

João Victor Luiz de Andrade

**Desenvolvimento de Hardware e Software para
Depurador Multiprotocolo Multiplexador**

UBERLÂNDIA

2024

João Victor Luiz de Andrade

Desenvolvimento de Hardware e Software para Depurador Multiprotocolo Multiplexador

Trabalho de Conclusão de Curso da
Engenharia de Controle e Automação da
Universidade Federal de Uberlândia - UFU -
Câmpus Santa Mônica, como requisito para a
obtenção do título de Graduação em
Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU
Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Marcelo Barros de Almeida

UBERLÂNDIA

2024

Luiz de Andrade, João Victor
Desenvolvimento de Hardware e Software para Depurador Multiprotocolo
Multiplexador / **João Victor Luiz de Andrade. - UBERLÂNDIA, 2024-**
61 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Marcelo Barros de Almeida

Trabalho de Conclusão de Curso - Universidade Federal de Uberlândia - UFU Faculdade
de Engenharia Elétrica . **2017.**
Inclui bibliografia.

1. Sistemas embarcados. 2. Engenharia de testes. 2. Desenvolvimento de hardware e
Software. I. Marcelo Barros de Almeida. II. Universidade Federal de Uberlândia. III.
Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

João Victor Luiz de Andrade

Desenvolvimento de Hardware e Software para Depurador Multiprotocolo Multiplexador

Trabalho de Conclusão de Curso em Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, apresentado como requisito parcial para a obtenção do título de Bacharel em Engenharia de Controle e Automação.

Trabalho aprovado em de 2024

Comissão examinadora:

Prof. Dr. Marcelo Barros de Almeida
Orientador

Prof. Dr. Daniel Pereira de Carvalho
Membro Avaliador

Prof. Dr. Renato Ferreira Fernandes Júnior
Membro Avaliador

UBERLÂNDIA
2024

Agradecimentos

Em primeiro lugar, agradeço ao meu pai, Martines, à minha mãe, Claudia, e aos meus irmãos, Bruno e Luiz Gustavo, pela amizade, apoio constante e por sempre acreditarem em mim ao longo de toda a minha formação.

Expresso minha gratidão a Maxence Theron, engenheiro de validação eletrônica analógica na equipe MDG-GPM da STMicroelectronics em Grenoble, por me oferecer a oportunidade de realizar este trabalho. Agradeço também a Remy Quaglino, membro da equipe de validação, por sua orientação em design de hardware e pelo auxílio prestado ao longo do desenvolvimento do projeto. Estendo o meu agradecimento a toda equipe de validação por me acolherem e me propiciarem um ambiente de trabalho excelente.

Ao meu tutor de estágio, Luong-Viêt Phung, manifesto sincero agradecimento por acompanhar o desenvolvimento deste trabalho, por seu compromisso e pelas observações criteriosas que contribuíram significativamente para a qualidade final.

Expresso ainda a minha gratidão ao professor Marcelo Barros, orientador deste trabalho, por sua orientação e suporte inestimáveis ao longo do curso.

Agradeço aos meus amigos da turma 95 de Engenharia Elétrica pela amizade e apoio durante o curso, com destaque especial ao grupo “Fliperama”. Meu reconhecimento também se dirige aos professores e servidores da Universidade Federal de Uberlândia, cujo trabalho e dedicação são fundamentais para o funcionamento da instituição e para a formação de seus estudantes.

Por fim, agradeço à CAPES por fomentar a educação e possibilitar a realização do meu intercâmbio. Sem o apoio do programa BRAFITEC, a execução deste projeto não teria sido possível.

Resumo

O processo de desenvolvimento de dispositivos eletrônicos é complexo e, para garantir que funcionem conforme o esperado, são necessários procedimentos rigorosos de teste e validação. Esses procedimentos enfrentam desafios significativos, especialmente ao envolver múltiplos componentes que exigem protocolos de comunicação variados para se integrarem à bancada de testes. Assim, durante o teste e validação de dispositivos, é necessário utilizar métodos e ferramentas flexíveis para lidar com as especificidades de cada dispositivo. Neste contexto, este trabalho propõe o desenvolvimento de um depurador multiplexador multiprotocolo, projetado para facilitar o teste de dispositivos em ambientes de validação complexos. O depurador, que integra tanto hardware quanto software dedicados, adapta-se à interface de comunicação do dispositivo alvo sem a necessidade de recabeamento, permitindo testes robustos e eficientes com um único dispositivo. O presente documento descreve detalhadamente o processo de concepção, desenvolvimento e integração do módulo, além de documentar o fluxo de trabalho adotado por uma empresa de referência na área, oferecendo uma visão abrangente das práticas e técnicas utilizadas no setor.

Palavras-chave: Sistemas embarcados, Engenharia de testes, Desenvolvimento de hardware e Software.

Abstract

The process of developing electronic devices is complex, and to ensure they function as expected, rigorous testing and validation procedures are required. These procedures face significant challenges, especially when involving multiple components that require varied communication protocols to communicate with the test bench. Thus, during device testing and validation, it is essential to use flexible methods and tools to address the specific requirements of each device. In this context, this document proposes the development of a multiprotocol multiplexing debugger, designed to facilitate device testing in complex validation environments. The debugger, which integrates both dedicated hardware and software, adapts to the communication interface of the target device without the need for rewiring, enabling robust and efficient testing with a single device. This document provides a detailed description of the module's design, development, and integration process, in addition to documenting the workflow adopted by a leading company in the field, offering a comprehensive view of the practices and techniques used in the industry.

Keywords: Embedded systems. Test engineering. Hardware and software development.

Lista de ilustrações

Figura 1 : Fluxo de testes e validação de <i>Intellectual Property</i> (IP).....	13
Figura 2 : Exemplo de recabeamento durante os testes.	14
Figura 3 : Novo fluxo de testes de IP.....	15
Figura 4 : Princípio de funcionamento do módulo FLEXY.	15
Figura 5 : Diagrama de blocos dos microcontroladores STM32F7.....	17
Figura 6 : Exemplo de cronograma de dados – Protocolo JTAG	18
Figura 7 : Exemplo de topologia JTAG.....	19
Figura 8 : Exemplo de topologia SPI	20
Figura 9 : Exemplo de topologia I ² C	21
Figura 10 : Exemplo de topologia UART/USART	21
Figura 11 : Apresentação Altium Designer	22
Figura 12 : Apresentação TortoiseGit.....	23
Figura 13 : Apresentação STM32Cube	23
Figura 14 : Apresentação TouchGFX.....	24
Figura 15 : STM32F746G-DISCO (Esquerda) e STM32F769I-DISCO (Direita).....	25
Figura 16 : J-Link Base Classic.....	26
Figura 17 : STLink-V3SET da STMicroelectronics e seus módulos de extensão.....	27
Figura 18 : Apresentação Analisador Lógico Saleae.....	28
Figura 19 : Organização do projeto	29
Figura 20 : Estratégia de desenvolvimento do projeto.....	30
Figura 21 : Diagrama de Venn - FLEXY	31
Figura 22 : Modos de funcionamento – FLEXY.....	32
Figura 23 : Diagrama de blocos simplificado – Depurador FLEXY.....	33
Figura 24 : Arquitetura MVP - TouchGFX.....	34
Figura 25 : Cronograma de Escalonamento preemptivo TouchGFX.....	36
Figura 26 : Estrutura da API CMSIS-RTOS.....	37
Figura 27 : Exemplos e aplicações desenvolvidas com o TouchGFX.	38
Figura 28 : Topologia do buffer bidirecional para E/S	40
Figura 29 : Topologia da matriz de resistências.	41
Figura 30 : Tipos de conflitos na saída.....	41
Figura 31 : Topologia do circuito de detecção de conflitos de E/S.	42
Figura 32 : Diagrama de blocos completo – Depurador FLEXY.....	43
Figura 33 : Pinout FLEXY.....	43
Figura 34 : Montagem para a prova de conceito FLEXY	44
Figura 35 : Cronograma JTAG replicado F = 140kHz.	45
Figura 36 : Degradação do sinal TCK replicado em função de sua frequência.....	46
Figura 37 : Validação do princípio de redirecionamento de E/S e retransmissão via <i>bit banging</i> - Canal inicial: CH3 F = 50KHz / Canal final: CH0 F = 1MHz.	47
Figura 38 : Validação da comunicação entre o computador e o microcontrolador via a interface USB e validação do sistema de controle dos registros de segurança.	48
Figura 39 : Esquema funcional do protótipo 1	49
Figura 40 : Etapas de concepção do depurador FLEXY	50
Figura 41 : Exemplo de montagem para validação e exemplo de curva analisada.	50
Figura 42 : Formas de onda durante a inicialização do isolador 5V-5V. Rosa: 5V FLEXY - Azul: 5V Isolado - Amarelo: saída LDO 5V-3.3V.....	52
Figura 43 : Estrutura simplificada do software do módulo FLEXY.....	54
Figura 44 : Menu principal do firmware FLEXY	55
Figura 45 : Comparação entre o Protótipo 1 e o Protótipo 2 em nível de hardware	57

Lista de tabelas

Tabela 1 : Resultados dos testes e validação do FLEXY-PROTO-1	51
---	----

Lista de abreviaturas e siglas

DFT	Design For Test
DSI	Display Serial Interface
DUT	Device Under Test.
IP	Intellectual Property
I ² C	Inter-Integrated Circuit
JTAG	Join Test Action Group
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
MIPI	Mobile Industry Processor Interface
UART	Universal Asynchronous Receiver / Transmitter
USB	Universal Serial Bus
USART	Universal Synchronous / Asynchronous Receiver / Transmitter

Sumário

AGRADECIMENTOS.....	5
RESUMO.....	6
ABSTRACT.....	7
LISTA DE ILUSTRAÇÕES.....	8
LISTA DE TABELAS.....	9
LISTA DE ABREVIATURAS E SIGLAS.....	10
SUMÁRIO.....	11
1. INTRODUÇÃO.....	12
1.1. JUSTIFICATIVA.....	13
1.1. OBJETIVOS.....	15
2. REFERENCIAL TEÓRICO.....	16
2.1. SISTEMAS EMBARCADOS.....	16
2.1.1. <i>Microcontroladores.....</i>	<i>16</i>
2.2. PROTOCOLOS DE COMUNICAÇÃO.....	18
2.2.1. <i>JTAG.....</i>	<i>18</i>
2.2.2. <i>SPI.....</i>	<i>19</i>
2.2.3. <i>I²C.....</i>	<i>20</i>
2.2.4. <i>UART/USART.....</i>	<i>21</i>
2.3. FERRAMENTAS.....	22
2.3.1. <i>Altium Designer.....</i>	<i>22</i>
2.3.2. <i>TortoiseGit.....</i>	<i>22</i>
2.3.3. <i>STM32Cube.....</i>	<i>23</i>
2.3.4. <i>TouchGFX.....</i>	<i>24</i>
2.3.5. <i>Placas de desenvolvimento STM32-DISCOVERY.....</i>	<i>25</i>
2.3.6. <i>Depuradores.....</i>	<i>26</i>
2.3.6.1. <i>J-Link de Segger.....</i>	<i>26</i>
2.3.6.2. <i>STLink-V3 - STMicroelectronics.....</i>	<i>26</i>
2.3.7. <i>Analizador Lógico Saleae 8.....</i>	<i>28</i>
3. METODOLOGIA.....	29
3.1. ORGANIZAÇÃO E PLANO DE EXECUÇÃO DO PROJETO.....	29
4. DESCRIÇÃO DO SISTEMA.....	31
4.1. DESCRIÇÃO DO SISTEMA.....	31
4.2. MODOS DE FUNCIONAMENTO.....	32
5. ESTUDO DE VIABILIDADE.....	34
5.1. INTRODUÇÃO AO TOUCHGFX.....	34
5.1.1. <i>TouchGFX e RTOS.....</i>	<i>35</i>
5.1.2. <i>Integração TouchGFX e Hardware.....</i>	<i>38</i>
5.2. ANÁLISE DAS PLACAS DE REFERÊNCIA.....	39
5.2.1. <i>Buffer Bidirecional para E/S.....</i>	<i>39</i>
5.2.2. <i>Circuito de detecção de conflitos de E/S.....</i>	<i>41</i>
5.3. FLEXY - DIRETRIZES PARA O DESIGN DE HARDWARE.....	42
5.4. FLEXY - PROVA DE CONCEITO.....	44
6. DESENVOLVIMENTO DO PRIMEIRO PROTÓTIPO.....	49
6.1. <i>Plano de validação, metodologia de teste e resultados.....</i>	<i>50</i>
6.2. <i>Dificuldades e pontos de melhoria.....</i>	<i>51</i>
7. DESENVOLVIMENTO DO FIRMWARE EMBARCADO.....	54
8. DISCUSSÃO DOS RESULTADOS E PERSPECTIVAS FUTURAS.....	56
9. CONCLUSÃO.....	58
REFERÊNCIAS BIBLIOGRÁFICAS.....	59

1. Introdução

Nos últimos anos, os dispositivos eletrônicos registraram um crescimento significativo, transformando diversos aspectos do cotidiano. Esse avanço tem sido impulsionado pela constante inovação e pela demanda por soluções tecnológicas avançadas, especialmente em áreas como semicondutores, comunicações e computação. De acordo com a Fortune Business Insights (2023), o mercado global de eletrônicos de consumo, avaliado em USD 773,40 bilhões em 2023, deverá crescer aproximadamente 90% até 2032, refletindo uma tendência de expansão contínua.

Esses dispositivos dependem de microcontroladores, que estão presentes em uma ampla variedade de sistemas utilizados cotidianamente. Eles são encontrados em automóveis, eletrodomésticos, aparelhos médicos, dispositivos portáteis e até mesmo em aplicações industriais. Cada microcontrolador é composto por inúmeros circuitos que necessitam de testes e validações rigorosas para garantir o funcionamento seguro e adequado do sistema como um todo.

Para garantir o desempenho ideal desses equipamentos, é fundamental que os circuitos que os compõem sejam rigorosamente testados e validados. Cada componente possui características específicas e precisa passar por diversos testes, assegurando sua conformidade e segurança operacional. Esses procedimentos podem incluir ensaios de temperatura, testes de compatibilidade eletromagnética em câmaras isoladas e outros métodos especializados. Com o avanço constante da tecnologia, os circuitos se tornam cada vez mais complexos, exigindo um número maior de testes e padrões de controle mais rigorosos. A automação e padronização desses processos tornaram-se essenciais para atender a essa demanda crescente.

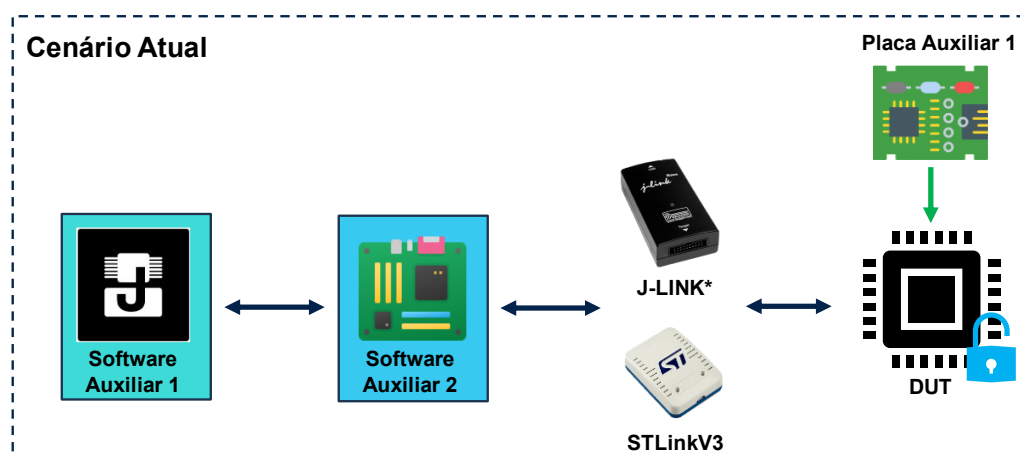
A engenharia de testes desempenha um papel fundamental nesse contexto, sendo a área responsável pelo desenvolvimento e execução desses procedimentos. Ela garante que cada componente e sistema integrado passe pelos critérios de qualidade e segurança necessários, colaborando diretamente para a confiabilidade e eficiência dos produtos.

Dessa forma, este trabalho tem como objetivo desenvolver um depurador multiprotocolo multiplexador, uma nova ferramenta que será empregada no processo de teste e validação de componentes. O projeto foi conduzido em colaboração com a STMicroelectronics, uma empresa líder global no setor de fabricação de semicondutores, reconhecida pela sua excelência em microcontroladores e inovação tecnológica.

1.1. Justificativa

Para uma melhor compreensão da problemática que foi abordada neste trabalho, é necessário compreender o fluxo de testes e validação de um componente eletrônico.

Figura 1 : Fluxo de testes e validação de *Intellectual Property* (IP).



Fonte : Autoria própria.

Conforme ilustrado na Figura 1, o processo de teste e validação de um componente inicia-se em um software interno denominado Software Auxiliar 1. Esta ferramenta é responsável por interpretar os comandos do usuário e encaminhá-los ao Software Auxiliar 2, que os converte em instruções. Essas instruções contêm endereços de registradores específicos para operações de leitura ou escrita, permitindo que o usuário interaja diretamente com o dispositivo em teste e verifique seu funcionamento (THERON, 2022)(FOUILLEUL, 2023).

Após essa conversão, as instruções são transmitidas pelo depurador STLinkV3 ao DUT, utilizando protocolos de comunicação de baixo nível, como: Joint Test Action Group (JTAG), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I²C), Serial Wire Debug (SWD) ou Universal Asynchronous Receiver / Transmitter (UART).

No uso do protocolo JTAG, destaca-se que, apesar de o depurador STLinkV3 ser uma ferramenta poderosa, sua capacidade de interpretar comandos é limitada, pois foi projetado exclusivamente para produtos ST. Como consequência, a equipe de engenheiros responsável deve atualizar sua base de comandos a cada novo produto desenvolvido para garantir a compatibilidade, um processo que demanda tempo e que geralmente não é aplicado a produtos ainda em desenvolvimento.

Em tais situações, utiliza-se o depurador J-Link, cuja principal vantagem em relação ao STLink é a capacidade de operar com uma gama maior de dispositivos. O J-Link possui

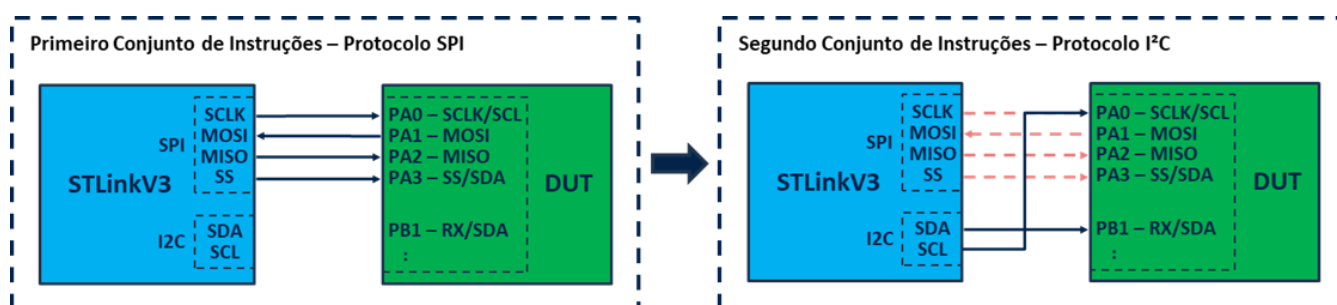
uma camada de abstração mais flexível, que permite acessar qualquer registro, desde que o dispositivo tenha uma arquitetura compatível. Os microcontroladores ST, baseados na série de núcleos ARM Cortex, estão entre os dispositivos compatíveis com o J-Link. É importante notar que, embora essa ferramenta seja eficaz no contexto do protocolo JTAG, ela não gerencia outras interfaces operadas pelo STLink.

O acesso a alguns dos registros utilizados nesse processo é restrito ao modo de teste. Para ativá-lo, utiliza-se uma ferramenta interna, aqui referida como Módulo Auxiliar 1. Esse módulo antigo opera de forma independente das demais ferramentas do engenheiro de teste e possui uma funcionalidade intuitiva: em sua memória, há uma sequência de comandos armazenados que são enviados ao dispositivo-alvo para ativação, acessados por um conjunto de botões.

Atualmente, esse módulo é essencial para a equipe de testes e validação. No entanto, ela representa uma vulnerabilidade em termos de segurança da informação, pois durante seu desenvolvimento, não foram previstos métodos para proteger as chaves registradas em sua memória. Assim, é possível recuperar essas informações por meio de engenharia reversa, conectando um analisador lógico às suas saídas.

Outra limitação no fluxo atual é a necessidade de recabeamento constante. Durante uma manipulação, pode ser necessário utilizar diversos protocolos de comunicação, e um mesmo pino do DUT pode ser compartilhado entre interfaces distintas. A Figura 2 ilustra uma sequência de testes em que o primeiro conjunto de instruções é transmitido via SPI, seguido de um novo conjunto pela interface I²C. Observa-se que o pino PA0 do DUT atua como SCLK no primeiro protocolo e como SDA no segundo. Por outro lado, o STLinkV3 possui pinos exclusivos para cada interface, exigindo a reorganização manual das conexões entre o depurador e o alvo. Essa necessidade de intervenção manual dificulta a automação do fluxo de testes e validação.

Figura 2 : Exemplo de recabeamento durante os testes.

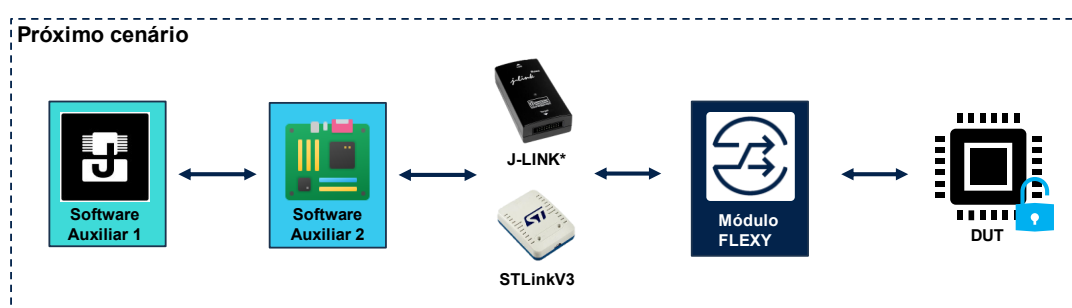


Fonte : Autoria própria.

1.1. Objetivos

A gestão de múltiplos depuradores e protocolos de comunicação apresenta desafios significativos, como a necessidade de recabeamento constante, que dificulta a automação do processo de teste e validação de componentes eletrônicos. Adicionalmente, o módulo Auxiliar 1, atualmente utilizado, está obsoleto, o que representa uma vulnerabilidade de segurança para a empresa. Com o intuito de resolver essas limitações, a equipe de testes e validação está comprometida em implementar melhorias no fluxo de trabalho atual. A figura 3 ilustra o novo processo.

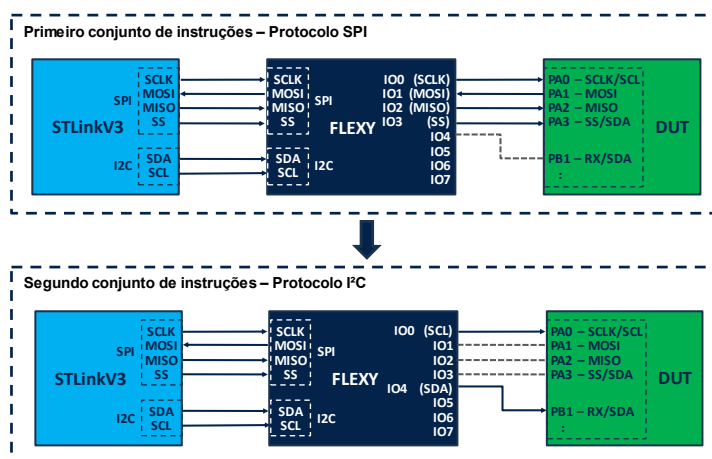
Figura 3 : Novo fluxo de testes de IP.



Fonte : Autoria própria.

O objetivo deste trabalho é desenvolver um módulo de extensão denominado FLEXY. Este módulo, conectado ao STLinkV3/J-Link, será capaz de atuar como um depurador multiprotocolo multiplexador. Conforme ilustrado na figura 4, sua função será redirecionar as informações recebidas para qualquer pino de saída, eliminando a necessidade de recabeamento ou intervenção manual, o que contribuirá para um processo de teste mais automatizado e seguro.

Figura 4 : Princípio de funcionamento do módulo FLEXY.



Fonte : Autoria própria

2. Referencial teórico

Este tópico apresenta algumas das principais teorias necessárias para uma melhor compreensão do trabalho realizado.

2.1. Sistemas embarcados

Os sistemas embarcados são sistemas computacionais projetados para realizar tarefas específicas e são frequentemente integrados em dispositivos maiores com uma finalidade definida. Esses sistemas combinam hardware e software em uma única estrutura e são amplamente empregados em aplicações que exigem alta eficiência, confiabilidade e controle preciso, como automóveis, eletrodomésticos, dispositivos médicos, equipamentos de telecomunicações e aeronaves (BERGER, 2002; SHANMUGAVEL, B.; T, R.; R, K, 2018).

Conforme Shanmugavel (2018), um sistema embarcado consiste em uma combinação de componentes eletrônicos e mecânicos, projetado para realizar uma função específica em tempo real, com alto desempenho e eficiência. A seleção de arquitetura e componentes nesses sistemas busca garantir operações precisas, aspecto essencial em aplicações de natureza crítica.

Diferentemente dos sistemas de propósito geral, como desktops e laptops, que possuem capacidade para executar uma ampla gama de tarefas, um sistema embarcado é otimizado para uma função ou um conjunto restrito de funções. Esses sistemas são chamados de embarcados porque o microprocessador ou microcontrolador, que controla o sistema, está embutido diretamente no dispositivo, integrando-se à sua estrutura para maximizar a eficiência e reduzir o consumo de energia (SHANMUGAVEL, B.; T, R.; R, K, 2018; TOULSON; WILMSHURST, 2017).

2.1.1. Microcontroladores

Os microcontroladores (MCUs) desempenham um papel fundamental em sistemas embarcados, atuando como elementos centrais compostos por um núcleo de processamento, memórias e periféricos. A principal diferença entre um microcontrolador e um microprocessador é que o microcontrolador integra em seu interior periféricos, como sensores, entradas e saídas digitais/analógicas e interfaces de comunicação, o que facilita sua interação direta com o ambiente (MESTANZA, 2024). Já o microprocessador, por sua vez, depende de circuitos externos para realizar essas funções, sendo mais adequado para sistemas de uso geral que exigem maior capacidade de processamento.

A Figura 5 apresenta o diagrama de blocos de um microcontrolador da família STM32F7, exemplificando a arquitetura e os principais componentes internos dessa linha (STMicroelectronics, 2023).

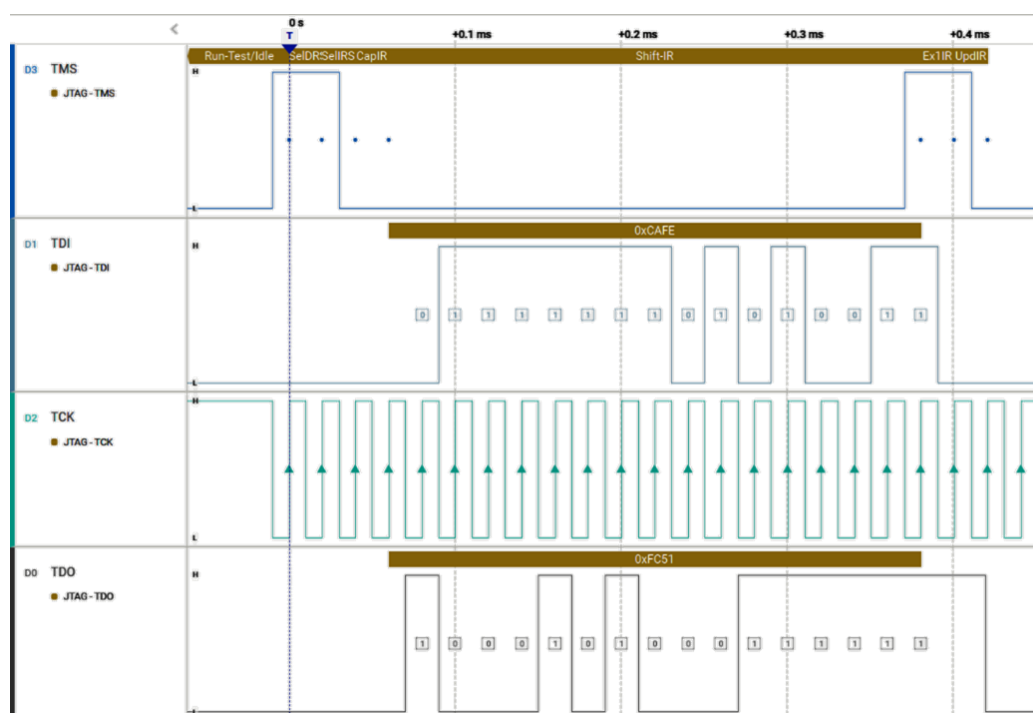
[illegible]

Fonte: STMicroelectronics

2.2. Protocolos de comunicação

Os protocolos de comunicação são conjuntos de regras e convenções que regulam a troca de informações entre dispositivos. Esses protocolos definem como os dados devem ser formatados, transmitidos, recebidos e interpretados, assegurando que a comunicação seja corretamente compreendida por todas as partes envolvidas. Além disso, eles especificam aspectos como a estrutura dos dados, o método de transmissão (por exemplo, via cabo ou sem fio), o controle de erro e de fluxo, bem como as medidas de segurança necessárias para proteger os dados contra acessos não autorizados, interceptações, e modificações maliciosas durante a transmissão. Observa-se na figura 6 como a informação é transmitida em nível de sinais elétricos.

Figura 6 : Exemplo de cronograma de dados – Protocolo JTAG



Fonte : Autoria própria.

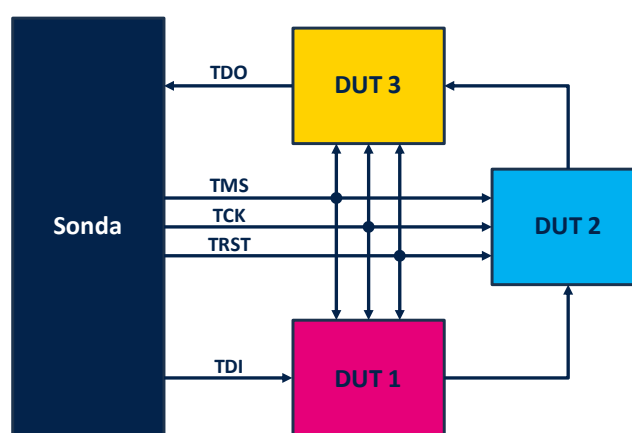
No contexto deste projeto, são abordados de forma específica os aspectos relacionados aos sinais elétricos, bem como os protocolos JTAG, SPI, I²C e UART. É importante ressaltar que os diagramas a seguir tem como objetivo ilustrar os principais sinais dos protocolos, dessa forma os sinais de alimentação estão implícitos.

2.2.1. JTAG

O protocolo JTAG, ou Test Access Port and Boundary-Scan Architecture (IEEE 1149.1), foi desenvolvido pelo Joint Test Action Group, um consórcio de empresas da

indústria de semicondutores. Introduzido como padrão para o teste de circuitos integrados, o JTAG fornece uma interface serial que permite o acesso direto e eficiente aos componentes eletrônicos em dispositivos complexos, como microprocessadores e chips integrados. Sua principal característica é a realização de testes diagnósticos avançados, depuração e programação por meio de uma interface baseada em um registro de deslocamento, possibilitando a verificação da funcionalidade dos dispositivos e a identificação de falhas de conexão através da técnica de Boundary Scan (ALTERA, 2002). A figura 7 ilustra como os dispositivos são conectados em uma rede JTAG.

Figura 7 : Exemplo de topologia JTAG.



Fonte : Autoria própria.

Os sinais comumente utilizados neste protocolo são:

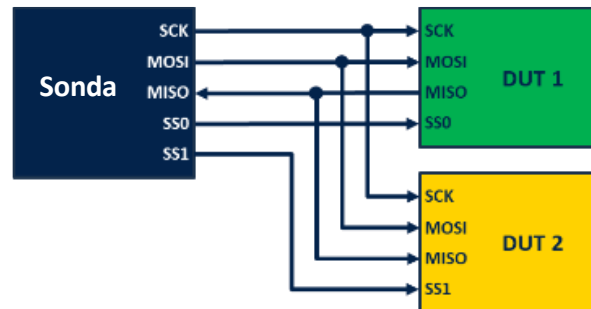
- Test Data In (TDI): Entrada de dados de teste. As informações são deslocadas para dentro do dispositivo através deste pino.
- Test Data Out (TDO): Saída de dados de teste. As informações são deslocadas para fora do dispositivo através deste pino.
- Test Clock (TCK): Sinal de relógio utilizado para sincronizar as operações de teste.
- Test Mode Select (TMS): Seleciona o modo de operação do JTAG.
- Test Reset (TRST): Opcional, utilizado para reinicializar o controlador JTAG.

2.2.2. SPI

O protocolo SPI é um padrão de comunicação serial desenvolvido pela Motorola na década de 1980. Esse padrão é amplamente utilizado na comunicação entre microcontroladores e periféricos, como sensores, memórias e displays, fornecendo uma interface simples e eficiente para a troca de dados em alta velocidade. Seu princípio fundamental baseia-se na arquitetura mestre-escravo, na qual o dispositivo mestre controla

a comunicação e se conecta a vários periféricos escravos através de um barramento comum. O protocolo é valorizado por sua versatilidade e eficiência, especialmente em aplicações que exigem transferências rápidas de dados (DHAKER, 2018). Podemos observar a topologia de uma rede SPI na figura 8.

Figura 8 : Exemplo de topologia SPI



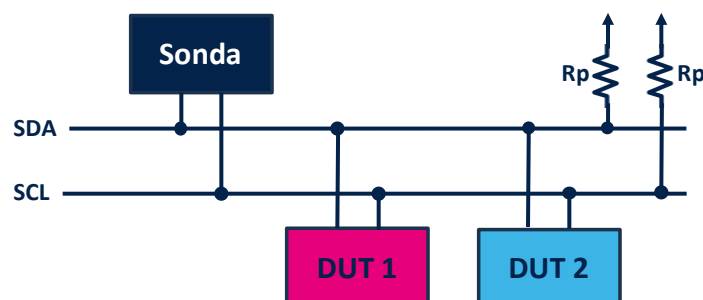
Fonte : Autoria própria.

Os principais sinais envolvidos são:

- Master Out Slave In (MOSI): Pino de dados pelo qual o mestre envia informações aos escravos.
- Master In Slave Out (MISO): Pino de dados pelo qual os escravos enviam informações ao mestre.
- Serial Clock (SCK): Sinal de relógio gerado pelo mestre para sincronizar a transferência de dados.
- Slave Select (SS): Linha utilizada pelo mestre para selecionar o periférico escravo específico.

2.2.3. I²C

O protocolo I²C foi desenvolvido pela Philips Semiconductor na década de 1980. Este padrão é comumente empregado em sistemas embarcados para permitir a comunicação entre vários dispositivos, como microcontroladores, sensores, memórias e outros periféricos, utilizando apenas duas linhas de comunicação. Os dispositivos conectados ao barramento I²C são classificados como mestre ou escravo. O dispositivo mestre é responsável por gerar os sinais de relógio e iniciar a comunicação, enquanto os dispositivos escravos respondem às instruções do mestre. Cada dispositivo no barramento possui um endereço único, permitindo que o mestre identifique e se comunique com cada escravo individualmente (WU, 2018). A figura 9 oferece ilustra como uma rede I²C é configurada.

Figura 9 : Exemplo de topologia I²C

Fonte : Autoria própria.

Os principais sinais do I²C são:

- Serial Data Line (SDA): Linha de dados bidirecional utilizada para a transferência de dados entre os dispositivos.
- Serial Clock Line (SCL): Linha de relógio gerada pelo mestre para sincronizar a transferência de dados.

2.2.4. UART/USART

Os protocolos UART e USART são amplamente utilizados para comunicação serial em sistemas embarcados. Desenvolvidos com o objetivo de facilitar a transferência de dados entre dispositivos, esses protocolos desempenham um papel fundamental em uma variedade de aplicações, abrangendo desde a comunicação entre microcontroladores até a interface com periféricos (PEÑA, 2020) (TEXAS INSTRUMENTS, 2010). A Figura 10 ilustra a conexão entre dois dispositivos que utilizam o protocolo UART/USART para comunicação.

Figura 10 : Exemplo de topologia UART/USART



Fonte : Autoria própria.

As principais conexões desses protocolos são:

- Transmit (TX): Linha de transmissão de dados, utilizada para enviar informações do dispositivo.
- Receive (RX): Linha de recepção de dados, utilizada para receber informações no dispositivo.

inicial para seus projetos, gerando o código de inicialização com base nas opções selecionadas.

- STM32CubeIDE: É um ambiente de desenvolvimento integrado (IDE) baseado na plataforma Eclipse, que oferece um ambiente completo para escrever, compilar, depurar e testar aplicações para dispositivos STM32.
- STM32CubeHAL: Essa camada de abstração de hardware fornece uma interface padronizada para acessar os recursos do microcontrolador, independentemente do modelo. A STM32CubeHAL simplifica o desenvolvimento de aplicações e facilita a portabilidade do código entre diferentes dispositivos STM32.
- STM32Cube Expansion Packages: São pacotes de software que oferecem suporte a funcionalidades adicionais que não estão incluídas na biblioteca padrão, como o protocolo Universal Serial Bus (USB) e sistemas operacionais em tempo real.

2.3.4. TouchGFX

O TouchGFX é um framework desenvolvido pela STMicroelectronics, projetado para criar interfaces gráficas de usuário (GUI) sofisticadas e de alto desempenho em dispositivos embarcados. Conforme ilustrado na figura 14, a ferramenta é amplamente utilizada em aparelhos com telas sensíveis ao toque, tornou-se especialmente popular em aplicações que demandam interfaces interativas e visualmente atrativas, como eletrodomésticos, equipamentos médicos, automação industrial e eletrônicos de consumo (STMicroelectronics, 2024).

Figura 14 : Apresentação TouchGFX



Fonte: TouchGFX.(STMicroelectronics)

2.3.5. Placas de desenvolvimento STM32-DISCOVERY

As placas STM32DISCOVERY, comumente denominadas STM32DISCO, são placas de desenvolvimento fornecidas pela STMicroelectronics com o objetivo de simplificar o processo de criação de aplicações que utilizam microcontroladores da família STM32. Projetadas para servir como plataformas de prototipagem rápida e desenvolvimento de software, essas oferecendo uma ampla gama de funcionalidades e periféricos integrados, incluindo botões, conectores USB e Ethernet, LEDs, sensores e até interfaces gráficas.

Existem diversos modelos de placas STM32DISCOVERY, cada um equipado com um microcontrolador STM32 diferente, adaptado para atender às necessidades específicas de cada projeto. Conforme ilustrado na figura 15, durante o desenvolvimento deste trabalho, foram utilizadas duas referências principais:

Figura 15 : STM32F746G-DISCO (Esquerda) e STM32F769I-DISCO (Direita)



Fonte: Adaptado STMicroelectronics

- STM32F746G-DISCO: Esta placa é baseada no microcontrolador STM32F746NGH6, que utiliza a arquitetura ARM Cortex-M7. Oferecendo uma ampla gama de funcionalidades, incluindo uma frequência de clock de até 216 MHz, até 1 MB de memória flash e 340 KB de SRAM, esta placa é ideal para aplicações que exigem alto desempenho. Além disso, ela conta com uma variedade de periféricos integrados, como botões, memórias externas do tipo SDRAM e Quad-SPI, conectividade para câmeras, sistemas de áudio e cartões microSD; uma tela LCD de 4,3 polegadas com tela sensível ao toque e interface RGB paralela, bem como interfaces de comunicação como USB FS e USB HS, UART, SPI, I²C e VCP (STMicroelectronics, 2023).
- STM32F769I-DISCO: Baseada no microcontrolador STM32F769NIH6, esta placa também utiliza a arquitetura ARM Cortex-M7. Com características semelhantes ao

modelo anterior, a STM32F769I-DISCO se destaca pela capacidade de suas memórias, oferecendo 2 MB de memória flash e 532 KB de SRAM. As funcionalidades presentes nesta placa são similares às do modelo anterior; no entanto, ela possui apenas uma interface USB HS e sua tela LCD mede 4 polegadas, conectada por meio de uma interface MIPI DSI (STMicroelectronics, 2023).

2.3.6. Depuradores

Os depuradores são ferramentas utilizadas para estabelecer a comunicação entre um computador e um microcontrolador.

2.3.6.1. J-Link de Segger

O J-Link Base é um depurador (figura 16) utilizado para programar e depurar códigos embarcados, com suporte para as interfaces JTAG e SWD. Uma de suas principais vantagens é a capacidade de operar com uma variedade de microcontroladores de diferentes fabricantes, o que o torna uma escolha prática para desenvolvedores que atuam em diversas plataformas. Além disso, com atualizações regulares de firmware e suporte contínuo da Segger, o J-Link Base se estabelece como uma solução confiável e sustentável ao longo do tempo (Segger, 2024).

Figura 16 : J-Link Base Classic



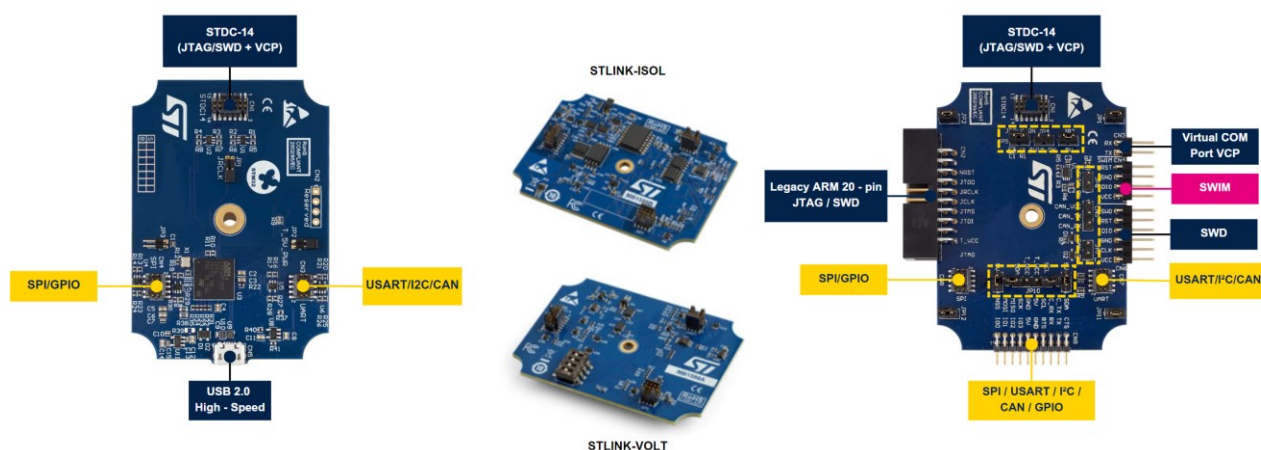
Fonte: Segger

2.3.6.2. STLink-V3 - STMicroelectronics

O STLink-V3SET é um depurador modular empregado na programação e depuração de microcontroladores das famílias STM8 e STM32. Conforme ilustrado na figura 17, ele consiste em uma placa principal, que abriga um microcontrolador STM32F7 responsável

pela gestão do sistema, e uma placa de extensão, a qual facilita o acesso do usuário a diversas interfaces de comunicação (STMicroelectronics, 2023).

Figura 17 : STLink-V3SET da STMicroelectronics e seus módulos de extensão.



Fonte: STMicroelectronics

Na programação e depuração, utiliza-se as interfaces JTAG ou SWD, e a comunicação entre o usuário e a placa alvo é realizada por meio de uma porta COM virtual. Além desses protocolos, o STLink permite que o usuário utilize as interfaces SPI, USART, I²C, CAN, SWIM e quatro General-Purpose Input/Output(*GPIO*) para a implementação de protocolos personalizados. Essas interfaces são acessíveis por meio da API STLINK-V3-BRIDGE, um conjunto de arquivos fonte que possibilita ao usuário desenvolver suas próprias aplicações.

A arquitetura modular do STLink-V3SET permite a adição de dois módulos auxiliares que ampliam suas funcionalidades:

- **STLINK-VOLT:** Esse módulo permite que o depurador ajuste seus níveis lógicos de tensão para se adequar aos da placa alvo, com uma faixa de operação de 1,6V a 3,3V.
- **STLINK-ISOL:** Este módulo adiciona uma camada de isolamento galvânico entre o depurador e o DUT. Essa funcionalidade é útil em ambientes ruidosos e protege o STLink em caso de mau funcionamento no DUT ou vice-versa. Além disso, assim como o módulo VOLT, ele também possibilita o ajuste dos níveis lógicos de tensão para adequar-se aos da placa alvo.

2.3.7. Analisador Lógico Saleae 8

O analisador lógico Saleae é uma ferramenta projetada para capturar múltiplos sinais de sistemas eletrônicos. Ele converte os dados adquiridos em diagramas temporais, facilitando a visualização e a análise do comportamento dos sinais capturados. No contexto deste projeto, o Saleae foi utilizado como uma ferramenta essencial para a decodificação de protocolos de comunicação, monitoramento de entradas e saídas, além do acompanhamento de máquinas de estado (Saleae, 2024);

Figura 18 : Apresentação Analisador Lógico Saleae.



Fonte : Adaptado Saleae

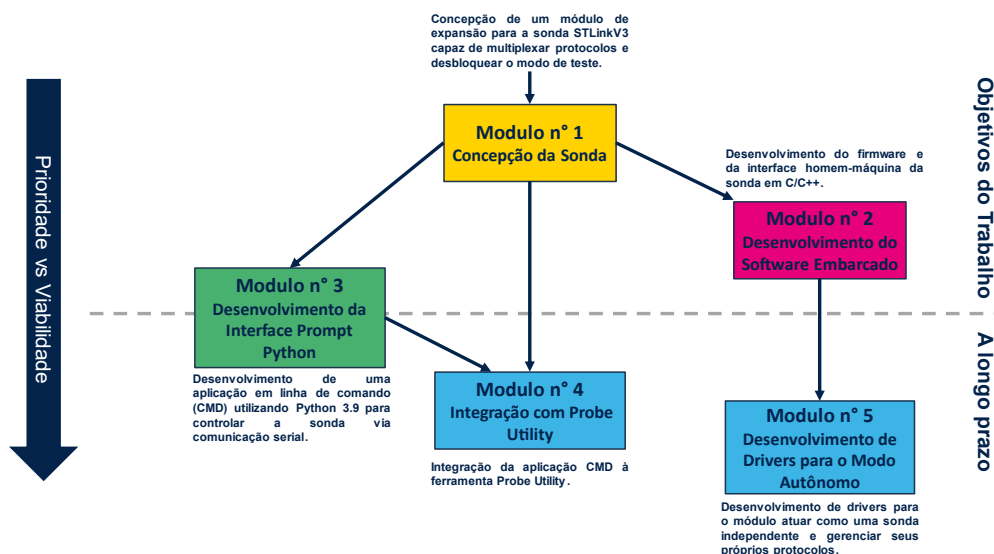
3. Metodologia

Esse capítulo apresenta a metodologia adotada para o desenvolvimento do projeto, abordando os principais módulos do sistema.

3.1. Organização e plano de execução do projeto

Este trabalho abrangeu diversas áreas da informática e da eletrônica, que se encontram intimamente relacionadas. A Figura 19 apresenta os diferentes módulos que compõem o projeto, com ênfase nas metas de curto e longo prazo. As atividades descritas neste trabalho foram realizadas entre fevereiro e agosto de 2024 contribuíram diretamente para o desenvolvimento de longo prazo, com a execução dos dois primeiros módulos e parte do terceiro.

Figura 19 : Organização do projeto



Fonte : Autoria própria.

O Módulo 1 concentrou-se na concepção da placa de expansão. Considerando a complexidade do design de hardware, realizou-se estudos de caso para verificar a viabilidade das funcionalidades descritas no documento de especificação do depurador FLEXY, bem como análises dos circuitos das placas de referência. A partir dessas informações, desenvolveu-se um primeiro protótipo.

O Módulo 2 teve como objetivo o desenvolvimento do *firmware* a ser embarcado no dispositivo. Nesta fase, foram conduzidos estudos de arquitetura de software e realizadas comparações de desempenho de ferramentas para o desenvolvimento de interfaces homem-máquina e validação de funções básicas

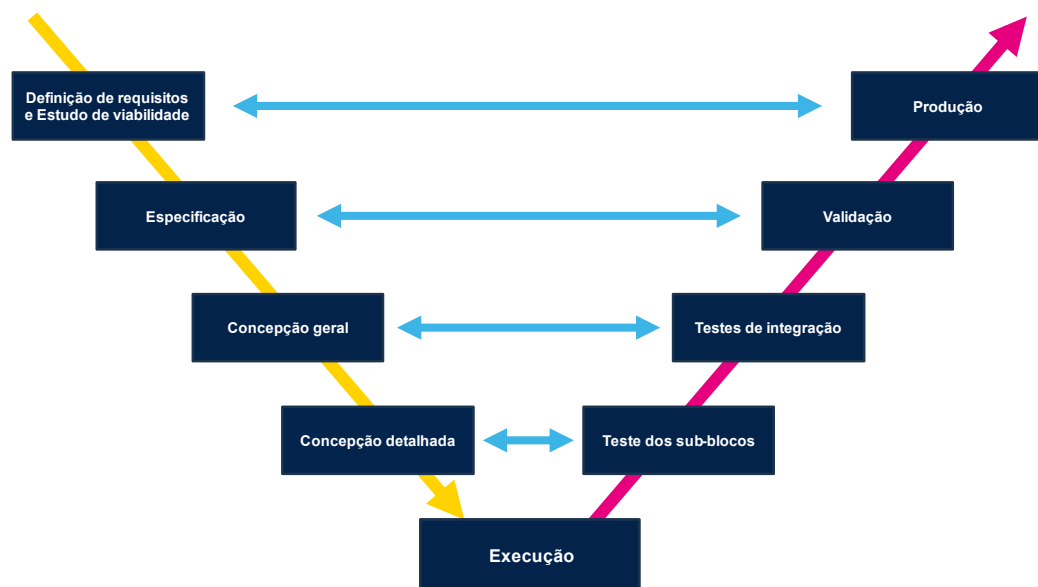
O Módulo 3 abordou o desenvolvimento de uma camada responsável pela interpretação dos comandos enviados pelo usuário via computador. Nesta etapa, a lógica de processamento foi estruturada e as primeiras funcionalidades foram integradas, ficando o restante do desenvolvimento para etapas posteriores.

Em uma perspectiva de longo prazo, o Módulo 4 visa a integração de uma interface de linha de comando em Python com a ferramenta gráfica de alto nível *Probe Utility*, facilitando a interação do usuário com o módulo de expansão.

Por fim, o Módulo 5 consistirá no desenvolvimento de drivers que permitirão à placa de expansão gerenciar seus próprios protocolos de comunicação, possibilitando o acesso e manipulação dos registradores do DUT sem a necessidade de um depurador STLink ou J-Link.

Conforme ilustrado na figura 20, para o desenvolvimento do projeto, adotou-se uma estratégia de desenvolvimento em ciclo em V. Essa abordagem permitiu definir com clareza o processo a ser seguido na concepção do dispositivo e facilitou a verificação e validação em cada etapa. Essa metodologia contribuiu para a mitigação de riscos nas fases iniciais e favoreceu a qualidade final do trabalho. Deve-se ressaltar que embora todas as etapas tenham sido seguidas durante o desenvolvimento desse projeto, algumas são descritas de forma implícita no decorrer desse trabalho.

Figura 20 : Estratégia de desenvolvimento do projeto



Fonte : Autoria própria.

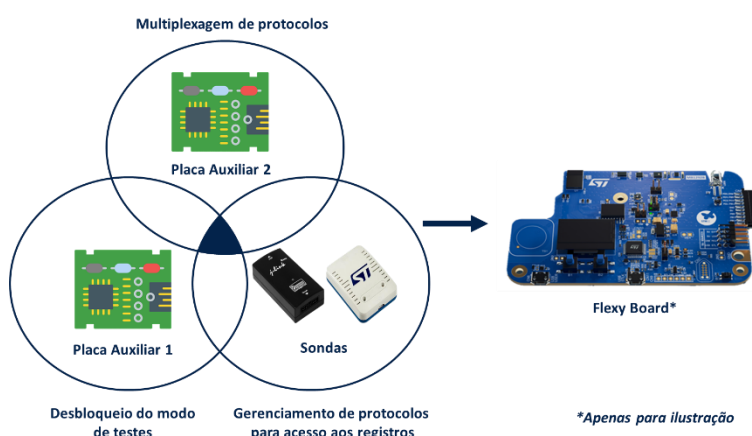
4. Descrição do Sistema

Conforme ilustrado na figura 20, a primeira etapa desse trabalho consistiu na definição de requisitos do projeto e em um estudo de viabilidade. Nessa sessão, são apresentados os requisitos do sistema e os modos de funcionamento previstos.

4.1. Descrição do Sistema

O módulo FLEXY integra funcionalidades específicas para aprimorar o processo de testes, conforme ilustrado na Figura 21, que destaca os dispositivos que servem de referência para seu desenvolvimento. As funcionalidades principais incluem:

Figura 21 : Diagrama de Venn - FLEXY



Fonte : Autoria própria.

- **Módulo Auxiliar 1:** Foi incorporado ao FLEXY a funcionalidade de desbloqueio do modo de teste, essencial para o fluxo de trabalho da equipe, junto com um sistema de chaves de segurança para aumentar a proteção. A chave de acesso será armazenada em uma memória volátil e poderá ser apagada por comando do usuário ou após inatividade prolongada caso o certificado de segurança não seja renovado por um servidor externo.
- **Módulo Auxiliar 2:** Este módulo é utilizado pelos engenheiros de validação para testar as entradas e saídas do DUT. Dentre suas principais funcionalidades, foram integradas ao módulo a capacidade de adaptação à tensão do dispositivo alvo, a capacidade de detectar conflitos entre as E/S, e a capacidade de alocar dinamicamente os pinos utilizados no protocolo JTAG.

- **STLink e J-Link:** A placa FLEXY foi desenvolvida de modo a garantir a compatibilidade com ambos os depuradores, assegurando a integração com os protocolos gerenciados por cada um.

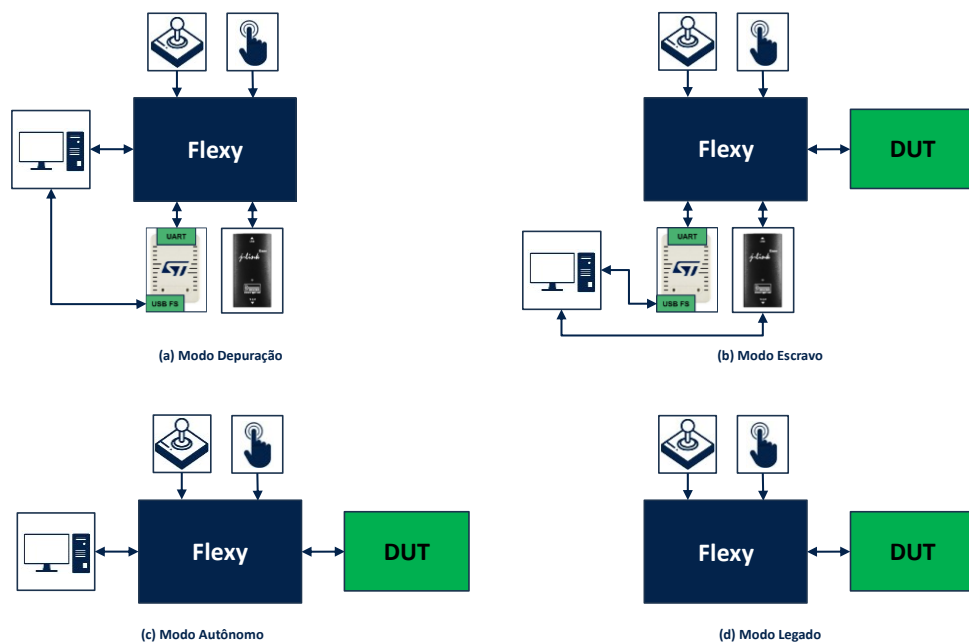
Além disso, o módulo foi galvanicamente isolado do equipamento em teste, com o objetivo de melhorar a integridade dos sinais transmitidos, garantir que não haja interferências decorrentes das diferenças de potencial entre os dispositivos e minimizar os danos nos componentes de ambos os circuitos em caso de mau funcionamento.

Por fim, o depurador é capaz de gerenciar os protocolos I²C, JTAG, SPI e UART/USART.

4.2. Modos de funcionamento

Conforme ilustrado na figura 22, o módulo FLEXY possui quatro modos de operação distintos, cada um com funcionalidades específicas:

Figura 22 : Modos de funcionamento – FLEXY



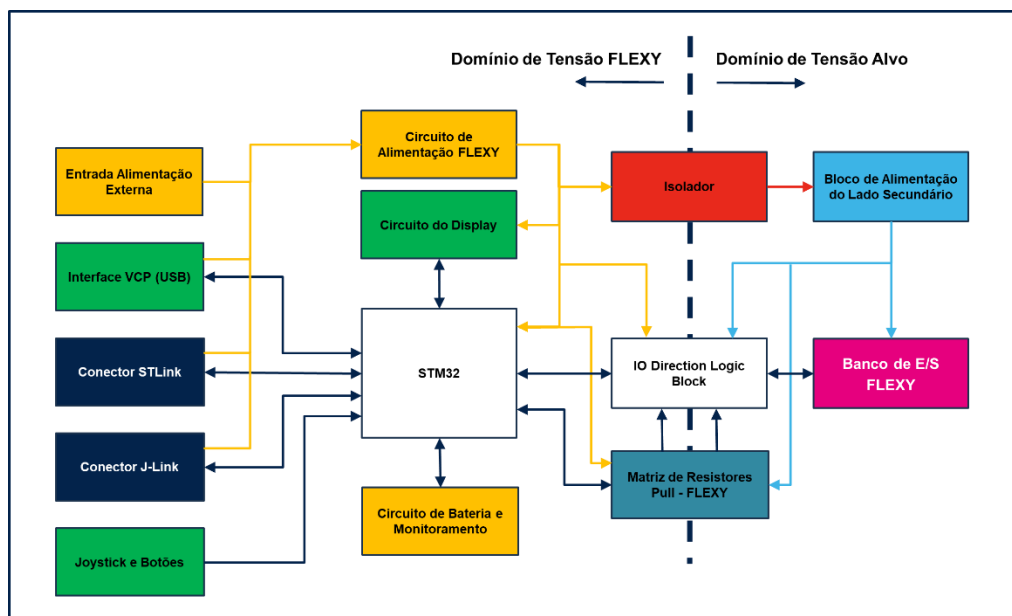
Fonte : Autoria própria.

- **Modo Depuração:** Neste modo, o STLink e o J-Link estarão conectados ao FLEXY. O usuário poderá enviar instruções ao módulo para desbloquear o modo de testes e redirecionar as instruções recebidas. Além disso, será possível solicitar informações, como a versão do firmware, gerenciar as chaves de segurança e depurar seu funcionamento.

- **Modo Autônomo:** Esta configuração é uma variação do primeiro modo, na qual apenas o computador está conectado ao módulo. Semelhante ao modo default, será possível obter informações sobre a módulo FLEXY e desbloquear o modo de teste. No futuro, o módulo terá a capacidade de atuar como um depurador, gerenciando seus próprios protocolos de comunicação.
- **Modo Escravo:** Neste modo, apenas os depuradores estão conectados ao módulo. Sua manipulação será feita através de uma tela LCD sensível à toques e de uma interface de botões. Nessa configuração, estão previstas as mesmas funcionalidades do primeiro modo.
- **Modo Legacy:** Nesta configuração, a placa FLEXY funciona como o módulo auxiliar 1 e será controlado através da interface LCD e dos botões. Nesse contexto, o módulo é utilizado exclusivamente para desbloquear o modo de testes no DUT, visando garantir a compatibilidade com o fluxo de teste atual.

A Figura 23 apresenta um diagrama de blocos simplificado que resume a organização do hardware do módulo.

Figura 23 : Diagrama de blocos simplificado – Depurador FLEXY



Fonte : Autoria própria.

5. Estudo de viabilidade

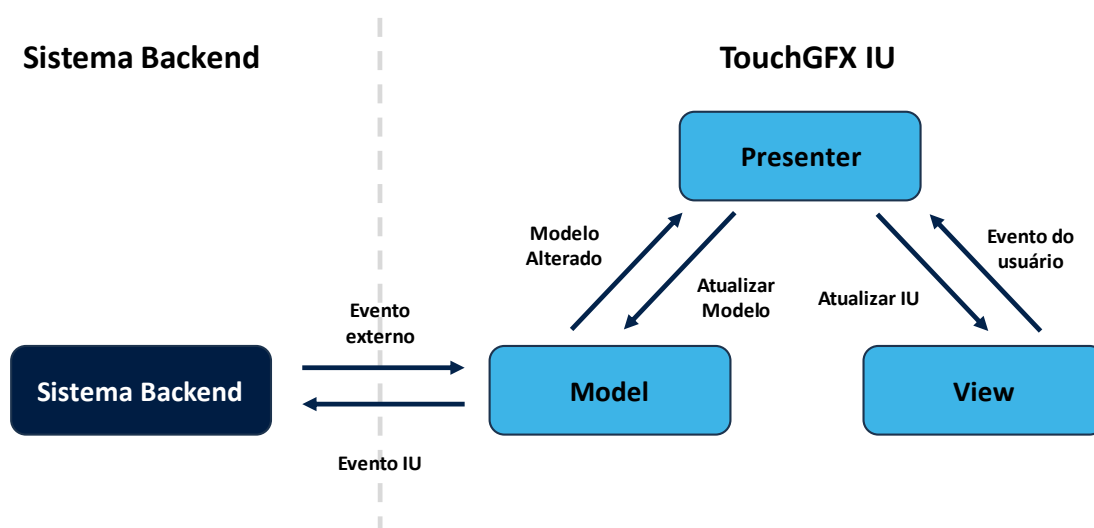
O projeto do módulo FLEXY abrangeu diversas áreas, como desenvolvimento de hardware, firmware, interface homem-máquina e gestão de protocolos de comunicação. Assim, a primeira etapa desse trabalho consistiu em uma análise de viabilidade da solução proposta, com o objetivo de compreender as interações da ferramenta TouchGFX com o ecossistema STM32, explorar suas limitações e examinar opções de hardware adequadas às necessidades do projeto.

5.1. Introdução ao TouchGFX

O TouchGFX é um framework desenvolvido em C++ que proporciona alto desempenho e fluidez para interfaces gráficas, sendo especialmente otimizado para microcontroladores da família STM32. Sua integração com ferramentas e ambientes da STMicroelectronics, como o STM32CubeMX e o STM32CubeIDE, facilita a criação de projetos e permite configurá-los corretamente. Além disso, o TouchGFX oferece suporte completo por meio de uma vasta documentação oficial e de uma comunidade ativa de desenvolvedores, acelerando o processo de criação de aplicações.

O TouchGFX adota a arquitetura *Model-View-Presenter* (MVP) para organizar o código embarcado, uma estrutura que facilita a modularidade e a manutenção, sendo especialmente útil em aplicações complexas. Essa arquitetura é ilustrada na figura 24.

Figura 24 : Arquitetura MVP - TouchGFX



Fonte : Adaptado TouchGFX

Os componentes do sistema são definidos da seguinte forma (STMicroelectronics):

- **View:** Interface passiva responsável por exibir os dados provenientes do modelo e encaminhar os comandos do usuário (eventos) ao *presenter*.
- **Presenter:** Componente intermediário entre *View* e *Model*, é responsável por atualizar o *Model* conforme as interações do usuário e por preparar e enviar os dados para exibição no *view*.
- **Model:** Armazena e organiza os dados exibidos ou manipulados pela interface do usuário, além de se comunicar diretamente com o software embarcado de baixo nível no microcontrolador (*backend*).

O motor gráfico do TouchGFX opera por meio de um loop principal estruturado em três etapas sequenciais:

- **Coletar:** O motor gráfico coleta eventos externos, como toques e pressionamentos de botões, convertendo-os em eventos específicos, como cliques, deslizamentos e gestos.
- **Atualizar:** Os eventos coletados são enviados para o objeto gráfico correspondente (botão, barra de progresso, tela, etc.), que gerencia a ocorrência. Normalmente, uma ou mais propriedades desse objeto serão modificadas, como cor, estado ou visibilidade. Em seguida, o objeto solicita ao motor gráfico que seja redesenhado no próximo frame. Esse método otimiza o processo de renderização, pois apenas as áreas modificadas são redesenhadas.
- **Renderizar:** A etapa de atualização resulta em uma lista de áreas a serem redesenhadas, chamadas de áreas invalidadas. Durante a etapa de renderização, o motor gráfico percorre essa lista, escaneia o modelo de cena (cada uma das camadas sobrepostas nessa área) e chama o método de desenho desses widgets, começando pela camada inferior até a camada superior.

Por fim, após um sinal de sincronização, a interface é atualizada e o motor gráfico começa a preparar a próxima imagem.

5.1.1. TouchGFX e RTOs

Os dispositivos embarcados estão se tornando cada vez mais avançados. A maioria dos sistemas não apenas gerencia a interface gráfica do usuário, mas também executa algoritmos complexos e múltiplas tarefas. Da mesma forma, o módulo FLEXY

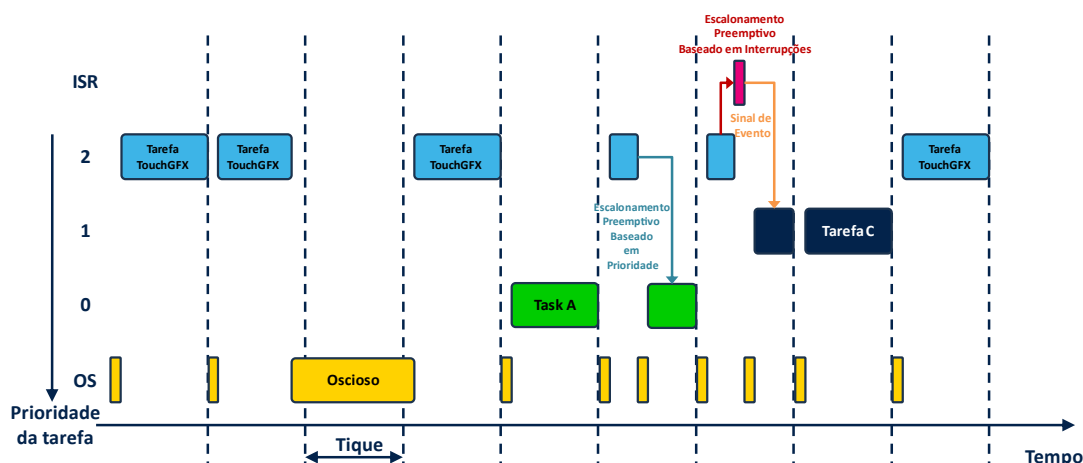
deverá ser capaz de lidar com protocolos de comunicação, processar instruções provenientes do computador e garantir a reatividade da interface gráfica para proporcionar uma experiência satisfatória ao usuário. Para isso, será utilizado um sistema operacional em tempo real (RTOS).

Um RTOS é projetado para gerenciar a execução de múltiplas tarefas em um ambiente onde a precisão no tempo de resposta é crítica. Seu funcionamento baseia-se na criação de tarefas, cada uma com prioridade e uma janela de execução definida. Essa abordagem é especialmente vantajosa em microcontroladores com um único núcleo, proporcionando a sensação de simultaneidade ao usuário, embora apenas uma tarefa seja executada por vez.

O núcleo de um RTOS é o Kernel, responsável pela gestão de recursos do sistema, como memória, dispositivos (por meio de drivers), comunicação e sincronização entre tarefas, por meio de filas e mutexes. O escalonador é o componente do Kernel que gerencia a fila de tarefas, decidindo qual tarefa será executada em seguida com base na política de prioridade, além de administrar interrupções e mudanças de contexto.

A Figura 25 ilustra um exemplo de escalonamento de tarefas em uma aplicação TouchGFX em um sistema de apenas um núcleo. A tarefa TouchGFX engloba os processos de coleta, atualização e renderização, além das interações básicas entre a interface gráfica e o modelo. Observa-se que essa tarefa é interrompida pela Tarefa A devido à sua prioridade mais alta. Graças ao princípio de preempção dos sistemas em tempo real, ocorre uma mudança de contexto, permitindo que a Tarefa A seja executada (HYMEL, 2021).

Figura 25 : Cronograma de Escalonamento preemptivo TouchGFX.

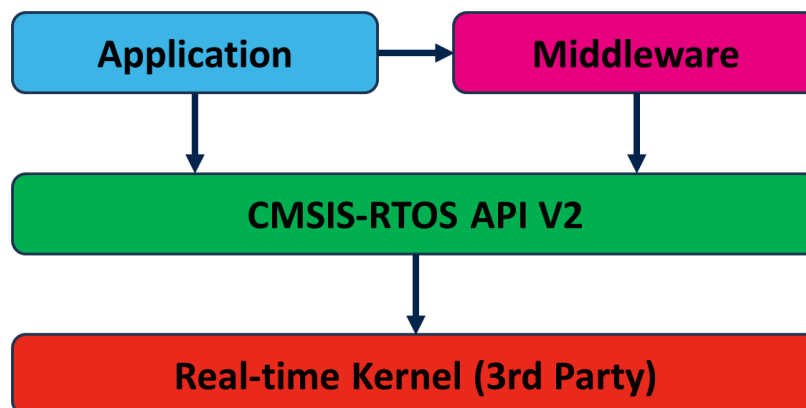


Fonte : Autoria própria.

Além do escalonamento baseado em prioridade, existe também o escalonamento baseado em interrupções. As interrupções são eventos definidos pelo usuário e possuem prioridade superior a qualquer tarefa, garantindo a reatividade do sistema em relação a estímulos externos. Quando uma rotina de interrupção é acionada, ela sinaliza à Tarefa C que um recurso necessário para sua execução está disponível. Em seguida, o escalonador a define como a próxima atividade a ser executada.

Os microcontroladores STM32, baseados na arquitetura Cortex-M, utilizam o conjunto de ferramentas CMSIS, composto por APIs e frameworks projetados para simplificar o desenvolvimento e a reutilização de software. Neste projeto, utilizou-se o CMSIS-RTOS2, segunda versão da interface padrão desenvolvida pela ARM para sistemas operacionais em tempo real. Como ilustrado na figura abaixo, essa aplicação atua como uma camada de *wrapper*, reduzindo o número de funções com as quais o desenvolvedor precisaria lidar em uma aplicação baseada em RTOS. Isso também assegura a portabilidade do firmware entre diferentes controladores Cortex-M e sistemas RTOS, facilitando a migração e a reutilização de código (ARM, 2022).

Figura 26 : Estrutura da API CMSIS-RTOS.



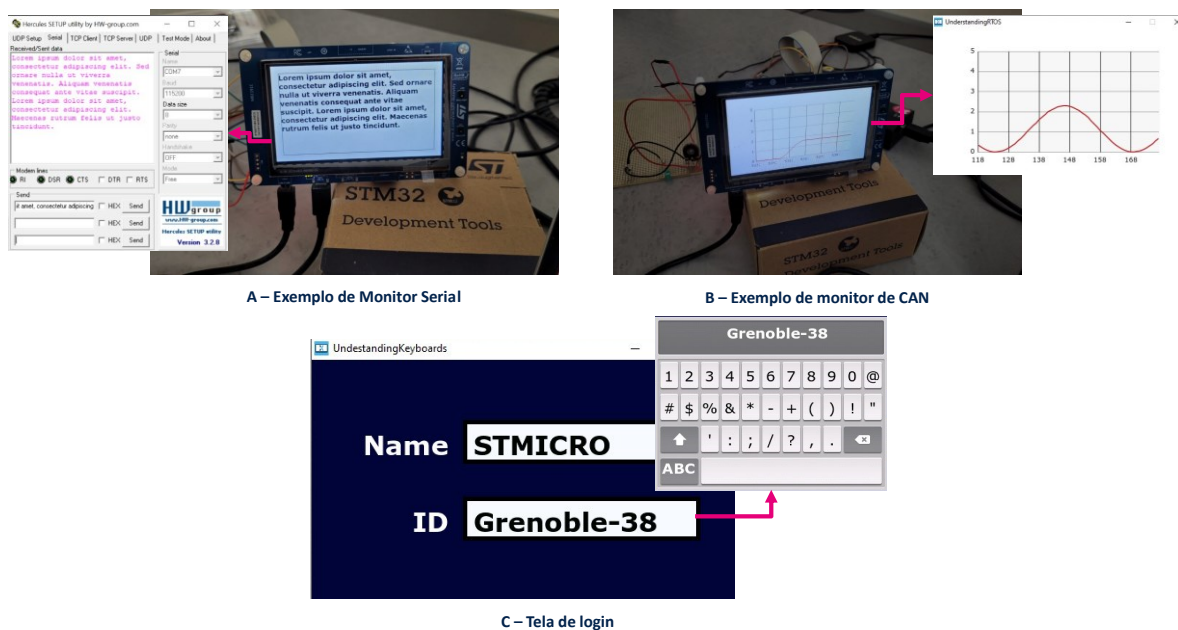
Fonte: ARM

Por fim, o FreeRTOS é um sistema operacional em tempo real, desenvolvido em colaboração com algumas das principais empresas de semicondutores do mundo e distribuído sob licença de código aberto. Sua escolha para o desenvolvimento deste projeto se deve à robustez, à compatibilidade com microcontroladores STM32, ao baixo consumo de memória e à extensa comunidade de desenvolvedores, que assegura suporte e melhorias constantes.

5.1.2. Integração TouchGFX e Hardware

Com base nos estudos apresentados na seção anterior, foi proposta a validação da integração entre o ecossistema TouchGFX e os microcontroladores STM32. Para esse propósito, utilizou-se a placa STM32F746G, na qual foram desenvolvidas as aplicações ilustradas na Figura 27.

Figura 27 : Exemplos e aplicações desenvolvidas com o TouchGFX.



Fonte: Autoria própria

- **Osciloscópio:** Esta aplicação inicial foi desenvolvida para explorar a gestão de múltiplas tarefas em um sistema RTOS, verificar a comunicação entre tarefas por meio de filas de mensagens e validar a interação entre as camadas gráfica e física. A aplicação é estruturada em duas tarefas principais:
 - ❖ **Tarefa ADC:** Com a prioridade mais alta, esta tarefa coleta amostras de um conversor analógico-digital (ADC) conectado a um potenciômetro. Os dados obtidos são enviados para a tarefa TouchGFX através de uma fila de mensagens. Após o envio, a tarefa ADC entra em modo inativo por 10 ms.
 - ❖ **Tarefa TouchGFX:** A cada 1 ms, esta tarefa verifica a fila de mensagens em busca de novos valores. Quando um valor é recebido pelo modelo, o gráfico é atualizado com as medições recentes.
- **Tela de login:** Esta aplicação foi desenvolvida para explorar os widgets do TouchGFX e analisar como interagem com o firmware. A tela inicial exibe um

formulário de login, no qual o usuário insere dados por meio de um teclado personalizado. Embora simples, sua implementação exige a sobreposição de múltiplos elementos para criar campos de entrada e botões invisíveis, que delimitam áreas específicas para a interação do usuário.

- **Monitor serial:** Esta aplicação foi desenvolvida com o objetivo de integrar o ambiente TouchGFX a um computador. Para isso, utilizou-se a interface VCP do STLINKV3, que recebe dados da interface USB do computador e os retransmite ao microcontrolador principal via protocolo UART. Com esta aplicação, foi possível compreender a gestão de buffers de entrada e o processamento de instruções provenientes de um computador, etapa essencial na automação de processos de teste.

As aplicações desenvolvidas permitiram validar os conceitos fundamentais para a integração do TouchGFX com o hardware. Elas constituem a base para o desenvolvimento da aplicação final, facilitando a implementação das próximas etapas

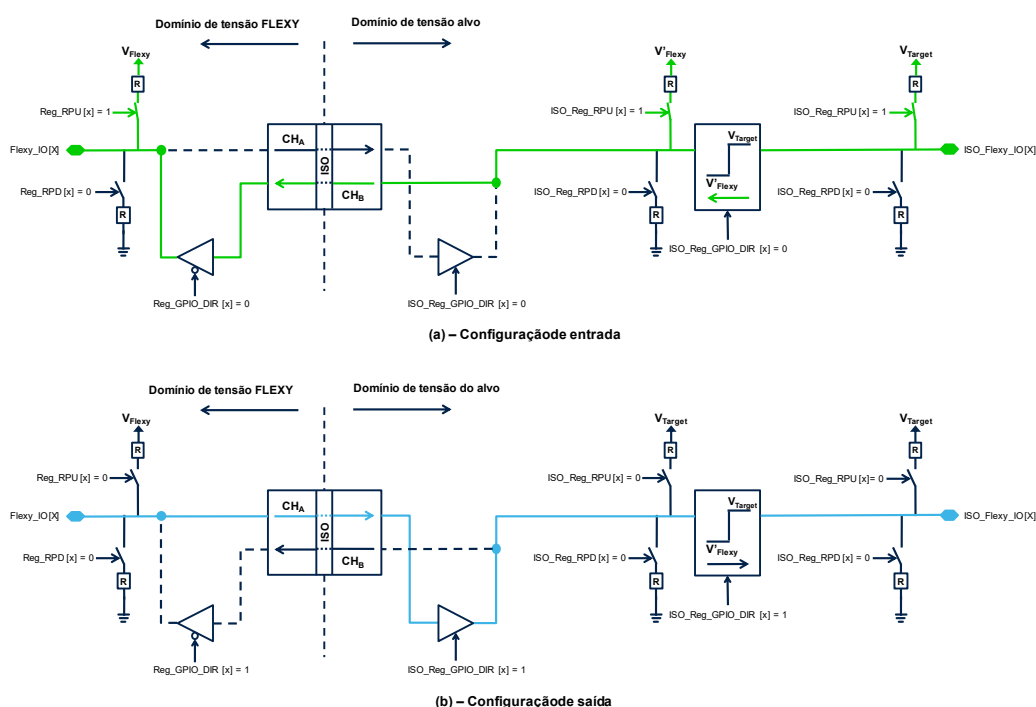
5.2. Análise das placas de referência

A segunda parte do estudo de viabilidade consistiu em uma análise detalhada dos módulos de referência. Foram realizados testes práticos com ambos os instrumentos para avaliar suas capacidades, seguidos de um estudo dos esquemas eletrônicos, com o objetivo de compreender seu funcionamento físico e catalogar os componentes eletrônicos utilizados em seus projetos.

5.2.1. Buffer Bidirecional para E/S

Com base nos resultados dessa segunda etapa, desenvolveu-se um circuito para garantir a isolação entre o módulo FLEXY e o DUT, permitindo a comunicação entre os dispositivos. Conforme ilustrado na Figura 28, o circuito utiliza um isolador bidirecional que permite a passagem de sinais elétricos em uma única direção por vez, empregando dois buffers de três estados controlados pelo microcontrolador integrado ao módulo FLEXY. Os buffers possuem lógicas de controle complementares: o buffer de entrada opera no domínio de tensão do microcontrolador, enquanto o buffer de saída opera no domínio de tensão do DUT.

Figura 28 : Topologia do buffer bidirecional para E/S



Fonte : Autoria própria

Na configuração de entrada, um sinal de controle em nível lógico baixo é enviado pelo módulo de extensão para os buffers. O buffer de saída coloca o canal A em estado de alta impedância, impedindo que os sinais do módulo sejam direcionados ao DUT. Simultaneamente, o buffer de entrada permite a passagem dos sinais do DUT para o módulo. O funcionamento do circuito é complementar na configuração de saída.

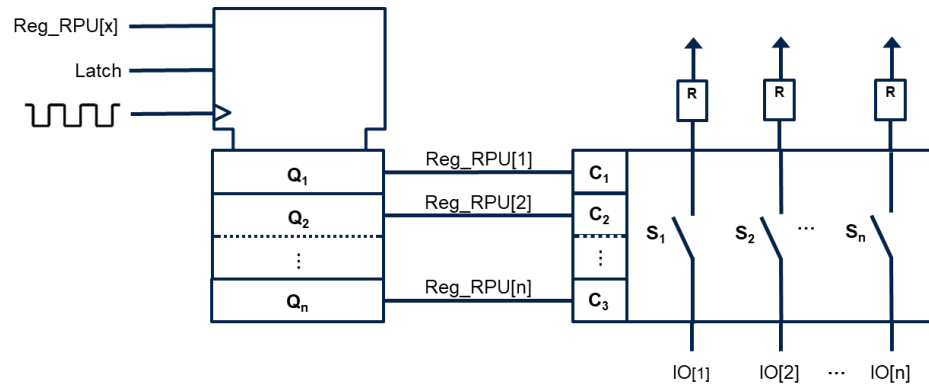
Os sinais elétricos são ajustados conforme sua direção por meio de um regulador de nível. Para os sinais de entrada, os níveis lógicos variam entre 0 V e 3,3 V; para os sinais de saída, os níveis lógicos ficam entre 0 V e 1,65~3,3 V, adaptados à tensão do DUT.

Além da gestão da direção das E/S, o circuito integra resistores de *pull-up* e *pull-down* que podem ser conectados à linha de transmissão conforme as necessidades do protocolo, controlados por uma matriz de contatos gerida pelo módulo.

O módulo possui 8 E/S para conexão com o DUT. A fim de garantir a escalabilidade do projeto e minimizar o número de pinos necessários para o controle das resistências e da direção das E/S, utilizaremos registradores de deslocamento série-paralelo, que permitem controlar múltiplas saídas com apenas quatro pinos do microcontrolador. As saídas do registro de deslocamento são conectadas em uma

matriz de contatos. A Figura 29 ilustra o arranjo para a gestão dos resistores de *pull-up*; o mesmo princípio é aplicado para o arranjo de *pull-down* e para o controle dos *buffers* de E/S.

Figura 29 : Topologia da matriz de resistências.

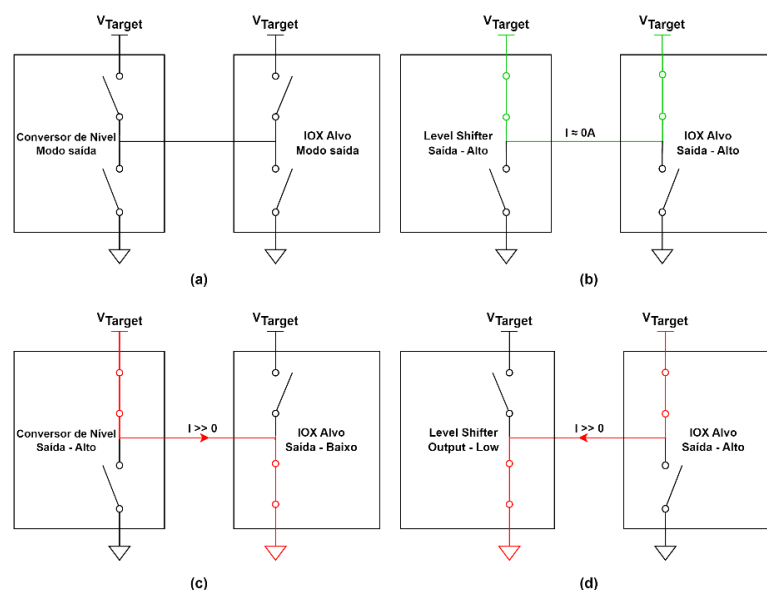


Fonte : Autoria própria

5.2.2. Circuito de detecção de conflitos de E/S

A multiplexação dos protocolos exigirá a reconfiguração constante do banco de E/S do depurador, o que pode resultar em conflitos. Nesse contexto, um conflito ocorre quando há uma configuração inadequada de um pino de saída do módulo que está conectado a um pino de saída do DUT. Conforme ilustrado na figura 30, se esses pinos apresentarem estados lógicos diferentes, ocorrerá um curto-circuito que pode resultar em possíveis danos a ambos os circuitos.

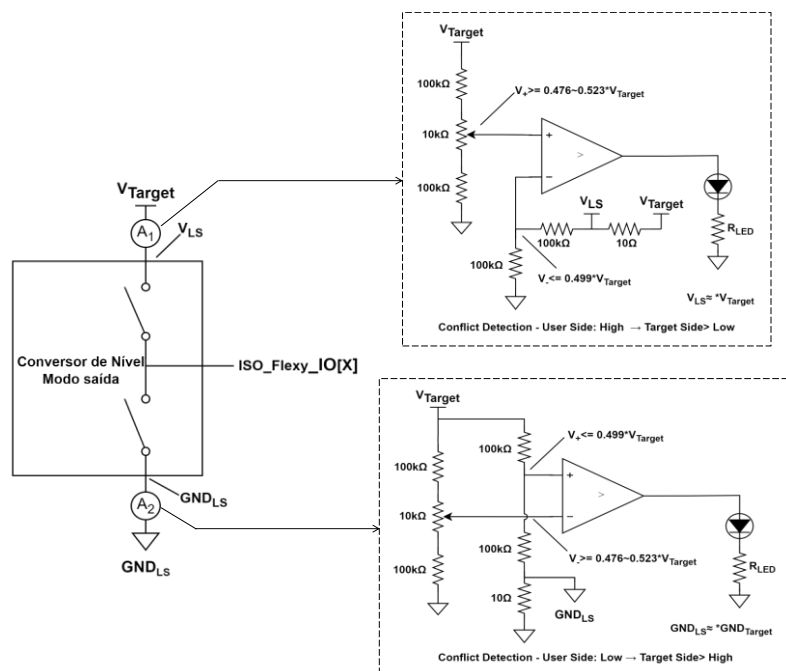
Figura 30 : Tipos de conflitos na saída.



Fonte : Autoria própria

Para atenuar esse problema, utilizamos o circuito de detecção de conflitos de E/S do módulo auxiliar 2, que fornece um retorno visual ao operador ao detectar um curto-circuito, permitindo uma correção imediata. A Figura 31 ilustra os dois circuitos utilizados.

Figura 31 : Topologia do circuito de detecção de conflitos de E/S.



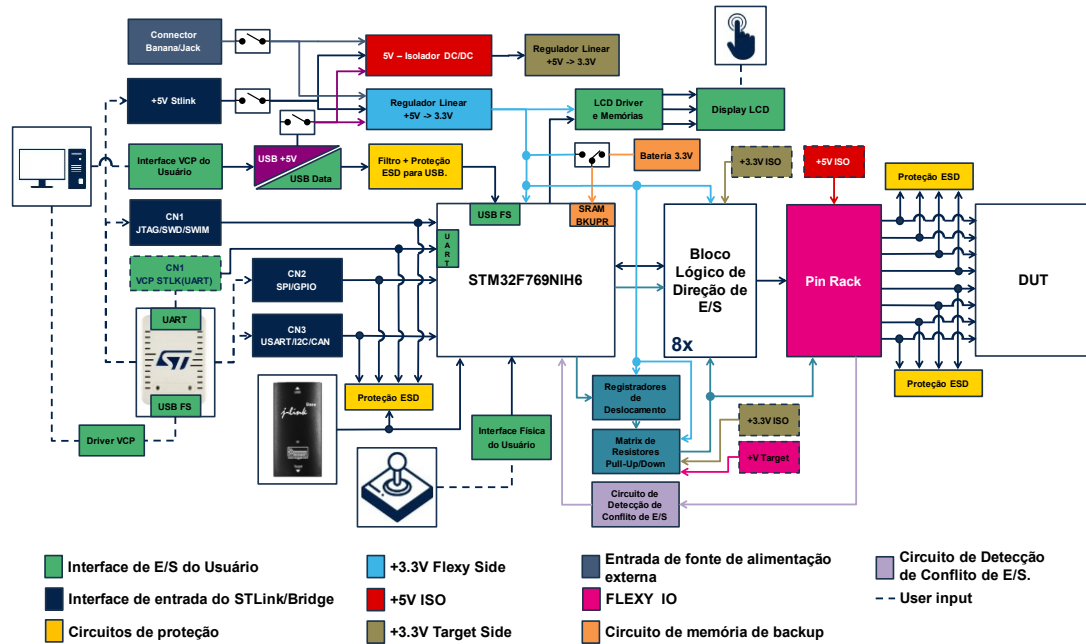
Fonte: Autoria própria.

Essa configuração utiliza dois comparadores que atuam como amperímetros, conectados à alimentação e à referência dos reguladores de nível. Na presença de uma sobrecorrente, a tensão no divisor de tensão varia, acionando um dos comparadores que acende um LED. O limite de corrente pode ser ajustado individualmente para cada circuito por meio de um potenciômetro.

5.3. FLEXY - Diretrizes para o design de hardware

Com base no estudo realizado nas seções anteriores e nos requisitos do sistema, foi elaborado o diagrama funcional do módulo FLEXY. Como ilustrado na Figura 32, o módulo será equipado com quatro modos de alimentação: via STLinkv3, J-Link, USB ou uma fonte externa através de um conector *jack* DC, oferecendo flexibilidade para diferentes cenários de uso. Conectores individuais serão disponibilizados para os depuradores J-Link e STLink, além de um conector USB para conexão com o computador. Por fim, um circuito com bateria e monitoramento será usado para alimentar a memória volátil onde a chave de segurança será armazenada.

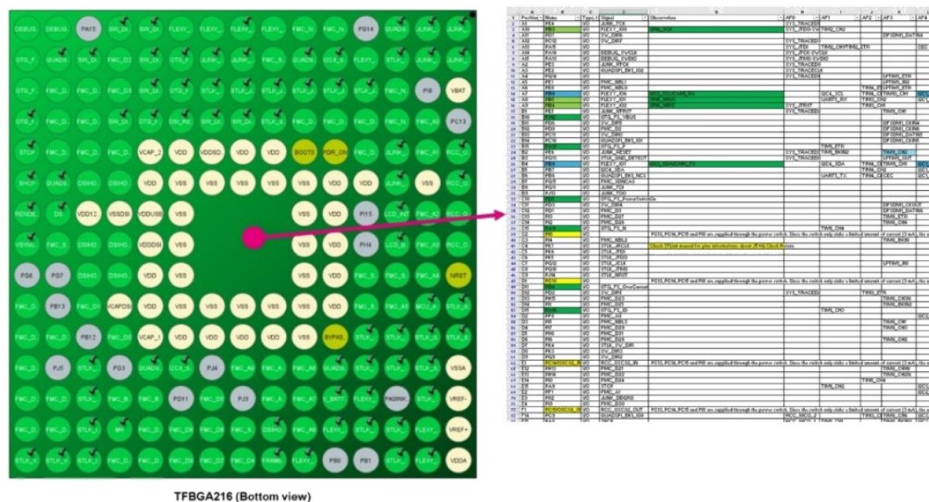
Figura 32 : Diagrama de blocos completo – Depurador FLEXY



Fonte: Autoria Própria.

O microcontrolador STM32F769NIH6 foi escolhido para o projeto devido ao seu excelente desempenho, ideal para aplicações como a gestão e o multiplexação de protocolos. Como ilustrado na Figura 33, o circuito utiliza um encapsulamento TFBGA216 e oferece 159 GPIOs, uma característica essencial para nosso projeto que requer muitas portas disponíveis. Inicialmente, planejamos utilizar três bancos de E/S com oito pinos cada; no entanto, devido à ocupação da maioria desses pinos na configuração atual, nossa escolha inicial se revelou impossível. Cada banco de E/S inclui os circuitos descritos na seção 5.2.

Figura 33 : Pinout FLEXY



Fonte: Autoria Própria.

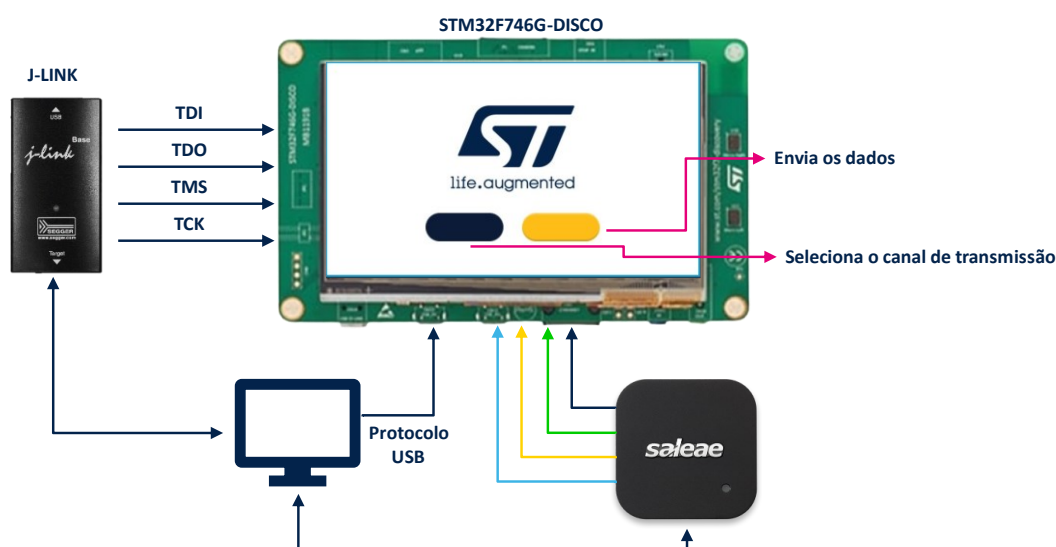
Para a interface gráfica, avaliou-se as soluções presentes nas duas placas DISCO utilizadas no projeto. A STM32F46G-DISCO dispõe de uma interface RGB paralela, que é compatível com muitos displays e de fácil implementação, mas exige um grande número de pinos. Em contrapartida, a STM32F769I-DISCO possui uma interface MIPI DSI, reconhecida pela eficiência e pela significativa redução no número de conexões necessárias. Optou-se pela última devido à limitação de pinos do microcontrolador e à oportunidade de otimizar os custos do projeto, utilizando a solução comercial oferecida pela ST: o *display touch* capacitivo de 4" B-LCD40-DSI1. Assim, não foi necessário desenvolver o módulo gráfico, e pôde-se utilizar parte do esquema eletrônico da STM32F769I-DISCO.

Por fim, optou-se pelo uso dos componentes catalogados durante a análise dos circuitos das placas de referência, o que dispensou a necessidade de validação adicional, uma vez que esses componentes já foram implementados em produtos comerciais. Para componentes obsoletos ou indisponíveis no mercado, foi realizado um estudo para identificar referências compatíveis.

5.4. FLEXY - Prova de Conceito

Com base nas aplicações descritas na seção 5.1.2, desenvolveu-se uma primeira demonstração do princípio de multiplexação de protocolos, com o objetivo de avaliar a viabilidade do projeto em nível de software e identificar as limitações dos microcontroladores da família STM32F7. A montagem está ilustrada na figura 34.

Figura 34 : Montagem para a prova de conceito FLEXY

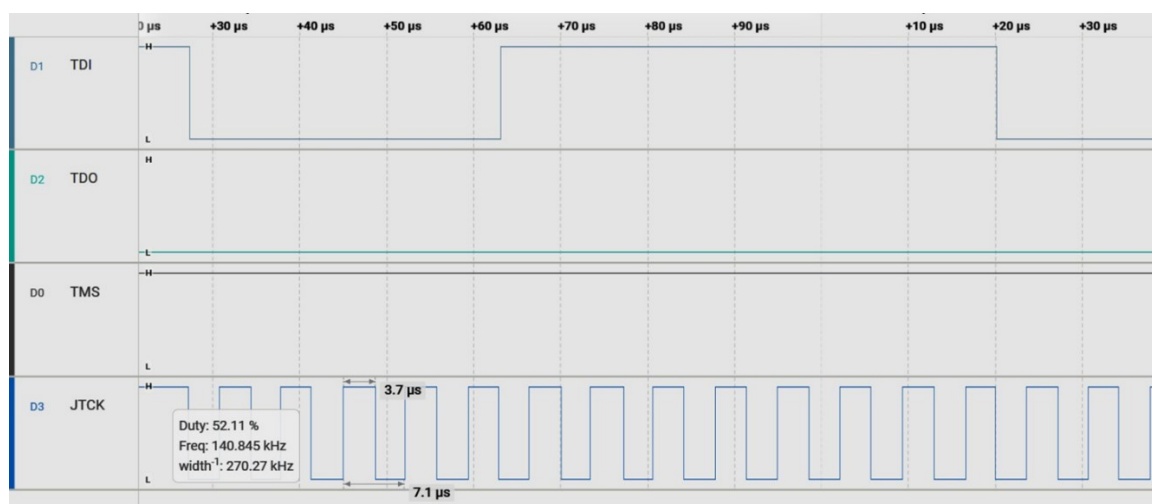


Fonte: Autoria Própria.

A aplicação desenvolvida consiste em uma placa STM32F46G-DISCO conectada a um J-Link. Os sinais emitidos pelo depurador foram programados através da ferramenta auxiliar 1 e replicados em quatro pinos de entrada/saída da placa Discovery, monitorados pelo analisador lógico Saleae. Por fim, o dispositivo foi configurado como um periférico (*Device*) e conectado a um computador (*Host*), que transmite comandos via protocolo USB.

A primeira fase da prova de conceito consistiu em verificar a frequência máxima de aquisição de dados do microcontrolador. Para isso, foram empregadas rotinas de interrupção sincronizadas com as transições de subida e descida do sinal TCK. Essas rotinas têm a função de monitorar o estado dos quatro sinais de entrada e reproduzi-los nas saídas correspondentes. A figura abaixo apresenta uma réplica do cronograma dos dados transmitidos a uma frequência de 140 kHz.

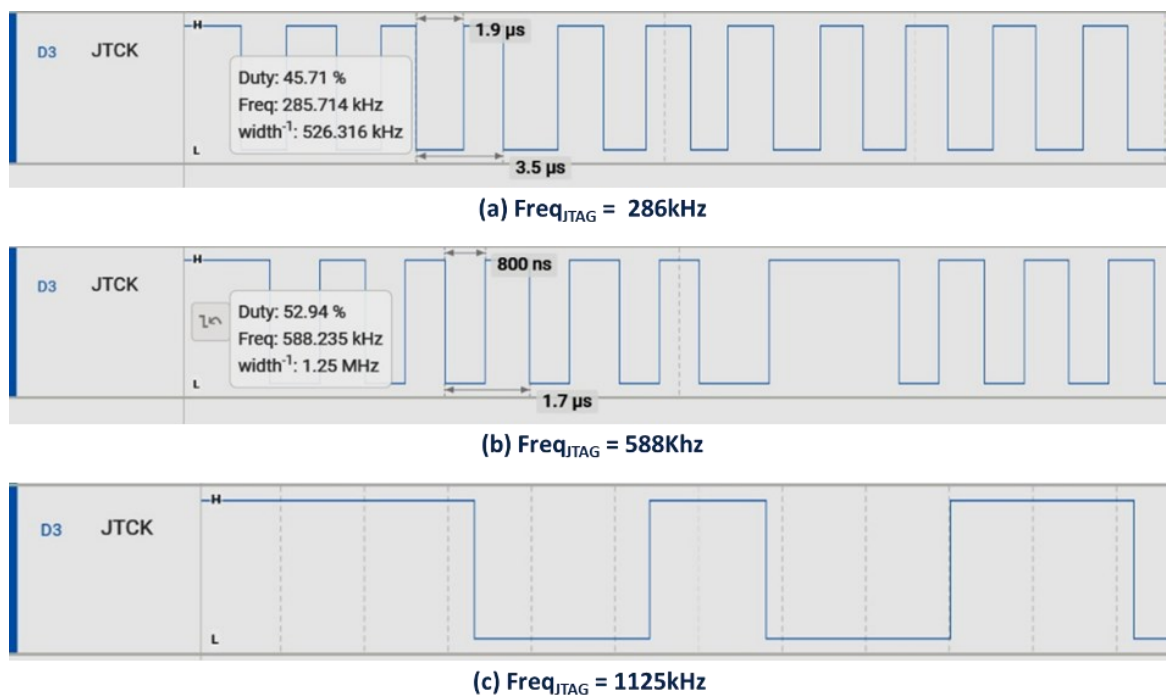
Figura 35 : Cronograma JTAG replicado F = 140kHz.



Fonte: Autoria Própria.

Em seguida, foi realizado um estudo para determinar a frequência máxima do sinal de entrada que o sistema é capaz de amostrar, tomando como referência as frequências padrão do protocolo JTAG. Conforme ilustrado na Figura 36, observou-se que, à medida que a frequência aumenta, o sinal TCK replicado começa a se deformar. A partir de 588 kHz, foram identificadas perdas, e, com uma frequência de entrada de 1,12 MHz, torna-se impossível reconstruir o sinal. Esse fenômeno é atribuído à latência das interrupções, ou seja, ao intervalo entre a geração de uma interrupção por um evento e a execução da rotina de serviço de interrupção (ISR), o que resulta na perda de alguns ciclos de clock.

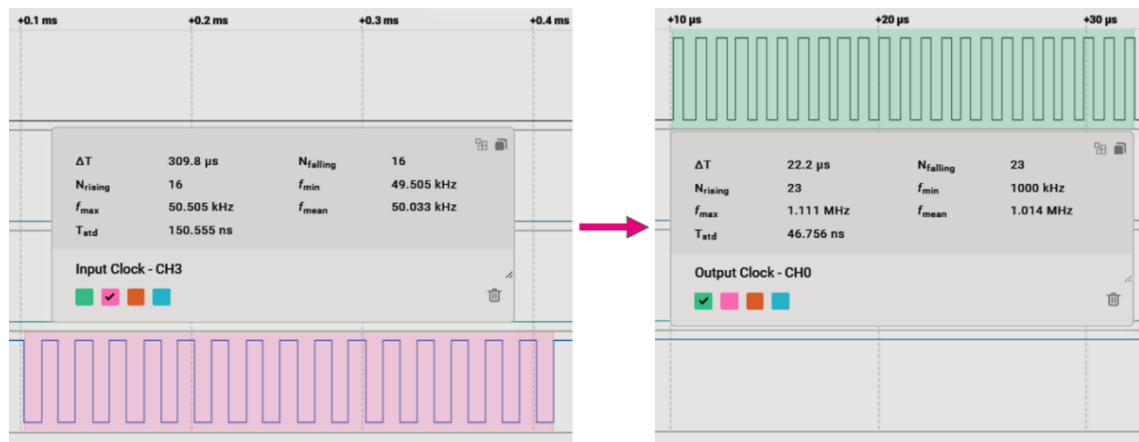
Figura 36 : Degradação do sinal TCK replicado em função de sua frequência.



Fonte: Autoria Própria.

A segunda fase da demonstração teve como objetivo aprimorar a capacidade de retransmissão do microcontrolador por meio da implementação da técnica de *bit banging*. Dessa forma, os dados recebidos pelo módulo são armazenados em um buffer e retransmitidos a uma frequência controlada pelo usuário. Essa abordagem permite garantir a integridade do sinal recebido ao trabalhar com frequências mais baixas na recepção e é particularmente útil em aplicações unidirecionais, nas quais se deseja apenas transmitir dados para o DUT. A retransmissão dos dados armazenados é iniciada pelo acionamento do botão amarelo da interface gráfica. Além disso, ao pressionar o botão azul, o usuário pode alternar o canal de retransmissão, validando assim a capacidade de multiplexação dos sinais. Os resultados dessa etapa estão ilustrados na figura 37.

Figura 37 : Validação do princípio de redirecionamento de E/S e retransmissão via *bit banging* - Canal inicial: CH3 F = 50KHz / Canal final: CH0 F = 1MHz.



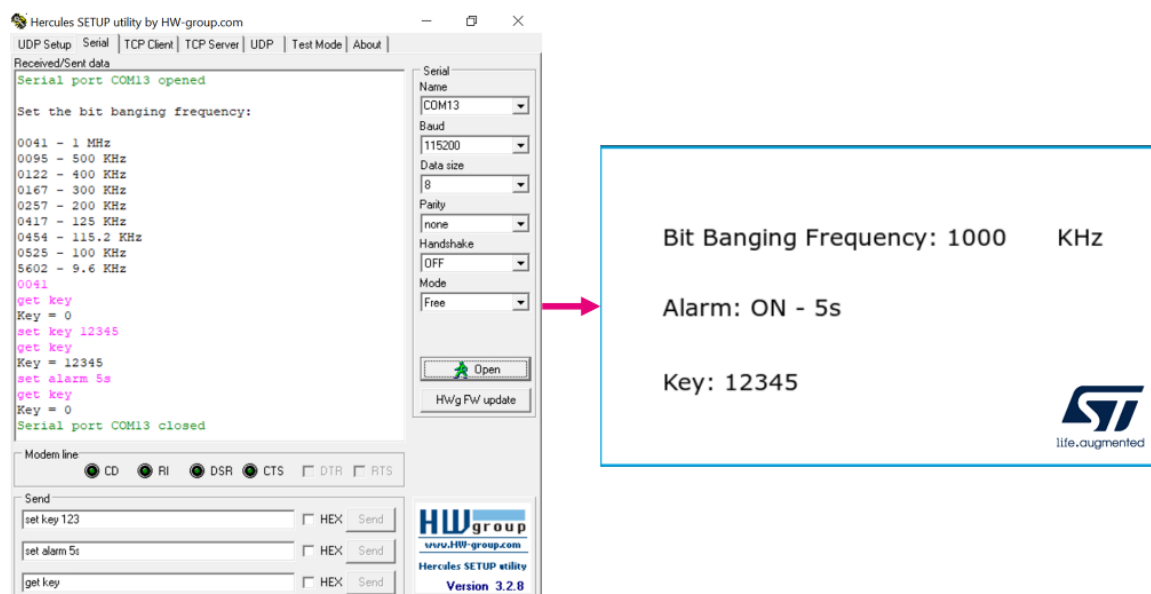
Fonte: Autoria Própria.

A terceira fase do projeto consistiu na integração do firmware desenvolvido com o computador e na implementação de um sistema simples de chaves de segurança. Para essa finalidade, utilizou-se a biblioteca USB fornecida pela ST, em conjunto com o software Hercules da HW Group, que permitiu tanto a transmissão dos comandos do usuário quanto a recuperação dos dados da placa.

Por meio dos comandos implementados, o usuário pode definir a frequência de retransmissão dos dados e enviar um token de segurança. Esse token é armazenado nos registros RTC (Real-Time Clock), também conhecidos como registros de backup, que mantêm os dados em uma memória volátil do tipo SRAM. Ressalta-se que esses dados podem ser redefinidos somente quando a alimentação da memória é interrompida ou por meio de um comando de software.

A Figura 38 apresenta as instruções executadas durante experimento. Inicialmente, uma série de comandos foi enviada para configurar a frequência de transmissão, verificar a chave de segurança armazenada e, em seguida, definir uma nova. Posteriormente, ativou-se um contador para reinicialização com um período de cinco segundos. Após esse intervalo, foi enviado o comando de verificação, constatando-se que o valor armazenado havia sido redefinido para zero.

Figura 38 : Validação da comunicação entre o computador e o microcontrolador via a interface USB e validação do sistema de controle dos registros de segurança.



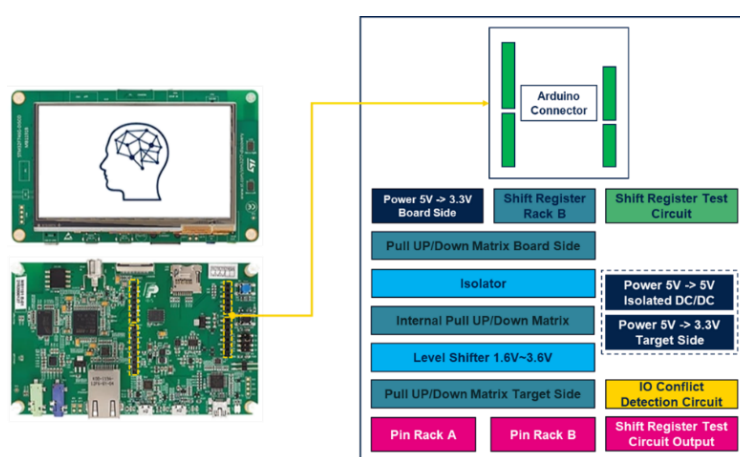
Fonte: Autoria Própria.

Finalmente, os resultados obtidos a partir dessa montagem permitiram verificar a viabilidade do projeto quanto à capacidade de multiplexação de protocolos, gerenciamento de chaves de segurança e integração com os depuradores e o computador, além de identificar suas limitações.

6. Desenvolvimento do primeiro protótipo

Com base nos resultados do estudo de viabilidade, decidiu-se desenvolver um primeiro protótipo para validar os circuitos estudados na seção 5.2. Conforme ilustrado na Figura 39, o protótipo inicial inclui circuitos de teste para a matriz de resistências, o buffer bidirecional para E/S e o circuito de detecção de conflitos entre E/S. Adicionalmente, foram incorporados circuitos de teste para os joysticks, que serão integrados à interface física do sistema, bem como um circuito de alimentação com isolamento galvânico. Por fim, uma interface de conectores permitirá conectar ao protótipo uma placa STM32F769I-DISCO que será responsável por operá-lo.

Figura 39 : Esquema funcional do protótipo 1

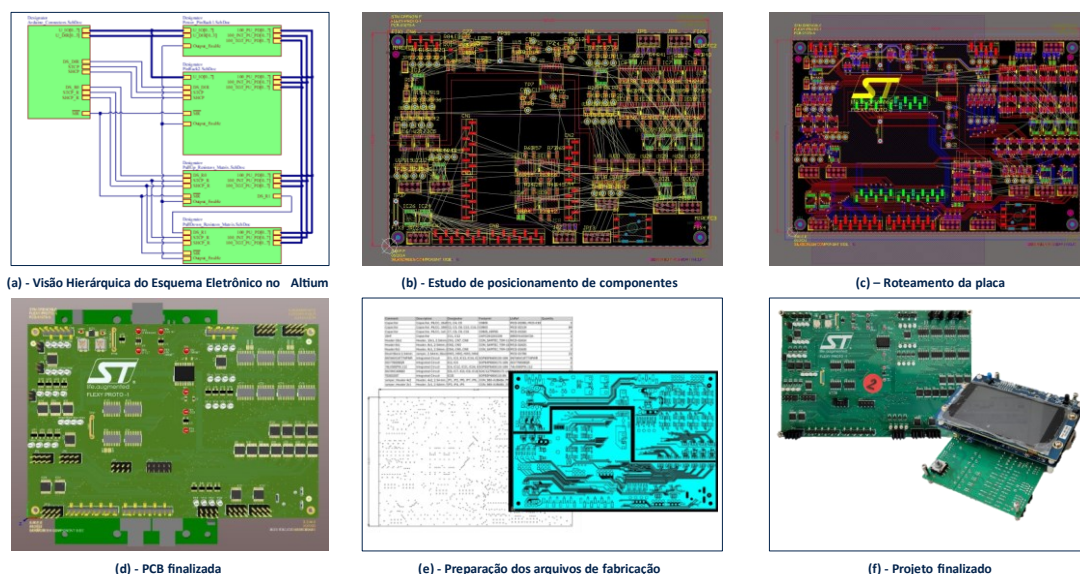


Fonte: Autoria Própria.

O esquema eletrônico da placa foi desenvolvido utilizando o software Altium Designer, enquanto o layout do PCB foi terceirizado para a empresa parceira SMEE. A concepção de um PCB é um processo complexo que envolve diversos fatores, como integridade do sinal, número de camadas, posicionamento dos componentes e largura das pistas. Durante o desenvolvimento do protótipo, várias reuniões foram organizadas para discutir esses aspectos, definir as restrições mecânicas do módulo e substituir componentes com prazos de entrega longos, que poderiam comprometer o cronograma do projeto.

O processo de fabricação, desde a concepção do esquema eletrônico até a entrega do protótipo, teve duração de 48 dias. A Figura 40 ilustra algumas etapas do desenvolvimento do módulo até sua versão final, e foram fabricados três protótipos para realização de testes e validações.

Figura 40 : Etapas de concepção do depurador FLEXY

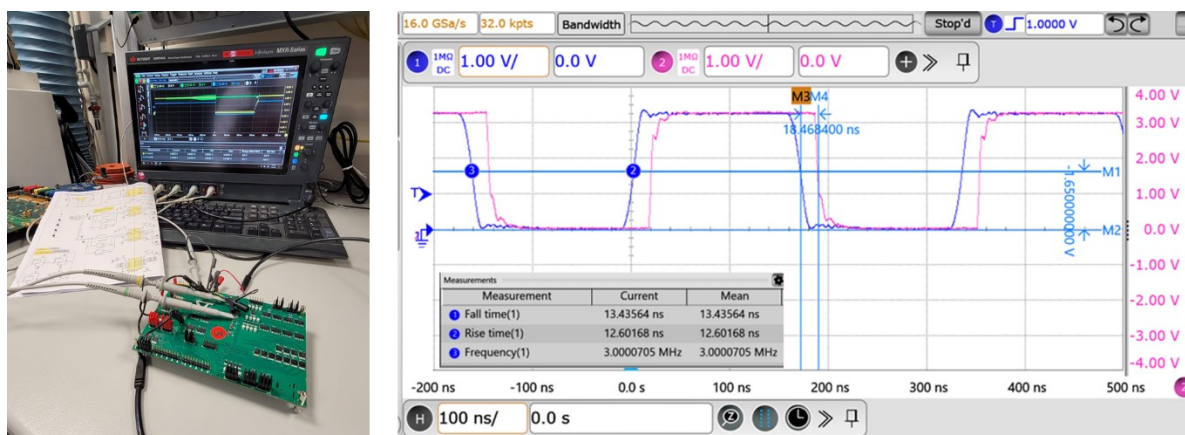


Fonte: Autoria própria.

6.1. Plano de validação, metodologia de teste e resultados

Com a chegada do protótipo, iniciou-se o processo de validação. Essa etapa é fundamental para verificar o funcionamento do protótipo, identificar incoerências no projeto e compreender os limites do circuito, como o tempo de propagação e a frequência máxima de operação. Além disso, busca-se garantir que as grandezas do circuito, como correntes de alimentação máxima e nominal, correspondam aos valores calculados e aos observados na prática. A Figura 41 apresenta uma das montagens realizadas e exibe uma das curvas obtidas durante os testes e validação do módulo. Nessa curva, analisa-se o tempo de propagação entre o sinal de origem em azul e a sua versão replicada em rosa.

Figura 41 : Exemplo de montagem para validação e exemplo de curva analisada.



Fonte: Autoria própria.

A tabela abaixo resume, de forma geral, os testes realizados e seus respectivos resultados. O protótipo 1 apresentou bom desempenho, permitindo validar o funcionamento dos circuitos propostos. No entanto, o circuito de alimentação não operou conforme o esperado; esse ponto será detalhado na seção 6.2.

Tabela 1 : Resultados dos testes e validação do protótipo 1.

Teste	Descrição	Resultado
Inspeção visual da placa	Verificação das soldas, trilhas e número de série dos componentes principais.	✓
Validação do circuito de alimentação	Validação dos níveis lógicos dos circuitos de alimentação e consumo de corrente da placa com o auxílio de um osciloscópio e uma fonte de alimentação externa. Análise da isolação entre os lados primário e secundário.	✗
Medição de corrente de fuga com sonda conectada ao alvo	Conexão de uma alimentação externa em modo 4 quadrantes à saída.	✓
Validação do circuito de detecção de conflitos de E/S	Indução de conflitos entre os pinos de saída com diferentes níveis lógicos.	✓
Validação do circuito de registrador de deslocamento	Validação da configuração do componente com o auxílio do analisador lógico SALEAE. Escrita de 16 bits nos bancos CN7 e CN8.	✓
Validação da matriz de resistências	Validação do cronograma do componente com o auxílio do analisador lógico SALEAE. Escrita de 16 bits nos pontos de teste FLEXY Side/ Internal/ Target Side.	✓
Validação do buffer de E/S	Configuração e validação dos buffers de E/S com o auxílio do analisador lógico SALEAE. Testes de entradas e saídas com o auxílio do osciloscópio e uma fonte de alimentação. Medição dos tempos de subida, descida e propagação.	✓
Verificação da frequência máxima de transferência de dados	Configuração dos modos de saída Push-Pull/Open-Drain e controle dos retardos pela função _NOP%. Variação controlada para o comportamento da frequência.	✓
Validação dos botões do usuário	Configuração da entrada em modo Pull-UP e manipulação do joystick para enviar um nível lógico 0 aos pinos correspondentes. Cronograma validado pelo SALEAE.	✓

Fonte: Autoria própria.

Por fim, o protótipo foi utilizado como ferramenta de validação no desenvolvimento do firmware do módulo FLEXY.

6.2. Dificuldades e pontos de melhoria

Durante a manipulação do protótipo, foi identificado um problema no circuito de alimentação: o isolador DC/DC apresentou dificuldades de partida e não conseguiu fornecer os 5V necessários ao lado secundário. Para dar continuidade aos testes, o componente foi desconectado do circuito, e uma fonte externa foi utilizada para alimentar o lado secundário. Paralelamente ao processo de validação do módulo,

foram formuladas hipóteses sobre o mau funcionamento do circuito, e as fichas técnicas dos componentes utilizados foram analisadas a fim de identificar a possível origem do problema.

Figura 42 : Formas de onda durante a inicialização do isolador 5V-5V. Rosa: 5V FLEXY - Azul: 5V Isolado - Amarelo: saída LDO 5V-3.3V.



(a) Départ incorrect



(b) Départ idéal

Fonte: Autoria própria.

Conforme ilustrado na figura 42, foi observado que durante a partida do isolador o regulador de tensão 5V-3.3V do lado secundário solicitava uma corrente elevada, comprometendo o funcionamento correto do sistema. Ao substituir o isolador por uma fonte externa capaz de fornecer a corrente necessária, o circuito de alimentação partia conforme o esperado e operava normalmente após a remoção da fonte externa. A análise da corrente de partida revelou um pico de aproximadamente 200 mA, excedendo a capacidade máxima de 100 mA do isolador, o que impedia o circuito de alimentação de inicializar corretamente e o colocava em estado de bloqueio.

Após a análise do problema, concluiu-se que a solução consistia em implementar

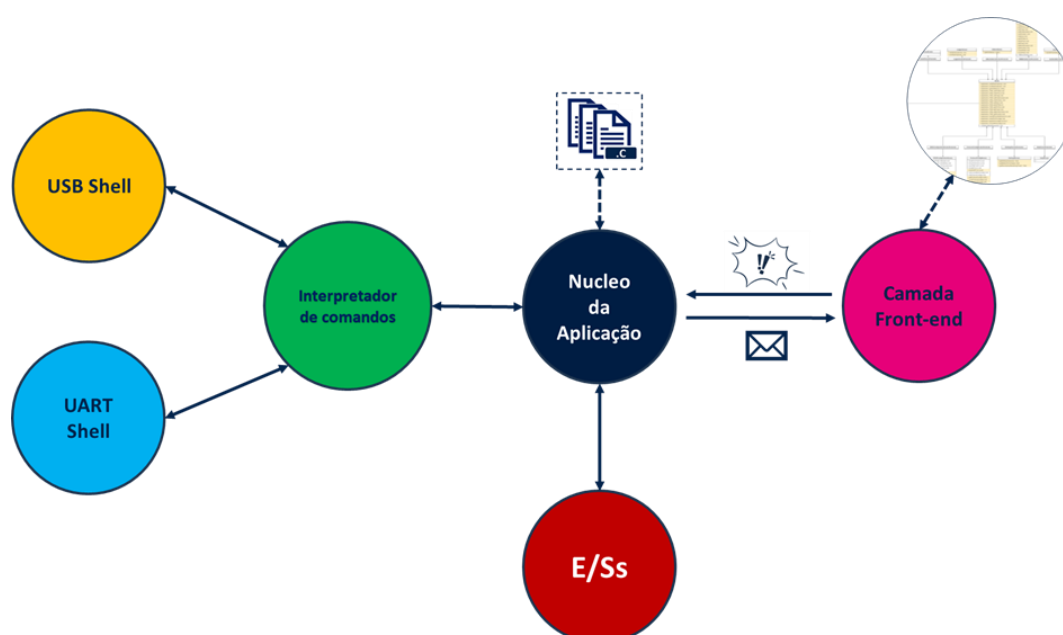
um sequenciamento para a inicialização dos componentes do lado secundário e aumentar a capacidade de corrente do circuito de alimentação. Para isso, novos componentes foram encomendados para realizar testes adicionais e, após a validação, o esquema foi atualizado para o segundo protótipo.

Finalmente, durante a manipulação do protótipo, foi identificado a necessidade de incluir pontos de conexão para fontes de alimentação externas e de utilizar jumpers para isolar circuitos críticos em caso de mau funcionamento.

7. Desenvolvimento do firmware embarcado

Como discutido na seção 1.3, o firmware do módulo FLEXY deve ser capaz de gerenciar o redirecionamento dos sinais recebidos a qualquer saída. Além disso, deve ser capaz de se comunicar com o computador e com os depuradores J-Link e STLink, bem como administrar quatro modos de operação distintos. A figura abaixo apresenta como o firmware foi organizado para atender a esses requisitos.

Figura 43 : Estrutura simplificada do software do módulo FLEXY.



Fonte: Autoria própria.

A camada de *frontend* é responsável por gerenciar a interface do sistema, sendo estruturada segundo o modelo de arquitetura MVP (*Model-View-Presenter*) e gerenciada pelo TouchGFX. Cada tela da interface é representada por uma classe que utiliza um apresentador exclusivo para interagir com o modelo, permitindo uma interface modular e flexível.

O *backend*, desenvolvido em linguagem C, não suporta diretamente o paradigma de programação orientada a objetos. No entanto, foi adotado um modelo de design modular, no qual as funcionalidades são agrupadas de acordo com suas funções e telas correspondentes em arquivos .C. Essa camada é composta por quatro módulos: o *USB shell* e o *UART shell*, que gerenciam, respectivamente, a comunicação via USB e via VCP; o interpretador de comandos, que processa as instruções recebidas e administra os dados a serem transmitidos ao computador; e, por fim, o Núcleo da Aplicação, responsável por processar os dados recebidos do interpretador de

comandos, além de monitorar os estados das E/S e os dados provenientes da camada de *frontend*.

Figura 44 : Menu principal do firmware FLEXY



Fonte: Autoria própria.

A figura 44 ilustra o menu principal da aplicação desenvolvida, onde oito botões direcionam o usuário para diferentes telas:

- Autotest: Permite ao usuário executar uma rotina de verificação para testar o funcionamento dos pinos de entrada e saída do módulo.
- IO Bank Status: Nessa tela, o usuário pode verificar a configuração atual de cada pino de E/S.
- Settings: O menu de configurações possibilita ao usuário visualizar informações sobre o estado do módulo, como a versão do firmware e a carga da bateria que alimenta os registros de backup.
- Help: Oferece um guia de utilização do módulo.
- Logger: Utilizado em modo de depuração, permite ao usuário monitorar os comandos recebidos e seu processamento pelo módulo.
- TME Mode: Dá acesso às funções do módulo auxiliar 1 e desbloqueia o modo de teste nos dispositivos.
- IO Configuration: Permite a configuração individual de pino de E/S do módulo.
- Protocol Configuration: Associa os pinos configurados a um protocolo selecionado pelo usuário.

8. Discussão dos Resultados e Perspectivas Futuras

Pode-se observar ao longo deste documento que a adoção da metodologia em ciclo V foi fundamental para a organização e execução eficaz do projeto. A etapa inicial de estudo das placas de referência permitiu a identificação e reaproveitamento de circuitos que serviram como base para o desenvolvimento, além de possibilitar a criação de novos circuitos ajustados às especificidades do sistema. O estudo do *framework* TouchGFX também proporcionou uma compreensão aprofundada de sua arquitetura e impacto no desenvolvimento de *firmwares* com interfaces homem-máquina, facilitando o progresso nas fases subsequentes.

A prova de conceito desempenhou um papel essencial na identificação das limitações do sistema e na definição de pontos críticos para o seu aprimoramento, orientando ajustes que foram incorporados ao projeto. Com o sucesso no desenvolvimento do primeiro protótipo, foi possível validar o funcionamento do módulo e identificar melhorias a serem implementadas no segundo protótipo. Esse avanço permitiu, ainda, o desenvolvimento parcial do firmware que será integrado à versão final do projeto, consolidando a base para a continuidade e o aperfeiçoamento do sistema.

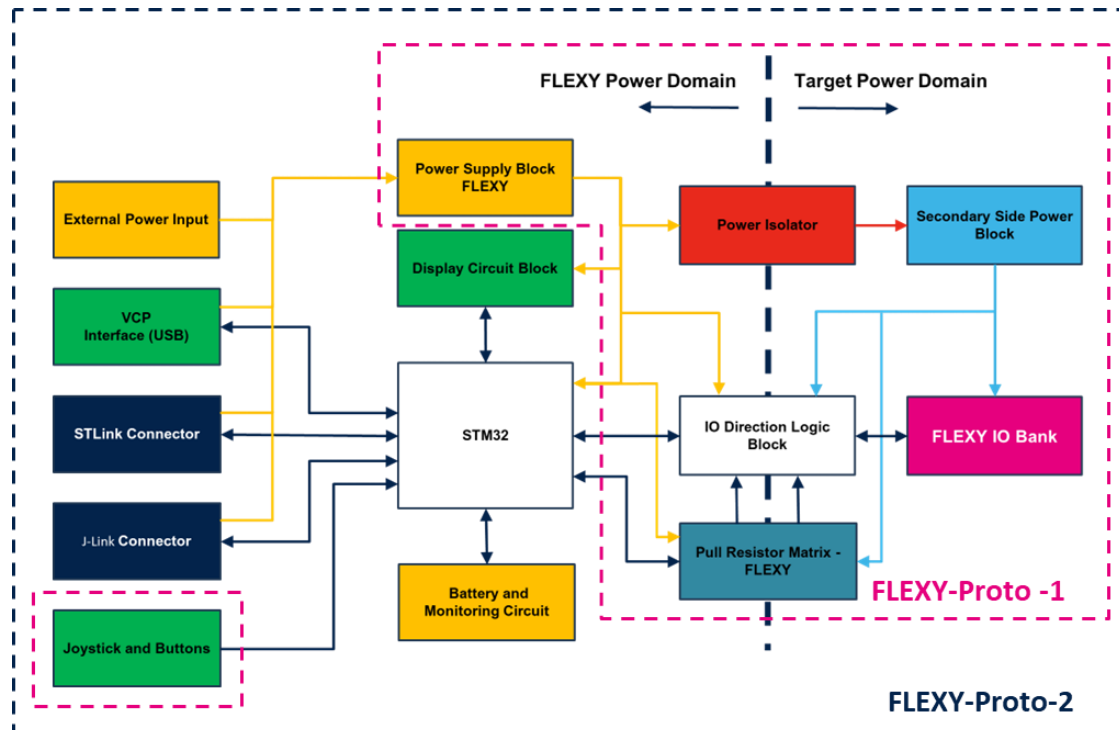
No último mês do projeto, concentrou-se na finalização do esquema eletrônico do segundo protótipo e no acompanhamento do processo de fabricação. Além disso, foi elaborado um plano de validação para o firmware desenvolvido. A Figura 45 ilustra de forma simplificada, em diagrama de blocos, os componentes de cada versão do módulo.

O cronograma apresentado na Figura 19 demonstra a contribuição significativa deste projeto para os dois primeiros módulos, que servirão como base para o desenvolvimento contínuo. Em relação ao módulo 3, a contribuição foi pontual, abrangendo a criação de um dicionário de comandos a serem implementados no software em Python e a integração da camada de comunicação USB, possibilitando a interação entre o módulo FLEXY e um computador.

A documentação do projeto abrange relatórios, apresentações e guias de uso, e todos os repositórios foram armazenados no servidor da equipe. Adicionalmente, todo o software desenvolvido foi documentado em uma *branch* GIT dedicada ao projeto FLEXY e arquivado na base de dados da empresa. Comentários explicativos

foram inseridos no código, acompanhados de um documento que descreve os procedimentos necessários para sua utilização.

Figura 45 : Comparação entre o Protótipo 1 e o Protótipo 2 em nível de hardware



Fonte: Autoria própria.

Atualmente, o segundo protótipo encontra-se em desenvolvimento. A Figura 45 apresenta as diferenças de hardware entre as duas versões. De forma geral, o segundo módulo incorpora todos os circuitos da primeira versão, além de adicionar um microcontrolador, um display, conectores para os depuradores de desenvolvimento, um circuito para verificação da bateria do sistema de armazenamento de chaves e uma interface USB para conexão com o computador.

Por fim, a continuidade do projeto exigirá que o próximo colaborador integre o firmware desenvolvido ao segundo protótipo e avance no desenvolvimento dos módulos 4 e 5, seguindo a documentação fornecida.

9. Conclusão

A realização deste projeto culminou no desenvolvimento de um depurador multiprotocolo multiplexador, projetado para otimizar e automatizar processos de teste e validação de dispositivos eletrônicos em ambientes de semicondutores. Desde o início, a metodologia em ciclo V mostrou-se eficaz, permitindo uma abordagem sistemática e incremental, onde cada etapa foi validada, minimizando retrabalhos e garantindo a confiabilidade do sistema. O estudo de viabilidade das placas de referência e a análise do TouchGFX forneceram a base para projetar circuitos e funcionalidades que atendem às exigências específicas de validação em múltiplos protocolos de comunicação.

Por fim, o projeto proporcionou uma experiência rica em desafios e aprendizado, que contribuíram significativamente para o desenvolvimento das habilidades técnicas e de gerenciamento de projeto. A experiência prática com metodologias de engenharia de testes e desenvolvimento de sistemas embarcados solidificou os conhecimentos adquiridos ao longo do curso e reafirmou a importância de uma abordagem organizada e colaborativa, especialmente em um ambiente de alta complexidade como o de semicondutores.

O primeiro protótipo desenvolvido possibilitou testar e validar as funcionalidades essenciais do sistema, contribuindo para identificar melhorias que foram incorporadas na versão subsequente do projeto. Essa abordagem incremental permitiu o avanço seguro para o segundo protótipo, onde o hardware e o firmware foram refinados para otimizar o desempenho, segurança e integração do depurador com diferentes interfaces de comunicação, como JTAG, SPI, I²C e UART.

Finalmente, o projeto evidenciou a importância de uma documentação rigorosa e a padronização de processos, essenciais para garantir a continuidade e a escalabilidade do trabalho.

Referências Bibliográficas

ALTIVUM. Altium Designer Documentation. Disponível em: <https://www.altium.com/documentation/altium-designer?version=20.0>. Acesso em: 16 mai. 2024. Citado na página 28.

ALTERA CORPORATION (INTEL). AN39: IEEE 1149.1 JTAG Boundary-Scan Testing. San José: ALTERA Corporation, 2002. 44 p. Citado na página 22.

ARM. CMSIS-RTOS2: Generic RTOS Interface. Disponível em: https://arm-software.github.io/CMSIS_5/RTOS2/html/genRTOS2IF.html. Citado na página 38.

BERGER, Arnold S. Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. 2. ed. Burlington: Elsevier Science, 2002. Citado na página 19.

DHAKER, Piyu (Analog Devices). Introduction to SPI Interface. Disponível em: <https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html>. Acesso em: 19 mai. 2024. Citado na página 22.

FORTUNE BUSINESS INSIGHTS. Consumer electronics market. Disponível em: <https://www.fortunebusinessinsights.com/consumer-electronics-market-104693>. Acesso em: 1 nov. 2024. Citado na página 12.

FOUILLEUL, Alex. Développement d'outils Python pour sondes hardware multi-protocoles dédiés à du DFT. Thèse Discipline. Polytech Grenoble - INP, UGA, 2023. 38 p. Citado na página 13.

HYMEL, Shawn (Digikey). Introduction to RTOS - Solution to Part 3 (Task Scheduling). Disponível em: <https://www.digikey.com/en/maker/projects/introduction-to-rtos-solution-to-part-3-task-scheduling/8fbb9e0b0eed4279a2dd698f02ce125f>. Publicado em: 18 jan. 2021. Acesso em: 16 mai. 2024. Citado na página 37.

MESTANZA, Thiago Fernando Cuevas. Desenvolvimento de um Robô Assistente Pessoal para Aprimorar a Autonomia e Qualidade de Vida de Idosos com Limitações Motoras. 2024. 77 f. Tese de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) - Universidade Federal de Uberlândia, Uberlândia, 2024. Citado na página 19.

PEÑA, Eric (Analog Devices). UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. Disponível em: <https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>. Acesso em: 19 mai. 2024. Citado na página 24.

SALEAE. Saleae Logic 8 Analyzer - Debug hardware like the pros. Disponível em: <https://www.saleae.com/products/saleae-logic-pro-16>. Acesso em: 16 mai. 2024. Citado na página 34.

SHANMUGAVEL, B.; T, R.; R, K. Applications and future scope of robotics—a review. Journal of Advanced Robotics Research, v. 32, n. 6, p. 453-460, 2018. Disponível em: https://www.researchgate.net/publication/326293920_Applications_and_Future_scope_of_Robotics-A_Review. Acesso em: 10 nov. 2024. Citado na página 19.

SEGGER. J-Link BASE: The JTAG/SWD debug probe with USB interface. Disponível em: <https://www.segger.com/products/debug-probes/j-link/models/j-link-base>. Acesso em: 16 mai. 2024. Citado na página 32.

STMICROELECTRONICS. Datasheet - STM32F765xx, STM32F767xx, STM32F768Ax, STM32F769xx - Arm Cortex-M7 32-bit MCU. Agosto de 2023. Disponível em: <https://www.st.com>. Acesso em: 7 nov. 2024. Citado na página 21.

STMicroelectronics. STM32Cube. Disponível em: <https://www.st.com/en/ecosystems/stm32cube.html>. Acesso em: 16 mai. 2024. Citado na página 29.

STMicroelectronics. TouchGFX Documentation. Disponível em: <https://support.touchgfx.com/docs/introduction/welcome>. Acesso em: 27 mai. 2024. Citado na página 30.

STMicroelectronics. UM1907 User manual: Discovery kit for STM32F7 Series with STM32F746NG MCU. Genève: STMicroelectronics, 2023. 37 p. Citado na página 31.

STMicroelectronics. UM2033 User manual: Discovery kit with STM32F769NI MCU. Genève: STMicroelectronics, 2023. 47 p. Citado na página 32.

STMicroelectronics. UM2448 User manual: STLINK-V3SET debugger/programmer for STM8 and STM32. Genève: STMicroelectronics, 2023. 51 p. Citado na página 33.

THERON, Maxence. DFT Link Utility V3 - The Definitive User Guide - Revision 1.0. Grenoble: STMicroelectronics, 2022. Citado na página 13.

TEXAS INSTRUMENTS. Universal Synchronous Asynchronous Receive/Transmit USART. Dallas: Texas Instruments. 52 p. Citado na página 24.

TOULSON, R.; WILMSHURST, T. Fast and Effective Embedded Systems Design: Applying the ARM mbed. 2. ed. Newnes, 2017. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780081008805000013>. Acesso em: 10 nov. 2024. Citado na página 19.

TORTOISEGIT. About TortoiseGit. Disponível em:
<https://www.altium.com/documentation/altium-designer?version=20.0>. Acesso em: 16
mai. 2024. Citado na página 28.

WU, Joseph (Texas Instruments). A Basic Guide to I2C. Dallas: Texas Instruments,
2022. 32 p. Citado na página 24.