INSTITUTO HARDWARE BR

CAPACITAÇÃO EM SISTEMAS EMBARCADOS

IGOR KENZO MARTINS SEBATA

PROJETO FINAL:
SISTEMA DE ALARME PARA TAREFAS DIÁRIAS

CAMPINAS

2025

1. Apresentação

O presente projeto tem como objetivo demonstrar o desenvolvimento de um sistema embarcado de alarmes para tarefas cotidianas na placa BitDogLab. O sistema permite que o usuário configure alarmes de curta duração (até 24 horas) ou de longa duração (até 336 horas) através de uma interface gráfica simples, contando com um modo silencioso, onde o disparo do alarme é indicado de forma totalmente visual por meio da matriz de LEDs, ou seja, não utilizando os buzzers (alerta sonoro).

2. Objetivos

2.1 Objetivo Principal

- Desenvolver um sistema que possibilite a configuração e disparo de alarmes, para tarefas diárias, de curta ou longa duração, com alertas sonoros e/ou visuais, por meio de uma interface simples.

2.2 Objetivos Específicos

- Implementar uma interface simples e intuitiva para a configuração do alarme.
- Permitir a configuração de alarmes por meio de botões e joystick.
- Implementar um modo silencioso, em que o disparo do alarme será indicado apenas visualmente.
- Assegurar que o sistema seja compacto, de baixo custo e fácil de utilizar.
- Assegurar que a operação do sistema seja independente da internet.

3. Principais Requisitos

3.1 Requisitos Funcionais

RF01	**Duração do Alarme**
O sistema deverá permitir que o usuário configure alarmes de curta duração (até 24 horas) ou de longa duração (até 336 horas).	

RF02	**Configuração do Alarme**
O sistema deverá permitir que o alarme seja configurado manualmente, utilizando os botões e o joystick.	

RF03	Visualização da Configuração
O sistema deverá exibir todas as etapas de configuração do alarme para o usuário através do display OLED.	

RF04	Modo Silencioso
O sistema deverá permitir que o usuário escolha habilitar o modo de alerta silencioso na interface gráfica.	

RF05	Visualização da Configuração
O sistema deverá emitir um alerta sonoro e/ou visual no disparo do alarme configurado pelo usuário.	

RF06	Independência de Rede
O sistema não deverá estabelecer conexão com a internet durante a sua operação.	

RF07	Desativar Alarme
O sistema deverá permitir que o usuário desative o alarme pressionando o botão A, B ou SW do joystick.	

RF08	Frequência Sonora
O sistema deverá produzir, pelos buzzers, um alerta sonoro na frequência de 520Hz por 30 segundos ou até a ser desativado manualmente.	

RF09	Fonte de Energia
O sistema deverá permitir que a pilha recarregável seja carregada por mini painel solar.	

3.2 Requisitos Não Funcionais

RNF01	Tempo de Resposta
O sistema deverá processar e atualizar o display em menos de 100 ms.	

RNF02	Acionamento do Alarme
O sistema deverá ativar os alertas após, no máximo, 200 ms do disparo do alarme.	

RNF03	Tempo de Configuração
O sistema deverá permitir que o usuário configure um alarme curto em menos de 1 minuto.	

4. Descrição de Funcionamento

O sistema será configurado através dos botões e do joystick, sendo a configuração exibida por meio do display OLED. Inicialmente, o usuário pode escolher o modo do alarme:

- Modo Curto: a configuração do alarme se dá pela forma de uma contagem regressiva, no formato HH:MM:SS, de até 24 horas.
- Modo Longo: o usuário deverá configurar o horário e dia da semana atual e então escolher se a semana de disparo do alarme será a atual ou a próxima, então a configuração do alarme será feita escolhendo o horário (no formato HH:MM) e dia da semana de disparo.

Após escolher o modo e configurar o horário do alarme, o sistema irá permitir que o usuário escolha se o alarme deverá ser silencioso ou não:

- Modo Silencioso: no disparo do alarme, os buzzers não serão utilizados, sendo a notificação de disparo feita por padrões luminosos na matriz de LEDs.
- Modo Normal (não silencioso): no disparo do alarme, tanto os buzzers quanto a matriz de LED serão utilizados para notificar o disparo.

5. Justificativa

Marcar o tempo foi, desde os primórdios da civilização, uma das necessidades mais vitais do ser humano. Sendo fundamental para as questões mais básicas de sobrevivência, a evolução das formas de medir o tempo moldou a evolução humana[2]. Atualmente, embora dispositivos como celulares ou relógios inteligentes ofereçam amplas funções avançadas de alarme e agenda, o sistema embarcado dedicado a alarmes apresenta algumas vantagens específicas:

- A possibilidade de carregamento da bateria por painel solar oferece ao sistema certa autonomia energética, reduzindo a dependência de redes elétricas, sendo ideal para uso em locais com acesso limitado a energia.
- A interface simplificada, focada nas funcionalidades de alarme, proporciona uma interação intuitiva e simples, com uma configuração de parâmetro de cada vez, atrativa a usuários que preferem dispositivos com propósito específico e operação objetiva.

- Relógios inteligentes e celulares são caros e não acessíveis para todas as pessoas[26], a utilização de componentes simples e relativamente baratos torna o sistema uma alternativa viável quando se necessita desta simples função.
- Celulares muitas vezes são fontes de distração, com redes sociais, notificações e aplicativos que podem desviar a atenção do usuário[4]. Em momentos de foco, um dispositivo que não contém diversas distrações pode ser benéfico ao diminuir interrupções e desvios de foco.

6. Originalidade

Em projetos de pequena escala ou feitos por hobby, observa-se que grande parte dos sistemas de alarme baseados em Raspberry Pi Pico é desenvolvida em MicroPython, frequentemente utilizando bibliotecas e funções de alto nível para sua implementação, mesmo quando aplicam o conceito do alarme de forma diferente deste projeto[5][6][15][21][22][24][25] . O próprio repositório do GitHub da BitDogLab[7] contém muitos exemplos de projetos programados em MicroPython, utilizando também bibliotecas de alto nível sem explorar os recursos de comunicação de baixo nível.

Com esta visão, este projeto se contrasta ao utilizar a linguagem C, que proporciona um maior controle sobre o hardware e possibilita a implementação de baixo nível, ao explorar os PIOs (Programmed Input/Output) para o controle eficiente da matriz de LED 5x5, e ao implementar todas estas funcionalidades na placa BitDogLab.

7. Diagrama de Blocos – Hardware

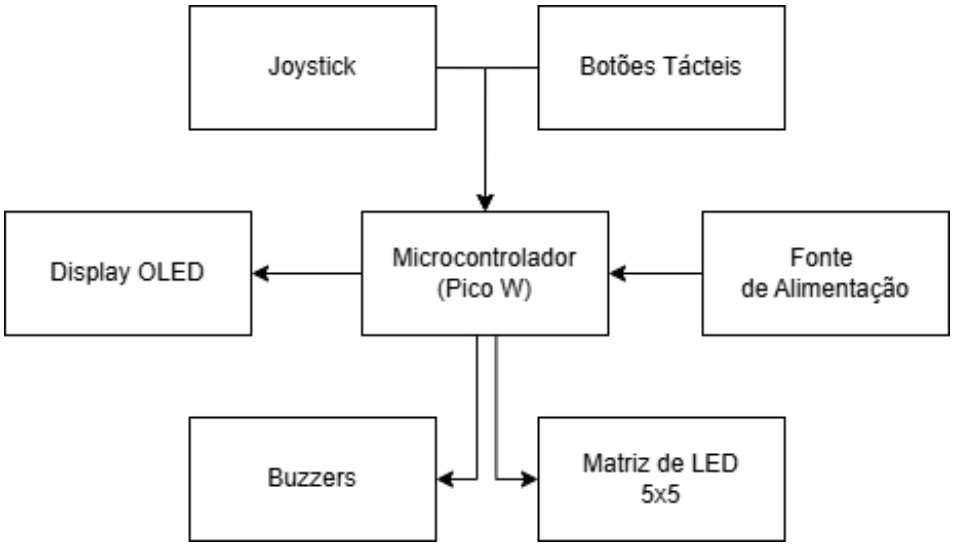

Figura 1 - Diagrama de Blocos do Hardware. Fonte: Autor.

- Microcontrolador – Responsável por gerenciar os periféricos e a lógica de alarme; envia instruções para o display, matriz de LED e buzzers.
- Matriz de LED 5x5 (WS2812B) – Responsável por indicar visualmente quando houve o disparo de um alarme, especialmente no modo silencioso.
- Display OLED – Responsável por exibir a interface gráfica para a configuração dos alarmes.
- Joystick e Botões Tácteis – Responsável por permitir a navegação e interação do usuário com o sistema.
- Buzzers – Responsáveis por indicar sonoramente quando houve o disparo de um alarme.
- Fonte de Alimentação – Alimenta todo o sistema, pode ser uma bateria de lítio ou cabo USB-C.

8. Configuração dos Blocos

- Microcontrolador:
  - Alimentação: 3.3V
  - Memória: 264kb SRAM e 2MB Flash
  - Clock: 133 MHz
  - Periféricos Utilizados: I^2C, PIO, ADC, PWM, Timer de Hardware;
- Matriz de LED 5x5 (WS2812B):
  - Comunicação: PIO
  - Alimentação: 5V
  - Frequência PIO: 800kHz
- Display OLED (SSD1306):
  - Comunicação: I^2C
  - Alimentação: 3.3V
  - Frequência I^2C: 100kHz
  - Porta I^2C: i2c1
  - Endereço: 0x3c
- Joystick:
  - Comunicação: ADC
  - Modo ADC: One-shot
  - Resistores para SW: Pull-Up
- Botões Tácteis:
  - Resistores: Pull-Up
- Buzzers Passivos:
  - Comunicação: PWM
  - Frequência PWM: 520Hz
- Fonte de Alimentação:
  - Regulador de Tensão: 3.3V

9. Atendimento aos Requisitos

Os componentes e suas configurações atendem aos requisitos do sistema uma vez que:

- O microcontrolador (Raspberry Pi Pico) é mais do que o suficiente para gerenciar todas as funcionalidades de maneira eficiente e rápida;
- Para a interface com o usuário, o display, joystick e botões tácteis permitem uma configuração manual e intuitiva dos alarmes, possibilitando a visualização e seleção das opções e parâmetros do alarme;
- O sistema de alerta é implementado por meio de dois mecanismos, os buzzers passivos para alertas sonoros, ativados na frequência de 520Hz, e a matriz de LEDs para alertas visuais. A combinação permite tanto o modo normal (ambos) quanto o silencioso (visual);
- O sistema aceita múltiplas fontes de alimentação, podendo ser alimentado por uma bateria de lítio recarregável (que pode ser carregada por mini painel solar de 12V) ou USB-C;
- O sistema opera de forma independente de conexão à internet, sendo todas as funcionalidades processadas localmente;
- As altas frequências de operação, tanto do sistema quanto de seus periféricos, asseguram que as atualizações visuais e limites de tempos sejam respeitados.

10. Lista de Materiais (Bill of Materials)

Nome	Descrição	Quantidade
Raspberry Pi Pico W	Microprocessador da BitDogLab, versão com WiFi	1
MÓDULO WS2812B	Módulo de Matriz de LEDS RGB 5x5 5050	1
SSD1306	Display OLED 0.96” I2C 128x64	1
JOYSTICK	Versão do Joystick sem o módulo	1
PUSH BUTTON	Botão Táctil Push Button	2
BUZZERS	Buzzers Passivos	2

11. Descrição da Pinagem

GPIO	Função
5	Entrada para o Botão Táctil A (esquerdo)
6	Entrada para o Botão Táctil B (direito)
21	Saída PWM para o Buzzer A (esquerdo)
10	Saída PWM para o Buzzer B (direito)

14	Pino SDA (sinal de linha de dados) do protocolo I²C para comunicação com o display
15	Pino SCL (sinal de linha de clock) do protocolo I²C para comunicação com o display
26	Entrada ADC para o Joystick Y (vertical)
27	Entrada ADC para o Joystick X (horizontal)
22	Entrada para o Botão SW do Joystick
7	Saída para o DIN (Data In) do primeiro LED da Matriz 5x5

12. Desenho Completo do Circuito

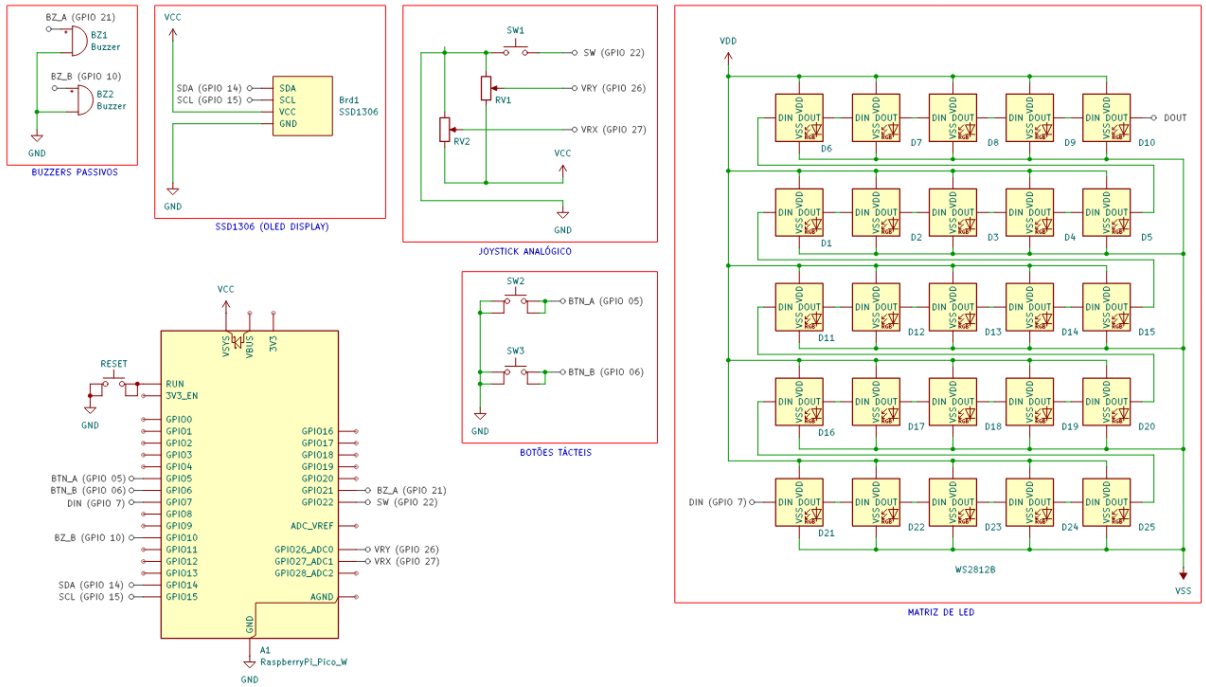

Figura 2 - Desenho Esquemático do Sistema. Fonte: Autor.

DESENHO_CIRCUITO.pdf

13. Diagrama de Blocos – Software

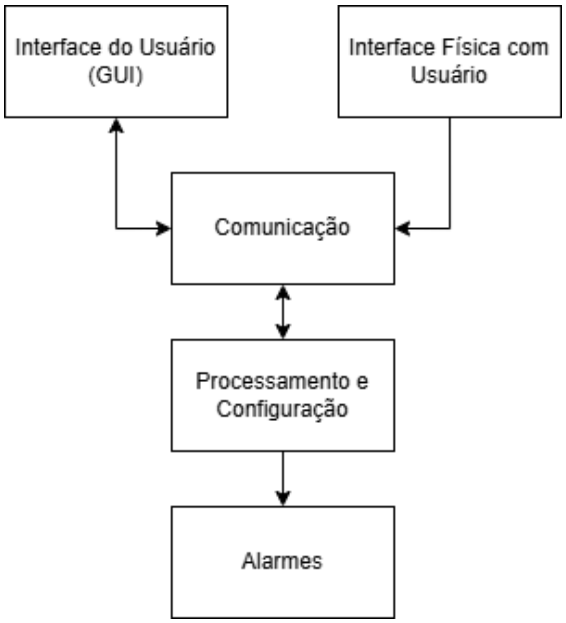

Figura 3 - Diagrama de Blocos das Camadas do Sistema. Fonte: Autor.

- Interface do Usuário – Responsável por exibir as opções e parâmetros de configuração dos alarmes para o usuário.
- Interface Física com Usuário – Responsável por enviar os sinais de entrada feitos pelo usuário (mover joystick, apertar botão, etc.)
- Comunicação – Responsável por enviar e receber dados, fornecendo-os para as outras camadas, especialmente a de processamento.
- Processamento e Configuração – Responsável por gerenciar as outras camadas, interpretar e armazenar dados, e configurar alarmes.
- Alarmes – Responsável por gerenciar o alarme atual, acionando os alertas visuais e/ou sonoros quando disparado.

14. Definição das Variáveis

- **current_timer_struct:** variável que contém os dados necessários para a configuração do alarme, como a data atual, data selecionada, modo silencioso, se o alarme é na semana atual e se o alarme é curto.
- **rule_set:** um array de regras de seleção de opções para cada estado do display, se comporta como um mapa, relacionando os estados com uma tupla: quantidade de opções na coluna 1 e quantidade de opções na coluna 2.
- **current_display_state:** variável global que contém o estado atual do display.
- **display:** variável que contém os dados necessários para operar o display, como comprimento e altura do display, endereço, instância I^2C etc.

- **on_loop:** variável que determina se as interrupções geradas pelos botões são válidas naquele momento.
- **next_triggered** e **prev_triggered**: variáveis alteradas pelas interrupções dos botões, determinam se o processo de configuração deve progredir ou regredir.
- **remaining_iterations**: variável que armazena a quantidade de vezes que o hardware timer precisa ativar para completar a quantidade necessária de tempo de alarme (devido à limitação do timer de 32 bits).

15. Fluxograma

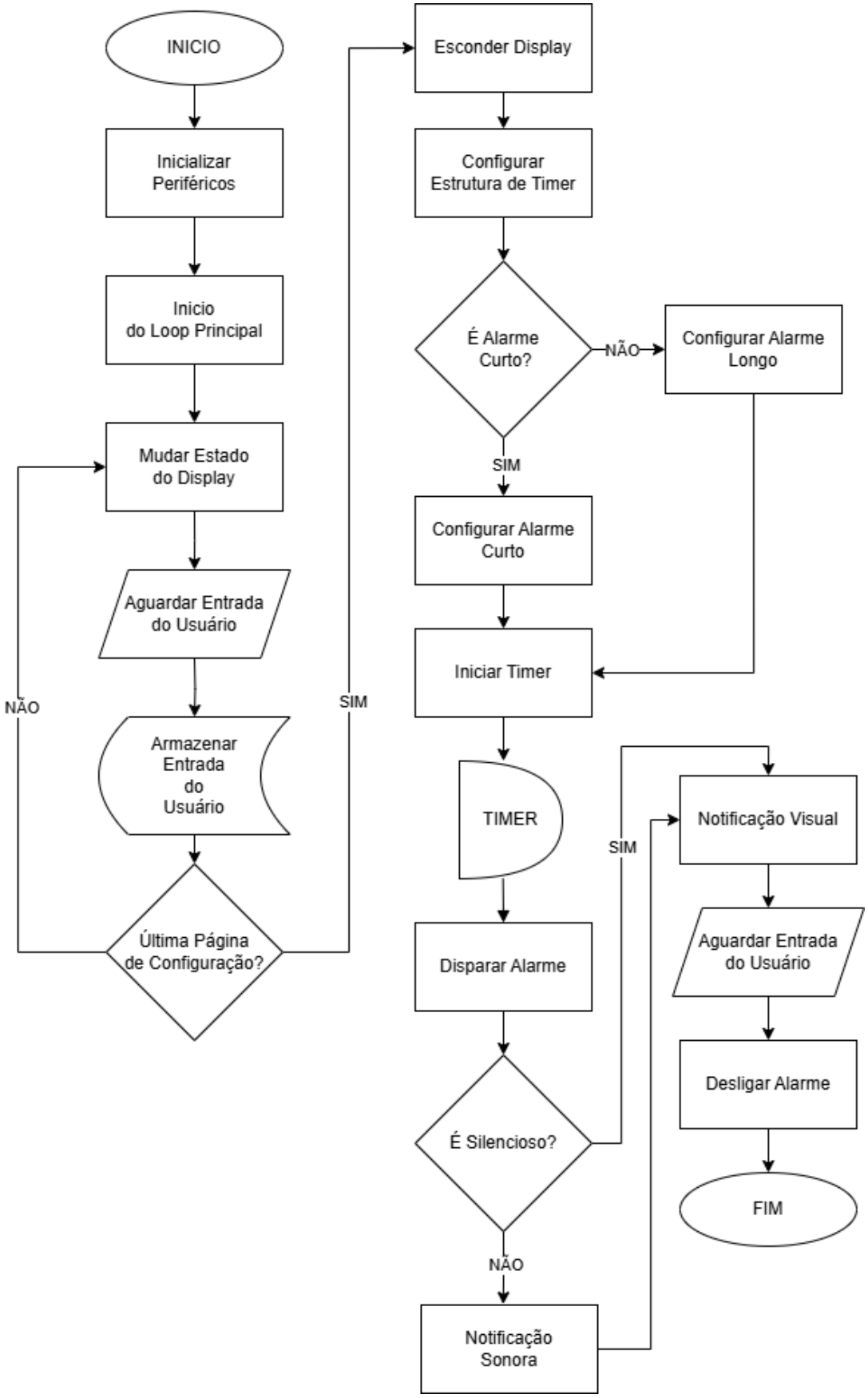

Figura 4 - Fluxograma do Sistema. Fonte: Autor.

16. Inicialização

O programa inicia pela função **main**, responsável, neste caso, por inicializar: as funções de entrada e saída padrão linkadas com o arquivo binário; o timer (insere o ID do timer a ser utilizado no registrador e inicializa o handler de interrupção); o display (inicia o **I²C** e configura os GPIOs, e inicia a estrutura do display para a biblioteca do **ssd1306**); os buzzers (inicia o **PWM** e configura a frequência); os leds (obtém uma **state machine** livre do **PIO**, carrega e configura o programa); o joystick (inicia o **ADC** e os GPIOs); e os botões (configura os botões e ativa a interrupção com a flag **GPIO_IRQ_EDGE_FALL**). Após isso, o sistema inicia o loop de funcionamento, checando os sinais dos botões para avançar ou regredir no procedimento.

17. Configurações de Registro

GPIO	I/O	Resistor
5	Entrada	PULL-UP
6	Entrada	PULL-UP
21	PWM - Saída	N/A
10	PWM - Saída	N/A
14	SDA	PULL-UP
15	SCL	PULL-UP
26	ADC - Entrada	N/A
27	NÃO UTILIZADA	N/A
22	Entrada	PULL_UP
7	SIDE-SET (SAÍDA)	N/A

PWM – Frequência: 520Hz; Wrap: 4096; GPIOs: 21 e 10

Hardware Timer – ID: 0; IRQ: TIMER_IRQ_0

PIO – Frequência Base: 800kHz; Frequência Real: 8MHz; GPIO Side-Set: 7; Tamanho da OSR: 24 bits; TX FIFO Join

I²C – Frequência (Baud Rate): 100kHz; SDA: GPIO 14; SCL: GPIO 15; I²C PORT: i2c1; Endereço: 0x3C

18. Endereço de Memória

Display OLED: Endereço I²C – 0x3C

19. Protocolos

I²C – interface síncrona com 2 fios de comunicação (SCL, SDA) que permite a conexão de múltiplos dispositivos em um só barramento.

PIO – método de transmissão da dados via entrada/saída, protocolos de comunicação são configuráveis.

20. Formato de Pacote de Dados

A Figura 4 retrata a composição do pacote de dados necessário para controlar os leds da matriz WS2812B.

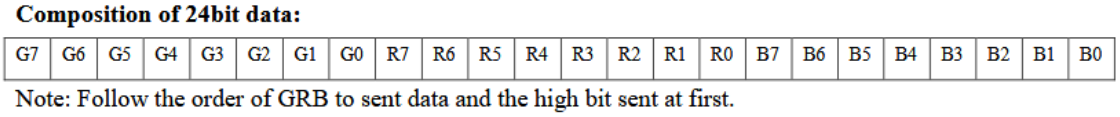

Figura 5 - Pacote de Dados de 24 Bits. Fonte: Adafruit[1].

21. Metodologia

Inicialmente, foram levantados todos os requisitos do documento providenciado, visando entender e definir claramente as prioridades para a execução do projeto. Após isso, foi conduzida uma busca sobre diferentes projetos envolvendo sistemas embarcados com objetivo de encontrar inspiração e entende mais sobre as aplicações deles em diferentes áreas, um processo que não rendeu tantos resultados quanto o esperado, visto que a grande maioria dos projetos necessitava de componentes externos ou não eram atrativos o suficiente.

O tema foi escolhido, finalmente, por análise da vida pessoal em busca de dispositivos úteis para o cotidiano, e então foi delimitado o escopo e determinados os objetivos do projeto, também definindo os requisitos funcionais e não funcionais que atendiam às necessidades e ao escopo. Então, foram identificados os blocos do sistema, hardware e software, e definidos os componentes e suas conexões.

Foram conduzidas diversas pesquisas, principalmente sobre os PIOs[16][23] e suas configurações e utilidades, também sobre a linguagem C e o Raspberry Pi Pico W e os componentes da placa BitDogLab.

Finalmente, o processo de desenvolvimento foi começado utilizando o VSCode com o Pico SDK e extensão de apoio ao desenvolvimento do Raspberry Pi. A estrutura do código foi dividida em módulos, cada um com seus respectivos cabeçalhos (*header files*). Cada módulo foi testado individualmente antes da sua integração com os outros.

22. Testes de Validação

- Configuração de Alarme Curto:
    - Método: acompanhar a configuração do alarme e cronometrar o tempo que foi necessário.
    - Objetivo: validar os requisitos: RF01, RF02, RF03 e RNF03
- Configuração de Alarmes Longos:
    - Método: configurar e monitorar o alarme longo.
    - Objetivo: validar o requisito RF01
- Modo Silencioso:
    - Método: configurar um alarme curto e um alarme longo com o modo silencioso.
    - Objetivo: validar os requisitos RF04 e RF05
- Acionamento do Alarme:
    - Método: medir o tempo entre o momento de acionamento esperado e o momento real.
    - Objetivo: validar o requisito RNF02
- Teste de Desativação:
    - Método: esperar o disparo do alarme e testar todas as entradas de desativação (Botão A, B e S)
    - Objetivo: validar o requisito RF07

23. Conclusão e Discussão dos Resultados

O desenvolvimento foi bem-sucedido, atendendo aos requisitos estabelecidos e se destacando por sua interface simples, confiabilidade na ativação dos alarmes e precisão de tempo. O projeto é funcional e viável para uso diário, funcionando de forma independente da internet e diminuindo as possíveis distrações.

Inicialmente, o projeto foi pensado com a intenção de incluir o **reconhecimento de voz** totalmente independente da internet, utilizando tecnologias/bibliotecas como o TensorFlow Lite[20], Pocketsphinx[11], Edge Impulse[3] e Picovoice[9]. Isto com o objetivo de permitir que o sistema processasse os comandos localmente. Contudo, a limitação do hardware do Raspberry Pi Pico impossibilitou a implementação desse recurso, visto que o processamento de voz precisaria de mais memória do que a disponível.

Mesmo com estas limitações, a ideia não tinha sido totalmente abandonada, tendo sido planejada uma alternativa por meio do reconhecimento de voz via API, utilizando o Amazon Transcribe, onde os comandos seriam capturados pelo microfone da BitDogLab, armazenados utilizando DMA e enviados a um serviço serverless, como o Amazon Lambda, processados e enviados de volta em forma de JSON. Infelizmente, essa abordagem também foi

descartada por limitação no tempo, alta complexidade e prováveis custos de operação (dos serviços Amazon), além de não se alinhar completamente com a ideia inicial do projeto de independência da internet.

Apesar de tudo, o projeto atendeu a todos os requisitos principais, proporcionando um sistema de alarme eficiente e simples. Algumas sugestões de melhoria seriam, obviamente, integrar o sistema de reconhecimento de voz e implementar o painel solar para carregar a pilha embutida.

24. Video de Demonstração

https://www.youtube.com/watch?v=2yAQodvzWOg

25. Repositório do Projeto

Igor-KMS/projeto_final: Projeto Final

26. Referências

[1] ADAFRUIT. WS2812B Datasheet. Disponível em: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf.

[2] COMCIÊNCIA. Para conseguir contar o tempo, foi uma questão de tempo. Disponível em: https://www.comciencia.br/para-conseguir-contar-o-tempo-foi-uma-questao-de-tempo/.

[3] EDGE IMPULSE. Machine Learning at the Edge. Disponível em: https://edgeimpulse.com/.

[4] EL PAÍS. Smartphone, uma arma de distração em massa. Disponível em: https://brasil.elpais.com/brasil/2017/06/23/tecnologia/1498217993_075316.html.

[5] EMBARCADOS. Raspberry Pi Pico – Timer One Shot com MicroPython. Disponível em: https://embarcados.com.br/raspberry-pi-pico-timer-one-shot-com-micropython/.

[6] EMBARCADOS. Raspberry Pi Pico – Timer Periódico com MicroPython. Disponível em: https://embarcados.com.br/raspberry-pi-pico-timer-periodico-com-micropython/.

[7] GITHUB. BitDogLab. Disponível em: https://github.com/BitDogLab/BitDogLab.

[8] GITHUB. Mozilla DeepSpeech. Disponível em: https://github.com/mozilla/DeepSpeech.

[9] GITHUB. Picovoice. Disponível em: https://github.com/Picovoice/picovoice.

[10] GITHUB. Pico SSD1306 Library. Disponível em: https://github.com/daschr/pico-ssd1306.

[11] GITHUB. Pocketsphinx. Disponível em: https://github.com/cmusphinx/pocketsphinx.

[12] GOOGLE DOCS. Banco de Informações de Hardware (BIH) da BitDogLab. Disponível em: https://docs.google.com/document/d/13-68OqiU7ISE8U2KPRUXT2ISeBl3WPhXjGDFH52eWlU/edit?tab=t.0.

[13] IEEE XPLORE. Continuous speech recognizer for low-end embedded devices. Disponível em: https://ieeexplore.ieee.org/document/7181940.

[14] I-PROGRAMMER. The Pico In C - Basic PWM. Disponível em: https://www.i-programmer.info/programming/hardware/14849-the-pico-in-c-basic-pwm.html?start=2.

[15] OSOYOO. Raspberry Pi Pico Learning Kit – Intruder Alarm System. Disponível em: https://osoyoo.com/2021/07/20/raspberry-pi-pico-learning-kit-intruder-alarm-system/.

[16] RASPBERRY PI. Raspberry Pi Documentation. Disponível em: https://www.raspberrypi.com/documentation/pico-sdk/.

[17] SOLID STATE SECURITY. Most Effective Hearing Frequency: What You Need to Know. Disponível em: https://www.solidstatesecurity.co.uk/post/most-effective-hearing-frequency-what-you-need-to-know.

[18] TECNOLOGIA. Amazon Echo Dot ou Google Nest Mini: qual é melhor? Disponível em: https://tecnoblog.net/guias/amazon-echo-dot-ou-google-nest-mini-qual-e-melhor/.

[19] TECHTUDO. Echo ou Google Nest? Conheça as diferenças das smart speakers. Disponível em: https://www.techtudo.com.br/listas/2023/07/echo-ou-google-nest-conheca-as-diferencas-das-smart-speakers-edinfoeletro.ghtml.

[20] TENSORFLOW. TensorFlow Lite Guide. Disponível em: https://www.tensorflow.org/lite/guide?hl=pt-br.

[21] YOUTUBE. How to Make a Motion-Activated Burglar Alarm with Raspberry Pi Pico and MicroPython. Disponível em: https://www.youtube.com/watch?v=Qtxmb1nTbs8.

[22] YOUTUBE. Curso de Raspberry Pi Pico - Alarme com sensor de movimento. Disponível em: https://www.youtube.com/watch?v=OGd66MkO5Q0.

[23] YOUTUBE. In-depth: Raspberry Pi Pico's PIO – programmable I/O. Disponível em: https://www.youtube.com/watch?v=yYnQYF_Xa8g.

[24] YOUTUBE. 15 - TIMER | Periódico - Curso Raspberry Pi Pico com #Micropython. Disponível em: https://www.youtube.com/watch?v=FYmz5SDm6N0.

[25] YOUTUBE. How to use Timers on the RPi PICO. Disponível em: https://www.youtube.com/watch?v=HKhY1qV8JbY.

[26] O TEMPO. Tecnologia é mais cara no Brasil. Disponível em: https://www.otempo.com.br/economia/tecnologia-e-mais-cara-no-brasil-1.1581660.