INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* BETIM
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

MATHEUS ROCHA GUERRA

USO DE *HARDWARE-IN-THE-LOOP* E RECONHECIMENTO DE IMAGEM PARA VALIDAÇÃO DE CENTRAIS AUTOMOTIVAS

Betim
2025

MATHEUS ROCHA GUERRA

USO DE *HARDWARE-IN-THE-LOOP* E RECONHECIMENTO DE IMAGEM PARA VALIDAÇÃO DE CENTRAIS AUTOMOTIVAS

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Msc. Bruno de Souza Baptista

Coorientador: Prof. Msc. Maurício Monteiro da Silva

Betim

2025

FICHA CATALOGRÁFICA

G934u Guerra, Matheus Rocha

Uso de Hardware-in-the-Loop e reconhecimento de imagem para validação de centrais automotivas / Matheus Rocha Guerra. – 2025.

79 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2025.

Orientação: prof. Me. Bruno de Souza Baptista
Coorientação: prof. Me. Maurício Monteiro da Silva

1. Centrais automotivas. 2. Sistemas embarcados. 3. Visão computacional. 4. Reconhecimento computacional. 5. Engenharia de Controle e Automação. I. Guerra, Matheus Rocha. II. Título.

CDU: 629.33

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS

Campus Betim
Diretoria de Ensino
Docentes Mecânica

Rua Itamarati - CEP 32677-564 - Betim - MG

3135976360 - www.ifmg.edu.br

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos 13 dias do mês de julho do ano de 2025, às dezenove horas, nas dependências do IFMG – Campus Betim, reuniu-se a banca examinadora presidida por mim, Bruno de Souza Baptista e demais membros, Leandro Freitas de Abreu e Felipe Augusto Rocha da Silva. Nesta ocasião o discente Matheus Rocha Guerra, do curso de Bacharelado em Engenharia de Controle e Automação, com registro acadêmico de número 0034095 do IFMG – Campus Betim, defendeu seu Trabalho de Conclusão de Curso (TCC) intitulado "Uso de Hardware-In-The-Loop e Reconhecimento de Imagem Para Validação de Centrais Automotivas" e foi APROVADO, com 92 (noventa e dois) pontos.

Este resultado reflete o cumprimento parcial dos critérios de avaliação estabelecidos pelo curso e reconhece os esforços e a dedicação do discente e seu orientador no desenvolvimento do seu TCC. O lançamento da nota e o consequente encerramento do respectivo processo está condicionado ao cumprimento dos procedimentos pós-defesa conforme previstos nos regulamentos vigentes. Tais procedimentos pós-defesa devem ser finalizados dentro do prazo limite de 30 dias, a contar da data desta ata. O descumprimento destes procedimentos até a data estipulada implicará em atribuição de nota 0 (zero) e consequente reprovação.

A sessão foi encerrada às vinte horas e quarenta e cinco minutos. Para constar, eu, Bruno de Souza Baptista, redigi a presente ata que após lida publicamente, foi aprovada e assinada pelos membros da banca examinadora.

Betim, 20 de agosto de 2025.

Documento assinado eletronicamente por **Bruno de Souza Baptista**, **Professor**, em 20/08/2025, às 20:20, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Felipe Augusto Rocha da Silva**, **Professor EBTT**, em 20/08/2025, às 21:19, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leandro Freitas de Abreu**, **Professor**, em 25/08/2025, às 14:16, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2425409** e o código CRC **847C8A74**.

23792.001307/2025-11 2425409v1

AGRADECIMENTOS

A minha mãe, Adriane, que nunca deixou de me apoiar durante todos os anos do curso e esteve presente em todos os momentos, me dando forças para continuar e acreditando que tudo é possível. Sem você, eu não estaria onde estou e não seria quem sou.

Aos Franciscos, meus dois avôs, que sempre desempenharam o papel de pais, amigos e fonte de apoio emocional ao longo desta trajetória. Ao Chico, o meu mais sincero agradecimento e, ao Chefe, a minha eterna gratidão e saudade.

Às minhas avós Nadir e Marília, que participaram com seus conselhos e recepções calorosas, recarregando as minhas energias para cada nova jornada e cada retorno para casa.

Aos meus amigos Cleiton e Frederico, que completaram o trio e foram essenciais para que eu vencesse esta etapa, apoiando as minhas ideias, aconselhando e estando presentes quando mais precisei.

À minha irmã Letícia, por ser uma companheira tão especial e por sempre me mandar lembranças de casa.

Ao Zezinho, à Maria e ao Murilo, que trazem a alegria mais pura para a minha vida e, mesmo sem saber, são as maiores fontes de energia e motivação.

Aos meus amigos de infância, que fazem parte da minha história e, mesmo à distância ou com o passar do tempo, continuam presentes nas memórias e no coração, fortalecendo quem eu sou.

A todos os demais amigos, servidores e professores do IFMG Betim, por todo o conhecimento, momentos, histórias e risadas que compartilhamos ao longo desta jornada.

Aos meus orientadores, Bruno Baptista e Maurício Monteiro, que de última hora aceitaram embarcar comigo nesta jornada do TCC e contribuíram muito para o meu crescimento técnico e profissional.

Ao professor Leandro Freitas, que iniciou o desafio comigo e me ajudou a pavimentar o caminho para a conclusão dessa etapa.

RESUMO

A crescente complexidade dos sistemas eletrônicos embarcados em veículos modernos demanda métodos de validação mais eficientes e confiáveis. Este trabalho apresenta a aplicação de técnicas de *Hardware-in-the-Loop* (HIL) e reconhecimento de imagem para validação do *Instrument Panel Cluster* (IPC) em centrais automotivas, com foco em otimizar o processo, reduzir custos e melhorar a confiabilidade dos testes. O HIL foi integrado a ferramentas como o dSPACE SCALEXIO, MATLAB Simulink e câmeras Cognex In-Sight Série 7000, utilizando protocolos de comunicação, como CAN e TCP/IP, para simular e validar sistemas em tempo real. O processo envolveu a criação de modelos de simulação, treinamento de imagem para análise automatizada e desenvolvimento de sequências de testes no AutomationDesk. A metodologia garantiu uma análise abrangente do desempenho do IPC, verificando funcionalidades visuais e operacionais com alta precisão. Os resultados demonstraram que a abordagem proposta ajuda a reduzir o tempo de validação, é capaz de encontrar eventuais falhas e melhora a eficiência do uso de recursos humanos e computacionais. Este estudo contribui para o avanço das práticas de validação na indústria automotiva, destacando a relevância de ferramentas HIL e tecnologias de visão computacional.

Palavras-chave: *Hardware-in-the-Loop*, *Instrument Panel Cluster*, validação automotiva, softwares de simulação, reconhecimento de imagem, rede CAN, sistemas eletrônicos embarcados.

ABSTRACT

The increasing complexity of embedded electronic systems in modern vehicles demands more efficient and reliable validation methods. This study presents the application of *Hardware-in-the-Loop* (HIL) techniques and image recognition to validate the Instrument Panel Cluster (IPC) in automotive control units, focusing on process optimization, cost reduction, and improved test reliability. HIL was integrated with tools such as dSPACE SCALEXIO, MATLAB Simulink, and Cognex In-Sight Series 7000 cameras, using communication protocols like CAN and TCP/IP to simulate and validate systems in real-time. The process involved: creation of simulation models, image training for automated analysis, and the development of test sequences in AutomationDesk. The methodology ensured a comprehensive analysis of the IPC's performance, verifying visual and operational functionalities with high precision. The results demonstrated that the proposed approach helps reducing validation time, detection of potential failures, and improves the efficiency of human and computational resources. This study contributes to the advance of validation practices in the automotive industry, highlighting the relevance of HIL tools and computer vision technologies.

Keywords: *Hardware-in-the-Loop*, Instrument Panel Cluster, automotive validation, dSPACE, simulation softwares, image recognition, CAN networks, embedded electronic systems.

LISTA DE ILUSTRAÇÕES

Figura 1: Algumas funcionalidades de um veículo moderno 1
Figura 2: Exemplo de um painel de instrumentos (IPC) 2
Figura 3: Exemplo de um IPC misto 6
Figura 4: Exemplo de um IPC digital 7
Figura 5: Exemplo de ligação entre ECUs de um veículo utilizando rede CAN 8
Figura 6: Frame padrão e estendido da arquitetura da mensagem de dados CAN 9
Figura 7: Representação do teste *Hardware-in-the-Loop* 10
Figura 8: Representação do processo de testes utilizando o SCALEXIO 12
Figura 9: Aplicação do Simulink para simular as funções de uma ECU 13
Figura 10: Interface do ConfigurationDesk 15
Figura 11: Interface do AutomationDesk 16
Figura 12: Câmera Cognex In-Sight Série 7000 18
Figura 13: Exemplo de funcionamento do PatMax 19
Figura 14: Organograma do projeto 22
Figura 15: IPC utilizado nos testes 23
Figura 16: Montagem dos hardwares 24
Figura 17: Montagem da câmera e do IPC na base 25
Figura 18: Caixa projetada para reduzir o efeito da iluminação externa 26
Figura 19: Montagem final 26
Figura 20: Sequência de atuação dos softwares 28
Figura 21: Sequência do treinamento da câmera 29
Figura 22: Imagem utilizada para o treinamento da câmera 29
Figura 23: Treinamento para identificar o pop-up 31
Figura 24: Treinamento para identificar o acionamento da telltale 31
Figura 25: Treinamento da cor da barra de combustível 32
Figura 26: Treinamento da cor do ícone da bomba de combustível 33
Figura 27: Treinamento de cor para acionamento de outras telltales 34
Figura 28: Mapeamento final 35
Figura 29: Sequência de testes 37
Figura 30: Variáveis de controle 38
Figura 31: Ausência da imagem da bomba de combustível 40
Figura 32: Ausência do texto “Necessário Abastecimento” 40
Figura 33: Ausência da telltale de combustível baixo 41

Figura 34: Presença de telltale em outras regiões do IPC 42
Figura 35: Teste para verificar se há diferenciação entre branco e o amarelo treinado 42
Figura 36: Relatório contendo o resultado da seleção do treinamento (*job*) da câmera 44
Figura 37: Parte do relatório contendo o resultado para ignição desligada.....45
Figura 38: Parte do relatório contendo o resultado para ignição ligada 46
Figura 39: Parte do relatório contendo o resultado para ignição e motor ligados.....46
Figura 40: Cabeçalho do relatório quando há falha.....47
Figura 41: Indicação de qual etapa do teste falhou.....47
Figura 42: Exibição dos resultados no AutomationDesk 48
Figura 43: Previsão econômica com utilização da automação.....50
Figura 44: Variáveis de controle de tensão mapeadas 56
Figura 45: Mapeamento das variáveis da câmera.....57
Figura 46: Variáveis de seleção do treinamento no *mapping*.....57
Figura 47: Variáveis que manipularão o IPC.....58
Figura 48: Variáveis de controle.....58
Figura 49: Estrutura condensada da sequência de testes 59
Figura 50: Seleção de qual o valor do *job* será usado no teste.....60
Figura 51: Envio do comando para selecionar o *job* 61
Figura 52: Leitura da seleção do *job* no AutomationDesk.....62
Figura 53: Veredito final do SelectJOB 63
Figura 54: Inicialização dos parâmetros através do Exec 64
Figura 55: Início da sequência do TestCase 64
Figura 56: Etapa inicial da execução da sequência de teste.....65
Figura 57: Sequência de acionamento da indicação e leitura (início bloco While)....66
Figura 58: Parte final da sequência de testes do bloco While.....67
Figura 59: Verificação dos resultados do teste dentro de cada condição de ignição 67
Figura 60: Linha de código do Exec1 68
Figura 61: Final da execução do laço For 68
Figura 62: Blocos finais no AutomationDesk.....69

LISTA DE ABREVIATURAS E SIGLAS

ADAS *Advanced Driver Assist Systems* (Sistemas Avançados de Assistência ao Motorista)

AEB *Autonomous Emergency Braking* (Frenagem Automática de Emergência)

CAN *Controller Area Network* (Rede de Controle de Área)

CRC *Cyclic Redundancy Check* (Verificação de Redundância Cíclica)

DBC *Database CAN* (Base de dados CAN)

ECU *Electronic Control Unit* (Unidade de Controle Eletrônico)

FCW *Forward Collision Warning* (Alerta de Colisão Frontal)

FMU *Functional Mock-Up Unit* (Unidade de *Mock-Up* Funcional)

HDR *High Dynamic Range* (Alta Faixa Dinâmica)

HIL *Hardware-in-the-Loop* (Hardware no Circuito)

HSI *Hue, Saturation, Intensity* (Matiz, Saturação e Intensidade)

ID *Identification* (Identificação)

I/O *Input/Output* (Entrada/Saída)

IPC *Instrument Panel Cluster* (Painel de Instrumentos)

LIN *Local Interconnect Network* (Rede de Interconexão Local)

PWM *Pulse Width Modulation* (Modulação por Largura de Pulso)

RGB *Red, Green, Blue* (Vermelho, Verde, Azul)

RTI CAN *Real-Time Interface for CAN* (Interface em Tempo Real para CAN)

SDRAM *Synchronous Dynamic Random-Access Memory* (Memória de Acesso Aleatório Dinâmica Sincronizada)

TCP/IP *Transmission Control Protocol/Internet Protocol* (Protocolo de Controle de Transmissão/Protocolo de Internet)

SUMÁRIO

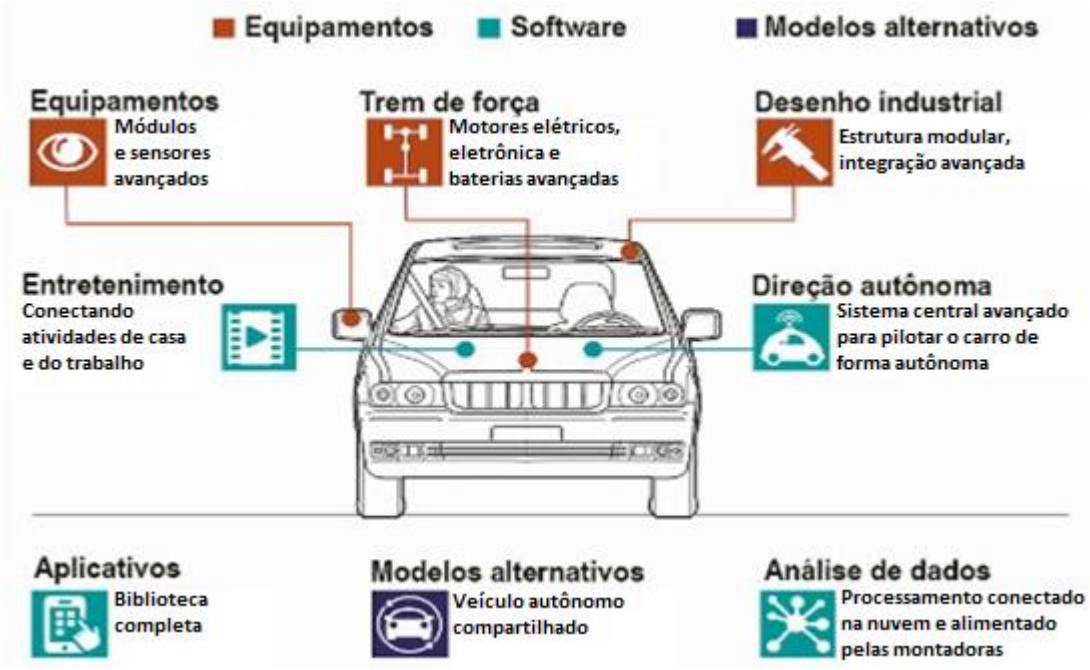
1 INTRODUÇÃO 1
1.1 Justificativa 3
2 OBJETIVOS 4
2.1 Objetivos específicos 4
3 FUNDAMENTAÇÃO TEÓRICA 5
3.1 ECU IPC (Instrument Panel Cluster) 5
3.2 Rede CAN (Controller Area Network) 7
3.3 *Hardware-in-the-Loop* 10
3.4 Visão computacional 11
3.4.1 Modelagem de sistemas com Simulink 12
3.4.2 Ferramentas de suporte a automação 14
3.4.3 Captura de Imagem em sistemas de visão computacional 17
3.4.4 Ferramentas de reconhecimento visual 18
4 METODOLOGIA 21
4.1 Materiais Utilizados 24
4.2 Conexão dos hardwares e montagem do setup 24
4.3 Criação do modelo no Simulink e integração entre softwares 27
4.4 Reconhecimento de imagem 28
4.5 Elaboração da sequência de testes 35
5 RESULTADOS 39
5.1 Validação do treinamento de imagem 39
5.2 Validação do teste automático 43
5.3 Impacto financeiro 48
5.4 Impacto na produtividade 50
6 CONCLUSÃO 52
6.1 Sugestões para trabalhos futuros 53
REFERÊNCIAS 54
APÊNDICES 56

1 INTRODUÇÃO

A validação de centrais automotivas é um dos processos mais críticos no desenvolvimento de veículos modernos. À medida que a complexidade dos sistemas eletrônicos embarcados cresce, impulsionada pela evolução de tecnologias como redes CAN (*Controller Area Network*) e controladores avançados, também aumenta a necessidade de garantir a confiabilidade, a segurança e a eficiência desses sistemas. Os métodos tradicionais de validação frequentemente demandam tempo e mão de obra intensiva, criando desafios para indústrias que buscam agilidade e redução de custos no desenvolvimento de produtos.

No atual contexto do mercado automotivo global, marcado pela transição para veículos elétricos e híbridos, por sistemas avançados de assistência ao condutor (ADAS) e pela digitalização do *cockpit*, o peso da eletrônica embarcada está cada vez mais evidente, como pode ser visto na Figura 1. Segundo Górski e Janiszewski (2022), a integração de múltiplos sensores, unidades de controle e interfaces de comunicação aumenta exponencialmente a complexidade dos sistemas eletrônicos e evidencia a importância de métodos de teste e simulação capazes de reproduzir situações realistas e detectar falhas precocemente.

Figura 1: Algumas funcionalidades de um veículo moderno



Fonte: Adaptado de OficinaBrasil

A importância do *Instrument Panel Cluster* (IPC), componente mostrado na Figura 2, neste contexto torna-se particularmente clara. Por representar a interface direta entre o veículo e o motorista, o IPC não é apenas um componente estético, mas um sistema para a comunicação de informações sobre o status do veículo. Qualquer anomalia ou inconsistência no seu funcionamento pode comprometer a percepção do condutor e prejudicar a segurança e a experiência de condução.

Figura 2: Exemplo de um painel de instrumentos (IPC)

Fonte: AutosSegredos

Nesse contexto, o uso de técnicas inovadoras, como *Hardware-in-the-Loop* (HIL) e o reconhecimento de imagem, surgem como alternativas promissoras para aprimorar os processos de validação. Segundo Zhang e Wang (2017), o HIL permite realizar testes complexos em sistemas automotivos combinando modelos simulados com hardware real, enquanto o reconhecimento de imagem automatiza a verificação de parâmetros visuais, como aqueles exibidos no IPC de veículos. Essas tecnologias têm demonstrado benefícios significativos em termos de eficiência e redução de custos no desenvolvimento automotivo (SOMMER et al., 2016). Østergaard et al. (2019) destaca que o HIL permite verificar e validar ECUs como o IPC antes mesmo de estarem integradas ao veículo físico, ampliando a cobertura de testes e reduzindo riscos de falhas não detectadas

1.1 Justificativa

A indústria automotiva onde o projeto foi desenvolvido possui diversas outras ECUs sendo validadas com o uso de HIL, mas nenhum desses *setups* é preparado para validação de IPC. Sendo a central eletrônica que fornece informações sobre o funcionamento do veículo para o motorista, é necessário garantir que o que é exibido pelo IPC seja correto e confiável. O uso de HIL e validação por imagem é importante para auxiliar no processo de validação, trazendo uma maior uniformidade, confiabilidade, produtividade, redução do tempo e dos custos.

2 OBJETIVOS

O objetivo deste trabalho é desenvolver, testar e validar a utilização de HIL na validação da ECU IPC através de um modelo real aplicado na indústria automotiva, com foco em aumentar a confiabilidade e produtividade das validações e liberar mão de obra para validação de novas funcionalidades dos IPCs e desenvolvimento de outros projetos.

2.1 Objetivos específicos

- Projetar e construir um ambiente de validação da ECU IPC utilizando HIL;
- Determinar quais funções do IPC podem ser automatizadas;
- Determinar quais hardware*s* de HIL melhor atendem as necessidades do projeto;
- Integrar as diferentes tecnologias de hardware e software para construir o ambiente;
- Validar o ambiente construído;
- Quantificar os impactos financeiros e de produtividade em ambiente de produção;
- Reduzir o tempo gasto em testes de verificação das funcionalidades do software após novas versões;

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos e tecnologias fundamentais para o desenvolvimento deste trabalho, abordando inicialmente a arquitetura e funcionamento da ECU IPC (*Instrument Panel Cluster*) e a comunicação por meio da rede CAN. Em seguida, discute-se a aplicação da técnica de *Hardware-in-the-Loop* (HIL) e os princípios de visão computacional utilizados para análise e validação de sistemas automotivos.

3.1 ECU IPC (Instrument Panel Cluster)

As ECUs (*Electronic Control Units*) são unidades eletrônicas responsáveis por controlar e monitorar diferentes funções e/ou componentes do veículo, como motor, transmissão, sistemas de segurança e entretenimento. Cada ECU é programada para executar tarefas específicas e interagir com outras unidades por meio de redes de comunicação internas, como a CAN (*Controller Area Network*) (BOYCE, 2014).

O IPC, por sua vez, é uma ECU onde os motoristas visualizam as informações essenciais do veículo. O seu funcionamento envolve a interação de diversos sensores, unidades de controle eletrônico e redes de comunicação como a CAN, para coletar e transmitir informações de maneira eficiente e em tempo real para o motorista. A principal função do IPC é permitir que o motorista acompanhe, de forma clara e precisa, o comportamento do veículo, garantindo uma condução segura e eficiente. O IPC é composto por telas analógicas, digitais ou mistas (ponteiros analógicos e um display digital por exemplo, como mostra a Figura 3), e sua confiabilidade é crítica para a segurança e a experiência do usuário no veículo (GOSSLING et al., 2019).

Figura 3: Exemplo de um IPC misto

Fonte: AprovaDetran

A arquitetura do IPC é altamente integrada aos sistemas eletrônicos do veículo, utilizando interfaces digitais e analógicas para representar as informações necessárias. Os painéis de instrumentos modernos, ao contrário dos modelos tradicionais analógicos, frequentemente empregam *displays* digitais de alta resolução, gráficos dinâmicos e até telas sensíveis ao toque. Esses sistemas permitem a adaptação das informações conforme as condições do veículo e as preferências do motorista. Os *displays* digitais não só proporcionam uma visualização mais clara e intuitiva, mas também possibilitam a personalização da interface, o que melhora a interação entre o veículo e o motorista. A Figura 4 exemplifica essa evolução ao apresentar a arquitetura e a disposição visual de um IPC digital.

Figura 4: Exemplo de um IPC digital

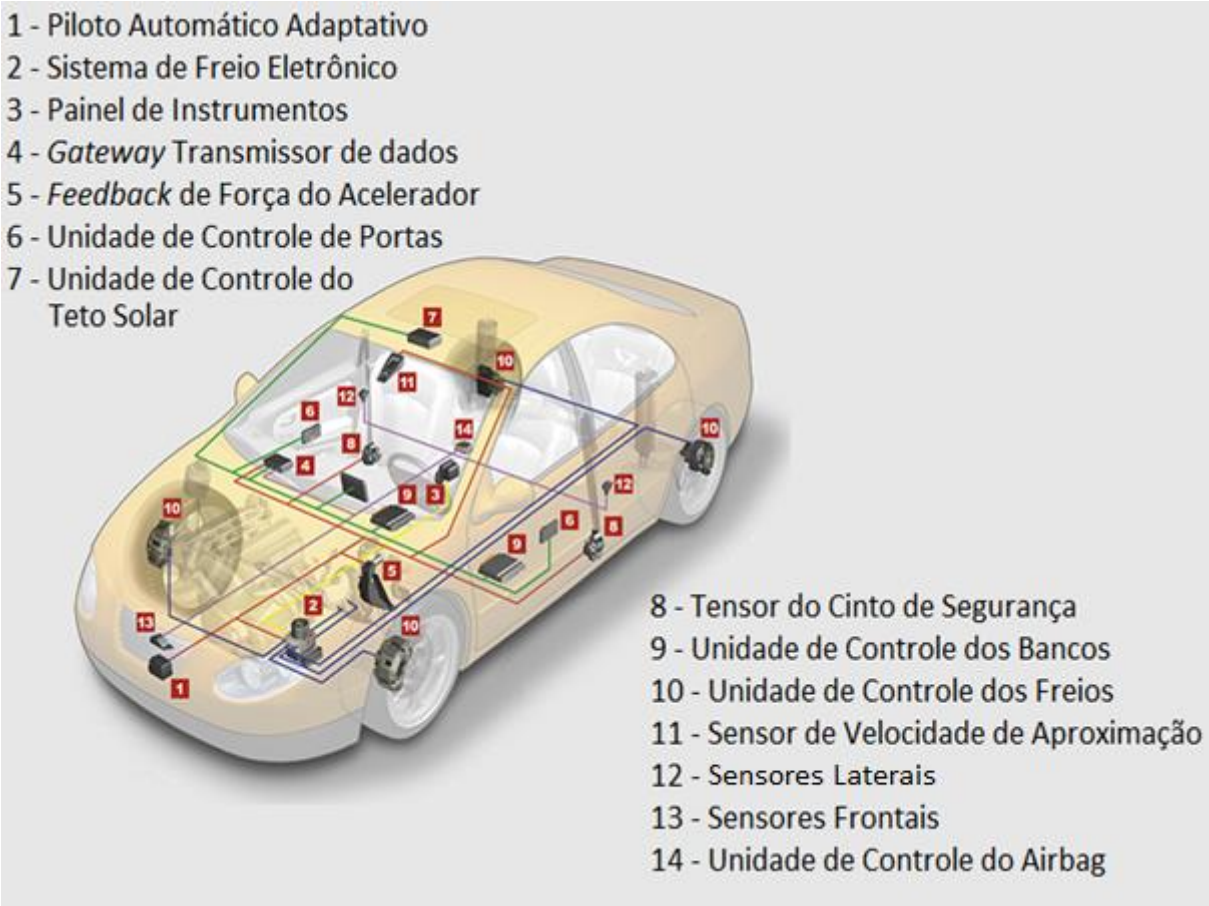
Fonte: Portal Lubes

A validação do IPC é um aspecto fundamental para garantir o bom funcionamento do sistema e, por consequência, a segurança do veículo. A realização de testes rigorosos e uma validação robusta são essenciais para verificar a confiabilidade do painel, garantindo que ele forneça informações precisas em tempo real, sem falhas ou comportamentos inesperados. A validação de sistemas críticos, como o IPC, envolve testes de integridade e resiliência, simulando diferentes condições de operação para assegurar que o painel continue funcionando de maneira eficiente, mesmo em situações adversas. Esse processo de validação assegura que o sistema seja resistente a falhas, mantendo a precisão das informações transmitidas, permitindo a tomada de decisões rápidas e seguras durante a condução.

3.2 Rede CAN (Controller Area Network)

A *Controller Area Network* (CAN) foi desenvolvida pela Bosch na década de 1980 com o objetivo de fornecer comunicação eficiente entre as ECUs (Unidades de Controle Eletrônico) em automóveis, através de uma rede com menos cabos e melhor organização, conforme mostra a Figura 5. De acordo com a Bosch (1991), a CAN foi projetada para suportar ambientes adversos típicos dos automóveis, como vibrações e interferências eletromagnéticas. Sua principal função é utilizar um barramento de comunicação compartilhado, permitindo que múltiplas ECUs transmitam dados sem a necessidade de um controlador centralizado.

Figura 5: Exemplo de ligação entre ECUs de um veículo utilizando rede CAN



Fonte: Adaptado de AA1Car

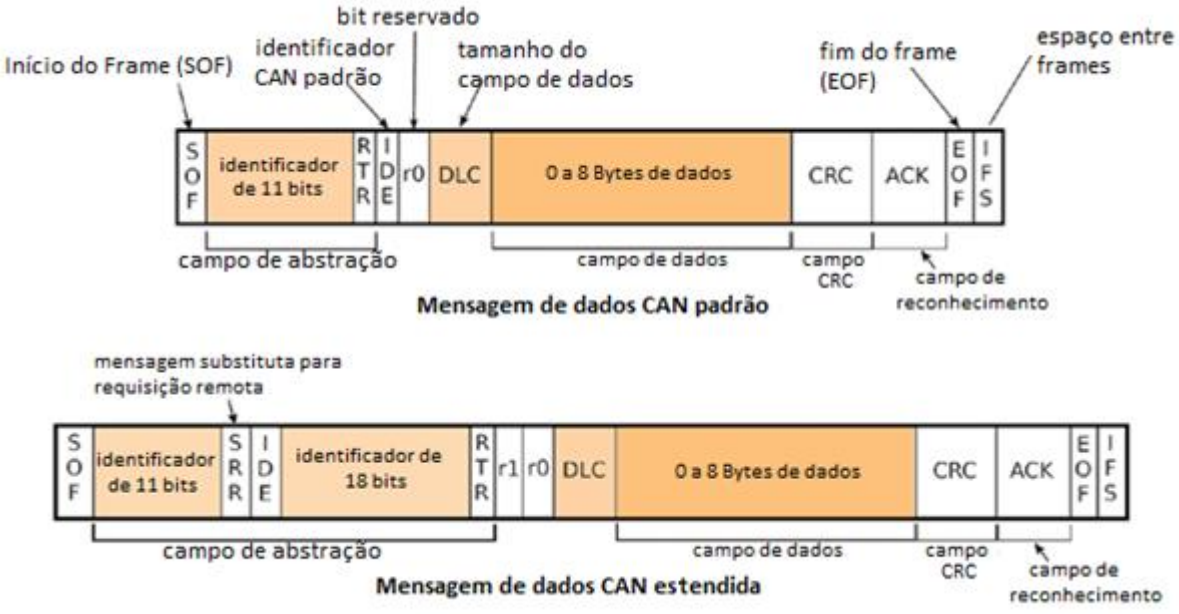
A comunicação em uma rede CAN ocorre através de dois fios principais: CAN *High* e CAN *Low*. Esses dois fios transmitem sinais de forma diferencial, ou seja, os dados são transferidos com níveis de tensão opostos. Isso garante melhor imunidade a ruídos elétricos e interferências, comuns em ambientes automotivos. O uso dessa técnica de transmissão diferencial contribui significativamente para as capacidades da rede. Mesmo em situações de alto ruído, o receptor pode calcular a diferença de tensão entre os dois fios para recuperar as informações de forma rápida e segura.

O protocolo CAN é multi-mestre. Isso significa que qualquer ECU (unidade de controle eletrônico) na rede de barramento CAN pode iniciar a comunicação. Porém, o controle do barramento é distribuído entre as ECUs participantes. Esse modelo de comunicação descentralizado oferece mais flexibilidade e robustez à rede. Portanto, não há necessidade de depender de um único ponto de falha. A comunicação na rede CAN é assíncrona. Isso significa que não é necessário

compartilhar um relógio comum entre a ECU e todos os dispositivos conectados. Isso reduz a complexidade da rede. De acordo com Ribeiro (2018), essa mesma arquitetura é o que torna as redes CAN tão confiáveis, pois não há um único ponto de falha.

A transmissão de dados na rede CAN é feita por meio de quadros, que contêm dados de identificação, dados reais e alguns campos de verificação. Cada um desses quadros possui um identificador de 11 bits (na versão padrão) ou de 29 bits (na versão estendida), conforme mostra a Figura 6. Esse identificador é necessário para determinar a prioridade da mensagem. A priorização das mensagens é feita por meio do conceito de arbitragem. Isso significa que, quando mais de uma ECU tenta transmitir ao mesmo tempo, a ECU com a mensagem de maior prioridade é a que será transmitida primeira no barramento, seguida pelas mensagens das demais ECUs, de acordo com a sua ordem de prioridade.

Figura 6: Frame padrão e estendido da arquitetura da mensagem de dados CAN



Fonte: Adaptado de Dewesoft

A CAN também utiliza um mecanismo de verificação de erros para garantir que os dados transmitidos sejam válidos. O protocolo implementa técnicas como a verificação de redundância cíclica (CRC) e *bit stuffing*. Essas técnicas são responsáveis por detectar e corrigir erros de transmissão, assegurando a integridade dos dados em condições de operação adversas. Esse alto nível de confiabilidade,

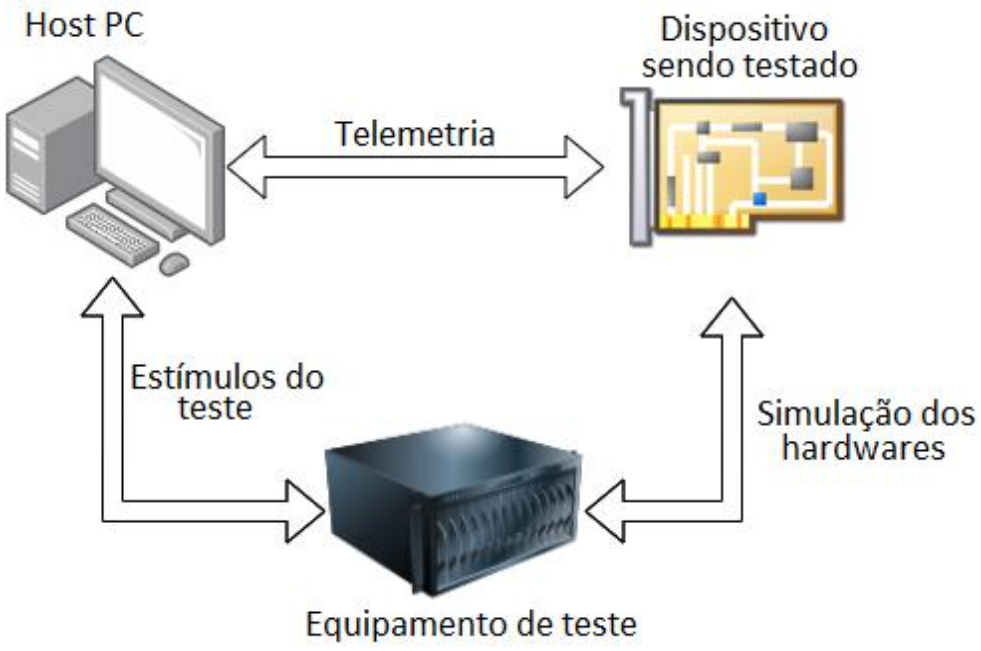
aliado à capacidade de detectar e corrigir erros rapidamente, faz com que a CAN seja a escolha ideal para aplicações automotivas onde a segurança e a precisão são determinantes.

Compreender a comunicação entre ECUs através da rede CAN é fundamental para analisar o funcionamento de sistemas como o Instrument Panel Cluster (IPC), um dos principais objetos de estudo deste trabalho.

3.3 *Hardware-in-the-Loop*

Hardware-in-the-Loop (HIL) é uma técnica de validação usada para testar sistemas embarcados, especialmente encontrados em veículos modernos. A Figura 7 mostra como esse método combina componentes de hardware reais com um ambiente de simulação computacional, permitindo a interação entre o sistema em desenvolvimento e o ambiente virtual. De acordo com Sommer et al. (2016), o HIL oferece uma plataforma de testes onde os sistemas físicos operam ao lado das simulações, tornando possível analisar o comportamento do sistema em um ambiente controlado e seguro. O HIL é amplamente utilizado no desenvolvimento de sistemas automotivos, como unidades de controle eletrônico (ECUs), para testar sua resposta sob diversas condições. Isso reduz os custos e os riscos de falhas nas primeiras etapas do desenvolvimento.

Figura 7: Representação do teste *Hardware-in-the-Loop*



Fonte: Adaptado de Altium

O funcionamento do HIL depende de dois componentes principais: o modelo de simulação e o hardware real. O modelo de simulação, que pode incluir um modelo de veículo ou sistemas do veículo (como motor, transmissão ou sistema de frenagem), interage com o hardware real por meio de uma interface que permite a troca de dados em tempo real. A principal vantagem dessa abordagem é que ela permite testar o comportamento completo do sistema, incluindo a interação com sensores e atuadores, antes da implementação final. Segundo Zhang e Wang (2017), o uso do HIL torna o processo de validação mais rápido e eficiente, pois erros podem ser identificados em estágios iniciais e corrigidos sem a necessidade de testes físicos do protótipo. O HIL facilita o desenvolvimento de software, permitindo testar e ajustar código de controle em tempo real, sem depender da disponibilidade de hardware físico.

O *Hardware-in-the-Loop* utiliza plataformas como o dSPACE SCALEXIO, Typhoon HIL, OPAL-RT, entre outros, para conectar simulações computacionais a hardware real. Essa integração será detalhada a seguir.

3.4 Visão computacional

A validação de sistemas embarcados veiculares, como o Instrument Panel Cluster (IPC), envolve a verificação combinada de respostas funcionais e visuais. A integração entre simulação em tempo real e inspeção automatizada por imagem tem sido amplamente adotada na literatura, especialmente em aplicações onde repetibilidade e precisão são requisitos frequentes (HILLEBRAND et al., 2020).

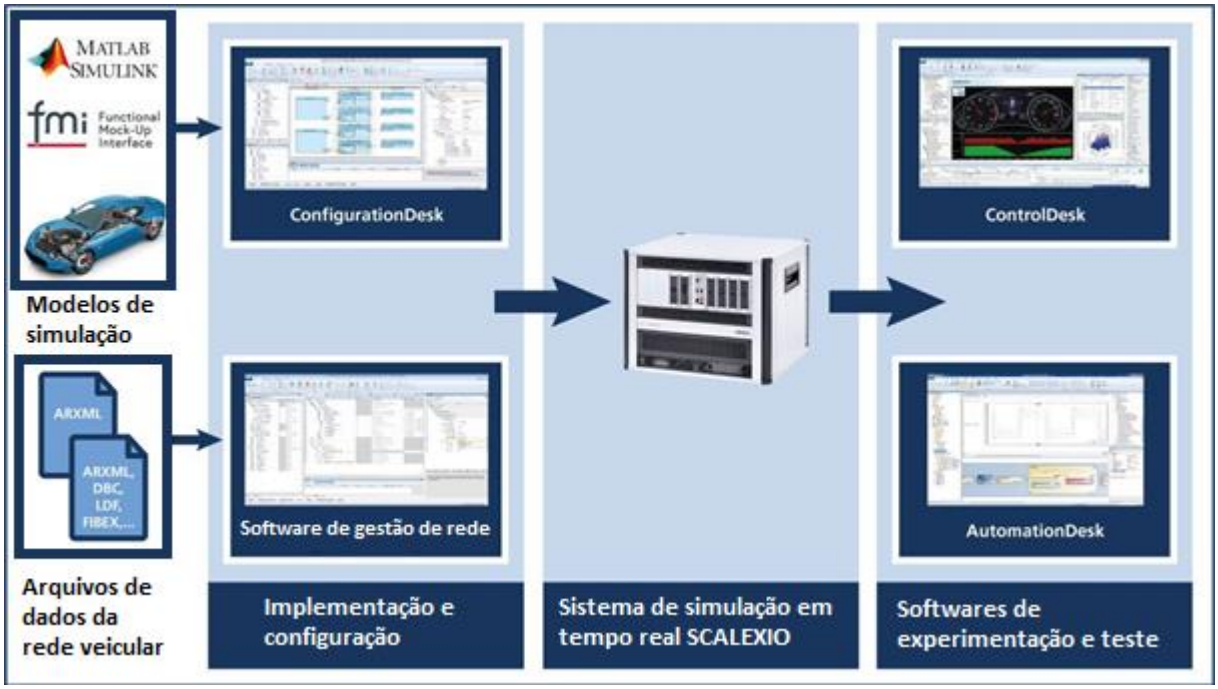
Essa integração é viabilizada por dois componentes tecnológicos complementares: a técnica de Hardware-in-the-Loop (HIL), aplicada à simulação de unidades de controle eletrônico (ECUs), e o uso de sistemas de visão computacional para a análise dos elementos gráficos exibidos ao condutor. Esta seção apresenta os fundamentos teóricos dessas abordagens e destaca os principais algoritmos e ferramentas empregados em sua aplicação.

Além disso, considerando que os dados exibidos no IPC são provenientes de redes embarcadas, como a CAN (Controller Area Network), falhas na integridade das mensagens — como erros de sincronização, checksum inválido ou identificadores incorretos — podem comprometer a exibição das informações visuais. Por esse

motivo, a inspeção por imagem atua como uma etapa complementar à validação lógica.

A Figura 8 apresenta o fluxo de desenvolvimento e teste de sistemas embarcados utilizando a plataforma SCALEXIO da dSPACE. O processo inicia com a modelagem do sistema no MATLAB Simulink e a configuração de redes veiculares no ConfigurationDesk, incluindo arquivos com as informações sobre os sinais que trafegam na rede do veículo. Após essa etapa, o modelo é executado em tempo real no SCALEXIO. Durante os testes, o AutomationDesk automatiza a execução dos cenários e o ControlDesk permite o monitoramento e a calibração de sinais, garantindo uma validação integrada entre simulação, comunicação e análise de resultados.

Figura 8: Representação do processo de testes utilizando o SCALEXIO



Fonte: Adaptado de dSPACE

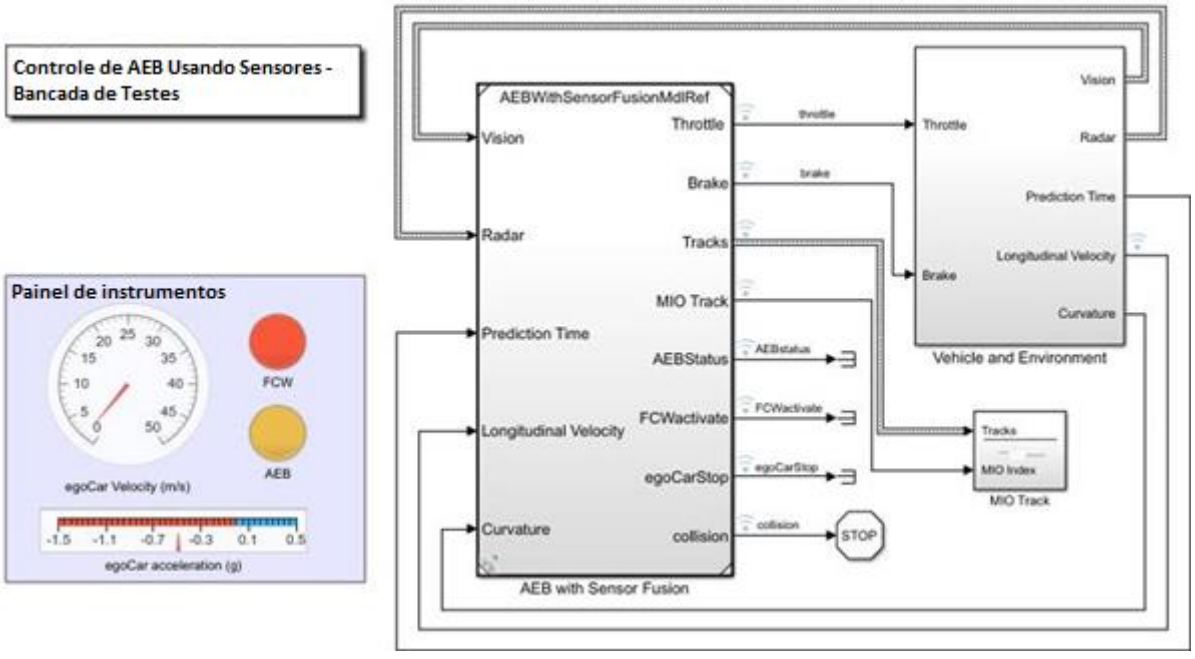
3.4.1 Modelagem de sistemas com Simulink

A modelagem de sistemas para testes HIL pode ser realizada com ferramentas gráficas como o MATLAB Simulink. Essa plataforma permite representar a lógica de funcionamento de uma ECU por meio de blocos interconectados, simulando seu comportamento sob diferentes condições operacionais (MATHWORKS, 2025).

Para aplicações automotivas, o Simulink pode ser integrado à biblioteca RTI CAN (*Real-Time Interface for CAN*), desenvolvida pela dSPACE. Essa biblioteca permite a configuração de canais de comunicação compatíveis com a rede CAN, facilitando a simulação e monitoramento de mensagens em tempo real. Com ela, é possível mapear sinais diretamente dos modelos no Simulink, transmitindo e recebendo mensagens CAN com precisão e sincronização com o restante do ambiente HIL. A geração automática de código e a compatibilidade com ambientes de automação de testes tornam a ferramenta adequada para aplicações industriais.

A Figura 9 mostra uma bancada de testes em Simulink para um sistema de frenagem autônoma de emergência (AEB) com fusão de sensores. Dados simulados de radar e visão são processados para calcular variáveis como tempo de predição, velocidade e risco de colisão. Com essas informações, o sistema decide quando acionar alertas (FCW) ou a frenagem automática (AEB), enquanto um painel exibe em tempo real a resposta do veículo. A simulação permite validar o desempenho do AEB em diferentes cenários de tráfego.

Figura 9: Aplicação do Simulink para simular as funções de uma ECU



Fonte: Adaptado de Mathworks

Após a modelagem dos sistemas e a configuração das comunicações no ambiente Simulink, ferramentas específicas são utilizadas para dar suporte à automação e ao monitoramento dos testes em ambientes Hardware-in-the-Loop (HIL).

Entre essas ferramentas destacam-se o AutomationDesk e o ConfigurationDesk, que facilitam a execução dos testes e a análise dos resultados em tempo real.

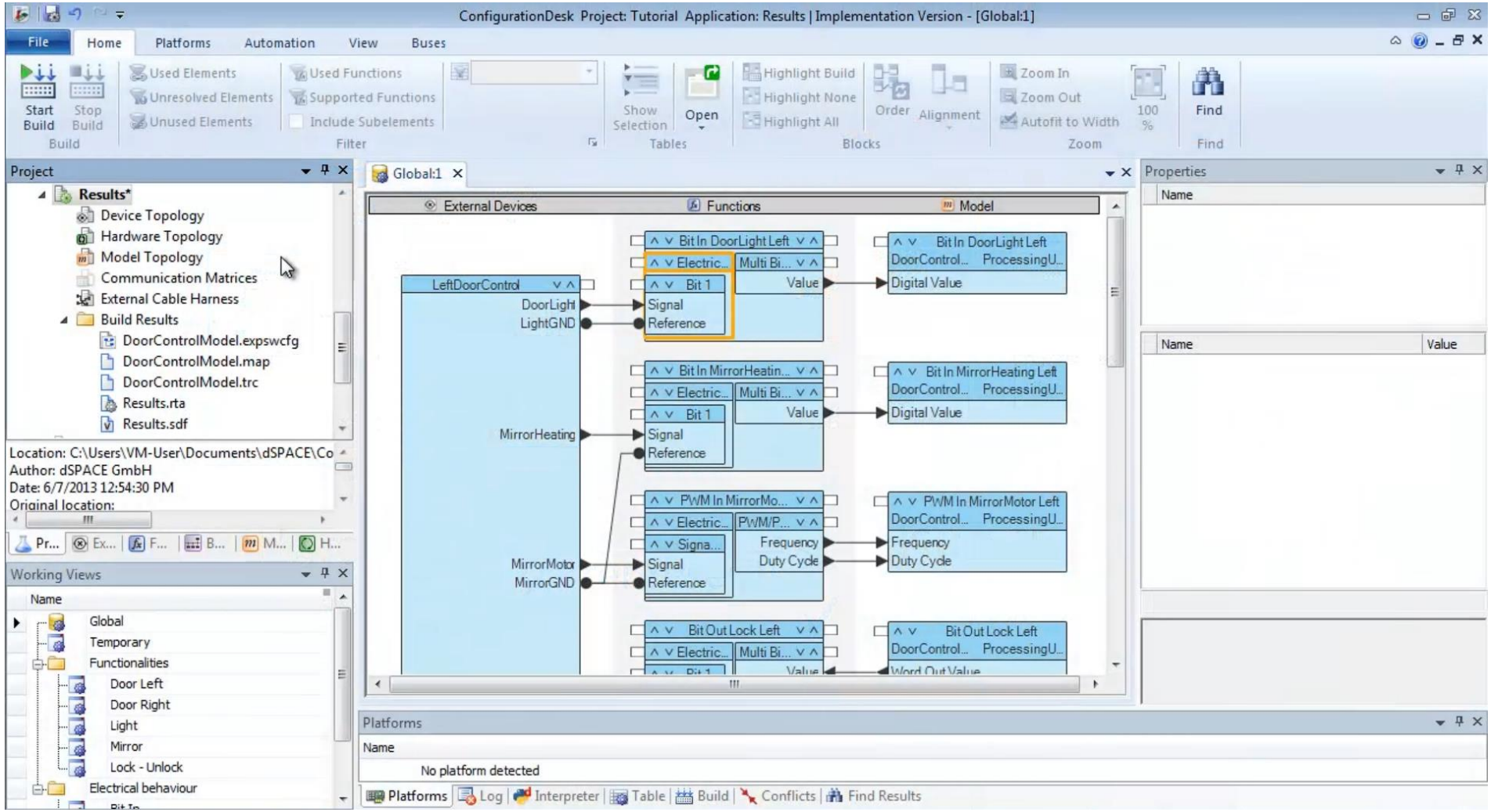
3.4.2 Ferramentas de suporte a automação

Na validação de sistemas embarcados automotivos a sincronização entre sinais eletrônicos e respostas visuais é fundamental. O IPC recebe dados via rede CAN (*Controller Area Network*), que transporta mensagens de múltiplas unidades de controle eletrônico (ECUs) para exibir informações ao condutor, tais como velocidade, nível de combustível e alertas visuais. A complexidade dessa comunicação exige ferramentas que permitam a orquestração precisa dos testes e o monitoramento simultâneo de sinais e comportamentos visuais.

No contexto de simulação em tempo real e testes Hardware-in-the-Loop (HIL), o ConfigurationDesk configura e mapeia sinais de entrada e saída (I/Os) definidos no modelo MATLAB Simulink. A ferramenta estabelece a interface entre os sinais virtuais e o hardware físico. Também permite o monitoramento em tempo real dos valores desses sinais durante os testes, facilitando a validação e calibração para garantir a correta interação entre o modelo e o sistema físico.

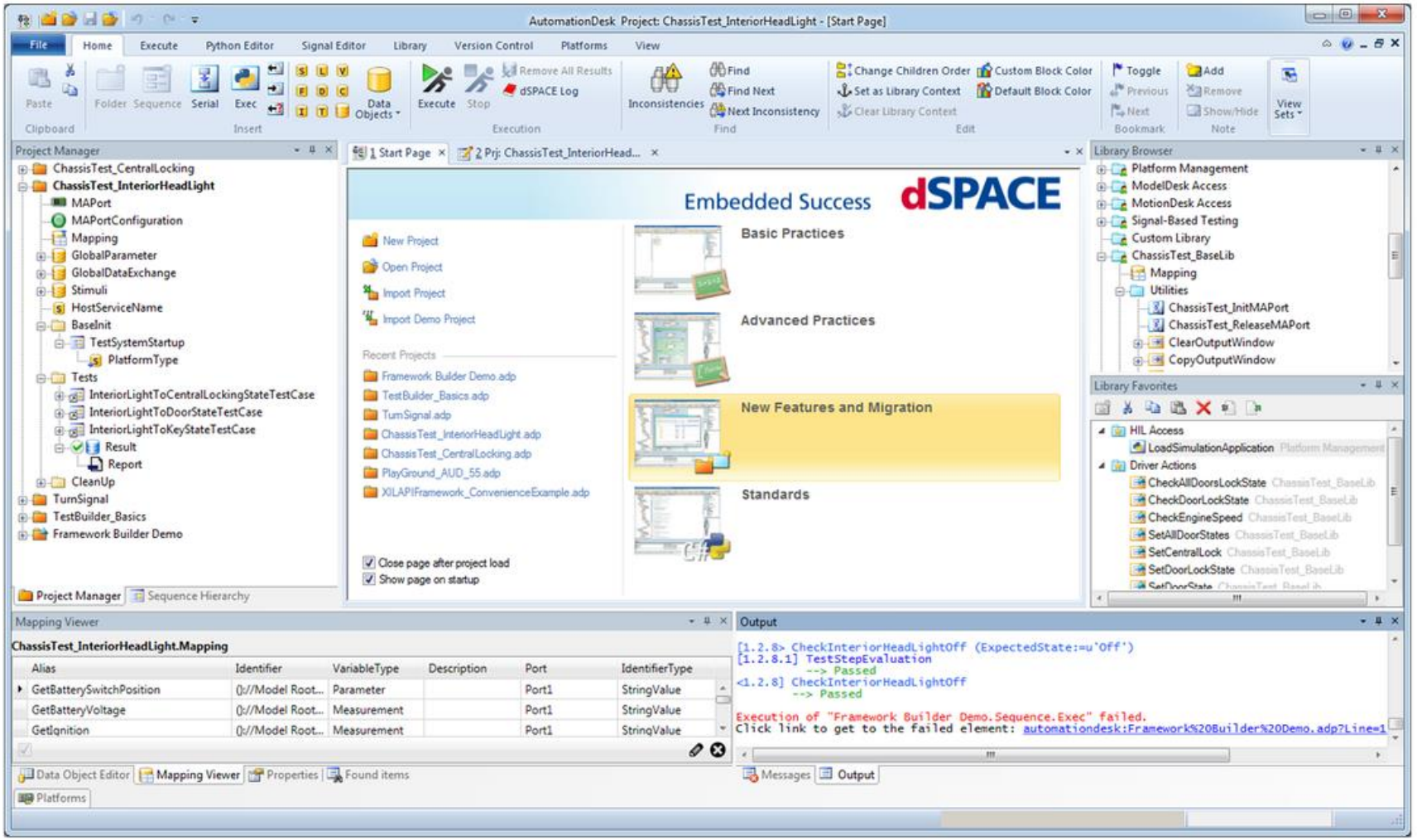
O AutomationDesk é utilizado para a criação e execução de sequências automatizadas de testes em ambientes HIL. Essa ferramenta possibilita a simulação controlada de mensagens CAN que alimentam o IPC, permitindo a verificação do comportamento da interface gráfica sob diferentes condições simuladas. O AutomationDesk gerencia o envio de pacotes CAN, sincronização com eventos internos e coleta de dados, garantindo que os testes sejam repetíveis e documentados. A automação reduz a ocorrência de erros humanos e facilita a realização de testes complexos, envolvendo múltiplas variáveis de entrada.

Figura 10: Interface do ConfigurationDesk



Fonte: dSPACE

Figura 11: Interface do AutomationDesk



Fonte: dSPACE

A integração dessas ferramentas com sistemas de visão computacional, por exemplo, câmeras inteligentes configuradas via In-Sight Explorer, viabiliza a análise sincronizada da resposta visual do IPC. A captura de imagens pode ser disparada em momentos específicos do ciclo de testes, correlacionando alterações visuais detectadas por algoritmos como PatMax ou pela análise de cor com os sinais CAN simulados e monitorados. Essa abordagem possibilita uma validação mais abrangente, ao verificar não somente os dados eletrônicos, mas também a correta exibição visual das informações ao usuário final.

3.4.3 Captura de Imagem em sistemas de visão computacional

A captura de imagem é a etapa inicial nos sistemas de visão computacional, responsável por fornecer os dados visuais a serem analisados. A qualidade da imagem adquirida depende de fatores como resolução, tempo de exposição, profundidade de cor e taxa de quadros. Estes fatores influenciam diretamente a acurácia dos algoritmos de inspeção (GONZALEZ; WOODS, 2018).

Variações de iluminação, ruído eletrônico e reflexos podem afetar negativamente o processo de análise. Por esse motivo, é comum empregar técnicas de pré-processamento, como filtros espaciais e equalização de histograma, para melhorar a qualidade da imagem (RUSSELL et al., 2016). A escolha da câmera e dos parâmetros de aquisição deve considerar as características da aplicação, como contraste, velocidade de resposta e estabilidade da cena.

Em sistemas embarcados, a sincronização entre a captura da imagem e os eventos internos — como a atualização de um sinal CAN — é frequentemente realizada com o uso de sinais digitais de disparo (*triggers*), de modo a alinhar a aquisição de imagem com o estado do sistema analisado. Empresas especializadas em visão computacional, como Cognex (Figura 12), Basler e Teledyne DALSA, oferecem câmeras industriais e ferramentas de desenvolvimento voltadas a aplicações embarcadas, com recursos para controle de captura, integração com protocolos de comunicação e suporte a algoritmos de inspeção em tempo real.

Figura 12: Câmera Cognex In-Sight Série 7000

Fonte: Cognex

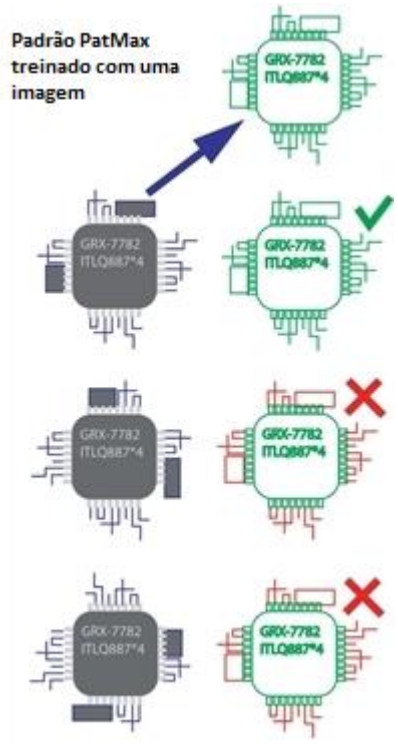
3.4.4 Ferramentas de reconhecimento visual

Dentre os algoritmos aplicados na indústria para reconhecimento visual, destacam-se aqueles baseados em correspondência geométrica (*pattern matching*) e análise espectral (*color analysis*). Ambos são empregados em tarefas de inspeção visual com requisitos de confiabilidade e repetição.

O algoritmo PatMax realiza a detecção de padrões com base na orientação dos gradientes de contorno, e não na intensidade dos pixels. Essa abordagem reduz a sensibilidade do método a mudanças de iluminação, rotação ou escala (COGNEX, 2025b). Durante o processo de treinamento, o PatMax extrai contornos da imagem de referência e constrói um modelo vetorial que representa as bordas relevantes e sua estrutura espacial, conforme é mostrado na Figura 13. Esse modelo é então usado como base para busca em imagens subsequentes.

Durante a inspeção, o algoritmo percorre a imagem adquirida procurando regiões que apresentem gradientes de contorno semelhantes ao modelo. A correspondência é feita por meio de um índice de similaridade, que considera a sobreposição geométrica e a orientação dos gradientes. Caso o índice calculado ultrapasse um limiar pré-definido, o padrão é considerado detectado. O método é particularmente eficaz em ambientes industriais devido à sua tolerância a ruídos, distorções geométricas e pequenas variações de foco ou brilho.

Figura 13: Exemplo de funcionamento do PatMax



Fonte: Adaptado de Cognex

Em aplicações com superfícies reflexivas, ruído visual elevado ou contornos mal definidos, a eficácia do PatMax pode ser reduzida (ZHANG et al., 2019). Nesses casos, ajustes no pré-processamento e nos parâmetros de sensibilidade são recomendados para melhorar o desempenho.

O algoritmo Color analisa a distribuição de cores em espaços RGB (*Red, Green, Blue*) ou HSI (*Hue, Saturation, Intensity*), permitindo a identificação de regiões com tonalidades semelhantes àquelas treinadas. A segmentação cromática é útil para verificar alterações visuais como mudanças de ícones ou indicadores, especialmente em sistemas que utilizam cores para sinalizar estados de operação.

Apesar de ser eficaz em muitos contextos, esse método apresenta limitações quando há variações de iluminação ambiente ou sombras que alterem a percepção da cor (SIVARAMAN; TRIVEDI, 2013). A calibração da câmera e o controle do ambiente de iluminação são estratégias comuns para mitigar esses efeitos.

O uso combinado dos dois algoritmos permite abordar casos onde há necessidade de verificar tanto a forma quanto a coloração dos elementos inspecionados. A literatura também aponta que a aplicação de visão computacional

pode complementar outras abordagens de validação embarcada, permitindo detectar falhas visuais que não seriam captadas apenas por testes lógicos ou elétricos (SANTOS et al., 2020).

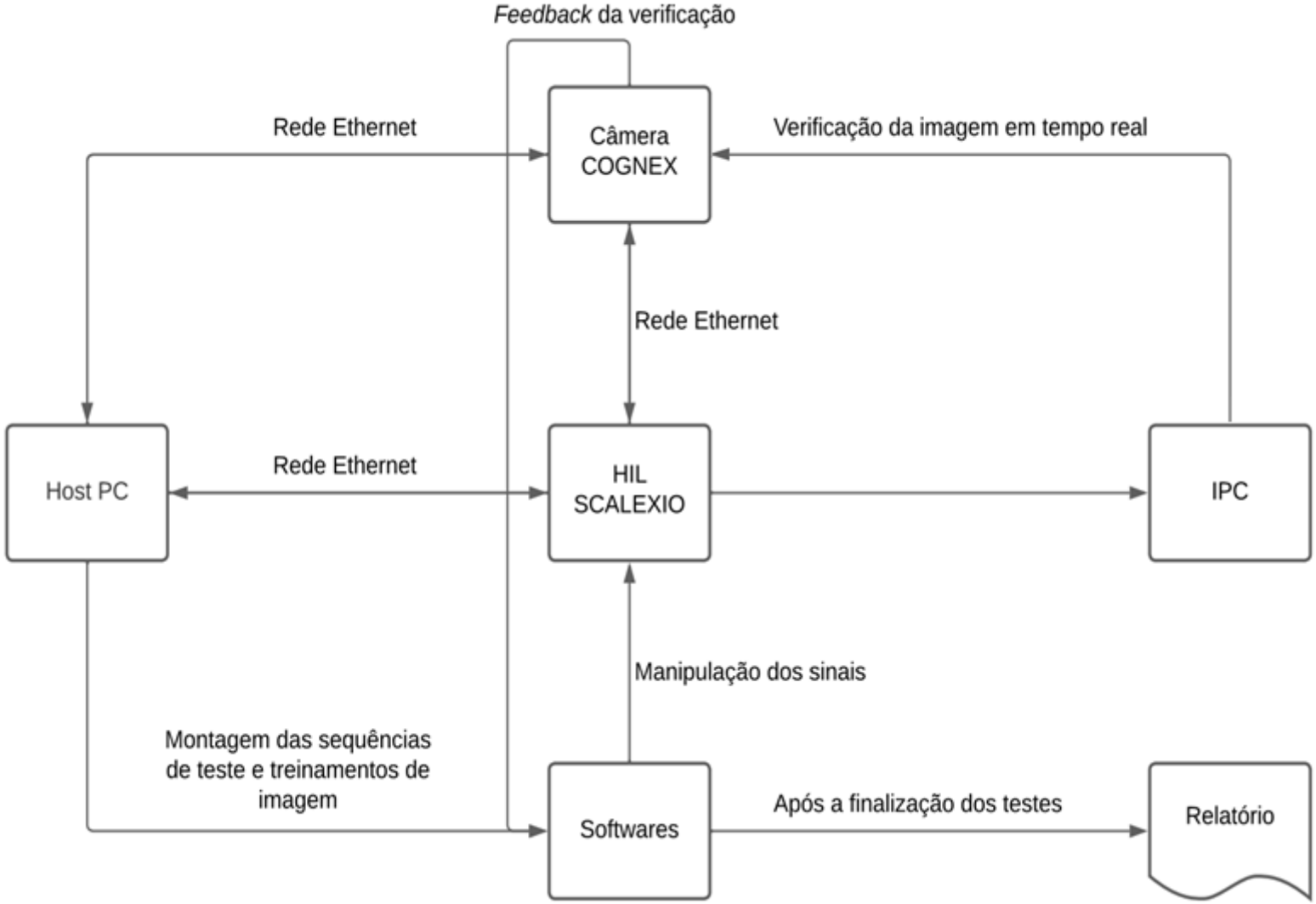
4 METODOLOGIA

Apesar de o HIL ser uma prática consolidada para validar ECUs automotivas dentro da empresa, sua aplicação para o IPC ainda não estava implementada. O IPC desempenha uma função importante ao informar o motorista sobre as condições do veículo e por isso requer testes abrangentes, capazes de verificar não só a lógica interna, mas também a exibição correta e clara das informações visuais.

Neste contexto, foi desenvolvido e validado um ambiente de testes HIL integrado à inspeção por imagem para ECUs IPC, simulando uma ampla diversidade de situações e cenários críticos. O sistema construído alia hardware e software especializados para replicar e verificar todas as funções do IPC, desde a comunicação interna com demais ECUs até a exibição gráfica para o usuário. Dessa forma, tornou-se possível automatizar o *smoke-test* (que é uma verificação sistemática de todas funções do IPC a cada nova versão de software) aumentando a cobertura e a repetibilidade das análises e permitindo a identificação de problemas que estavam fora do escopo de alteração.

A Figura 14 ilustra a arquitetura do ambiente de testes desenvolvido para validar o IPC por meio de HIL e inspeção por imagem. O Host PC contém os softwares necessários para o projeto, que executam a montagem das sequências de teste, treinamentos de imagem, envio de comandos para o HIL SCALEXIO e para a câmera Cognex via rede Ethernet. O HIL SCALEXIO executa as simulações em tempo real e envia os sinais simulados para o IPC e os comandos para a câmera, esta última realiza a captura e processamento das imagens apresentadas no IPC, encaminhando o *feedback* para os softwares, que geram os relatórios detalhados após a finalização de cada ciclo de testes.

Figura 14: Organograma do projeto



Fonte: Autor

Foi escolhido um IPC de 10 polegadas, mostrado na Figura 15, para validação do ambiente construído, por possuir um volume maior de indicações, softwares e consumir um maior tempo durante o processo de validação em relação a outros IPCs (3,5 e 7 polegadas) produzidos pela empresa. Essa escolha também está alinhada com a direção dos próximos projetos da empresa, que estão migrando para o uso de IPCs 100% digitais. É importante destacar que a empresa possui, até o momento da escrita deste trabalho, outros quatro projetos que utilizam IPCs de tamanho e arquitetura semelhantes, o que facilita a utilização do mesmo setup para a validação desses sistemas.

Em contrapartida, IPCs analógicos apresentam maior variabilidade nas indicações visuais, devido a fatores como diferenças mecânicas dos ponteiros e calibração individual. Essa variabilidade pode impactar a consistência do reconhecimento por imagem, exigindo um maior número de treinamentos e ajustes contínuos nos modelos para garantir a confiabilidade dos testes. Além disso, a natureza analógica pode limitar a escalabilidade e a repetibilidade dos processos automatizados, tornando o ambiente de validação mais complexo e menos eficiente. Ainda assim, o processo de treinamento descrito pode ser usado também nesses modelos de IPC, uma vez que os fundamentos para criação dos testes são os mesmos.

Figura 15: IPC utilizado nos testes

Fonte: AutoEntusiatas

4.1 Materiais Utilizados

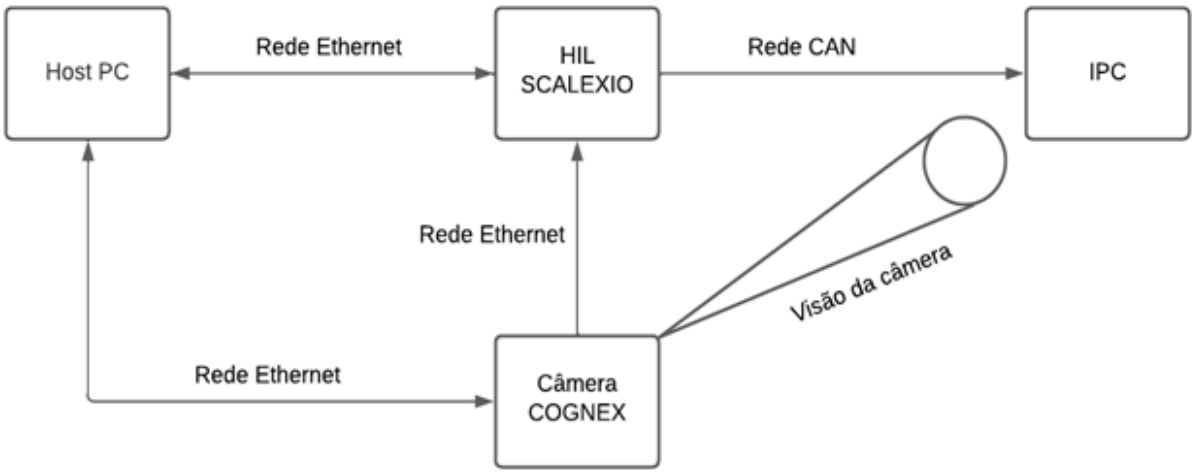
Para a execução do trabalho, foram utilizados os seguintes hardwares e softwares:

- Computador HP Z4.
- dSPACE SCALEXIO com unidade de I/O DS2680.
- Câmera Cognex In-Sight Série 7000.
- Caixa escura com película tipo *blackout*.
- Switch TP-Link Green 5 portas
- MATLAB Simulink, software para simular as ECUs do veículo.
- ConfigurationDesk, software integrar as ECUs simuladas com o HIL.
- AutomationDesk, software para criar as sequências de teste.
- In-Sight Explorer, software para criar os treinamentos de imagem e processá-los.

4.2 Conexão dos hardwares e montagem do setup

O esquemático mostrado na Figura 16 mostra como são organizados os componentes físicos do sistema. O HIL SCALEXIO, por padrão, comunica-se através da rede Ethernet com o computador que atuará como *host* e a câmera, responsável por capturar e processar as imagens. A comunicação com o IPC ocorre pela rede CAN[1].

Figura 16: Montagem dos hardwares

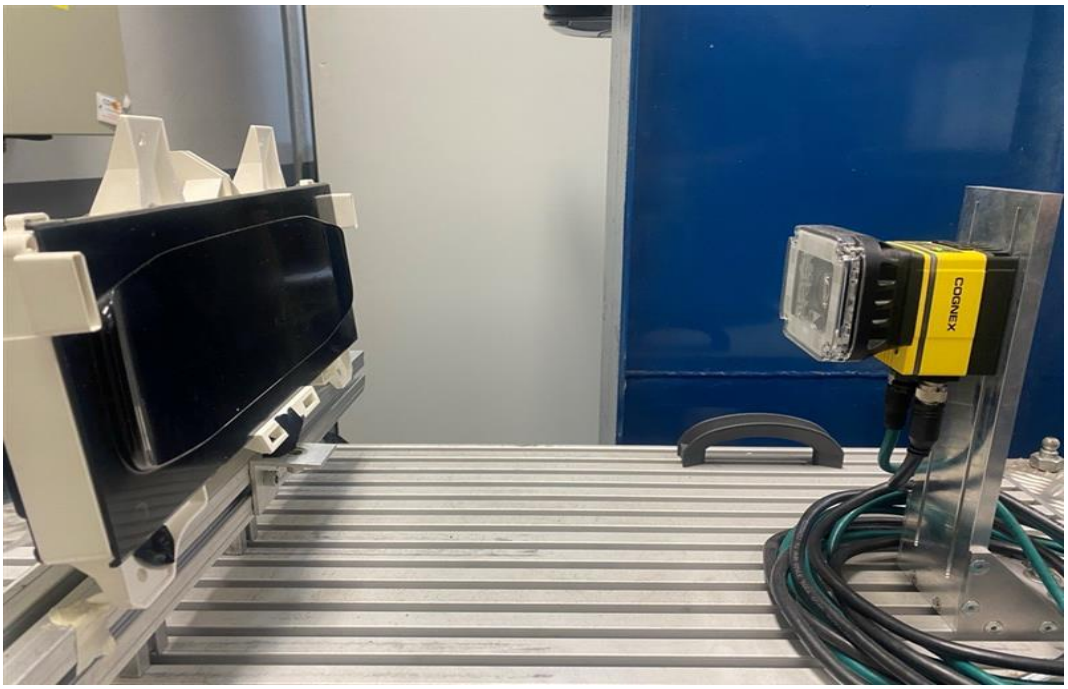


Fonte: Autor

[1] Por ser segredo industrial, a configuração do pinout para este projeto e o conector do IPC não podem ser mostrados neste trabalho.

Sobre uma mesa de suporte, foi projetada uma base de alumínio com 80 cm de comprimento e 50 cm de largura com regulagem por trilhos onde serão alocados a câmera e o IPC. Optou-se por utilizar a câmera de maneira fixa, na lateral direita, evitando perturbações que venham a afetar o bom funcionamento. Já para o IPC, foi elaborado um suporte móvel que se desloca horizontalmente na base, permitindo que, conforme a complexidade da imagem ou caractere a ser avaliado, o cluster possa ser aproximado da câmera. Esse modelo possibilita a implementação de outras ECUs que possam se aproveitar do uso das técnicas de validação empregadas. A montagem final pode ser vista na Figura 17, com o IPC posicionado na lateral esquerda sobre o suporte móvel e a câmera, posicionada a direita.

Figura 17: Montagem da câmera e do IPC na base



Fonte: Autor

Para minimizar os efeitos de iluminação externos da sala, uma caixa escura foi projetada para cobrir os componentes da base. Foi utilizada uma estrutura de alumínio e acrílico, para ser de fácil manuseio e baixo custo. No revestimento do acrílico, foi utilizada uma película *blackout*, garantindo mínima exposição a iluminação externa e maior precisão na avaliação da câmera, conforme a Figura 18.

Figura 18: Caixa projetada para reduzir o efeito da iluminação externa

Fonte: Autor

A Figura 19 mostra a montagem final dos hardwares. Nela, a direita temos o Host PC utlizado no projeto. Posicionada acima da mesa está a caixa preta, que cobre tanto a câmera quanto o IPC e abaixo da mesa está o dSPACE SCALEXIO.

Figura 19: Montagem final

Fonte: Autor

4.3 Criação do modelo no Simulink e integração entre softwares

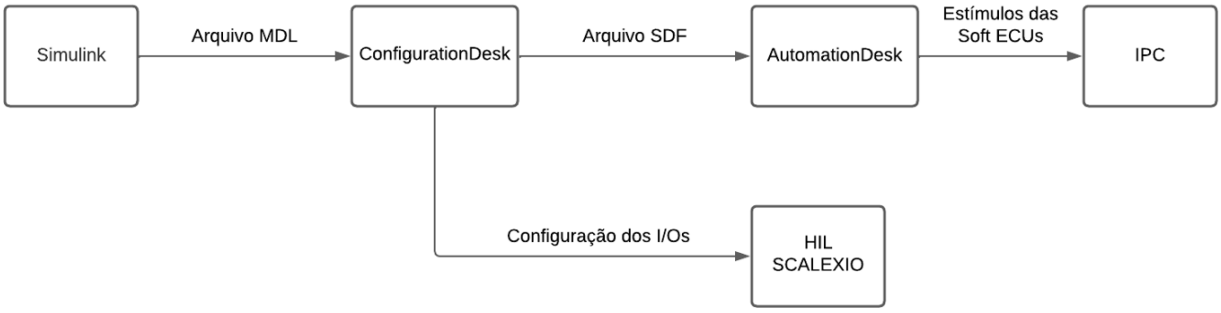
A integração entre as diferentes ferramentas inicia-se com o modelo (MDL) no MATLAB Simulink. Nessa etapa, é utilizado DBC (Database CAN), que contém todos os sinais e mensagens trocados entre as ECUs. O DBC contém também os parâmetros de identificação, estrutura e prioridade de cada mensagem enviada na rede CAN. Foi utilizada a biblioteca RTI CAN (*Real-Time Interface for CAN*), desenvolvida pela dSPACE para o ambiente Simulink, destinada à configuração e simulação de redes CAN em modelos de controle.

A biblioteca RTI CAN possibilita a configuração das interfaces de comunicação, parametrização do barramento CAN e também a definição dos I/Os (*Inputs/Outpus*) de cada ECU simulada. Os I/Os recebem identificadores únicos, alinhados aos respectivos IDs de mensagem especificados no DBC, garantindo consistência entre o modelo funcional e as interfaces físicas emuladas no HIL. Dessa maneira, o ConfigurationDesk realiza o mapeamento direto entre o modelo desenvolvido no Simulink e a infraestrutura de simulação em hardware (HIL), promovendo comunicação sincronizada e determinística[2]. Um exemplo das ligações do software pode ser visto na Figura 10.

A Figura 20 ilustra o processo no qual o arquivo com extensão .sdf (System Description File), produzido pelo ConfigurationDesk, é carregado no AutomationDesk. A partir desse arquivo, o AutomationDesk passa a interpretar e manipular as variáveis e sinais simulados para realizar o acionamento e validação das funções específicas do IPC.

[2] As imagens correspondentes à simulação das SoftECUs e seus I/Os não podem ser apresentadas neste trabalho por estarem protegidas por sigilo industrial, conforme acordado com a empresa responsável.

Figura 20: Sequência de atuação dos softwares



Fonte: Autor

4.4 Reconhecimento de imagem

O treinamento das imagens é realizado por meio do software específico da câmera, que disponibiliza as funções para configuração e ajuste dos parâmetros de detecção e processamento. Essa ferramenta possibilita o processamento de imagens em tempo real e comunicação direta com o hardware, contribuindo para aumentar a eficiência e a consistência nas etapas de calibração e ajuste do sistema de visão.

A interface gráfica do software facilita a configuração e o ajuste dos algoritmos de detecção, reduzindo o tempo necessário para preparação dos testes. Além disso, sua estrutura intuitiva contribui para tornar o processo de treinamento acessível a diferentes níveis de experiência técnica, ampliando a facilidade de integração e manutenção do sistema.

O processo apresentado na Figura 21, mostra como o treinamento do sistema é feito. A validação foi feita utilizando um IPC que já possuía um software validado manualmente e aprovado pela equipe responsável para preparar o sistema, ao invés de ir ao repositório de imagens do projeto e pegar a imagem de cada ícone, mensagem e cor que seria avaliada pela câmera. Além disso, carregar uma imagem em alta definição no software da câmera pode fazer com que o reconhecimento não seja executado de maneira eficaz, por problemas como: diferentes frequências entre a tela e o obturador, resolução do IPC e ocorrência do efeito moiré[3].

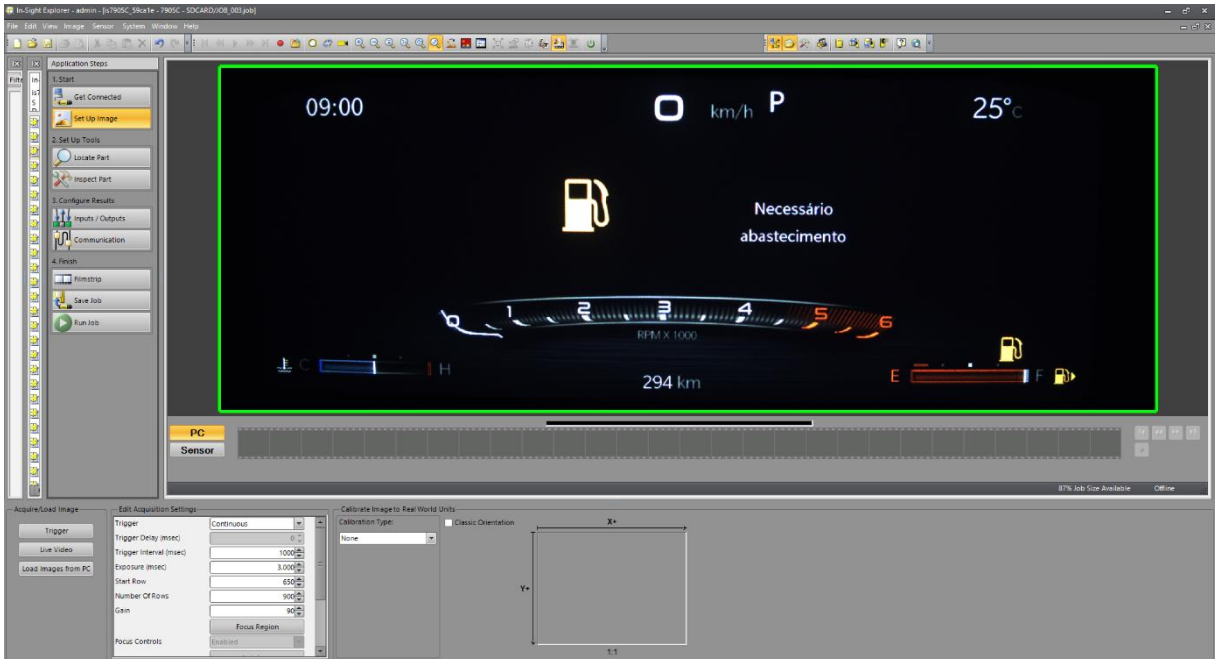
[3] O efeito moiré ocorre quando dois padrões, como retículas de cores diferentes ou linhas finas em uma imagem, se sobrepõem de forma inadequada, criando um padrão visual indesejado de linhas ou ondulações irregulares.

Figura 21: Sequência do treinamento da câmera

Fonte: Autor

Tendo isso em mente, o IPC que serve como base foi montado no suporte e a indicação a ser avaliada pela câmera foi acionada através do AutomationDesk. Em seguida, capturou-se a imagem através do botão de trigger no software e foram feitos os ajustes no foco, tempo de exposição, área da imagem capturada e avaliada pela câmera e intervalo entre as capturas de tela subsequentes. A Figura 22 mostra a indicação de baixo nível de combustível, que foi escolhida por conter todos os elementos que podem ser avaliados manualmente quando um software está sendo validado: *pop-up* contendo imagem e texto, acionamento de *telltale*[4] e mudança de cor em itens que são padrão do IPC durante o funcionamento normal.

Figura 22: Imagem utilizada para o treinamento da câmera



Fonte: Autor

[4] Um *telltale* (ou luz espia) é uma luz ou indicador no painel de instrumentos de um veículo que alerta o motorista para uma condição ou status específico.

O software In-Sight Explorer faz o reconhecimento de padrões, cores, texto e formas em uma imagem. Optou-se por utilizar duas funções que atendem a demanda de validação: PatMax e Color. O primeiro possui a capacidade de identificar padrões de imagem mesmo com variações de ângulo e profundidade. O segundo permite identificar cores de acordo com a necessidade, buscando diferentes intensidades da mesma cor em uma área pré-determinada.

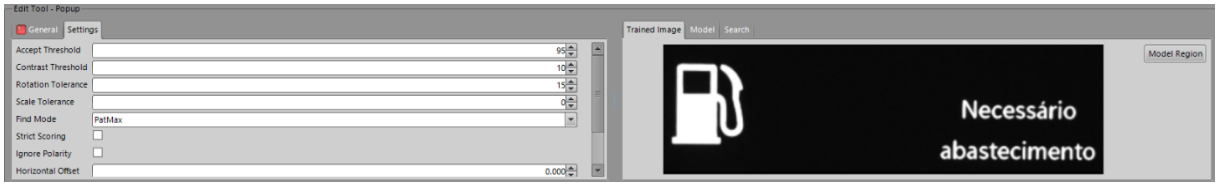
Para identificar a presença do *pop-up* (região central do IPC) contendo a imagem da bomba de combustível e o texto “Necessário abastecimento”, foi utilizado o PatMax. Como o *pop-up* existe no repositório do projeto como sendo uma única figura que contém o texto e o ícone juntos e todas as imagens são previamente validadas por uma equipe dedicada a isso, não é necessário avaliar os caracteres individualmente.

Durante o processo de treinamento de imagem, foi selecionada apenas a região da imagem relevante para o reconhecimento, excluindo elementos irrelevantes do campo de análise. Em seguida, foram realizados diversos testes, nos quais a tela foi capturada repetidas vezes sob as mesmas condições visuais, simulando o comportamento real durante os ciclos de validação. O objetivo desses testes era ajustar a taxa de aceitação, que representa o nível mínimo de similaridade entre a imagem capturada no momento do teste e a imagem de referência previamente treinada. Em termos práticos, trata-se de um parâmetro de sensibilidade do algoritmo de reconhecimento: quanto maior a taxa, mais exigente será a correspondência entre as imagens; valores muito altos podem levar a falsos negativos, ou seja, situações em que o sistema deixa de reconhecer uma imagem válida por pequenas variações visuais.

Durante os experimentos, concluiu-se que taxas de aceitação acima de 97% resultavam em rejeições indevidas de imagens visualmente idênticas, o que comprometia a confiabilidade do teste. Por outro lado, taxas acima de 80% já garantiam detecção adequada em todos os casos analisados. Assim, com o intuito de manter um equilíbrio entre robustez e tolerância a pequenas variações, adotou-se uma taxa de aceitação de 95%, assegurando precisão na validação sem comprometer a confiabilidade do sistema.

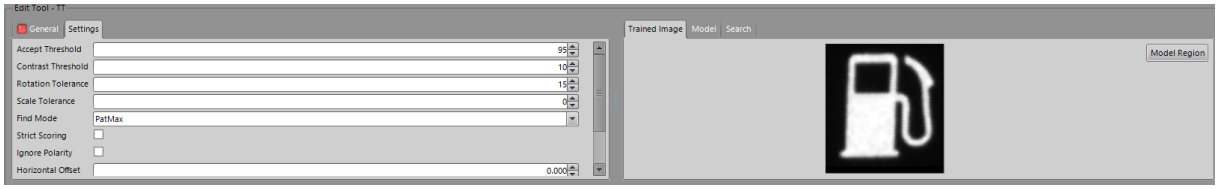
Esse processo foi aplicado para validação do *pop-up* (conteúdo central da imagem na Figura 22) e a *telltale* (localizada no canto direito, acima da barra de nível da mesma figura), conforme mostram as Figuras 23 e 24 abaixo.

Figura 23: Treinamento para identificar o pop-up



Fonte: Autor

Figura 24: Treinamento para identificar o acionamento da telltale



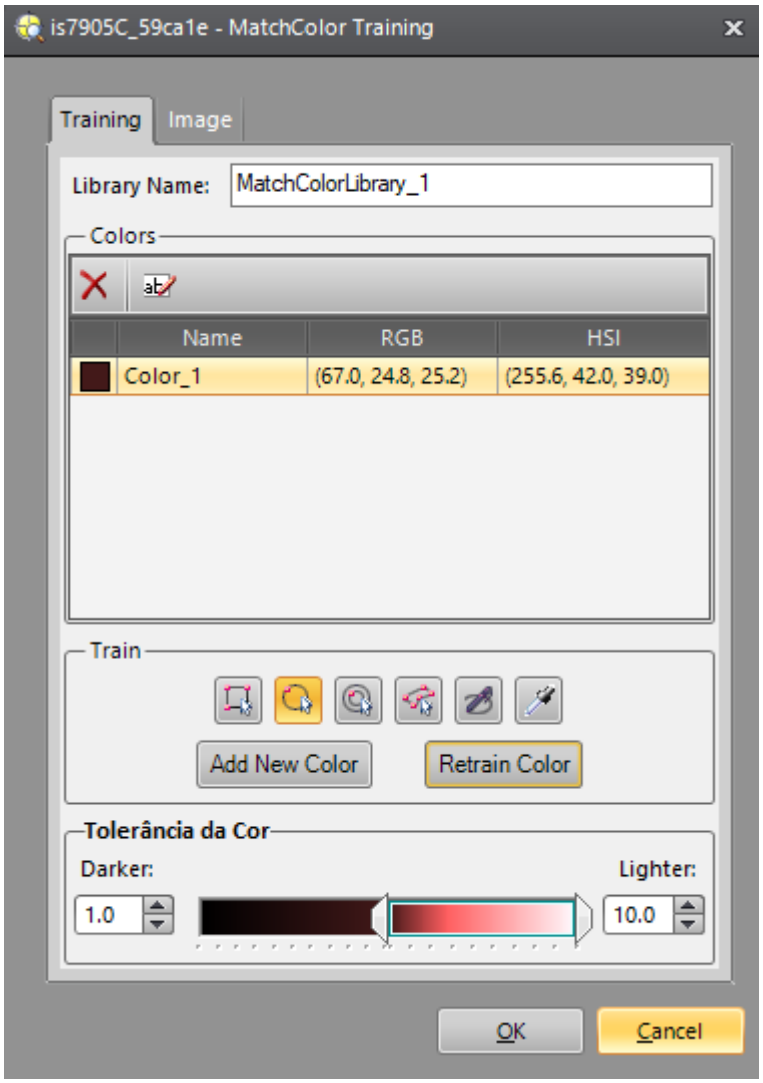
Fonte: Autor

Para realizar a detecção da coloração da barra de nível de combustível, foi utilizada a ferramenta Color, nativa do software *In-Sight Explorer*. Inicialmente, o sinal de nível de combustível foi ajustado para 100%, preenchendo completamente a barra e permitindo a captura da tonalidade correspondente à condição de alerta. Com a área de interesse selecionada, utilizou-se a função de extração de cor, que coletou uma amostra do vermelho escuro exibido. A partir dessa amostra, foi ajustado o tolerância da cor, que representa diferentes tons da cor extraída, desde o tom exato até variações mais claras ou escuras, para incluir variações mais claras de vermelho, considerando que a barra não apresenta coloração totalmente uniforme em toda a sua extensão.

Esse ajuste foi feito diretamente na escala RGB (*Red, Green, Blue*), delimitando os valores mínimo e máximo de cada componente para abranger a variação observada. Embora a ferramenta também apresente os valores equivalentes na escala HSI (*Hue, Saturation, Intensity*), essa representação foi mantida conforme a configuração automática, sem necessidade de intervenção manual. Dessa forma, o reconhecimento foi configurado para responder de forma consistente às diferentes tonalidades da barra quando preenchida em vermelho.

É importante destacar que, como a coloração padrão da barra é azul, e a tonalidade vermelha só aparece em situações de alerta, essa calibração é fundamental para evitar falsos positivos. A Figura 25 mostra a interface da ferramenta Color, com o nome da cor, os valores das escalas RGB e HSI e a tolerância da cor.

Figura 25: Treinamento da cor da barra de combustível

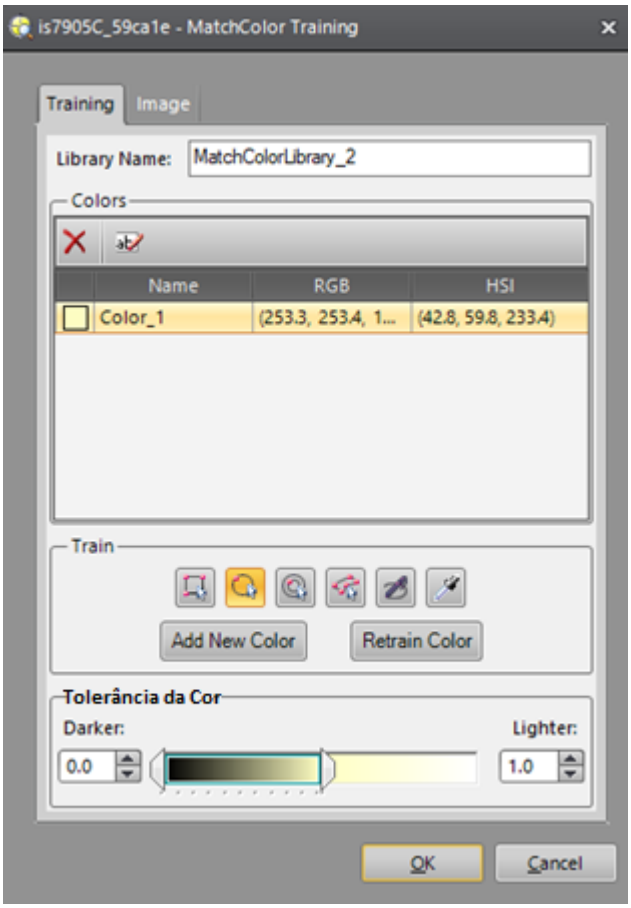


Fonte: Autor

A Figura 26 mostra o mesmo procedimento para o treinamento da cor amarela do ícone da bomba de combustível, localizado ao lado da barra de nível. Em condições normais, esse ícone possui coloração branca, mas passa a ser exibido em amarelo quando a indicação de combustível baixo é acionada, caracterizando uma condição de alerta visual no painel.

O tom de amarelo selecionado foi definido com base na aparência real do ícone durante os testes, buscando uma correspondência visual com a imagem exibida na tela do software da câmera. Como a representação da cor segue a escala RGB, os valores foram ajustados manualmente para abranger variações mais escuras de amarelo, de modo a contemplar possíveis alterações na renderização da imagem. Foram avaliadas diferentes tonalidades até se identificar uma faixa que se adequasse ao comportamento visual observado durante a execução dos testes.

Figura 26: Treinamento da cor do ícone da bomba de combustível



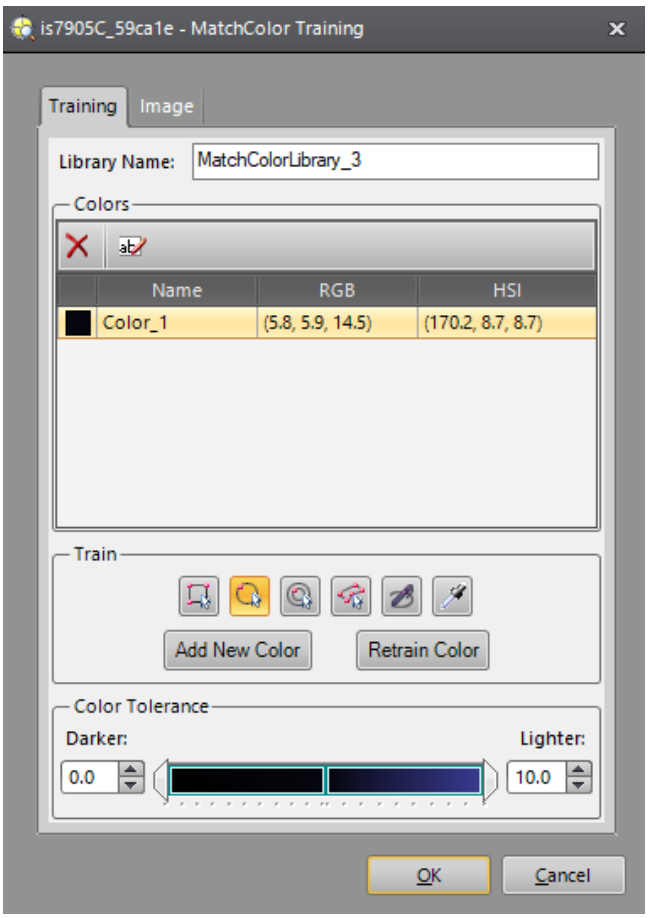
Fonte: Autor

Com o intuito de aumentar a confiabilidade do processo de validação, foi realizado um terceiro treinamento utilizando a ferramenta Color, desta vez voltado à detecção indevida de *telltales* acionadas fora do contexto previsto. O objetivo foi identificar a presença de indicadores visuais não relacionados ao sinal de combustível baixo, garantindo que nenhuma outra *telltale* fosse ativada incorretamente durante a execução do teste.

Para isso, com base na documentação funcional do projeto, foram mapeadas as regiões do IPC que normalmente exibem *telltales*. Em seguida, foi criado um treinamento voltado à identificação da cor de fundo padrão dessas áreas, que consiste em um tom de azul com gradiente para preto, característica típica da interface gráfica do painel em condição normal. A detecção foi configurada para reconhecer o as tonalidades que esse plano de fundo pode assumir, com variações em tons escuros de azul e preto.

A metodologia aplicada considera que, na ausência de *telltales*, a área deve manter esse padrão cromático. Assim, caso uma telltale de cor distinta — como vermelho, verde, laranja ou amarelo — seja ativada indevidamente em uma dessas posições mapeadas, o software será capaz de indicar a anomalia automaticamente. A Figura 27 ilustra a configuração do treinamento, mostrando o intervalo de cores definido para representar o estado neutro do fundo do IPC.

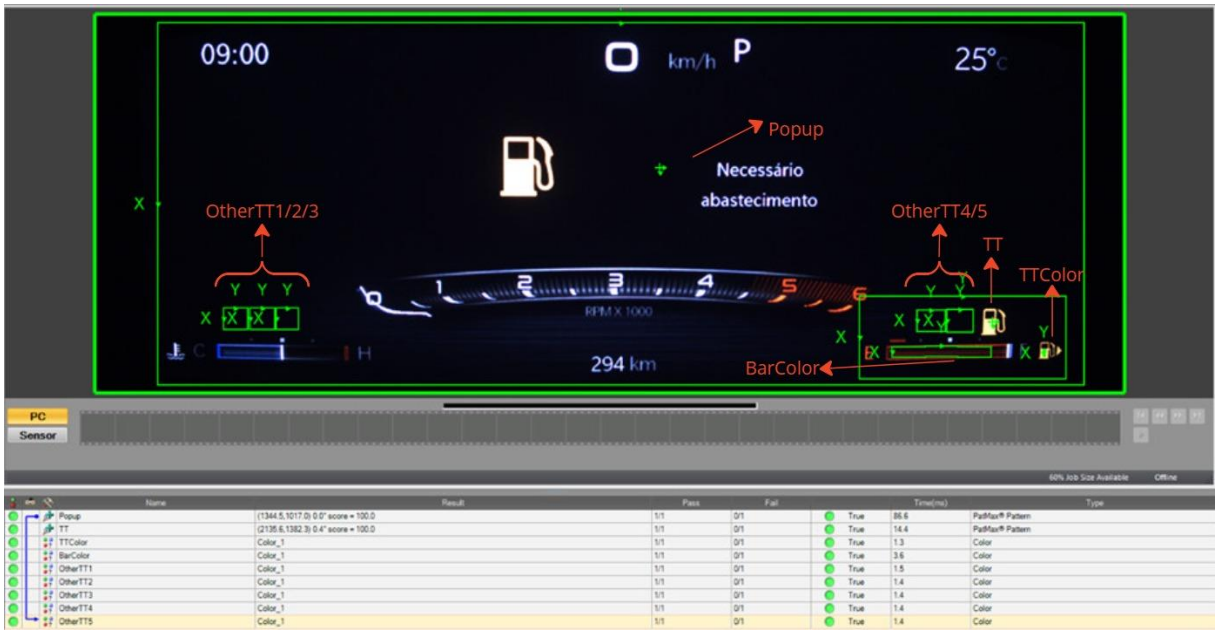
Figura 27: Treinamento de cor para acionamento de outras telltales



Fonte: Autor

Por fim, foi definido que todos os treinamentos seguintes dependem da execução correta da primeira verificação, que corresponde à identificação do pop-up por meio da ferramenta PatMax. Dessa forma, as outras ferramentas de reconhecimento só são acionadas se o pop-up for validado, e o resultado final será considerado positivo apenas se todas identificarem corretamente seus respectivos elementos. Essa decisão foi tomada porque o pop-up é o principal feedback visual recebido pelo motorista. A Figura 28, mostra o mapeamento final das áreas do IPC onde cada um dos treinamentos realizados anteriormente é executado.

Figura 28: Mapeamento final



Fonte: Autor

4.5 Elaboração da sequência de testes

O AutomationDesk permite a criação de rotinas de teste por meio de uma interface gráfica, dispensando a necessidade de programação avançada. Apresenta integração com ferramentas como Simulink, hardwares da dSPACE, incluindo o SCALEXIO, e equipamentos de visão computacional, como câmeras COGNEX. Sua arquitetura modular orientada a componentes facilita a manutenção e a escalabilidade dos sistemas de teste. O software suporta personalizações via linguagem Python e bibliotecas externas, como NumPy e Pandas, para análises de dados. Além disso, gera relatórios com gráficos, tabelas, logs e imagens para documentação e avaliação dos testes.

Dentro do ambiente do AutomationDesk já existem bibliotecas que possuem lógicas básicas de programação (for, and, or, if, while, entre outras) implementadas e que serão utilizadas na elaboração dos testes automáticos. Isso reduz a necessidade de se escrever manualmente cada função durante o processo, permitindo maior eficiência e menos erros.

A Figura 29 apresenta o fluxograma da sequência de testes automatizados. Inicialmente, define-se o número de testes a serem realizados, a variável responsável por acionar a indicação a ser validada e o treinamento da câmera correspondente a essa indicação.

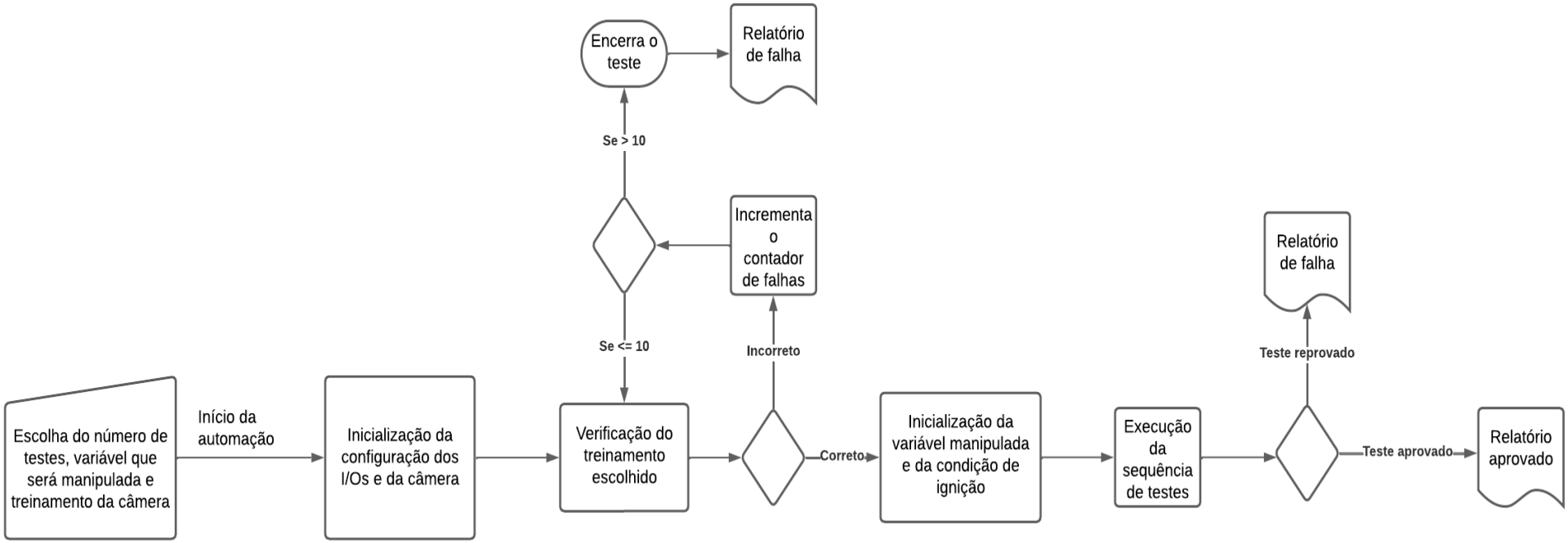
Em seguida, realiza-se o carregamento das configurações das entradas e saídas (I/Os) do SCALEXIO, da câmera e do IPC, além da inicialização da câmera, que realiza capturas de imagem em intervalos de um segundo. A sequência de testes é programada para efetuar a leitura do resultado do processamento da câmera somente após cada alteração na variável que aciona a indicação.

Após a conclusão da inicialização, o sistema verifica o número do treinamento da câmera selecionado, com o objetivo de identificar eventuais erros de digitação ou entradas inválidas, evitando a execução do teste sem o reconhecimento da imagem. Para essa validação, são realizadas até dez tentativas de seleção do treinamento. Caso o número seja reconhecido corretamente pelo software, o teste prossegue; caso contrário, o procedimento é encerrado e um relatório de falha é gerado.

Na etapa seguinte, a variável responsável por acionar a indicação é posicionada em seu estado inicial, com a indicação desativada, e a condição da ignição é configurada para desligada. A sequência de testes é então executada, repetindo-se conforme a quantidade previamente definida e abrangendo as três principais condições de ignição do veículo. Embora existam outras condições de ignição, estas possuem duração mínima e, portanto, não são incluídas na avaliação.

Por fim, é gerado um relatório ao término de cada sequência de testes, contendo informações sobre aprovação ou reprovação, comportamento do sinal manipulado durante o processo e resposta obtida pela câmera. Esse relatório facilita a interpretação dos resultados e a distribuição das informações, sendo detalhado na seção de avaliação dos resultados do teste automático, no item 5.2.

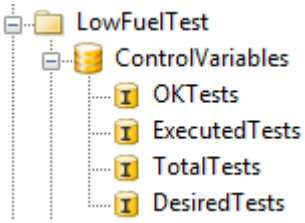
Figura 29: Sequência de testes



Fonte: Autor

Esta organização da sequência de testes foi pensada para servir como uma base genérica e que pode ser usada para a validação de outras indicações e funções do cluster, alterando apenas o sinal que está sendo manipulado e qual o treinamento da câmera é utilizado, passando pelas três principais condições de chave do veículo (ignição desligada, ignição ligada e motor ligado). Foram criadas também quatro variáveis que são responsáveis por: escolher quantos testes deseja-se fazer em cada indicação, quantos foram executados, total efetivamente executado (apenas uma redundância caso haja alguma interrupção no ciclo) e número de aprovações. Cada indicação que aparece no IPC possui estas quatro variáveis.

Figura 30: Variáveis de controle



Fonte: Autor

Cada sequência de testes é concluída, em média, em 10 segundos. Já o tempo necessário para a troca entre diferentes indicações ou conjuntos de testes é de aproximadamente três minutos, considerando o carregamento de novas configurações, seleção do treinamento e reinicialização dos sistemas envolvidos. Esse tempo de execução contribui para uma rotina de testes eficiente, com ritmo adequado à demanda de validações em ambiente automatizado.

5 RESULTADOS

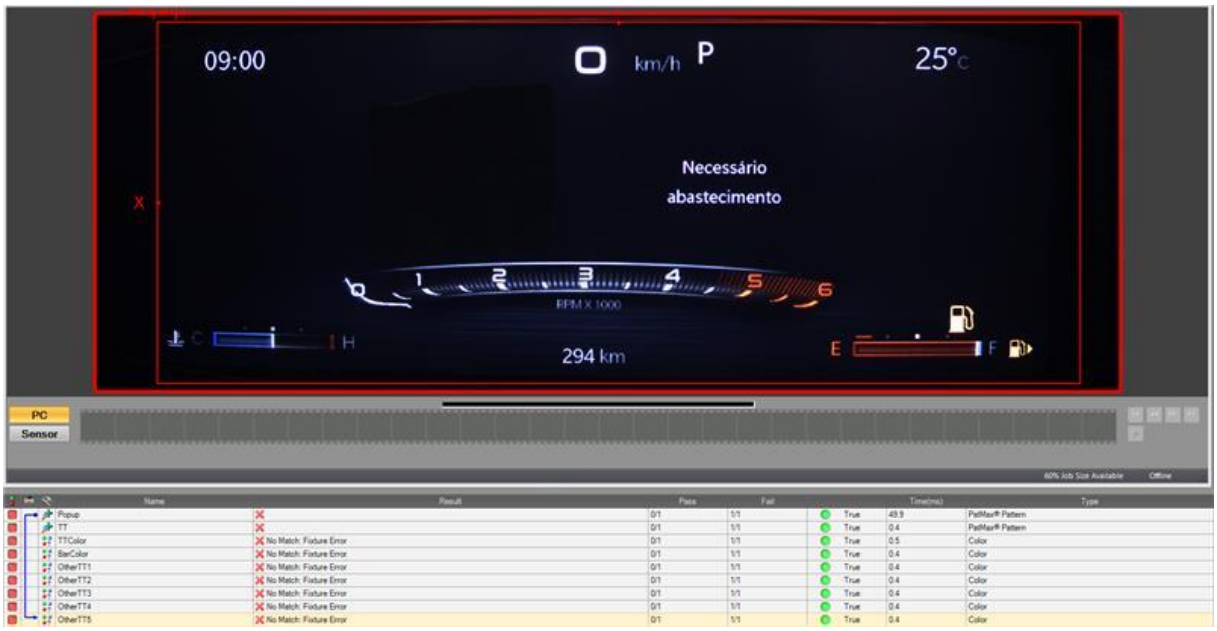
Este capítulo apresenta os resultados obtidos com a aplicação do método proposto, contemplando desde a validação individual dos treinamentos de imagem até a execução de testes automáticos em larga escala. São avaliadas a robustez e a confiabilidade do reconhecimento visual, a eficiência da sequência de testes automatizados e os impactos financeiros e produtivos da solução.

5.1 Validação do treinamento de imagem

Os resultados do treinamento de imagem foram avaliados realizando-se o teste de reconhecimento da indicação através de ciclos de teste. Dentro de cada um desses, foram simuladas algumas condições de falha para poder verificar se o treinamento executado conseguiria identificar o problema e se essa identificação era feita corretamente. Para simular falhas no *pop-up* e na *telltale*, foi utilizada uma fita adesiva preta para cobrir parte da indicação.

A Figura 31 mostra o resultado ao cobrir apenas a imagem da bomba de combustível no *pop-up*. Nela, é possível identificar que o comportamento imposto no final do item 4.4 é respeitado. Ao não conseguir identificar o *pop-up* conforme o treinamento, o software retorna falha para todos os outros componentes do treinamento na seguinte ordem: *telltale* da bomba de combustível acima da barra de nível de combustível (TT), cor da *telltale* a direita da barra de nível de combustível (TTColor), mudança de cor da barra de combustível (BarColor) e regiões onde aparecem outras *telltales* no IPC (OtherTT1, OtherTT2, OtherTT3, OtherTT4 e OtherTT5).

Figura 31: Ausência da imagem da bomba de combustível



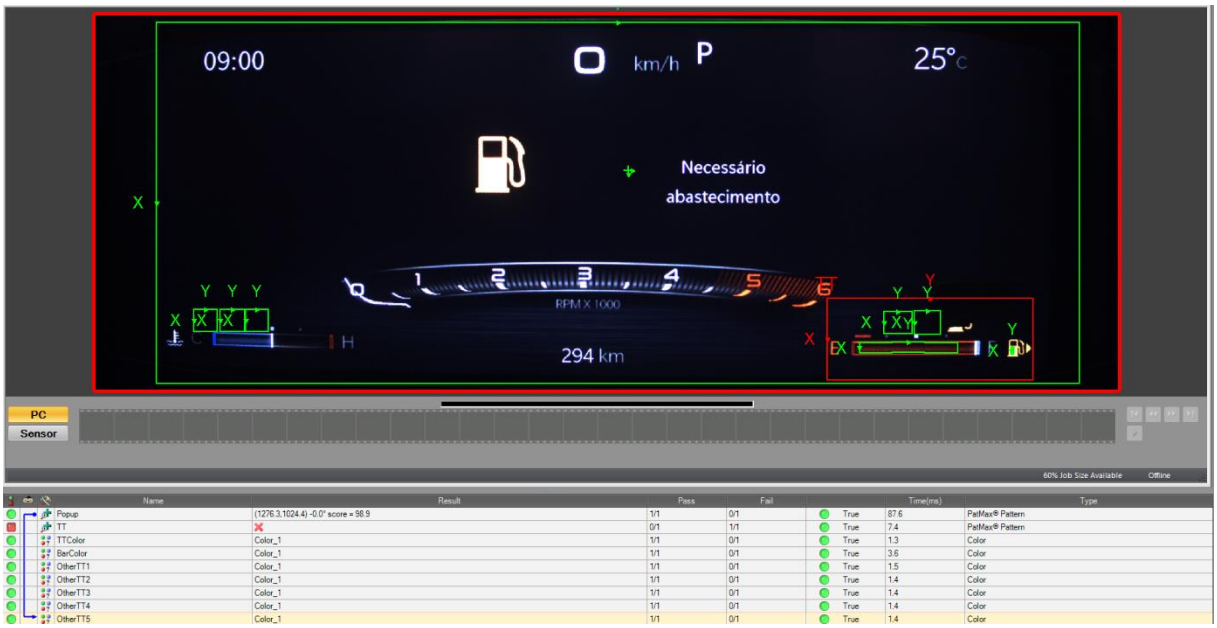
Fonte: Autor

Nas Figuras 32 e 33, respectivamente, é exibido o resultado obtido ao se realizar o mesmo procedimento para a mensagem de texto e a *telltale*, o resultado obtido foi satisfatório, já que em nenhum dos ciclos de teste houve um falso positivo.

Figura 32: Ausência do texto “Necessário Abastecimento”

Fonte: Autor

Figura 33: Ausência da telltale de combustível baixo

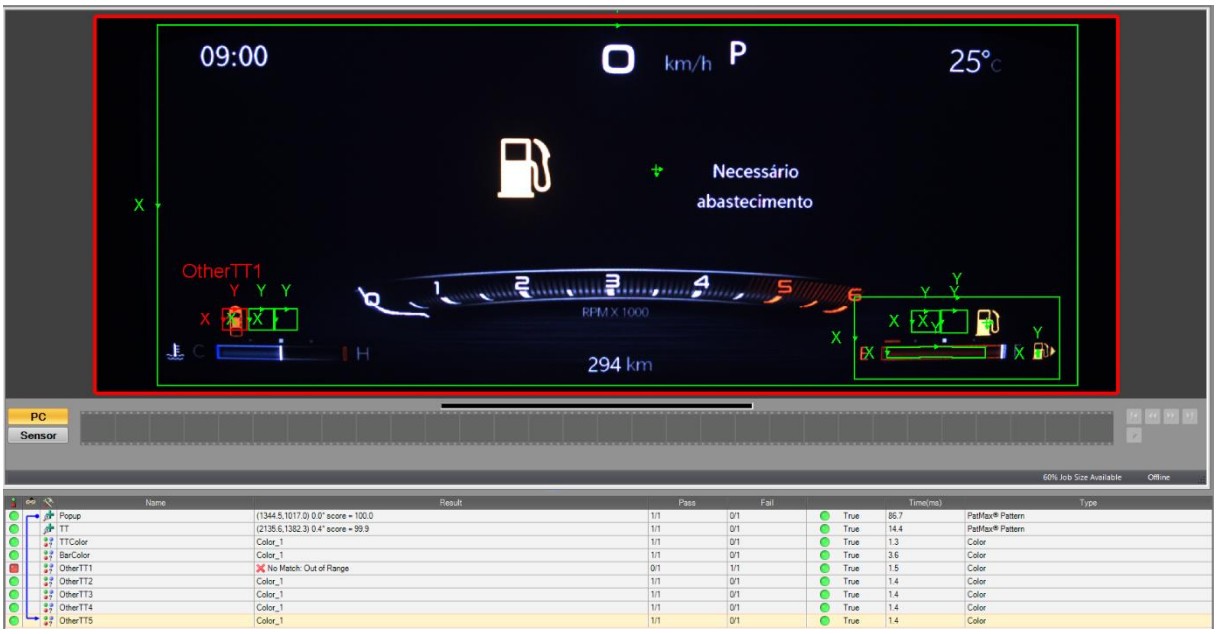


Fonte: Autor

A validação da ferramenta para identificar a presença de outras *telltales* foi feita apenas em uma das regiões, uma vez que o plano de fundo na região onde as outras *telltales* podem aparecer é idêntico ao da região testada, porque houve um acordo de discrição com a empresa responsável. Portanto foi utilizada apenas uma outra *telltale* que pode ser vista de maneira simples por qualquer pessoa que conduza o veículo.

A Figura 34 mostra o resultado ao executar os ciclos de teste, onde é possível observar que a ferramenta se comporta conforme esperado. Em testes realizados para as outras regiões onde as *telltales* podem aparecer, o resultado também foi positivo, mas, como essas regiões apresentam *telltales* de indicações menos comuns ou de funções em desenvolvimento, ficou acordado com a empresa que as imagens referentes seriam mantidas em sigilo.

Figura 34: Presença de telltale em outras regiões do IPC



Fonte: Autor

Na Figura 35 é mostrado o resultado da validação da ferramenta responsável por identificar a mudança da cor do ícone da bomba de combustível e verificar se ele seria capaz de identificar corretamente a diferença entre o branco padrão e o amarelo da indicação de combustível baixo. Como a mudança de branco para amarelo é direta, o teste foi feito ativando o *trigger* da câmera em intervalos de 1 segundo através do software enquanto a indicação não estava ativa. Em nenhum momento ocorreu um falso positivo.

Figura 35: Teste para verificar se há diferenciação entre branco e o amarelo treinado

Fonte: Autor

A ferramenta de diferenciação da cor da barra teve seu comportamento avaliado conforme os testes foram executados e, conforme mostram as Figuras 33 e 34, ela foi capaz de identificar corretamente a mudança de azul para vermelho em todos os testes. Analisando individualmente cada ferramenta e o conjunto como um todo, é possível afirmar que o treinamento da câmera para identificar a indicação possui robustez e capacidade de identificar corretamente cada mudança que ocorre no IPC durante a ativação.

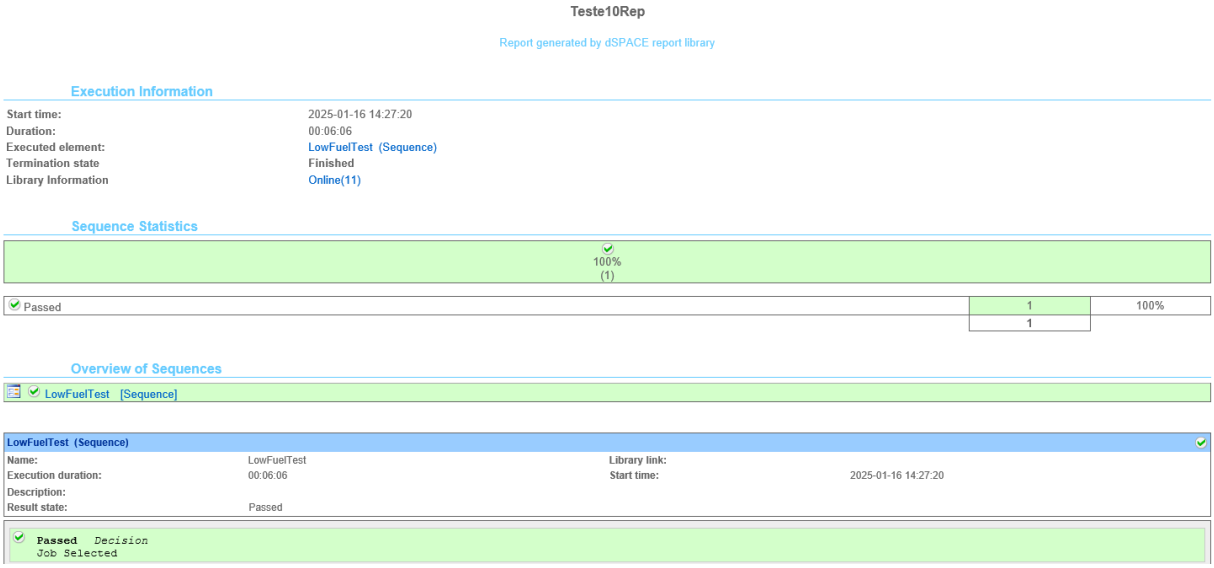
5.2 Validação do teste automático

A sequência elaborada mostrada na Figura 29 foi validada sendo submetida a um teste de estresse, onde foram realizados cerca de 10000 ciclos (cerca de 10 segundos por ciclo) de ativação da indicação de combustível baixo. A execução do teste foi acompanhada de duas formas: através das variáveis mostradas na Figura 30 e visualmente, observando-se o acionamento da indicação no painel de instrumentos dentro do período para se executar cada ciclo.

Ao fim da execução do teste, é gerado um relatório que mostra o comportamento da variável manipulada e a resposta da câmera ao longo da execução do teste. O relatório permite uma visualização gráfica do resultado e facilita a identificação de eventuais falhas, uma vez que ele mostra tanto o comportamento dos sinais quanto a condição de ignição do veículo no momento do acontecimento.

Os resultados da sequência de teste mostrados abaixo foram realizados numa escala menor, para melhor visualização dos gráficos. Na Figura 36, é mostrado o cabeçalho do relatório, mostrando que a sequência executada (*LowFuelTest*) foi executada corretamente e teve um índice de aprovação de 100%. Também é possível ver que a sequência de seleção do treinamento da câmera foi executada com êxito.

Figura 36: Relatório contendo o resultado da seleção do treinamento (*job*) da câmera

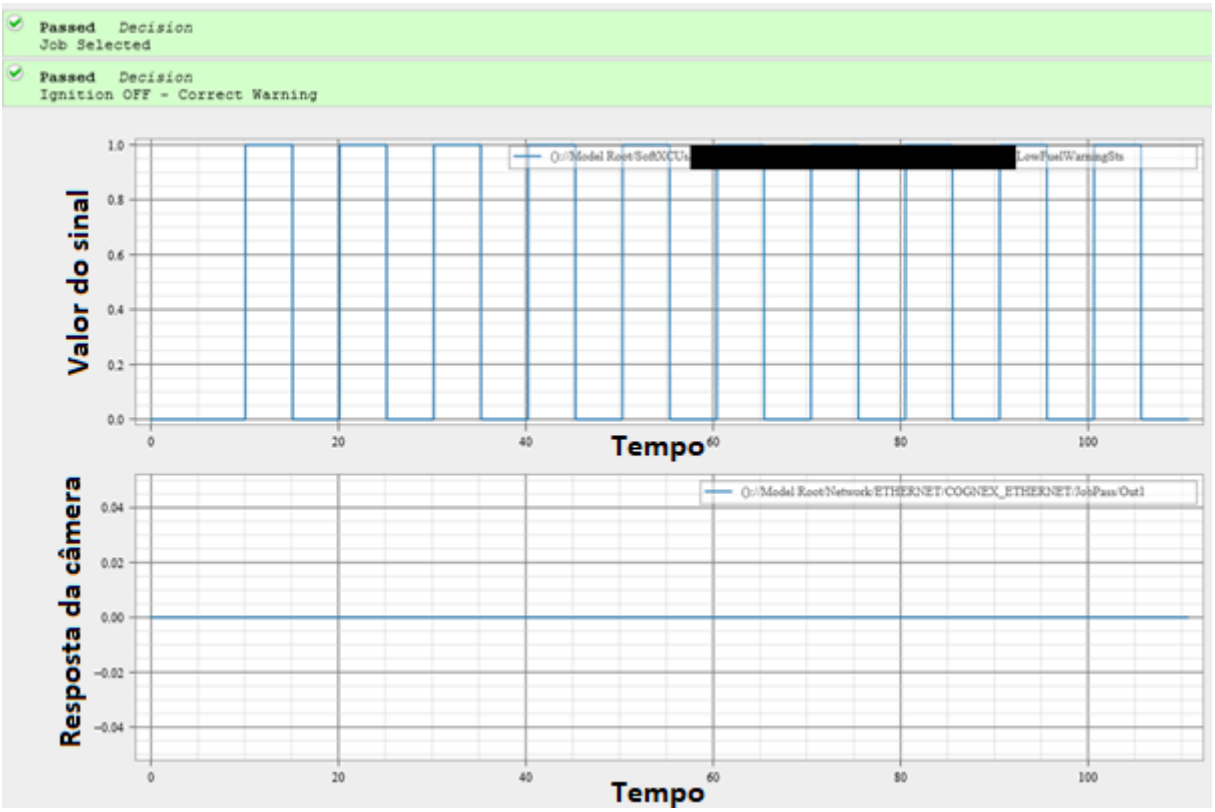


Fonte: Autor

A Figura 37 mostra o resultado do teste para a condição de ignição desligada (*Ignition OFF*). No primeiro gráfico, é possível verificar a variação do valor do sinal no decorrer do tempo (s). No segundo gráfico é exibido o resultado do processamento de imagem obtido pela câmera. Um resultado 0 indica que o treinamento não foi aprovado.

Para essa condição de ignição em específico, o relatório é dado como aprovado porque a indicação de nível baixo de combustível não deve ser exibida com o veículo desligado.

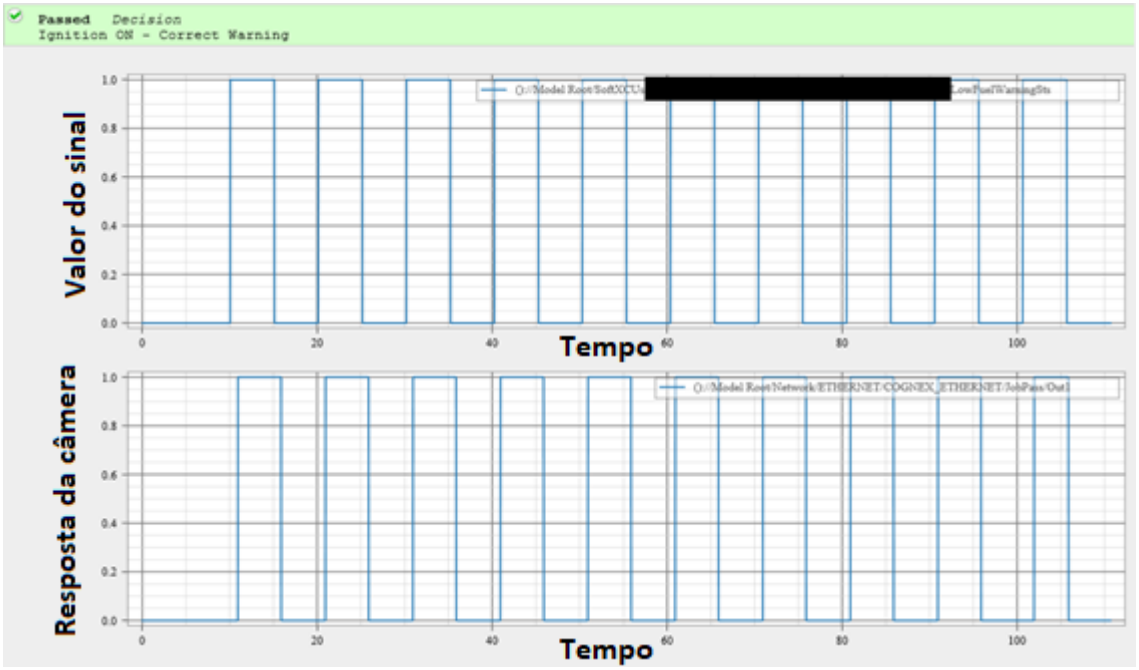
Figura 37: Parte do relatório contendo o resultado para ignição desligada



Fonte: Autor

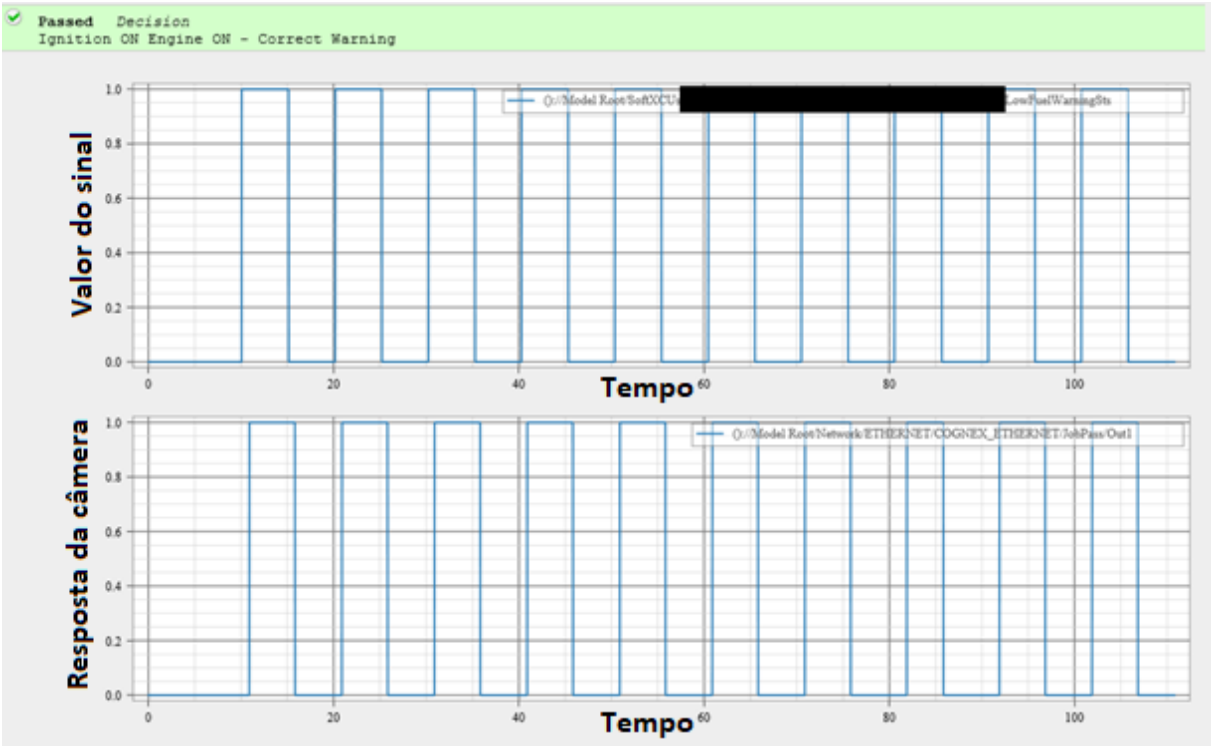
As Figuras 38 e 39 mostram o resultado do teste para as outras duas condições de ignição – ligada e motor ligado – e, nesse caso, é possível observar que durante a ativação da indicação (valor do sinal igual a 1), a câmera também retorna um valor 1, indicando que a indicação de nível baixo de combustível foi identificada corretamente, conforme o treinamento.

Figura 38: Parte do relatório contendo o resultado para ignição ligada



Fonte: Autor

Figura 39: Parte do relatório contendo o resultado para ignição e motor ligados

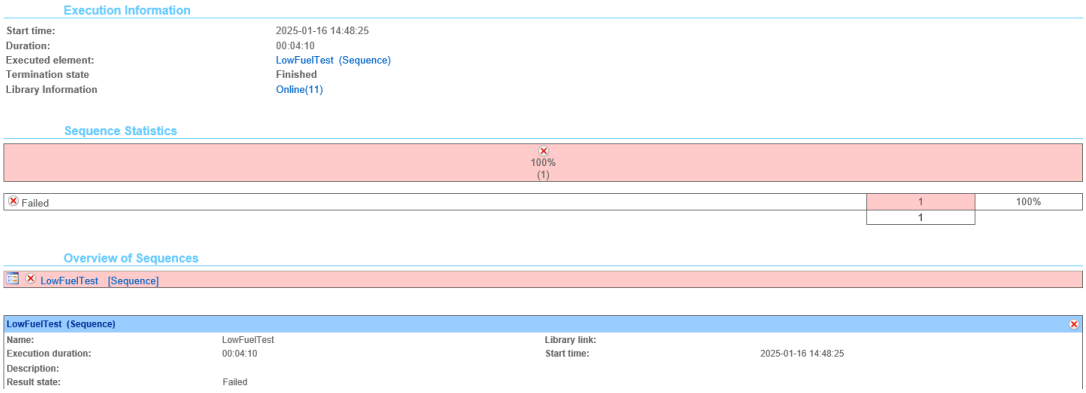


Fonte: Autor

Foi simulada uma condição de falha ao se tampar completamente a câmera, não permitindo a visualização da indicação. Como uma condição imposta na

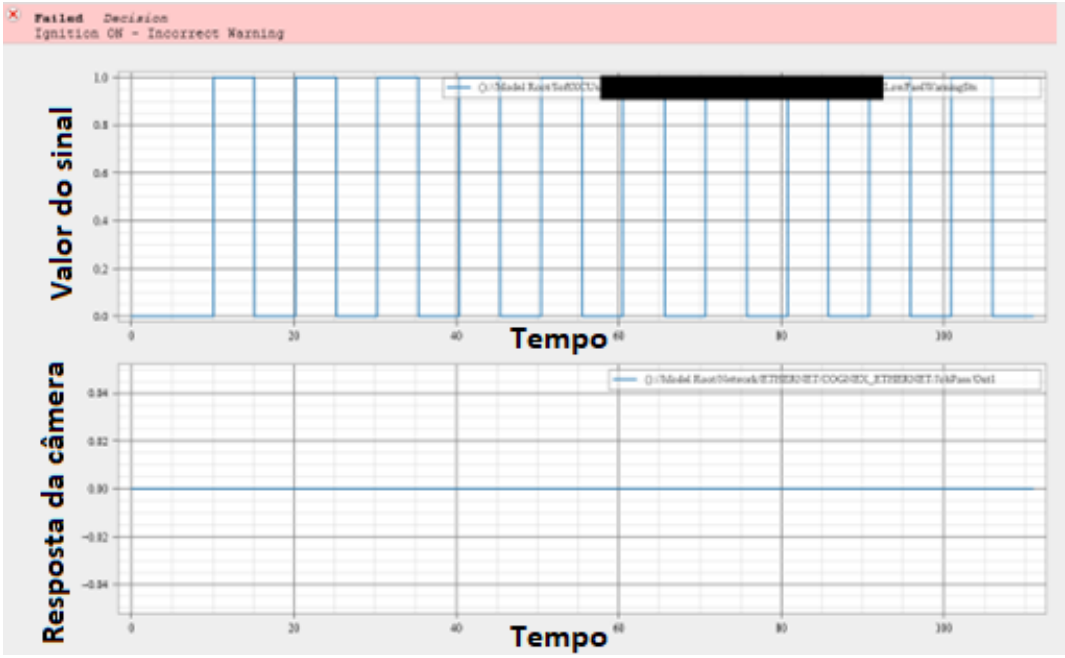
sequência de testes não foi atendida, o teste é finalizado e o relatório é exibido da seguinte forma:

Figura 40: Cabeçalho do relatório quando há falha



Fonte: Autor

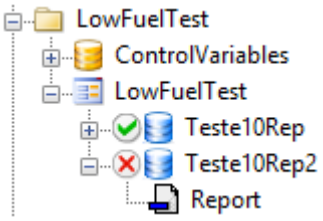
Figura 41: Indicação de qual etapa do teste falhou



Fonte: Autor

Na Figura 42, são exibidas as duas condições: aprovação e falha. O item Teste10Rep, que obteve um resultado positivo é mostrado com uma marcação verde ao lado. Já o item Teste10Rep2 em que houve a simulação da falha é exibido com uma marcação vermelha ao lado.

Figura 42: Exibição dos resultados no AutomationDesk



Fonte: Autor

5.3 Impacto financeiro

A implementação do método proposto requer, inicialmente, a estruturação completa do ambiente de testes baseado em Hardware-in-the-Loop (HIL). Essa etapa é conduzida por um engenheiro responsável pela montagem e configuração do setup, o qual envolve tanto a preparação dos hardwares quanto a integração dos elementos de software necessários para a execução dos testes automatizados.

A configuração do ambiente tem início com a definição do modelo do sistema a ser validado e a inserção dos arquivos de descrição da rede (DBC), que são fundamentais para a interpretação correta dos sinais CAN utilizados durante os testes. Em seguida, realiza-se a montagem física do HIL, incluindo o rack, os módulos de entrada e saída, as interfaces de comunicação e a conexão com a câmera industrial responsável pela captura das imagens do IPC.

Além da parte física, há a necessidade de preparar os sistemas que simulam o comportamento das ECUs em ambiente virtual (soft ECUs), configurando parâmetros, mensagens e reações conforme o escopo do teste. Posteriormente, é realizada a integração entre os softwares AutomationDesk, ControlDesk, ConfigurationDesk, In-Sight Explorer e MATLAB, permitindo o controle dos sinais, a captura de imagens e o processamento dos resultados de forma sincronizada.

Na etapa final, realiza-se o treinamento das imagens por meio das ferramentas PatMax e Color, com base nas indicações visuais do cluster. Essa preparação visual é essencial para que a câmera possa identificar corretamente os elementos gráficos apresentados no painel durante os testes.

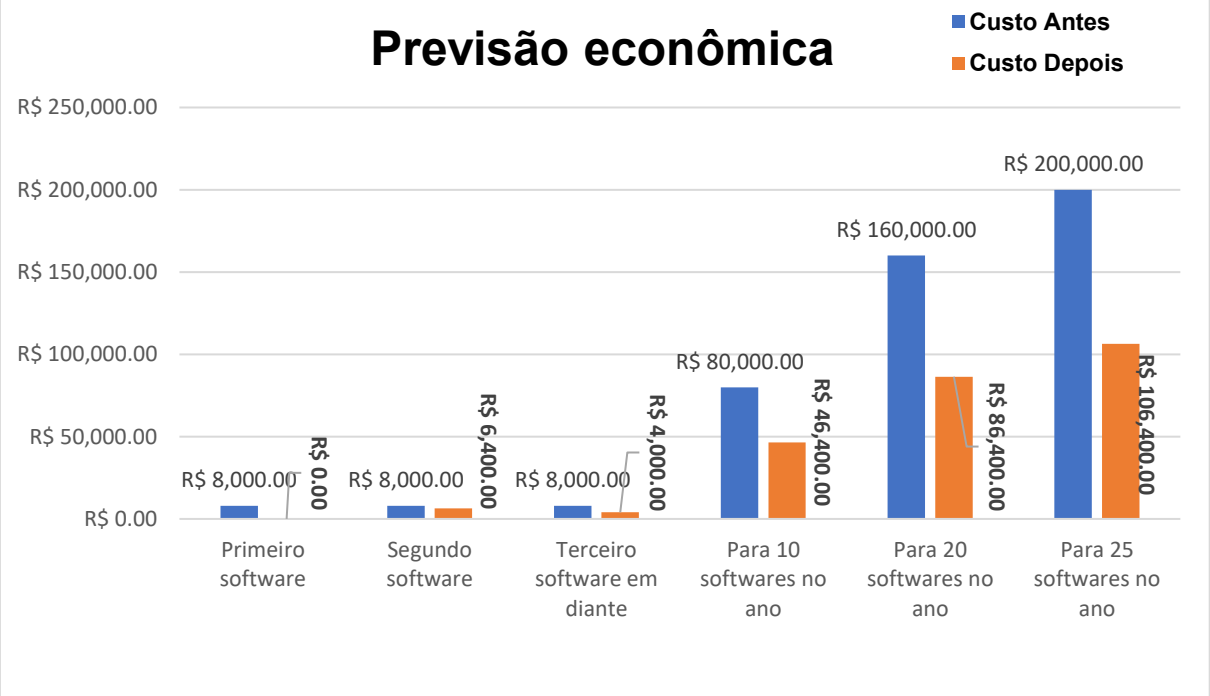
O tempo total necessário para a montagem, configuração e validação do setup é de aproximadamente um mês, considerando também os ajustes finos, testes preliminares e eventuais incompatibilidades entre os sistemas. Embora esse período represente um investimento inicial significativo em tempo e recursos, os ganhos

observados nos ciclos de validação subsequentes indicam um retorno técnico e operacional consistente.

Para estimar o custo associado à implementação do ambiente, foram utilizados valores médios de mercado com base em cotações disponíveis publicamente. A plataforma HIL SCALEXIO foi estimada em aproximadamente R$ 630.000, a câmera Cognex da série 7000 em torno de R$ 26.250,00 e o conjunto de licenças de software (incluindo AutomationDesk, ControlDesk, ConfigurationDesk e visão computacional) em cerca de R$ 183.750,00. Além disso, estima-se um custo de mão de obra para o engenheiro responsável pela montagem e configuração do setup equivalente a 220 horas de trabalho, a um custo médio de R$ 100,00 por hora, totalizando R$ 22.000,00. Assim, o investimento inicial total aproximado é de R$ 862.000,00. Ressalta-se que esses valores são apenas projeções aproximadas, utilizadas com finalidade ilustrativa para a análise econômica, e podem variar conforme o escopo do projeto, a configuração dos equipamentos e os contratos com fornecedores.

Considerando que o custo médio de um ciclo de validação sem automação é de aproximadamente R$ 8.000,00, e que, com a automação, esse valor é reduzido para R$ 4.000,00, projeta-se uma economia de R$ 4.000,00 por ciclo. Com base nessa diferença, o *payback* estimado do investimento ocorre após cerca de 216 ciclos de validação. Em contextos onde há um elevado número de versões de software, ou em projetos com múltiplas plataformas que compartilham a mesma arquitetura, como ocorre nos IPCs de 10 polegadas da empresa, esse retorno tende a se concretizar em cerca de 5 anos.

Figura 43: Previsão econômica com utilização da automação



Fonte: Autor

Essa análise demonstra que, além dos ganhos operacionais, a adoção do método automatizado pode representar uma alternativa economicamente viável para projetos de validação que demandam repetibilidade, rastreabilidade e maior eficiência na execução dos testes.

5.4 Impacto na produtividade

A aplicação dos métodos descritos neste trabalho, em um cenário de validação para uma das releases de software da ECU IPC, mostrou que o procedimento criado possui impactos positivos durante o processo de validação. Além de reduzir o tempo necessário para validação, foi possível realizar o *stress test* em todas as funções que se encaixavam no padrão estabelecido no trabalho.

O processo na empresa demanda 2 semanas de validação, ou seja, 10 dias de trabalho (80 horas). Com a utilização do ambiente criado, esse tempo pode ser dividido de maneira mais eficiente: as indicações similares a de baixo nível de combustível foram validadas de maneira automática utilizando-se o HIL, enquanto partes que demandam maior complexidade de preparação, como testes de

velocímetro, tacômetro, cálculo de consumo, entre outros, foram validados pela pessoa responsável pelo projeto.

Essa divisão no trabalho permitiu que o tempo de validação final fosse de 8 dias de trabalho (64 horas) para o primeiro *smoke test*, representando uma economia de tempo de 20%. O cálculo inclui o tempo necessário para o treinamento das imagens e troca das variáveis manipuladas no AutomationDesk. Para softwares subsequentes, o volume de alterações é pouco e consiste majoritariamente em *bug fixes*. Como as imagens já haviam sido treinadas anteriormente, as alterações que são feitas nos treinamentos serão aquelas referentes apenas as correções de falhas e, caso hajam, implementação de novas funções. Foi verificado que o tempo de validação para IPCs de 10 polegadas em softwares seguintes é de 5 dias (40 horas).

Essa abordagem atende primariamente a aplicação da proposta do trabalho, mas existem outros benefícios financeiros e de produtividade que dificilmente podem ser mensurados financeiramente, destacando-se o uso de mão de obra para validar outros projetos durante a execução dos testes automáticos, o que reduziria também o tempo de validação para IPCs que não são o foco da automação e a possibilidade de rodar testes 24 horas por dia, 7 dias por semana.

Além disso, o método desenvolvido apresenta potencial de reaplicação em outros IPCs com tela de 10 polegadas que compartilham a mesma arquitetura eletrônica. Nesses casos, a reutilização da infraestrutura de testes exige apenas a criação de novos treinamentos de imagem, sem a necessidade de reconfigurar o sistema como um todo. Essa característica amplia o alcance do sistema automatizado, permitindo sua utilização em múltiplos projetos com esforço adicional mínimo. O que aumentaria a quantidades de softwares que podem ser avaliados no período de 1 ano.

6 CONCLUSÃO

Este trabalho descreveu o projeto, construção e testes de um ambiente para validação do *Instrument Panel Cluster* (IPC) utilizando *Hardware-in-the-Loop* (HIL) associado ao reconhecimento de imagem. Os resultados demonstraram que a integração das ferramentas permitiu automatizar os testes de maneira confiável e eficiente, garantindo um aumento de produtividade e ganhos financeiros para a empresa.

Um dos aspectos mais relevantes foi o aumento da confiabilidade nos resultados obtidos durante a validação. O uso do dSPACE SCALEXIO permitiu uma integração precisa entre simulações e hardware real, enquanto a câmera Cognex In-Sight Série 7000 viabilizou a verificação detalhada de parâmetros visuais, como a ativação correta de *telltales*, cores do painel e exibição de *pop-ups*. Essa abordagem garantiu que os resultados dos testes refletissem com precisão o comportamento do IPC em situações reais de uso.

A possibilidade de realizar *stress tests* também foi um diferencial importante do projeto. Esses testes simularam cenários extremos, como ciclos contínuos de ativação de alertas e mudanças rápidas de mensagens, além de variações no estado de ignição e em condições críticas de operação. A automação permitiu reproduzir essas situações de forma controlada, facilitando a identificação de limitações e aprimorando a robustez do sistema.

A automação também reduziu significativamente o tempo necessário para condução dos testes. Processos que antes demandavam longos períodos de validação manual podem ser executados automaticamente, sem comprometer a qualidade dos resultados e a possibilidade de operação contínua — 24 horas por dia, 7 dias por semana — o que amplia a capacidade de validação da equipe técnica. Isso também permitiu a realocação de profissionais para outras atividades de maior valor agregado, como o desenvolvimento de novos projetos e a melhoria de processos.

Do ponto de vista econômico, a automação reduziu o custo por ciclo de validação de R$ 8.000,00 para R$ 6.400,00 na primeira *release* de software e, a partir das versões seguintes, para R$ 4.000,00, já com os treinamentos de imagem previamente realizados. Considerando essa redução e um investimento inicial estimado em R$ 862.000,00, o retorno financeiro é projetado para ocorrer após cerca de 216 ciclos de validação. A estrutura desenvolvida também pode ser reaproveitada em projetos que utilizem IPCs com arquitetura digital semelhante, exigindo apenas a

criação de novos treinamentos de imagem. Para os modelos analógicos, embora exista maior variação visual entre os painéis e uma necessidade maior de ajustes na fase de preparação, a metodologia continua aplicável, mantendo os princípios de reconhecimento e automação.

Dessa forma, a solução proposta mostrou-se eficiente, replicável e alinhada às necessidades atuais da validação de sistemas embarcados. Sua aplicação contribui para padronizar os testes, otimizar recursos e preparar a indústria para lidar com projetos cada vez mais complexos e exigentes.

6.1 Sugestões para trabalhos futuros

Para trabalhos futuros, recomenda-se a aplicação dos hardware*s* e software*s* utilizados na validação de outras centrais automotivas, especialmente unidades de rádio e sistemas de infoentretenimento. A extensão da metodologia para sistemas mais complexos, como assistência avançada ao motorista (ADAS), alerta de ponto cego e sistemas de direção autônoma, é uma possibilidade promissora, dado que esses sistemas exigem testes rigorosos para garantir segurança e desempenho adequado.

Além disso, recomenda-se aprofundar ainda mais a exploração das ferramentas disponíveis nos software*s* utilizados, garantindo uma validação ainda mais robusta. Explorar o uso de novas técnicas de aprendizado de máquina para aprimorar a precisão do reconhecimento de imagem também pode agregar valor ao processo.

Por fim, sugere-se a investigação de novas tecnologias de hardware e software que possam ampliar o leque de aplicações do método para outras indústrias além da automotiva, como a indústria aeroespacial e sistemas industriais de monitoramento.

REFERÊNCIAS

BOSCH, Robert. CAN Specification 2.0. Bosch GmbH, 1991.

BOYCE, M. C. Automotive Electronics Handbook. 3. ed. New York: McGraw-Hill, 2014.

COGNEX. In-Sight Explorer: Reference Guide. 2025b. Disponível em: https://www.cognex.com/pt-br/resources. Acesso em: 8 jan. 2025.

DSPACE Learning Center. Disponível em: https://www.dspace.com/en/pub/home/learning-center.cfm. Acesso em: 6 jan. 2025.

GÓRSKI, P.; JANISZEWSKI, M. Automotive Electronic Systems Design. 2. ed. New York: Springer, 2022.

GONZALEZ, Rafael C.; WOODS, Richard E. Digital Image Processing. 4. ed. Upper Saddle River: Pearson, 2018.

GOSSLING, H.; GOLDSCHMIDT, A.; KUPFER, S. Handbook of Automotive Electronics and Electrical Systems. 2. ed. Berlin: Springer, 2019.

HILLEBRAND, Jan; KROGER, Stefan; KOCH, Tobias. Simulation-based testing of embedded automotive systems: Trends and challenges. *Journal of Automotive Engineering*, v. 44, n. 3, p. 45-58, 2020.

MATHWORKS. Simulink Documentation. 2025. Disponível em: https://www.mathworks.com/help/simulink/. Acesso em: 6 jan. 2025.

ØSTERGAARD, J.; WU, Q.; NIELSEN, A. H. Real Time Hardware-in-the-Loop Testing for Power Electronics Controllers. *Asia Pacific Power and Energy Engineering Conference (APPEEC)*, 2012. Disponível em: https://doi.org/10.1109/APPEEC.2012.6307219. Acesso em: 15 jun. 2025.

RIBEIRO, Renan Airton B. Multiplexação de pacotes CAN em quadros Ethernet. 2018. Disponível em: https://www.formiga.ifmg.edu.br/documents/2017/PublicacoesTCCsBiblioteca/CienciadaComputacao/monografia--Renan-Airton-B-Ribeiro---Ciencia-da-Computacao.pdf. Acesso em: 6 jan. 2025.

RUSSELL, Stuart et al. Artificial Intelligence: A Modern Approach. 3. ed. Upper Saddle River: Prentice Hall, 2016.

SANTOS, André L. dos et al. Avaliação da interface gráfica de sistemas automotivos com inspeção automatizada. *Revista de Engenharia de Automação*, v. 12, n. 1, 2020.

SIVARAMAN, Sai; TRIVEDI, Mohan M. Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection. *IEEE Transactions on Intelligent Transportation Systems*, v. 14, n. 4, p. 1774-1795, 2013.

SMITH, Grant Maloy. O que é barramento CAN (Controller Area Network) e como ele se compara a outras redes de barramento de veículos. Disponível em: https://dewesoft.com/pt/blog/que-e-barramento-can. Acesso em: 6 jan. 2025.

SOMMER, Eike et al. Real-time Hardware-in-the-Loop simulation for the automotive industry: Benefits and practical applications. *Automotive Technology Journal*, v. 33, p. 109-116, 2016.

SZELISKI, Richard. Computer Vision: Algorithms and Applications. 1. ed. London: Springer, 2011.

TANENBAUM, Andrew S. Operating Systems: Design and Implementation. 3. ed. Rio de Janeiro: Elsevier, 2016.

TANG, Qiyuan et al. Vision-based validation of automotive HMIs. *International Journal of Automotive Technology*, v. 22, n. 2, p. 305-318, 2021.

ZHANG, Wei; WANG, Xian. The role of Hardware-in-the-Loop in automotive control systems testing. *International Journal of Automotive Technology*, v. 28, n. 1, p. 65-72, 2017.

ZHANG, Yong; WANG, Wei. Hardware-in-the-loop simulation for automotive control systems. 2017.

ZHANG, Wei et al. Improving pattern matching robustness in vision systems. *Journal of Imaging Science*, v. 10, n. 4, p. 50-60, 2019.

APÊNDICES

Apêndice A – Sequência de testes

A primeira etapa é configurar o fornecimento de energia para o *cluster* e permitir que o AutomationDesk consiga manipular corretamente os diferentes estados de chave do veículo, influenciando assim o IPC. Através das variáveis ***ActiveSupply*** (com o valor 0 indicando a ausência de tensão e 1 indicando a presença) e ***BatteryVoltage*** (variando de 0 a 18), a simulação recebe o valor de tensão que chega da bateria da fonte analógica e informa as demais ECUs simuladas, permitindo que todas atuem de maneira síncrona. A variável ***TE30IPC*** é responsável por permitir ou não que a tensão fornecida pela fonte analógica chegue até o hardware físico que será testado, controlando um *switch*, onde 0 indica que o *switch* está aberto e 1 fechado. ***Control_Value2*** faz a função da chave do veículo, permitindo que a simulação identifique corretamente em qual estado de ignição o veículo está, fazendo com o que o IPC ligue ou desligue conforme ocorreria em uma situação real. Aqui, a variável pode assumir valores de 0 a 8. Para os testes utilizados, a variável assume os valores 2 (ignição desligada), 4 (ignição ligada) e 8 (ignição e motor ligados). ***Conversion_Out1*** é responsável por fazer a conversão do sinal analógico para o sinal digital.

Figura 44: Variáveis de controle de tensão mapeadas

Mapping Viewer

Project [redacted].Mapping

Drag a column header here to group by that column

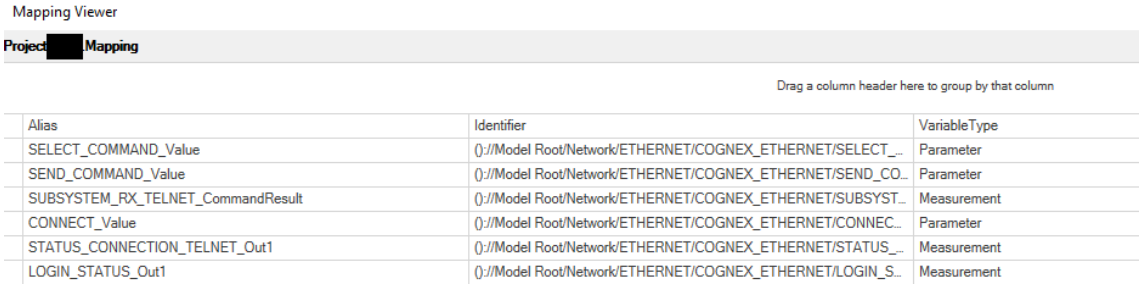
Alias	Identifier	VariableType
ActiveSupply	():/Model Root/IOUserInterface/Simulator/Power Supply/Shutdown_Rel...	Parameter
BaterryVoltage	():/Model Root/IOUserInterface/Simulator/Power Supply/BatteryVoltage...	Parameter
TE30IPC	():/Model Root/IOUserInterface/Simulator/Power Supply/CTRL_TE30_I...	Parameter
Control_Value2	():/Model Root/IOUserInterface/Simulator/Power Supply/KEY [-]/Control...	Parameter
Conversion_Out1	():/Model Root/IOUserInterface/Simulator/Power Supply/Vbat1Measure...	Measurement

Fonte: Autor

A configuração das variáveis que atuam na câmera COGNEX In-Sight 7905C também são mapeadas nesta etapa. Primeiro temos a variável ***SELECT_COMMAND_Value***, responsável por receber um input de qual comando se tem a intenção de executar. ***SEND_COMMAND_Value*** envia o comando para a câmera quando assume o valor 1. ***SUBSYSTEM_RX_TELNET_CommandResult*** retorna se o input da primeira variável é válido (1) ou inválido (-5). Em seguida, temos

o comando de conexão da câmera, para que ela fique online na rede, gerido pela variável ***CONNECT_Value***, que assume valor 0 quando se tem a intenção de desconectar e 1 para conectar. ***STATUS_CONNECTION_TELNET_Out*** é responsável por dar o *feedback* para o sistema se a câmera está online (1) ou não (0). ***LOGIN_STATUS_Out1*** também é uma variável de retorno, mas, diferente da anterior, esta é responsável por informar o restante da simulação se o *login* na câmera foi executado da maneira correta (1) ou não (0).

Figura 45: Mapeamento das variáveis da câmera


Mapping Viewer

Project[redacted].Mapping

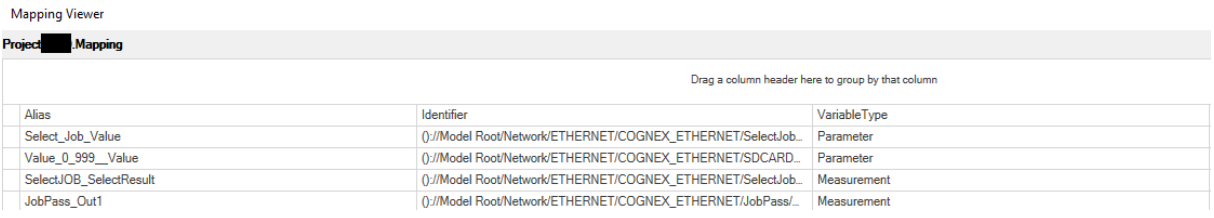
Drag a column header here to group by that column

Alias	Identifier	VariableType
SELECT_COMMAND_Value	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SELECT_...	Parameter
SEND_COMMAND_Value	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SEND_CO...	Parameter
SUBSYSTEM_RX_TELNET_CommandResult	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SUBSYST...	Measurement
CONNECT_Value	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/CONNEC...	Parameter
STATUS_CONNECTION_TELNET_Out1	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/STATUS_...	Measurement
LOGIN_STATUS_Out1	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/LOGIN_S...	Measurement

Fonte: Autor

Partindo para a integração entre o treinamento de imagem e o AutomationDesk. Primeiro temos ***Value_0_999__Value***, responsável por converter o input de um número inteiro para a *string*, respeitando o formato JOB_xxx, onde xxx é o valor digitado anteriormente, que é então enviado para o software da câmera, através do comando ***Select_Job_Value*** (1 envia o comando contendo a informação). ***SelectJOB_SelectResult*** é responsável por dar o *feedback* para o se o job foi selecionado corretamente (1) ou não (0). Já a variável ***JobPass_Out1*** é informa ao sistema se a imagem capturada pela câmera é a mesma do treinamento (1 para sim e 0 para não.

Figura 46: Variáveis de seleção do treinamento no *mapping*


Mapping Viewer

Project[redacted].Mapping

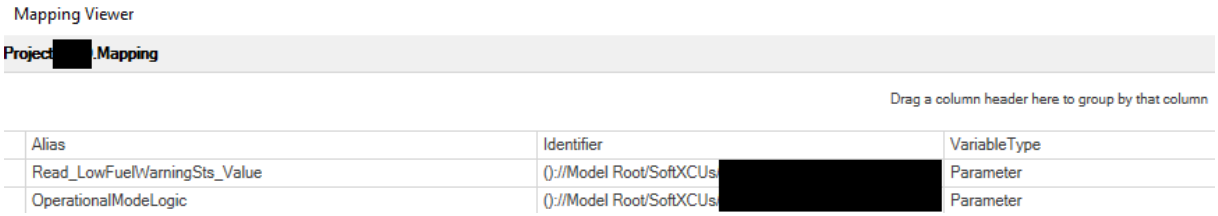
Drag a column header here to group by that column

Alias	Identifier	VariableType
Select_Job_Value	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SelectJob...	Parameter
Value_0_999__Value	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SDCARD...	Parameter
SelectJOB_SelectResult	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/SelectJob...	Measurement
JobPass_Out1	():://Model Root/Network/ETHERNET/COGNEX_ETHERNET/JobPass/...	Measurement

Fonte: Autor

Por fim, são mapeadas as variáveis responsáveis por atuar diretamente no IPC. A primeira é ***OperationalModeLogic***, que atua de maneira síncrona a variável ***Control_Value2***, controlando os estados de ignição do IPC conforme os valores descritos anteriormente. Já ***Read_LowFuelWarningSts_Value*** aciona a indicação (caso o valor seja 1) de baixo nível de combustível no quadro de instrumentos.

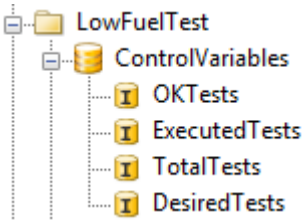
Figura 47: Variáveis que manipularão o IPC



Fonte: Autor

Por fim, são criadas outras 4 variáveis, dentro do próprio AutomationDesk, que atuam com objetivo de contabilizar a quantidade de testes realizados no total (***TotalTests***), quantos testes foram efetivamente executados (***ExecutedTests***), o número de aprovações (***OKTests***) e um input para o usuário escolher quantos testes ele deseja realizar para cada iteração (***DesiredTests***). Essas variáveis ficam dentro da lista pertencente a sequência lógica que será testada, como pode ser visto abaixo:

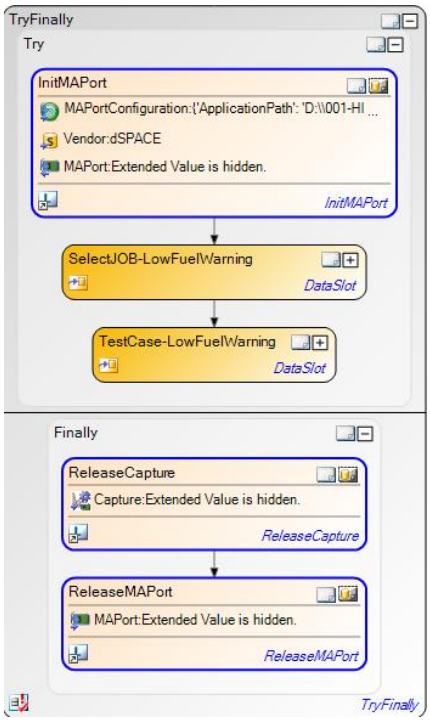
Figura 48: Variáveis de controle



Fonte: Autor

Tendo mapeado todas as variáveis necessárias para a execução do teste, podemos enfim começar a montar qual a sequência de execução. Esse processo é constituído por 5 etapas, que serão descritas individualmente mais abaixo. O intuito é gerar uma estrutura sequencial, conforme mostrado na figura abaixo:

Figura 49: Estrutura condensada da sequência de testes

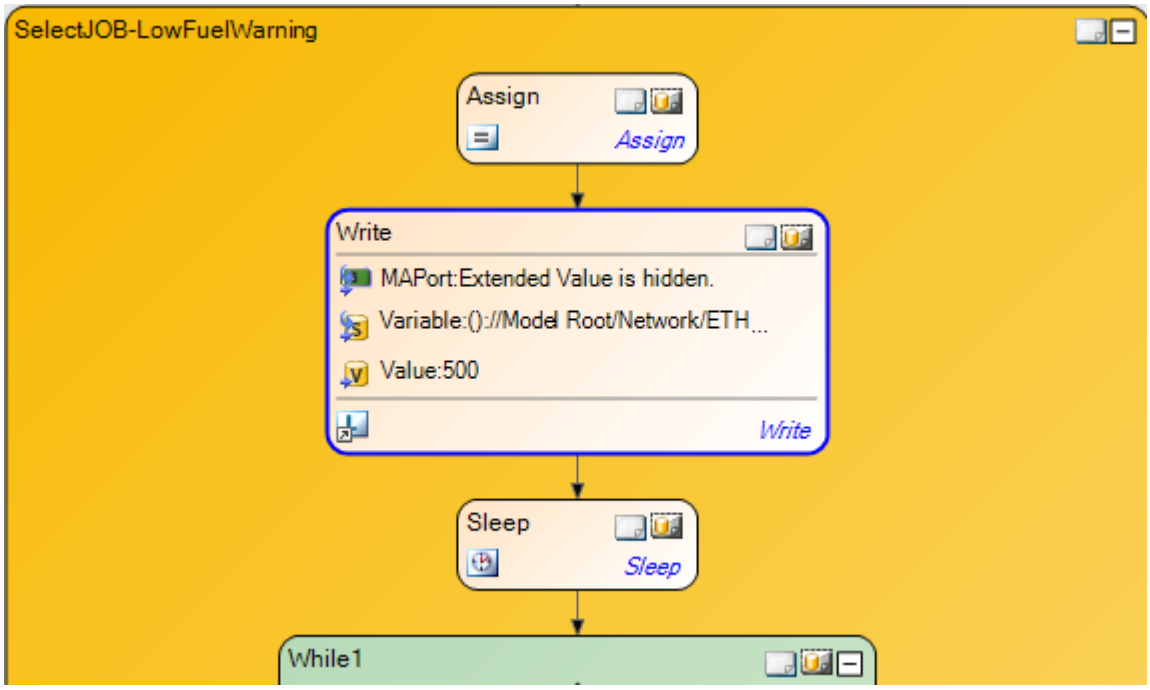


Fonte: Autor

A primeira etapa consiste em inicializar a configuração dos I/Os, carregando no bloco ***InitMAPort*** o ***MAPortConfiguration***, que foi mostrado anteriormente. Em seguida, é fornecida uma string que vai ser responsável por buscar qual a XIL API será usada. Ao digitar dSPACE, a XIL API utilizada será a que é compatível com a versão do AutomationDesk utilizada. Por fim, é dado um *output* para o software contendo a configuração que será utilizada.

Na segunda etapa, o AutomationDesk comunica-se com a câmera para selecionar qual o treinamento (*job*) será usado no teste. Primeiro, é designado o valor 1 para uma variável chamada TrySelectJob, responsável por quantificar o total de tentativas que foram feitas para selecionar o treinamento. No primeiro bloco ***Write***, é escrito na variável ***Value_0_999__Value*** o valor do *job*, sendo este um valor inteiro e em conformidade com o que foi descrito anteriormente. Nesse caso, o número é 500. Temos então um ***Sleep*** para garantir que o processamento ocorra, conforme imagem abaixo:

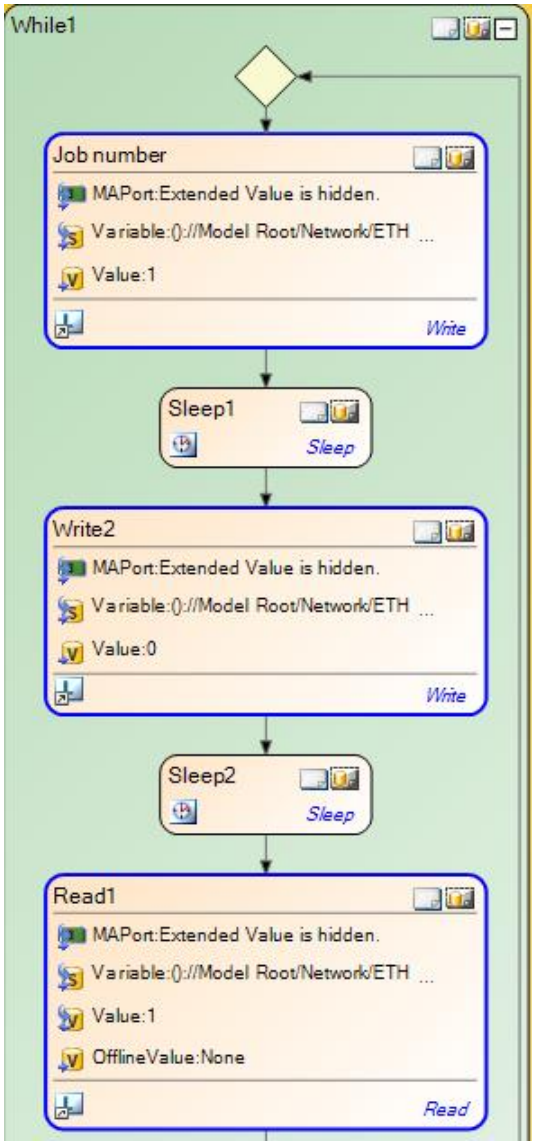
Figura 50: Seleção de qual o valor do *job* será usado no teste



Fonte: Autor

O bloco Job number (***Write***) é responsável por enviar o valor 1 para o comando ***Select_Job_Value***. Em seguida, o software aguarda 1 segundo e logo em seguida escreve o valor 0 no mesmo comando, garantindo que a instrução para selecionar o *job* seja enviada apenas uma vez. Após outro ***Sleep*** (1 s), o AutomationDesk lê, no bloco ***Read1***, o *feedback* da câmera em relação ao comando dado, através da variável ***SUBSYSTEM_RX_TELNET_CommandResult***.

Figura 51: Envio do comando para selecionar o *job*

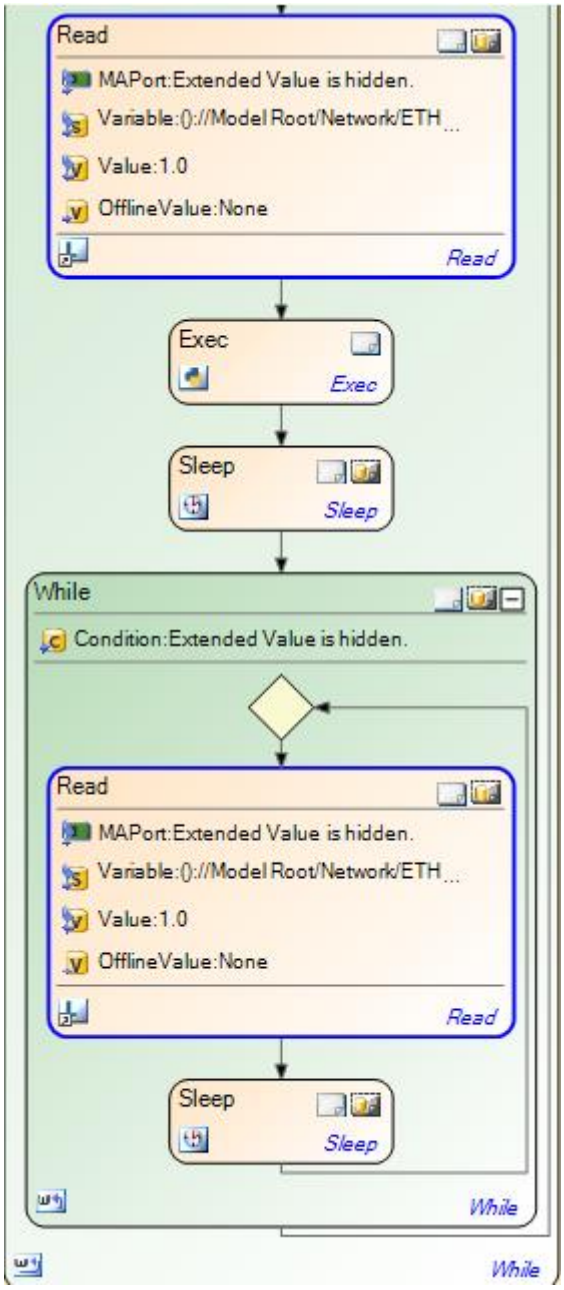


Fonte: Autor

Ainda dentro do laço ***While1*** da figura anterior, após a execução do bloco ***Read1***, temos o ***Read***, que é responsável ler o *feedback* da variável ***SelectJOB_SelectResult***, enquanto o ***Exec*** incrementa o parâmetro que contabiliza o número de tentativas realizadas de selecionar o job. O ***Sleep*** (1 s) garante que a leitura terá tempo suficiente de ser realizada antes de executar o próximo passo. Dentro do próximo laço, chamado de ***While*** (condição *SelectJOB_SelectResult != 0 or SelectJOB_SelectResult != 1*), será avaliado se a leitura da variável ***SelectJOB_SelectResult*** retornou um valor válido (0 ou 1). Caso não atenda a condição do laço, é realizada uma nova leitura do *feedback* fornecido pela câmera no bloco ***Read***, com um ***Sleep*** (1 s) logo em seguida, que tem o mesmo propósito do

anterior. Esse loop se repete até que seja possível selecionar o valor correto do treinamento ou o número máximo de tentativas seja atingido.

Figura 52: Leitura da seleção do *job* no AutomationDesk

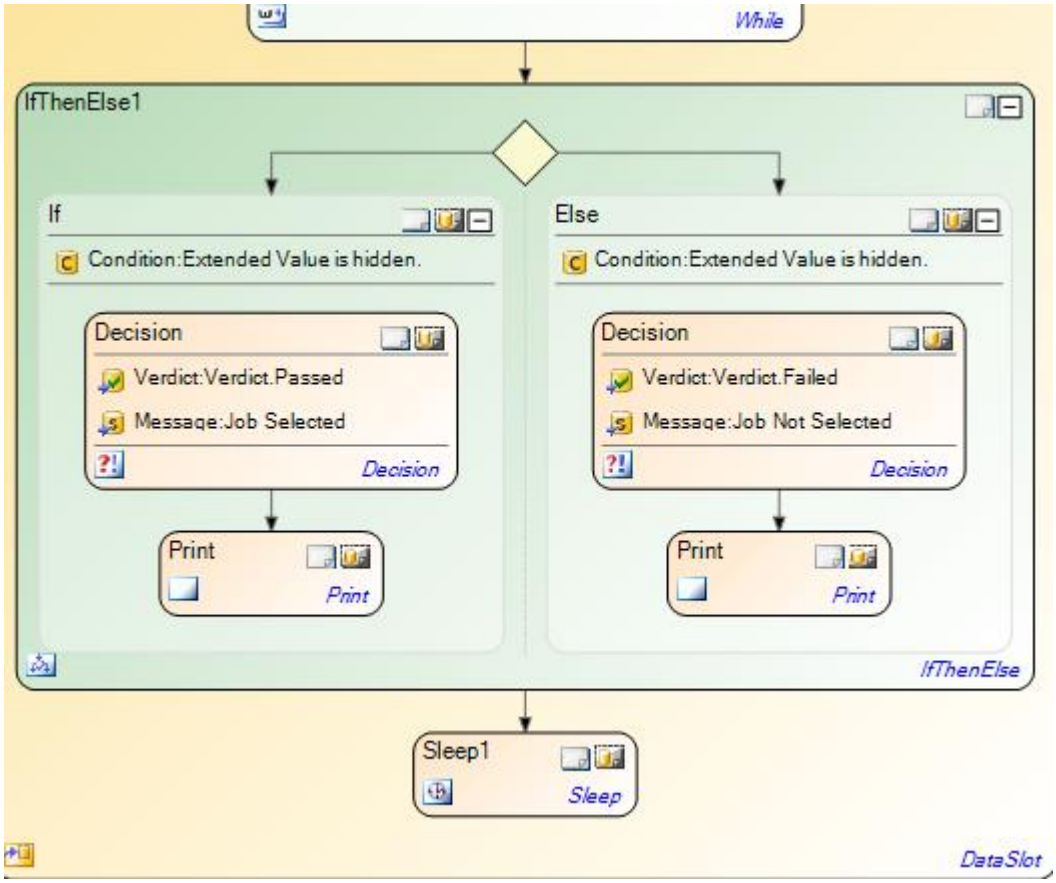


Fonte: Autor

Para finalizar a etapa de seleção do job, temos um *feedback* para o usuário, que será escrito no relatório final do teste. Dentro do bloco ***IfThenElse1***, temos duas condições: se a variável ***SelectJOB_SelectResult*** possuir valor 1 (indicando que a seleção foi executada), o software fornece um veredito positivo (bloco ***Decision***) e

escreve no relatório a seguinte mensagem: “Job Selected”, através do ***Print***. Caso o valor seja 0, o software dará o veredito negativo e a mensagem será: “Job Not Selected”, finalizando logo em seguida a execução do teste. O ***Sleep*** está presente para que o processamento possa ser concluído antes de prosseguir para a etapa 3.

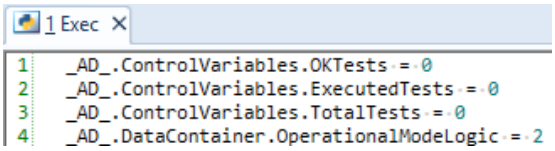
Figura 53: Veredito final do SelectJOB



Fonte: Autor

Tendo finalizado corretamente a seleção do treinamento de imagem, o AutomationDesk passa então para a terceira etapa do processo, que é executar o acionamento da indicação e a receber o *feedback* da câmera se o que foi exibido no IPC condiz com aquilo que está no *job*. Primeiro, é executado um bloco de comando (***Exec***) que é responsável por deixar os parâmetros necessários na sua condição inicial, conforme pode ser visto abaixo:

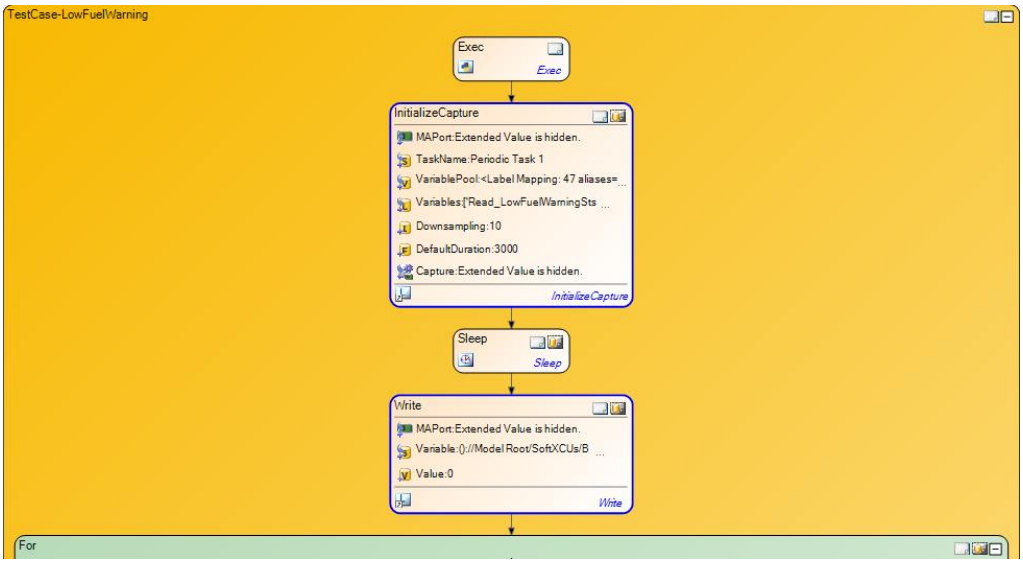
Figura 54: Inicialização dos parâmetros através do Exec



Fonte: Autor

Em seguida, executa-se a inicialização da câmera, através do ***InitializeCapture***. Aqui, os inputs são novamente o ***MAPortConfiguration***, que contém as informações sobre os I/Os da câmera, também é inserido o ***VariablePool***, que nada mais é que o mapeamento das variáveis do sistema, conforme explicado no início desde tópico. Em seguida, define-se quais os parâmetros serão necessários para a execução dessa sequência de testes em específico (neste caso ***OperationalModeLogic*** e ***Read_LowFuelWarningSts_Value***, além das variáveis relacionadas a câmera). O ***Downsampling*** armazena o valor dos parâmetros de *feedback* da câmera a cada 10 amostras e o ***DefaultDuration*** indica que a duração da captura para cada iteração seja de 3000 segundos (valor escolhido arbitrariamente, a intenção é garantir que a câmera sempre esteja disponível para realizar a captura da imagem). Após o tempo definido no ***Sleep*** (4 s), o software escreve (***Write***) na variável ***Read_LowFuelWarningSts_Value*** o valor 0, garantindo que a indicação não esteja ativa no início da sequência de testes.

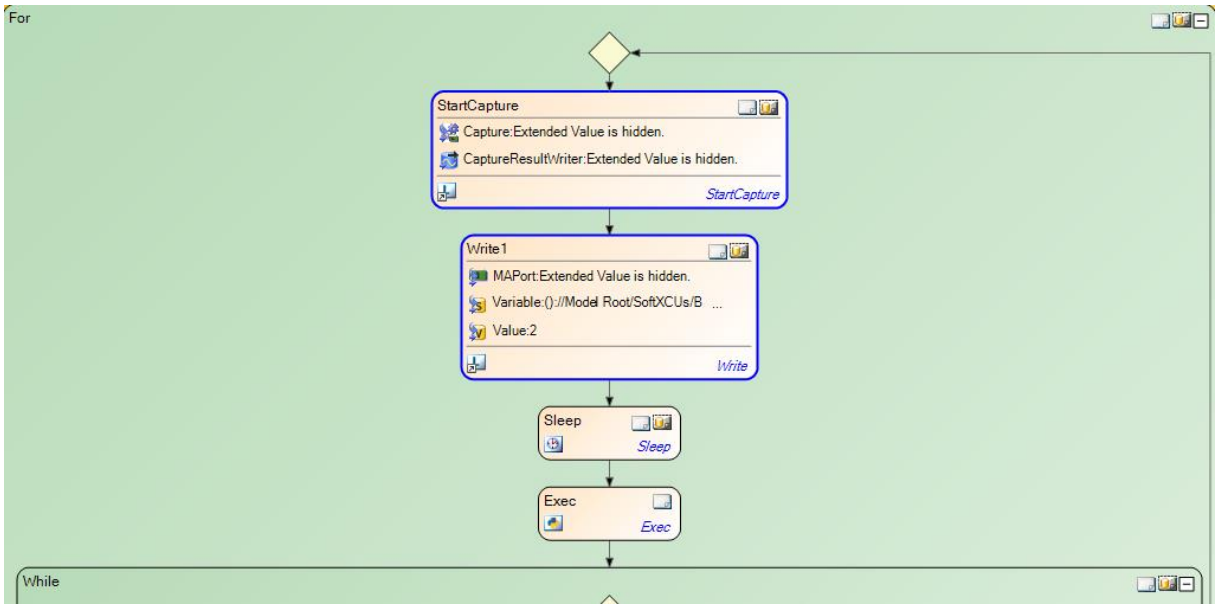
Figura 55: Início da sequência do TestCase



Fonte: Autor

Inicia-se então um laço ***For***, que inicia-se em 1 e vai até 3, garantindo que o test case seja realizado de maneira cíclica até se atingir todos os estados de ignição desejados (3 loops no total). Em seguida, o bloco ***Write1*** irá escrever na variável ***OperationalModeLogic*** o valor inicial igual a 2, representando a condição de ignição desligada. Após um tempo de 10 s (***Sleep***), executa-se então o comando de ***Exec***, responsável por zerar o valor das variáveis responsáveis por contabilizar o total de testes executados, quantos foram realmente executados e o número de aprovações.

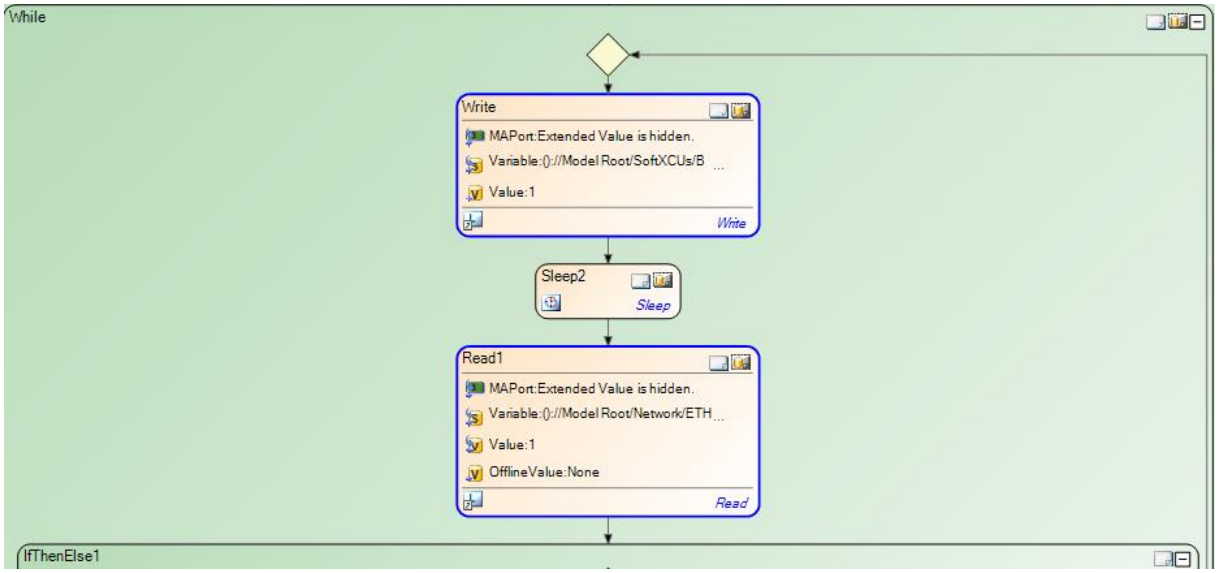
Figura 56: Etapa inicial da execução da sequência de teste



Fonte: Autor

No bloco ***While*** que vem a seguir, existe uma condição que é responsável por executar a sequência de testes enquanto o número total de testes for menor que a quantidade de testes desejados. A partir daí, o AutomationDesk irá escrever (***Write***) o valor 1 na variável ***Read_LowFuelWarningSts_Value***, ativando a indicação de combustível baixo e, após 4 s, será lido o ***JobPass_Out1*** no bloco ***Read1***, que contém o *feedback* da câmera, com a informação se a indicação acionada está conforme o treinamento ou não.

Figura 57: Sequência de acionamento da indicação e leitura (início bloco While)

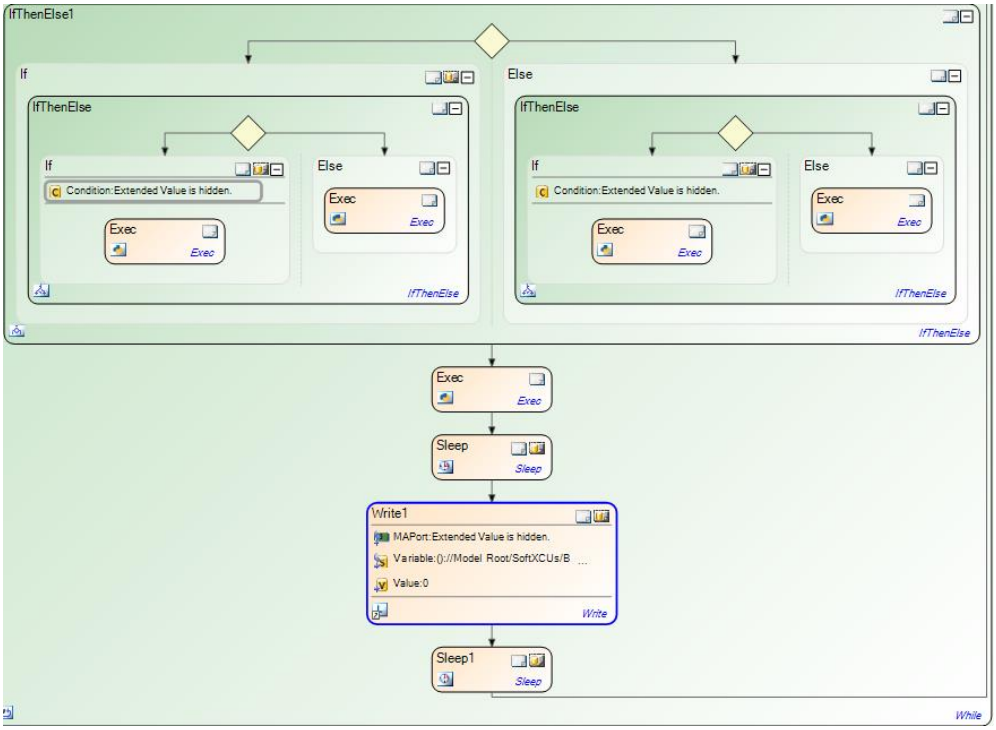


Fonte: Autor

A seguir, o ***IfThenElse1*** é condicionado da seguinte forma: primeiro, ele irá verificar o valor da variável ***OperationalModeLogic*** para definir se o funcionamento do IPC está correto ou não. No primeiro *loop* de teste, que é executado com a ignição desligada, a leitura feita do ***JobPass_Out1*** tem que ser igual a 0, uma vez que a indicação de combustível baixo não é exibida para esse estado de ignição. Nos próximos 2 *loops* de teste, que são realizados primeiro com a ignição ligada e depois com a ignição e o motor ligados, o alerta de combustível baixo precisa ser exibido, portanto o valor de ***JobPass_Out1*** precisa ser igual a 1. Validando essas condições, o AutomationDesk irá incrementar, através do ***Exec*** a variável respectiva para teste bem-sucedido. Caso contrário, será incrementado na variável de teste mal-sucedido. Também se incrementa a variável com o total de testes executados com sucesso.

Após finalizar a verificação no bloco anterior, o software irá incrementar a variável ***TotalTests***, aguardar 1 segundo e escrever (***Write1***) em ***Read_LowFuelWarningSts_Value*** o valor de 0, desativando a indicação e retornando a condição inicial para a execução do próximo *loop* de testes.

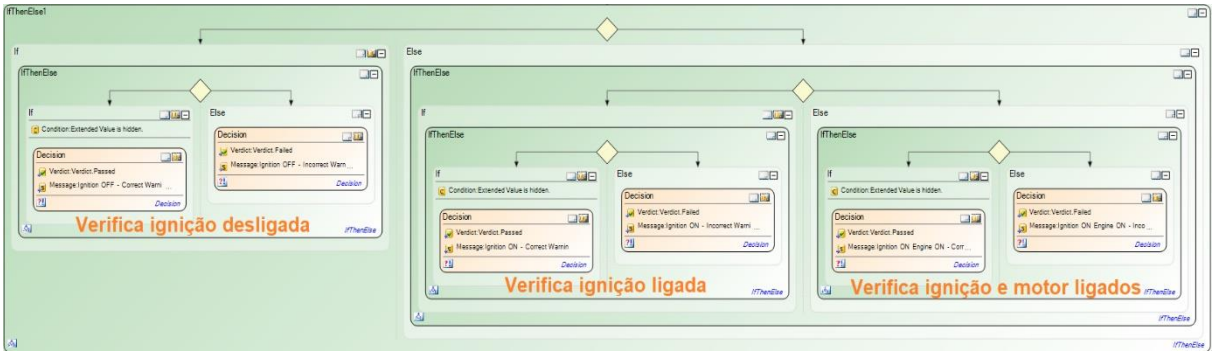
Figura 58: Parte final da sequência de testes do bloco While



Fonte: Autor

Após a finalização do laço ***While*** anterior, é então executado um outro bloco ***IfThenElse***, que é responsável por verificar a proporção entre o total de testes realizados e o total de testes que passaram (***OKTests/ExecutedTests***), esse valor precisa ser igual a 1, o que indica que 100% dos testes foram bem-sucedidos. Essa verificação é feita para cada uma das condições de ignição (através do valor da variável ***OperationalModeLogic***) descritas anteriormente, dentro de cada um dos blocos ***IfThenElse*** adjacentes, conforme é mostrado na imagem abaixo:

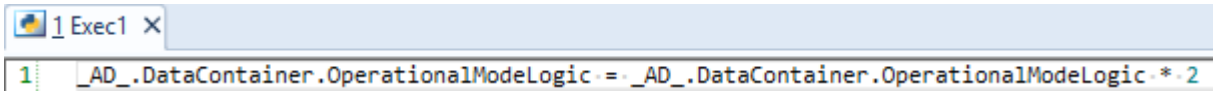
Figura 59: Verificação dos resultados do teste dentro de cada condição de ignição



Fonte: Autor

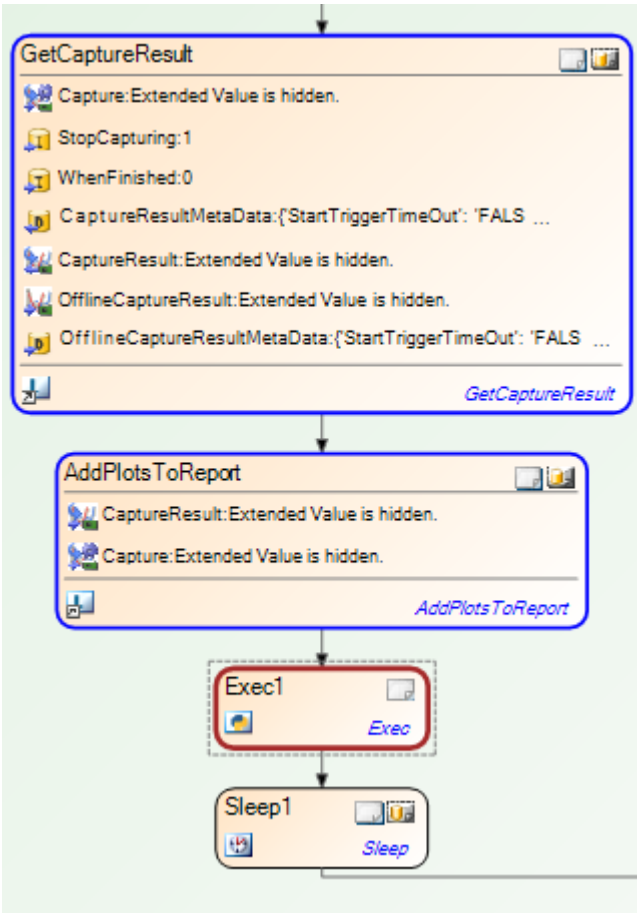
Finalizando essa etapa de verificação dos resultados dos testes, chegamos ao fim da execução do laço ***For*** com os últimos 4 blocos da sequência: primeiro, é coletado adquirido o resultado da captura da câmera, que fica armazenado internamente na câmera (CaptureResult). Em seguida, são adicionados os gráficos ao relatório final. Esses gráficos contém o valor ao longo do tempo dos parâmetros ***Read_LowFuelWarningSts_Value*** e ***JobPass_Out1***, mostrando a relação entre a condição de ativação da indicação e o *feedback* da câmera. Em seguida, no bloco ***Exec1*** a condição de ignição é alterada para o próximo valor, através da linha de código mostrada na Figura 60. A sequência final é mostrada na figura 61.

Figura 60: Linha de código do Exec1



Fonte: Autor

Figura 61: Final da execução do laço For

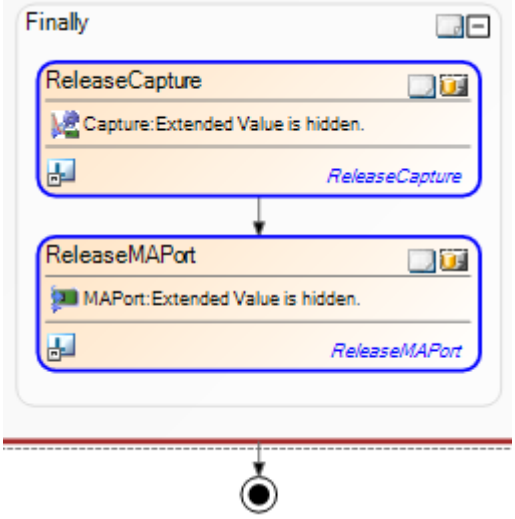


Fonte: Autor

O laço ***For*** será então repetido para as próximas duas condições de chave, executando os mesmos passos descritos anteriormente.

Após a execução do bloco ***TestCase-LowFuelWarning***, mostrado na Figura 55, o AutomationDesk irá executar mais 2 passos antes de gerar o relatório final do teste. Primeiro, ele irá desligar a câmera através do ***ReleaseCapture*** e em seguida liberar os I/Os do hardware pelo ***ReleaseMAPort***.

Figura 62: Blocos finais no AutomationDesk



Fonte: Autor