SINALIZAÇÃO FERROVIÁRIA E ANÁLISE DE DADOS: UMA ABORDAGEM PARA MANUTENÇÃO PREDITIVA

MORAES, Leonardo de[1]
PICCININI, Marco Aurélio[2]
ABRITTA, Camila do Carmo Almeida[3]
TEIXEIRA, Wesley Carminati[4]

Linha de pesquisa: Automação

RESUMO

A manutenção preditiva é uma estratégia crucial para a gestão de ativos em setores que demandam alta confiabilidade, como o ferroviário. Este artigo traz inicialmente uma revisão de literatura sobre diversos conceitos atrelados à manutenção preditiva, e posteriormente propõe métodos de desenvolvimento de sistemas preditivos no contexto da sinalização ferroviária, com ênfase em soluções automatizadas de coleta e análise de dados gerados por controladores lógicos. Por meio da análise de registros operacionais, busca-se antecipar falhas e reduzir intervenções corretivas não planejadas. O estudo detalha o desenvolvimento softwares aplicativos que otimizam o processo de tratamento de dados, possibilitando a construção de uma base robusta que, futuramente, poderá servir como insumo para a aplicação de técnicas de inteligência artificial na previsão de falhas. A iniciativa destaca os benefícios da integração entre automação e manutenção preditiva, como aumento da eficiência operacional e redução de custos com falhas.

Palavras-chave: Manutenção Preditiva. Sinalização Ferroviária. Automação de Processos. Análise de Dados.

[1] Graduando em Engenharia Elétrica pelo Centro Universitário Academia - UniAcademia
[2] Professor do curso de Engenharia Elétrica do Centro Universitário Academia - UniAcademia.
[3] Professora do curso de Engenharia Elétrica do Centro Universitário Academia - UniAcademia.
[4] Professor do curso de Engenharia Elétrica do Centro Universitário Academia - UniAcademia.

1 INTRODUÇÃO

A manutenção preditiva é considerada uma das estratégias mais eficientes na gestão de ativos, especialmente em setores que demandam alta confiabilidade e disponibilidade destes, como ocorre no setor ferroviário. Esse modelo de manutenção visa prever falhas por meio da análise de alarmes e dados históricos referentes à operação de equipamentos, diferentemente da manutenção preventiva tradicional baseada em ciclos de funcionamento. Justamente por conta disso, este método de manutenção torna-se mais eficiente, uma vez que o número de intervenções da manutenção sobre os dispositivos é reduzido. Essa abordagem contrasta ainda com a manutenção corretiva, que é realizada quando ocorre a falha e/ou o defeito de determinado componente.

No contexto ferroviário, entende-se por sinalização um sistema complexo de equipamentos eletroeletrônicos interligados e intertravados, que juntos, desempenham um papel crucial na coordenação e controle da movimentação de trens com eficiência e segurança. Exemplos de componentes deste sistema são desde os sinaleiros, que através de diferentes aspectos de cor indicam ao maquinista a condição da via a sua frente, até os controladores lógicos programáveis (CLP's) que monitoram, controlam e registram operações de diversos componentes, inclusive dos próprios sinaleiros.

Neste contexto, o trabalho seguirá a linha de pesquisa de "Automação" e "Controle", e não apenas documenta o desenvolvimento e implementação de um software em específico, mas também explora a aplicação de métodos de manutenção preditiva em sistemas eletrônicos ferroviários através da análise de dados. Ao ditar os passos sobre como estruturar uma base de dados robusta, este projeto abre caminho para futuras aplicações de inteligência artificial (IA), que poderão aprender com os dados devidamente tratados e estruturados, possibilitando métodos cada vez mais eficientes e confiáveis para previsão de falhas.

Por fim, essa iniciativa busca destacar a importância e as vantagens de se adotar uma visão abrangente dos sistemas integrados à manutenção preditiva, aumentando a eficiência operacional e reduzindo os custos relacionados a falhas e interrupções indevidas do sistema.

1.1 MOTIVAÇÃO

Com o avanço da tecnologia, os componentes da sinalização ferroviária tornam-se cada vez mais sofisticados, tornando-se capazes de gerar registros com grande riqueza de informação. Estes controladores geralmente possuem a capacidade de gerar registros (*logs*) detalhados de suas operações, arquivos que tradicionalmente são utilizados para analisar falhas e defeitos ocorridos em algum dos componentes controlados por estes CLP's.

Dada a grande quantidade de informações geradas pelos equipamentos, surge a demanda/necessidade de utilizar esses dados não apenas para investigar anomalias já ocorridas, mas sim para antever possíveis defeitos, aprimorando assim o processo de análise de dados hoje existente, e implementando métodos de manutenção preditiva ao modelo tradicional. Contudo, devido a grande quantidade destes equipamentos presentes em uma ferrovia, e consequentemente um volume excessivamente grande de dados, torna-se inviável que essa análise de dados seja realizada de forma manual.

Surgiu então uma dor latente no setor: existe uma grande quantidade de dados sendo gerados pelos equipamentos que não são aproveitados para análises preditivas, tornando imprescindível o emprego de técnicas de automação, e foi a partir dessa carência observada no ambiente de trabalho real que nasceu o desafio de desenvolver soluções capazes de automatizar os processos de coleta, armazenamento e análise de dados dos CLP's que compõem os sistemas de sinalização ferroviária. Deste modo, este desenvolvimento tem como objetivo elaborar um método de utilização destes dados operacionais de forma preditiva, identificando através de um software de análise automática, desvios e anomalias nos equipamentos de sinalização antes que estes apresentem falhas, reduzindo assim, interrupções indevidas no tráfego ferroviário ocasionadas por falhas nos equipamentos da sinalização de uma ferrovia.

1.2 ORGANIZAÇÃO E ESTRUTURAÇÃO

Na primeira seção é apresentada a introdução, motivação e a estrutura do trabalho.

Na segunda é apresentado todo o referencial teórico do trabalho, dividido em subtópicos para cada uma das técnicas e conceitos que serão aplicados durante o desenvolvimento prático. Todas as informações nesta seção são provenientes de fontes literárias existentes.

Na terceira é descrito detalhadamente o processo prático para o desenvolvimento de softwares automatizados para análises preditivas, que podem ser aplicados a diferentes equipamentos.

Na quarta seção são levantados e evidenciados o potencial das ferramentas e métodos desenvolvidos, através de dados reais de falhas ocorridas em ambiente empresarial que poderiam ter sido evitadas com a aplicação dos softwares desenvolvidos.

Na quinta seção, o presente trabalho é concluído, fornecendo um panorama que sintetiza os principais pontos discutidos ao longo de sua elaboração.

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentados e discutidos os conceitos iniciais, teorias e modelos que embasam a presente pesquisa. Esta etapa é fundamental para fornecer uma compreensão abrangente e fundamentada em bases acadêmicas dos assuntos que serão abordados no desenvolvimento do trabalho. A compreensão desses conceitos é essencial para contextualizar as abordagens que serão propostas.

Serão explicitados conceitos relacionados à manutenção preditiva, bem como suas diferentes abordagens em relação a manutenções corretivas e preventivas. Serão discutidos modelos teóricos que sustentam as práticas empregadas, incluindo metodologias e técnicas. Haverá ainda uma introdução sobre a sinalização ferroviária, o papel dos controladores lógicos e sua relevância na sinalização moderna.

Após a realizada a contextualização básica, serão abordados tópicos referentes a diferentes técnicas de programação, armazenamento e processamento de dados, entre outros, que são essenciais para a implementação de soluções automatizadas.

A revisão de literatura, bem como a apresentação dos modelos teóricos, tem como objetivo proporcionar uma base uniforme ao leitor para maior compreensão dos problemas e desafios a serem investigados, além de proporcionar uma visão crítica

das soluções propostas. Esta revisão ajudará a identificar lacunas na pesquisa existente, bem como justificar os métodos utilizados no desenrolar do trabalho.

2.1 MANUTENÇÃO PREDITIVA

A manutenção preditiva é hoje uma estratégia essencial na gestão de ativos, visando maximizar a eficiência operacional e reduzir custos associados a quebras inesperadas. De acordo com Neponuceno (1989), a finalidade da manutenção é conservar um equipamento ou sistema em plenas condições de funcionamento com o menor custo possível, minimizando sua deterioração. Ao associar a redução de custos operacionais à menor frequência de paradas, é evidente que a manutenção busca minimizar as perdas financeiras causadas por interrupções, tanto planejadas quanto não planejadas.

A manutenção corretiva é aquela voltada diretamente ao restabelecimento de um ativo que apresenta falha, “Uma falha é qualquer enguiço num sistema ou circuito que permanece até que sejam tomadas providências corretivas” (Neponuceno, 1989). Em contraste a corretiva, a manutenção preventiva segundo Gregório (2018) é aquela realizada visando justamente evitar o aparecimento da falha em um ativo e/ou sistema, podendo ser sistemática, ou seja, baseada em ciclos e intervalos de tempo pré-definidos (por exemplo, a troca de óleo de um motor de veículo, que pode ocorrer anualmente ou a cada certa quantidade de quilômetros percorridos) ou mesmo por oportunidade, aproveitando possibilidades de intervalos concedidos pelas equipes de operação (quando por algum motivo a área de operação deixa de demandar determinado equipamento de forma não planejada, a manutenção pode usar este tempo para atuar sobre o ativo).

Segundo Kardec (1998) a manutenção preditiva surge como uma grande quebra de paradigma na manutenção em geral, citando sua imensa capacidade de evolução no decorrer dos avanços em conhecimento tecnológicos. Essa abordagem se diferencia dos métodos tradicionais de manutenção, como a preventiva e a corretiva, mencionando ainda sobre a possibilidade de preparação/planejamento de serviços de forma antecipada, o que de certa forma torna esse tipo de manutenção algo próximo a manutenção corretiva planejada.

Esse modelo de manutenção consiste no “monitoramento de um ou mais parâmetros de um item com objetivo de realizar a troca de óleo de um motor de veículo, que pode ocorrer anualmente ou a cada certa quantidade de quilômetros percorridos” (Gregório, 2018). Em linhas gerais, esse método se faz extremamente útil em setores que exigem alta confiabilidade de seus ativos e têm restrições quanto ao tempo de inatividade, como é o caso do setor ferroviário.

Ainda de acordo com Kardec (1998), existem alguns requisitos que devem ser atendidos para adoção de técnicas preditivas, como: equipamento/sistema deve possuir capacidade de monitoramento e/ou medição, deve existir uma relação de custo/benefício entre as paradas que se deseja predizer e o custo de implementação dessas técnicas, as falhas devem ser oriundas de causas que possam ser medidas e que tenham progressão temporal etc. Ou seja, deve existir uma análise minuciosa sobre o equipamento, seus modos de falhas e custos envolvidos no processo para garantir um processo de preditiva eficiente.

Adotar a conceitos de manutenção preditiva permite otimizar os processos e reduzir significativamente o tempo de inatividade não planejado, melhorando a eficiência geral e a confiabilidade dos ativos.

2.2 MANUTENÇÃO CENTRADA NA CONFIABILIDADE

A confiabilidade de um ativo “está associada à operação bem-sucedida de um produto ou sistema, na ausência de quebras ou falhas” (Fogliato, 2009). Ou seja, refere-se à capacidade de um determinado equipamento, máquina ou sistema em realizar sua função de forma constante e sem falhas durante determinado período de tempo. Em termos matemáticos, é a probabilidade de o ativo funcionar de maneira adequada durante determinado período. Preservar a confiabilidade de um ativo é crucial para maximizar seu tempo de operação, e consequentemente, minimizar a quantidade de interrupções. Uma alta taxa de confiabilidade implica em um menor tempo de indisponibilidade do ativo, e consequentemente menos custos com reparos e maior lucratividade de uma operação.

A Manutenção Centrada na Confiabilidade (MCC), segundo Gregório (2018) surge como uma evolução do modelo tradicional de manutenção, iniciando com a análise de funções e falhas, onde basicamente são identificadas as funções críticas

(primárias) de determinado ativo, e como falhas nessas funções podem impactar a operação e/ou segurança de um determinado sistema/planta, as falhas por sua vez são classificadas em modos de falha e analisadas, é necessário compreender as causas e consequências das falhas para traçar estratégias eficientes. A terceira etapa do processo é justamente desenvolver estratégias de manutenção que previnam as possíveis falhas identificadas, através de implementações de métodos de manutenção preditiva e preventiva.

De acordo com Siqueira (2005), a MCC trata-se de um método bem estruturado para elencar uma sequência de passos a serem seguidos a fim de garantir os resultados de manutenção desejados, resultados esses que se alinham ao conceito primordial da MCC: garantir a aplicação da manutenção mais adequada para cada modo de falha em diferentes equipamentos. Dentre os benefícios desse modelo de manutenção, destaca-se principalmente a otimização de recursos tanto materiais quanto humanos, uma vez que a manutenção é bem dimensionada e direcionada para onde estes serão mais necessários e eficazes. Como consequência, existe uma melhoria na disponibilidade, uma vez que o tempo de operação do ativo tende a aumentar, e uma redução de custos referentes a interrupções indevidas.

2.3 AUTOMAÇÃO ROBÓTICA DE PROCESSOS

Conforme os estudos realizados por Filippo Filho (2014) a automação em linhas gerais busca reduzir as necessidades de intervenção humana no controle de processos, visando garantir ganhos relacionados à produtividade, padronização, qualidade, custo, etc. Ainda segundo Filippo Filho (2014), pode-se classificar a automação em dois modelos primários, sendo eles: a automação com função de produção que trata da automação de equipamentos e processos ligados diretamente à uma cadeia de produção, ou seja, automação de equipamentos instalados em determinada planta; automação de suporte, implementada geralmente com o uso de softwares visando reduzir a ação humana em tarefas repetitivas, a “Automação Robótica de Processos” (ARP) se enquadra nesta segunda classificação.

A ARP permite que tarefas repetitivas sejam realizadas de forma mais rápida e eficaz, ou seja, minimizando erros frequentes associados a tarefas repetitivas e manuais. Uma das aplicações mais comuns de ARP é a que se dá através da

implementação de lógicas e algoritmos que geralmente buscam emular um processo humano de realização de atividades específicas. Esse tipo de automação é descrito por Mamede (2021) como automação assistida, que é um software desenvolvido visando automatizar aplicações em execução no desktop de um operador humano. De forma geral, toda ARP assistida é única para determinado processo e/ou aplicação. Dentre as vantagens da aplicação dessas técnicas, destaca-se a redução da necessidade de intervenção humana em tarefas rotineiras, gerando resultados com maior velocidade e consistência (uma vez que o mesmo processo é realizado sempre da mesma forma)

Porém, é importante ressaltar que antes de se buscar implementar uma automatização de um processo, deve-se garantir que ele esteja devidamente otimizado, automatizar um processo mal otimizado, ou ainda mal desenhado, irá amplificar problemas já existentes, ocasionando em falhas sistêmicas, erros repetitivos, travamentos etc. Com isso, a revisão do fluxo de trabalho deve proceder o processo de automação.

Na atualidade, as técnicas de ARP são cada vez mais bem vistas pelas empresas e por seus funcionários de forma geral, conforme citado por Mamede (2021) em sua obra “Automação de Processos com RPA”, existe hoje o entendimento geral de que a aplicação dessas ferramentas não implica na substituição de pessoas por máquinas, e sim na possibilidade de libertarem os funcionários para inovar e buscarem novas oportunidades de negócios e inovações.

A ARP é comumente empregada em processos repetitivos dos mais variados, como: preenchimento de formulários, captura e raspagem de dados de sites web, tratativas de grandes volumes de dados, dentre outros. Muniz et al. (2022) descreve o processo de raspagem de dados em tela (*screen scraping*) como um passo essencial para criação de RPA. Sobre esse método, em contrapartida às vantagens relacionadas a ganho de produção por substituir um trabalho outrora manual, o autor destaca a dificuldade de se implementar esse tipo de automação, uma vez que depende de conhecimentos específicos dos profissionais da determinada área de negócio em que se deseja implantar, logo, existe uma dificuldade deste profissional em repassar o processo para o programador para que ele realize a automação.

Por fim, as técnicas de ARP não apenas diminuem os tempos de execução de tarefas, contribuem ainda com a redução de custos operacionais, uma vez que elimina o tempo útil gasto por funcionários na realização de tarefas simples/repetitivas. Representa ainda uma melhora na segurança da informação, pois diminui o contato humano com os dados extraídos, diminuindo o risco de erros que ocasionam vazamento de informação.

2.4 MODELAGEM DE DADOS

Segundo Silva et al. (2023), a indústria 4.0 é caracterizada pela forte presença da internet (interconectividade e comunicação em geral), este conceito é também definido como “um sistema produtivo, integrado por computador e dispositivos móveis interligados à internet ou à intranet” Lima (2018). Essa capacidade de interconexão entre sensores através da rede possibilita diferentes aplicações, desde monitoramento de parâmetros, gerenciamento de informação, controle de dados etc. Dados estes, que são gerados e armazenados de forma contínua, a essa enorme massa de dados armazenada e tratada dá-se o nome de *Big Data*.

Sobre o conceito de *Big Data*, de acordo com Almeida (2019), pode ser definido pelo uso de técnicas que visam combinar dados de diferentes fontes e através dessas combinações obter novos insights. Sobre os tipos de dados utilizados, podem ser classificados de acordo com a maneira como são armazenados e/ou tratados. Estes dados são classificados em duas categorias macro: Dados estruturados, que são aqueles que seguem um formato de organização bem definido, facilitando seu armazenamento em bancos de dados e posterior análise, geralmente dispostos em formato tabular; dados não estruturados, os que não possuem um formato de organização padronizado, tornando complexo o processo de extração de informação, geralmente dispostos em formatos diversos, como textos, imagens, áudios, etc.

Taurion (2013) menciona sobre a necessidade de se aplicar técnicas visando a integração de diferentes tipos de dados e de fontes distintas, para obter-se um resultado final satisfatório. É citado como exemplo o setor da saúde, onde são utilizados dados de diferentes fontes como: medições de sensores através de monitoramento remoto, uso de informações disponibilizadas por pacientes em formulários, dados financeiros etc. de forma integrada para atestar a viabilidade de

determinado desenvolvimento / pesquisa. A origem destes dados pode ser das mais diversas, como: dados de operação do sistema, dados ambientais, dados de medições de equipamentos, etc.

A combinação desses dados de diferentes fontes pode gerar insights para que os planos de manutenção sejam ajustados baseados em dados específicos, otimizando o processo de manutenção. De forma geral, segundo Padilha (2022), faz-se necessário a análise de texto para compreender e trabalhar de forma mais eficiente essas fontes de dados, o principal objetivo dessa análise é formar dados estruturados a partir de textos não estruturados disponíveis.

Com o monitoramento de dados contínuo e a disponibilidade massiva destes dados, as técnicas de manutenção preditiva também são impulsionadas. Segundo Quintino (2019), com o advento da "Internet das Coisas" os equipamentos nessa era industrial passam a estar diretamente interconectados através da internet. Logo, pode-se abstrair desse conceito que não mais existe a necessidade de, como era no passado, os operadores de manutenção coletarem dados desses equipamentos de forma manual e periódica, o que resultava em uma quantidade baixa de informações, além de suscetível a erros e falhas humanas na aquisição.

2.5 SINALIZAÇÃO FERROVIÁRIA

De acordo com Theeg (2019), durante muitos anos não houve uma definição global dos conceitos de sinalização ferroviária, ficando estes conceitos muito específicos para cada país e cada ferrovia. Essa falta de padronização criou desafios para a modernização das redes ferroviárias. Ainda na obra, é mencionado sobre os primórdios dos processos de operação ferroviária, que era pautada em regras escritas ou até mesmo faladas, onde um responsável pela circulação dos trens ditava diretamente ao maquinista responsável pela composição como a circulação deveria ser realizada (via oral ou escrita), processo esse extremamente sujeito a erros humanos, o que além de limitar a eficiência operacional, botava em risco a segurança.

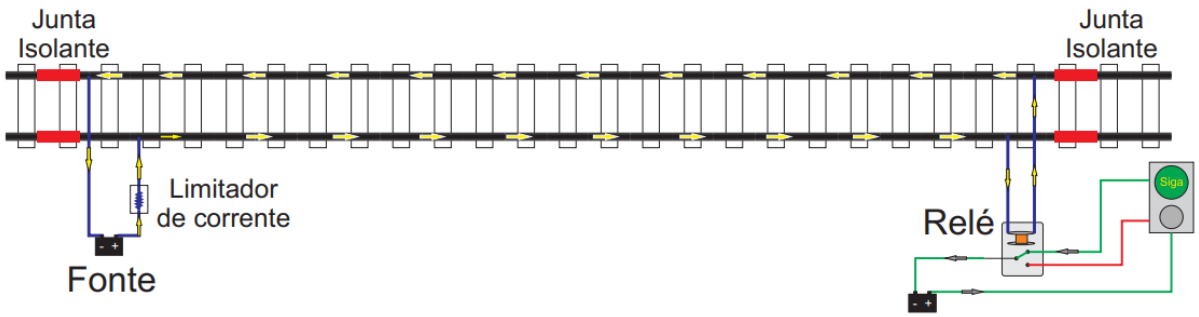
Com o passar dos anos e o advento de novas tecnologias, a sinalização ferroviária passa a se utilizar de sistemas e/ou equipamentos elétricos e eletromecânicos, como relés. Para se detectar a presença de trens sobre as seções da malha ferroviária, existe o conceito de Circuito de Via. De acordo com Junior (2018)

circuito de via é um circuito elétrico que utiliza os trilhos como condutores no lugar dos cabos e um relé no final do circuito. Durante a presença de trens, ocorre um curto-circuito durante a passagem do rodeiro metálico sobre o circuito, possibilitando assim detectar a presença de trens. É possível ainda detectar anomalias como trilho partido, uma vez que o circuito fica em aberto.

Segundo Bento et al. (2023), existem diferentes formas de se realizar a detecção de um trem, com diferentes equipamentos, desde conjuntos de baterias e relés até sistemas sofisticados gerenciados por controladores lógicos programáveis.

A Figura 1 representa um circuito de via de corrente contínua composto por um relé de indicação e uma fonte de alimentação, em que os trilhos do trem funcionam como “conectores” entre esses dois elementos, formando um circuito elétrico simples.

FIGURA 1 - Representação do circuito de via com relé de indicação



Fonte: Alvarenga (2018).

Contudo, embora as tecnologias da época tenham trazido avanços significativos, segundo Theeg (2019), o sistema ainda era totalmente dependente de interações humanas, o que gerava vulnerabilidade. A segurança ferroviária, apesar de aprimorada com o uso de blocos elétricos e sinaleiros (equipamento composto por fonte luminosa que indica se a seção à sua frente está livre da presença de outro trem), ainda sofria com riscos relacionados a falhas humanas, devido à ausência de bloqueios, intertravamentos e automação.

Alguns anos após a implementação dos primeiros modelos de sinaleiros e circuitos de via, surge o sistema conhecido como “Controle de Tráfego Centralizado”

(CTC): “um sistema composto por circuitos de via, relés elétricos e sinais indicativos de tráfego, ligados a um centro de controle de onde eram comandados os sinais” (Santos, 2021). Nesse modelo, um “Centro de Controle Operacional” (CCO) consegue operar, comandar e gerenciar o tráfego de uma ferrovia de forma centralizada e com visão do todo, diferente do que era no início das operações ferroviárias.

Santos (2021) destaca ainda a aplicação dos conceitos de intertravamento nos dispositivos elétricos da sinalização ferroviária, em que através uma lógica de conexões elétricas, a presença do trem em um determinado bloco permitia ou não a operação de outros elementos, além de proporcionar funcionalidades automáticas ao sistema: “Quando uma composição entrava num trecho livre, isto é, foco verde, a corrente elétrica no circuito de via aciona o foco vermelho” S, 2021)

Ribeiro (1999) menciona a aplicação de lógicas de intertravamento de segurança, bem como os motivos para sua implementação, entre os quais se destacam: a proteção do sistema e dos operadores, a prevenção de danos e a eficiência do processo, entre outros.

Em sequência, a sinalização ferroviária acompanhou o desenvolvimento das tecnologias de automação, como descrito em Nagpur (2013). A necessidade de adotar velocidades de circulação mais altas e com mais segurança motivou a busca por sistemas de controle e licenciamento de trens automáticos, adotando modelos de “Controle Automático de Trens” (ATC), que tem como base o sistema de “Controle de Trens Baseado em Comunicação” (CBTC).

O CBTC é, segundo Junior (2018), um sistema em que existe troca contínua de informações em tempo real entre a locomotiva de comando de uma composição (passando informações como posicionamento, velocidade etc.) e recebendo informações sobre as condições de circulação (trens a frente, condição e estado dos sinaleiros e circuitos de via, equipes de manutenção etc.). Essa comunicação ocorre através de um sistema de telecomunicações interligado diretamente com a sinalização de campo. A implementação do sistema oferece melhorias tanto ao maquinista, que obtém maior visibilidade sobre o trajeto e possíveis interrupções, quanto oferece ganhos relacionados à produtividade como um todo, uma vez que o sistema garante otimização de paradas e reduz a possibilidade de falha humana na operação.

Atualmente a sinalização está interligada aos conceitos de Indústria 4.0, adotando cada vez mais equipamentos e trens sensorizados conectados à rede e gerando dados. Esses avanços permitem monitorar de forma contínua o estado dos equipamentos da sinalização, possibilitando a prevenção de falhas através dos conceitos de manutenção preditiva.

2.6 CONTROLADOR LÓGICO PROGRAMÁVEL

Um CLP, de acordo com Filippo Filho (2014), é um equipamento capaz de executar algoritmos e desempenhar funções de acionamento, controle, sequenciamento, etc. E tem como principal objetivo controlar processos dos mais diversos tipos de máquinas. No caso da sinalização ferroviária, um dos CLP's empregados no modelo de licenciamento CBTC é o Electrologixs (IXS).

O IXS está inserido no contexto da sinalização ferroviária microprocessada, que opera "através de diversos equipamentos transmite pulsos de ondas quadradas, gerando os códigos de entrada e saída nos cartões Electrologixs" (Bento et al., 2013). Para isso, são utilizados os microprocessadores do IXS para gerar os pulsos, transmiti-los e recebê-los, o pulso por sua vez é transmitido pelos próprios trilhos de trem, similar ao conceito básico do circuito de via supracitado, porém de forma mais tecnológica e menos suscetível a interferências externas.

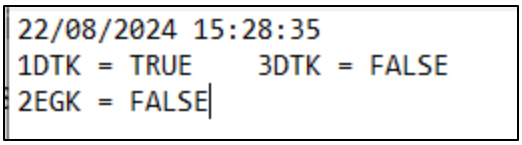
Esse equipamento, por ser um CLP, possui uma arquitetura que permite a integração de múltiplos módulos de entradas e saídas através de cartões (input e output). Possui a flexibilidade de ser programado para atender às necessidades específicas de cada operação, atendendo às lógicas de controle especificadas pelas devidas áreas de negócio. Além disso, é um sistema robusto projetado para operar de forma confiável sob as mais diversas adversidades, como: vibração, oscilações de temperatura, umidade, dentre outras que são frequentemente encontradas nas instalações ferroviárias.

O IXS possui intertravamento diferente dos modelos de sinalização convencional (apesar de possuir conceitos correlatos), de forma resumida, existe comunicação microprocessada entre um IXS e os outros de seu entorno, através de envio de pulsos de código pelos próprios trilhos. Ou seja, o equipamento está constantemente enviando suas informações vitais (status de circuitos e chaves, por

exemplo) para os que estão ao seu redor, garantindo o bloqueio e o intertravamento da seção de bloqueio.

Outro aspecto fundamental do equipamento é sua capacidade de comunicação, diferente do que existia nos sistemas de sinalização convencional, ele pode integrar-se a sistemas de supervisão, possui interface visual para configuração de parâmetros, e possui ainda a capacidade de armazenamento de dados de forma volátil em sua memória. Ou seja, todos os equipamentos por ele controlados são também monitorados, e suas mudanças de estado são armazenadas em forma de arquivos de log. Na Figura 2 observa-se um exemplo desse armazenamento de informações, onde na data de 22/08/2024 15:28:35 as variáveis '3DTK' e '2EGK' tiveram seu estado alterado para "FALSE", enquanto a variável "1DTK" teve seu estado alterado para "TRUE". A riqueza de detalhes presente nesse armazenamento contribui fortemente para o processo de análise das falhas ocorridas no sistema.

FIGURA 2 - Exemplo de armazenamento de informações no DataLog do Electrologixs



Fonte: Elaborado pelo autor (2024).

Em suma, a adoção do sistema de sinalização microprocessada vinculada ao CBTC em uma ferrovia acarreta resultados significativos em termos de eficiência operacional e de segurança, melhora ainda a capacidade de análise e resposta a incidentes e acidentes, garantindo maior segurança de passageiros ou da carga transportada.

Diferentes tipos de CLP's podem ser empregados no controle da sinalização de uma ferrovia, independente do modelo de sinalização empregado. Para o desenvolvimento deste trabalho, é essencial que o equipamento utilizado possua a capacidade de gerar os arquivos de log que armazenam suas informações ao longo do tempo.

2.7 *PYTHON*

Python é uma linguagem de programação criada por Guido van Rossum e lançada oficialmente em 1991. Segundo Wazlawick (2017), é uma linguagem criada buscando simplificar o processo de programação, minimizando a quantidade de decisões e sintaxe necessárias para um programador escrever um programa.

Trata-se de uma linguagem de alto nível, ou seja, ainda de acordo com Wazlawick (2017), é focada na compreensão do usuário, mais intuitiva e com elementos que se aproximam da linguagem humana, exigindo menos linhas de códigos e possuem elementos mais intuitivos quando comparada às linguagens de baixo nível (Assembly, por exemplo).

Ao longo dos últimos anos, *Python* tornou-se uma das linguagens de programação mais populares no meio da análise de dados, segundo Perkovic (2016), isso se dá por alguns motivos, sendo os principais deles justamente fornecer uma escrita legível e de fácil compreensão (alto nível), e por possuir uma vasta biblioteca de funções, possibilitando seu uso nas mais diversas aplicações. Dentre as vantagens da sua utilização, além do que já foi citado, pode-se destacar o fato de ser uma linguagem interpretada, ou seja, o usuário não precisa executar o processo de compilação, o código fonte é executado diretamente por um interpretador, linha a linha, o que facilita ainda o processo de identificação de erros. Outro ponto positivo é a possibilidade de utilização dos conceitos de programação orientada a objetos, que de forma geral oferece uma forma estrutural de organização de códigos que facilita a compreensão e manutenção de um script.

Existe uma vasta gama de bibliotecas disponíveis para utilização e outras em constante desenvolvimento por membros de diferentes comunidades (linguagem possui código aberto, o que facilita e impulsiona novos desenvolvimentos) que fornecem ferramentas robustas para as mais diversas aplicações. No campo de análise de dados, Behrman (2023) destaca “*Pandas*” e “*Numpy*”. Sendo Pandas um módulo que oferece estrutura e ferramentas de alto desempenho para análise de dados, possibilitando a capacidade de leitura, manipulação, modelagem, tratamento e organização dos dados. “O principal objeto deste módulo é o “*DataFrame*”, uma estrutura de dados rotulada bidimensional com colunas de tipos potencialmente diferentes. Você pode pensar nele como uma planilha, uma tabela SQL ou um

dicionário de objetos series" (The Pandas Development Team, 2024). Outro ponto de destaque para essa forma de estrutura é que os dados armazenados através do uso de ferramentas dessa biblioteca podem ser facilmente utilizados em modelos de inteligência artificial através de outras bibliotecas do próprio *Python* (biblioteca *Seaborn*, por exemplo). Já *Numpy*, é comumente utilizada na computação científica, oferecendo métodos e funções que suportam as mais diversas operações matemáticas, com foco em *Arrays* e matrizes multidimensionais, utilizada principalmente em "manipulação de forma, ordenação, seleção, entrada/saída, transformadas de Fourier discretas, álgebra linear básica, operações estatísticas básicas, simulação aleatória e muito mais." (Numpy Developers, 2024).

Um dos elementos cruciais e amplamente utilizado no desenvolvimento de softwares na atualidade, a Programação Orientada a Objetos, ou POO, trata-se de organizar o código em torno de "Objetos" de certa forma genéricos que são instâncias de uma classe. Perkovic (2016) define a POO como um "paradigma de desenvolvimento de software que alcança modularidade e portabilidade de código", através da implementação de conceitos como classes e objetos. Por se tratar de uma abordagem mais genérica de tratar a informação, é reutilizável para construção de programas e tratamento de diferentes arquivos dentro de uma mesma aplicação, justamente por isso é destacada a portabilidade de um código escrito dessa maneira.

Os dois conceitos mais elementares de POO que devem ser compreendidos são o de classe e objeto. Classe pode ser entendida como uma estrutura macro que define o conjunto de atributos e métodos que os objetos possuirão. Um objeto por sua vez, é uma instância de uma determinada classe, ou seja, algo concreto no código para representar informações sobre a classe a qual pertence.

2.8 BANCO DE DADOS

Um banco de dados é, segundo Pichetti (2024), um sistema para armazenar, gerenciar e manipular grandes volumes de dados de forma segura e estruturada. Tais sistemas são controlados pelos chamados "Sistemas de Gerenciamento de Banco de Dados" (SGBDs), sendo os mais comuns: SQL, *MySql*, *PostgreSQL* e *Oracle*.

No contexto da manutenção, organizar os dados é essencial para possibilitar a realização de análises de informações históricas e assim identificar os padrões de

falha. Contudo, é comum que os equipamentos da sinalização ferroviária gerem dados de forma não estruturada, dados estes que precisam ser tratados e reorganizados para que possam ser armazenados em sistemas de banco de dados relacionais.

O *MySql* é um banco de dados relacional, que segundo Alves (2014) permite armazenar e consultar informações de maneira simples e organizada. Sua utilização no *Python* é facilitada por bibliotecas como o sqlite3 (Python Software Foundation, 2024).

2.9 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) é, resumidamente, a capacidade de sistemas de computadores em realizar tarefas que tradicionalmente exigem inteligência humana, como reconhecimento de padrões e tomada de decisão. Segundo Kaufman (2022), não se trata apenas de fazer as máquinas pensarem como seres humanos, e sim de trabalhar algoritmos de previsão por meio de modelos estatísticos.

Neste trabalho em si, a IA não será aplicada diretamente, mas os dados obtidos e devidamente estruturados podem servir como base para futuras aplicações, principalmente em técnicas de aprendizado supervisionado (*machine learning*). Nestes modelos, a máquina é alimentada com uma vasta base de dados e resultados passados, e são posteriormente treinados com essas informações para identificação de padrões e determinar resultados com base no que ocorreu no passado. Segundo Lima (2014), essas aplicações tratam do emprego de algoritmos que buscam descobrir a correlação entre diferentes variáveis de um sistema, e qual a relevância dessas entradas no resultado final. Nestes modelos, a máquina é alimentada com uma vasta base de dados e resultados passados, e são posteriormente treinados com essas informações para identificação de padrões e determinar resultados com base no que ocorreu no passado

Com isso, a medida em que a base de informações se torna mais robusta e enriquecida, poderá ser possível o emprego de algoritmos deste tipo para automatizar a análise de desempenho de ativos, aumentando ainda mais a precisão na tomada de decisões e contribuindo diretamente para uma maior eficiência da manutenção preditiva.

3 DESENVOLVIMENTO PRÁTICO

Os conceitos teóricos introduzidos nos tópicos anteriores são de suma importância para o avanço do desenvolvimento prático que foi realizado ao longo do projeto. Tudo o que foi exposto servirá como fundamento no desenvolvimento de técnicas de manutenção preditiva através de softwares aplicativos que utilizam a linguagem *Python* para processar e analisar dados de equipamentos da sinalização ferroviária, e assim atender ao objetivo deste trabalho: reduzir interrupções no tráfego ferroviário que tem como origem falha nos equipamentos e/ou sistemas da sinalização ferroviária, com destaque para as máquinas de chave. Falhas essas que possuem origens elétricas e/ou eletromecânicas, como mau contato em conexões, resistência elétrica em contatos de relés, desgaste de contatos elétricos e outros comportamentos anormais dos componentes.

As técnicas que serão apresentadas, apesar de serem voltadas para diferentes equipamentos, possuem um fluxo de construção em comum, que pode ser numerado da seguinte forma:

1. Coleta de Dados**:** Desenvolver o método para realizar a coleta dos dados de forma automatizada, e armazená-los em um ambiente acessível;
2. Estruturação de Dados: Estabelecer um sistema para estruturar os dados coletados, que tradicionalmente possuem formato de texto, em uma estrutura tabular;
3. Armazenamento dos Dados: Garantir o armazenamento das informações estruturadas em ambiente acessível;
4. Análise Automática de Modos de Falha: Implementação do algoritmo de análise automática, identificando padrões anormais de funcionamento dos ativos;
5. Exibição dos resultados e tratativas: Exibir os resultados da análise para um usuário final, que deve por sua vez realizar o acionamento dos times de manutenção da companhia para correção dos desvios.

Os dados operacionais que serão insumo deste software são informações registradas sobre a operação de um equipamento e/ou sistema. Estes registram as condições/comportamentos do equipamento durante o desempenho de suas funções. Importante ressaltar que tradicionalmente esse tipo de informação não indica necessariamente uma falha no sistema, contudo, é possível identificar anomalias com

base nos seus padrões de funcionamento. Dentre as informações contidas nesses arquivos pode-se destacar: sequência de operação do equipamento, ações de controle, mudança de estado de relés/válvulas, etc.

Os elementos abordados neste trabalho serão aprofundados ao longo do desenvolvimento prático, com a devida análise e detalhamento de cada aspecto.

3.1 COLETA DE DADOS

A etapa fundamental na construção do sistema de análise automática é a coleta dos dados operacionais dos equipamentos. Esta coleta por sua vez pode ocorrer de duas formas principais: coletando os dados diretamente dos equipamentos ou utilizando dados já disponíveis em servidores específicos.

Existem cenários em que os próprios equipamentos armazenam seus registros de operação em sua memória interna (como ocorre com o Electrologixs). Nesses casos é necessário desenvolver uma solução automatizada com o emprego de técnicas de RPA para realizar a coleta das informações. No presente trabalho foi feito o uso das bibliotecas "*Pyautogui*" e "*Selenium*" para emular o processo de coleta manual das informações. Basicamente, foi feito um mapeamento de como uma coleta manual é realizada, e posteriormente esses passos foram transcritos em forma de código em um script *Python* que realiza as seguintes ações:

- Acessar o equipamento remotamente via navegador através de seu endereço de IP (endereço único para um equipamento dentro de uma rede de comunicação interna;
- Navegar até a aba "*Diagnostics*" do equipamento;
- Navegar até o submenu "*DataLog*";
- Clicar no botão "download log";
- Aguardar a realização do download;
- Fechar a página;

Essa sequência pode ser verificada na função da Figura 3.

UniAcademia
Centro Universitário

FIGURA 3 - Função para realizar o download do arquivo de log

```
37    def realizar_abertura(self):
38        try:
39            self.google.get("http://" + self._ip) # Endereco WEB que será acessado
40            self.google.implicitly_wait(60) # Aguarda o retorno do comando anterior antes de iniciar a próxima linha
41
42            # Autenticação na pagina web:
43            self.google.find_element('xpath', '//*[@id="username"]').send_keys(self._usuario) # Caixa de texto do usuario
44            self.google.find_element('xpath', '//*[@id="password"]').send_keys(self._senha) # Caixa de texto da senha
45            self.google.find_element('xpath', '//*[@id="login"]').click() # Botao de login
46            self.google.implicitly_wait(60)
47            self.google.find_element('xpath', '//*[@id="diagnostics_data_log_link"]').click() # Navegação dentro da página
48            self.google.find_element('xpath', '//*[@id="NavForm"]/table[1]/tbody/tr[1]/td[13]/input[2]').click() #Botao download
49            time.sleep(600) # Aguardar até o download terminar
50
51        except Exception as e:
52            str_falha = f'Houve um erro durante o acesso ao equipamento: {e}'
53            return str_falha
```

Fonte: Elaborado pelo autor (2024).

Para o cenário em que as informações já são transmitidas e armazenadas automaticamente para servidores centrais, como ocorre com algumas das remotas de sinalização, o processo de extração de dados torna-se mais simplificado, bastando identificar o local onde os arquivos estão salvos e desenvolver um script capaz de copiar e colar as informações de um ponto ao outro. Para isso, a biblioteca “OS” nativa do *Python* pode ser utilizada, conforme pode ser visto no script da Figura 4.

FIGURA 4 - Função para acessar arquivos de log armazenados localmente

```
54  import os # Importar a biblioteca OS
55
56  caminho = "C:\\Users\\Documents\\Logs" # Caminho da pasta que contém os arquivos de log
57
58  for arquivo in os.listdir(caminho): # Laço FOR para iterar sobre os arquivos do caminho
59      caminho_arquivo = os.path.join(caminho, arquivo) # Caminho completo do arquivo
60      with open(caminho_arquivo,'r'): # Abre o arquivo em modo de leitura
61          print(caminho_arquivo) # Exibir o arquivo, validando o funcionamento da função
62
```

Fonte: Elaborado pelo autor (2024).

Conforme supracitado nos itens anteriores, apesar de existirem diferentes formas de obter o dado, o intuito dessa etapa é o mesmo: Coletar o dado bruto (sem tratamento) do equipamento e disponibilizá-lo em uma pasta padrão.

3.2 TRATAMENTO DOS DADOS

Após possuir os dados devidamente armazenados em uma pasta, a próxima etapa está relacionada ao tratamento dessas informações. Em geral, neste estudo, foram utilizados arquivos no formato de texto (*.txt*), que conforme supracitado no desenvolvimento teórico, são dados do tipo não estruturado. Trabalhar com dados brutos é inviável, pois não permitem análises estruturadas ou relacionamentos entre variáveis, e nem as dispor ao longo do tempo.

No geral, o intuito é transformar o dado inicial em um formato de tabela, utilizando o recurso do *DataFrame* da biblioteca *Pandas*. Conforme já foi explanado, trata-se de um formato de armazenamento de dados bidimensional similar a uma tabela, que permite manipulação simples dos dados, além da possibilidade de serem transformados em tabelas de bancos de dados relacionais posteriormente. Para converter o arquivo de texto nesse formato estruturado, existem duas formas que serão explicadas utilizando exemplos reais:

3.2.1 Iteração linha-a-linha do arquivo de texto e armazenamento direto das variáveis:

Utilizando como exemplo o arquivo *DataLog* do IXS, ao abrir o arquivo é observado o padrão de escrita de dados evidenciado na Figura 5.

- As primeiras linhas contêm informações sobre o equipamento em si, como seu nome e versões de software nele instaladas;
- Após a declaração feita nas linhas iniciais, o formato segue um mesmo padrão, uma linha com informações de data/hora seguida por linhas com as informações das variáveis;

Após a análise preliminar da estrutura de armazenamento, foi necessário validar com a área de negócio se necessariamente todos os equipamentos seguirão esses moldes. Feito isso, foi aplicada uma técnica de varredura de arquivo linha-a-linha, onde para cada linha percorrida pelo código é feita a validação das informações contidas nessa linha. Sabe-se que as linhas de data/hora contêm a palavra "*Recorder Entry*", e que as linhas seguintes a ela contêm as variáveis a serem coletadas e armazenadas no *DataFrame*.

FIGURA 5 - Arquivo *DataLog* bruto do IXS

```
 5
 6  APPLICATION INFORMATION
 7       NAME              0fwb_112877v
 8       EPT CRC           27C4
 9       EPT Checksum      36EF
10
11  EXECUTIVE INFORMATION
12       PN                --
13       VER               --
14
15
16  09-12-23 14:55:51 Non-Vital Recorder Entry
17
18  1BLK          =F   3ALK          =F   3BLK          =F
19
20  09-12-23 14:55:51 Vital Recorder Entry
21
22  1BLR          =T   3ALR          =T   3ALSR         =T   3BLR          =T
23  3ESR          =T   4DHYR         =T   4D_4ESTOPR    =F   4D_C6_OUT     =F
24
25
26  09-12-23 14:55:51 Non-Vital Recorder Entry
27
28  3TK           =F
29
30  09-12-23 14:55:51 Vital Recorder Entry
31
32  3TPER         =T
33
34  09-12-23 14:55:50 Vital Recorder Entry
35
36  2EBPTER       =T
```

Fonte: Elaborado pelo autor (2024).

Para realizar a transformação do arquivo, foi utilizado o código presente na Figura 6.

FIGURA 6 – Função para montar *DataFrame* com as informações lidas

```python
import pandas as pd

def montar_df(self, dataLog):
        dataFrameTime = {} # Dicionario que vai armazenar temporáriamente as informações
        tamanho = len(dataLog) # Tamanho do arquivo recebido
        for i in range (0,tamanho): # Laço RANGE para iterar e realizar a leitura do arquivo
            msg_data = '' # Cria e declara a variável 'msg_data'
            if 'Recorder' in str(dataLog[i]): # Verifica presenca de 'recorder', indicando que é uma linha com data/hora
                msg = str(dataLog[i]).split() # Aloca a informação da linha na variavel 'msg'
                msg_data = str(msg[0] + ' ' +  msg[1]) # Organiza a data/hora na variavel 'msg_data'
                data = str(pd.Timestamp(msg_data)) # Conversão da data/hora para o formato pd.Timestamp
            elif len(str(dataLog[i])) > 1 : # Verifica se a linha contém informação (tamanho > 1)
                msg = dataLog[i].replace(' ','') # Remove os espaços da mensagem e armazena na variavel 'msg'
                msg = msg.replace('\n','') # Retira a quebra de linha da 'msg'
                while len(msg) > 0: # Iterar sobre a variável 'msg' enquanto houver conteúdo
                    primeiro_igual = msg.index('=') # Mapea a primeira ocorrência do caractere '='
                    tag = msg[0:primeiro_igual] # Identifica a tag da linha de mensagem
                    value = msg[primeiro_igual+1:primeiro_igual+2] # armazena o valor da tag (TRUE ou FALSE)
                    dataFrameTime[tag] = value # Cria a coluna referente a tag e armazena  seu valor
                    msg = msg[primeiro_igual+2:len(msg)] # Remove a parte já lida da variavel 'msg'
        dfDadosFinal = pd.DataFrame(dataFrameTime) # Cria uma variavel DataFrame com os dados do dicionario "DataFrameTime"
        return dfDadosFinal
```

Fonte: Elaborado pelo autor (2024).

O resultado após a execução desse script é o DataFrame da Figura 7.

FIGURA 7 - *DataFrame* final após estruturação do arquivo de texto

dfmaqChave (30825, 440)

	datetime	1TPERP	1TK	1TPR	5TPERP
0	2023-05-25T11:48:08.000	F	T	F	
1	2023-05-25T11:47:58.000				F
2	2023-05-25T11:47:57.000				
3	2023-05-25T11:47:23.000				
4	2023-05-25T11:47:09.000				
5	2023-05-25T11:47:08.000				
6	2023-05-25T11:46:53.000				
7	2023-05-25T11:32:34.000				
8	2023-05-25T11:32:32.000				
9	2023-05-25T11:32:31.000				
10	2023-05-25T11:32:30.000				
11	2023-05-25T11:32:29.000				
12	2023-05-25T11:32:24.000				
13	2023-05-25T11:32:23.000				
14	2023-05-25T11:32:22.000				
15	2023-05-25T11:16:12.000				
16	2023-05-25T11:15:58.000	T	F		
17	2023-05-25T11:15:42.000			T	
18	2023-05-25T11:14:31.000				T
19	2023-05-25T11:14:16.000				
20	2023-05-25T11:14:10.000				
21	2023-05-25T11:13:55.000				

Fonte: Elaborado pelo autor (2024).

3.2.2 Iteração linha-a-linha utilizando identificação das variáveis antes do armazenamento:

Em alguns formatos de armazenamento, diferente do que foi visto no *DataLog* do IXS, todas as informações de variáveis são expressas a cada momento de tempo, o que torna a iteração apresentada no item anterior inviável. Para contornar essa situação, foi utilizado as funcionalidades da biblioteca "*match*" do *Python*.

O processo em si é bem similar, o arquivo de texto é aberto e ocorre a iteração linha-a-linha visando identificar as informações dos estados das variáveis armazenadas. Porém, como neste caso uma mesma linha de data/hora irá conter diversas informações de variáveis, é utilizado o recurso "*findall*" para identificar tanto as variáveis quanto seu estado (*True* ou *False*). Para exemplificar essa situação, foi utilizado um log de sinalização ferroviária obtido de um servidor de sinalização específico. De forma sucinta, este servidor faz interface com as remotas de sinalização convencional (RTU's) instaladas em trechos da ferrovia onde não é utilizado o IXS (neste caso, existe a sinalização convencional via relé).

A Figura 8 demonstra o formato bruto de armazenamento deste tipo de arquivo, onde nota-se que para o mesmo instante de tempo, todas as informações de cada bit da remota é exibida. O processo de "raspagem" desses dados para armazenamento no *DataFrame* tende a ser mais trabalhoso, pois cada arquivo pode ter uma quantidade de informações distintas.

FIGURA 8 - Armazenamento do log em formato distinto

```
10-14-2024 00:00:15.942    FirstIndicationReceived:ALIAN?A (FAI) 2
10-14-2024 00:00:15.942    PROCESS IND:    10 (ALIAN?A (FAI) 2) (10000001000010000000000000000000)
10-14-2024 00:00:15.942    IND MNEM:       <(01)MTK=True    (02)BLANK=False (03)BLANK=False (04)BLANK=False (05)2EAK=False  (06)1TK=False   (07)BLANK=False >
10-14-2024 00:00:15.942    IND MNEM:       <(08)1NWK=True   (09)1RWK=False  (10)BLANK=False (11)BLANK=False (12)2EGK=False  (13)2DGK=True   (14)BLANK=False >
10-14-2024 00:00:15.942    IND MNEM:       <(15)BLANK=False (16)BLANK=False (17)BLANK=False (18)BLANK=False (19)BLANK=False (20)BLANK=False (21)BLANK=False >
10-14-2024 00:00:15.942    IND MNEM:       <(22)BLANK=False (23)BLANK=False (24)BLANK=False (25)BLANK=False (26)BLANK=False (27)BLANK=False (28)BLANK=False >
10-14-2024 00:00:15.942    IND MNEM:       <(29)INSTAVEL=False(30)BLANK=False (31)BLANK=False (32)POK-BAT=False>
```

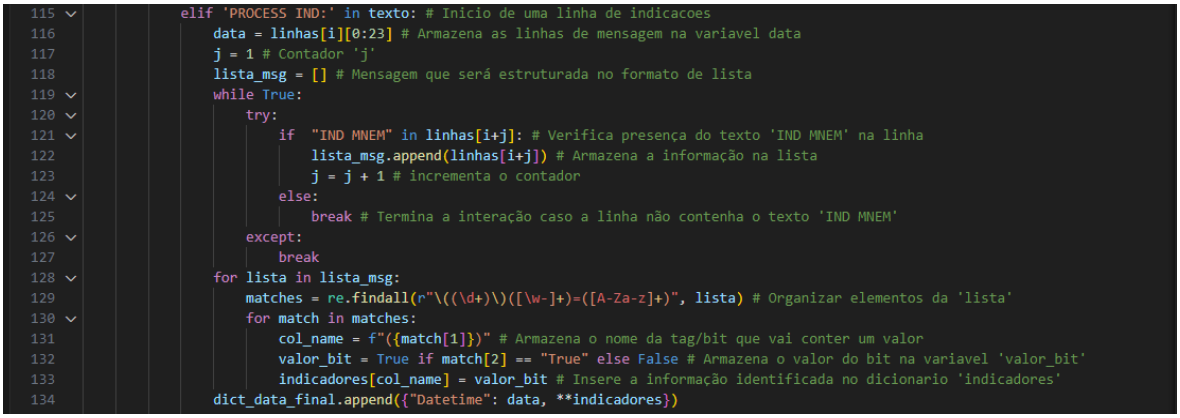
Fonte: Elaborado pelo autor (2024).

Com isso, o script torna-se mais complexo, sendo evidenciado na Figura 9 o processo de obtenção das *tags* e seus respectivos valores.

Com os processos de armazenamento e estruturação da informação coletada bem definidos, as informações agora estão dispostas em um formato de

armazenamento estruturado, possibilitando seguir com a sequência do desenvolvimento dos algoritmos de análises automáticas.

FIGURA 9 - Armazenamento das variáveis e valores identificados nas linhas



```
115         elif 'PROCESS IND:' in texto: # Inicio de uma linha de indicacoes
116             data = linhas[i][0:23] # Armazena as linhas de mensagem na variavel data
117             j = 1 # Contador 'j'
118             lista_msg = [] # Mensagem que será estruturada no formato de lista
119             while True:
120                 try:
121                     if  "IND MNEM" in linhas[i+j]: # Verifica presença do texto 'IND MNEM' na linha
122                         lista_msg.append(linhas[i+j]) # Armazena a informação na lista
123                         j = j + 1 # incrementa o contador
124                     else:
125                         break # Termina a interação caso a linha não contenha o texto 'IND MNEM'
126                 except:
127                     break
128             for lista in lista_msg:
129                 matches = re.findall(r"\((\d+)\)([\w-]+)=([A-Za-z]+)", lista) # Organizar elementos da 'lista'
130                 for match in matches:
131                     col_name = f"({match[1]})" # Armazena o nome da tag/bit que vai conter um valor
132                     valor_bit = True if match[2] == "True" else False # Armazena o valor do bit na variavel 'valor_bit'
133                     indicadores[col_name] = valor_bit # Insere a informação identificada no dicionario 'indicadores'
134             dict_data_final.append({"Datetime": data, **indicadores})
```

Fonte: Elaborado pelo autor (2024).

3.3 ANÁLISE AUTOMÁTICA DE DADOS

Após os dados serem coletados e devidamente tratados, ocorre a etapa de implementar a análise automática dessas informações. Nessa etapa, busca-se analisar eventos que podem ter ocorrido durante a operação do equipamento e que evidenciam possíveis falhas nos mesmos. Para isso, foi feito novamente um levantamento de informações com a área de negócio, onde foi repassado por eles modos de falhas que poderiam ser previstos e como cada um deles será visto através das informações contidas no log.

A Máquina de Chave (MCH), resumidamente, é o equipamento responsável por realizar a movimentação das agulhas e permitir a mudança de direção da composição entre duas linhas. Esse ativo é operado remotamente pelo controlador de operação da ferrovia, e após o comando de movimentação, o ativo deve indicar novo posicionamento final conforme esperado.

O componente principal da MCH é o motor elétrico, que exerce uma força sobre barras de acionamento que promovem a movimentação dos trilhos, e consequentemente permitem a mudança de sentido de circulação na via. O sistema possui ainda diversos elementos eletromecânicos, como contatos de fim de curso e relés que geram a informação do posicionamento atual dos trilhos, informação crucial

para a circulação ferroviária. Estes componentes por sua vez estão sujeitos a falhas elétricas, desgastes e intempéries que podem comprometer a confiabilidade de seu funcionamento, gerando por exemplo, indicações falsas referente ao posicionamento das agulhas.

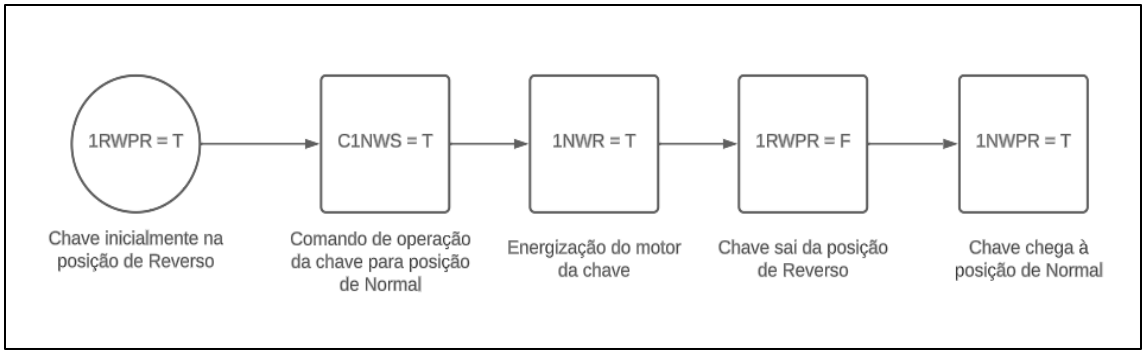
Um exemplo de falha recorrente é quando ocorre oscilação nas indicações de posicionamento dessas agulhas. A chave está fisicamente em uma posição estável, mas devido a um mau contato elétrico (ou outra falha no equipamento), o sistema recebe uma informação errônea de sua posição. Tal condição não apresenta risco direto à circulação, uma vez que a chave em si não apresentou movimentação, mas com adoção dos sistemas modernos de sinalização (CBTC) essas indicações “falsas” podem promover, por segurança, a parada imediata de um trem que circula em um trecho próximo ao equipamento, impactando diretamente na eficiência operacional de uma ferrovia.

Tais informações referentes a posição e comandos dados aos equipamentos são registradas através de *tags* nos arquivos de log. A sequência lógica do fluxo na Figura 10 representa as informações registradas em log durante a movimentação de uma MCH.

Com isso, sabe-se que a operação de uma chave possui um caminho fechado e bem determinado, contendo início e fim, sendo que o início é justamente a variável que representa o comando de operação do ativo (neste exemplo, a variável C1NWS), e o fim é a indicação do ativo na posição previamente desejada (neste exemplo, a variável 1NWPR).

Conhecendo o fluxo de operação normal do ativo, outras anomalias podem ser identificadas, como por exemplo, a perda da indicação de *tag* de posicionamento sem que haja um comando para tal (justamente o exemplo descrito anteriormente neste mesmo tópico).

FIGURA 10 - Sequência lógica de operação de uma MCH



Fonte: Elaborado pelo autor (2024).

Outro ativo que terá seu funcionamento analisado de forma autônoma é o Detector de Descarrilamento (DD). Este equipamento é capaz de identificar se houve a perda do contato roda/trilho em uma composição, ou seja, se o trem está em processo de descarrilamento. Trata-se de uma barra instalada transversalmente aos trilhos, quando essa barra é quebrada o DD é acionado. Automaticamente ocorre a aplicação de frenagem da locomotiva através dos sistemas modernos de sinalização (CBTC), e um alarme é emitido no painel do controlador de operação. Contudo, para que o alarme seja emitido, geralmente é necessário que a indicação de quebra permaneça estável por alguns segundos, como forma de garantia que houve de fato uma quebra. Bem como as MCH's, por se tratar de um equipamento que possui contatos elétricos (as barras possuem fios que são conectados ao CLP local) é esperado que o mesmo esteja sujeito a diversas falhas, como por exemplo, o mau-contato de seus terminais, que podem ocasionar também uma falsa indicação de descarrilamento e uma parada indevida de trem. A Figura 11 evidencia uma barra instalada na ferrovia.

FIGURA 11 - Detector de descarrilhamento instalado em campo

Fonte: Elaborado pelo autor (2024).

Conforme descrito no desenvolvimento teórico deste trabalho, o Circuito de Via (CDV) é um dos componentes fundamentais da sinalização ferroviária, tendo como função primária a identificação da presença de trens. Conforme também mencionado, existem diferentes formas e equipamentos utilizados para realizar essa detecção, sendo o mais simples aquele compreendido por um circuito com fonte de alimentação, cabos e relés. Assim como os outros ativos já mencionados, novamente existe uma condição em que componentes eletroeletrônicos estão instalados em condições climáticas adversas, e consequentemente, sujeitos a diferentes modos de falha elétricos, como: resistência de contato em relé, mau-contato em conexões, baixa isolação em cabos, falhas de aterramento do circuito, dentre outras. O funcionamento,

conforme descrito no desenvolvimento teórico, é bem simplório, e assim como nos outros ativos já mencionados, as informações de seu funcionamento são armazenadas em *tags* dos arquivos de log dos CLP's. Basicamente, as *tags* dos circuitos oscilam entre nível lógico alto (*true*) e baixo (*false*), geralmente o nível lógico alto indicando a presença de trem no circuito. A proposta dessa análise automática é identificar oscilações nesta indicação, que sinalizam uma das possíveis falhas elétricas citadas nos componentes.

Tendo o entendimento amplo do funcionamento dos ativos, e como estes geram registros de informações, o próximo passo é criar os algoritmos de detecção de anomalias em seus funcionamentos, utilizando scripts na linguagem *Python*. Como exemplo destes scripts, estão evidenciados nas figuras 12 e 13 os scripts utilizados na identificação de anomalias em DD's e CDV's, respectivamente.

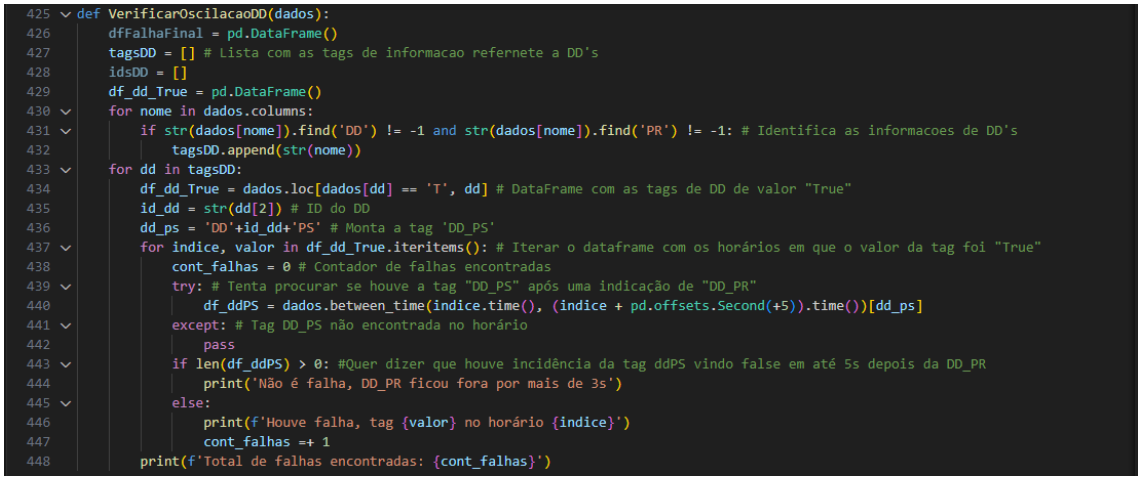
Em ambas funções, conforme pode-se analisar com o código comentado, o *DataFrame* de indicações do CLP é varrido (iterado) buscando por condições específicas que indicam a potencial falha. Com posse das informações da possível falha, o time de manutenção de campo da companhia deve ser acionado para investigação in loco da causa raiz do defeito.

Em posse dos dados operacionais devidamente estruturados, existe uma vasta gama de desenvolvimentos possíveis, para diferentes modos de falha. Em suma, a criação de uma nova função passa por um fluxo básico:

1. Área de negócio (análise de falhas) identifica possibilidade de criação de modo de falha novo;
2. Área de negócio determina os padrões e parâmetros a serem adotados;
3. Desenvolvedor cria a função e implementa em produção;
4. Resultados são analisados em conjunto com a área de negócios e correções são realizadas;
5. Time de manutenção passa a ser acionado para novas intervenções.

UniAcademia
Centro Universitário

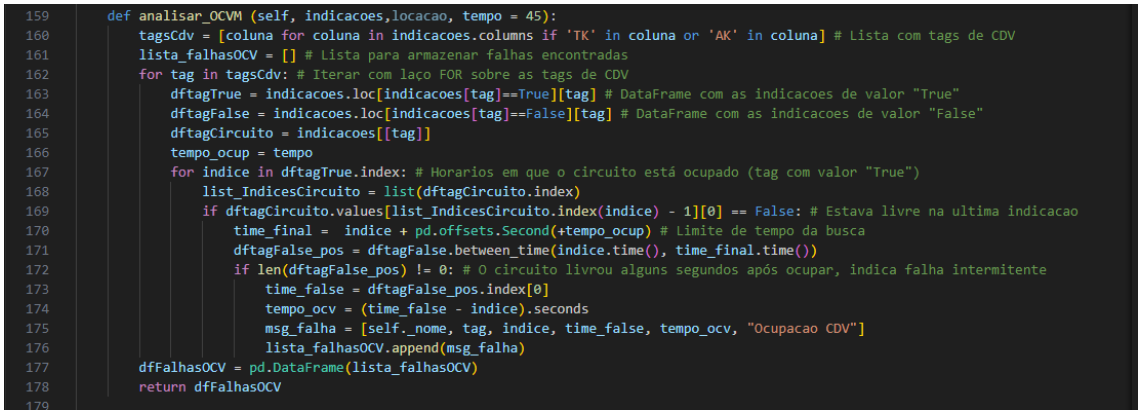
FIGURA 12 - Função para detecção de falhas em DD's



```python
def VerificarOscilacaoDD(dados):
    dfFalhaFinal = pd.DataFrame()
    tagsDD = [] # Lista com as tags de informacao refernete a DD's
    idsDD = []
    df_dd_True = pd.DataFrame()
    for nome in dados.columns:
        if str(dados[nome]).find('DD') != -1 and str(dados[nome]).find('PR') != -1: # Identifica as informacoes de DD's
            tagsDD.append(str(nome))
    for dd in tagsDD:
        df_dd_True = dados.loc[dados[dd] == 'T', dd] # DataFrame com as tags de DD de valor "True"
        id_dd = str(dd[2]) # ID do DD
        dd_ps = 'DD'+id_dd+'PS' # Monta a tag 'DD_PS'
        for indice, valor in df_dd_True.iteritems(): # Iterar o dataframe com os horários em que o valor da tag foi "True"
            cont_falhas = 0 # Contador de falhas encontradas
            try: # Tenta procurar se houve a tag "DD_PS" após uma indicação de "DD_PR"
                df_ddPS = dados.between_time(indice.time(), (indice + pd.offsets.Second(+5)).time())[dd_ps]
            except: # Tag DD_PS não encontrada no horário
                pass
            if len(df_ddPS) > 0: #Quer dizer que houve incidência da tag ddPS vindo false em até 5s depois da DD_PR
                print('Não é falha, DD_PR ficou fora por mais de 3s')
            else:
                print(f'Houve falha, tag {valor} no horário {indice}')
                cont_falhas =+ 1
        print(f'Total de falhas encontradas: {cont_falhas}')
```

Fonte: Elaborado pelo autor (2024).

FIGURA 13 - Função para detecção de falhas em CDV's



```python
    def analisar_OCVM (self, indicacoes,locacao, tempo = 45):
        tagsCdv = [coluna for coluna in indicacoes.columns if 'TK' in coluna or 'AK' in coluna] # Lista com tags de CDV
        lista_falhasOCV = [] # Lista para armazenar falhas encontradas
        for tag in tagsCdv: # Iterar com laço FOR sobre as tags de CDV
            dftagTrue = indicacoes.loc[indicacoes[tag]==True][tag] # DataFrame com as indicacoes de valor "True"
            dftagFalse = indicacoes.loc[indicacoes[tag]==False][tag] # DataFrame com as indicacoes de valor "False"
            dftagCircuito = indicacoes[[tag]]
            tempo_ocup = tempo
            for indice in dftagTrue.index: # Horarios em que o circuito está ocupado (tag com valor "True")
                list_IndicesCircuito = list(dftagCircuito.index)
                if dftagCircuito.values[list_IndicesCircuito.index(indice) - 1][0] == False: # Estava livre na ultima indicacao
                    time_final =  indice + pd.offsets.Second(+tempo_ocup) # Limite de tempo da busca
                    dftagFalse_pos = dftagFalse.between_time(indice.time(), time_final.time())
                    if len(dftagFalse_pos) != 0: # O circuito livrou alguns segundos após ocupar, indica falha intermitente
                        time_false = dftagFalse_pos.index[0]
                        tempo_ocv = (time_false - indice).seconds
                        msg_falha = [self._nome, tag, indice, time_false, tempo_ocv, "Ocupacao CDV"]
                        lista_falhasOCV.append(msg_falha)
        dfFalhasOCV = pd.DataFrame(lista_falhasOCV)
        return dfFalhasOCV
```

Fonte: Elaborado pelo autor (2024).

3.4 EXIBIÇÃO DOS RESULTADOS E INTERFACE GRÁFICA

Por fim, com os alarmes de possíveis anomalias geradas na etapa da análise automática, é necessário definir como essas informações serão visualizadas pelo usuário final. As falhas identificadas são gravadas em arquivos de formato Excel e/ou texto e são verificadas pelo usuário do sistema.

Com posse das informações de anomalia detectadas, o usuário realiza o acionamento da devida equipe de manutenção da companhia que realiza a inspeção do ativo em campo.

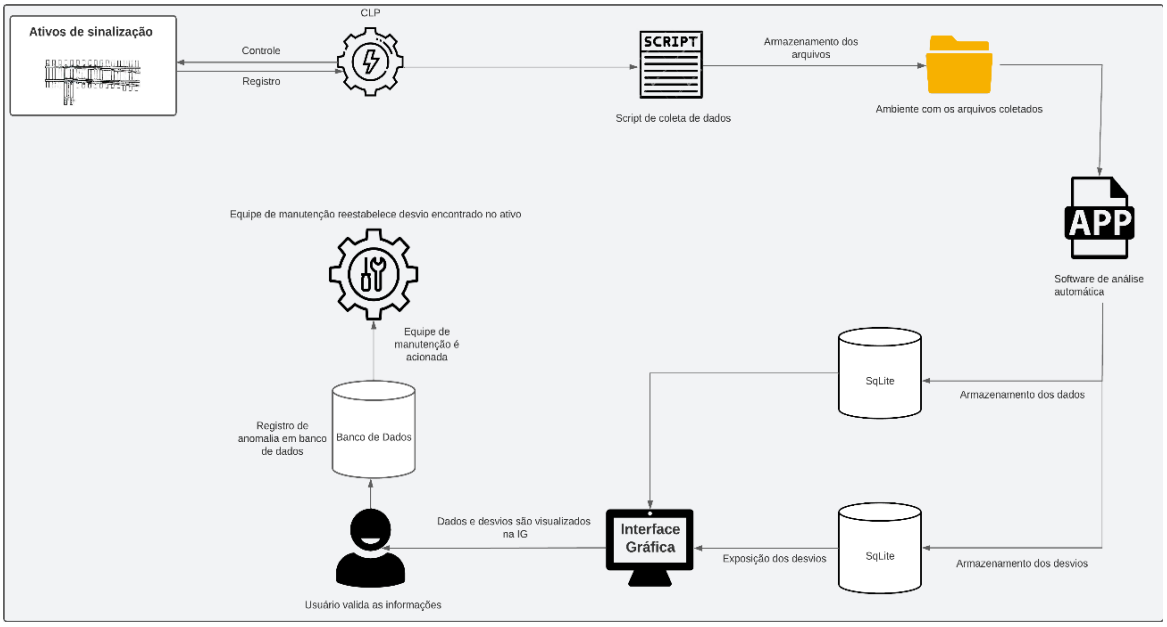
3.5 CONCLUSÃO DO DESENVOLVIMENTO

Para concluir o desenvolvimento do projeto, é possível consolidar todas as etapas já realizadas e organizá-las em uma estrutura na qual seja possível elaborar um procedimento operacional, com o intuito de definir algumas etapas. Primeiramente, ocorre a leitura dos dados diretamente de um ambiente de rede específico. Em seguida, procede-se ao processamento dos dados através do algoritmo de análise automática, e os resultados da análise são revisados pelo usuário responsável e tratados por ele.

- Todos os algoritmos mencionados nos tópicos anteriores são consolidados em um único software, que estará disponível para a execução pelo usuário;
- O usuário executará o software em intervalos previamente definidos;
- Com os resultados exibidos na interface gráfica à disposição, o usuário procederá com os acionamentos e fará os registros das anomalias identificadas no sistema de Planejamento de Recursos Empresariais (ERP) da empresa;

A operação da aplicação, como pode ser observado, é bastante simples e quase inteiramente automatizada, atribuindo ao usuário a responsabilidade de seguir a rotina estabelecida. Foi elaborado ainda um fluxograma na Figura 14 com uma visão global das etapas do funcionamento padrão do software.

FIGURA 14 - Fluxo global de funcionamento do software



Fonte: Elaborado pelo autor (2024).

4 RESULTADOS E DISCUSSÕES

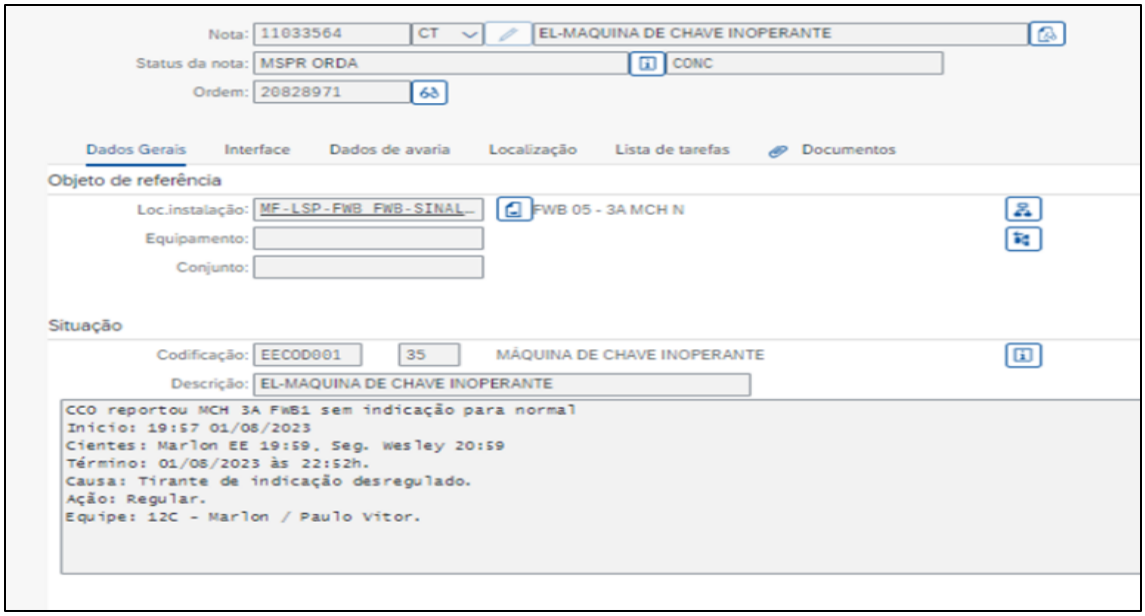
Foram realizadas de forma regular etapas de validação do processo, ou seja, os resultados obtidos no *software* passaram pelo crivo técnico do time de análise de falhas da empresa, validando o algoritmo. Foi possível confirmar a acurácia do sistema tanto na detecção de desvios que já haviam gerado impacto, quanto naqueles em que ainda era possível realizar atuação preditiva. Serão evidenciados abaixo exemplos de verificação em que obtivemos êxito.

4.1 EXEMPLO DE RESULTADO 01

No dia 01/08/2023 foi reportado pelo time de operação da empresa que uma MCH não apresentava indicação do posicionamento “normal”. Foi realizado o devido registro da falha no sistema ERP da empresa conforme evidenciado na Figura 15.

UniAcademia
Centro Universitário

FIGURA 15 - Registro no ERP da falha do exemplo 01



Fonte: Elaborado pelo autor (2024).

Nota-se que nesse caso existia falha em um componente mecânico do ativo (tirante de indicação), que foi corrigido após atuação da manutenção. Os resultados da análise indicam que a MCH apresentava desvios em seu funcionamento desde o dia 27/07/2023. Os resultados estão evidenciados na Figura 16. Ou seja, a anomalia poderia ter sido identificada cerca de cinco dias antes ao relato do time de operação.

FIGURA 16 - Resultado da análise automática na falha do exemplo 01

datetime	C3NWS	C1NWS	3ANWPR	3ARWPR	1BNWPR	1BRWPR	Alarme
2023-08-01 22:28:39	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-08-01 19:55:43	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-08-01 19:54:57	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-08-01 19:54:25	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-08-01 19:18:33		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-08-01 19:17:31		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-08-01 19:16:50		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-08-01 19:16:38	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-08-01 19:15:52		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-08-01 19:15:24		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-08-01 19:14:14		T			Ñ indicou	F	Alarme - HOUVE COMANDO C1NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 1BNWPR
2023-07-31 17:14:32	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-07-27 23:28:15	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-07-27 18:44:11	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR
2023-07-27 18:43:45	T		Ñ indicou	F			Alarme - HOUVE COMANDO C3NWS PARA NORNAL COM MOVIMENTAÇÃO DA CHAVE SEM INDICAR POSIÇÃO FINAL 3ANWPR

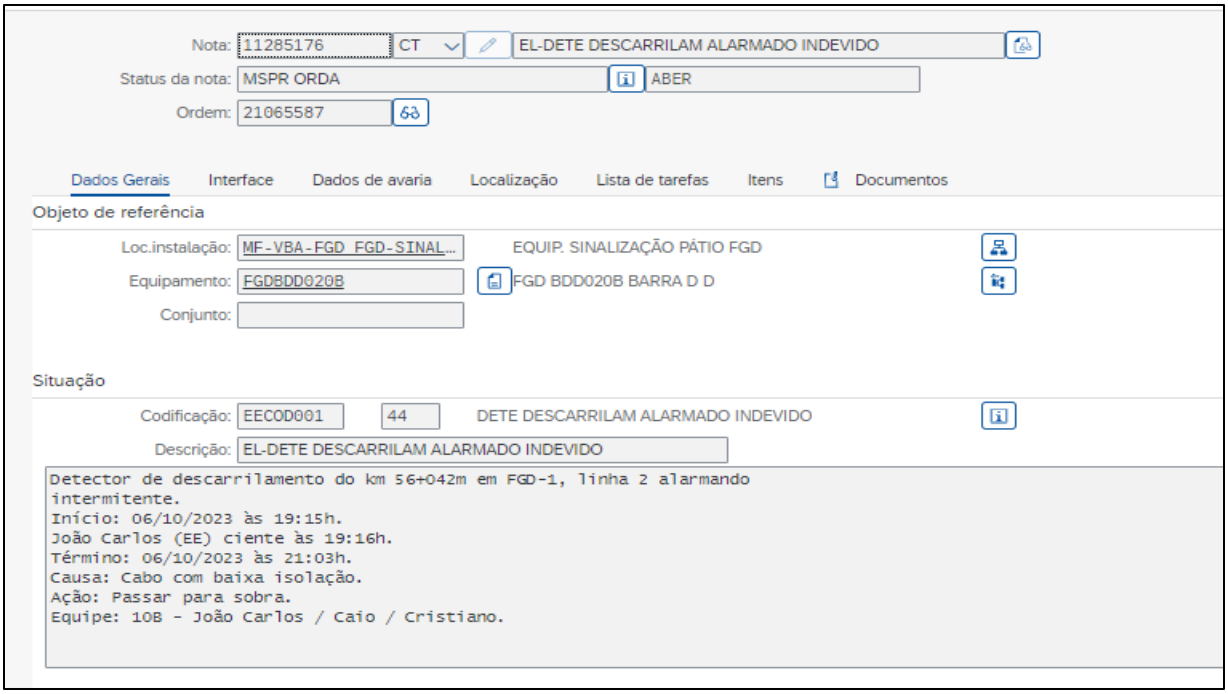
Fonte: Elaborado pelo autor (2024).

UniAcademia
Centro Universitário

4.3 EXEMPLO DE RESULTADO 02

No dia 06/10/2023 foi reportado pelo time de operação que um DD alarmava de forma indevida, ou seja, gerou-se o alarme no respectivo DD sem a presença de um trem sobre o circuito. A Figura 17 evidencia o registro da anomalia no ERP.

FIGURA 17 - Registro no ERP da falha do exemplo 02



Fonte: Elaborado pelo autor (2024).

Neste caso, observa- se que a causa do problema foi cabeamento com baixa isolação, essa condição anômala fazia com que as indicações do equipamento ficassem distorcidas. Bem como no exemplo anterior, os arquivos de log foram submetidos ao algoritmo de análise automática, constatando que o defeito se iniciou cerca de 17h antes da percepção da falha no ativo. Ou seja, existia possibilidade de atuação antes que o ativo gerasse o impacto. A Figura 18 evidencia o resultado da análise obtido.

FIGURA 18 - Resultado da análise automática na falha do exemplo 02

*Sem título - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:44:13
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:44:02
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:58
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:57
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:55
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:48
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:45
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:44
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:41
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:32
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:43:19
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:42:54
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:42:37
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:25:46
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:06:36
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:03:23
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:03:05
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:00:45
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:00:39
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 16:00:24
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 11:25:46
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 10:01:43
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 10:01:39
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:56:14
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:56:02
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:55:24
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:55:22
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:55:18
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:55:11
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:54:49
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:54:45
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:54:39
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 09:26:49
FALHA ENCONTRADA: Tag DD2BPR oscilou por menos de 3s em 2023-10-06 02:09:53
```

Fonte: Elaborado pelo autor (2024).

Apesar do software não estar implementado e em funcionamento no ambiente empresarial até o desenvolvimento deste trabalho, através destas análises de falhas já ocorridas foi possível confirmar sua eficácia na detecção de anomalias, confirmando que a ferramenta apresenta ganhos para manutenção elétrica preditiva da companhia. Foi realizado um estudo de eficácia na detecção de anomalias em fevereiro de 2025, onde 10 falhas ocorridas em MCH's foram submetidas ao processo de análise automática, e destas, em 6 foi possível identificar a falha em até dois dias antes do impacto à circulação. Ou seja, o software desenvolvido apresentou uma eficácia de 60%.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de técnicas para monitoramento e predição de falhas em ativos de sinalização ferroviários baseada na análise de dados de operação, enfatizando as etapas de coleta e implementação das análises automáticas. A abordagem adotada, conforme pode ser visto na etapa de resultados,

é uma solução bastante promissora para redução de interrupções indevidas no tráfego ferroviário, aprimorando assim a confiabilidade operacional dos sistemas.

Este estudo revela ainda a importância dos *logs* operacionais dos equipamentos como uma fonte relevante e estratégica para o desenvolvimento de soluções preditivas, mesmo não sendo um modelo convencional deste segmento, que normalmente é pautado no monitoramento dos parâmetros dos equipamentos (temperatura, corrente, vibração etc.). A automação do processo analítico dessas informações mostrou-se bastante efetiva na identificação de anomalias em ativos.

Por fim, o estudo visou contribuir de forma consistente para a evolução na gestão de ativos ferroviários, alinhando-se assim com os princípios da Indústria 4.0 e evidenciando o potencial de desenvolvimento ainda existente com as tecnologias de automação existentes. Os resultados obtidos reforçam ainda a necessidade de desenvolvimento de soluções integradas que proporcionem maiores possibilidades de manter ativos de forma não emergencial, agregando valor para diferentes áreas da operação ferroviária.

Diante dos resultados alcançados e já mencionados, existe ainda um grande potencial a ser explorado nesta linha de desenvolvimentos. A estruturação de dados realizada permitiu criar uma base sólida de informações que podem ser aplicadas a técnicas de inteligência artificial e/ou aprendizado de máquina, possibilitando ainda mais capacidade preditiva. Existe ainda a perspectiva de expansão do algoritmo para outros sistemas de sinalização ferroviária (e áreas correlatas) que possuam mecanismos historiadores (arquivos de *log*). Uma outra possibilidade está na integração destes algoritmos de detecção com o sistema ERP da empresa, agregando ainda mais dinamismo e capacidade de velocidade de resposta das equipes de manutenção para as anomalias detectadas.

ABSTRACT

Predictive maintenance is a crucial strategy for asset management in sectors that demand high reliability, such as the railway industry. This paper initially 36ontexto36 a literature review on various concepts related to predictive maintenance and subsequently proposes methods for developing predictive systems within the 36ontexto of railway signaling, with na emphasis on automated solutions for data collection and analysis generated by programmable logic controllers. Through the

analysis of operational records, the study aims to anticipate failures and reduce unplanned corrective interventions. It details the development of application software that optimizes data processing, enabling the construction of a robust database that may serve as a foundation for future implementation of artificial intelligence techniques for fault prediction. The initiative highlights the benefits of integrating automation with predictive maintenance, such as increased operational efficiency and cost reduction related to failures.

Keywords: Predictive Maintenance. Railway Signaling. Process Automation. Data Analysis

REFERÊNCIAS

ALMEIDA, Paulo Samuel de. **Indústria 4.0**: princípios básicos, aplicabilidade e implantação na área industrial. Rio de Janeiro: Érica, 2019.

ALVARENGA, Tiago Araujo. **Identificação e localização de falhas em circuitos de via de ferrovias baseada em reflectometria no domínio da frequência**. 2018. Tese (Doutorado) – Universidade Federal de Juiz de Fora, Juiz de Fora, 2018.

ALVES, William P. **Banco de dados**. Rio de Janeiro: Érica, 2014.

BEHRMAN, Kennedy R.; BRODECK, Henrique**. Fundamentos de Python para ciência de dados**. Porto Alegre: Bookman, 2023.

BENTO, Thauan Farias; MARIA, Márcio Barbosa; FERREIRA, Alex Franco. **Sistema de sinalização ferroviária, circuito de via** = Railway signaling system, track circuit. [S.l.: s.n.], [s.d.].

FERNANDES FILHO, G. E. F. **Automação de processos e sistemas**. São Paulo: Editora Érica, 2014.

FOGLIATO, F. F. **Confiabilidade e manutenção industrial**. Rio de Janeiro: GEN LTC, 2009.

GREGÓRIO, G. F. P.; SANTOS, D. F.; PRATA, A. B. **Engenharia de manutenção**. Porto Alegre: SER - SAGAH, 2018.

KAUFMAN, Dora. **Desmistificando a inteligência artificial**. São Paulo: Autêntica Editora, 2022.

KARDEC, A.; NASCIF, J. **Manutenção função estratégica**. Rio de Janeiro: Qualitymark Editora Ltda, 1998.

LIMA, A. W. B. et al. **Indústria 4.0**: conceitos e fundamentos. São Paulo: Blucher, 2009.

LIMA, Isaías. **Inteligência artificial**. Rio de Janeiro: GEN LTC, 2014.

LUTZ, Mark; ASCHER, David. **Aprendendo Python**. Porto Alegre: Bookman, 2007.

MAMEDE, H. S. **Automatização de processos com RPA**. Lisboa: FCA, 2021.

M. C. JÚNIOR, Jarbas et al. (ed.). **Sinalização e comunicação: elementos de sinalização e comunicação**. Juiz de Fora: Bright Treinamento Ltda, 2018.

MRS LOGÍSTICA. **Conheça o CBTC**. Disponível em: https://www.mrs.com.br/post-blog-inovacao/conheca-o-cbtc/. Acesso em: 27 out. 2024.

MUNIZ, A.; RODRIGUES, A. C.; MARTINS, L.; STRAFACCI, G. **Jornada RPA e hiperautomação**: como acelerar a transformação digital somando tecnologia e processos inteligentes. Rio de Janeiro: BRASPORT Livros e Multimídia Ltda, 2022.

NEPONUCENO, L. X. **Técnicas de manutenção preditiva**: volume 1. São Paulo: Blucher, 1989.

NUMPY DEVELOPERS. **NumPy Documentation**. Disponível em: https://numpy.org/doc/stable/. Acesso em: 27 out. 2024.

PADILHA, Juliana et al. **Analytics para big data**. Porto Alegre: SAGAH, 2022.

PERES, Tarcísio de Souza. **Programação orientada a objetos**. São Paulo: Editora Sol, 2024.

PERKOVIC, Ljubomir. **Introdução à computação usando Python**: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.

PICHETTI, Roni F.; VIDA, Edinilson S.; CORTES, Vanessa S. M. P. **Banco de dados**. Porto Alegre: SAGAH, 2021.

PYTHON SOFTWARE FOUNDATION. sqlite3 — DB-API 2.0 **Interface for SQLite Databases**. Disponível em: https://docs.python.org/3/library/sqlite3.html. Acesso em: 27 out. 2024.

QUINTINO, Luis F. et al. **Indústria 4.0**. Porto Alegre: SAGAH, 2019.

RIBEIRO, Marco Antônio. **Automação industrial**. Salvador: Tek Treinamento e Consultoria, 1999.

SANTOS, Silvio dos. **Transporte ferroviário**: histórias e técnicas. Florianópolis: Ed. do Autor, 2021.

SILVA, Luiz F. C. et al. **Banco de dados não relacional**. Porto Alegre: SAGAH, 2021.

SIQUEIRA, I. P. **Manutenção centrada na confiabilidade**: manual de implementação. Rio de Janeiro: Qualitymark Editora Ltda, 2005.

TAURION, Cezar. **Big data**. Rio de Janeiro: Brasport, 2013.

THE PANDAS DEVELOPMENT TEAM. **Pandas Documentation**. Disponível em: https://pandas.pydata.org/pandas-docs/stable/. Acesso em: 27 out. 2024.

THEEG, Gregor; VLASENKO, Sergej. **Railway signalling and interlocking**. Bergisch Gladbach: Trackomedia, 2019.

WAZLAWICK, Raul. **Introdução a algoritmos e programação com Python**: uma abordagem dirigida por testes. Rio de Janeiro: GEN, 2017.