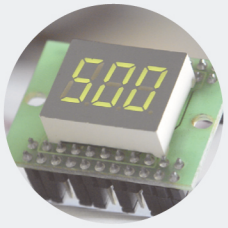
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






SIMULADOR DE VALIDAÇÃO DE SISTEMAS DE SINALIZAÇÃO FERROVIÁRIA

ROBERTO XAVIER DA COSTA SOUSA
novembro de 2022

POLITÉCNICO
DO PORTO

Instituto Politécnico do Porto

Instituto Superior de Engenharia do Porto

Simulador de validação de sistemas de sinalização ferroviária

Roberto Xavier Costa Sousa

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Outubro, 2022

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Roberto Xavier Costa Sousa, 1141269@isep.ipp.pt
Orientação Teórica: Manuel Gericota, mgg@isep.ipp.pt
Empresa: EFACEC, mktransportes@efacec.com
Orientação: João Marques Martins, joao.martins@efacec.com

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Outubro, 2022

Resumo

O ritmo ao qual novas funcionalidades estão a ser solicitadas nos produtos de sinalização, bem como em todos os outros tipos de sistemas, está em constante aceleração. O que, por sua vez, aumenta significativamente a pressão nas atividades de validação, aumentando o risco na implementação de sistemas com um nível de maturidade insuficiente. De forma a mitigar estes riscos, associados às novas funcionalidades dos produtos de sinalização SIL4, um protótipo de um sistema precisa de ter em conta, durante o processo de desenvolvimento, as funcionalidades dos sistemas nos quais serão implementados, permitindo ajustes tanto ao nível de requisitos como de implementação. O protótipo do sistema é composto por diversos componentes. Durante a fase de comissionamento, alguns dos componentes são substituídos pelos componentes finais e podem correr em diferentes alvos. Assim de forma a garantir o funcionamento do protótipo, existe a necessidade de implementação de novas interfaces.

O objetivo deste projeto é a implementação de uma nova interface que permite que equipamentos de simulação de via possam comunicar com produtos de sinalização que estão a operar em sistemas finais, nomeadamente nos controladores lógicos programáveis (PLC). Assim, numa etapa inicial, a interface foi disponibilizada ao nível da camada de aplicação, enquanto que na fase de integração uma nova interface é fornecida através de um protocolo de comunicação, uma vez que os dois componentes são executados em unidades de processamento diferentes.

Palavras-Chave

Prototipagem, Padrões Ferroviários, Princípios de Encravamento, Princípios de Sinalização, PLC.

Abstract

The rate new functionalities are being requested in signaling products, like in all other systems types, is constantly accelerating. Additionally, it significantly increases pressure on validation activities, increasing the risk of implementing systems with an insufficient level of maturity. To mitigate these risks associated with the new functionalities of the SIL4 signaling products, a prototype of the system needs to consider the functionalities of the systems where it will be integrated while allowing adjustments both in terms of requirements and implementation. The system prototype is composed of several components. During the commissioning phase, some of them are replaced by the final ones and may run on different targets. Thus, to ensure the correct prototype operation, it is necessary to implement new interfaces.

The project objective is to implement a new interface that allows trackside simulation equipment to communicate with signaling products operating in final systems, namely in programmable logic controllers (PLC). Therefore, in the initial stage, the interface is made available at the application layer level. In the integration stage, a new interface is provided through a communication protocol since the two components run on different processing units.

Keywords

Prototyping, Railway Standards, Interlocking Principles, Signalling Principles, PLC.

Conteúdo

Conteúdo **i**

Lista de Figuras **v**

Lista de Tabelas **vii**

Glossário **ix**

Acrónimos **xiii**

Agradecimentos **xv**

1 Introdução **1**
1.1 Contextualização 1
1.2 Objetivos 2
1.3 Empresa EFACEC 3
1.4 Organização do Relatório 4

2 Estado da arte **7**
2.1 Evolução da Indústria Ferroviária 7
2.1.1 Indústria Férrea em Portugal 9
2.2 Encravamento e Sinalização Ferroviária 10
2.2.1 Princípios de Encravamento 11
2.2.1.1 Acoplamento de Elementos 13
2.2.1.2 Bloqueio Unidirecional 14
2.2.1.3 Bloqueio Bidirecional Simples 14
2.2.2 Itinerário 15
2.2.2.1 Definições de Termos 16
2.2.2.2 Extensão de Itinerário e Restrições de Velocidade Associadas 17
2.2.3 Encravamento de Itinerários Contraditórios 18

2.2.3.1 Outras Funções de Encravamento 19
2.2.4 Proteção de Flanco 20
2.2.4.1 Proteção de Flanco Ramificado e por Transferência 22
2.2.5 Princípios de Formação de Itinerário no Plano da Via . . . 24
2.3 Protocolos de Comunicação Industrial 28
2.3.1 Profinet 28
2.3.1.1 Profinet IO 29
2.3.2 OPC 30
2.3.2.1 OPC UA 31
2.3.3 Modbus 32
2.3.3.1 Modbus TCP/IP 32

3 Arquitetura do Sistema **35**
3.1 Contexto Inicial 35
3.2 Protocolo de Comunicação 37
3.3 Ambiente de Desenvolvimento Integrado 37
3.3.1 Windows Presentation Foundation 38
3.3.1.1 XAML 39
3.3.1.2 .NET Framework 39
3.4 Controlador Lógico Programável 40
3.5 Comunicação de Teste EasyModBus 41
3.6 Configuração Teste por Ficheiro CSV 43

4 Implementação do Sistema **47**
4.1 Introdução 48
4.2 Desenvolvimento 48
4.2.1 Interface Gráfica *DEN Depot Zone* 48
4.2.2 Aegis Trackside Equipment Standalone Simulator 50
4.2.2.1 Classe Read Configuration CSV 50
4.2.2.2 Classe *ScadeDisplayManager* 52
4.2.2.3 Classe *SD Host* 52
4.2.2.4 Classe *GUI* 52
4.2.2.5 Classe PLC 54
4.2.3 Interface Gráfica *Control and Maintenance Center* 56

5 Testes e Resultados **59**
5.1 Conexão 59
5.2 Comunicação entre as Interfaces Gráficas 61
5.2.1 Cenário de Ocupação de Secção de Via 61
5.2.2 Cenários de Criação de um Itinerário 62
5.2.2.1 Cenário do Itinerário SR3 (*Signal Box S1*) 62
5.2.2.2 Cenário do Itinerário SR2 (*Signal Box S2*) 63

5.3 Cenário de Itinerário, Sinalização e *Feedback* 65

6 Conclusão **69**
6.1 Considerações Finais 69
6.2 Principais Dificuldades Encontradas 70
6.3 Trabalho Futuro 70

Referências Bibliográficas **73**

Lista de Figuras

1.1 Logótipo e slogan da empresa. 3

2.1 Turway Diolkos, Grécia. 8
2.2 Logótipo da empresa. 9
2.3 Esfera global do sistema ferroviário. Adaptado. 11
2.4 Medidas de segurança e perigo em operações de ferrovia. 12
2.5 Comutação de dois elementos acoplados. 14
2.6 Dependência unidirecional entre o sinal principal e o sinal distante. . 14
2.7 Dependência sequencial entre pontos e sinalização. 15
2.8 Dependência sequencial entre pontos e descarrilador. 15
2.9 Partes lógicas de um itinerário. 17
2.10 Encravamento de Itinerário Simples e Especial. 19
2.11 Métodos para evitar a libertação prematura de um itinerário. 20
2.12 Métodos de proteção de flanco. 21
2.13 Proteção de flanco por transferência. 22
2.14 Proteção de flanco ramificado. 23
2.15 Solicitação de proteção de flanco por dois itinerários (caso da esquerda) e pelo mesmo itinerário (caso da direita). 24
2.16 Plano para a matriz de bloqueio de itinerário, segundo a matriz da tabela 2.1. 25
2.17 Mapa de conexão do conjunto de relé simplificado para o encravamento topológico. 26
2.18 Esforço técnico para o princípio tubular e topológico no encravamento de relés. 27
2.19 Processo de dados ASE. 30
2.20 Diferença entre OPC Clássico e o OPC UA. 31
2.21 Diagrama de construção de um pacote de dados Modbus TCP. 33

3.1 Diagrama de blocos da arquitetura do sistema. 36
3.2 Arquitetura de vinculação do objeto. Adaptado. 39

3.3 Painel Frontal: Himatrix F30. 40
3.4 Arquitetura da rede de teste. 41
3.5 Cliente *EasyModBus*. 42
3.6 Servidor *EasyModBus*. 43
3.7 Configuração *Modbus* em ficheiro csv. 43
3.8 Classe de *getters* e *setters* criado para a estrutura das listas de configuração *Modbus*. 44
3.9 Estrutura da lista *ReadBus* por posição. 45

4.1 Arquitetura de Projeto Final. 47
4.2 Interface Gráfica *DEN Depot Zone*. 49
4.3 Fluxograma de funcionamento da classe *Read Configuration CSV*. . . . 51
4.4 Fluxograma de funcionamento da classe *GUI*. 53
4.5 Fluxograma de funcionamento da classe PLC 55
4.6 Interface Gráfica *Control and Maintenance Center*. 57

5.1 Notificação de conexão ao PLC com sucesso. 60
5.2 Notificação de perda de conexão com o PLC. 60
5.3 Teste de ocupação de via com recurso às interfaces gráficas: *DEN Depot Zone* e *Control Maintenance Center*. 61
5.4 Teste de criação do itinerário SR3: *DEN Depot Zone*. 62
5.5 Teste de criação do itinerário SR3: *Control Maintenance Center*. 63
5.6 Teste de criação do itinerário SR2 com recurso às interfaces gráficas: *DEN Depot Zone* e *Control Maintenance Center*. 64
5.7 Teste Cenário Final: *DEN Depot Zone*. 65
5.8 Teste Cenário Final: *Control Maintenance Center*. 66

Lista de Tabelas

2.1 Matriz de bloqueio de itinerário (exemplo). Adaptado. 25
2.2 Comparação entre os princípios tabular e geográfico. 26
2.3 Valor relativo do *jitter* calculado em relação à periodicidade. Adaptado 28

3.1 RJ-45 Led's - Estados de comunicação. Adaptado. 40

Glossário

- ***Assembly***: Linguagem de programação de baixo nível.
- **Datagrama**: Unidade de transferência básica associada a uma rede de comutação de pacotes. Os datagramas são normalmente estruturados em secções de cabeçalho e carga útil. Os datagramas fornecem um serviço de comunicação sem conexão numa rede comutada por pacotes.
- **Encravamento**: Interdependência entre os manípulos de comando ou os circuitos elétricos de comando dos diferentes aparelhos, agulhas, sinais ou outros, tornando impossível qualquer simultaneidade de posições incompatíveis do ponto de vista da segurança, nomeadamente quando defeitos no sistema de aferrolhamento possam por em causa a segurança das circulações e enquanto não se efetua a sua reparação.
- **EN ISO 13849-1**: A EN ISO 13849-1, como sucessora da EN 954-1, é a norma central para o dimensionamento de comandos de segurança orientados à segurança de máquinas.
- **EN 50128**: A EN 50128 é uma norma de segurança funcional adaptada às exigências específicas da indústria ferroviária. Ela fornece um conjunto de requisitos para o desenvolvimento, implantação e manutenção de qualquer software relacionado à segurança destinado a aplicações de controlo e proteção ferroviária.
- ***Framework***: É uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Um *framework* pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação.
- ***Gateway***: No contexto da Internet e das configurações de conexão, refere-se às ferramentas que funcionam como intermediário para troca de informações.

- **GitHub**: Trata-se de um serviço baseado em nuvem que hospeda um sistema de controlo de versão (VCS) chamado Git. Permite que os desenvolvedores colaborem e façam mudanças em projetos compartilhados enquanto mantêm um registo detalhado do seu progresso.

- **IEC 61508**: Criada pela IEC (*International Electrotechnical Commission*) em 1985, a norma IEC 61508 consiste nos padrões internacionais genéricos de segurança funcional voltada para dispositivos elétricos, eletrónicos ou eletronicamente programáveis.

- **IEC 61158**: A norma IEC 61158 da *International Electrotechnical Commission* abrange as comunicações de dados digitais para medição e controlo. Esta série de normas foi desenvolvida para protocolos de redes informáticas industriais, oferecendo controlo distribuído em tempo real.

- **IEC 61511**: A norma IEC 61511 é uma norma técnica que estabelece práticas de engenharia de sistemas que garantem a segurança de um processo industrial através do uso de instrumentação. Esses sistemas são chamados de sistemas de segurança . O título da norma é "Segurança Funcional - Sistemas de Segurança para o Setor da Indústria de Processos ".

- **IEC 62061**: A IEC 62061 especifica os requisitos de segurança para o projeto e implementação de sistemas de controlo de máquinas.

- **Itinerário**: Porção de via delimitada, na origem, por um sinal de protecção e, no destino, normalmente por um outro sinal, que pode ser percorrida em segurança quando esteja estabelecido e aberto o correspondente sinal de protecção. Nesta situação, estarão encravados todos os aparelhos de via com ele relacionados.

- **Jitter**: É uma variação estatística do atraso na entrega de dados numa rede, ou seja, pode ser definida como a medida de variação do atraso entre os pacotes sucessivos de dados.

- **Secção**: Troço de via localizado entre dois pontos singulares (estações e aparelhos de mudança de via, entre outros).

- **SIL**: SIL ou Nível de Integridade de Segurança (SIL) é baseado no valor de redução de risco associado a uma Função Instrumentada de Segurança (SIF) que protege contra um evento perigoso específico, ou como o risco deve ser reduzido para atingir um nível aceitável.

- **Sinalização Ferroviária**: Conjunto dos equipamentos técnicos, organização e regulamentos destinados a garantir a segurança e a eficácia do tráfego ferroviário.

- **Unidades Móveis**: Termo utilizado para designar veículos concebidos para medições de controlo, controlo de funções, protecção contra incêndios, evacuação, manutenção, reparação e reboque de veículos.
- **Via**: Infra-estrutura de transporte guiado cujo perfil transversal apresenta uma só via que pode ser percorrida nos dois sentidos. Conjunto de elementos que servem de base de sustentação e encaminhamento dos comboios.
- **.dll**: Também conhecidos como arquivos *Dynamic Link Library*, trata-se de uma biblioteca dinâmica que contém dados que podem ser acessados por mais do que um programa e que pode conter código, dados ou recursos (ícones, fontes, cursores, entre outros). Compostas por sub-rotinas armazenadas em disco, podem ser carregadas na memória e executadas quando uma aplicação realiza o seu acesso.

Acrónimos

Abreviatura	Descrição
A&E	*Alarm and Events*
ASE	*Application Service Element*
CBA	*Component Based Automation*
CF	*Caminho de Ferro*
CP	*Comboios de Portugal*
CSV	*Comma-separated values*
DA	*Data Acess*
EUA	*Estados Unidos da América*
EPS	*EFACEC Power Solutions*
GUI	*Graphical User Interface*
HDA	*Historical Data Acess*
IDE	*Integrated Development Environment*
IG	*Interface Gráfica*
IP	*Infraestruturas de Portugal*
MAC	*Media Access Control*
NRT	*Non Real Time*
OPC	*OLE (Object Linking and Embedding) Process Control*
OSI	*Open Systems Interconnection*
PLC	*Programmable Logic Controller*
REFER	*Rede Ferroviária Nacional*
RI	*Revolução Industrial*
RT	*Real Time (Tempo Real)*
SCADA	*Supervisory Control and Data Acquisition*
SOA	*Service-Oriented Architecture*
TCP	*Transmission Control Protocol*
VF	*Via Férrea*
VPN	*Virtual Private Network*
WPF	*Windows Presentation Foundation*

Agradecimentos

Com a realização e conclusão desta dissertação termina simultaneamente mais um capítulo da minha vida académica. Como tal, aproveito aqui para manifestar o meu maior agradecimento a algumas das pessoas que contribuíram para este sucesso.

Em primeiro lugar, gostaria de expressar o meu agradecimento ao Professor Doutor Manuel Gericota pela disponibilidade, orientação, contributo, apoio e boa disposição que sempre demonstrou ao longo do trabalho desenvolvido e que permitiu que este decorresse da melhor forma possível.

Um agradecimento especial à empresa Efacec, em particular ao meu orientador, o Engenheiro João Martins, que para além de me oferecer a oportunidade de realizar este trabalho, esteve sempre disponível, com um sentido de orientação, contributo e boa disposição que sempre demonstrou ao longo do estágio, bem como o resto da equipa do departamento de IDE da Efacec, que também contribuíram com um ambiente profissional, mas amigável e descontraído.

De seguida, gostaria de agradecer aos meus amigos, pelas opiniões, incentivos e bons momentos que permitiram tornar esta etapa académica especial e ainda mais agradável.

Por último, gostaria de oferecer o agradecimento mais importante aos meus pais, irmã e namorada, por todo o esforço, preocupação, incentivo e inúmeros outros aspetos fundamentais que me permitiram atingir este objetivo, pois só eles sabem o que foi necessário para chegar a este ponto.

Os meus sinceros agradecimentos!

Capítulo 1

Introdução

Neste capítulo é feita uma introdução ao tema, abordando o objetivo principal do projeto e a sua contextualização. Também faz parte deste capítulo a apresentação da EFACEC, empresa onde foi realizado este projeto/estágio. O capítulo termina com o planeamento do trabalho e a organização do relatório.

1.1 Contextualização

Os transportes são fundamentais para a sociedade providenciando mobilidade e facilitando as atividades económicas, nomeadamente o comércio e a indústria. Desde tempos imemoriais que o comércio foi um dos principais motivos de deslocações entre comunidades. Um longo caminho foi percorrido, até hoje, desde os tempos em que os mercados das cidades eram um ponto de reunião de comerciantes, vindos de vários locais para efetuar trocas comerciais. Após o fim das Guerras Napoleónicas, no período sensivelmente entre 1820 e 1870, é comummente aceite que começa a espalhar-se o conceito do livre comércio, num contexto de redução do custo dos transportes (começa o desenvolvimento dos caminhos de ferro). Já na segunda metade do século XIX, como consequência da Revolução Industrial, temos aquela que é considerada como a terceira revolução dos transportes e logística europeus. O aumento da eficiência da máquina a vapor acentua a revolução nos transportes, nomeadamente no caminho-de-ferro.

A história do transporte ferroviário em Portugal começa em 1856, quando foi aberta a primeira linha ferroviária. A rede foi gradualmente expandida tanto ao sul do Tejo, como ao norte do pais, bem como nas áreas metropolitanas de Lisboa e Porto e para a Espanha. A ferrovia foi encarada como um símbolo de progresso, instrumento para unificar um país dividido pelas distâncias e por diversos obstáculos geográficos, ferramenta para aproximar o pais desde a pe-

riferia europeia até ao seu centro [1]. Apesar das diversas alterações de uso e manutenção da linha férrea nacional, em 1976 a concessão das linhas passou totalmente para a administração da CP. Portugal, infelizmente, encontra-se em contra-ciclo com o resto da Europa, com uma rede ferroviária a reduzir de dimensão e a ficar privada de pequenos troços essenciais para a sua conectividade. Apesar de todo o investimento realizado com o objetivo de aumentar e melhorar o transporte de passageiros e mercadorias, estão associadas as normas de segurança e sinalização que preveem aumentar o bom funcionamento das linhas férreas, bem como garantir a inexistência de problemas e acidentes associados.

Assim, contextualizando a proposta anteriormente referida, este projeto incide no desenvolvimento de um simulador de validação para sistemas de sinalização ferroviária. Um *software* de comunicação entre uma interface gráfica (*DEN Depot Zone*), e um *target* final (PLC), isto porque apesar de a interface gráfica já ser uma aplicação utilizada para a verificação de sistemas de sinalização, não atuava no *target* final, o que o *software* desenvolvido ao longo desta dissertação irá permitir (uma comunicação entre a interface gráfica (IG) e o PLC de forma a permitir essas mesmas atividades numa fase mais avançada do desenvolvimento. Logo é importante que o projeto siga o sistema de verificação e validação de sistemas. Verificação e validação são procedimentos independentes que são usados em conjunto para verificar se um produto, serviço ou sistema atende aos requisitos e especificações e se cumpre a sua finalidade, dado a existência de novos *softwares* críticos e seguros *safety-critical software*, *software* cuja falha pode causar perdas humanas. Segundo o glossário *standard* do IEEE aplicado à engenharia de *software*, o processo de verificação e validação é uma fase que determina se:

- Os requisitos para um sistema ou *software* estão completos e corretos;
- Os resultados de cada fase de desenvolvimento cumprem os requisitos ou condições impostos pela fase prévia;
- O sistema ou *software* final cumpre com as especificações impostas.

As atividades de verificação e validação servem para assegurar que o *software* funciona de acordo com o que foi especificado. A verificação tem como objetivo averiguar se o *software* está de acordo com as especificações preestabelecidas, e a validação é o processo de confirmação de que o sistema está apropriado e consistente com os requisitos. [2]

1.2 Objetivos

O objetivo principal deste projeto passa pela implementação de uma interface que permita a simulação de comunicação entre os equipamentos e a sina-

lização que opera no final dos sistemas alvo, nomeadamente no controlador lógico programável (PLC). O protocolo de comunicação terá impacto na verificação da sinalização, pelo que deve estar em conformidade com as normas ferroviárias em vigor, nomeadamente o EN50128 [3] (padrão para sistemas de software). Assim, fazem parte deste projeto os seguintes objetivos:

- Entender os princípios de prototipagem.
- Entender os princípios das normas ferroviárias, nomeadamente a norma EN50128.
- Entender os princípios dos produtos de sinalização.
- Implementar um protocolo de comunicação que providencie uma interface entre os equipamentos de simulação e o sistema final: Uma vez que já existe uma aplicação utilizada para a verificação de sistemas de sinalização, este ponto visa a introdução de um protocolo de comunicação entre uma interface gráfica e o *target* final (PLC), de forma a permitir essas mesmas atividades numa fase mais avançada do desenvolvimento do *software*.

1.3 Empresa EFACEC

A Efacec (Figura 1) criada em 1948, é uma empresa portuguesa que opera nos setores da energia, da engenharia e da mobilidade, conhecida nos dias de hoje como a Efacec Power Solutions. A EPS foi constituída em 14 de Agosto de 2014, tendo como objetivo a gestão de participações sociais como forma indireta de exercício de atividades económicas. No final de 2014 a EPS passou a constituir, ela própria, um grupo de empresas que reúne todos os meios de produção, tecnologias e competências técnicas e humanas para o desenvolvimento de atividades nos domínios das soluções de Energia, Engenharia, Ambiente, Transportes e Mobilidade Elétrica, abrangendo ainda uma vasta rede de filiais, sucursais e agentes espalhados por quatro continentes. Por outro lado, a empresa assenta sobre valores como fiabilidade, sustentabilidade, competência, audácia e humanismo.

Figura 1.1: Logótipo e slogan da empresa. [4]

.

1.4 Organização do Relatório

No primeiro Capítulo é feita a apresentação do trabalho, dando introdução ao tema e ao contexto envolvente. É também apresentada a empresa EFACEC, onde o estágio foi realizado, os objetivos do trabalho e a estrutura do relatório.

No segundo Capítulo é feita a apresentação do estado da Arte. Este aborda as normas ferroviárias, a sinalização das linhas férreas, bem como protocolos de comunicação industrial e mais especificamente o protocolo que será usado no projeto.

No terceiro Capítulo é apresentada a arquitetura do sistema, bem como a seleção do *hardware* e *software* necessários para a concretização do projeto.

O quarto Capítulo é composto pela arquitetura final criada que soluciona o problema em questão, apresentando o conceito central do trabalho e a implementação do sistema.

O quinto Capítulo faz referência aos testes e resultados finais realizados ao *software* desenvolvido.

O sexto e último Capítulo faz o balanço do projeto, apresentado as suas conclusões e as principais dificuldades sentidas bem como possíveis melhorias.

.

Capítulo 2

Estado da arte

Neste capítulo é apresentada uma visão geral das tecnologias envolvidas neste projeto, bem como a apresentação de soluções, existentes no mercado, na área de sinalização ferroviária e encravamento. Com este capítulo é possível compreender as tecnologias em questão bem como oferecer uma visão alargada sobre as soluções já desenvolvidas, ambiente em que estas se inserem e também a escolha dos equipamentos e protocolos de comunicação que mais se adequam ao projeto em questão.

2.1 Evolução da Indústria Ferroviária

No século VI a.C., na Grécia Antiga, apareceram vestígios da primeira via-férrea. No entanto, o conceito nessa altura não assentava numa via com duas linhas de ferro. Numa civilização em que o trabalho escravo aparecia associado às mais diversas funções, o primeiro transporte com semelhança a uma carruagem de transporte era puxado por escravos como força de tração, através de sulcos de calcário, que funcionavam como carris [5]. Algumas das vias, construídas por volta de 600 a.C., eram muito ambiciosas. Nessa época, os habitantes da antiga Corinto, construíram Diolkos (Figura 2.1), que é considerado por muitos como o primeiro caminho de ferro (CF) do mundo. Diolkos, é um exemplo extenso, servindo de ligação entre o Golfo de Corinto e o Golfo de Salónica que evitou a longa e perigosa viagem, por mar, em torno de Peloponeso **??**.

Apesar dos primeiros vestígios reportarem à época da Grécia Antiga, outras áreas e necessidade aguçaram o engenho e deram o seu contributo para o desenvolvimento e aparecimento do verdadeiro sentido do CF.

Foi o caso da indústria da mineralização [5]. Esta indústria dos tempos ancestrais teve grande importância e relevo, nomeadamente a mineralização de

Figura 2.1: Turway Diolkos, Grécia. [5]

.

ouro no tempo do Império Romano no noroeste de Espanha e em Portugal. Contudo, foi entre 1603 e 1604 que, oficialmente, em Inglaterra, através de uma parceria, que foi construído o primeiro *wagonway* (WW) [6]. Este foi considerado um passo significativo para o desenvolvimento dos CF, tendo circulado em duas milhas de trilhos feitos em madeira, entre Strelley e Wollaton, com o intuito de auxiliar o transporte do carvão. Mas o desgaste dos trilhos de madeira era muito grande à medida que o peso transportado aumentava. Assim, começaram-se a reforçar as vias de madeira com barras de ferro, de forma a reduzir custos de manutenção e facilitar a passagem de vagões. A primeira linha de transporte público de mercadorias era operada com WW e recurso à força de cavalos. Mais tarde, a *Middleton Railway* tornou-se no primeiro CF comercial a usar locomotivas a vapor. A invenção de trilhos, do motor de vapor e da locomotiva, apontaram, para a época, uma revolução no sistema de transporte, já que o transporte se operava com recurso à tração animal. Toda esta conjuntura fez com que a Grã-Bretanha fosse considerada a pátria da ferrovia [7]. Nos primeiros anos de operação das locomotivas a vapor, estas apenas eram utilizadas como força motriz em superfícies planas e subidas, enquanto para descidas era explorada a inércia. Nesta altura, já se observava a classificação dos passageiros, uma vez que alguns destes eram figuras notáveis, que seguiam num vagão apelidado de "*The Experiment*". O *boom* da via-férrea (VF) começa a ocorrer um pouco por todo o mundo. Na Europa continental, a primeira VF é construída e aberta à exploração, em Junho de 1827. Nos EUA iniciou-se a construção de uma VF, tendo sido aberta em 1830. Para muitos céticos, esta não era uma oportunidade de progresso e de desenvolvimento, mas antes uma ameaça aos empregos, aos ritmos e modos de vida. Contudo, o baixo custo do novo sistema de transporte permitiu o crescimento económico generalizado, estimulando a produção e o fornecimento de mercados, quer internos quer externos. O aparecimento das redes europeias é claramente assente na experiência britânica, cujo *know-how* era altamente evoluído. É na década de 80 (século XIX) que surge o primeiro proto-

colo entre estados que uniformiza as normas técnicas para a construção de VF, criado em 1887, revisto e aprovado em Berna em 1907, na Conferência Internacional de Berna. Desde os anos 50, que o desenvolvimento ferroviário ajudou à mudança de vários climas políticos e revoluções. Novas exigências económicas emergiam com a necessidade de se reduzirem os custos de transporte das matérias-primas provenientes das minas, bem como da produção das fábricas e aumento da manufaturação para venda no mercado interno e externo [7].

2.1.1 Indústria Férrea em Portugal

Com a Revolução Industrial (RI) , industrias como a extração mineira e a metalurgia ganharam relevo, revolucionando o mundo, no século XIX. Nesta época surge o vapor e o consequente comboio e o CF. Portugal não foi exceção. Todavia, não foi um processo simples pois existia uma grande resistência na aceitação das vantagens que este traria [8]. Com a modernização da *Companhia de Caminhos de Ferro Portugueses*, em 1950, ocorreu a eletrificação das linhas urbanas de Lisboa, mais precisamente na Linha de Sintra, uma vez que a linha de Cascais já tinha sido eletrificada em 1926. Esta será uma das mais antigas linhas em exploração comercial de passageiros, em Portugal, datada de 1887, a par da Linha de Cascais, data de 1889 [9]. Em 1997, a rede ferroviária é liberalizada, passando a ser gerida por uma entidade independente e partilhada por vários operadores. Desta forma, é criada a *Rede Ferroviária Nacional* (REFER), empresa que faria a gestão da infraestrutura, que atualmente, após a fusão com a Junta Autónoma das Estradas se denomina *Infraestruturas de Portugal* (IP). Antes da liberalização, apenas existia a CP (*Comboios de Portugal*), (Figura 3), que estava dividida em CP Passageiros e CP Carga. Após a reorganização e recente venda da CP Carga (agora MedRail) a CP é estruturada em:

- Comboios Urbanos de Lisboa, Porto e Coimbra;
- Comboios Alfa Pendular, Intercidades e Internacional;
- Comboios Regionais e InterRegionais.

Figura 2.2: Logótipo da empresa. [10]

.

As primeiras locomotivas a *diesel* a serem introduzidas em Portugal datam de 1948, de origem americana [8]. Entre 1988 e 2009, Portugal perdeu 43% dos passageiros de comboio. Se em 1988 foram realizadas 231 milhões de viagens, em 2009 este número reduziu para 131 milhões. O número de passageiros por quilómetro percorrido também baixou. Entre 1990 e 2008 a quota de mercado do transporte ferroviário caiu 66%. Entre as causas para a perda de passageiros em Portugal está o encerramento de linhas complementares, que alimentavam as linhas principais, a falta de investimento no setor e o investimento excessivo na construção de estradas e autoestradas [11]. Todo o sistema ferroviário implica a implementação de sinalização e encravamentos que visam garantir a segurança e bom funcionamento dos transportes e do sistema em si.

2.2 Encravamento e Sinalização Ferroviária

Desde o inicio do século XX que o sistema de encravamento e sinalização ferroviária entrou em uso para proteger contra falhas de maquinistas para parar perante um sinal de perigo ou em caso de excederem a velocidade permitida. Estes sistemas foram desenvolvidos a partir de sistemas mais simples que inicialmente apenas forneciam uma verificação no caso de um comboio já ter passado um sinal vermelho. Durante o século XX, as tecnologias de sinalização mecânica foram substituídas progressivamente pela sinalização elétrica e mais tarde pela tecnologia micro-eletrónica. Funções adicionais e sofisticadas foram adicionadas ao longo do tempo, mas os princípios de sinalização ferroviária e encravamento permaneceram inalterados em relação aos estabelecidos nos primeiros anos de implementação.

O sistema ferroviário, do ponto de vista técnico, é composto por três segmentos principais (Figura 2.3):

- Infraestrutura de via onde se move o tráfego ferroviário.
- Veículos ferroviários que incluem as unidades móveis para o transporte de passageiros e mercadorias.
- Sistemas de sinalização e encravamento na via que garantem uma maior segurança.

Estes três componentes compõem a operação ferroviária como um todo, juntamente com as suas regras e processos associados [12].

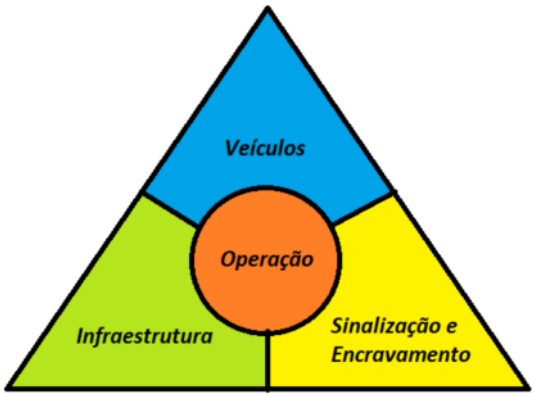


Figura 2.3: Esfera global do sistema ferroviário. Adaptado [12]

.

2.2.1 Princípios de Encravamento

O encravamento ferroviário cumpre a função de "processamento de informações". O encravamento é a função central que garante tecnicamente que um comboio se move de forma segura. De forma a alcançar esta medida, o sistema de encravamento obtém informação sobre a ocupação da linha, bem como a posição de elementos móveis. Assim, avalia a informação adquirida e permite o movimento via sinalização. Entre outros, é preciso garantir dois princípios base:

- Um sinal apenas permite o movimento de um comboio se todos os elementos móveis estiverem na posição certa e bloqueados (dependência entre pontos e sinais), e os elementos móveis têm de permanecer bloqueados enquanto forem usados pelo comboio.

- A um comboio apenas é permitido a entrada numa dada secção se esta estiver livre, e assim nenhum outro comboio deve ter permissão de entrar nessa mesma secção atribuída.

A todos os diferentes sistemas de encravamento - mecânico, relé, eletrónico, são aplicados os mesmos princípios lógicos. Os princípios de encravamento estão ligados às filosofias de operação e baseados no seu desenvolvimento histórico. Assim podem ser divididos segundo a sua influência, seja ela britânica, germânica ou americana. Em contraste com outros métodos de transporte, as ferrovias têm duas características decisivas que determinam os requisitos de segurança:

- O atrito é reduzido, logo as distâncias de travagem são maiores. Isto reduz a capacidade do condutor prevenir colisões com outros comboios, unidades móveis ou outro obstáculo externo.

- Os comboios são guiados pelos carris, assim a sua manutenção é importante para garantir a segurança.

Assim é possível concluir que seguindo as funções de proteção das ferrovias é necessário incorporar (Figura 2.4):

- Proteção contra movimentos sequenciais;
- Proteção contra movimentos opostos;
- Proteção contra movimentos de flanco;
- Segurança em elementos móveis;
- Definição de velocidades permitidas;
- Supervisão e controlo de velocidade;
- Proteção em passagens de nível;
- Proteção contra objetos externos.

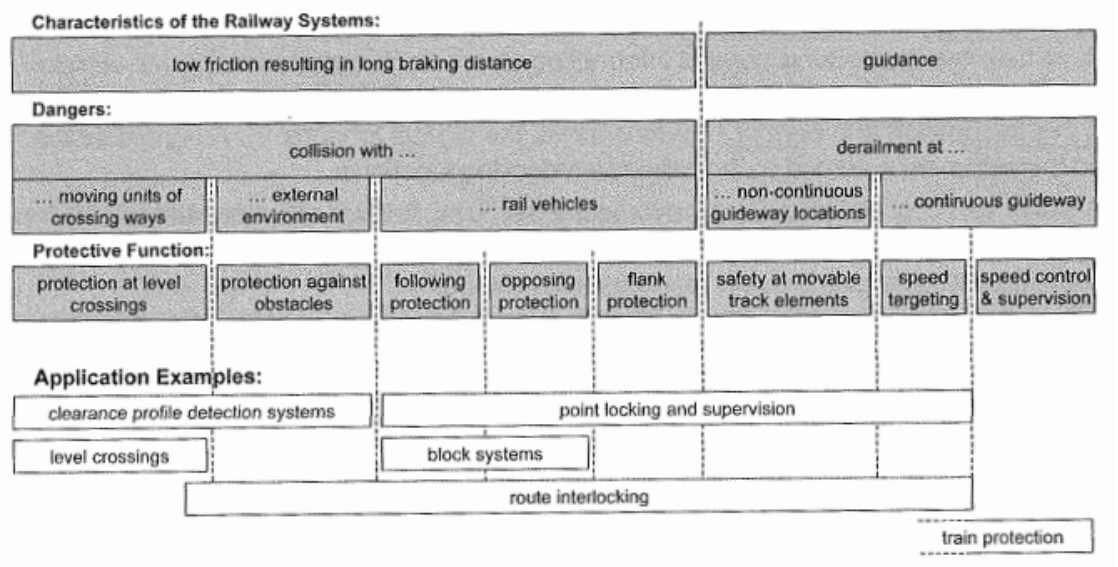


Figura 2.4: Medidas de segurança e perigo em operações de ferrovia. [13]

.

No que diz respeito aos princípios básicos de protecção de itinerário existem no que toca a métodos de protecção, podemos destacar:

- Itinerário - Todo o itinerário incluindo as posições dos elementos de via móvel e elementos de deteção apenas são verificados com pedido, normalmente quando é definido o itinerário antes de libertar a sinalização (esta libertação é supervisionada até o comboio entrar no itinerário). Normalmente estes métodos seguem o comboio em medida de proteção e segurança de flanco em elementos móveis, mas também contribuem para a velocidade de direcionamento em elementos de via móvel. Pode ainda incorporar passagens de nível e deteção de obstáculos.

- Informação de Bloqueio - Após um comboio libertar uma secção, a mensagem de confirmação é gerada e transmitida para o ponto de entrada da secção. Assim a informação é armazenada de forma a permitir que mais tarde seja possível a passagem de um outro comboio. Este princípio apenas fornece proteção oposta. Pode ainda incorporar também passagens de nível e deteção de obstáculos.

Historicamente, a diferença surge na deteção do itinerário pela observação de sinalização. Como tal é também importante falar das dependências dos elementos. Dependências entre elementos individuais são a forma mais simples de bloqueio. Estes elementos podem ser tanto passagens de nível, como sinalização, como elementos móveis. Assim, de acordo com os arranjos lógicos, as dependências dos elementos podem ser divididas nas seguintes funções de bloqueio elementares:

- Acoplamento de dois ou mais elementos

- Bloqueio Unidirecional

- Bloqueio Bidirecional, podendo este ser entre dois elementos e três ou mais elementos.

2.2.1.1 Acoplamento de Elementos

Elementos acoplados são operados pelo mesmo elemento operacional e só podem ser comutados em operação regular. O caso mais típico é o de dois elementos móveis dando protecção de flanco um ao outro, como dois conjuntos de pontos de um cruzamento ou um conjunto de pontos e uma protecção de descarrilamento desse mesmo conjunto de pontos (Figura 2.5).

Este tipo de bloqueio é comum na lógica de bloqueio influenciada pelos britânicos e norte americanos, mas não o germânico. Muitas vezes, ambos os conjuntos de pontos acoplados têm o mesmo número de identificação, normalmente distinguidos pelos sufixos A e B. A desvantagem é que se um dos elementos falhar, ambos não podem ser comutados pelo comando de operação normal. Ou-

tra desvantagem é o bloqueio mecânico associado: o operador de via precisa do dobro do esforço para alterar o estado de ambos.

Figura 2.5: Comutação de dois elementos acoplados. [13]

.

2.2.1.2 Bloqueio Unidirecional

No bloqueio unidirecional de dois elementos, existe um elemento independente e um elemento dependente. O elemento independente pode ser movido livremente enquanto que o elemento dependente apenas pode ser definido para uma determinada posição se o elemento independente também estiver numa determinada posição e sai desta imediatamente após o elemento independente também sair da sua posição. Com mais do que dois elementos, a posição do elemento dependente depende da combinação das posições dos elementos independentes. O caso mais comum é a dependência entre o sinal principal e o sinal distante associado (Figura 2.6). Como a capacidade de alterar o estado do sinal principal para o estado de *Stop* em qualquer altura deve ser preservada por motivos de segurança, o sinal principal pode ser comutado sem considerar a posição do sinal distante. Contudo o estado do sinal distante está sempre dependente do sinal principal.

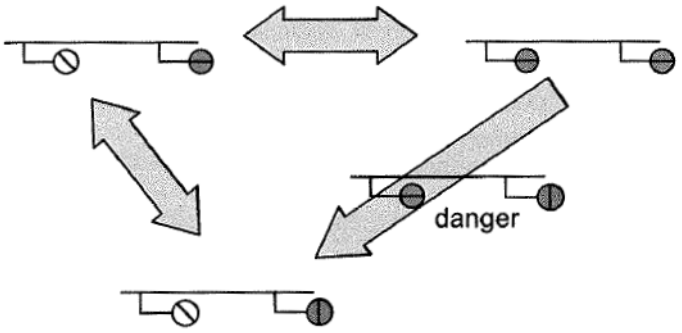


Figura 2.6: Dependência unidirecional entre o sinal principal e o sinal distante. [13]

.

2.2.1.3 Bloqueio Bidirecional Simples

No bloqueio bidirecional, dois ou mais elementos são bloqueados de forma a que uma combinação de posições seja impossível e cada elemento seja bloque-

ado se os outros estiverem numa respetiva posição. O exemplo mais comum é o caso de dependência entre posições e sinalização (Figura 2.7). O sinal é bloqueado na posição de paragem se os pontos divergirem conforme o exemplo. Assim os pontos são bloqueados na posição "reta" se o sinal for libertado. Isto significa que uma certa combinação de posições é impossível. Os elementos devem alterar o seu estado numa sequência definida. Primeiro as posições tem de estar na posição de "passagem" e só depois é que o sinal é libertado. Este tipo de bloqueio também é chamado de bloqueio sequencial.

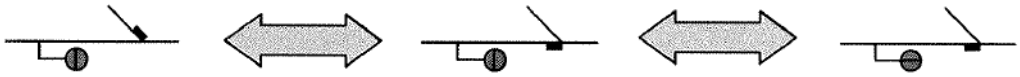


Figura 2.7: Dependência sequencial entre pontos e sinalização. [13]

.

O mesmo principio é aplicado a pontos e descarriladores (Figura 2.8) ou a dois conjuntos de pontos que oferecem protecção de flanco um ao outro, particularmente na lógica de bloqueio alemã. A lógica é a mesma que a anterior, a diferença é que elementos têm de ser alterados em primeiro e segundo lugar. Em sistemas mecânicos, o bloqueio sequencial pode resolver-se através de fechaduras manuais, com um destravamento num ou noutro elemento de posição bloqueado.

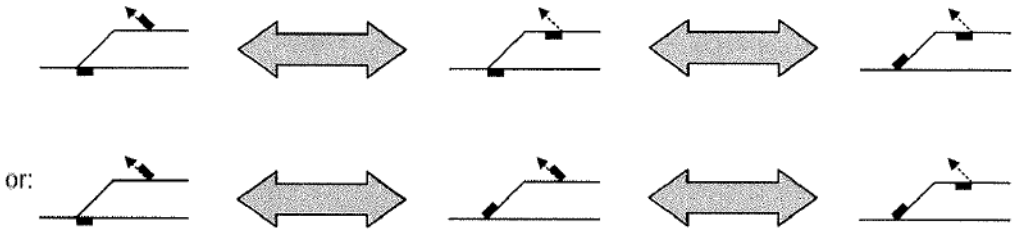


Figura 2.8: Dependência sequencial entre pontos e descarrilador. [13]

.

2.2.2 Itinerário

Os itinerários são uma forma de salvaguardar um percurso completo de um comboio ao longo de todo o comprimento, ao longo no qual o movimento é permitido. Uma das bases fundamentais de um itinerário incide nas dependências entre elementos, entre pontos e sinais, mas também na verificação de secções antes de permitir o movimento (pré-condição importante). Estas dependências contribuem para as seguintes funções de proteção:

- Prevenção de descarrilamento em locais de guias não contínuas: Elementos móveis devem estar nas suas posições.

- Prevenção de descarrilamento em locais de guia contínua pela escolha da velocidade adequada de acordo com a geometria da linha e dos raios na divergência.

- Proteção dos movimentos de flanco.

- Proteção opcional nas passagens de nível, caso estas estejam incluídas no itinerário.

- Proteção opcional contra obstáculos, se esses sistemas de deteção estiverem incluídos nas dependências da rota.

Estas funções também podem incluir uma certa proteção contra erros humanos.

2.2.2.1 Definições de Termos

Quando todos os elementos móveis da via estão nas posições adequadas para que um comboio possa tomar uma determinada direção, este caminho é chamado de itinerário. Antes de permitir qualquer movimento, certas pré-condições devem ser comprovadas. As pré-condições mais importantes são:

- Os elementos móveis devem estar nas posições corretas e bloqueados para que não possam ser trocados enquanto o comboio passar sobre o elemento.

- Os caminhos tem de estar desimpedidos e ao mesmo tempo, prevenir movimentos de conflito.

- Os itinerários precisam de um alvo distinto.

O itinerário pode ser salvaguardado tanto por questões técnicas como não técnicas sobre a responsabilidade de operários. Um caminho tecnicamente protegido com um ponto de entrada e um ponto de saída definido é chamado de itinerário. Em países onde os movimentos dos comboios são distinguidos como diferentes classes de movimento, os itinerários podem ser distinguidos como itinerários de comboios e itinerários de manobras. Nos casos em que são fornecidos itinerários de manobra separados, as suas funções de encravamento são geralmente menos complexas do que as utilizadas nos itinerários de comboios em operação normal.

2.2.2.2 Extensão de Itinerário e Restrições de Velocidade Associadas

As partes do itinerário que devem ser salvaguardadas são (Figura 2.9):

- O caminho - as vias que são regularmente ocupadas pelo comboio durante a sua travessia do itinerário. Inclui secções com e sem elementos móveis.
- A sobreposição - que se pode estender por uma secção curta para proteger contra uma pequena ultrapassagem do comboio causada por um erro de avaliação de travagem ou que pode até acomodar toda a distância de travagem da velocidade máxima até a paragem total.
- Áreas de flanco - que protegem o comboio contra movimentos de flanco não autorizados.
- O fim da secção de proteção - que protege contra movimentos opostos.
- Secção inicial - situada na retaguarda do sinal de entrada do itinerário, mas ocupada pelo comboio ou situada entre a cabeceira do comboio e o sinal de entrada do itinerário.

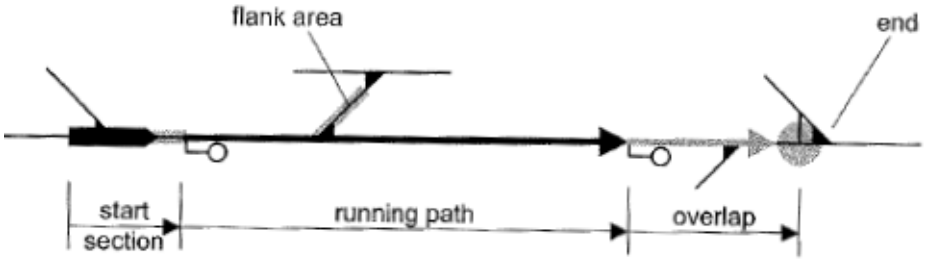


Figura 2.9: Partes lógicas de um itinerário. [13]

.

A consideração prática destas partes de encravamento diferem entre ferrovias. Embora o bloqueio do caminho seja essencial, existem grandes diferenças em relação a outras partes:

- Algumas ferrovias não fornecem sobreposições, argumentando que bons sistemas de protecção de comboios evitam invasões de linha com segurança.
- Outras ferrovias não fornecem protecção de flanco particular, argumentando que todos os movimentos assumem que movimentos de flanco não podem ocorrer.
- A protecção frontal é considerada em diferentes graus.

- O encravamento de secções iniciais é importante apenas se houver elementos móveis dentro desta secção.

2.2.3 Encravamento de Itinerários Contraditórios

O conflito de itinerários consiste em diferentes itinerários que usam uma parte comum da infraestrutura. O encravamento de dois métodos para conflito de itinerários pode ser usado da seguinte maneira (Figura 2.10):

- **Encravamento de itinerário simples**: Itinerários que exigem que pelo menos um elemento móvel esteja numa posição diferente são encravados uns com os outros pela posição desse mesmo elemento, não necessitando de encravamento separado. Estes encravamentos são aceites pela maioria das ferrovias. Caso contrário, o encravamento especial deve ser fornecido em todos os casos.

- **Encravamento de itinerário especial**: Conflito de itinerários que não diferem na posição de qualquer elemento móvel da via devem ser encravadas umas às outras por meios especiais. Este método requer esforço adicional na lógica de encravamento.

Os casos mais importantes de encravamento particular são (Figura 2.10):

- **O caso contrário**: Os itinerários opostos na mesma via devem sempre ser excluídos. Para movimentos de manobra e movimentos de comboios, os requisitos diferem entre países e dependem do comprimento da secção da via. Em alguns casos os movimentos são excluídos, enquanto outros visam obter uma operação mais rápida na montagem dos comboios. Uma solução para cumprir esta função de encravamento é o encravamento individual das vias, outra é o uso de encravamento separado para cada elemento móvel.

- **O caso consecutivo**: Em algumas situações técnicas e operacionais, itinerários consecutivos precisam de se interligar novamente para prevenir através de movimentos o uso de determinada secção. Os requisitos também diferem para um itinerário de um comboio e um itinerário de manobra consecutiva. Em algumas situações e por algumas ferrovias, a exclusão é necessária para evitar que uma unidade móvel que entrou como um comboio continue sem parar como um movimento de manobra. Um exemplo típico é o de um itinerário de receção para comboios de carga. Em outros casos, e por diferentes razões, uma certa sequência de configurações dos itinerários.

- **O caso do perfil de cruzamento**: Mesmo que a travessia em diamante não tenha uma parte móvel, os movimentos em ambas as pistas ao mesmo tempo devem ser evitados. O mesmo se refere a qualquer situação em que os perfis de folga de vias vizinhas se sobreponham.

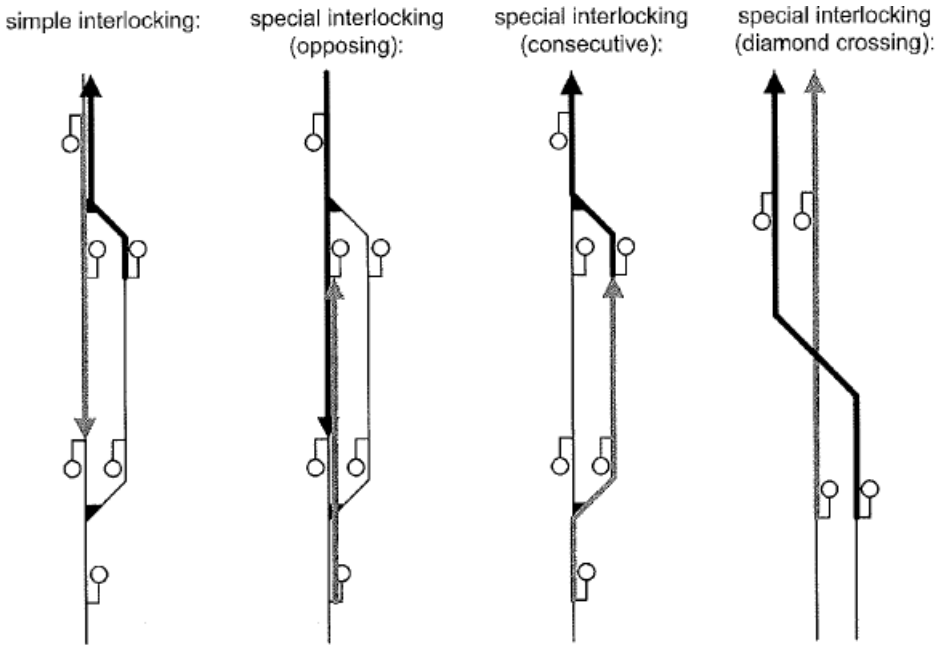


Figura 2.10: Encravamento de Itinerário Simples e Especial. [13]

2.2.3.1 Outras Funções de Encravamento

Além das funções básicas mencionadas, várias outras funções de encravamento são fornecidas por algumas ferrovias. Algumas delas estão relacionadas com a segurança, outras servem apenas para evitar entraves operacionais. A maioria das ferrovias utiliza alguma forma de prevenção de libertação prematura de vias devido a erros de deteção de desobstrução da via, que resultam em consequências perigosas. Isto é particularmente importante quando são usados circuitos de via, pois a probabilidade de falsa ocupação é relativamente alta. Diferentes estratégias são aplicadas (Figura 2.11):

1. A via só pode ser libertada após a ocupação sequencial e a desobstrução de pelo menos dois troços de via vizinhos, na ordem adequada.

2. É usada uma combinação de um contacto ferroviário (que deteta a ocupação com segurança) com um circuito de via curta (que deteta a libertação com segurança).

3. Atrasos temporais são aplicados entre a passagem do comboio e a libertação da secção do itinerário relacionado. Isto apenas evita consequências perigosas de erros na deteção de curta duração, mas não erros duradouros.

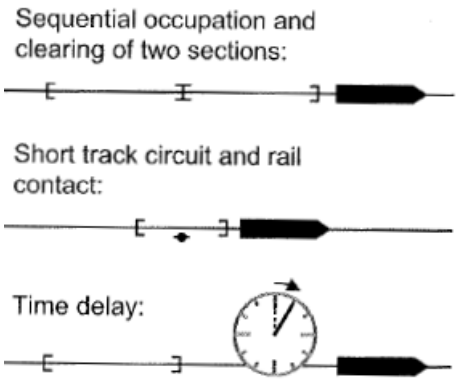


Figura 2.11: Métodos para evitar a libertação prematura de um itinerário. [14]

2.2.4 Proteção de Flanco

Movimentos perigosos de flanco podem originar-se através dos seguintes tipos de atividade:

- Comboios e manobras em rotas de conflito. Isto é excluído pela lógica de encravamento.
- Desvio sem rota.
- Veículos estacionados.

A maioria das ferrovias fornece protecção de flanco pelo menos para itinerários de comboios. A área entre a via e o elemento de protecção é frequentemente comprovada. Assim a protecção de flanco pode ser dada pelas seguintes medidas de protecção (Figura 2.12):

1. Elementos móveis, como pontos e pontos de captura que direcionem os movimentos ofensivos para fora da rota ou até com a possibilidade de descarrilamento. Isto pode ser resolvido ao encravar os pontos de um cruzamento ou ao incluir elementos de protecção nas funções de encravamento do itinerário.
2. Sinalização bloqueada no estado de paragem. Para permitir a proteção total, a lâmpada de sinalização vermelha deve estar ligada. Com segurança reduzida, algumas ferrovias também aceitam um sinal de cor escura como

proteção de flanco, pois o maquinista é obrigado a considerar esse sinal distinto como sinal de paragem.

3. Regulamentos que impedem movimentos de flanco. Tais regulamentos podem proibir manobras ou estacionamento de veículos em determinadas pistas ou até mesmo proibir manobras em geral.

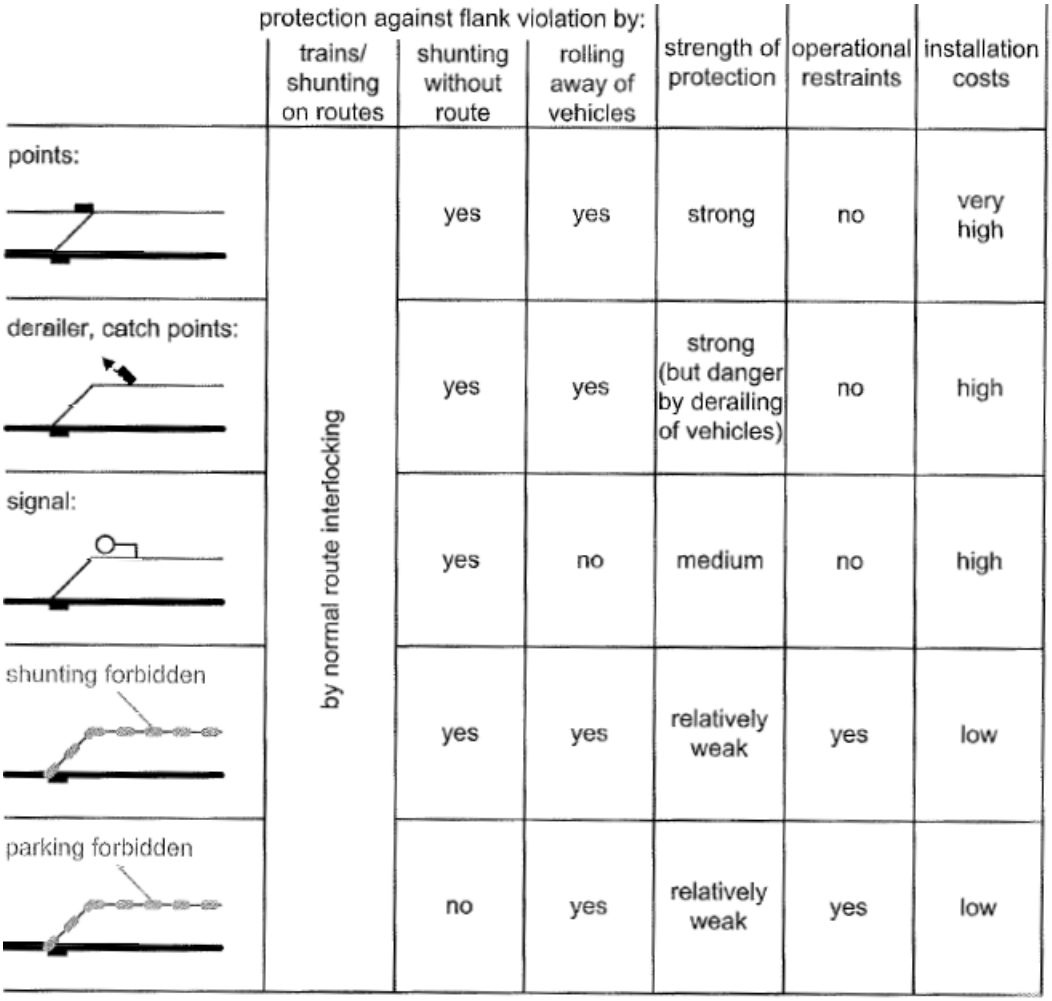


Figura 2.12: Métodos de proteção de flanco. [14]

Destes métodos, o método 1 é o mais forte, enquanto que os outros são mais fracos, uma vez que dependem da condição de obediência às regras por parte das pessoas. Nem todos os tipos de medidas protegem contra todos os tipos de movimentos perigosos. Portanto, muitas vezes é usada uma combinação de dois métodos, um sinal e uma proibição de deixar os veículos na via em questão. Já os requisitos para aplicação e a escolha da proteção de flanco diferem. Uma questão especialmente controversa é a de fornecer ou não proteção de flanco,

pois isso só é relevante se ocorrerem dois erros independentes ao mesmo tempo. Estas são a ultrapassagem do comboio e o movimento não autorizado de outros veículos, o que é muito improvável.

2.2.4.1 Proteção de Flanco Ramificado e por Transferência

Se o elemento de pista adjacente àquele que requer proteção de flanco não for capaz de a oferecer, a solicitação é transferida para o próximo elemento mais distante. Assim a área do flanco que deve ser comprovadamente limpa torna-se mais longa. Dois casos típicos são explicados de seguida:

1. **Proteção de flanco transferida (secundária)**: No caso da figura 2.13 se o elemento que normalmente deveria dar proteção de flanco não o puder fazer, ele transfere a solicitação para o próximo elemento. Exemplos de aplicação são solicitações de proteção dupla e elementos de deteção. Se também neste caso o sinal não estiver a funcionar corretamente, a secção B da via pode fornecer proteção de flanco. Isto é implementado em algumas situações. Uma possibilidade adicional é transferir a proteção de flanco mesmo no caso de um elemento de proteção de flanco de rota já ativo ser perturbado.

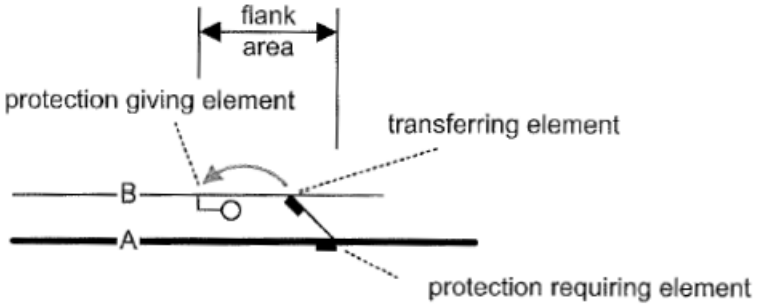


Figura 2.13: Proteção de flanco por transferência. [13]

2. **Proteção de flanco ramificado**: No caso da figura 2.14, o ponto dois não pode dar proteção de flanco para o ponto um devido ao esquema da pista. Portanto o ponto dois deve ser transferido para a solicitação de proteção de flanco em direção a ambas as extremidades. Num ramal, o ponto três pode dar proteção contra um movimento da via C, enquanto que no outro ramal, o sinal quatro pode dar proteção contra movimentos da via B. A área do flanco estende-se, o que significa que o ponto dois também deve ser detetado como livre.

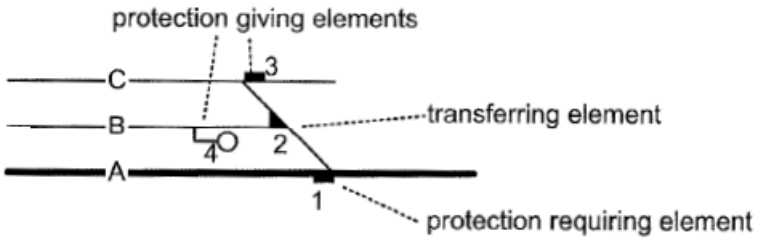


Figura 2.14: Proteção de flanco ramificado. [13]

Um caso particular de proteção de flanco por transferência é o de conflito de solicitações por dois itinerários (ou pelo mesmo itinerário). Casos de requisições de conflito por dois itinerários não consecutivos (figura 2.15a) podem ser resolvidos da seguinte forma, ao se utilizar o principio topológico:

1. O primeiro itinerário solicitado recebe proteção de flanco pelos pontos de proteção duplos.
2. O segundo itinerário solicitado obtém proteção de flanco remota por outro elemento mais distante do itinerário a ser protegido.
3. Quando o primeiro itinerário solicitado (que significa que o itinerário parcial para a qual os pontos de proteção duplo fornecem proteção de flanco) foi libertado, os pontos de proteção duplos podem alternar e dar proteção de flanco para o outro itinerário. O recurso e implementação desta opção difere entre ferrovias e sistemas.

Alguns sistemas de encravamento oferecem a possibilidade de definir prioridades entre as duas solicitações de proteção de flanco. O itinerário prioritário obtém sempre a proteção de flanco dos elementos de proteção duplo, enquanto o outro itinerário obtém proteção de flanco pelo elemento de proteção dupla somente se não houver solicitação do itinerário prioritário. Caso contrário, obtém proteção remota de outro elemento, num local mais distante. Por esta razão, o elemento de proteção dupla não é bloqueado no itinerário não prioritário. Assim, pode ser comutado se uma solicitação de proteção de flanco for emitida pelo itinerário prioritário. No caso de solicitações pelo mesmo itinerário (figura 2.15b) (incluindo variantes e formas mistas) são possíveis os seguintes casos:

- Uma prioridade é definida e a solicitação de proteção de flanco não prioritária obtém proteção remota. A prioridade pode ser definida por diferentes critérios, como o comprimento das áreas de flanco.
- A situação é resolvida dinamicamente. Inicialmente, o elemento de proteção dupla protege o elemento de pista de rolamento que é a primeira

direção em que o comboio se move. Após o comboio ter libertado esta parte do itinerário, o elemento de proteção dupla comuta para fornecer proteção de flanco para o outro elemento da via. Como o tempo costuma ser curto, esta solução é raramente aplicada.

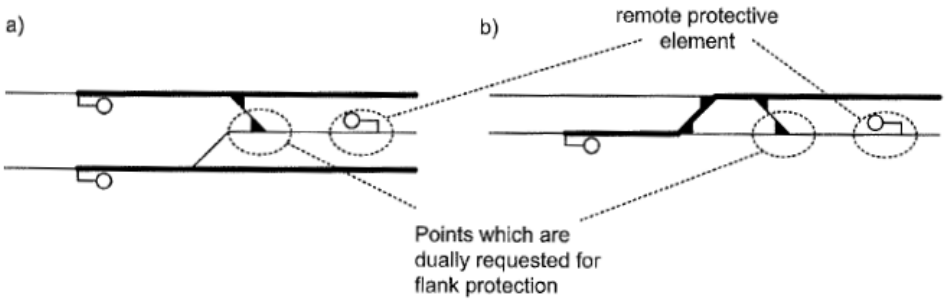


Figura 2.15: Solicitação de proteção de flanco por dois itinerários (caso da esquerda) e pelo mesmo itinerário (caso da direita). [13]

Em casos especiais, as solicitações de proteção de flanco para um elemento de proteção dupla geralmente podem ser transferidas para outro elemento mais atrás para evitar comutação. Isto é particularmente adequando se ambas as solicitações de proteção de flanco ocorrerem com frequência e geralmente alternadamente, para pontos de estacionamento e inversão de vias entre a via principal em ferrovias metropolitanas, com horários de intervalo fixos. Este pode ser o caso descrito na figura 2.15a. Nas ferrovias metropolitanas onde não existem veículos não-tratores estacionados, mas apenas unidades fixas acopladas com pelo menos um veiculo de tração, a perda de segurança ao fornecer proteção de flanco por um sinal em vez de pontos é relativamente pequena.

2.2.5 Princípios de Formação de Itinerário no Plano da Via

Nesta seção, são discutidos os princípios que definem quais os elementos que pertencem a um determinado itinerário. Existem três princípios básicos de formação de itinerário: tabular, cascata e topológico, este último muitas vezes chamado de princípio geográfico. Todos os três princípios são aplicáveis para sistemas com bloqueio irreversível de rota antes e depois da libertação do sinal, embora nem todas as combinações sejam de importância prática.

O **princípio da cascata** era o princípio mais antigo na Grã-Bretanha e na lógica de encravamento da América do Norte em sistemas mecânicos. Nos dias de hoje, só é possível encontrar esta aplicação em instalações mais antigas que usem máquinas de quadro de alavanca. O princípio básico é que cada elemento de via móvel é encravado por elementos de via móveis decisivos ou pelo sinal.

Assim, cada elemento geralmente pode ser libertado individualmente após a passagem do comboio. Devido à sua baixa importância nos dias de hoje, não será descrito em detalhe.

O **princípio tabular** é a forma tradicional originada no encravamento mecânico na lógica de encravamento influenciada pela lógica alemã. Uma das razões históricas para o seu desenvolvimento foi a necessidade de manobras livres (sem sinais de manobra e com pontos livremente comutáveis), na Alemanha, o que não era possível com o bloqueio em cascata. Este princípio é ainda amplamente aplicado em encravamentos mecânicos, de relé e eletrónico em todo o mundo. Todas as rotas possíveis são predefinidas numa matriz, indicando exatamente que elementos pertencem à respetiva rota e em qual posição. É possível analisar um exemplo através da figura 2.16 e da tabela 2.1. As rotas que não podem estar ativas ao mesmo tempo também são predefinidas e distinguidas entre exclusões de itinerários simples e especiais.

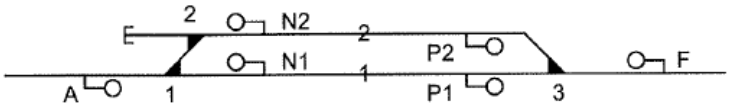


Figura 2.16: Plano para a matriz de bloqueio de itinerário, segundo a matriz da tabela 2.1. [13]

Tabela 2.1: Matriz de bloqueio de itinerário (exemplo). Adaptado [14]

Routes	Conflicting Routes								Points			Flank Protection Signals		
	A/1	A/2	N1	N2	P1	P2	F/1	F/2	1	2	3	N1	P1	P2
A/1		|	||	|			||		+	+				
A/2	|		|	||		||		||	-	-		S		
N1	||	|		|					+	+				
N2	|	||	|					||	-	-		S		
P1						|	||	|			+			S
P2		||			|		|	||			-		S	
F/1	||				||	|		|			+			S
F/2		||		||	|	||	|				-		S	
A/1 : Route beginning at signal A into track 1														
N1 : Route beginning at signal N1 (if there is only one route from this signal)														
+ : Point in normal (here straight) position														
- : Point in reverse (here: diverging) position														
S : Signal at Stop														
| : Simple exclusion of routes by movable track element in different positions														
|| : Special exclusion of routes														

O **princípio geográfico** apareceu pela primeira vez na Alemanha na década de 50 (1950), foi desenvolvido para o encravamento de relés e, até hoje, é usado internacionalmente para relés de encravamento eletrónico em larga escala.

Os elementos da via são definidos com as suas relações de vizinhança entre si (Figura 2.17).

Quando um determinado itinerário tem de ser definido, o caminho, a sobreposição e a proteção de flanco são pesquisados por essas relações topológicas. Portanto, todos os itinerários possíveis de acordo com o plano da via podem ser selecionados automaticamente, a menos que sejam deliberadamente suprimidos.

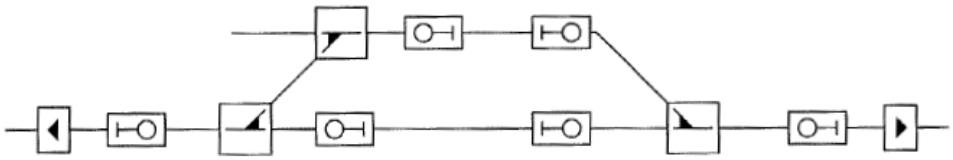


Figura 2.17: Mapa de conexão do conjunto de relé simplificado para o encravamento topológico. [14]

À medida que as características, vantagens e desvantagens do princípio topológico e tabular se tornam mais óbvias no encravamento de relés, elas são comparadas primeiramente com a tecnologia de relés (tabela 2.2).

Tabela 2.2: Comparação entre os princípios tabular e geográfico. Adaptado [13]

	Princípio Tabular	**Princípio Topológico**
Esforços para o desenvolvimento do sistema de relés	Relativamente baixo	Alto
Esforços de equipamentos de retrasmissão por áreas de encravamento	Baixo	Alto (funções centrais)
Esforços de equipamentos de retransmissão para cada unidade de comutação adicional	Alto	Relativamente baixo
Esforços para mudanças posteriores no encravamento do relé	Relativamente alto	Relativamente baixo
Esforços para mudanças posteriores no encravamento eletrónico	Alto	Alto
Adaptabilidade a casos especiais	Normalmente fácil	Fácil
Aplicação eficiente para ...	Estações peqenas (relé) / baixo número de aplicações do sistema (relé/eletrómnico)	Estações grandes (relé) / grande númerop de aplicações do sistema (relé/eletrónico)

Em sistemas topológicos de encravamento de relés, os relés são posicionados em conjuntos de relés, cada um representando um determinado elemento no plano da via. Os conjuntos de relés são padronizados e podem ser fabricados e testados automaticamente nas fábricas. Isto reduz a necessidade de cablagem no local de implementação ao mínimo de conexões no conjunto de relés por

cabos padronizados. Isso facilita também os processos de ligações e testagem, bem como a adaptação do encravamento às alterações no plano da via. Esta é uma vantagem sobre o princípio tabular, quanto mais elementos forem incluídos na área de encravamento. Por outro lado, uma desvantagem do princípio topológico é o grande esforço a ser colocado nos conjuntos de relés. A cablagem associada dentro dos conjuntos é geralmente mais complicada e, devido aos conjuntos de relés padronizados, um grande número de relés que não são necessários para o caso especifico devem estar fisicamente presentes.

No encravamento eletrónico, algumas das vantagens e desvantagens de ambos os princípios perdem mais importância. Portanto, ambos os princípios são adequados para pequenas e grandes áreas de encravamento. Permanece assim, uma diferença quanto ao número de instalações de um determinado tipo. O desenvolvimento de um novo tipo de encravamento é mais complexo com o princípio topológico, mas uma vez desenvolvido, projetar uma nova instalação é mais simples. Juntos, ambos os princípios são aplicados no encravamento eletrónico, diferenciados por fabricantes e tipos de encravamento.

A figura 2.18 ilustra o esforço necessário para cada princípio na tecnologia de relés, dependendo do tamanho da estação/junção a ser encravada. Analisando a figura 2.18, o princípio tabular é vantajoso para esquemas menores, enquanto o princípio topológico ganha com maiores áreas de encravamento.

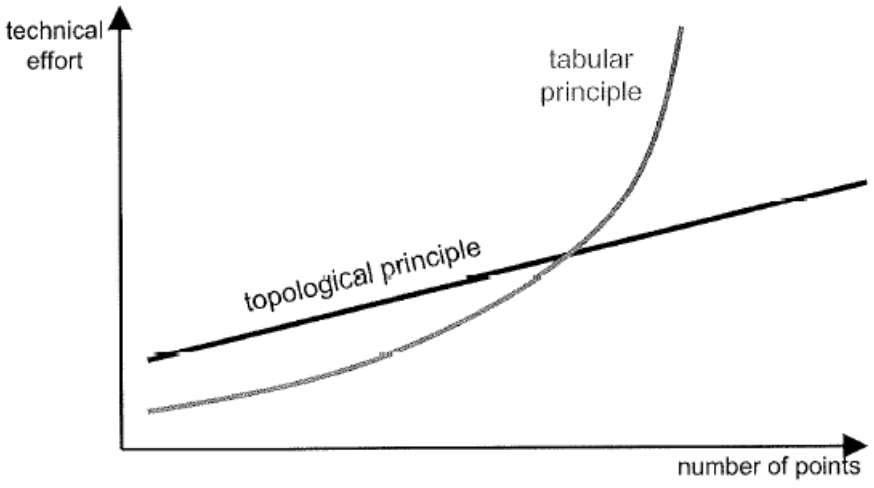


Figura 2.18: Esforço técnico para o princípio tubular e topológico no encravamento de relés. [13]

2.3 Protocolos de Comunicação Industrial

As redes de comunicação industrial são usadas para a monitorização e controlo de fabrico e aplicações de controlo de processos. Estas diferem das redes tradicionais de uma forma fundamental: parte das mensagens transferidas pela rede têm de satisfazer restrições de tempo com o perigo de danos e perdas de vidas humana se essas restrições não forem atendidas **??**. A evolução das redes industriais passou pelo avanço de possibilidades de topologias, velocidades de comunicação, aumento de *throughput*, redução de latência e, para o caso de tecnologias baseadas em Ethernet, as garantias de envio e recebimento de dados.

2.3.1 Profinet

O protocolo Profinet é um protocolo que faz uso da Ethernet física e controlo de acesso ao meio por camadas (MAC). Atualmente, compreende dois protocolos de camada alta diferentes, destinados a empregar diferentes níveis de comunicação, nomeadamente Profinet CBA para o nível alto, Profinet IO para o nível do dispositivo. Destes níveis de comunicação, apenas o Profinet CBA está incluído no IEC61784 [15] como o perfil de comunicação (Profinet) da família de perfis de comunicação, relevante para a rede IEC61158 [16] [17] [18].

Profinet CBA (Automação baseada em componentes) é baseado na definição de um ambiente orientado a objetos no qual os relacionamentos entre objetos (componentes Profinet) definem as variáveis que são trocadas na rede.

Profinet IO permite implementar uma solução mais rápida na transferência de dados entre controladores e dispositivos de campo. O protocolo é projetado para satisfazer as exigências mais criticas ao nível do dispositivo em termos tanto de periodicidade como de tempo real [18].

Dependendo das restrições de tempo, o protocolo Profinet define três tipos de tráfego, conforme é possível analisar através da tabela 2.3:

Tabela 2.3: Valor relativo do *jitter* calculado em relação à periodicidade. Adaptado [14]

Tipo de tráfego	**Periodicidade/Tempos de reação**	**Jitter**
Tempo Não Real (TNR)	$\geq 100ms$	≥ 100
Tempo Real, Classe 1	$\geq 5ms$	≥ 15
Tempo Real, Classe 2	$\geq 250us$	$\leq 0,4(1us)$

Analisando a tabela 2.3, conclui-se que o tráfego do tipo tempo não real (NRT) é normalmente aplicável com Profinet CBA, enquanto tempo real (RT) classe 1 pode tanto ser aplicável com Profinet CBA como Profinet IO, RT classe 2 é apenas aplicável com Profinet IO. Um aspeto importante do Profinet é que

todos os três tipos de tráfego ocorrem na mesma rede física. Portanto, é necessário um gerenciamento de tráfego específico para garantir que as mensagens urgentes (ou seja, aquelas relacionadas a ambas as classes de RT) são entregues primeiro [17]. Para questões de análise e comparação, este documento vai apenas focar-se no Profinet IO, uma vez que incide na camada do dispositivo.

2.3.1.1 Profinet IO

Profinet IO define quatro tipos de dispositivos que podem estar presentes na rede: controlador IO, dispositivo IO, supervisor IO e servidor de parâmetros IO.

Os controladores IO são dispositivos inteligentes como, por exemplo, computadores pessoais ou controladores programáveis que realizam tarefas de automação.

Dispositivos IO são dispositivos que realizam a interface entre os controladores e o campo. Exemplos típicos são: sensores, atuadores, válvulas, terminais eletrónicos, etc.

Os supervisores são dispositivos que trocam dados de configuração e diagnóstico com controladores e/ou dispositivos.

Os servidores de parâmetros são dispositivos usados para trocar dados de configuração relevantes para os aplicativos com os dispositivos.

O tráfego de dados gerado por ambos os supervisores de parâmetros normalmente ocorre durante as fases *off-line* e não implica aplicações em tempo real. Por outro lado, a troca de dados entre controladores e dispositivos está principalmente relacionada com tarefas de automação e, portanto, tem como requisito um tempo crítico. A especificação da camada de aplicação do Profinet IO é baseado no conceito de *Application Service Element* (ASE), definido no modelo de referência *Open System Interconnection* (OSI) para interconexão de sistemas abertos. De facto, um ASE fornece um conjunto de serviços destinados a realizar a comunicação entre aplicativos de processos distribuídos e, portanto, a troca de inscrição de objetos [18].

Através da figura 2.19 é possível analisar o tipo de processo aplicado pelo Profinet IO. O controlador atualiza o *buffer* de saída com a função *Set Output*. Os dados são então disponibilizados para o dispositivo através da saída CR cíclica RT. Tanto as funções *Set Output* como a *Get Output* podem ser tanto assíncronas como síncronas. No caso assíncrono, se a função *Set Output* realizar um pedido antes da função *Get Output*, então, nem todos os valores escritos no *buffer* de saída são adquiridos pelo dispositivo. No caso síncrono, as solicitações dos valores de saída são transmitidos e adquiridos exatamente na altura em que são atualizados. Este comportamento permite implementar a classe síncrona do Profinet IO [17].

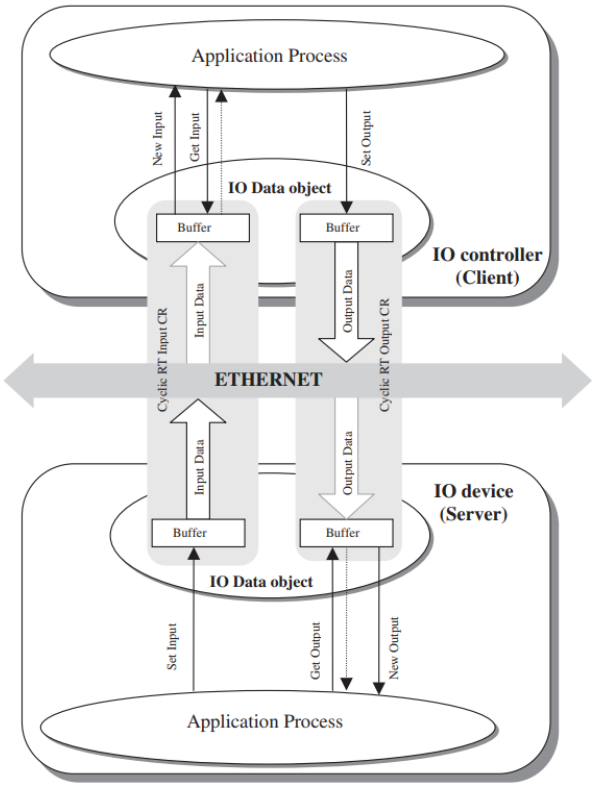


Figura 2.19: Processo de dados ASE. [19]

2.3.2 OPC

Para conhecer o significado da palavra OPC é preciso saber que se trata de uma abreviação para *OLE for Process Control*. Na prática é um conjunto de comunicação de dados para a indústria com base no *OLE*, tecnologia da Microsoft utilizada no sistema operacional Windows, que permite a conexão entre objetos de dados [19]. Esta tecnologia utiliza a interface COM/DCOM, permitindo a troca de dados entre aplicações e dispositivos, por exemplo, um PLC e um sistema SCADA conectados com um OPC. O OPC Clássico possui três especificações:

- **DA (Data Access):** para troca de dados em tempo real.
- **A&E (Alarm and Events):** dados e mensagens de eventos de estados.
- **HDA (Historical Data Access):** dados para análise histórica de eventos.

Contudo o OPC Clássico mostrou-se limitado para alguns desafios [20]. As principais limitações que surgiram foram:

- Problemas de configuração com o DCOM.
- Inexistência de *timeouts* configuráveis.
- Implementação apenas com recurso a sistema Microsoft Windows (Figura 2.20).
- Segurança simples.
- Nenhum controlo sobre o DCOM.

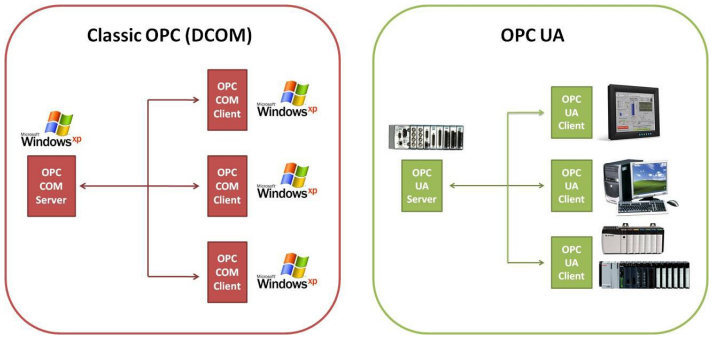


Figura 2.20: Diferença entre OPC Clássico e o OPC UA. [21]

Assim foi necessário desenvolver uma tecnologia que resolvesse estas limitações, surgindo assim o OPC UA.

2.3.2.1 OPC UA

O OPC UA (*Unified Architecture*) foi a tecnologia desenvolvida para suceder ao OPC Clássico, sendo um padrão aberto de comunicação de dados industriais. Ao contrário do OPC Clássico, o OPC UA utiliza um modelo de informação orientado a objetos, que suporta estruturas e máquinas de estado, além de ser independente de um sistema operacional. Além disso, é possível ainda referir algumas vantagens do OPC UA:

- Suporta arquitetura orientada a serviços (SOA) que permite a fácil personalização do OPC UA, para diversos tipos de dispositivos e aplicações.
- Possibilita a troca de dados brutos e informações pré-processadas entre os sistemas incorporados nos sensores e nos dispositivos de campo e os sistemas de ERP, MES e de visualização de processos (IHM).
- Possui segurança robusta de dados.

Em relação à comunicação, o OPC UA suporta dois formatos, UA Binário e XML. O remetente codifica os dados para o formato relevante e o recetor descodifica o conteúdo (com todas as permissões necessárias, bem como a segurança associada) de forma a reconstruir os dados originais. No que toca a protocolos, o OPC UA suporta um protocolo baseado em TCP - OPC/TCP (OLE for Process Control / Transmission Control Protocol), permitindo um canal full-duplex entre o servidor e o cliente, onde a comunicação é realizada com recursos a pacotes binários, permitindo um canal seguro, onde apenas os clientes OPC são capazes de receber informações OPC/TCP [20].

2.3.3 Modbus

O protocolo Modbus foi introduzido em 1979 pela empresa *Modicon*, líder no mercado de controladores lógicos programáveis (PLC). O *Modbus* foi concebido como o protocolo interno de comunicação ponto a ponto entre PLCs *Modicon* e os painéis de programação usados para programar os controladores. Após algumas aquisições, a *Modicon* faz agora parte da *AEG Schneider Automation* [22]. Normalmente usado em ambientes de *Supervisory Control And Data Acquisition* (SCADA) para monitorização remota, o protocolo continua a prosperar, pois é um protocolo de fácil compreensão para a maioria. Além disso, o protocolo é um sistema aberto que pode ser usado sem qualquer custo, assim é comum encontrar diversas aplicações do protocolo na indústria de automação, incluindo automação residencial.

O protocolo *Modbus* é um protocolo mestre-escravo e estes termos predominam até aos dias de hoje. Este protocolo permite apenas um mestre e até duzentos e quarenta e sete escravos. Com o protocolo Modbus, apenas o mestre pode iniciar uma mensagem. Portanto, se um escravo tiver a informação de uma ação, o escravo não consegue informar o mestre, até que este envie uma mensagem de conhecimento ao escravo [23].

2.3.3.1 Modbus TCP/IP

O Modbus TCP/IP é uma variante da família Modbus de comunicação simples e neutra para fornecedores destinados à supervisão e controlo de equipamentos de automação. Especificamente, abrange o uso de mensagens Modbus num ambiente de "intranet"ou "internet"utilizando os protocolos TCP/IP. O TCP/IP refere-se ao protocolo de controlo de transmissão e protocolo de internet, que fornece o meio de transmissão para mensagens Modbus. O uso mais comum deste protocolo atualmente é para a conexão Ethernet de PLCs, módulos I/O e *gateways* a outros barramentos de campo simples ou redes de I/O [24].

No protocolo Modbus, as transações de dados são tradicionalmente independentes, tornando-se altamente resistentes a ruído e ainda exigindo que in-

formações de recuperação mínimas sejam mantidas. As operações de programação, por outro lado, esperam uma abordagem orientada à conexão. O Modbus TCP/IP lida com ambas as situações. A principal função do TCP é garantir que todos os pacotes de dados sejam recebidos corretamente, enquanto o IP garante que as mensagens sejam endereçadas e roteadas corretamente. De realçar que a combinação TCP/IP é meramente um protocolo de transporte, e não define qual o meio de dados ou como os dados devem ser interpretados. Uma conexão é facilmente reconhecida no nível do protocolo e uma única conexão pode transportar várias transações independentes. Além disso, o TCP permite um número grande de conexões simultâneas. [25]. Assim, a partir do seguinte diagrama (figura 2.21) podemos verificar como é construido um pacote Modbus TCP [26]:

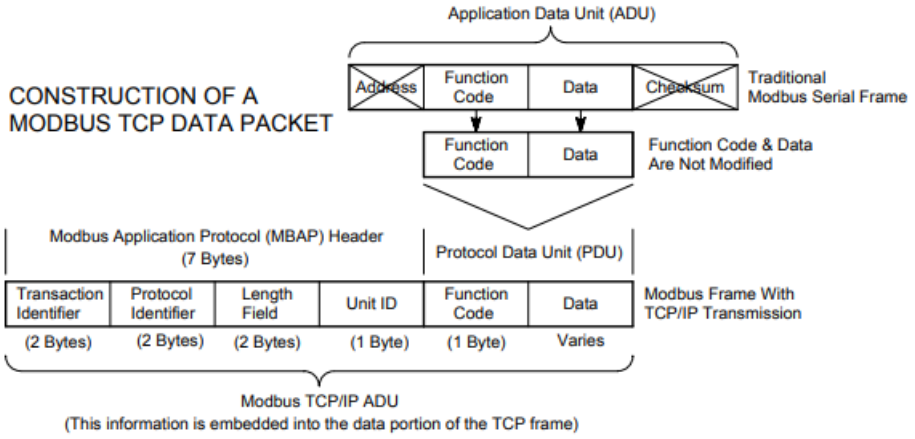


Figura 2.21: Diagrama de construção de um pacote de dados Modbus TCP. [26]

Os comandos Modbus e os dados do utilizador são encapsulados num pacote de dados de um datagrama TCP/IP sem serem modificados de forma alguma. No entanto, o campo de verificação de erros do Modbus (checksum) não é utilizado, pois o método padrão de verificação da camada Ethernet TCP/IP é usado para garantir a integridade dos dados. A partir da figura 2.21, verificamos que o código da função e os campos de dados são absorvidos na sua forma original. A operação Modbus na Ethernet é quase transparente para a estrutura de registo/comando Modbus.

Feito o enquadramento do trabalho, no que se refere à evolução histórica do caminho-de-ferro, nomeadamente os princípios de segurança que foram sendo adotados e que vigoram atualmente, bem como os protocolos de comunicação que permitem a troca de informação entre os sistemas que implementam no terreno esses princípios, o capítulo seguinte abordará a arquitetura do simulador para validação de sistemas de sinalização ferroviária desenvolvido, listando-se os seus requisitos, explicando e justificando as diferentes opções tomadas ao longo do projeto.

Capítulo 3

Arquitetura do Sistema

Esta secção consiste no trabalho de pesquisa científica e técnica relacionada com a arquitetura do projeto. Uma vez adquiridos todos os conhecimentos necessários para a compreensão e avaliação do estado da arte dos conceitos e tecnologias relacionadas, segue-se a escolha do *hardware* e *software* necessários para levar a cabo a concretização do objetivo deste projeto.

3.1 Contexto Inicial

Primeiramente, um dos objetivos traçados para este trabalho foi a criação de um *software* de comunicação entre uma interface gráfica e um PLC. Isto porque, através da criação de um *software* de comunicação, era possível validar, testar e aplicar as diferentes funcionalidades de sinalização e encravamento da interface gráfica.

Inicialmente, tendo em conta a proposta apresentada, surgiu a necessidade de comparar as diferentes tecnologias de comunicação industrial, de forma a apresentar uma solução ótima no que diz respeito a este tipo de tecnologias e a aplicação em si, conhecer a comunicação *ModBus* e procurar entender o funcionamento do *hardware*/*software* disponibilizado pela empresa.

Com este capítulo, foi possível compreender um pouco melhor os recursos utilizados, bem como oferecer uma visão alargada sobre as soluções existentes. De forma a compreender melhor a arquitetura de sistema deste projeto, iniciamos o capitulo com a análise do diagrama de blocos associado (Figura 3.1).

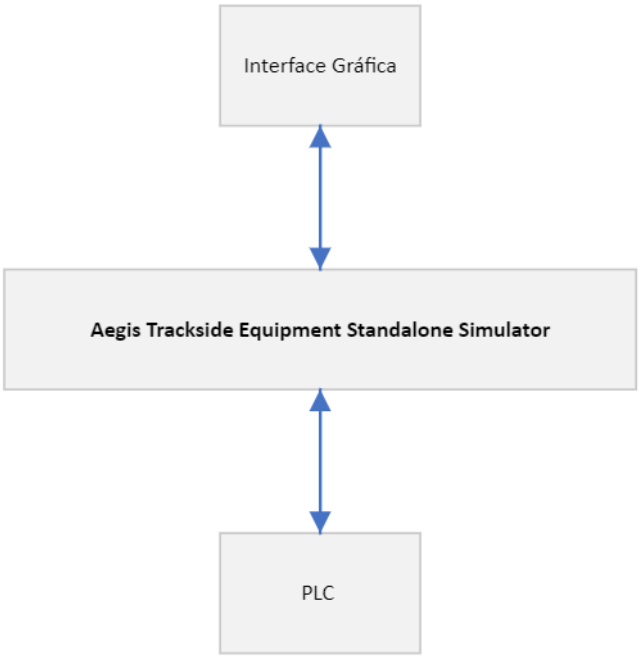


Figura 3.1: Diagrama de blocos da arquitetura do sistema.

Analisando a figura 3.1 verificamos que o projeto incide na simulação de cenários com recurso a uma interface gráfica. A interface gráfica é a janela de comunicação com o utilizador, onde todos os pedidos são passados ao PLC. Esta permite ao utilizador selecionar qual o estado de cada um dos equipamentos a simular. O *software* desenvolvido tem como função realizar a comunicação entre a interface gráfica e o PLC, equipamento onde ocorre o sistema de encravamentos e com o qual o simulador de equipamentos comunica através de uma interface protocolar, realizando o tratamento dos pedidos com base nas suas especificações. Este *software* apresenta também uma série de configurações base que são apresentadas a partir de um ficheiro .csv que dá a conhecer ao *software* quais os endereços alvo do PLC, o tipo de função *ModBus* (leitura ou escrita de *coil* (atribuição de memória interna ou definição de saída do PLC como resultado da operação lógica que é atribuída a um endereço) ou registos) e o nome da função.

Na ótica do módulo de comunicações para simuladores de equipamentos, este é responsável por reencaminhar a informação simulada de cada um dos equipamentos até ao PLC (onde ocorre o sistema de encravamento) e no sentido contrário, os comandos requeridos pelo sistema de encravamento.

3.2 Protocolo de Comunicação

Como referido no segundo capitulo e após uma larga pesquisa no sentido dos fatores a favor e contra no que respeita ao tipo de protocolo de comunicação a utilizar que melhor se adequava ao projeto apresentado e a solução que se pretendia foi utilizado o protocolo *ModBus*. Um protocolo de comunicação da camada de aplicação (modelo OSI) com utilização via *ethernet*, uma vez que, um dos requisitos do projeto implicava a operação remota por IP do software com o PLC. Ora o protocolo do tipo TCP/IP não garante que todo o tipo de *hardware* consiga converter com sucesso os dados, mas apenas que as mensagens são transferidas entre dispositivos com sucesso [27]. Através da comunicação *ModBus* TCP os dados são encapsulados em formato binário de frames TCP para a utilização do meio físico *ethernet*. Quando o *ModBus* TCP é utilizado, o mecanismo de controlo de acesso é o CSMA-CD (próprio da rede ethernet), utilizando o modelo cliente-servidor. [28].

Verificou-se que de entre algumas das hipóteses disponibilizadas a biblioteca *EasyModBus* era a que melhor se adequava ao projeto e a que ia ao encontro do objetivo de funcionamento do *software* a desenvolver. Está é uma biblioteca .NET que permite um acesso rápido e seguro de um computador ou sistema embebido a sistemas PLC e outros componentes de automação industrial. Toda a informação e documentação associada pode ser encontrada em *easymodbus.net* com todos os direitos associados a Stefan Rossmann Engineering Solutions (Copyright 2015-2020) [29].

3.3 Ambiente de Desenvolvimento Integrado

Microsoft Visual Studio é um ambiente de desenvolvimento integrado (IDE) da Microsoft. Este ambiente de desenvolvimento é utilizado para desenvolver programas de computador, sites, aplicações web, serviços web e aplicações móveis. O *Visual Studio* utiliza plataformas de desenvolvimento de *software* da Microsoft como

- **API Windows** - Conjunto base de interfaces de programação API para o sistema operativo Microsoft Windows. Quase todos os programas para Windows interagem com o Windows API;

- **Windows Forms** (WinForms) - É uma biblioteca de classes gráficas (GUI) gratuita e de código aberto incluída como parte do Microsoft .NET e .NET Framework, fornecendo uma plataforma para criação de aplicações cliente para desktop. No que respeita à sua arquitetura, o *Windows Forms* é um aplicativo orientado a eventos, ou seja, a aplicação utiliza grande parte

do tempo, simplesmente à espera de ações por parte do utilizador, como preencher uma caixa de texto ou clicar em um botão;

- **Windows Presentation Foundation** (WPF) - Trata-se de um subsistema gráfico no .NET Framework que utiliza a linguagem de marcação, conhecida como XAML para desenvolvimento de GUIs. Esta linguagem oferece um modelo consistente de programação para a construção de aplicações, numa clara separação entre interface com o utilizador e lógica de negócios.

Assim o *Visual Studio* pode produzir código nativo e código gerenciado [30] [31].

3.3.1 Windows Presentation Foundation

No que respeita à arquitetura inicial do projeto, a plataforma de desenvolvimento que vai ao encontro dos seus objetivos é o *Windows Presentation Foundation* com implementação baseada no *.NET Framework*. Como descrito anteriormente trata-se de um subsistema gráfico do .NET Framework que usa a linguagem de marcação conhecida como XAML para desenvolvimento de GUIs. A maioria das aplicações é criada para fornecer ao utilizador os meios para visualizar e editar dados. Para aplicações WPF, o trabalho de armazenamento e acesso a dados já é fornecido por muitas bibliotecas de acesso a dados *.NET* diferentes, como SQL e *Entity Framework Core*. Depois de acessar os dados e de carregar nos objetos criados por um aplicativo, começa o trabalho árduo dos aplicativos WPF [31]. Essencialmente, este processo envolve duas etapas:

- Cópia dos dados dos objetos criados em controlos, onde os dados podem ser exibidos e editados;

- Garantir que as alterações feitas nos dados usando controlos sejam copiadas de volta para os objetos criados.

De forma a simplificar o desenvolvimento de aplicações, o WPF fornece um poderoso mecanismo de vinculação de dados para lidar automaticamente com essas etapas. A unidade principal do mecanismo de vinculação de dados é a classe vinculação, cujo trabalho é vincular um controlo (destino da vinculação) a um objeto de dados (origem da vinculação). É possível ilustrar esta relação através da figura 3.2.

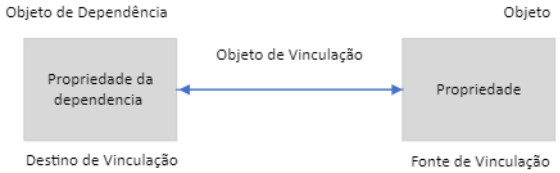


Figura 3.2: Arquitetura de vinculação do objeto. Adaptado [31]

3.3.1.1 XAML

XAML é uma linguagem de marcação declarativa. Conforme aplicado ao modelo de programação *.NET Core*, o XAML simplifica a criação de uma interface do utilizador para um aplicativo *.NET Core*, ou seja, é possível criar elementos de interface do utilizador visíveis na marcação XAML declarativa e, em seguida, separar a definição de interface do utilizador da lógica de tempo de execução utilizando arquivos *code-behind* que são associados à marcação por meio de definições de classe parciais. XAML representa diretamente a instanciação de objetos em um conjunto específico de tipos de suporte definidos em *assembly*. O XAML permite um fluxo de trabalho em que partes separadas podem trabalhar na interface do utilizador e na lógica de um aplicativo, utilizando ferramentas potencialmente diferentes. [32]

3.3.1.2 .NET Framework

Este subsistema trata-se de uma estrutura de interface do utilizador, independente da resolução e utiliza um mecanismo de renderização baseado em vetor, criado para aproveitar o *hardware* gráfico moderno. O WPF fornece um conjunto abrangente de recursos de desenvolvimento de aplicações, controlos, vinculação de dados, *layout* gráfico 2D e 3D, animação, estilos, modelos, documentos, texto e tipografia. Existem duas implementações do WPF:

- Versão *.NET* - Uma implementação de código aberto do WPF hospedada no *GitHub* executada no *.NET 5* ou superior. Embora o *.NET* seja uma tecnologia multiplataforma, o WPF é executado apenas no Windows.

- Versão *.NET Framework* - A implementação do *.NET Framework* do WPF é somente para Windows e é considerado um componente do sistema operativo Windows. Esta versão do WPF é distribuída com o *.NET Framework* [31].

3.4 Controlador Lógico Programável

Como referenciado anteriormente e segundo as especificações do projeto, após uma breve descrição da comunicação industrial a utilizar, bem como do ambiente de desenvolvimento é preciso também apresentar um dos *hardware* necessários para a arquitetura do projeto, o PLC. Este projeto foi desenvolvido em ambiente empresarial tendo sido utilizado o PLC Himatrix F30 da *Hima* (Figura 3.3), uma vez que são os PLCs utilizados pela Efacec.

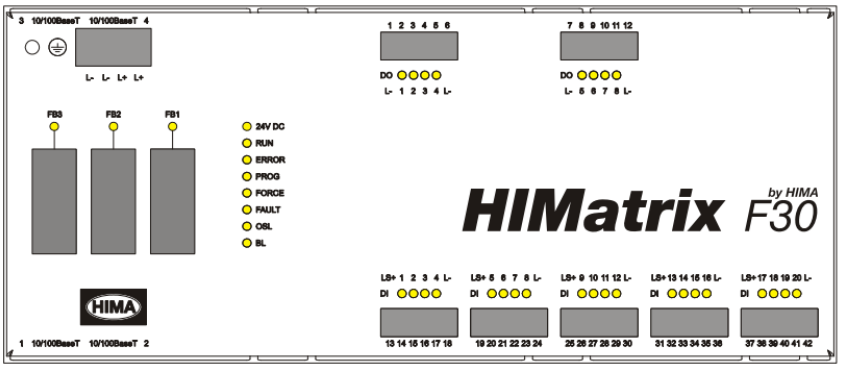


Figura 3.3: Painel Frontal: Himatrix F30. Adaptado [33]

Este controlador é um sistema compacto que possui vinte entradas digitais e oito saídas digitais. Este PLC foi certificado pela TUV (fornecedor global de serviços técnicos, de segurança e de certificação) para aplicações relacionadas à segurança até SIL 3 (IEC 61508, IEC 61511 e IEC 62061), Cat.4 (EN 954-1) e PLe (EN ISO 13849-1). Analisando a figura 3.3 podemos entrar um pouco no detalhe técnico do PLC. Focando o PLC no que diz respeito à comunicação utilizada no projeto, para comunicações via IP/TCP o *Himatrix F30* possui quatro portas RJ-45 do tipo half e full-duplex com taxas de transmissão de 10/100 Mb/s com endereço IP e máscara de sub-rede de livre configuração, reservando de fábrica a porta 502 para a comunicação *ModBus* (esta porta pode ser alterada pelo utilizador).

LED	Estado	Descrição
Verde	Ligado	Operação *full-duplex*
Verde	Piscar	Colisão de comunicação
Verde	Desligado	Operação *half-duplex* sem colisão
Amarelo	Ligado	Comunicação disponível
Amarelo	Piscar	Atividade de interface
Amarelo	Desligado	Sem qualquer comunicação disponível

Tabela 3.1: RJ-45 Led's - Estados de comunicação. Adaptado [33]

Analisando a tabela 3.1, é possível verificar que as entradas RJ-45 são fornecidas com um LED correspondente que sinaliza diferentes estados. Os parâmetros de conexão são baseadas no seu endereço MAC definido durante o seu fabrico e é também equipado com um *switch* integrado para comunicações de *Ethernet* seguras (*safe ethernet*) [33].

3.5 Comunicação de Teste EasyModBus

Assim, a primeira fase passou pela criação de um sistema que simulasse o funcionamento básico de comunicação (Figura 3.4) entre um cliente e um servidor. Neste teste o objetivo é criar um cliente com recurso a biblioteca *EasyModBus* capaz de se conectar a um servidor, onde é possível efetuar testes de conexão, escrita e leitura de valores nos diferentes campos, obtendo *feedback* das alterações e dos valores escritos e lidos no servidor. O objetivo foi testar unitariamente o cliente antes de passar para os testes integrados com o PLC. Primeiramente os testes passaram apenas pela alteração dos valores das *coils* (*true or false*). Subsequentemente foram testados a escrita e leitura de valores nos outros campos, que por sua vez implicava a escrita e leitura de valores inteiros.

Figura 3.4: Arquitetura da rede de teste.

Analisando a figura 3.5 é possível verificar que foram criados campos para a conexão ao servidor (campo onde se insere o IP do servidor e se realiza a sua conexão ou desconexão). Para efeitos de teste o IP utilizado foi o 127.0.0.2, IP registado como um endereço de *loopback*, também conhecido como *local host*. De seguida foram criados os campos de leitura para os endereços *coil*, *discrete input*, *holding register* e *input register*. Consequentemente, dependendo do tipo de campo, foram criados campos de escrita: no caso do *coil* a opção de escrita *true or false* e no caso do *input register* e do *holding register* o campo para escrita de valores. Este simulador foi testado tanto em comunicação com o servidor fornecido pela biblioteca *EasyModBus* como remotamente conectado ao PLC alvo da empresa. Em ambos os casos todos os testes realizados foram comprovados com sucesso. Este cliente foi criado com recurso a uma arquitetura do tipo Windows Forms com *.NET Framework* no ambiente de desenvolvimento *Visual Studio*.

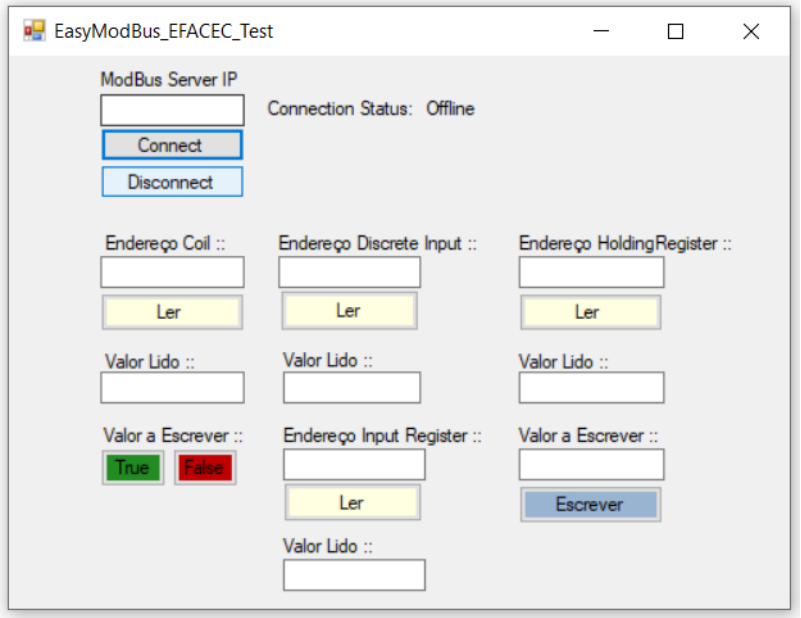


Figura 3.5: Cliente *EasyModBus*.

Após conexão do cliente ao servidor (figura 3.6), além das alterações dos estados é ainda possível verificar a informação de protocolo, ou seja, é visível os pedidos do cliente e a resposta do servidor para cada iteração feita, bem como a alteração direta dos estados ou dos valores de forma a receber diferentes leituras por parte do cliente. É ainda possível ativar ou desativar o tipo de função que queremos a funcionar durante a simulação. O servidor faz a simulação de um PLC e apresenta todas as funcionalidades da biblioteca *EasyModBus*:

- *Discrete Inputs*;
- *Coils*;
- *Input Registers*;
- *Holding Registers*.

A interface também apresenta o número de conexões em tempo real que estão a ser realizadas ao servidor (número de clientes conectados). Além de simulação para protocolo por TCP/IP também é possível simular conexões RTU. Este simulador de servidor é fornecido pelo mesmo autor da biblioteca *EasyModBus*.

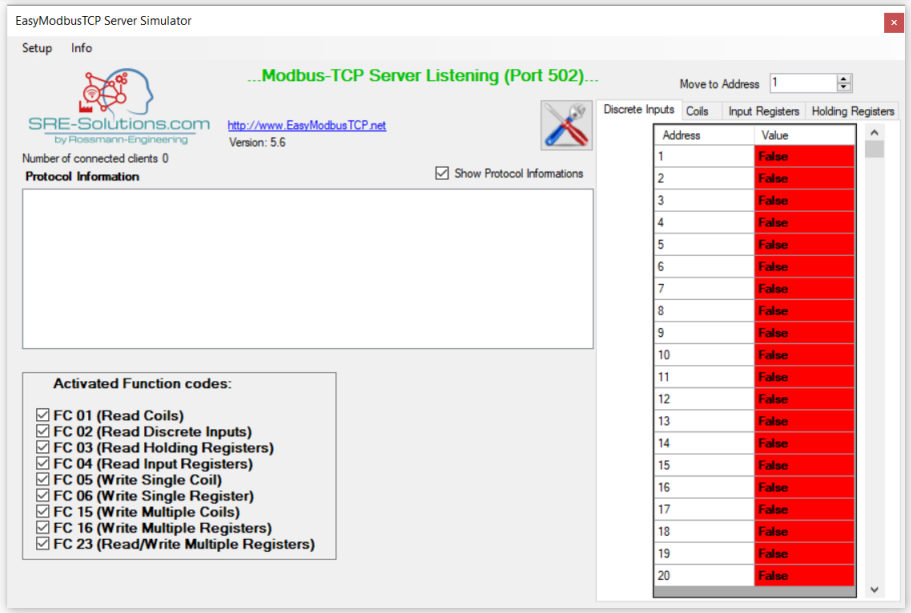


Figura 3.6: Servidor *EasyModBus*. [29]

3.6 Configuração Teste por Ficheiro CSV

Ficheiros .csv são ficheiros sem formatação em que os valores estão separados por vírgulas e cada linha apresenta um registo diferente, ou seja, um artigo, cliente ou fornecedor por linha. Estes ficheiros podem ser criados a partir de qualquer *software* de folha de cálculo, sendo os mais conhecidos *Microsoft Excel* e o *OpenOffice Calc*. Para criar um ficheiro num destes programas, cada linha tem de corresponder a um registo diferente, sendo que deve ser colocado cada valor numa célula diferente, como é possível verificar na figura 3.7 [34].

	Name	ModbusFunction	Address	AddressBit
2	o_SG_1_Lamp1Feedback_Stop	WriteCoil	0	
3	o_SG_1_Lamp2Feedback_Straight	WriteCoil	1	
4	o_SG_1_Lamp3Feedback_Left	WriteCoil	2	
5	o_SG_1_Lamp5Feedback_Amber	WriteCoil	3	
6	o_SG_2_Lamp1Feedback_Stop	WriteCoil	4	
7	o_SG_2_Lamp2Feedback_Straight	WriteCoil	5	
8	o_SG_2_Lamp3Feedback_Left	WriteCoil	6	
9	o_SG_2_Lamp4Feedback_Right	WriteCoil	7	
10	o_SG_2_Lamp5Feedback_Amber	WriteCoil	8	
11	o_SG_3_Lamp1Feedback_Stop	WriteCoil	9	
12	o_SG_3_Lamp2Feedback_Straight	WriteCoil	10	

Figura 3.7: Configuração *Modbus* em ficheiro csv.

Analisando a figura 3.7, verificamos que para este caso em especifico são ne-

cessários três campos obrigatórios: nome da função, tipo de função *ModBus* e endereço. O endereço *bit* foi um parâmetro extra, adicionado para uma futura aplicação. Neste caso, não é necessário adicionar vírgulas a cada valor, pois o programa adiciona automaticamente (dependendo das configurações selecionadas ao exportar ou guardar o ficheiro em formato .csv). Estes três campos são bastante importantes, pois o *software* quando é lançado começa por realizar a leitura deste ficheiro .csv de forma a conhecer os endereços alvo no PLC, bem como o tipo de função associada a esse endereço e o nome da função (de forma a que o *software* saiba se aquele endereço ou o nome da variável está associado a uma leitura ou escrita, para assim reencaminhar o evento criado por essa variável para a função da classe PLC correspondente). Sempre que o *software* é lançado e o ficheiro de configuração inicial .csv é lido, são criadas duas listas diferentes com formato de estrutura, ou seja, é criada uma lista do tipo *ReadModbus* (lista que contém as configurações com o tipo de função *ReadCoil* associada) e uma lista do tipo *WriteModbus* (lista que contém as configurações com o tipo de função *WriteCoil*). Analisando a figura 3.8 é possível verificar o tipo de estrutura que foi criado para a lista.

```
1 reference
public class VariablesConfiguration
{
    1 reference
    public string Name { get; set; }

    1 reference
    public string Adress { get; set; }

    0 references
    public string ModbusFunction { get; set; }

}
```

Figura 3.8: Classe de *getters* e *setters* criado para a estrutura das listas de configuração *Modbus*.

Uma lista é uma coleção de objetos que podem ser acessados por índice e com métodos para classificar, pesquisar e modificar a lista. Uma vez que se trata de uma coleção genérica, é preciso especificar um tipo de parâmetro para o tipo de dados que ela pode armazenar [35]. Assim, através da figura 3.10 verificamos que após a leitura do ficheiro .csv, as listas são criadas apresentando esta estrutura de três parâmetros. Caso as listas não fossem criadas com o tipo de estrutura apresentada na figura 3.8, nunca seria possível adicionar os três parâmetros associados (nome, tipo de função e endereço) por posição da lista.

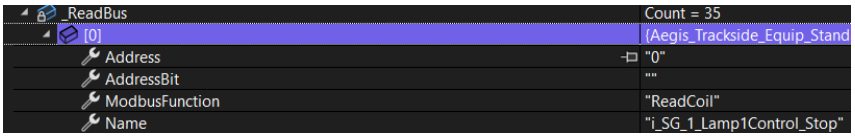


Figura 3.9: Estrutura da lista *ReadBus* por posição.

Tal como explicado anteriormente, podemos verificar na figura 3.9 que a cada posição está associado um endereço, um endereço *bit*, uma função *ModBus* e um nome. Para esta aplicação, foi necessário utilizar a biblioteca *CsvHelper* através do *NuGet* do *Visual Studio*, ferramenta que possibilita a adição de bibliotecas ou pacotes aos projetos. A biblioteca *CsvHelper* é uma biblioteca *open source* para leitura e escrita de ficheiros .csv, flexível e fácil de usar que também suporta leitura e escrita de objetos de classe personalizados.

Neste capítulo foi apresentada a arquitetura do sistema desenvolvido (modelos experimentais e testes realizados com as diferentes bibliotecas e aplicações necessárias), tendo em conta os requisitos definidos anteriormente. O próximo capítulo apresenta a forma com o sistema foi implementado, apresentando uma versão final do *software*.

Capítulo 4

Implementação do Sistema

Esta secção consiste no trabalho de aplicação relacionado com o *software* do projeto. Uma vez adquiridas todas as ferramentas necessárias para a aplicação, segue-se a explicação do *software*, decisões, arquitetura final (figura 4.1) e conhecimentos necessários para levar a cabo a concretização do objetivo final deste projeto. Assim, foi definida uma solução que abrange as especificações procuradas, tendo em conta os equipamentos disponibilizados. Este capítulo tem como objetivo a interligação entre a visão inicial, a conceção dos vários protótipos de teste e a implementação final deste projeto.

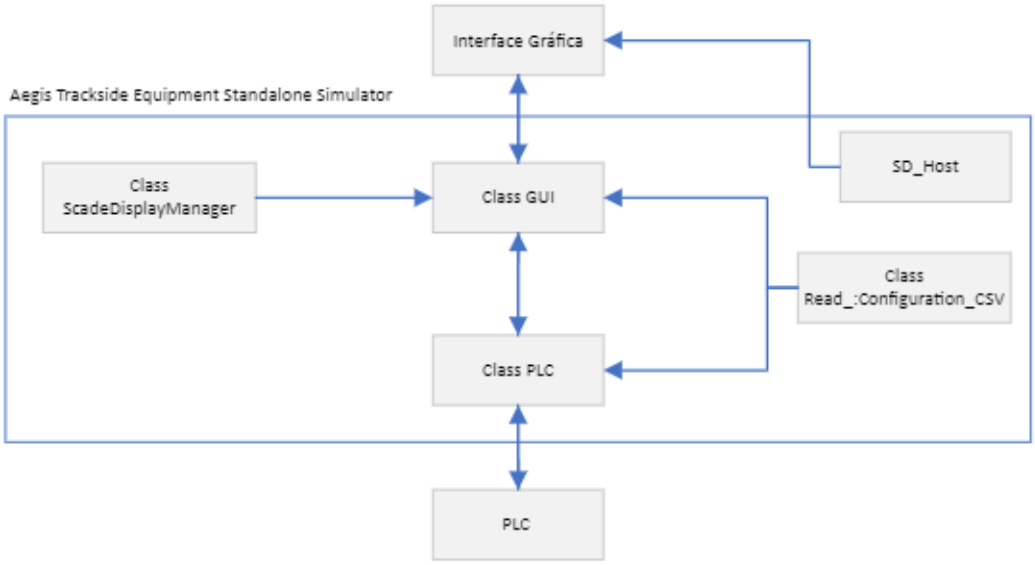


Figura 4.1: Arquitetura de Projeto Final.

4.1 Introdução

Ao nível do *software* desenvolvido foi necessário criar três classes distintas que realizam a configuração inicial e a comunicação entre as duas interfaces gráficas, uma primeira interface gráfica que realiza os pedidos de escrita e recebe o *feedback* dos comandos ativos e uma segunda interface que realiza o *feedback* gráfico dos comandos de escrita da primeira interface. Foi necessário criar um ficheiro .csv com: nome, tipo de função *modbus* e o endereço associado para os comandos de aplicação ao utilizador associado à classe ***Read Configuration CSV***. Temos ainda uma classe ***GUI*** que trata os pedidos do utilizador lançados pela primeira interface bem como o reencaminhamento dos dados lidos da segunda interface e uma terceira classe **PLC** que recebe os pedidos do utilizador provenientes dos comandos da primeira interface que faz a filtragem do tipo de função *modbus* e que realiza a devida escrita do comando na segunda interface. Existem também outras duas classes:

- ***ScadeDisplayManager*** (carrega os ficheiros .dll das interfaces gráficas, trata da inicialização da interface gráfica e ainda do tratamento dos eventos associados ao *software*);
- ***SD Host*** (trata da inicialização e processamento da janela da primeira interface gráfica. Uma vez que esta interface gráfica foi fornecida por parte da empresa como parte inicial do projeto final, é preciso associá-la ao *software* e realizar o lançamento da mesma na arquitetura do projeto WPF).

Ao longo deste capítulo será feita a explicação das diferentes classes, as bibliotecas e pacotes utilizados e ainda o funcionamento do *software*.

4.2 Desenvolvimento

Com as informações adquiridas nos capítulos anteriores, destacando-se o capítulo Arquitetura do Sistema, na qual foram realizados testes com diferentes sistemas, foi possível chegar à definição dos equipamentos e bibliotecas que se adequavam à arquitetura final do projeto. Esta secção fará a descrição da arquitetura final do projeto e a descrição de funcionamento e aplicação de cada classe.

4.2.1 Interface Gráfica *DEN Depot Zone*

Esta interface gráfica é fornecida pela empresa como base da proposta do projeto. Assim, esta subsecção é apresentada na secção do desenvolvimento para dar a conhecer a interface gráfica que é lançada com o *software* e na qual o

utilizador faz toda a interação de aplicação com o projeto. Como explicado anteriormente, apesar de toda a arquitetura do *software*, na aplicação final o utilizador apenas visualiza a interface gráfica *DEN Depot Zone* na qual faz os pedidos de escrita ao PLC. Analisando a figura 4.2 é então possível visualizar a interface gráfica *DEN Depot Zone*.

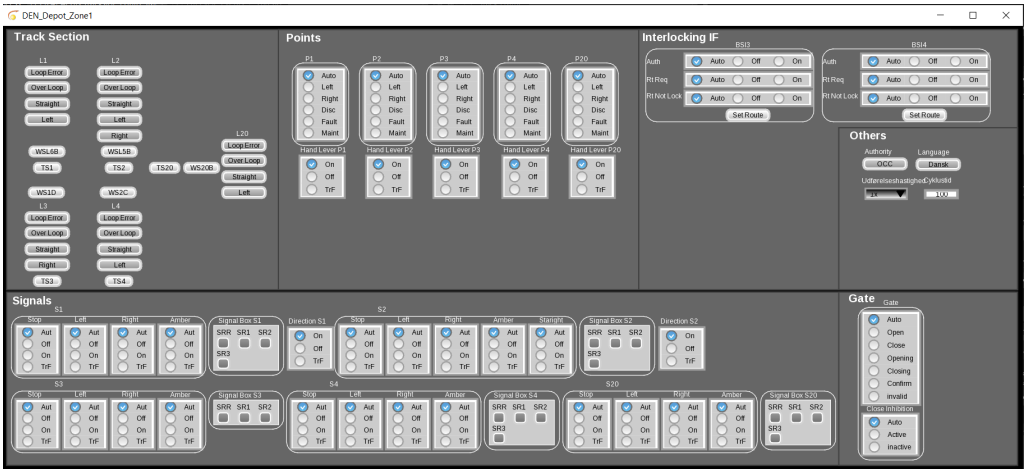


Figura 4.2: Interface Gráfica *DEN Depot Zone*.

É possível verificar que a interface gráfica *DEN Depot Zone* está dividida em seis secções diferentes:

- ***Track Section***: Nesta secção existem cinco secções diferentes (L1, L2, L3, L4 e L20) com diferentes comandos associados como:
  - *Loop Error* : Permite detetar veículos num dado ponto da rede. Adicionalmente, permite a comunicação com o veiculo para pedidos de itinerários;
  - *Over Loop* : Simula a ocupação de um veículo naquele equipamento;
  - TSx : Simula a ocupação de uma secção de linha por parte de um veículo.
- ***Points***: Nesta secção existem cinco agulhas diferentes, onde é possível ativar seis opções diferentes como:
  - Px : Neste comando é possível ativar a agulha com diferentes posições (esquerda, direita, etc) ou estado de manutenção;
  - *Hand Level Px* : Neste comando é simulada a ativação da opção de encravamento manual da agulha.
- ***Interlocking IF***: Esta secção apresenta as opções de controlo de encravamento referentes ao utilizador com autoridade:

  - *Auth* : Certos comandos necessitam de uma autorização especifica, que este comando ativa;
  - *Rt Req* : Simula o pedido de um encravamento externo quando é feito um itinerário;
  - *Rt Not Lock* : Indica o estabelecimento de um itinerário por parte de um encravamento externo.

- ***Others*** : Nesta secção é possível alterar a linguagem da interface gráfica e o tipo de autoridade;
- ***Signals*** : Nesta secção além da ativação de diferentes estados da sinalização é ainda possível ativar algumas opções diferentes como:
  - *Signal Box Sx* : Permite o estabelecimento e remoção de itinerários através de um IF junto ao sinal físico.
- ***Gate*** : Nesta secção existem dez opções diferentes associados ao portão como abertura, ou estado do portão fechado, etc.

4.2.2 Aegis Trackside Equipment Standalone Simulator

Esta secção diz respeito ao *software* desenvolvido para a comunicação entre a interface gráfica e o PLC. Assim são detalhadas, em cada uma das subsecções seguintes, as diferentes classes que fazem parte da arquitetura final do *software*, explicando o que cada classe faz, as bibliotecas ou pacotes utilizados e qual o seu papel no funcionamento do mesmo.

4.2.2.1 Classe Read Configuration CSV

No que diz respeito à configuração inicial do *software*, como anteriormente explicado, o *software* precisa de conhecer os endereços alvo, nos quais vão atuar os comandos solicitados pelo *DEN Depot Zone*. Assim é preciso criar duas listas iniciais para os endereços associados à leitura: *Read ModBus*, e os endereços associados à escrita: *Write ModBus*, com o nome e o tipo de função associado (é com o tipo de função que esta classe faz a filtragem de a qual lista adicionar as diferentes funções). Este processo inicia-se através da rotina de configuração do processo de leitura: *CsvConfiguration(CultureInfo.InvariantCulture)*, onde são especificadas as características do ficheiro .csv como: delimitações dos campos com recurso ao ponto e vírgula e um cabeçalho que separa o tipo de informação (nome, tipo função *ModBus* e o endereço). Assim o *software* sabe que ao fazer a leitura do ficheiro cada registo ocupa uma linha, sendo a informação dos três campos distribuída por três colunas (Figura 3.7).

De seguida são usados os comandos *StreamReader* que especifica o caminho onde está guardado o ficheiro .csv, e o comando *CSVReader* que faz a leitura do

ficheiro .csv com as configurações especificadas com o comando mencionado no parágrafo anterior. Posteriormente toda a informação é guardada numa lista geral. Após este passo é realizado um ciclo *foreach* (executa uma instrução ou um bloco de instruções para cada elemento em uma instância do tipo que implementa a interface *System Collections IEnumerable* ou *System Collections Generic IEnumerable <T>*) a esta lista. É neste ciclo que é feita a procura de todas os nomes associados aos diferentes tipos de função *modbus* (*Read Coil, Read Input Register, Write Coil, Write Register* e *Read Holding Register*) de forma a separar corretamente cada função para cada uma das duas listas de leitura e escrita criadas inicialmente. É possível verificar todo este processo através da análise da figura 4.3.

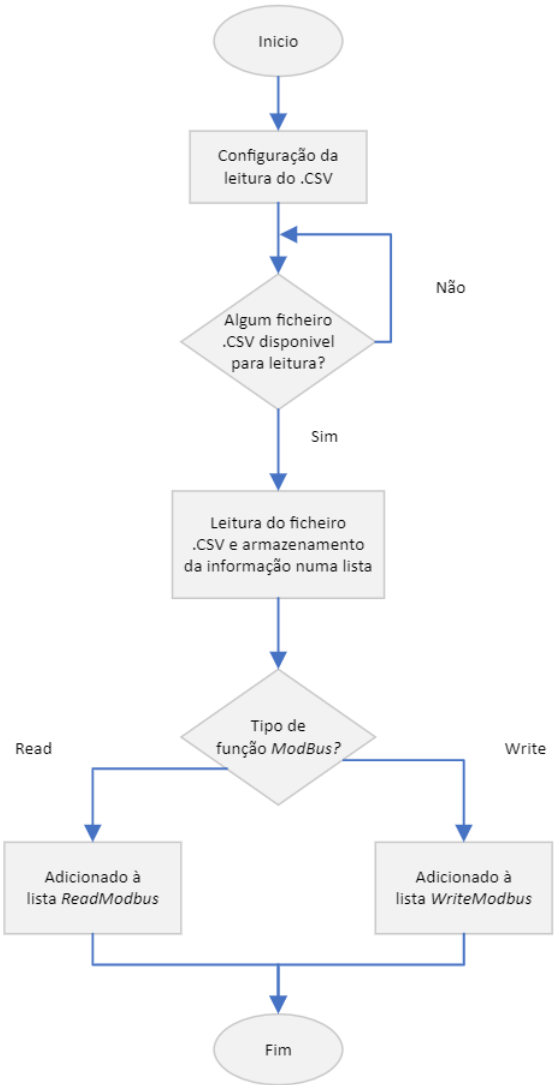


Figura 4.3: Fluxograma de funcionamento da classe *Read Configuration CSV*.

Analisando a figura 4.3 verificamos que todos os dados lidos do ficheiro .csv são guardados numa lista geral. Depois no ciclo *foreach* verificamos que o *loop* analisa toda a lista e faz a separação do tipo de função *ModBus* guardando essa informação numa variável e adicionando essa variável à sua lista correspondente seja ela leitura ou escrita.

4.2.2.2 Classe *ScadeDisplayManager*

Esta classe trata as linhas de comando responsáveis pelo tratamento da interface gráfica *DEN Depot Zone*. Esta classe não será explicada com detalhe, uma vez que a arquitetura base de código da classe foi fornecida pela empresa como código necessário para a utilização da interface gráfica. Contudo é importante analisar a classe e perceber que é aqui que o *software* faz o tratamento da IG. Inicialmente esta classe procura o ficheiro .dll correspondente à IG, onde adquire toda a informação necessária do ficheiro. É na classe *ScadeDisplayManager* que o *software* guarda a definição das funções de aquisição das variáveis e definição de novos valores caso exista alguma alteração de valores. A classe *GUI* herda os atributos desta classe para que assim seja possível a comunicação da interface com o *software* e posteriormente com o PLC no que respeita à visualização gráfica da alteração de valores e pedidos de comandos na IG. Esta classe também possui a função que realiza a inicialização da interface gráfica.

4.2.2.3 Classe *SD Host*

Nesta subsecção será feita uma breve explicação da classe *SD Host*, uma vez que esta classe foi criada com uma pequena escrita de código apenas para realizar o tratamento da janela da interface gráfica *DEN Depot Zone*. O *software* foi desenvolvido com recurso ao ambiente de desenvolvimento *Visual Studio* numa arquitetura do tipo WPF. Assim, é preciso hospedar a janela da interface gráfica *DEN Depot Zone* como um elemento dentro do conteúdo do WPF. Assim esta classe foi criada derivando da classe *HwndHost*, criando uma janela (*DEN Depot Zone*) como um filho da janela mãe que é passado como método. Após criar esta classe com as dependências por hierarquia da classe *HwndHost* é preciso especificar alguns parâmetros da janela como: largura, altura, estilo da janela, entre outras características de uma janela. Sem esta classe a janela da interface gráfica *DEN Depot Zone* nunca seria criada.

4.2.2.4 Classe *GUI*

Esta classe trata todo o código necessário para a comunicação da interface gráfica *DEN Depot Zone* com o PLC. Como referido anteriormente, a classe *GUI* herda atributos da classe *ScadeDisplayManager*, bem como a lista *WriteBus* da classe *Read Configuration CSV*. Aqui é importante realçar que não é possível pas-

sar a lista *WriteBus* da classe *Read Configuration CSV* diretamente. Assim é necessário criar uma lista local e através do construtor, passar os valores da lista *WriteBus* da classe *Read Configuration CSV* para a lista *WriteBus* local da classe *GUI*. Esta classe trata de escrever com valor falso, todos os valores lidos da lista *WriteBus*, como se realizasse uma espécie de *reset* à interface gráfica CMC associada ao PLC. Este comportamento é verificado sempre que o *software* é lançado. Só após este processo estar concluído é que é possível ao utilizador interagir com a interface gráfica *DEN Depot Zone*. De forma a entender melhor o funcionamento da classe, iremos analisar a figura 4.4 referente ao fluxograma da classe *GUI*.

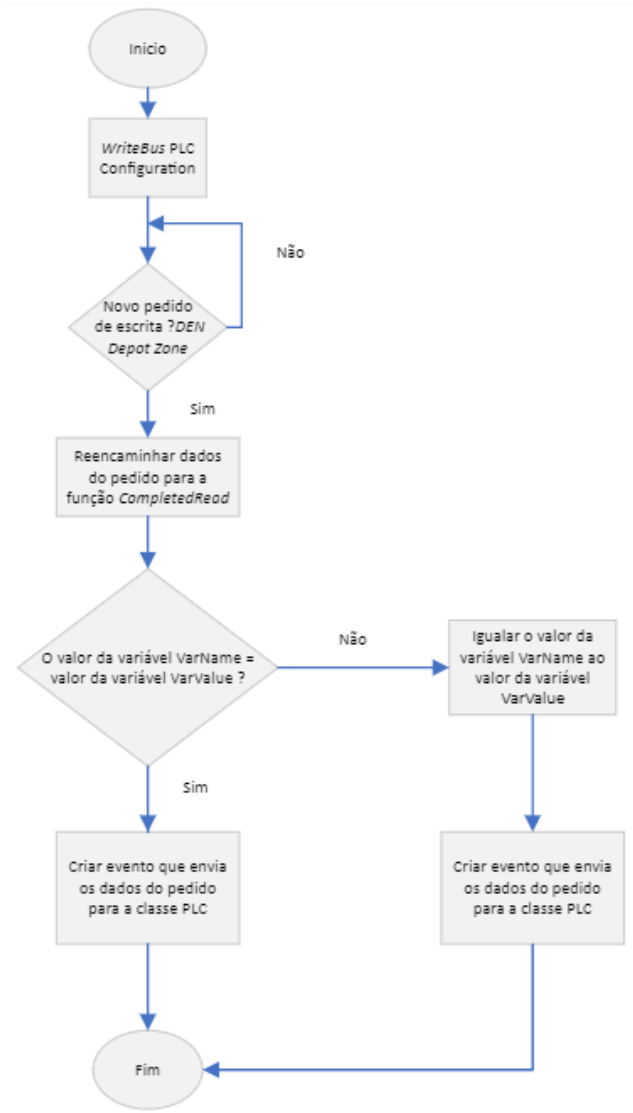


Figura 4.4: Fluxograma de funcionamento da classe *GUI*.

Analisando o fluxograma da figura 4.4, é possível entender o funcionamento da classe. Podemos dividir a explicação da classe em três fases diferentes:

- Primeiramente, como referido anteriormente, sempre que o *software* é lançado, a classe *GUI* lê os valores da lista *WriteBus* e cria os eventos de pedido de escrita nos endereços da lista com o valor *false*, como um *reset* à interface gráfica (CMC) associada ao PLC.

- De seguida a classe fica à espera de novos pedidos por parte da interface gráfica *DEN Depot Zone*. Sempre que o utilizador ativa um comando na IG, existe uma alteração do valor associado a uma determinada variável (*writeVar.Name*). De seguida reencaminha os dados: *writeVar.Name, varValue, writeVar.Address* e *writeVar.ModbusFunction* para a função *CompleteRead()*. Esta função tem associado um dicionário onde são guardados os *VarName* que são reencaminhados pela função *AcquisitionProcessRunning()*. Assim, sempre que é reencaminhado um novo pedido esta função passa por uma condição inicial: *if (MemorizedReadVariables.ContainsKey(VarName))*, ou seja, esta variável *VarName* já existe no dicionário ?, se sim é apresentada uma nova condição: *if (MemorizedReadVariables[VarName] != VarValue*, ou seja, esta variável VarName que já existe no dicionário tem valor diferente da variável VarValue ?, se sim então a função escreve na variável *VarName* o mesmo valor da variável *VarValue* e lança o evento para o PLC com os dados: *VarAddress, ModBusFunction* e *VarValue* de forma a realizar a escrita no PLC. Caso contrário, se não se confirmar a primeira condição de todas, a função automaticamente adiciona as variáveis *VarName* e *VarValue* ao dicionário para futuras comparações e lança o evento para o PLC com os dados anteriormente referidos.

- Esta classe apresenta ainda mais duas funções. Uma de escrita de valores na *DEN Depot Zone*: sempre que são alterados valores de variáveis que não tenham sido requeridos pelo *software* os valores na IG alteram como que um *feedback*; e uma segunda função responsável pela atualização dos estados das variáveis sempre que se perde a conexão ao PLC, isto é, caso o utilizador faça um pedido a partir da IG e entretanto se a conexão com o PLC for perdida, esta função limpa os dados anteriores, de forma a não repetir eventos já lançados e assim que a conexão é estabelecida lança o último evento que o utilizador tentou escrever antes da conexão se perder.

4.2.2.5 Classe PLC

No que diz respeito à comunicação com o PLC é necessário referir algumas configurações iniciais. Como em outras classes, também a classe PLC usa uma biblioteca externa: *EasyModBus* (referenciada e explicada no capítulo anterior),

bem como o uso da lista *ReadBus*, uma vez, que tal como na classe *GUI* não é possível passar diretamente a classe *ReadBus* da classe *Read Configuration CSV*. Também aqui é criada uma lista *ReadBus* local e através do construtor passam-se os valores da lista *ReadBus* da classe *Read Configuration CSV* para a lista local *ReadBus* da classe PLC.

Assim de forma a melhor entender o funcionamento desta classe, iremos analisar a figura 4.5 referente ao fluxograma da classe PLC:

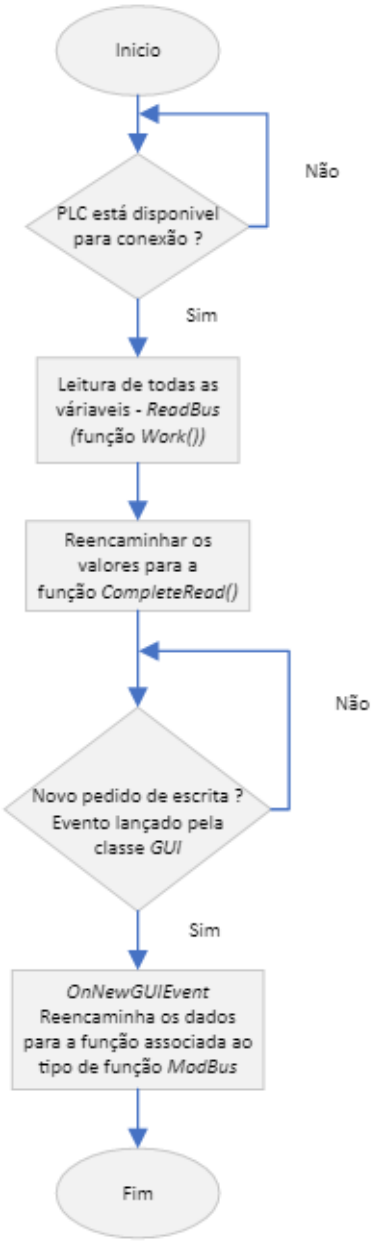


Figura 4.5: Fluxograma de funcionamento da classe PLC.

Analisando o fluxograma da figura 4.5, é possível entender o funcionamento da classe PLC, podendo dividir a análise em três fases:

- Numa primeira fase quando o *software* arranca esta classe começa por realizar a conexão ao PLC e consequentemente corre a função *Work ()*, onde percorre todas as posições da lista local *ReadBus*, lendo o valor de todos os endereços e retornando o valor à classe *GUI* para *feedback* dos valores na interface gráfica *DEN Depot Zone*.

- Numa segunda aplicação desta classe, a mesma fica à espera de eventos lançados pela classe *GUI* com os pedidos de escrita no PLC. Assim que um pedido é lançado, entramos na função *OnNewGUIEvent*, função que tem o papel de verificar a condição de qual o tipo de função *ModBus* associada ao evento para assim reencaminhar a variável *varValue* e o endereço associado à função *EasyModBus* correspondente: *Write Coil* ou *Write Register*.

- Nesta classe também são aplicadas algumas medidas de segurança. Sempre que o *software* entra em qualquer uma das funções de leitura ou escrita existe uma condição referente ao estado de ligação do PLC, isto é, de forma a garantir que um valor é escrito ou uma leitura é efetuada com sucesso é preciso garantir que o *software* está conectado ao PLC. Assim todas as funções têm implementada a condição: *if (modbusClient.Available(1000))*. Esta é uma condição da biblioteca *EasyModBus* que confere se a comunicação com o cliente está disponível e realiza essa verificação numa janela temporária (*timeout*) de um segundo. Caso não se verifique disponível a conexão, automaticamente o *software* entra na função *ReConnect ()* que lança um *pop-up* de alerta ao utilizador dizendo que a conexão foi perdida e que uma nova conexão será realizada. Realizada a nova conexão o *software* lança um evento de atualização dos últimos valores lidos e escritos, garantindo que não são repetidos eventos de escrita e lança também um *pop-up* de notificação da nova conexão realizada.

4.2.3 Interface Gráfica *Control and Maintenance Center*

Esta interface gráfica *Control and Maintenance Center* (CMC - figura 4.6) é fornecida pela empresa como base da proposta do projeto. Assim esta subsecção é apresentada na secção do desenvolvimento para dar a conhecer a interface gráfica que apresenta o *feedback* gráfico dos comandos realizados pela interface gráfica do *DEN Depot Zone*. Para visualizar esta interface gráfica é preciso possuir credenciais de acesso ao servidor *Aegis CC* onde esta e outras interfaces gráficas estão alocadas. O sinótico da área CMC permite a visualização do estado dos equipamentos de sinalização reportados pelo encravamento, sendo que o encra-

vamento recebe os estados destes equipamentos via a ferramenta de simulação de equipamentos de sinalização desenvolvida no contexto desta dissertação

É importante realçar com a análise à figura 4.6 que é possível ativar comandos diretamente na interface gráfica. Assim confirma-se o propósito da função *OnNewPLCEvent* da classe *GUI* que se trata de uma função que escreve alterações de valores na interface gráfica *DEN Depot Zone* que foram feitos no PLC mas não necessariamente por comandos do *software* deste projeto. Este *software* foi criado com o propósito de funcionar com diferentes interfaces gráficas que possam estar associadas a um PLC. Logo mecanismos de *feedback* e segurança são importantes e necessários.

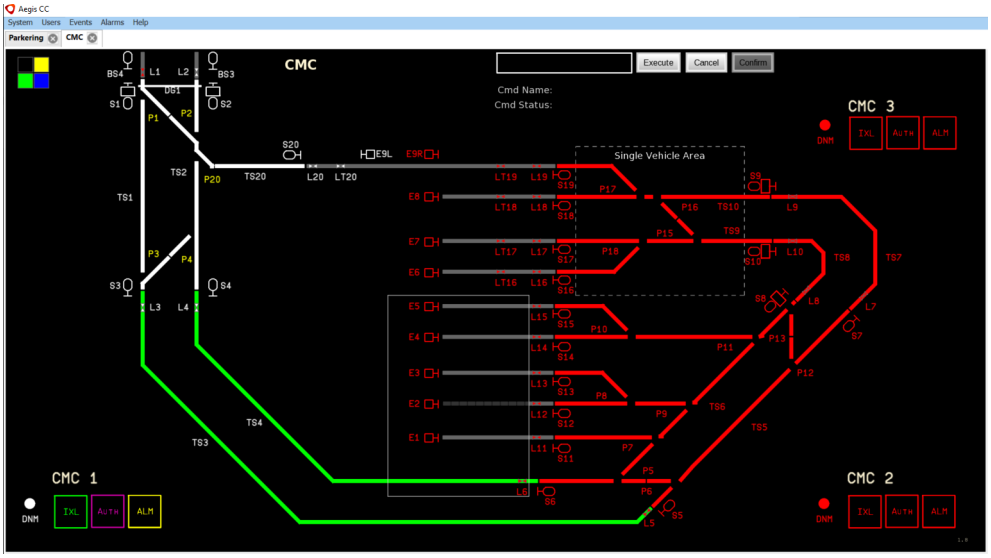


Figura 4.6: Interface Gráfica *Control and Maintenance Center*.

Neste capitulo foi apresentado todo o desenvolvimento do sistema final, tendo em conta os requisitos definidos anteriormente. Além do sistema final foram também explicadas tanto a interface gráfica do utilizador (*DEN Depot Zone*), como a interface gráfica do PLC (CMC). De realçar que este *software* não funciona apenas como simulador ao nível gráfico. Caso o PLC esteja inserido num sistema real, conectado a diferentes componentes físicos, este *software* realiza a simulação real desse mesmo sistema. O próximo capítulo apresenta os testes e resultados obtidos com a aplicação do sistema final.

Capítulo 5

Testes e Resultados

O objetivo deste Capítulo é a apresentação dos vários ensaios realizados ao sistema. Por último são expostas as conclusões tiradas nos ensaios. Faz também parte deste capítulo o esclarecimento das soluções encontradas para contornar os problemas verificados nos ensaios.

5.1 Conexão

Um dos parâmetros mais importantes a testar era a conexão com o PLC. Isto porque sem uma conexão com sucesso ao PLC era impossível o funcionamento do simulador. Assim, optou-se por realizar diversos testes no sentido de verificar, numa aplicação real, a conexão e reconexão ao longo de uma simulação e compreender se existe um resultado de sucesso no que toca ao âmbito da aplicação na rede.

De antemão eram conhecidos alguns fatores que condicionariam o desempenho dos testes, tais como o meio no qual se realizaram as conexões, isto é, para acessar o PLC é preciso estar conectado a uma rede Ethernet da empresa, ou seja, caso se tente realizar uma conexão fora da empresa é preciso conectar uma VPN.

O objetivo destes testes é a verificação de sucesso da conexão do *software* ao PLC, bem como a reconexão, caso se perca a conexão inicial. Assim, como anteriormente descrito, foram criadas algumas mensagens de notificação (pop-ups) para indicar a conexão ao PLC, como a perda de conexão. Assim é possível verificar que caso se perca a última conexão com sucesso, o último comando realizado na interface gráfica *DEN Depot Zone* será realizado assim que se verifique uma nova conexão ao PLC, bem como o envio do *feedback* dos comandos ativos.

A figura 5.1 apresenta a mensagem de notificação de conexão com sucesso ao PLC quando se lança o *software*. Nestes testes foram realizadas várias tentativas de conexão ao PLC com o PLC conectado e não conectado à rede, de forma a verificar que o *software* estava a realizar as conexões.

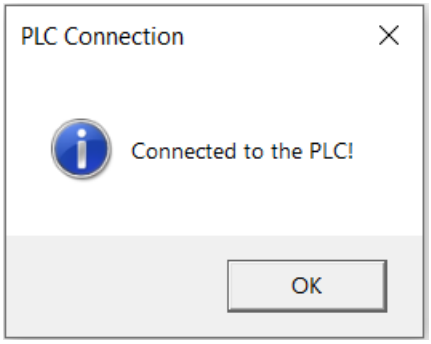


Figura 5.1: Notificação de conexão ao PLC com sucesso.

Com o *software* a funcionar, é importante conhecer o estado da conexão. Assim sempre que a conexão se perca, seja por motivos da conexão do *software* à rede, seja por motivos de perda de ligação pelo próprio PLC, sempre que se verifica a perda de conexão entre os dois meios, é lançada uma janela de notificação de perda de conexão (figura 5.2). Assim que a conexão for restabelecida (o *software* entra num ciclo loop de conexão até ter sucesso), é lançada novamente a mensagem de notificação da figura 5.1.

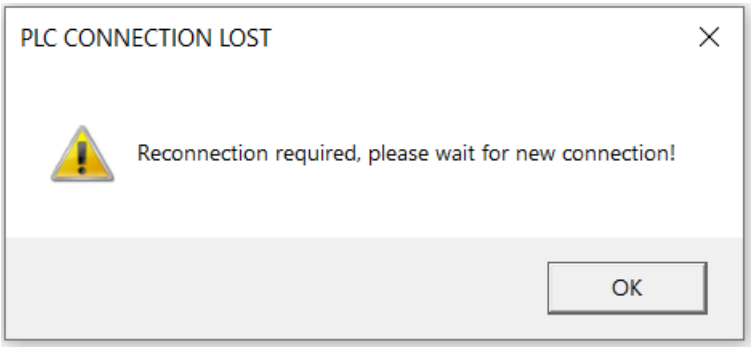


Figura 5.2: Notificação de perda de conexão com o PLC.

Estas mensagens representam um papel importante no que respeita ao bom funcionamento do *software*, isto porque, numa fase inicial do projeto, quando se realizaram os primeiros testes de interação entre o *software* e o PLC, por vezes perdia-se a conexão, mas sem um *feedback* desta alteração na comunicação entre os dois meios, o utilizador simplesmente ativava comandos, contudo não se verificavam alterações na interface gráfica CMC. Nesta fase de testes o conhecimento desta situação está presente, mas numa aplicação futura com um

utilizador que não tenha conhecimento deste fator, podiam verificar-se complicações.

5.2 Comunicação entre as Interfaces Gráficas

Esta secção explora a análise de resultados da interação entre as duas interfaces gráficas com recurso ao *software* desenvolvido. Será possível verificar os comandos ativos na interface gráfica *DEN Depot Zone* e o resultado da ativação desses comandos na interface gráfica *Control Maintenance Center*.

5.2.1 Cenário de Ocupação de Secção de Via

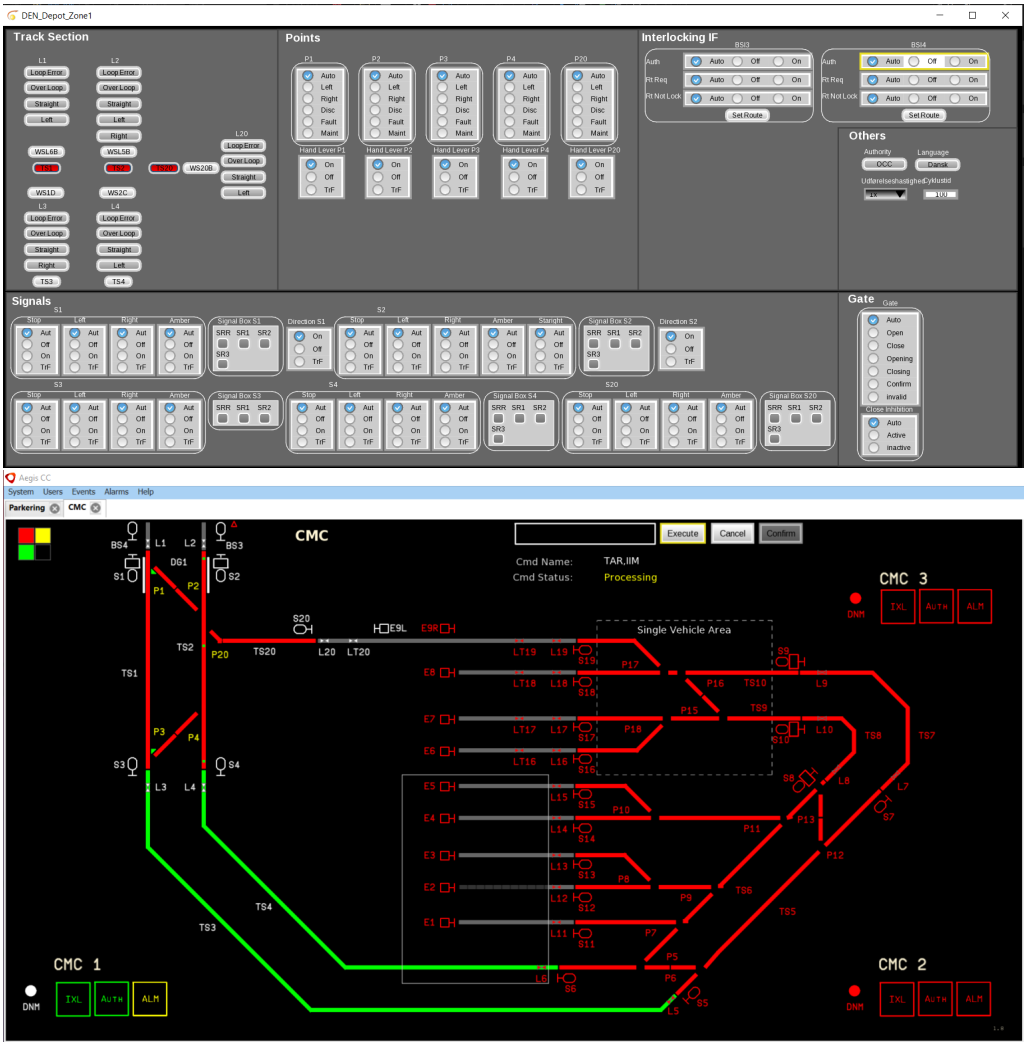


Figura 5.3: Teste de ocupação de via com recurso às interfaces gráficas: *DEN Depot Zone* e *Control Maintenance Center*.

Analisando a figura 5.3 verificamos que estão ativos na *DEN Depot Zone* os comandos TS1, TS2 e TS20 (cor vermelha) na secção *Track Section*, ou seja, foram ativados os comandos que ditam que aquelas secções de via se encontram ocupadas. Como resultado na interface gráfica *Control Maintenance Center* as secções TS1, TS2 e TS20 ficaram com a cor vermelha, ao contrária das secções TS3 e TS4 que, uma vez que não têm o comando ativo na *DEN Depot Zone* apresentam uma cor verde, o que significa que as secções de via estão livres.

5.2.2 Cenários de Criação de um Itinerário

Nesta secção serão apresentados dois cenários semelhantes, mas com algumas diferenças, que serão descritas na sua subsecção correspondente. Iremos comparar casos e analisar as diferenças.

5.2.2.1 Cenário do Itinerário SR3 (*Signal Box S1*)

O objetivo deste cenário é a verificação do encravamento de um itinerário já pré-definido na interface gráfica *DEN Depot Zone*, isto é, se ativarmos o comando *SR3* da janela *Signal Box S1* é possível verificar a ativação de um itinerário na interface gráfica *Control Maintenance Center*. Os itinerários são visíveis através das secções de via com cor verde entre dois pontos, neste caso em concreto, entre o ponto BS4/S1 e o ponto L20/S20.

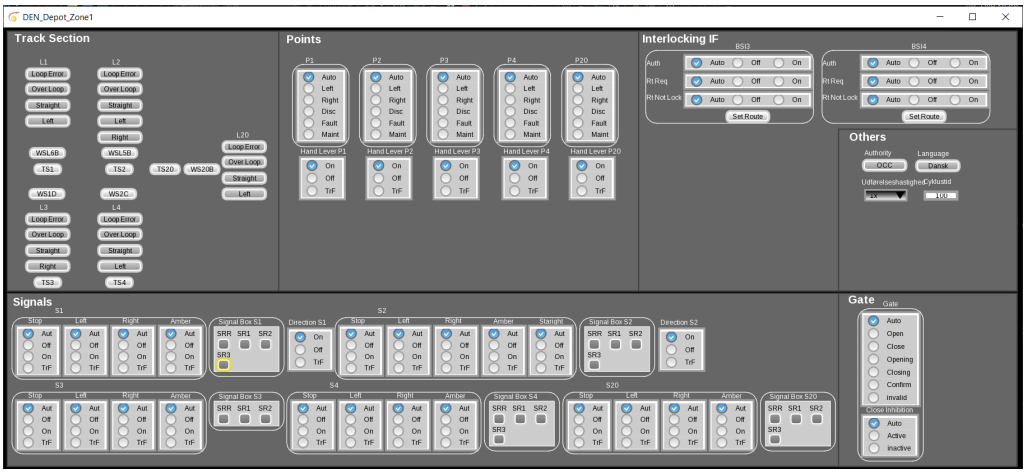


Figura 5.4: Teste de criação do itinerário SR3: *DEN Depot Zone*.

Se analisarmos com atenção a interface gráfica *DEN Depot Zone* para este cenário em particular, verificamos que os comandos referentes ao *Signal Box S1* não apresentam qualquer feedback de ativação de comando, ao contrário do que se verificou na figura 5.3 com a ativação dos comandos TS1, TS2 e TS20 (cor vermelha), isto acontece porque os comandos da *Signal Box* são pontuais en-

quanto o botão da secção indica o estado da mesma num dado momento. Como referido anteriormente SR1, SR2 e SR3 ativam diferentes itinerários (consoante definido). Para libertar o itinerário e os respetivos encravamentos e sinalizações basta ativar o comando SRR. Este comando limpa todos os itinerários anteriormente ativos.

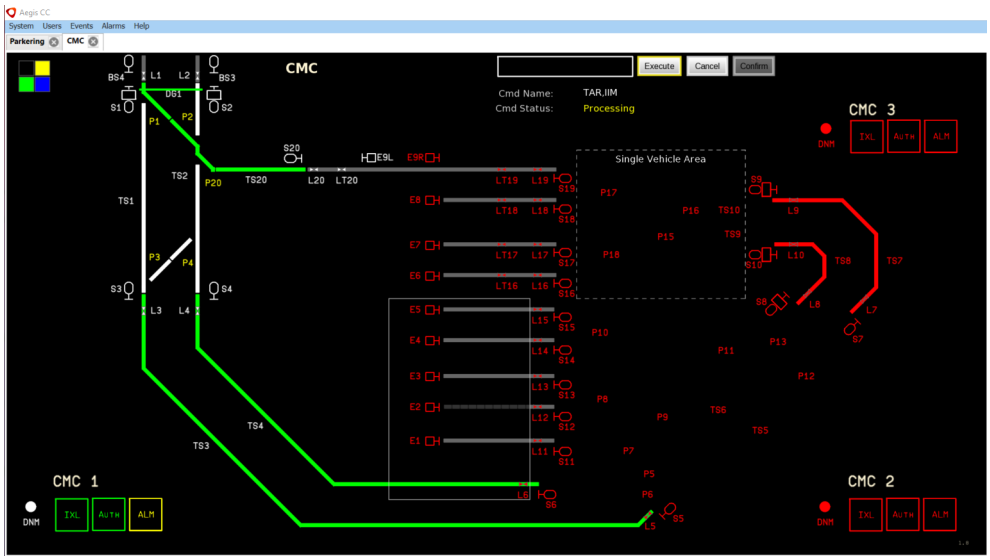


Figura 5.5: Teste de criação do itinerário SR3: *Control Maintenance Center*.

Analisando a figura 5.5 verificamos o itinerário criado entre o ponto BS4/S1 e o ponto L20/S20. O trajeto entre os dois pontos apresenta a cor verde, bem como o encravamento das agulhas P1, P2 e P20 apresentam uma cor amarela. Neste cenário é importante realçar que, uma vez que na definição do itinerário (*DEN Depot Zone*) não foram ativados os comandos de sinalização S1 com a orientação à direita , nem o comando do *gate* (portão) aberto, o DG1 apresenta uma linha verde entre S1 e S2 (*gate* fechado), apenas se o *gate* DG1 estivesse aberto é que a sinalização S1 estaria ativa com sinalização orientada à direita.

5.2.2.2 Cenário do Itinerário SR2 (*Signal Box S2*)

O objetivo deste cenário é a verificação do encravamento de um itinerário já pré-definido da interface gráfica *DEN Depot Zone*, como analisado no caso anterior, com a diferença do comando *gate* (DG1) ativo, bem como a sinalização S2 e o *feedback* da sinalização S2 na interface gráfica *DEN Depot Zone*. Como descrito anteriormente, este *feedback* é importante para o utilizador verificar o encravamento de equipamentos ou sinalização realizados com a criação de itinerários pré-definidos, ou por ativação direta na interface gráfica *Control Maintenance Center*. Como referido anteriormente, no caso da interface gráfica *Control Maintenance Center* é possível realizar comandos diretamente através da interface grá-

fica. Este *software* não foi desenvolvido apenas com o propósito de comunicação entre as duas interfaces gráficas, mas sim entre quaisquer outras interfaces gráficas.

Tal como no caso anterior, iremos analisar a figura 5.6 que representa a criação de um itinerário SR2 (*Signal Box S2*).

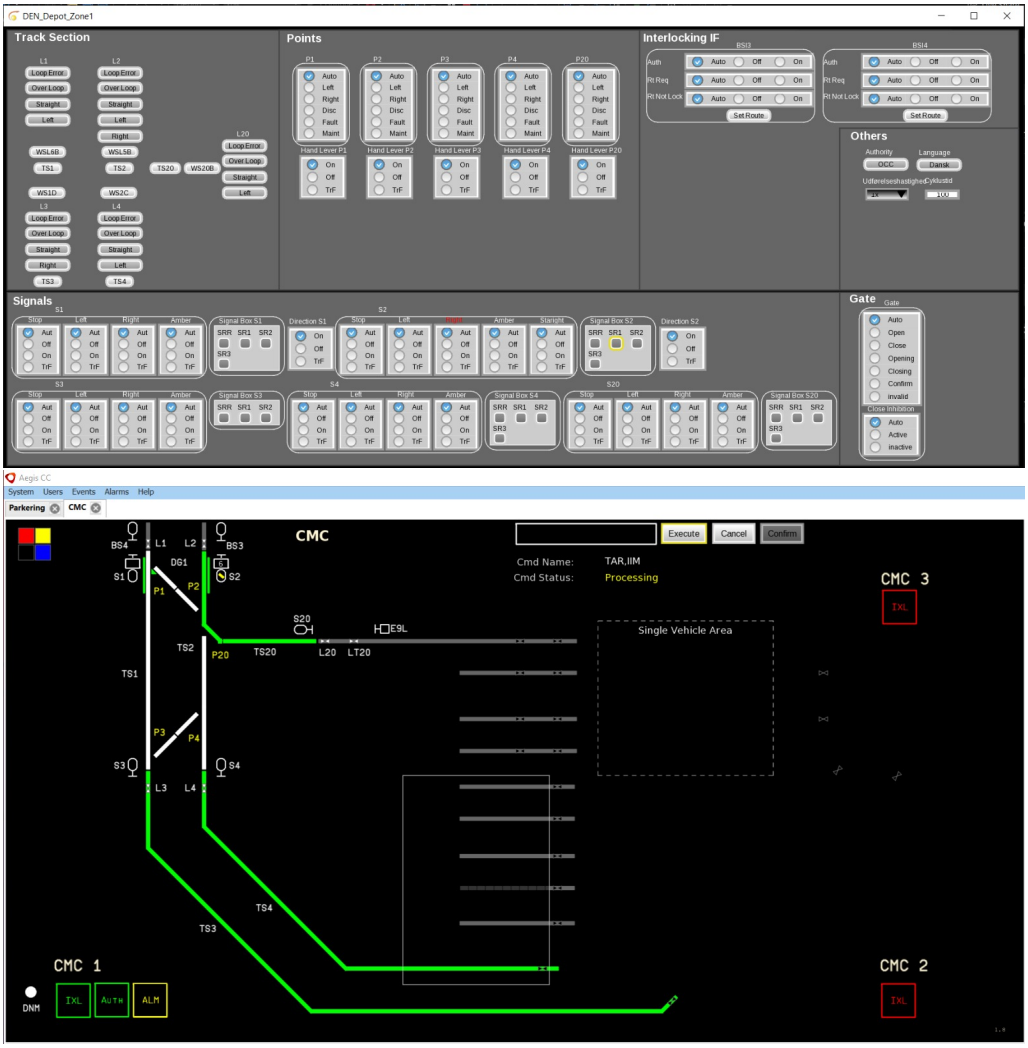


Figura 5.6: Teste de criação do itinerário SR2 com recurso às interfaces gráficas: *DEN Depot Zone* e *Control Maintenance Center*.

Analisando a figura 5.6, verificamos que na interface gráfica *DEN Depot Zone* foi ativo o comando SR2 (*Signal Box S2*) que cria o itinerário entre o ponto BS3/S2 e o ponto L20/S20. Como resultado visualizamos na interface gráfica *Control Maintenance Center* o itinerário entre os pontos BS3 e L20 com a cor verde. Ao contrário do caso anterior, o *gate* DG1 foi ativo na interface gráfica *DEN Depot Zone* (janela *Gate* como *Open*), como resultado verificamos que perpendicular às

sinalizações S1 e S2 é possível visualizar dois traços verdes que representam o *gate* DG1 como aberto e assim a sinalização S2 também apresenta uma orientação à direita com a cor amarela, ao contrário do caso anterior, uma vez que o *gate* estava fechado. Como resultado é enviado um *feedback* para a interface gráfica *DEN Depot Zone* que sinaliza o comando *Right* da sinalização S2 com a cor vermelha, o que significa que este comando está ativo, apesar de não ter sido ativado diretamente na interface gráfica.

Após analisar as semelhanças e as diferenças entre os dois casos de encravamento de itinerários é importante realçar que a cor verde nas secções (TS1, TS2, TS3, TS4 e TS20) significa que as secções estão livres, ou seja, não existe nenhum veículo presente nessa secção, ao contrário da cor vermelha que como descrito anteriormente representa a ocupação da secção por um veículo. Uma vez que os veículos entram nas via através das secções L1 e L2, caso não exista nenhum veiculo presente na secções TS1 e TS2 e não exista nenhum itinerário ativo, estas duas secções apresentam a cor cinzenta.

5.3 Cenário de Itinerário, Sinalização e *Feedback*

Nesta secção, será apresentado o último cenário de teste para análise. À semelhança dos casos anteriores, está ativo o comando SR1 (*Signal Box S4*) e ainda alguns comandos referentes a diferentes sinalizações. Este cenário é apresentado para consolidação dos diferentes comandos que é possível ativar através da interface gráfica *DEN Depot Zone*.

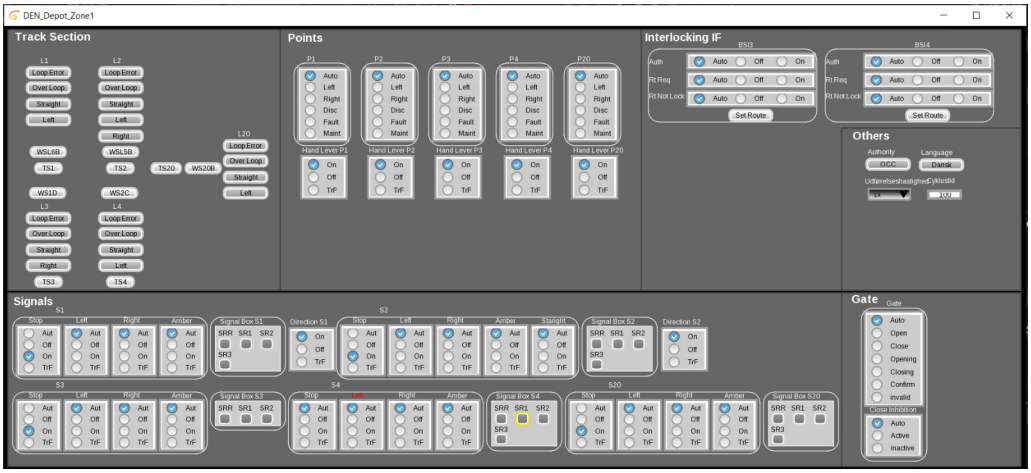


Figura 5.7: Teste Cenário Final: *DEN Depot Zone*.

Analisando a figura 5.7 é possível verificar que estão ativos os seguintes comandos:

- SR1 - *Signal Box S4*;
- *Stop - On* - S1;
- *Stop - On* - S2;
- *Stop - On* - S3;
- *Stop - On* - S20.

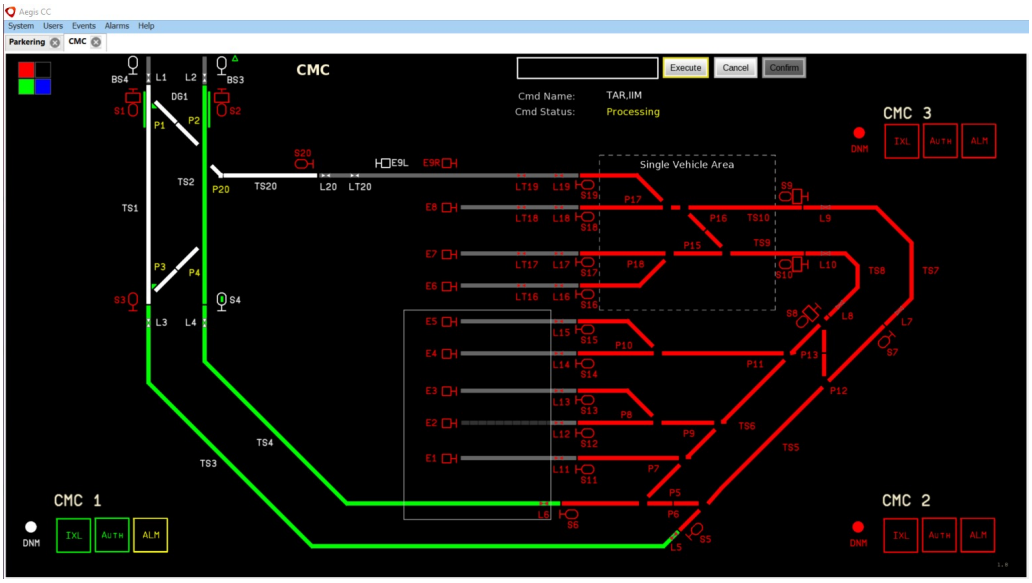


Figura 5.8: Teste Cenário Final: *Control Maintenance Center*.

Analisando a figura 5.8 podemos verificar visualmente os comandos ativos pela interface gráfica *DEN Depot Zone*. Com o comando SR1 (*Signal Box S4*) verificamos o encravamento da secção entre os pontos BS3/S2 e os pontos L4/S4 representada pela cor verde. Como resultado do comando SR1 do *Signal Box S4*, apesar do comando *Open* do *Gate* (DG1) não estar no modo *Open*, o *Gate* apresenta-se aberto, representado pelos traços verdes perpendiculares à sinalização S1 e S2, a sinalização S4 apresenta também uma cor verde, que como resultado envia um *feedback* para a interface gráfica *DEN Depot Zone*, com o comando *Left* do S4 a vermelho. Como resultado da ativação do *On* dos comandos *Stop* de S1, S2, S3 e S20, visualmente na interface gráfica *Control Maintenance Center* é visível uma cor vermelha intermitente, o que significa que a lâmpada está no estado inválido e consequentemente o aspeto do sinal é vermelho intermitente representando o estado inválido daquele elemento.

Neste capitulo foram apresentados alguns cenários de teste realizados entre a comunicação da interface gráfica *DEN Depot Zone* com a interface gráfica *Control*

Maintenance Center. Além da visualização gráfica a partir das figuras apresentadas, foi também explicado os comandos atuados na *DEN Depot Zone* e o resultado gráfico na *Control Maintenance Center*. Como referido anteriormente, estes testes foram todos simulados dentro da rede Efacec, uma vez que qualquer comunicação fora dessa rede tem de ser realizada com recurso a VPN da empresa. Estes foram testes satisfatórios, uma vez que se obtiveram resultados dentro do previsto. No próximo capitulo serão apresentadas todas as conclusões obtidas após o desenvolvimento e conclusão do *software*.

Capítulo 6

Conclusão

Neste capítulo são apresentadas as considerações finais relativas ao trabalho realizado. Além disso, são abordadas as principais dificuldades sentidas e propostas algumas ideias para trabalhos futuros.

6.1 Considerações Finais

No que diz respeito aos objetivos deste trabalho é possível concluir que os mesmos foram atingidos, uma vez que partindo dos equipamentos fornecidos e do conhecimento adquirido através do trabalho de pesquisa, foi criado um *software* de comunicação entre a interface gráfica CMC que envia comandos de escrita e a consequente atuação desses comandos no PLC. Esta comunicação pode ser consultada pelo utilizador através da interação entre as interfaces gráficas *DEN Depot Zone* e *Control Maintenance Center*. Assim, foi necessário, numa primeira fase, efetuar um estudo acerca dos conceitos relacionados com os sistemas ferroviários, mais concretamente sinalização e encravamento, e de algumas tecnologias associadas a este, bem como a aquisição de alguns conhecimentos relativos ao segmento da comunicação industrial, da programação em *C Sharp* e do ambiente gráfico de *Windows Presentation Foundation* em combinação com a *framework .NET*.

Seguidamente foi definida uma arquitetura para o sistema. Nesta foram definidos não só todos os componentes necessários para o desenvolvimento de um *software* baseado na filosofia dos sistemas ferroviários, mas também outras tecnologias e ferramentas fundamentais para a implementação deste sistema, tendo em foco os princípios de sinalização e prototipagem.

Uma vez elaborado todo o trabalho de pesquisa, onde ficaram assentes as normas ferroviárias, procedeu-se à implementação do sistema. Devido à com-

plexidade imposta pelas tecnologias associadas a estes procedeu-se à divisão do trabalho em duas fases distintas: o teste ao protocolo de comunicação: *Modbus* e o teste à leitura de ficheiros .csv. De modo a facilitar a implementação, cada um destes sistemas foi desenvolvido e testado individualmente e posteriormente integrados na solução final.

Relativamente ao projeto como um todo, o sistema desenvolvido apresenta-se como um sistema com as características necessárias para ser aplicado e adaptado para diversas soluções, não apenas para este caso em específico de comunicação entre a interface gráfica *DEN Depot Zone* e a interface gráfica *Control Maintenance Center*, mas também com outras interfaces gráficas.

Concluindo, apesar de algumas das dificuldades encontradas durante o desenvolvimento desta dissertação, todos os objetivos estabelecidos inicialmente foram atingidos. Este trabalho permitiu a aquisição de conhecimentos na área ferroviária e sobre diversas tecnologias. Além disso, espera-se que este projeto possa vir a ser uma contribuição, maior ou menor, ou até mesmo um ponto de partida para projetos semelhantes na mesma área ou até mesmo em outras áreas.

6.2 Principais Dificuldades Encontradas

Uma das principais dificuldades sentidas ao longo do trabalho revelou-se a nível prático com a integração dos diversos conceitos e tecnologias num só sistema. Este problema sentiu-se especialmente na integração do sistema final com recurso ao *Windows Presentation Foundation* devido à pouca experiência e conhecimentos relativos a este sistema gráfico. Inicialmente, o avanço ou pequenas mudanças no trabalho, demoravam muito mais tempo do que seria esperado, tempo que poderia ter sido utilizado para melhorar outros aspetos do trabalho.

Relativamente ao sistema final, a principal dificuldade prendeu-se com o envio do último comando realizado na interface gráfica *DEN Depot Zone* para a interface gráfica *Control Maintenance Center*, quando uma conexão entre o *software* e o PLC era perdida, para que não se realizasse a repetição de eventos já enviados com sucesso anterior a perda da conexão. Porém, esta dificuldade foi ultrapassada.

6.3 Trabalho Futuro

Tendo em conta o ambiente industrial para o qual se desenvolveu este *software* seria interessante implementar um protocolo de comunicação para casos em que a conexão via *ethernet* não é possível.

Uma outra ideia é a implementação de um sistema de atualização de *feedback* para alterações efetuadas diretamente na interface gráfica *Control Maintenance*

Center, uma vez que esta interface permite a interação direta.

Por último seria interessante uma aplicação no sistema que lançasse automaticamente a interface gráfica *Control Maintenance Center* sem ser necessário o l*ogin* externo ao cliente *Aegis CC* para lançar a interface gráfica de forma a verificar as alterações gráficas resultantes dos pedidos realizados pela interface gráfica *DEN Dept Zone*.

Referências Bibliográficas

[1] H. S. Pereira, "Quando Portugal acreditou no progresso: a ferrovia nacional na década de 1880," *Boletim do Arquivo da Universidade de Coimbra*, vol. 31, no. 2, pp. 105–127, 2018. [Quoted on p. 2]

[2] L. G. Peña and M. L. D. O. E. Souza, "Uma discussão sob o processo de verificação e validação de software crítico e seguro; definição de conceitos, limitações, normas, objetivos e técnicas aplicadas," *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, vol. 1, no. 1, 2013. [Quoted on p. 2]

[3] E. Irrazábal, I. Sambrana, and C. Pinto Luft, "Proceso para el diseño de la arquitectura software en un sistema crítico ferroviario según la norma EN-50128," in *XIX Simposio Argentino de Ingeniería de Software (ASSE)-JAIIO 47 (CABA, 2018)*, 2018. [Quoted on p. 3]

[4] Efacec Power Solutions. Disponível em: `https://www.efacec.pt`. Acessado em: 5 de Fevereiro de 2022. [Quoted on p. 3]

[5] M. Lewis, "Railways in the Greek and Roman world," in *Early Railways: A Selection of Papers from the First International Early Railways Conference*, pp. 8–19, 2001. [Quoted on p. 7, 8]

[6] S. Hylton, *The grand experiment: the birth of the railway age, 1820-45*. Ian Allan, 2007. [Quoted on p. 8]

[7] L. Scaggiante, "Le strade ferrate: l'Italia preunitaria e gli elementi dell'identità nazionale," 2014. [Quoted on p. 8, 9]

[8] D. A. T. Carmona *et al.*, "Contributo biobibliográfico para o estudo do caminho-de-ferro em Portugal (1856–2006)," Master's thesis, 2013. [Quoted on p. 9, 10]

[9] E. d. F. Ribeiro, "A Gazeta dos Caminhos de Ferro e a promoção do turismo em Portugal (1888-1940)," Master's thesis, Universidade de Évora, 2006. [Quoted on p. 9]

[10] Comboios de Portugal. Disponível em: `https://www.cp.pt/institucional/pt/empresa`. Acessado em: 5 de Fevereiro de 2022. [Quoted on p. 9]

[11] Wikipedia. Disponível em: `https://pt.wikipedia.org/wiki/`. Acessado em: 6 de Fevereiro de 2022. [Quoted on p. 10]

[12] G. Theeg and S. Vlasenko, "Railway Signalling & Interlocking," *International Compendium. Hamburg, Eurail-press Publ*, vol. 448, 2009. [Quoted on p. 10, 11]

[13] G. Theeg and S. Vlasenko, "Railway Signalling & Interlocking: Edition," *Germany, Leverkusen PMC Media House GmbH*, 2020. [Quoted on p. 12, 14, 15, 17, 19, 22, 23, 24, 25, 26, 27]

[14] O. N. D. S. J. P. G. M. h. T. J. T. T. W. Gregor Theeg, ulrich Maschek, *Interlocking Principles*. 2015. [Quoted on p. 20, 21, 25, 26, 28]

[15] H. Sasajima, "Intelligent field devices and fieldbus solutions in PA industries," in *SICE 2004 Annual Conference*, vol. 2, pp. 1473–1478, IEEE, 2004. [Quoted on p. 28]

[16] S. Vitturi, "Some features of two fieldbuses of the IEC 61158 standard," *Computer Standards & Interfaces*, vol. 22, no. 3, pp. 203–215, 2000. [Quoted on p. 28]

[17] P. Ferrari, A. Flammini, and S. Vitturi, "Performance analysis of PROFINET networks," *Computer standards & interfaces*, vol. 28, no. 4, pp. 369–385, 2006. [Quoted on p. 28, 29]

[18] J. Feld, "PROFINET-scalable factory communication for all applications," in *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.*, pp. 33–38, IEEE, 2004. [Quoted on p. 28, 29]

[19] T. Hannelius, M. Shroff, and P. Tuominen, "Embedding OPC unified architecture," *Automaatio XVIII Seminari, Helsinki*, vol. 63, 2009. [Quoted on p. 30]

[20] M. H. Schwarz and J. Börcsök, "A survey on OPC and OPC-UA: About the standard, developments and investigations," in *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, pp. 1–6, IEEE, 2013. [Quoted on p. 30, 32]

[21] National Instruments Corporation, "Engineer Ambitiously." Disponível em: `https://www.ni.com/en-us/innovations/white-papers/12/why-opc-ua-matters.html`. Acessado em: 20 de Abril de 2022. [Quoted on p. 31]

[22] G. Thomas, "Introduction to the ModBus protocol," *The Extension*, vol. 9, no. 4, pp. 1–4, 2008. [Quoted on p. 32]

[23] The Modbus Organization. Disponível em: `https://modbus.org/`. Acessado em: 22 de Abril de 2022. [Quoted on p. 32]

[24] B. Dutertre, "Formal modeling and analysis of the ModBus protocol," in *International Conference on Critical Infrastructure Protection*, pp. 189–204, Springer, 2007. [Quoted on p. 32]

[25] A. Swales *et al.*, "Open ModBus/TCP Specification," *Schneider Electric*, vol. 29, pp. 3–19, 1999. [Quoted on p. 33]

[26] Acromag, "ModBus TCP/IP." Disponível em: `https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf`. Acessado em: 22 de Abril de 2022. [Quoted on p. 33]

[27] N. Muskinja, B. Tovornik, and M. Terbuc, "Use of TCP/IP protocol in industrial environment," in *IEEE International Conference on Industrial Technology, 2003*, vol. 2, pp. 896–900 Vol.2, 2003. [Quoted on p. 37]

[28] M. Mellia, M. Meo, and C. Casetti, "TCP Smart-Framing: using smart segments to enhance the performance of TCP," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 3, pp. 1708–1712 vol.3, 2001. [Quoted on p. 37]

[29] Rossmann Engineering, "EasyModBus." Disponível em: `https://easymodbustcp.net`. Acessado em: 25 de Abril de 2022. [Quoted on p. 37, 43]

[30] Microsoft Corporation, "Visual Studio Documentation." Disponível em: `https://docs.microsoft.com/en-us/visualstudio/windows/?view=vs-2022`. Acessado em: 15 de Maio de 2022. [Quoted on p. 38]

[31] Microsoft Corporation, "Desktop Guide (WPF .NET)." Disponível em: `https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-6.0`. Acessado em: 16 de Maio de 2022. [Quoted on p. 38, 39]

[32] Microsoft Corporation, "Xaml overview (WPF .NET)." Disponível em: `https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0`. Acessado em: 16 de Maio de 2022. [Quoted on p. 39]

[33] HIMA Paul Hildebrandt GmbH, "HIMatrix F30 Manual." Disponível em: https://www.eic2.com/pdf/HI_800_145_E_HIMatrix_F30.pdf. Acessado em: 5 de Junho de 2022. [Quoted on p. 40, 41]

[34] Respostas Exatas, "O que é um ficheiro CSV?." Disponível em: https://respostasexatas.pt/biblioteca/palestra/read/1893-o-que-e-um-ficheiro-csv. Acessado em: 5 de Junho de 2022. [Quoted on p. 43]

[35] Microsoft Corporation, "List<T> Class." Disponível em: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-6.0. Acessado em: 6 de Junho de 2022. [Quoted on p. 44]