INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Sistema de Rastreabilidade em Ambiente Industrial

Nuno Guedes - 27112

Trabalho realizado sob a orientação de

Prof. Dr. Paulo Leitão

Prof. Dr. José Lima

Mestrado em Engenharia Industrial, ramo Eletrotécnica

2019-2020

Sistema de Rastreabilidade em Ambiente Industrial

Dissertação do Curso de Mestrado em Engenharia Industrial - Ramo Eletrotécnica

Escola Superior de Tecnologia e Gestão

Nuno Guedes - 27112

2019-2020

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Declaro que o trabalho descrito neste relatório é da minha autoria e é da minha vontade que o mesmo seja submetido a avaliação.

Nuno Guedes - 27112

Dedicatória

Dedico este trabalho à minha família, que sempre me apoiou incondicionalmente para desenvolver o trabalho ao meu melhor nível, e em especial à minha esposa e filha que criaram em mim uma vontade de aprender mais e de vencer qualquer tipo de desafio.

Agradecimentos

Estou muito agradecido devido ao facto de, em todo este percurso, ter conseguido adquirir novos conhecimentos e amizades que serão muito valorizados. Em especial, um agradecimento aos professores orientadores Paulo Leitão e José Lima, que sempre se mostraram disponíveis para esclarecimento de dúvidas e para enriquecer o trabalho desenvolvido. Não esquecendo os colegas, que sempre se mostraram disponíveis para contribuir para a superação deste desafio, um obrigado e votos de muito sucesso em todo o seu percurso de vida.

Resumo

A 4ª revolução industrial procura satisfazer o objetivo de inserir avanços tecnológicos no modelo de produção, flexível em termos de produtos e serviços produzidos digitalmente, e que combinam a comunicação em tempo real entre todas as partes e instalações que se relacionam durante o processo de fabrico e inspeção. Com os desenvolvimentos recentes na tecnologia e nos sistemas de comunicação, existe uma maior disponibilidade e acessibilidade por parte de sensores mais capacitados, sistemas de aquisição de dados e redes de computadores, e a natureza competitiva da atual indústria procura cada vez mais as fábricas-inteligentes através da implementação de novas tecnologias.

Esta dissertação descreve um processo de interligação de informações geradas através de máquinas, enquadradas num processo de inspeção industrial robotizado, a fim de criar uma aplicação móvel de rastreabilidade dos produtos. No ambiente industrial, os dispositivos que geram informação são um controlador lógico programável (PLC) interligado com uma câmara de processamento de imagem para obter e registar informações contidas no QR-CODE de cada produto, um robô colaborativo (UR3) com uma garra de pressão e uma câmara de processamento acopladas que realizam os testes aos botões dos produtos e que detetam a quantidade de erros nos LCD's que estes possam possuir. Acessível a qualquer utilizador, foi desenvolvida uma aplicação num computador industrial para rastrear características de cada produto inspecionado, tais como o número de série, datas de inspeção, tempos médios de processamento, entre outras. Todos os dispositivos foram ligados numa mesma rede, onde se encontrava um servidor capaz de armazenar toda a informação do processo e enviá-la para a plataforma elaborada pelo autor, tornando assim possível visualizar de forma organizada e estruturada os dados relativos a qualquer

objeto inspecionado na bancada industrial. Fazem parte do conceito de "Indústria 4.0" o processamento de informações (BIG-DATA) geradas por diferentes dispositivos capazes de se comunicar entre si e presentes em cloud, acessíveis a partir de um sítio na internet, o robô colaborativo e autónomo aquando da deteção de um objeto específico e, ainda, garantir a segurança no transporte das informações. São abordados diferentes tipos de linguagem de programação para os diferentes softwares utilizados, como a linguagem de consulta estruturada, Structured Query Language (SQL) para a interface do sistema de gestão de base de dados MySQL, Ladder diagram (LD), blocos funcionais, Function Block Diagram (FBD) para comandar o autómato, Java Script para programar a ferramenta visual baseada em fluxos Node-RED e também código Phyton relativo à programação do processo de inspeção de um robô. O O programa Node-RED consiste numa interface de programação de aplicações, Application Programming Interface (API) capaz de aceder a qualquer informação registada na base de dados acerca do processo de inspeção industrial, em tempo real, permite formatações de dados passados de forma rápida e acessível a todos, através de gráficos, imagens e texto relativos ao processo decorrido.

Palavras-chave: Rastreabilidade, Industria 4.0, Aplicação em WEB.

Abstract

The 4th Industrial Revolution aims to achieve the goal of implementing the newest advancements into the production model, which is flexible in terms of products and services that are digitally produced, and combine communication in real time among all parts and installations involved in the process, creating relations that are active during the process of production and inspection. With the recent and newest advancements in technology and communication systems, there is an increasing availability and accessibility in terms of more skilled sensors, data acquisition systems and computer networks, and therefore the competitive nature of Industry, nowadays, demands an increasing search for intelligent factories through the implementation of new technologies.

The present dissertation describes a process of links and interconnection of data generated through machines that develop a robotic inspection process, aiming to create a mobile WEB application to ensure the traceability of products. In the industrial environment, the devices responsible for generating data are a programmable logic controller (PLC) connected to an image-processing camera in order to obtain and register information present in the QR-CODE of each product; a collaborative robot (UR3) equipped with a pressure claw and an attached processing camera, that performs the tests to the buttons in the products, and detects the amount of errors in the LCD's that they may display.

Accessible to any user, a WEB application was developed in an industrial computer in order to trace the characteristics of any product inspected, such as the serial number, the dates of inspection, the average times of processing, among other factors. By using a local network (similar to the Internet of Services) all the devices were inter-connected,

as well as a server with the ability to store all the data in the process, and afterwards send it to the platform previously developed by the author. This way, it is possible to visualize in an organized and structured way the data related to any inspected object in the industrial workbench.

As part of the concept of "Industry 4.0", we have the processing of data (BIG-DATA) generated by different devices that are able to communicate among themselves, which are located in clouds accessible by an Internet website, the collaborative and autonomous robot when in the detection of a specific object and, finally, to ensure the safety in the transportation of data.

Throughout the present dissertation, different types of programming languages are approached in relation to the various softwares, such as the Structured Query Language (SQL) for the interface of the data base management system MySQL, the Ladder Diagram (LD) and function blocks to command the automaton. The programming language Python is also approached in order to program the visual tool, which is based in Node-RED fluxes.

This visual tool consists of a Programmable Interface Application (API) which is able to access any data that is registered in the process of industrial inspection, in real time, and that allows the formatting of data that was transported and passed in a fast and accessible way to everybody, by using graphics and text information.

Keywords: Traceability, Industry 4.0, WEB Application.

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

AI Artificial intelligence. 2

API Application Programming Interface. x

FBD Function Block Diagram. x, 16

GUI Graphical User Interface. 13

HMI Human-machine interface. 5, 12, 20, 21

IBM International Business Machines. 15

ID Identifier. 9

IOT Internet of Things. 3

LD Ladder diagram. x, 16

MySQL My Structured Query Language. 5

PLC Programmable logic controller. 5, 11, 16

SQL Structured Query Language. x

ST Structured text. 16

TI Tecnologias da Informação. 10

UR3 Universal Robots. 5

Capítulo 1

Introdução

A Indústria 4.0 exige que cada componente seja individualmente identificado e localizado dentro da cadeia de abastecimento. Informações sobre origem, armazenamento, estado e localização de materiais, componentes e produtos devem estar disponíveis em todo o tempo. Providenciando dados históricos ou o estado atual de qualquer objeto ou parte que o constitui, engloba ferramentas básicas para análise operacional, sugerindo caminhos e processos alternativos e mais eficientes de produção. Este trabalho tem a tarefa de conseguir centralizar todas as informações relacionadas com um processo de controlo de qualidade em displays, usando diferentes tecnologias que constituem a estação de inspeção robotizada e automatizada. Usando diferentes tipos de programação em diferentes plataformas, torna as máquinas capazes de se conectar numa mesma rede privada para apresentar o processo de controlo decorrido, através de aplicação móvel e por meio de gráficos e informações estatísticas [1].

1.1 Enquadramento

Um dos grandes desafios das indústrias em geral é a rastreabilidade dos seus produtos. Tendo por base este conceito, e utilizando um conjunto de práticas de identificação e monotorização de produtos através de etiquetas com códigos de barras e câmaras inteligentes conectadas a controladores lógicos programáveis, bem como computadores capazes

de guardar toda a informação do produto, foi elaborada uma interface gráfica para aceder e dispor dessa informação de forma intuitiva. Sistemas de rastreabilidade são de elevada utilidade em inúmeros serviços, como no caso da indústria farmacêutica, na qual já podem ser consultados dados disponibilizados pela consultoria tecnológica Accenture, em parceria com a DHL, entidade responsável pelo transporte, relativos ao desenvolvimento de um protótipo de rastreio de produtos farmacêuticos desde o seu ponto de origem até ao consumidor, para evitar adulterações e erros. Ou seja, quanto melhor for o processo de rastreabilidade mais confiável será o produto. Com o avanço da automação industrial, o conceito de Indústria 4.0, ou *"smart-factories"*, combina o uso de tecnologias emergentes, como a internet das coisas (Internet of Things - IoT), dados (BigData), Artificial intelligence (AI) e Computação na nuvem (Cloud computing), originando sistemas de informação cada vez mais minuciosos sobre todo o processo de fabrico, desde o chão de fábrica até aos sistemas superiores de gestão e controlo de qualidade [2].

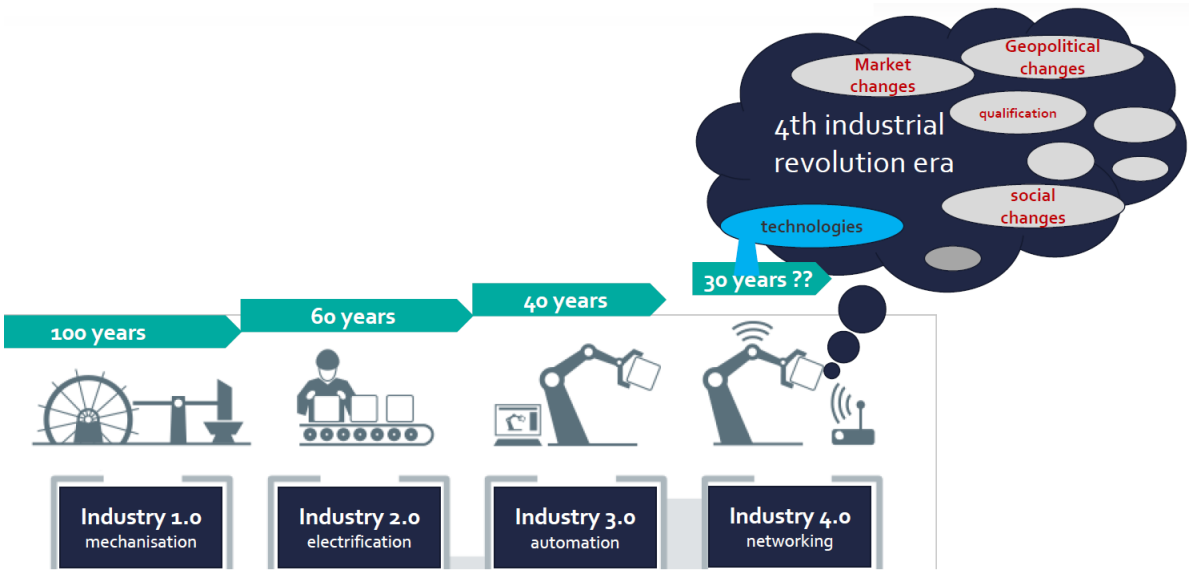

Figura 1.1: Evolução Histórica Fonte: [3]

Os avanços tecnológicos têm impulsionado um aumento dramático na produtividade desde o início da Revolução Industrial. Atualmente, com os avanços na área da internet das coisas, comunicações móveis e comércio via internet (e-commerce), está a começar

uma quarta onda de avanço tecnológico: a ascensão da nova tecnologia industrial digital, conhecida como Indústria 4.0. É uma transformação que é alimentada por alguma tecnologia aplicada a sensores, máquinas, peças e sistemas da Internet of Things (IOT) conectados ao longo da cadeia de produção de toda a fábrica. Com a evolução das tecnologias industriais e dos seus requisitos para a recolha eficiente e divulgação de dados de produção, a rastreabilidade torna-se um elemento essencial do processo de fabrico, podendo ser implementada em qualquer nível da cadeia de produção [4]. Estes sistemas conectados podem interagir uns com os outros utilizando protocolos padrão baseados na Internet, e analisar dados gravados. O conceito de Indústria 4.0 tornará possível reunir e analisar dados através de máquinas permitindo, de forma mais rápida e através de processos mais flexíveis e eficientes, produzir bens de alta qualidade a custos reduzidos. São apresentados alguns dos avanços na tecnologia necessários à transição para a Indústria 4.0, que irão transformar a produção, tais como: isolamento de processos, uso de células otimizadas e automatizadas, e otimização do fluxo de produção para aumentar a eficiência e mudar as relações de produção tradicionais entre fornecedores, produtores e clientes, bem como na relação homem e máquina [5]. Os sistemas de rastreabilidade asseguram o controlo completo sobre a complexidade da informação e sobre os dados dentro de uma operação, independentemente do seu volume. A rastreabilidade de peças individuais em todo o processo de fabrico, ao longo da cadeia de fornecimento e na sua distribuição no mercado, no qual são aplicadas apresentam ferramentas fundamentais para o funcionamento de fábricas altamente automatizadas e inteligentes (as smart-factories) [6]. Rastreabilidade é também um elemento-chave para a melhoria contínua [5].

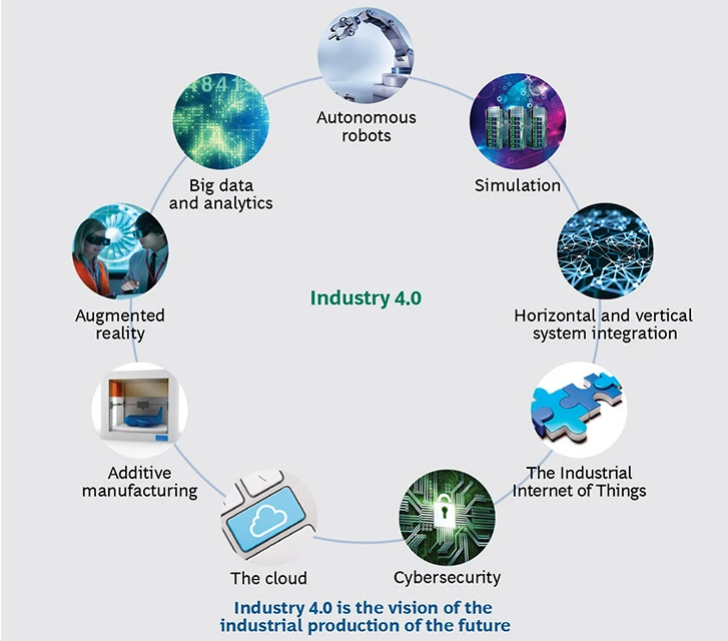

Figura 1.2: Pilares da Industria 4.0 Fonte: [7]

1.2 Objetivos

No processo de inspeção de displays do presente estudo de caso, pretende-se aglutinar todo o tipo de informações relativas a cada peça, número de série, qual o operador que autorizou o processo, tempos de inspeções, quantidades de objetos, entre outras características, para poderem ser rapidamente consultadas por um gestor de serviços/produção.

Com a informação toda reunida numa base de dados, é feito um sistema capaz de fazer o rastreio a qualquer objeto inspecionado, bem como a um conjunto deles. A interface desenvolvida disponibiliza ao utilizador estas informações realizando pesquisas, através do número de série ou do tipo de cada objeto e, ainda, num determinado período de tempo. O mecanismo de identificação de cada peça a ser inspecionada funciona através da validação por código de barras bidimensional (código QR), o qual permite armazenar uma grande quantidade de informação, e que é lido através de uma câmara da Teledyne Dalsa, acoplada a um autómato munido de comunicação ModBus, por forma a enviar mensagens a uma base de dados. Constitui-se desta forma um sub-sistema de gestão de informação com o

Programmable logic controller (PLC) (Programmable Logic Controller) da Omron NJ101-9020, para receber os dados obtidos pela câmara, e acionar as saídas e entradas analógicas e digitais a fim de controlar outras variáveis do processo industrial. O autómato também controla a interface homem máquina Human-machine interface (HMI) da Omron model NA5-9W001B, touch screen, que foi utilizada para introdução de informação relacionada com o produto, e para validar qualidades do mesmo.

É através de um sistema de gestão de base de dados, o My Structured Query Language (MySQL), que recebe informação do autómato e a deixa disponível para futuros processamentos. Na procura por um recurso visual, explorou-se e implementou-se uma interface gráfica através do Node-RED, uma ferramenta de ambiente de código aberto voltada para a criação de mashups para IoT. É através deste programa, facilmente instalado em qualquer sistema operativo ou microcontrolador, como o RaspberryPi, que são elaborados gráficos e janelas capazes de mostrar todos os dados dos processos selecionados e acompanhar todo o sistema em tempo real. Assim, o projeto mostra desde o desenvolvimento à implementação, um robusto sistema de rastreabilidade que posteriormente poderá ser incorporado a outras bancadas de produção, possibilitando uma linha de produção monitorizada. O objetivo é o de integrar tecnologias presentes numa bancada de inspeção, sendo elas um robô colaborativo industrial da Universal Robots (UR3), duas câmaras de processamento de imagem e um autómato com a respetiva consola HMI, conectados a um computador industrial capaz de disponibilizar em tempo real a informação que por eles seja gerada. Em processos industriais como o do presente estudo de caso, a realização de inspeções, a verificação de códigos e a inspeção feita pelo robô UR3 proporciona enormes vantagens competitivas, quando aglomeradas todas estas informações na nuvem ou num servidor local relativo ao processo. É um sistema composto por vários subsistemas que se encontram totalmente interligados através de um protocolo de comunicação, muito utilizado em ambiente industrial e do conhecimento de todas as máquinas que constituem a bancada, e que se denomina por ModBus. A adoção de rastreabilidade torna mais rápido isolar um problema ocorrido, permitindo reduzir custos no fabrico de peças com defeitos e possibilitando o reajuste de forma mais eficiente para qualquer bancada de inspeção, em

relação a trabalhos futuros.

1.3 Estrutura do documento

Os capítulos que seguem estão estruturados da seguinte forma: Capítulo 2: Apresenta a contextualização do trabalho, mostrando o cenário atual na rastreabilidade na indústria e uma introdução geral às tecnologias utilizadas. Capítulo 3: Apresenta as ferramentas utilizadas no trabalho, incluindo dificuldades encontradas durante o projeto e as formas encontradas para as contornar. Capítulo 4: Apresenta as funcionalidades implementadas em todos os softwares utilizados. Capítulo 5: Apresenta os testes realizados para verificar que o projeto desenvolvido cumpre os objetivos assumidos e resolve, de facto, o problema descrito na Análise/Modelação. Capítulo 6: Apresenta a conclusão, a sintetização e proporciona uma perspetiva unificadora do trabalho efetuado, bem como várias propostas de trabalhos futuros.

Capítulo 2

Motivação

Este trabalho assenta no desenvolvimento de uma aplicação compatível com qualquer navegador, inclusive em dispositivos móveis, na qual toda a informação sobre um processo de inspeção em displays está disponível, oferecendo uma visão detalhada acerca do histórico de dados à cerca dos produtos inspecionados pela bancada industrial. Ainda neste capítulo é feita uma abordagem do atual estado de alguns sistemas de rastreabilidade, com especial atenção para as suas funcionalidades, tecnologias existentes e alguns dos problemas atuais. Por fim é descrita uma lista de programas essenciais para a recolha, a gravação e a exposição das informações que fazem parte da aplicação móvel [1].

2.1 Rastreabilidade

Com o desenrolar da Indústria 4.0, as novas visões acerca das ferramentas digitais, da Internet das Coisas, da manipulação de grande quantidade de dados e da possibilidades de peças e matérias-primas serem rastreadas sofrem um considerável avanço. A adoção antecipada é agora um passo essencial no caminho para a plena implementação da Indústria 4.0. Matérias-primas, peças e práticas de rastreabilidade dos produtos acabados têm sido, tradicionalmente, alvo de operações altamente reguladas e que são legalmente obrigatórias, por forma a demonstrar a conformidade, como é o caso de produtos farmacêuticos e na indústria aeroespacial e automóvel. Cresce assim uma base industrial muito

maior com a promessa da Indústria 4.0, e o impacto que esta virá a ter na economia de fabrico e logística. Questões de rastreabilidade assumem cada vez mais uma maior importância para as empresas que operam em todas as fases da cadeia de abastecimento [8].

A rastreabilidade tem sido sempre um aspeto vital na indústria, desde a produção de produtos farmacêuticos e o seu fornecimento, até aos setores como o aeroespacial e o automóvel, dado que acarretam um risco real para o consumidor final. Além desse factor, também a reputação do fabricante é averiguada, devendo cada produto ser devidamente classificado durante o processo de fabrico. É o caso da indústria farmacêutica, na qual já podem ser consultados dados disponibilizados pela consultoria tecnológica Accenuture em pareceria com a DHL, entidade responsável pelo transporte, relativos ao desenvolvimento de um protótipo de rastreio de produtos farmacêuticos desde o seu ponto de origem até ao consumidor, para evitar adulterações e erros [9].

Na aeronáutica, como explicou Paul Stein, Diretor do Departamento Científico da Rolls Royce à revista Forbes, "estão no ar 500 motores Rolls Royce em aviões de transporte civil e 150 em aviões das forças armadas, todos eles monitorizados pelo seu sistema de gestão de vida do motor. Com motores aeronáuticos civis tão confiáveis, a ênfase muda para os manter em funcionando ao máximo, economizando combustível às companhias aéreas e cumprindo os seus horários. A análise de dados ajuda a Rolls-Royce a identificar ações de manutenção dias ou semanas antes do tempo, para que as companhias aéreas possam agendar o trabalho sem que os passageiros sofram qualquer interrupção"[10].

Na industria automóvel, como acontece na marca de automóveis elétricos Tesla, é possível personalizar um automóvel por cliente, o que implica ter um elevado nível de controlo em todo o tipo de componentes, como datas de produção e de entregas. Ou seja, quanto melhor for o processo de rastreabilidade mais confiável será o produto.

No entanto, a Indústria 4.0 define uma prioridade maior para a rastreabilidade entre os fabricantes tradicionais; metodologias de produção eficientes exigem uma cadeia ininterrupta, totalmente conectada desde o abastecimento, ao longo do qual os participantes são capazes de controlar não apenas o movimento de peças e materiais em tempo real,

mas também a história do seu fabrico. A rastreabilidade de peças individuais ao longo do processo de fabrico de toda a cadeia de fornecimento, e a sua distribuição no mercado, são fundamentais para o funcionamento de fábricas automatizadas e 'inteligentes' [11].

O sistema de produção em fábrica 'inteligente' requer Tecnologias da Informação e da Comunicação (ICT, do inglês Information and Communication Technologies) baseadas na integração dos sistemas de produção do chão de fábrica, e o planeamento dos recursos empresariais. Deste modo, ambientes de produção sem papel onde as intervenções humanas são raras ou inexistentes, bem como rastreabilidade, serão fundamentais para as operações.

Uma elevada qualidade em codificação legível, e uma eficiência na captura de dados são elementos essenciais para um robusto sistema de rastreabilidade. O processo começa com um dispositivo de marcação ao qual se aplica uma marca de identificação única e permanente Identifier (ID), a qual pode ser, por exemplo, um código de barras bidimensional - código QR lido através de câmaras tecnológicas de processamento de imagem [11].

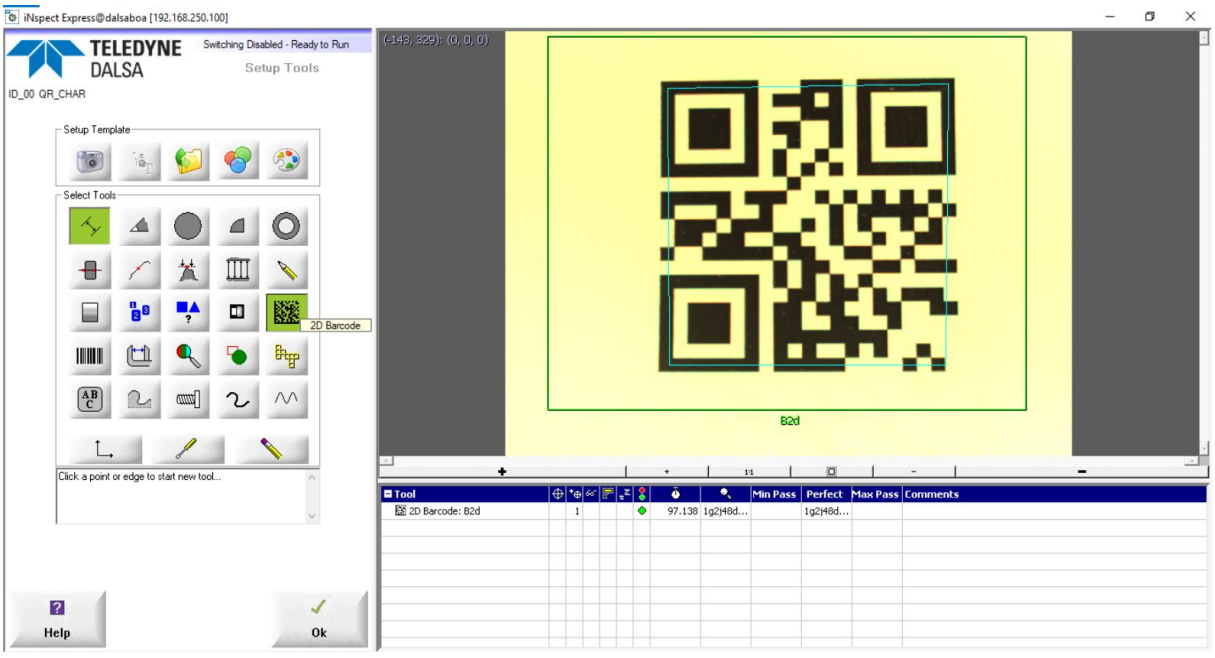

Figura 2.1: Utilização da ferramenta de captura de QR code Fonte: Adaptado pelo autor, baseado no software iNspect Express BOA Cameras.

Outra utilidade e vantagem de um sistema de rastreabilidade surge no presente estudo de caso, gerando dados correspondentes a cada objeto inspecionado. Assim que o objeto entra no processo de transformação ou outro tipo de processamento, é feita a sua identificação em tempo real por meio de um sistema de visão provido de câmaras de processamento de imagem, e realiza-se o registo do número de série num servidor(Figura 2.1). Esses dados são correlacionados com sistemas de planificação da produção, processo a processo e passo a passo, para garantir que nenhuma etapa é esquecida, e que são concluídas na ordem correta, acumulando assim um 'historial' de produção relativo a cada objeto. Ao rastrear peças individuais e ao armazenar uma variedade de parâmetros de produção relacionados com a mesma, é possível identificar exatamente como, quando e onde pode ter ocorrido um problema, de forma rápida e eficaz. Uma identificação do local onde terá ocorrido o erro, e podendo assim este ser corrigido antes que se transforme num problema maior, pode significar a diferença entre a recolha da produção de um mês inteiro para simplesmente a mudança de peças defeituosas individuais.

2.2 Base de Dados da Informação

Atualmente, a produção e o acesso à informação é conseguido por diversos meios, principalmente através dos meios eletrónicos. No entanto, para que tenhamos chegado a este ponto, as tecnologias de suporte têm-se baseado numa forte evolução tecnológica, tais como nas redes de comunicação, nas ferramentas e nas linguagens de programação, bem como nos dispositivos ou terminais utilizados na produção e visualização da informação. Na área das Tecnologias da Informação (TI) foram desenvolvidas soluções a fim de se lidar com limitações causadas pelo aumento de diferentes dispositivos que acedem ao mesmo recurso, onde se processam os dados por eles gerados [1]. MySql é um manipulador de base de dados que pode armazenar tabelas de dados. Alguns dos benefícios que o MySQL proporciona são a flexibilidade, escalabilidade e a simplicidade de implementação. Além destas vantagens, o MySQL é de código aberto e é suportado em várias plataformas [12].

Uma base de dados no MySQL consiste em tabelas constituídas por linhas e colunas,

onde as colunas são os atributos que cada linha conterá; uma ilustração da estrutura pode ser vista na tabela 1.

Coluna0 /Linha0	Coluna1 /Linha0	Coluna2 /Linha0
Coluna0 /Linha1	Coluna1 /Linha1	Coluna2 /Linha1
Coluna0 /Linha2	Coluna1 /Linha2	Coluna2 /Linha2

Tabela 2.1: Uma ilustração de como uma tabela está estruturada no MySQL.

2.3 Autómato e Câmara de Processamento de Imagem

Na indústria em geral utilizam-se dispositivos capazes de efetuar operações lógicas, como computadores e controladores lógicos programáveis PLC. São equipamentos capazes de, em alguns casos, substituirem tarefas humanas ou realizarem outras que o ser humano não consegue. Entre os dispositivos mecânicos, destacam-se motores, atuadores hidráulicos e pneumáticos. É um equipamento projetado para comandar e monitorizar máquinas ou processos industriais, como um computador especializado, baseado num microprocessador que desempenha funções de controlo através de softwares desenvolvidos pelo utilizador. Deverá possuir um processador com software de controlo, sendo que cada marca tem um próprio, e hardware que suporte operações em ambientes industriais. Este software, que é específico para automação e controlo, possui um sistema operacional de tempo real, algo indispensável para o controlo de processos indústriais. São sistemas dotados de redes de campo abertas como MODBUS-RTU, PROFIBUS e, mais recentemente, o PROFINET, de uso muito comum em aplicações mais complexas. As linguagens de programação utilizadas são maioritariamente a Ladder e a Function Block. Já ao nível de Hardware, este deverá suportar as condições extremas de mudanças de temperatura, humidade, pressão ou outras situações nas quais um computador padrão não suportaria essas condições. Existem inúmeros fabricantes de PLC, destacando-se a Omron, uma companhia Japonesa de componentes eletrónicos com sede em Kyoto e fundada em 1933,

que apresenta já um longo historial nesse ramo, e que disponibiliza ótimos produtos. O PLC NJ101-9020 pertence a um nova série NJ1, sendo um novo controlador Sysmac para sequência lógica e controlo de movimento destinado a máquinas compactas.

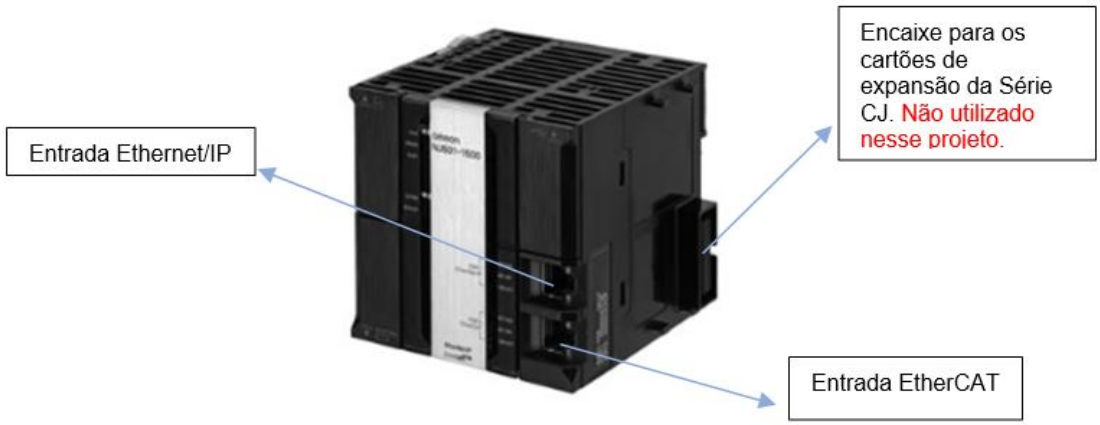

Figura 2.2: CPU Unit NJ101-9020 Fonte: Adaptado do catalogo da Omron da Série NJ.

A capacidade de comunicar através de EtherCAT e de EtherNet/IP, assegurando uma elevada velocidade e qualidade na comunicação na ligação com o cliente e com o banco de dados MySQL, e o facto de possuir uma integração prévia a HMI NA5-9W001B por meio do software da empresa Sysmac Studio, são algumas das características mais relevantes deste autómato.

Acoplada à unidade CPU do Nj-101-9020 está um conjunto de slots de entradas e saídas digitais e analógicas, controladas pelo software de programação do PLC por meio da comunicação com EtherCAT. Na figura 2.2 pode observar-se a CPU Unit NJ101-9020 juntamente com a sua fonte de alimentação, que é um módulo à parte e que pode ser encaixado na mesma.

A Câmara Teledyne Dalsa possui um software integrado que pode ser acedido através do navegador do computador, e já incorpora programas desenvolvidos, como a leitura de código de barras bidimensional (usualmente denominado por QR-Code) que se utiliza na

presente tese para identificar todo objeto a ser inspecionado pelo braço robótico. Além de incluir programas para medir distâncias, medidas e escritas, é de realçar a elevada capacidade de meios de comunicação como TCP/IP, Modubus, Profinet, entre outros.

Figura 2.3: Câmara Teledyne Dalsa Fonte: Adaptado do catalogo da TELEDYNE DALSA.

2.4 Estação de inspeção - UR3 e barreiras de proteção

A estação de inspeção em estudo está equipada com um robô colaborativo da Universal Robots UR3, o qual executa as tarefas de posicionamento do sistema de aquisição de imagens e inspeção dos botões dos displays. Trata-se de um robô colaborativo de mesa para tarefas leves de montagem e automação, equipado com uma caixa de controlo e uma interface de utilizador de programação.

Utiliza a linguagem de programação Python que pode ser usada para criar qualquer coisa, de scripts curtos a uma Graphical User Interface (GUI) completa. O Python é reconhecido por ter sintaxe simples e ser fácil de entender, especialmente para qualquer pessoa com experiência anterior em programação [13].

Constitui a escolha ideal para aplicações que exigem recursos de 6 eixos onde o tamanho, segurança e custos são factores críticos.

Além do software descrito anteriormente, foram investigadas linguagens de programação e protocolos de comunicação, como é o caso da linguagem Python para manipular

a informação gerada pelo robô, Function Block no caso do autómato e o protocolo de comunicação ModBus em ambos para a transferência da informação com a base de dados.

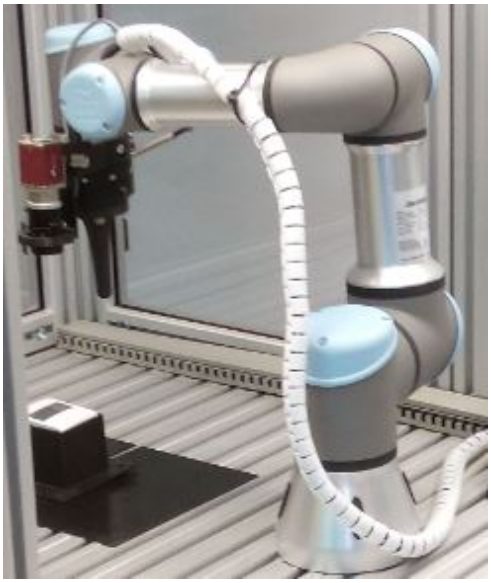

Figura 2.4: Robô Colaborativo UR3.

2.4.1 Equipamento de proteção

Por se tratar de uma área colaborativa onde o operador e o robô partilham o mesmo espaço de trabalho, e por forma a alterar a peça que será inspecionada, alguns equipamentos de segurança estão instalados a fim de melhorar a sua segurança, tais como um botão de paragem de emergência, luzes para indicar quando o robô está em operação e uma barreira de sensores da Omron, respetivamente.

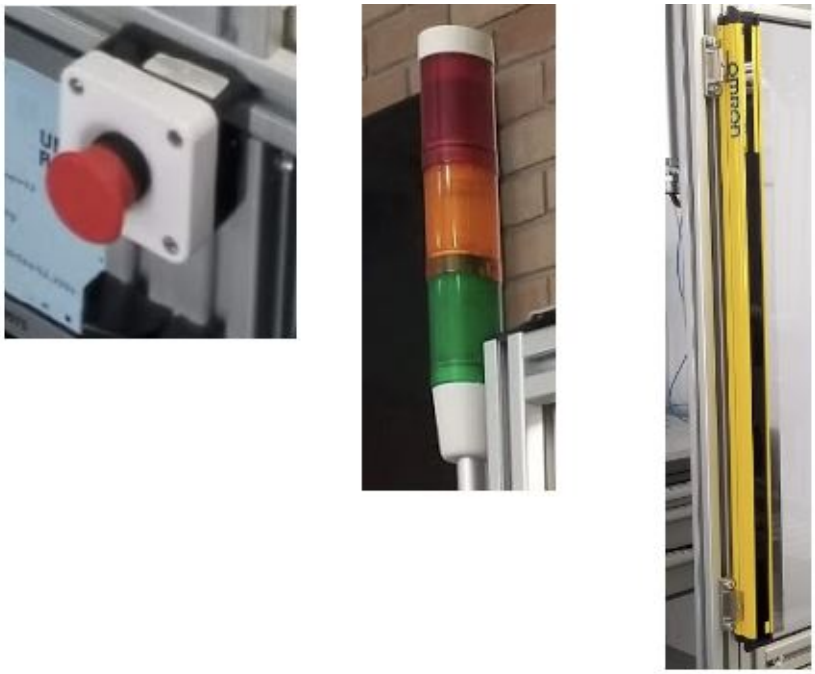

Figura 2.5: Equipamentos de Segurança constituintes da bancada industrial.

2.5 Ferramenta para manipulação das Pesquisas

O Node-RED é uma ferramenta visual de ambiente de código aberto (licença Apache 2.0), desenvolvido por uma equipa da empresa International Business Machines (IBM) e que recentemente foi integrada na JS Foundation. Utiliza na sua construção JavaScript, mais especificamente o Node.js, como o principal framework.

Desta forma, Node-RED consegue tirar proveito do modelo I/O dirigido a eventos (event-driven) não bloqueante (non-blocking) do Node.js, adequado para aplicações que lidam com manipulação intensa de dados em tempo real. Pode ser executada em diferentes plataformas, tais como os sistemas operacionais comumente utilizados em máquinas locais (Windows, Mac, Linux), em microcoprocessaddores como o Raspberry PI (bastante utilizados na construção de aplicações de IoT), além de soluções em cloud [14]. O modelo de programação está baseado em fluxos, direcionados para a Internet das Coisas. Sendo assim, toda a sua programação é realizada com base na construção de workflows entre nós

(também chamados de caixas ou blocos) e conexões (também chamados de fios ou arcos). Dentre os motivos, foi escolhida para ser utilizada na implementação desta tese por ser especialmente orientado para a construção de mashups (aplicações web) para IoT [15], e por possuir uma grande comunidade ativa que diariamente lança novas funcionalidades. Os nós consistem, geralmente, num par de arquivos (arquivo JavaScript), responsável por definir a funcionalidade do nó, ou seja, toda a programação do nó fica nesse arquivo. No Node-RED cada nó é desenvolvido com uma finalidade. Eles podem ser subdivididos em três tipos distintos: entrada, processamento e saída. Nó de entrada (inputs): são responsáveis pela entrada de dados numa aplicação. Assim sendo, todos os nós que lidam com a entrada de dados a partir de fontes externas são categorizados como nós de entrada; Nó de processamento (functions): são as funções que manipulam dados oriundos dos nós de entrada. Esses são os nós mais comuns, pois eles processam e disponibilizam estes dados para uma determinada saída; Nó de saída (outputs): são responsáveis pela saída dos dados, ou seja, externalizam os dados tratados para um outro meio. A figura 2.6 exemplifica um fluxo criado com três nós, um de entrada, um de processamento e um de saída.

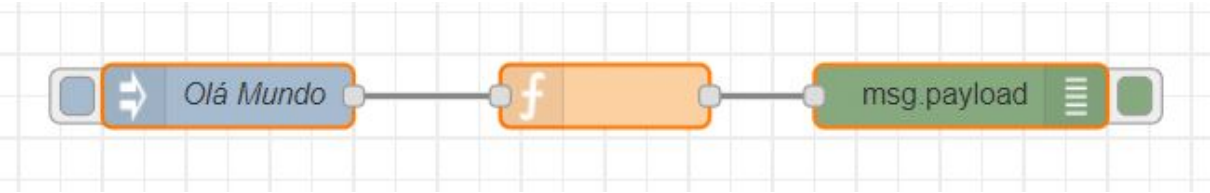

Figura 2.6: Agrupamento de nós em flow do Node-RED

A linguagem Ladder está entre as cinco linguagens de programação de Controladores Lógicos Programáveis PLC definidas pela norma IEC 61131-3 : FBD(Function Block Diagram, Diagrama de Blocos), LD (Ladder Diagram, Diagrama Ladder), Structured text (ST) (Texto Estruturado) e foi originalmente desenvolvida para construir e documentar circuitos a relés, utilizados em processos de produção. O Modbus é um protocolo de comunicação da camada de aplicação (modelo OSI) e pode utilizar o RS-232, RS-485 ou

a Ethernet como meios físicos - equivalentes à camada de enlace (ou link) e à camada física do modelo. O protocolo possui comandos para envio de dados discretos (entradas e saídas digitais) ou numéricos (entradas e saídas analógicas).

Desta forma todos os dispositivos conseguem comunicar com a base de dados, fornecendo à aplicação desenvolvida em Node-RED todos os meios necessários para o correto funcionamento do sistema de rastreabilidade que se pretende implementar na bancada industrial.

Capítulo 3

Estudo de caso

3.1 Descrição do estudo de caso

A bancada industrial é composta por diversa tecnologia, sendo que os dispositivos que a compõem contribuem para a um processo de inspeção a diferentes tipos de displays, averiguando o seu correto estado de funcionamento/fabrico em dois aspetos diferentes. A nível físico conta-se com a ajuda de uma ferremanta de pressão que, quando acoplada num robô UR3, permite determinar o estado dos botões; "ok"quando o a pressão no botão está conforme, "not-ok"para os botões nos quais não se deteta qualquer tipo de pressão ou "rigid"para aqueles em que a pressão exercida no botão é maior que a normalizada. O segundo estado de inspeção diz respeito às imagens processadas por uma câmara de alta resolução para processos industriais, quando os displays mostram defeitos no display. Os equipamentos do robô UR3 são conectados via TCP-IP num computador industrial, montado e configurado previamente por outros alunos que desenvolveram projetos anteriores a este.

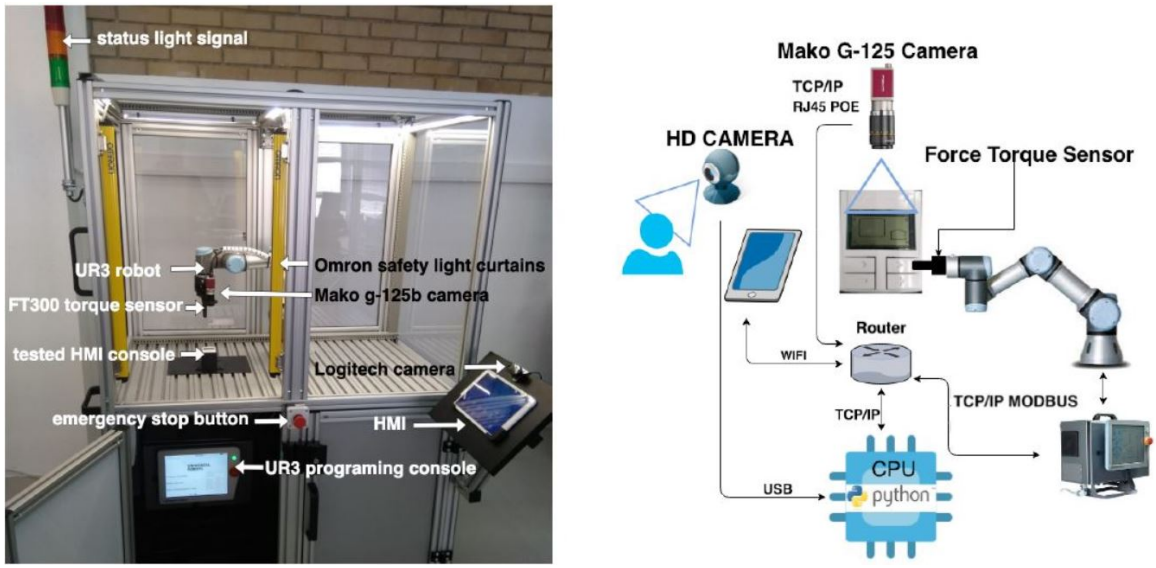

Figura 3.1: Organização do Hardware da estação de inspeção robotizada Fonte:[16]

À bancada industrial existente foi acrescentado o autómato programável NJ101-9020 da Omron, juntamente com uma slot de entradas e saídas analógicas e digitais, uma consola HMI e uma câmara de processamento de imagem TENDEY BOA.

No passado, os processamentos de imagem eram complicados de realizar, pois os softwares de desenvolvimento eram complexos e apresentavam a necessidade da configuração de diferentes parâmetros. Atualmente, com o uso de câmaras como a BOA, o processemento de imagem apresenta-se mais facilitado, dado que deixa de ser necessário desenvolver as complexas programações em softwares, devido ao facto de serem as próprias câmaras a conterem um software, possível de aceder diretamente através do navegador, e que contém alguns programas com funcionalidades muito úteis para diferentes tipos de tarefas[17]. Das diferentes funcionalidades possíveis foi usada a de leitura de códigos de barras dimensionais, QR-Code. Além da utilidade de diferentes funções, também é de elevado interesse a variedade de meio de comunicação que são suportados pela mesma, como TCP/IP, modbus e RS2332. Esta combinação de tecnologia encarrega-se de fazer a leitura de cada objeto a ser inspecionado, por meio da câmara, e envia a informação para uma base de dados. O mesmo processo de transmissão de dados acontece com os

valores registados nas entradas analógicas e na consola HMI. Todas as máquinas utilizadas para o processo de inspeção de displays possuem um meio eficaz de comunicar entre si, um protocolo de comunicação muito utilizado em processos de supervisão, denominado MODBUS. É um dos protocolos mais utilizados em automação industrial, graças à sua simplicidade e facilidade de implementação, podendo ser utilizado em diversos padrões de meio físico, como RS-232, RS-485 e Ethernet TCP/IP (MODBUS TCP).

3.2 Abordagem

O objetivo é o de aglomerar informações dos dispositivos já existentes e também daqueles adicionados, por forma a fornecer uma detalhada visão da informação que passou pelo processo industrial de inspeção de displays, e de forma organizada por meio de uma aplicação gráfica. É através de um servidor virtual que se realiza a gravação dos dados gerados pelo processo numa base de dados. A partir desta base de dados a aplicação desenvolvida consegue organizar das mais variadas formas toda a informação, e mostrá-la ao utilizador. Todos os dados têm de ser conduzidos desde a sua origem até ao servidor para de seguida ser feito o apropriado tratamento, em dados estatísticos, operações aritméticas, gráficos didáticos ou pesquisas personalizadas. Ao estarem todos os dados na base de dados, e utilizando uma aplicação totalmente personalizada, o NODE-RED, monitorizar e mostrar o processo de fabrico, bem como a informação mais relevante do mesmo, torna-se possível. No estudo de caso procura-se saber a que horas teve início a inspeção, qual o operador que estava responsável pelo lote, o número de série de cada objeto e o seu tipo, o estado de cada botão inspecionado, a relação de erros dos LCD (Liquid Cristal Display) dos displays, valores analógicos registados, estado do teste visual e mecânico, entre outros valores, que são enviados através das máquinas, nomeadamente o autómato e o robô industrial.

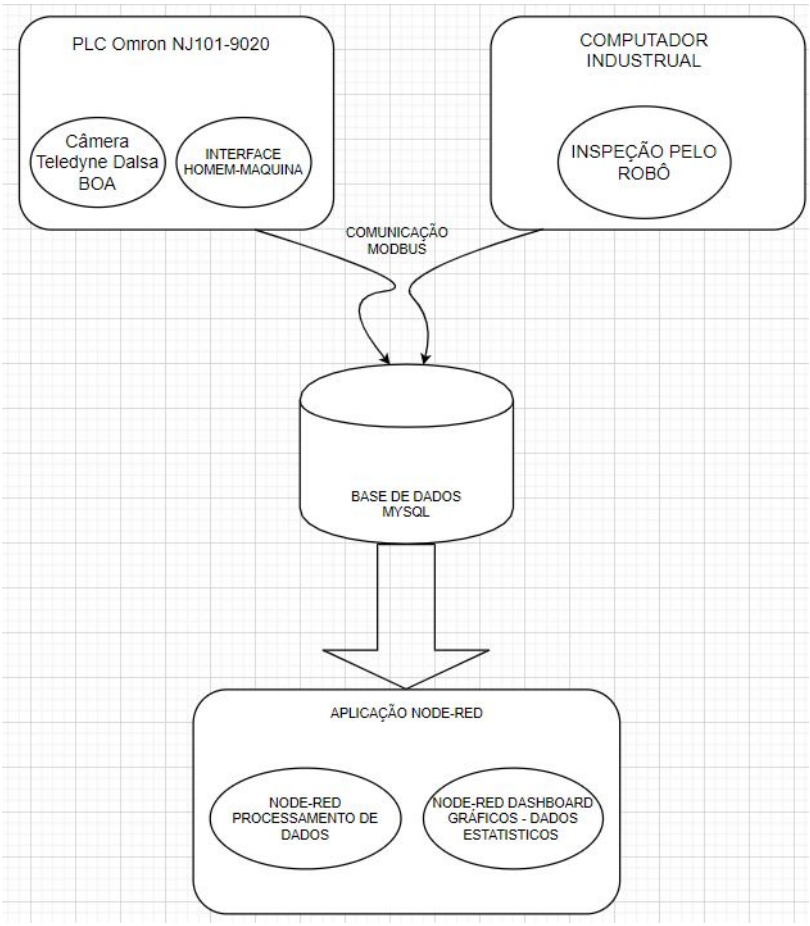

Figura 3.2: Arquitetura da Rede

No computador industrial, onde se encontra o código existente de programação do robô colaborativo, é instalado um servidor local e uma base de dados, constituída por tabelas a serem preenchidas, agrupando em colunas as informações provenientes do autómato e do robô colaborativo [13]. No mesmo computador é ainda instalado o programa de tratamento dos dados Node-RED, juntamente com a extensão para ambiente gráfico - Node-RED Dashboard. O autómato possui um software próprio instalado no computador do autor, e aí foi desenvolvida a programação necessária para a conexão e o envio de informações para a base de dados.

A base de dados é formada por tabelas, sendo que cada uma delas tem inúmeras colunas, o que tornaria muito difícil enviar informações coluna a coluna, desde o autmómato.

Por esse motivo, é possível enviar para a base de dados apenas uma variável capaz de transportar toda a informação, inclusive de diferentes tipos. As Structures são um tipo diferente das variáveis comuns e têm a vantagem de enviar toda a informação desejada de forma compatível com a Base de Dados[11]. A relação do tipo de variáveis entre o Sysmac Studio e o das colunas da base de dados MySQL encontram-se no Anexo I. Essa tabela foi retirada do manual "NJ/NX-series Database Connection Unidade Central de Processamento (CPU)Units" da Omron que foi extremamente útil na correta programação do autómato. É de salientar que os nomes dos elementos das variáveis criadas no autómato e das variáveis criadas na base de dados são exatamente iguais. Outra observação a realçar é que nem todas as colunas da tabela precisam de estar na Structure, pois em alguns casos trata-se de valores auto incrementais como é o caso da coluna "id" e "data-horário".

Após esta iformação chegar corretamente à base de dados apareceu o desafio de, utilizando a aplicação Node-RED, aceder aos dados guardados na base de dados do servidor para fazer a sua organização de forma a mostrá-los de forma clara e organizada ao utilizador, por meio de uma interface gráfica. É uma aplicação muito utilizada no mundo da Internet das Coisas devido ao facto de possuir, entre muitas funcionalidades, a capacidade de ser totalmente personalizada e de conter diversas ferramentas utilizadas no transporte de informação, via MySQL, MQ Telemetry Transport (MQTT), entre outros e ainda uma interface gráfica dedicada onde o utilizador consegue realizar o processo de rastrabilidade dos produtos.

Capítulo 4

Implementação do sistema de rastreabilidade

Este capítulo descreve o desenvolvimento e a aplicação de ferramentas necessárias, a nível de hardware e software, para que a estação de inspeção robotizada combine as diferentes tecnologias no transporte de informação numa aplicação, rapidamente acessível por qualquer dispositivo móvel que se encontre na mesma rede local. É feita referência às principais etapas para articular o processo e aos resultados obtidos após cada etapa implementada. De salientar que, tanto para o robô como para o autómato, serão aqui descritas algumas citações de código, adicionados aos que já se encontravam em funcionamento, elaborados por alunos, uma vez que o objetivo é o de aglutinar a informação proveniente de diferentes tecnologias na aplicação, esta sim, inteiramente desenvolvida pelo autor desta tese. Será ainda exemplificado como configurar a o Node-RED, mostrando para o efeito linhas de código devidamente justificadas ilustrando as suas funcionalidades. Na figura 4.1 indica a funcionalidade do sistema de rastreabilidade implementado.

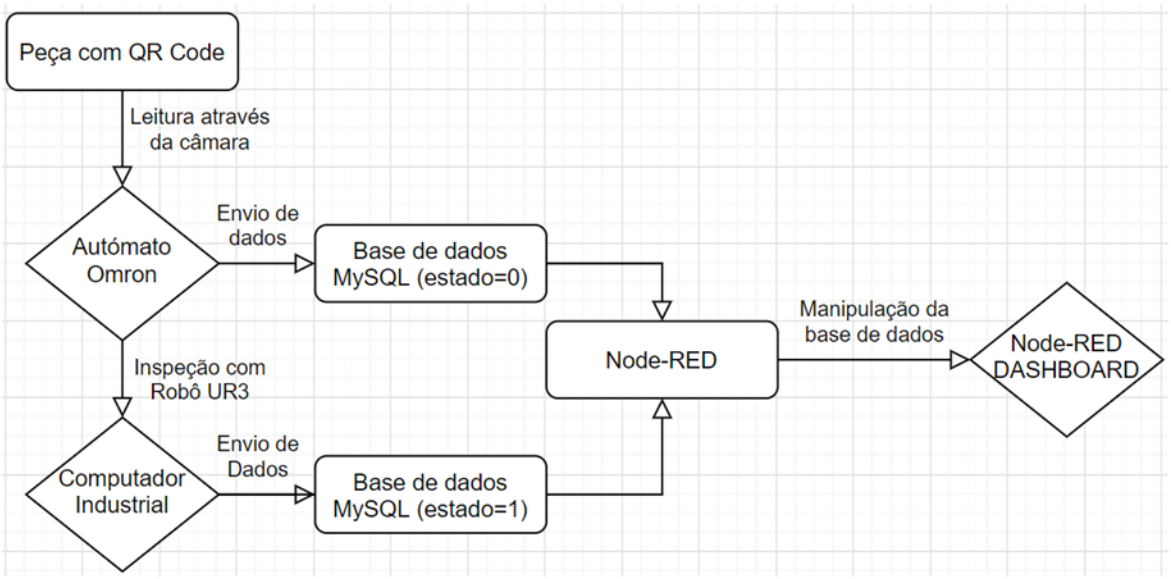

Figura 4.1: Esquema de funcionamento do Sistema

4.1 Sistema de Gestão de Base de Dados

Um sistema de gestão de base de dados é indispensável para armazenar e poder consultar os dados gerados em tempo real. Na figura 4.2 é ilustrada a interface do MySQL onde foi criada uma base de dados chamada "leitura"com uma tabela chamada "peça", a qual é responsável por receber, guardar e disponibilizar as informações dos dispositivos que operam na bancada de inspeção.

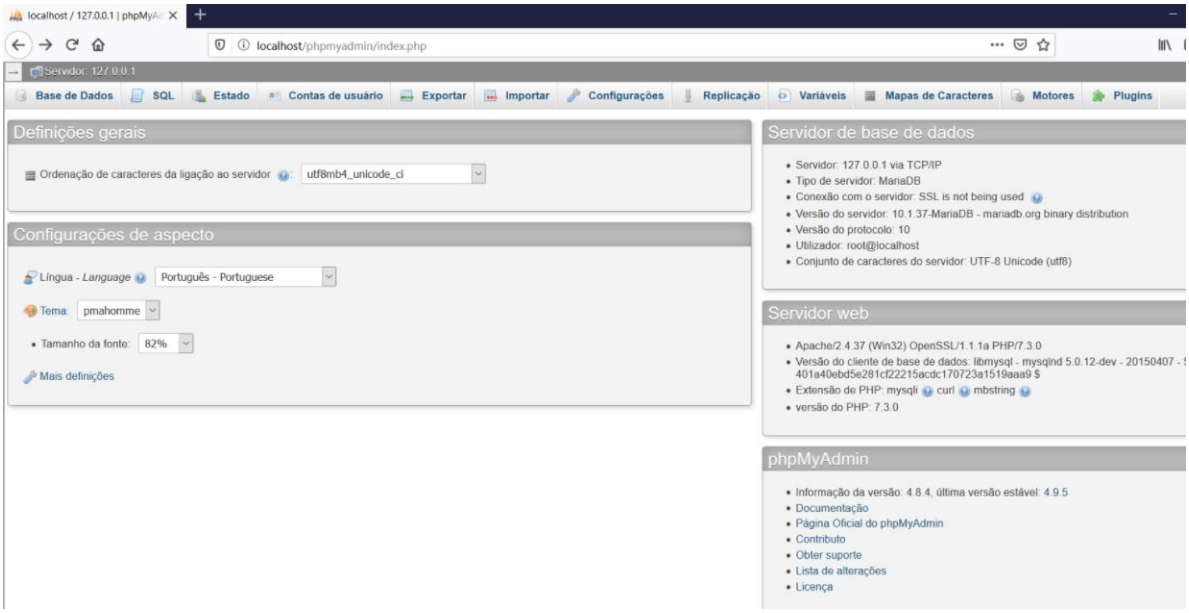

Figura 4.2: Interface Inicial MySQL

Após a criação da tabela foram criados os campos, que contêm os locais onde serão alojadas as informações provenientes das máquinas. Estes dizem respeito ao valor código QR, identificado por "referência", "valor analógico"lido através de um potenciómetro conectado a uma entrada do autómato, e também o "teste analógico", o "teste visual"e o "teste mecânico"provenientes da consola HMI da Omron. Desde este último campo de entrada até ao final, dizem respeito os dados provenientes do computador industrial, no qual está em execução o programa que controla o braço robótico e a câmara a ele acoplada. A variável "estado"representa um valor predefinido no MySQL e tem uma função crucial, a de colocar um determinado valor aquando dos dados recebidos do PLC, pois é sempre o primeiro dispositivo a escrever na base de dados, e para evitar que os dados provenientes do robô (enviados através de um protocolo de comunicação ModBus) sejam preenchidos numa nova linha, pois isso iria dificultar a operação de consulta na tabela. Este último, quando tem informação a enviar para a base de dados, vai procurar uma linha onde tenha esta variável com o valor "0", procedendo ao registo dos valores e alterando o valor da variável para "1"no final de cada inspeção. Desta forma é possível assegurar que a linha fica completa com a informação relativa ao objeto inspecionado.

	#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
☐	1	**referencia**	varchar(50)	latin1_swedish_ci		Sim	*NULL*			Muda Elimina Mais
☐	2	**valor_analogico**	int(11)			Sim	*NULL*			Muda Elimina Mais
☐	3	**teste_analogico**	varchar(50)	latin1_swedish_ci		Sim	*NULL*			Muda Elimina Mais
☐	4	**teste_visual**	varchar(50)	latin1_swedish_ci		Sim	*NULL*			Muda Elimina Mais
☐	5	**teste_mecanico**	varchar(50)	latin1_swedish_ci		Sim	*NULL*			Muda Elimina Mais
☐	6	**tempo_leitura**	timestamp			Não	CURRENT_TIMESTAMP			Muda Elimina Mais
☐	7	**estado**	int(1)			Não	0			Muda Elimina Mais
☐	8	**tipodisp**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais
☐	9	**estadobutonA**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais
☐	10	**estadobutonB**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais
☐	11	**estadobutonC**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais
☐	12	**estadobutonD**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais
☐	13	**tempopc**	timestamp			Não	0000-00-00 00:00:00			Muda Elimina Mais
☐	14	**relationErro**	int(100)			Não	*None*			Muda Elimina Mais
☐	15	**operador**	varchar(20)	latin1_swedish_ci		Não	*None*			Muda Elimina Mais

Figura 4.3: Variáveis criadas em MySQL

4.2 Conexão com Base de Dados e Transferência de Informações

Para programar o PLC NJ-1001 da Omron e a consola de interface homem-máquina existe um software da própria marca que possibilita a configuração de sensores, câmaras de processamento de imagem, o Slave Terminal, entre outros drivers. Este software agiliza a utilização de cada componente pela forma de os acoplar digitalmente, utilizando uma conexão fácil e intuitiva apoiada por diversos documentos específicos do fabricante, os quais fornecem o suporte técnico necessário aos programadores. As linguagens de programação são mais comuns no mundo dos autómatos: Ladder, texto estruturado (ST) ou blocos funcionais (FB). A conexão entre o autómoto e o computador que o irá programar é feita através de cabo Ethernet, tal como indica a figura 4.4 que mostra um switch para multiplicar o número de entradas de Ethernet, pois sem ele não seria possível conectar, em simultâneo, o autómato, o Slave Terminal, a consola HMI e o computador.

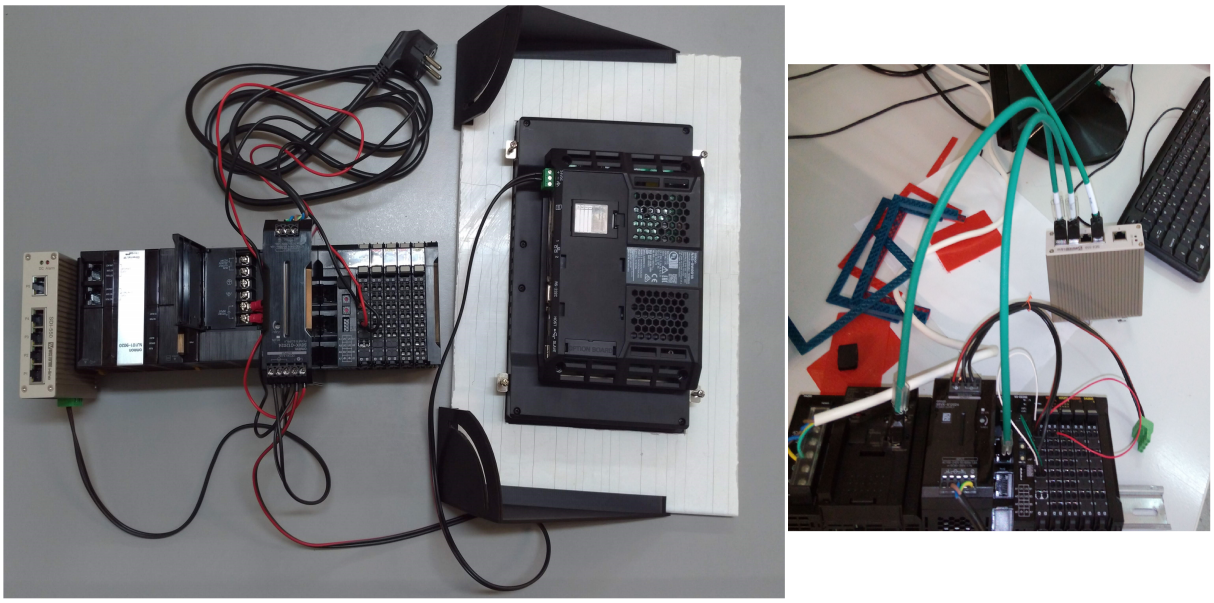

Figura 4.4: Autómato, Consola HMI e Switch para ligações

Após a conexão de todos os elementos ao autómato e a progrmação da consola HMI terem sido feitos, tendo sido elaborado para o efeito um manual pelo colega internacional Douglas Schauren, tornou-se possível a introdução de valores, validação de outros e escolhas de opções. A realização da conexão entre o autómato e a base de dados, uma opção existente no autómato NJ1001-900, foi simplificada através da consulta a uma tabela retirada do manual "NJ/NX-series Database Connection CPU Units" da Omron, respeitando a relação do tipo de variáveis do Sysmac Studio e da base de dados MySQL. A tabela em questão pode ser consultada no Anexo I.

A base de dados, como já foi visto, é formada por tabelas, sendo que cada uma pode conter inúmeras colunas. No entanto, enviar coluna a coluna para a base de dados seria pouco eficiente, pois pode ser necessário enviar um grande número de valores, o que poderia sobrecarregar a rede e levar a bloqueios no processo de inspeção. Para o efeito, o PLC contém um novo tipo de variáveis: as Structure. Consiste numa ferramenta

essencial para este projeto, uma vez que é capaz de transportar diferentes tipos de variáveis ao mesmo tempo para a base de dados. Cada tabela na base de dados deve ter uma Structure criada no Sysmac Studio, e os elementos deverão ter exatamente o mesmo nome e serem do mesmo tipo. Na figura que se segue exemplifica-se a variável Structure criada com as variáveis de igual nome e tipo, daquelas que estão presentes na base de dados, e respeitando o tipo com base na tabela do Anexo I.

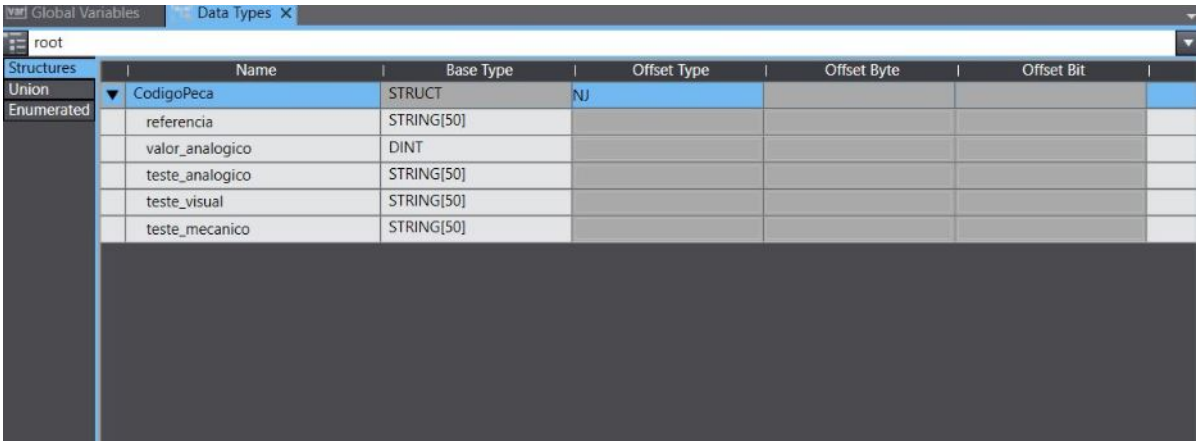

Figura 4.5: Variável Stucture do Sysmac Studio

Com este novo tipo de variável criada e acedendo ao "Global Variables", onde estão as restantes variáveis, é então criada uma com o nome "Insert-DB"do tipo "CodigoPeca", que corresponde ao nome da Structure. A forma de obter a informação referente ao elemento "referencia"da Structure, por exemplo, faz-se da seguinte forma: "Insert-DB.referencia". Para os restantes elementos contidos na Structure o processo é feito da mesma forma. Este processo é necessário aquando da programação da conexão do PLC com a base de dados, para proceder ao envio dos valores obtidos no processo de inspeção realizado pelo autóimato. A programação deste processo é na linguagem de blocos estruturados (FB-Function Block). Incialmente é necessário abrir a conexão entre o autómato e a base de dados, por meio do bloco FB “DB-Connect” que se encontra na Toolbox ->DB Connect. Na figura 4.6, observa-se o modo de utilizar, com o Trigger ligado no “Execute”, podendo esse ser uma ação automática do PLC, ou um comando do utilizador a partir da consola HMI. Em “DBConnectionName”, é inserido o nome da conexão criada anteriormente. Ao criar a

conexão, é gerada uma chave de identificação (ID) da conexão, que deve ser utilizada para posteriormente inserir os dados na base de dados. A variável que é adicionada no campo DBConnection é do tipo DWORD. Aberta a conexão, "DBConnection01", é necessário realizar um mapa entre a Structure e a tabela por meio do bloco "DB-CreatMapping", a fim de estabelecer ligação entre os elementos da Structure e as suas colunas na tabela, respetivamente. Começa-se por colocar o ID da conexão, criada no bloco "DB-Connect", seguido do nome, entre aspas, da tabela à qual se irá interligar e gravar as informações (visível em "TabelName"de "DB-CreatMapping"). Para inserir os dados utiliza-se um comando SQL -DB-SQLTYPE-INSERT"seguido por "."e a variável em questão. Todo este processo descrito pode ser visto na figura 4.6.

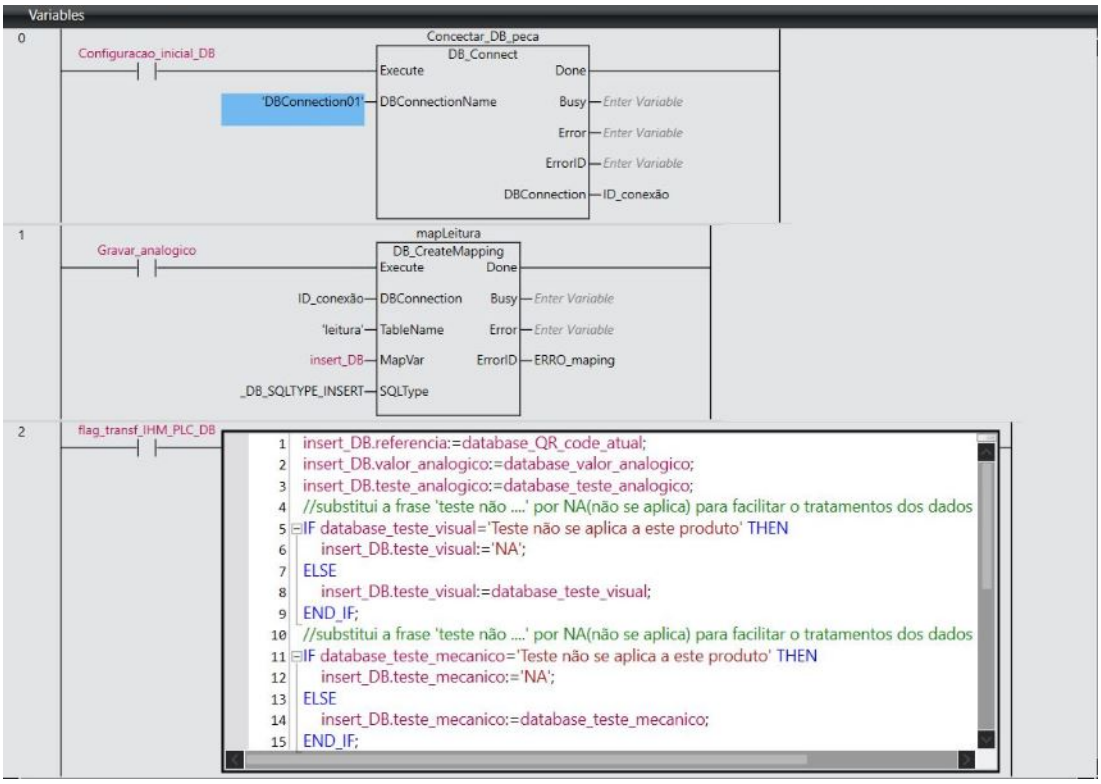

Figura 4.6: Função "variáveis"no SysmacStudio para enviar informação para base de dados

Criado o mapa, os dados são enviados para a base de dados pelo bloco "DB-Insert"onde é necessária a introdução da ID da conexão aberta, o nome da variável a enviar e o tempo que a base de dados estará à espera de valores, antes de considerar algum tipo de erro,

sendo este o "TimeOut"do tipo TIME. Após os dados terem chegado ao destino, a conexão deve ser fechada, usando para isso o bloco "DB-Close"que tem a função de informar o ID da conexão que termina aqui, e a informação a ser enviada para uma linha da tabela na qual se está a trabalhar.

4.3 Programação Phyton no PC Industrial

Do computador industrial, proveniente do robô colaborativo acoplado a uma garra de pressão e a uma câmara de processamento de imagem, é igualmente necessário enviar informações sobre os resultados obtidos para a base de dados, no final da inspeção a cada objeto. Nesta etapa, ao programa desenvolvido por colegas e que está em funcionamento no computador industrial com o objetivo de realizar os testes aos objetos, identificando o seu tipo, estados de algumas características e ainda a pessoa responsável pela ordem de execução, foram acrescentadas linhas de código para que estas informações cheguem à base de dados.

Por ser uma base de dados partilhada, pelo autómato e pelo computador industrial, houve um enorme desafio de sincronização, devido ao facto de, após uma das máquinas enviar informações, que são inseridas numa linha da tabela, a máquina seguinte a enviar os seus dados iria procurar preencher uma nova linha da tabela. Como o objetivo é o de armazenar a informação proveninte de ambas as máquinas na mesma linha, para depois se poder aceder aos dados completos de uma determinada referência e número identificativo do objeto, foi necessário encontrar uma alternativa funcional. O código da programação do robô encontra-se devidamente estruturado e comentado, o que agiliza a sua interpretação para a implementação de uma nova função capaz de fazer a conexão com a base de dados, e de transmitir os resultados obtidos do processo de inspeção.

A função "DataAquire"desenvolvida pelo autor, estabelece a ligação à base de dados, interligando-se através do IP do servidor no qual a base de dados está alojada, identificando o utilizador, "user", a palavra passe e ainda o nome da base de dados na qual irá fazer as operações de escrita. Nesta função são definidos como parâmetros de entrada

os valores que se querem enviar para a base de dados, provenientes do código que faz a inspeção aos objetos. O tempo durante o qual ocorre esta operação e o local onde vão ser inseridos, que neste caso deve ser na linha que contenha o valor da variável estado a "0"proveniente dos dados inseridos pelo autómato, e passando o valor para "1"a fim de indicar que a linha relativa a um objeto fica completa, têm de ser enviados para completar informações enviadas através do autómato. A imagem 4.7 faz referência ao código da inspeção de um objeto sem display e ao envio dessa informação através da função DataAquire, no final do processo realizado.

```
import mysql.connector #biblioteca para aceder ao MySQL
from datetime import datetime #valor da presente data e hora

def data(FinaLabel,button1,button2,button3,button4,relationErro):
#paramtros de entrada da funcao
    mydb = mysql.connector.connect(     #conexao com a base de dados
        host = "10.20.38.14",   #localhost
        user = "root",
        passwd = "root123",
        database = "peca",
    )


    my_cursor = mydb.cursor()
    my_cursor.execute("SELECT * FROM leitura WHERE estado =  0")
#procura a linha onde o valor da variavel estado seja 0
    myresult = my_cursor.fetchall()

    for x in myresult:
        print(x)
nowpc = datetime.now()  # tempo do pc
formatted_date = nowpc.strftime('%Y-%m-%d %H:%M:%S')    #formatacao da
data e hora para um formato que a base de dados consiga interpretar


sql_update_query = "Update leitura set estado = %s, tipodisp = %s,
estadobutonA = %s, estadobutonB = %s, estadobutonC = %s, estadobutonD
= %s, relationErro = %f, tempopc = %s, WHERE estado=0"
input = (1, FinaLabel, button1, button2, button3, button4,
relationErro, formatted_date)    #valores que sao enviados para a base
de dados com a alteração da variável estado para o valor de "1"
my_cursor.execute(sql_update_query, input)      #processo de escrita
na base de dados
mydb.commit()#fecho da conexão
```

Figura 4.7: Função DataAquire para enviar a informação do computador industrial para base de dados.

Além destes dados, também as imagens dos objetos e dos operadores se encontram

guardadas na memória na computador industrial e são utilizadas para enriquecer a interface da aplicação de rastreabilidade. As linhas de código abaixo dizem respeito às alterações que tiveram de ser feitas ao código existente, para conseguir reunir o conjunto de informações a serem transmitidas através da função "DataAquire".

```
if (FinalLabel == "objetoquadradosemdisplay"):
    # Display tipe TFT
    self.MsgStatus.config(text="Console Detected: \n Without TFT
Display \n Display wiil not be tested " )
    root.update()
    # ******START BUTTON INSPECTION************
    # Inspection Button A
    self.MsgStatus.config(text="Console Detected: \n Without TFT
Display \n Executing Button Inspection ")
    root.update()
    DataSensorStr = ModBusRobot.ModBusComunnication(30)
    maxi, mean, std, rms, GrdMax, GrdMin, GrdRms =
DataSensorPreProcess.PreProcess(DataSensorStr)
    DataButton = np.zeros((1, 7))
    DataButton[0] = [maxi, mean, std, rms, GrdMax, GrdMin, GrdRms]
    prediction = knn.predict(DataButton)
    if (str(prediction[0]) == 'OkButton'):
        StatusBtn = 'Ok'
    elif (str(prediction[0]) == 'NotOkButton'):
        StatusBtn = 'Not Ok'
    elif (str(prediction[0]) == 'RigidButton'):
        StatusBtn = 'Rigid'
    print(DataButton[0])
    MsgButton = 'Button A: ' + StatusBtn + '\n'
    self.MsgBtn.config(text=MsgButton)
    root.update()
    button1= str(prediction[0]) # Variável button1 guarda a informação
do estado do botão 1

(…)

# Inspection Button D
DataSensorStr = ModBusRobot.ModBusComunnication(33)
maxi, mean, std, rms, GrdMax, GrdMin, GrdRms =
DataSensorPreProcess.PreProcess(DataSensorStr)
DataButton = np.zeros((1, 7))
DataButton[0] = [maxi, mean, std, rms, GrdMax, GrdMin, GrdRms]
prediction = knn.predict(DataButton)
if (str(prediction[0]) == 'OkButton'):
    StatusBtn = 'Ok'
elif (str(prediction[0]) == 'NotOkButton'):
    StatusBtn = 'Not Ok'
elif (str(prediction[0]) == 'RigidButton'):
    StatusBtn = 'Rigid'
print(DataButton[0])
MsgButton = MsgButton + 'Button D: ' + StatusBtn
self.MsgBtn.config(text=MsgButton)
root.update()
button4 = str(prediction[0])  # estado do botão 4
relationErro = 0    # 0 pois não tem LCD
DataAquire.data(FinalLabel, button1, button2, button3,
button4,relationErro)
# Envio da informação para a função DataAquire para depois chegar à
base de dados
ModBusRobot.ModBusComunnication(10)    # Protocolo de comunicação
self.MsgStatus.config(text="Inspection Done \n Waiting New Inspection
\n or Finish Turn ")
```

Figura 4.8: Excerto do código principal do robô com inspeção de consola do tipo TFT sem LCD ao botão A e ao botão D, envio da informação para a função DataAquire.

4.4 Desenvolvimento da Aplicação de Rastreabilidade em Ambiente Gráfico

A criação e a configuração das ferramentas ligadas à aplicação NODE-RED vão constituir o motor de pesquisa à base de dados para que seja possível apresentar a informação de forma pretendida.

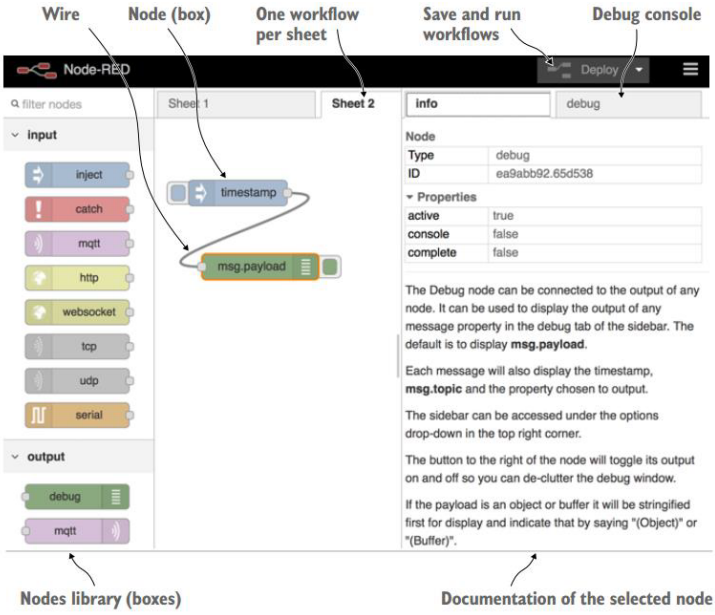

Figura 4.9: Interface de criação e programação da aplicação Node-RED Fonte: [18]

O ambiente de desenvolvimento desta aplicação consiste numa interface web subdividida em três secções. A primeira mais à esquerda agrupa todos os nós disponíveis na biblioteca, sejam eles providos pela instalação do Node-RED ou instalados posteriormente. Para elaborar um fluxo, como o apresentado na figura (do Cap 3), o utilizador só necessita de arrastar e soltar os nós contidos na secção à esquerda para dentro do canvas (workplace central) e conectar as respectivas portas dos nós[19][20]. A segunda secção é a workplace central, na qual quando nó é colocado no centro é possível fazer configurações no mesmo. A terceira e última parte mostra a documentação dos nós atualizada, ao selecionar os diferentes tipos de nós [4]. De igual forma, ela contém uma aba que corresponde

à consola de debug.

A Figura 4.9 demonstra o visual do node-RED destacando as diferentes partes nele inseridas. Procurou-se neste trabalho utilizar a aplicação Node-RED para obter o acesso a dados guardados na base de dados alojada no servidor, e fazer a sua organização para os mostrar de forma clara e ordenada ao utilizador através de uma interface gráfica. Como já foi referido, é uma aplicação muito utilizada no mundo da Internet das Coisas por possuir, entre muitas funcionalidades, a capacidade de ser totalmente personalizada e possuir extensões úteis e fáceis de instalar que permitem a ligação com MySQL, MQ Telemetry Transport (MQTT), entre outras. Pode ser considerada, no presente projeto, como uma ferramenta composta por duas estruturas.

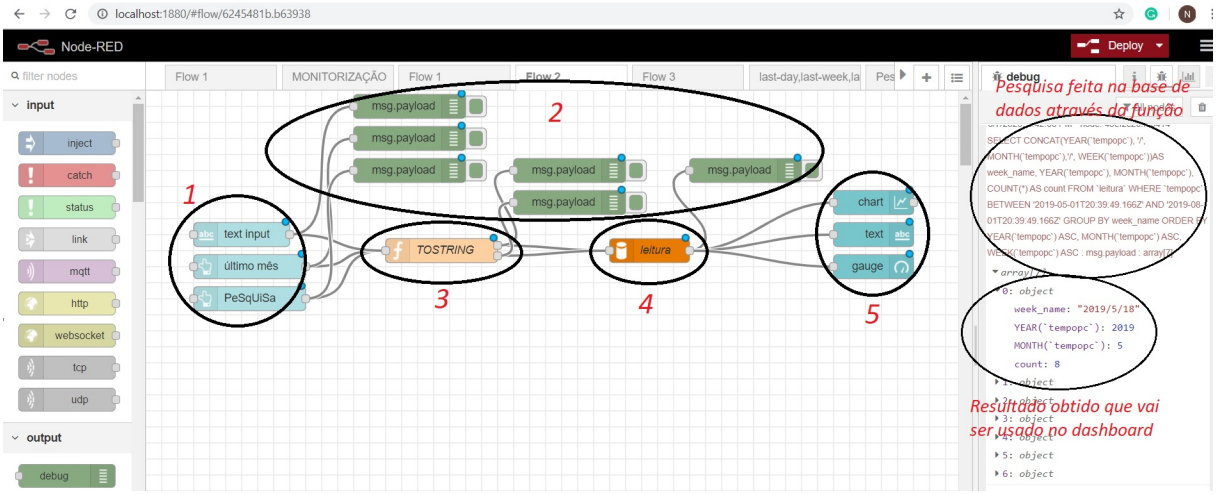

Figura 4.10: Ambiente de programação da aplicação Node-Red com as partes importantes identificadas.

Na primeira, e tal como indica a figura 4.10, são evidenciadas as zonas onde são introduzidas as entradas (1-Inputs) como botões, campos de texto e seleção de datas como inputs, as saídas para observar os resultados obtidos como texto ou gráficos (2-outpus), as funções onde são introduzidas as programações para o efeito pretendido, como tipo de pesquisa, contadores ou delimitar gráficos (3-function), o tipo de armazenamento como ficheiros Excel ou base de dados (4-Storage) e ainda elementos para a construção

do ambiente gráfico como gráficos, caixas de texto ou caixas de imagem (5-dashboard). Por se encontrar na mesma rede local que os demais dispositivos, este ambiente é acedido através do endereço "localhost:1880".[21]. A segunda estrutura é constituída pelo ambiente gráfico configurado na aba "Dashboard"identificada pelo balão número 5 da figura 4.10, por oferecer ao programador a oportunidade de criar os outputs que desejar e, desta forma, o utilizador consegue ter acesso aos resultados de várias pesquisas. A biblioteca do Node-RED não possui todos os nós existentes e, por isso, foram instalados alguns necessários para a elaboração deste projeto. Destacam-se o "mysql"no separador storag, nó utilizado para estabelecer a ligação entre Node-Red e a base de dados a fim de realizar operações de leitura e escrita na mesma, e os nós constituintes do separador dashboard, necessários para criar e configurar a interface gráfica que vai estar ao dispor do utilizador para que consiga realizar os testes de rastreabilidade aos produtos inspecionados.

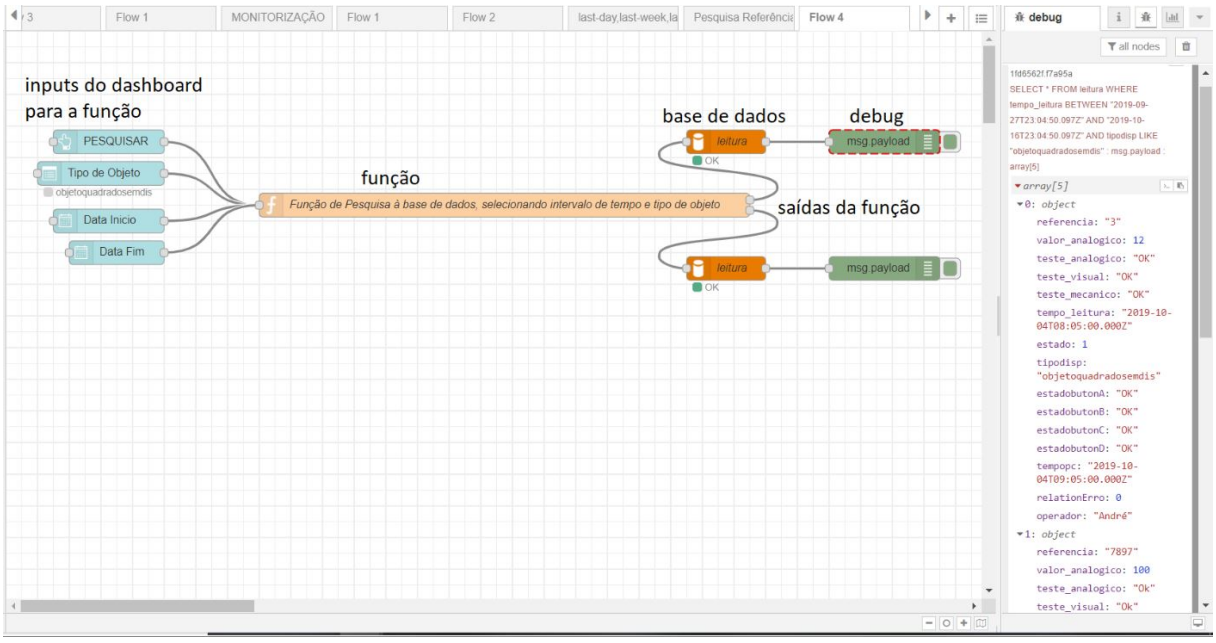

Figura 4.11: Nós Constituintes de uma operação de pesquisa da quantidade de objetos de determinado tipo e seu estado de ”ok” ou ”não ok” em Node-RED.

No flow da figura 4.11, é um exemplo da primeira implementação das funcionalidades

da aplicação desenvolvida em Node-Red para este projeto. Nele estão presentes nós de entrada (inputs que pertecem ao separador Dashboard), um nó de processamento com duas saídas (função que contem código capaz de interpretar os dados selecionados pelo utilizador), dois nós de conexão com base de dados e ainda um nó de debug, utilizado essencialmente para mostrar os valores que saem de um nó. Este flow tem como objetivo o de pesquisar na base de dados a quantidade de objetos que foram inspecionados num determinado intervalo de tempo, e o estado em que se encontram, quantos "Ok"ou "Nok". O tipo de objeto e o intervalo de tempo são atributos a selecionar pelo utilizador, servindo-se dos Inputs do Dashboard. São os Inputs que levam à função as características escolhidas pelo utilizador, sendo estas processadas a fim de retornarem algo para as suas saídas, podendo ser várias. Os nós "leitura"dizem respeito à base de dados, e vê-se que a ligação está estabelecida através do "OK", local onde está armazenada toda a informação gerada por diferentes dispositivos. Na zona do debug, lado direito da figura 4.11, observa-se o resultado de uma das ordens transmitidas pela função à base de dados, resultando num total de cinco objetos com todas as suas informações, neste caso do tipo "objeto-quadradosemdis"(visível em "tipodisp:"). A figura 4.12 apresenta o código presente na função que se localiza no centro do workplace, e está munido de comentários para facilitar a interpretação das suas funcionalidade, à excepção do "return", por ser um parâmetro que indica as saídas da função, ou seja, o resultado das pesquisas realizadas à base de dados. Esses dados podem ser consultados na aba "debug". O "return"é composto por dois tópicos, sendo o primeiro referente à primeira saída da função, e o segundo tópico correspondente à segunda saída. No primeiro tópico são selecionados todos os elementos (utilizando o caractere "*") provenientes da tabela "leitura", compreendidos no intervalo de tempo e tipo de display selecionados pelo utilizador. Já o segundo tópico faz a seleção dos mesmos objetos com a condição de selecionar apenas aqueles que cumpriram as normas impostas pelo processo de controlo da qualidade.

```
var select_ref= context.get("select_ref")  //criação de uma variável para guardar o valor selecionado em tipo de objeto

var select_data_min= context.get("select_data_min")    //variável que guarda o valor da data do inicio do periodo a fazer a pesquisa

var select_data_max= context.get("select_data_max")    //variável que guarda o valor da data do final do periodo a fazer a pesquisa

  if (msg.topic=="input_ref"){      //quando o Input no dashboard tiver sido selecionado um tipo de objeto vai acontecer o seguinte:

     context.set("select_ref", msg.payload)     //pega na opção selecionada

       flow.set("select_ref", msg.payload);      //guarda essa opção

}

  if (msg.topic=="input_data_min"){  //quando o Input no dashboard tiver sido selecionado uma data para o inicio do periodo a pesquisar vai acontecer o seguinte:

     context.set("select_data_min", msg.payload)    //pega na opção selecionada

       flow.set("select_data_min", msg.payload);      //guarda essa opção

}

  if (msg.topic=="input_data_max"){  //quando o Input no dashboard tiver sido selecionado uma data para o inicio do periodo a pesquisar vai acontecer o seguinte:

     context.set("select_data_max", msg.payload)    //pega na opção selecionada

       flow.set("select_data_max", msg.payload);      //guarda essa opção

}

  if (msg.topic=="PESQUISAR"){      //quando é pressionado o botão com o topico "PESQUISAR" é exucatada a seguinte operação

  return [{topic: 'SELECT * FROM leitura WHERE tempo_leitura BETWEEN "'+ new Date(select_data_min).toISOString()+'" AND "'+ new Date(select_data_max).toISOString()+'" AND tipodisp LIKE "'+select_ref+'"'},

     {topic: 'SELECT * FROM leitura WHERE  `teste_analogico` NOT LIKE "NOK" AND `teste_visual` NOT LIKE "NOK" AND `teste_mecanico` NOT LIKE "NOK" AND `estadobutonA` LIKE "OK" AND `estadobutonB` LIKE "OK" AND `estadobutonC` LIKE "OK" AND `estadobutonD` LIKE "OK" AND `relationErro` = 0 AND tempo_leitura BETWEEN "'+ new Date(select_data_min).toISOString()+'" AND "'+ new Date(select_data_max).toISOString()+'" AND tipodisp LIKE "'+select_ref+'"'},

  ]}
```

Figura 4.12: Função de Pesquisa à base de dados, selecionando intervalo de tempo e tipo de objeto.

Uma vez criada a função e conectada à base de dados, já existem informações para colocar na interface do utilizador. Para o efeito são adicionados nós e usados Outputs do dashboard do tipo formato de texto e gráficos .[21] Os gráficos utilizados são do tipo calibre

e gauge, e foram configurados para indicar os valores pretendidos. De entre as diversas formas de os configurar, optou-se por criar quatro novas funçõe:, duas que trasnsportam as informações recebidas da base de dados para cada saída, e outras duas para enviar a parametrização dos gráficos. A que tem o nome de "Todos os objetos", conectada à primeira saída, guarda numa variável o valor dos objetos provenientes da pesquisa, um total de cinco, como indicado na aba debug da figura 4.10. Na segunda saída a lógica é a de guardar numa nova variável o número de objetos "OK". Por fim, as restantes duas funções realizam cálculos baseados nas informações recebidas das anteriores para formatar as escalas e os valores a serem exibidos no gráfico. O código contido em cada uma das quatro funções encontra-se nas figuras 4.13 e 4.14 , acrescido de comentários.

```
----------Função "Todos os objetos"----------
var objetos_todos = (msg.payload.length);          //Criação de uma variável
"objetos_todos" que vai guardar o número de objetos provenientes da pesquisa
flow.set("objetos_todos", objetos_todos);          //Grava esse valor na variável
return {payload: objetos_todos};                   //Retorna o valor contido na variável
"objetos_todos"
--------------------------------------------------------------------------------------------------------

----------Função "Objetos OK"----------
var objetos_ok = (msg.payload.length);      //Criação de uma variável "objetos_ok"
que vai guardar o número de objetos provenientes da pesquisa
flow.set("objetos_ok", objetos_ok);         //Grava esse valor na variável
return {payload: objetos_ok};               //Retorna o valor contido na variável
"objetos_ok"
--------------------------------------------------------------------------------------------------------
```

Figura 4.13: Código das funções objetos-todos e objetos-ok.

```
----------Função "gauge ok"----------

var min = 0;    //Ciração de uma variável com o valor '0' para definir o início da escala do gráfico

var max = (flow.get("objetos_todos"));      //Criação de uma variável "max" com o valor recebido pela variável "objetos_todos"

var data = flow.get("objetos_ok");          //Criação de uma variável "data" com o valor recebido pela variável "objetos_ok"

msg = {payload:data,ui_control:{min:min,max:max}}       //Criação do Gauge (em ui_control) com o valor armazenado na variável "data" a exibir através do apontador do gráfico e e definição do valor mínimo (0) e o valor máximo (max) da escala do gráfico

return msg;    //Envio da informação para o nó de saída, onde será ligado o Gauge

----------

----------Função "gauge nok"----------

var min = 0;    //Ciração de uma variável com o valor '0' para definir o início da escala do gráfico

var max = (flow.get("objetos_todos"));      //Criação de uma variável "max" com o valor recebido pela variável "objetos_todos"

var ok = flow.get("objetos_ok");            //Criação de uma variável "ok" com o valor recebido pela variável "objetos_ok"

var data = max – ok;            //Criação de uma variável "data" com o valor da operação de subtração realizada pelos objetos_ok aos objetos_todos o que origina os objetos_não_ok

msg = {payload:data,ui_control:{min:min,max:max}}       //Criação do Gauge (em ui_control) com o valor armazenado na variável "data" a exibir através do apontador do gráfico e e definição do valor mínimo (0) e o valor máximo (max) da escala do gráfico

return msg;    //Envio da informação para o nó de saída, onde será ligado o Gauge

----------
```

Figura 4.14: Código das funções de criação dos Gauge.

No workbench do Node-Red, e após serem interligados estes nós e feitas as configurações dos mesmos, é criada a interface gráfica para uso do utilizador, e também esta foi configurada em termos de tamanhos das janelas, da disposição entre elas e a origem das informações a mostrar. Na figura 4.15 é mostrada a ligação entre todos os nós descritos.

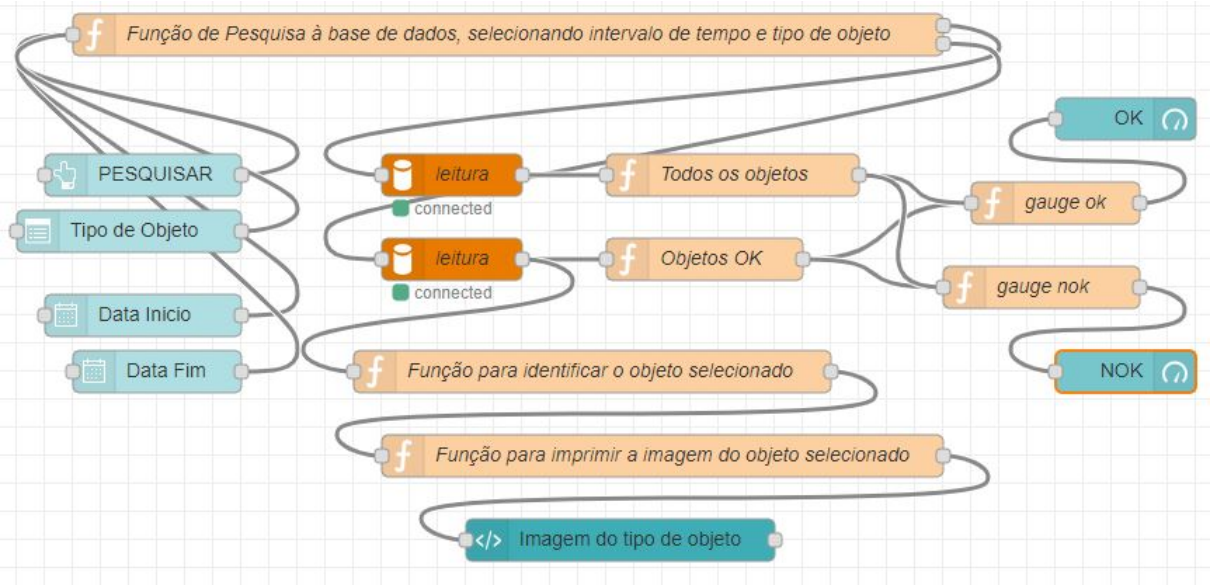

Figura 4.15: Configuração dos nós no flow da elaboração de gráficos com informações recebidas pela pesquisa à base de dados ordenada pelo utilizador.

A figura 4.16 mostra a visão do utilizador na interface gráfica após ter selecionado o tipo de objeto, definido o intervalo de tempo e pressionado o botão "PESQUISAR". É indicado o número de objetos inspecionados no perído de tempo entre 28 de Setembro e 17 de Outubro do ano 2020, e desses quais os que cumpriram as normas impostas pelos testes de controlo da qualidade, e quais não.

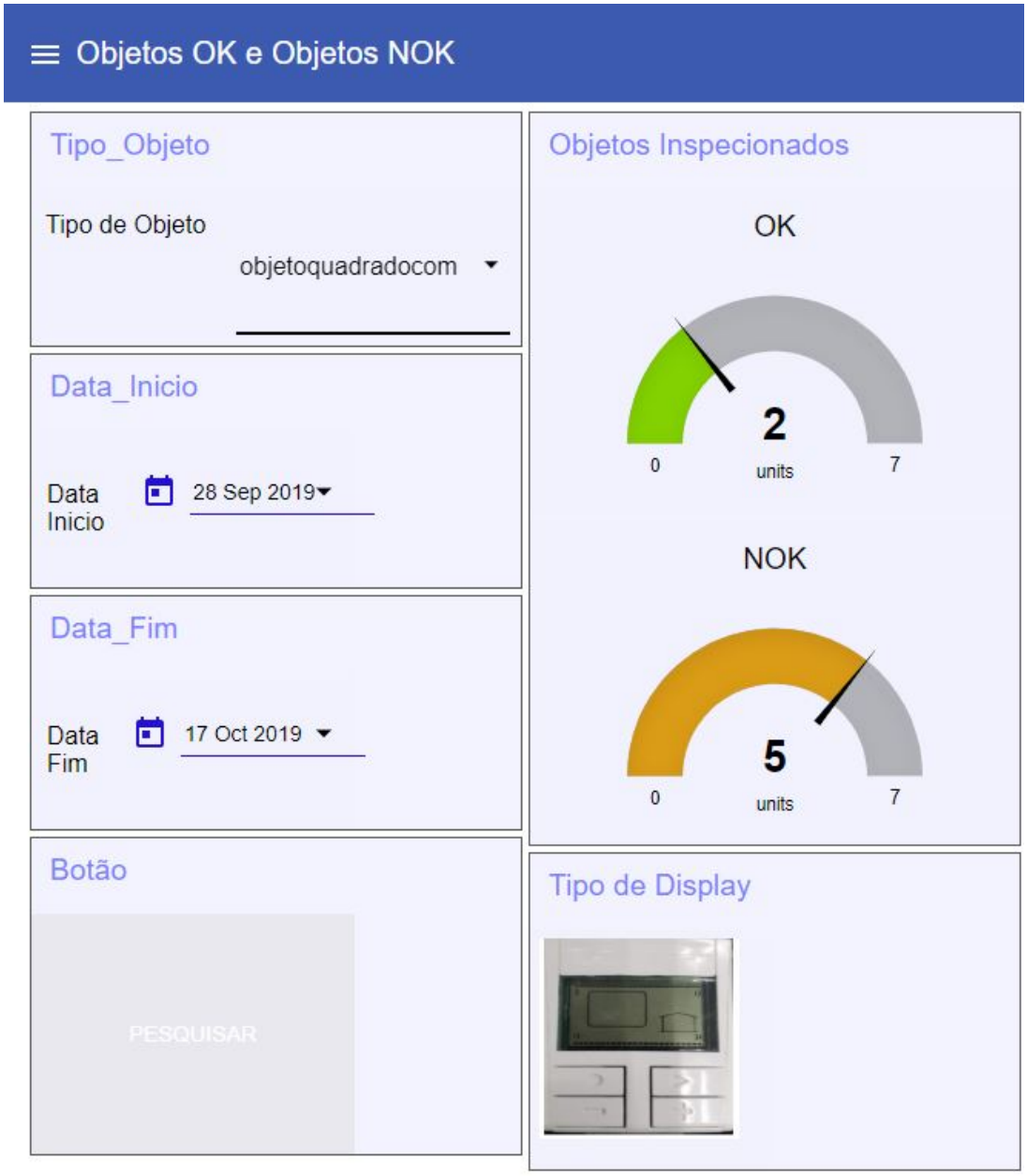

Figura 4.16: Resultado final no Dashboard dos nós que constituem o flow acima descrito.

4.4.1 Funcionalidades das funções implementadas no sistema de rastreabilidade Node-RED

Neste subcapítulo discutem-se os nós que deram origem ao sistema de rastreabilidade, de que forma são conectados e as suas configurações.

No sistema a usar por parte do utilizador existe a opção de pesquisa por número de objeto, e pesquisa por tipo de objeto num intervalo de tempo. Na figura 4.17 encontram-se os nós que constituem a primeira opção, composta por nós de entrada do dashboard, nós de função, nós de base de dados e nós de saída do dashboard.

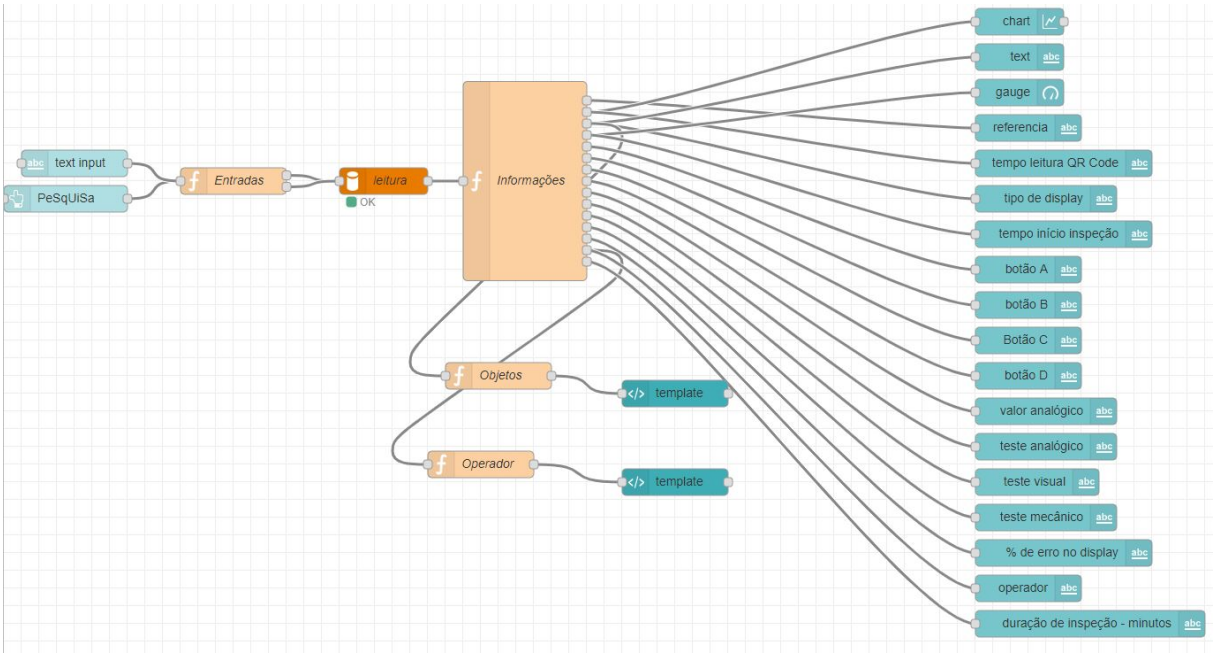

Figura 4.17: Nós constituintes do flow Pesquisa por Referenica.

Esta funcionalidade recolhe todas as informações relacionadas com o objeto identificado pelo número de série introduzido na caixa de texto, e dispõe os resultados em janelas. O flow que contém os nós necessários para esta opção encontra-se na figura 4.17.

Após a introdução do número de série e pressionado o botão "PESQUISAR", a ordem entra num nó que irá ser ligado ao nó da base de dados, e de seguida é feita a ligação a um outro nó do tipo função programada para aceder às informações retornadas, que são

agrupadas num conjunto de dados chamado strings. Para serem visualizados no ambiente gráfico, estes dados são ligados a nós do dashboard, uma ligação por cada saída da função, pois a função assim foi programada. O primeiro nó que recebe os parâmetros das entradas do dashboard é do tipo função, e tem o objetivo de criar variáveis com os valores inseridos pelo utilizador para elaborarem a pesquisa à base de dados. O código implementado neste nó está indicado na figura 4.18, com comentários para facilitar a compreensão da lógica utilizada.

```
var buff_input= context.get("buff_input")        //Variável que guarda o valor selecionado
pelo utilizador

if (msg.topic=="inputvalue"){

   context.set("buff_input", msg.payload)        //Comando para guardar o valor selecionado
pelo utilizaro

}

if (msg.topic=="button"){        //Ciclo que inicia após ser pressionado o botão de pesquisa

   return [{topic: "select * from leitura where referencia="+buff_input},          //Retorna tudo
que esteja na base de dados com o valor "referência" igual ao selecionado pelo utilizador

        {topic:"SELECT TIMESTAMPDIFF(MINUTE,`tempo_leitura`, `tempopc`) as dif_time from
leitura where referencia="+buff_input}          //Calcula a diferença ao minuto dos tempos
em que foi lido o objeto pelo autómato e o tempo no qual o computador industrial enviou os
dados relativos à inspeção

]

   }
```

Figura 4.18: Código fonte da função Escolha do número de série.

Este flow é responsável por mostrar as informações guardadas na base de dados relativas a cada objeto inspecionado. Assim, após ter numa função as variáveis criadas com os valores introduzidos pelo utilizador, são ligados os nós função à base de dados por já estarem reunidas as condições para realizar a pesquisa. Uma vez na aplicação Node-RED todas as informações provenientes da base de dados, criou-se uma nova função ("Informações") capaz de extrair os dados retornados pela base de dados. Desta forma foram criadas várias saídas na função para poderem ser ligados os nós de saída do dashboard, a fim de transportar para a interface gráfica toda a informação que o utilizador tem direito a visualizar acerca de um objeto. Na função "Informações"são criadas variáveis capazes de armazenar os dados proveninentes da base de dados, tal como indica a figura 4.19.

```
var msg2 ={topic: "var2",payload:msg.payload[0].referencia}     //Guarda na variável msg2 o valor da referência do objeto selecionado na pesquisa

var msg3 ={topic: "var3",payload:msg.payload[0].tempo_leitura}    //Guarda na variável msg3 o valor da data de leitura do QR-Code pelo autómato do objeto selecionado na pesquisa

var msg4 ={topic: "var4",payload:msg.payload[0].tipodisp}       //Guarda na variável msg4 o tipo de display do objeto selecionado na pesquisa

var msg5 ={topic: "var5",payload:msg.payload[0].tempopc}      //Guarda na variável msg5 o valor da data enviada pelo computador industrial após a isnpeção do objeto selecionado na pesquisa

var msg6 ={topic: "var6",payload:msg.payload[0].estadobutonA}          //Guarda na variável msg6 o estado do botão A do objeto selecionado na pesquisa

var msg7 ={topic: "var7",payload:msg.payload[0].estadobutonB}          //Guarda na variável msg7 o estado do botão B do objeto selecionado na pesquisa

var msg8 ={topic: "var8",payload:msg.payload[0].estadobutonC}          //Guarda na variável msg8 o estado do botão C do objeto selecionado na pesquisa

var msg9 ={topic: "var9",payload:msg.payload[0].estadobutonD}          //Guarda na variável msg9 o estado do botão D do objeto selecionado na pesquisa

var msg10 ={topic: "var10",payload:msg.payload[0].valor_analogico}    //Guarda na variável msg10 o valor analógico registado por um potenciómetro conectado no autómato do objeto selecionado na pesquisa

var msg11 ={topic: "var11",payload:msg.payload[0].teste_analogico}    //Guarda na variável msg11 o valor do teste analógico registado na consola do autómato do objeto selecionado na pesquisa

var msg12 ={topic: "var12",payload:msg.payload[0].teste_visual}          //Guarda na variável msg12 o valor do teste visual registado na consola do autómato do objeto selecionado na pesquisa

var msg13 ={topic: "var13",payload:msg.payload[0].teste_mecanico}    //Guarda na variável msg13 o valor do teste mecânico registado na consola do autómato do objeto selecionado na pesquisa

var msg14 ={topic: "var14",payload:msg.payload[0].relationErro}          //Guarda na variável msg14 o valor em percentagem do número de erros detetados no display pela câmara conectada ao robô do objeto selecionado na pesquisa

var msg15 ={topic: "var15",payload:msg.payload[0].operador}  //Guarda na variável msg5 o nome do operador que deu a ordem de inspeção para o objeto selecionado na pesquisa

var msg20 ={topic: "var20",payload:msg.payload[0].dif_time}    //Guarda na variável msg20 o valor em minutos da diferença das datas e hora na qual é registado o objeto pela câmara do autómato e quando o computador industrial envia os dados do objeto selecionado na pesquisa

return
[msg2,msg3,msg4,msg5,msg6,msg7,msg8,msg9,msg10,msg11,msg12,msg13,msg14,msg15,ms
g20];              //Retorna para a saída da função os valores guardados nas variáveis
```

Figura 4.19: Variáveis criadas na função Pesquisa por Referência.

As funções "Objetos"e "Operador"encarregam-se de fazer a leitura da informação proveniente da base de dados para que, formatada, consiga ir buscar a imagem correspondente à memória do Node-RED presente no diretório "myimg", e transportando para o ambiente gráfico as imagens correspondentes.

Na figura 4.20 está o código de cada uma destas funções.

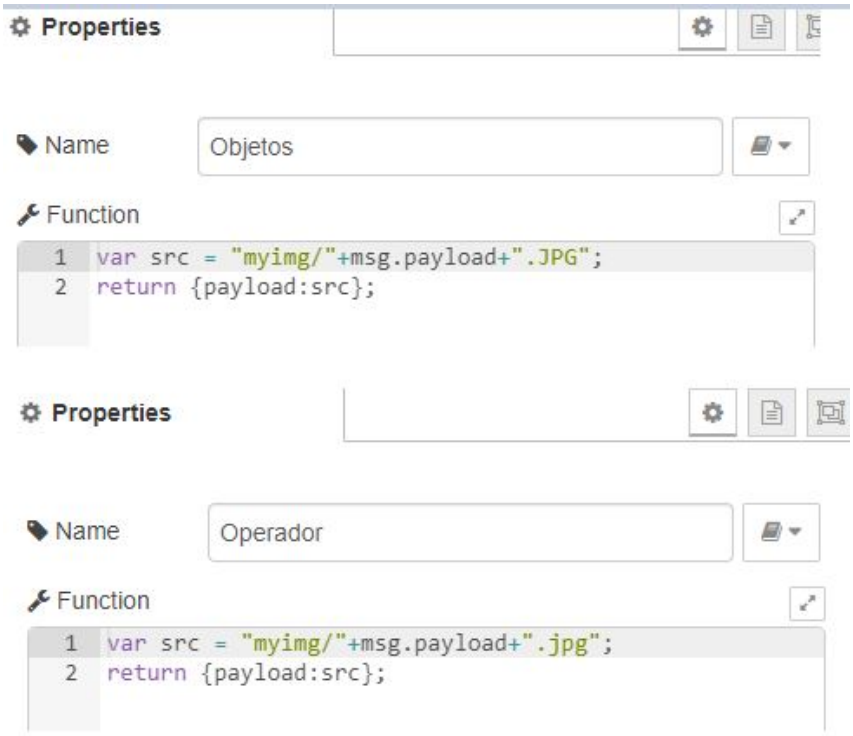

Figura 4.20: Códigos das funções do nós responsáveis por transformar a informação lida na base de dados em imagem, do objeto inspecionado e do operador respetivamente.

Desta forma, a opção de pesquisa por número de série reúne todas as características necessárias para o seu funcionamento, como se mostra no seguinte capítulo.

A segunda funcionalidade deste sistema de rastreabilidade consegue obter as informações de um grupo de objetos compreendidos num determinado intervalo de tempo. Tem funcionalidades de criação de gráficos de diferentes tipos, o de barras e do tipo piza, e os nós que constituem o flow necessário para o correto funcionamento apresentam-se na figura 4.21.

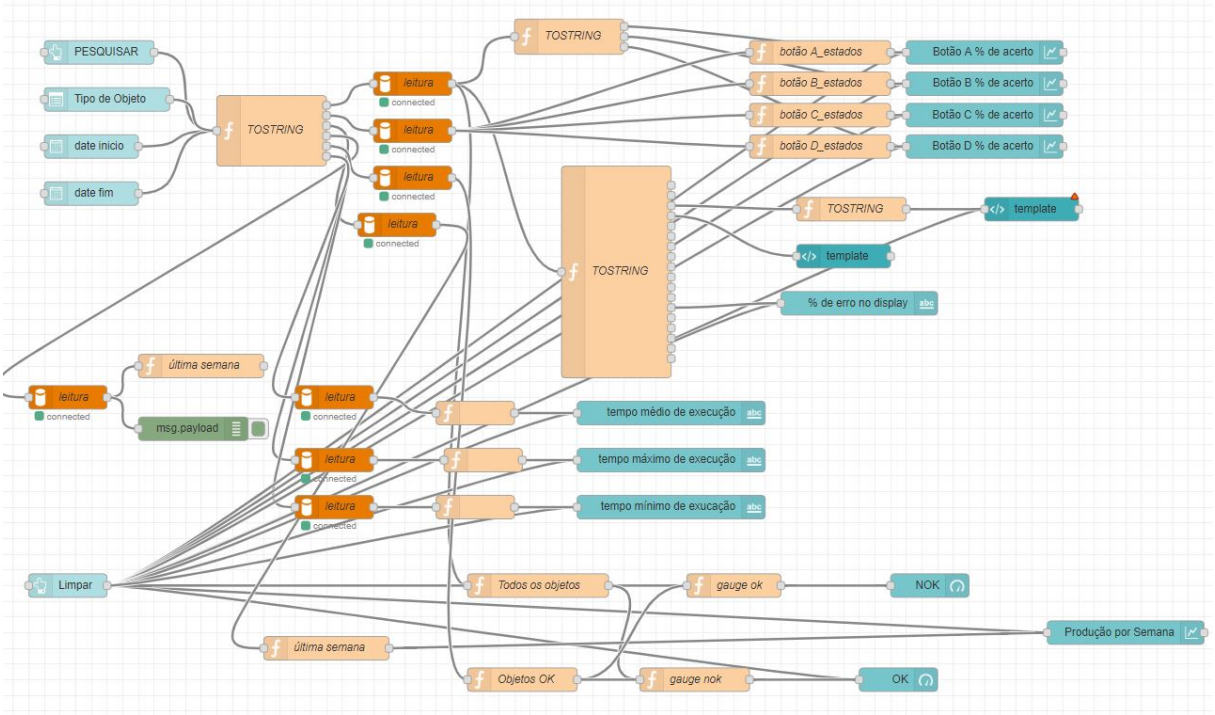

Figura 4.21: Nós constituintes do flow "Pesquisa por tipo de objeto".

No flow de pesquisa por tipo de objeto, e compreendido num intervalo de tempo, são utilizados quatro nós de entrada do Dashboard para a criação e gravação dos dados introduzidos pelo utilizador a partir da interface gráfica num nó de função, que é ligado a vários nós de base de dados para fazer diferentes pesquisas.

Além das funcionalidades presentes na pesquisa por objeto, que são as de organizar na base de dados informações relativas a cada objeto, bem como as imagens guardadas na memória do Node-RED, existem mais operações configuradas neste flow, nomeadamente as de originar gráficos. O primeiro nó criado neste flow do tipo função é responsável por criar e gravar em variáveis os elementos selecionados na interface gráfica. A figura 4.22 contém as linhas de código utilizadas para a configuração deste primeiro nó.

```
var select_button= context.get("select_button")          //Variável que guarda opção escolhida do tipo de botão a ser considerada na pesquisa

var select_data_min= context.get("select_data_min")  //Variável que guarda a data a ser considerada para o início do período para a pesquisa

var select_data_max= context.get("select_data_max")  //Variável que guarda a data a ser considerada para o fim do período para a pesquisa

if (msg.topic=="input_button"){          //Ciclo que vai ler a opção escolhida do tipo de display e guardá-la quando o utilizador pressionar o botão  pesquisar para ser utilizada em futuras pesquisas

   context.set("select_button", msg.payload)

   flow.set("select_button", msg.payload);

}

if (msg.topic=="input_ref"){      //Ciclo que vai ler e guardar a opção da referência a ser levada em conta para a pesquisa

   context.set("select_ref", msg.payload)

   flow.set("select_ref", msg.payload);

}

if (msg.topic=="input_data_min"){       //Ciclo que vai ler e guardar a opção escolhida na data que vai constar o início da pesquisa

   context.set("select_data_min", msg.payload)

   flow.set("select_data_min", msg.payload);

}

if (msg.topic=="input_data_max"){       //Ciclo que vai ler e guardar a opção escolhida na data que vai constar o fim da pesquisa

   context.set("select_data_max", msg.payload)

   flow.set("select_data_max", msg.payload);

}

if (msg.topic=="PESQUISAR"){   //Ciclo que vai ordenar as pesquisas à base de dados quando for pressionado o botão "PESQUISAR"
```

Figura 4.22: Código dos parâmetros de entrada da função do nó "Opções selecionadas" pelo utilizador.

O código que faz parte do return, parâmetro de saída da função, é composto por quatro tipos de pesquisa com as seguintes funcionalidades: Pesquisa na base de dados de todas as informações relativas ao tipo de display escolhido no período de tempo selecionado pelo utilizador; Pesquisa na base de dados de todas as informações relativas ao tipo de

display escolhido no período de tempo selecionado pelo utilizador e elaboração de um mapa capaz de guardar no Node-RED temporariamente a quantidade de botões "ok"e botões "nok"para a elaboração de gráficos; Pesquisa na base de dados da quantidade de objetos inspecionados correspondentes por semana; Pesquisa na base de dados de todas as informações relativas ao tipo de display escolhido no período de tempo selecionado pelo utilizador para fazer a diferença do tempo em que foi registado o número de série do objeto, e quando termina o processo de inspeção executado pelo robô industrial, dando origem ao tempo de inspeção; O código utilizado para realizar as tarefas acima descritas encontra-se na figura 4.23.

```
return [{topic: 'SELECT * FROM leitura WHERE tempo_leitura BETWEEN "'+ new
Date(select_data_min).toISOString()+'" AND "'+ new Date(select_data_max).toISOString()+'"
AND tipodisp LIKE "'+select_ref+'"'},      //Pesquisa na base de dados todas as informações relativas
ao tipo de display escolhido pelo utilizador no período de tempo selecionado pelo utilizador

        {topic:'SELECT SUM(`estadobutonB` LIKE "Ok") AS Ok_B, SUM(`estadobutonB` LIKE "Not
Ok") AS Not_Ok_B,SUM(`estadobutonB` LIKE "Rigid") AS Rigid_B, SUM(`estadobutonA` LIKE
"Ok") AS Ok_A, SUM(`estadobutonA` LIKE "Not Ok") AS Not_Ok_A,SUM(`estadobutonA` LIKE
"Rigid") AS Rigid_A, SUM(`estadobutonC` LIKE "Ok") AS Ok_C, SUM(`estadobutonC` LIKE "Not
Ok") AS Not_Ok_C,SUM(`estadobutonC` LIKE "Rigid") AS Rigid_C, SUM(`estadobutonD` LIKE
"Ok") AS Ok_D, SUM(`estadobutonD` LIKE "Not Ok") AS Not_Ok_D,SUM(`estadobutonD` LIKE
"Rigid") AS Rigid_D FROM `leitura` WHERE tempo_leitura BETWEEN "'+ new
Date(select_data_min).toISOString()+'" AND "'+ new Date(select_data_max).toISOString()+'"
AND tipodisp LIKE "'+select_ref+'"'},      //Pesquisa na base de dados todas as informações relativas
ao tipo de display escolhido pelo utilizador no período de tempo selecionado pelo utilizador e faz um
mapa para o Node-RED guardar temporariamente a quantidade de botões ok e botões não ok para
construção de gráficos

        {topic: "SELECT CONCAT(YEAR(`tempopc`), '/', MONTH(`tempopc`),
'/',WEEK(`tempopc`))AS week_name, COUNT(*) AS count  FROM `leitura` WHERE `tempopc`
BETWEEN '"+ new Date(select_data_min).toISOString()+'" AND "'+ new
Date(select_data_max).toISOString()+'" AND tipodisp LIKE "'+select_ref+"' GROUP BY
week_name ORDER BY YEAR(`tempopc`) ASC, MONTH(`tempopc`) ASC, WEEK(`tempopc`)
ASC"},          //Esta ordem tem a finalidade de conseguir identificar por semana a quantidade de
displays inspecionados para serem vistos através de gráfico de barras horizontal

        {topic:'SELECT SUM(`estadobutonB` LIKE "OK") AS ok_B, SUM(`estadobutonB` LIKE
"NOK") AS nok_B,SUM(`estadobutonB` LIKE "RIGID") AS rigid_B, SUM(`estadobutonA` LIKE
"OK") AS ok_A, SUM(`estadobutonA` LIKE "NOK") AS nok_A,SUM(`estadobutonA` LIKE "RIGID")
AS rigid_A, SUM(`estadobutonC` LIKE "OK") AS ok_C, SUM(`estadobutonC` LIKE "NOK") AS
nok_C,SUM(`estadobutonC` LIKE "RIGID") AS rigid_C FROM `leitura` WHERE
DATE_SUB(CURDATE(),INTERVAL 30 DAY) <= `tempopc`'},      //Pesquisa na base de dados
todas as informações relativas à inspeção de displays nos últimos 30 dias da data em que é feita a
pesquisa

        {topic: 'SELECT * FROM leitura WHERE  `teste_analogico` NOT LIKE "NOK" AND
`teste_visual` NOT LIKE "NOK" AND `teste_mecanico` NOT LIKE "NOK" AND tempo_leitura
BETWEEN "'+ new Date(select_data_min).toISOString()+'" AND "'+ new
Date(select_data_max).toISOString()+'" AND tipodisp LIKE "'+select_ref+'" AND `estadobutonA`
LIKE "OK" AND `estadobutonB` LIKE "OK" AND `estadobutonC` LIKE "OK" AND `estadobutonD`
LIKE "OK" AND `relationErro` = 0 '},      //Pesquisa na base de dados todas as informações relativas
ao tipo de display esxolhido pelo utilizador no período de tempo selecioando pelo utilizador que
contenham todas as características em conformidade

        {topic: 'SELECT TIMESTAMPDIFF(MINUTE,`tempo_leitura`, `tempopc`) as dif_time from
leitura WHERE tempo_leitura BETWEEN "'+ new Date(select_data_min).toISOString()+'" AND
"'+ new Date(select_data_max).toISOString()+'" AND tipodisp LIKE "'+select_ref+'"'},//Ordem
que vai fazer a diferença dos tempos de entrada do objeto no autómato e no computador industrial
para calcular o tempo médio de inspeção

]}
```

Figura 4.23: Código dos parâmetros de saída da função Opções selecionadas pelo utilizador.

Na elaboração de gráficos é necessário não só definir quais os valores de entrada a serem mostrados, mas também os valores que vão fazer parte dos eixos, de forma a tornar os dados de fácil interpretação. Existem dois tipos de gráficos distintos na aplicação desenvolvida, sendo eles os gráficos de barras e os gráficos tipo piza. O gráfico que mostra a quantidade de objetos inspecionados por semana é do tipo de barras, no qual é definido o eixo das ordenadas, como a semana na qual teve origem a inspeção, e no eixo das abcissas a quantidade de objetos. A figura 4.24 mostra o código da função utilizada para a elaboração deste tipo de gráfico.

```
var label = [];    //Criação de uma variável array para atribuir valor no eixo das abcissas
var count = [];   //Criação de uma variável array contar o número de objetos inspecionados e
valor esse a ser colocado  no eixo das coordenadas
var i=0;          //Criação de uma variável com o valor 0 para o ciclo percorrer todos os dados
recebidos pelo nó base de dados
for( i=0; i<msg.payload.length; i++ ) {    //Ciclo para elaboração do gráfico com os valores da
semana na qual foram inspecionados os objetos e a quantidade deles
  label[i]=msg.payload[i].week_name;          //eixo das abcissas com a identificação da
semana na qual foram inspecionados os objetos
  count[i]=msg.payload[i].count;        //Eixo das coordenadas com a quantidade de objetos
inspecionados
}
var m={          //Variável m que irá construir o gráfico

  "series":label,
  "data":[count],
  "labels":label,
};
return {payload:[m]};    //Retorno do gráfico pronto a ser conectado a um nó do Dashboard
```

Figura 4.24: Código presente na função do nó "Inspeções por semana"e coloca a informação em formato de ser observada através do gráfico de barras.

O outro tipo de gráfico utilizado é o do tipo piza, usado para indicar o estado dos botões inspecionados. Na programação das funções que lhe dão origem está presente um ciclo capaz de obter a informação do número total de objetos selecionados, e de os

organizar no gráfico por ordem de "ok", "nok"e "rigid", dependendo da informação que provém da base de dados. A figura 4.25 mostra o código utilizado para criar um gráfico, o do estado dos botões A inspecionados.

```
var i=0;
for( i=0; i<msg.payload.length; i++ )      //Ciclo que percorre todos os objetos chegados da
base de dados
var m2 = {        //Criação do gráfico
"labels" : ["ok_A", "nok_A", "rigid_A"],            //Eixo das abcissas
"data" : [[msg.payload[0].Ok_A, msg.payload[0].Not_Ok_A, msg.payload[0].Rigid_A,
        ]],        //Eixo das coordenadas com os valores dos estados dos botões
"series" : ["estado_button"]     //Título do gráfico
}
return {payload:[m2]};            //Envio do gráfico com os dados para um nó do Dashboard
```

Figura 4.25: Código presente na função do nó que indica o estado do botão-A e coloca a informação em formato de gráfico tipo Gauge.

Além destas funcionalidades, o sistema de rastreabilidade para a opção de pesquisa por tipo de objeto num intervalo de tempo também indica a quantidade de objetos inspecionados, juntamente com a indicação do número de objetos ok e objetos não ok, mostrando ainda a imagem correspondente ao tipo de objeto selecionado, tal como ilustrado na figura 4.16.

Desta forma o sistema de rastreabilidade encontra-se configurado relativamente às funcionalidades de consulta e disponibilização da informação do processo de inspeção, de forma organizada e ordenada por data de inspeção de todos os os elementos guardados até ao momento. Por último é necessário configurar os elementos que vão constituir a interface de utilização. Em todos os objetos que constituem a interface visual é possível definir o tamanho, a posição dos grupos, os valores de entrada e o aspeto visual. O comando "msg.payload"guarda a informação que se quer visualizar, contendo a informação transportada à saída de cada nó.

A figura 4.26 mostra a edição de um template do flow de pesquisa por número de série. Selecionada a aba do dashboard é possível visualizar as páginas criadas e os grupos que compõem cada uma.

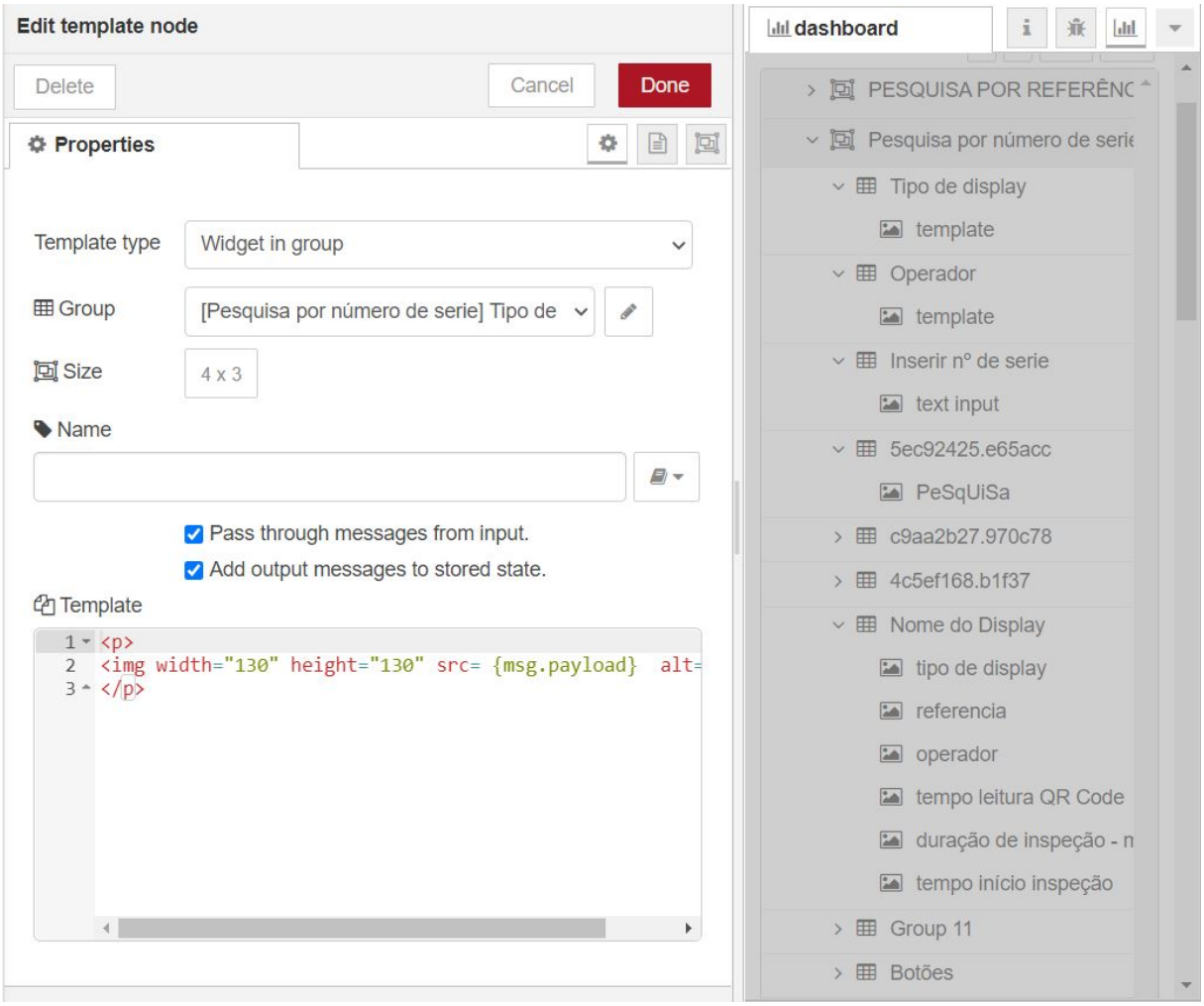

Figura 4.26: Configuração do nó template que mostra a imagem do operador que deu a ordem de inspeção.

Dentro de cada grupo existem espaçamentos que, no caso do template, pode ser selecionado o tamanho, e podem ser escritas linhas de código. Além do comando "msg.payload", que transporta a carga da mensagem, pode também ser utilizado o "Topic".

Na figura 4.27 é configurado um dos espaçamentos de seleção de data, o início do

período no qual se quer saber a quantidade de um tipo de objetos e as suas características. Após ser atribuído o grupo ao nó de escolha de data, selecionou-se o tamanho a ocupar na interface, foi-lhe atribuído um nome e criou-se um "Topic"com o valor "input-data-min"a ser interpretado pelo nó função, antes de elaborar a pesquisa à base de dados.

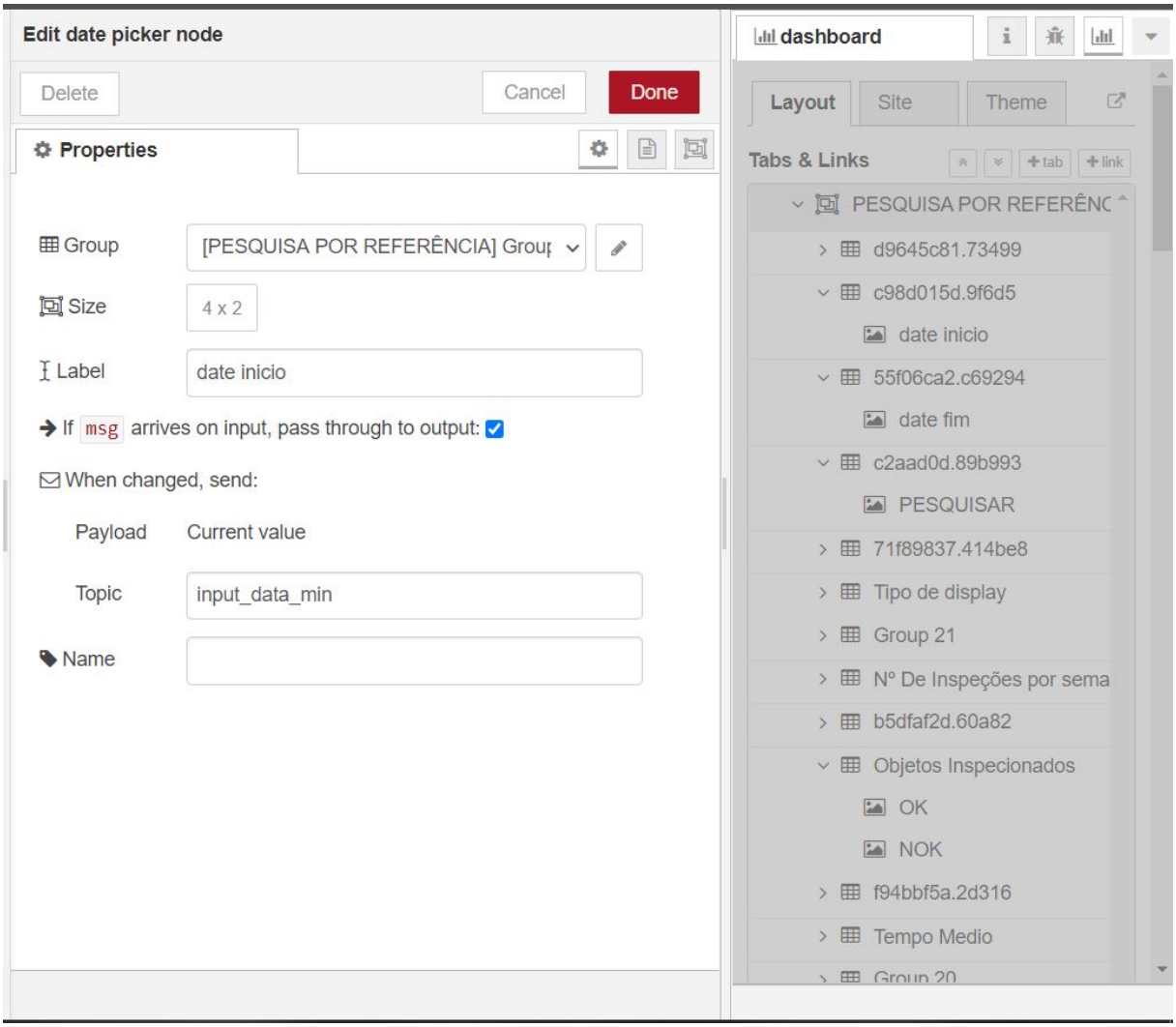

Figura 4.27: Configuração de um nó de entrada no dashboard com o valor definido pelo utilizador da data correspondente ao início do intervalo de tempo da pesquisa.

A forma de configurar cada botão, neste caso o de "PESQUISAR"está ilustrado na figura 4.28 e, de igual modo, foi agrupado o nó button ao grupo do dashboard, definido o

tamanho, atribuido o nome a mostrar no ambiente gráfico e a informação a transportar para o nó função que vai originar a pesquisa à base de dados.

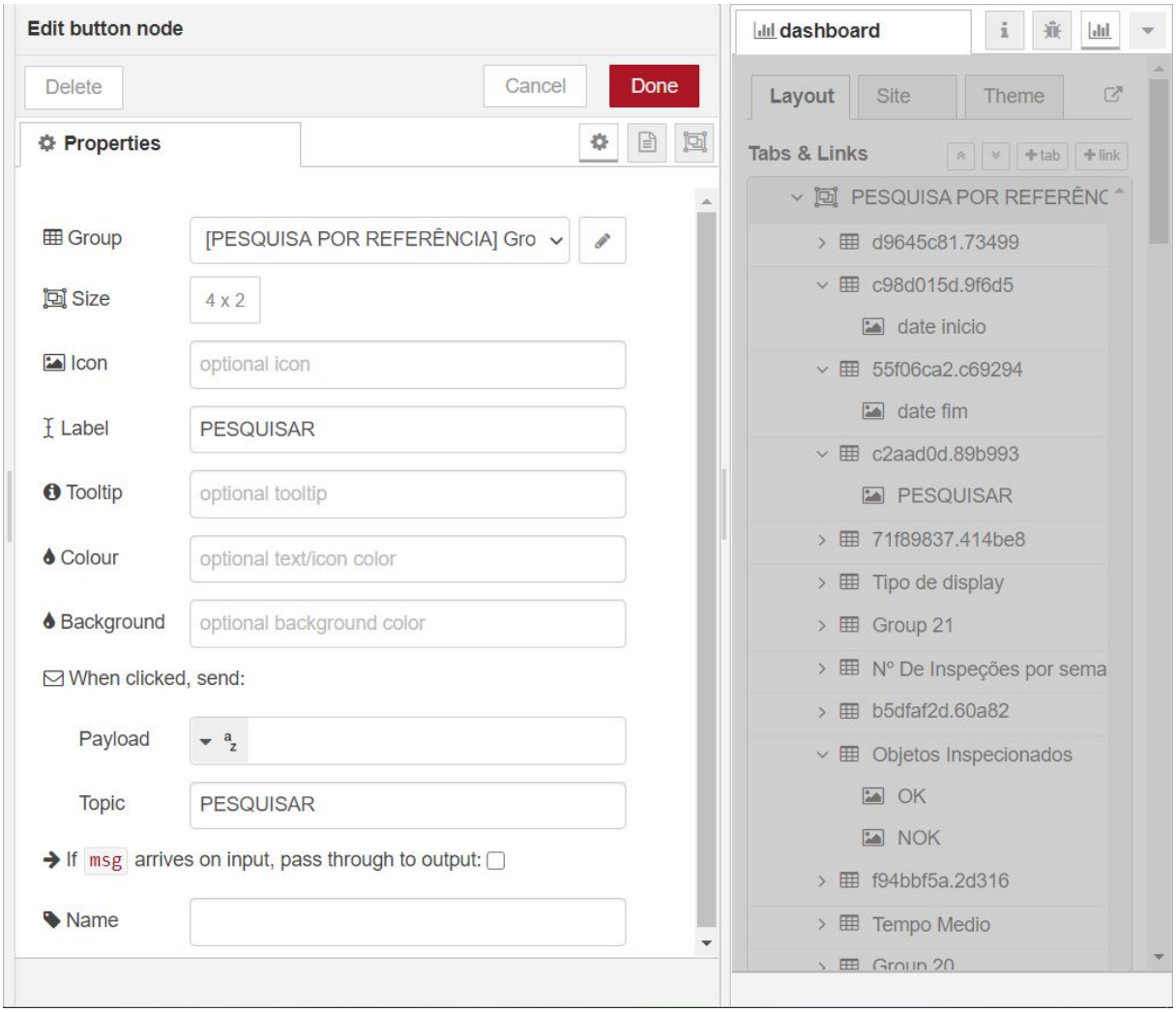

Figura 4.28: Configuração do botão PESQUISAR do dashboard.

O texto a ser visualizado no dashboard tem de ser configurado, agrupando o nó ao grupo correspondente, definido o nome do espaçamento e indicando o valor a ser mostrado. Pode ser um valor pré-definido ou, utilizando o comando "msg.payload", um valor recebido de outro nó.

Na figura 4.29 pode ver-se a configuração do nó pretencente ao dashboard que indica

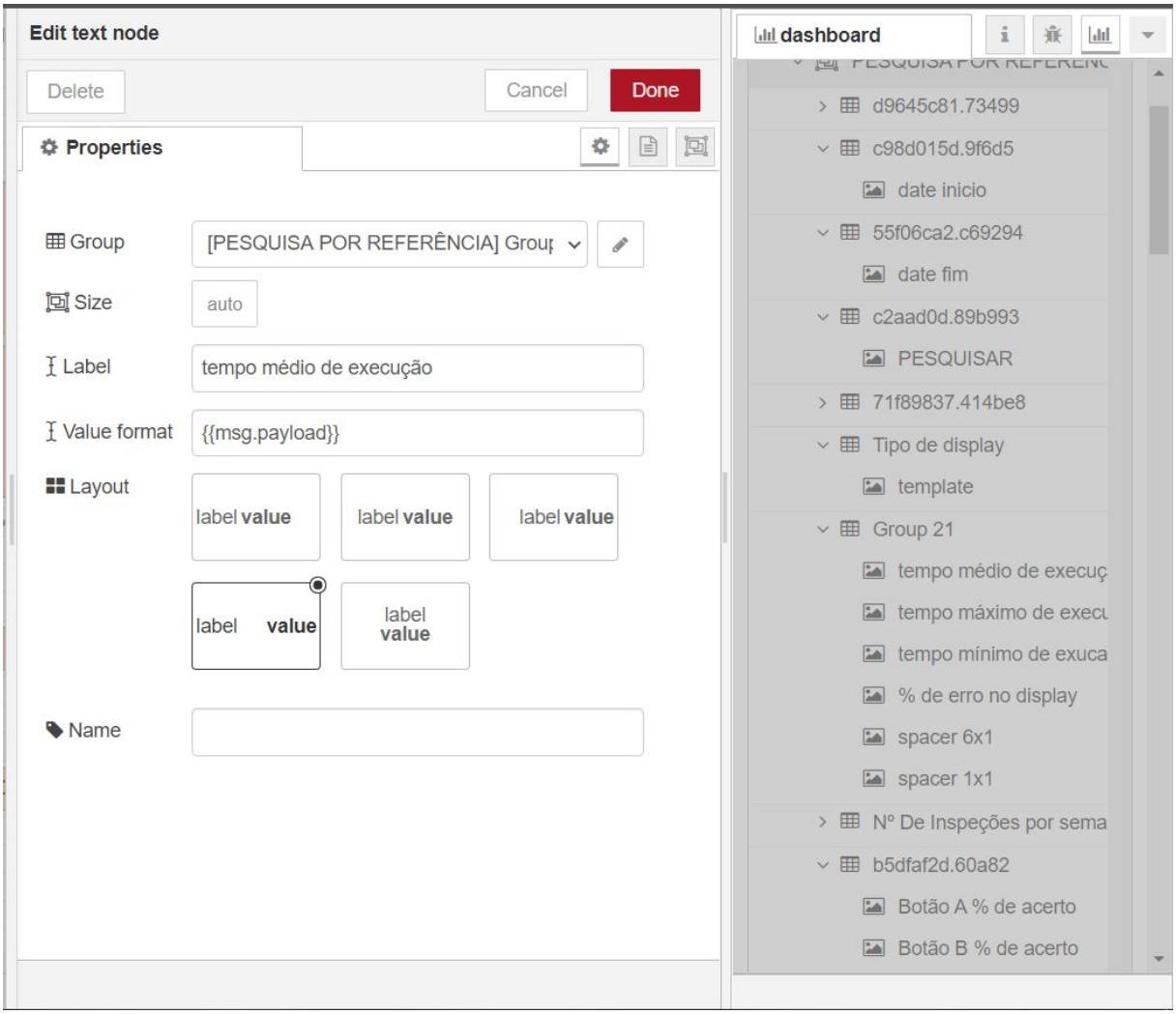

Figura 4.29: Configuração do nó que indica o tempo médio por inspeção de objeto.

o tempo médio de execução da inspeção aos objetos.

Uma vez que os gráficos também fazem parte do ambiente gráfico, é necessário definir a posição que ocupam entre os grupos existentes, o tamanho, o tipo de gráfico, o nome que o identifica, os dados que lhe vão dar origem, a escala, as cores, entre outros.

As figuras 4.30 e 4.31 mostram uma possível forma de configurar os gráficos de barras e do tipo piza, respectivamente. Desta forma a interface a ser utilizada encontra-se configurada e corretamente disposta entre si. O aspeto que lhe foi conferida durante esta implementação encontra-se no seguinte capítulo com a discussão dos resultados obtidos.

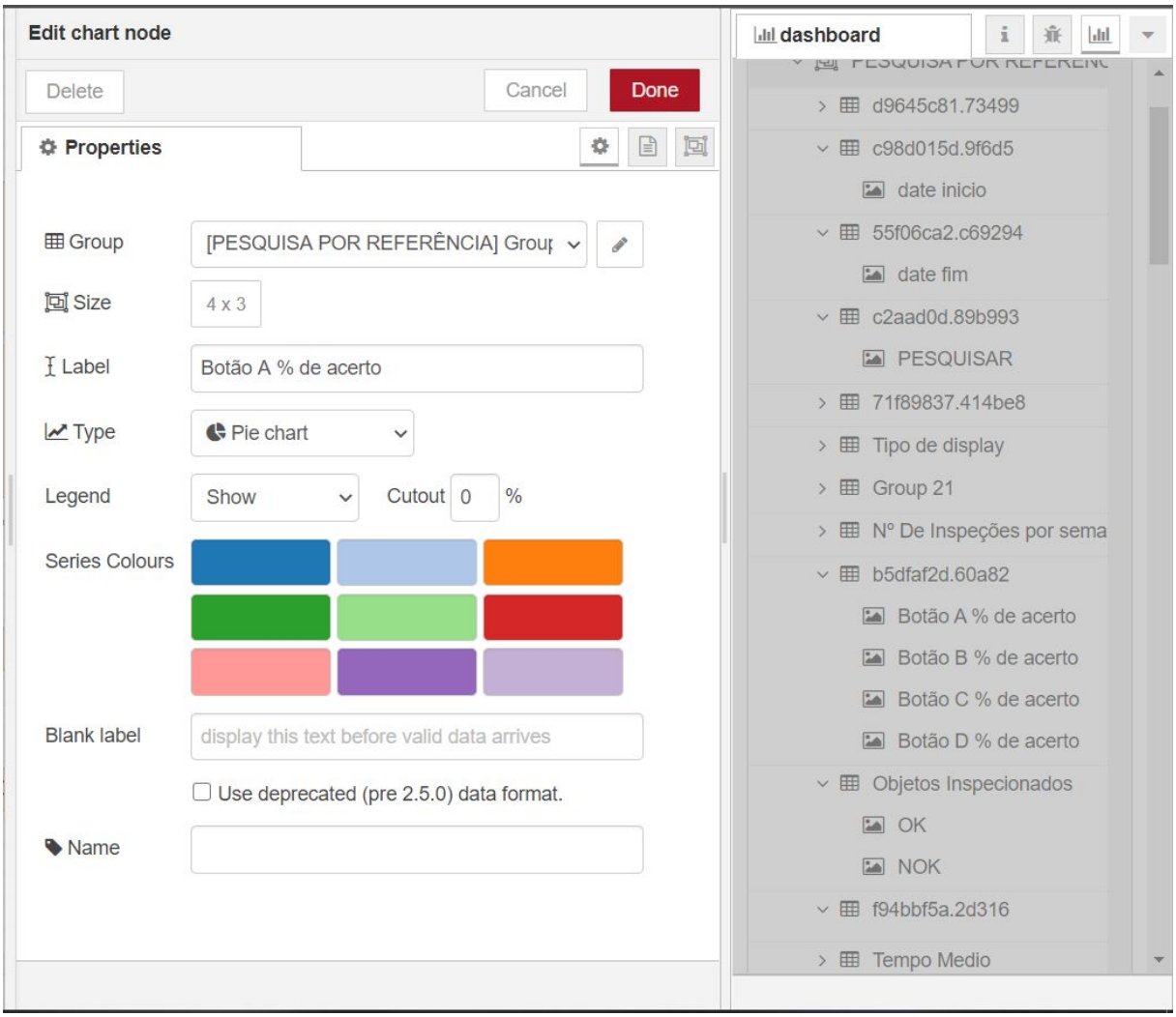

Figura 4.30: Configuração do gráfico tipo piza relativo ao estado do botão A dos objetos inspecionados.

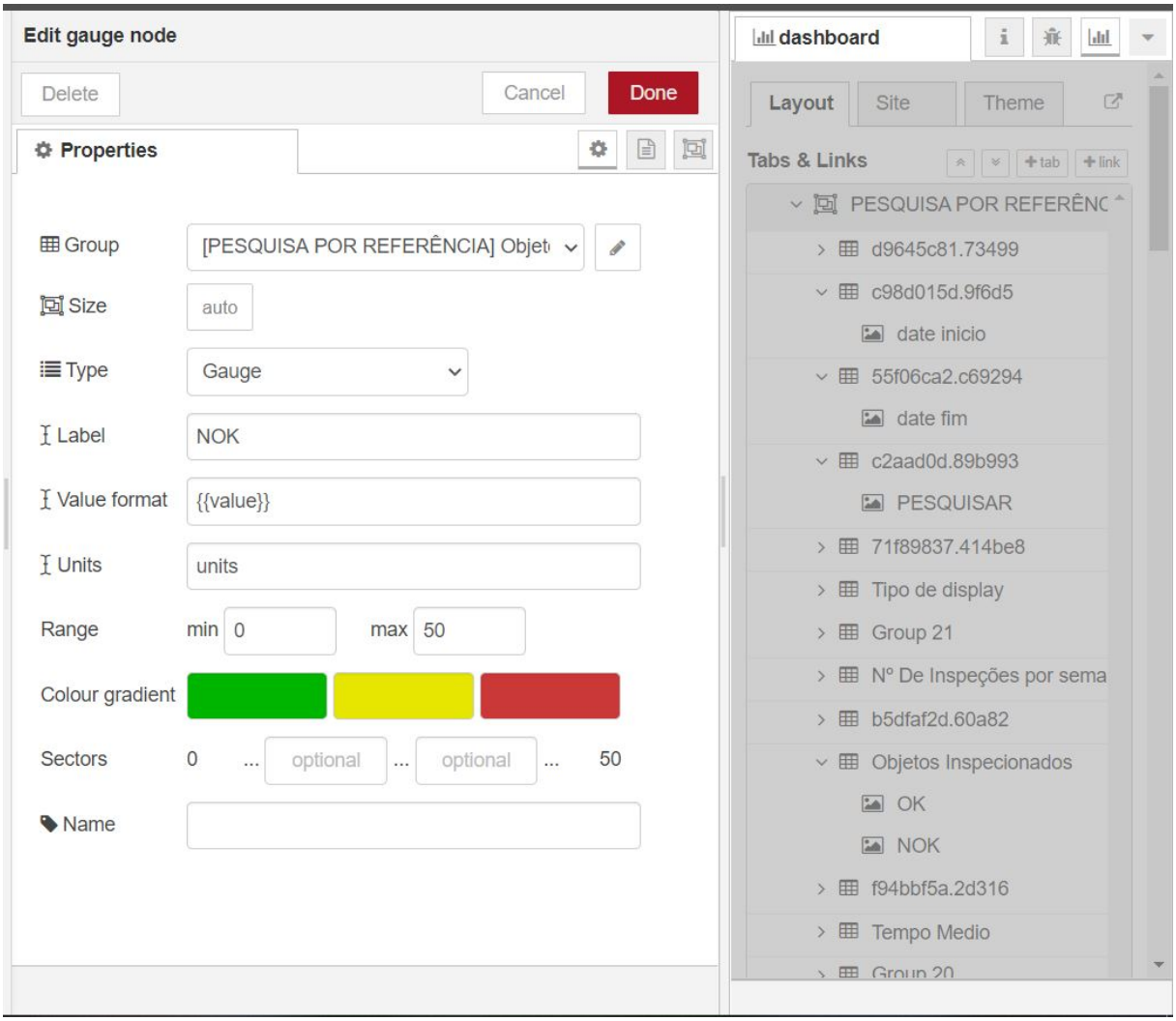

Figura 4.31: Gráfico Gauge do número de objetos NOK.

Capítulo 5

Testes Experimentais e Discussão de Resultados

5.1 Softwares Instalados para o Projeto Final

Para este projeto foram instalados, num computador com processador da marca Intel Core i7 - 8750H, com 16 Gigabytes de memória RAM e 4 Gigabytes de pláca gráfica dedicada aos softwares Sysmac Studio para a a programação do autómato, um sistema de gestão de dados proveniente do phpMyAdmin que é o MySQL, um editor de texto de JavaScript para editar o código do robô industrial - pycharm -, e ainda a aplicação Node-RED acoplada ao node.js para poder ser programada em estrutura web, ou seja, no browser. Dentro do Node-RED foram adicionadas bibliotecas, como a da base de dados e do Dashboard - ambiente gráfico para a aplicação final, acessível a qualquer utilizador.

A máquina onde foi desenvolvido o trabalho mostrou-se compatível com todas as necessidades, não tendo havido quaisquer problemas na configuração dos diferentes dispositivos.

5.2 Leitor de códigos QR

A leitura dos códigos de barra bidimensionais foi um sucesso, tanto a nível de software como de hardware, devido à utilização de uma câmara inteligente da marca Teledyne Dalsa, com resolução até 1280x960 e por possuir um software intuitivo próprio e de fácil acesso, utilizando o navegador do próprio computador, e contendo programas já pré-instalados específicos para a função pretendida. Aliado a esta tecnologia encontra-se o CPU Unit NJ101-9020 da Omron, que contém as funcionalidades de conectividade à base de dados mySQL e de comunicação TCP/IP, o que facilita muito o processamento de dados deste género, com elevada eficiência.

5.3 Aplicação web Node-RED

Tal como em todas as tecnologias implementadas neste trabalho, foi necessário investigar e aprender a configurar os softwares utilizados. A aplicação Node-RED foi aquela que requereu mais atenção e à qual mais tempo teve de ser dedicado. Mostrou-se ser uma aplicação muito versátil e capaz de superar o desafio proposto da criação duma aplicação capaz de fornecer ao utilizador de um processo de supervisão industrial, uma visão de rastreabilidade dos produtos inspecionados, mostrando o registo da quantidade de peças boas para continuarem na cadeia de produção, e por outro lado aquelas que não estariam em condições de seguir o percurso.

O flow de Node-RED mostrado por ser dividído em três partes principais:

- Entradas: Seleção das variáveis a pesquisar; Introdução Manual;
- Interface: De programação - Sítio localhost:1880; De Visualização - Sítio localhost:1880/ui;
- Saídas;

5.4 Ambiente gráfico do Node-RED - Dashboard

Neste trabalho de rastreabilidade, dentro do sistema de inspeção de displays foram desenvolvidas duas interfaces gráficas, para obtenção dos dados registados na base de dados e selecionados pelo utilizador. São designados por parâmetros de entrada as opções e valores escolhidos pelo utilizador na interface visual, por serem estes os dados que ditam a forma como serão feitas as pesquisas à base de dados. Caso a interface gráfica não seja prenchida por valores admissíveis, após premir o comando "PESQUISAR"não irá criar gráficos nem devolver valores. Os nós que integram cada flow já foram descritos no capítulo anterior, juntamente com as funcionalidades que conseguem realizar e a forma de configurar o ambiente gráfico.

5.5 Seleção das opções pelo utilizador e resultados obtidos

O utilizador tem ao seu dispor duas interfaces de pesquisa: a pesquisa por número de série, onde apenas tem de colocar o número que identifica o objeto, e a pesquisa por tipo de display, onde faz a escolha do tipo de objeto e define o intervalo de tempo. Em ambas as opções, é necessário pressionar o botão de pesquisa para validar a escolha feita, pois o Node-RED funciona através de acionamentos que provêm de comandos de entrada nas funções, os botões. A figura 5.1 corresponde à interface visual de pesquisa por escolha do número de série. Nesta janela apenas existe um sítio onde pode ser inserido o número de série do objeto. Pode servir para o caso de um objeto apresentar algum defeito após o processo de inspeção ter sido concluído, e o utilizador consiga saber informações relativas ao objeto, como a data e hora da verificação, valores que lhe estavam associados e, ainda, o operador responsável pelo lote de objetos. A título de exemplo, foi selecionado o objeto com o número "7889"que deu origem à informação disponível na figura 5.2.

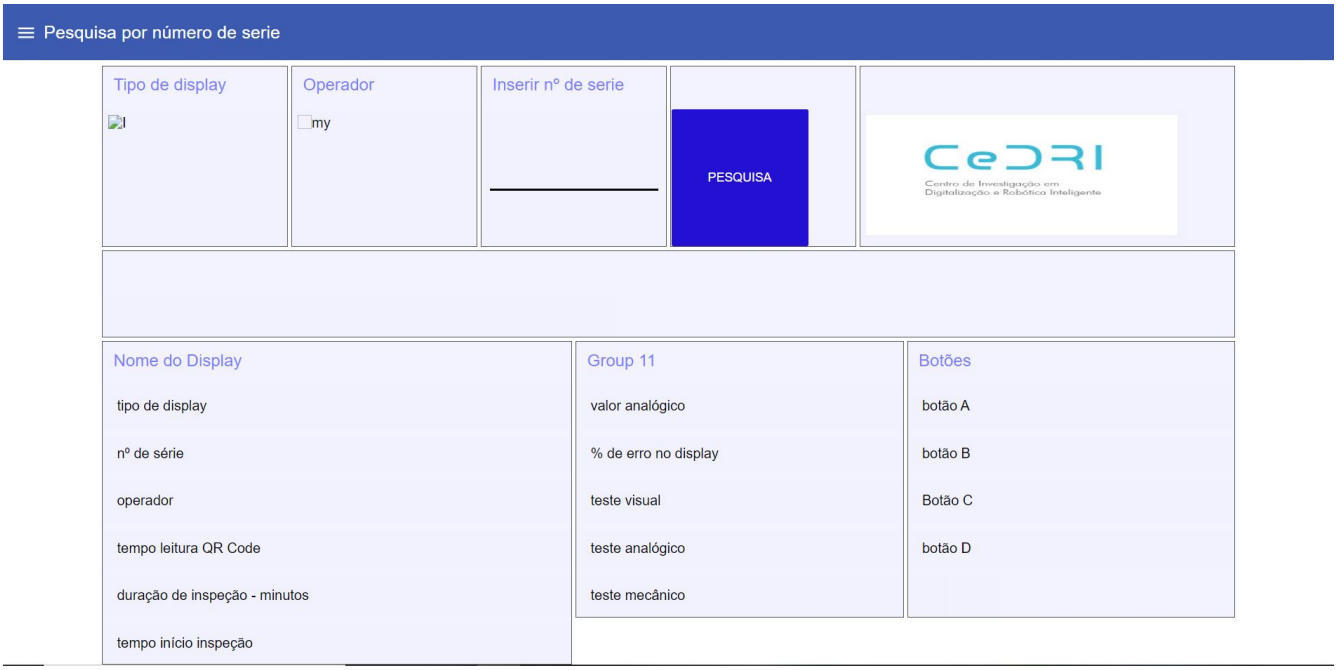

Figura 5.1: Interface visual da pesquisa por número de série/referência.

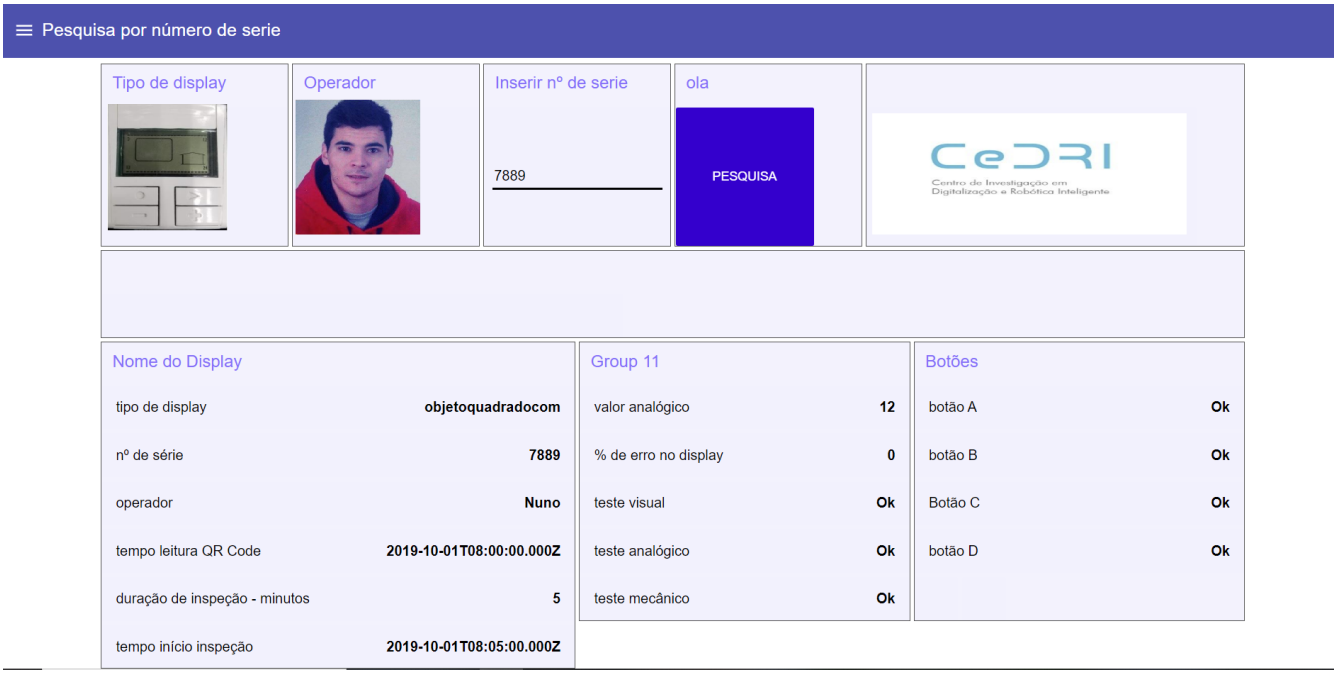

Figura 5.2: Informação retornada pela aplicação à pesquisa realizada pelo utilizador, relativa ao objeto com o número de série 7889.

Outra opção disponível ao utilizador é a de selecionar um tipo de objeto e definir um intervalo de tempo no qual terão decorrido as inspeções aos objetos na bancada industrial. O layout com que se depara o utilizador, ao utilizar esta ferramenta, está presente na figura 5.3. Após ter sido selecionado o objeto do tipo "objetoquadradocom"compreendido no intervalo de 28 de Setembro de 2019 a 17 de Outubro de 2019, e pressionado o botão "PESQUISAR", obtiveram-se os resultados presentes na figura 5.4.

Figura 5.3: Interface visual da opção de pesquisa por escolha de tipo de objeto e definido o intervalo de tempo.

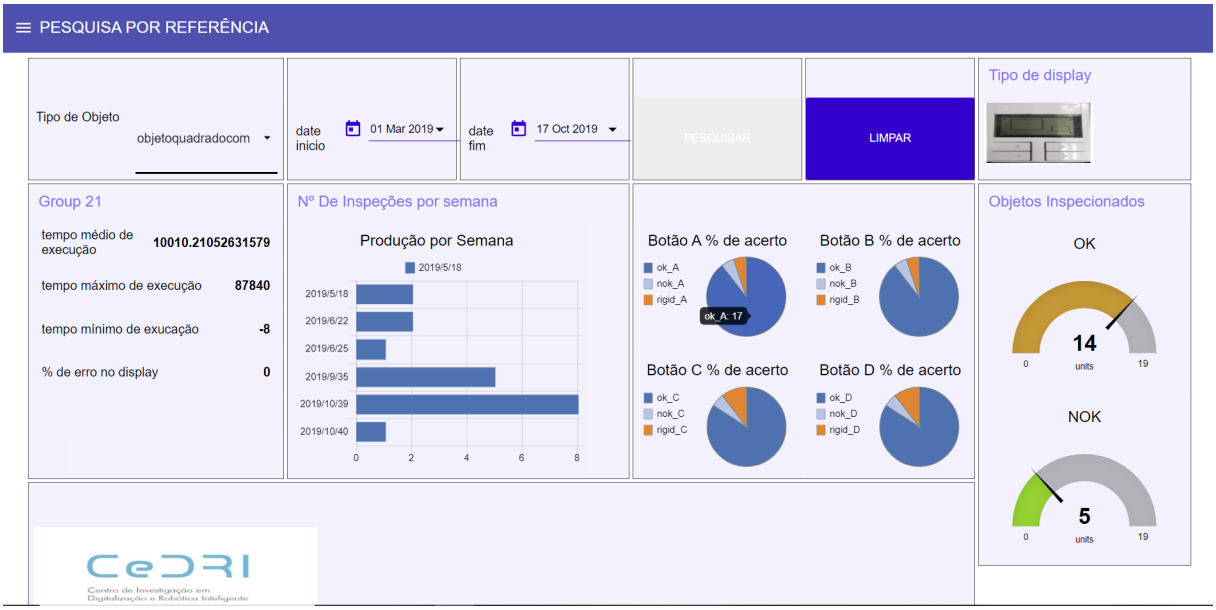

Figura 5.4: Informação retornada pela aplicação à pesquisa por tipo de objeto.

5.6 Avaliação dos resultados obtidos através das pesquisas na interface disponível para o utilizador

É possível concluir que os resultados foram os esperados, pois correspondem exatamente às informações contidas na tabela da base de dados na qual foram inseridas todas as informações no decorrer do processo de inspeção de objetos, provenientes da câmara de processamento de imagem acoplada ao autómato, bem como da consola HMI e do computador industrial, com dados provenientes da tarefa realizada pelo robô colaborativo. Os gráficos mostraram capacidade de se adaptar quando possuem mais e menos dados.

Para finalizar, poderiam ter sido criadas mais interfaces gráficas para o utilizador, nomeadamente uma que fizesse alusão aos objetos no decorrer da inspeção. No entanto, como ainda não constam mais objetos para o robô inspecionar, tal não foi realizado, podendo vir a ser implementado em trabalhos futuros. Também será possível adicionar funcionalidades como alertas, quando um objeto não esteja conforme o pretendido, podendo ser enviados através de aplicações em redes sociais como o Twitter; este factor poderia ter interesse, já que representaria uma produção monotorizada ao instante.

Capítulo 6

Conclusões e Trabalho Futuro

Abraçando o desafio com conhecimentos limitados em Java (Node-RED) e Python (JetBrains), foi possível dominá-los o suficiente para desenvolver o sistema de rastreabilidade de peças que agora se encontra disponível a qualquer utilizador, presente na estação de inspeção de displays. O desenvolvimento desta aplicação ajudou a perceber alguns dos problemas enfrentados pela indústria em relação à monotorização da produção em tempo real, ajudando de igual forma a responder a todas as declarações de problemas anteriormente detetados.

A primeira ideia para este desafio era o de ajudar a controlar a inspeção de peças, como se de uma produção em série se tratasse, de forma a identificar de forma rápida e eficaz objetos defeituosos. A resposta é que sim, com a aplicação desenvolvida pelo autor, a partir da combinação de diferentes tecnologias, é possível cumprir o objetivo de identificar peças defeituosas e obter dados históricos da produção. O tempo médio de 5 segundos por inspeção não deve ser levado a sério, pois a estação de inspeção apenas realiza testes pontuais mas, com todas as ferramentas desenvolvidas facilmente poderá ser posta em infinito funcionamento. Uma vez iniciados os programas, de gestão de base de dados e da aplicação, os dados serão registados de forma autónoma libertando tempo da equipa que se encontre a gerir todo o processo, para que a mesma se possa concentrar noutras tarefas, em vez de realizar a ação de registar valores. Este sistema pode ser considerado uma mais valia, por reduzir os erros no registo de peças, já que além de ser extremamente rápido

a transportar a informação, os dados nunca poderão ser mal interpretados, como poderia acontecer com pessoas, devido à escrita e registo menos rigorosos ou às dificuldades na leitura desses valores.

Até agora, o sistema não apresentou erros de impressão ou erros que levaram à interpretação incorreta dos dados recebidos das máquinas. É altamente improvável que a informação proveniente do autómato ou do robô, em algum ponto do caminho, forneça outro valor que não seja o recebido pela câmara e pela garra de push, pois os dados são encaminhados e não são adulterados. No entanto, o que poderia levar a um erro é o facto de, por algum motivo, o sistema parar de atualizar ou no caso de receber dados em duplicado, pois as amostras iriam aumentar mas os resultados não, facto que poderia conduzir a erros estatísticos.

A aplicação mostra-se fiável no registo dos dados e pode servir para retirar trabalho moroso e minucioso a um operador, que ganha desta forma tempo e autonomia para geir outros assuntos, ou seja, enriquece a cadeia de inspeção e o sistema de rastreabilidade de produtos.

6.1 Trabalho futuro

Medidas simples podem ser implementadas para impedir a ocorrência de falhas, exibindo um aviso quando a conexão acontece. Seria interessante desenvolver um sistema de reiniciamento que seja ativado quando algum tipo de falha ocorre, ficando registado o tipo de falha, a causa que lhe foi associada e data e hora da mesma. Na programação do autómato pode ser aprofundada a automatização do processo, ajustando flags de erros nos blocos, por exemplo.

No Node-red, mais serviços devem ser desenvolvidos para permitir um nível ainda mais alto de flexibilidade. Pode tratar-se de qualquer tipo de operação, desde nós pessoais em que as informações de um conjunto de máquinas são adicionadas por padrão, até um nó que permita que os dispositivos se conectem usando outras formas de comunicação, como Wi-Fi. Este é um sistema diversificado, visto que é composto por muitos componentes

distintos. Por esse motivo, esses exemplos mencionados aqui são apenas sugestões recomendadas. Com um pouco de imaginação, provavelmente é possível criar muito mais recursos que poderão ser adicionados.

Podem ainda ser usadas estas informações armazenadas para elaborar previsões de tempos de produção de determinados produtos, e também para escolher as posições de máquinas e robôs, a fim de tornar o processo ainda mais rápido e fiável aquando da adição de novas tecnologias, iguais ou idênticas às existentes.

Capítulo 7

Anexo I

- MySQL

Data type category	Data type in DB	Data type in NJ/NX-series Controllers
Numbers*1	BIT	BOOL
	BOOL BOOLEAN	BOOL
	TINYINT	SINT USINT
	SMALLINT	INT UINT
	MEDIUMINT	DINT UDINT
	INT	DINT UDINT
	BIGINT	LINT ULINT TIME
		*2
	DECIMAL(1)	BOOL
	DECIMAL(3)	SINT
	DECIMAL(5)	INT
	DECIMAL(10)	DINT
	DECIMAL(20)	LINT
	DECIMAL(3)	USINT
	DECIMAL(5)	UINT
	DECIMAL(10)	UDINT
	DECIMAL(20)	ULINT
	DECIMAL(20)	TIME
	FLOAT	REAL
	DOUBLE	LREAL
Date and time	DATE	DATE
	DATETIME	DATE_AND_TIME
	TIMESTAMP	DATE_AND_TIME
	TIME	TIME_OF_DAY
String	CHAR	STRING*3
	VARCHAR	STRING*3
	TINYTEXT	STRING*3
	TEXT	STRING*3
	MEDIUMTEXT	STRING*3
	LONGTEXT	STRING*3
Binary	BINARY	None
	VARBINARY	None
	TINYBLOB	None
	BLOB	None
	MEDIUMBLOB	None
	LONGBLOB	None
Others	ENUM	None
	YEAR	None
	SET	None

NJ/NX-series Database Connection CPU Units User's Manual (W527)

Figura 7.1: Variáveis Compatíveis entre o Sysmac Studio e o MySQL base de dados Fonte: seção “3.2 Creating a Structure Data Type”, manual Omron "NJ/NX-series Database Connection CPU Units”.

Bibliografia

[1] D. Zuehlke, "Smartfactory—towards a factory-of-things", mar. de 2010.

[2] Y. Liao, E. Loures e F. Deschamps, "Industrial internet of things: A systematic literature review and insights", mar. de 2018.

[3] *"industry 4.0." in*, https://en.wikipedia.org/wiki/Industry_4.0, cit. on pp. 15, 16 de 2016.

[4] M. Peris-Ortiz, J. Álvarez-García e C. Rueda-Armengot, "Achieving competitive advantage through quality management", Fev de 2015.

[5] M. Groover, "Automated assembly system, automation, production systems, and computer-integrated manufacturing", 2007.

[6] Q. Lu e X. Xu, "Adaptable blockchain-based systems: A case study for product traceability", nov. de 2017.

[7] P. Gerbert, M. Lorenz e M. Harnisch., *"industry 4.0: The future of productivity and growth in manufacturing industries."*, https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.aspx, Abril, 2015 de pag 16, 17.

[8] S. Hartmann, *Dhl and accenture unlock the power of blockchain in logistics*, https://newsroom.accenture.com/news/dhl-and-accenture-unlock-the-power-of-blockchain-in-logistics.htm, mar. de 2018.

[9] Accenture, *Dhl and accenture unlock the power of blockchain in logistics*, https://newsroom.accenture.com/news/dhl-and-accenture-unlock-the-power-of-blockchain-in-logistics.htm, dez. de 2018.

[10] B. Marr, *How big data drives success at rolls-royce*, https://www.forbes.com/sites/bernardmarr/2015/06/01/how-big-data-drives-success-at-rolls-royce/#3230e6a31d69, jan. de 2015.

[11] R. Mayr, "Industry 4.0 and the importance of product traceability", nov. de 2018.

[12] O. Corporation, *About mysql*, http://www.mysql.com/about/, mai. de 2016.

[13] O. Corporation, *Machine automation controller nj-nx-series database connection cpu units*, 1st. Kyoto, JAPAN: Regional Headquarters OMRON EUROPE B.V., 2012, ISBN: 123456789.

[14] M. Sheng, Y. Qin, L. Yao e B. Benatallah, *Managing the web of things linking the real world to the web*, 1st. Burlington, Massachusetts, EUA: Morgan Kaufmann, 2017, ISBN: 123456789.

[15] NODE-RED, *Documentação*, https://nodered.org/docs/, jan. de 2020.

[16] V. Luís, *"machine learning applied to an intelligent and adaptive robotic inspection station"*, 2019 de cit. pag 24.

[17] M. P. de Albuquerque e M. P. de Albuquerque, "Processamento de imagens: Métodos e análises", *Centro Brasileiro de Pesquisas Físicas – CBPF/MCT*,

[18] GUINARD e TRIFA, *Node-red*, 1st. 20167, ISBN: 123456789.

[19] R. P. Foundation, *Raspberry pi model a+*, https://www.Raspberrypi.org/blog/Raspberry-pi-model-a-plus-on-sale/, 2019.

[20] NODE-RED, *The telegraph. mini raspberry pi computer goes on sale for £22*, http://www.telegraph.co.uk/technology/news/9112841/Mini-Raspberry-Pi-computer-goes-on-sale-for-22.html, 25/05 de 2019.

[21] Node-RED, *About node-red*, https://nodered.org/about/, 2016.