Aluno: Miguel Carvalho de Medeiros
Cidade: Brasília - DF
GitHub: `https://github.com/Miguel-MCM/BitDogCalendar`

1 Escopo do Projeto

1.1 Apresentação do Projeto

Este relatório detalha o desenvolvimento de um calendário inteligente utilizando a placa BitDogLab, baseada em uma Raspberry Pi Pico W. O projeto visa criar um dispositivo que permita ao usuário gerenciar seu compromisso e ser notificado por um alarme.

1.2 Título do Projeto

BitDog Calendar

1.3 Objetivos do Projeto

- Desenvolver um sistema de calendário eletrônico funcional.
- Implementar uma interface de usuário intuitiva para marcação de eventos e configuração de alarmes.
- Integrar funcionalidades de alarme sonoro e/ou visual para notificação de eventos.
- Criar um projeto de código aberto, documentado e de fácil reprodução

1.4 Principais Requisitos

- Cadastro de Eventos com:
  - Nome do Evento (ex "Reunião com equipe");
  - Data (dia, mês e ano);
  - Hora (horário exato do evento).
- Visualização dos Eventos diários:
  - O usuário deve possuir algum meio de visualizar os eventos marcado para o dia atual.
- Alarmes:
  - O sistema deve alertar o usuário do momento de início de um evento por meio de um alarme sonoro.
- Interface Simples e Intuitiva:
  - A interface deve ser fácil de usar, com botão analógico para seleção de opções e botões físicos para confirmação e retorno;
  - Display claro e legível para exibição de informações;
  - O usuário não deve precisar de conhecimento técnico para operar o sistema.

1.5 Descrição de Funcionamento

Visualização de Eventos com Matriz de LEDs

- O sistema utiliza a matriz de LEDs para representação dos horários, cada LED corresponde a uma hora específica do dia;
- A cor de cada LED indica o tipo ou categoria do evento agendado para aquela hora permitindo uma visualização rápida e intuitiva dos compromissos diários.

Interface de Usuários para Gerenciamento de Eventos

- O display OLED integrado exibe o horário atual em tempo real;
- O usuário pode utilizar o controle analógico para navegar entre as opções do sistema. Ao mover o controle, é possível alternar da exibição do horário atual para o menu de criação de novos eventos;
- Ao pressionar o botão **A**, o usuário confirma a seleção da opção destacada, como iniciar o processo de criação de um novo evento;
- Processo de Criação de Evento:
  - O usuário é guiado por uma série de telas, onde pode inserir detalhes do evento, como nome, data e hora de início e fim e cor do LED correspondente;
  - Durante a seleção dos dados o controle analógico permite a entre os campos disponíveis;
  - O botão **A** irá confirmar as informações apresentadas;
  - O botão **B** poderá ser utilizado para retornar à tela anterior, caso necessite corrigir algo ou alterar alguma informação.

Notificações e Alarmes

- Quando o horário atual coincide com o de um evento agendado, o sistema ativa um alarme sonoro para notificar o usuário sobre o compromisso iminente.

1.6 Justificativa

A organização do tempo é crucial na vida moderna. Um calendário inteligente com alarmes personalizados pode auxiliar na gestão de compromissos e prazos, aumentando a produtividade e reduzindo o esquecimento de tarefas importantes. Embora existam aplicativos de calendário para smartphones e computadores, um dispositivo dedicado oferece a vantagem de independência de outros aparelhos, além de personalização conforme as necessidades específicas do usuário. A Raspberry Pi Pico W oferece uma plataforma acessível e poderosa para a criação de um dispositivo personalizado, com baixo custo e consumo de energia.

Além disso, o projeto serve como uma excelente oportunidade para explorar as capacidades da placa, incluindo sua conectividade Wi-Fi, que pode ser utilizada para expandir as funcionalidades no futuro.

1.7 Originalidade

Embora existam projetos que utilizam o Raspberry Pi Pico W para criar relógios com alarmes, como o "Raspberry Pi Pico - Alarm Clock"[1] , e iniciativas que integram calendários familiares com displays, como o "How to build a Wall Mounted Family Calendar and Dashboard with a Raspberry Pi and cheap monitor"[2], e também o projeto "Workday Progressbar With Google

Calendar Integration"[3], utilizando a API do google calendar para o gerenciamento dos eventos, o projeto se destaca por combinar várias funcionalidades de maneira única:

- Matriz de LEDs para Visualização de Eventos: A utilização de uma matriz de LEDs, onde cada LED representa uma hora do dia e sua cor indica o tipo de evento, oferece uma representação visual imediata e intuitiva dos compromissos diários;
- Interface de Controle Analógico: A implementação de um controle analógico para navegação entre opções e criação de eventos proporciona uma interação mais fluida e natural para o usuário.
- Integração de Alarmes Sonoros e Visuais: A combinação de notificações sonoras e visuais assegura que o usuário seja alertado de maneira eficaz sobre eventos iminentes.

2 Hardware

2.1 Diagrama em Blocos

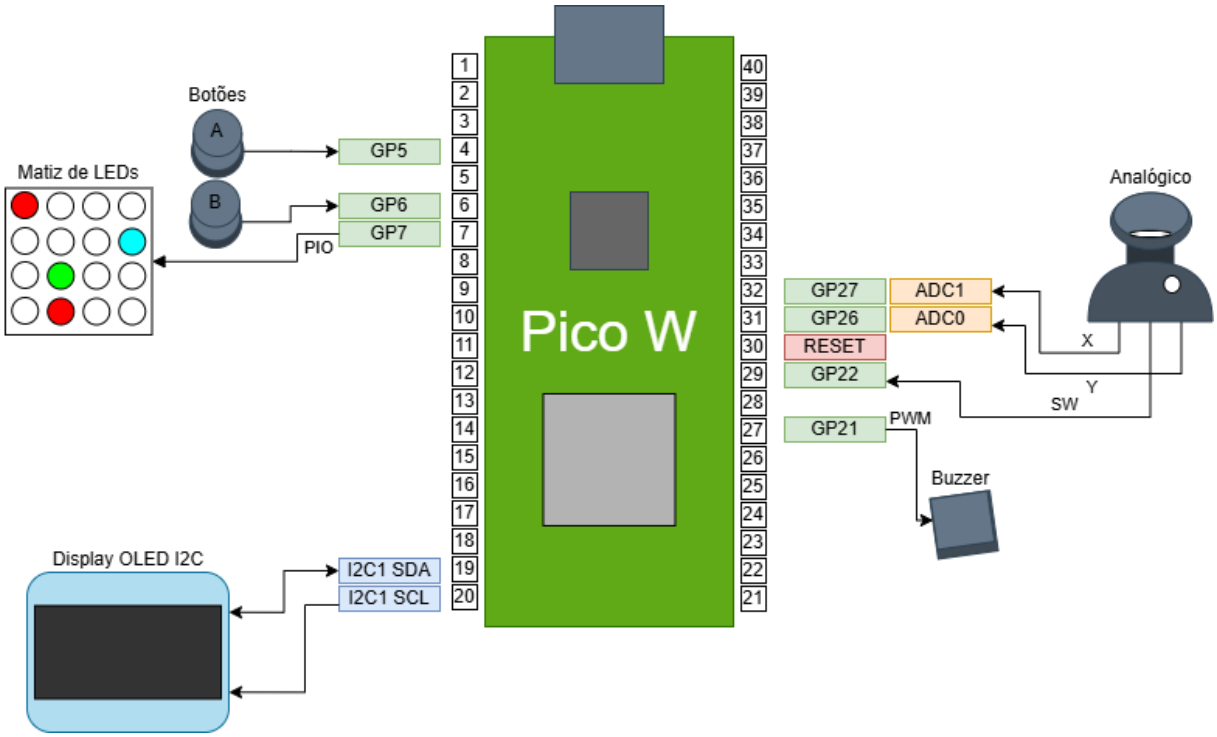

Figura 1: Diagrama de Blocos do Hardware

Neste diagrama já se assume a devida conexão de **Ground** e **VCC**, para os que requerem alimentação, dos componentes.

2.2 Função de Cada Bloco

Pico W

Representa a placa Raspberry Pi Pico W, nela está o microcontrolador RP2040, responsável por toda a lógica do programa, além dos outros módulos de comunicação, PIO, RTC, ADC e todos os módulos centrais do sistema.

Botão A

Botão interpretado por meio da entrada GP5 da placa. Responsável por sinal de confirmação e seleção.

Botão B

Botão interpretado por meio da entrada GP6 da placa. Responsável por sinal de negação e retroceder.

Matriz de LEDs

Representa a matriz de LEDs RGB 5x5 WS2812 que está conectada à placa pela saída GP7. Utilizando PIO, a placa envia o padrão que deve estar aceso na matriz.

A matriz é utilizada para a representação dos horários, cada LED corresponde a uma hora específica do dia, e sua cor indica se há e qual é o evento nesse horário.

Display OLED I2C

Representa o display OLED 128x64 de 0,96 polegadas, ssd1306, que se conecta a placa por meio das entradas de I2C1, SDA e SCL.

Através da comunicação I2C a placa define a informação que será apresentada na tela.

A tela é utilizada para a exibição do horário atual, e também para apresentar as telas de preenchimento de informações para cadastro de eventos.

Analógico

Joystick Analógico de 13x13mm Multi-Dir ROHS, conectado à entrada de ADC1 e ADC2 do placa, com leituras de posição X e Y, respectivamente.

A placa periodicamente faz a leitura da posição do joystick e utiliza essa informação para a navegação e seleção de opções presentes na tela.

O Joystick também está conectado a entrada GP21 da placa, para a leitura de seu pressionamento. Mas essa função não é utilizada.

Buzzer

O buzzes está conectado à saída GP21 da placa e é ativado, quando necessário, por meio de PWM.

A buzina é utilizada para alertar o usuário do momento de início de um de seus eventos.

2.3 Configuração de Cada Bloco

Pico W

Utilizada em modo padrão, com interrupções e um loop principal a cada 250 ms.

Botão A

Configurado por GPIO para leitura, com resistor de pull-up.

Botão B

Configurado por GPIO para leitura, com resistor de pull-up.

Matriz de LEDs

Conectada como saída da Pico W, utilizando PIO, de forma a enviar os dados seguindo o padrão de comunicação da matriz de LED.

Display OLED I2C

Se comunica com a placa por meio de I2C, a uma frequência de 400 Hz.

A placa age como mestre, utilizando a interface I2C1.

O display interage como escravo. Ele está configurado para estar completamente ligado e sem scrool.

Analógico

Conectado as entrada de ADC da placa.

Buzzer

O buzzer está conectado à saída GP21 da placa e é ativado, quando necessário, por meio de PWM, com uma frequência de 1760 Hz.

2.4 Especificações

Pico W

A placa Raspberry Pi Pico W é o núcleo do sistema, sendo responsável por toda a lógica de controle do calendário inteligente. O microcontrolador RP2040 gerencia a execução do programa e a comunicação entre os diferentes periféricos, garantindo a sincronização adequada das funcionalidades do sistema. Os módulos PIO e ADC são utilizados para interfacear com a matriz de LEDs e o joystick analógico, enquanto o RTC permite manter a exibição precisa do horário.

Botão A

Funciona como entrada digital e está configurado para leitura com resistor de pull-up, permitindo ao usuário confirmar seleções e avançar na interface do sistema.

Botão B

Também configurado com resistor de pull-up, é utilizado para cancelar ações e retroceder dentro das telas do sistema.

Ambos os botões garantem uma interação eficiente com o calendário, proporcionando controle intuitivo na navegação dos menus e criação de eventos.

Matriz de LEDs

A matriz de LEDs é controlada via PIO para exibir informações sobre os eventos ao longo do dia. Cada LED representa uma hora específica e sua cor varia conforme o tipo de evento agendado, permitindo uma visualização rápida dos compromissos do usuário.

Display OLED I2C

O display OLED exibe as principais informações do calendário, como o horário atual e as telas de configuração dos eventos. Ele garante uma atualização fluida da interface.

Analógico

O joystick analógico permite a navegação pelo sistema e a seleção de opções exibidas no display OLED.

Buzzer

O buzzer é responsável por emitir alertas sonoros sempre que um evento programado estiver prestes a ocorrer. Isso garante que o usuário receba notificações audíveis de seus compromissos.

2.5 Lista de Materiais

Tabela 1: Componentes Utilizados no Projeto

Nome	**Descrição**	**Quantidade**
Raspberry Pi Pico W	Placa com microcontrolador RP2040	1
Chave Táctil 12x12x7,5 mm	Botões A e B	2
Matriz de LEDs RGB 5x5 WS2812	Matriz de LEDs endereçáveis	1
Display OLED SSD1306 I2C 128x64	Display OLED 0.96", controlado via I2C	1
Joystick Analógico	Joystick de 13x13mm	1
Buzzer	Buzzer para notificações sonoras	1
Fonte de Alimentação	Fonte USB ou bateria 3,7V presente na BitDogLab	1

2.6 Descrição da Pinagem

Tabela 2: Pinagem Utilizada no Projeto

Pino da Pico W	**Componente Conectado**	**Descrição da Conexão**
GP5	Botão A	Entrada digital para confirmação e seleção (com resistor de pull-up)
GP6	Botão B	Entrada digital para negação e retrocesso (com resistor de pull-up)
GP7	Matriz de LEDs RGB 5x5 WS2812	Saída digital para controle da matriz de LEDs via PIO
GP20 (SDA)	Display OLED I2C	Comunicação I2C (dados) para o display OLED
GP21 (SCL)	Display OLED I2C	Comunicação I2C (clock) para o display OLED
GP26 (ADC0)	Joystick Analógico (Eixo X)	Entrada analógica para leitura do eixo X do joystick
GP27 (ADC1)	Joystick Analógico (Eixo Y)	Entrada analógica para leitura do eixo Y do joystick
GP22	Buzzer	Saída PWM para controle do buzzer
3V3	Matriz de LEDs, Display e Joystick	Alimentação de 3.3V para os componentes
GND	Todos os componentes	Conexão de terra (GND) para os componentes

2.7 Circuito Completo

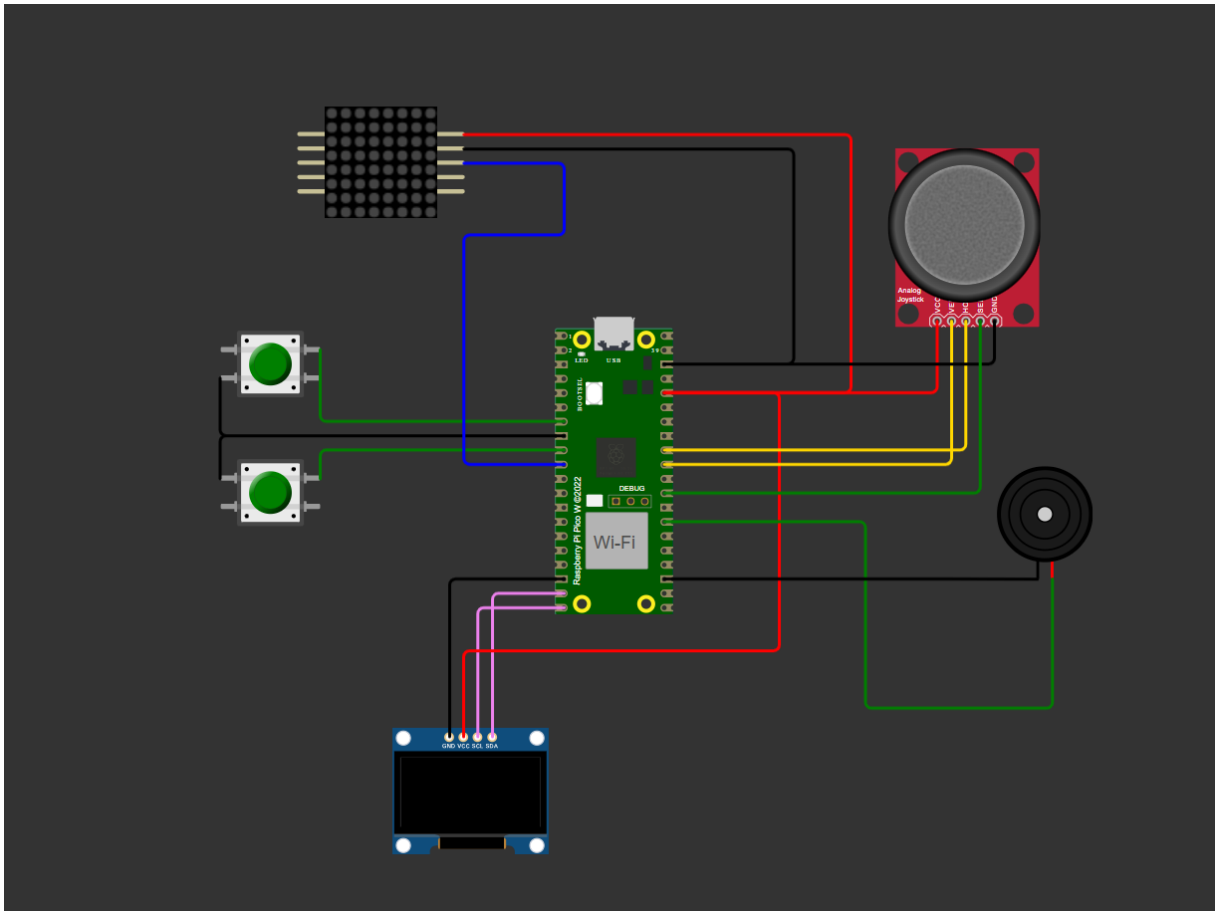

Figura 2: Circuito Completo do hardware

3 Software

3.1 Blocos Funcionais

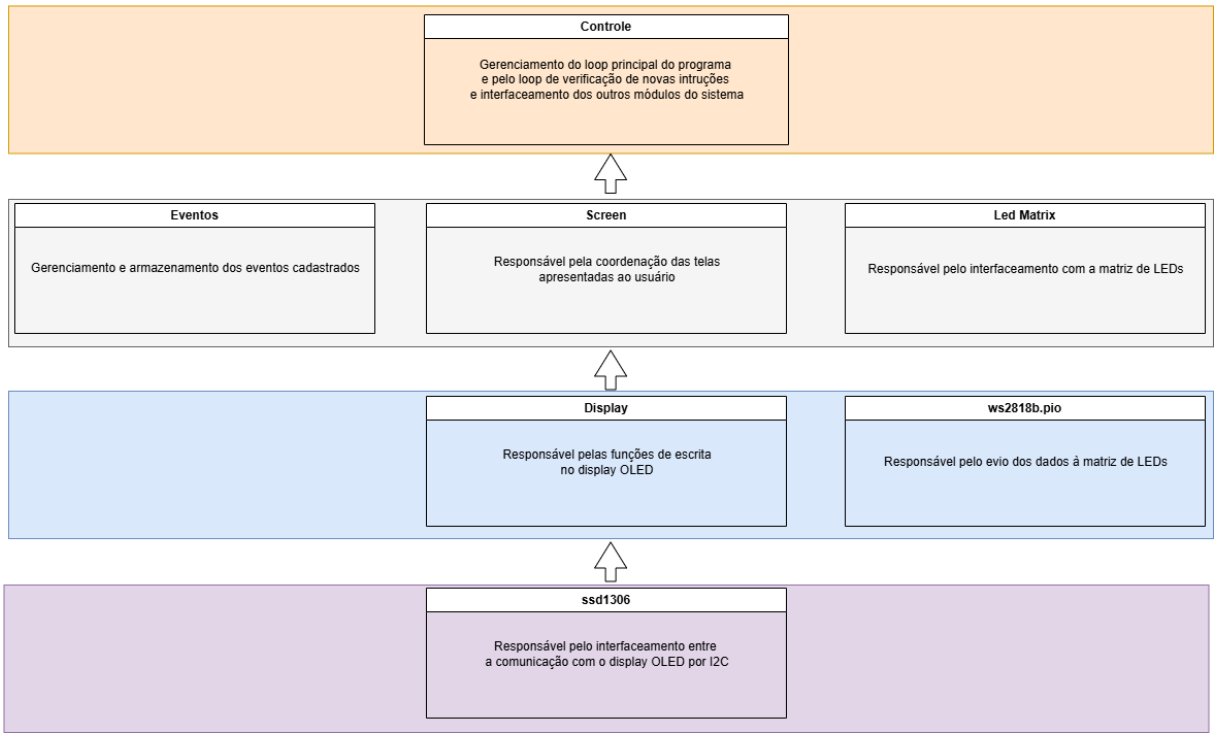

Figura 3: Diagrama de Blocos com as principais camadas do sistema.

3.2 Descrição das funcionalidades

Control

tua como o módulo de controle central e orquestrador do sistema. É responsável pela inicialização de todos os outros módulos, coordena a interação entre eles e gerencia o fluxo geral da aplicação.

Principais funcionalidades:

- `Ctr_setupAll`: Configura todos os módulos do sistema (timer AON, entrada, buzzer). É a função de arranque principal.
- Fornece funções para inicializar componentes de hardware individuais como o timer Always-On (AON) (`Ctr_setup_aon_timer`), o sistema de entrada (`Ctr_setupInput`), e o buzzer (`Ctr_setupBuzzer`);
- Oferece uma função para adicionar um novo evento ao sistema;
- `Ctr_buzzerBeep`: Permite gerar um som "beep"usando o buzzer por uma duração especificada;
- `Ctr_addEvent`: Oferece uma função para adicionar um novo evento ao sistema;
- `Ctr_menuLoop`: Contém o ciclo principal da aplicação, responsável por processar entradas do utilizador, atualizar a tela e gerir o fluxo da aplicação entre diferentes "telas"ou menus.

Led Matrix

Responsável por controlar a matriz de LEDs, que é usada para exibir as horas e indicação de eventos diários.

O código para a interação com a matriz física por PIO é o código de exemplo da BitDogLab para operação com a matriz de leds [4].

Principais funcionalidades:

- `LM_setup`: Configura o hardware necessário para controlar a matriz de LEDs (PIO, timer para blinking, etc.).
- `LM_setHourColor`: Permite definir a cor RGB de um LED específico, correspondente a uma hora ou posição na matriz.
- `LM_nextBlinkLed, LM_setBlinkLed e LM_update`: Implementam a funcionalidade de um LED piscar. Permite avançar para o próximo LED a piscar, definir um LED específico para piscar e atualizar o estado do "blinking"na matriz de LEDs.

Events

Lida com o armazenamento e a gestão dos eventos do calendário.

- `Ev_newEvent`: Permite adicionar um novo evento à lista de eventos armazenados.
- `Ev_updateEvent`: Permite modificar os detalhes de um evento já armazenado.
- `Ev_getEvent`: Permite recuperar um evento específico usando o seu índice único.
- `Ev_searchEvent`: Permite procurar por um evento que esteja ativo (dentro do seu período de tempo) num determinado momento.

Screen

Gerencia a lógica de diferentes "telas"ou vistas da aplicação, como o menu principal e a tela de criação de novos eventos. Define como a interface do utilizador é apresentada e como o utilizador interage com a aplicação em cada estado.

- `Sc_changeScreen`: Permite alternar entre diferentes telas da aplicação, limpando o display e definindo a função de atualização correta para a nova tela.
- `Sc_Menu_update`: Implementa a lógica específica para a tela do menu principal, incluindo a exibição de opções de menu, navegação com o joystick analógico e seleção de ações com botões.
- `Sc_New_Event_update`: Implementa a lógica para a tela onde o utilizador pode criar um novo evento, guiando-o através de várias etapas (seleção de hora, nome, alarme, cor) e processando a entrada do utilizador em cada etapa.
- Funções Auxiliares de Input (e.g., `getInputInt`, `Sc_New_Event_select_time`, etc.): Inclui funções auxiliares para simplificar a obtenção de inputs do utilizador e a gestão de etapas dentro das telas mais complexas, como a tela de "Novo Evento".

Display

Abstrai a interação com o display OLED (SSD1306), fornecendo funções de alto nível para desenhar texto e outros elementos visuais no display.

- `Dp_setup`: Configura a comunicação I2C e inicializa o controlador do display OLED (SSD1306).
- `Dp_drawString` e `Dp_drawChar`: Permite desenhar strings e caracteres individuais em posições específicas do display.
- `Dp_drawUInt`: Permite desenhar números inteiros sem sinal com opção de preenchimento com zeros à esquerda.
- `Dp_update`: Envia o conteúdo do buffer de memória para o display físico, tornando as operações de desenho visíveis.
- `Dp_clear`: Limpa o buffer de memória do ecrã, apagando o conteúdo visual atual.

SSD1306

Fornecem as ferramentas de baixo nível necessárias para controlar um display OLED SSD1306.
Foi utilizado o código de exemplo da BitDogLab para configuração e uso do display OLED [5].

- `ssd1306_init`: Esta função executa a sequência de inicialização do controlador SSD1306 através da interface I2C.
- `ssd1306_draw_string`: Desenha uma string de texto no buffer de memória do display OLED.
- `ssd1306_draw_char`: Desenha um único caractere no buffer de memória do display OLED.
- `ssd1306_clear`: Limpa o buffer de memória do display OLED, preenchendo-o com pixels "apagados".
- `render_on_display`: Transfere o conteúdo do buffer de memória para o display OLED físico, efetivando as operações de desenho no display.
- `calculate_render_area_buffer_length`: Calcula o tamanho em bytes do buffer necessário para uma dada área de renderização no display SSD1306.

3.3 Definição das Variáveis

Serão apresentadas as principais variáveis definidas para cada módulo.

Control

- `Input_t input_buf[INPUT_BUFF_SIZE]`: Guarda os últimos inputs ainda não processados.
  - `Input_t` é um enum com as possíbilades de entrada do usuário.
  - `INPUT_BUFF_SIZE` é uma constante definida no arquivo `Types.h`.
- `struct tm now`: variável constantemente atualizada com o valor atual de tempo.
- `Event event`: struct que guarda as informações do evento sendo utilizado pela tela atual.

  - O tipo `Event` é definido no arquivo `Types.h`. Ele armazena os dados de nome, hora de começo e fim, cor do LED e se é necessário o alarme de um evento.

- `Event dayEvents[MAX_DAY_EVENTS]`: Array armazenando os eventos que ocorrerão no dia atual.
- `uint8_t numDayEvents`: variável que armazena o número de eventos que ocorrerão no dia atual.
- `static Screen screen`: Armazena os dados da tela atual.
- `static Sc_Type screen_type`: Amazena a informação de qual é o tipo de tela atual.
  - Os tipos `Screen` e `Sc_Type` são definidos no arquivo `Types.h`.

Led Matrix

- `PIO np_pio` e `sm;`: Informações do PIO utilizado para o controle da matriz de LEDs.

Events

- `Event events[MAX_EVENT_NUM]`: Array que armazena todos os eventos cadastrados no sistema.
- `EventIndex_t current_event_id`: Posição que o último evento adicionado se encontra no array de eventos.
  - Caso o array fique completamente cheio o index atual irá para o começo do array.
- `EventIndex_t events_num`: Número total de eventos armazenados no array.

Screen

Todas as funções de update das Screen recebem os seguintes argumentos

- `Event *event`: Evento que será manipulado para a utilidade que a tela necessitar.
- `Input_t *input_buf`: Buffer de entradas ainda não processadas.

Display

- `uint8_t ssd[ssd1306_buffer_length]`: Armazena as informações que serão enviadas ao display, principalmente informações de atualização da tela.
- `struct render_area frame_area`: A área de atualização do display.

3.4 Fluxograma

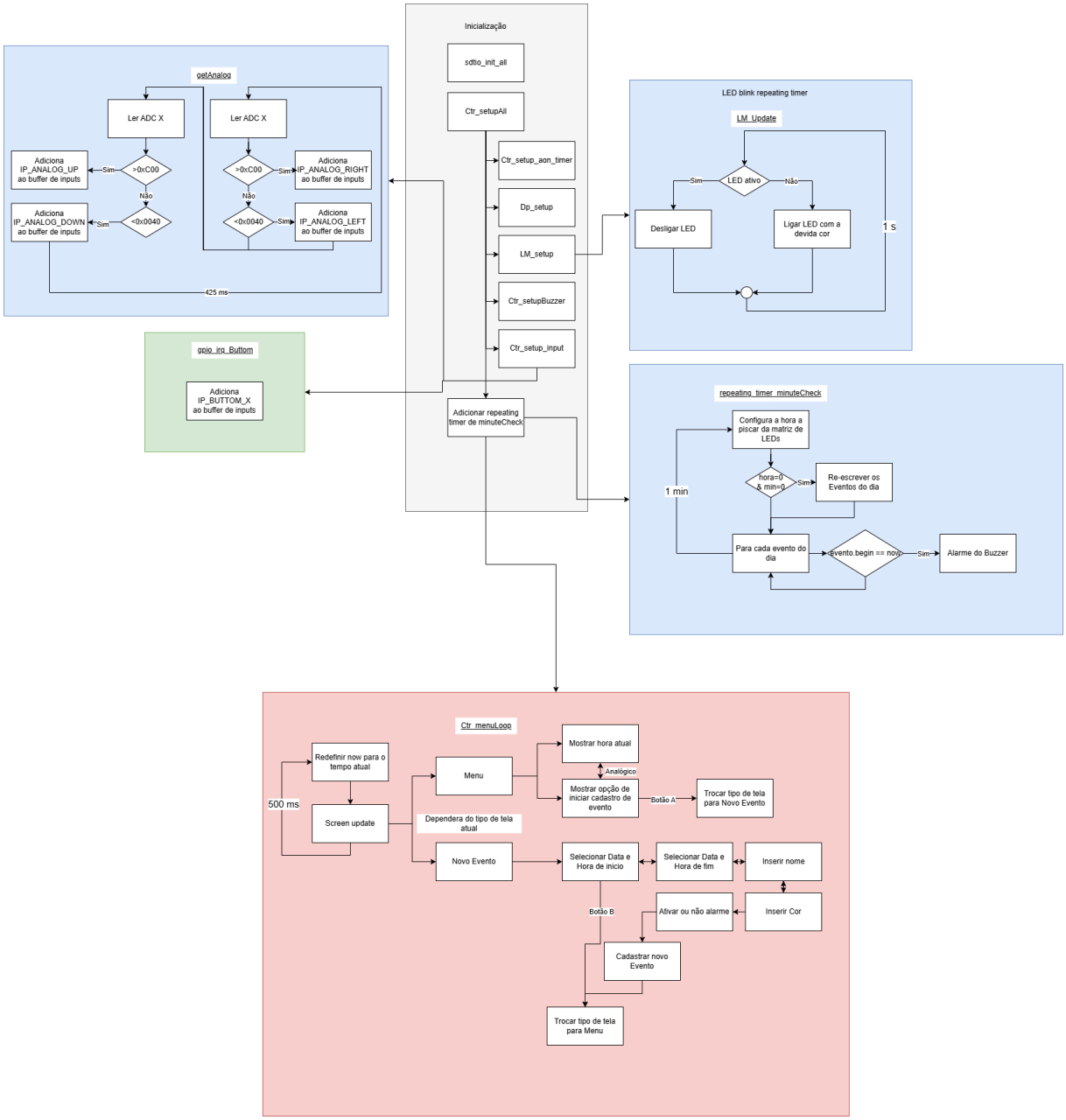

Figura 4: Fluxograma do sistema

3.5 Inicialização

Cada função de inicialização está apresentada a seguir:

- `Ctr_setupAll`: Esta é a função de inicialização mestre. O seu propósito é coordenar e chamar todas as funções de configuração necessárias para preparar o sistema de calendário para o funcionamento.
- `Ctr_setup_aon_timer`: Inicializar o Always-On Timer (AON) do microcontrolador Pico. Este timer é essencial para manter o tempo real (relógio) do calendário, mesmo quando o resto do sistema pode estar em baixo consumo ou em diferentes modos de operação.
  - O valor incial de tempo é igual ao tempo de compilação do projeto.

- `Ctr_setupInput`:Inicializar o sistema de input do utilizador, que neste projeto envolve botões físicos e um joystick analógico.
  - É ativado um repeating timer para, periodicamente, ler a entrada analógica.
  - A informação de pressionamento dos botões é obtida por meio de interrupção.
- `Ctr_setupBuzzer`: Inicializar o buzzer para gerar alertas sonoros.
- `LM_setup`: : Inicializar o módulo da Matriz de LEDs (`LedMatrix`).
- `Dp_setup`: Inicializar o módulo do Display (`Display`), que gere o ecrã OLED SSD1306.
- Por fim inicializa um repeating_timer para, uma vez por minuto, atualizar a matriz de LEDs e observar se há algum evento nesse horário

3.6 Estruturas e formatos de dados

`Color` (Estrutura)

`Color` é uma estrutura que representa uma cor no formato RGB (Vermelho, Verde, Azul).
Formato de Dados:

- Estrutura (`struct`) composta por três membros: `r`, `g` e `b`.
- Cada membro (`r`, `g`, `b`) é do tipo `uint8_t` (inteiro não assinado de 8 bits), representando valores de 0 a 255 para cada componente de cor.

`Event` (Estrutura)

`Event` define a estrutura para representar um evento de calendário, contendo todas as informações necessárias para um evento agendado.
Formato de Dados:

- Estrutura (`struct`) com seis membros de tipos variados:
  - `begin` e `end`: Do tipo `struct tm` (da biblioteca `<time.h>`), para data e hora de início e fim.
  - `name`: Array de `char` de tamanho 32 (`char name[32]`), para o nome do evento (máximo 31 caracteres + null terminator).
  - `color`: Do tipo `Color`, para a cor associada ao evento.
  - `alert`: Do tipo `bool`, indicando se o alarme está ativo para o evento.
  - `index`: Do tipo `uint`, um índice único para identificar o evento.

`Sc_Type` (Enumeração)

`Sc_Type` é uma enumeração que define os diferentes tipos de "ecrãs"ou estados da aplicação de calendário.
Formato de Dados:

- Enumeração (`enum`) com nomes simbólicos representando valores inteiros:
  - `SC_MENU`: Ecrã do menu principal (valor 0).
  - `SC_NEW_EVENT`: Ecrã de criação de novo evento (valor 1).
  - `SC_NEW_EVENT_F`: Ecrã de evento finalizado (valor 2).

`EventIndex_t` **(Tipo Definido - Typedef)**

`EventIndex_t` é um tipo definido como sinónimo de `uint8_t`, usado para representar o índice de um evento.

Formato de Dados:

- Sinónimo de `uint8_t`, um inteiro não assinado de 8 bits (valores 0-255).

5. `Input_t` **(Enumeração)**

`Input_t` é uma enumeração que define os tipos de eventos de entrada do utilizador.

Formato de Dados:

- Enumeração (`enum`) com nomes simbólicos para tipos de input:
  - `IP_NONE`: Ausência de input.
  - `IP_BUTTOM_A`: Botão A pressionado.
  - `IP_BUTTOM_B`: Botão B pressionado.
  - `IP_ANALOG_UP`, `IP_ANALOG_DOWN`, `IP_ANALOG_LEFT`, `IP_ANALOG_RIGHT`: Movimentos do joystick analógico.

`Screen` **(Estrutura)**

`Screen` é uma estrutura que representa uma "tela"ou vista da aplicação, essencial para a arquitetura baseada em telas.

Formato de Dados:

- Estrutura (`struct`) com um único membro:
  - `update`: Ponteiro para função do tipo `Sc_Type (*update)(Event *event, Input_t *input_buf)`.
    - Retorna um valor de `Sc_Type` para indicar o próximo ecrã.
    - Recebe parâmetros: `Event *event` (evento atual) e `Input_t *input_buf` (inputs do utilizador).

3.7 Protocolos de Comunicação

Foi utilizado o protocolo de comunicação I2C para conctar-se ao display e o protoco personalizado da matriz de LEDs para enviar informações a ela.

4 Execução do Projeto

4.1 Metodologia de Desenvolvimento

1. Conceção e Planeamento Inicial

Etapas:

- **Definição de Requisitos:** Clarificar o propósito do calendário, funcionalidades desejadas (ex: mostrar hora, agendar eventos, alarmes), interfaces de utilizador (LED Matrix, Display OLED, botões, joystick), e restrições de hardware.
- **Arquitetura do Sistema:** Esboçar a arquitetura geral do sistema, identificando os módulos principais (Controlo, Display, Eventos, LedMatrix, Screen, Types) e as suas interações.
- **Seleção de Hardware e Software:** Escolher o microcontrolador (Pico RP2040), display OLED (SSD1306), matriz de LEDs, sensores de input, e ferramentas de desenvolvimento (SDK do Pico, compilador, IDE).

2. Desenvolvimento Modular e Implementação

Etapas:

- **Implementação de Módulos Individuais:** Desenvolver cada módulo separadamente (ex: `Display`, `LedMatrix`, `Events`, `Screen`, `Control`). Começar pelos módulos de nível mais baixo (ex: `Display`, `LedMatrix`) e progredir para os de nível mais alto (`Screen`, `Control`).
- **Codificação e Documentação:****Implementar as funcionalidades de cada módulo em C, seguindo as especificações definidas no planeamento. Documentar o código de forma clara e concisa, especialmente nos ficheiros de cabeçalho (`.h`) para descrever as interfaces dos módulos.
- **Testes Unitários:**** Para cada módulo, realizar testes unitários para verificar se as funções e componentes individuais funcionam corretamente (ex: testar funções de desenho do `Display`, funções de gestão de eventos do `Events`).

3. Integração e Testes de Sistema

Após testar os módulos individualmente, é crucial integrá-los e verificar o sistema como um todo.

Etapas:

- **Integração Progressiva:**** Integrar os módulos gradualmente, começando com os módulos base. Por exemplo, integrar `Display` e `LedMatrix` primeiro, depois adicionar `Events`, `Screen` e, finalmente, `Control`.
- **Testes de Integração:** Testar as interações entre os módulos integrados para garantir que comunicam e funcionam em conjunto corretamente.
- **Testes de Sistema e Validação:** Realizar testes de sistema completos para verificar se o calendário cumpre todos os requisitos definidos na fase de conceção. Validar se todas as funcionalidades (ex: agendar evento, exibir hora, alarmes, interação do utilizador) operam como esperado em diferentes cenários de uso.
- **Debugging e Correção de Erros:** Identificar e corrigir erros encontrados durante os testes de integração e sistema. Utilizar ferramentas de debugging para analisar o comportamento do software e hardware.

4. Documentação Final e Manutenção

A documentação é essencial para a compreensão, manutenção e evolução do projeto.

Etapas:

- **Documentação do Projeto Completo:** Compilar a documentação de todas as fases do projeto, incluindo requisitos, arquitetura, design de módulos, código documentado, procedimentos de teste e resultados.
- **Manual do Utilizador (Opcional):** Criar um manual do utilizador simples, se necessário, para explicar como interagir com o calendário.
- **Manutenção e Melhorias Contínuas:** Planear a manutenção do projeto, incluindo a correção de bugs que possam surgir após a implementação e a consideração de futuras melhorias ou funcionalidades adicionais.

4.2 Testes e Validação

Após o desenvolvimento de cada módulo se realizava um teste com a placa para testar seu devido funcionamento, como o alarme de eventos, criação de eventos, update da matriz de LEDs e etc.

4.3 Discussão dos Resultados

O desenvolvimento do sistema de calendário embutido alcançou um progresso significativo, resultando num protótipo funcional que integra diversas funcionalidades essenciais. Através da metodologia de desenvolvimento modular, foi possível construir e documentar cada componente do sistema, desde os tipos de dados fundamentais até à lógica de controlo e interface de utilizador.

Funcionalidades Implementadas e Validadas

- **Visualização da Hora e Data:** O sistema é capaz de manter e exibir a hora e data atuais utilizando o timer Always-On (AON) do microcontrolador Pico RP2040. A informação temporal é apresentada tanto na matriz de LEDs (possivelmente de forma analógica) quanto no display OLED, oferecendo redundância e flexibilidade na visualização.
- **Agendamento e Gestão de Eventos:** Foi implementado um sistema básico de gestão de eventos, permitindo ao utilizador criar e armazenar eventos. Cada evento inclui detalhes como hora de início e fim, nome, cor associada e a opção de configurar um alarme. O armazenamento de eventos, embora limitado no protótipo atual, demonstra a capacidade do sistema em lidar com dados de calendário.
- **Interface de Utilizador Gráfica (GUI):** O display OLED serve como a principal interface de utilizador, permitindo apresentar menus, informações de eventos, e guiar o utilizador através do processo de criação e edição de eventos. A navegação nos menus e a interação com a interface são possibilitadas pelos botões físicos e pelo joystick analógico.
- **Feedback Audível e Visual:** O buzzer foi integrado para fornecer alertas sonoros, nomeadamente para alarmes de eventos. A matriz de LEDs, para além da indicação da hora, pode ser utilizada para fornecer feedback visual adicional, como destacar eventos.
- **Arquitetura Modular e Documentada:** O projeto foi estruturado em módulos bem definidos (Types, Control, LedMatrix, Events, Display, Screen), facilitando a compreensão, manutenção e expansão do código. A documentação detalhada, especialmente nos ficheiros de cabeçalho (`.h`), torna o projeto mais acessível e colaborativo.

Limitações e Oportunidades de Melhoria

Embora o protótipo demonstre as funcionalidades base de um calendário, existem áreas com potencial para refinamento e expansão:

- **Persistência de Dados:** Atualmente, os eventos são provavelmente armazenados em memória volátil (RAM). A adição de memória não volátil (ex: Flash interna ou externa) seria crucial para garantir que os eventos agendados sejam preservados mesmo após o sistema ser desligado ou reiniciado.
- **Interface de Utilizador Avançada:** A interface do utilizador no display OLED pode ser aprimorada para ser mais intuitiva e rica em funcionalidades. Isto pode incluir layouts de menu mais elaborados, melhor formatação da informação de eventos, e talvez até elementos gráficos mais complexos.

- **Funcionalidades de Alarme Avançadas:** O sistema de alarme atual pode ser básico (um simples "beep"). Melhorias poderiam incluir alarmes recorrentes, diferentes tipos de sons de alarme, e opções de "snooze".
- **Sincronização de Tempo Mais Precisa:** O timer AON, embora útil para manter o tempo de forma autónoma, pode acumular deriva ao longo do tempo. Mecanismos de sincronização de tempo mais precisos, como NTP (Network Time Protocol) através de uma interface WiFi, poderiam ser adicionados.

Perspectivas Futuras e Funcionalidades Adicionais

Para além do refinamento das funcionalidades existentes, o projeto pode ser significativamente expandido através da incorporação de novas capacidades, nomeadamente focando na economia de energia e na conectividade WiFi.

4.4 Vídeo

`https://youtube.com/shorts/Yz97WCfgbEI?feature=share`

Referências

[1] **Raspberry Pi Pico - Alarm Clock Project**, from the NerdCave YouTube channel. `https://youtu.be/EOMcPAKL6RM?si=pNSk2qMvinZMr-He`

[2] **How to build a Wall Mounted Family Calendar and Dashboard with a Raspberry Pi and cheap monitor**, por Scott Hanselman. `https://www.hanselman.com/blog/how-to-build-a-wall-mounted-family-calendar-and-dashboard-with-a-raspberry-pi-and-cheap`

[3] **Workday Progressbar With Google Calendar Integration**, por danionescu. `https://www.instructables.com/Workday-Progressbar-With-Google-Calendar-Integrati/`

[4] **BitDogLab-C neopixel_pio**. `https://github.com/BitDogLab/BitDogLab-C/tree/main/neopixel_pio`

[5] **BitDogLab-C display_oled**. `https://github.com/BitDogLab/BitDogLab-C/tree/main/display_oled`