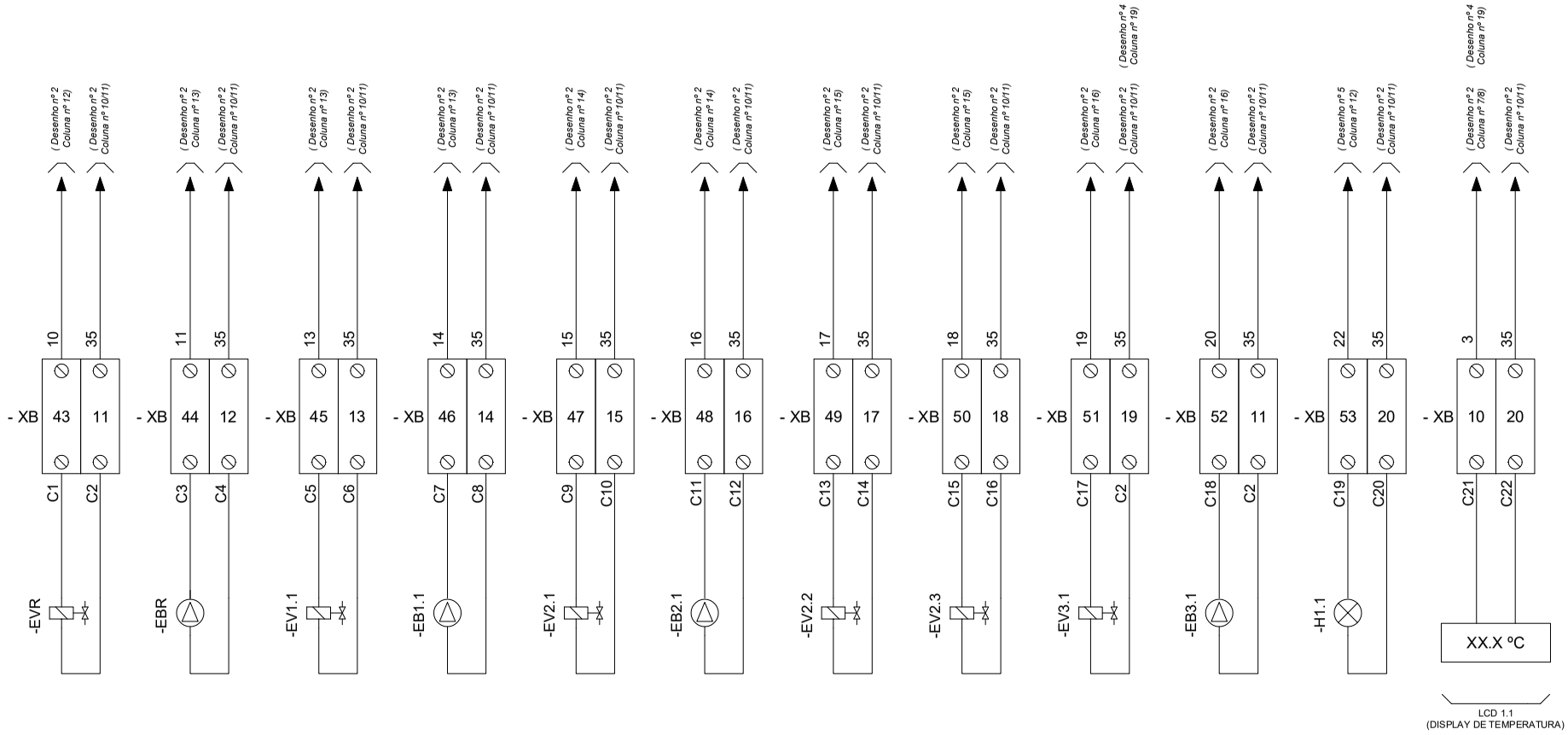
PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION
PRODUCED BY AN AUTODESK STUDENT VERSION



	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
LIGAÇÕES DAS ENTRADAS BINÁRIAS E ANALÓGICAS		Desenho nº 4 Ano Letivo 2020/2021 Substitui Substituido



	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
	LIGAÇÕES DAS SAIDAS	Desenho nº 5 Ano Letivo 2020/2021 Substitui Substituido



	Secção de Automação e Robótica ISEL	ISEL Área Departamental de Engenharia Electrotécnica de Energia e Automação
BORNES DE SAÍDA		Desenho nº 6 Ano Letivo 2020/2021 Substitui Substituido

PRODUCED BY AN AUTODESK STUDENT VERSION