INSTITUTO HARDWARE BR

Eduardo Pereira Tejada

Projeto Final

BitDorme: Monitoramento de sono e despertador inteligente

EMBARCATECH

CAMPINAS, SP
2025

1. Escopo do projeto

1.1. Apresentação do projeto – breve descrição sobre o que trata o projeto.

O projeto consiste em um sistema de monitoramento do sono utilizando o microfone da BitDogLab, aliado a um alarme interativo que inclui um puzzle para facilitar o despertar de forma mais eficiente.

1.2. Título do projeto – pequeno título para o projeto.

“BitDorme: Monitoramento de sono e despertador inteligente”

1.3. Objetivos do projeto – descreva os objetivos do projeto.

Busca-se fazer um sistema capaz de coletar dados de áudio durante a noite para que assim seja possível identificar padrões e ajudar a melhorar a qualidade do sono dos usuários

Também se deseja criar um jeito de fazer com que o usuário resolva um puzzle para fazer com que o alarme pare de tocar. Assim estimulando a atividade cognitiva fazendo com que não seja tão simples desligar o alarme e voltar a dormir.

Com a combinação de ambas tarefas o projeto contribui para uma rotina matinal mais produtiva e uma qualidade de sono melhor.

1.4. Principais requisitos – podem ser estipulados pelo usuário ou cliente, ou até por você (fazendo o papel de um deles).

1.4.1. Requisitos funcionais:

1.4.1.1. Monitoramento de áudio: O sistema deve captar sons do ambiente, como respiração e ronco, para análise da qualidade do sono.

1.4.1.2. Armazenamento de dados: Deve registrar informações relevantes durante a noite para posterior análise.

1.4.1.3. Ativação do alarme: O despertador deve ser acionado em um horário pré-definido.

1.4.1.4. Desbloqueio do alarme por puzzle: Para desligar o alarme, o usuário deve resolver um desafio (exemplo: sequência lógica, cálculo matemático, jogo simples).

1.4.1.5. Interface de usuário: Deve ter uma forma de configuração e visualização de dados.

1.4.2. Requisitos não funcionais:

1.4.2.1. Custo acessível: O projeto deve buscar componentes de baixo custo para viabilizar sua produção.

1.4.2.2. Baixa latência no processamento: A análise de áudio deve ser feita em tempo real ou com um pequeno atraso para garantir a precisão.

1.4.2.3. Durabilidade: Os componentes devem ser resistentes o suficiente para suportar o uso diário sem falhas.

1.5. Descrição do funcionamento – descreva as funcionalidades do projeto.

Os sons captados são processados em tempo real e armazenados para futura análise, possibilitando que o usuário visualize informações relevantes sobre seu sono.

Há uma interface de configuração, permitindo ao usuário configurar o horário do despertador.

Para garantir que o usuário realmente acorde, o alarme só pode ser desligado após a resolução de um jogo da memória, estimulando a atividade mental e reduzindo as chances de voltar a dormir.

É mostrado no display OLED dos resultados coletados durante o sono, bem como é disponibilizado um código em Python capaz de gerar um na tela do computador para melhor visualização.

1.6. Justificativa – mostre que a execução do projeto se justifica.

O sono é fundamental para a vida de todos, e garantir que ele esteja sendo realizado com qualidade é uma tarefa importante. Durante o sono o corpo realiza diversos processos essenciais e distúrbios no sono podem causar diversos problemas não só em relação ao comportamento mas também podem levar ao desenvolvimento de doenças crônicas.

Por outro lado, despertar de um sono profundo nem sempre é uma tarefa fácil, muitas pessoas, mesmo que sem querer, desligam o alarme e voltam a dormir sem nem perceber. Para combater isso, diversos aplicativos usam desafios para estimular o pensamento nos primeiros segundos do dia, assim ajudando a trazer o cérebro para a consciência mais rápido. Esses podem ser problemas matemáticos ou desafios de lógica, mas todos buscam minimizar a sonolência matinal e melhorar a disposição ao longo do dia.

Por conta disso o uso de um dispositivo que não só ajuda a monitorar o sono, coletando dados que podem indicar diversos problemas mas também que auxilia o usuário a acordar tende a ser muito benéfico para a saúde, rotina e bem-estar.

1.7. Originalidade – mostre através de uma pesquisa que existem projetos correlatos, mas não iguais.

Sleep Quality Monitor: Um projeto semelhante que utiliza de mais sensores, permitindo conclusões mais precisas para a análise, mas não permite que os dados sejam salvos, visualizados através do computador e fazendo uma análise mais complexa. Além de utilizar ATMEGA328P, o que limita o poder de processamento do projeto. Ainda, não há a parte do despertador;

A wearable wristband designed for sleep monitoring and analysis: Esse projeto trás uma proposta interessante de uma pulseira que para fazer a detecção dos parâmetros, também dispõe de mais sensores e é feito com ATMEGA328P como no projeto anterior. O projeto também se torna um pouco diferente por se tratar não só de um detector para o sono, mas sim para ser usado 24 horas. Ainda, não há a parte do despertador;

I made an alarm clock that you can't turn off from bed: Como não foi encontrado nenhum projeto de sistema embarcado de um despertador que para de tocar através de um puzzle, esse é algo semelhante, mas ao invés de um jogo da memória ele obriga o usuário a levantar da cama para desligar, porém não contém a parte de monitoramento de sono.

2. Hardware (2,5 pontos no total)

2.1. Diagrama em blocos – diagrama mostrando os blocos e sua interligação.

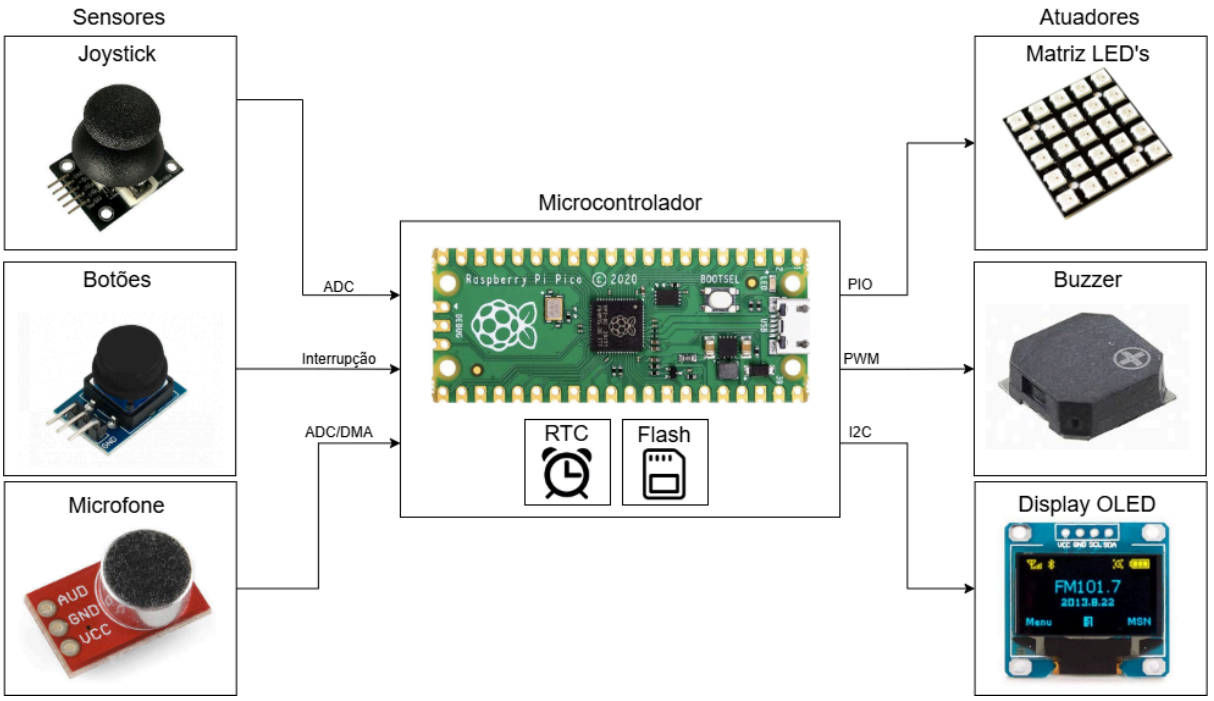


Figura 1: Diagrama em Blocos

2.2. Função de cada bloco – descreva a função que cada bloco terá no projeto.

2.2.1. Microcontrolador: Responsável pelo gerenciamento central do sistema, o microcontrolador executa o código programado, coordena a comunicação entre os componentes, processa os dados captados pelo microfone e sensores, além de armazenar e transmitir informações relevantes para o funcionamento do dispositivo;

2.2.1.1. RTC: Recurso interno que conta o tempo do despertador;

2.2.1.2. Flash: Recurso interno que armazena os dados para análise futura;

2.2.2. Joystick: Utilizado para navegação no menu e como recurso principal no jogo da memória, através do ADC;

2.2.3. Botões: Utilizado principalmente para navegação e para parar o alarme, lidos utilizando gpio e interrupção;

2.2.4. Microfone: Principal sensor responsável por captar nível sonoro que será guardado e servirá para análises futuras, lido utilizando DMA e ADC;

2.2.5. Matriz de LED’s: O jogo da memória é jogado diretamente na matriz de LED’s, ela também gera um feedback luminoso do despertador, acionados através do PIO;

2.2.6. Buzzer: Toca o som do alarme do despertador, essencial para a aplicação, acionado através de PWM;

2.2.7. Display OLED: Responsável por dar instruções, exibir o menu de navegação e exibir gráficos, acionado através de I2C.

2.3. Configuração de cada bloco – descreva a configuração usada em cada bloco.

2.3.1. Microcontrolador: Raspberry Pi Pico W (RP2040), sua frequência de operação é de 133 MHz, foram utilizadas as interfaces I2C, UART, recursos como PWM, PIO, DMA, timers, entre outros;

2.3.1.1. RTC: Foi configurado com a hora atual, com leituras periódicas e com uma função de callback para despertar em um horário determinado;

2.3.1.2. Flash: Usada para armazenar e recuperar dados do microfone para monitoramento de sono, antes de armazenar ela é apagada com um offset para evitar sobrescrever o firmware;

2.3.2. Joystick: Configurado lendo os eixos x e y através do ADC em dois inputs diferentes operando com 12 bits de resolução cada;

2.3.3. Botões: Mecânicos, configurados com resistores de pull up e lidos através de interrupções nas bordas de subida e descida;

2.3.4. Microfone: Lido através do ADC, operando com 12 bits de resolução e referência de 3.3V, o DMA é usado para capturar 200 amostras;

2.3.5. Matriz de LED’s: 25 LED's organizados em uma matriz 5x5. O protocolo WS2812B usa um sinal serial de 800 kHz. O código utiliza uma máquina de estado PIO para enviar os sinais necessários para os LEDs.

2.3.6. Buzzer: buzzer ativo roda a 100 Hz acionado através de PWM;

2.3.7. Display OLED: Tipo OLED, comunicação I2C, 128x64.

2.3.8. Alimentação: Conversor 5V -> 3v3;

2.4. Especificações – descreva como as especificações técnicas atendem os requisitos do cliente ou usuário.

2.4.1. Requisitos funcionais:

2.4.1.1. Monitoramento de áudio: Atendido pelo Microfone, capaz de capturar sons do ambiente, incluindo respiração e ronco;

2.4.1.2. Armazenamento de dados: Atendido pela Memória Flash, os dados do áudio são armazenados em uma alocação dinâmica durante o tempo de execução do código e também na Flash interna do microcontrolador, permitindo posterior análise;

2.4.1.3. Ativação do alarme: Atendido pelo RTC, Buzzer e Matriz de LEDs, o RTC ativa o alarme no momento programado, o buzzer gera o som do alarme e a Matriz de LEDs fornece feedback visual sincronizados com o alarme. É importante notar, contudo, que o RTC inteiro da RP2040 tem uma determinada imprecisão;

2.4.1.4. Desbloqueio do alarme por puzzle: atendido pelo Joystick, botões e Matriz de LEDs, o Joystick e os é usado para interagir com o desafio do alarme e a matriz de LEDs exibe o jogo da memória;

2.4.1.5. Interface de usuário: atendido pelo display OLED, Joystick e Botões, o display OLED exibe menus e instruções para configuração do alarme, o joystick e os botões permitem a navegação no menu.

2.4.2. Requisitos não funcionais:

2.4.2.1. Custo acessível: Atendido pela escolha dos componentes, RP2040, display OLED, Matriz de LEDs e o Buzzer são componentes de baixo custo;

2.4.2.2. Baixa latência no processamento: atendido pelo RP2040, DMA, PIO e interrupções, o microcontrolador (133 MHz) é mais do que suficiente para o processamento do projeto, o DMA reduz a carga da CPU ao coletar dados do microfone, a matriz de LEDs usa PIO garantindo atualização rápida dos LEDs sem travamentos, ainda, as interrupções e a alocação dinâmica facilitam o processamento e ajudam a economizar recursos;

2.4.2.3. Durabilidade: Atendido pela seleção de componentes confiáveis.

2.5. Lista de materiais – incluindo descrição e quantidade.

Item	Quantidade
RP2040	1
Botão	3
Resistor	7
Transistor	1

Buzzer	1
Capacitor	31
Microfone	1
Regulador 5v -> 3v3	1
LED WS2818B	25
Joystick	1
Display OLED	1

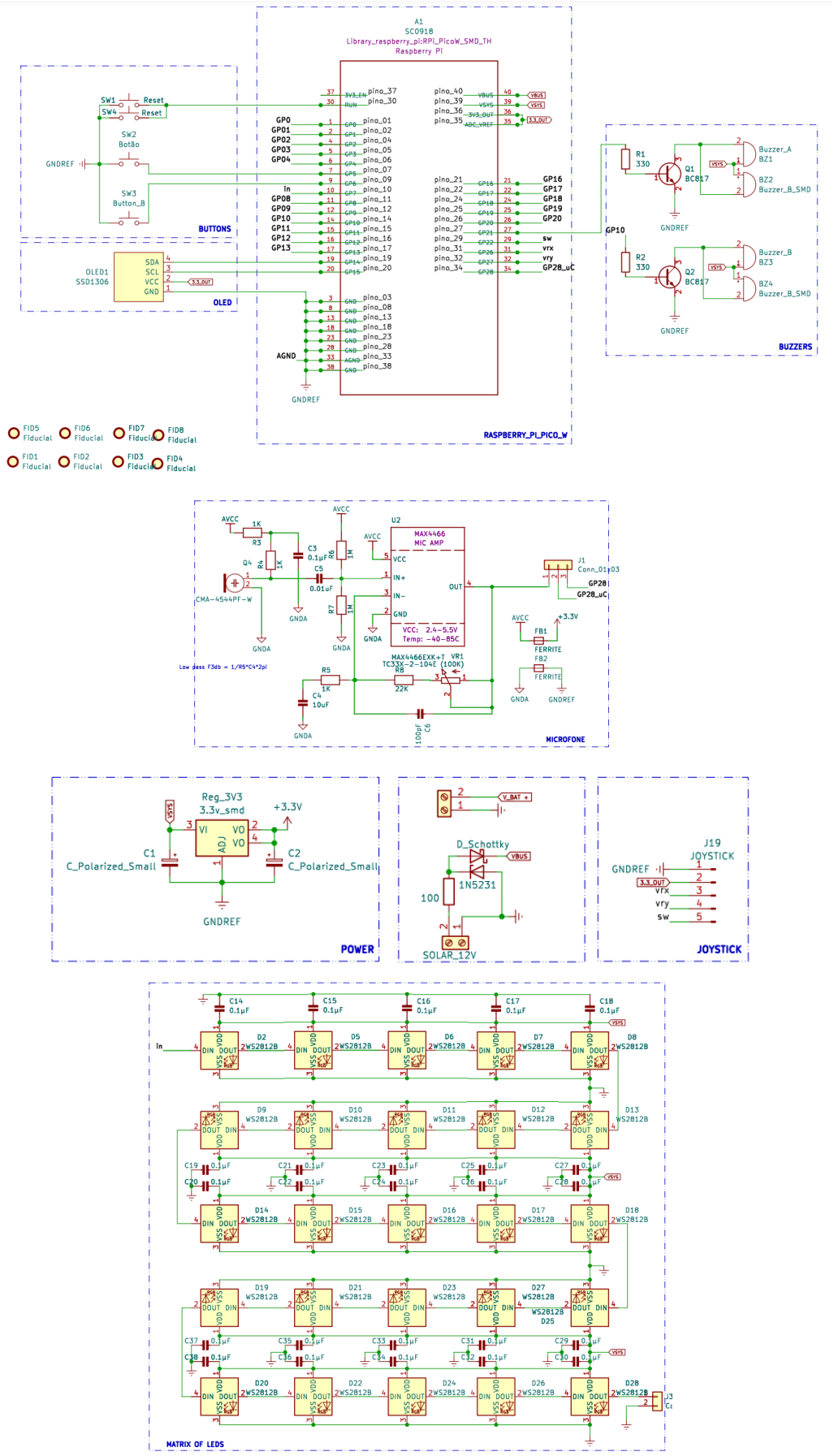
Tabela 1: Lista de materiais

2.6. Descrição da pinagem usada – quais pinos do hardware foram usados e sua função. Você pode usar uma tabela para indicar como os pinos do microcontrolador se conectam com os periféricos. OBS: Apesar de você estar usando a BitDogLab e estas conexões já estarem estabelecidas, monte um mapa do seu projeto só com aquilo que está usando.

Componente	**Pino**
Botão A	GP5
Botão B	GP6
OLED SDA	GP14
OLED SCL	GP15
Buzzer	GP21
Joystick VRY	GP26
Joystick VRX	GP27
Microfone	GP28
Canal do microfone	2
Matriz de LED’s	GP7

Tabela 2: Mapa de pinos

2.7. Circuito completo do hardware – faça o desenho do circuito completo do hardware.



Figuras 2 até 6: [Diagrama da BitDogLab v5.3]

3. Software (2,5 pontos no total, sendo 0,25 cada item)

3.1. Blocos funcionais – mostre um diagrama das camadas do software e suas funções.

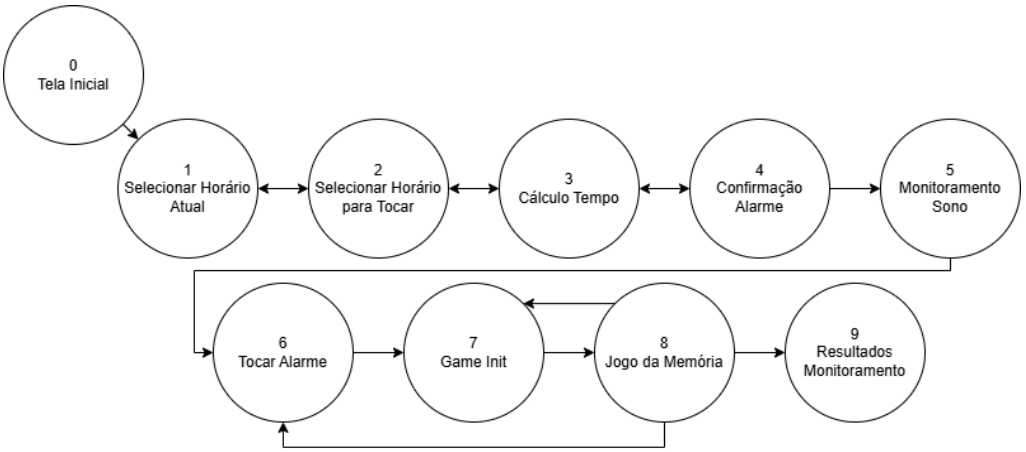


Figura 7: Máquina de Estados Principal

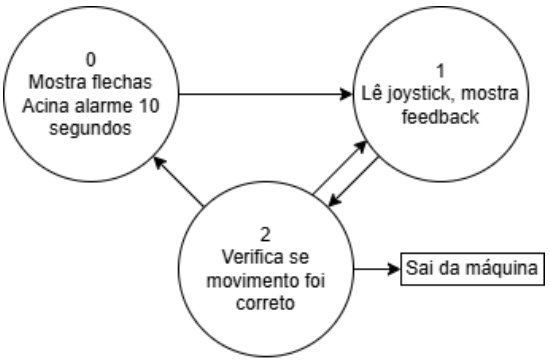


Figura 8: Máquina de Estados do Jogo da Memória

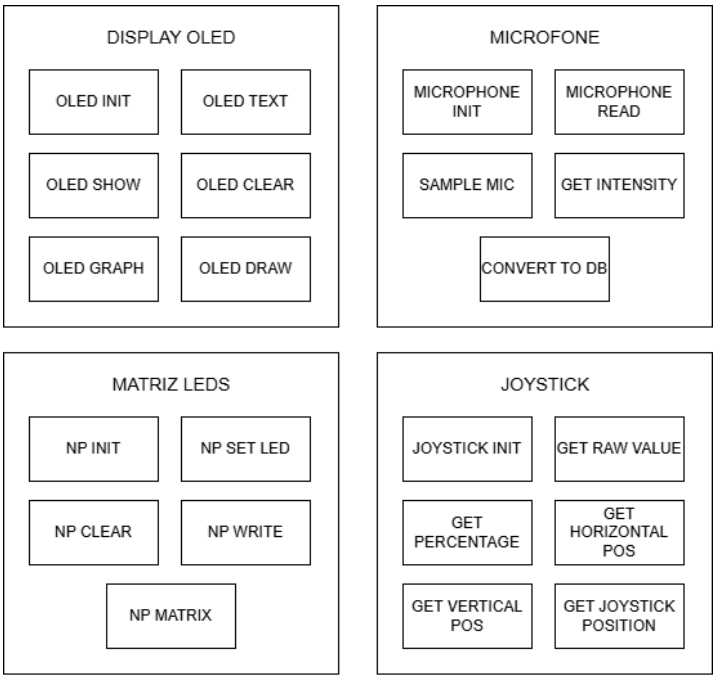


Figura 9: Diagrama dos arquivos externos ao código principal e suas funções

3.2. Descrição das funcionalidades – breve descrição das funções dos blocos de software.

Começando pela máquina de estados principal

Estado 0: Exibe a tela inicial no display OLED mostrando a "logo" do projeto, troca de estado com uma interação no botão A (usa a função oled_draw);

Estado 1: Utiliza uma função para fazer a seleção do horário, que utiliza o display OLED e o Joystick, passando para o próximo estado com o botão B;

Estado 2: Utiliza a mesma função mencionada no estado 1 mas para a selecionar o horário que o alarme irá tocar, voltando pro estado anterior com o botão A e passando para o próximo com o botão B;

Estado 3: Calcula o intervalo de tempo entre os horários atual e do alarme, aloca dinamicamente espaço na memória para salvar os dados do microfone a cada segundo e indica no display OLED o tempo de sono;

Estado 4: Caso o usuário pressione o botão A volta ao estado 2, caso pressione o botão B salva os dados no RTC, configura o alarme para tocar, aciona o timer repetitivo a cada 1 segundo para coletar amostras do microfone e passa para o próximo estado;

Estado 5: Estado de monitoramento de sono o código funciona somente por interrupções de callback do timer repetitivo, coletando amostras, salvando na alocação dinâmica e exibindo informações no display, finalmente, quando o callback do RTC é ativado significa que o alarme precisa tocar e então cancela-se o timer repetitivo e passa para o próximo estado;

Estado 6: Ativa a matriz de LED's e começa a tocar o som programado através do Buzzer até que algum botão seja pressionado, então passando para o próximo estado;

Estado 7: Serve configurar o jogo, alocando a estrutura do tipo "memoryGame" que contém parâmetros do jogo da memória e sorteando a direção das flechas durante o jogo;

Estado 8: Jogo da memória, o estado 8 é complexo e contém uma máquina de estados dentro dele mesmo (demonstrada nas imagens acima):

> Subestado 0: Escreve na matriz de LED's as flechas necessárias para o nível atual, por exemplo no nível 3 são exibidas 3 flechas (utilizando a função show_arrow e a lookup table para facilitar). Então ativa-se um alarme para os próximos 10 segundos e passa para o próximo subestado;
>
> Subestado 1: Lê-se o joystick através da função get_joystick_position, caso não seja detectado movimento o alarme acionado no subestado anterior

disparar, voltando a dois estados anteriores e tocando novamente o despertador. Caso tenha movimento no joystick, é indicado na matriz de LED's, cancela-se o alarme e passa para o próximo subestado;

Subestado 2: Verifica-se se a flecha é igual a esperada para essa posição, em caso negativo exibe uma mensagem de "Você perdeu" no OLED e retorna pro estado anterior para configurar novamente o jogo. Em caso positivo, verifica se é a última flecha do nível (por exemplo, terceira flecha do terceiro nível), em caso negativo volta ao subestado 1, passa para detectar a próxima flecha e aciona o alarme dos 10 segundos novamente, em caso positivo verifica se passou por todos os níveis, em caso positivo passa para o próximo estado, em caso negativo zera a contagem de flechas, ativa o alarme de 10 segundos e volta ao subestado 1.

Estado 9: Lê o joystick através da função get_joystick_horizontal_pos, exibe os gráficos dos dados coletados anteriormente através da função oled_graph, como o display OLED é pequeno para exibir todos os dados utiliza-se o joystick para passar o gráfico para o lado, contudo, ainda é possível coletar os dados via UART e rodar o código em Python disponibilizado nos arquivos para gerar um gráfico pelo computador.

3.3. Definição das variáveis – descreva as principais variáveis usadas.

3.3.1. Variáveis Globais:

3.3.1.1. `int state`: Variável de estado da máquina de estados, determina qual função será executada no loop principal;

3.3.1.2. `volatile int button_press_count_a` e `button_press_count_b`: Contador de pressões dos botões A e B, respectivamente, modificado dentro da interrupção de borda e subida em ambos terminais;

3.3.1.3. `volatile int counter`: Contador usado para rastrear o número de amostras coletadas pelo microfone;

3.3.2. Ponteiros:

3.3.2.1. `uint8_t* data_sono`: Ponteiro para um array que armazena os dados coletados pelo microfone durante o monitoramento do sono;

3.3.2.2. `memoryGame* game`: Ponteiro para uma estrutura do tipo memoryGame, usado para gerenciar o jogo da memória, incluindo a sequência de direções, o nível atual e o estado interno do jogo;

3.3.2.3. `horario_atual[2]`, `horario_alarme[2]`, `horario_faltando[2]`: Arrays que armazenam o horários (horas e minutos) configurados pelo usuário ou calculados pela diferença dos dois primeiros;

3.3.3. Variáveis Locais:

3.3.3.1. `int micro_value`: Armazena o valor lido do microfone em um determinado instante, local à função de callback do timer repetitivo de 1 segundo;

3.3.3.2. `int display_pos`: Controla a posição atual no gráfico de dados do sono exibido no display OLED;

3.3.3.3. `bool mesmo_dia`: Indica se o alarme está configurado para o mesmo dia ou para o dia seguinte;

3.3.3.4. `int posicao_joystick, posicao_joystick_display`: controlam a posição atual do cursor no display na tela de configuração de horários;

3.3.3.5. `int variacao` e `bool changed`: servem para indicar se houve alguma movimentação no joystick na função de configuração de horários para evitar que o display seja limpado excessivamente fazendo com que ele fique piscando;

3.3.4. Estruturas:

3.3.4.1. memoryGame: Estrutura que armazena o estado do jogo da memória

3.3.4.1.1. `int substate`: Subestado do jogo (ex.: exibindo sequência, aguardando entrada do usuário);

3.3.4.1.2. `int level`: Nível atual do jogo;

3.3.4.1.3. `int flecha`: Posição atual na sequência de direções;

3.3.4.1.4. `enum Directions sequencia[LEVELS_MEMORY_GAME]`: Array que armazena a sequência correta de direções do jogo.

3.3.4.2. datetime_t: Estrutura fornecida pela biblioteca do RTC para armazenar data e hora

3.3.4.2.1. `int year, month, day, dotw, hour, min, sec`: Campos que representam a data e a hora.

3.3.5. Constantes:

3.3.5.1. `#define FLASH_TARGET_OFFSET (256 * 1024)`: Offset de 256 KB na memória flash, onde os dados do sono serão armazenados;

3.3.5.2. `#define DEBOUNCE_TIME_US 50000`: Tempo de debounce de 50 ms para os botões;

3.3.5.3. `#define MICROPHONE_SAMPLING_TIME 1000`: Intervalo de tempo (1 segundo) entre as leituras do microfone;

3.3.5.4. `#define LEVELS_MEMORY_GAME 6`: Número de níveis no jogo da memória;

3.3.5.5. `extern const npLED_t (*lookup_table_arrow[4])[5]`: No arquivo matriz_leds, armazena o padrão as flechas utilizadas no jogo para fácil acesso;

3.3.5.6. `extern const npLED_t (*lookup_table_indicaator[4])[5]`: No arquivo matriz_leds, armazena o indicador do movimento do usuário;

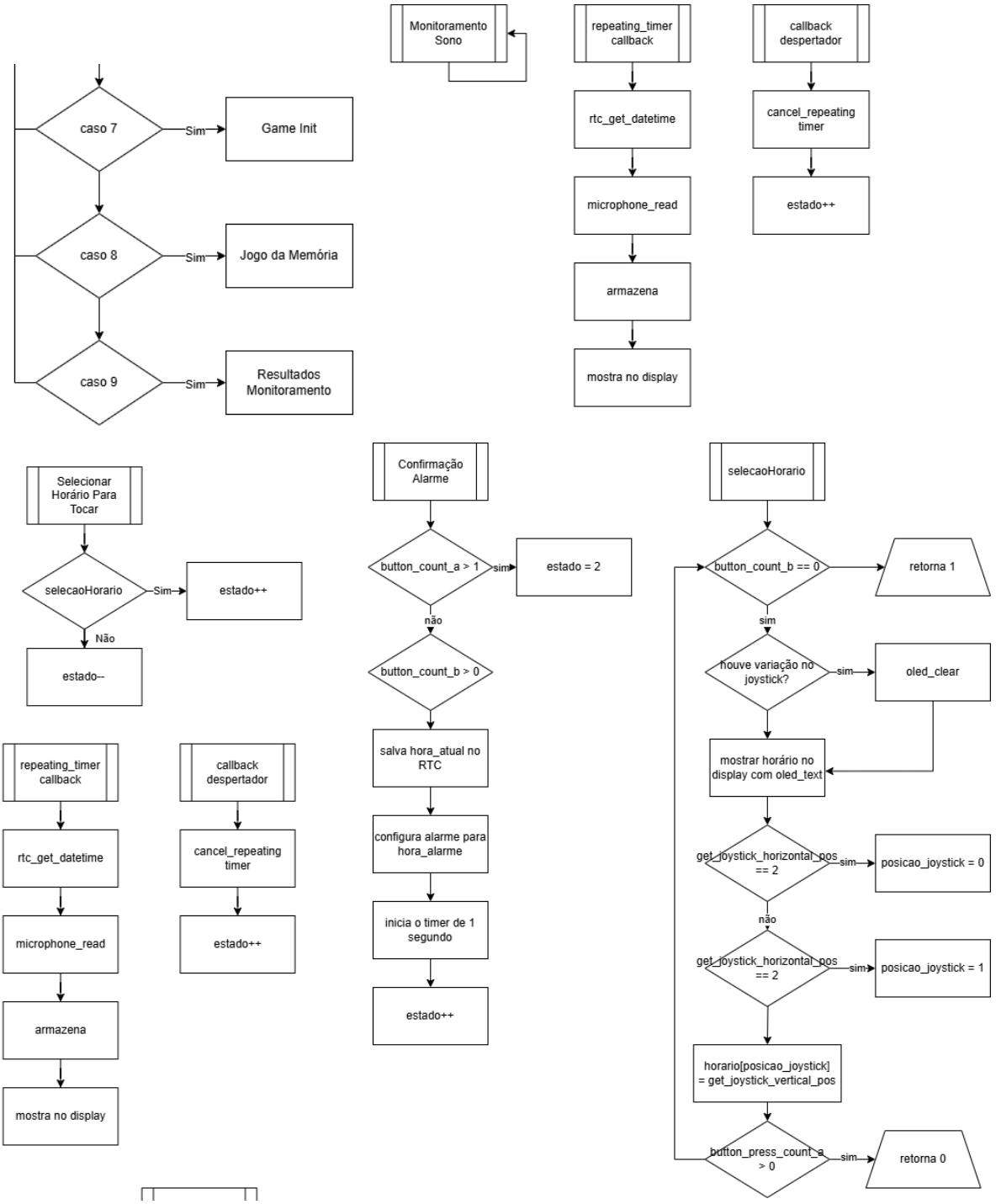
3.3.5.7. `#define SAMPLES 200`: dentro do arquivo microphone_dma, número de amostras que serão feitas do ADC;

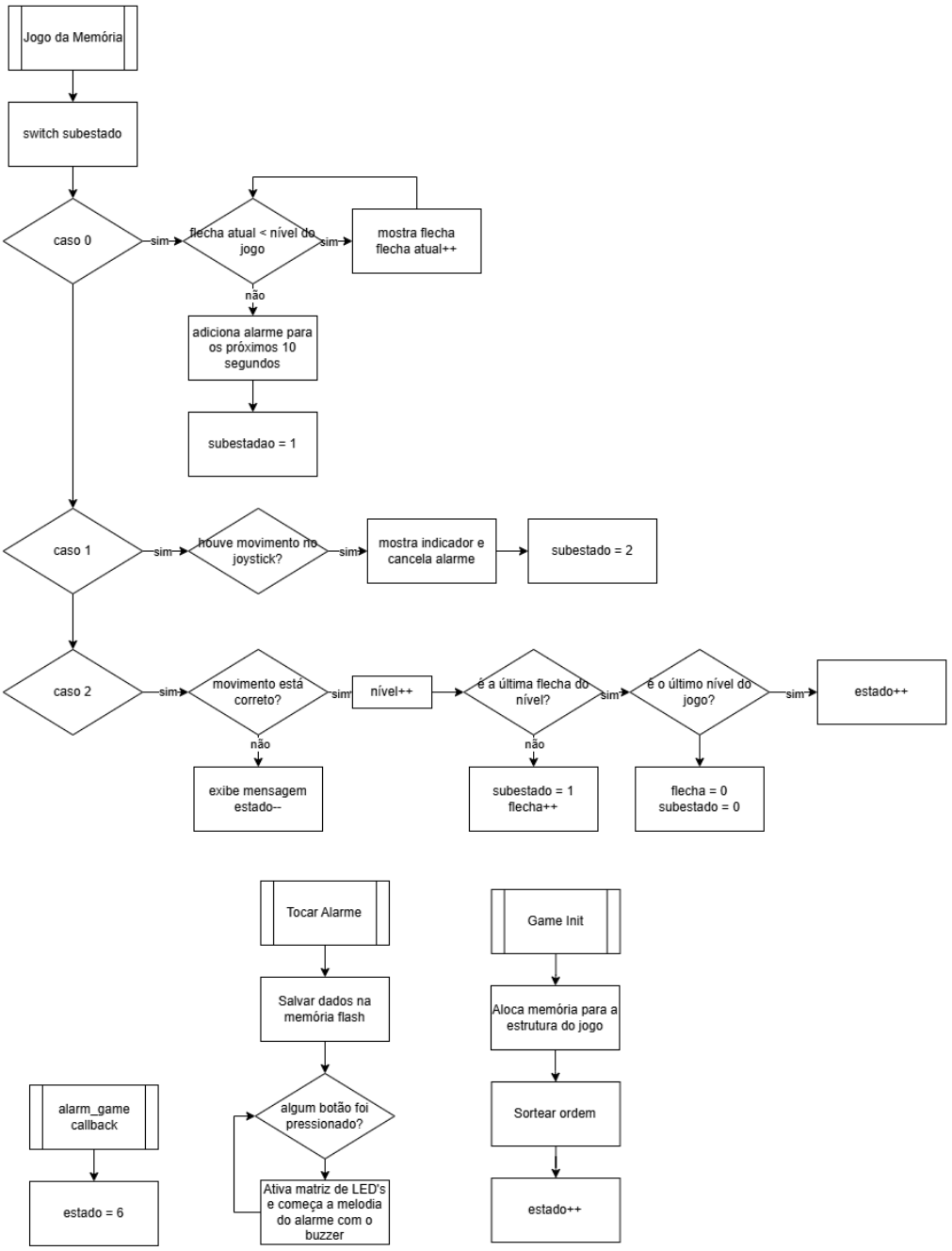
3.3.5.8. `define ADC_ADJUST(x) (x * 3.3f / (1 << 12u) - 1.65f)`: dentro do arquivo microphone_dma, ajuste do valor do ADC para volts;

3.3.5.9. `#define ADC_STEP (3.3f/5.f)`: dentro do arquivo microphone_dma, intervalos de volume do microfone;

3.3.5.10. `#define V_REF 0.006f`: dentro do arquivo microphone_dma, tensão mínima mensurável (~6mV como referência).

3.4. Fluxograma – desenhe o fluxograma completo do software.


caso 7
Sim
Game Init
caso 8
Sim
Jogo da Memória
caso 9
Sim
Resultados Monitoramento
Monitoramento Sono
repeating_timer callback
rtc_get_datetime
microphone_read
armazena
mostra no display
callback despertador
cancel_repeating timer
estado++
Selecionar Horário Para Tocar
selecaoHorario
Sim
estado++
Não
estado--
repeating_timer callback
rtc_get_datetime
microphone_read
armazena
mostra no display
callback despertador
cancel_repeating timer
estado++
Confirmação Alarme
button_count_a > 1
sim
estado = 2
não
button_count_b > 0
salva hora_atual no RTC
configura alarme para hora_alarme
inicia o timer de 1 segundo
estado++
selecaoHorario
button_count_b == 0
retorna 1
sim
houve variação no joystick?
sim
oled_clear
mostrar horário no display com oled_text
get_joystick_horizontal_pos == 2
sim
posicao_joystick = 0
não
get_joystick_horizontal_pos == 2
sim
posicao_joystick = 1
horario[posicao_joystick] = get_joystick_vertical_pos
button_press_count_a > 0
sim
retorna 0



Figuras 10 até 17: Fluxograma do código

3.5. Inicialização – descreva o processo de inicialização do software.

3.5.1. stdio_init_all: Inicializa a comunicação serial;

3.5.2. oled_init: Configura o display oled, configurando o I2C e a biblioteca do ssd1306;

3.5.3. joystick_init: Configura o adc e os pinos associados;

3.5.4. button_init: Configura os pinos com resistor de pullup, ativa a interrupção de subida e descida para ambos e salva o tempo inicial para lógica de debounce;

3.5.5. buzzer_init: Configura o PWM a 100 Hz no pino do buzzer;

3.5.6. rtc_init: Ativa o RTC;

3.5.7. npInit: Cria, toma posse e inicia o programa em uma máquina PIO;

3.5.8. microphone_init: Configura o ADC para esse pino
3.5.9. srand(time(NULL)): Inicializa o gerador de números aleatórios.

3.6. Configurações dos registros – descreva as funções de configuração dos registros.

3.6.1. Configuração de GPIOs
3.6.1.1. SIO.GPIO_OUT: Define o estado dos pinos de saída;
3.6.1.2. SIO.GPIO_OE: Habilita ou desabilita a saída dos pinos;
3.6.1.3. IO_BANK0.GPIOx_CTRL: Configura o modo de funcionamento do pino (entrada, saída, função alternativa);
3.6.2. Configuração do Clock
3.6.2.1. CLOCKS.CLK_REF_CTRL: Define a fonte do clock de referência;
3.6.2.2. CLOCKS.CLK_SYS_CTRL: Configura a fonte e o divisor do clock do sistema;
3.6.2.3. CLOCKS.CLK_PERI_CTRL: Ajusta o clock dos periféricos;
3.6.3. Configuração de Timers
3.6.3.1. TIMER.TIMELR: Define o valor de contagem do timer;
3.6.3.2. TIMER.ALARM: Configura alarmes para gerar interrupções;
3.6.4. Configuração de PWM
3.6.4.1. PWM.CHx_CSR: Controla o funcionamento do canal PWM;
3.6.4.2. PWM.CHx_DIV: Define a divisão de clock para o PWM;
3.6.4.3. PWM.CHx_CC: Ajusta o duty cycle;
3.6.5. Configuração de UART
3.6.5.1. UARTx_BAUD: Define a taxa de transmissão;
3.6.5.2. UARTx_CTRL: Habilita e configura os modos de operação;
3.6.5.3. UARTx_FR: Indica status da fila de transmissão/recepção;
3.6.6. Configuração do RTC
3.6.6.1. RTC_SETUP: Configura o RTC antes de habilitá-lo;
3.6.6.2. RTC_CTRL: Controla a ativação e desativação do RTC;
3.6.6.3. RTC_IRQ_SETUP: Configura alarmes para gerar interrupções;
3.6.6.4. RTC_1: e RTC_0: Registradores que armazenam o tempo e a data atuais;
3.6.7. Configuração da memória Flash:
3.6.7.1. XIP_CTRL: Controla a ativação do acesso XIP;
3.6.7.2. XIP_SSI_SR: Indica o status do controlador Flash;
3.6.7.3. XIP_SSI_DR0: Permite enviar e receber dados;
3.6.7.4. XIP_SSI_CTRLR0: Configura os modos de operação da Flash;
3.6.7.5. XIP_CACHE_CTRL: Controla o cache;
3.6.7.6. FLASH_CFG: Define parâmetros de configuração da memória Flash;
3.6.8. Configuração de PIO
3.6.8.1. CTRL: Configura e habilita o bloco PIO;
3.6.8.2. FSTAT: Indica o status das FIFOs de entrada e saída;
3.6.8.3. FDEBUG: Fornece informações de depuração sobre as FIFOs;
3.6.8.4. TXF: FIFOs de transmissão para os quatro estados da máquina;
3.6.8.5. RXF: FIFOs de recepção para os quatro estados da máquina;

3.6.8.6. IRQ: Configuração e status das interrupções PIO;
3.6.8.7. SMx_CLKDIV: Define o divisor de clock para a máquina de estado;
3.6.8.8. SMx_EXECCTRL: Configura o comportamento da execução do programa;
3.6.8.9. SMx_SHIFTCTRL: Configura deslocamento de bits e alinhamento nas FIFOs;
3.6.8.10. SMx_PINCTRL: Define quais pinos GPIO a máquina de estado pode controlar;

3.6.9. Configuração de DMA:
3.6.9.1. CHx_CTRL: Configura o canal de DMA;
3.6.9.2. CHx_READ_ADDR: Endereço de origem da transferência;
3.6.9.3. CHx_WRITE_ADDR: Endereço de destino da transferência;
3.6.9.4. CHx_TRANS_COUNT: Número de bytes/palavras a serem transferidos;
3.6.9.5. CHx_AL1_CTRL: Estado atual do canal de DMA;
3.6.9.6. INTR: Indica interrupções de DMA pendentes;
3.6.9.7. TIMER: Configura temporizadores para sincronização de transferências.

3.7. Estrutura e formato dos dados – descreva os dados específicos usados no seu software.

3.7.1. Dados de Monitoramento do Sono: Usa o formato de um número inteiro de 1 byte (8 bits), usado para armazenar na memória flash os dados então alocados no heap de memória;

3.7.2. Horário Atual, Horário do Alarme, Horário Faltando: cada um contém 2 inteiros horas (0–23) e minutos (0–59);

3.7.3. Estrutura do Jogo da Memória:
substate: Controla o estado do jogo;
level: Nível atual do jogo;
flecha: Posição atual na sequência de direções;
sequencia: Array que armazena a sequência de direções gerada aleatoriamente;

3.7.4. Dados do RTC (Real-Time Clock):
year: Ano (ex.: 2025);
month: Mês (1–12);
day: Dia (1–31);
dotw: Dia da semana;
hour: Hora (0–23);
min: Minutos (0–59);
sec: Segundos (0–59);

3.7.5. Dados imprimidos na UART para análise em Python: “HH:MM:SS - VVV”, onde representam hora, minuto, segundo e volume.

3.8. Organização da memória – descrição dos endereços de memória que você usou.

Tipo de Memória	Endereço Inicial	Descrição
Flash (Programa)	0x10000000	Armazena o código do programa (firmware).
Flash (Dados)	0x10040000	Armazena os dados do sono (data_sono)
RAM (.data)	0x20000000	Armazena variáveis globais inicializadas.
RAM (.bss)	0x20000000 + .data	Armazena variáveis globais não inicializadas
RAM (Heap)	Após .bss	Usado para alocação dinâmica de memória (ex.: data_sono)
RAM (Stack)	0x20041F00 (Core 0)	Usado para variáveis locais e contexto de funções
Registradores GPIO	0x40014000	Configuração e dados do periférico
Registradores RTC	0x4005C000	Configuração e dados do periférico
Registradores Timer	0x40054000	Configuração e dados do periférico
Registradores ADC	0x4004C000	Configuração e dados do periférico
Registradores DMA	0x50000000	Configuração e dados do periférico
Registradores PWM	0x40050000	Configuração e dados do periférico

Tabela 3: Endereços de memória utilizados

3.9. Protocolo de comunicação – descreva o protocolo, se existir.

3.9.1. I2C: A comunicação com o display OLED é feita através do protocolo I2C (Inter-Integrated Circuit), um protocolo serial síncrono que utiliza dois fios: SDA (dados) e SCL (clock). Cada dispositivo na rede I2C é chamado de "slave" e tem um endereço (ssd1306: 0x3C), a não ser o microcontrolador que comanda tudo, o "master". É considerado uma comunicação lenta, com uma velocidade padrão de 100 kbps, modo rápido 400 kbps e alta velocidade 3,4 Mbps;

3.9.2. WS2812B: É um protocolo específico dos LED's RBG de neopixel, ele usa comunicação serial com um único fio de dados para enviar informações para os LEDs. Cada LED recebe 24 bits de dados (8 bits para vermelho, 8 bits para verde e 8 bits para azul)

3.10. Formato do pacote de dados – descreva a formação dos pacotes, se existir.

3.10.1. Dados de sono no formato: "HH:MM:SS - VVV", hora, minuto, segundo e volume;

3.10.2. Dados do RTC no formato: {year, month, day, dotw, hour, min, sec}.

4. Execução do projeto (3 pontos no total)

4.1. Metodologia – descrição da execução das etapas do projeto: pesquisas realizadas, escolha do hardware, definição das funcionalidades do software, inicialização da IDE, programação na IDE, depuração.

Inicialmente, foram exploradas diferentes possibilidades dentro do escopo do projeto, considerando os recursos disponíveis na BitDogLab. Após essa análise, optou-se pelo desenvolvimento de um sistema de monitoramento de sono e despertador iterativo. Embora a inclusão de sensores adicionais, como batimentos cardíacos, movimento e temperatura, pudesse enriquecer o projeto, concluiu-se que a análise acústica por meio do microfone seria suficiente para obter resultados relevantes.

A escolha do hardware foi baseada nos periféricos já disponíveis na placa, visto que não havia outros dispositivos acessíveis no momento. Com isso, cada periférico necessário foi configurado separadamente, organizando suas implementações em arquivos distintos para modularizar o código. Esse processo contou com o apoio de vídeos instrucionais sobre a configuração do ambiente de trabalho e com referências do repositório BitDogLab-C.

Na etapa de desenvolvimento do software, foram definidos os estados do sistema e programados de forma independente, garantindo a separação das funcionalidades e evitando interferências entre as diferentes partes do código.

Para a depuração, foram utilizados diversos recursos, como a comunicação UART, permitindo a impressão de valores de variáveis para análise, além do LED RGB da placa, que foi empregado para indicar estados específicos da execução. Adicionalmente, os recursos de debugging do Visual Studio Code foram aproveitados para identificar e corrigir possíveis falhas no funcionamento do sistema.

4.2. Testes de validação – descreva os testes realizados para validação do funcionamento.

Para validar o funcionamento do sistema, foram realizados testes individuais dos periféricos, verificando a captura de áudio pelo microfone, a resposta do LED RGB e a comunicação via UART. Em seguida, testes funcionais avaliaram a detecção de padrões sonoros, o acionamento do despertador e a resposta em tempo real do sistema.

4.3. Discussão dos Resultados – analise os resultados e conclua sobre a confiabilidade e aplicabilidade do projeto.

Os resultados demonstraram que o sistema foi capaz de detectar padrões sonoros e ativar o despertador de forma eficiente, validando sua funcionalidade. A confiabilidade foi assegurada por meio de testes repetidos, que mostraram um desempenho consistente na captação e processamento do áudio. No entanto, limitações como a sensibilidade do microfone e possíveis interferências ambientais foram observadas, exigindo refinamentos na filtragem dos dados. Apesar disso, a aplicabilidade do projeto se mostrou promissora, especialmente como uma solução acessível para monitoramento de sono, podendo ser aprimorada com sensores adicionais para maior precisão e com novas funcionalidades através da conexão Wi-Fi, podendo gerar o gráfico automaticamente pelo website.

4.4. Faça um vídeo de no máximo 3 minutos mostrando seu projeto funcionando. Inclua o link do vídeo no seu relatório de entrega.

Apresentação e demonstração (VÍDEO OFICIAL)

Links Úteis:

- Projeto no GitHub: https://github.com/EduardoTejada/BitDorme
- Apresentação: Link
- LinkedIn: https://www.linkedin.com/in/eduardo-pereira-tejada/
- Vídeo de 3 minutos: Apresentação e demonstração (VÍDEO OFICIAL)

REFERÊNCIAS

OSUL. A influência da qualidade do sono na saúde: especialista analisa três doenças relacionadas à falta do descanso. Disponível em: https://www.osul.com.br/a-influencia-da-qualidade-do-sono-na-saude-especialista-analisa-tres-doencas-relacionadas-a-falta-do-descanso/. Acesso em: 15 fev. 2025.

HOSPITAL ISRAELITA ALBERT EINSTEIN. Semana do Sono: quantidade e qualidade. Disponível em: https://vidasaudavel.einstein.br/semana-do-sono-quantidade-e-qualidade/. Acesso em: 15 fev. 2025.

COOL-MANIA. Sleep Dot: analisador da qualidade do sono. Disponível em: https://www.cool-mania.pt/gadgets/gadgets-fabulosos/sleep-dot-analisador-da-qualidade-do-sono. Acesso em: 16 fev. 2025.

SECAD. Aplicabilidade e confiabilidade dos diferentes métodos de monitoração do sono. Disponível em: https://portal.secad.artmed.com.br/artigo/aplicabilidade-e-confiabilidade-dos-diferentes-metodos-de-monitoracao-do-sono. Acesso em: 16 fev. 2025.

CANALTECH. Melhores aplicativos de alarme. Disponível em: https://canaltech.com.br/apps/melhores-aplicativos-alarme/. Acesso em: 16 fev. 2025.

CONSULTA CARIOCA. A importância da qualidade do sono para a saúde. Disponível em: https://www.consultacarioca.com.br/a-importancia-da-qualidade-do-sono-para-a-saude/. Acesso em: 20 fev. 2025.

Documentação técnica e datasheets:

RASPBERRY PI. Pico SDK - High-Level Documentation. Disponível em: https://www.raspberrypi.com/documentation/pico-sdk/high_level.html. Acesso em: 19 fev. 2025

GITHUB - BITDOGLAB. BitDogLab-C. Disponível em: https://github.com/BitDogLab/BitDogLab-C/tree/main. Acesso em: 18 fev. 2025.

FRUETT - BITDOGLAB. GitHub. Disponível em: https://github.com/BitDogLab/BitDogLab/tree/main/kicad/bitdoglabsmd/bitdoglab_main. Acesso em: 18 fev. 2025.

RASPBERRY PI. RP2040 Datasheet. Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 18 fev. 2025.

ADAFRUIT. SSD1306 Datasheet. Disponível em: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf. Acesso em: 20 fev. 2025.

TME. WS2818B Datasheet. Disponível em: https://www.tme.eu/Document/ddc250a349c0084fadc3ded4c327f335/WS2818B.pdf. Acesso em: 20 fev. 2025.

Livros e dissertações:

CUGNASCA, C. E. Projetos de Sistemas Embarcados. Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP, 2018.

Projetos relacionados:

SRVASSILI. Sleep Quality Monitor. Disponível em: https://projecthub.arduino.cc/SrVassili/sleep-quality-monitor-48a511. Acesso em: 17 fev. 2025.

NICOLAS HU. Arduino Sleep Monitor. Disponível em: https://github.com/NicolasHu11/Arduino-Sleep-Monitor. Acesso em: 17 fev. 2025.

REDDIT. I made an alarm clock that you can't turn off from bed. Disponível em: https://www.reddit.com/r/arduino/comments/c2x9ww/i_made_an_alarm_clock_that_you_can_t_turn_off_from/. Acesso em: 17 fev. 2025.