CEPEDI - Centro de Pesquisa, Desenvolvimento Tecnológico e Inovação

EMBARCATECH – Residência Tecnológica em Sistemas Embarcados

ALPHA SEGURANÇA

Sistema Integrado de Autenticação Multifatorial para Ambientes de Alta Segurança

Trabalho de Conclusão da fase 1

Luis Felipe Pereira de Carvalho

[Número de Matrícula]

São Gonçalo dos Campos - Bahia

2025

CEPEDI - Centro de Pesquisa, Desenvolvimento Tecnológico e Inovação

EMBARCATECH – Residência Tecnológica em Sistemas Embarcados

ALPHA SEGURANÇA

Sistema Integrado de Autenticação Multifatorial para Ambientes de Alta Segurança

Trabalho apresentado como requisito parcial para a obtenção do certificado de conclusão da primeira fase do projeto de residência em sistemas embarcados.

Luis Felipe Pereira de Carvalho

[Número de Matrícula]

São Gonçalo dos Campos - Bahia

2025

1.Introdução

O sistema **Alpha Segurança** é um sistema de segurança multifatorial projetado para ambientes críticos, como portas de acesso, cofres e áreas restritas. A solução integra três métodos de autenticação – senha, reconhecimento de voz e verificação de íris – e utiliza uma arquitetura modular composta por:

• Microcontrolador Raspberry Pi Pico W: Responsável pelo processamento central e gerenciamento das interfaces.

• Sensores e Entradas: Incluindo joystick, microfone e botões, que permitem a captura de dados analógicos e digitais essenciais para os processos de autenticação e interação do usuário.

• Atuadores e Saídas: Compreendendo LEDs indicadores, buzzers, display OLED e LED Matrix, que fornecem feedback visual e sonoro durante as operações do sistema.

• Interfaces de Comunicação: Tais como ADC, PWM, UART, I2C, PIO e GPIO, que asseguram a integração e a comunicação eficiente entre os diversos módulos do sistema.

• Módulo de Display: Com driver SSD1306 para a geração de gráficos e textos via I2C, possibilitando a exibição clara de mensagens e informações de status.

• Módulo Debouncer e Botões: Responsável pelo tratamento dos sinais de entrada, eliminando ruídos (bounce) e garantindo que os acionamentos dos botões sejam processados de forma confiável.

Este documento descreve detalhadamente cada módulo, suas funções, interligações e configurações, permitindo uma compreensão completa do projeto Alfa Segurança.

1.1 Apresentação do Hardware utilizado

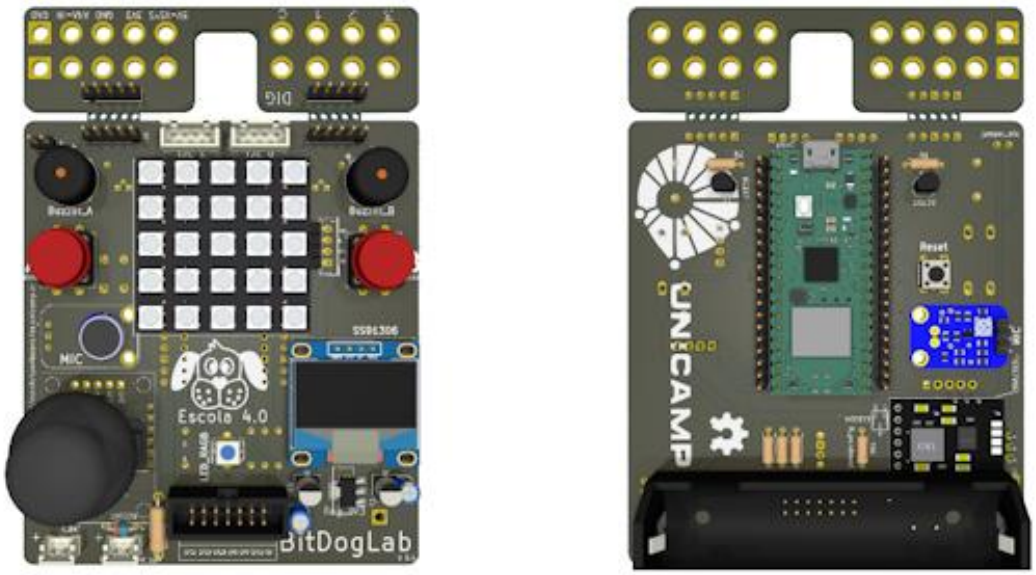
O **Raspberry Pi Pico W** possui o chip **RP2040** com suporte a Wi-fi e bluetooth que é um microcontrolador desenvolvido pela Raspberry Pi Foundation que combina alta performance e baixo consumo energético. O RP2040 possui uma arquitetura de duplo núcleo, 264 KB de memória SRAM e 2 MB de memória flash, o que o torna ideal para aplicações exigentes e sistemas embarcados complexos.

Figura: Raspberry Pi Pico W (RP2040)

Fonte: pishop.us

A **Bitdoglab** é a placa de desenvolvimento desenvolvida pela UNICAMP, projetada para experimentos avançados em sistemas embarcados. Ela integra o Raspberry Pi Pico W e oferece uma plataforma robusta que facilita a interligação de diversos periféricos, como ADC, PWM, UART, I2C, PIO e GPIO. Essa integração permite que todos os módulos do sistema – desde sensores e atuadores até displays e interfaces de comunicação – sejam gerenciados de forma harmoniosa pelo RP2040.

Figura: Placa de Desenvolvimento BitDogLab



Fonte: Github da BitDogLab

Metodologia

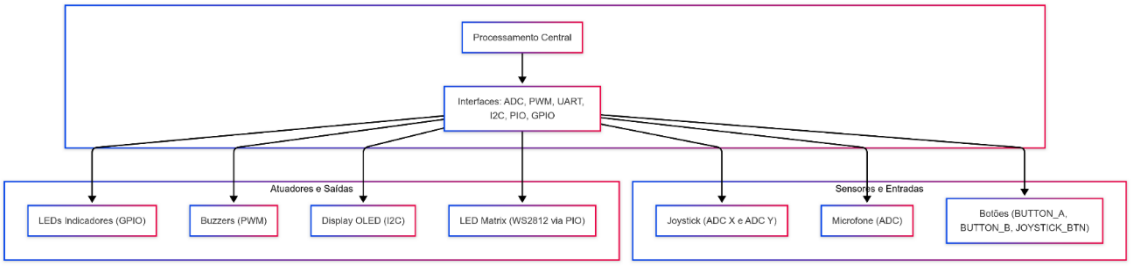
O desenvolvimento do sistema foi organizado em quatro etapas principais, complementares às seções específicas de hardware, firmware e testes:

1. **Levantamento de Requisitos:** Realizou-se uma análise de casos de uso e estudo das tecnologias, definindo as necessidades funcionais e de desempenho do sistema.
2. **Definição das Funcionalidades:** Foram estabelecidas as principais funções do software, com a divisão modular do firmware para gerenciar os métodos de autenticação e implementar as interfaces de usuário e feedback.
3. **Configuração do Ambiente de Desenvolvimento:** Preparou-se a IDE e estruturou-se o projeto em módulos separados para facilitar a manutenção e escalabilidade, abrangendo tanto funções de baixo nível quanto de alto nível.
4. **Depuração e Validação:** Adotaram-se práticas de debugging e ajustes de timing para garantir a correta integração dos módulos e a robustez do sistema durante a operação.

Mais detalhes da metodologia serão explicados nas sessões especificas que tratam do funcionamento das partes do sistema.

2. Visão Geral do Sistema

2.1. Diagrama Geral do Sistema

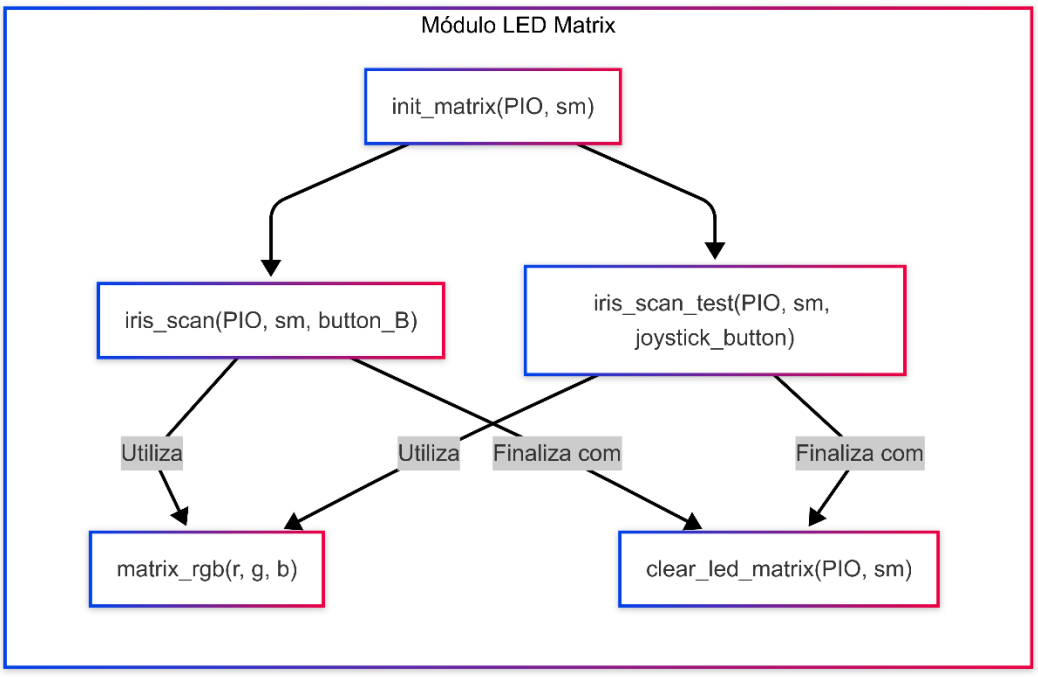


Descrição Geral:

- **Processamento Central:** O firmware no Raspberry Pi Pico orquestra a aquisição de dados dos sensores, processa as informações e emite comandos para os atuadores.
- **Interfaces:** Cada tecnologia (ADC, PWM, UART, I2C, PIO e GPIO) conecta o microcontrolador aos dispositivos periféricos.
- **Sensores/Entradas:** Responsáveis pela coleta de dados do ambiente e interação do usuário.
- **Atuadores/Saídas:** Exibem informações e fornecem feedback visual e sonoro (display, LED Matrix, LEDs, buzzers).

3. Módulo LED Matrix

3.1. Diagrama de Bloco – LED Matrix



3.2. Descrição Detalhada

- **init_matrix(PIO, sm):** Carrega o programa específico no PIO para controlar a matriz de 5x5 LEDs

(WS2812) no pino definido. Esta função prepara a state machine para enviar dados de cor à matriz.

- **matrix_rgb(r, g, b):**
  Converte valores de intensidade (0.0 a 1.0) para um único valor uint32_t no formato RGB, padronizando a comunicação de cores com os LEDs.

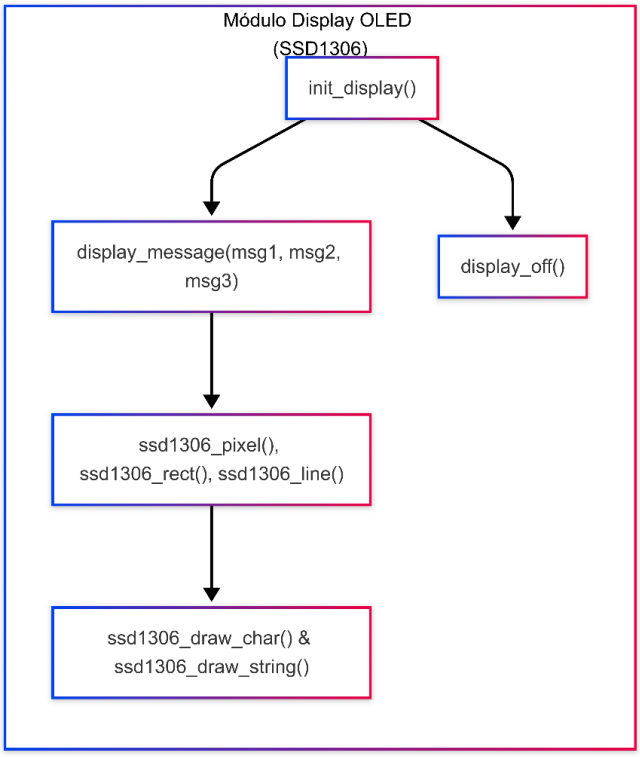
- **clear_led_matrix(PIO, sm):**
  Apaga a matriz de LEDs enviando valor zero a cada LED, garantindo que a tela esteja limpa para novas exibições.

- **iris_scan(PIO, sm, button_B):**
  Realiza a simulação da leitura da íris: exibe um frame azul padrão, aguarda um intervalo e, dependendo se o botão B é pressionado, exibe um frame vermelho (erro) ou verde (acesso concedido).

- **Display():**
  Executa um teste dinâmico da matriz, variando as cores e monitorando o botão do joystick para detectar problemas no scanner.

4. Módulo Display – Driver SSD1306 e Interface I2C

4.1. Diagrama de Bloco – Display



4.2. Descrição Detalhada

- **init_display():**
  Inicializa a interface I2C configurando os pinos SDA e SCL com resistores de pull-up e define a comunicação com o display SSD1306. O display é limpo e preparado para receber comandos gráficos.

- **display_message(msg1, msg2, msg3):**
  Limpa o display e exibe três linhas de texto em posições predefinidas. Essa função é utilizada para fornecer feedback visual durante processos de autenticação e testes.

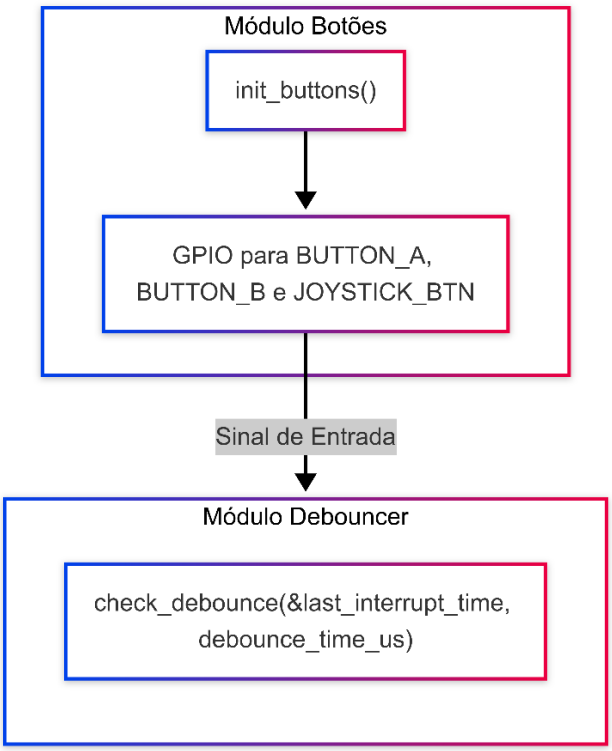
- **display_off():**
  Desliga o display limpando seu conteúdo, útil para economizar energia ou indicar que nenhuma informação está sendo exibida.

- **Funções Gráficas (ssd1306_pixel, ssd1306_rect, ssd1306_line):**
  São responsáveis por desenhar pixels, retângulos e linhas na tela, permitindo a criação de gráficos e layouts personalizados.

- **Funções de Texto (ssd1306_draw_char e ssd1306_draw_string):**
  Utilizam uma fonte pré-definida para desenhar caracteres e strings, possibilitando a exibição de informações textuais com clareza.

5. Módulo Debouncer e Botões

5.1. Diagrama de Bloco – Debouncer e Botões



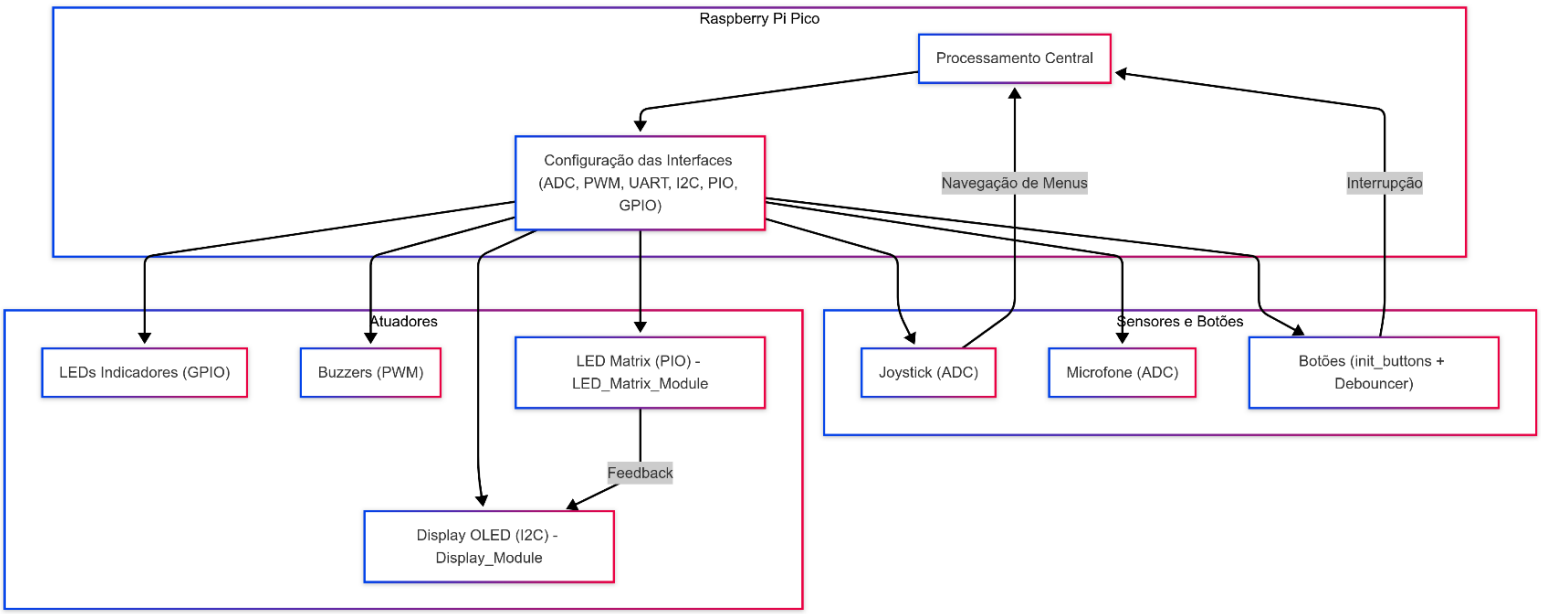
5.2. Descrição Detalhada

- **init_buttons():**
  Inicializa os pinos configurados para os botões (BUTTON_A, BUTTON_B e JOYSTICK_BTN) como entradas com resistores de pull-up. Essa configuração garante que os sinais sejam estáveis e evita falsos disparos devido a ruídos.

- **check_debounce():**
  Implementa uma lógica de "debounce" que verifica se o intervalo de tempo definido (em microssegundos) foi ultrapassado desde a última interrupção. Essa função é essencial para evitar múltiplas detecções de um único acionamento de botão devido a oscilações mecânicas.

6. Integração e Fluxo de Dados Entre os Módulos

6.1. Diagrama de Integração dos Módulos



6.2. Fluxo de Operação

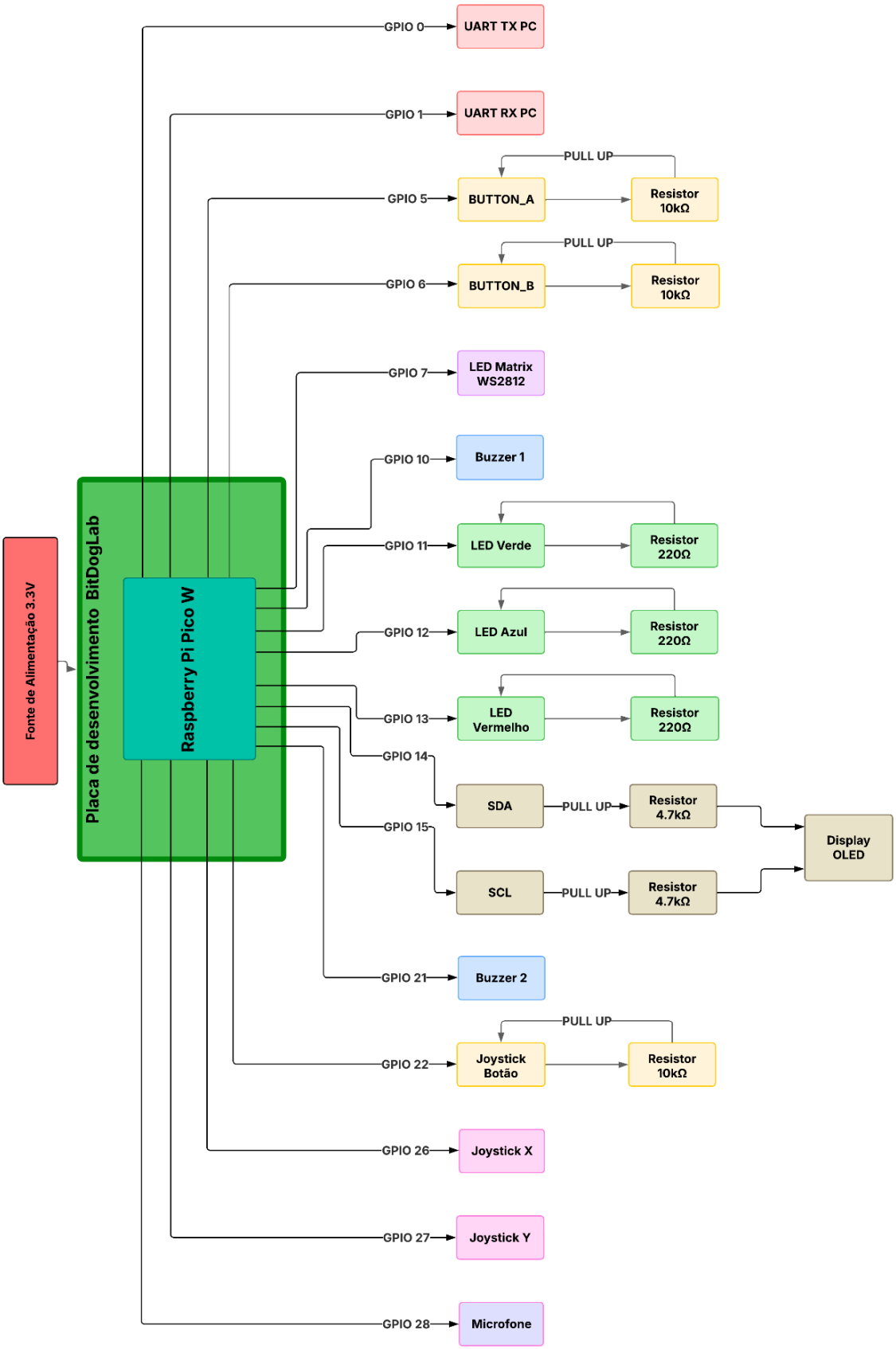
1. **Entrada e Processamento:**
    - O usuário interage via botões e joystick.
    - O módulo Debouncer garante que os sinais sejam processados sem ruídos.
    - Dados analógicos do joystick e microfone são lidos via ADC.
2. **Autenticação e Feedback:**
    - Após a validação de senha e reconhecimento de voz, o sistema aciona o módulo LED Matrix para exibir a verificação de íris.
    - O Display OLED mostra mensagens de status (ex.: “IRIS RECONHECIDA”, “ERRO NO LEITOR”) e informações de navegação no menu.
    - Buzzers fornecem feedback sonoro para acertos ou erros.
3. **Comunicação e Atualização:**
    - O microcontrolador utiliza UART para comunicação serial com um PC, permitindo a entrada de dados e depuração.
    - Funções de desenho (no módulo Display) atualizam o conteúdo gráfico e textual do SSD1306 via I2C.

7. Descrição da Pinagem e Configuração dos Blocos

Função	Pino	Descrição/Observações
LED Indicador Verde	11	Indica operação normal ou sucesso.
LED Indicador Vermelho	13	Indica erro ou alerta.
LED Indicador Azul	12	Usado para testes e feedback adicional.
Buzzer 1	10	Emite sinais sonoros via PWM.
Buzzer 2	21	Complementa o feedback sonoro.
Microfone	28	Entrada analógica (ADC) para captação de áudio.
Joystick ADC X	26	Leitura analógica da posição horizontal do joystick.
Joystick ADC Y	27	Leitura analógica da posição vertical do joystick.
Joystick BTN	22	Botão integrado do joystick para ações adicionais.
Display OLED (I2C)	SDA/SCL 14/15	Comunicação I2C para controle do display SSD1306.
LED Matrix (WS2812)	7	Linha de dados para controle da matriz de 5x5 via PIO.
UART TX	0	Comunicação serial (envio de dados).
UART RX	1	Comunicação serial (recepção de dados).
Botões (BUTTON_A e BUTTON_B)	A/B 5/6	Usados para navegação e acionamento de funções no menu.

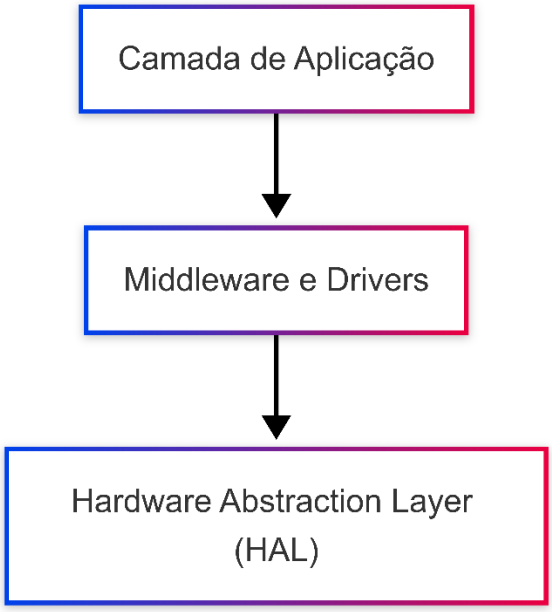
8. Circuito Completo do Hardware

A seguir, é apresentado um esboço do circuito que integra todos os módulos:



9. Blocos Funcionais – Camadas do Software

Diagrama das Camadas do Firmware



- **Camada de Aplicação:**
  Responsável pelas funções de alto nível, como o controle do fluxo de autenticação (senha, reconhecimento de voz e verificação de íris), navegação no menu, testes dos componentes e respostas aos eventos do usuário.

- **Middleware e Drivers:**
  Agrupa os módulos que interagem diretamente com os dispositivos periféricos, como o driver do Display (SSD1306), o módulo LED Matrix, funções de debounce para os botões e os drivers de comunicação (UART, I2C, PIO). Essa camada fornece uma abstração para que a camada de aplicação não precise lidar com detalhes de baixo nível.

- **Hardware Abstraction Layer (HAL):**
  Engloba as bibliotecas e funções que configuram e gerenciam o acesso direto aos periféricos do microcontrolador, como ADC, PWM, UART, I2C, PIO e GPIO, garantindo que os registros e interfaces estejam corretamente configurados para a operação do sistema.

9.2. Descrição das Funcionalidades dos Blocos de Software

- **Camada de Aplicação:**

- **Processamento de Autenticação:** Realiza a sequência de verificação (senha, voz e íris).
- **Navegação e Interface do Usuário:** Gerencia o menu, atualiza a seleção com base no joystick e exibe mensagens no display OLED.
- **Testes e Diagnósticos:** Executa funções de teste para o LED Matrix, buzzers, microfone e leitor de íris, fornecendo feedback ao usuário.

- **Middleware e Drivers:**
  - **Driver do Display (SSD1306):** Inicializa o display via I2C, gera pixels, retângulos, linhas e desenha caracteres e strings com base em uma fonte pré-definida.
  - **Driver LED Matrix:** Controla a matriz de LEDs (5x5) através do PIO, gerando padrões para simulação do leitor de íris e testes dinâmicos.
  - **Módulo Debouncer:** Implementa funções para eliminar ruídos (bounce) dos sinais dos botões, usando um timer para verificar intervalos mínimos entre acionamentos.
  - **Gerenciamento de Botões:** Inicializa e configura os botões com resistores de pull-up, além de associar interrupções para ações imediatas.
- **Hardware Abstraction Layer (HAL):**
  - **Configuração de Periféricos:** Inicializa e configura os módulos ADC, PWM, UART, I2C, PIO e GPIO.
  - **Acesso Direto aos Registros:** Utiliza funções de baixo nível (por exemplo, gpio_set_function, pwm_set_wrap, adc_select_input) para configurar os registros dos periféricos conforme as necessidades do firmware.

10. Definição das Principais Variáveis

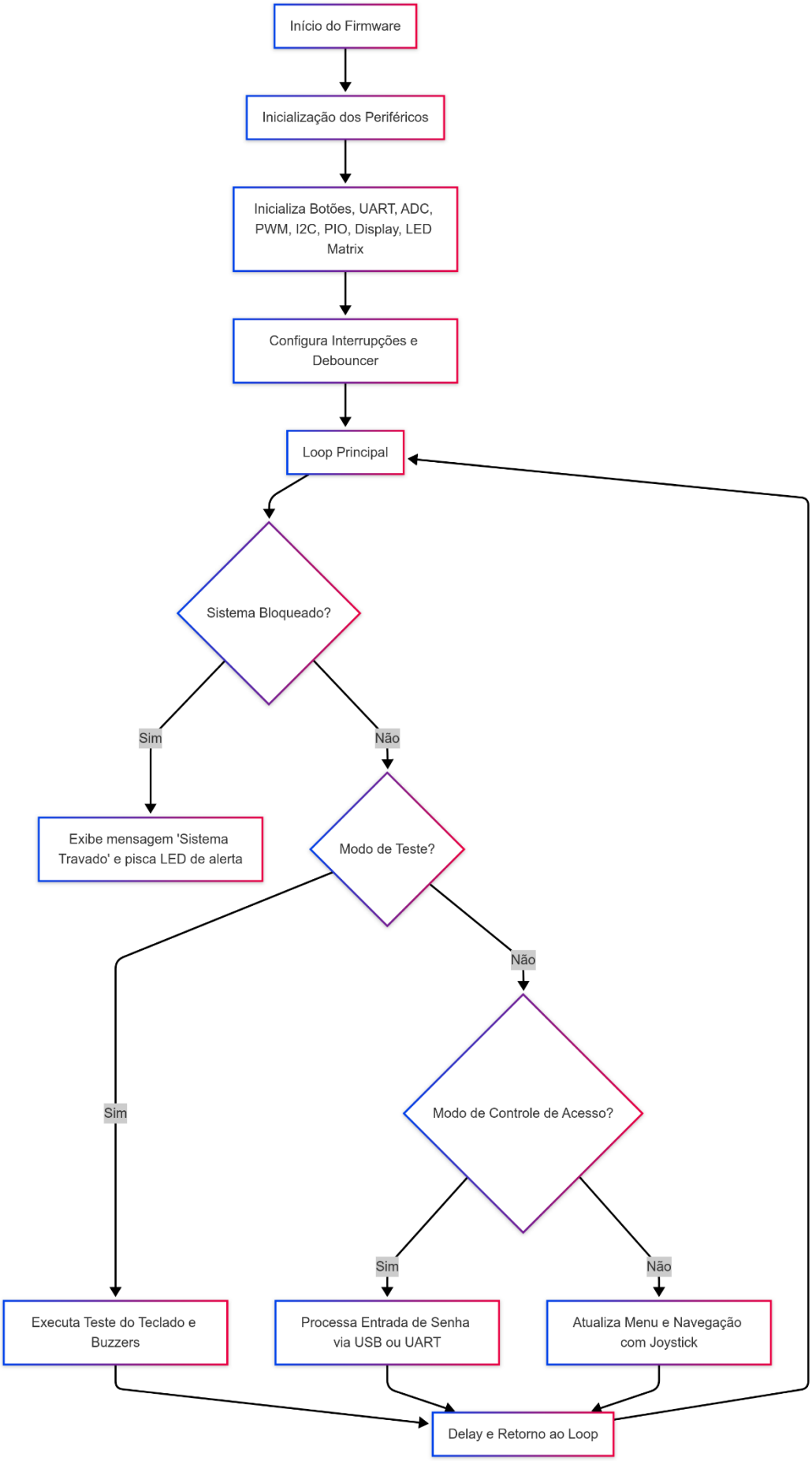
- **Variáveis de Controle de Acesso:**
  - volatile bool access_control_mode;
    Indica se o sistema está no modo de controle de acesso (entrada de senha, voz, etc).

- char entered_code[CODE_LENGTH + 1];
  Armazena os dígitos digitados para a senha.
- int code_index;
  Índice para o preenchimento da senha.

- **Variáveis para Debounce e Interrupções:**
  - uint32_t last_interrupt_time_A, last_interrupt_time_B, last_interrupt_time_JOYSTICK;
    Registram o tempo do último acionamento dos botões, garantindo que o debounce seja aplicado.
- **Variáveis de Status e Flags:**
  - volatile bool keypad_test_mode;
    Indica se o sistema está no modo de teste do teclado.
  - bool keypad_fault, buzzer_fault;
    Flags para sinalizar falhas em componentes.
  - bool scan_problem;
    Indica problemas detectados no leitor de íris (através do LED Matrix).
- **Estrutura do Display:**
  - ssd1306_t ssd;
    Estrutura que mantém os parâmetros e buffers do display SSD1306 (largura, altura, buffer de RAM, etc).
- **Buffers e Dados do Display:**
  - ssd->ram_buffer
    Buffer que armazena os dados a serem enviados para o display, com um tamanho calculado a partir do número de páginas e da largura do display.

11. Fluxograma Completo do Firmware

Fluxo Principal do Firmware


Início do Firmware
Inicialização dos Periféricos
Inicializa Botões, UART, ADC, PWM, I2C, PIO, Display, LED Matrix
Configura Interrupções e Debouncer
Loop Principal
Sistema Bloqueado?
Sim
Não
Exibe mensagem 'Sistema Travado' e pisca LED de alerta
Modo de Teste?
Sim
Não
Modo de Controle de Acesso?
Sim
Não
Executa Teste do Teclado e Buzzers
Processa Entrada de Senha via USB ou UART
Atualiza Menu e Navegação com Joystick
Delay e Retorno ao Loop

Descrição do Fluxograma:

- O firmware inicia com a **inicialização dos periféricos** e a configuração das interfaces (botões, UART, ADC, PWM, I2C, PIO e GPIO).
- Em seguida, o **debouncer** e as interrupções são configurados para garantir que os eventos de entrada sejam tratados corretamente.
- No **loop principal**, o firmware verifica se o sistema está bloqueado ou se está em algum modo específico (teste ou controle de acesso) e, de acordo com a condição, executa as funções correspondentes.
- Após processar as entradas (senha, testes ou navegação de menu via joystick), o sistema aguarda um curto período antes de reiniciar o ciclo.

12. Processo de Inicialização do Firmware

1. **Inicialização das Bibliotecas e Periféricos:**
    - stdio_init_all() para inicializar a comunicação padrão (USB/ serial).
    - Inicialização do ADC para leitura dos sinais do joystick e microfone com adc_init() e adc_gpio_init().
    - Configuração do PWM para os buzzers com gpio_set_function(pin, GPIO_FUNC_PWM) e funções PWM.
    - Inicialização da UART com uart_init() e configuração dos pinos TX e RX.
    - Configuração do I2C para o display, incluindo a ativação de resistores de pull-up.
    - Configuração do PIO para o controle da LED Matrix via init_matrix().
2. **Configuração dos Botões e Debouncer:**
    - Os botões são configurados como entradas com gpio_set_dir() e gpio_pull_up(), utilizando a função init_buttons().
    - O debouncer é preparado através da função check_debounce() para evitar múltiplos acionamentos.
3. **Inicialização dos Módulos de Interface:**

- **Display:** Chama init_display() para configurar o SSD1306, limpar a tela e preparar os comandos gráficos.
- **LED Matrix:** Inicializa a matriz de LEDs para o controle do leitor de íris e testes.

4. **Configuração de Interrupções:**
   - As interrupções dos botões são configuradas com gpio_set_irq_enabled_with_callback(), associando-as à função de callback que gerencia as ações de menu e testes.

13. Configurações dos Registros

- **UART:**
  Configurada via uart_init(UART_ID, BAUD_RATE), onde os registros internos definem a taxa de transmissão (115200 bps), o tamanho dos dados (8 bits), e o modo sem paridade.
- **ADC:**
  Inicializado com adc_init() e configurado com adc_gpio_init() para os canais dos sensores. O registro interno do ADC é ajustado com adc_select_input() para a leitura dos canais.
- **PWM:**
  Utiliza funções como pwm_set_wrap() e pwm_set_chan_level() para definir a frequência e o ciclo de trabalho dos buzzers.
- **I2C:**
  Inicializado com i2c_init(I2C_PORT, 400 * 1000), configurando os pinos SDA/SCL com resistores de pull-up e os registros do controlador I2C para comunicação estável com o display.
- **PIO:**
  O programa PIO é carregado e inicializado com pio_add_program() e pio_matrix_program_init(), configurando a state machine para enviar os dados à LED Matrix.
- **GPIO:**
  Cada pino é configurado com gpio_init(), gpio_set_dir() e, para os botões,

gpio_pull_up(). As interrupções são registradas para responder a eventos de entrada.

14. Estrutura e Formato dos Dados

- **Display OLED (SSD1306):**
  - **Buffer de RAM:** ssd->ram_buffer contém os dados dos pixels do display.
  - **Formato:** O primeiro byte é o controle (0x40 para dados), seguido por bytes que representam os pixels organizados em páginas (cada página contém 8 linhas de pixels).
- **LED Matrix:**
  - Cada LED recebe um valor de 32 bits (formato RGB) gerado pela função matrix_rgb().
- **Dados de Autenticação:**
  - **Senha:** Armazenada em entered_code (string com tamanho definido).
  - **Flags de Status:** Variáveis booleanas (ex.: access_control_mode, keypad_test_mode, scan_problem) para controlar o fluxo e o estado do sistema.
- **Buffer de Comunicação UART:**
  - Utilizado para enviar e receber caracteres individuais (como dígitos da senha e feedback, representado por '*' para cada dígito).

15. Testes de Validação

Para assegurar o funcionamento e a confiabilidade do sistema, foram realizados os seguintes testes:

- **Teste de Autenticação por Senha:**
  Foram simuladas diversas entradas de senha via USB/UART, com testes para senhas corretas e incorretas. O feedback visual (display OLED e LEDs) e sonoro (buzzers) foi verificado em cada cenário.
- **Teste de Reconhecimento de Voz:**
  Utilizando o microfone, foram realizados testes de captação de áudio com

diferentes níveis de som. Os valores lidos pelo ADC foram comparados com o limiar pré-definido para confirmar a resposta do sistema.

- **Teste de Leitor de Íris (LED Matrix):**
  Foram executados os testes do módulo LED Matrix (através das funções iris_scan() e iris_scan_test()), verificando a exibição de frames em cores (azul, vermelho e verde) e a reação a interferências (como o pressionamento do botão B ou do joystick).

- **Teste de Botões e Debouncer:**
  Foram validados os tempos de debounce para evitar múltiplos acionamentos e garantir que os botões (BUTTON_A, BUTTON_B e JOYSTICK_BTN) respondessem apenas aos eventos intencionais.

- **Teste de Comunicação I2C e UART:**
  A comunicação com o display OLED e com o PC via UART foi monitorada para assegurar a integridade dos dados e a atualização correta das mensagens.

Discussão dos Resultados

- **Confiabilidade do Sistema:**
  Os testes demonstraram que o sistema responde de forma consistente às entradas do usuário. O sistema de autenticação multifatorial foi validado, com cada método (senha, voz e íris) apresentando feedback claro e confiável.

- **Integração dos Módulos:**
  A arquitetura modular permitiu que cada componente (display, LED Matrix, botões e sensores) funcionasse de maneira independente e integrada, facilitando a identificação e correção de possíveis falhas.

- **Desempenho e Robustez:**
  A implementação dos mecanismos de debounce e a correta configuração dos periféricos (através dos registros) asseguraram que os sinais de entrada fossem processados sem ruídos, contribuindo para a robustez do sistema.

- **Feedback ao Usuário:**
  A combinação de feedback visual (display OLED e LEDs) e sonoro (buzzers) forneceu uma experiência de usuário satisfatória, permitindo a rápida identificação de erros e a confirmação de sucessos na autenticação.

17. Originalidade

A pesquisa sobre soluções correlatas mostrou que existem diversos projetos que exploram o uso do Raspberry Pi Pico, especificamente para funções de controle de acesso e interação com múltiplos periféricos, mas nenhum deles integra exatamente o conjunto de funcionalidades presente no código apresentado e detalhado nos tópicos anteriores. Foram encontradas as seguintes soluções correlatas, os links estão dispostos nas referências:

Controle de Acesso com RFID e MQTT: Este sistema integra tecnologia RFID com comunicação MQTT, permitindo controle de acesso remoto e monitoramento em tempo real. Embora utilize o Raspberry Pi Pico, foca na autenticação via RFID e comunicação em rede (1).

Controle de Pinos GPIO com Teclado Matricial 4x4: Projeto que implementa o controle de LEDs RGB e buzzer através de um teclado matricial conectado ao Raspberry Pi Pico W. O código é desenvolvido em C e simulado no Wokwi, destacando-se pelo uso de periféricos básicos para controle de hardware (2).

Aplicação de Controle de Acesso com RFID e AWS DynamoDB: Este projeto utiliza o Raspberry Pi Zero W para implementar controle de acesso com tags RFID, integrando-se ao AWS DynamoDB para gerenciamento de dados. Foca na autenticação via RFID e armazenamento em nuvem (3).

Sistema de Controle de Acesso por Reconhecimento Facial: Projeto que combina um servidor local com um Raspberry Pi para implementar controle de acesso baseado em reconhecimento facial. Embora utilize o Raspberry Pi, a autenticação é realizada por biometria facial (4).

Reconhecimento de voz com Raspberry Pi 4: Projeto onde o microfone é utilizado para acionar funções do sistema, mostrando que a integração do áudio para verificação de acesso também é uma abordagem conhecida, mas implementada de forma distinta (5).

18. Problemas e Erros Durante a Execução

Durante a execução do sistema Alpha Segurança, foram observados alguns problemas intermitentes relacionados ao funcionamento do display OLED. Em determinadas ocasiões, os pixels do display são renderizados de forma estranha, resultando em imagens distorcidas ou fragmentadas, enquanto em outros momentos o display não liga, mesmo quando o sistema continua executando o loop principal (while true).

19. Repositório do Projeto e Vídeo de Demonstração

Para facilitar o acesso ao código-fonte e à demonstração do funcionamento do sistema Alfa Segurança, disponibilizamos os seguintes links:

- **Repositório no GitHub:**
  Acesse o repositório do projeto para visualizar o código-fonte e a documentação completa:
  https://github.com/LuisBaiano/FinalProject-EMBARCATECH
- **Vídeo de Demonstração:**
  Confira o vídeo que demonstra as funcionalidades e o funcionamento do sistema Alfa Segurança:

  https://drive.google.com/drive/u/0/folders/1htpyT061CxRiVs5t-WJD19F9LfIxuQYs

20. Conclusão

O firmware do sistema **Alpha Segurança** é estruturado em camadas bem definidas, desde a abstração de hardware até a aplicação de alto nível, garantindo uma integração robusta e modular dos diversos componentes do sistema. Através de funções dedicadas aos drivers do display, LED Matrix, tratamento de botões e debounce, o software assegura a correta aquisição dos dados, o processamento das autenticações e o fornecimento de feedback visual e sonoro.

Demonstração em vídeo:

Referências

1. **Controle de Acesso com RFID e MQTT**. Disponível em: https://github.com/EduPDX/RFID_PICO. Acesso em: 25 fev. 2025.
2. **Controle de Pinos GPIO com Teclado Matricial 4x4**. Disponível em: https://github.com/adimael/controle-gpio-teclado-matricial-rp2040. Acesso em: 25 fev. 2025.
3. **Aplicação de Controle de Acesso com RFID e AWS DynamoDB**. Disponível em: https://github.com/rafaelbcastilhos/RFID-dynaberry. Acesso em: 25 fev. 2025.
4. **Sistema de Controle de Acesso por Reconhecimento Facial**. Disponível em: https://github.com/heyuall/access-control. Acesso em: 25 fev. 2025.
5. **RECONHECIMENTO DE VOZ COM RASPBERRY PI 4 (CONTROLE A GPIO).** Disponível em: https://www.youtube.com/watch?v=zL0NkObD9B4. Acesso em: 25 fev. 2025.
6. **RASPBERRYPI/PICO-EXAMPLES - GitHub**. Disponível em: https://github.com/raspberrypi/pico-examples. Acesso em: 25 fev. 2025.
7. **RASPBERRY PI FOUNDATION.** Raspberry Pi Pico Datasheet. Disponível em: https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf. Acesso em: 25 fev. 2025.
8. **RASPBERRY PI FOUNDATION.** Pico SDK Documentation. Disponível em: https://github.com/raspberrypi/pico-sdk. Acesso em: 25 fev. 2025.
9. **SSD1306 OLED Driver Datasheet.** Disponível em: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf. Acesso em: 25 fev. 2025.
10. **BROWN, Michael.** Techniques for Button Debouncing in Embedded Systems. Disponível em: https://www.embedded.com/debouncing-techniques. Acesso em: 25 fev. 2025.
11. **JOHNSON, Alex.** LED Matrix Control with Raspberry Pi Pico and PIO. Disponível em: https://github.com/username/ledmatrix-tutorial. Acesso em: 25 fev. 2025.
12. **VISUAL STUDIO CODE**. Visual Studio Code, 2025, Docs. Disponível em: < https://code.visualstudio.com/docs>, Acesso em: 25 fev. 2025.

13. **CARVALHO, Ricardo; SILVA, Fernanda.** Segurança em Sistemas Embarcados: Desafios e Soluções. São Paulo: Editora Técnica, 2020.