

INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E
TECNOLOGIA DE SANTA CATARINA
CÂMPUS ITAJAÍ

GABRIEL MOTA

**DESENVOLVIMENTO DE UM MAIN CONTROL DISPLAY UNIT PARA
AERONAVES EMBRAER E-195 PARA USO EM SIMULADORES DE VOO
X-PLANE**

ITAJAÍ
2026

Gabriel Mota

Desenvolvimento de um *Main Control Display Unit* para aeronaves Embraer E-195
para uso em simuladores de voo X-Plane

Monografia apresentada ao curso de Engenharia Elétrica do Instituto Federal de Santa Catarina, para obtenção do título de bacharel em Engenharia Elétrica.

Área de concentração: Circuitos elétricos, magnéticos e eletrônicos.

Orientador: Prof. Sergio A. B. Petrovcic, Dr. Eng.

Ficha de Identificação da obra elaborada pelo autor, através do cadastro de ficha de identificação disponível no portal discente do Sistema Integrado de Gestão Acadêmica - SIGAA, do IFSC.

Mota, Gabriel

Desenvolvimento de um Main Control Display Unit para aeronaves Embraer E-195 para uso em simuladores de voo X-Plane / Gabriel Mota ; orientador(a): Sergio A. B. Petrovcic. -- Itajaí : 2026.

107 p.

Trabalho de conclusão de curso (Graduação) - Instituto Federal de Santa Catarina, Campus Itajaí. Curso de Bacharelado em Engenharia Elétrica, Itajaí, 2026.

Inclui referências.

1. Main Control Display Unit. 2. Embraer E-195. 3. Simuladores de voo. 4. Sistemas embarcados. 5. X-Plane. I. Petrovcic, Sergio A. B.. II. Instituto Federal de Santa Catarina, Curso de Bacharelado em Engenharia Elétrica. III. Título.

Gabriel Mota

Desenvolvimento de um *Main Control Display Unit* para aeronaves Embraer E-195
para uso em simuladores de voo X-Plane

Trabalho de Conclusão de Curso apresentado ao
curso de Engenharia Elétrica do Instituto Federal
de Santa Catarina, para obtenção do título de
bacharel em Engenharia Elétrica.

Área de concentração: 3.04.03.03-0

Itajaí, 06 de Agosto de 2026.



Documento assinado digitalmente
FERNANDA ISABEL MARQUES ARGOUD...SILVA
Data: 02/09/2026 18:17:14-0300
CPF: ***.421.939-**
Verifique as assinaturas em <https://v.ufsc.br>

Dra. Fernanda Isabel Marques Argoud
Instituto Federal de Santa Catarina



Documento assinado digitalmente
SERGIO AUGUSTO BITENCOURT PETROVCIC
Data: 01/09/2026 14:46:07-0300
CPF: ***.649.159-**
Verifique as assinaturas em <https://v.ufsc.br>

Dr. Sergio A. B. Petrovcic
Instituto Federal de Santa Catarina



Documento assinado digitalmente
TIAGO DRUMMOND LOPES
Data: 01/09/2026 16:09:45-0300
CPF: ***.244.128-**
Verifique as assinaturas em <https://v.ufsc.br>

Dr. Tiago Drummond Lopes
Instituto Federal de Santa Catarina

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por ter me concedido toda a condição de concluir esta etapa importante da minha vida. Agradeço à minha esposa, familiares e amigos pelo apoio durante não somente o trabalho de conclusão, mas também durante todo o curso. Agradeço ao meu orientador por ter me apoiado, acreditado no meu projeto, e ter me ajudado a alcançar o melhor do meu trabalho. Agradeço também aos colegas de faculdade que me ajudaram durante o curso, assim como todo o corpo docente, que me deu o conhecimento e ajuda necessária para concluir o curso. Por fim, agradeço às pessoas que me conectei durante a execução do projeto, como a Embraer, por me conceder uma reunião e sanar dúvidas.

RESUMO

O uso de simuladores de voo tem crescido tanto para capacitação e treinamento de pilotos quanto para fins recreativos. Entretanto, observa-se a ausência de dispositivos dedicados, que reproduzam de maneira fiel, o *Main Control Display Unit* (MCDU) utilizado nas aeronaves Embraer E-195. Diante deste cenário, este trabalho teve como objetivo desenvolver um protótipo funcional de MCDU, incluindo o desenvolvimento e integração de *hardware* e *software* necessários para comunicação com simuladores de voo X-Plane. Para isso, buscou-se projetar o MCDU compatível com o equipamento presente nas aeronaves Embraer E-195, construir o protótipo funcional e realizar testes para a validação do sistema desenvolvido. Sendo assim, foram realizadas buscas sobre o modelo real a fim de atribuir ao protótipo as suas características, além de análises e extração dos dados do simulador de voo X-Plane, o desenvolvimento de uma arquitetura de comunicação baseada em protocolos TCP/IP e WebSocket, implementação de *hardware* utilizando um Raspberry Pi 4 Model B e o desenvolvimento das placas de circuito impresso necessárias para o completo funcionamento do dispositivo. Os resultados demonstraram a viabilidade técnica do projeto, permitindo a comunicação entre o simulador de voo e o protótipo físico de maneira bidirecional, com exibições em tempo real e envio de comandos. É possível concluir que os objetivos propostos foram atingidos, resultando em um dispositivo funcional, compatível com simuladores de voo X-Plane, reproduzindo de maneira fiel o MCDU utilizado nas aeronaves Embraer E-195.

Palavras-chave: *Main Control Display Unit*; Embraer E-195; Simuladores de voo; Sistemas embarcados; X-Plane.

ABSTRACT

The use of flight simulators has increased both for pilot qualification and training purposes, as well as for recreational use. However, there is a lack of dedicated devices capable of faithfully reproducing the Main Control Display Unit (MCDU) used in Embraer E-195 aircraft. In this context, the objective of this work was to develop a functional MCDU prototype, including the development and integration of the hardware and software required for communication with X-Plane flight simulators. To achieve this, the project sought to design an MCDU compatible with the equipment used in Embraer E-195 aircraft, build a functional prototype, and perform tests to validate the developed system. Therefore, research was conducted on the real equipment in order to incorporate its characteristics into the prototype, in addition to the analysis and extraction of data from the X-Plane flight simulator, the development of a communication architecture based on TCP/IP and WebSocket protocols, the implementation of hardware using a Raspberry Pi 4 Model B, and the development of the printed circuit boards required for the complete operation of the device. The results demonstrated the technical feasibility of the project, enabling bidirectional communication between the flight simulator and the physical prototype, with real-time information display and command transmission. It can be concluded that the proposed objectives were achieved, resulting in a functional device compatible with X-Plane flight simulators and capable of faithfully reproducing the MCDU used in Embraer E-195 aircraft.

Keywords: Main Control Display Unit; Embraer E-195; Flight simulators; Embedded systems; X-Plane.

LISTA DE FIGURAS

Figura 1 - Alberto Santos Dumont no primeiro voo do 14-bis.....	21
Figura 2 - Linha do tempo na aviação.....	22
Figura 3 - Exemplo de sistemas embarcados de uma aeronave.....	23
Figura 4 - Superfícies de uma aeronave.....	24
Figura 5 - Sistema <i>Fly-By-Wire</i>	25
Figura 6 - <i>Cockpit</i> analógico Embraer T-27 Tucano.....	26
Figura 7 - Comparação entre sistema dedicado e IMA.....	27
Figura 8 - EFIS Embraer E-Jets.....	28
Figura 9 - Diagrama de um FADEC.....	29
Figura 10 - <i>Flight Management System</i> GE 2907C1.....	30
Figura 11 - <i>Cockpit</i> Embraer T-27 Tucano atualizado com <i>Glass Cockpit</i>	32
Figura 12 - Conjunto do computador de gerenciamento de voo Honeywell.....	33
Figura 13 - Descrição de um MCDU.....	34
Figura 14 - Conjunto de controle e navegação.....	35
Figura 15 - Simulador de Antoinette.....	36
Figura 16 - Link Trainer.....	37
Figura 17 - <i>Flight Training Device</i> DELTA 5 Simuladores.....	38
Figura 18 - <i>Flight Training Device</i> CAE 600XR.....	40
Figura 19 - <i>Full Flight Simulator</i> CAE 7000XR.....	41
Figura 20 - Menu inicial X-Plane 12.....	45
Figura 21 - Menu de novo voo X-Plane 12.....	46
Figura 22 - Ambiente de simulação.....	46
Figura 23 - DataRef Tool.....	47
Figura 24 - MCDU Airbus Winctrl.....	50
Figura 25 - <i>Flight Deck</i> Epic Load.....	52
Figura 26 - MCDU Load 27 Honeywell.....	53
Figura 27 - Datarefs do E-195 desenvolvido pela X-Crafts.....	55
Figura 28 - Exibição do MCDU no momento da captação dos dados.....	56

Figura 29 - Teste de extração dos dados via <i>prompt</i>	58
Figura 30 - Código presente no TcpClient.js.....	59
Figura 31 - Informação recebida no terminal do Visual Studio.....	59
Figura 32 - Linhas da página Rádio no terminal do Visual Studio.....	60
Figura 33 - Informações da página Rádio convertidas para <i>string</i>	61
Figura 34 - Dados da página de Rádio organizados via LLCCEE.....	62
Figura 35 - Fluxograma do sistema.....	63
Figura 36 - Primeiro envio de dados para página <i>web</i>	65
Figura 37 - Dados misturados na página <i>web</i>	66
Figura 38 - Classes do texto para o CSS.....	67
Figura 39 - Dados recebidos na página <i>web</i> utilizando <i>grid</i> 14x24.....	68
Figura 40 - Informações exibidas no <i>display</i> do MCDU no simulador.....	69
Figura 41 - MCDU com botões adicionados.....	70
Figura 42 - Raspberry Pi 4 Model B.....	71
Figura 43 - Esquema elétrico teclado matricial.....	72
Figura 44 - Acionamento tecla 1.....	73
Figura 45 - Protótipo PCB do teclado em placa de fenoliteilhada.....	74
Figura 46 - Parte do esquema elétrico da PCB teclado.....	76
Figura 47 - Visualização 3D da PCB do teclado no Altium Designer.....	77
Figura 48 - Circuito típico sugerido pelo <i>datasheet</i>	78
Figura 49 - Parte do esquema elétrico da PCB de controle.....	79
Figura 50 - Visualização das trilhas em 2D no Altium Designer.....	80
Figura 51 - Visualização em 3D no Altium Designer.....	80
Figura 52 - Visualização em 3D do conjunto case e PCBs.....	81
Figura 53 - PCBs recebidas após fabricação.....	84
Figura 54 - PCB montada.....	85
Figura 55 - Protótipo montado.....	87
Figura 56 - Comparação com modelo real.....	94

LISTA DE TABELAS

Tabela 1 - Lista de materiais de <i>hardware</i> e custo.....	83
Tabela 2 - Custo total do protótipo.....	93

LISTA DE ABREVIATURAS E SIGLAS

ANAC	Agência Nacional de Aviação Civil
BOM	<i>Bill of Materials</i>
CAN	<i>Controller Area Network</i>
CB	<i>Circuit Breakers</i>
CDU	<i>Control Display Unit</i>
DME	<i>Distance Measuring Equipment</i>
EASA	<i>European Union Aviation Safety Agency</i>
EICAS	<i>Engine Indicating and Crew Alert System</i>
EFIS	<i>Electronic Flight Instrument System</i>
FAA	<i>Federal Aviation Administration</i>
FADEC	<i>Full Authority Digital Engine Control</i>
FBW	<i>Fly-By-Wire</i>
FCU	<i>Flight Control Unit</i>
FFS	<i>Full Flight Simulator</i>
FMC	<i>Flight Management Computer</i>
FMS	<i>Flight Management System</i>
FSTD	<i>Flight Simulator Training Device</i>
FTD	<i>Flight Training Device</i>
GPS	<i>Global Positioning System</i>
GPWS	<i>Ground Proximity Warning System</i>
HUD	<i>Head-Up Display</i>
IFR	<i>Instrument Flight Rules</i>
IMA	<i>Integrated Modular Avionics</i>
IRS	<i>Inertial Reference System</i>
JPR	<i>Jon Peddie Research</i>
LSK	<i>Line Select Key</i>
MCDU	<i>Main Control Display Unit</i>
ND	<i>Navigation Display</i>
PCB	<i>Printed Circuit Board</i>
PFD	<i>Primary Flight Display</i>
RBAC	Regulamento Brasileiro da Aviação Civil

TCAS	<i>Traffic Collision Avoidance System</i>
TCP/IP	<i>Transmission Control Protocol / Internet Protocol</i>
VOR	<i>VHF Omnidirectional Range</i>

SUMÁRIO

1 INTRODUÇÃO.....	15
1.1 OBJETIVO GERAL.....	17
1.2 OBJETIVOS ESPECÍFICOS.....	17
1.3 JUSTIFICATIVA.....	17
1.4 METODOLOGIA.....	18
1.4.1 Projeto do MCDU.....	18
1.4.2 Construção do Protótipo.....	19
1.4.3 Testes e Validação.....	19
2 FUNDAMENTAÇÃO TEÓRICA.....	20
2.1 AVIAÇÃO E SISTEMAS EMBARCADOS.....	20
2.1.1 Surgimento da Eletrônica Embarcada.....	22
2.1.2 Conceito de Aviônicos.....	25
2.1.2.1 EFIS.....	28
2.1.2.2 FADEC.....	29
2.1.2.3 FMS.....	29
2.1.2.4 Glass Cockpit.....	31
2.1.3 MCDU.....	32
2.2 SIMULADORES DE VOO.....	35
2.2.1 Simuladores de Voo Para Capacitação.....	37
2.2.1.1 Flight Training Device.....	37
2.2.1.2 Full Flight Simulator.....	40
2.2.2 Simuladores de Voo de Uso Doméstico.....	42
2.2.2.1 Interface de Um Simulador de Uso Doméstico.....	44
3 PRODUTOS RELACIONADOS.....	49
4 DESENVOLVIMENTO.....	52
4.1 PROJETO DO MCDU.....	54
4.1.1 Obtenção dos Dados do Simulador.....	54
4.1.2 Tratamento dos Dados.....	58
4.1.3 Arquitetura de Comunicação.....	62
4.1.4 Desenvolvimento da Interface Gráfica.....	64
4.1.5 Envio de Comandos ao Simulador.....	69
4.2 CONSTRUÇÃO DO PROTÓTIPO.....	71
4.2.1 Desenvolvimento do Teclado Matricial.....	72
4.2.2 Desenvolvimento das Placas de Circuito Impresso.....	75
4.2.3 Sistema de Alimentação.....	77
4.2.4 Integração Mecânica do Protótipo.....	80
4.2.5 Fabricação e Montagem do Protótipo.....	81
4.3 TESTES E VALIDAÇÃO.....	87
4.3.1 Teste de Alimentação e Estabilidade Elétrica.....	88

4.3.2 Testes Funcionais e Integrados do Protótipo.....	90
5 RESULTADOS.....	91
5.1 RESULTADOS DOS TESTES E VALIDAÇÃO.....	91
5.2 ANÁLISE DE CUSTOS.....	92
5.3 COMPARAÇÃO COM O MCDU DE REFERÊNCIA.....	93
6 CONCLUSÃO.....	95
6.1 LIMITAÇÕES DO PROJETO.....	97
6.2 PROPOSTAS PARA TRABALHOS FUTUROS.....	98
REFERÊNCIAS.....	100
APÊNDICE A – Circuito Elétrico PCB Teclado.....	105
APÊNDICE B – Circuito Elétrico PCB Controle.....	106
APÊNDICE C – Vídeo de Demonstração de Funcionamento do Protótipo.....	107

1 INTRODUÇÃO

A aviação é um importante e complexo setor da economia global, não apenas por exercer o papel de modelo de transporte eficiente, mas também pelo fator econômico, tecnológico e industrial associado à produção de aeronaves. Em solo brasileiro, está sediada a Embraer, fundada em 1969 e atualmente considerada a terceira maior fabricante de jatos comerciais do mundo, atrás apenas da Boeing e da Airbus, sendo líder mundial no segmento de aeronaves até 150 assentos (BBC, 2025b). Apesar de sua posição de destaque, a tecnologia dos aviônicos embarcados em suas aeronaves não é produzida pela Embraer, mas sim por parceiros comerciais que firmam contrato com a empresa brasileira e fornecem os equipamentos os quais equipam as aeronaves. Dentre as fornecedoras de material eletrônico, está a Honeywell, a qual fornece os equipamentos para os sistemas de comunicação, controle de voo e navegação de algumas das aeronaves da fabricante brasileira.

Entre os diversos equipamentos produzidos pela Honeywell, encontra-se o *Flight Management Computer* (FMC), ou computador de gerenciamento de voo, componente responsável pelo controle e interação de diferentes sistemas da aeronave e variáveis de um voo. Dentre as funções do FMC, destacam-se gerenciamento do plano de voo, controle das frequências de rádio, ajuste de parâmetros do motor, controle dos *circuit breakers* (CB) também conhecidos como disjuntores, e outras funções essenciais à operação. A interface física utilizada pelos pilotos para interação com esse sistema, é o *Main Control Display Unit* (MCDU), instalado no *cockpit* e composto por um display e teclado dedicados.

A complexidade e custo desses equipamentos restringem seu uso a aeronaves reais, mas seu princípio de funcionamento inspirou o desenvolvimento de sistemas equivalentes em simuladores de voo, um setor que movimenta bilhões de dólares e possui relevância tanto recreativa como profissional. Segundo a Jon Peddie Research (JPR, 2020), estimou-se que através do Microsoft Flight Simulator 2020, por exemplo, o mercado de *hardware* venderia aproximadamente 2,6 bilhões de dólares em dispositivos como placas de vídeo, telas, processadores, e dispositivos de controle e gerenciamento de voo, como pedais, *joysticks*, manetes de potência e módulos de controle como o MCDU. O uso de simuladores de voo, porém, não é feito apenas por usuários que querem se divertir com a simulação das

operações reais, mas também por usuários que os utilizam para o treinamento de pilotos. Segundo a Aero Magazine, o uso de simuladores de voo pode auxiliar no desenvolvimento do aluno durante as aulas práticas:

O treinamento prático em simuladores domésticos, como o Microsoft Flight Simulator e o X-Plane, pode melhorar o rendimento dos estudantes do curso de piloto privado, podendo reduzir em até 5,5 horas de voo reais o treinamento prático (Aero Magazine, 2024).

Historicamente, os simuladores de voo surgiram através da necessidade de se obter dados e testar parâmetros sem que se colocasse em risco a vida das pessoas e os equipamentos, e que se mantivessem ativos para treinamentos (Fonseca; Monteiro, 2022 apud Costa, 2008). O avanço da tecnologia permitiu que os simuladores se tornassem cada vez mais realistas, e hoje, são utilizados para a formação dos pilotos e também para a capacitação das tripulações nas empresas aéreas. A partir da certificação fornecida pelos órgãos regulamentadores, como a Agência Nacional de Aviação Civil (ANAC) no Brasil, a *European Union Aviation Safety Agency* (EASA) na Europa e a *Federal Aviation Administration* (FAA) nos Estados Unidos, as empresas podem utilizar simuladores de voo, entretanto, existem classificações e tipos de simuladores para esse tipo de uso.

Os equipamentos conhecidos como *Flight Simulator Training Device* (FSTD) e *Full Flight Simulator* (FFS) são os dispositivos utilizados para o treinamento e capacitação de pilotos, e podem ter a função *motion*, ou movimento, para que o nível de realismo do treinamento seja elevado. Esses equipamentos são classificados de acordo com seu nível de fidelidade, variando de FTD-4 a FTD-7 e FFS A a FFS D, sendo o último o modelo o qual possui o maior realismo. Ambos os simuladores fornecem réplicas fiéis às aeronaves simuladas, sendo o FSTD um simulador para introdução de tarefas ou treinamento de elemento, e o FFS para um treinamento completo. No Brasil, a ANAC detalha o desenvolvimento e uso de simuladores FSTD e FFS no RBAC 60 (ANAC, 2020).

Apesar da evolução dos simuladores e da relevância da Embraer no cenário mundial da aviação, ainda existe uma lacuna entre os sistemas embarcados utilizados em aeronaves reais e os dispositivos disponíveis para simuladores de voo de uso doméstico. Essa diferença impacta diretamente na qualidade e imersão da experiência e eficiência do aprendizado. Atualmente, não existe disponível para

aquisição um modelo fiel ao MCDU presente nos E-Jets da Embraer, para uso com os simuladores de voo.

Dessa forma, este trabalho propõe-se a estudar e desenvolver um modelo funcional de MCDU, baseado em *hardware* e *software* acessíveis, buscando aproximar a interação e o comportamento do dispositivo real com as condições de uso em simuladores de voo de uso doméstico. Tal iniciativa visa contribuir com o avanço tecnológico nacional e com o fortalecimento das aplicações educacionais e de pesquisa na área de sistemas embarcados e aviação.

Vale destacar que o modelo de MCDU abordado neste trabalho é adotado na linha E-Jets da Embraer, tanto nas aeronaves E-195 E1 quanto nas aeronaves E-195 E2.

1.1 OBJETIVO GERAL

Desenvolver um protótipo de MCDU, incluindo o desenvolvimento de *software* e *hardware* necessários para o pleno funcionamento do dispositivo em comunicação com simuladores de voo.

1.2 OBJETIVOS ESPECÍFICOS

- Projetar MCDU para aeronaves Embraer E-195;
- Construir um protótipo de MCDU funcional;
- Realizar testes e validar o protótipo.

1.3 JUSTIFICATIVA

Atualmente, não existe disponível para aquisição um modelo de MCDU dedicado a aeronaves Embraer, apenas para aeronaves fabricadas pela Boeing ou pela Airbus. Essa ausência limita o realismo dos simuladores, reduzindo a representatividade da aviação nacional nos ambientes de simulação. Mesmo que esses dispositivos que funcionam para essas aeronaves possam ser convertidos para operar nos modelos Embraer, sua característica física, como a disposição do teclado e funções não são compatíveis com o modelo desejado. Sendo assim, observa-se a necessidade de desenvolver um modelo de MCDU compatível com os

modelos fabricados pela Honeywell e presentes nas aeronaves fabricadas pela Embraer. Dada a importância que a empresa brasileira possui no mercado internacional, esperava-se que os desenvolvedores de *hardware* pudessem disponibilizar modelos físicos dos componentes presentes nas aeronaves brasileiras.

Levando em consideração o consumo de periféricos para simuladores de voo, e a posição da Embraer como terceira maior produtora de aeronaves do mundo, desenvolver um MCDU deste modelo fortalece a comunidade de simuladores de voo, em especial, os brasileiros. Além de contribuir para a comunidade de simuladores, o desenvolvimento deste dispositivo incentiva o aprendizado prático em sistemas embarcados, integração de *hardware-software* e eletrônica aplicada.

1.4 METODOLOGIA

Este trabalho caracteriza-se como uma pesquisa aplicada e experimental, com o foco no desenvolvimento de um protótipo funcional de um *Main Control Display Unit* (MCDU), compatível com sistemas da família E-Jets da Embraer. O projeto foi desenvolvido utilizando o simulador de voo X-Plane, desenvolvido pela Laminar Research, para aeronaves E-195, desenvolvidas pela X-Crafts para X-Plane 12, permitindo a simulação de sistemas embarcados em ambiente virtual.

1.4.1 Projeto do MCDU

Nesta etapa, foi realizada a análise de funcionamento do MCDU presente na aeronave Embraer E-195 no simulador X-Plane, bem como as características do equipamento real, incluindo suas funcionalidades, *layout* físico e lógica de operação. Foram identificados os principais sistemas integrados ao seu funcionamento, bem como sua interface com o *Flight Management System* (FMS).

Em seguida, após identificadas as principais características do MCDU e sua interface, iniciou-se a etapa de projeto para comunicação com o ambiente virtual, através de *datarefs* e *commands* disponibilizados no X-Plane, e a arquitetura de *software* que deveria integrar o protótipo, bem como os requisitos funcionais do dispositivo. Também foram estabelecidos os parâmetros de comunicação entre o simulador e o sistema externo, utilizando protocolos baseados nos sistemas *Transmission Control Protocol / Internet Protocol* (TCP/IP) e WebSockets, com apoio

do *plugin* ExtPlane, para acesso aos dados internos do simulador, e do programa DataRef Tool para obtenção dos *datarefs* específicos da aeronave.

1.4.2 Construção do Protótipo

A etapa de desenvolvimento envolveu a implementação dos sistemas de *software* e *hardware* necessários para o pleno funcionamento do MCDU. Para o *hardware*, optou-se por utilizar comunicação via rede, e portanto, o Raspberry Pi 4 Model B surgiu como opção, em conjunto com um display de 5.6 polegadas, visando garantir capacidade de processamento e fidelidade visual ao sistema.

No desenvolvimento de *software*, foram utilizadas ferramentas como o Visual Studio para implementação da lógica de comunicação, processamento de dados e renderização das informações para o *display*. A integração dos sistemas foi feita através da leitura dos *datarefs* e envio dos *commands* ao simulador, permitindo assim integração em tempo real. Por fim, foi realizada a integração entre *hardware* e *software*, garantindo o funcionamento do sistema desenvolvido.

Para a *case* do produto, foi utilizado um modelo 3D, e todos os botões, suportes e conexões foram pensados para que pudessem se encaixar entre o *hardware* do projeto e sua parte impressa na impressora, para dar acabamento ao projeto.

1.4.3 Testes e Validação

A validação do protótipo foi realizada por meio de testes funcionais, utilizando o simulador X-Plane. Os testes foram realizados para verificar a fidelidade funcional e visual do dispositivo em relação ao modelo real. Foram realizados testes de leitura de dados, envio de comandos e resposta do sistema, avaliando a consistência das informações exibidas e a capacidade de integração com o simulador.

Os resultados foram registrados por meio de capturas de tela, registros fotográficos e análise do comportamento do sistema, permitindo demonstrar a viabilidade do desenvolvimento de um dispositivo físico, que seja compatível com simuladores de voo, sistemas embarcados e que demonstrem fidelidade ao sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Na seção atual, serão abordados os conceitos fundamentais relacionados à aviação e sistemas embarcados, o uso dos simuladores de voo tanto para treinamento como em uso doméstico, o funcionamento de um MCDU real embarcado em uma aeronave Embraer E-195, os métodos de desenvolvimento do MCDU para aplicação em simuladores de voo, e por fim, os dados dos componentes eletrônicos necessários para o desenvolvimento de *hardware* e também de *software* para aplicação embarcada. Com base nesses tópicos, é possível compreender os princípios teóricos e tecnológicos que sustentam o desenvolvimento do protótipo proposto.

2.1 AVIAÇÃO E SISTEMAS EMBARCADOS

No início do século XX, o mundo presenciou uma das descobertas mais importantes da história da humanidade. Em 1906, o brasileiro Alberto Santos Dumont realizou o primeiro voo de uma aeronave autopropulsada, o 14-Bis, marco reconhecido internacionalmente como o início da aviação moderna. Sua criação demonstrou que o homem era capaz de voar por meios próprios, sem depender de fontes externas de propulsão. Há divergências históricas sobre o primeiro voo realizado, em uma disputa entre o brasileiro e os americanos Orville e Wilbur Wright, conhecidos como Irmãos Wright, que dizem ter voado antes de Santos Dumont, em 1903. A grande diferença entre os acontecimentos, é a necessidade de uma espécie de catapulta que era necessária para que a aeronave dos irmãos Wright pudesse voar, e a aeronave de Santos Dumont era propulsionada (BBC, 2025a). Na Figura 1, é possível visualizar Alberto Santos Dumont a bordo de sua aeronave 14-bis, na realização do primeiro voo.

Figura 1 – Alberto Santos Dumont no primeiro voo do 14-bis.



Fonte: FAB (2020).

Com o avanço das décadas, a aviação evoluiu rapidamente, acompanhando o desenvolvimento tecnológico mundial. No Brasil, esse progresso culminou em 1969, com a fundação da Embraer, liderada por Ozires Silva, e o lançamento do Embraer Bandeirante, primeira aeronave nacional de transporte militar e civil. Esse marco consolidou a indústria aeronáutica brasileira no cenário internacional e abriu caminho para uma série de inovações tecnológicas que tornaram o país referência mundial no setor (Embraer, 2023).

O avanço tecnológico ao longo do século XX também se refletiu na aviação. O período pós segunda guerra trouxe avanços significativos para vários setores, pois as tecnologias antes restritas ao meio militar começaram a se difundir, e não foi diferente com a aviação. Sistemas de navegação e controle por meios eletrônicos, instrumentação digital e simuladores se espalharam além do âmbito militar, e atingiram a aviação civil, contribuindo para seu desenvolvimento. O uso de simuladores para treinamento de pilotos ganhou força, e para além disso, foi fundamental para a simulação de diferentes cenários de voo, possibilitando o desenvolvimento do voo por instrumentos conhecidos como IFR (*Instrument Flight Rules*). Esses simuladores desempenharam um papel essencial no aperfeiçoamento de sistemas eletrônicos embarcados, e também na validação de novas tecnologias aeronáuticas. Com o avanço da aviação e da eletrônica, esses equipamentos se tornaram cada vez mais fiéis, atribuindo movimento, computação, e possibilitando a

criação de cenários com a interferência humana (Matsuura, 1995). A fim de criar uma melhor visualização dos acontecimentos, a Figura 2 demonstra uma linha do tempo que conecta a aviação, a evolução da eletrônica e os simuladores de voo.

Figura 2 – Linha do tempo na aviação.



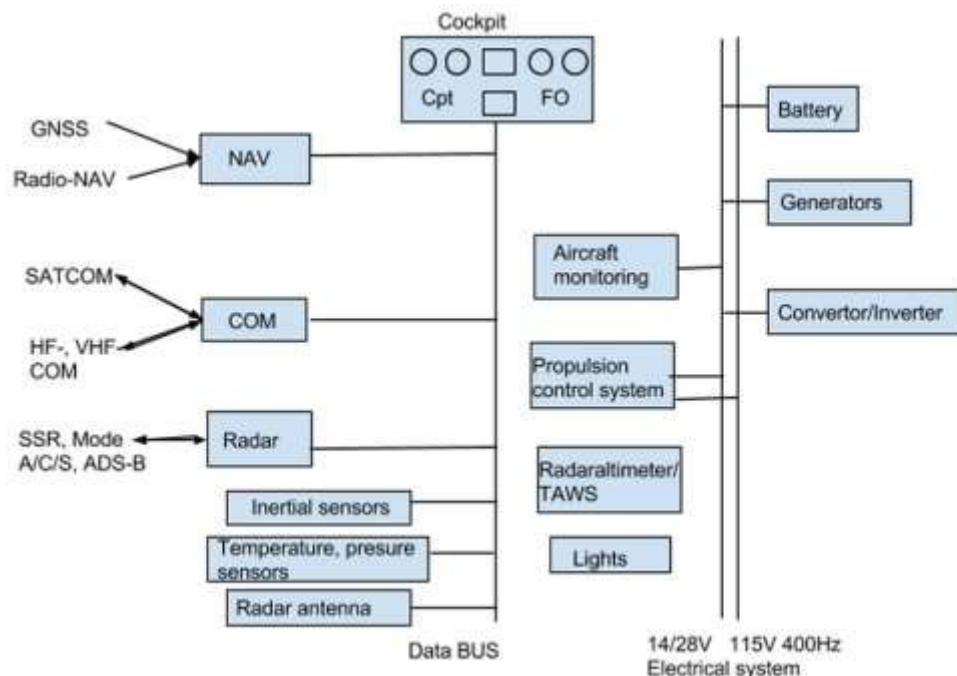
Fonte: O autor (2025).

2.1.1 Surgimento da Eletrônica Embarcada

A partir da década de 1950, com a miniaturização de componentes a partir da invenção do transistor, a aviação entrou na era da eletrônica embarcada. Até então, o controle das aeronaves era predominantemente mecânico e analógico. Os primeiros circuitos eletrônicos passaram a ser empregados em sistemas de navegação, controle automático e comunicação aérea, permitindo precisão e segurança cada vez maiores. Segundo Barr (1999), os sistemas embarcados são caracterizados por executarem tarefas específicas de forma contínua, utilizando microcontroladores ou processadores dedicados de propósito restrito. Na aviação, esses sistemas assumem papel crítico, pois controlam funções vitais da aeronave, exigindo confiabilidade e redundância elevadas. Os sistemas embarcados comunicam-se entre si através de barramentos e compartilham dados que permitem a leitura e interpretação em diferentes componentes, e, na Figura 3, é demonstrado

um sistema embarcado de aeronave, onde os sistemas se comunicam entre si, compartilhando dados e alimentados pelo sistema elétrico da aeronave.

Figura 3 – Exemplo de sistemas embarcados de uma aeronave.



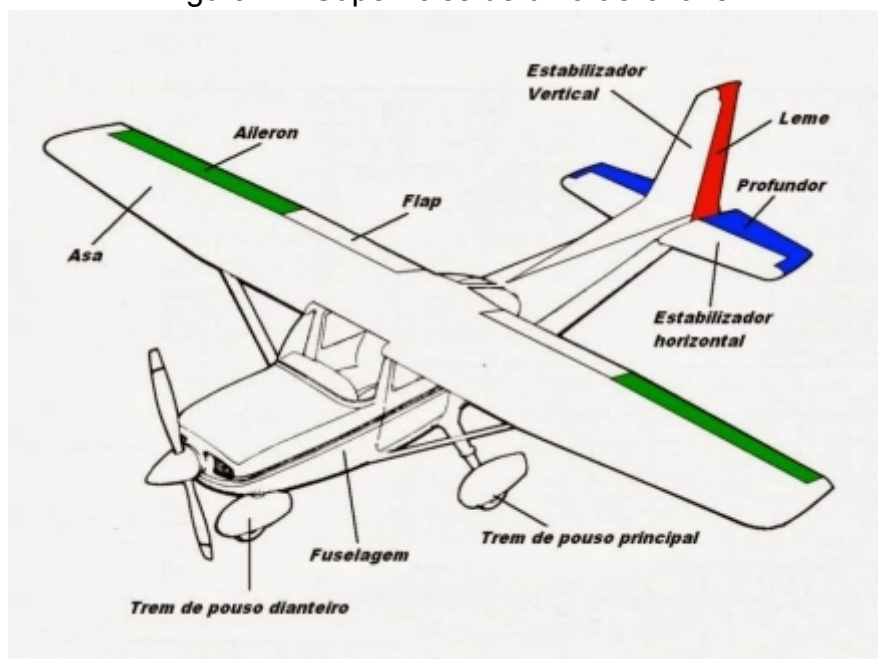
Fonte: BALMUS (2015, p. 165).

Durante a primeira e segunda guerra mundial, a aeronáutica sofreu uma intensa aceleração tecnológica. As aeronaves militares da época passaram a incorporar sistemas de rádio comunicação, sistema de navegação por rádio, piloto automático e, no caso da Segunda Guerra Mundial, passaram a utilizar radares embarcados, sendo todos esses elementos restritos a aeronaves militares (Kayton, 2003; Moir; Seabridge, 2006). A necessidade operacional impôs que a aviação adotasse soluções eletrônicas confiáveis em ambientes extremos, dando origem aos primeiros sistemas embarcados de relevância. Mais tarde, essas tecnologias migraram para a aviação comercial, impulsionando o conceito contemporâneo de aviônicos e preparando o cenário para a integração de sistemas mais sofisticados de controle e gerenciamento de voo.

Em 1988, quando o consórcio europeu batizado de Airbus, formado por França, Alemanha, Inglaterra e Espanha, entregou o primeiro A320 para a companhia francesa Air France, a eletrônica embarcada na aeronave já mostrava uma grande evolução frente aos seus concorrentes, principalmente contra a americana Boeing. O A320, além de painéis e sistemas eletrônicos precisos, mudou

o rumo do desenvolvimento de aeronaves civis. Ele era, e é até hoje, equipado com o sistema chamado de *Fly-By-Wire* (FBW), da tradução direta de “Voar Por Fios”, por literalmente ter o controle mecânico de suas superfícies feitas por um sistema eletrônico controlado por robustos computadores. O conjunto é projetado para realizar o controle das superfícies primárias como os *ailerons* e os profundos demonstrados na Figura 4, permitindo o controle da aeronave no ar. Neste caso, são vários computadores que controlam tais superfícies e criam uma redundância que impede que o sistema falhe e cause problemas em pleno voo. O sistema foi bem projetado, e adotado posteriormente por outras fabricantes, como Boeing e Embraer, consolidando a ideia do uso dos sistemas eletrônicos a bordo de aeronaves.

Figura 4 – Superfícies de uma aeronave.

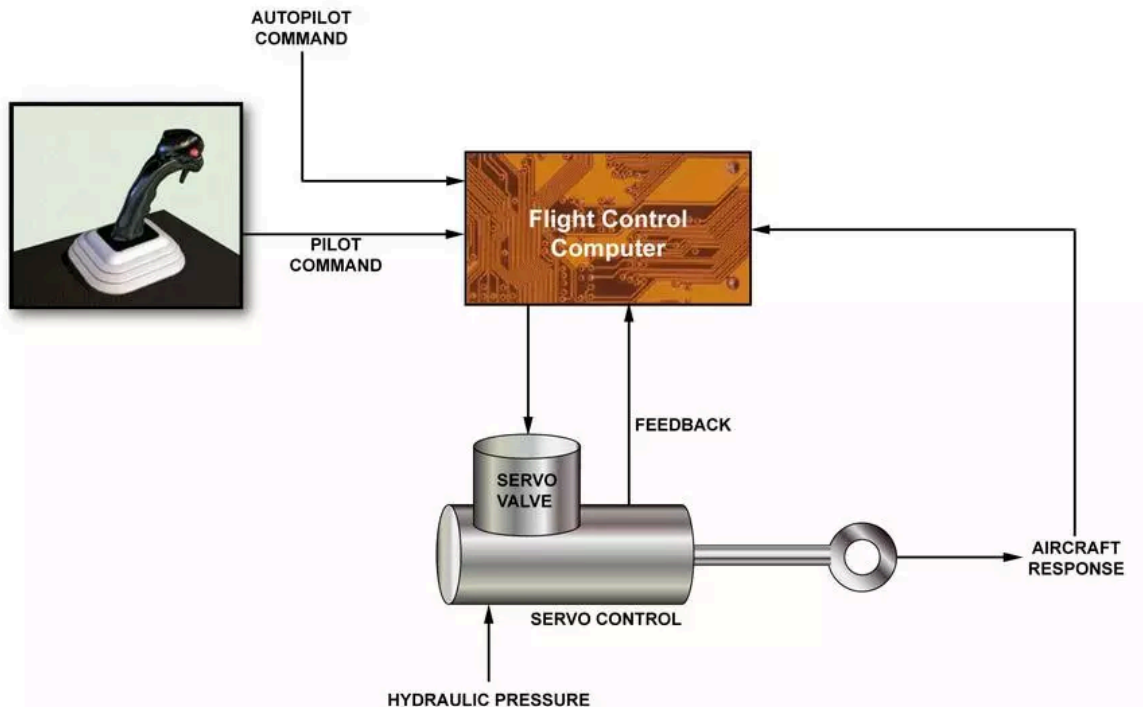


Fonte: Guia do Aviador (2017).

Porém, o sistema *fly-by-wire*, demonstrado na Figura 5, não foi inventado pela Airbus. Inicialmente, esse sistema de controle de voo eletrônico foi implementado em aviões militares na década de 1970, com destaque para o NASA F-8 *Digital Fly-By-Wire*, que em 1972 realizou o primeiro voo totalmente dependente de um sistema eletrônico de controle de voo, com uma aeronave F-8 Crusader modificada (NASA, 2021). Posteriormente, o sistema foi reaproveitado pela fabricante de aviões europeia e implantado nas suas aeronaves, reforçando o fato de que as tecnologias eram utilizadas estritamente em operações militares, mas que aos poucos foram

aproveitadas e implantadas na aviação civil, marcando dessa forma a adoção de sistemas digitais na linha aérea.

Figura 5 – Sistema *Fly-By-Wire*.



Fonte: Airbus (2022).

A consolidação do FBW digital na aviação civil redefiniu a interação homem-máquina e a filosofia de proteção de envelope, garantindo segurança de voo e trazendo maior confiabilidade aos sistemas. Portanto, o desenvolvimento de sistemas eletrônicos e aviônicos influenciou tanto o desenho de sistemas embarcados quanto a simulação de voo.

2.1.2 Conceito de Aviônicos

O termo aviônicos, que vem do inglês *avionics*, junção de *aviation* e *electronics*, se refere ao conjunto de sistemas eletrônicos embarcados responsáveis por navegação, comunicação, controle e gerenciamento do voo (Skybrary, 2023). Esses sistemas são integrados por meio de barramentos digitais, que conectam sensores, atuadores e módulos de processamento, formando uma rede embarcada capaz de reagir em tempo real às condições de voo. Em outras palavras, é como um

sistema nervoso e sensorial de uma aeronave, que coleta informações, processa dados e comanda atuadores e mecanismos essenciais.

No início da aviação, o *cockpit* das aeronaves era composto apenas por instrumentos analógicos, como altímetro, velocímetro, horizonte artificial, bússola, e alguns manômetros com indicações dos motores, podendo ser visualizados na Figura 6. Cada sistema operava de forma independente, e o piloto precisava interpretar as informações e tomar atitudes baseadas nas informações obtidas no painel, para manter um voo seguro. Esse tipo de conjunto ficou conhecido como *cockpit* analógico ou *steam gauge cockpits* (Moir; Seabridge, 2006).

Figura 6 – Cockpit analógico Embraer T-27 Tucano.



Fonte: Tecnodefesa (2018).

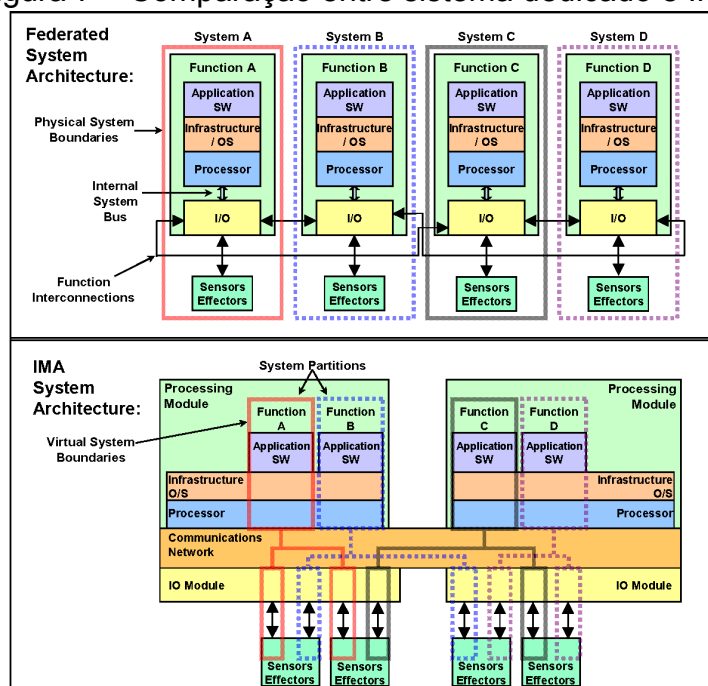
A partir da segunda metade do século XX, a complexidade operacional das aeronaves e a necessidade de precisão e segurança conduziram à digitalização e integração desses sistemas, dando início aos aviônicos.

Os aviônicos modernos compõem um sistema distribuído, em que sensores, processadores, atuadores e interfaces se comunicam por meio de barramentos

digitais determinísticos, os quais garantem e permitem prever o tempo de entrega das informações, como o *Controller Area Network* (CAN) e MIL-STD-1553B (ITA, 2012). Essa rede embarcada permite o funcionamento integrado de diversos subsistemas, como controle de voo, navegação, motores, pressurização, gerenciamento elétrico, dentre diversos outros existentes, por meio de computadores de bordo conectados entre si.

O resultado desse sistema é uma aeronave mais eficiente, com capacidade de autodiagnóstico, compensação de falhas e otimização de desempenho em tempo real (Collinson, 2014). Em aeronaves comerciais e executivas, essa integração cria uma arquitetura conhecida como *Integrated Modular Avionics* (IMA), na qual múltiplos sistemas compartilham processadores e barramentos, reduzindo peso, complexidade e custos de manutenção (RTCA, 2005). Ao invés de cada módulo ter um computador dedicado, a IMA é capaz de distribuir várias aplicações críticas em plataformas computacionais comuns, cada uma executando tarefas independentes e isoladas logicamente. A Figura 7 apresenta uma comparação entre os dois tipos de sistemas, mostrando que o sistema IMA possui melhor funcionamento e vantagens operacionais.

Figura 7 – Comparação entre sistema dedicado e IMA.



Fonte: WATKINS; WALTER (2006).

Nas próximas subseções, serão listados os principais sistemas aviônicos modernos.

2.1.2.1 EFIS

O *Electronic Flight Instrument System* (EFIS) é uma substituição dos instrumentos analógicos presentes nos painéis das aeronaves por um conjunto de telas digitais. O sistema apresentado na Figura 8 demonstra o EFIS de uma aeronave Embraer E-195, onde os painéis digitais formam o conjunto de instrumentos do *cockpit* da aeronave. O conjunto é formado por três sistemas, sendo eles:

- *Primary Flight Display* (PFD): *Display* com informações sobre altitude e altímetro, velocidade e horizonte artificial;
- *Navigation Display* (ND): *Display* que exibe informações da rota programada, alertas de tráfego aéreo por meio de um sistema chamado de *Traffic Alert and Collision Avoidance System* (TCAS) e informações meteorológicas;
- *Engine Indicating and Crew Alert System* (EICAS): *exibe informações de monitoramento de motores, sistemas elétricos, hidráulicos, pneumáticos, freios, abertura de portas, entre outras informações.*

Figura 8 – EFIS Embraer E-Jets.



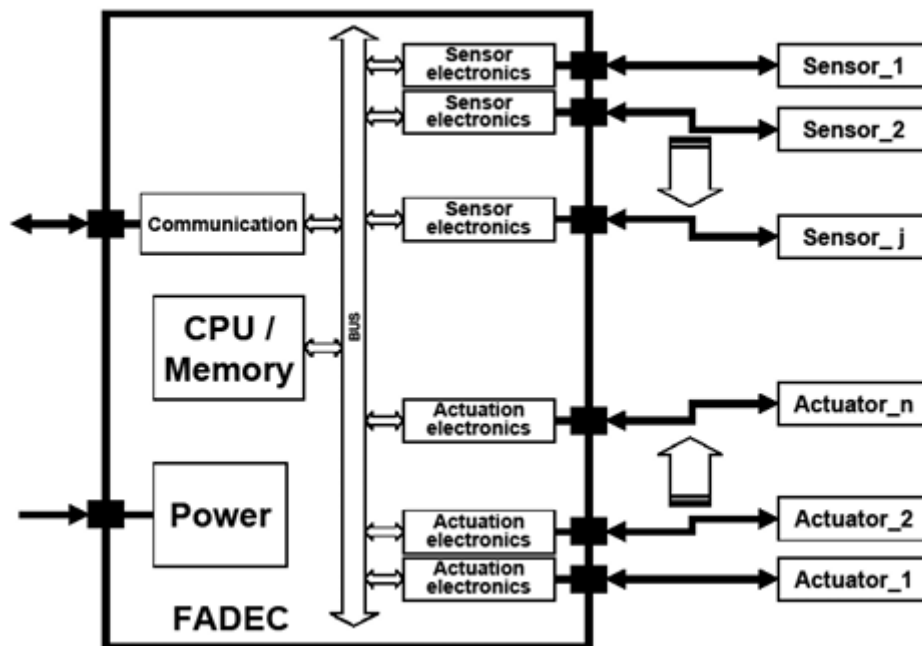
Fonte: Embraer (2023).

O EFIS pode ser considerado como uma grande evolução, devido ao fato de reduzir a carga de trabalho dos pilotos, ao concentrar as informações de maneira precisa e direta nos *displays*.

2.1.2.2 FADEC

Full Authority Digital Engine Control é a sigla de FADEC, um sistema digital responsável pelo controle de todos os parâmetros do motor de uma aeronave, permitindo a medição de temperatura, pressão, vibração, umidade, tensão e corrente (NASA, 2007). O modelo é apresentado na Figura 9, onde os sensores, atuadores e sistema de comunicação apresentados substituem os controles mecânicos e cabos, e se comunicam com outros módulos, como o FMC para o controle de parâmetros dos motores.

Figura 9 – Diagrama de um FADEC.



Fonte: NASA (2007).

2.1.2.3 FMS

O *Flight Management System* (FMS) é um sistema de gerenciamento de voo capaz de calcular e controlar rotas, realizar o controle de velocidade e altitude em pontos específicos da rota, ajustar frequências de rádio para operação, controlar e

definir parâmetros do motor para uma operação mais eficiente, dentre outros parâmetros, como definidos pela GE Aviation:

O *Flight Management System* (FMS) integra recursos avançados de navegação, como RNP/RNAV e GPS, permitindo gerenciamento automatizado do voo e cálculo preciso de trajetórias laterais e verticais. Ele realiza funções de planejamento e desempenho de voo, controle de rádios e navegação. O banco de dados de navegação inclui procedimentos de partida, rota, chegada e aproximação, assegurando operações seguras, eficientes e automatizadas. Esse sistema constitui o núcleo do gerenciamento de voo moderno e pode ser adaptado a aeronaves existentes para compatibilidade com os procedimentos de última geração (GE Aviation, 2022).

Este sistema é o cérebro por trás do dispositivo com o qual os pilotos interagem, e se comunica com diversos outros dispositivos e sensores na aeronave, sendo capaz de realizar um controle preciso e eficaz. O dispositivo é apresentado na Figura 10, e como demonstrado, é um computador robusto, encapsulado em uma case metálica.

Figura 10 – *Flight Management System* GE 2907C1.



Fonte: GE Aviation (2022).

O sistema do FMS é composto pelo computador central, o qual se comunica com o *Main Control Display Unit* (MCDU), dispositivo de interação entre os pilotos e a máquina. O sistema é composto também por sensores do sistema de navegação, como os conhecidos por IRS (*Inertial Reference System*) e GPS (*Global Positioning System*), e comunicadores de rádio com frequência como o VHF *Omnidirectional Range* (VOR) e o *Distance Measuring Equipment* (DME).

2.1.2.4 Glass Cockpit

A partir da utilização dos novos painéis digitais nas aeronaves, os antigos mostradores *steam gauges* deram lugar aos dispositivos que compõem o *Glass Cockpit*. O *cockpit* é o ambiente físico onde os sistemas aviônicos convergem e ocorre a interação entre os pilotos e os sistemas, porém, com a transição para o novo modelo composto por telas digitais e maior automação, o papel do piloto mudou, de um executor de tarefas para um gerenciador de sistemas.

Hoje, o *cockpit* é projetado sob os princípios da ergonomia e engenharia cognitiva, com uma ênfase na interface de comunicação entre o homem e a máquina, reunindo informações com maior organização, trazendo redundância a cada sistema e acessibilidade (FAA, 2013). Para seguir essa linha de raciocínio, os *cockpits* atuais utilizam, além dos já mencionados EFIS e EICAS, sistemas como o *Head-Up Display* (HUD), que projeta informações no campo de visão do piloto, o *Traffic Collision Avoidance System* (TCAS) e *Ground Proximity Warning System* (GPWS) que fornecem alertas automáticos sobre a proximidade da aeronave com outras aeronaves no espaço aéreo através do TCAS e também alertas de proximidade com o solo através do GPWS.

O sistema de *Flight Control Unit* (FCU) também integra o sistema, e por meio dele, é possível realizar comandos diretos na aeronave, atribuindo a ela informações diretas do FMS, e que serão controladas pelo piloto automático. Essas tecnologias garantem que, mesmo em situações de alta carga de trabalho, o piloto tenha acesso rápido e intuitivo às informações necessárias para manter o controle e a segurança de voo. Na Figura 11, é apresentado o mesmo *cockpit* da Figura 6 após passar por uma atualização. Na aeronave da Figura 11, o *cockpit* foi atualizado com telas digitais, fornecendo o sistema *Glass Cockpit* ao piloto.

Figura 11 – Cockpit Embraer T-27 Tucano atualizado com *Glass Cockpit*.



Fonte: Tecnodefesa (2018).

2.1.3 MCDU

O *Main Control Display Unit*, que pode ser chamado apenas de *Control Display Unit* (CDU), é o dispositivo que se conecta ao computador de gerenciamento de voo. Ele é a interface física com a qual os pilotos interagem, e, através dela, realizam controle e configuração de determinados parâmetros da aeronave. A Rockwell Collins, desenvolvedora de dispositivos eletrônicos em aeronaves, desenvolve o modelo Collins FMS-4200, e o descreve como:

O Sistema de Gerenciamento de Voo FMS-4200 inclui o Computador de Gerenciamento de Voo (FMC), a Unidade de Controle e Exibição (CDU) e uma Unidade de Banco de Dados (DBU). O sistema é usado para controlar

os displays EFIS, o FMS, o AFIS, a sintonia de rádio e outras funções da aeronave (Rockwell Collins, 1999).

Portanto, é possível diferenciar e distinguir os itens que compõem o conjunto de controle do computador de gerenciamento de voo. Na Figura 12, é possível visualizar o MCDU, com seu *display* e teclados, e conectados ao FMC, composto pela case atrás do MCDU.

Figura 12 – Conjunto do computador de gerenciamento de voo Honeywell.



Fonte: Duncan Aviation (2024).

O MCDU possui uma organização física característica, independente da fabricante que desenvolve o produto. Ele é composto por um display, o qual mostra as informações relevantes, os campos a serem preenchidos e os parâmetros que compõem cada etapa do voo, é uma tela de exibição, como qualquer outra, dividido em três partes: o campo do título da página atual no campo superior, no centro as informações, e no campo inferior o *scratchpad*, uma área onde é mostrada a inserção de dados do teclado. Nas laterais do display, estão os botões de seleção, chamados de *Line Select Keys* (LSK), que possuem a função de selecionar os campos e inserir dados. Um pouco abaixo da área do display, estão os botões de navegação, responsáveis por navegar pelos menus e sistemas de configuração disponíveis para determinado modelo de MCDU. E por fim, abaixo das teclas de navegação, está o teclado alfa numérico, composto por letras de A a Z, números de

0 a 9, e caracteres, como o símbolo de barra “/”, o ponto final, tecla de espaço, e botões de *delete* e *clear*, para apagar informações de determinados campos, e limpar os dados inseridos pelo teclado. Para melhor visualização, a Figura 13 destaca todos os itens comentados.

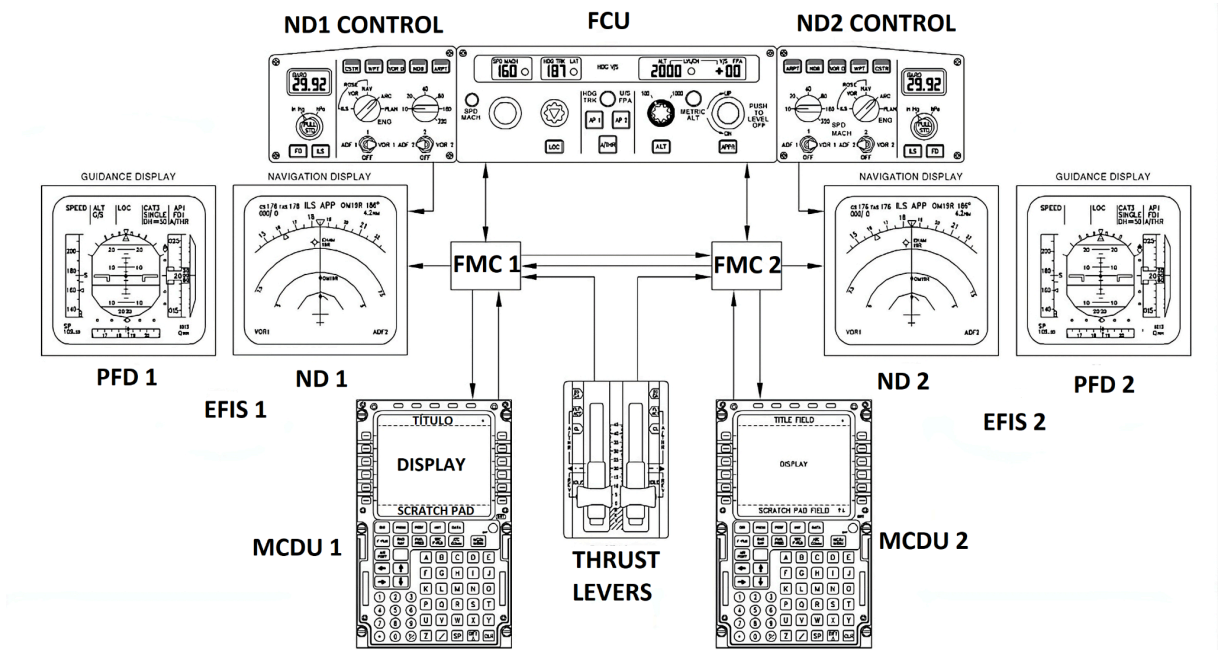
Figura 13 – Descrição de um MCDU.



Fonte: CPFlight (2025).

O MCDU, portanto, se comunica com o FMS por meio de barramento de dados. Os dados inseridos pelo piloto através do MCDU, são recebidos e manipulados pelo FMS, que retorna as informações de volta para os displays de exibição, como rotas, mensagens, e velocidades, por exemplo. A fim de consolidar o conceito dos dispositivos anteriormente mencionados, a Figura 14 demonstra a interação dos sistemas de uma aeronave da Airbus, e todos os dispositivos nela contidos, já foram anteriormente explicados.

Figura 14 – Conjunto de controle e navegação.



Fonte: Honeywell (2016).

O MCDU é uma das interfaces mais complexas e críticas de um *cockpit*, e a sua reprodução é essencial para a simulação de cenários e desenvolvimento de sistemas embarcados. Ele é, e todo seu conjunto, parte de um sistema essencial para o voo de uma aeronave comercial.

2.2 SIMULADORES DE VOO

Para que os pilotos possam se adequar aos sistemas, realizar treinamentos e para que possam ser realizados testes com as aeronaves sem que se coloque em risco a vida e os equipamentos, foram criados os simuladores de voo. Por volta de 1910, surgiram os primeiros simuladores de voo, compostos de um aeroplano fixo ao solo, conhecidos como *Sanders Teacher* e o invento de Eardley Billing. Esses modelos de simulador evoluíram para o que ficou conhecido como “Barril de Aprendizado de Antoniette”, apresentado na Figura 15. O simulador de Antoniette era composto por duas metades de tambores, e não estava mais fixo no solo (FAB, 2020).

Figura 15 – Simulador de Antoinette.



Fonte: FAB (2020).

De acordo com a evolução da aviação, os simuladores também evoluíram. Entre 1927 e 1929, Edwin Link desenvolveu seu simulador, o *Link Trainer*. Na Figura 16, é possível visualizar o simulador de Link, baseado em mecanismos pneumáticos, e possuía uma bomba de sucção de acionamento elétrico que era responsável por alimentar válvulas de controle operados pela vara do leme, porém, mesmo com toda a evolução, eles não se tornaram ainda uma substituição de um voo real. O *Link Trainer*, mesmo assim, foi muito importante e colaborou para o avanço da aviação. Devido a uma necessidade da *Army Air Corps*, Link realizou um voo teste, em condições de zero visibilidade, o qual serviu como teste para o contrato de 10.000 unidades vendidas para os EUA e os países aliados. Ele também provou que era possível realizar um voo inteiro por instrumentos, sem que se tivesse visão do solo (FAB, 2020).

Figura 16 – *Link Trainer*.

Fonte: FAB (2020).

Ao longo dos anos, os simuladores evoluíram cada vez mais, e com a melhoria nos circuitos eletrônicos e avanços computacionais, tornou-se possível simular diversos cenários, configurar diferentes parâmetros, reproduzir de maneira mais fiel o espaço aéreo e as condições de voo, além de capacitar melhor os profissionais que estariam nos céus.

2.2.1 Simuladores de Voo Para Capacitação

Para que os pilotos pudessem se capacitar e aprimorar técnicas de manobra e treinar cenários de emergência, os simuladores deveriam seguir padrões para garantir a qualidade e fidelidade dos modelos a serem simulados. Para isso, órgãos como a FAA, EASA e ANAC criaram requisitos que os simuladores deveriam seguir, para serem certificados a operar como método de treinamento e capacitação de pilotos.

2.2.1.1 *Flight Training Device*

Os simuladores conhecidos como *Flight Training Device*, ou FTD, são simuladores fixos, sem plataforma de movimento, projetados para treinamento

parcial de procedimentos e sistemas da aeronave. Apesar de não simularem as forças físicas de um voo, eles possuem painéis, controles e equipamentos idênticos às aeronaves reais, permitindo o treinamento de operação de sistemas elétricos, hidráulicos, de navegação, entre outros. De maneira resumida, o voo nesse tipo de simulador é predominantemente visual e operacional, não sensorial, e o piloto em treinamento não sente as acelerações, vibrações ou forças físicas do voo. Durante a realização deste trabalho, o autor realizou uma visita ao FTD da empresa DELTA 5 Simuladores, e a versão apresentada na Figura 17 corresponde ao simulador de voo de um Airbus A320.

Figura 17 – *Flight Training Device* DELTA 5 Simuladores.



Fonte: O autor (2025).

Para que seja possível compreender a importância de um equipamento FTD, é preciso entender que pilotar uma aeronave moderna abrange muito mais funções do que apenas controlar manche, pedais e manete de potência. O piloto de uma aeronave moderna interage com sistemas complexos, altamente tecnológicos, integrados entre si, e a operação, manuseio e interpretação correta dos sistemas são requisitos obrigatórios para segurança de voo e precisão operacional. Dessa forma, os FTDs permitem que o treinamento ocorra sem risco, seja humano ou material, sendo essa a base para que os simuladores de voo existam.

De acordo com o Regulamento Brasileiro da Aviação Civil (RBAC) 60, na página 4 (ANAC, 2020), é definido um FTD como “uma réplica dos instrumentos, equipamentos, painéis e controles de uma aeronave [...]. Inclui os equipamentos e programas necessários para representar a operação da aeronave em solo e em voo”. Para que possa ser determinado o nível de fidelidade de um simulador do tipo FTD, há uma classificação que varia em níveis progressivos entre FTD-4 a FTD-7, conforme as regulamentações da FAA, EASA e ANAC. Quanto maior a fidelidade dos sistemas presentes no simulador, complexidade dos sistemas e precisão da simulação, maior o nível.

Para os níveis mais básicos da classificação, são realizados procedimentos específicos mais simples, como por exemplo a inspeção pré-voo, procedimento de partida dos motores e operação de aviônicos, e, conforme o nível aumenta, mais complexa é a operação, como um voo com um motor inoperante. As classificações FTD-4 a FTD-7 estão presentes no apêndice B do RBAC 60 fornecido pela ANAC para regulamentação no Brasil, seguindo padrões internacionais, e contêm inúmeras páginas detalhando cada nível e função de acordo com a classificação do simulador.

Apesar da limitação dos simuladores FTD quanto ao movimento, eles possuem um papel essencial no treinamento moderno, permitindo a redução de custos, a segurança no treinamento de falhas e recuperação de situações, padronização de procedimento das tripulações, e a familiarização com os equipamentos das aeronaves. Dessa forma, o treinamento de pilotos em FTDs permite que o profissional desenvolva as habilidades necessárias para a operação segura do equipamento antes de enfrentar o ambiente real. Assim como apresentado na Figura 17, a Figura 18 representa um simulador de voo FTD, este desenvolvido pela CAE, tornando visível a estrutura em base fixa da cabine de simulação.

Figura 18 – *Flight Training Device* CAE 600XR.



Fonte: CAE (2025a).

2.2.1.2 *Full Flight Simulator*

Para uma experiência mais imersiva no treinamento dos pilotos, têm-se disponíveis os *Full Flight Simulators*, ou FFS, representando um nível de simulação mais avançado. Em contrapartida aos FTDs, que são dispositivos fixos e focados na operação de sistemas, os FFS reproduzem de maneira mais precisa e completa toda a experiência do voo, incluindo, além dos aspectos operacionais, o movimento e a física, tornando o treinamento mais sensorial.

Um FFS é composto por uma réplica de um *cockpit* da aeronave real, incluindo todos os comandos, painéis, e sistemas embarcados funcionando conforme a aeronave real atua. Essa cabine de comando é montada sob uma

plataforma de movimento, capaz de simular aceleração, vibrações e inclinações da aeronave a depender da fase do voo. Diferente do modelo apresentado nas Figuras 17 e 18, o simulador exibido na Figura 19 é o FFS desenvolvido pela CAE, apresentando seus mecanismos de movimento responsáveis por tornar a simulação neste tipo de simulador mais real.

Figura 19 – *Full Flight Simulator* CAE 7000XR.



Fonte: CAE (2025b).

A classificação dos simuladores de voo do tipo FFS é definida pela FAA, EASA e ANAC, dividida em níveis que variam de A a D, dependendo do nível de fidelidade apresentado pelo simulador. O nível D é a representação mais fiel do FFS, possuindo alta fidelidade aerodinâmica, ampla gama de teste de movimento e capacidade de representar vibrações semelhantes à aeronave (ANAC, 2020). Ainda de acordo com o RBAC 60, o simulador FFS é usado para a simulação de cenários considerados críticos, como recuperação de voo em baixa altitude e situações de *Stall*, onde há perda de sustentação da aeronave, com o ar se desprendendo da superfície da asa, devido a sua capacidade de movimento.

O custo de um FFS pode alcançar valores elevados, na casa dos milhões de dólares, devido à complexidade apresentada pelos sistemas. De acordo com a Adrenaline, o custo de um simulador CAE 7000XR, utilizado pela Azul Linhas Aéreas no treinamento de seus pilotos de Airbus A320Neo, é de 12 a 15 milhões de dólares. Apesar do alto investimento, os FFS desempenham um papel essencial no treinamento de pilotos, principalmente referente à segurança de voo, a partir do treinamento preciso de determinadas situações de emergência (Adrenaline, 2021).

Assim, enquanto os simuladores FTD desempenham seu papel no treinamento de procedimentos e sistemas, os FFS ampliam a experiência sensorial ao incorporar os aspectos físicos ao seu funcionamento. Apesar do elevado nível de fidelidade e certificação, os custos de simuladores do tipo FTD e FFS dificultam o acesso a esses equipamentos, fazendo com que os simuladores de voo domésticos se tornem uma alternativa acessível para estudo de sistemas e experimentação no desenvolvimento de sistemas embarcados.

2.2.2 Simuladores de Voo de Uso Doméstico

Com o avanço computacional, os simuladores puderam se difundir para diferentes arquitetura de computadores, e tornar-se uma opção para quem gosta de aviação poder ter um contato maior com a área. Os números de venda do Microsoft Flight Simulator mostram a aderência do público aos simuladores. Lançado em 2006, a empresa norte-americana vendeu cerca de 1 milhão de cópias do *Flight Simulator X* (Levvvel, 2023). Já a versão lançada em 2020, conhecida como Microsoft Flight Simulator 2020, vendeu, segundo o chefe do produto na Microsoft Jorg Neumann, até junho de 2024, 15 milhões de unidades (Stormbirds, 2024). Ainda em 2020, quando lançado, o Flight Simulator 2020 alcançou um milhão de cópias vendidas e 26 milhões de voos, o equivalente ao número de voos do ano de 2018 (AeroIn, 2020). A Laminar Research, por sua vez, desenvolvedora do simulador conhecido como X-Plane, não divulga oficialmente seu número de vendas, mas, de acordo com sites de estatísticas de vendas de jogos, a versão X-Plane 11, lançado em 2017 vendeu um pouco mais de 600 mil cópias, enquanto a versão mais atualizada, conhecida como X-Plane 12 e lançado em 2022, vendeu aproximadamente 160 mil cópias (Gamalytic, 2025).

Por meio dos dados apresentados, é possível notar um aumento no uso dos simuladores domésticos, apesar dos números do X-Plane apresentarem uma queda. Alguns fatores contribuem para a queda dos números da produtora Laminar Research, como o lançamento seguido com pouco espaço de tempo entre as versões 11 e 12, e o mesmo ocorre com os simuladores da Microsoft versões 2020 e 2024, em que na versão 2020 foram vendidas mais de 15 milhões de cópias, mas seu sucessor lançado 4 anos depois vendeu menos de 400 mil cópias (PlayTracker, 2025). Outros aspectos, como a diferença gráfica existente entre as versões X-Plane 12 e Microsoft Flight Simulator 2020 e 2024 e os *softwares* adicionais para cada modelo de simulador, fazem com que os usuários optem pelo modelo desenvolvido pela Microsoft.

Porém, existem outras diferenças que fazem com que os usuários escolham um determinado modelo de simulador. O Microsoft Flight Simulator é um simulador com um modelo de voo baseado em um modelo de *Surface Element Model*, onde os modelos de superfície de voo são responsáveis pelas forças físicas atuantes sob as aeronaves durante uma simulação. A aeronave é dividida em diferentes superfícies aerodinâmicas, como asas, fuselagem e estabilizadores, e para cada superfície, são calculados os modelos matemáticos do voo, sendo capaz de simular o fluxo do ar em 3D ao redor da aeronave, independentemente da direção do vento. A física do voo é calculada por meio de equações diferenciais que calculam o próximo estado da aeronave com base no estado atual e na entrada de comandos do piloto, ou seja, a cada instante do voo o simulador calcula a posição da aeronave, a orientação ou atitude da aeronave, a velocidade linear e a velocidade angular. Além desses dados, o *software* também calcula o centro de massa e centro de gravidade da aeronave. O modelo de movimento é dividido em 6 graus de liberdade, sendo 3 translacionais chamados de avanço, lateral e vertical, e 3 rotacionais, chamados de *roll*, *pitch* e *yaw*, e a orientação é representada por ângulos de Euler (Microsoft, 2023).

O simulador da Microsoft ainda apresenta, quando comparado ao concorrente X-Plane, melhor qualidade gráfica, porém, apesar da complexidade dos cálculos para a física do voo, o X-Plane apresenta maior precisão quanto aos modelos de voo.

O modelo de voo apresentado pelo X-Plane é diferente do Microsoft Flight Simulator. O produto desenvolvido pela Laminar Research utiliza o modelo BET, *Blade Element Theory*, modelo o qual além de dividir a aeronave em diferentes

superfícies, divide cada superfície simulada em pequenos pedaços, ou seja, são criados vários pequenos segmentos de asa, de fuselagem, de estabilizador, e para cada pequeno segmento, são calculadas as forças aerodinâmicas, que depois são somadas e integradas para obter velocidade, aceleração e posição. Para o fluxo de ar, o simulador calcula velocidade do ar, ângulo de ataque e forças aerodinâmicas. Para o cálculo das forças, em cada elemento são calculados *lift*, *drag* e momentos de força, e essas forças são convertidas em aceleração, rotação e mudança de trajetória. A integração das forças acontece diversas vezes por frame, aumentando a precisão (Laminar Research, 2025b).

Para fins de comparação, um estudo feito na Rússia desenvolveu um modelo de aeronave do tipo UAV, não tripulada, no simulador X-Plane, com base em um modelo matemático. Após o desenvolvimento do modelo para o ambiente de simulação e o modelo físico, foram realizados testes comparativos entre os modelos. A conclusão foi de que a resposta dinâmica do modelo implementado no X-Plane é muito próxima do modelo matemático, e o erro entre os modelos foi de cerca de 2% após a estabilização. Portanto, o X-Plane pode ser usado como modelo para simulação de alta fidelidade, tanto para o teste de alguns modelos matemáticos e comportamentos físicos de um voo, como desenvolvimento de sistemas, por exemplo, um sistema de piloto automático (Figueiredo; Saotome, 2012).

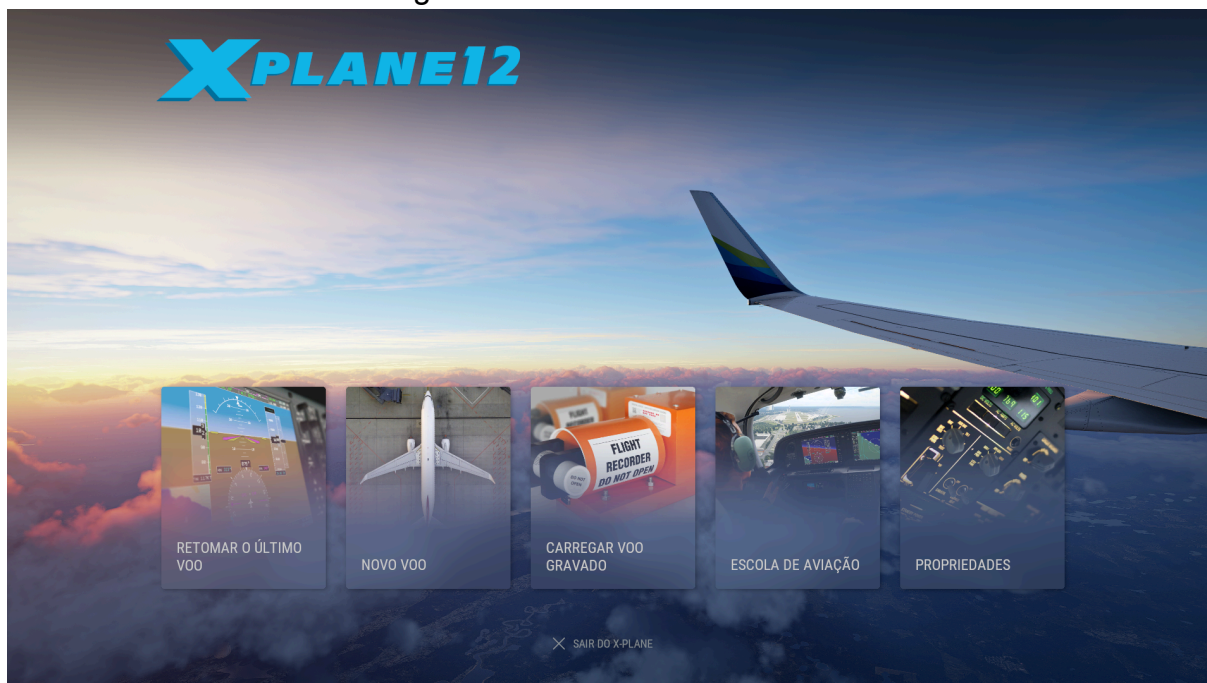
Embora os simuladores domésticos não sejam certificados como os FFS ou FTD, os princípios utilizados nos simuladores são parecidos, desde o cálculo dos modelos matemáticos, o motor gráfico e os modelos de sistemas. A Laminar Research oferece em seu site, as versões *home-use* e *professional*. Segundo a desenvolvedora, o X-Plane Pro “contém todas as características da versão de uso residencial do X-Plane 12 Global”, e ainda diferencia a versão *professional* da versão *home-use* “fornece as verificações de *hardware* e taxa de quadros necessários para a certificação FAA do simulador” (Laminar Research, 2025a).

2.2.2.1 Interface de Um Simulador de Uso Doméstico

No desenvolvimento deste trabalho, optou-se por utilizar o X-Plane 12 como ferramenta de simulação, devido à alta fidelidade do ambiente. A interface do simulador conta com um menu inicial, logo após iniciá-lo, com algumas opções, como iniciar um novo voo, e o menu de configurações, onde por exemplo, se

configuram periféricos, qualidade gráfica, e configuração de rede e dados de saída. O menu, apresentado na Figura 20, é simples, porém funcional.

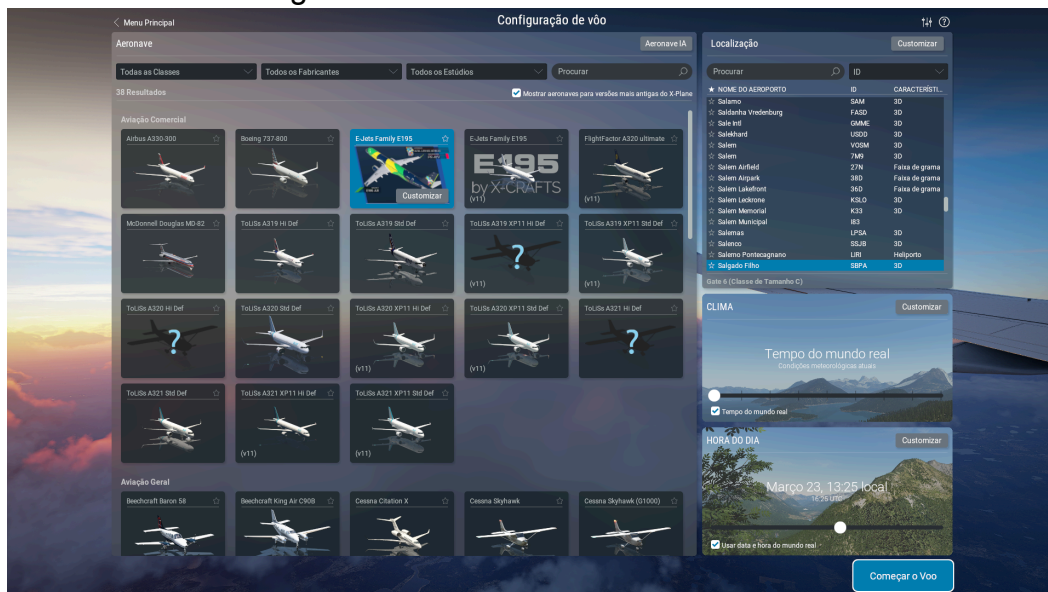
Figura 20 – Menu inicial X-Plane 12.



Fonte: O autor (2026).

Para que se possa iniciar um voo, é preciso definir alguns parâmetros, e podem ser feitos alguns ajustes. É necessário que o usuário escolha a aeronave a ser utilizada, e nesse momento algumas opções podem ser selecionadas, como o tipo de pintura da aeronave, e podem ser configuradas falhas, a fim de que o usuário possa simular e treinar situações de crise, a exemplo de falha nos motores, danos elétricos, ou de qualquer outra natureza. Peso, balanceamento e combustível também podem ser predefinidos neste momento, porém o comum é que se faça durante a simulação, para que se aumente o nível do realismo. Também é necessário que se escolha o aeroporto de origem, com a posição de estacionamento, ou *parking*, preestabelecida. Por fim, deve-se definir informações como data e hora, e o clima, podendo seguir os padrões locais atuais, ou informações personalizadas, em caso de necessidade de treinamentos específicos, como o de baixa visibilidade. Todas as configurações são feitas no menu exibido na Figura 21.

Figura 21 – Menu de novo voo X-Plane 12.



Fonte: O autor (2026).

Após definir os parâmetros, o voo é iniciado, carregando a aeronave na posição preestabelecida, devidamente configurada, e a partir deste momento, a simulação pode ser seguida, e pode ser encerrada a qualquer momento. Na Figura 22 é demonstrado o ambiente de simulação, onde está carregada a aeronave Embraer E-195. Na figura, é possível reconhecer os itens já mencionados na Figura 8, como o EFIS completo da aeronave, reforçando a precisão dos sistemas simulados das aeronaves.

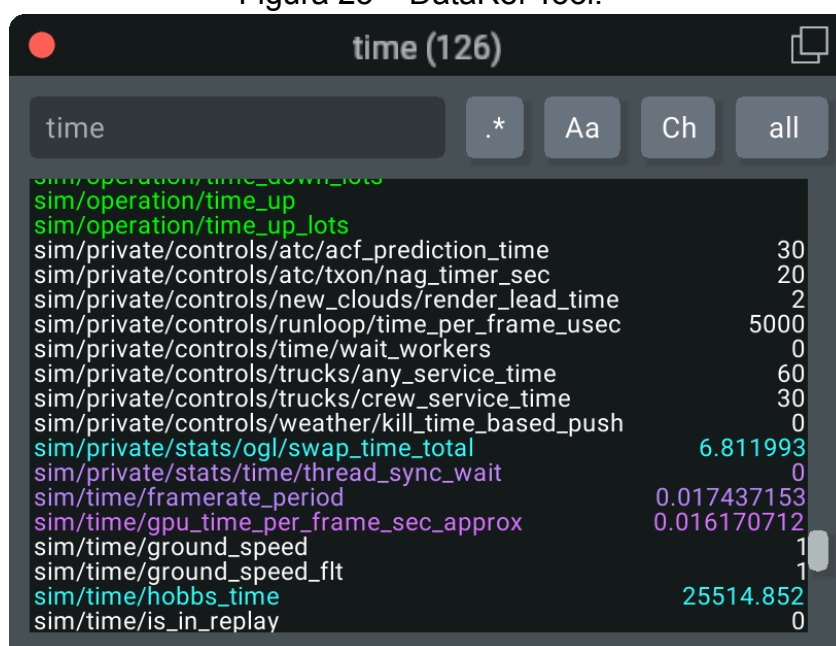
Figura 22 – Ambiente de simulação.



Fonte: O autor (2026).

A interface gráfica, portanto, é composta por estes elementos, porém, para o desenvolvimento de *plugins* e soluções para simuladores, é preciso observar as saídas de dados e como é possível alterá-las. Para viabilizar a visualização dos dados de uma simulação, é necessário utilizar uma ferramenta do simulador chamada de *DataRef Tool*. A ferramenta, exibida na Figura 23, é capaz de exibir os *datarefs*, que são as variáveis internas do simulador, correspondentes a diferentes parâmetros associados ao voo.

Figura 23 – DataRef Tool.



Fonte: DataRef Tool (2014).

Na figura acima, é possível visualizar os *datarefs* do X-Plane, associados a um voo. No lado esquerdo, são exibidos os parâmetros para cada sistema presente no voo, seja relacionado à aeronave, ao cenário, ao tempo ou ao clima, portanto, é através deles que se obtém acesso aos parâmetros. No lado direito da figura, cada valor associado à variável é um estado que a representa. Sendo assim, é através destas variáveis que é possível desenvolver soluções específicas para o X-Plane. É possível ler o estado atual ou valor de uma variável ou escrever nela. Para um display, por exemplo, é possível ler cada parâmetro separadamente, e posteriormente, organizar os dados em um display. Para botões, existe outra função, denominada de *commands*, que são funções com início e fim (DataRef Tool, 2014).

A partir dos *datarefs* e *commands*, torna-se possível desenvolver sistemas externos, capazes de interagir com o simulador em tempo real. Os dispositivos físicos podem ser utilizados como interfaces homem-máquina, e são capazes de enviar comandos ao simulador, e receber dados diversos. A partir desta capacidade, é possível a criação de painéis dedicados, como rádios, *autopilots* e unidades de MCDU, fazendo com que a experiência doméstica se aproxime do ambiente real do *cockpit*.

3 PRODUTOS RELACIONADOS

Atualmente, são encontradas diversas soluções de *hardware* para o mercado de simuladores de voo, destinado para o aumento da fidelidade da experiência do usuário. Dispositivos como manetes, *joysticks*, pedais, painéis de rádio e controles, voltados à reprodução de sistemas embarcados de aeronaves comerciais, estão disponíveis. De um modo geral, estes equipamentos têm como objetivo aprimorar a experiência do usuário com o simulador, reduzindo uso de *mouse* e teclado, permitindo maior imersão durante a simulação.

Grande parte das soluções comerciais existentes são desenvolvidas para aeronaves de maior presença no mercado de simuladores, especialmente modelos das fabricantes Airbus e Boeing. Isso ocorre porque aeronaves como Airbus A320 e Boeing 737 possuem ampla utilização em simuladores domésticos, grande quantidade de *add-ons* disponíveis e uma comunidade ativa de usuários, o que torna economicamente mais viável o desenvolvimento de periféricos dedicados a estes modelos.

Entre os dispositivos disponíveis, estão os MCDUs, que se destacam por reproduzirem fisicamente a interface utilizada pelos pilotos para interação com o sistema de gerenciamento de voo. Esses equipamentos possuem *display* e teclado próprios, conexões via cabo como HDMI e USB e compatibilidade com *softwares* específicos de simulação. No entanto, seu modelo físico, incluindo disposição de teclas e *display*, nomenclatura dos comandos e suporte de *software* são projetadas para aeronaves específicas, o que limita o uso em aeronaves diferentes.

No caso das aeronaves Embraer da linha E-Jets, não foram identificados até o momento, dispositivos comerciais que reproduzam de maneira fiel os diferentes equipamentos utilizados neste modelo de aeronave, como o MCDU. Por mais que existam os dispositivos para outros modelos, as adaptações para o modelo da Embraer são limitadas, tanto no aspecto físico quanto no funcional. Como exemplo, pode-se citar um modelo comercializado pela fabricante Winctrl, apresentado na Figura 24. O modelo é de um MCDU compatível com aeronaves Airbus, com aparência física e suporte compatíveis para simulação com este tipo de aeronave.

Figura 24 – MCDU Airbus Winctrl.



Fonte: WINCTRL (2026).

Este disponível encontra-se disponível em plataformas de importação, com valor aproximado de US\$150,00. Considerando a conversão cambial, os custos de frete e a incidência de tributos, como os custos de importação e o ICMS, o final aproximado para aquisição no Brasil atinge aproximadamente R\$2.158,60 com base nos dados de março de 2026.

Além de soluções comerciais disponíveis no mercado, existem também projetos desenvolvidos por entusiastas da simulação de voo, geralmente voltados para uso doméstico. Estes projetos utilizam plataformas como Arduino para comunicação com o computador, uso de Raspberry ou conexão via HDMI com o computador para reprodução das imagens, formando os dispositivos físicos integrados aos simuladores.

Grande parte das soluções é baseada em projetos independentes compartilhados em fóruns, repositórios públicos e comunidades de simulação, utilizando ferramentas como MobiFlight, um *software* amplamente utilizado para a construção de *cockpits* domésticos. Entretanto, muitos destes projetos possuem foco voltado para aeronaves como Airbus e Boeing, limitando a representação adequada dos sistemas presentes nas aeronaves Embraer. Além disso, a ausência de padronização entre os sistemas implementados em aeronaves e a dificuldade de acesso às interfaces internas dos simuladores tornam o desenvolvimento de

dispositivos dedicados uma tarefa complexa, exigindo integração entre diferentes ambientes e dispositivos.

Alguns trabalhos, como de Pedro Toth, disponível em seu *blog* (Toth, 2018), e de Shahada Abubakar, disponível no GitHub do usuário “Dotsha747” (DOTSHA747, 2017), são exemplos de desenvolvimento de MCDUs dos modelos de Airbus e Boeing, respectivamente. Ambos os projetos serviram de inspiração para o desenvolvimento de um MCDU destinado ao uso de aeronaves Embraer da linha E-Jets.

Sendo assim, é evidente que há uma lacuna no mercado de simuladores domésticos, no que se refere à disponibilidade de *hardware* dedicado para aeronaves Embraer, especialmente considerando a relevância da empresa brasileira no cenário mundial da aviação. Este fato limita a fidelidade da simulação e dificulta a reprodução da interface homem-máquina destes sistemas. Nesse contexto, torna-se relevante o desenvolvimento de uma solução dedicada, que seja capaz de reproduzir as características físicas e funcionais do MCDU utilizado nos E-Jets, permitindo uma maior fidelidade operacional e contribuindo para aplicações educacionais e experimentais no ambiente de simulação de voo.

4 DESENVOLVIMENTO

Neste capítulo, será apresentado o desenvolvimento do protótipo do dispositivo proposto, incluindo o entendimento do ambiente de simulação, os métodos de comunicação, bem como a arquitetura de *hardware* e *software* a serem implementados. A partir dos conceitos apresentados na fundamentação teórica, busca-se demonstrar na prática a integração entre o simulador de voo e o dispositivo físico de um MCDU, com funcionamento completo, e com fidelidade para aeronaves Embraer E-195.

Os aviônicos presentes nas aeronaves Embraer E-195 séries E1 e E2 são formados por dispositivos Honeywell da linha Primus Epic, sendo o MCDU um dos dispositivos incluídos no pacote do *flight deck*, presente na Figura 25. O sistema NGFMS (*Next Generation Flight Management System*) representa uma evolução dos sistemas modernos de gerenciamento de voo integrados a aeronaves comerciais. Este sistema é responsável pelo gerenciamento das principais funções relacionadas à navegação, planejamento de voo, gerenciamento de performance e integração de sensores da aeronave como GPS, IRS e VOR/DME. Ainda, conforme o manual oficial do sistema, o NGFMS realiza funções de navegação lateral (LNAV) e navegação vertical (VNAV), e utiliza uma base de dados, conhecida como AIRAC (*Aeronautical Information Regulation And Control*), com informações a respeito de aeroportos, frequências de rádio e navegação (Honeywell, 2015).

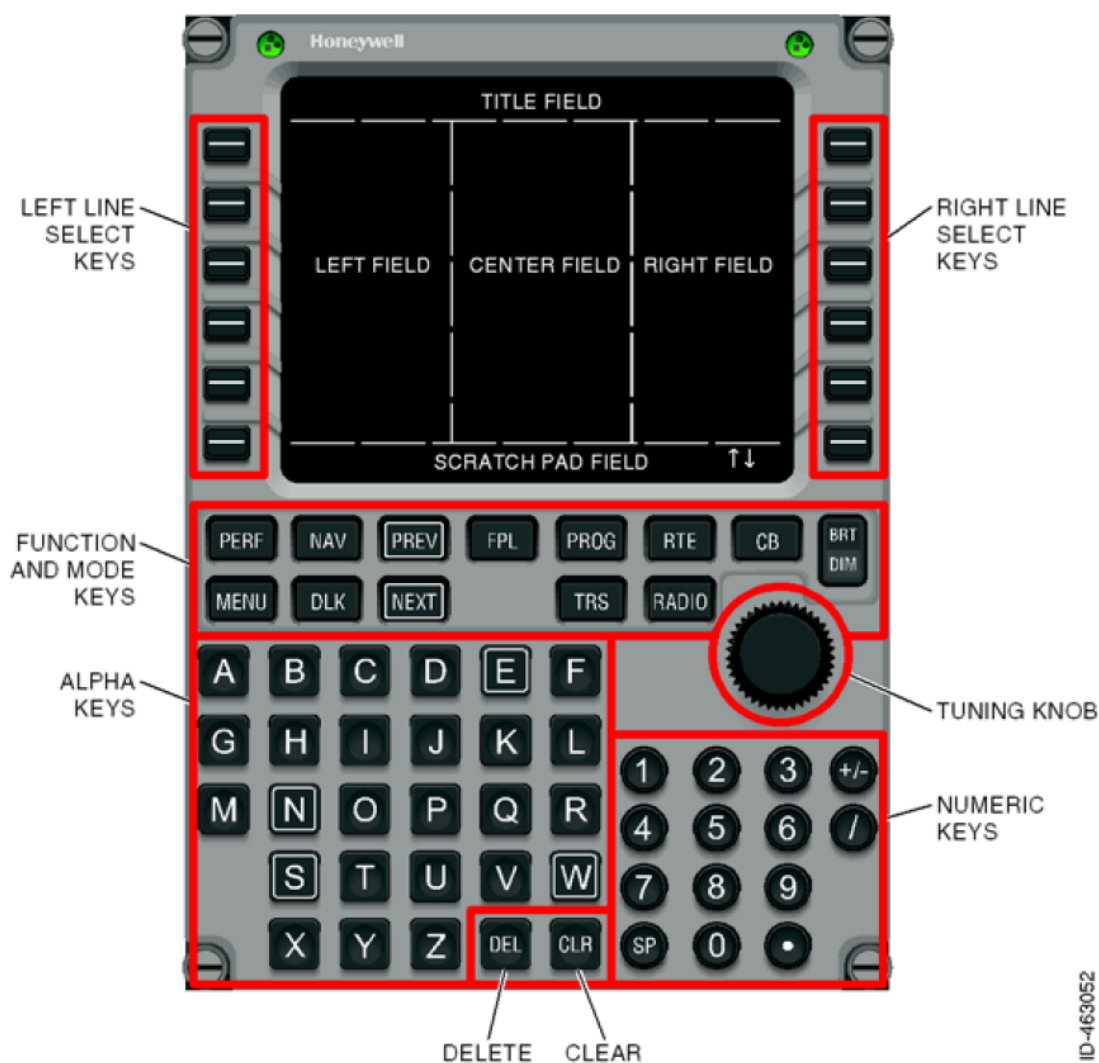
Figura 25 – *Flight Deck Epic Load*.



Fonte: Honeywell (2023).

O MCDU, por sua vez, é equipado com um teclado composto por 68 teclas, incluindo teclas de seleção dispostas ao lado do *display* conhecidas como LSK, teclas de navegação, numéricas e alfabéticas, entre outras teclas como de *delete* e *clear*, e ajuste de brilho do *display*. Através do teclado a tripulação pode interagir com o sistema, inserindo dados e gerenciando informações. O *display* equipado no MCDU é dividido em partes, disposto em 14 linhas, e cada uma das linhas contém 24 caracteres. A linha 1 é utilizada para exibição do título da página, e a linha 14 é para o *scratch pad*, onde os dados inseridos no teclado aparecem. Além disso, a área de exibição é dividida em lado esquerdo e direito e a área central. O dispositivo também conta com um *encoder* para os ajustes de frequência de rádio. A estrutura descrita do MCDU pode ser visualizada na Figura 26.

Figura 26 – MCDU Load 27 Honeywell.



O manual para pilotos fornecido pela Honeywell também contém um guia com um exemplo completo de um voo, e, em posse das imagens fornecidas por ele, foi possível comparar com as funções disponibilizadas pelo MCDU presente na aeronave desenvolvida pela X-Crafts. As funções e páginas disponíveis são fornecidas pela aeronave desenvolvida pela X-Crafts, e não desenvolvidas para uso no protótipo do MCDU, visto que as informações foram extraídas diretamente do simulador.

Com o entendimento do dispositivo e em posse das informações relevantes, tornou-se possível desenvolver um protótipo fiel ao dispositivo real, desenvolvido pela Honeywell. Nos próximos capítulos, serão demonstradas as etapas e processos que envolveram o desenvolvimento do protótipo do MCDU.

4.1 PROJETO DO MCDU

Para iniciar o trabalho, foi realizada a análise de funcionamento do MCDU presente no X-Plane e início do desenvolvimento do *software*. A aeronave em questão é um Embraer E-195 E1, e foi desenvolvido pela empresa X-Crafts para o simulador X-Plane 12. Essa análise teve como objetivo a observação das funcionalidades disponíveis, a organização das páginas e layout de interface. Embora o estudo tenha sido conduzido no ambiente de simulação, buscou-se manter a fidelidade ao sistema real, por meio da comparação de materiais disponíveis como vídeos, fotos ou manuais da linha E-Jets. A partir dessa análise, foram identificadas as páginas, o fluxo do uso e a lógica de interação com o dispositivo.

4.1.1 Obtenção dos Dados do Simulador

Para que se tornasse possível o acesso aos dados do dispositivo, foi utilizado um *plugin* para X-Plane, chamado de DataRef Tool, cuja funcionalidade é mostrar, durante uma simulação, todos os *datarefs* e *commands* que estão sendo utilizados. Esses parâmetros podem ser padrão do X-Plane, para serviços do próprio simulador, aeronaves desenvolvidas pelo próprio X-Plane e que acompanham o simulador por padrão, ou podem ser criados pelos desenvolvedores de *plugins* ou aeronaves, como pela X-Crafts. Para o Embraer E-195 utilizado, a X-Crafts possui

seus próprios *datarefs* e *commands*, e para que seja possível ter acesso a esses dados, utilizou-se a ferramenta descrita anteriormente, o DataRef Tool. Na Figura 27, estão presentes os *datarefs* obtidos através do DataRef Tool. Na coluna da esquerda, estão os nomes dos parâmetros e na coluna da direita, seu valor ou informação atual.

Figura 27 – *Datarefs* do E-195 desenvolvido pela X-Crafts.

XCrafts/FMS/CDU 1 01	010121
XCrafts/FMS/CDU 1 02	011010RADIO
XCrafts/FMS/CDU 1 03	0122201/2
XCrafts/FMS/CDU 1 04	020120 COM1 COM2
XCrafts/FMS/CDU 1 05	030110@ @
XCrafts/FMS/CDU 1 06	
XCrafts/FMS/CDU 1 07	030212122.800 127.100
XCrafts/FMS/CDU 1 08	040220
XCrafts/FMS/CDU 1 09	050210126.850 121.325
XCrafts/FMS/CDU 1 10	060120 NAV1 NAV2
XCrafts/FMS/CDU 1 11	060922FMS FMS
XCrafts/FMS/CDU 1 12	070922AUTOAUTO
XCrafts/FMS/CDU 1 13	070110@ @
XCrafts/FMS/CDU 1 14	070212110.30 110.30
XCrafts/FMS/CDU 1 15	090210113.80 117.35
XCrafts/FMS/CDU 1 16	100922
XCrafts/FMS/CDU 1 17	101822
XCrafts/FMS/CDU 1 18	101920XPDR1
XCrafts/FMS/CDU 1 19	110118\$TCAS/XPDR
XCrafts/FMS/CDU 1 20	1120121200
XCrafts/FMS/CDU 1 21	122022
XCrafts/FMS/CDU 1 22	130212STBY
XCrafts/FMS/CDU 1 23	130720TA/RA
XCrafts/FMS/CDU 1 24	130110b
XCrafts/FMS/CDU 1 25	131910IDENT\
XCrafts/FMS/CDU 1 26	050110
XCrafts/FMS/CDU 1 27	
XCrafts/FMS/CDU 1 28	
XCrafts/FMS/CDU 1 29	
XCrafts/FMS/CDU 1 30	
XCrafts/FMS/CDU 1 31	
XCrafts/FMS/CDU 1 32	
XCrafts/FMS/CDU 1 33	
XCrafts/FMS/CDU 1 34	
XCrafts/FMS/CDU 1 35	
XCrafts/FMS/CDU 1 36	
XCrafts/FMS/CDU 1 37	
XCrafts/FMS/CDU 1 38	
XCrafts/FMS/CDU 1 39	
XCrafts/FMS/CDU 1 40	

Fonte: O autor (2026).

Os *datarefs* do *display* da aeronave escolhida, apresentados na Figura 27, se referem ao estado atual do *display* do MCDU, que nesse caso se encontra na página de rádio. A fim de que fique claro como se relacionam os dados e o tratamento do conteúdo para exibição no *display* do dispositivo, a Figura 28 traz a visualização da tela do aparelho no momento da captação dos *datarefs*.

É possível notar um padrão entre os dados apresentados e o conteúdo exibido no MCDU, e que a empresa desenvolvedora da aeronave optou por uma codificação, uma espécie de modelo organizacional onde o código que antecede a informação é capaz de organizá-la corretamente para exibição no *display*. Portanto, sabendo que os *datarefs* obtidos são referentes ao MCDU 1, que se refere ao lado

esquerdo da aeronave, o lado do comandante, é necessário exportar os dados via rede para que possam ser exibidos em um *display* externo, em um dispositivo fora do simulador.

Figura 28 – Exibição do MCDU no momento da captação dos dados.



Fonte: O autor (2026).

Porém, antes de definir como exportar esses dados, deve-se pensar à frente e definir como será feita a exibição dos dados. O conjunto de *hardware* deve ser analisado já nesse momento, para que toda e qualquer linha de código escrita seja compatível com o modelo escolhido. Dentre os diferentes métodos possíveis, optou-se por escolher o compartilhamento da informação via *web*, ou seja, os dados serão extraídos do simulador, e serão enviados para a rede local, para que um outro dispositivo possa acessá-lo. A escolha desse método se justifica por dois motivos, o primeiro deles é a capacidade de se usar um Raspberry Pi 4 para ser o cérebro do dispositivo, fazendo com que o acesso seja feito via rede WI-FI, minimizando a utilização de cabos externos ao dispositivo, sendo necessária apenas a fonte, e o segundo motivo é o desenvolvimento *web* que se torna uma opção viável, tanto para receber dados, quanto para enviá-los. Define-se então o uso de um Raspberry Pi 4 Model B para ser o *hardware* do projeto, e a comunicação via rede para troca de informações em tempo real.

A comunicação via rede com o X-Plane realizou-se através de um *plugin* chamado de ExtPlane. Por mais que o X-Plane 12 tenha uma função nativa de comunicação via rede, os dados que podem ser acessados são somente alguns específicos do X-Plane, não se estendendo ao compartilhamento de *datarefs* específicos, portanto, o ExtPlane surge como opção, para que o acesso via rede seja possível.

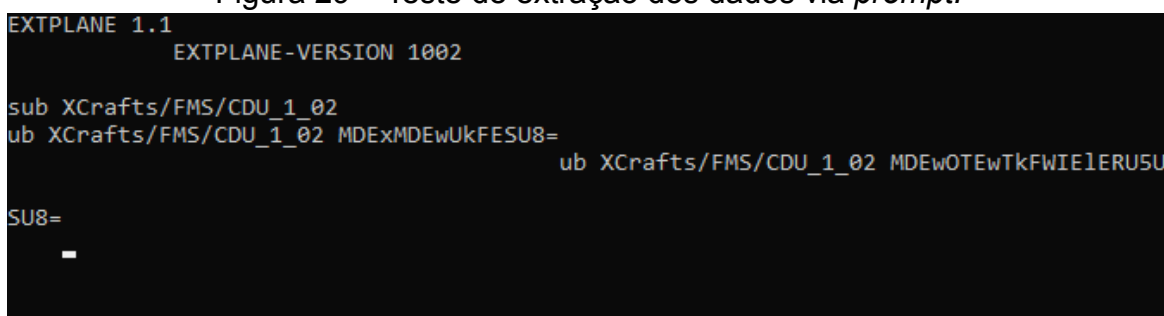
De acordo com seu criador, o usuário “vranki” no GitHub, o ExtPlane é descrito como “um plugin para X-Plane e outros simuladores que permite comandar a simulação a partir de programas externos através de um protocolo TCP fácil de usar” (Vranki, 2018). O *plugin* existe desde o X-Plane 9, e é amplamente utilizado para compartilhamento de dados do simulador via rede. Ele utiliza o método de comunicação via protocolo TCP, e dentre as suas características estão o múltiplo compartilhamento de dados, a possibilidade de acessar e alterar informações de *datarefs*, executar *commands*, obter variáveis de diferentes tipos, entre outras características. Ao instalar o *plugin* no X-Plane, ele funciona como um servidor TCP, lendo as informações do simulador e deixando disponível o acesso via rede.

Antes de iniciar a escrita de qualquer código, utilizou-se o teste da documentação padrão do ExtPlane. Abrindo o *prompt* do Windows, executou-se o seguinte comando “*telnet localhost 51000*”. Esse comando permite o acesso ao simulador via protocolo TCP, na rede local, na porta 51000. A documentação sugere obter informações do simulador e fornece uma lista de comandos utilizados para se comunicar com os *datarefs*, que quando utilizados, enviam os comandos como *strings* para o simulador, que através do *plugin*, retorna com os dados solicitados. O comando a ser testado é o *sub*, responsável por se inscrever em uma determinada variável do simulador, e, enquanto escrito, torna-se capaz de receber as informações da variável e atualizações dela, sempre que ela se altera.

Deste modo, utilizou-se do teste do comando *sub* para se obter as primeiras informações do simulador via rede. Foi utilizado o comando “*sub XCrafts/FMS/CDU_1_02*” para obter a informação da variável presente em XCrafts/FMS/CDU_1_02. Através do DataRef Tool, sabia-se que a informação a ser extraída seria “011010RADIO”, porém, a informação recebida no *prompt* foi “MDExMDEwUkFESU8”. O formato apresentado é conhecido como Base64, utilizado para codificação de dados. O Base64 é definido pela RFC 4648 como um esquema de codificação para representar dados binários em texto, em uma

sequência de 64 caracteres ASCII (*American Standard Code For Information Interchange*), permitindo que os dados sejam transmitidos com segurança em canais baseados em texto (IETF, 2006). Para a interpretação dos dados, foi necessária a decodificação para *string*, possibilitando assim a análise do conteúdo. Após a decodificação, o resultado demonstrado foi “011010RADIO”, confirmando que a informação recebida do simulador, via o ExtPlane, estava correta. A fim de facilitar a visualização, a Figura 29 contém uma exibição do CMD do Windows após o teste para obtenção de dados do simulador, com os dados em Base64. Para confirmar que a comunicação realizasse atualizações automáticas, alterou-se a página do MCDU no simulador, a fim de testar a inscrição no *dataref* solicitado, e quando alterada a página no simulador, alterando consequentemente o *dataref*, a informação no *prompt* também foi alterada, o que garantiu a comunicação via rede local entre o simulador e um meio externo e confirmando suas atualizações de informação.

Figura 29 – Teste de extração dos dados via *prompt*.



```

EXTPLANE 1.1
EXTPLANE-VERSION 1002

sub XCrafts/FMS/CDU_1_02
ub XCrafts/FMS/CDU_1_02 MDEwOTUwTkFWIE1ERUSU8=
ub XCrafts/FMS/CDU_1_02 MDEwOTUwTkFWIE1ERUSU8=
SU8=
-
  
```

Fonte: O autor (2026).

4.1.2 Tratamento dos Dados

Confirmada a comunicação e o funcionamento do ExtPlane, iniciou-se a escrita do código, para que, através de uma página *web*, o *display* pudesse ser exibido, e posteriormente, comandos fossem enviados. O primeiro código escrito foi o *TcpClient.js*, sendo ele o responsável, no início, por abrir a comunicação com o simulador, não sendo mais necessário o uso do comando via *prompt*. Portanto, o *TcpClient.js* tornou-se o comunicador com a porta 51000 aberta pelo ExtPlane, e ele, além de se comunicar com o simulador, envia comandos de inscrição para obter os dados do simulador. Com a confirmação de recebimento dos dados brutos, via *TcpClient.js*, iniciou-se a fase de interpretação dos dados. Antes mesmo de poder

tratá-los e enviá-los para uma interface web, os dados deveriam ser interpretados, para que fosse possível recriar a lógica de montagem do display fora do simulador, e para isso, foi necessário analisar a imagem do *display* e os dados brutos, para buscar compreender qual a conexão havia entre eles. O código do *TcpClient.js* mostrado na Figura 30, iniciou recebendo um dado, após a inscrição no primeiro *dataref* de linha disponível. O terminal do Visual Studio mostrou o dado bruto, em Base64, da linha solicitada, e o resultado pode ser verificado na Figura 31.

Figura 30 – Código presente no *TcpClient.js*.

```

1  const net = require('net');
2
3  // configuração da r
4  const host = '127.0.0.1'; //maquina local
5  const port = 51000; //porta aberta pelo extplane
6
7  // Cria cliente TCP
8  const client = new net.Socket();
9
10 // Conecta ao ExtPlane
11 client.connect(port, host, () => {
12   console.log(`Conectado ao ExtPlane em ${host}:${port}`);
13
14   // inscreve em um dataref
15   client.write("sub XCrafts/FMS/CDU_1_01\n");
16 });
17
18 // Recebe dados
19 client.on('data', (data) => {
20   console.log("📡 DADO RECEBIDO:");
21   console.log(data.toString());
22 });
23
24 // Erro
25 client.on('error', (err) => {
26   console.error("❌ Erro:", err.message);
27 });
28
29 // Conexão fechada
30 client.on('close', () => {
31   console.log("🔌 Conexão encerrada");
32 });

```

Fonte: O autor (2026).

Figura 31 – Informação recebida no terminal do Visual Studio.

```

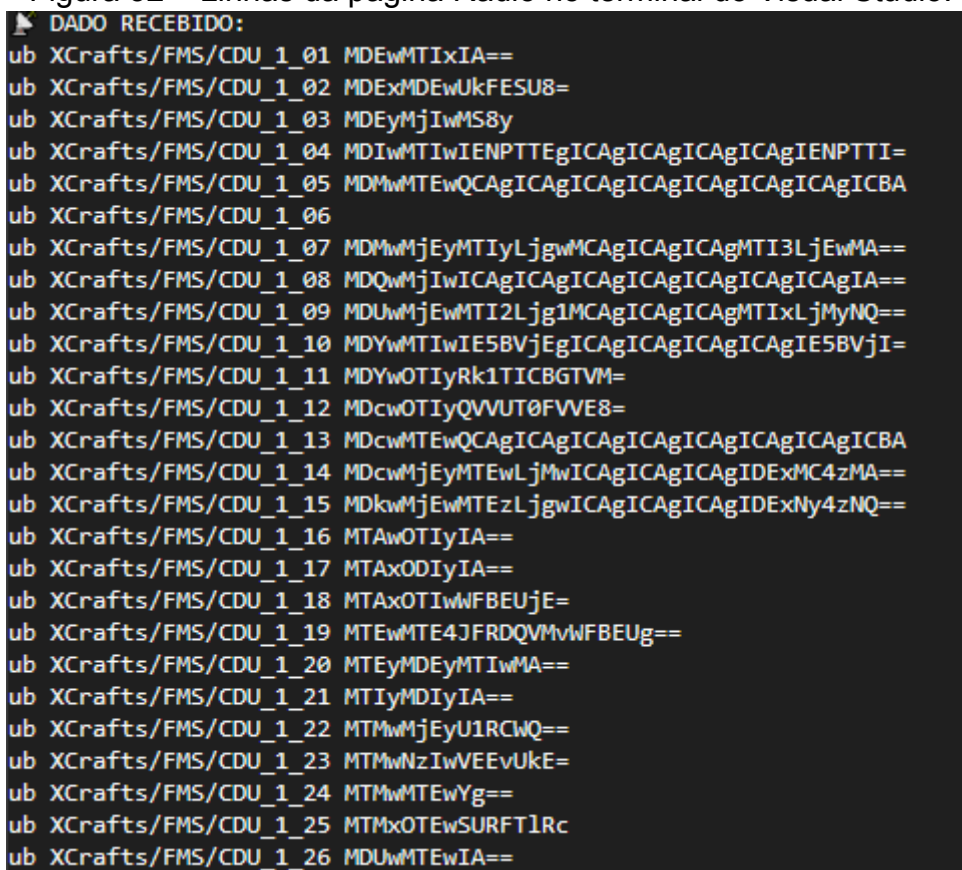
PowerShell do Desenvolvedor
+ PowerShell do Desenvolvedor | [ ] [ ] [ ]
📡 DADO RECEBIDO:
EXTPLANE 1.1
EXTPLANE-VERSION 1002
ub XCrafts/FMS/CDU_1_01 MDEwMTIxIA==

```

Fonte: O autor (2026).

Sendo assim, para recriar o *display*, o código solicitou todas as linhas existentes na página de rádio, informadas pela ferramenta DataRef Tool. Desse modo, foi possível solicitar as linhas de 01 a 26, as usadas pela página de rádio. O resultado obtido, presente na Figura 32, é a lista em sequência das linhas 01 a 26 dos *datarefs* do *display* do MCDU.

Figura 32 – Linhas da página Rádio no terminal do Visual Studio.



```

DADO RECEBIDO:
ub XCrafts/FMS/CDU_1_01 MDEwMTIxIA==
ub XCrafts/FMS/CDU_1_02 MDEwMDEwUkFESU8=
ub XCrafts/FMS/CDU_1_03 MDEyMjIwMS8y
ub XCrafts/FMS/CDU_1_04 MDIwMTIwIENPTTEgICAgICAgICAgICAgIENPTTI=
ub XCrafts/FMS/CDU_1_05 MDMwMTEwQCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_06
ub XCrafts/FMS/CDU_1_07 MDMwMjEwMTIyLjgwMCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_08 MDQwMjIwICAgICAgICAgICAgICAgICAgICAgICAgICAgIA==
ub XCrafts/FMS/CDU_1_09 MDUwMjEwMTI2Ljg1MCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_10 MDYwMTIwIE5BVjEgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_11 MDYwOTIyRk1TICBGTVM=
ub XCrafts/FMS/CDU_1_12 MDcwOTIyQVVUT0FVVE8=
ub XCrafts/FMS/CDU_1_13 MDCwMTEwQCAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_14 MDCwMjEwMTIyLjgwMCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_15 MDkwMjEwMTIyLjgwMCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_16 MTAwOTIyIA==
ub XCrafts/FMS/CDU_1_17 MTAxODIyIA==
ub XCrafts/FMS/CDU_1_18 MTAxOTIwWFBEUjE=
ub XCrafts/FMS/CDU_1_19 MTEwMTE4JFRDQVMwWFBEUg==
ub XCrafts/FMS/CDU_1_20 MTEyMDEyMTIwMA==
ub XCrafts/FMS/CDU_1_21 MTIyMDIyIA==
ub XCrafts/FMS/CDU_1_22 MTMwMjEwMTIyLjgwMCAgICAgICAgICAgICAgICAgICAgICBA
ub XCrafts/FMS/CDU_1_23 MTMwNzIwVEEvUkE=
ub XCrafts/FMS/CDU_1_24 MTMwMTEwYg==
ub XCrafts/FMS/CDU_1_25 MTMxOTEwSURFTlRc
ub XCrafts/FMS/CDU_1_26 MDUwMTEwIA==

```

Fonte: O autor (2026).

O modo bruto enviado pelo simulador pôde ser convertido, através da classe *buffer* do Node.js que permite converter Base64 para uma representação textual (Node.js, 2024b). Após a conversão obteve-se a lista completa dos dados da página de rádio, que podem ser visualizados na Figura 33, onde é mostrado o resultado dos dados obtidos no terminal do Visual Studio. Através dos dados extraídos, e com um método comparativo, é possível retomar, não apenas a página de rádio, como todas as outras páginas, e montá-las novamente com características semelhantes ao simulador e ao dispositivo real.

Um ponto de atenção nesse momento, observando os dados convertidos, é que não necessariamente a linha 1 do *dataref* corresponda à linha 1 da página

aberta naquele momento. No caso da página de rádio, o *dataref* da linha 1 traz a informação “010121”, que é o código utilizado pela X-Crafts para organizar o conteúdo, mas não necessariamente é o que irá aparecer.

Figura 33 – Informações da página Rádio convertidas para *string*.

```
Conectado ao ExtPlane em 127.0.0.1:51000
Linha 01: 010121
Linha 02: 011010RADIO
Linha 03: 0122201/2
Linha 04: 020120 COM1          COM2
Linha 05: 030110@             @
Linha 07: 030212122.800      127.100
Linha 08: 040220
Linha 09: 050210126.850      121.325
Linha 10: 060120 NAV1        NAV2
Linha 11: 060922FMS  FMS
Linha 12: 070922AUTOAUTO
Linha 13: 070110@             @
Linha 14: 070212110.30      110.30
Linha 15: 090210113.80      117.35
Linha 16: 100922
Linha 17: 101822
Linha 18: 101920XPDR1
Linha 19: 110118$TCAS/XPDR
Linha 20: 1120121200
Linha 21: 122022
Linha 22: 130212STBY
Linha 23: 130720TA/RA
Linha 24: 130110b
Linha 25: 131910IDENT\
Linha 26: 050110
Linha 17: 101822@
Linha 17: 101822
```

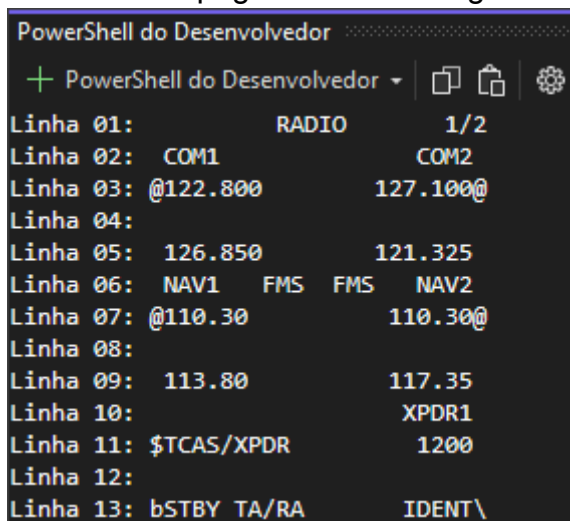
Fonte: O autor (2026).

Neste ponto do projeto, para que a reorganização dos dados seja realizada com sucesso, buscou-se entender como funcionava a divisão do *display*, e, através do número de caracteres, foi possível entender que o *display* é organizado em uma grade de 14 linhas por 24 colunas. A informação se comprova ao analisar os dados. A linha 3 do MCDU contém as frequências ativas nos canais de comunicação COM1 e COM2. Observando a Figura 26, é possível extrair a informação 122.800 e 127.100, além dos símbolos de troca de frequência. As mesmas informações podem ser observadas na linha 07 dos dados extraídos do simulador, que informam “030212122.800 127.100”. Sabe-se que as informações anteriores são pertencentes à linha 3 do MCDU, e que subtraindo a informação esperada dos

dados que foram convertidos, restaria a informação “030212”, já que até os espaços são contados como caracteres. Desse modo, espera-se que os dois primeiros caracteres correspondam ao número da linha da informação. Através da Figura 26 sabe-se também que a informação inicia na coluna 2, visto que a coluna 1 e a coluna 24 são utilizadas para exibir os símbolos de troca de frequência. Sendo assim, os próximos dois dígitos correspondem ao número da coluna de início da informação. Por fim, os últimos dois dígitos definem o tipo de fonte utilizada, definindo o tamanho e a cor apresentada. O método de codificação então é descrito por uma sequência “LLCCEE TEXTO”, onde “LL” representa a linha, “CC” representa a coluna e “EE” representa o estilo de texto, já o “TEXTO” é o conteúdo exibido.

Neste momento, para fins de organização da informação, utilizaram-se somente as informações de linhas e colunas, para que fosse possível organizar de maneira clara a informação recebida do simulador, e portanto, como demonstrado na Figura 34, exibir no console utilizando os códigos de organização da própria desenvolvedora da aeronave.

Figura 34 – Dados da página de Rádio organizados via LLCCEE.



```

PowerShell do Desenvolvedor
+ PowerShell do Desenvolvedor
Linha 01:          RADIO          1/2
Linha 02:   COM1          COM2
Linha 03: @122.800      127.100@
Linha 04:
Linha 05:   126.850      121.325
Linha 06:   NAV1   FMS   FMS   NAV2
Linha 07: @110.30      110.30@
Linha 08:
Linha 09:   113.80      117.35
Linha 10:
Linha 11: $TCAS/XPDR      1200
Linha 12:
Linha 13: bSTBY TA/RA      IDENT\
  
```

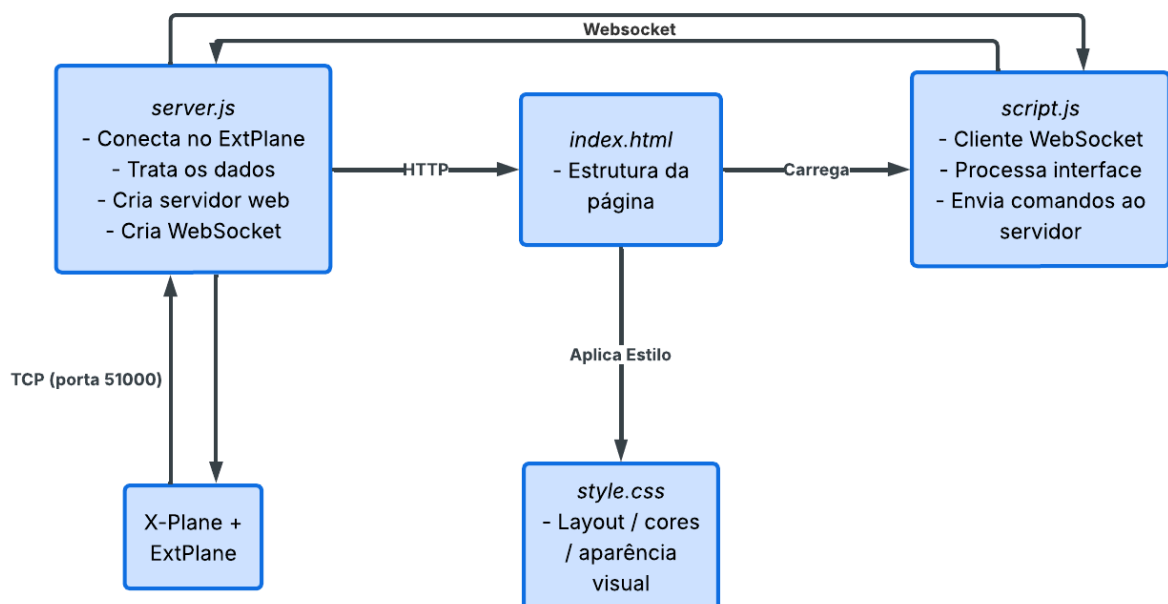
Fonte: O autor (2026).

4.1.3 Arquitetura de Comunicação

Em posse dos dados organizados e compatíveis com a informação que a página de rádio apresentava, buscou-se compartilhar a informação para uma página web via rede local. Para isso, o sistema foi estruturado em uma arquitetura baseada

em uma comunicação entre o simulador, um servidor local e a interface *web*. Esse modelo de organização permitiu separar a aquisição dos dados, o tratamento das informações, a comunicação em tempo real e a renderização gráfica do MCDU. Para que a nova estrutura pudesse ser colocada em prática, foram criados diferentes arquivos que formam a estrutura do *software*. Sendo assim, foram criados os arquivos *server.js*, *script.js*, *index.html* e *style.css*. O fluxo de trabalho é variado. O X-Plane é capaz de enviar os dados através do ExtPlane, porém também os recebe em forma de comandos. A comunicação entre o simulador e a interface externa é feita agora pelo *server.js*, que também é responsável por criar um servidor HTTP para criação da página *web*, e se comunicar através de WebSocket com o *script.js*. O HTTP se comunica na *web* através do *index.html*, o qual é responsável por carregar tanto o *script.js* quanto o *style.css*. A Figura 35 a seguir demonstra o fluxograma de conexão entre os novos arquivos.

Figura 35 – Fluxograma do sistema.



Fonte: O autor (2026).

A comunicação com o ExtPlane foi implementada no ambiente Node.js, utilizando o módulo nativo *net*, responsável pela criação de conexões TCP. De acordo com a documentação técnica do Node, o módulo *net* fornece uma API (Application Programming Interface) ou Interface de Programação de Aplicações, para a criação tanto de clientes quanto servidores TCP, permitindo uma

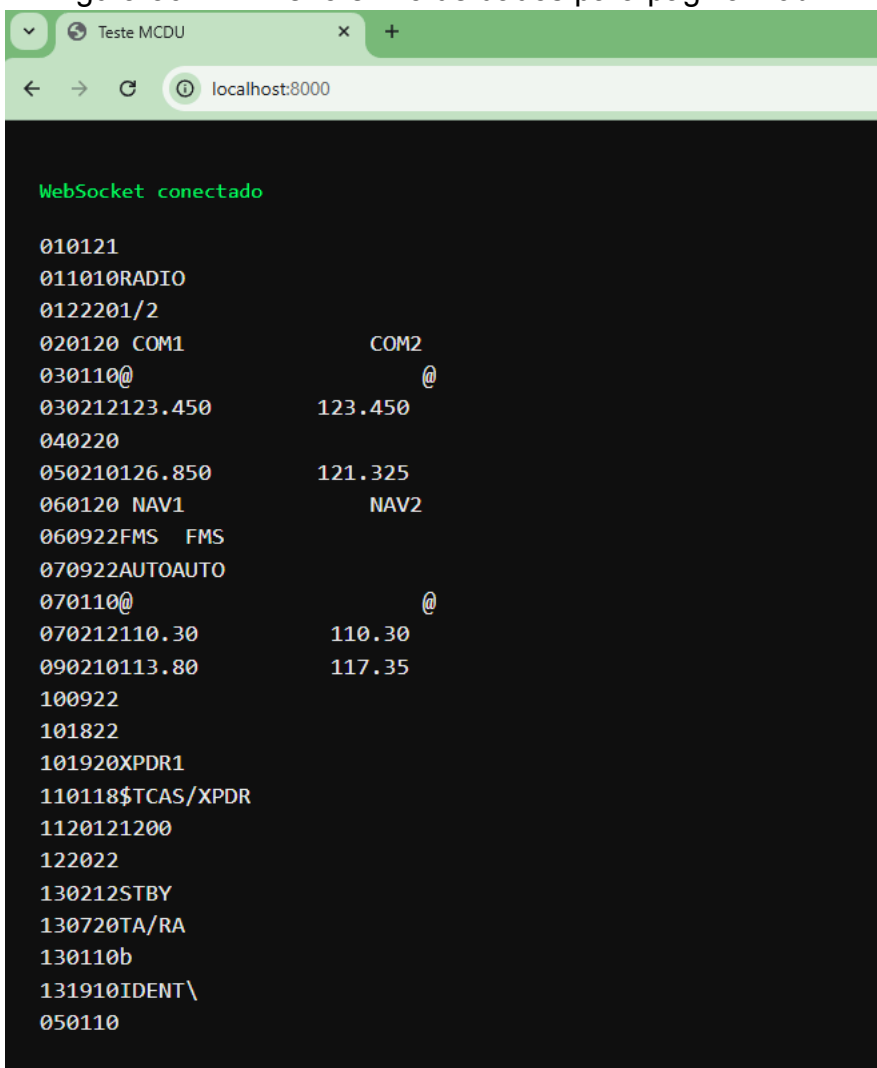
comunicação direta com serviços externos disponíveis, por meio de *sockets* (Node.js, 2024a). Os *sockets* podem ser definidos como um processo de comunicação que permite que dois diferentes processos se comuniquem e troquem informações entre si, nesse caso, entre cliente e servidor (IBM, 2022). Dessa forma, o estabeleceu conexão com o simulador na porta 51000, enviando os comandos de inscrição *sub* para receber informações e atualizações contínuas dos *datarefs* solicitados, referentes às linhas do *display* do MCDU.

4.1.4 Desenvolvimento da Interface Gráfica

A interface gráfica implementada para visualização, foi estruturada através do HTML, CSS e JavaScript. O servidor, responsável por disponibilizar a interface, desenvolvido em Node.js, utiliza o *framework* Express, que facilita a criação de aplicações *web* e o gerenciamento de rotas e arquivos estáticos (Express, 2024). *Framework* é um conjunto de bibliotecas e ferramentas que fornecem uma estrutura pré-definida para o desenvolvimento de *softwares* (Alura, 2021). Além disso, o servidor HTTP foi implementado por meio do módulo *http* do Node, o qual permitiu a criação do servidor capaz de responder as requisições *web*. Neste contexto, o servidor é responsável por disponibilizar o arquivo *index.html*, que contém a estrutura da interface, bem como os arquivos auxiliares *script.js* e o *style.css*, responsável pela apresentação visual da página.

A Figura 36 apresenta o primeiro envio de dados para a página *web*, que contou com o envio dos dados após a conversão, apenas para testar o envio dos dados obtidos do simulador para a página.

Figura 36 – Primeiro envio de dados para página web.



The screenshot shows a web browser window with the title 'Teste MCDU' and the address bar displaying 'localhost:8000'. The main content area has a black background with green text. At the top, it says 'WebSocket conectado'. Below this, there is a list of datarefs, some of which are grouped into columns. The datarefs listed are:

```

010121
011010RADIO
0122201/2
020120 COM1          COM2
030110@              @
030212123.450        123.450
040220
050210126.850        121.325
060120 NAV1          NAV2
060922FMS FMS
070922AUTOAUTO
070110@              @
070212110.30         110.30
090210113.80         117.35
100922
101822
101920XPDR1
110118$TCAS/XPDR
1120121200
122022
130212STBY
130720TA/RA
130110b
131910IDENT\
050110
  
```

Fonte: O autor (2026).

Após a validação da recepção dos dados da página de rádio, para fins de teste, alterou-se a página do MCDU no simulador. A interface exibiu os dados convertidos de outra página do simulador na interface *web*, porém, dois problemas ocorreram. O primeiro problema foi a mistura dos dados. A página de rádio utiliza as linhas de 01 a 26 do *dataref*, outra página porém, utiliza apenas as linhas de 01 a 10. Isso fez com que os dados dos *datarefs* das linhas de 01 a 10, ao alterar a página, se atualizassem, porém, as linhas de *datarefs* de 11 a 26 não se alteraram, e fez com que os dados exibidos na página *web* fossem um misto de duas páginas, e este erro pode ser notado no destaque da Figura 37. O segundo problema também estava relacionado às linhas dos *datarefs*. Uma página, como a de rádio, pode usar apenas 26 linhas, porém a aeronave da X-Crafts dispõe de 70 linhas de *datarefs* para que as páginas utilizem. Sendo assim, algumas páginas não mostraram

totalmente seu conteúdo, devido a uma limitação no código em relação ao número de linhas inscritas, visto que o código estava inscrito apenas em 26 linhas.

Figura 37 – Dados misturados na página *web*.

```

WebSocket conectado

010121
010810PERF INDEX
0122201/1
030120$PERF INIT
050120$CRUISE
070120$DESCENT
090120$APP SPD
110120$LANDING
031720TAKEOFFv
051920CLIMBv

060922FMS FMS
070922AUTOAUTO
070110@
070212110.30 110.30
090210113.80 117.35
100922
101822
101920XPDR1
110118$TCAS/XPDR
1120121200
122022
130212STBY
130720TA/RA
130110b
131910IDENT\
050110
  
```

Fonte: O autor (2026).

Para solucionar os problemas, foi necessário criar um mapa de páginas no arquivo *server.js*, chamado no código de *pageMap*. Porém, para criar um mapa de páginas, além de identificar quais linhas eram utilizadas, tornou-se necessário também criar uma identificação para cada página. Através da ferramenta DataRefTool, foi descoberto o *dateref* “XCrafts/FMS/CDU1_page”, o qual contém um

número para cada página diferente do MCDU. Sendo assim, criou-se o *pageMap* para cada número de página, contendo somente o conjunto das linhas necessárias a serem inscritas. Além disso, foi necessário criar uma lógica para que, sempre que houvesse uma troca de página, os dados anteriores fossem apagados e se exibisse apenas os novos dados apresentados, de acordo com a página. Desse modo, foram corrigidos os problemas de mistura de dados entre páginas e a falta de informações.

O arquivo *style.css* define os aspectos visuais do sistema, incluindo as definições das letras de acordo com o código “LLCCSS” e o *layout*. Segundo o código “LLCCSS”, os últimos dois números se referem ao estilo do texto, como tamanho e cor. Para isso, através de um comando no *script.js*, foram atribuídas classes para cada tipo de código diferente recebido do simulador, fazendo com que os dados fossem exibidos com as características recebidas através do servidor. Além da classe de letras, também foi criada uma classe para que os símbolos pudessem ser exibidos da maneira correta, e, também, o CSS é responsável por criar o *grid* de exibição das informações, onde cada caractere ocupa um espaço. A estrutura de classes é apresentada na Figura 38, criando uma codificação para a correta caracterização do conteúdo do MCDU.

Figura 38 – Classes do texto para o CSS.

```
// Classes para o CSS
1 reference
function styleToClass(code) {
  const n = parseInt(code, 10);
  if ([10, 16, 20].includes(n)) return 'w';
  if ([12, 22].includes(n)) return 'g';
  if ([11, 21].includes(n)) return 'c';
  if ([13].includes(n)) return 'y';
  if ([14].includes(n)) return 'mgt';
  if ([18, 26].includes(n)) return 'gy';
  return 'w';
}
```

Fonte: O autor (2026).

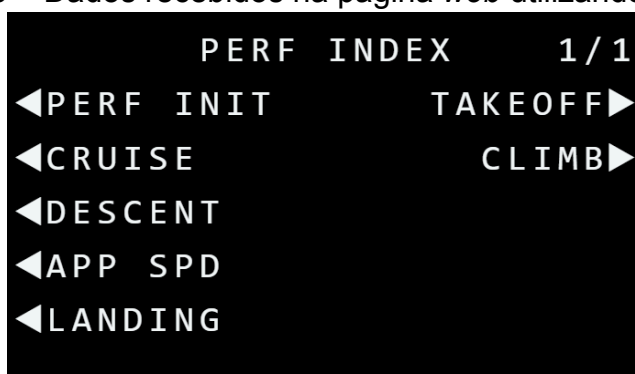
O arquivo *script.js* por sua vez, atua como o cliente WebSocket. O protocolo WebSocket foi utilizado para a comunicação em tempo real entre o servidor e a interface *web*, que, diferentemente do modelo tradicional HTTP, permite comunicação bidirecional contínua, no modelo *full-duplex*, mantendo uma conexão sempre ativa entre cliente e servidor. De acordo com a documentação da MDN, este

tipo de abordagem permite o envio de dados de maneira assíncrona, sem a necessidade de múltiplas requisições, sendo assim a mais adequada para aplicações que necessitam de atualizações em tempo real (MDN, 2023). Deste modo, o *script.js* atuou de modo a estabelecer conexão com o servidor, receber os dados processados e atualizar dinamicamente o conteúdo da interface. Ao receber os dados vindos do *server.js*, o *script.js* também é responsável por trocar os caracteres símbolos para o formato correto, visto que a aeronave produzida pela X-Crafts utiliza de substituição de caracteres para os símbolos corretos.

No lado do servidor, a implementação do WebSocket foi realizada utilizando a biblioteca *ws*, utilizada no Node para a criação de servidores WebSocket. A biblioteca permite gerenciar conexões simultâneas e enviar mensagens a múltiplos clientes conectados, mesmo que para o protótipo do MCDU se utilize apenas um cliente. Desta forma, o sistema desenvolvido integra múltiplas tecnologias de forma conjunta, estabelecendo conexão entre o simulador, o servidor e a interface *web*, permitindo assim, a reprodução fiel do MCDU em um ambiente externo ao simulador.

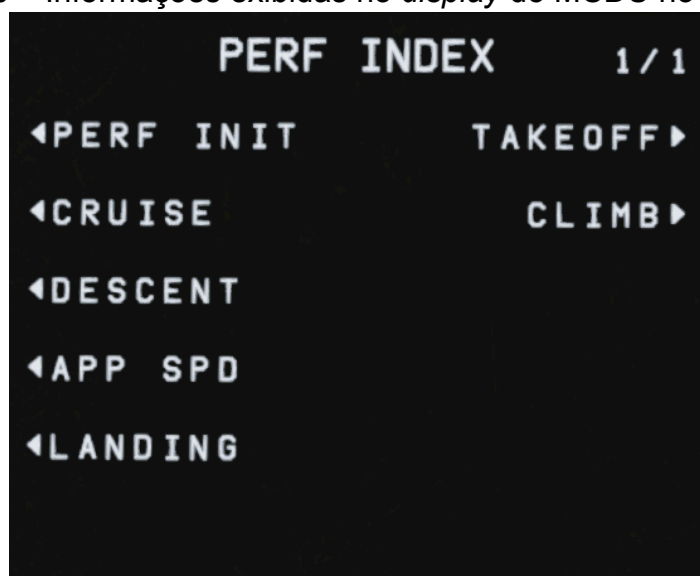
Como demonstrado na Figura 39, após os ajustes necessários para estilizar os dados no arquivo CSS utilizando as classes criadas e concentrá-los em um *grid*, a interface gráfica do *display* do MCDU se reproduziu em uma página *web*, garantindo o recebimento em tempo real dos dados recebidos, e tornando possível a exibição em um *display* externo ao simulador. Para efeito de comparação, na Figura 40 é apresentada uma captura de tela do MCDU diretamente no simulador, demonstrando que o conteúdo pode ser reapresentado via página *web* com precisão.

Figura 39 – Dados recebidos na página *web* utilizando *grid* 14x24.



Fonte: O autor (2026).

Figura 40 – Informações exibidas no *display* do MCDU no simulador.

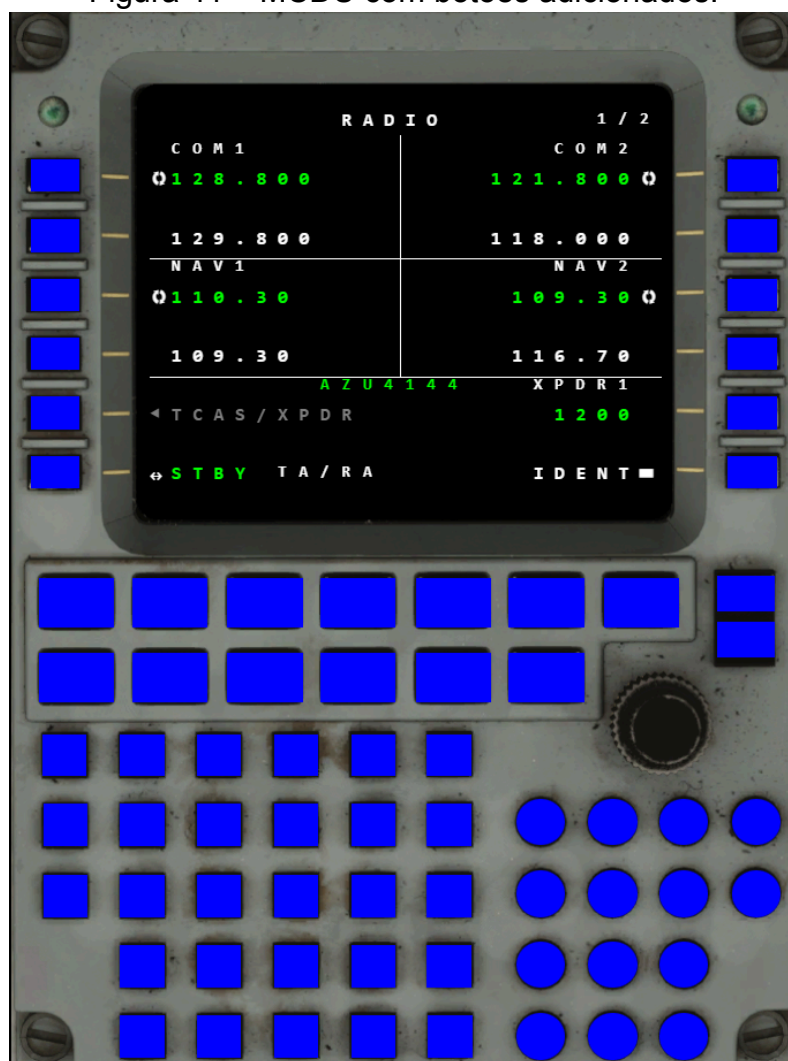


Fonte: O autor (2026).

4.1.5 Envio de Comandos ao Simulador

Neste momento, a página *web* era capaz de demonstrar, em tempo real, o *display* do MCDU presente no simulador. Para o completo funcionamento da interface *web*, tornou-se necessário criar a estrutura para envio de dados para o simulador, como os comandos de teclado alfabético e numérico, teclado de funções e teclas para seleção. Para isso, utilizou-se do WebSocket para se comunicar da página *web* até o servidor, o qual realizou o envio de comandos para o simulador. Em um primeiro momento, foi necessário inserir uma imagem do MCDU na página *web*, a fim de criar os botões e testar o envio de comandos. Sendo assim, adicionou-se a imagem e sob ela, foram criados todos os botões, de maneira visível, para que fosse possível testar seu funcionamento. Na Figura 41, é possível visualizar a imagem do MCDU que foi carregada na interface *web*. Para cada botão, foi criado um quadrado azul com função de comando, e a cada clique em determinado quadrado azul, um comando era enviado ao simulador. Posteriormente, o envio de comandos será alterado para envio via os pinos GPIO do Raspberry, porém, para o início dos testes de envio, o desenvolvimento deste tipo de botão tornou-se crucial para garantir o funcionamento da interface.

Figura 41 – MCDU com botões adicionados.



Fonte: O autor (2026).

Assim como informado anteriormente, cada botão criado envia um determinado comando para o simulador. Os comandos do MCDU da aeronave E-195 da X-Crafts foram obtidos através do DataRef Tool, e atribuídos para cada botão. Quando clicado na página *web*, o botão envia para o *server.js* um código, referente à tecla pressionada. Quando recebido no servidor, o código é interpretado e então, envia-se para o simulador um *command* específico da tecla pressionada. Através da documentação do ExtPlane, sabe-se que *commands* são as variáveis de comando para controle de diferentes sistemas no simulador. Portanto, ao receber o comando específico via ExtPlane, o simulador o executa, e deste modo, através do envio do comando, é possível ter um teclado funcional para o MCDU.

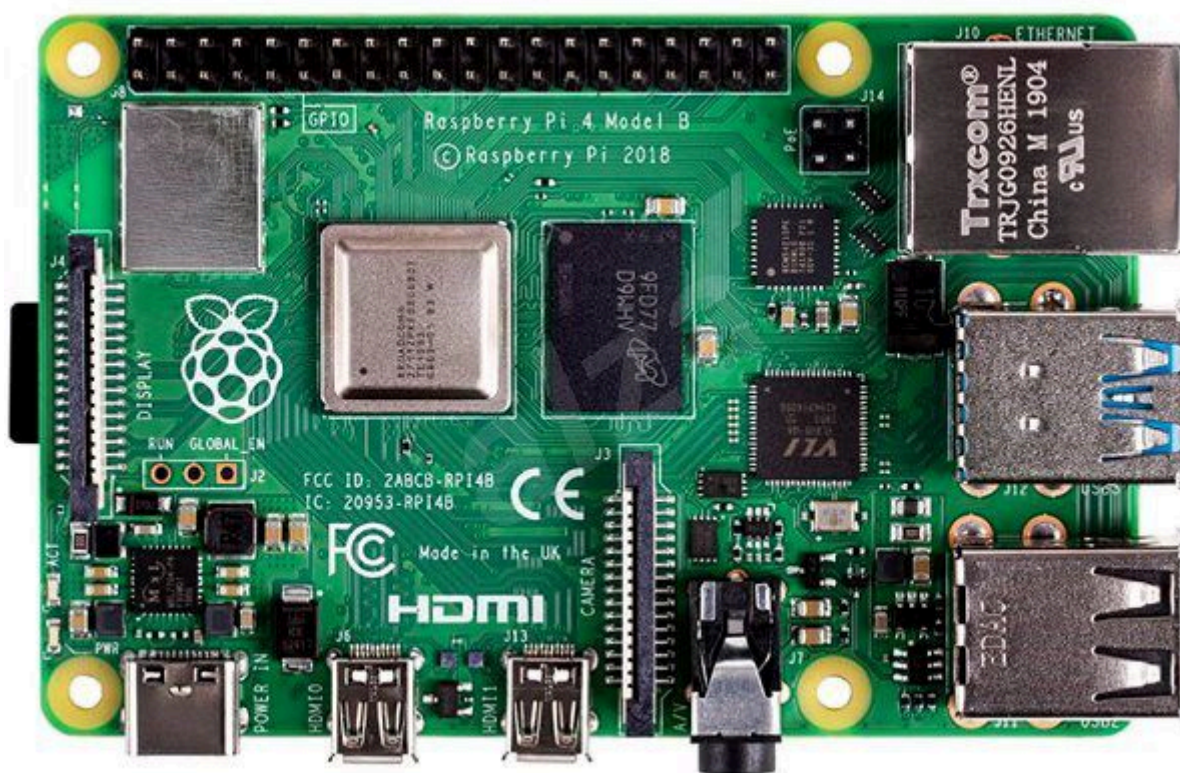
Portanto, com o *software* modelado para utilização em uma página *web*, capaz de ser acessado via Raspberry Pi 4, seguiu-se para o desenvolvimento de

hardware do projeto, definindo as necessidades do protótipo, modelagem e integração dos sistemas, e produção dos circuitos elétricos e eletrônicos para o pleno funcionamento do produto.

4.2 CONSTRUÇÃO DO PROTÓTIPO

Para a construção do protótipo, entende-se a necessidade de integração do sistema, para a criação de um modelo funcional. Optou-se então por utilizar um Raspberry Pi 4 Model B, apresentado na Figura 42, sendo ele o responsável pelo processamento do MCDU. Através de sua saída HDMI, foi possível enviar a imagem para um *display*, o qual equipa o protótipo.

Figura 42 – Raspberry Pi 4 Model B.



Fonte: Raspberry (2019).

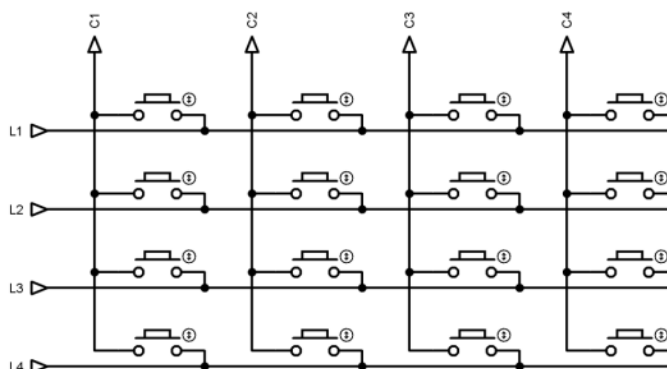
O Raspberry escolhido possui capacidade de processamento suficiente para ser o controlador do MCDU, visto que o produto necessita de acesso à *web* e processamento de vídeo para visualização em *display*. O dispositivo conta com 4GB de RAM e um processador 64 *bits* ARM Cortex A-72 de 1,5GHz, além disso conta

com conexões GPIO e suporte nativo a comunicação de rede. Para a exibição, em um primeiro momento, utilizou-se de um *display* de 5 polegadas já disponível para testes. Portanto, iniciou-se o Raspberry, e através do acesso à página *web* no endereço de *localhost*, foi possível receber os dados do *display* e com o uso de teclado e *mouse*, enviar comandos para o simulador. Para que o protótipo pudesse ser montado, tornou-se necessário desenvolver um teclado, no formato do MCDU original, para que se comunicasse com o Raspberry e então, o comando pudesse ser enviado para o simulador.

4.2.1 Desenvolvimento do Teclado Matricial

Para o desenvolvimento do teclado, através do *software* Altium Designer com licença de estudante, iniciou-se o desenvolvimento da *Printed Circuit Board* (PCB) ou placa de circuito impresso para o teclado. O Raspberry não conta com 68 portas digitais para entrada de dados, portanto, outra abordagem deveria ser escolhida para que o teclado pudesse funcionar. Sendo assim, testou-se um teclado em formato matricial 2x2, onde eram necessários 4 pinos digitais para serem utilizadas 4 teclas. O teste foi feito através de um tutorial fornecido pelo *blog* Embarcados para um teclado matricial 4x4 comercial, porém adaptado para um 2x2 com chaves de toque em uma *protoboard*. Após garantir a funcionalidade, estendeu-se o teclado de 2x2 para 8x9, criando 72 teclas disponíveis e utilizando apenas 17 pinos digitais do Raspberry. Na Figura 43, é possível visualizar o esquema de um teclado matricial 4x4.

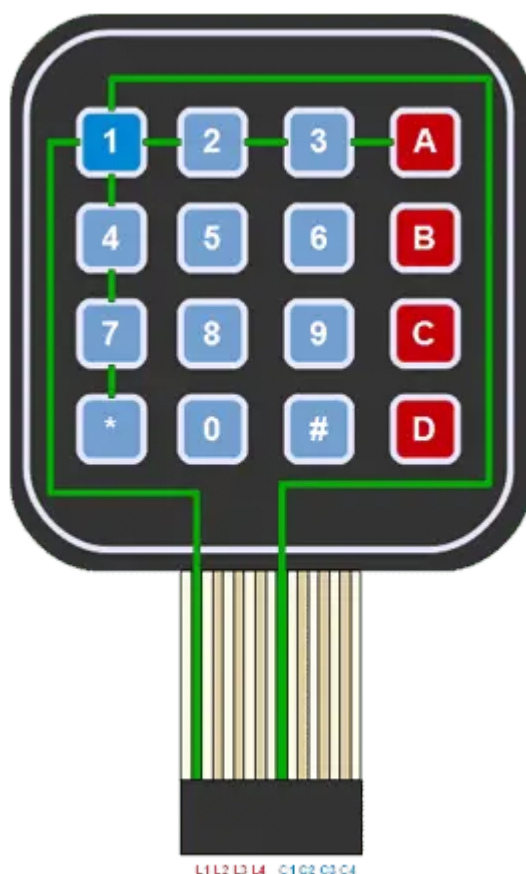
Figura 43 – Esquema elétrico teclado matricial.



Fonte: Vida De Silício (2017).

O funcionamento do teclado matricial se dá através da interconexão das linhas e colunas ligadas aos pinos digitais do Raspberry. No código de controle do dispositivo, os pinos de coluna foram configurados como *input*, com resistores internos de *pull down*, utilizando o comando `rpio.open(pin, rpio.INPUT, rpio.PULL_DOWN)`. Sendo assim, o estado lógico padrão é *low*. Durante a varredura do teclado, as linhas são ativadas individualmente em *high*. Quando uma tecla é pressionada, o sinal *high* da linha ativa é conduzido através da tecla até a coluna correspondente. O Raspberry então detecta a alteração do estado lógico da coluna de *low* para *high*, permitindo identificar qual tecla foi pressionada. A Figura 44 apresenta o teclado matricial 4x4 referente ao tutorial do *blog* Embarcados utilizado como referência.

Figura 44 – Acionamento tecla 1.

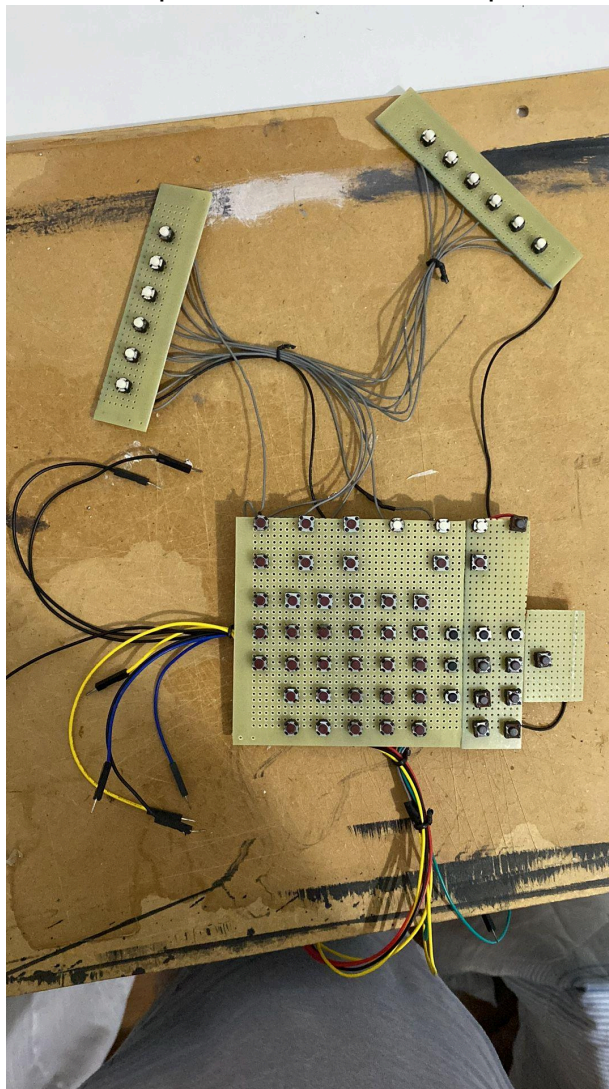


Fonte: Embarcados (2015).

Para validação de todo o teclado matricial, foi desenvolvido um teclado provisório utilizando uma placa de fenolite ilhada com algumas chaves de toques reaproveitadas de aparelhos de sucata, e este teclado de validação está presente na

Figura 45. Isso garantiu o baixo custo para o teste do teclado, e também a possibilidade de validar a teoria do teclado matricial, antes de enviar para produção as placas de circuito impresso. As chaves de toque foram soldadas na placa de fenoliteilhada e conectadas entre si com cabos de soquetes de aparelhos eletrônicos usados. Nesta etapa não foram utilizados diodos ou resistores externos *pull down*, apenas as chaves. O protótipo do teclado permitiu validar experimentalmente o funcionamento da lógica de varredura do teclado matricial, bem como identificar problemas relacionados a acionamentos indevidos, como o *ghosting* ou “comandos fantasmas”. Esses efeitos ocorreram devido à ausência de componentes eletrônicos de estabilização, evidenciando a necessidade de implementação de melhorias no projeto da placa de circuito impresso definitiva.

Figura 45 – Protótipo PCB do teclado em placa de fenoliteilhada.



Fonte: O autor (2025).

4.2.2 Desenvolvimento das Placas de Circuito Impresso

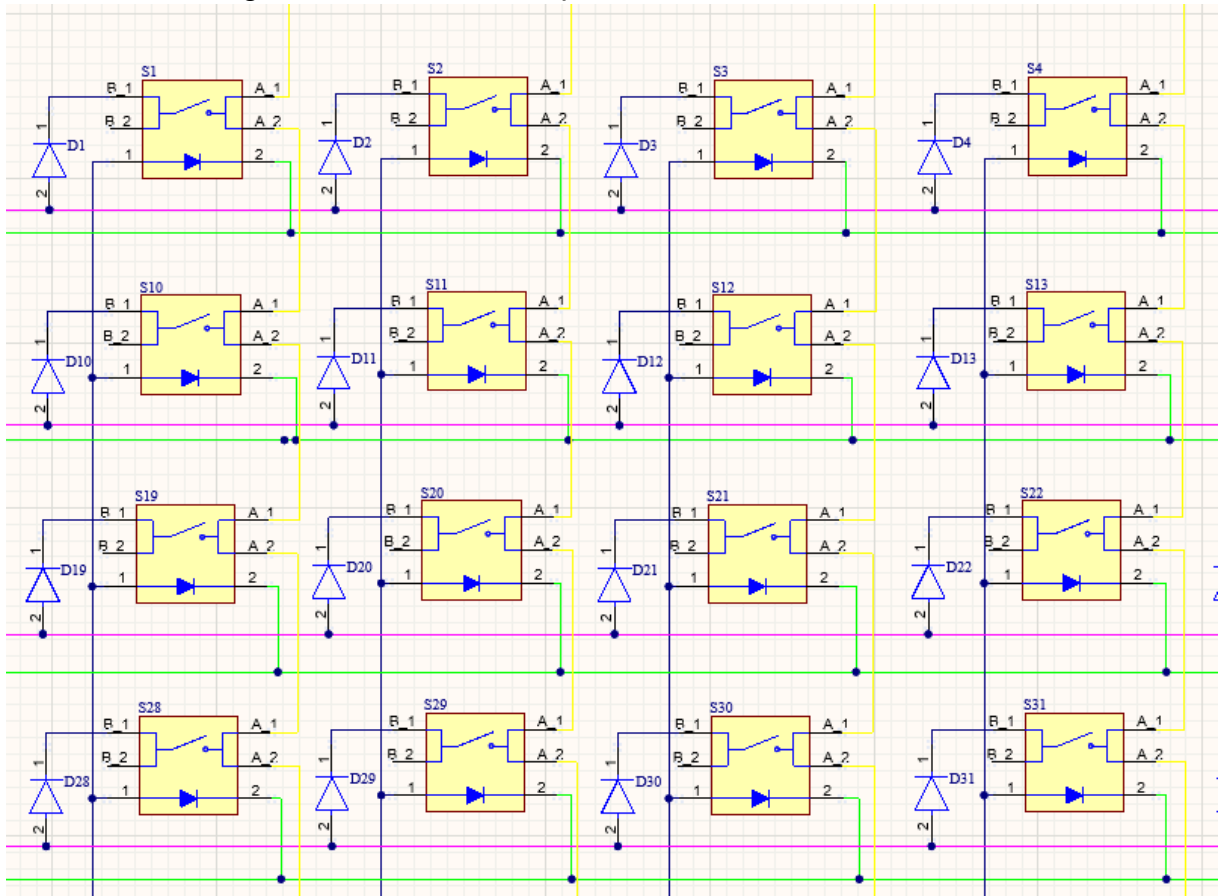
Sendo assim, iniciou-se o desenvolvimento da PCB a ser utilizada no protótipo do dispositivo. Para iniciá-lo, era necessário possuir as medidas do dispositivo real, fazendo com que o protótipo pudesse ser o mais semelhante possível ao dispositivo original. Foi utilizado então um modelo 3D fornecido na página *web* 3D Cults de uma réplica da *case* frontal do dispositivo visto que o equipamento real não estava disponível para medição. Através do modelo, foi possível retirar as medidas e convertê-lo para abrir em anexo à PCB no Altium Designer. A utilização do modelo permitiu uma construção mais precisa, tanto da PCB, quanto do protótipo como um todo.

Para a montagem do esquema elétrico da placa de circuito impresso do teclado, foi necessário definir os componentes a serem utilizados. As chaves de toque utilizadas no projeto deveriam possuir um LED, para criar um teclado retroiluminado. A chave de toque escolhida foi do modelo TS27 desenvolvida pela Same Sky. Ela possui um LED embutido na tecla, garantindo a iluminação do teclado, possui uma boa durabilidade e fácil montagem. Para realizar o controle dos LEDs na PCB, foi necessário utilizar um controlador de LEDs, para isso, utilizou-se o módulo HT16K33, escolhido devido à sua ampla utilização em aplicação de matrizes de LEDs e teclados retroiluminados, além de possuir interface I2C para comunicação com o Raspberry (Adafruit, 2025). Entretanto, para adicionar um módulo controlador à PCB, foi necessário desenvolver uma segunda placa, utilizada apenas para conectar os controladores. Neste momento, dividiu-se o projeto em duas PCBs, uma dedicada apenas ao teclado, e uma segunda, contendo conexão para o Raspberry e para o controlador dos LEDs.

Na montagem da placa de circuito impresso do teclado, foram posicionadas as chaves de toque, e em cada entrada de sinal da chave, foi adicionado um diodo 1N4148, a fim de garantir que a corrente elétrica do sinal de tecla pressionada existisse em apenas um sentido, evitando o *ghosting*. Ainda, foram adicionados os conectores para a conexão entre o Raspberry e a matriz do teclado matricial, bem como um conector para comunicação entre o módulo HT16K33 e os LEDs. Por fim, foi adicionado à PCB um *encoder* duplo, dispositivo que compõe o produto e é responsável por realizar os ajustes de frequência de rádio. O *encoder* foi testado em uma *protoboard* e seu esquema de ligação inserido na PCB, junto a um conector

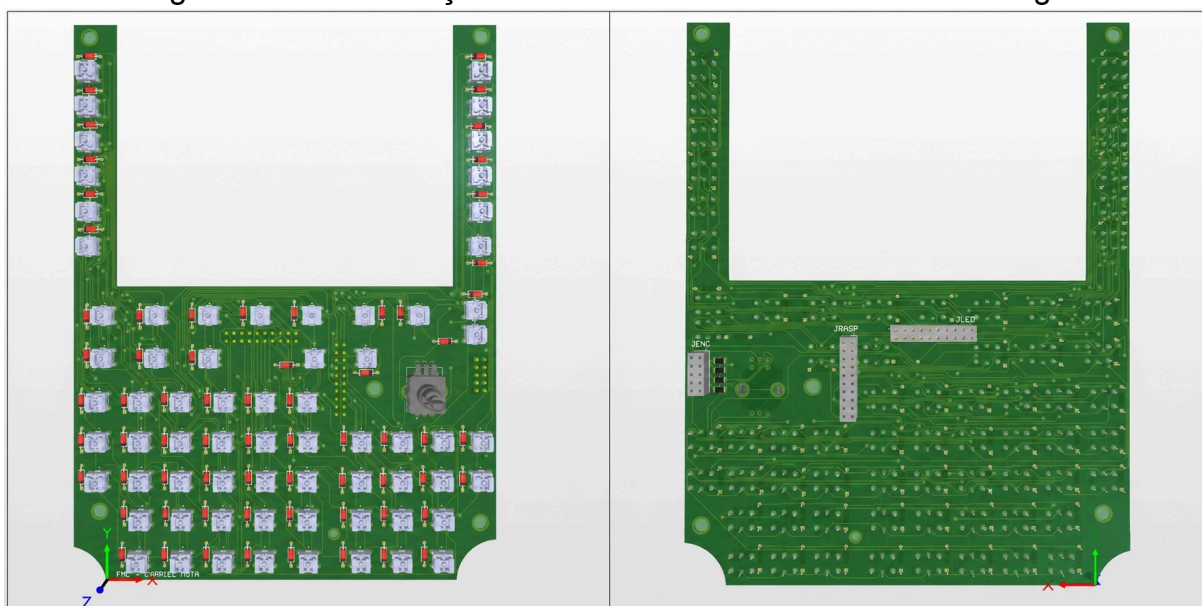
para se comunicar com a placa de controle do dispositivo. Sendo assim, na visão frontal do dispositivo, ficaram as chaves de toque posicionadas, juntamente aos diodos e o *encoder*, e na parte traseira da PCB os conectores para comunicação com os controladores. A placa foi desenvolvida no modelo *double layer*, ou seja, existem trilhas de comunicação nos dois lados da placa, devido ao alto número de conexões, permitindo assim melhor roteamento das trilhas, evitando cruzamentos. Na Figura 46 é apresentado parte do esquema elétrico da PCB do teclado, que pode ser visto por completo no Apêndice A. Por sua vez, a Figura 47 é a representação em 3D do esquema elétrico da Figura 46.

Figura 46 – Parte do esquema elétrico da PCB teclado.



Fonte: O autor (2025).

Figura 47 – Visualização 3D da PCB do teclado no Altium Designer.



Fonte: O autor (2025).

Para a placa de circuito impresso responsável pelo controle, foi necessário adicionar conectores para inserção do Raspberry e módulo HT16K33. Resistores foram adicionados em série com a saída dos pinos que são responsáveis pelo controle das linhas do teclado matricial, a fim de garantir a proteção dos pinos GPIO do Raspberry, além de controlar a corrente que chega ao circuito, funcionando também como uma segunda camada de proteção ao efeito *ghosting*. A comunicação entre o módulo HT16K33 e o Raspberry é simples, exigindo apenas uma alimentação de 5V, uma conexão GND e os pinos de controle SDA e SCL conectados ao Raspberry, mas seu uso foi fundamental, evitando o aumento de linhas de sinais para os LEDs.

4.2.3 Sistema de Alimentação

Uma alteração no projeto foi necessária devido ao uso do *display*. O modelo de testes não se adequou ao tamanho da case e portanto, foi substituído. A versão a qual serviu para o tamanho de área útil disponível para o *display* foi o AT056TN53, fabricado pela InnoLux. A versão foi importada, devido à falta de estoque em solo brasileiro. Sua conexão, entretanto, alterou o projeto da PCB de controle. Devido a sua alimentação ser de 12V, foi necessário inserir um conversor *buck* de 12V para 5V. Isso se dá devido ao requisito de utilizar apenas 1 fonte para funcionamento do

projeto. A fonte de 12V alimenta o conversor *buck* e o *display*, e a saída do conversor embutido na PCB de controle alimenta o Raspberry e os componentes do teclado. Esta alteração permitiu o pleno funcionamento do projeto. Para o desenvolvimento do conversor *buck*, utilizou-se o circuito integrado XL4015, com especificações que permitem o funcionamento conjunto do projeto. A alimentação necessária para o Raspberry, por padrão, é 5V de tensão e 3A de corrente. O XL4015, por sua vez, garante uma saída de 5V com até 5A de corrente, o suficiente para alimentar o projeto. A tensão de saída do conversor é definida pelos resistores R1 e R2 localizados na saída. A relação de tensão de saída é definida pela seguinte equação:

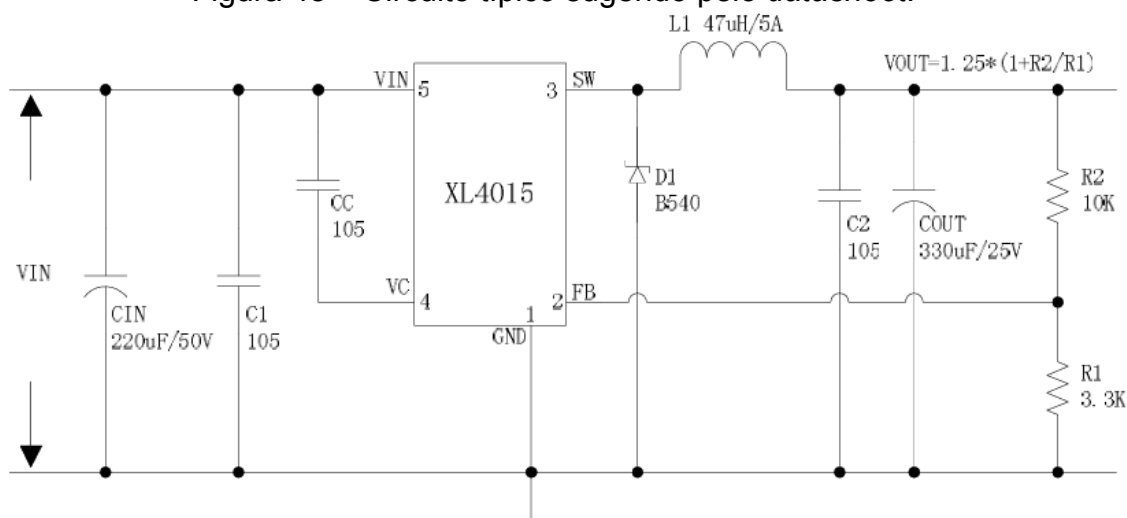
$$V_{out} = 1.25 * (1 + R2/R1) \quad (1)$$

Através do *datasheet* do componente, foi possível obter o circuito típico de aplicação para um conversor de 12V para 5V, com até 5A de corrente de saída para a carga. O circuito típico é apresentado na Figura 48. O documento sugere um resistor R1 de 3.3k ohm e R2 de 10k ohm.

$$V_{out} = 1.25 * (1 + 10/3.3) \quad (2)$$

$$V_{out} = 5,03V \quad (3)$$

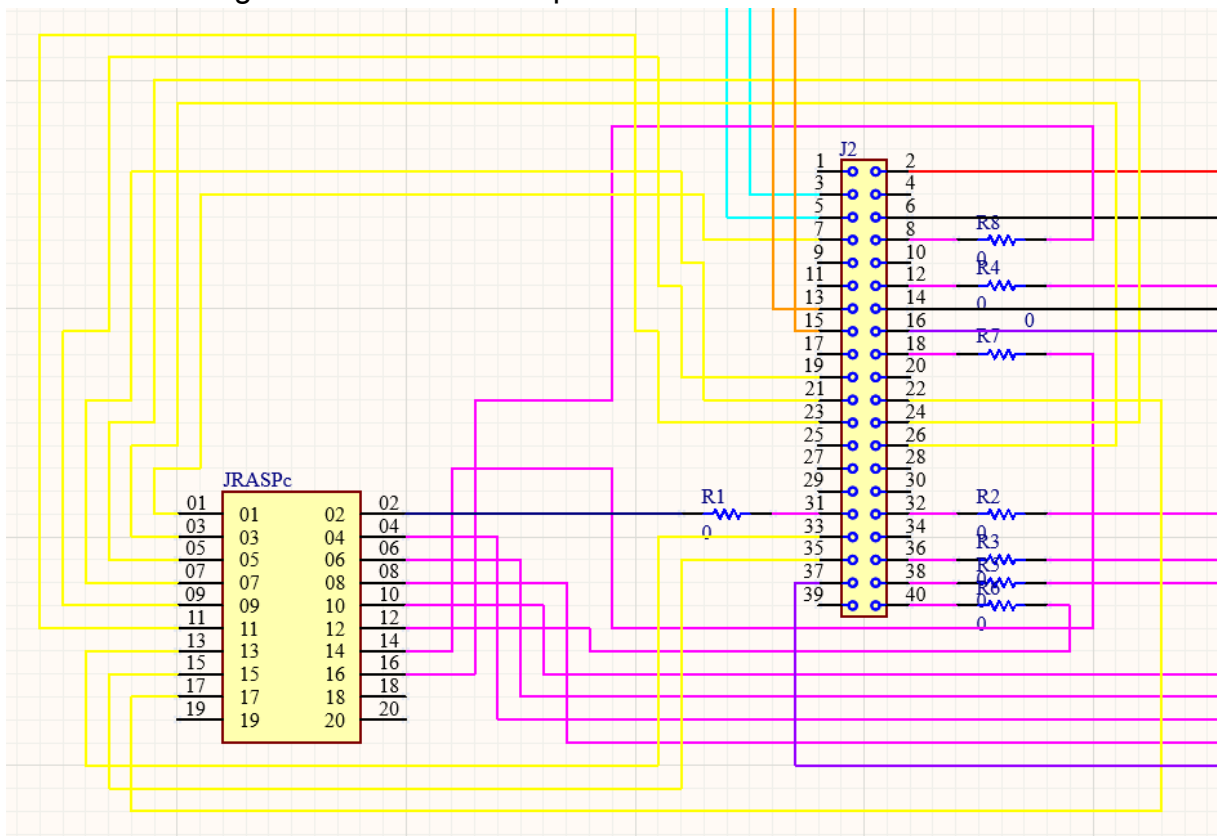
Figura 48 – Circuito típico sugerido pelo *datasheet*.



Fonte: Elecrow (2025).

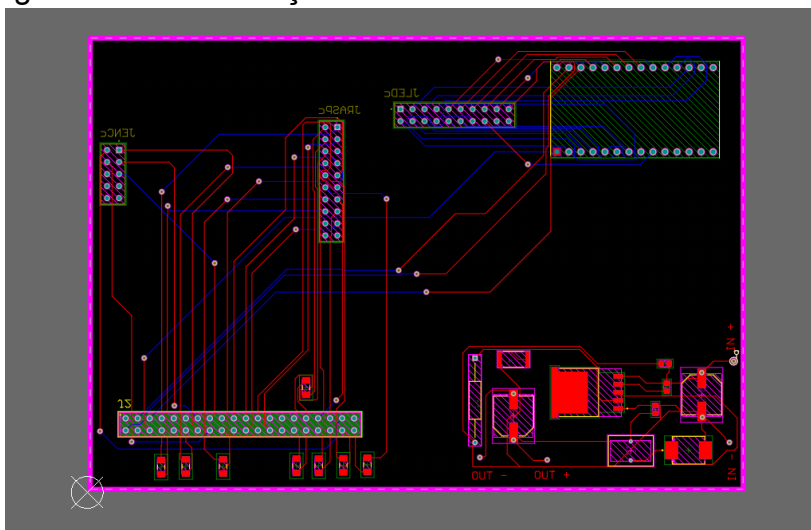
Portanto, a partir da definição do circuito, realizou-se a inserção do conversor no diagrama esquemático da PCB de controle. Deste modo, a PCB controladora foi finalizada, contando com conectores para o Raspberry e o módulo HT16K33, além de contar com um conversor *buck* para alimentação dos dispositivos. O posicionamento do conversor foi escolhido distante dos demais circuitos, a fim de evitar quaisquer problemas de cruzamento de sinais na PCB. A Figura 49 apresenta uma parte do esquema elétrico da PCB, que pode ser visto por completo no Apêndice B. Conforme disponibilizado pelo Altium Designer, é possível visualizar a PCB em 2D, e utilizá-la para distribuição das trilhas, como apresentado na Figura 50, e também é possível visualizar o resultado da placa em 3D, conforme a Figura 51.

Figura 49 – Parte do esquema elétrico da PCB de controle.



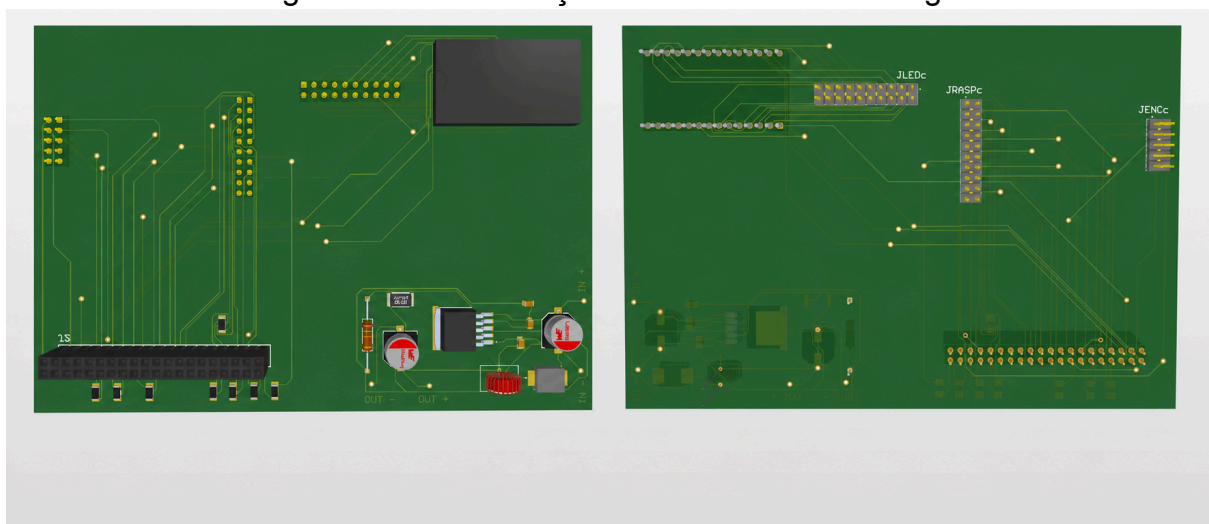
Fonte: O autor (2025).

Figura 50 – Visualização das trilhas em 2D no Altium Designer.



Fonte: O autor (2025).

Figura 51 – Visualização em 3D no Altium Designer.



Fonte: O autor (2025).

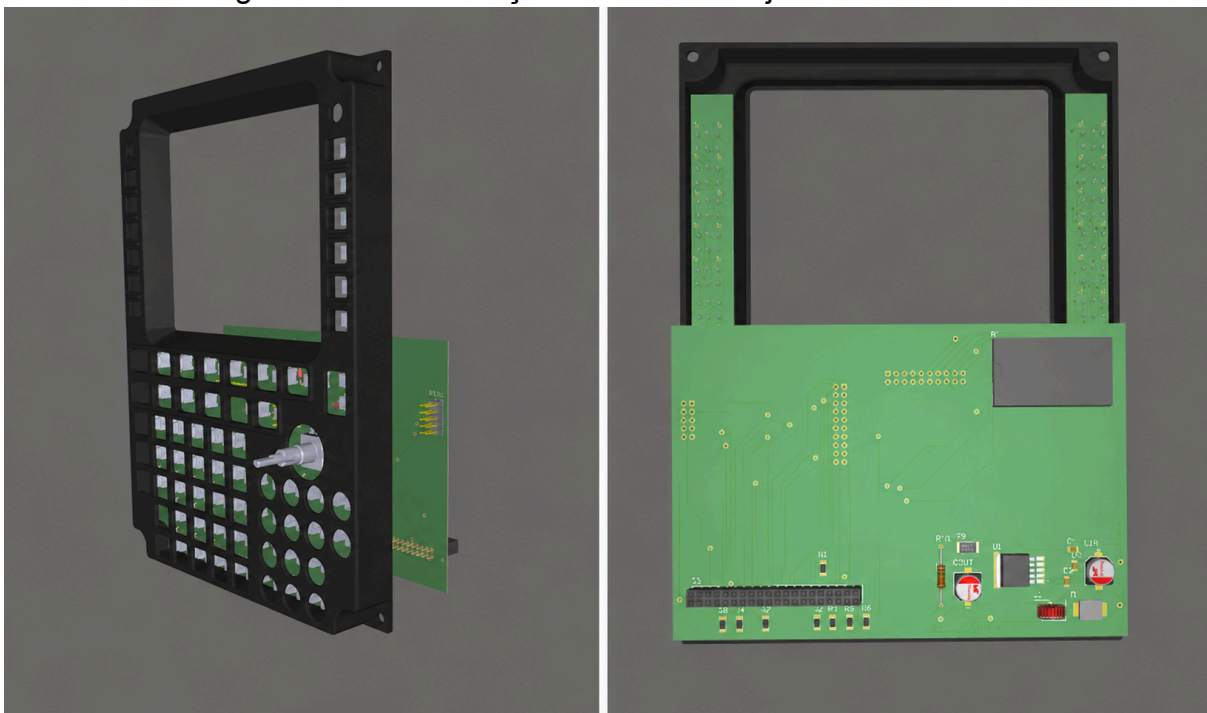
4.2.4 Integração Mecânica do Protótipo

A ferramenta de visualização em 3D permitiu integrar virtualmente a *case* frontal desenvolvida com a PCB do teclado e a PCB de controle, possibilitando uma análise completa do posicionamento físico dos componentes antes da fabricação das placas de circuito impresso na China. A utilização deste tipo de visualização teve um papel fundamental no processo de desenvolvimento do protótipo, uma vez que permitiu identificar previamente possíveis incompatibilidades entre os componentes eletrônicos e a estrutura física da *case*. Durante esta etapa, foi

possível validar dimensões, alinhamento entre conectores, o posicionamento dos botões de acordo com os espaços dispostos na case frontal, altura dos componentes eletrônicos e o espaço interno disponível, reduzindo a probabilidade de retrabalho após a fabricação das PCBs.

Outro fator relevante proporcionado pela visualização 3D foi a possibilidade de projetar os suportes internos de fixação da placa de teclado. A partir da análise do posicionamento da placa e altura dos componentes, tornou-se possível projetar na PCB os espaços para conexão com os suportes mecânicos para manter a PCB do teclado fixadas à estrutura frontal da case. Isso contribuiu para maior robustez estrutural do conjunto, fazendo com que, ao pressionar o teclado, a fixação melhore a experiência de uso. A Figura 52 apresenta a visualização de ambas as PCBs e a case frontal, facilitando o *design* do modelo.

Figura 52 – Visualização em 3D do conjunto case e PCBs.



Fonte: O autor (2025).

4.2.5 Fabricação e Montagem do Protótipo

Após a validação das PCBs, e a interação com a case frontal, foram gerados os arquivos conhecidos como *gerber files*, os quais são os arquivos padrão da indústria de produção de PCBs, contendo todas as informações das placas de

circuito impresso como camadas de cobre, mascara de solda, legenda e furação. Após a exportação dos arquivos, eles foram enviados para produção na China através da empresa JLCPCB, a qual realiza a fabricação, testes físicos e elétricos, e o envio das placas. O custo de impressão e envio foram contabilizados no valor final do projeto.

Durante o período de fabricação das PCBs, foi realizada a definição da lista de materiais necessários para montagem do protótipo. Esta etapa envolveu tanto a análise dos componentes já disponíveis quanto a identificação dos componentes eletrônicos que ainda precisavam ser adquiridos para permitir o funcionamento completo do sistema. Para auxiliar no levantamento dos componentes, utilizou-se a ferramenta de geração automática de BOM (*Bill of Materials*) disponibilizada pelo Altium Designer. A BOM consiste em uma lista detalhada de todos os componentes utilizados no circuito eletrônico, incluindo informações do componente, como encapsulamento, especificações técnicas e quantidade necessária, além de informar, em alguns casos, a fabricante e o código comercial. A utilização desse recurso foi fundamental para organizar o processo de aquisição dos materiais, reduzindo erros de compra e facilitando a conferência dos componentes durante a etapa de montagem.

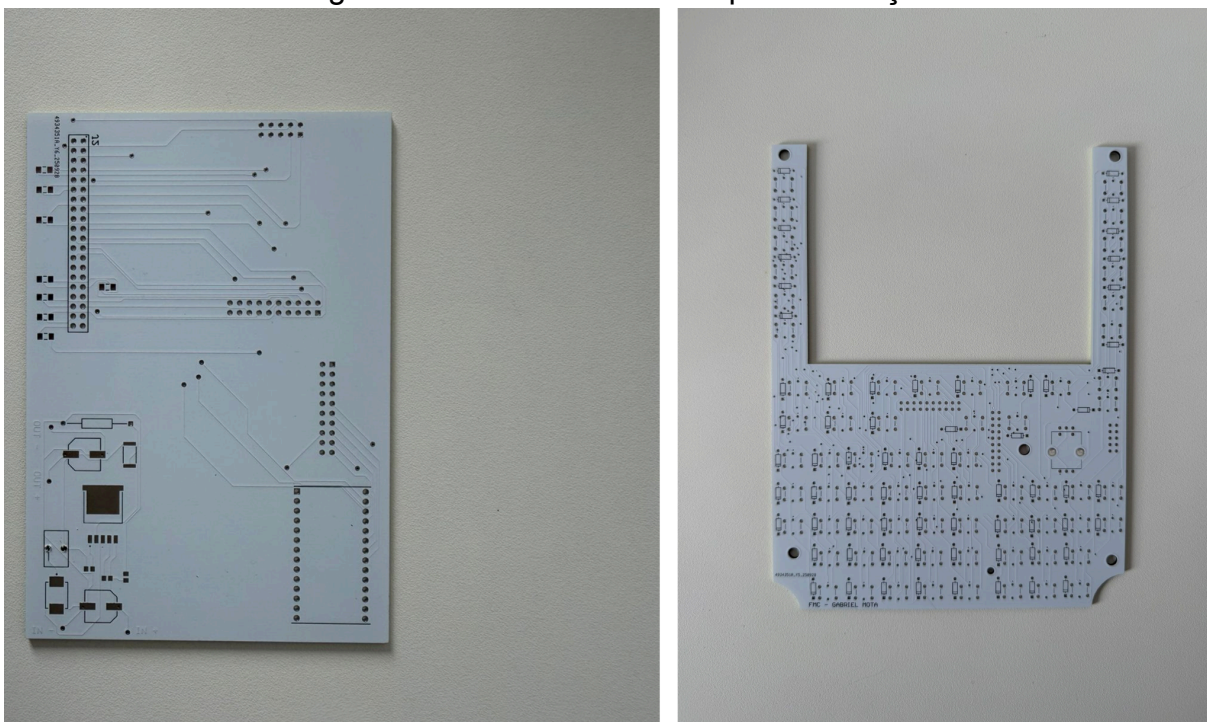
Parte dos componentes eletrônicos foi adquirida em fornecedores nacionais, como resistores, capacitores e conectores. Entretanto, alguns componentes específicos do projeto, como chaves de toque, componentes comprados em quantidade como os diodos, e o módulo HT16K33 por exemplo, foram importados da China devido à maior disponibilidade de estoque e custo reduzido em comparação ao mercado nacional. As PCBs, por sua vez, foram impressas em lote de 5 unidades, a quantidade mínima impressa pela JLCPCB. O valor total apresentado na Tabela 1, denominada tabela de custos, representa o valor gasto em *hardware* de uma unidade de protótipo. Na Figura 53, é possível visualizar as PCBs de controle e teclado após a fabricação.

Tabela 1 – Lista de materiais de *hardware* e custo.

Quantidade	Descrição	Valor Final (Em Reais)
1	Raspberry Pi 4 Model B	R\$588,05
1	Display AT056TN53	R\$368,97
1	Módulo HT16K33	R\$31,11
1	Encoder EC11	R\$45,19
68	Chave de Toque TS27	R\$103,97
68	Diodo 1N4148	R\$17,68
4	Resistores 10k ohm 1206 SMD 1/4W	R\$1,24
8	Resistores 330 ohm 1206 SMD 1/4W	R\$2,48
1	CI XL4015	R\$22,37
1	Capacitor Eletrolítico SMD 220uF 50V	R\$2,16
1	Capacitor Eletrolítico SMD 330uF 25V	R\$2,74
3	Capacitor Cerâmico 1uF 105 SMD	R\$0,39
1	Resistor 10k ohm 2512 1W	R\$0,63
1	Resistor 3,3k ohm 1W	R\$0,56
1	Indutor 47uH 5A	R\$27,79
1	Diodo Schottky B540 40V 5A	R\$13,49
2	Barramento Conexão Fêmea 2 x 10	R\$5,02
1	Barramento Conexão Fêmea 2 x 5	R\$1,64
1	Barramento Conexão Fêmea 2 x 20	R\$5,80
2	Barramento Conexão Fêmea 1 x 14	R\$7,74
1	Barramento Conexão Macho 2 x 5	R\$1,74
2	Barramento Conexão Macho 2 x 10	R\$5,37
1	Conjunto PCB (1 de teclado e 1 de controle)	R\$92,89
Valor Final		R\$1.349,02

Fonte: O autor (2026).

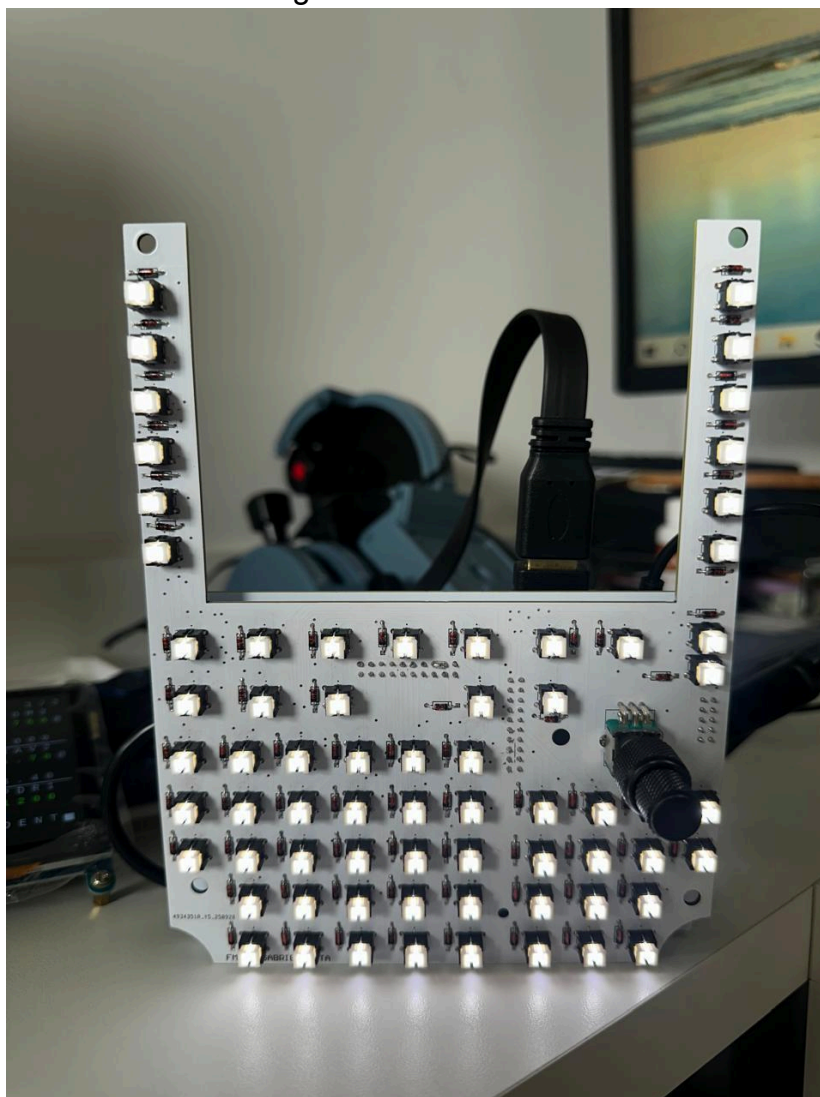
Figura 53 – PCBs recebidas após fabricação.



Fonte: O autor (2025).

Após o recebimento de todos os componentes e placas, iniciou-se a montagem dos circuitos. Para isso, foram utilizados equipamentos como estação de solda, sugador, estanho, pasta de solda e alicate, para que fosse possível realizar a montagem das placas. Para a limpeza dos circuitos, foi utilizado álcool isopropílico e um pincel anti-estático, com o intuito de eliminar quaisquer resíduos existentes no circuito. Após a montagem, com a utilização de um multímetro, realizou-se uma análise das trilhas, a fim de verificar se todas foram impressas corretamente, e verificar também a existência de curto-circuito em algum componente. Nenhum curto-circuito foi encontrado, porém, houve uma pequena falha na impressão, e algumas trilhas não foram inseridas na placa do teclado, no setor do *encoder*. Para corrigir o problema, foram soldados fios para fechar os circuitos os quais não continham a ligação correta prevista no diagrama esquemático. Após a correção e finalização, a PCB foi finalizada, conforme apresentado na Figura 54.

Figura 54 – PCB montada.



Fonte: O autor (2026).

Para que a montagem do circuito pudesse ser testada no Raspberry, foi necessário inserir ao código do servidor, sendo ele o principal arquivo de funcionamento do dispositivo, os comandos para acionar e controlar a matriz de LEDs com o HT16K33. Para isso, foi necessário habilitar o I2C no Raspberry, e na sequência, foi inserido ao código os comandos de acionamento via I2C dos pinos SDA e SCL do módulo, permitindo o acionamento dos LEDs sempre que o servidor estiver ativo. Também foi necessário inserir no código os comandos para a matriz do teclado. Previamente, no desenvolvimento da PCB do teclado, foram selecionados os pinos GPIOs do Raspberry para cada linha e coluna. Sendo assim, um mapeamento de linhas e colunas permitiu identificar as teclas, fazendo com que, ao

clicar uma tecla, o servidor a identifique e envie para o simulador o comando específico.

Para finalização do produto, foi necessário imprimir as *cases*, frontal e traseira, e os botões para cada tecla do teclado. A modelagem realizou-se através do *software* Fusion 360 desenvolvido pela Autodesk, e acessado com uma licença de estudante. Como citado anteriormente, a *case* frontal foi impressa em 3D através de um arquivo disponibilizado na *web*, e a partir dela, foi possível modelar o restante das peças necessárias para impressão. Inicialmente, foi impressa a *case* frontal apenas para teste, com baixa qualidade, para verificar encaixes e para que pudessem ser desenvolvidas as teclas e testado o encaixe da tela. Sendo assim, iniciou-se o desenvolvimento das teclas. Com o auxílio do modelo 3D da *case*, foi possível medir os espaços disponíveis para modelar as teclas. O primeiro modelo de tecla foi feito em peça única, com o filamento do corpo em preto e o filamento da letra em transparente, porém, o encaixe com chave de toque não funcionou, assim como o filamento transparente permitiu muita passagem de luz. Desse modo, foi dividida a tecla em duas partes, a base e o acabamento. A base foi feita de acordo com o modelo da tecla, alfabética, numérica, de navegação ou de seleção, cada uma com encaixe preciso na chave de toque. Os acabamentos das teclas possuíam a informação referente à tecla, de acordo com o teclado, e seu corpo era preto com o filamento das letras impressos com filamento branco, que trouxe um acabamento melhor, mesmo se tratando de um protótipo. Foram impressas as 68 teclas do teclado, divididas em base e acabamento, com filamento preto e branco do tipo PLA. O custo de impressão por tecla foi de R\$2,87 considerando o filamento e gasto energético da impressora Bambu Lab A1, totalizando R\$195,16 para a impressão de todas as teclas.

Após a impressão e montagem das teclas, foi desenvolvida uma nova versão da *case* frontal. Houve a necessidade devido ao fato de que o *display* a ser utilizado era maior do que era previsto, portanto, adicionou-se na parte superior uma extensão da *case* para que a tela se mantivesse encaixada. Além disso, foi necessário criar um rebaixo maior para a área útil do *display* para refinar o encaixe. Também foram criados os suportes de fixação da PCB do teclado, com espessura para pequenos parafusos, de 3mm. A primeira impressão com um acabamento melhor mostrou alguns pequenos ajustes a serem realizados, tanto no encaixe do *display* quanto nos suportes de fixação, e na segunda impressão corrigida, a *case*

frontal foi finalizada. A case traseira foi desenvolvida baseado nos tamanhos da case frontal, com uma base inferior maior e reta permitindo que o dispositivo fique em pé durante o uso. Na parte superior do aparelho, a case permitiu um acabamento fino ao produto. Por fim, foi necessário desenvolver um suporte interno para o *display* ficar preso, permitindo usar o produto mesmo sem a case traseira. As cases tiveram um custo de R\$64,95 cada, além do suporte interno para o *display* o qual também foi impresso, com um custo aproximado de R\$18,74. O custo total com impressão 3D para a montagem física do produto totalizou R\$343,80. Na Figura 55 é apresentado o protótipo montado na case, e em funcionamento. É possível visualizar na parte traseira que há apenas a conexão da fonte de alimentação, conforme indicado anteriormente.

Figura 55 – Protótipo montado.



Fonte: O autor (2026).

4.3 TESTES E VALIDAÇÃO

A etapa de testes teve início ainda durante o desenvolvimento do projeto. Cada fase de implementação, tanto de *hardware* quanto de *software*, foi acompanhada por pequenos testes e validações, realizados com o objetivo de garantir o funcionamento adequado dos sistemas antes de seguir avançando para as etapas seguintes. Essa abordagem permitiu que cada módulo desenvolvido fosse

validado individualmente, estabelecendo pontos seguros de retorno, em caso de falhas ou necessidades de modificações futuras. Dessa forma, funcionalidades relacionadas à comunicação com o simulador, renderização de informações e funcionamento de *hardware* foram constantemente validadas ao longo do desenvolvimento. Além das verificações realizadas durante as etapas intermediárias, foram também executados testes finais com o protótipo completamente integrado, visando avaliar o funcionamento conjunto dos sistemas, a fidelidade operacional e a estabilidade durante o uso.

Com o objetivo de verificar o funcionamento do sistema desenvolvido, foram realizados testes de comunicação, renderização, operação do *hardware* e estabilidade funcional do protótipo. Os testes tiveram como finalidade validar tanto o funcionamento individual dos módulos quanto a integração completa entre todos os componentes do dispositivo, de *hardware* e de *software*, e o simulador de voo. Os procedimentos de validação foram executados progressivamente, porém, após a montagem completa do protótipo, foram realizados testes completos de operação utilizando o simulador X-Plane em tempo real.

Um teste de comunicação foi realizado entre o simulador X-Plane e o protótipo do MCDU, com o objetivo de verificar o tempo de resposta na comunicação entre os sistemas e na atualização das informações recebidas e enviadas. Durante o teste, foram realizadas alterações em parâmetros do MCDU diretamente no simulador, observando-se o tempo necessário para as atualizações ocorrerem também no *display* do MCDU físico. Também foram realizados comandos para verificar o envio do MCDU físico para o simulador, verificando-se o tempo de resposta. Os resultados demonstraram atualização sem atrasos perceptíveis visualmente durante a operação normal do sistema.

4.3.1 Teste de Alimentação e Estabilidade Elétrica

O sistema de iluminação do teclado também foi testado, a fim de garantir o funcionamento integral dos LEDs presentes em cada tecla do dispositivo. Durante o teste, utilizando uma fonte de 12V 2.5A, verificou-se que a alimentação foi insuficiente para alimentar o sistema, visto que nem todos os LEDs acenderam e o Raspberry também acusou baixa potência de alimentação. Após a substituição da fonte para uma de 12V 5A, verificou-se que os LEDs acenderam com um pouco

mais de constância, porém ainda com algumas falhas. Porém a alimentação foi suficiente para o Raspberry, sem ocorrência de falhas.

Por conta da necessidade de trocar a fonte de alimentação após o teste dos LEDs, foi feita uma análise para realizar a substituição. Para isso, estimou-se a fonte a partir do consumo dos circuitos do produto. Considerou-se o consumo da placa controladora do *display*, do Raspberry e do módulo HT16K33 responsável pelos LEDs, os quais totalizam 68 embutidos nas chaves de toque. O consumo a ser considerado por LED é de aproximadamente 20mA, totalizando 1,36A de corrente necessária para o funcionamento completo dos LEDs.

$$I_{5V} = I_{Raspberry} + I_{LEDs} \quad (4)$$

$$I_{5V} = 3A + 1,36A \quad (5)$$

$$I_{5V} = 4,36A \quad (6)$$

A partir da descoberta da potência necessária para alimentar o sistema, foi necessário converter o valor para obter a corrente de operação em 12V. Sabe-se porém, de acordo com a documentação do circuito integrado XL4015, que a eficiência do conversor é de 87%, portanto, considerando-a, obtém-se o valor de potência corrigida, e assim, o valor da corrente para a fonte de 12V.

$$P_{5V} = 4,36A * 5V = 21,8W \quad (7)$$

$$P_{5V} = \frac{21,8W}{0,87} = 25,05W \quad (8)$$

$$I_{12V} = \frac{25,05W}{12V} = 2,09A \quad (9)$$

Após obter o valor da corrente necessária em 12V, deve-se somar o consumo da placa controladora do *display*, que, segundo a documentação, é de 12V 2A. Sendo assim, é possível concluir que é necessária a utilização de uma fonte de 12V 5A, atribuindo uma margem de tolerância, e seu custo deve ser adicionado ao custo geral do projeto. Após a substituição da fonte, realizou-se um teste de estabilidade

elétrica, com o intuito de verificar o funcionamento do sistema em conjunto com a nova fonte de energia. O dispositivo permaneceu conectado ao simulador em funcionamento por aproximadamente três horas. Durante o período de teste, percebeu-se que o Raspberry não notificou por insuficiência de alimentação e os LEDs funcionaram durante todo o período, bem como o *display*, que não apresentou qualquer anomalia. Ao final do teste, então, não foram observados desligamentos inesperados, falhas de comunicação ou instabilidades elétricas no sistema.

4.3.2 Testes Funcionais e Integrados do Protótipo

O teclado matricial foi submetido a testes individuais de acionamento, com o objetivo de validar o funcionamento de todas as teclas do dispositivo. Durante os testes, verificou-se a correta leitura dos comandos do teclado, bem como a ausência de problemas como *ghosting*. A estrutura mecânica das teclas, formada pelas peças impressas em impressora 3D, também demonstrou funcionamento adequado, sem qualquer travamento ou fragilidade.

Foram realizados testes de renderização das páginas do MCDU, verificando a integridade das informações exibidas, alinhamento dos textos, atualização dinâmica das informações obtidas do simulador e representação dos estilos visuais. Os testes demonstraram funcionamento adequado do sistema, sendo necessárias apenas pequenas correções na renderização de alguns símbolos e pequenos ajustes relacionados ao tamanho das informações.

Após a integração completa de todo o sistema, realizou-se um voo completo utilizando o protótipo desenvolvido em conjunto com o simulador X-Plane. Durante o voo, foram executadas operações de navegação, alteração de páginas, inserção de dados, troca de frequências de rádio, formando uma interação completa com o sistema. O teste permitiu validar o funcionamento integrado do dispositivo em condições de uso durante sua utilização com o ambiente de simulação, demonstrando a viabilidade do sistema desenvolvido.

5 RESULTADOS

Após todas as etapas de desenvolvimento, foi possível obter um protótipo funcional de um *Main Control Display Unit* compatível com a aeronave Embraer E-195, para utilização com simuladores de voo X-Plane. O dispositivo desenvolvido foi capaz de reproduzir as funcionalidades do dispositivo real, em conjunto com o simulador de voo, permitindo a interação entre o ambiente virtual e o *hardware*. A comunicação entre o simulador de voo X-Plane e o protótipo ocorreu de forma satisfatória, possibilitando o recebimento de informações para exibição no *display*, bem como o envio de comandos para o simulador, através do teclado desenvolvido. Os testes de desenvolvimento e uso demonstraram atualização das informações em tempo real, sem atrasos perceptíveis durante a operação do sistema.

O sistema de renderização foi capaz de reproduzir as páginas do MCDU da aeronave E-195 incluindo textos, estilos visuais e elementos característicos da interface original. A utilização dos métodos de comunicação escolhidos, como o WebSocket, permitiu a construção de um modelo flexível e compatível com comunicação bidirecional simultânea.

5.1 RESULTADOS DOS TESTES E VALIDAÇÃO

Os testes realizados permitiram validar o funcionamento dos diferentes sistemas desenvolvidos para o protótipo. Em relação à comunicação, foi possível realizar tanto o recebimento das informações fornecidas pelo simulador quanto o envio dos comandos realizados através do teclado físico, estabelecendo a comunicação bidirecional necessária para o funcionamento do dispositivo. Durante os testes realizados, não foram observadas falhas que impedissem a utilização do sistema, sendo possível realizar a alteração das informações tanto através do simulador quanto pelo protótipo desenvolvido.

Em relação ao *hardware*, após a substituição da fonte de alimentação para o modelo de 12V e 5A, o sistema apresentou funcionamento estável durante o período de teste de aproximadamente três horas. Durante esse período, o Raspberry não apresentou notificações relacionadas à insuficiência de alimentação, enquanto o *display* e o sistema de iluminação permaneceram em funcionamento. Ao final do

teste, não foram observados desligamentos inesperados, falhas de comunicação ou instabilidades elétricas.

Os testes realizados no teclado demonstraram o correto funcionamento das teclas e dos comandos associados, sem a ocorrência de *ghosting* durante as verificações realizadas. A estrutura mecânica desenvolvida também apresentou funcionamento adequado, permitindo o acionamento das teclas sem travamentos. Para a renderização, foi possível reproduzir as páginas do MCDU, incluindo as informações recebidas do simulador e os diferentes estilos utilizados na interface, sendo necessárias apenas pequenas correções relacionadas à representação de símbolos e ao tamanho das informações exibidas.

Por fim, o teste realizado durante um voo completo permitiu avaliar o funcionamento conjunto do protótipo. Durante a utilização, foram realizadas operações de navegação, alteração de páginas, inserção de informações e troca de frequências de rádio através do dispositivo desenvolvido. Dessa forma, foi possível validar de maneira integrada a comunicação, o teclado, a interface gráfica e o *hardware* desenvolvido durante a utilização do sistema no ambiente de simulação.

O funcionamento integrado do protótipo durante sua utilização com o simulador de voo pode ser visualizado no vídeo disponibilizado no Apêndice C.

5.2 ANÁLISE DE CUSTOS

Quando comparado às soluções comerciais atualmente disponíveis, o protótipo apresenta como principal diferencial a compatibilidade com aeronaves Embraer E-195 equipadas com o modelo de MCDU fabricado pela Honeywell, segmento no qual não foram encontradas soluções dedicadas disponíveis para aquisição durante o desenvolvimento deste trabalho. Além disso, o custo estimado de desenvolvimento mostrou-se inferior ao valor estimado para aquisição e importação de equipamentos similares disponíveis no mercado. O valor final para desenvolvimento do protótipo, apresentado na Tabela 2, foi de aproximadamente R\$1.738,32, em contrapartida, o valor para importar um modelo de MCDU, não específico para o Embraer, é de R\$2.158,60. Sendo assim, além da redução de custos, o protótipo apresenta compatibilidade específica com aeronaves Embraer E-195 com os dispositivos Honeywell.

Tabela 2 – Custo total do protótipo.

Quantidade	Descrição	Valor Final (Em Reais)
1	Conjunto de <i>hardware</i>	R\$1.349,02
1	Fonte 12V / 5A	R\$45,50
1	Custos de impressão	R\$343,80
Valor Final		R\$1.738,32

Fonte: O autor (2026).

Conforme apresentado na Tabela 2, o custo total para desenvolvimento de uma unidade do protótipo foi de R\$1.738,32. Desse valor, R\$1.349,02 correspondem ao conjunto de *hardware*, enquanto R\$45,50 correspondem à fonte de alimentação utilizada após os testes realizados e R\$343,80 aos custos relacionados à impressão 3D das peças utilizadas na construção do dispositivo. Dessa forma, observa-se que o conjunto de *hardware* representa a maior parcela do custo total para construção do protótipo.

5.3 COMPARAÇÃO COM O MCDU DE REFERÊNCIA

Para fins de comparação, a Figura 56 apresenta o protótipo do MCDU, localizado à esquerda, e o MCDU presente nas aeronaves Embraer, localizado à direita.

Figura 56 – Comparação com modelo real.



Fonte: O autor (2026).

A partir da figura acima, compara-se o protótipo desenvolvido com o modelo real do MCDU da Honeywell presente nas aeronaves Embraer. A fotografia do modelo original foi tirada durante uma visita à cabine de comando da aeronave Embraer E-195 E2 de matrícula PS-AEG após autorização da tripulação. A captura tornou-se fundamental para comparação entre o dispositivo desenvolvido e o modelo real, apresentando uma adequada compatibilidade.

6 CONCLUSÃO

A partir do desenvolvimento deste trabalho, foi possível realizar o projeto, construção e validação de um protótipo funcional de um *Main Control Display Unit*, baseado no equipamento presente nas aeronaves Embraer E-195 e destinado à utilização com simuladores de voo X-Plane. O desenvolvimento permitiu integrar diferentes áreas da Engenharia Elétrica, envolvendo sistemas embarcados, eletrônica, desenvolvimento de placas de circuito impresso, programação e comunicação entre sistemas, resultando em um dispositivo físico capaz de interagir em tempo real com o simulador de voo.

Durante o desenvolvimento, buscou-se compreender inicialmente o funcionamento do MCDU real e sua relação com o sistema de gerenciamento de voo da aeronave. A partir dessas informações, foi possível comparar as características presentes no dispositivo real e no simulador, e estipular as características necessárias para que o protótipo pudesse reproduzir as principais funções do dispositivo. Além das características físicas, como a disposição do teclado, display e teclas de seleção, foi necessário compreender a forma como as informações eram apresentadas e como a interação realizada pelo usuário deveria ser transmitida ao simulador de voo.

Na etapa de desenvolvimento de *software*, a obtenção das informações disponibilizadas pelo X-Plane permitiu estabelecer uma comunicação entre o simulador e um sistema externo. A utilização dos *datarefs* e *commands*, em conjunto com o ExtPlane, possibilitou receber as informações do MCDU presente no ambiente virtual e enviar comandos realizados através do dispositivo físico. Para isso, foi desenvolvida uma arquitetura de comunicação baseada em TCP/IP e WebSocket, permitindo que as informações obtidas do simulador fossem processadas e posteriormente apresentadas na interface desenvolvida para o protótipo.

A interface gráfica desenvolvida permitiu reproduzir as informações apresentadas no MCDU do simulador, incluindo a posição dos caracteres, estilos e diferentes páginas do equipamento. Durante o desenvolvimento, problemas relacionados à organização das informações, troca de páginas e representação dos caracteres foram identificados e corrigidos, permitindo que a interface final apresentasse as informações recebidas de maneira compatível com o dispositivo

presente no simulador. Dessa forma, foi possível estabelecer o caminho de comunicação desde a obtenção dos dados no X-Plane até sua exibição no *display* físico do protótipo.

Além do recebimento das informações, o desenvolvimento do teclado possibilitou realizar o caminho contrário da comunicação. A matriz de teclas desenvolvida permitiu identificar os acionamentos realizados pelo usuário e enviá-los ao simulador, fazendo com que as ações executadas no protótipo fossem reproduzidas no MCDU virtual. Sendo assim, a integração entre o recebimento das informações e o envio dos comandos permitiu obter uma comunicação bidirecional entre o hardware desenvolvido e o simulador de voo.

Para a construção do hardware, a utilização do Raspberry Pi 4 Model B permitiu concentrar o processamento necessário para execução do sistema e comunicação com o simulador, além da geração da interface utilizada no display. As placas de circuito impresso desenvolvidas possibilitaram integrar o teclado, sistema de iluminação, alimentação e demais componentes necessários para o funcionamento do protótipo. Durante a etapa de montagem e testes, foram identificadas necessidades de alterações no projeto, como a adequação da alimentação elétrica, resultando na utilização de uma fonte de 12V e 5A capaz de fornecer a potência necessária para o funcionamento do conjunto.

Os testes realizados permitiram verificar o funcionamento do sistema em diferentes condições. O teste de estabilidade elétrica demonstrou que, após a substituição da fonte, o protótipo permaneceu em funcionamento por aproximadamente três horas sem apresentar desligamentos inesperados, falhas de comunicação ou instabilidades elétricas. O teclado matricial também apresentou funcionamento adequado durante os testes individuais das teclas, sem a ocorrência de *ghosting*, enquanto os testes de renderização permitiram verificar a atualização das informações, alinhamento dos textos e representação dos estilos visuais.

Por fim, a realização de um voo completo utilizando o protótipo integrado ao X-Plane permitiu avaliar o sistema em uma condição de utilização conjunta. Durante o teste, foram realizadas operações de navegação, alteração de páginas, inserção de dados e troca de frequências de rádio, permitindo validar o funcionamento integrado do dispositivo. A comunicação ocorreu de forma satisfatória, com atualização das informações em tempo real e sem atrasos perceptíveis durante a operação.

Em relação ao custo de desenvolvimento, o protótipo apresentou valor total de R\$1.738,32, considerando o conjunto de *hardware*, a fonte de alimentação e os custos relacionados à impressão 3D. O valor obtido foi inferior ao valor estimado de R\$2.158,60 para aquisição e importação de um equipamento semelhante analisado durante o desenvolvimento deste trabalho. Além da diferença de custo, o protótipo desenvolvido possui como característica a compatibilidade específica com o MCDU utilizado nas aeronaves Embraer E-195 equipadas com o modelo da Honeywell, para o qual não foram encontradas soluções dedicadas disponíveis para aquisição durante o desenvolvimento do projeto.

Dessa forma, os resultados obtidos demonstram a viabilidade técnica do desenvolvimento de dispositivos embarcados para simuladores de voo, utilizando componentes eletrônicos acessíveis e ferramentas amplamente disponíveis. O projeto atingiu os objetivos propostos, incluindo o projeto, construção e validação do MCDU, resultando em um protótipo funcional capaz de reproduzir as principais características funcionais do MCDU utilizado nas aeronaves Embraer E-195.

Além dos resultados obtidos, o projeto demonstrou aplicação prática de conceitos de sistemas embarcados, eletrônica e programação, integrando os conhecimentos adquiridos ao longo da graduação.

6.1 LIMITAÇÕES DO PROJETO

Apesar dos resultados obtidos, algumas limitações relacionadas ao desenvolvimento e à validação do protótipo devem ser consideradas. Os testes realizados tiveram como objetivo principal verificar o funcionamento dos sistemas desenvolvidos e sua integração com o simulador de voo, sendo realizados no ambiente utilizado durante o desenvolvimento do projeto. Dessa forma, não foram realizados testes com diferentes configurações de *hardware*, redes ou versões do simulador, limitando a validação às condições utilizadas durante o desenvolvimento deste trabalho.

Em relação à comunicação entre o protótipo e o simulador, a avaliação do tempo de resposta foi realizada de maneira qualitativa, através da observação do comportamento do sistema durante sua utilização. Não foram realizadas medições quantitativas de latência entre o envio ou recebimento das informações e sua

respectiva atualização. Portanto, as considerações relacionadas ao tempo de resposta do sistema estão limitadas à percepção durante os testes realizados.

Os testes de estabilidade elétrica também foram realizados durante um período limitado de aproximadamente três horas. Embora não tenham sido observados desligamentos inesperados, falhas de comunicação ou instabilidades elétricas durante esse período, não foram realizados testes de longa duração que permitissem avaliar o comportamento do dispositivo durante períodos prolongados de utilização.

Por fim, a validação funcional foi realizada pelo próprio desenvolvedor durante os testes e durante a realização do voo completo no ambiente de simulação. Não foram realizados testes com diferentes usuários ou avaliações relacionadas à ergonomia e experiência de utilização do dispositivo. Além disso, o protótipo desenvolvido é destinado exclusivamente à utilização em ambiente de simulação de voo doméstico, não possuindo qualquer finalidade de utilização em aeronaves reais.

6.2 PROPOSTAS PARA TRABALHOS FUTUROS

Durante o desenvolvimento do projeto, foram identificadas oportunidades de melhoria, que podem ser consideradas em versões futuras do dispositivo, visando aumentar sua robustez, reduzir dimensões e facilitar processos de montagem e manutenção.

Uma das principais melhorias está relacionada à arquitetura eletrônica adotada. O uso de duas placas de circuito impresso distintas permitiu simplificar o desenvolvimento e os testes durante a fase de prototipagem, porém aumentou o volume interno necessário para a colocação dos componentes. Em uma segunda versão futura, recomenda-se a integração dos circuitos em uma única PCB, reduzindo a quantidade de conexões internas, simplificando a montagem e reduzindo a espessura final. Outro caminho, seria a melhoria da disposição dos componentes e conectores da PCB de controle, melhorando a distribuição dos componentes, o roteamento das trilhas e o acesso às conexões. Durante a montagem do protótipo, observou-se também a necessidade de pequenos recortes na PCB de controle devido à interferência com pontos de fixação da estrutura, situação que pode ser eliminada após uma revisão da PCB e do *layout* final do produto.

Na parte mecânica, recomenda-se a implementação de suportes dedicados para a fixação do *display* diretamente na estrutura frontal, proporcionando maior resistência e melhor alinhamento. Os pontos de fixação da PCB do teclado também devem ser ampliados, distribuindo melhor os esforços mecânicos e aumentando a rigidez. Pequenos ajustes dimensionais nas teclas de seleção lateral podem ser realizados para aprimorar o encaixe.

O projeto apresenta potencial para expansão funcional. Entre as possibilidades identificadas, está a compatibilidade com outros simuladores de voo, como o Microsoft Flight Simulator, implementação do controle de brilho do *display* e desenvolvimento de módulos complementares de *cockpits* para simuladores de voo.

A tecnologia segue evoluindo, e trazendo consigo melhorias em diversas áreas. Na aviação, observa-se que as novas aeronaves estão cada vez mais integradas com dispositivos eletrônicos modernos. Atualmente, já existem aeronaves que utilizam telas sensíveis ao toque para interação com sistemas embarcados, reduzindo a dependência de teclados físicos. Dessa forma, futuras gerações de aeronaves Embraer poderão incorporar novas versões de MCDU ou até mesmo diferentes conceitos de interação, acompanhando a evolução tecnológica.

REFERÊNCIAS

- ADAFRUIT. **HT16K33 datasheet**. 2025. Disponível em: <https://cdn-shop.adafruit.com/datasheets/ht16K33v110.pdf>. Acesso em: 12 dez. 2025.
- ADRENALINE. **SIMULADOR DE 12 MILHÕES DE DÓLARES! Testamos um Full Flight nível Delta**. 2021. Disponível em: <https://www.adrenaline.com.br/hardware/simulador-de-12-milhoes-de-dolares-testamos-um-full-flight-nivel-delta/>. Acesso em: 10 mar. 2026.
- AERO MAGAZINE. **Voar em simuladores de voo domésticos economiza até 5,5 horas de voo em instrução real**. 2024. Disponível em: <https://aeromagazine.uol.com.br/artigo/voar-em-simuladores-de-voo-domesticos-economiza-ate-55-horas-de-voo-em-instrucao-real.html>. Acesso em: 10 set. 2025.
- AEROIN. **Flight Simulator bate 1 milhão em vendas**. 2020. Disponível em: <https://aeroin.net/flight-simulator-bate-1-milhao-vendas/>. Acesso em: 29 nov. 2025.
- AGÊNCIA NACIONAL DE AVIAÇÃO CIVIL (ANAC). **Regulamento Brasileiro da Aviação Civil nº 60 (RBAC 60)**. 2020. Disponível em: <https://www.anac.gov.br/assuntos/legislacao/legislacao-1/rbha-e-rbac/rbac/rbac-60>. Acesso em: 12 set. 2025.
- AIRBUS. **Fly-By-Wire Flight Control System**. Airbus, 2022. Disponível em: <https://www.airbus.com/en/newsroom/stories/2022-06-safety-innovation-1-fly-by-wire-fbw>. Acesso em: 02 out. 2025.
- ALURA. **Framework: o que é e pra que serve essa ferramenta?** 2021. Disponível em: https://www.alura.com.br/artigos/framework?srsId=AfmBOooKjWxUG1PWpYU2rmsyYI06ynSYOe2YWMhTaHUTF_uZy4FApUst. Acesso em: 03 fev. 2026.
- BALMUS, Elena. **Avionics and ATC Technology for Mission Control**. INCAS Bulletin, v. 7, n. 3, p. 161–171, 2015.
- BARR, Michael. **Programming Embedded Systems in C and C++**. 1. ed. Sebastopol: O'Reilly, 1999.
- BBC. **Afinal, quem inventou o avião? A resposta provavelmente não é a que você pensou**. 2025a. Disponível em: <https://www.bbc.com/portuguese/articles/c1w3l9ej8pxo>. Acesso em: 19 set. 2025.
- BBC. **A empresa brasileira de alta tecnologia que pode ser alvo de tarifas de Trump**. 2025b. Disponível em: <https://www.bbc.com/portuguese/articles/c80pnz99y29o>. Acesso em: 10 set. 2025.
- CAE. **Flight Training Devices**. CAE Inc., 2025a. Disponível em: <https://www.cae.com/civil-aviation/aviation-simulation-equipment/flight-training-devices/>. Acesso em: 01 nov. 2025.

CAE. **Full-Flight Simulators**. CAE Inc., 2025b. Disponível em: <https://www.cae.com/civil-aviation/aviation-simulation-equipment/full-flight-simulators>. Acesso em: 01 nov. 2025.

COLLINSON, Richard. **Introduction to Avionics Systems**. 3. ed. Dordrecht: Springer, 2014.

CPFLIGHT. **MCDU Airbus A320**. 2025. Disponível em: https://cpflight.com/prodotto/mcd�_a320/. Acesso em: 21 out. 2025.

DATAREF TOOL. **DataRef Tool for X-Plane**. 2014. Disponível em: <https://datareftool.com/>. Acesso em: 10 set. 2025.

DOTSHA747. **Pi-XPlane-FMC-CDU**. GitHub Repository. 2017. Disponível em: <https://github.com/dotsha747/Pi-XPlane-FMC-CDU/tree/master>. Acesso em: 08 ago. 2025.

DUNCAN AVIATION. **Do not delay. Your Honeywell MCDUs are facing impeding component obsolescence**. 2024. Disponível em: <https://www.duncanaviation.aero/intelligence/do-not-delay-your-honeywell-mcdus-are-facing-impending-component-obsolescence>. Acesso em: 21 out. 2025.

ELECROW. **XL4015 Step-Down DC-DC Converter Datasheet**. 2025. Disponível em: https://www.elecrow.com/download/XL4015_datasheet.pdf?srsId=AfmBOoptJ6Rv-RkUHuvXSubfynNGAbGrID3595uDe803v2McthCjDF7. Acesso em: 10 jan. 2026.

EMBARCADOS. **Raspberry Pi: teclado matricial 4x4 com Python**. 2015. Disponível em: <https://embarcados.com.br/raspsberrypi-teclado-4x4-com-python/>. Acesso em: 08 fev. 2026.

EMBRAER. **E-195**. Embraer, 2023. Disponível em: <https://www.embraer.com/e-jets/e195/pt/>. Acesso em: 20 set. 2025.

EXPRESS. **Basic Routing. Express.js Documentation**. 2024. Disponível em: <https://expressjs.com/pt-br/5x/starter/basic-routing/>. Acesso em: 02 fev. 2026.

FEDERAL AVIATION ADMINISTRATION (FAA). **Glass Cockpit Systems**. FAA, 2013. Disponível em: https://www.faa.gov/documentlibrary/media/advisory_circular/ac_25-11b.pdf. Acesso em: 18 out. 2025.

FIGUEIREDO, Helosman V.; SAOTOME, Osamu. **Simulation Platform for Quadricopter: Using Matlab/Simulink and X-Plane**. Instituto Tecnológico de Aeronáutica, 2012. Disponível em: https://www.sige.ita.br/edicoes-antiores/2012/st/IV_2.pdf. Acesso em: 03 nov. 2025.

FONSECA, Matheus Muniz da; MONTEIRO, Raul Francé. **Simuladores de voo: a importância do seu uso para a formação do aeronauta**. Revista Brasileira de Aviação Civil & Ciências Aeronáuticas, v. 2, n. 1, p. 79–104, 2022.

FORÇA AÉREA BRASILEIRA (FAB). **História dos simuladores de voo**. FAB, 2020. Disponível em: <https://www2.fab.mil.br/musal/index.php/projeto-av-hist?view=article&id=470:os->. Acesso em: 20 set. 2025.

GAMALYTIC. **X-Plane 12 - Steam Stats**. 2025. Disponível em: <https://gamalytic.com/game/269950>. Acesso em: 30 nov. 2025.

GE AVIATION. **Flight Management System GE 2907C1**. 2022. Disponível em: <https://www.geaerospace.com/sites/default/files/737C1-FMC-Datasheet.pdf>. Acesso em: 10 out. 2025.

GUIA DO AVIADOR. **Superfícies de controle de uma aeronave**. 2017. Disponível em: <https://guiadoaviador.wordpress.com/2017/03/>. Acesso em: 01 out. 2025.

HONEYWELL. **NG Flight Management System for Embraer 170/175/190/195 Pilot's Guide**. Honeywell Aerospace, 2015.

HONEYWELL. **DU 1080 Cockpit Display**. Honeywell Aerospace, 2023. Disponível em: <https://www.honeywellaerospace.com/us/en/products-and-services/products/cabin-and-cockpit/avionics/cockpit-and-flight-displays/du-1080-cockpit-display>. Acesso em: 20 jan. 2026.

HONEYWELL. **Airbus Pegasus FMS for the Airbus A330 and A320 series**. Honeywell Aerospace, 2016. Disponível em: https://aerospace.honeywell.com/content/dam/aerobt/en/documents/learn/platforms/brochures/C61-1647-000-000_ATR_TechnicalSummary_PegasusFMS_Airbus_A330_A320.pdf. Acesso em: 21 out. 2025.

IBM. **Socket Programming**. 2022. IBM Documentation. Disponível em: <https://www.ibm.com/docs/pt-br/i/7.5.0?topic=communications-socket-programming>. Acesso em: 02 fev. 2026.

INSTITUTO TECNOLÓGICO DE AERONÁUTICA (ITA). **Sistemas aviônicos e barramentos aeronáuticos**. ITA, 2012.

INTERNET ENGINEERING TASK FORCE (IETF). **RFC 4648: The Base16, Base32 and Base64 Data Encodings**. IETF, 2006. Disponível em: <https://datatracker.ietf.org/doc/html/rfc4648>. Acesso em: 30 jan. 2026.

JON PEDDIE RESEARCH (JPR). **Flight Simulator 2020 Hardware Impact Study**. JPR, 2020. Disponível em: <https://www.jonpeddie.com/news/microsoft-flight-simulator-2020-will-stimulate-billions-of-dollars-in-pc-ga/>. Acesso em: 10 set. 2025.

KAYTON, Myron. **Avionics Navigation Systems**. 2. ed. New York: John Wiley & Sons, 2003.

LAMINAR RESEARCH. **X-Plane 12**. Laminar Research, 2025a. Disponível em: <https://www.x-plane.com>. Acesso em: 21 ago. 2025.

LAMINAR RESEARCH. **How X-Plane Works**. Laminar Research, 2025b. Disponível em: <https://www.x-plane.com/desktop/how-x-plane-works/>. Acesso em: 01 dez. 2025.

LEVVEL. **Microsoft Flight Simulator Statistics**. 2023. Disponível em: <https://levvel.com/microsoft-flight-simulator-statistics/>. Acesso em: 29 nov. 2025.

MATSUURA, John. **Flight Simulation**. Blue Ridge Summit: TAB Books, 1995.

MDN WEB DOCS. **WebSocket API**. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/WebSocket>. Acesso em: 04 fev. 2026.

MICROSOFT. **Microsoft Flight Simulator**. Microsoft Corporation, 2024. Disponível em: <https://www.flightsimulator.com>. Acesso em: 08 ago. 2025.

MICROSOFT. **Flight Model Physics**. Microsoft Corporation, 2023. Disponível em: https://docs.flightsimulator.com/html/Samples_And_Tutorials/Primers/Flight_Model_Physics.htm. Acesso em: 01 dez. 2025.

MOIR, Ian; SEABRIDGE, Allan. **Military Avionics Systems**. Chichester: John Wiley & Sons, 2006.

NASA. **Full Authority Digital Engine Control (FADEC)**. National Aeronautics and Space Administration, 2007. Disponível em: <https://ntrs.nasa.gov/api/citations/20070010763/downloads/20070010763.pdf>. Acesso em: 11 out. 2025.

NASA. **F-8 Digital Fly-By-Wire Aircraft**. National Aeronautics and Space Administration, 2021. Disponível em: https://www.nasa.gov/wp-content/uploads/2021/09/120282main_fs-024-dfrc.pdf. Acesso em: 02 out. 2025.

NODE.JS. **NET**. 2024a. Disponível em: <https://nodejs.org/api/net.html>. Acesso em: 02 out. 2025.

NODE.JS. **Buffer API Documentation**. 2024b. Disponível em: <https://nodejs.org/api/buffer.html>. Acesso em: 02 out. 2025.

PLAYTRACKER. **Microsoft Flight Simulator 2024**. 2025. Disponível em: <https://playtracker.net/insight/game/101017>. Acesso em: 30 nov. 2025.

RASPBERRY. **Raspberry Pi 4 Model B**. 2019. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Acesso em: 08 nov. 2025.

ROCKWELL COLLINS. **FMS-4200 Flight Management System**. Rockwell Collins, 1999.

RTCA. **DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations**. RTCA, 2005.

SKYBRARY. **Flight Management System (FMS)**. 2023. Disponível em: <https://skybrary.aero/articles/flight-management-system-fms>. Acesso em: 30 set. 2025.

STORMBIRDS. **Microsoft Flight Simulator 2020 four years later a retrospective**. 2024. Disponível em: <https://stormbirds.blog/2024/09/02/microsoft-flight-simulator-2020-four-years-later-a-retrospective/>. Acesso em: 29 nov. 2025.

TECNODEFESA. **O projeto T-27M - O Tucano “Modernizado”**. 2018. Disponível em: <https://tecnodefesa.com.br/o-projeto-t-27m-tucano-modernizado/>. Acesso em: 10 out. 2025.

TOTH, Eduardo. **FMC Project**. 2018. Disponível em: <https://toth.eti.br/2018/07/09/day-0-an-idea/>. Acesso em: 08 ago. 2025.

VIDA DE SILÍCIO. **Teclado matricial e multiplexação**. 2017. Disponível em: <https://portal.vidadesilicio.com.br/teclado-matricial-e-multiplexacao/>. Acesso em: 08 fev. 2026.

VRANKI. **ExtPlane**. 2018. GitHub Repository. Disponível em: <https://github.com/vranki/ExtPlane>. Acesso em: 07 set. 2025.

WATKINS, Christopher B.; WALTER, Randy. **Transitioning from Federated Avionics Architectures to Integrated Modular Avionics**. AFRL, 2006.

WINCTRL. **Airbus MCDU Hardware Panel**. 2026. Disponível em: <https://ea.winctrl.com/view/goods-details.html?id=945>. Acesso em: 04 abr. 2026.

X-CRAFTS. **Embraer E195 for X-Plane 12**. Disponível em: <https://www.xcrafts.com/ejets-family>. Acesso em: 10 ago. 2025.

APÊNDICE A – Circuito Elétrico PCB Teclado



APÊNDICE C – Vídeo de Demonstração de Funcionamento do Protótipo

O vídeo disponibilizado neste apêndice apresenta o funcionamento do protótipo desenvolvido, demonstrando a interação entre o MCDU físico e o simulador de voo X-Plane, incluindo a atualização das informações exibidas no *display* e o envio de comandos por meio do teclado físico.

Vídeo disponível em: <https://youtu.be/zKu22vTFc1U>

QR Code:

