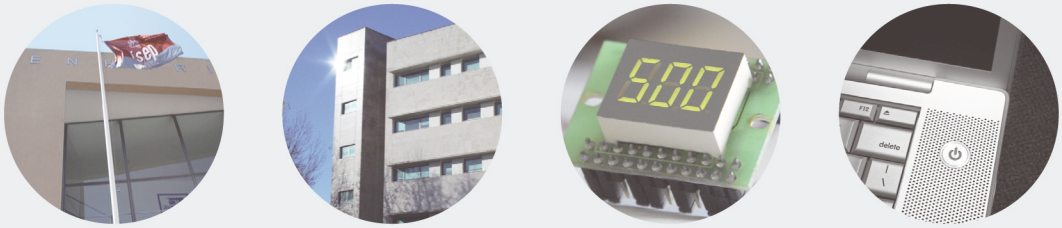
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema automatizado de teste de continuidade elétrica

PEDRO MIGUEL BARBOSA DA COSTA
outubro de 2024

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Sistema automatizado de teste de continuidade elétrica

Pedro Miguel Barbosa da Costa

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, 2024

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: Pedro Miguel Barbosa da Costa, Nº 1190955,
1190955@isep.ipp.pt

Orientação Científica: Prof. Dr. Jorge Mamede, jbm@isep.ipp.pt

Empresa: Maxiglobal, Lda.

Orientador: Rui Ferreira, rui.ferreira@maxiglobal.pt

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

Outubro, 2024

Agradecimentos

Pretendo expressar a minha sincera gratidão a todos aqueles que, de alguma forma, participaram na concretização desta dissertação.

Em primeiro lugar, gostaria de agradecer ao meu orientador, Professor Doutor Jorge Mamede pelo seu contributo constante com correções e sugestões ao longo da realização desta dissertação.

Agradeço à empresa Maxiglobal e Engenheiro Rui Ferreira pelo seu apoio técnico e pessoal disponibilizado e por ter proporcionado um bom ambiente de trabalho durante o desenvolvimento deste projeto.

Por fim, quero expressar um profundo agradecimento aos meus pais, e amigos que me acompanharam durante todo o meu percurso académico.

Resumo

Na fase de produção das indústrias, existe a necessidade de validar o correto funcionamento de um dado produto, de forma a verificar que este está a operar como previsto. A verificação da continuidade elétrica é fundamental para que se confirme a integridade do produto final.

A motivação deste projeto recai na vontade da empresa possuir um sistema capaz de efetuar testes de continuidade automatizados, visto que, até à atualidade, este procedimento tem sido realizado com recurso a multímetro, sendo propício a erros humanos e demora um tempo considerável na validação da cablagem. Este projeto foca-se no desenvolvimento de um sistema automatizado capaz de efetuar testes de continuidade na cablagem e validar o funcionamento dos sensores presentes na máquina da empresa, o *microdatacenter*. No desenvolvimento do sistema responsável por efetuar o teste de continuidade automatizado, recorreu-se a componentes de baixo custo financeiro como microcontrolador e *multiplexers*. O sistema rapidamente verifica as conexões elétricas das diferentes zonas da máquina e o resultado do teste de continuidade é apresentado numa *Graphical User Interface* (GUI), permitindo ao operador verificar em tempo real a continuidade da cablagem testada e retificar eventuais erros detetados.

Para além do teste de continuidade, o presente projeto, recorrendo à interface gráfica, também é capaz de efetuar leituras de dados provenientes de sensores baseados em Modbus e SNMP. A leitura de dados é crucial para verificar o correto funcionamento de diversos sensores presentes na máquina. Se o teste de continuidade validar a integridade da cablagem mas os dados dos sensores não são lidos, o sistema notifica uma potencial falha no equipamento, ajudando o operador a identificar e substituir o sensor danificado.

As principais vantagens deste sistema, são a velocidade, fiabilidade e baixo custo financeiro do sistema desenvolvido. O impacto deste projeto é consideravelmente significativo para a empresa, uma vez que esta ferramenta ao ser utilizada pelos operadores diminui erros humanos e tempo de verificação da cablagem, aumentando a velocidade de produção e consequentemente produzindo mais máquinas em menos

tempo.

Palavras-Chave: Teste de continuidade, Teste de funcionalidade, GUI, Sistema, Microcontrolador, *Multiplexers*, Cabos de teste

Abstract

In the industrial production stage, there is a need to validate the correct functioning of a given product, in order to verify that it is operating as expected. In products that involve wire harness, checking for eletrical continuity is essential to confirm the integrity of the final product.

The motivation for this project lies in the company's desire to have a system capable of automated continuity testing, since this procedure has been carried out using a multimeter to this day, which is prone to human errors and takes a considerable amount of time to validate the cabling. This project aims on the development of an automated system capable of performing continuity tests in the wiring and to validate the correct functionality of the sensors present in the company machine, the microdatacenter. During the development of the system responsible for carrying out the automated test continuity, low financial cost components were used such as microcontroller and multiplexers. The system quickly checks electrical connections of different sectors of the machine and the results on the continuity test are presented in a *Graphical User Interface* (GUI), allowing the operator to check in real time the continuity of the test cabling, and in some cases rectify any errors detected.

In the addition to the continuity test, this project, using the GUI is also capable of reading data from low-end sensors based on Modbus and SNMP. Reading data is crucial to verify the correct operability of various sensors present in the machine. If the continuity test validate the integrity of the cabling, but sensor data can't be read, the system notifies a pontential equipment failure, helping the operator to identify and replace the damaged sensor.

The main advantages of using this system are evident in the speed, reliability and low financial cost of the developed system. The positive impact of this project, is considerably significant for the company, since this tool can be used by operators, reduces human error and time for checking the cabling, increasing production speed and consequently, producing more in less time.

Keywords: Continuity test, Functionality test, GUI, System, Microcontroller, Multiplexers, Cabling test

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADU	*Application Data Unit*
AI	*Artificial Intelligence*
ASCII	*American Standard Code for Information Interchange*
CRC	*Cyclic Redundancy Check*
DLO	*Deformable Linear Objects*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
FDR	*Frequency Domain Reflectometry*
FTDI	*Future Technology Devices International*
GUI	*Graphical User Interface*
IC	*Integrated Circuits*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
LAN	*Local Area Network*
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diode*
MAC	*Media Access Control*
MBAP	*Modbus Application Protocol*
MIB	*Management Information Base*
MTU	*Master Terminal Unit*
NFF	*No Fault Found*
NMS	*Network Management Station*
OID	*Object Identifier*

PCB	*Printed Circuit Board*
PDU	*Protocol Data Unit*
PLC	*Programming Logic Controller*
RTU	*Remote Terminal Units*
SCADA	*Supervisory Control And Data Acquisition*
SMI	*Structure of Management Information*
SNMP	*Simple Network Management Protocol*
TCP	*Transmission Control Protocol*
TDR	*Time Domain Reflectometry*
UART	*Universal Asynchronous Receiver Transmitter*
UDP	*User Datagram Protocol*
WAN	*Wide Area Network*

Capítulo 1

Introdução

Nos últimos anos tem-se notado cada vez mais a presença de sistemas automatizados na indústria, uma vez que estes providenciam mais eficiência, precisão e segurança na execução das mais diversas tarefas. A automatização de sistemas capazes de realizar testes de continuidade elétrica, já se verifica em algumas indústrias como telecomunicações, automóvel e na aviação. No entanto, devido ao custo elevado destes sistemas, empresas mais pequenas ainda estão na fase inicial de implementação de tais sistemas de automação.

1.1 Contextualização

A utilização de sistemas automatizados capazes de verificar a continuidade elétrica de equipamentos nas mais diversas áreas da indústria acelerou consideravelmente as etapas de produção. Validar a continuidade elétrica e certificar que as conexões elétricas estão executadas como esperado, permite que a empresa assegure que os seus produtos estejam a funcionar de acordo com o expectável. Caso se verifique que as conexões elétricas não estão definidas como planeadas, estes sistemas são capazes de notificar os operadores para que estes corrijam as ligações elétricas atempadamente, sem lançar produtos inválidos para o mercado.

Os sistemas de continuidade automatizados enfrentam diversos desafios. No entanto, é certo que cada vez mais empresas comecem a realizar o seu desenvolvimento de testes automatizados, uma vez que, a solução alternativa, recorrendo a testes manuais, já não é eficiente. O processo de assemblagem de cablagem elétrica e outros

componentes é uma das etapas de uma linha de produção que mais tempo consome. Para além do tempo dedicado na assemblagem, caso ainda se tenha que realizar o teste de continuidade da mesma, ainda mais tempo é perdido por parte dos operadores. Mesmo tendo em conta o tempo dedicado a esta etapa, é certo que realizar testes de continuidade manuais é propício a se cometerem erros humanos, reduzindo a qualidade e segurança da cablagem elétrica efetuada.

A presença de ferramentas como sistemas automatizados de teste de continuidade, não apenas simplifica e agiliza a identificação de falhas elétricas, como também contribui para a manutenção proativa dos sistemas, reduzindo assim custos operacionais (tanto a nível monetário, como a nível de tempo investido) e aumentando a eficiência operacional, evitando assim interrupções inesperadas nos serviços. Como tal, a adoção de tais ferramentas tendem cada vez mais a ser uma necessidade que qualquer empresa, na área de produção necessita.

Assim sendo, a presente dissertação foca-se no desenvolvimento de um sistema automatizado de testes de continuidade elétrica, com o intuito de oferecer uma solução que permita à empresa acelerar o processo de validação de cablagem elétrica nas suas máquinas. Desta forma, os operadores podem alocar o seu tempo a outras tarefas que têm em mãos. Para além do teste de continuidade, este projeto implementa a funcionalidade de leitura de dados, provenientes de sensores, que utilizam os protocolos de comunicação Modbus *Remote Terminal Units* (RTU), Modbus *Transmission Control Protocol* (TCP) e *Simple Network Management Protocol* (SNMP). Com o desenvolvimento de uma ferramenta, que integra ambas as funcionalidades, de realizar teste de continuidade e executar leituras de dados, permite obter informações tanto da assemblagem elétrica, como também valida a correta instalação e configuração dos sensores presentes na máquina.

Sendo assim, este projeto pretende dar resposta à necessidade de uma solução eficiente, fiável e de baixo custo para a realização de testes de continuidade elétrica, complementando com a funcionalidade de executar leituras dos sensores da máquina.

1.2 Objetivos

A presente dissertação vem de encontro à necessidade da empresa pretender uma ferramenta que consiga realizar testes de continuidade automatizados nos diversos fios que constituem todo o setor de cablagem da máquina. Complementando com a leitura de dados, permite à empresa detetar a origem do problema em sensores. Isto é, caso a leitura do sensor não seja efetuada com sucesso, chama à atenção da empresa e assim facilmente usam a outra funcionalidade de testes (teste de continuidade) nos fios associados aos sensores, detetando rapidamente se o problema é de uma configuração errada do protocolo nos equipamentos ou se o problema é na camada física correspondendo assim à descontinuidade dos cabos.

Tanto o teste de continuidade, como a leitura de dados estão incorporados na mesma ferramenta. A forma como se abordou e desenvolveu este projeto, permite que a equipa de manutenção (que por vezes se dirigem às empresas de clientes) consiga transportar, esta ferramenta portátil, conseguindo assim realizar os mesmos testes na empresa do cliente e assim detetando se algum problema de manutenção se deve à continuidade de cabos ou configuração errada do protocolo.

Sendo assim, o trabalho desenvolvido e exposto nesta dissertação, visa oferecer, numa única ferramenta:

- Teste de continuidade na cablagem elétrica da máquina, apresentando a partir de uma *Graphical User Interface* (GUI) os resultados obtidos. Tal necessidade advém do facto da equipa de produção anteriormente recorrer a aparelhos como multímetro de forma a validar a continuidade dos cabos, demorando um tempo consideravelmente superior.
- Teste de pares de fios trocados;
- Leitura de dados, provenientes de equipamento com protocolo Modbus/ *Transmission Control Protocol* (TCP), Modbus/*Remote Terminal Units* (RTU) e SNMP;

1.3 Organização da Dissertação

Este documento é composto pelo presente capítulo e outros quatro capítulos.

No Capítulo 2 aborda-se o estado de arte que está divido em duas partes principais: teste de continuidade e teste de funcionalidade de sensores. Em relação ao teste de continuidade, inicialmente discute-se o conceito de continuidade elétrica e os fundamentos do teste de continuidade. Posteriormente são apresentadas soluções e metodologias abordadas, por outros autores, em relação ao desenvolvimento de sistemas automatizados capazes de testar a continuidade elétrica. Posteriormente, são abordados testes complementares aos testes de continuidade evidenciando as suas mais valias e métodos que permitem alcançar estes testes complementares. De seguida são abordadas as limitações e tendências futuras relativamente aos testes de continuidade, apresentando os principais desafios presentes na indústria e como tecnologias cada vez mais sofisticadas pretendem combater estas limitações. Relativamente ao teste de funcionalidade, são introduzidos conceitos relativamente aos protocolos Modbus e SNMP, finalizando com métodos de realizar leituras de dados baseado nestes protocolos.

No Capítulo 3 discute-se a arquitetura do sistema, onde se revê os requisitos e funcionalidades do sistema. Após a análise da funcionalidade do sistema detalham-se os componentes de *hardware* utilizados que permitiram o desenvolvimento do sistema. De seguida mencionam-se tecnologias utilizadas que permitiram a interligação entre *software* e o *hardware* utilizado. Por último, analisa-se de maneira

geral uma solução projetada, resumindo o funcionamento deste sistema, de forma a auxiliar o leitor a compreender melhor o capítulo seguinte.

No Capítulo 4 apresenta-se o protótipo final e como este foi alcançado. Inicialmente exibe-se o esquema de *hardware* de forma a permitir averiguar as ligações elétricas entre os diversos componentes utilizados. De seguida aborda-se o sistema responsável por efetuar teste de continuidade, detalhando o seu desenvolvimento desde as inicializações, funções de controlo dos *multiplexers* e a estratégia de como se realiza a obtenção, tratamento e transmissão dos resultados do teste de continuidade. Posteriormente explica-se como se desenvolveram os cabos de teste, permitindo definir uma interface entre os terminais elétricos da máquina e o sistema responsável pelo teste de continuidade. De seguida é abordado o desenvolvimento da *Graphical User Interface* (GUI), começando por uma breve apresentação do *layout* da mesma, seguindo-se as estratégias implementadas para realizar o tratamento de dados provenientes do microcontrolador e como os dados do teste de continuidade são apresentados ao leitor. De seguida é abordado como foi implementado a funcionalidade de leitura de dados provenientes de sensores baseados em Modbus e SNMP. Por fim, são apresentados os resultados de testes efetuados e discussão dos mesmos.

O Capítulo 5 apresenta uma conclusão do documento com um sumário dos objetivos expectáveis e funcionalidades implementadas, reforçando a contribuição relevante deste projeto e sugerindo possíveis melhorias a serem implementadas.

Capítulo 2

Estado de Arte

Existe um grande número de empresas em distintas indústrias que nos seus diversos produtos, como carros, aviões, máquinas de indústria, envolvem uma grande quantidade de fios elétricos que transportam dados, potência ou sinais de controlo. No entanto, durante a assemblagem do produto, existe sempre a possibilidade deste não estar com as ligações elétricas devidas, quer seja por erro humano, onde se cometeu a troca de fios, quer seja por erro de fabrico, onde estes fios estão por ventura estão danificados desde o fornecedor. Independentemente do motivo é certo que um produto com as suas cablagens incorretas, não tem o funcionamento expectável, podendo inclusive danificar o equipamento em casos de curtos-circuitos. Como tal, o diagnóstico e deteção de erros, cometidos durante a assemblagem do produto, é uma etapa essencial, de forma a garantir segurança, integridade e a performance otimizada do produto. Existem dispositivos ou sistemas, que realizam testes de continuidade elétrica, desde dispositivos manuais, como multímetro digital até sistemas automatizados capazes de executar a mesma tarefa a uma maior velocidade e garantindo maior fiabilidade. Os sistemas automatizados têm por vantagem o teste de continuidade num maior número de fios em simultâneo, permitindo velocidades muito superiores e sem erro humano na validação da assemblagem, quando comparados com sistemas manuais.

Na indústria, máquinas possuem sensores de diversos tipos e transmitindo informação que reúnem a partir de diversos protocolos. Como tal, uma ferramenta auxiliar que permita executar testes de funcionalidade a sensores baseados em Modbus e SNMP permite que haja a deteção de um possível erro de configuração do

protocolo. Isto é, ao testar a continuidade de toda a cablagem de uma máquina industrial que possua sensores, eventualmente fios associados a sensores também são testados. Caso se garanta a continuidade elétrica na camada física dos sensores e as conexões elétricas estão devidamente estabelecidas, se um sensor não estiver a transmitir dados como esperado, pode haver problemas na configuração do equipamento, inclusive, relativamente a configurações de protocolo. Ainda durante a fase de produção de uma máquina, para além do teste de continuidade à cablagem elétrica, é importante verificar o funcionamento dos seus sensores. Como tal, efetuar testes de funcionalidade aos sensores, respeitando o funcionamento dos seus protocolos, é uma estratégia de averiguar se este sensor está a operar como desejado e se transmite dados expectáveis.

A partir da revisão de literatura, no decorrer deste capítulo, são analisados o conceito de teste de continuidade, os procedimentos para testes manuais e também os sistemas automatizados que têm vindo a ser desenvolvidos para o mesmo efeito. De forma complementar, também é alvo de estudo o desenvolvimento de plataformas capazes de lidar com a verificação e implementação de certos protocolos de comunicação, embebidos nos sensores distribuídos na máquina. Ao analisar estes componentes essenciais, o propósito é relacionar o progresso que tem decorrido e os métodos que definem o *design* e as funcionalidades dos correntes sistemas que efetuam testes de continuidade e outros testes complementares.

Este capítulo é dividido em dois subcapítulos relativos a teste de continuidade elétrica e a testes de funcionalidade. No que toca a testes de continuidade, inicia-se pelo conceito de continuidade elétrica num circuito elétrico. De seguida, abordam-se os conceitos teóricos e métodos utilizados, de forma a desenvolver um sistema automatizado capaz de testar a continuidade. Posteriormente são abordados testes complementares aos testes de continuidade e as limitações e tendências futuras dos testes de continuidade automatizados. Finalmente, em relação a testes de funcionalidade, estes são abordados no final deste capítulo. É realizada uma introdução teórica dos protocolos Modbus e SNMP e métodos/comandos capazes de executar leituras de dados provenientes de sensores que tiram proveito dos protocolos anteriormente mencionados.

2.1 Teste de continuidade elétrica

Sistemas automatizados capazes de realizar teste de continuidade têm-se relevado cada vez mais essenciais na indústria moderna, particularmente em indústrias com grande volume de cablagem como a automóvel, aviação e manufatura, onde a eficiência é crucial. A necessidade se garantir a integridade de circuitos elétricos exige a implementação de soluções automatizadas que permitam uma validação rápida e

precisa. Estes sistemas desempenham um papel fundamental na redução de falhas elétricas garantindo a qualidade dos produtos finais.

Durante esta secção são abordados conceitos e métodos que permitem o desenvolvimento de um sistema automatizado que efetua teste de continuidade. Nesta revisão de literatura, exploram-se as metodologias implementadas que permitiram implementar sistemas capazes de efetuar teste de continuidade, com diversas funcionalidades complementares e finaliza-se com as limitações e tendências futuras desta área de indústria.

2.1.1 Continuidade elétrica

Considera-se que um circuito elétrico tem continuidade elétrica quando existe um fluxo contínuo de corrente elétrica, ao longo do circuito, sem interrupções [1]. Isto implica que existe um caminho elétrico, que permite o fluxo contínuo de corrente desde a fonte de alimentação, através dos vários componentes que integram o circuito, e retornam à fonte de alimentação, sem qualquer quebra no percurso. Em contraste, um circuito não tem continuidade, quando não existe um fluxo contínuo de corrente elétrica ao longo do circuito (circuito aberto). Num outro ponto de vista, verifica-se a continuidade de um circuito quando este tem uma resistência baixa, pois qualquer condutor elétrico, incluindo o melhor material condutor, tem um nível de resistência associado, por baixa que seja, proporcional ao seu comprimento. Verifica-se a falta de continuidade quando o circuito está aberto, ou seja, assume-se que o circuito no ponto onde está aberto, tem uma resistência muito elevada associada, pois é o ponto onde a corrente deixa de ter fluxo no circuito elétrico [2].

Em meios de transporte como comboios, aviões e carros, estes possuem uma grande quantidade de cablagem elétrica e, durante o funcionamento destes equipamentos, estes estão suscetíveis a condições adversas como: aquecimento, vibrações e humidade. Tais condições, são estimulantes para fios se soltarem, provocando perda de continuidade [3]. A perda de continuidade num circuito, pode-se também dever a motivos mais gerais como:

- Fio soltar-se;
- Fio partido;
- Fio sujeito a corrosão;
- Quebra da solda que une dois fios;
- Junção de fios mal soldada;
- Oxidação de pinos dos conectores;

Apesar destes problemas serem, na maioria dos casos, relativamente simples de corrigir, é fundamental detetar a falta de continuidade antes de proceder à reparação. Como tal, de forma a ser possível realizar a reparação de um circuito elétrico danificado, primeiramente é necessário estar a par da falta de continuidade do circuito. De forma a diagnosticar a falta de continuidade, existem equipamentos e/ou sistemas capazes de realizar testes de continuidade, que são frequentemente utilizados por técnicos e engenheiros, tanto na fase de produção e manufatura do produto, como também, em fase de manutenção de equipamentos. Se a falha for detetada rapidamente, pode ser retificada rapidamente e o dispositivo volta ao seu funcionamento correto, minimizando os custos do dispositivo estar com funcionamento interrompido.

2.1.2 Fundamentos do Teste de Continuidade Elétrica

De forma a garantir que um circuito elétrico esteja a operar como esperado, o teste de continuidade é uma etapa indispensável para assegurar o correto funcionamento do circuito. O teste de continuidade é realizado enquanto o circuito está desligado e consiste em determinar a presença de um percurso elétrico, capaz de estabelecer uma ligação entre dois pontos num circuito elétrico [4, 5]. É um dos testes mais importantes que um engenheiro ou técnico realiza para validar a integridade de um sistema elétrico. Este teste permite determinar se uma conexão elétrica existe no sítio correto, se uma conexão elétrica existe onde não devia (curto-circuito), e, se uma conexão não existe onde devia (circuito aberto) [1]. Sendo assim, a partir do teste de continuidade, é possível diagnosticar as seguintes falhas num cabo [6]:

- Circuito aberto - Onde o fio de um cabo está completamente partido ou interrompido.
- Curto circuito - Quando dois ou mais fios de um cabo, entram em contacto entre si. Frequentemente esta falha sucede quando o isolamento do cabo está danificado.
- Alta impedância - Também conhecido como *shunt*, esta falha do cabo advém da existência de um caminho resistivo para o *ground* que é demasiado alto quando comparado com a impedância do próprio cabo. Pode ocorrer, quando qualquer um dos fios do cabo entra em contacto com o *ground*, devido ao rompimento da bainha do cabo.

Para necessidade de teste de continuidade em poucos fios, existem sistemas simples para essa finalidade. Inicialmente, métodos rudimentares eram utilizados para detetar continuidade elétrica, por exemplo, com recurso a uma bateria e lâmpada, como se demonstra na Figura 2.1.

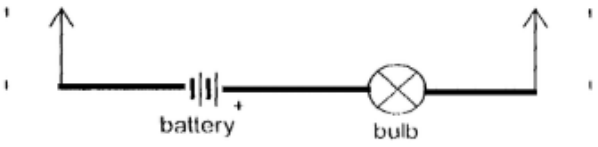


Figura 2.1: Teste continuidade com recurso a lâmpada [2]

Neste procedimento, a bateria e a lâmpada eram conectadas a ambos os terminais do dispositivo em teste, podendo ser um fio, ou componente elétrico. Caso a lâmpada acenda, indica continuidade entre os terminais. Ou seja, por consequência, o elemento a ser testado também possui continuidade. Caso a lâmpada não acenda, significa que o circuito está aberto, ou seja, não há continuidade entre estes dois pontos, provando que o dispositivo a ser testado não tem continuidade. No entanto, este método não é de todo o ideal, visto que, consoante a corrente que a bateria gere, pode danificar o componente a ser testado. Entretanto, com o aparecimento do *Light Emitting Diode* (LED), tornou este teste mais seguro, visto que, este componente minimiza a corrente na ordem dos 20 mA, quando acompanhado com um valor de resistência apropriado, limitando a corrente dependendo da tensão de alimentação. Na Figura 2.2 é possível ver um circuito que realiza o teste de continuidade de forma mais segura, baseando-se no mesmo princípio descrito anteriormente, indicando continuidade caso o LED acenda [2].

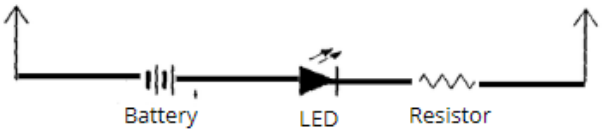


Figura 2.2: Teste continuidade com recurso a LED e resistência [2]

À medida que a complexidade de sistemas elétricos aumentam, estas abordagens revelaram-se ineficientes, tanto em termos de tempo como de precisão, devido à suscetibilidade de erros humanos, comprometendo a veracidade dos testes realizados. Neste contexto, torna-se evidente a necessidade de desenvolvimento de sistemas de teste de continuidade automatizados, capazes de lidar com grande volume de componentes elétricos, de forma rápida e viável.

2.1.3 Teste de continuidade automatizados

Devido ao crescimento exponencial da complexidade dos sistemas elétricos nas diferentes indústrias como automóvel, aeronáutica, telecomunicações e máquinas industriais, *Integrated Circuits* (IC) , *Printed Circuit Board* (PCB) e o facto da existência de cabos subterrâneos [7, 8, 9, 4], existe uma ampla variedade de testes de continuidade automatizados indispensáveis para a indústria moderna. Limitações associadas a métodos manuais demoram tempo considerável e a própria subjetividade das conclusões tiradas do teste de continuidade manual são propícias a erros, pois dependem de inspeção visual por parte de um técnico [9]. Para além disso, em setores industriais, como setor automóvel, cada carro ligeiro de passageiros tende a possuir um maior número de fios elétricos, como se ilustra na Figura 2.3. A fadiga associada a verificação de 1500 conexões facilmente invocam diversos erros de teste associados [8]. Tais erros podem resultar em interrupções significativas ou até mesmo riscos de segurança, sendo que, o recurso a tecnologias automatizadas é essencial para garantir a confiabilidade dos sistemas.

Figura 2.3: Evolução do comprimento total de fios elétricos em carros ligeiros [10]

Através de sistemas automatizados, estes métodos de teste são capazes de identificar problemas de continuidade e curtos-circuitos em tempo real. Para além disso, possibilitam a execução de um grande volume de verificações, num curto espaço de tempo, quer seja de cabos, componentes ou até mesmo circuitos, em simultâneo, assegurando que as etapas de produção são concretizadas sem falhas. O objetivo de sistemas automatizados é de substituir os técnicos que realizavam testes de continuidade, aumentando o volume de produção e sem falhas. No entanto, é importante ter bem definido, a função que se pretende obter por parte destes sistemas, de forma a desenvolver um sistema que corresponde às necessidades específicas da empresa [8, 9]. É importante realçar que processos de automação mais economicamente acessíveis, dependem muitas vezes, da sinergia encontrada entre os operadores e os sistemas de automatização [8]. Sendo assim, de seguida são expostos e analisados diversos sistemas automatizados de teste de continuidade, em diversas áreas da indústria. Iniciam-se com sistemas automatizados mais simples e financeiramente acessíveis, com recursos a componentes de baixo custo como microcontroladores, até sistemas mais robusto e dispendioso.

- ***Wire Harness Continuity Testing***

Um *wireharness*, ou chicote de fios, refere-se a um conjunto de fios, geralmente unidos por fitas, terminais, conectores ou outros tipos de proteção. A título de exemplo, observe-se a Figura 2.4, onde se encontra um exemplo de chicotes de fios elétricos. Estes são usados para transmitir sinais elétricos, potência elétrica e para fornecimento de energia em sistemas elétricos e/ou eletrónicos [8]. Por exemplo, na área automóvel, existem imensos *wireharnesses* para conectar sistemas como motores, sistemas de iluminação, travões e faróis [11]. Os chicotes de fios, são amplamente usados para simplificar o processo de produção de uma empresa. Durante a produção tem-se em atenção os fios que fazem sentido serem agrupados num mesmo *wireharness*, de acordo com a necessidade e percurso que cada *wireharness* atravessa no produto final, reduzindo assim o risco de curtos-circuitos, falha elétricas.

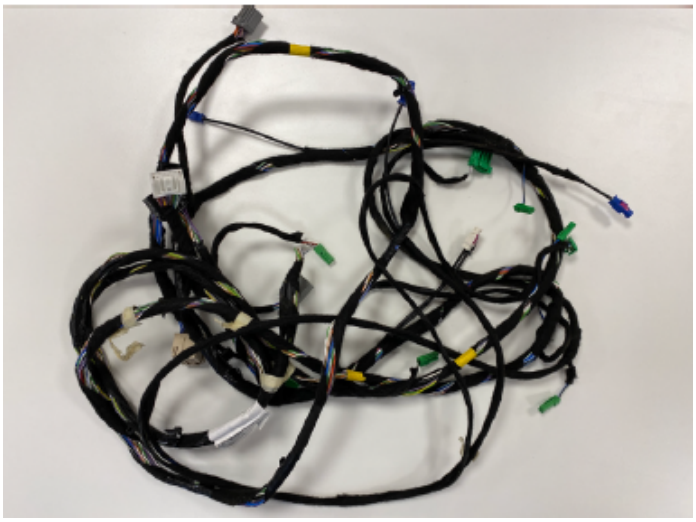


Figura 2.4: Exemplo de um *wire harness* [12]

De seguida são apresentadas diferentes abordagens de como automatizar o teste de continuidade e como este apresenta os resultados.

Kunaraj et al. [13] recorreu a um microcontrolador (atmega328p), resistências, transístores, LEDs e *Buzzer*. A solução encontrada por [13] baseou-se em dividir o seu sistema em dois testes distintos: realizar teste de continuidade a um único fio e a múltiplos fios. O teste de continuidade de um único fio é simplesmente a partir de duas *test probes*, onde caso o fio tenha continuidade, fecha o circuito e consequentemente ativa um *buzzer* e um *LED*. Já o teste a múltiplos fios é realizado a partir de um circuito projetado para verificar a continuidade de vários condutores simultaneamente. No caso do autor, foi projetado para testar a continuidade a cabos VGA ou RJ45. O circuito utiliza dois *Integrated Circuits* (IC)s principais: o CD4017 é um contador decimal de impulsos de *clock* e contém dez saídas, correspondendo a dez estados de contagem diferentes. Já o IC NE555 permite gerar pulsos. A

integração entre estes dois ICs baseia-se no facto de cada vez que o IC NE555 emite um pulso de relógio, o CD4017 incrementa a contagem, ativando sequencialmente cada uma das suas saídas. Por exemplo, o primeiro pulso gerado pelo NE555 ativa o C0 (canal 0) do CD4017, o segundo impulso ativa o canal C1, e por aí em diante. O autor tirou proveito desta integração, de forma a que cada um dos fios a serem testados, são conectados a um pino de entrada do CD4017, que é responsável por identificar qual dos fios está a ser testado em cada instante. Após a colocação dos fios nas respetivas entradas do CD4017, o teste é iniciado, e o IC NE555 gera pulsos que são enviados através dos fios. O CD4017 ativa a saída correspondente ao fio que está a ser testado, permitindo que a corrente flua através dele. Caso o fio esteja operacional e com continuidade, a corrente flui sem interrupções, ativando um LED e um *buzzer*, indicando assim continuidade. Ao utilizar um LED para cada fio, permite que o operador identifique de forma individual, a continuidade de cada um dos fios que compõe o cabo VGA ou RJ45.

Kakkeri et al. [14] realiza o teste de continuidade de múltiplos fios a partir o microcontrolador ARM LPC2148, uma placa de expansão de pinos I/O, obtendo um total de 16 pinos de I/O. A partir do *Integrated Circuits* (IC) 24C04A, previamente à realização do teste de continuidade, o microcontrolador escreve neste *Electrically Erasable Programmable Read-Only Memory* (EEPROM) números decimais associados a cada fio a ser testado. Por fim, um *Liquid Crystal Display* (LCD) faz parte deste sistema que é responsável por apresentar o resultado dos testes. Agora que se está a par dos elementos constituintes do sistema [14] explora-se o seu funcionamento e a apresentação dos resultados. O funcionamento deste sistema divide-se em três partes: Montagem do *harness*, o carregamento dos dados *standard* no IC 24C04A, relativos ao *harness* que se vai testar, e a realização do teste em si. A montagem do *harness* baseia-se em associar os fios a serem testados, aos pinos da placa de expansão de I/o. Como segunda etapa, introduz-se no IC 24C04A como foram realizadas as conexões da primeira etapa, de forma a que, o microcontrolador seja capaz de saber quais seriam as conexões, caso não haja continuidade, de forma a poder indicar via se existe ou não continuidade para certo fio. O teste é a última etapa, onde o autor disponibiliza três testes diferentes, mas sequenciais: teste de continuidade, fios trocados e conexões erradas.

- teste de continuidade - verifica se existe conexão entre dois pontos como seria de estar à espera, consoante os dados pré carregados no IC 24C04A.
- fios trocados - dois fios dizem-se estar trocados caso os dados pré carregados no IC 24C04A tenham sido por exemplo, dois pares de conexões corretas: 1-16 e 2-15 e quando se efetua este teste, identificam-se os pares de conexões 1-15 e 2-16, e de acordo com os dados previamente definidos como *standard*, assume-se que estes fios estão trocados.

- Conexões erradas - Quando as conexões do *harness* não estão como seria de esperar.

Todos os fios do *harness* a ser testado passam por estes três testes de forma sequencial, e caso os três testes sejam efetuados com sucesso, o teste a este *harness* é considerado como bem sucedido. Os resultados são apresentadas recorrendo a um LCD.

Por fim, Kustija et al. [4] de forma a automatizar o teste de continuidade utiliza dois microcontroladores atmega328p, onde um é utilizado como *master* e o outro como *slave*. De forma a ambos os microcontroladores comunicarem entre si, o autor tira proveito do módulo *Wireless* NRF24l91+ , e por último o microcontrolador *master* conecta-se a um computador e transmite dados a partir de comunicação série, de forma a que o resultado do teste de continuidade possa ser exibido numa interface gráfica. Quando o microcontrolador *master* envia "endereços" que definem o teste que se pretende realizar ao microcontrolador *slave*. Este último, por sua vez, controla *multiplexers*, selecionando os canais a serem ativados, de acordo com os fios que têm que ser testados, a partir das instruções fornecidas pelo microcontrolador *master*. De forma a efetuar o teste de continuidade, o microcontrolador *slave* aplica uma tensão através da linha e mede a resposta, verificando se a conexão está intacta. Após o teste continuidade ser realizado, o atmega328p envia os resultados do teste, de volta para o microcontrolador *master*. Este por sua vez, é responsável de reenviar os resultados para o computador, que a partir da interface gráfica, apresentam os resultados do teste ao operador. O autor menciona que o teste automatizado é 46,2% mais rápido do que a utilizar multímetro digital, para além de reduzir o risco de confusões e erros por parte do operador.

De notar que uma solução encontrada num sistema automatizado para além de se ter em atenção o custo financeiro, também tem que ser flexível para acomodar as mais distintas tarefas possíveis no próprio sistema [8]. Como tal, segue-se um estudo de sistemas que desempenham outros tipos de testes, mas que facilmente complementam o teste de continuidade.

2.1.4 Testes complementares aos testes de continuidade

Em certas áreas mais complexas como aviação, automóvel ou telecomunicações, os testes de continuidade podem nem sempre ser suficientes. A integridade de um sistema não depende apenas se a corrente flui sem interrupções através de fios, por vezes, parâmetros como a resistência mecânica de fios, correta polaridade, existências de falhas intermitentes e deteção de localização da falha de continuidade num fio, são parâmetros igualmente importantes e que contribuem para um sistema que realiza um maior número de testes complementares ao teste de continuidade. Desta forma, apresentam-se de seguida, abordagens complementares ao teste de continuidade,

cada uma com a sua função específica. São funcionalidades que se pode acrescentar a sistemas de teste de continuidade, desenvolvendo um sistema com uma combinação destes testes complementares ao teste de continuidade, proporcionando uma análise mais abrangente e robusta do estado do sistema elétrico.

- **Tensão mecânica nos fios**

Este teste complementa o de continuidade ao validar a integridade física dos fios quando sujeitos a forças externas, ou até mesmo devido a propriedades de flexibilidade dos mesmos. Fios elétricos que possam por condições severas devido ao meio que estão inseridos, estão sujeitos a ter quebras internas que podem não ser visíveis. Um sistema capaz de testar tensão mecânica e continuidade elétrica, garante a validação de um circuito que se mantém funcional mesmo sob condições adversas.

A tensão mecânica de um fio necessita de estar num intervalo específico de valores, de forma a que esteja no seu funcionamento nominal. Uma tensão mecânica muito alta pode causar uma quebra no fio ou deformar a estrutura que está a segurar um cabo. Por outro lado, uma tensão mecânica num fio demasiado baixa pode levar a ruído nos sinais transmitidos através do fio [15]. Sabendo os valores de comprimento e densidade do conjunto de fios a serem testadas, relativamente à tensão mecânica que possuem. A tensão mecânica pode ser determinada ao medir a *wire sagitta* a partir de deflexão ou a partir da frequência fundamental do fio, através da aplicação de excitação no mesmo. Esta última abordagem, pode ser categorizada pela forma como a excitação é alcançada, podendo ter origem em [15]:

- Força mecânica [16, 17];
- Força magnética [18];
- Força elétrica [19, 20];

Sendo que as medições baseadas nos métodos de deflexão ou excitação mecânica e magnética necessitam de acesso físico aos fios, ou têm a necessidade do fio estar inserido num campo magnético, não são os métodos mais acessíveis. Por outro lado, os métodos de excitação elétrica são os mais adequados [15].

Prince et al. [15] desenvolveu um instrumento digital que permite a medição de tensão mecânica num conjunto de fios, assumindo que se conhece o comprimento e densidade. O autor optou pela abordagem de excitação elétrica e como tal, tem que realizar uma análise da frequência fundamental do fio. Tal é alcançado ao aplicar altos valores de tensão elétrica aos fios vizinhos do fio que está a ser testado (*middle wire*) a realizar o varrimento da frequência de uma tensão alternada que está a ser aplicada aos vizinhos, em simultâneo. Posteriormente, na leitura do sinal do *middle wire* quando a frequência da tensão alternada coincide com a frequência fundamental.

Como tem tensão aplicada, o autor tirou proveito e também introduziu o teste de continuidade e teste de isolamento ao sistema que desenvolveu.

Quando a densidade linear, λ, comprimento, L e a frequência fundamental, f_0, de um fio são conhecidos, a sua tensão mecânica, T, pode ser calculada através da equação 2.1.

$$T = 4\lambda L^2 f_0^2. \tag{2.1}$$

Como tal, a medição de T torna-se a medição da frequência fundamental de um conjunto de fios com comprimento e densidade conhecidas, tal como mencionado anteriormente. Por sua vez, para determinar a frequência fundamental, os fios vizinhos têm que estar igualmente espaçados e paralelos entre si. A excitação do fio a ser testado (*middle wire*) é induzida através de uma combinação de corrente alternada e corrente contínua, a altos valores de tensão nos fios vizinhos. Tensões de estímulo $V_+(t)$, aplicadas num dos fios vizinhos, e $V_-(t)$, aplicadas noutro fio vizinho, definem a equação 2.2:

$$V_\pm(t) = V_{ac}\sin(\omega t) \pm V_{dc}. \tag{2.2}$$

Onde $\omega = 2\pi f$, sendo f a frequência tensão AC. A tensão DC, de polaridade inversa aplicada nos fios vizinhos, cria um campo elétrico que rodeia o fio em teste. A tensão AC que está presente nos fios vizinhos, acaba por carregar eletricamente o *middle wire*. Por sua vez, as oscilações em frequência do *middle wire* alteram a capacitância do sistema de fios, que dão origem a uma ressonância bipolar na amplitude de corrente do *middle wire*, para frequências AC perto dos valores para f_0 [15].

A vantagem desta abordagem remete ao facto de ser possível realizar três testes distintos: teste de continuidade, isolamento elétrico e tensão mecânico de fio, mesmo não tendo acesso físico aos fios. O teste de isolamento é realizado a partir do facto de que se o fio a ser testado não estiver isolado dos fios vizinhos, a sua tensão vai estar com o mesmo valor (alto) da tensão de excitação elétrica.

- **Métodos de *fault location* nos fios**

Em áreas urbanas, uma grande parte da instalação de cabos elétricos é efetuada no subterrâneo, pois é considerado mais eficiente, tanto no ponto de vista da organização como na própria transmissão de sinais elétricos. Esta abordagem reduz o impacto visual e a exposição dos cabos a condições climatéricas adversas. No entanto, surgem desafios quando uma falha ocorre num fio elétrico subterrâneo e há a necessidade de detetar a localização exata da falha.

Quando ocorre uma falha nos cabos subterrâneos, é extremamente difícil detetar a exata localização da origem de uma falha, com a intenção de reparar o cabo

danificado. A deteção de falhas visa encontrar o ponto onde ocorreram problemas como curto-circuitos, circuitos abertos ou quebra de isolamento [6].

Reflectometria é um método de alta no intervalo de alta frequência com a vantagem de usar apenas um cabo (o próprio cabo de teste) e permite obter resultados do estado do cabo enquanto se realizam as medições apenas numa das pontas do cabo. De forma semelhante ao radar, reflectometria injeta um sinal numa das pontas do cabo a ser testado. O sinal é propagado ao longo do cabo e em cada impedância de descontinuidade encontrada (devido a algum defeito ou dano no cabo), retorna uma porção da energia do sinal para a origem. A base deste método encontra-se na análise que se tem a realizar em relação ao sinal recebidos [21].

Já existem métodos baseados em reflectometria para detetar a localização de falhas, e estes podem ser classificados em duas categorias distintas: métodos *phasor-based* (domínio da frequência) e métodos *travelling wave* (domínio dos tempos) [21, 22]:

- *Time Domain Reflectometry* (TDR) - Periodicamente é injetado um sinal numa *probe* que se conecta ao cabo, e medição do sinal basicamente é obtido a partir de múltiplas porções do sinal atrasado no tempo. Para cada porção do sinal, o atraso é o tempo necessário para chegar até ao ponto de descontinuidade, partindo do ponto onde o sinal foi inicialmente injetado. Este sinal que é obtido, chama-se *reflectogram*. Ao analisar a velocidade de propagação do sinal permite localizar o ponto de descontinuidade ou defeito, pois é a partir dos pontos de descontinuidade que cada porção de sinal é criado, devido à variação de impedância. O formato e amplitude do sinal retornado, fornece informação da natureza do defeito encontrado no cabo a ser testado.

- *Frequency Domain Reflectometry* (FDR) - Este método mede a frequência, magnitude e fase de *sine waves* para determinar a distância até o ponto de falha. Neste método, em vez de enviar um pulso de tensão e monitorizar o tempo de retorno como é no TDR, o FDR emite sinais com diferentes valores de frequência. Estes sinais fluem através do cabo a ser testado, e as reflexões geradas por descontinuidades são analisadas em função da variação da frequência.

Tanto TDR, como FDR são métodos utilizados para determinar localização de uma falha. No entanto, de acordo com a aplicação final e o ambiente onde o teste é realizado, cada um destes métodos tem as suas mais valias. No domínio do tempo, TDR é uma técnica amplamente implementada em ambientes onde estão cabos curtos ou sistemas de cablagens simples, como em edifícios ou redes industriais, utilizado em redes elétricas. Já no domínio da frequência, FDR, é um método utilizado em ambientes ruidosos, devido à emissão de um sinal sinusoidal onde se analisam as

respostas retornadas a cada alteração no valor da frequência do sinal variável, sendo então menos propício a possuir interferência ou ruídos eletromagnéticos nas suas leituras. Geralmente este método é aplicado na área das telecomunicações, onde os cabos percorrem distâncias maiores, sendo mais eficaz para detetar múltiplas falhas ao longo de cabos de grande extensão.

- **Falhas de intermitência**

Falhas de intermitência nas conexões elétricas são reconhecidas como uma das principais fontes dos eventos *No Fault Found* (NFF). Falhas de intermitência refere-se à falha esporádica e não constante de um fio ou componente, ou seja, conectores e *wire harnesses* sujeitos a condições ambientais severas podem experienciar vibrações, mudanças de temperatura, humidade, que temporariamente alteram as características elétricas do componente por curtos períodos de tempo, daí o conceito de "intermitência".

A corrosão e degradação de conectores e cabos elétricos, ao longo do tempo, causam falhas de intermitência devido à não igualdade de impedância no mesmo equipamento, fazendo com que os sinais elétricos não se propaguem de igual forma, dentro do mesmo cabo elétrico, por exemplo [3]. O principal desafio dos testes de intermitência é o facto da falha não estar presente no equipamento no momento do teste, tendo por vezes que simular vibrações, de forma a que estimule a falha de intermitência, em cabos que se sabe que já estão suscetíveis a este tipo de falhas.

Qualquer interrupção, mesmo que breve, pode ser detetada com recurso a equipamentos que monitorizam de forma constante a integridade de um circuito de teste, ao longo de tempo. Os testes de intermitência baseiam-se na aplicação de tensão e corrente constantes a um circuito ao longo de um intervalo de tempo prolongado, e, em simultâneo, simula as condições adversas que estes equipamentos encontram no seu funcionamento real. Ou seja, realiza-se o teste de intermitência enquanto se induz vibração, ou altera-se a temperatura em volta do circuito de forma drástica e repetitivamente, por exemplo. Se uma falha for detetada, mesmo que por um curto espaço de tempo, esta é registada permitindo posteriormente a análise e localização da falha. Equipamentos como o osciloscópio, são capazes de registar estas variações mínimas de falha do sinal, visto que são equipamentos que são capazes de estar conectados constantemente ao circuito a ser testado, e, ao visualizar a forma de onda da corrente ou tensão, ao longo do tempo, qualquer interrupção pontual pode ser observada em tempo real, provando a falha de intermitência no circuito a ser testado [3].

- ***Network Cable Tester***

No caso de cabos de rede, existem equipamentos mais apropriados para efetuar teste de continuidade como os *network cable tester*. Alguns equipamentos executam

pouco mais do que a verificação de continuidade e o *pin-out* de um cabo (os fios individuais estão conectados ao par apropriado). Já outros equipamentos são extremamente sofisticados e caracterizam completamente as propriedades elétricas do cabo. Consoante as necessidades de cada consumidor, existem equipamentos capazes de realizar testes para além do teste de continuidade. Em equipamentos mais sofisticados, geralmente consistem num par de unidades, ou seja, tem-se o *cable tester* e um dispositivo auxiliar, que, quando conectado, forma um *loop* do sinal. Estes equipamentos geralmente fazem as seguintes verificações[23]:

- *Pin-outs* - Verifica se os pinos de cada extremo do cabo estão correspondentemente conectados.
- *Near End Cross-Talk* - medição que permite perceber a quantidade de interferência de sinal que um cabo está a impor num cabo adjacente. Valores muito altos talvez seja um indicador que foi instalado o tipo errado de cabo/conectores.
- Atenuação - Executa uma medição que verifica a porção do sinal perdido desde a origem até chegar ao destino. É um parâmetro importante a ter em consideração pois traduz a taxa máxima de informação que o cabo aguenta.
- Impedância - Advém da resistência e indutância do cabo. A medição da impedância permite descobrir onde há incompatibilidade de impedância que pode causar os sinais a ser refletidos onde há aglomerados de cabos.
- Capacitância - A energia do campo elétrico que é capaz de ser alocada no cabo. Valores fora do normal, traduz-se em problemas no cabo como curtos ou fios partidos.
- Comprimento - Ao temporizar a receção de um determinado sinal, previamente emitido, é possível descobrir o comprimento do cabo. Promovendo informação do comprimento de cabo que está dentro das paredes, por exemplo.

Os melhores *cable testers* podem ser pré-programados com valores apropriados para os diferentes tipos de cabo, permitindo uma identificação rápida dos parâmetros que estão fora das especificações *standard* [23].

2.1.5 Limitações e tendências futuras

A automação na assemblagem de *wire harnesses* tem sido uma área de crescente interesse e desenvolvimento, inclusive com a integração de tecnologias avançadas e modernas como a visão computacional e inteligência artificial. No entanto, apesar dos avanços significativos, também existem limitações associadas, uma vez que é uma área específica de manipulação de objetos deformáveis [10].

Estudos atuais, sugerem que a implementação de reconhecimento visual na assemblagem de chicotes de fios é fundamental, no entanto, ainda não se atingiu uma solução completamente robusta para manipulação de *wire harnesses*. Existem dois desafios notórios no que toca a aplicações robóticas baseadas em visão computacional: a primeira, baseia-se a na robustez e precisão de manipulação destes chicotes, a segunda remete às características físicas dos *wire harnesses*, que devido aos seus diversos formatos e imensos componentes, torna-se difícil o reconhecimento de cada elemento, por parte de um computador. Nas limitações atuais, destacam-se as seguintes [10, 12, 24, 25]:

- Desafios reconhecimento visual - Um dos principais obstáculos na automação de chicotes elétricos é a capacidade dos sistemas de visão computacional reconhecer e rastrear diversos componentes elétricos, integrantes no processo de assemblagem. A complexidade do *design* e condições de iluminação insatisfatórias, comprometem a eficácia dos sistemas de visão, resultando em falhas de deteção e manipulação dos componentes.

- Custo de implementação - O investimento inicial elevado, em conjunto com os custos de manutenção e atualização das tecnologias, é um fator a ter em consideração para a implementação de soluções robóticas em grande escala.

- Falta padrões e normas - A corrente ausência de padrões aceites para a manipulação robótica de chicotes elétricos limita a escalabilidade e interoperabilidade entre distintas soluções desenvolvidas. A falta de diretrizes explícitas e claras, dificulta a comparação de desempenho entre diferentes abordagens de sistemas desenvolvidos. A criação de normas e certificações é essencial para promover a confiança na tecnologia e facilitar a sua implementação global.

- Manipulação de *Deformable Linear Objects* (DLO) - A natureza dos *wire harnesses* são maioritariamente flexíveis e deformáveis, que dificultam a sua manipulação automatizada. A manipulação eficaz destes objetos, requer que robôs possuam capacidades de perceção e controlo. A implementação de estratégias de controlo de força aplicada aos fios elétricos, têm que garantir que a força aplicada pelo manipulador não exceda os limites físicos permitidos pela natureza dos cabos. Ou seja, há a necessidade de um controlo preciso e adaptável, uma vez que, algumas partes dos cabos podem ser rígidas, mas mesmo assim, ainda apresentam características deformáveis. Consequentemente, a falta de um sistema de controlo de força eficaz, pode resultar em danos nos componentes, comprometendo a integridade do chicote elétrico, resultando numa fraca qualidade do produto final.

- Questões de confiança - A introdução de tecnologias baseadas em inteligência artificial levanta preocupações sobre a aceitação destas metodologias, por

parte de operadores humanos. A falta de compreensão relacionada a como *Artificial Intelligence* (AI) toma decisões, remete a resistência à adoção dessas tecnologias.

Relativamente a tendências futuras, no campo de automação de *wire harnesses*, destacam-se na integração de tecnologias avançadas relativamente a sensores e inteligência artificial, proporcionando uma melhoria significativa na capacidade de robôs manipularem *Deformable Linear Objects* (DLO). Essencialmente, estes sistemas têm de ser desenvolvidos no âmbito de serem capazes de lidar com uma variedade de formas e tamanhos dos chicotes elétricos, como também, se ajustarem a diferentes configurações de montagem. Nas tendências futuras de automatização na área de assemblagem de fios, destacam-se as seguintes:

- Avanços em sistemas de visão computacional - A integração de dados multimodais, como informações táteis e sonoras, nos algoritmos de visão computacional vai melhorar a robustez e precisão do reconhecimento de componentes, proporcionando uma perceção mais abrangente e confiável nos objetos a serem manipulados durante a assemblagem.

- Inovação em *design* de produtos - Um *design* claro, explícito e padronizado de formas e cores, bem como a implementação de marcadores visuais que ajudem na identificação de componentes deve ser efetuado. A partir deste *design* mais explícito, permite que os sistemas de computação visual reconheçam e manipulem os chicotes de forma mais generalizada, facilitando a automação e reduzindo a complexidade dos processos de montagem.

- Colaboração entre operador e robô - A sinergia entre operadores humanos e sistemas robóticos irá resultar num aumento significativo de eficiência e qualidade do processo de montagem. A interação entre operadores humanos e sistemas robóticos irá aumentar a eficiência e segurança, permitindo que os trabalhadores se dediquem a tarefas que requerem habilidades cognitivas e criativas, enquanto os sistemas de automatização se concentram nas tarefas repetitivas e fisicamente exigentes.

- Estabelecimento de normas e certificações - A implementação de padrões e certificações para a manipulação robótica de objetos deformáveis é essencial para promover a confiança na tecnologia, facilitando a sua adoção em larga escala. A colaboração entre investigadores, profissionais na indústria é crucial para desenvolver diretrizes que orientem a pesquisa e implementação de soluções robóticas.

Em conclusão, embora existam limitações significativas na automação do manipulação de chicotes elétricos, as tendências futuras indicam um caminho promissor

para o desenvolvimento de soluções mais eficazes e adaptáveis, aumentando significativamente o ritmo de produção e assemblagem, assegurando a qualidade do produto final, reduzindo custos e atrasos durante o processo de manufatura e assemblagem.

2.2 Teste de funcionalidade

De forma a validar uma máquina para além de validar a continuidade elétrica na camada física, também é relevante aferir que algumas funcionalidades do sistema estão operacionais.

Esta secção vem de encontro à verificação de outras funcionalidades do sistema, como confirmar funcionalidades de transmissão e receção de *requests* e *responses* por parte de sensores integrados na máquina. Como a empresa utiliza os protocolos Modbus/*Remote Terminal Units* (RTU), Modbus/*TCP* e *Simple Network Management Protocol* (SNMP) para implementar a comunicação entre a máquina, nesta secção inicia-se com a apresentação de fundamentos teóricos destes protocolos. Posteriormente é analisado métodos e comandos que permitem efetuar leituras de sensores e, no caso do SNMP, verificar o acesso a *Object Identifier* (OID) e *Management Information Base* (MIB) do sistema.

2.2.1 Modbus

Desenvolvido pela empresa Modicon em 1979, Modbus é dos mais antigos protocolos, mas dos mais usados para controlo industrial, tendo portanto, uma grande relevância na comunicação onde envolve *Programming Logic Controller* (PLC) [26]. É um protocolo amplamente conhecido devido à sua simplicidade, viabilidade e versatilidade, sendo então um protocolo utilizado tanto para sistemas embebidos de pequena escala como para rede industriais de grande escala. Atualmente a Siemens adquiriu a empresa fundadora deste protocolo, tornando-o um protocolo aberto, não requerendo assim licenças e permitindo que marcas de PLC distintas, consigam interagir entre si, destacando assim a importância da interoperabilidade.

Modbus define a estrutura da mensagem e regras de comunicação a serem usadas pelos sistemas de controlo de processo, de forma a permutar informação *Supervisory Control And Data Acquisition* (SCADA), de forma a operar e controlar processos industriais [26]. A partir de dois estilos distintos de comunicação: *query/response* (comunicação entre *master/slave*) ou do tipo *broadcast* onde o *master* envia uma mensagem para todos os seus *slaves* [27, 28]. Os dispositivos *master* iniciam as *queries*, quer sejam para transmissão, como também pode ser para monitorização, enquanto os dispositivos *slaves* respondem de acordo com a as *querie* que receberam. De salientar que os considerados *slaves* não podem iniciar a comunicação, estando exclusivamente à espera que o *master* inicialize o canal de comunicação. Este tipo de

comunicação tipicamente ocorre via comunicação série, nomeadamente RS-232 e RS-485 ou rede Ethernet, embora outros meios físicos sejam possíveis. A versatilidade deste protocolo é ainda mais amplificada pelas diversas extensões disponíveis, cada uma adaptada para atender a necessidades específicas de comunicação. No entanto, no âmbito do projeto corrente, apenas se entrará em detalhe nas principais extensões, sendo elas: Modbus série (RTU) e Modbus TCP/IP. Embora sejam extensões com processos diferentes,complementar ambas as extensões, beneficia a arquitetura de rede. Como tal, estudando cada uma das extensões é bem possível integrar ambas as extensões, de forma a obter uma rede mais robusta e escalável. Nos subcapítulos seguintes, apresentam-se breves conceitos relativamente a cada uma delas.

- **Modbus *Remote Terminal Units* (RTU)**

Nesta comunicação, apenas existe um *master*, por cada linha série, para múltiplos *slaves*. As mensagens entre dispositivos *master* e *slave* são transmitidas via linhas série, usando os modos de transmissão RTU ou *American Standard Code for Information Interchange* (ASCII) [26, 27] como ilustra a Figura 2.5.

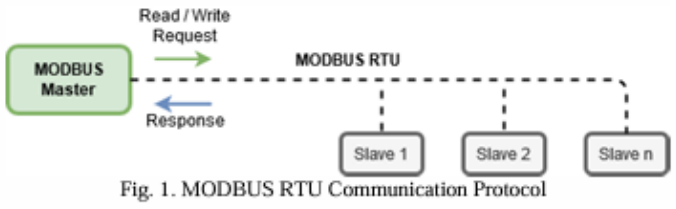


Figura 2.5: Comunicação Modbus RTU entre *master* e *slave* [29]

Em relação à estrutura da mensagem, são constituídas por três componentes [26]: *slave address*, aplicação de *Protocol Data Unit* (PDU)(que por sua vez é composto pelos *function codes* e *data field*) e um *error checking field*.

Começando pelo formato da mensagem consiste num pedido do *master* para os respetivos *slaves* e a respetiva resposta. Sendo uma comunicação bidirecional, ambas as mensagens são formatadas de acordo com especificações das mensagens do protocolo Modbus. Sendo assim, cada mensagem consiste numa série de *bytes* agrupados em quatro campos distintos:

- *Slave Address* - Numa mensagem do tipo *request* este parâmetro identifica o destinatário. Já o endereço correspondente numa mensagem de resposta, identifica o escravo que respondeu ao *request*. Mensagem *unicast* tem um intervalo de endereços compreendido entre [1,247][26, 27, 28].

- Código de função (*Function code*) - Cada trama ,de um *Byte*, ao realizar o *request* contém o código da função que define a ação esperada por parte do

controlador, ou seja, o significado do *request* depende do código preenchido no respetivo campo [27]. Já os códigos de função especificam: operações de leitura e escrita nos *slaves*, funções de diagnóstico e condições de erro. Modbus série possui três tipos de códigos de função: códigos públicos, códigos definidos pelo utilizador e códigos reservados. Os *function codes* podem ser de três tipos [26, 30]:

- Códigos públicos - Associados a funções bem definidas pela norma, validadas pela comunidade de MODBUS-IDA.org e que têm documentação pública disponível [30]. Os códigos públicos válidos estão alocados em intervalos compreendidos entre [1,64],[73,99], [111,127] [26].
- Códigos definidos pelo utilizador - Associados a funções alocadas nos intervalos entre [65,72] e [100,110], onde nestes intervalos o utilizador tem a possibilidade de definir as suas próprias funções. No entanto, não é possível garantir a compatibilidade com outras funções [26, 30].
- Código reservados - Associados a funções atualmente utilizadas por algumas empresas, possuindo o direito exclusivo do seu uso [30]. Códigos das funções compreendidos entre [128,255] indicam condições de erro em mensagens de resposta [26].

Segue-se a Tabela 2.1 de forma a analisar os principais *function codes* e respetiva descrição, visto serem essenciais para formatar a mensagem como já vimos anteriormente. O protocolo Modbus trabalha com dados discretos ou numéricos, e, do ponto de vista de *standards* são associadas algumas nomenclaturas individuais aos respetivos blocos de memória e permissões. Os valores discretos dizem respeito às "*coils*" e aos "*Discrete Inputs*", enquanto que os valores destinados a receber dados numéricos são os "*Holding Register*" e os "*Input Registers*"

Tabela 2.1: Function Codes [27]

Data Type	Absolute Addresses	Relative Addresses	Function Codes	Description
Coils	00001 to 09999	0 to 9998	01	Read Coil Status
Coils	00001 to 09999	0 to 9998	05	Force Single Coil
Coils	00001 to 09999	0 to 9998	15	Force Multiple Coils
Discrete Inputs	10001 to 19999	0 to 9998	02	Read Input Status
Input Registers	30001 to 39999	0 to 9998	04	Read Input Registers
Holding Registers	40001 to 49999	0 to 9998	03	Read Holding Register
Holding Registers	40001 to 49999	0 to 9998	06	Preset Single Register
Holding Registers	40001 to 49999	0 to 9998	16	Preset Multiple Registers
-	-	-	07	Read Exception Status
-	-	-	08	Loopback Diagnostic Test

- Campo de informação (*Data Field*) - varia em dimensão de memória de acordo com o código de função que foi previamente imposto no seu campo dedicado. Num *request* do *master* especifica a operação a realizar pelo *slave* [26]. No entanto, este mesmo campo, também pode, por exemplo, incluir informação que sabe de antemão que será necessária o *slave* possuir de forma a realizar a sua função. Ao contrário, numa resposta por parte do *slave*, este campo vem acompanhado com informação que o *master* lhe tinha requisitado [26, 27].

- Campo de verificação de erro (*Error Check Field*) - Nos últimos dois bytes é comprimido o campo para realizar a verificação de erros a partir do mecanismo de *Cyclic Redundancy Check* (CRC), que consiste num algoritmo usado para detetar erros na transmissão de informação. O algoritmo essencialmente realiza uma *check-sum* de verificação. A partir do valor obtido na *check-sum*, este valor é indexado aos dados a serem enviados. Do lado do recetor, apenas faz a verificação deste valor, e caso o valor seja idêntico, não houve erros detetados. Este método de verificação de erro assegura que os dispositivos da rede não irão reagir a mensagens que podem ter sido alteradas durante a transmissão, independentemente da razão pela qual foi alterada [27].

  As mensagens de resposta têm a mesma estrutura das mensagens de *request*. A especificação Modbus define as respostas em duas categorias: resposta positiva e resposta negativa. Quando a resposta é positiva, esta informa o *master* que o *slave* sucedeu a realizar a tarefa que lhe foi associada, e neste caso, o código da função é incluído na mensagem de resposta. Em contraste, uma resposta negativa, notifica o *master* de que a tarefa não foi possível ser realizada pelo *slave* a que foi endereçado esta tarefa (no campo do *slave address*). O código de função, para uma resposta negativa, é obtido a partir de somar o valor de 128 ao código de função da mensagem de *request*. Logo, códigos de função, compreendidos entre [128,255] indicam as condições de erro. Uma resposta negativa também inclui um código de exceção (como parâmetro de função) que oferece informação sobre a causa do erro [26].

Em sistemas de automação industriais, *Remote Terminal Units* (RTU) estão conectados a sensores e atuadores para interagir com a camada física da máquina. Sistemas de controlo industriais consistem num *Master Terminal Unit* (MTU) que estabelece conexão com as RTU's, através de uma ligação de comunicação [29].

A partir da contextualização do Modbus RTU que foi realizada, pretende-se agora transpor para um contexto prático, onde se analisam implementações para extrair dados de sensores e atuadores que estejam a comunicar via Modbus [29].

A partir de uma conexão modbus estabelecida, o MTU envia sinais de comando aos dispositivos finais, consoante o *slave addresses* de cada um. Após os *slaves* efetuarem a sua função, retornam um sinal à MTU. No modo RTU, o meio de

comunicação cada mensagem de 8-*bits* (1 *byte*) é encapsulada por: 1 *Start bit*, 8 *Data bits*, 0 ou 1 *Parity Bit* e 1 ou 2 *Stop bit* [29]. Como tal, o formato da trama de uma mensagem Modbus RTU é apresentada na Figura 2.6.

Address Code	*Function Code*	*Data*	*CRC-Check*
8 *bits*	8 *bits*	n x 8 *bits*	16 *bits*

Figura 2.6: Formato da trama de mensagem Modbus RTU [29]

De forma a Herath et. al [29] implementar o sistema de aquisição de dados teve em atenção a:

- Configurar os parâmetros de comunicação do dispositivo a utilizar para a comunicação baseada em Modbus RTU. Os parâmetros da comunicação são o ID do dispositivo, velocidade de transmissão, paridade, *data bit* e *stop bit*.
- Desenvolver um algoritmo para as *requests* e *responses* entre os dispositivos e MTU.
- Desenvolveu uma *Graphical User Interface* (GUI) a partir da linguagem de programação C#.

Após os parâmetros que estabelecem comunicação estarem definidos, existe a necessidade das mensagens *request* e *response*. A mensagem de *request* é composta por:

- Endereço do dispositivo - *Slave ID*.
- *Function Code* - Para ler *holding registers*, *Function Code* 03.
- Endereço inicial - registo Modbus onde a leitura é iniciada.
- Endereço de quantidade - número total de registos que se pretende ler.
- CRC - *checksum* para a mensagem a ser enviada.

Por sua vez, a mensagem de *response* é composta por:

- Endereço do dispositivo - *slave id*.
- *Function Code* - Função à qual o *slave* está a dar resposta.
- *Byte count* - Quantidade de *bytes* a serem lidos.
- *Register Values* - representa os valores lidos.
- CRC - *checksum* da mensagem recebida.

O estudo do protocolo Modbus RTU, parâmetros de implementação da comunicação e troca de mensagens Modbus, forneceu conhecimento para desenvolver métodos de leitura de dados provenientes de sensores que estabelecem comunicação Modbus.

- **Modbus *Transmission Control Protocol* (TCP)**

O protocolo Modbus TCP assegura conectividade numa rede, do tipo Modbus *Local Area Network* (LAN), composta pelo *master* e *slaves* e também pelas próprias redes IP interconectadas entre si, onde aqui, já aparecem múltiplos *masters*, cada um com múltiplos *slaves*. A razão pela qual Modbus TCP é capaz de abranger uma maior rede funcional, deve-se ao facto de estender o Modbus série, no aspeto de permitir a um *master* conseguir múltiplas trocas de mensagem com os *slaves* em simultâneo (pois aqui não há a ideologia do *bus* poder estar ocupado), e também, um *slave* conseguir interagir em comunicações com múltiplos *masters*, por exemplo, num cenário em que dois distintos *masters* precisavam ambos da mesma informação que podia ser apresentada exclusivamente por um *slave*, não tendo assim que estar à espera que o *slave* responda ao *master* que primeiro lhe solicitou tal informação.

Na Figura 2.7 é possível visualizar a conexão de múltiplos *slaves* numa rede IP. De realçar que o *master* também pode estar conectado a outros recursos no centro de controlo, como em *"databases"* e *"historians"*.

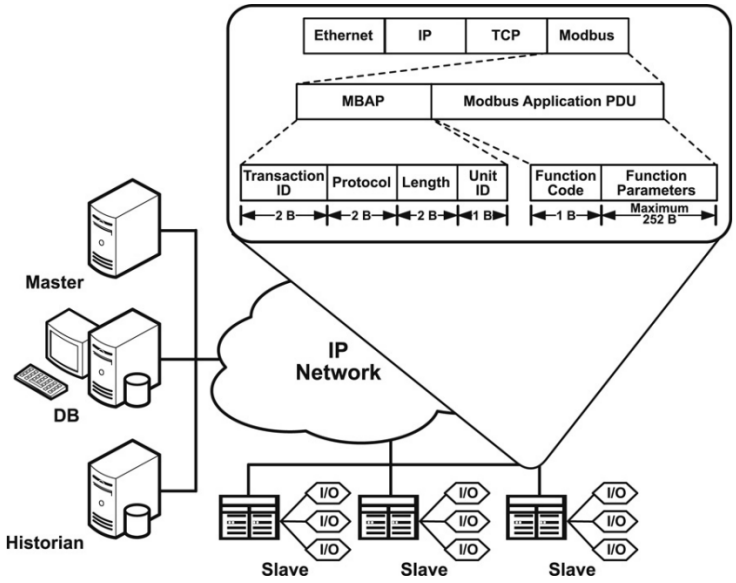


Figura 2.7: Modbus TCP/IP [26]

As trocas de mensagem são realizadas de forma idêntica às mensagens série. Ambos os dispositivos trocam informação sobre PDUs, mas presentemente é adicionado o encapsulamento na trama TCP [26]. Consequentemente, na extensão Modbus TCP, o campo do PDU inclui o *Modbus Application Protocol* (MBAP) em adição ao simples PDU utilizado no protocolo série.

Tal como se pode averiguar na Figura 2.7, o cabeçalho MBAP é composto por quatro campos: "*transaction identifier*", "*protocol identifier*", "*length*" e "*unit identifier*". Já o cabeçalho PDU é composto pelo "*function code*" e "*function parameters*" que são os dados necessários para efetuar a operação indicada pelo *function code*. Ou seja, caso se pretenda ler um registo, o *function code* é 03 e posteriormente, *function parameters* é composto pelo *starting register* do primeiro registo a ser lido e *quantity of register* que define o número total de registos que se vai ler, começando pelo *starting register* [26].

- Transaction Identifier - Permite aos dispositivos inicializar um canal de comunicação de acordo com os *requests* e respetivas respostas.
- Protocol Identifier - Indica o protocolo de aplicação encapsulado pelo cabeçalho MBAP (zero no caso Modbus).
- Lenght - Indica o tamanho, em *bytes*, dos campos restantes (unit identifier e PDU).
- Unit Identifier - Indica qual é o *slave* associado à transmissão.

Os campos de endereçamento e de verificação de erros varia de acordo com a tecnologia de transporte a estar a ser utilizada. No Modbus série, como visto anteriormente, contém o *Slave ID* e utiliza CRC para deteção de erro. No caso do Modbus TCP/IP, o campo de endereçamento é substituído pelo cabeçalho MBAP, enquanto que o campo de deteção de erro, aqui é removido pois as capacidades de deteção de erro do próprio protocolo TCP/IP é superior ao mecanismo de CRC, podendo este ser removido da trama [31]. Adicionalmente, na estrutura da trama Modbus TCP/IP, é chamada de *Application Data Unit* (ADU) que encapsula o cabeçalho MBAP e o campo definido pelo PDU [31].

Num contexto de implementação de leituras de dados provenientes de dispositivos Modbus TCP, Mageshkumar et al. [32] recorreu ao uso de um PLC e um arduino. O PLC foi configurado como Modbus *slave*. Este *setup* envolve ativar o Modbus via Ethernet, atribuindo um específico endereço IP ao PLC. O autor desenvolveu o programa *ladder logic* do PLC de forma a que este seja capaz de ler valores de sensores a partir do seu módulo analógico. Por sua vez, os valores lidos são guardados nos *holding registers* do PLC, onde podem ser acedidos para futuros *requests* [32]. Já o arduino foi configurado como Modbus *master*, conectando-se ao *port Ethernet* do PLC, de forma a permitir a comunicação entre os dois dispositivos.

Após ambos os dispositivos estarem configurados, o arduino estabelece conexão ao enviar uma trama *request* para o PLC. Este *request* é bem definido e estruturado, incluindo campos como: *transaction identifier*, *protocol identifier*, *unit identifier*, *function code*, *starting address* do primeiro conjunto de dados a ser acedido e finalmente o *number of registers* a serem lidos. Esta trama, ilustrada na Figura 2.8

garante que o PLC interpreta corretamente o *request* recebido e retorna uma *response* apropriada.

Frame Count	Request Indication	Slave ID	Function Code	Starting Address Register	Register Count

Figura 2.8: Formato da trama *request* [32]

Por sua vez, a *response* do PLC retorna uma trama que contém a informação pretendida proveniente dos seus *holding registers*. A Figura 2.9 apresenta os campos que definem a *response* do Modbus TCP. Por fim, os valores que foram acedidos aos *holding registers* do PLC, são apresentados ao utilizador no *serial monitor* do Arduino IDE, fornecendo os dados em tempo real.

Frame Count	Response Indication	Slave ID	Function Code	Starting Address Register	Float Data DC	Float Data BA	CRC

Figura 2.9: Formato da trama *response* [32]

O estudo do protocolo Modbus TCP, parâmetros da comunicação, estrutura das mensagens Modbus, e uma implementação prática forneceu conhecimento para desenvolver métodos de leitura de dados provenientes de sensores que estabelecem comunicação Modbus TCP.

2.2.2 SNMP

Simple Network Management Protocol (SNMP) introduzido em 1988 é um protocolo destinado a dar resposta à necessidade de monitorizar e gerir dispositivos como: switches, routers, firewalls, servidores, dispositivos sem fio, entre outros, que conecta a rede via *Internet Protocol* (IP). SNMP fornece a possibilidade de múltiplos utilizadores gerirem, de forma remota a partir de uma série de funções simples. Por exemplo, SNMP pode ser usado para monitorizar a temperatura de um *switch* e avisar o gestor da rede quando tiver uma temperatura demasiado elevada.

É um protocolo a nível da camada de aplicação definida pela *Internet Architecture Board* na RFC 1157 (evolução do RFC 1098). As organizações usam SNMP para monitorizar e gerir dispositivos quer a nível local como *Local Area Network* (LAN), quer a nível mais abrangente como *Wide Area Network* (WAN).

Na arquitetura do SNMP existem dois elementos: gestores e agentes. Um gestor, neste contexto, é um servidor que está a correr algum algoritmo capaz de gerir tarefas para uma rede, daí, por vezes serem intitulados de *Network Management Station* (NMS). NMS é o responsável por realizar sondagens e receber *traps* pelos agentes na rede. Uma sondagem, no contexto de gestão de redes, é o ato de fazer um pedido de algum tipo de informação que lhe seja necessária [33, 34]. Por sua vez, a *trap* é um

mecanismo que permite ao agente, notificar o NMS responsável que algo aconteceu na rede. Esta notificação pode ser enviada para o NMS sem a necessidade de haver uma autorização para iniciar transmissão com o responsável da rede. Como tal, é da responsabilidade da NMS tomar uma ação consoante a informação que recebeu vinda do agente.

Já o agente, é uma porção de software que corre nos dispositivos que estão a ser geridos pelo respetivo NMS[33, 34]. Pode ser um programa separado ou pode ser um programa que está incorporado num sistema operativo. Hoje em dia, grande parte dos dispositivos IP vem com algum tipo de agente SNMP já embebido [33]. Só o facto dos fornecedores de tais dispositivos colocar tal protocolo no seu produto já expõe a relevância deste protocolo e tornando o trabalho do administrador da rede mais acessível, uma vez que não tem que estar a configurar o dispositivo. Após a resolução de um possível problema (por parte do NMS, visto estes serem os responsáveis de tomar ação), muitas vezes, o agente reenvia uma *trap* "*all clear*" de forma a esclarecer que qualquer que fosse o problema, este já está resolvido e assim a rede está a operar de forma eficaz [33].

É importante de realçar, que neste protocolo tanto o agente como o NMS podem comunicar quando lhes convém, não tendo que estar à espera de autorização (a sondagem e a *trap* podem inclusive ocorrer em simultâneo).

Outros elementos de grande importância estrutural deste protocolo são: *Structure of Management Information* (SMI) e *Management Information Base* (MIB). SMI fornece uma mforma de definir os objetos que estão a ser supervisionados e o respetivo comportamento. Cada agente tem em sua posse uma lista, que se entende por ser o coletivo de informação respetiva dos objetos que supervisiona [33]. "Objeto"pode ser a interface do estado operacional de um router, por exemplo.

Já o MIB pode ser considerado como uma base de dados de objetos geridos que o agente supervisiona, sendo então uma estrutura que define o formato da informação a ser transmitida num sistema SNMP. Qualquer tipo de estado ou informação estatística que possa ser acedida pelo NMS é definida pelo MIB [33].

SNMP usa o *User Datagram Protocol* (UDP) como protocolo para transmitir dados entre os gestores e os agentes. Efetivamente UDP substitui o TCP para transmissão de dados, uma vez que, não há conexão *end-to-end* entre o agente e NMS quando as tramas são transmitidas entre entidades [33]. Sendo que a comunicação entre NMS e o agente é assíncrona, percebe-se que o protocolo UDP é mais adequado, ao invés do TCP, sendo que, pelo menos o UDP permite que os dispositivos enviem e recebam mensagens sem a necessidade de estabelecer uma conexão de longa duração, diminuindo assim a latência de comunicação. Para além desta razão, encapsular em TCP é mais complexo e exige mais recursos de hardware e software do que o UDP e o SNMP é frequentemente usado em dispositivos de rede com recursos limitados, onde a eficiência de recursos é fundamental.

O UDP por si só não garante a entrega ou *acknowledgment* dos datagramas terem sido recebidos com sucesso ao destino. No entanto, o SNMP pode implementar mecanismos de reenvio de dados, caso o datagrama tenha sido perdido, a partir de um *timeout*. O NMS envia um *request* UDP a um agente e fica à espera de resposta. O intervalo de tempo em que o NMS fica à espera depende da configuração prévia para o máximo tempo aceitável de espera [33]. Em casos que o tempo máximo de espera é ultrapassado, o NMS reenvia a mensagem e assim em diante e pode ser também configurado o número de vezes que o NMS reenvia a mesma mensagem.

Mecanismos como os anteriormente mencionados sugere que o UDP acaba por conseguir ter um certo controlo de confirmações de mensagem, pois no pior dos casos o NMS envia um *request* e nunca obtém resposta [33]. No entanto, para o caso das *traps* (relembrando que entende-se pela mensagem enviada pelo agente para o gestor quando ocorre um evento alarmante) é diferente. Se o agente enviar uma *trap* e a *trap* nunca chega ao destino, o NMS não tem como saber que a mensagem foi sequer enviada por parte do agente [33]. Para além de que o agente fica num ponto de indecisão pois não tem como saber se tem que reenviar a *trap* pois o NMS não tem como requisito confirmar a receção da mensagem vinda do agente. De salientar que o UDP por si não fornece garantias de entrega, em sistemas que exigem confirmação de entrega, esses mecanismos devem ser implementados na *layer* de aplicação, como no caso do SNMP.

SNMP usa o port 161 para enviar e receber *requests* e o port 162 para receber *traps* pelos dispositivos geridos. SNMP funciona ao enviar trama de dados, conhecidos como SNMP GET *requests* para a rede. Estas comunicações são monitorizadas e as ferramentas de gestão usam os GET *requests* para extrair dados do SNMP. De notar que o fluxo de dados dentro da rede tem diversas origens e é da responsabilidade do SNMP gerir toda a rede e os dispositivos nas mesmas.

Segue-se a Figura 2.10 que demonstra o encapsulamento das camadas do protocolo e também o mecanismo da troca de *requests* e *responses*. Importante notar que nesta disposição de encapsulamento, cada camada usa a informação que advém da camada imediatamente acima e realiza uma ação/serviço na camada imediatamente abaixo.

Quando um NMS ou agente desejam realizar uma função SNMP os seguintes eventos ocorrem os seguintes eventos:

- *Application* - Primeiramente a aplicação SNMP decide que operação deseja realizar. Por exemplo, pode enviar um SNMP *request* para um agente, ou o agente enviar uma *trap* para o NMS. Tal operação é alocada na camada de aplicação.

- UDP - Permite que dois *hosts* comuniquem entre si. O cabeçalho UDP contém informações sobre o port do dispositivo de destino, relembrando que no caso

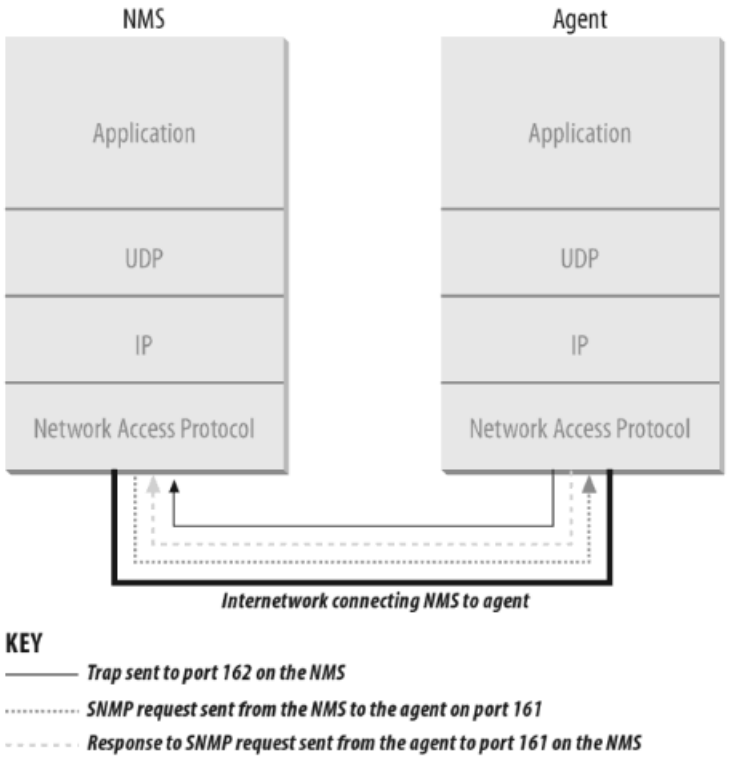


Figura 2.10: SNMP *protocol stack* [33]

do SNMP será o port 161 para *request* e 162 para *trap*.

- IP - Esta camada entrega a trama SNMP para o destino desejado, especificado pelo enderço IP.
- *Media Access Control* (MAC) - Neste último evento, é onde efetivamente a informação chega ao destino físico. A camada MAC é composta pelo próprio hardware que configurado através de software consegue alocar os dados num elemento físico como por exemplo um fio de rede, ou um cabo ethernet. A camada MAC também é responsável pela mensagem inversa, ou seja, recebe informação pela rede física e reencaminha a informação pelo encapsulamento acima, chegando eventualmente à camada superior.

Como mencionado anteriormente, SNMP é previamente configurado nos dispositivos, e quando o protocolo está ativo, os dispositivos vão alocar dados estatísticos de performance nos ficheiros MIB. Na prática, os GET *requests* vão extrair os dados alocados nos MIBs de forma a permitir que a rede funcione em concordância com a performance que está a ter. Sendo assim, o SNMP tem comandos que ajudam a extrair, gerir, modificar e transmitir dados, tornando a gestão da rede mais acessível. Os comandos de leitura de dados provenientes do agente são:

- GET - o GET *request* é iniciado pelo NMS, que envia um *request* para o agente. O agente recebe o pedido e dá a melhor resposta possível, isto porque alguns dispositivos estão sobrecarregados, como por exemplo os *routers*. Nos instantes que os dispositivos estão sobrecarregados não enviam qualquer tipo de informação para o NMS (o que fará com que o NMS reenvie o pedido GET mais tarde). Caso o agente consiga reunir os dados desejados, envia resposta de volta para o NMS. Como é que o agente sabe que informação o NMS está à procura? A partir da uma variável de valor que identifica exatamente o que o NMS procura. Variável de valor pode ser vista como associação de parâmetros de forma simples do tipo *Object Identifier* (OID) = *value* o que torna fácil a extração dos dados realmente desejados [33].

- GET NEXT - Similar ao comando GET, mas usado para solicitar a próxima instância de uma variável. Visto que a procura começa no topo do SMI e vai pela lista abaixo até encontrar a instância especificada pelo OID. O NMS continua a responder até enviar uma mensagem de erro, significando que os objetos retidos no MIB já foram todos enviados.

- GET BULK - Comando que permite à aplicação de gestão obter uma extensa tabela de informação de uma só vez. Como tal, por vezes até adquire mais que um objeto MIB de uma vez, no entanto, há que ter cuidado e consideração com o tamanho máximo de transmissão de dados, por parte do agente, que conseguem ser enviados de uma vez. Caso não consiga enviar a informação pedida de uma só vez (devido a razões de tamanho da mensagem) acaba por não enviar nada e retorna um erro. Ou seja, é uma operação que pode provocar congestionamento no fluxo de dados. No entanto, caso seja bem configurado, este comando pode ser muito útil. Por exemplo, se dividir em dois campos distintos: *nonrepeaters*(N) e *max-repititions* (M). Os *nonrepeaters* indicam ao comando GET BULK que os primeiros N objetos podem ser extraídos a partir da simples operação GET NEXT. Já as *max-repititions* indicam ao comando GET BULK para tentar usar o comando GET NEXT até um máximo de M vezes, de forma a tentar extrair a informação restante [33].

Capítulo 3

Arquitetura do sistema

No presente capítulo é exposto o conjunto de hardware utilizado no contexto deste projeto. Explora-se os detalhes técnicos e as especificações de cada componente. Para além disso, também são expostas as tecnologias utilizadas para, posteriormente, ser possível compreender a integração dos componentes na arquitetura do sistema final, composto por diversos componentes, destacam-se: atmega328P e *multiplexer*.

3.1 Funcionalidades do sistema

Tal como foi explicitado no Subcapítulo 1.2 os requisitos do sistema focam-se na necessidade de desenvolver um sistema capaz de realizar o teste de continuidade na cablagem da máquina de forma automatizada. Neste teste de continuidade é necessário a deteção de continuidade, a falta de continuidade e a deteção de fios trocados baseado nas ligações elétricas expectáveis da máquina, que a empresa disponibilizou através de um esquema das conexões elétricas da máquina. Associa-se também a necessidade de disponibilizar ao operador a identificação do fio testado, de forma a informá-lo da necessidade da correção no caso de fios com pinos trocados, ou substituição no caso de fios danificados. Para além disso existe a necessidade de efetuar leituras de dados provenientes de sensores que utilizam Modbus RTU, Modbus TCP e SNMP. No caso do SNMP é importante salientar que esta leitura de dados apenas averigua o correto funcionamento do "agente", um dos elementos envolventes no protocolo. De forma a dar resposta ao teste de continuidade automatizado implementou-se um sistema, composto por microcontrolador, *multiplexers*.

Posteriormente desenvolveram-se cabos de teste, de forma a ser possível conectar as ligações elétricas da máquina ao sistema responsável pela realização do teste de continuidade. Com o propósito de apresentar os testes de continuidade desenvolveu-se uma *Graphical User Interface* (GUI), que comunica via *Universal Asynchronous Receiver Transmitter* (UART) com o microcontrolador, apresentando os resultados do teste de continuidade, destinguindo casos de continuidade, não continuidade e fios trocados. Para além dessa funcionalidade da GUI, é esta que também efetua leituras de dados provenientes dos sensores baseados nos protocolos de comunicação Modbus RTU, Modbus TCP e SNMP.

Durante o Capítulo 2 evidenciou-se uma distinção entre os testes de continuidade elétrica na cablagem da máquina e os testes de funcionalidade, de forma a averiguar se as características operacionais dos sensores estão ativas. Esta distinção deve-se ao facto de serem testes com diferentes propósitos, no entanto, ambos os testes complementam-se. Isto é, na máquina da empresa é necessária a verificação tanto da cablagem elétrica como a verificação do correto funcionamento dos sensores presentes na máquina. De forma a validar a cablagem elétrica, o teste de continuidade elétrica é o suficiente. No entanto, de forma a verificar o funcionamento dos sensores, estes têm as suas ligações elétricas (camada física) verificada pelo teste de continuidade e complementa-se com a realização de leitura de dados provenientes dos sensores. Ou seja, caso as ligações elétricas associadas aos sensores se verifiquem com continuidade elétrica, e as suas conexões foram estabelecidas como previsto, durante a fase de produção da máquina existe a necessidade do teste complementar de forma a verificar se as características operacionais dos sensores estão ativas. Sendo que os sensores presentes na máquina são baseados em Modbus ou SNMP, efetuam-se as leituras de dados respeitando a estrutura e os parâmetros do protocolo associado a cada sensor.

Tanto o teste de continuidade, como o teste funcional podem ser realizados de forma independente ou de forma complementar. No entanto, em zonas da máquina que presenciam fios elétricos em conjunto com sensores, o teste de continuidade, em cojunto com o teste de funcionalidade, permite detetar a origem de um possível erro na configuração dos sensores. De seguida são abordados estes três casos de uso:

- Teste de continuidade na cablagem elétrica.
- Teste de funcionalidade onde efetua leituras de dados provenientes de sensores baseados nos protocolos Modbus ou SNMP.
- A integração do teste de continuidade com o teste de funcionalidade, originando o "teste complementar", permitindo a deteção de erros de configuração do sensor.

A Figura 3.1 ilustra as quatro etapas do teste de continuidade elétrica. A primeira etapa envolve o *setup* do teste de continuidade. Ou seja, de forma a ser possível

testar a continuidade elétrica da cablagem da máquina, existe a necessidade de criar uma interface entre a cablagem da máquina com o sistema responsável pelo teste de continuidade. Esta interface é realizada a partir de cabos de teste. É importante definir que os "cabos de teste" não são os fios que se tenciona testar a continuidade. Os cabos de teste, são os cabos que permitem prolongar os fios elétricos da máquina até ao sistema responsável por efetuar o teste de continuidade. Devido aos diversos conectores que a máquina possui, estes cabos de teste, têm na sua extremidade, os conectores correspondentes aos já existentes na máquina. Este bloco remete ao conjunto de cabos de teste e respetivos conectores, associados a cada zona da máquina a ser testada. A segunda etapa é a seleção da zona da máquina que se pretende testar, de acordo com o *setup* efetuado previamente. Por sua vez, a GUI estabelece comunicação UART com o microcontrolador. Após a conexão estar estabelecida, a GUI envia uma mensagem, informando o microcontrolador qual o teste que se pretende realizar. Na terceira etapa, o sistema responsável pelo teste de continuidade, constituído pelo microcontrolador e *multiplexers* tem de ser capaz de efetuar o teste de continuidade nos diferentes fios elétricos da zona selecionada. Por sua vez, o microcontrolador tem de receber e interpretar a mensagem proveniente da GUI, de forma a saber qual a zona da máquina a testar a continuidade. Após efetuar o teste de continuidade, o microcontrolador também é responsável por formatar e transmitir a mensagem com o resultado do teste de continuidade para a interface gráfica. Na quarta etapa a interface gráfica tem de ser capaz de extrair o resultado dos testes de continuidade, transmitidos pelo microcontrolador. Posteriormente, há a necessidade de descodificação do formato da mensagem e apresentação dos resultados de forma a permitir ao operador validar e confirmar o teste de continuidade realizado.

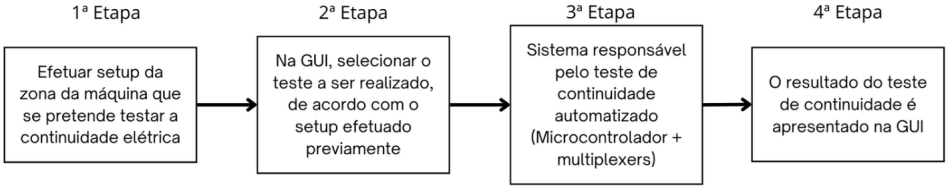


Figura 3.1: Etapas do teste de continuidade

Relativamente ao teste de funcionalidade, a Figura 3.2 ilustra as quatros etapas. A primeira etapa diz respeito ao utilizador garantir que o sensor está a ser alimentado. No caso de sensores Modbus TCP e SNMP, os sensores e o computador que está a correr o executável da aplicação GUI têm de estar na mesma *network*. Na segunda etapa, o utilizador introduz os parâmetros que são solicitados, relativamente ao sensor que se pretende efetuar leituras de dados. Estes parâmetros solicitados são os que permitem estabelecer conexão e instruir, na mensagem de *request*, as ações que o *master* (GUI) pretende que o *slave* (sensor) execute. Os parâmetros solicitados pela GUI, são os campos necessários da trama *request*, como foi revisto

no Subcapítulo 2.2. Sendo que, consoante o sensor que está a ser testado, os parâmetros solicitados são associados ao protocolo utilizado pelo sensor. Já na terceira etapa, caso os parâmetros introduzidos estejam corretos, a conexão entre o *master* (GUI) e o *slave* (sensor a ser testado) é estabelecida e posteriormente efetua-se a leitura dos dados. Finalmente, na quarta etapa, os dados recebidos são apresentados na GUI.

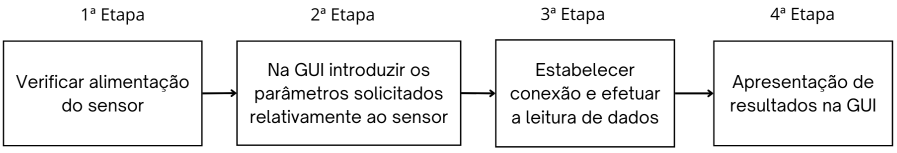


Figura 3.2: Etapas do teste de funcionalidade

Por fim, apresentam-se as etapas do teste complementar. Ou seja, quando se realiza o teste de continuidade elétrica na zona da máquina onde se encontram os sensores, e caso se verifique a continuidade nos fios do sensor, sucede-se a leitura de dados do sensor. Caso haja continuidade nos fios elétricos associados ao sensor (fios de alimentação, sinal, rede), mas, não seja possível efetuar a leitura de dados do sensor, diagnostica-se assim um problema de configuração no sensor. Reforçando mais uma vez, a intenção deste teste complementar é de detetar erros de configuração do sensor, não sendo o objetivo de identificar a causa nem efetuar o conserto do mesmo. A mais valia de diagnosticar um possível erro de configuração, permite ao operador, ainda na fase de produção, trocar o sensor danificado por um sensor que esteja operável. De salientar que este teste complementar é a junção do total das etapas anteriormente mencionadas. No entanto, com a intenção do diagrama de blocos não ficar exageradamente extenso, as três etapas na Figura 3.3 estão mais generalizadas. A primeira etapa está associada ao *setup* do teste de continuidade na zona da máquina que contém o sensor que se pretende verificar a configuração. O microcontrolador em conjunto com os *multiplexers* efetua o teste de continuidade e envia o resultado do teste de continuidade para a GUI, e são apresentados na interface gráfica. Na segunda etapa o utilizador introduz os parâmetros solicitados, e, por sua vez, a GUI estabelece conexão com o sensor e efetua a leitura de dados. A terceira etapa é relativa à apresentação do teste de continuidade e do teste de funcionalidade na GUI. Assumindo que o operador introduziu os parâmetros corretos relativamente a informações do sensor e se não conseguir visualizar os dados lidos, mas em simultâneo, verifica que o teste de continuidade confirma a integridade de todos os fios, concluirá que há um erro de configuração no sensor, o que resultará na sua substituição.

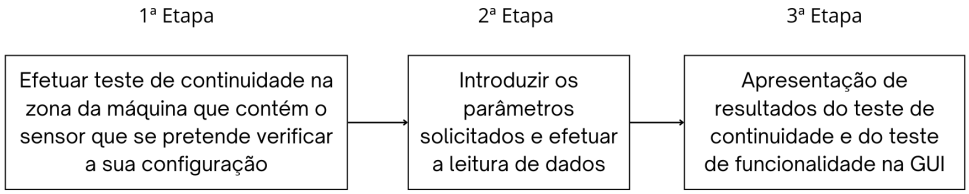


Figura 3.3: Etapas do teste complementar

3.2 Hardware Utilizado

De forma a permitir a implementação do sistema responsável por efetuar o teste de continuidade, recorreu-se ao uso do microcontrolador atmega328p, *multiplexers* e cabos de teste. Para além disso, de forma a estabelecer uma conexão ente o sistema responsável pelo teste de continuidade e o computador que contém a interface gráfica, utilizou-se o conversor usb/serial FT232 que permite a comunicação via *Universal Asynchronous Receiver Transmitter* (UART).

De seguida, abordam-se os componentes utilizados e as suas especificações técnicas, necessárias de conhecer, de forma a ser possível perceber a integração entre os periféricos no sistema responsável por efetuar o teste de continuidade.

3.2.1 Atmega328P

O Atmega328P é um microcontrolador de 8 bits da família AVR, fabricado pela Microchip Technology (anteriormente pela Atmel Corporation). É dos microcontroladores mais utilizados para projetos eletrónicos como em sistemas embebidos. A principal característica que distingue o Atmega328 do Atmega328P é o consumo de energia. Na própria nomenclatura, chama logo à atenção o acréscimo da letra "P". No Atmega328P, a letra "P", indica a presença de um pino de alimentação (*Power*) de energia de baixa potência (PWRDOWN). Devido a este pino, o Atmega328P, tem a opção de ser utilizado em modo de baixo consumo de energia, caso necessário, ao contrário do Atmega328 que não possui tal opção.

Atmega328P baseia-se numa arquitetura modificada de Harvard de 8 bits. “Modificada”, pois, permite que a partição da memória que contém as instruções, possa ser acedida. Deste hardware, vale a pena salientar as seguintes características [35]:

- Memória: Contém memória de programa Flash de 32 KB onde o código é armazenado. Conta com 2 KB de memória RAM para armazenamento temporário de dados durante a execução do programa. Por fim, inclui 1 KB de memória EEPROM onde pode ser usada para armazenar dados persistentes mesmo quando o microcontrolador é desligado.

Tabela 3.1: Características do Atmega328P

Características	**Descrição**
Arquitetura	AVR de 8 bits
Memória Flash	32KB
Memória RAM	2KB
Memória EEPROM	1 KB
Clock	Até 20 MHz
Pinos GPIO	23 (Incluindo pinos analógicos)
Comunicação Série	USART/I2C/SPI
Temporizadores	3 (16-bit)
Comparadores Analógicos	1

- Clock e Temporizadores: Opera com frequência de processamento máximo até 20MHZ. Possui três temporizadores/counters, interrupções internas e externas de clock e PWM.
- Comunicação: Suporta comunicação série USART, SPI (Serial Peripheral Interface) e I2C (Inter-Integrated Circuit).
- GPIO (General-Purpose Input/Output): Possui 23 pinos GPIO para entrada e saída digital, alguns dos quais podem ser utilizados como pinos analógicos.
- ADC (Conversor analógico-digital): Possui conversor analógico-digital de 10 bits integrado, permitindo a leitura de sinais analógicos.
- Interrupções: Oferece um sistema de interrupções de eventos.

Na Tabela 3.1 é possível realçar as características principais deste microcontrolador.

De forma a concluir os pormenores deste microcontrolador, segue-se a Figura 3.4 onde é representado o *PINOUT* deste microcontrolador.

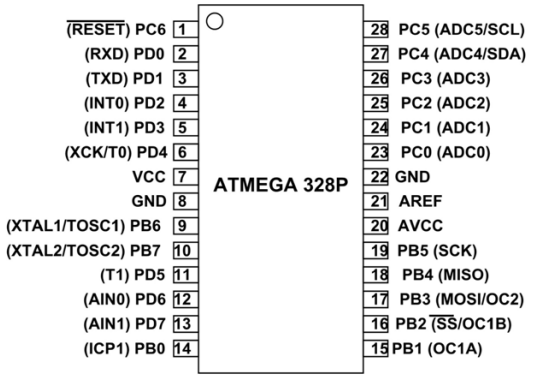


Figura 3.4: *PINOUT* de Atmega328P [35]

A escolha deste componente deve-se ao seu baixo custo financeiro e por ser capaz de exercer as funções expectáveis para efetuar um teste de continuidade em fios elétricos. No sistema encarregado por efetuar o teste de continuidade, o atmega328p é o elemento principal, essencialmente devido às suas capacidade de emitir sinais digitais, detetar a sua receção e de permitir a conexão série, de forma a transmitir os dados resultantes do teste de continuidade. Embora microcontroladores como o atmega2560, quando comparado com o atmega328P, possua um maior número de pinos digitais (54 pinos digitais) mesmo assim continua a não ter pinos suficientes para testar a continuidade de toda a cablagem elétrica da máquina e seria mais do dobro do custo financeiro. De forma a colmatar a necessidade de pinos digitais, recorreu-se ao uso de *multiplexers*, permitindo assim aumentar o número de fios a serem testados em simultâneo.

3.2.2 Módulo multiplexer CD74HC4067

O *multiplexer/demultiplexer* CD74HC4067 é um módulo eletrónico capaz de operar de duas formas distintas, mas baseando-se no mesmo princípio e modo de operação. No funcionamento como *multiplexer*, ao ser conectado a um microcontrolador permite que o microcontrolador selecione várias entradas (analógicas/digitais) e forneça uma única saída (analógica/digital). Neste modo de funcionamento, é capaz de a partir de 2^n linhas de entrada obter uma linha de saída, onde "n"é o número de linhas de seleção. Em contraste, no funcionamento como *demultiplexer*, a partir de único sinal de entrada e direciona-o para uma de dezasseis linhas de saída. No contexto deste projeto, o componente CD74HC4067 está a operar como *demultiplexer* devido ao número limitado e insuficiente de pinos, aumentando assim o número de pinos de forma a dar respota aos testes de continuidade. A partir de quatro pinos do atmega328p, conectam-se aos quatro pinos do *demultiplexer* aos pinos de controlo do demultiplexer S0 a S3 (n = 4) e um pino SIG que é o pino de entrada/saída comum, este pino é o responsável para selecionar em qual dos dezasseis (n = 4: 2^4) canais o sinal aparecerá. Ou seja, dependendo da combinação binária atribuída aos pinos de controlo, um dos dezasseis canais é conectado ao canal comum (SIG). Na Tabela 3.2 é possível verificar a manipulação binária a realizar nos pinos de controlo, de forma a obter determinado canal final. A título de exemplo, imagine-se que se pretende direcionar o sinal comum (SIG) para o canal 8 (CH8), a partir do atmega328p configuram-se os pinos de controlo S3 com valor binário um, e os restantes pinos de controlo: S2, S1, S0 com valor binário zero.

Tabela 3.2: Combinação binária para seleção de canal pretendido

EN	**S3**	**S2**	**S1**	**S0**	**Canal Selecionado**
1	X	X	X	X	Nenhum (desabilitado)
0	0	0	0	0	CH0
0	0	0	0	1	CH1
0	0	0	1	0	CH2
0	0	0	1	1	CH3
0	0	1	0	0	CH4
0	0	1	0	1	CH5
0	0	1	1	0	CH6
0	0	1	1	1	CH7
0	1	0	0	0	CH8
0	1	0	0	1	CH9
0	1	0	1	0	CH10
0	1	0	1	1	CH11
0	1	1	0	0	CH12
0	1	1	0	1	CH13
0	1	1	1	0	CH14
0	1	1	1	1	CH15

Na Tabela 3.3 é possível verificar de forma resumida as características mais importantes sobre este componente.

Tabela 3.3: Características do CD74HC4067

Características	**Descrição**
Tensão de operação	2V a 6V
Número de canais	16, CH0 a CH15
Número pinos de controlo	4, S0 a S3
Número pino comum	1, SIG

A seleção deste equipamento, quando comparado com outros *multiplexers* encontrados no mercado deve-se ao facto de possuir 16 pinos de seleção e por ser amplamente utilizado na indústria, devido à sua versatilidade fiabilidade. Este componente é integrado no sistema responsável pelo teste de continuidade, devido ao facto de a partir das suas 16 saídas, permitir apoiar o teste de continuidade num maior número de fios elétricos em simultâneo, aumentando assim a velocidade e capacidade numérica da execução do teste de continuidade. Por meio do operador, os cabos de teste são conectados aos pinos dos *multiplexers*, de forma a ser possível realizar o teste de continuidade.

3.2.3 Conectores implementados nos cabos de teste

Em relação aos conectores utilizados, de forma a realizar o teste de continuidade em cada cabo, utilizaram-se os conectores SP e JDS. Devido ao número de pinos pretendidos, utilizaram-se os conectores SP: SP2912/S24, SP2912/S17, SP2912/S3,SP2112/S7, SP2112/S4, SP2112/S2, SP1712/S9, SP1712/S4, SP1712/S2. Os conectores JDS utilizados foram JDS25, JDS29 e JDS37. De acordo com as especificações do fabricante, de forma a compreender o significado do nome dos conectores, use-se o exemplo de um conector SP2912/S24.

- SP - Significa que este produto é relativamente ao modelo SP dos diversos modelos de conectores do fabricante;
- 29 - É o diâmetro do corte do conector, em mm.
- 12 - Nesta parte do nome do conector, pode variar entre 12 e 10. Quando o número é 12 significa que é um conector de chassi como o da Figura 3.5a ou caso seja o número 10 indica um conector de ficha.
- S - Este conector é do tipo fêmea. Caso estivesse um 'P' significaria que é um conector macho.
- 24 - Indica o número de pinos que este conector possui.

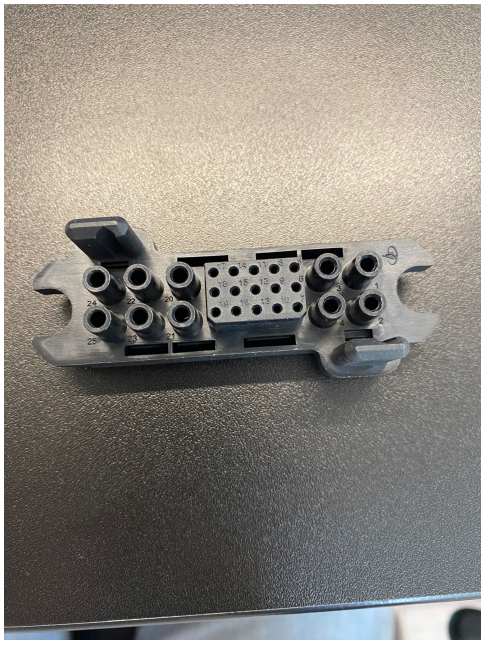
No caso dos conectores JDS a única variante é o número que sucede, indicando o número de pinos que o conector suporta. Segue-se na Figura 3.5a o exemplo de um conector do tipo SP e na Figura 3.5b um conector do tipo JDS.

Estes conectores abordados, são os utilizados na máquina da empresa. Consequentemente, estes conectores são utilizados em ambas as extremidades do cabo de teste desenvolvido, de forma a permitir estabelecer uma interface entre os terminais da máquina e os pinos de seleção dos *multiplexers*. É importante salientar que, num dado terminal da máquina, caso este seja do tipo "Macho", no cabo de teste desenvolvido, para a mesma localização, terá que se implementar o mesmo conector mas na versão "Fêmea".

3.2.4 FT232

O FT232, apresentado na Figura 3.6, é fabricado pela empresa *Future Technology Devices International* (FTDI) e permite a conversão de sinais USB em sinais série como a UART [36]. São amplamente utilizados, de forma a permitir a comunicação entre o microcontrolador e o computador, visto que os computadores modernos não têm porta série. O *chip* FT232 permite que o microcontrolador se conecte diretamente à porta USB do computador, sem necessitar de modificar o hardware existente.

(a) Conector SP



(b) Conector JDS

Figura 3.5: Conectores

O FT232 é o componente encarregado de estabelecer a comunicação entre o microcontrolador e o computador que aloca a interface gráfica, via UART. É a partir deste componente que a interface gráfica é capaz de comunicar com o microcontrolador. O FT232 permite transmitir a informação sobre o teste de continuidade a ser realizado instruído pela interface gráfica. Posteriormente ao teste de continuidade ser realizado, o FT232 permite transmitir o resultado do teste de continuidade, de forma a ser possível a receção do resultado por parte da interface gráfica, apresentando de seguida os resultados para o operador.

3.3 Tecnologias Utilizadas

Neste capítulo exploraram-se as *frameworks* existentes para a programação do microcontrolador e para o desenvolvimento da framework que será utilizada como *Graphical User Interface* (GUI). Para tal, inicialmente serão apresentadas algumas *frameworks* disponíveis, destacando as suas características e áreas de aplicação. De seguida, é realizada uma análise comparativa entre as opções disponiveís, indicando e justificando o porquê da decisão final. De forma a auxiliar esta análise, será exposta uma tabela comparativa, que permite resumir as características principais de cada opção. Por fim há uma explicação da comunicação série UART que se utilizou de forma a permitir a comunicação entre o sistema responsável pelo teste de continuidade (via microcontrolador) e o computador que aloca a interface gráfica.

Figura 3.6: Componente FT232

3.3.1 *Integrated Development Environment* (IDE) para Atmega328p

Optou-se pelo IDE Microchip Studio para a integração com o microcontrolador atmega328p, devido a este IDE ter sido desenvolvido pelos próprios fabricantes dos microcontroladores AVR, assegurando assim uma integração direta e com suporte robusto, proporcionando uma compatibilidade fiável com o hardware em questão. A familiaridade com esta ferramente também foi um pormenor impulsionador para a escolha deste IDE. É uma ferramenta amplamente utilizada na indústria, assegurando a confiabilidade necessária. Oferece acesso direto a um vasto conjunto de bibliotecas e documentação específica para os microcontroladores AVR, acelarando o processo de desenvolvimento. Em contraste a IDEs genéricos como Eclipse e PlatformIO, estes podem necessitar de configurações adicionais e soluções baseadas em GCC.

3.3.2 Escolha framework para desenvolvimento GUI

Neste sub-capítulo aborda-se a escolha da *framework* e fundamentos que justificam a sua escolha. Em relação à *framework* escolhida para o desenvolvimento da interface gráfica existiram mais condicionantes e cuidados a ter, quando comparado ao IDE para AVR.

Primeiramente pesquisou-se acerca de *frameworks* capazes do desenvolvimento de interfaces gráficas recorrendo à linguagem de programação Python, devido à familiaridade com a linguagem e pela sua simplicidade e versatilidade. Alguns dos principais fatores que se teve em conta para determinar a *framework* a utilizar foram [37]:

- Compatibilidade - Ao selecionar uma *framework* é importante certificar que é capaz de operar em modo *cross-platform*, isto é, a aplicação final é capaz de funcionar em qualquer sistema operativo como windows, linux, macOS, entre outros. É de igual importância certificar que a *framework* a ser utilizada é compatível com a versão de python que se utilizará no decorrer do projeto, de forma a não haver incompatibilidade de versões entre a framework e versão do python, e até mesmo, compatibilidade com bibliotecas que serão importadas para a aplicação.

- Documentação - Uma *framework* ao ter uma documentação adequada é essencial para qualquer tipo de desenvolvimento de *software* e *frameworks* GUI não são exceção. Ao possuir uma documentação de qualidade é possível acelerar consideravelmente o desenvolvimento da interface, tanto por associar-se uma aprendizagem mais clara, como também evitar perdas de tempo em problemas recorrentes, que já tenham sido enfrentados anteriormente por programadores, que já diagnosticaram o problema.

- Suporte da comunidade de desenvolvedores - É certo que uma *framework* que esteja ativa no presente é uma grande mais valia devido à possibilidade de acesso a soluções disponibilizadas em fóruns por outros programadores onde existem recursos partilhados de sugestões a efetuar *troubleshoot* em problemas comuns, especialmente para programadores com menos experiência.

- Licenças - Um fator importante a ter em conta, principalmente em casos que se pretendo comercializar a interface desenvolvida. Embora não sendo o caso deste projeto, não deixa de ter sido um fator a ter em conta devido a possíveis perspetivas para o futuro da aplicação desta interface. Assim sendo, algumas *frameworks* são de código aberto (*open-source*) e de uso gratuito, dentro de certas condições, enquanto outras *frameworks* podem exigir uma licença comercial. É importante compreender os termos de licença e certificar que se enquandram com os requesitos do projeto, de forma a evitar possiveís complicações financeiras e legais.

- Performance - É importante avaliar as características de performance que se estão a prever para cada projeto, de forma a decidir qual a *framework* ajustada para cada aplicação. Existem *frameworks* projetadas para lidar com alto volume de dados, e outras, para aplicações mais leves, como é o caso da interface gráfica deste projeto.

As *frameworks* que foram alvos de avaliação foram nomeadamente: Tkinter, PyQt5 e PyGUI. De forma a sumarizar e destacar os pormenores mais relevantes de cada framework tem-se a Tabela 3.4 [37, 38]:

Tabela 3.4: Comparação entre Tkinter, PyQt5 e Kivy

Framework	Prós	Contras
Tkinter	• Incluído na biblioteca Python, assegurando imediato acesso; • Compatibilidade em *cross-platform*; • Projetado para aplicações pequenas/médias; • Documentação bastante presente e suporte da comunidade razoável;	• Limitação no *design* e customização em termos estéticos; • Funcionalidades limitadas para GUIs complexas; • Esforço considerável para alcançar uma aparência moderna; • Não possui a melhor performance para aplicações mais complexas;
PyQt	• Elevado número de *Widgets* modernos e altamente personalizáveis; • Compatibilidade em *cross-platform*; • Grande conjunto de funcionalidades e suporte para desenvolvimento complexo, incluindo animações e *plugins* com aplicações externas; • Excelente documentação e comunidade ativa;	• Requer licença comercial GPL para aplicações projetadas para serem comercializadas; • Necessita de instalação adicional; • Curva de aprendizagem acentuada devido à complexidade;
Kivy	• Orientado para interfaces móveis, que recorrem a *multi-touch*; • Compatibilidade em *cross-platform*; • Utilização gratuita sem necessidade de licença para aplicações comerciais; • *Widgets* altamente customizáveis;	• Adequado para aplicações móveis, exigindo esforço extra para ajustar em aplicações de computador; • Curva de aprendizagem acentuada; • Suporte da comunidade em crescimento, mas menos extensivo do que PyQT;

Tendo em conta os fatores principais na seleção de uma *framework* complementando com os prós e contras de cada uma, a *framework* escolhida para o desenvolvimento da interface gráfica deste projeto foi o Tkinter. Essencialmente devido à sua simplicidade e integração com a própria biblioteca de Python, tendo em conta

a aplicação final, o *design* da interface não é um aspeto importante e finalmente devido a não ser necessário qualquer tipo de licença e ser uma *framework* orientada para interfaces não muito extensas.

3.3.3 *Universal Asynchronous Receiver Transmitter* (UART)

UART é uma interface de comunicação série entre periféricos amplamente utilizada em sistemas embebidos, principalmente entre específicos *chips* e a um utilizador via computador. Nomeadamente para propósitos de configuração ou para realizar *queries*, de forma a verificar o estado do *chip*. A maneira mais acessível de executar tal comunicação com um computador é a partir de um conversor USB-to-UART que facilmente se insere numa PORT do computador. UART é amplamente utilizada devido à sua versatilidade e simplicidade. Apenas requer dois fios para funcionar (não confundir com USART que usa quatro fios): Tx (transmissão) e Rx (receção). Permite comunicação assíncrona, ou seja, não existe um sinal de relógio para coordenar a comunicação, em vez disso, o transmissor e recetor sincronizam-se a partir da utilização da mesma *baud rate*, expressa em *bits* por segundo, sendo habitual o uso de *baud rate* igual a 9600 ou 115200 bps [39].

Organiza os dados a transmitir em pacotes, cada pacote tem um *start bit*, cinco a nove *bits* de dados, um *bit* opcional de paridade e um ou dois *stop bits*.

O modo de operação da UART assenta em [39]:

- Início de transmissão - A UART encontra-se num nível lógico *high* quando a linha de transmissão está em *idle*. O *start bit* é colocado antes dos *bits* de dados, de forma que, o recetor quando verifica a existência deste *bit* prepara-se para começar a receber os dados. O recetor consegue verificar a existência deste *bit* através de detetar que a linha de transmissão, que por norma está em *idle*, a certo momento foi do nível lógico *high* para *low*, significando que o transmissor pretende transmitir dados.

- *Bits* de dados - Os *bits* de dados compõe a informação que se pretende transmitir. O tamanho dos dados pode ser entre os cinco a nove *bits*. No entanto, por norma o número de *bits* costuma ser oito. Isto porque, para ser nove *bits* de dados, significa que não se está a recorrer ao uso de um bit de paridade;

- *Bit* de paridade - o *bit* de paridade consegue ajudar o recetor a verificar se a trama de dados sofreu alguma alteração durante a sua transmissão;

- End Bits - Define o fim da transmissão de um pacote de dados. Quando o *stop bit* é reconhecido, a linha de transmissão retorna a nível lógico *idle*, após ter voltado para o nível lógico *high*;

É importante refletir que há uma diferença entre USART e UART. Tal como o nome indica, a principal diferença baseia-se no facto da USART permitir a transmissão síncrona e assíncrona, consoante a necessidade do sistema. No entanto, o que acontece em bastantes casos, é utilizar a comunicação UART numa interface USART. Daí ser comum, por vezes visualizar conexões USART entre periféricos, mas na verdade a comunicação UART está a ser implementada a partir da interface USART. De salientar que é possível realizar a comunicação USART, no entanto, há o acréscimo de mais dois fios, resultando num total de quatro: Tx, Rx, XCK (clock) e XDIR(direção). Sendo assim, a UART permite comunicação *full-duplex*, enquanto a USART permite a comunicação *half-duplex*, no entanto, consegue possuir uma taxa de transmissão superior, quando comparada com a UART.

Em termos qualitativos, a UART conta com vantagens como:

- Simplicidade com a necessidade de apenas dois fios para estabelecer comunicação;
- Flexibilidade devido a não necessitar sinal de relógio;
- Uso de *bits* de paridade para deteção de erros de transmissão;
- Velocidade ajustável a partir da definição do valor de *baud rate* sendo facilmente ajustada para atender a requesitos específicos de um dado sistema;

Em aspetos menos positivos, tem-se de seguida desvantagens encontradas com a utilização da UART:

- Limitação de distância, uma vez que, é mais adequada para comunicações de curta distância devido às suas limitações de ruído e atenuação do sinal;
- Como é uma interface de comunicação assíncrona, por vezes pode levar a problemas de sincronização, caso o transmissor e recetor não estejam ambos a operar à mesma *baud rate*;
- Não possui mecanismos de deteção de erro. No entanto, facilmente a partir de manipulação de *software*, o utilizador consegue implementar mecanismos adicionais, como paridade ou *bits* de *stop*;
- Limite de nove bits para alocar dados na trama;
- Dificuldades em cenários que existem múltiplos dispositivos a quererem comunicar entre si;

Em relação às aplicações, a UART está presente em comunicações ponto a ponto em diversos campos. Essencialmente na utilização para comunicar entre microcontroladores e um computador pessoal de forma a configurar/analisar dados que possam estar a ser recolhidos por este, vindo de sensores de temperatura por exemplo.

É comum ver *standards* da UART a serem utilizados como RS-232 e RS-485. Estes *standards* permitem comunicação UART em cenários de longas distâncias e oferecem a possibilidade de criar redes com múltiplos escravos, aumentando a flexibilidade de aplicações UART.

3.4 Solução Projetada

Na Figura 3.7 é apresentado um esquema de como os elementos do sistema se integram, como os componentes de *hardware*, a interface gráfica e os cabos de teste. A arquitetura do sistema que se definiu atende às necessidades de realizar os testes de continuidade automatizados, como também a nível das leituras de dados provenientes de sensores embebidos com protocolo Modbus RTU/TCP e SNMP. A intenção desta figura é ilustrar a um nível geral os distintos blocos se integram no sistema final.

Na Figura 3.7, tal como se pode verificar, existem três blocos distintos, mas que se integram no sistema final:

- Interface Gráfica (Cor Amarela) - Esta interface gráfica, desenvolvida a partir do tkinter, é responsável por duas componentes: Uma das componentes é apresentar, de forma gráfica, o resultado do teste de continuidade realizado nos fios elétricos da máquina. De forma a ser possível testar a cablagem de certa zona da máquina, o operador tem que selecionar qual a zona que se pretende testar a partir do clique num botão presente na GUI. Após o clique, é enviado para o microcontrolador, via UART, a identificação da zona da máquina que se pretende testar naquele instante. Após o teste de continuidade ser efetuado, a interface gráfica é responsável por analisar e descodificar a mensagem relativa ao teste de continuidade efetuado. Após o tratamento dos dados, esta apresenta o resultado numa tabela, permitindo a verificação da cablagem por parte do operador. A outra componente é apresentar os dados recebidos provenientes dos diferentes sensores implementados na máquina.

- Sistema Teste de Continuidade(Cor Azul) - O sistema responsável por efetuar o teste de continuidade, composto essencialmente pelo atmega328p e quatro *multiplexers* (dois *multiplexers* a operarem como transmissor de sinal LOW e os restantes dois *multiplexers* a operarem como recetor de sinal LOW, é responsável por efetuar teste de continuidade da máquina. Numa fase inicial, estabelece conexão UART com a interface gráfica e recebe informação sobre os fios da zona da máquina que se pretendem testar. Após ser efetuado o teste de continuidade através da transmissão e receção de sinais digitais, as conexões onde se verificou continuidade são armazenadas num *buffer* e posteriormente transmite o resultado

do teste de continuidade de forma a que a GUI possa apresentar os resultados ao operador.

- Cabos de teste (Cor Verde) - Entende-se por este bloco, os cabos de teste desenvolvidos que permitem definir uma interface entre os conectores dos terminais elétricos da máquina e os *multiplexers* presentes no sistema responsável pelo teste de continuidade.

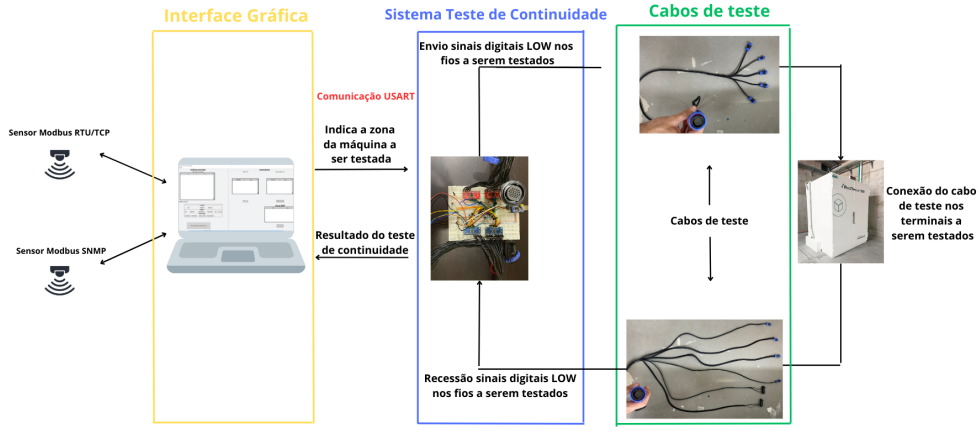


Figura 3.7: Arquitetura do Sistema

A importância deste esquema é apresentar uma ideia geral ao leitor de como o sistema se integra. Estes três blocos que foram mencionados, serão apresentados no capítulo seguinte a um nível mais detalhado. No entanto, ter uma ideia abstrata que existem estes três blocos e que se complementam é essencial para ter uma melhor compreensão de como se abordou o desenvolvimento do sistema final.

Capítulo 4

Protótipo Final

No capítulo que se segue, serão apresentadas as diversas etapas do projeto que permitiram obter o sistema final. De forma a proporcionar a melhor compreensão do sistema desenvolvido, assume-se que este é composto por três unidades individuais que se complementam:

- Sistema responsável pelo teste de continuidade.
- Processamento e comunicação dos resultados obtidos pelo teste de continuidade.
- Interface gráfica, responsável por apresentar os resultados de continuidade, e também responsável por efetuar leituras de dados provenientes de dispositivos que comunicam via Modbus e/ou SNMP.

Convém reforçar que a distinção entre estas três vertentes é meramente abstrata, dado que, como ferramenta, estas três vertentes integram-se e operam em conjunto.

Relativamente à estrutura deste capítulo, inicialmente é abordado o esquema elétrico do sistema responsável pelo teste de continuidade. De seguida será analisado o desenvolvimento entre o microcontrolador e os *multiplexers*, a respetiva funcionalidade do sistema e a forma como se alcançou a obtenção, processamento e envio de dados. Segue-se com uma explicação dos cabos de teste soldados que são uma peça fundamental, visto que permitem prolongar as ligações elétricas da máquina até ao sistema responsável pelo teste de continuidade, mais especificamente, até aos canais dos *multiplexers*. Quando o leitor está a par do funcionamento deste sistema,

é apresentada uma secção que demonstra a validação do sistema responsável pelo teste de continuidade, complementando uma análise de um exemplo prático.

Por fim, apresenta-se a interface gráfica, seguida das etapas de estabelecimento de conexão como microcontrolador e método implementado para permitir ao utilizador de selecionar a zona da máquina que pretende testar. Posteriormente realiza-se uma análise de como se efetuou o tratamento dos dados recebidos, provenientes do microcontrolador e respetiva apresentação dos resultados em forma de tabela. Analisar-se-á a outra componente desta interface gráfica, apresentando as suas funcionalidades como leitura de dados provenientes de sensores com protocolos de comunicação, como Modbus e SNMP. Por fim, apresentam-se os testes realizados, com o intuito de demonstrar como o sistema de teste de continuidade e a interface gráfica se integram.

4.1 Esquema elétrico do sistema responsável pelo teste de continuidade

É de seguida apresentado o esquema elétrico desenvolvido para realizar testes de continuidade na cablagem, durante o processo de produção da máquina da empresa, utilizando o microcontrolador atmega328p e quatro *multiplexers*. O microcontrolador atua como núcleo do sistema, coordenando os quatro *multiplexers* envolvidos. Na Figura 4.1 , elaborou-se o *layout* do esquema, o mais semelhante possível ao *layout* do sistema físico. É possível verificar que o microcontrolador está no centro do sistema, e está cercado por quatro *multiplexers*, chamados: MUX1, MUX2, MUX3, MUX4. Os *multiplexers* MUX1 e MUX3 são ambos considerados *multiplexers* transmissores e os restantes, MUX2 e MUX4 são ambos considerados *multiplexers* recetores. Ou seja, os *multiplexers* transmissores são os encarregados por enviar, em certos canais, a determinados instantes, um sinal digital LOW. Por sua vez, os *multiplexers* recetores são encarregados por receber e detetar em qual dos seus canais recebeu um sinal digital LOW.

Na Figura 4.1 é possível verificar que os pinos PB0, PB1, PB2, PB3 do microcontrolador, são partilhados por dois *multiplexers*: MUX1 e MUX3 e os pinos PC0, PC1, PC2, PC3 são partilhados pelos *multiplexers* MUX2 e MUX4.Ou seja, evidencia-se aqui que os *multiplexers* transmissores, podem partilhar entre si, os mesmos pinos encarregados como seletores de canal, pinos: S0, S1, S2 e S3 de cada *multiplexer*, e por sua vez, os par de *multiplexers* recetores podem também partilhar entre si o pinos encarregados como seletores de canal. A razão pela qual, quer o par de *multiplexers* transmissores, quer o par de *multiplexers* recetores, podem partilhar entre si os mesmos pinos destinados a seletores de canal, deve-se ao facto de economizar o maior número de pinos do microcontrolador, e porque, no sistema final, a cada instante estão apenas dois *multiplexers* em uso, um *multiplexer* transmissor e

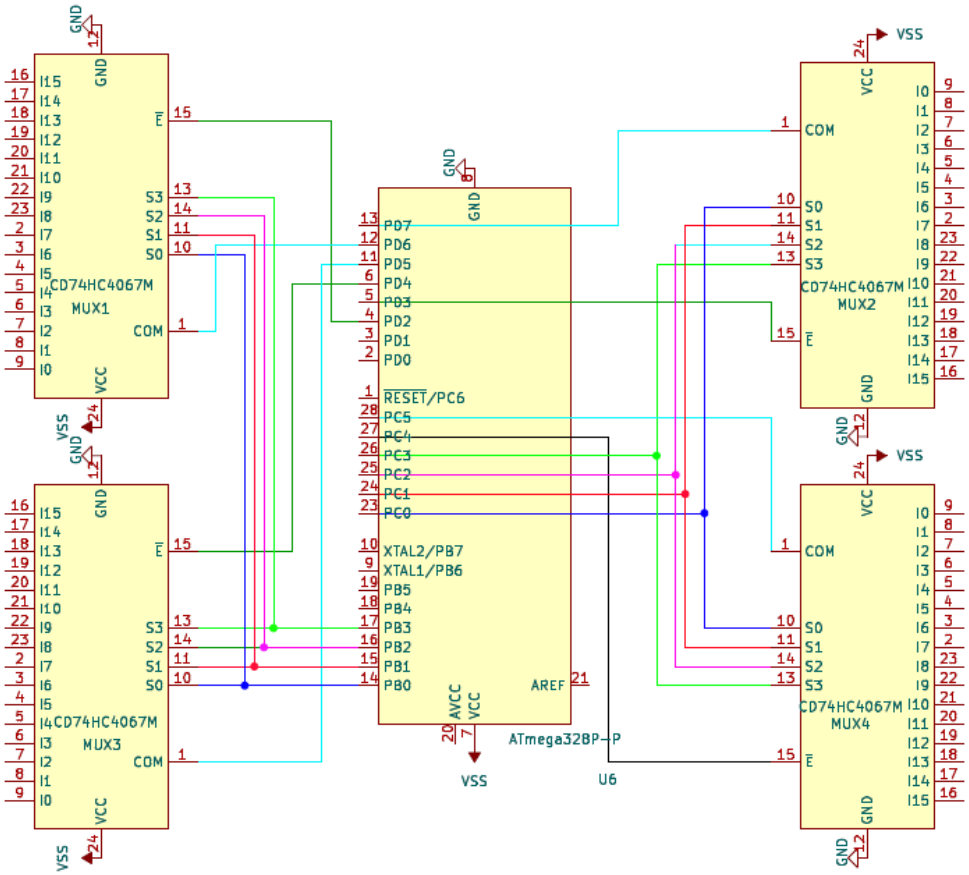


Figura 4.1: Esquema circuito elétrico

um *multiplexer* recetor, e como tal, é possível haver a partilha de pinos encarregados como seletor de canal, pois dos dois *multiplexers* do mesmo tipo (dois *multiplexers* transmissores, por exemplo), apenas um deles estará a ser usada em certos instantes. Como tal, é possível reduzir o número de pinos utilizados pelo microcontrolador, para preencher os pinos de seleção de canal, S0, S1, S2 e S3, de cada *multiplexer*. Caso não se realizasse esta economia de pinos, para quatro *multiplexers* iriam ser logo necessários dezasseis pinos do atmega328p, apenas para preencher os campos de seleção de canal dos *multiplexers*. No entanto, foi destacado o ponto de partilha destes pinos ser possível, mas com a condição de que é necessário haver uma seleção de qual dos *multiplexers* , do mesmo tipo, está ativo. Essa mesma seleção é obtida na configuração do microcontrolador e é importante realçar que cada *multiplexer*, independentemente da sua responsabilidade, têm que possuir um pino individual do microcontrolador, destinado ao pino "E" do *multiplexer*. Este "E" advém da palavra "*Enable*", e é a partir deste pino que é possível, no microntrolador, controlar qual

o *multiplexer* está ativo, para cada instante. Este pino *Enable*, por sua vez, já não pode ser partilhado entre os outros *multiplexers*, independentemente de servirem o mesmo propósito (transmissor ou recetor). É devido a este pino que é possível partilhar de pinos de seleção de canal, entre *multiplexers* do mesmo tipo. Por fim, resta o pino "COM" em que este, também poderia ser partilhado entre *multiplexers* do mesmo tipo, mas no que toca à economia de pinos, já não há a necessidade de partilhar um mesmo pino para o mesmo grupo de *multiplexers*, até porque, durante a apresentação da implementação é útil acompanhar o desenvolvimento do código a visualizar diferentes pinos atribuídos ao "COM" de cada multiplexer. Este "COM" é o responsável por aplicar o sinal emitido pelo microcontrolador, para o respetivo canal do seu *multiplexer*. Por fim, o conector FT232 é responsável pela comunicação UART, e em simultâneo alimenta o circuito do sistema.

4.2 Sistema responsável por efetuar teste de continuidade

Relativamente à primeira etapa do projeto, optou-se por desenvolver o sistema capaz de realizar testes de continuidade. o sistema responsável por efetuar teste de continuidade é apresentado na Figura 4.2 onde os *multiplexers* transmissores são os *multiplexers* de cor vermelha e os *multiplexers* recetores são os *multiplexers* de cor azul. Como nota, as cores dos *multiplexers* quando adquiridos, foi por mera coincidência, pois os quatro *multiplexers* são o mesmo modelo, apenas de fabricantes diferentes. Sendo que se possuía *multiplexers* de cores diferentes, aproveitou-se e efetuou-se uma distribuição lógica das cores: *multiplexers* vermelhos são os transmissores e os *multiplexers* azuis são os recetores. Também é possível verificar que existem dois conectores SP2912/S24, um conector SP2912/S24, tem os seus fios distribuídos pelos *multiplexers* transmissores, e o outro conector SP2912/S24 possui os seus fios distribuídos pelos *multiplexers* recetores. É também possível averiguar que os dezasseis canais (C0-C15) dos *multiplexers* MUX1 e MUX2 são preenchidos por dezasseis fios, com origem dos primeiros dezasseis fios do conector SP2912/S24. Tal como referido anteriormente, a nomenclatura "/S24" significa um total de vinte e quatro pinos, ou seja, os mutiplexers restantes, MUX3 e MUX4, são preenchidos apenas em sete dos seus canais (CH0-CH6),pelos restantes sete fios do mesmo conector S2912/S24. A soma de dezasseis fios, com estes sete fios, é de vinte e três fios utilizados, dos possíveis vinte e quatro existentes no conector. Isto porque, em nenhum dos testes da máquina, são necessários testar vinte e quatro fios de uma só vez.

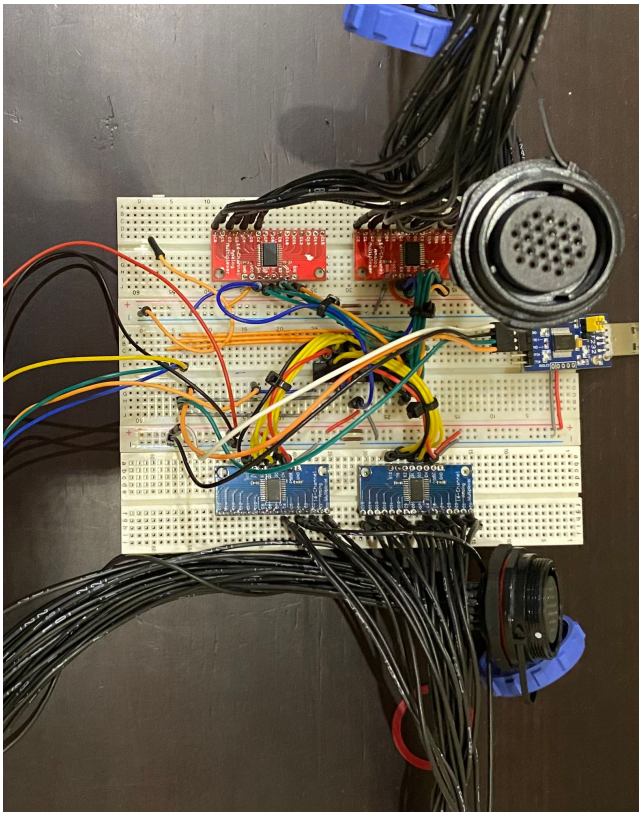


Figura 4.2: Sistema responsável por efetuar teste de continuidade

Durante este sub-capítulo são discutidas as funcionalidades de cada componente do sistema e a forma como se integram no sistema de continuidade final. Na fase de desenvolvimento do microcontrolador optou-se pela *framework* Microchip Studio, com auxílio ao *software* AVRdudes para configuração do atmega328p. O *software* RealTerm, como ferramenta auxiliar, permitiu monitorizar o fluxo de dados de comunicação série. Esta ferramenta auxiliar ajudou a realizar *debug* à medida que o código era desenvolvido e também se aproveitou para os dados já serem enviados via UART, com a formatação idêntica aos dados que são enviados no sistema final. Permitiu assim que no desenvolvimento da interface gráfica, já se desenvolveu o algoritmo responsável pela receção destes dados de forma orientada a antecipar a formatação dos dados que já se ia verificando no *RealTerm*.

4.2.1 Inicializações

A partir da Tabela 4.1 verificámos que em relação a MUX1 e MUX3, não existe distinção nos pinos do microcontrolador que se conectam aos pinos S0, S1, S2 e S3 dos *multiplexers* transmissores, tal como já foi mencionado anteriormente, na aprsentação das ligações elétricas da Figura 4.1. Para além disso, o microcontrolador inicializa SIG1 e SIG3 como saídas e com estado lógico HIGH.

Pinos *Multiplexer Transmissor*	**Pinos Microcontrolador**	**Inicialização**
S0_TX	PB0	Saída
S1_TX	PB1	Saída
S2_TX	PB2	Saída
S3_TX	PB3	Saída
EN1	PD2	Saída
EN3	PD4	Saída
SIG1	PD6	Saída
SIG3	PD5	Saída

Tabela 4.1: Inicializações pinos *multiplexers* transmissores

Relativamente à Tabela 4.2, nas suas inicializações, os pinos de seleção de canal dos *multiplexers* recetores usufruem de pinos comuns do microcontrolador. No entanto, os pinos SIG2 e SIG4 inicializam-se como entradas e as resistências *pull-up* internas são ativadas.

Pinos *Multiplexer*Recetor	**Pinos microcontrolador**	**Inicialização**
S0_RX	PC0	Saída
S1_RX	PC1	Saída
S2_RX	PC2	Saída
S3_RX	PC3	Saída
EN2	PD3	Saída
EN4	PC4	Saída
SIG2	PD7	Entrada
SIG4	PC5	Entrada

Tabela 4.2: Inicializações pinos *multiplexers* recetores

A necessidade da configuração com as resistências pull-up internas têm origem numa dificuldade presenciada durante o *debug* do sistema. Inicialmente, o projeto começou com as resistências pull-up desativadas, e realizava-se a leitura de sinais HIGH. À medida que o número de fios aumentava, de forma a ser testado a continuidade da máquina, observou-se que por vezes havia incoerências nas conexões enviadas. Ou seja, a *string* formatada relativa às conexões validadas, que é enviada via UART, por vezes, indicava que o mesmo canal do *multiplexer* recebeu um sinal digital HIGH com origem de 3 canais, curiosamente, distintos e ordenados, do *multiplexer* transmissor, quando cada canal recetor só pode ter um único canal transmissor associado a ele. Caso contrário, significaria que na máquina existem fios em *shunt*. Obviamente durante o processo de *debug*, percebeu-se que o *shunt* não seria a causa, mas sim algum erro da leitura. Tal como foi dito anteriormente, curiosamente, o erro era constante entre leituras seguidas e ordenadas. Ou seja, abstratamente, imagine-se que uma conexão correta (comparando a um mapeamento existente que será explicado mais à frente) seria a conexão entre o canal 0 (CH0)

do MUX1 e o canal 4 (CH4) do MUX2. O que deveria ser alcançado seria a associação de que o sinal digital HIGH com destino a CH4, adveio de CH0. No entanto, antes de implementar as resistências *pull-up* internas, o erro que chegava à porta série era a indicar que o canal CH4 recetor, recebeu sinais de CH0, CH1, CH2, CH3, CH4. Ou seja, tem uma conexão correta CH0-CH4 e três conexões incorretas. Para exatamente a mesma conexão, foi-se testando aumentar o intervalo de pausa na emissão de sinais digitais, de forma a conseguir estabilizar cada vez mais o sinal, e apercebeu-se que algumas ligações (erradas) já não iam aparecendo no terminal. Mesmo assim, num teste de bastantes fios em simultâneo, ainda ia aparecendo erro com duas ou três conexões. Percebendo que o tempo de *delay* é mais que suficiente, ponderou-se que poderia ter a haver com sinais de tensão residuais, que quando alocados nos canais do *multiplexer*, podiam induzir em erro a leitura de uma conexão. Determinou-se este erro nas leituras através de uma aproximação com um multímetro (idealmente seria um osciloscópio) que efetivamente havia instantes em que alguns canais do *multiplexer* tinham efetivamente tensões residuais de 1.9V, que, embora não seja uma tensão de se considerar HIGH, é uma tensão bastante próxima da que seria para alcançar o estado "HIGH". De forma a resolver esta incoerência nas conexões guardadas, conectaram-se resistências a operar como *pull-down* nos canais SIG dos *multiplexers*. Ao observar-se que estas conexões incorretas foram mitigadas, percebeu-se que esta seria a solução a seguir. Visto que o microcontrolador atmega328p possibilita as resistências pull-up internas, optou-se por ativá-las, não necessitando assim das 4 resistências *pull-down* externas. Acompanhando a implementação tardia das resistências de *pull-up* internas, houve a necessidade de ajustar o estado lógico do sinal que é enviado. Daqui em diante, o sinal digital que é transmitido e recebido, é o sinal digital LOW.

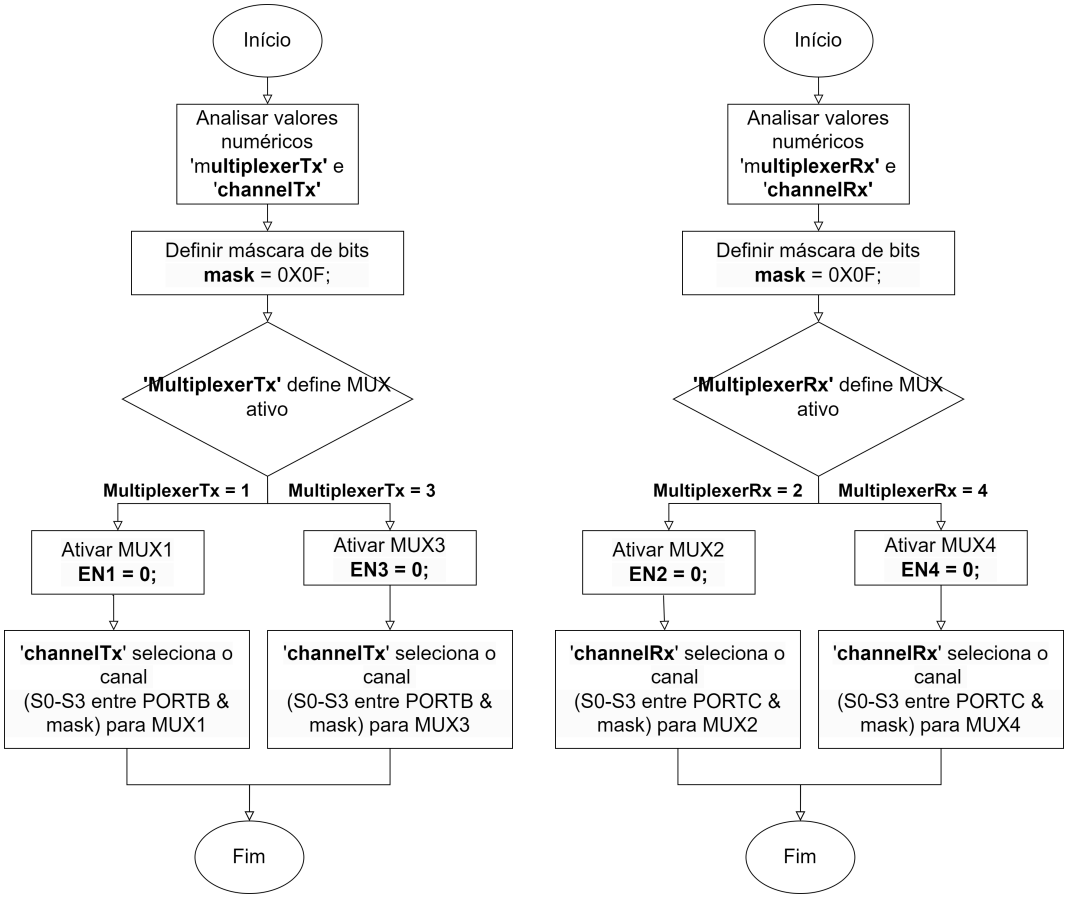
4.2.2 Funções controlo *multiplexers*

De forma a integrar o microcontrolador e os *multiplexers* no sistema de teste de continuidade, foram desenvolvidas as funções `void selectChannelTx(uint8_t multiplexerTx, uint8_t channelTx)` e `void selectChannelRx(uint8_t multiplexerRx, uint8_t channelRx)`. Estas funções, permitem ao microcontrolador selecionar dinamicamente o canal que se deseja manter ativo de um *multiplexer* específico, conforme a necessidade do teste de continuidade. A distinção entre *multiplexers* transmissores e recetores é claramente evidenciada tanto pelo próprio nome da função como os parâmetros que recebe. Tais parâmetros permitem definir o *multiplexer* em uso e o respetivo canal em uso. Como referido anteriormente, tanto os *multiplexers* transmissores, como os *multiplexers recetores*, podem partilhar entre si os pinos destinados a S0-S3 pois durante o teste de continuidade, exclusivamente o conjunto de um *multiplexer* transmissor e um *multiplexer* recetor está ativo.

Complementando, existe a necessidade destas duas funções que, embora distintas, executam um propósito bastante idêntico.

Observando o cabeçalho da função `'selectChannelTx'`, por exemplo, percebe-se que esta função é a responsável por selecionar o canal dos multiplexers transmissores. A função recebe dois argumentos: o primeiro, 'multiplexerTx', identifica o *multiplexer* a estar ativo, MUX1 ou MUX3 (recorde-se que esta nomenclatura remete aos *multiplexers* transmissores). A partir deste argumento, o pino de habilitação (EN) correspondente é colocado num nível lógico LOW, ativando o *multiplexer* e permitindo a passagem do sinal digital pelo pino de saída (SIG) do *multiplexer* ativo. Já o segundo argumento, 'channelTx', especifica o canal do *multiplexer* que será selecionado. A partir deste argumento, a seleção de canal é realizada configurando de forma binária os pinos S0 a S3 do *multiplexer* e, em cojunto com uma máscara de bits, seleciona-se o canal desejado. O canal ao estar selecionado, está preparado para enviar o sinal digital LOW. Analogamente, a função 'selectChannelRx' opera de forma similar, porém, é aplicada aos *multiplexers* recetores e está preparada para ler um sinal digital LOW no canal selecionado. Na Figura 4.3a é apresentado o fluxograma que descreve o processo de seleção de *multiplexer* e canal onde se pretende efetuar a transmissão do sinal digital LOW. Já na Figura 4.3b é exposto um fluxograma semelhante, mas este descreve o processo de seleção do *multiplexer* e canal onde se pretende ler o sinal digital LOW.

Estas funções proporcionam um controlo dinâmico dos *multiplexers*, permitindo transições rápidas e precisas nos canais ativos. Tal é importante, visto que, no contexto de teste de continuidade, a alta taxa de transição entre distintos canais selecionados é necessária para verificar múltiplas conexões em curtos intervalos de tempo. Assim, o sistema é capaz de testar sequencialmente vários fios da máquina, assegurando que cada canal esteja ativo no momento certo para uma avaliação adequada da continuidade elétrica.



(a) Seleção canal *multiplexer* transmissor (b) Seleção canal *multiplexer* recetor

Figura 4.3: Etapas da seleção de canal de cada *multiplexer*

4.2.3 Obtenção, Tratamento e Envio de Dados

Relativamente à obtenção e tratamento de dados foi projetada a função:

`void mapConnections (uint8_t multiplexerTx, uint8_t txInitial, uint8_t txLast, uint8_t multiplexerRx, uint8_t rxInitial, uint8_t rxLast)` e a interrupção `ISR(USART_RX_VECT)`.

Consoante a porção da máquina que se deseja testar, na interface gráfica, o utilizador ao clicar no botão correspondente ao tal teste, é automaticamente enviado um valor que atua como flag. O microcontrolador, ao ativar a interrupção por ter recebido dados via UART, irá realizar o teste correspondente ao botão clicado. Para tal, a função `mapConnections` está configurada para ativar os *multiplexers* e respetivos canais nos intervalos de tempo corretos. Quando os *multiplexers* estão a postos, o microcontrolador emite sinais digitais LOW através do *multiplexers* e canais selecionados e, ao realizar a leitura de um sinal nos canais dos *multiplexers* recetores, deteta a origem do sinal e guarda uma conexão com a respetiva origem e destino do sinal. Posteriormente, reunindo todas as conexões que foram guardadas, envia as conexões armazenadas via comunicação série num formato específico.

A função `mapConnections` recebe como argumentos informação relativa aos *multiplexers* que estarão ativos em determinado instante: um *multiplexer* transmissor ativo, `multiplexerTx`, e um *multiplexer* recetor ativo, `multiplexerRx`, bem como o canal inicial e final de cada *multiplexer* (`txInitial`, `txLast`, `rxInitial`, `rxLast`), permitindo assim que se reúna o total de canais necessários para efetuar o teste selecionado a partir da interface gráfica. Com as informações que estes argumentos fornecem, o microcontrolador é capaz de realizar um varrimento na seleção de canais totais através de todas as combinações possíveis de canais, garantindo uma validação que abrange todas as ligações do teste de continuidade em questão.

Com recurso ao fluxograma presente na Figura 4.4 analisa-se o funcionamento da função `mapConnections`. A partir dos parâmetros recebidos, a função determina inicialmente o *multiplexer* transmissor ('multiplexerTx') e o *multiplexer* recetor ('multiplexerRx') a ativar. Posteriormente, com recurso às variáveis `txInitial` e `txLast` definem o intervalo de canais que vão transmitir o sinal digital LOW. Analogamente, as variáveis `rxInitial` e `rxLast` definem o intervalo de canais que vão receber o sinal digital LOW. Posteriormente, a partir de um *nested loop*, o microcontrolador, quando deteta a receção de um sinal digital LOW num dos canais (variável 'j') do *multiplexer* recetor, tem como prioridade detetar a sua origem (variável 'i'). Ao detetar a continuidade do fio a ser testado, o microcontrolador certifica-se de guardar a conexão num *buffer*. Esta conexão é guardada temporariamente no formato de *nibble* com codificação hexadecimal. Um "*nibble*" é uma unidade de dados de 4 bits, que corresponde à metade de um byte (8bits). No contexto desta função, o uso de *nibbles* em conjunto com a codificação em dados hexadecimais, permite que a informação de dois canais diferentes, transmissor ('i') e recetor ('j') seja compactada num único *byte*. Quando é detetada uma conexão, os valores dos canais de transmissão e receção são armazenados num *buffer* num único byte em formato hexadecimal. A conexão é armazenada com a seguinte estrutura: os bits mais significativos representam o canal de transmissão ('i«4') e os 4 bits menos significativos representam o canal de receção ('j'). Esta abordagem tem então a vantagem de permitir que ambos os canais transmissores e recetores, sejam agregados e transmitidos num único bloco de memória, aumentando a eficiência do processo de comunicação e já ter um formato ideal para posteriormente esta string ser descodificada por parte da interface gráfica. A necessidade de codificação em dados hexadecimais advém da seguinte razão: recordando que os *multiplexers* possuem 16 canais, quando a conexão fosse, por exemplo, entre o canal 11 (transmissor) e o canal 4 (recetor). O formato de string iria indiciar que a conexão respetiva é: "114", ou seja, o canal recetor 4 recebeu um sinal digital LOW, originado pelo canal 11 transmissor. Mas seria complicado realizar o tratamento desse dado por parte da interface gráfica, pois "114", embora seja canal 11 conectado com canal 4, poderia também, ser erradamente interpretado, que é o canal 1 do transmissor e o canal 14 do recetor, o que estaria errado. O

facto de possuir uma mensagem no formato hexadecimal, a mesma conexão: pino 11 transmissor com pino 4 recetor, em vez de estar formatado como "114", que levantaria suscetibilidade a erros de leitura deste formato, tirando proveito da codificação hexadecimal, já ficaria "B4" e assim não há suscetibilidade para qualquer dúvida.

Após todos o término dos *nested loops*, todas as conexões, no formato de *nibble*, que foram armazenadas durante o teste de continuidade, são transmitidas via UART para a interface gráfica. Esta etapa finaliza o processo de obtenção, tratamento e transmissão de dados relacionados com o teste de continuidade, permitindo que os resultados sejam posteriormente analisados e interpretados, conforme necessário, por parte da interface gráfica. Após a transmissão de dados, o *buffer* é limpo, de forma a estar preparado para uma nova operação de teste que chegue via interrupção `ISR(USART_RX_VECT)`, garantindo que o sistema possa operar de maneira contínua e eficiente, sem reenviar conexões de testes anteriormente realizados.

Através da configuração dos *multiplexers* e da leitura sequencial dos sinais de conexão, a função assegura uma verificação completa e precisa de todas as ligações possíveis, desde que estejam dentro do intervalo definido pelos argumentos da função. A utilização do conceito de *nibbles* e formato de dados hexadecimal, para formatar a *string* onde indica a conexão entre o canal transmissor e o canal recetor num único *byte* reforça a eficiência e clareza na transmissão dos dados.

É importante salientar que não é da responsabilidade do microcontrolador certificar se a conexão elétrica encontrada está correta ou se está com os terminais dos fios da máquina trocados. Essa é a responsabilidade da interface gráfica, pois é nesta que possui um ficheiro .txt (que no seu conteúdo tem a formatação e estrutura exatamente igual a um ficheiro .json) para certificar se a conexão encontrada pelo microcontrolador está correta ou não. Tal tema será abordado mais adiante na dissertação, mas achou-se por bem salientar esta responsabilidade, de forma a que o leitor não assuma que é da responsabilidade da função 'mapConnections' a validação da conexão encontrada. O propósito essencial desta função é enviar um sinal digital LOW e do lado dos *multiplexers* recetores efetuar a sua leitura, averiguando continuidade. Caso exista a leitura do sinal digital LOW, mas num canal que não seria suposto, será então da responsabilidade da interface gráfica detetar esta troca de pinos e apresentar o resultado ao operador. Caso o sinal digital nunca seja lido por nenhum dos canais, assinala uma conexão sem continuidade.

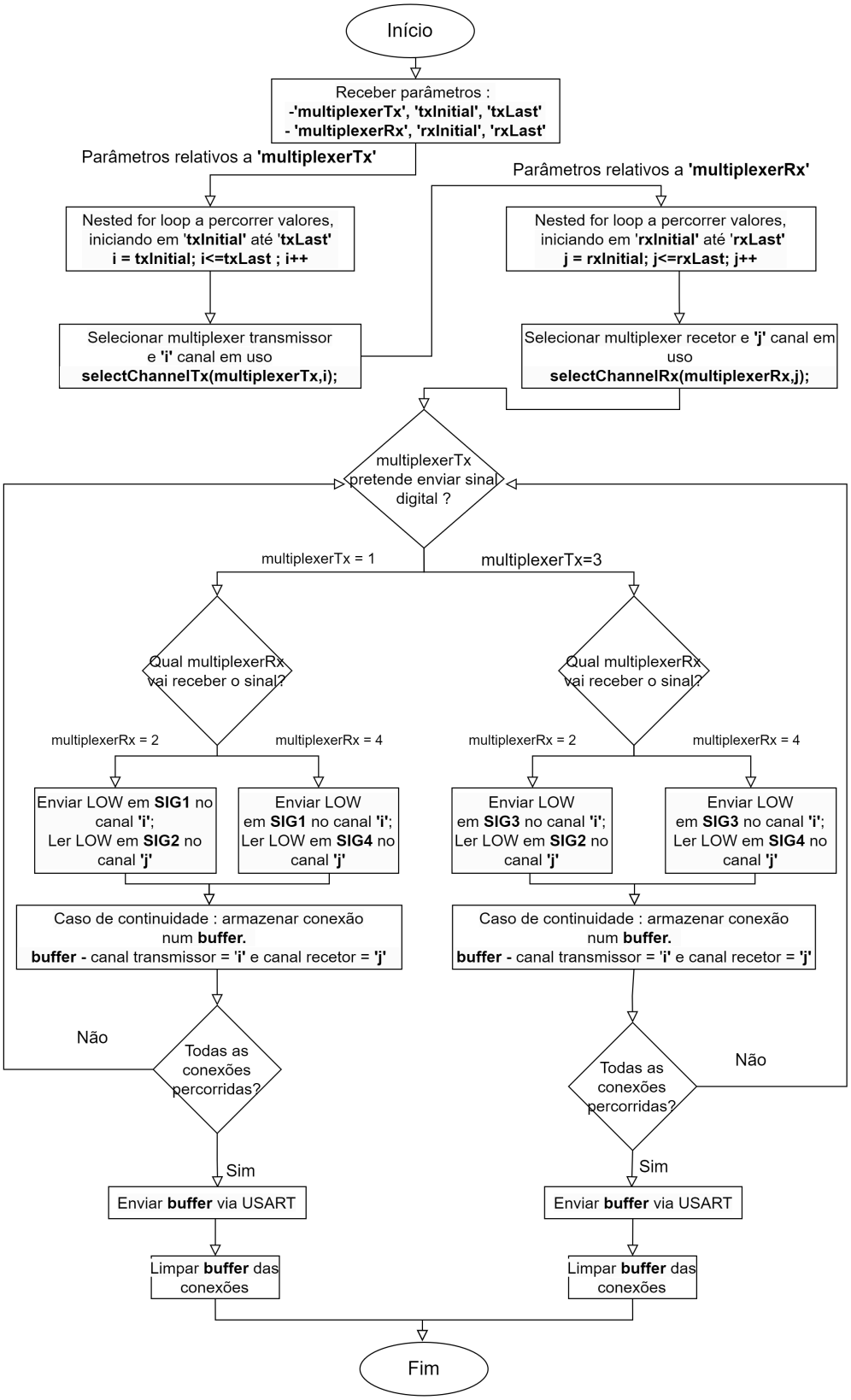


Figura 4.4: Fluxograma explicativo da função *mapConnections()*

4.2.4 Cabos de teste

Pretende-se testar a continuidade elétrica dos fios elétricos da máquina e os "cabos de teste"são os utensílios que permitem conectar os terminais elétricos da máquina e os canais dos *multiplexers*.

Estes cabos de teste, permitem assim criar uma interface que conecta os canais dos *multiplexers* aos conectores elétricos presentes no chassi da máquina. Ou seja, uma das extremidades do cabo de teste conecta-se aos conectores da máquina, de acordo com o teste que se espera realizar, e posteriormente, a partir da extremidade do cabo de teste, que contém o conector SP2912/P24 (Macho), conecta-se ao conhecido SP2912/S24 (Fêmea) que está ligado eletricamente aos canais dos *multiplexers*. Ou seja, sendo que os cabos de teste têm sempre numa das suas extremidades o conector SP2912/P24 (Macho), o que se altera, são os conectores presentes na outra extremidade, que por sua vez, depende da zona da máquina que se pretende testar. Sendo assim, de seguida apresentam-se os pontos que se teve em consideração para o desenvolvimento dos cabos de teste.

- De forma a poder conectar os cabos de teste aos *multiplexers*, executou-se uma solda individual do conector SP2912/S24 e conectou-se aos canais dos *multiplexers*. Tal como já foi referido anteriormente, um conector SP2912/S24 preenche um total de 23 canais do conjunto de *multiplexers* transmissores. Um outro, e distinto conector SP2912/S24 preenche um total de 23 canais do conjuntos de *multiplexers* recetores. Relembrando os conceitos mencionados anteriormente, a terminologia "S24"significa que é um conector do tipo fêmea e possui um total de 24 pinos (só são utilizados 23, por uma questão de ser o suficiente para testar cada zona da máquina).

- Se todos os *multiplexers* estão conectados, cada um deles, a um conector SP2912/S24 (fêmea), então numa das extremidades dos cabos de teste tem que existir sempre um conector SP2912/P24 ('P' associa-se ao conector macho). Já a outra extremidade destes cabos de teste, são soldados conectores para cada contexto, dependendo do conector que está presente no chassi da máquina, e que em simultâneo, remetem à origem e destino de certa ligação elétrica interna na máquina. No entanto, é certo que os conectores presentes nesta extremidade do cabo, são sempre do mesmo tipo do conector encontrado no chassi da máquina, mas com o encaixe sempre oposto.

  Dando um exemplo na "origem"de uma ligação, se no chassi da máquina estiver presente um conector, por exemplo SP2912/S17 (fêmea), então no cabo de teste encontra-se nesta extremidade um conector SP2910/P17(macho). de forma a ligar ao conector SP2912/S24 que está, por sua vez, conectado aos *multiplexers* transmissores. No "destino"das mesma ligação da máquina, se no chassi da máquina

estiver por exemplo um conector SP1710/S4 então na extremidade do cabo de teste irá ter um conector SP1710/P4(macho) que está ligado ao outro conector SP2912/S24 que está, por sua vez, conectado aos *multiplexers* recetores.

- Para cada zona da máquina que se pretende testar, consoante a sua quantidade de conexões, existem sempre, pelo menos, 2 cabos de teste (1 cabo de teste para "origem"das ligações e o restante cabo de teste para o "destino"das ligações elétricas da máquina.

  O cabo de teste transmissor (que está conectado entre os canais dos *multiplexers* transmissores e aos terminais da máquina onde está a origem das ligações elétricas da máquina) de teste possui: 1 extremidade sempre com conector SP2912/P24 (ligado aos *multiplexers* transmissores) e a outra extremidade possui a solda de diversos conectores que se conectam ao chassi nas origens das ligações elétricas.

  Analogamente, o cabo de teste recetor (que está conectado entre os canais dos *multiplexer* recetores e os terminais da máquina onde se encontram o destino das ligações elétricas)

- De salientar que, de acordo com os diversos testes, um mesmo conector do chassi da máquina, tanto pode ser a "origem"das ligações elétricas para um determinado teste, e ser o "destino"de ligações elétricas para um teste distinto. Daí haver a necessidade de o número das zonas diferentes da máquina serem (15 zonas) terem sempre, pelo menos, 15 cabos de teste.

A máquina da empresa possui, o que se pode considerar, um total de 15 zonas distintas a serem testadas. Para efeitos desta dissertação, apenas se executou a solda de 6 cabos de teste e 2 conectores SP2912/S24, permitindo assim apenas realizar teste de continuidade para 3 zonas da máquina. A razão pela qual apenas se soldou os cabos respetivos às zonas: "B1", "Bypass 2"e "UPS-U2" deve-se ao facto do tempo que se demora a soldar estes cabos de teste. De forma ao leitor ter uma noção, envolvendo a falta de experiência em soldar, estes 6 cabos de teste e 2 conectores SP2912/S24 demoraram aproximadamente 2 semanas. Embora não se tenha efetuado o desenvolvimento dos restantes cabos de teste, é importante salientar que o <mapeamento> dos cabos de teste em falta, para as restantes 12 zonas da máquina, e o mapeamento entre os cabos de teste e os *multiplexers* foram mapeados num ficheiro excel, onde foram fornecidas as instruções à empresa de como teriam que ser executadas as soldas dos cabos de teste em falta e de que forma teriam que estar conectados aos *multiplexers*, de forma a ir ao encontro do algoritmo presente no microcontrolador. Ou seja, tanto o mapeamento dos cabos de teste em falta, como o algoritmo presente no microcontrolador estão definidos, caso futuramente se desenvolvam os restantes cabos de teste, este sistema permite realizar o teste das 15

diferentes zonas da máquina. Os cabos de teste desenvolvidos permitem o teste de continuidade a: "B1", "Bypass2"e "UPS-U2".

O desenvolvimento dos cabos de teste permitem assim a criação de uma interface entre os *multiplexers* de transmissão e receção, sendo crucial para estabelecer um meio de envio e receção de sinais digitais robusta, entre os diferentes conectores da máquina, contribuindo para a possibilidade de realizar teste de continuidade. O facto de em cada zona da máquina, possuir sempre pelo menos 2 cabos de teste, permite fechar o circuito elétrico entre a máquina e os *multiplexers*, no qual os sinais digitais de teste podem ser enviados e recebidos, possibilitando assim a verificação da continuidade dos fios elétricos da máquina.

De seguida é apresentada a Tabela 4.3 idêntica à que foi fornecida pela empresa. Esta tabela, apresenta as ligações elétricas que se encontram na máquina, nas zonas "B1", "Bypass2"e "UPS-U2", onde descreve os tipos de conectores tanto na "Origem" contendo o conector da máquina que se ligará, por meio dos cabos de teste, aos *multiplexers* transmissores e o "Destino" onde se associa o conector que estará ligado, por meio de cabos de teste, aos *multiplexers* recetores.

Tabela 4.3: Ligações elétricas das 3 zonas da máquina

Zona Máquina	**Origem**	**Conector**	**Destino**	**Conector**	**Descritivo**
Zona B1	B1	SP2912/P3	BYP2	JDS25(A)-F (1-3)	(A): BYP2-B1-In
	B1	SP2912/P4	P5.2	SP2110/S24	AVAC
	B1	SP2912/S17(1-4)	P5.2	SP1710/S4	RS485
	B1	SP2912/S17 (5-6)	P3	SP1710/S2	P3-B1-PS
	B1	SP2912/S17 (7-8)	P3	SP1710/S2	B3-P1-EV
	B1	SP2912/S17 (9-10)	BYP2	JDS25(A)-F (9-10)	BYP2-B1-12V
	B1	SP2912/S3	BYP2	JDS25(D)-F(1-3)	BYP2-B1-OutA
	B1	SP2912/S3	BYP2	JDS25(D)-F(20-22)	BYP2-B1-OutB
Zona Bypass2	BY2	JDS25(20-22)(A) F	U2	JDS25(B) (1-3)	UPS-Input
	BY2	JDS25(A) (23-25) F	U2	JDS25(B)(20-21-4)M	UPS-Output
	BY2	JDS25(A) (5-6) F	U2	JDS25(B) (5-6) M	Sinal Bypass
	BY2	JDS25(A) (7-8)	P4	JDS25(B) (7-8) M	Inf. UPS
Zona UPS-U2	U2	JDS25(B) (22-25)M	BAT2	JDS25(C) (1-4)M	Lig. Baterias

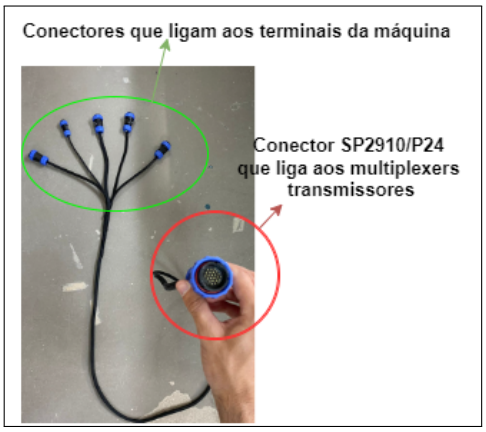
Nas colunas acerca dos conectores, o conteúdo presente em cada linha é relativo ao tipo de conector, mas em alguns casos existe a definição dos pinos em concreto utilizados. Por exemplo, observando a Tabela 4.3, na coluna "Origem" da "Zona B1", o conector SP2912/S17 aparece repetidamente. Isto deve-se ao facto de haver um esclarecimento de quais linhas em específico se conectam. Por exemplo, na linha da tabela que se introduz pela primeira vez o conector SP2912/S17 tem a nomenclatura: "SP2912/S17 "(1-4)". Ou seja, tem-se um total de 4 linhas do conector SP2912/S17, nomeadamente os fios compreendidos entre fio 1 e fio 4, ligados eletricamente ao conector SP1710/S24. Quando não existe a definição dos pinos em concreto de cada conector, significa que se testa o conector nos seus fios elétricos totais.

É importante relembrar que para cada zona da máquina que se pretende testar, existe sempre a necessidade de se desenvolver dois cabos de teste. Um cabo de teste

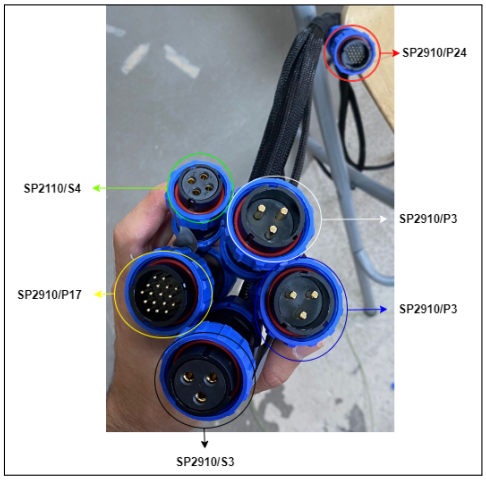
para conectar aos *multiplexers* transmissores (Coluna "Origem") e o outro cabo de teste para conectar aos *multiplexers* recetores (Coluna "Destino").

A título de exemplo, de seguida é apresentado o procedimento que se executou para efetuar o desenvolvimento dos cabos de teste para permitir o teste da zona "B1".

- Cabo teste transmissor - Na Figura 4.5 verifica-se que em relação aos cabos de teste, já não é novidade que uma das extremidades do cabo, possui o conector SP2910/P24. A outra extremidade possui os conectores designados na tabela para os conectores de "Origem". Ou seja, esta extremidade será composta por: 1 conector SP2912/S3, 2 conectores SP2912/S3, 1 conector SP2912/P4 e 1 conector SP2912/S17 (embora neste conector só se tira proveito de 10 pinos, dos totais 17, pois verifica-se pela tabela que este conector tem os pinos de 1-10 a serem usados). Por sua vez, este é o cabo de teste que se irá conectar ao conector SP2912/S24 que se encontra sempre conectado aos *multiplexers* transmissores. Realizando a soma dos pinos, de cada conector, temos o total de 23 pinos (3+3+3+4+10) a serem usados, tal como referido anteriormente.



(a) Visão geral cabo de teste transmissor

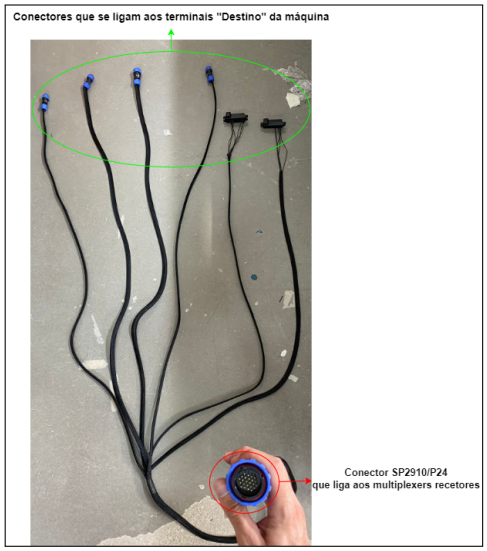


(b) Conectores presentes no cabo de teste transmissor

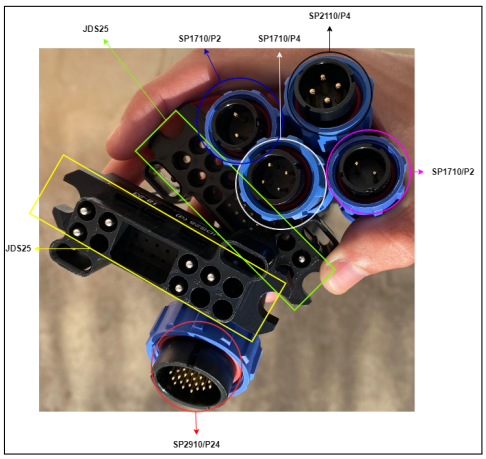
Figura 4.5: Cabo teste transmissor para zona "B1"

- Cabo teste recetor - Na Figura 4.6 observa-se que o cabo de teste recetor, composto pelo conector SP2910/P24 numa extremidade do cabo de teste. Já a outra extremidade possui os conectores designados na tabela para os conectores de "Destino". Ou seja, esta extremidade será composta por: 2 conectores JDS25: 1 JDS25 (chamemos JDS25(A)) tem 5 pinos, dos seus totais 25 preenchidos, a outra JDS25 tem 6 pinos preenchidos, 2 conectores SP1710/P2, 1 conector SP2110/P4 e por 1 conector SP1710/P4. Por sua vez, este é o cabo de teste que se irá conectar aos *multiplexers* recetores. Realizando a soma dos pinos, de cada conector, temos

o total de 23 pinos (5+6+2+2+4+4)a serem usados, tal como referido anteriormente.



(a) Visão geral cabo de teste recetor



(b) Conectores presentes no cabo de teste recetor

Figura 4.6: Cabo teste recetor para zona "B1"

O desenvolvimento dos cabos de teste permitem assim a criação de uma interface entre os *multiplexers* e os fios elétricos da máquina que se pretende testar. O facto de em cada zona da máquina, possuir sempre pelo menos 2 cabos de teste, permite fechar o circuito elétrico entre a máquina e os *multiplexers*, sendo crucial para estabelecer um meio de envio e receção de sinais digitais, entre os diferentes conectores da máquina, e respetivos *multiplexers*, contribuindo para a possibilidade de verificar a continuidade dos fios da máquina e se estão conectados nos seus locais apropriados. Caso um fio esteja num conector que não seria suposto, tal aparecerá na interface gráfica, de forma a avisar o operador que tem que trocar essa ligação errada.

Posteriormente ao desenvolvimento dos cabos de teste, existe a necessidade de efetuar o mapeamento com os *multiplexers*. Isto é, permitir que a interface gráfica, que vai realizar a descodificação do resultado do teste de continuidade, saber que conector e respetivo fio, estava conectado a cada canal dos *multiplexers*. Esta integração entre os *multiplexers* e os cabos de teste, é apresentado de seguida.

4.2.5 Integração entre os cabos de teste e os *multiplexers*

A integração entre os cabos de teste e os *multiplexers* é fundamental para a correta interpretação dos resultados de continuidade. Relembrando, os cabos de teste são constituídos por duas extremidades: uma extremidade tem sempre conector SP2912/P24 e a outra extremidade o conjunto de conectores necessários para efetuar

o teste de continuidade de determinada zona da máquina. A Figura 4.7 apresenta o mapeamento que permite a interpretação correta dos testes de continuidade. Esta figura divide-se no mapeamento dos cabos de teste transmissores com os *multiplexers* transmissores e os cabos de teste recetores com os *multiplexers* recetores. Para cada zona da máquina, presente nesta figura, atribui-se os fios de cada conector da máquina (mesmos conectores da Tabela 4.3) a cada canal individual dos *multiplexers*. Ou seja, ao se saber que o pino de um dado conector do cabo de teste, está conectado a um determinado canal dos *multiplexers*, é possível informar o microcontrolador onde se pretende que este aplique sinais digitais, de forma a validar a continuidade do sistema.

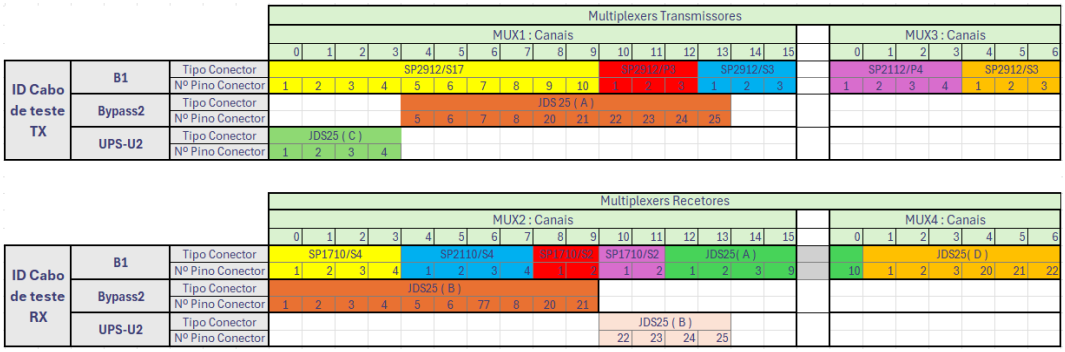


Figura 4.7: Conexões atribuídas entre conectores e respetivos *multiplexers*

Agora que o leitor já está a par da existência de um caractere que é recebido pelo microcontrolador, de forma a identificar o teste que se pretende realizar e também está a par da função 'mapConnections', segue-se o fluxograma presente na Figura 4.8 que auxilia a explicação da necessidade do mapeamento da Figura 4.7. Neste fluxograma, percebe-se que cada zona da máquina tem um identificador para o teste a realizar. De seguida, utiliza-se a função 'mapConnections' com parâmetros também já mencionados. A análise destes parâmetros é que justifica a necessidade do mapeamento da Figura 4.7. A título de exemplo, no teste "Bypass2"vemos a função mapConnections(1,4,13,2,0,9). De segudia, relembram-se o propósito destes parâmetros e a sua integração com o mapeamento associado:

- 1 - Ativa MUX1 para transmitir sinal digital LOW.
- 4 - O primeiro canal do MUX1 a enviar sinal digital LOW.
- 13 - O último canal do MUX1 a enviar sinal digital LOW.
- 2 - Ativa MUX2 para ler sinal digital LOW.
- 0 - O primeiro canal do MUX2 a ler sinal digital LOW.
- 9 - O último canal do MUX2 a ler sinal digital LOW.

Agora, observe-se que na Figura 4.7 o *multiplexer* transmissor que possui o conector JDS25 (A) é o MUX1 em que tem o primeiro fio elétrico no canal 4 do *multiplexer* e tem o último fio elétrico no canal 13 do *multiplexer*. Analogamente, na Figura 4.7 o *multiplexer* recetor que possui o conector JDS25 (B) localiza-se no MUX2, possuindo o primeiro fio elétrico no canal 0 e o último fio elétrico no canal 9 do MUX2. Analisando, são exatamente os mesmos valores que estão introduzidos como parâmetros da função 'mapConnections'.

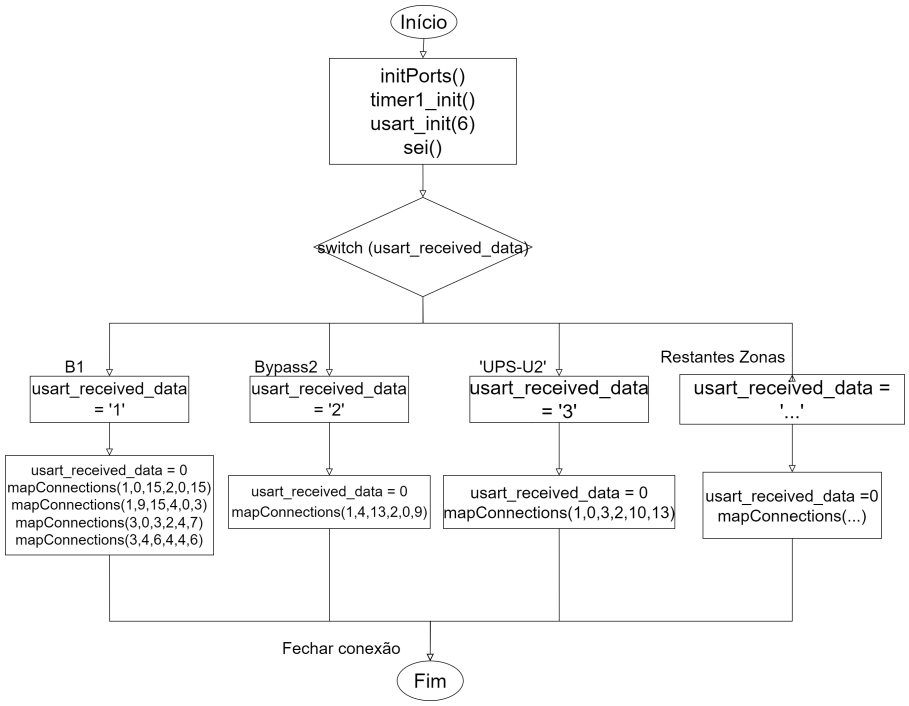


Figura 4.8: Funções de interpretação de teste a realizar, e respetiva execução

Concluindo, a integração entre os cabos de teste e os *multiplexers* é essencial, pois o resultado do teste de continuidade é apresentado no formato de *nibble* onde fornece informação sobre o transmissor (nos 4 bits mais significativos) e informação sobre o recetor (nos 4 bits menos significativos) para a mesma conexão. É possível interpretar os valores dessa conexão, devido ao mapeamento definido na Figura 4.7.

4.2.6 Validação do teste de continuidade

A intenção desta secção é de fornecer uma ideia do tipo de dados que irão futuramente ser transmitidos para a interface gráfica. Para tal, executaram-se os três testes (B1, Bypass2 e UPS-U2) e a receber o *feedback* via RealTerm. O objetivo é demonstrar o processo de validação do correto funcionamento do sistema desenvolvido e garantir que as operações e *work flow* estão em conformidade com as especificações projetadas. Para tal, este sub-capítulo está dividido em três partes principais:

- Os passos que o operador teria que seguir de forma a permitir efetuar o teste de continuidade.
- Análise da string que contém os dados relativos ao teste de continuidade, e perceber como interpretar a *string* que é futuramente enviada para a interface gráfica via UART.
- Ilustração das conexões que têm que ser implementadas tanto nos terminais elétricos da máquina como as conexões que se efetuam aos *multiplexers*, de acordo com a zona da máquina que se deseja testar.

De forma ao operador executar o teste de continuidade tem que seguir uma sequência de ações. Detalha-se de seguida quais seriam os passos a seguir de forma a testar uma zona da máquina:

1. Conectar FT232 entre o microcontrolador e o computador;
2. A partir dos cabos de teste desenvolvidos relativos à zona que se pretende testar, começa-se por: conectar extremidade do conector SP2912/S24 aos *multiplexers* respetivos e a outra extremidade do cabo de teste conecta-se aos respetivos terminais da máquina como é indicado na Tabela 4.3. Aproveita-se este passo para salientar que, consoante o teste da zona da máquina que se pretende testar, é sempre reaproveitado o conector SP2912/S24 que está fixamente conectado aos *multiplexers*, ou seja, para cada teste que se pretende realizar tem que se conectar o cabo de teste desenvolvido para o respetivo teste. No caso do teste de B1 conectava-se o cabo de teste da Figura 4.5b aos *multiplexers transmissores* e o cabo de teste da Figura 4.6b aos *multiplexers* recetores.
3. Na interface gráfica seleciona-se o botão da zona que se pretende testar, que por sua vez, irá enviar uma mensagem via UART ao microcontrolador, instruindo-o qual o teste que se pretende realizar.
4. O microcontrolador responde com uma *string* formatada e, no seu conteúdo, possui os resultados relativos ao teste de continuidade executado.
5. Interface gráfica descodifica essa *string* e apresenta os resultados numa tabela, de forma a ser acessível a visualização do teste de continuidade, por parte do operador.

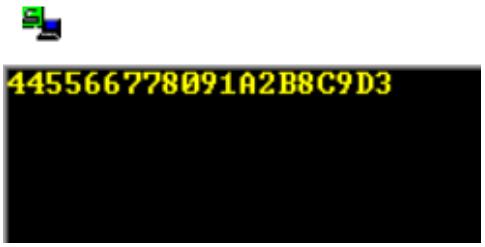
Tais sequência de passos são idênticos para qualquer zona da máquina que se pretenda testar. Simplesmente exige que o operador conecte sempre o cabo de teste apropriado aos terminais elétricos da máquina que se pretende testar.

De seguida serão demonstradas as três *strings*, visto que apenas se desenvolveu cabos de teste para três zonas distintas da máquina, tal como mencionado anteriormente. De forma a visualizar as funcionalidades deste sistema, os resultados do teste de continuidade apresentados irão ser para conexões corretas da máquina.

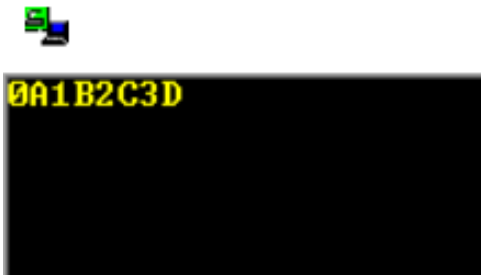
Relembrando, os valores contidos nesta *string* remetem aos canais dos *multiplexers* de forma a informar o transmissor e o recetor, para uma única conexão, no formato de *nibble*. Como tal, de forma a interpretar corretamente os valores, tem que se saber de antemão as conexões que foram implementadas entre os distintos conectores dos terminais máquina com os *multiplexers* do sistema de teste de continuidade.

Apresenta-se de seguida a Figura 4.9 do RealTerm que reúne as três *strings* relativas a cada zona da máquina que se testou a continuidade dos fios. Posteriormente, de forma a interpretar os resultados do teste de continuidade, é necessária a visualização da Tabela 4.3 com a Figura 4.7, de forma complementar.

(a) Mensagem relativa ao resultado correto do teste "B1"



(b) Mensagem relativa ao resultado correto do teste "Bypass2"



(c) Mensagem relativa ao resultado correto do teste "UPS-U2"

Figura 4.9: RealTerm: resultados corretos do teste de continuidade

As imagens presentes na Figura 4.9 demonstram as *strings* expectáveis quando os testes de continuidade são realizados e todos estão em conformidade, ou seja, sem pinos trocados e sem a existência de qualquer fio sem continuidade. Em cenários reais, caso aconteça não haver continuidade, o par de *nibbles* da respetiva conexão não apareceria nesta *string*. Caso houvesse continuidade mas com pinos trocados, essa conexão, embora errada, iria aparecer nesta *string*. É posteriormente, da responsabilidade da GUI verificar que a mesma conexão tem pinos trocados, a partir do ficheiro .txt que será posteriormente explicado.

De forma a elucidar o leitor do significado destas *strings* formatadas, dá-se agora um exemplo prático do teste de continuidade na zona "Bypass2", com auxílio à Figura 4.9b. Antes de se iniciar a explicação, é importante que o leitor analise novamente a Tabela 4.3 e a Figura 4.7 nas informações relativas à zona "Bypass2".

Segue-se a Figura 4.10 que ilustra a divisão desta string em pares de *nibbles*. Recordando que em cada par de *nibble*, o primeiro índex remete ao canal do *multiplexer* transmissor e o segundo índex remete ao canal do *multiplexer* recetor. Sendo assim, observando por exemplo o "Nibble1", onde está a conexão '44'. Isto significa, que o fio ligado ao canal '4' do *multiplexer* transmissor, está conectado ao canal '4' do *multiplexer* recetor. Outro exemplo, no "Nibble5", onde possui a conexão '80', significa que o canal '8' do *multiplexer* transmissor, está conectado ao canal '0' do *multiplexer* recetor. Presentemente, tira-se proveito destes dois exemplos, "Nibble1"e "Nibble5"de forma a explicitar o significado real destes dígitos.

- "Nibble1"(conexão '44') - Ao observar a Figura 4.7, verifica-se que o canal '4' transmissor corresponde ao pino 5 do conector JDS25(A). Já o canal recetor, o canal '4' remete ao pino 5 da JDS25(B).
- "Nibble5"(conexão '80') - De forma semelhante, observando a Figura 4.7 analisa-se que o canal '8' transmissor remete ao pino 20 do conector JDS25 (A). Já o canal '0' recetor, remete ao pino 1 da JDS25 (B).

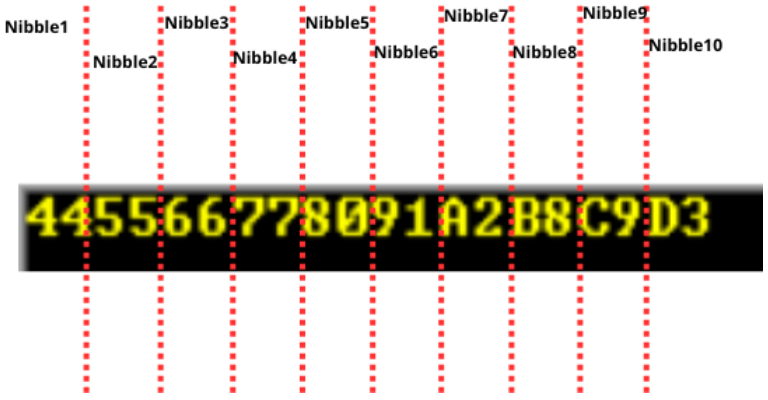


Figura 4.10: Divisão em *nibbles* a *string* resultante do teste "Bypass2"

Com o intuito de auxiliar o leitor na análise da Tabela 4.3, e, posteriormente analisar o mapeamento das conexões disponível na Figura 4.7, segue a Figura 4.11 que ilustra os passos a seguir na interpretação do exemplo "Nibble1"e "Nibble5". A cor preta representa os pontos chave que se deve analisar para fundamentar o exemplo "Nibble1", e, a cor azul, remete à análise dos pontos chave para explicar o exemplo "Nibble5".

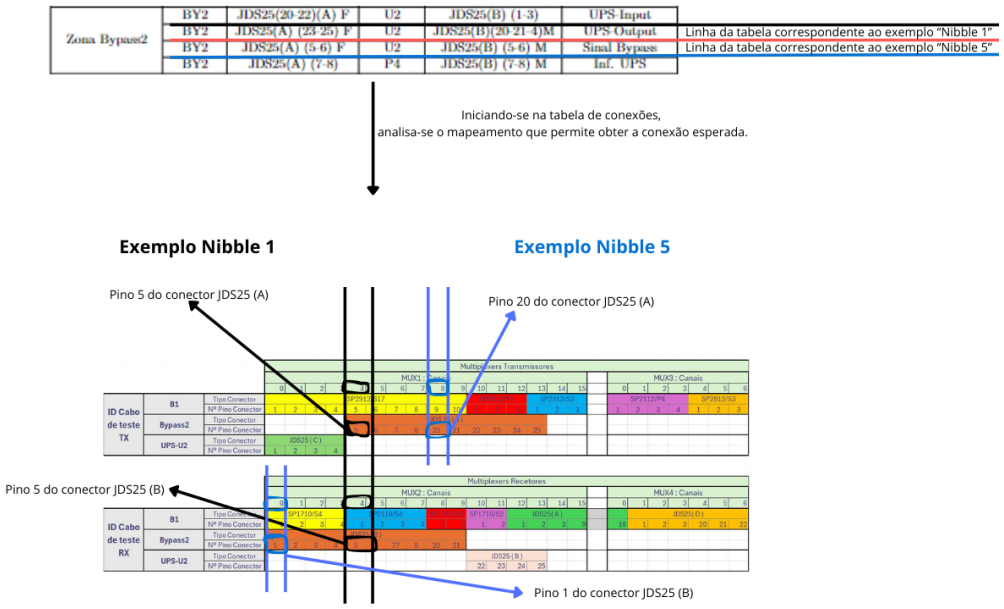


Figura 4.11: Etapas que permitem a correta análise da *string* do resultado do teste de continuidade

Ou seja, estes dois exemplos fornecem a seguinte informação: Nibble1, a partir da conexão '44', traduz-se que o pino 5 do conector JDS25(A) transmissor, está conectado ao pino 5 do conector JDS25 (B) recetor. E, por sua vez, a conexão '80' do Nibble5 traduz que o pino 20 do conector JDS25 (A) está conectado ao pino 1 da JDS25 (B). Agora, observe-se a Tabela 4.3 onde é possível verificar, que na zona "Bypass2", na terceira linha, efetivamente existe a conexão entre o pino 5 da JDS25 (A) e o seu correspondente pino 5 da JDS25 (B). De forma semelhante, é possível verificar que na primeira linha da tabela, na zona "Bypass2", evidencia-se que o pino 20 da JDS25 (A) está conectado ao pino 1 da JDS25 (B).

Embora se tenha utilizado o teste "Bypass2"para dar dois exemplos de funcionamento, esta abordagem é exatamente igual aos testes das outras zonas da máquina. Com esta explicação mais prática, pretende-se que o leitor compreenda mais facilmente o tratamento de dados que existe por parte da interface gráfica, tal como foi efetuado nestes dois exemplos, de forma a descodificar a *string* do resultado de teste de continuidade.

4.3 Interface Gráfica

De forma a permitir ao operador a visualização dos testes de continuidade executados pelo sistema baseado no microcntrolador, e, visualização de leitura de dados, provenientes de sensores embebidos com os protocolos: modbus RTU, modbus TCP e SNMP, desenvolveu-se a presente interface gráfica que oferece resposta às necessidades abordadas pela empresa.

A interface gráfica foi desenvolvida a partir da *framework* Tkinter, baseada na linguagem de programação Python. Tkinter foi escolhida pela sua simplicidade e flexibilidade, que permitiu a criação de uma ferramenta visual onde foi possível implementar as funcionalidades exigidas.

Durante este sub-capítulo entre os subcapítulos (4.3.2 e 4.3.6) serão expostas as funcionalidades da interface gráfica, relacionadas à apresentação dos resultados provenientes do sistema de teste de continuidade, como foi ilustrado na Figura 3.7. Num ponto de vista geral, abordar-se-à os seguintes tópicos:

- Estabelecimento da conexão e respetiva comunicação série (UART).
- Tratamento de dados provenientes do microcontrolador.
- A importância do ficheiro .txt relativamente ao mapeamento de valores provenientes do microcontrolador, de forma a permitir que a tabela apresente valores com significado para o utilizador, e não apenas uns números que à primeira vista não parecem ter qualquer significado.
- Respetiva atualização da tabela com os valores finais.

Entre os subcapítulos (4.3.7 e 4.3.9) é abordado o desenvolvimento desta aplicação, como ferramenta de leitura de dados provenientes de sensores que utilizam os protocolos Modbus e SNMP, visto serem os protocolos em uso por parte dos sensores da máquina. O propósito da implementação desta funcionalidade na interface gráfica, deve-se ao facto de por vezes os sensores não estarem a funcionar, sem inicialmente ter a certeza do motivo. Ao possuir uma interface gráfica que integra tanto a realização de testes de continuidade, como também a de leitura de dados, facilmente se conclui se o mau funcionamento dos sensores se deve a configurações de *firmware* do mesmo, ou se é devido à camada física (a partir dos testes de continuidade nos fios elétricos que conectam aos sensores). Ao longo destes subcapítulos, serão explorados os principais aspetos técnicos da implementação desta funcionalidade. Desde os parâmetros que o operador tem que inserir de forma a ter acesso aos dados dos sensores como também a respetiva apresentação de leituras na interface.

Por fim, apresentam-se os testes e resultados da interface gráfica, tanto como ferramenta de apresentação de testes de continuidade, como de leitura de dados provenientes de sensores.

4.3.1 Apresentação geral da interface gráfica

Ao abrir o GUI o é apresentada a única página desta aplicação, onde reúne todas as suas funcionalidades, como se pode visualizar na Figura 4.12. No entanto, em botões como: "TCP Config", "Modbus Config", "GET"e "WALK" existe a criação janelas adicionais, de forma a ser possível introduzir variáveis necessárias de acordo com a configuração do fabricante dos diversos sensores.

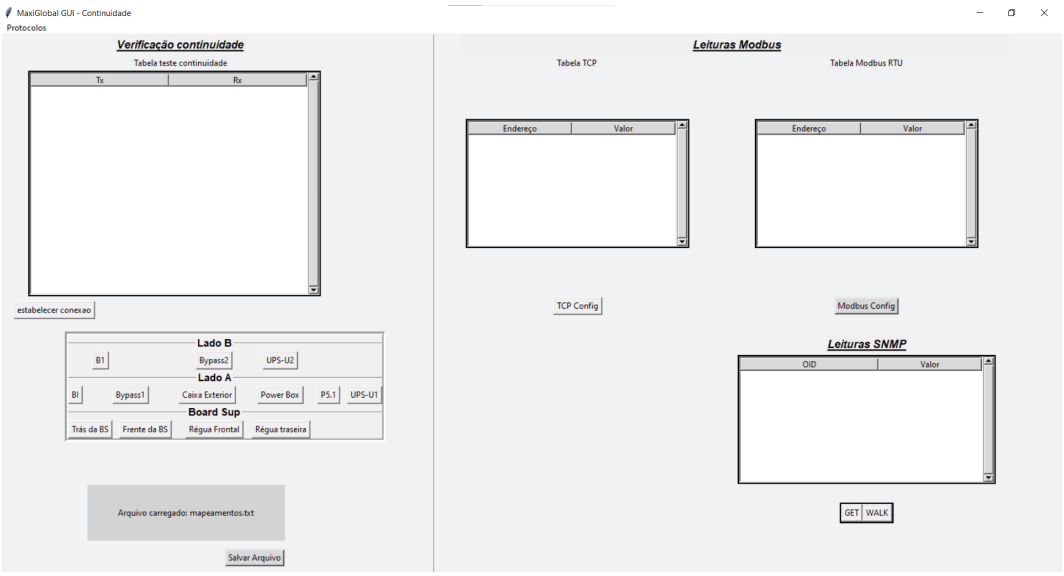


Figura 4.12: *Layout* da interface gráfica

Na Figura 4.12 chama de imediato à atenção a divisão desta interface:

- O lado esquerdo é responsável pelas funcionalidades associadas à apresentação dos resultados do teste de continuidade das diferentes zonas da máquina, obtidos a partir do microcontrolador. Como já foi mencionado anteriormente, a interface demonstra a nomenclatura das diferentes zonas da máquina. Os botões "B1", "Bypass2"e "UPS-U2" já são nomes familiares, visto terem sido estas as únicas zonas onde houve tempo para desenvolver e soldar os cabos de teste associados, no entanto, o raciocínio é exatamente o mesmo para as restantes 10 zonas. Por fim, na área sombreada apresenta-se a funcionalidade de "*Drag & Drop*" de ficheiros .txt onde estão contidos o mapeamento dos conectores e, onde os mesmos, estão conectados aos *multiplexers*, de forma a ser possível preencher a tabela com dados relevantes, e não, apenas os números das ligações entre os canais dos *multiplexers* transmissores e recetores. Tal como mencionado anteriormente, este ficheiro .txt foi o escolhido por ser mais cómodo para os operadores. No entanto, todo o conteúdo inserido neste ficheiro, tem o formato e estrutura de dados JSON.

- O lado direito diz respeito às funcionalidades da leitura de dados dos diversos sensores, a partir de leituras dos protocolos Modbus e SNMP, onde se teve em

atenção em desenvolver funcionalidades idênticas às de ferramentas como: Modbus Poll e MIB Browser. De salientar que para operações de leitura SNMP e Modbus TCP, o utilizador, no seu computador pessoal, tem que se inserir na rede do sensor que pretende realizar uma leitura.

No lado da funcionalidade de leitura de dados, os seguintes quatro botões presentes: '`TCP Config`', '`Modbus Config`', '`GET`' e finalmente '`WALK`', têm cada um deles uma janela associada. Em cada uma destas janelas é esperado o utilizador colocar dados associados ao dispositivo que pretendem ler e no formato respetivo. Posteriormente, ao clicar em "Save and Connect"a janela automaticamente é fechada e os dados inseridos são recolhidos pela aplicação, sendo posteriormente utilizados para efetuar diversas operações que serão explicadas no decorrer do capítulo.

4.3.2 Estabelecer comunicação entre microcontrolador e interface gráfica

De forma a permitir a comunicação série entre o microcontrolador e a interface gráfica tem-se em atenção os botões: "Atualizar Portas" e "Estabelecer Conexão". O utilizador quando se encontra na aplicação, primeiramente insere o conversor FT232 e, posteriormente, ao clicar no botão "Atualizar Portas" a aplicação irá detetar todos os dispositivos USB conectados ao computador, desde o teclado, rato, fones, entre outros. No caso do conversor FT232 utilizado nesta dissertação é 'COM6'. Após selecionar esta porta de comunicação, o utilizador clica no botão "Estabelecer Conexão". A função responsável por estabelecer tal conexão, '`connect_serial`', estabelece a comunicação com dois argumentos: porta de comunicação (PORT6) e a taxa de transmissão, *baudrate*, que está definida a 9600 bps. A razão para não permitir, por parte do utilizador, selecionar a taxa de transmissão deve-se ao facto do microcontrolador já estar a 9600 bps, ou seja, se o utilizador definir uma taxa de transmissão diferente, a comunicação não iria ser viável, uma vez que os dois extremos de comunicação estariam com diferentes taxas de transmissão. A conexão sendo iniciada, a aplicação aguarda um tempo breve para a inicialização da porta série. Em caso de sucesso, uma mensagem de confirmação é exibida ao operador, enquanto que em cenários de erro, estes são expostos por mensagens de erro apropriadas. Na Figura 4.13 é possível averiguar as etapas que permitem estabelecer conexão UART entre a GUI e o microcontrolador, com recurso ao conversor FT232.

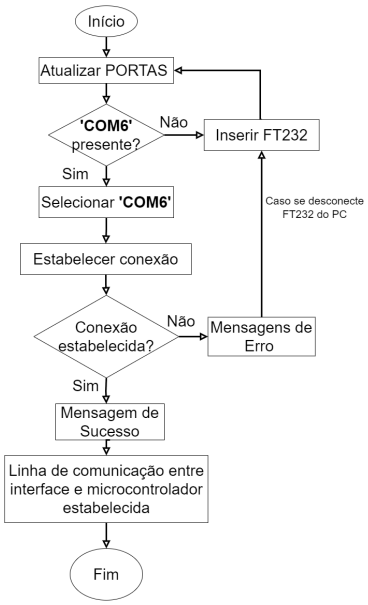
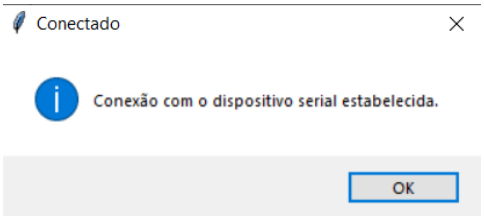


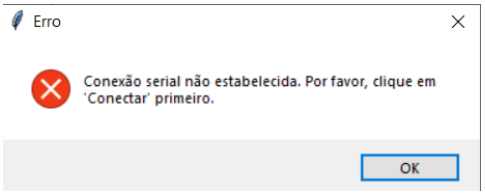
Figura 4.13: Fluxograma explicativo das etapas que permitem estabelecer conexão entre micocontrolador e interface gráfica

Durante a execução da aplicação, existe uma monitorização contínua, da responsabilidade da função '`check_serial()`' a averiguar se, por alguma razão, o FT232 se desconectar e interromper a comunicação, é apresentada uma mensagem de erro a notificar o utilizador. Esta função verifica periodicamente, em intervalos de 100 milissegundos, a presença do FT232.

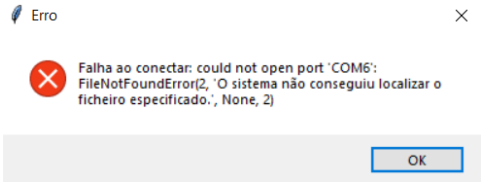
Na Figura 4.14 estão presentes as três mensagens informativas fornecidas pela aplicação relatavimente à conexão série. Começando pela Figura 4.14a esta mensagem indica uma conexão estabelecida com o FT232. Caso o operador clique num botão para realizar o teste de continuidade, em qualquer zona da máquina, se não estabeleceu previamente conexão série, a mensagem de erro que é apresentada é a da Figura 4.14b. Por fim, caso tenha sido efetuada uma conexão sucedida, mas se o componente FT232 for desconectado aparece a mensagem de erro presente na Figura 4.14c



(a) Mensagem informativa de conexão estabelecida com sucesso



(b) Mensagem de erro devido a operador tentar realizar teste, sem conectar previamente



(c) Mensagem de erro devido a FT232 se ter desconectado em certo instante

Figura 4.14: Mensagens informativas

4.3.3 Seleção do teste de continuidade que se pretende realizar

Uma vez estabelecida a conexão entre a interface gráfica e o sistema que efetua os testes de continuidade, existe a necessidade de selecionar a zona da máquina que o operador pretende testar.

Primeiramente, cada zona da máquina que se pretende testar tem um botão associado como já foi mencionado anteriormente. E, por sua vez, cada botão, ao ser clicado, envia um caractere (considerado como um identificador (ID) do respetivo teste) para o microcontrolador, de forma a informar o microcontrolador qual o teste que se pretende realizar. Para além do número enviado ao microcontrolador, consoante o botão clicado, a interface gráfica vai extrair o mapeamento das conexões .txt para cada teste, de forma a posteriormente esta ser capaz de atribuir o nome dos conectores às conexões encontradas, por parte do teste de continuidade. Este caractere pode ser visto como um identificador do teste que se pretende realizar. Tal como está representado no fluxograma da Figura 4.8, existe a atribuição dos números '1','2' e '3' para testar "B1", "Bypass2"e "UPS-U2", respetivamente. De forma complementar, tal como ilustra o fluxograma da Figura 4.15, os botões presentes na interface gráfica enviam os caracteres '1', '2' ou '3' para o microcontrolador, consoante o teste que se pretende realizar. Posteriormente o microcontrolador realiza os testes de acordo com o número (ID) que recebeu e retorna as conexões que encontrou. Para além disso, no momento de seleção do teste que se pretende realizar, a GUI automaticamente extrai os mapeamentos do ficheiro .txt para o teste que se está a realizar.

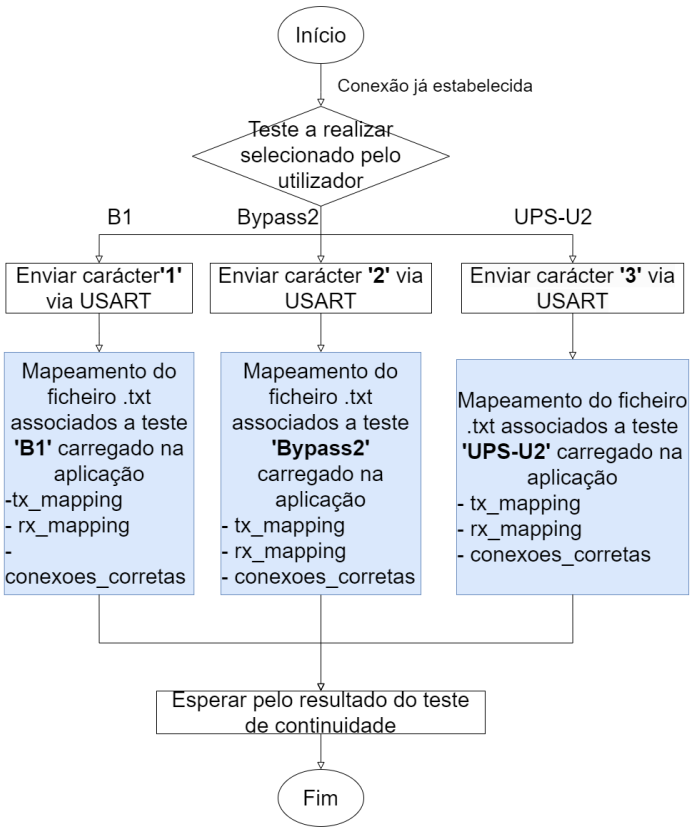


Figura 4.15: Fluxograma explicativo das etapas que permitem definir a zona da máquina a ser testada

4.3.4 Receção do resultado do teste de continuidade

Após a seleção do teste a realizar ser efetuada, o sistema responsável pelo teste de continuidade vai executar o teste, de acordo com o botão que o utilizador clicou. Posteriormente, o microcontrolador envia, via UART, o resultado desta operação.

De forma à interface gráfica ser capaz de captar os dados transmitidos pelo microcontrolador, esta, a partir da função '`read_serial()`', executa a leitura e armazena os dados num *buffer*. De forma à aplicação perceber quando se finalizou a receção de dados, existem as variáveis: 'timeout', 'current_time' e 'last_received_time'. O valor definido para 'timeout' é de 8 segundos. Na variável 'last_received_time' é alocado o instante de tempo em que a função '`read_serial()`' leu dados pela última vez, esvaziando o canal de comunicação. Já 'current_time', detetando que não estão a aparecer lados para ser lidos, vai incrementando o seu valor até que ao ponto em que se verifica a condição, onde a subtração entre 'current_time' e 'last_received_time' atinge o valor de 'timeout' (8 segundos), aqui a função '`read_serial()`' é instruída para finalmente alocar os dados no *buffer* e chamar as funções responsáveis pelo tratamento de dados. Esta etapa de receção do resultado proveniente do sistema responsável pelo teste de continuidade, é apresentada na Figura 4.16.

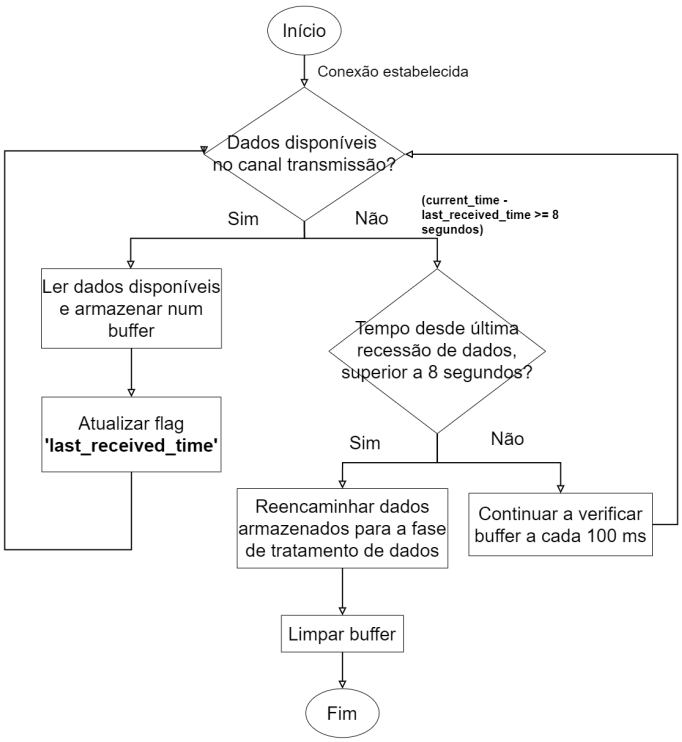


Figura 4.16: Fluxograma respetivo à receção de dados, por parte da GUI

4.3.5 Tratamento e descodificação do resultado do teste de continuidade

É então, da responsabilidade da interface gráfica de tratar e interpretar o resultado do teste de continuidade. Os dados enviados pelo microcontrolador, inicialmente são lidos pela função '`read_serial()`' e são convertidos para formato hexadecimal. A função '`processar_mensagem_nibbles()`' realiza a separação da *string* em pares de *nibbles* (unidades de 4 bits), anteriormente mencionados, e remove 2 ligações em particular, do teste 'B1', que são as ligações: '49' e '58'. A razão destas conexões serem removidas deve-se ao facto de, internamente, a máquina possuir uma resistência que realiza um divisor de tensão, tornando a leitura da conexão como um único ponto elétrico: '48','49','58','59'. No entanto, para não haver confusão por parte do operador, estas conexões são sempre removidas, mantendo então, as conexões corretas '48' e '59'.

De seguida, o mapeamento de valores é essencial pois expõe os dados relativos ao teste de continuidade, com significado real para o utilizador. Este mapeamento é realizado pelo conjunto da função '`mapeamento_for_same_key_values()`' com o ficheiro .txt que possui os mapeamentos. Por default existe já um ficheiro .txt na aplicação, no entanto, caso a máquina sofra alterações elétricas com o decorrer do tempo, este ficheiro pode ser atualizado com as novas ligações e assim este sistema

mantém-se funcional. Após atualizar o ficheiro .txt, basta arrastá-lo para a área sombreada respetiva, clicar em "Salvar Arquivo"e este torna-se o default daí em diante. A função '`mapeamento_for_same_key_value`' recebe então múltiplos pares de *nibbles* de acordo com o teste que o utilizador optou por realizar. Esta função, divide cada par de *nibbles* em metade, ou seja, uma conexão '8a', por exemplo, ficaria dividida em '8' e 'a'. Posteriormente, ao criar duas listas distintas, armazena na lista '`lista_tx_final`' o índex 0 de cada par de *nibble*, visto que estes são o identificador do canal dos *multiplexers* transmissores. De forma idêntica, a lista '`lista_rx_final`' armazena o índex 1, de cada par de *nibble*, sendo que, estes são o identificador do canal dos *multiplexers* recetores. Quando ambas as listas são preenchidas com todos os elementos das conexões do teste de continuidade, é nesta fase que o ficheiro .txt é necessário. No entanto, como já foi referido anteriormente, os *multiplexers* têm dados sobrepostos entre os dígitos 0 a 6 e para tal, é necessário que a interface interprete qual é o tipo de conector, e respetivo pino, para cada conexão. Como tal, existe a necessidade de duas variáveis que atuam como 'contadores', sendo elas: '`tx_occurrence_tracker`' para o mapeamento dos canais que se repetem dentro dos *multiplexers* transmissores, e '`rx_occurrence_tracker`' para o mapeamento dos canais que se repetem, dentro dos *multiplexers* recetores. Esta descodificação do resultado de teste de continuidade, é apresentado, sob a forma de fluxograma na Figura 4.17.

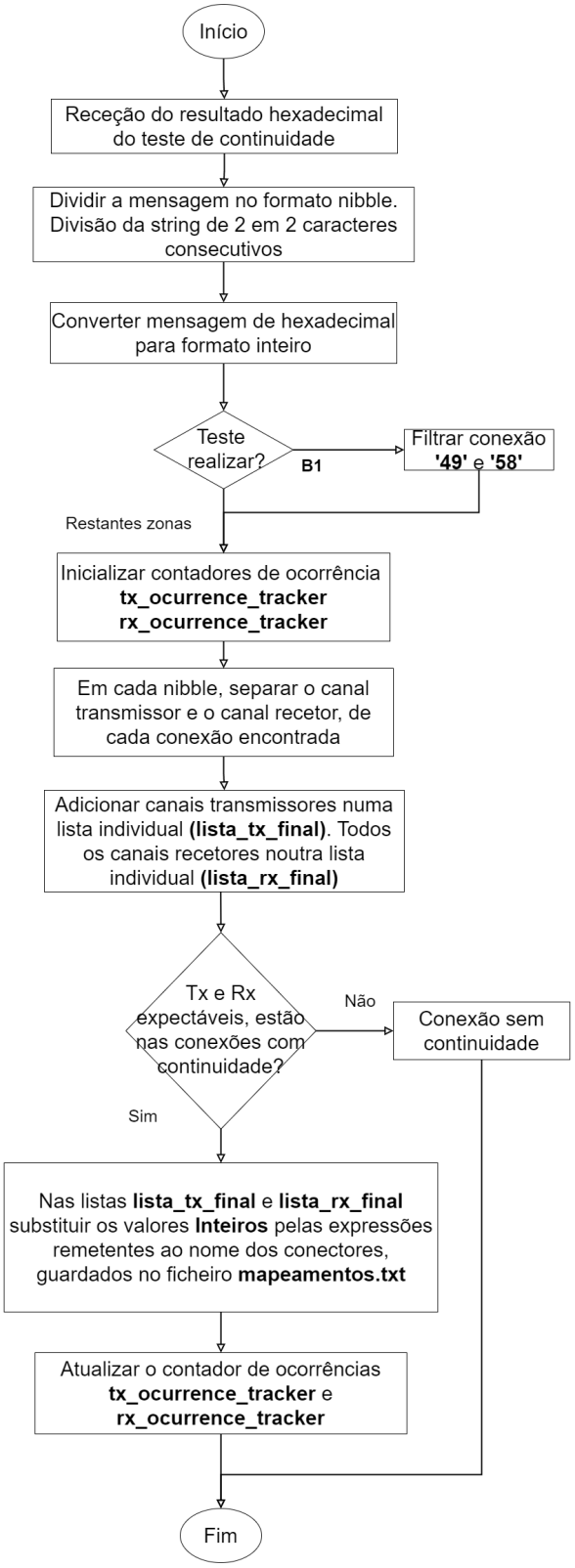


Figura 4.17: Fluxograma explicativo do tratamento de dados realizado ao resultado do teste de continuidade

4.3.6 Apresentação dos resultados ao operador

Após o tratamento dos dados existe a necessidade do resultado ser apresentado na interface gráfica sob a forma de tabela. De forma a implementar esta funcionalidade, desenvolveu-se a função `update_table()` que é responsável por atualizar a tabela de continuidade com os resultados do teste de continuidade realizado. Na Figura 4.18 é possível verificar que esta função começa por limpar qualquer tipo de dados existentes previamente na tabela. Recorrendo às duas listas criadas, '`lista_tx_final`' e '`lista_rx_final`' esta função compara os valores nestas listas com os valores expectáveis para cada teste que foram carregados a partir do ficheiro `mapeamentos.txt`.

Este ficheiro contém os valores corretos, de acordo com o esquema elétrico da máquina, para cada combinação entre Tx e Rx, que por sua vez, estão armazenadaos no elemento '`conexoes_corretas` do ficheiro `mapeamentos.txt`. A função, percorre todos os valores de Tx esperados, presentes nesse dicionário de formato *key/values*, e, verifica se o correspondente valor de Rx obtido pelo teste de continuidade, coincide com o valor de Rx esperado. Caso o valor de Rx obtido para um dado Tx, for igual ao valor de Rx esperado, a linha correspondente a essa conexão é colorida a verde na tabela. Se o valor de Rx não for o esperado, significa que houve uma troca de fios na produção da cablagem, e, como tal, a linha da tabela é colorida a bege. Por fim, caso um fio não possua continuidade, o teste de continuidade realizado nem sequer vai detetar a existência dessa conexão. Como tal, a função `update_table()` ao comparar o valor de Tx que não apareceu com os valores esperados, fornecidos por `mapeamentos.txt`, vai manter essa conexão na tabela, mas colorida a cor vermelha. O intuito da cor bege é de informar o utilizador que a conexão que corresponde a uma linha bege, significa que esta tem um dos extremos do fio no local errado. O utilizador, ao averiguar qual foi a conexão que se encontrou, facilmente retifica este fio, colocando-o no local esperado. No caso da cor vermelha, isto significa que esta conexão não possui continuidade, informando o operador que deve prosseguir à substituição do fio danificado.

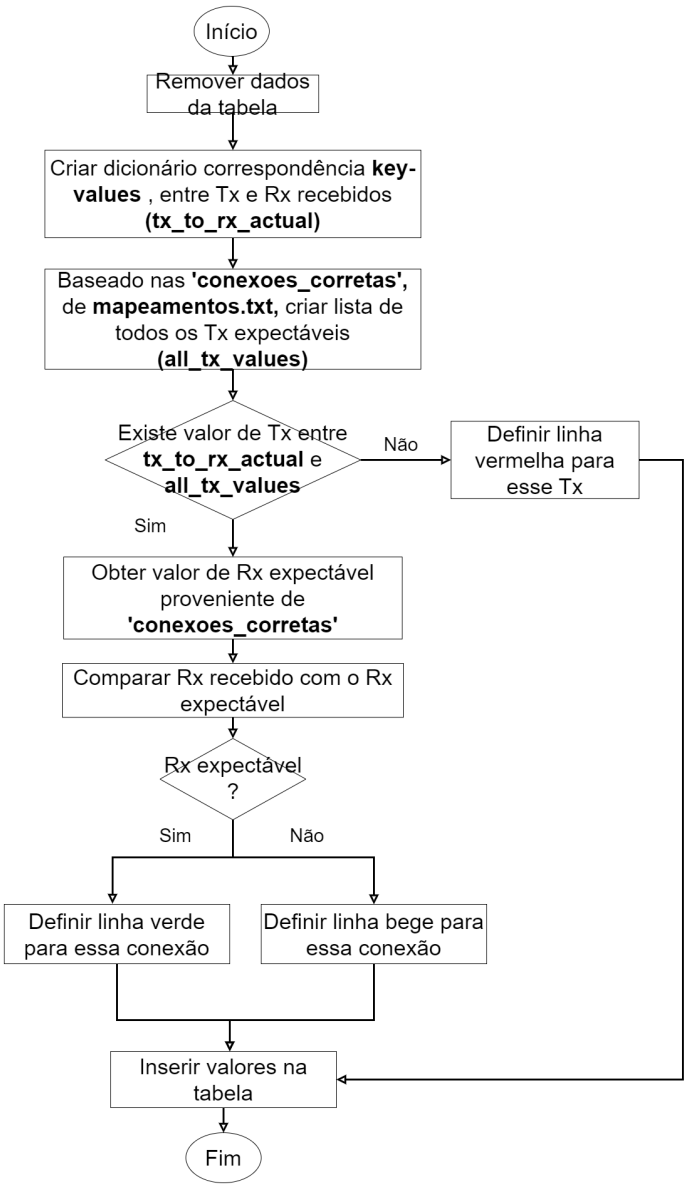
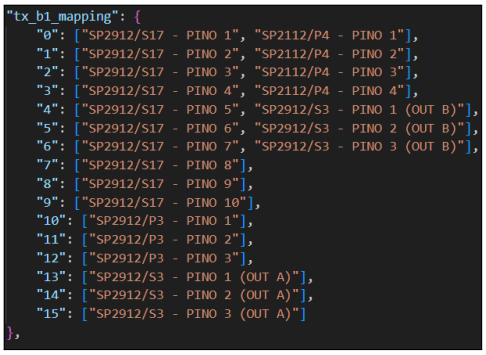


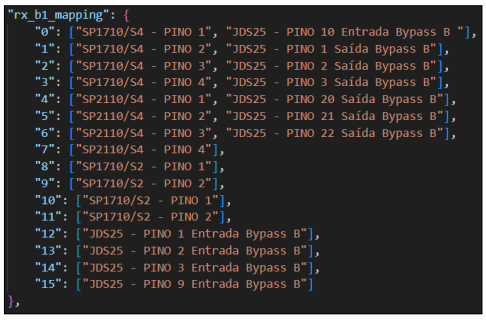
Figura 4.18: Fluxograma relativamente à atualização da tabela

Na Figura 4.19a apresenta-se o mapeamento relativo aos canais do *multiplexer* transmissor, na Figura 4.19b estão presentes as conexões referentes aos canais do *multiplexer* recetor, e, por fim, na Figura 4.19c está o mapeamento das conexões expectáveis. É a partir dos mapeamentos da Figura 4.19c que a interface gráfica consegue comparar as conexões obtidas com as conexões reais, permitindo assim a atribuição de cores em concordância com: falta de continuidade (cor vermelha), fios trocados (cor bege) e por fim, a cor verde confirma que as conexões estão de acordo com as expectáveis.



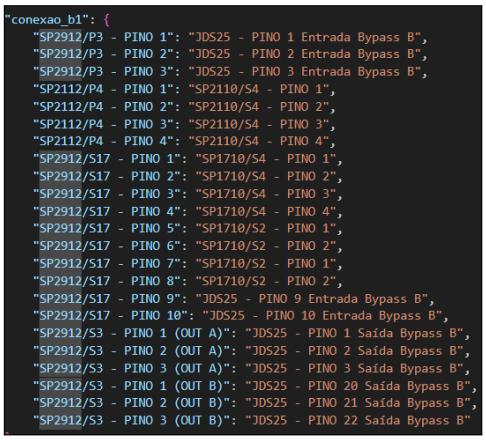
```
"tx_b1_mapping": {
    "0": ["SP2912/S17 - PINO 1", "SP2112/P4 - PINO 1"],
    "1": ["SP2912/S17 - PINO 2", "SP2112/P4 - PINO 2"],
    "2": ["SP2912/S17 - PINO 3", "SP2112/P4 - PINO 3"],
    "3": ["SP2912/S17 - PINO 4", "SP2112/P4 - PINO 4"],
    "4": ["SP2912/S17 - PINO 5", "SP2912/S3 - PINO 1 (OUT B)"],
    "5": ["SP2912/S17 - PINO 6", "SP2912/S3 - PINO 2 (OUT B)"],
    "6": ["SP2912/S17 - PINO 7", "SP2912/S3 - PINO 3 (OUT B)"],
    "7": ["SP2912/S17 - PINO 8"],
    "8": ["SP2912/S17 - PINO 9"],
    "9": ["SP2912/S17 - PINO 10"],
    "10": ["SP2912/P3 - PINO 1"],
    "11": ["SP2912/P3 - PINO 2"],
    "12": ["SP2912/P3 - PINO 3"],
    "13": ["SP2912/S3 - PINO 1 (OUT A)"],
    "14": ["SP2912/S3 - PINO 2 (OUT A)"],
    "15": ["SP2912/S3 - PINO 3 (OUT A)"]
},
```


(a) Mapeamento para *multiplexers* transmissores



```
"rx_b1_mapping": {
    "0": ["SP1710/S4 - PINO 1", "JDS25 - PINO 10 Entrada Bypass B "],
    "1": ["SP1710/S4 - PINO 2", "JDS25 - PINO 1 Saída Bypass B"],
    "2": ["SP1710/S4 - PINO 3", "JDS25 - PINO 2 Saída Bypass B"],
    "3": ["SP1710/S4 - PINO 4", "JDS25 - PINO 3 Saída Bypass B"],
    "4": ["SP2110/S4 - PINO 1", "JDS25 - PINO 20 Saída Bypass B"],
    "5": ["SP2110/S4 - PINO 2", "JDS25 - PINO 21 Saída Bypass B"],
    "6": ["SP2110/S4 - PINO 3", "JDS25 - PINO 22 Saída Bypass B"],
    "7": ["SP2110/S4 - PINO 4"],
    "8": ["SP1710/S2 - PINO 1"],
    "9": ["SP1710/S2 - PINO 2"],
    "10": ["SP1710/S2 - PINO 1"],
    "11": ["SP1710/S2 - PINO 2"],
    "12": ["JDS25 - PINO 1 Entrada Bypass B"],
    "13": ["JDS25 - PINO 2 Entrada Bypass B"],
    "14": ["JDS25 - PINO 3 Entrada Bypass B"],
    "15": ["JDS25 - PINO 9 Entrada Bypass B"]
},
```


(b) Mapeamento para *multiplexers* recetores



```
"conexao_b1": {
    "SP2912/P3 - PINO 1": "JDS25 - PINO 1 Entrada Bypass B",
    "SP2912/P3 - PINO 2": "JDS25 - PINO 2 Entrada Bypass B",
    "SP2912/P3 - PINO 3": "JDS25 - PINO 3 Entrada Bypass B",
    "SP2112/P4 - PINO 1": "SP2110/S4 - PINO 1",
    "SP2112/P4 - PINO 2": "SP2110/S4 - PINO 2",
    "SP2112/P4 - PINO 3": "SP2110/S4 - PINO 3",
    "SP2112/P4 - PINO 4": "SP2110/S4 - PINO 4",
    "SP2912/S17 - PINO 1": "SP1710/S4 - PINO 1",
    "SP2912/S17 - PINO 2": "SP1710/S4 - PINO 2",
    "SP2912/S17 - PINO 3": "SP1710/S4 - PINO 3",
    "SP2912/S17 - PINO 4": "SP1710/S4 - PINO 4",
    "SP2912/S17 - PINO 5": "SP1710/S2 - PINO 1",
    "SP2912/S17 - PINO 6": "SP1710/S2 - PINO 2",
    "SP2912/S17 - PINO 7": "SP1710/S2 - PINO 1",
    "SP2912/S17 - PINO 8": "SP1710/S2 - PINO 2",
    "SP2912/S17 - PINO 9": "JDS25 - PINO 9 Entrada Bypass B",
    "SP2912/S17 - PINO 10": "JDS25 - PINO 10 Entrada Bypass B",
    "SP2912/S3 - PINO 1 (OUT A)": "JDS25 - PINO 1 Saída Bypass B",
    "SP2912/S3 - PINO 2 (OUT A)": "JDS25 - PINO 2 Saída Bypass B",
    "SP2912/S3 - PINO 3 (OUT A)": "JDS25 - PINO 3 Saída Bypass B",
    "SP2912/S3 - PINO 1 (OUT B)": "JDS25 - PINO 20 Saída Bypass B",
    "SP2912/S3 - PINO 2 (OUT B)": "JDS25 - PINO 21 Saída Bypass B",
    "SP2912/S3 - PINO 3 (OUT B)": "JDS25 - PINO 22 Saída Bypass B"
```


(c) Mapeamento das conexões expectáveis

Figura 4.19: Mapeamentos do ficheiro .txt

4.3.7 Leitura de dados provenientes de sensores Modbus RTU

De forma a realizar a leitura de dados o utilizador primeiramente introduz os parâmetros como a taxa de transmissão (*baudrate*), *bits* de paridade, número de *bits* de dados que que se pretende ler e stop *bits*, permitindo a leitura dos registos do sensor. Tais parâmetros são introduzidos na janela da interface, apresentada na Figura 4.20.

Quanto à implementação desta funcionalidade, foi desenvolvida uma tabela, formada por duas colunas: 'Endereço' e 'Valor', permite a visualização dos dados dos sensores. Para além disso, existe uma *scroll bar* associada à tabela de forma a se poder navegar na mesma para leituras onde haja uma maior quantidade de dados. Para garantir que os *inputs* fornecidos pelo utilizador são válidas, foram definidas funções de validação, que apenas permitem a introdução de parâmetros em números inteiros ou formatos IP para campos desse propósito.

Após a introdução dos parâmetros do equipamento e clique no botão "Save and Connect", a função '`save_config()`' armazena em variáveis individuais, os respetivos dados definidos pelo utilizador, através do método `.get()` de forma a serem utilizados na leitura de dados do equipamento. De seguida, a partir da função

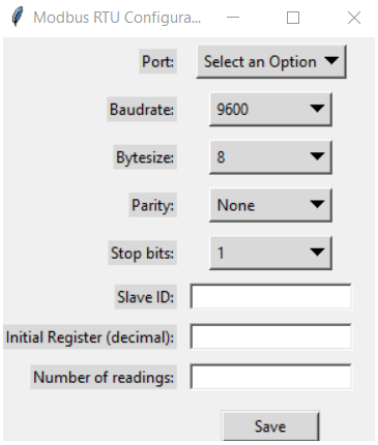


Figura 4.20: Parâmetros a serem introduzidos pelo utilizador, relativos a leituras Modbus RTU

'`config_modbus(port, baudrate, bytesize, parity, stopbits, slave_id, starting_register, count)`', reunindo os dados introduzidos pelo utilizador, estabelece conexão com o equipamento, e, caso a conexão seja bem sucedida e os dados fornecidos estejam corretos,efetua a leitura dos registos do mesmo e retorna os registos para a função `save_config()` e que por sua vez, atualiza a tabela. Caso contrário a função retorna uma mensagem de erro e termina.

4.3.8 Leitura de dados provenientes de sensores Modbus TCP

Previamete a executar leituras, o utilizador necessita alterar as definições da rede do próprio computador, sendo necessário estar na rede do equipamento, de forma a ser possível estabelecer conexão. De forma semelhante, ao clicar no botão 'RTU Config' é aberta uma janela que permite ao utilizador introduzir dados relativos a 'IP Address', 'Port', 'Start Register (decimal)' e 'Register Count', tal como se pode verificar na Figura 4.21. Existe uma verificação para que o utilizador apenas introduza números inteiros através da função '`only_integers()`', e, no campo de 'IP Address' a função '`validate_ip_address`' certifica-se que este campo é preenchido apenas por parâmetros no formato de endereço de IP.

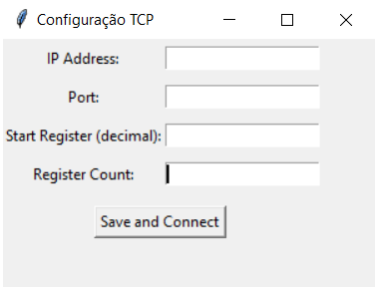


Figura 4.21: Parâmetros a serem introduzidos pelo utilizador, relativos a leituras Modbus TCP

Após o utilizador preencher os campos com informações do equipamento, a função '`save_and_connect()`' é responsável por limpar qualquer dado existente na tabela, armazenar em variáveis independentes os valores inseridos e apresentar na tabela os valores retornados pela função '`connect_read_tcp(ip, port, start_register, count`'. Esta última função, é responsável por estabelecer a conexão Modbus TCP usando esses parâmetros, de forma a ser possível ler os registos desejados. Inicialmente cria um cliente TCP e tenta conectar ao equipamento, caso conexão seja efetuada com sucesso, lê os registos especificados e retorna o resultado para a função '`save_and_connect`' que por sua vez atualiza a tabela com os dados obtidos. Caso a conexão não seja estabelecida com sucesso, retorna uma mensagem de erro.

4.3.9 Leitura de dados provenientes de sensores SNMP

À semelhança da operação de leitura do modbus TCP, na leitura de dados SNMP, o utilizador tem que colocar o próprio computador pessoal na mesma rede que o sensor se encontra, de forma a permitir realizar a leitura de dados com sucesso. No entanto, é importante salientar que no caso das leituras provenientes de sensores SNMP, apenas se está a testar a funcionalidade dos elementos "agentes"deste protocolo.

Esta interface permite que o utilizador introduza parâmetros como: o endereço IP do dispositivo, a *community* (refere-se a um tipo de *password* que permite o controlo de acesso do dispositivo), a porta de comunicação, e o *Object Identifier* (OID), que especifica o dado que se deseja obter do dispositivo, daí este último parâmetro estar apenas incluído na operação "GET". A Figura 4.22 apresenta a janela associada ao clique no botão "GET".

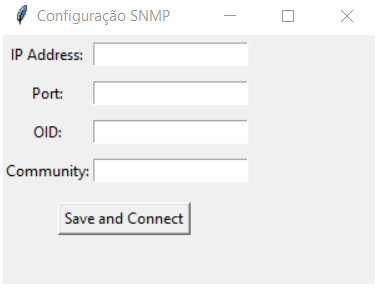


Figura 4.22: Parâmetros necessários para efetuar leitura GET

No caso do botão "GET", o utilizador ao desejar realizar uma leitura de um OID específico, insere então os parâmetros associados ao equipamento e posteriormente clica no botão "Save and Connect". Nos bastidores, a aplicação comunica com o dispositivo, enviando uma solicitação para obter o valor associado ao OID fornecido. A resposta por parte do dispositivo é então processada e exibida ao utilizador a partir da tabela presente na interface. Caso a conexão não seja estabelecida com sucesso, o utilizador receberá uma mensagem de *feedback* da origem do problema.

Já na operação relativa ao botão "WALK", em vez de solicitar um único valor, o sistema percorre uma série de OIDs a partir de um ponto inicial, coletando todos os valores disponíveis de uma subárvore de dados no dispositivo. A Figura 4.23 expõe os parâmetros necessários para efetuar uma leitura "WALK".

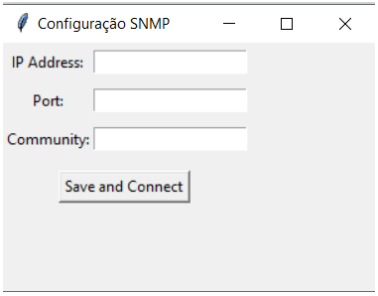


Figura 4.23: Parâmetros necessários para efetuar leitura WALK

Face a problemas que se encarou devido ao grande volume de dados que se tinham que ler com o comando "WALK"foi necessário a abordagem de *multithreading*. Isto porque, no desenvolvimento da aplicação, inicialmente os dados eram inseridos de forma semelhante às outras operações. Mas, com o grande volume de dados dos equipamentos SNMP da empresa, a aplicação demorava muito tempo a preencher a tabela pois esta só era preenchida quando todos os dados de uma subárvore fossem adquiridos. A partir da ferramenta MIB Browser verificou-se que as leituras de tais dados, chegavam a ser da ordem dos 1000, e, concluiu-se então que se tinha de solucionar este problema ao ir atualizando a tabela à medida que se recebiam os dados lidos. Como tal, se o dispositivo SNMP demorasse a responder ou se houvesse muitos OIDs, a restante aplicação ficava muito lenta, tornando uma experiência frustrante para o utilizador.

A partir do método de *multithreading*, essencialmente a leitura de dados SNMP é realizada numa *thread* paralela à *thread* da interface principal. Ou seja, enquanto a *thread* principal continua a funcionar normalmente, permitindo ao utilizador interagir com as restantes funcionalidades, sem interrupções, uma segunda *thread* é responsável pela tarefa que necessita de mais recursos: realizar a leitura de todos os dados associados a uma subárvore.

Esta abordagem, permite que à medida que os dados são recebidos em tempo real, eles são gradualmente inseridos na tabela da interface gráfica, criando uma experiência mais fluída e interativa, já que o utilizador é capaz de visualizar a tabela a ser preenchida conforme os dados chegam, ao invés de esperar que todo o conjunto de dados tenha sido processado. A partir desta abordagem, solucionou-se o problema de "congelamento"da aplicação, visto que *multithreading* permitiu que a interface gráfica continue responsiva, mesmo durante operações de leitura mais lentas devido ao maior volume de dados.

Uma outra funcionalidade criada no âmbito das leituras SNMP é a capacidade de calcular o volume de dados capazes de serem lidos num intervalo de tempo (*throughput*). Para tal, o sistema realiza duas leituras consecutivas dos contadores de octetos (dados transmitidos e recebidos) e, a partir das leituras, calcula a taxa de transmissão e receção de dados em *bits* por segundo. Esta métrica permite identificar interrupções no fluxo de dados.

4.4 Testes de interface

Durante este subcapítulo são apresentados os cenários de teste que permitiram validar a solução desenvolvida. Na execução destes cenários de teste, será possível analisar o comportamento do sistema, de forma a validar experimentalmente os resultados dos testes de continuidade realizados, comprovando a sua eficácia.

4.4.1 Cenários de teste

De forma a validar a solução desenvolvida durante esta dissertação são realizados testes relativos ao teste de continuidade e aos testes funcionais.

Em relação ao teste de continuidade, este foi executado na própria máquina da empresa, nas três diferentes zonas que se tem vindo a falar: "B1", "Bypass2"e "UPS-U2". O que se pretende alcançar com este teste é:

- Validar as conexões elétricas das três zonas da máquina, apresentando as tabelas expectáveis para cada zona em casos que se confirmem as corretas ligações da máquina. Para além disso, é apresentado uma análise comparativa do tempo necessário para testar cada zona da máquina recorrendo ao sistema apresentado nesta dissertação, comparando com o tempo necessário para testar as mesmas zonas da máquina, mas recorrendo ao multímetro, como tem sido realizado esta etapa por parte da equipa de produção da empresa.

- De forma a validar a funcionalidade de detetar fios sem continuidade e pares de fios trocados, recorreu-se a alterações propositadas nas ligações elétricas da máquina, abrindo o circuito num dos fios (forçando a não continuidade) e a troca propositada de 2 fios (forçando o par de pinos trocados). A intenção deste cenário de teste é avaliar o desempenho do sistema desenvolvido, em casos que a máquina não possui as ligações expectáveis, devido a erros humanos durante a fase de produção da mesma.

- O terceiro teste, baseia-se na manipulação do ficheiro "mapeamentos.txt"de forma a averiguar as suas capacidades de ajustar o funcionamento da GUI às possíveis alterações das ligações da máquina, uma vez que, consoante pedidos de clientes, algumas máquinas possuem ligações diferentes das *standard*. Assim, é possível

confirmar a versatilidade da solução desenvolvida, provando que esta apenas precisa de pequenos ajustes quando se sucedem alterações na máquina por parte da empresa.

Já em relação ao teste de funcionalidade, este vai ser avaliado recorrendo aos sensores presentes na máquina, verificando assim a receção dos dados em tempo real. Inicialmente são apresentados os parâmetros que se introduziu para cada sensor e posteriormente os resultados que se obtiveram.

4.4.2 Validação do teste de continuidade

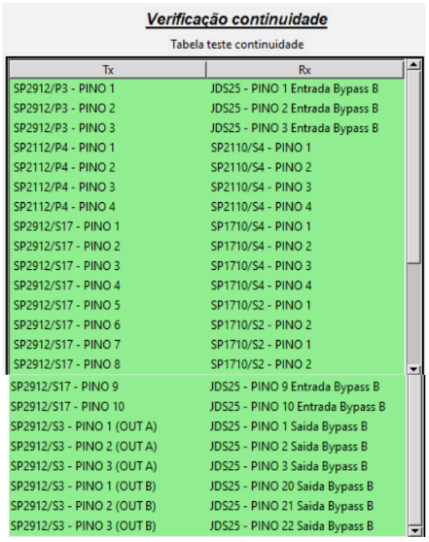
Relativamente ao primeiro cenário de teste, quando estes são realizados numa máquina que está com as ligações corretas, o aspeto visual que o utilizador terá acesso é ilustrado na Figura 4.24 para os testes nas zonas da máquina: 'B1', 'Bypass2' e 'UPS-U2', respetivamente. Tal como se tem vindo a mencionar, a cor verde transmite de forma imediata ao utilizador que as ligações existentes na máquina, no momento do teste, estão corretas.

Em relação à análise comparativa entre o uso do multímetro digital e o sistema desenvolvido nesta dissertação, realizaram-se os testes a: 'B1', 'Bypass2' e 'UPS-U2'. Os resultados, em segundos, obtidos a partir de uma média de medições, podem ser visualizados na Tabela 4.4.

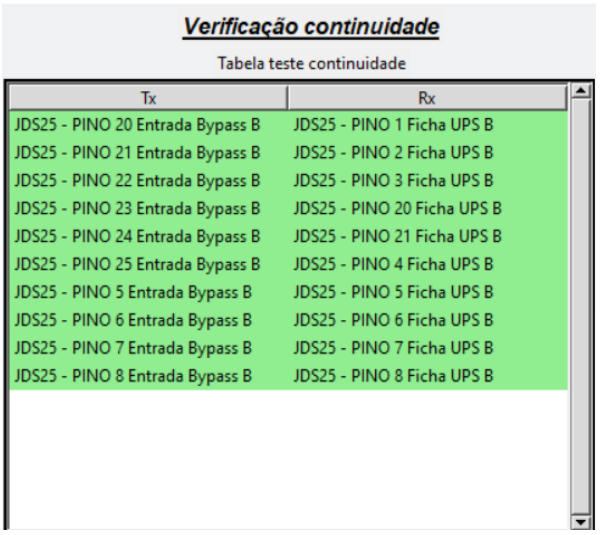
Tabela 4.4: Tempo demorado a realizar os mesmos testes, com ferramentas distintas

	B1	**Bypass2**	**UPS-U2**
Multímetro	127	47	41
Sistema Automatizado	24	13	13

De notar que o tempo de teste a partir do sistema automatizado, está incluído o valor de "*timeout*"equivalente a 8 segundos, anteriormente mencionado. Ou seja, na estratégia obtida, de forma a que a interface gráfica apresenta os resultados na tabela, esta espera 8 segundos até deixar de receber mais dados, assumindo assim que já não existem mais conexões que tenham sido verificadas. Caso se efetue esta sincronização a partir de outra estratégia, os testes realizados pelo sistema automatizado podiam ser reduzidos em pelo menos 8 segundos. Passando assim, a executar-se testes de continuidade em 16 segundos (24-8) para B1, e 5 (13-8) segundos para Bypass2 e UPS-U2. De notar que os tempos obtidos recorrendo ao multímetro digital advêm do teste ser realizado fio a fio e, também pelo facto das origens e destinos das ligações da máquina nem sempre estarem localizadas perto umas das outras.



(a) Teste continuidade lado B1



(b) Teste continuidade lado Bypass2

(c) Teste continuidade lado UPS-U2

Figura 4.24: Apresentação ao operador dos testes de continuidade

Em relação ao segundo cenário de teste, onde se alteraram as conexões elétricas da máquina, com a intenção de validar a deteção de erros na cablagem. Comparando a Figura 4.24a com a Figura 4.25, nota-se que houve a alteração das cores nas primeiras três linhas da tabela. Este resultado vai de encontro ao cenário de teste, uma vez que comprova a troca entre as ligações de dois fios. Visualizando a Tabela 4.3 era expectável que o operador tivesse realizado as ligações entre os conectores SP2912/S3 e JDS25 com os respetivos pinos coincidentes. Como tal, a cor bege nas linhas desta tabela, indicam ao operador que há a necessidade efetuar a troca desses dois fios. Já a cor vermelha, com o "X", indica que a ligação associada verificou a ausência de continuidade nesse fio.

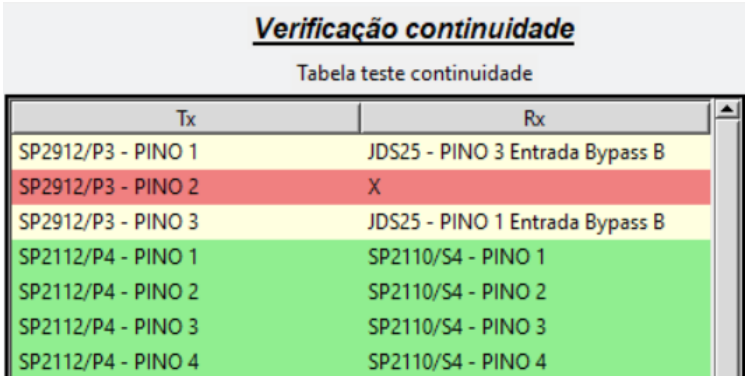


Figura 4.25: Teste de continuidade com circuito aberto num fio e troca de dois fios

Já no terceiro cenário de teste, onde se pretende averiguar as funcionalidades do ficheiro mapeamentos.txt apresenta-se a Figura 4.26 de forma a exemplificar como deve ser realizada a alteração dos mapeamentos no ficheiro .txt. Olhando para as quatro últimas linhas da tabela, verificam-se duas linhas a bege (que à primeira vista aparenta não haver razão, visto que a conexão está correta) e, nas últimas duas linhas observam-se a alteração do texto associado a este conector, onde se introduziu: "TESE SP1710/S24 - PINO 3"e "Mudanca eletrica". A razão para duas destas linhas estarem bege e as restantes a verde deve-se ao facto de como o utilizador altera o ficheiro .txt. Como se viu anteriormente, a utilidade deste ficheiro deve-se ao facto de se as ligações da máquina se forem alterando, e não houvesse a existência desta funcionalidade e deste ficheiro, este sistema ficava sem utilização, visto que não iria ser fiável para as novas ligações elétricas que a empresa poderia vir a implementar. No entanto, existe a necessidade de uma certa coerência de como se alteram estes valores no ficheiro. As linhas que ficaram a bege, que aparentam estar corretas, deve-se ao facto de, neste caso, terem sido efetuadas alterações no mapeamento da Figura 4.19b, sem se ter atualizado o mapeamento das conexões da Figura 4.19c em concordância. Ou seja, caso não se alterem os valores em ambos os mapeamentos, para a interface gráfica representam conexões diferentes, e daí a cor bege em conexões que aparentemente estão corretas.

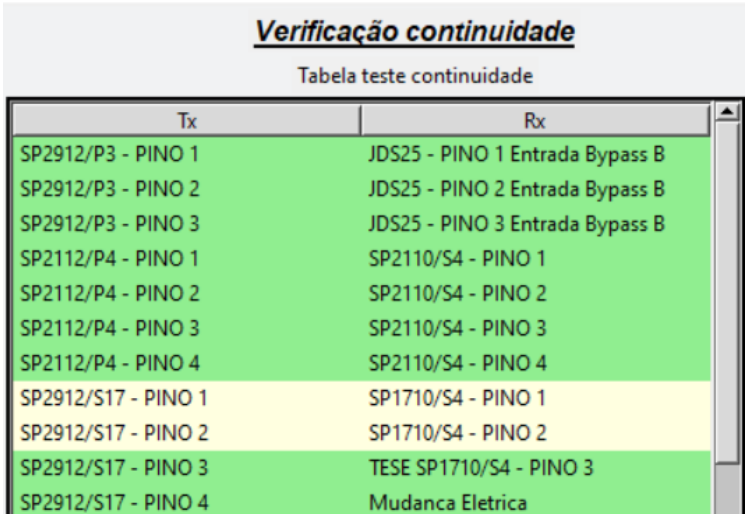
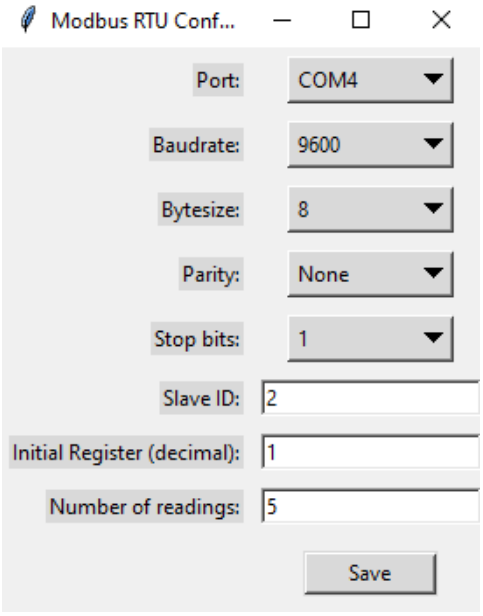


Figura 4.26: Visualização da tabela quando se realiza de forma errada alterações no ficheiro .txt

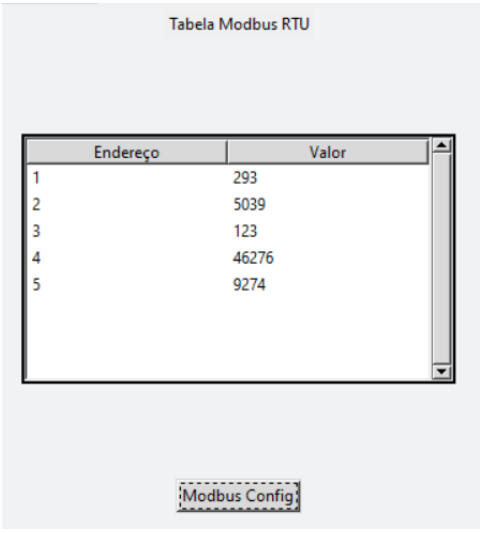
4.4.3 Validação dos testes funcionais

De forma a validar os testes funcionais, utilizaram-se sensores presentes na máquina. Inicialmente, introduziram-se os parâmetros relativos a cada sensor, e posteriormente verificou-se a receção de dados.

Começando pelo sensor baseado em Modbus RTU, inseriram-se os parâmetros presentes na Figura 4.27a e obtiveram-se os valores apresentados na Figura 4.27b.



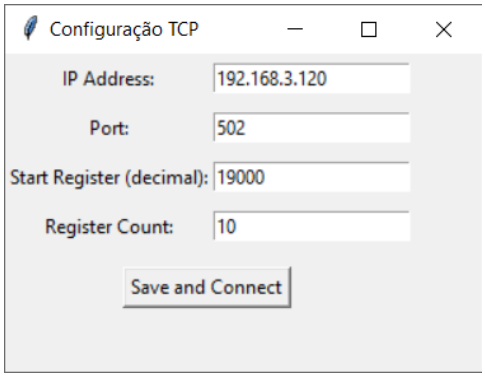
(a) Parâmetros introduzidos relativamente ao sensor testado



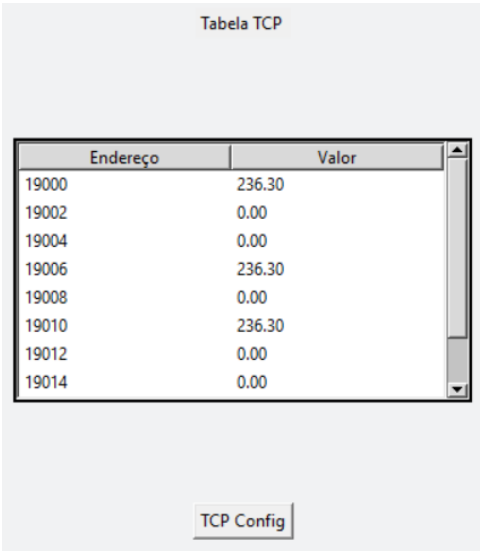
(b) Leituras obtidas do sensor

Figura 4.27: Teste de funcionalidade no sensor baseado em Modbus RTU

No sensor baseado em Modbus TCP introduziram-se os parâmetros expostos na Figura 4.28a e receberam-se as leituras ilustradas na Figura 4.28b.



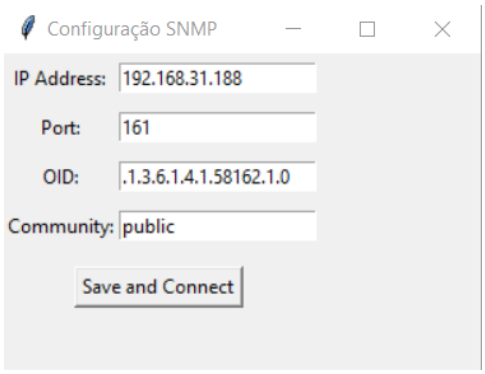
(a) Parâmetros introduzidos relativamente ao sensor testado



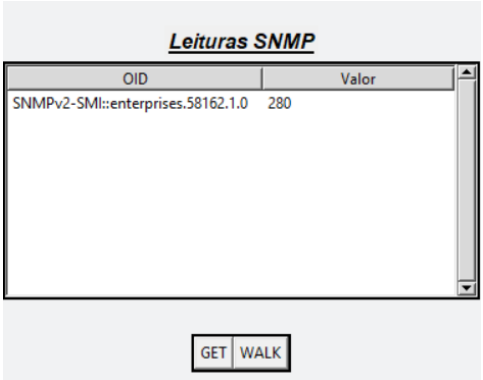
(b) Leituras obtidas do sensor

Figura 4.28: Teste de funcionalidade no sensor baseado em Modbus TCP

Já no sistema que recolhe dados SNMP, para o comando "GET", inseriram-se os parâmetros presentes na Figura 4.29a obtendo uma única leitura exposta na Figura 4.29b.



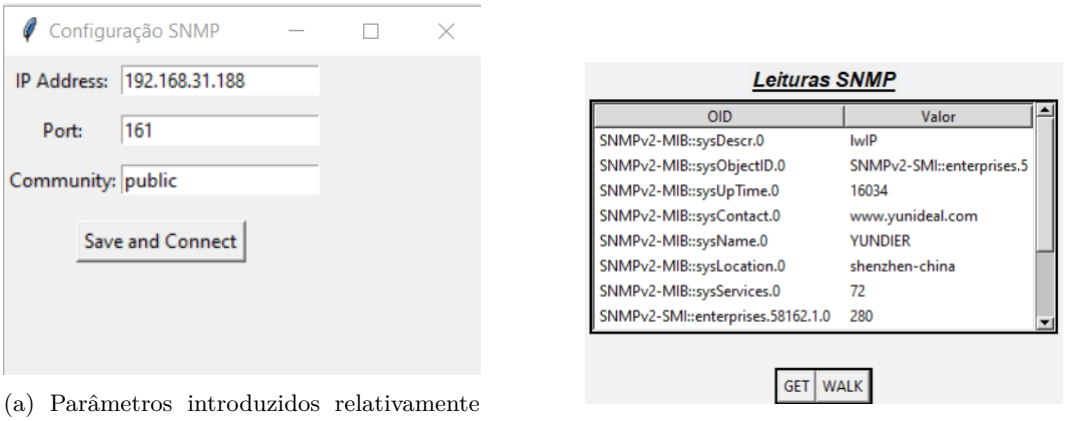
(a) Parâmetros introduzidos relativamente ao sensor testado



(b) Leituras obtidas do sensor

Figura 4.29: Teste de funcionalidade no sensor baseado em SNMP, comando "GET"

Por fim, relativamente ao sistema que coleta dados SNMP, para o comando "WALK", inseriram-se os parâmetros presentes na Figura 4.30a obtendo uma única leitura exposta na Figura 4.30b



(a) Parâmetros introduzidos relativamente ao sensor testado

(b) Leituras obtidas do sensor

Figura 4.30: Teste de funcionalidade no sensor baseado em SNMP, comando "WALK"

4.4.4 Discussão de resultados

Os resultados obtidos pelo sistema responsável por efetuar o teste de continuidade desenvolvido destacam que este foi capaz de responder de forma eficaz e precisa aos requisitos definidos no início do projeto. O sistema é capaz de realizar testes de continuidade de forma automatizada, com recurso a equipamento de baixo custo financeiro. É capaz de verificar corretamente a correspondência entre conexões elétricas esperadas e as conexões elétricas reais presentes nas diferentes zonas da máquina. A integração da interface gráfica é essencial uma vez que esta permite apresentar os resultados de teste de continuidade ao operador de forma clara e objetiva, tirando proveito de cores distintas para indicar o sucesso ou falha das conexões, facilitando a identificação de possíveis problemas, permitindo ao operador proceder a corrigir as conexões elétricas que se verificam erradas.

Para além disso, o sistema final mostrou versatilidade e flexibilidade devido à utilização do ficheiro mapeamentos.txt, uma vez que este permite o ajuste nos mapeamentos da máquina quando esta sofre alterações, sem a necessidade de alterações no código-fonte da interface gráfica, visto que este é um ficheiro externo que pode simplesmente ser carregado na aplicação, via *drag & drop*. Este fator contribui para a escalabilidade deste sistema, devido à facilidade de configurações para novas conexões elétricas da máquina.

Já na perspetiva dos testes funcionais, esta ferramenta demonstrou a sua eficiência na leitura e apresentação de dados provenientes de sensores baseados em Modbus ou SNMP. Esta funcionalidade permite assim visualizar valores em tempo real e com

o complemento do teste de continuidade, é possível identificar falhas em determinados sensores, caso estes não consigam estabelecer comunicação com a GUI, não enviando qualquer tipo de dados.

Capítulo 5

Conclusão

Esta dissertação promoveu o desenvolvimento de um sistema de monitorização e teste de continuidade, composto por duas principais vertentes: o sistema responsável pela realização dos testes de continuidade e a interface gráfica que permite a respetiva visualização dos resultados e também de efetuar a leitura de dados provenientes de sensores via Modbus e SNMP.

A primeira componente desta dissertação foi dedicada ao *design* e implementação do sistema responsável por efetuar os testes de continuidade nas diferentes zonas da máquina, possuindo como componentes integrantes deste sistema: microcontrolador, *multiplexers* e os cabos de teste desenvolvidos de forma a permitir executar os testes de continuidade. Este sistema desenvolvido com o intuito de testar de forma eficaz e automatizada diversas conexões elétricas, provou ser eficiente na identificação de falhas de continuidade nas distintas zonas da máquina. O uso de *multiplexers* e cabos de teste, permitiram testar múltiplas conexões elétricas em simultâneo, indo para além do que foi requisitado pela empresa, aumentando assim substancialmente a velocidade de cada teste e poupando tempo útil aos operadores. Este sistema foi sendo desenvolvido sempre com a integração da interface gráfica em mente, facilitando assim a integração do sistema responsável pelo teste de continuidade com a componente da interface gráfica. Proporcionou-se assim, uma interface *user-friendly*, onde, o utilizador ao definir o teste a realizar coloca os cabos de teste apropriados entre os terminais elétricos da máquina e os respetivos canais dos *multiplexers*, de forma a poder efetuar o teste de continuidade e visualizar os respetivos resultados de forma clara e fiável.

Já a segunda componente focou-se na interface gráfica, projetada para apresentar os resultados do teste de continuidade e também, realizar a leitura de dados de sensores a partir de protocolos como SNMP e Modbus, visto estes serem os protocolos usados pelos sensores integrantes da máquina. A interface foi desenvolvida com a intenção de ser responsiva e intuitiva, assegurando uma experiência agradável para o utilizador. A funcionalidade de *drag* & *drop* de ficheiros .txt, que têm no entanto no seu conteúdo dados no formato "json" provou ser uma grande mais valia devido à escalabilidade deste sistema, visto que, caso contrário, se houvessem alterações elétricas na máquina, este sistema ficaria obsoleto e sem qualquer propósito. Foi implementado *multithreading* para garantir uma fluidez na aplicação durante a leitura de grandes volumes de dados SNMP, o que evitou bloqueios e melhorou a performance global da aplicação. A integração com os protocolos SNMP e Modbus permitiu ampliar as funcionalidades da interface, que passou a oferecer não apenas efetuar testes de continuidade, mas também a leitura de dados em tempo real dos sensores, permitindo a validação do correto funcionamento dos mesmos.

5.1 Trabalho Futuro

Este projeto é flexível e facilmente escalável para diversas funcionalidades, no entanto, sugere-se o seguinte trabalho futuro:

- Cada zona da máquina a ser testada, tem um cabo de testes associado. Como trabalho futuro, finalizar o desenvolvimento dos cabos de teste para as restantes zonas da máquina, respeitando o mapeamento atual, de forma a permitir realizar os testes de continuidade em todas as zonas da máquina.
- Desenvolvimento de uma funcionalidade capaz de armazenar em memória o resultado de todos os testes de continuidade, de forma a, posteriormente, a partir do clique de um botão, reunir os testes de continuidade e criar um ficheiro de pdf. Ao ser possível imprimir o ficheiro, seria fornecido ao cliente com a intenção de garantir que os testes de continuidade foram realizados. Ou seja, um comprovativo de verificação das ligações elétricas.
- Tornar a interface gráfica com uma estética e *layout* mais agradável.
- À semelhança dos equipamentos "*ethernet cable tester*"referidos no capítulo 2, é uma mais valia integrar no sistema de teste continuidade esta funcionalidade de determinar a que distância o cabo tem a sua continuidade elétrica interrompida.
- Caso a empresa introduzisse uma "caixa de testes" no interior da máquina, que reunisse todas as ligações elétricas, em conjunto com um sistema semelhante ao desenvolvido nesta dissertação, e, por exemplo, a partir de um ESP permitiria uma monitorização constante da qualidade da cablagem. A partir das funcionalidades

do ESP, facilmente este sistema forneceria *feedback*, alocado num *website*. Ao monitorizar este *feedback*, caso se verificasse o desgaste da qualidade da cablagem nas máquinas dos clientes, facilmente se realizaria a manutenção nos equipamentos do cliente, antes da máquina ficar com interrupções de funcionamento devido à camada física, e assim prevenia o problema antes de acontecer.

Referências

[1] P. J. Bryant and T. J. Tillman, "Automated continuity tester for large wire-wrapped avionics chassis," in *2012 IEEE AUTOTESTCON Proceedings*, pp. 99–104, IEEE, 2012. [Citado nas páginas 7 e 8]

[2] C. ODURUKWE, *DESIGN AND CONSTRUCTION OF CONTINUITY TESTER AND CUT POINT DETECTOR.* PhD thesis, 2006. [Citado nas páginas vii, 7 e 9]

[3] W. S. Ahmad, S. Perinpanayagam, I. Jennions, and S. Khan, "Study on intermittent faults and electrical continuity," *Procedia Cirp*, vol. 22, pp. 71–75, 2014. [Citado nas páginas 7 e 17]

[4] J. Kustija, I. Purnama, I. Surya, D. Fahrizal, *et al.*, "Wireharness continuity test equipment design microcontroller-based aircraft module and atmega328p nrf24l01+ wireless," *International Journal of Science and Technology Research Archive*, vol. 4, no. 2, pp. 001–011, 2023. [Citado nas páginas 8, 10 e 13]

[5] P. Roy, S. Karmakar, P. Basak, S. Das, and S. Khatua, "Designing of a device for checking the polarity and continuity of any electrical and electronics circuit," *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 7, pp. 170–172, 2016. [Citado na página 8]

[6] H. Umadevi, B. Niketh, and V. T. Vijaydeep, "Underground cable fault monitoring & detection system using iot & arduino," *Journal for research*, 2018. [Citado nas páginas 8 e 16]

[7] N. Pawar, R. Nirhali, S. Kolhe, S. Darade, and R. Patil, "Design of arduino based underground cable fault detector," *Earth*, vol. 6, no. 04, 2019. [Citado na página 10]

[8] Z. Bi, C. Pomalaza-Ráez, D. Hershberger, J. Dawson, A. Lehman, J. Yurek, and J. Ball, "Automation of electrical cable harnesses testing," *Robotics*, vol. 7, no. 1, p. 1, 2017. [Citado nas páginas 10, 11 e 13]

[9] Z. Bi, C. Pomalaza-Ráez, A. Lehman, J. Dawson, D. Hershberger, J. Yurek, and J. Ball, "Automated testing of electrical cable harnesses," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2704–2709, IEEE, 2018. [Citado na página 10]

[10] H. Wang, O. Salunkhe, W. Quadrini, D. Lämkull, F. Ore, M. Despeisse, L. Fumagalli, J. Stahre, and B. Johansson, "A systematic literature review of computer vision applications in robotized wire harness assembly," *Advanced Engineering Informatics*, vol. 62, p. 102596, 2024. [Citado nas páginas vii, 10, 18 e 19]

[11] K. K. Asparuhova, S. Asenov, A. Chekichev, and D. A. Shehova, "Implementation of harness testing device using microcontroller," in *2022 XXXI International Scientific Conference Electronics (ET)*, pp. 1–5, IEEE, 2022. [Citado na página 11]

[12] H. Wang, O. Salunkhe, W. Quadrini, D. Lämkull, F. Ore, B. Johansson, and J. Stahre, "Overview of computer vision techniques in robotized wire harness assembly: Current state and future opportunities," *Procedia CIRP*, vol. 120, pp. 1071–1076, 2023. [Citado nas páginas vii, 11 e 19]

[13] A. Kunaraj, J. Joy Mathavan, and K. Jayasekara, "Portable multifunction tester design to check the continuity of wires and to measure the electrical parameters," in *Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020*, pp. 863–878, Springer, 2021. [Citado na página 11]

[14] R. Kakkeri, L. Inamdar, S. Bhambare, and A. Gund, "Distributed cable harness tester," *International Research Journal of Engineering and Technology (IRJET),(04)*, vol. 5, pp. 2278–2281, 2017. [Citado na página 12]

[15] S. Prince, P. Anand, J. Battat, R. Farnsworth, N. Felt, R. Guenette, S. Kubota, A. Li, E. Murdock, J. Oliver, *et al.*, "Digital wire analyzer of mechanical tension, electrical continuity, and isolation," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022. [Citado nas páginas 14 e 15]

[16] R. Stephenson and J. Bateman, "A non-contact method for measuring mwpc wire tensions," *Nuclear Instruments and Methods*, vol. 171, no. 2, pp. 337–338, 1980. [Citado na página 14]

[17] R. Acciarri, C. Adams, J. Asaadi, J. Danaher, B. Fleming, R. Gardner, S. Gollapinni, R. Grosso, R. Guenette, B. Littlejohn, *et al.*, "Construction and assembly of the wire planes for the microboone time projection chamber," *Journal of Instrumentation*, vol. 12, no. 03, p. T03003, 2017. [Citado na página 14]

[18] C. Bacci, C. Avanzini, F. Lacava, and D. Picca, "A method for measuring the wire tension in drift chambers using neodymium permanent magnets," in *1995 IEEE Nuclear Science Symposium and Medical Imaging Conference Record*, vol. 1, pp. 469–473, IEEE, 1995. [Citado na página 14]

[19] D. Carlsmith and S. Lusin, "Drift tube wire tension monitoring," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 364, no. 1, pp. 79–89, 1995. [Citado na página 14]

[20] P. Ciambrone, E. Dane, R. Dumps, M. Dwuznik, G. Felici, C. Forti, A. Frenkel, J.-S. Graulich, A. Kachtchouk, V. Kulikov, *et al.*, "Automated wire tension measurement system for lhcb muon chambers," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 545, no. 1-2, pp. 156–163, 2005. [Citado na página 14]

[21] F. Auzanneau, "Wire troubleshooting and diagnosis: Review and perspectives," *Progress In Electromagnetics Research B*, vol. 49, pp. 253–279, 2013. [Citado na página 16]

[22] C. M. Furse, M. Kafal, R. Razzaghi, and Y.-J. Shin, "Fault diagnosis for electrical systems and power networks: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 888–906, 2020. [Citado na página 16]

[23] J. D. Sloan, *Network Troubleshooting Tools: Help for Network Administrators*. "O'Reilly Media, Inc.", 2001. [Citado na página 18]

[24] X. Jiang, Y. Nagaoka, K. Ishii, S. Abiko, T. Tsujita, and M. Uchiyama, "Robotized recognition of a wire harness utilizing tracing operation," *Robotics and Computer-Integrated Manufacturing*, vol. 34, pp. 52–61, 2015. [Citado na página 19]

[25] X. Jiang, K.-m. Koo, K. Kikuchi, A. Konno, and M. Uchiyama, "Robotized assembly of a wire harness in a car production line," *Advanced Robotics*, vol. 25, no. 3-4, pp. 473–489, 2011. [Citado na página 19]

[26] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 2008. [Citado nas páginas vii, 21, 22, 23, 24, 26 e 27]

[27] G. Clarke, D. Reynders, and E. Wright, *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004. [Citado nas páginas ix, 21, 22, 23 e 24]

[28] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and implementation of a secure modbus protocol," in *Critical Infrastructure Protection III: Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Hanover, New Hampshire, USA, March 23-25, 2009, Revised Selected Papers 3*, pp. 83–96, Springer, 2009. [Citado nas páginas 21 e 22]

[29] H. Herath, S. Ariyathunge, and H. Priyankara, "Development of a data acquisition and monitoring system based on modbus rtu communication protocol," *Development*, vol. 5, no. 6, pp. 433–440, 2020. [Citado nas páginas vii, 22, 24 e 25]

[30] *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b*. [Citado na página 23]

[31] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: secure protocols, incidents, threats and tactics," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020. [Citado na página 27]

[32] G. Mageshkumar, N. Kasthuri, K. Tamilselvan, S. Suthagar, and A. Sharmila, "Design of industrial data monitoring device using iot through modbus protocol," *Int. J. Sci. Technol. Res*, vol. 9, pp. 1392–1396, 2020. [Citado nas páginas vii, 27 e 28]

[33] D. Mauro and K. Schmidt, *Essential SNMP: Help for System and Network Administrators*. "O'Reilly Media, Inc.", 2005. [Citado nas páginas vii, 28, 29, 30, 31 e 32]

[34] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol (snmp)," tech. rep., 1989. [Citado nas páginas 28 e 29]

[35] M. Technology, "Atmega328p datasheet." [Citado nas páginas vii, 37 e 38]

[36] A. Khandelwal and H. Sharma, "Physical system analysis using matlab," *vol*, vol. 4, p. 4, 2017. [Citado na página 41]

[37] Choosing the Best Python GUI Libraries: An Essential Guide for Developers. [Citado nas páginas 43 e 44]

[38] Top 10 Python GUI Frameworks for Developers. [Citado na página 44]

[39] J. Chen and S. Huang, "Analysis and comparison of uart, spi and i2c," in *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, pp. 272–276, IEEE, 2023. [Citado na página 46]