EMBARCATECH

RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS

OMNI-CLOCK

(Sistema Embarcado de Relógio Universal para Fusos Horários Sem Conectividade)

ANDREI LUIZ DA SILVA RODRIGUES

Matrícula: TIC370100444

JUAZEIRO - BA

FEVEREIRO– 2025

ÍNDICE

REPOSITÓRIO

https://github.com/andreirodriguess/Omni_clock

Omni Clock

a) ESCOPO

Apresentação

O **Omni-Clock** é um projeto de sistema embarcado desenvolvido para funcionar como um relógio universal, permitindo ao usuário sincronizar o horário conforme qualquer fuso horário desejado. Sua principal característica é a capacidade de fornecer a hora local de qualquer região do mundo sem depender de conexão com a internet.

Objetivo

O objetivo deste projeto é garantir que a hora de qualquer localidade esteja acessível independentemente do acesso à rede. Dessa forma, possibilita a sincronização de dispositivos em diferentes regiões do mundo, mesmo em cenários onde a conectividade à internet é limitada ou inexistente.

Descrição do Funcionamento

O sistema foi projetado para fornecer informações sobre horários em diferentes fusos horários, bem como permitir interações dinâmicas com o usuário. O funcionamento é dividido em etapas, conforme descrito a seguir:

1. **Recepção do Horário Atual**: O sistema inicialmente obtém o horário atual por meio do monitor serial, garantindo que a base de tempo esteja atualizada e precisa.
2. **Seleção do Fuso Horário**: O usuário tem a opção de selecionar o fuso horário correspondente à sua localização geográfica. Isso é feito de maneira simples e intuitiva, por meio de uma interface no monitor serial.
3. **Consulta de Outros Fusos Horários**: Uma vez selecionado o fuso horário inicial, o usuário pode consultar e visualizar o horário de outros fusos horários. Isso permite comparações diretas entre diferentes regiões do globo, facilitando o entendimento das diferenças de horário ao redor do mundo.
4. **Interatividade com o Joystick**: O sistema é equipado com um joystick que permite ao usuário selecionar diferentes pontos no globo. Conforme o joystick é movido, o LED RGB do sistema muda de cor, refletindo a proximidade do ponto selecionado em relação à linha do equador ou aos polos. O LED fica com um brilho **vermelho** mais forte quando o ponto se aproxima da linha do equador e com um brilho **azul** mais forte à medida que o ponto se aproxima dos pólos. Essa mudança de cor adiciona uma camada de interatividade visual ao sistema, tornando a experiência mais envolvente.

Justificativa

A implementação deste projeto se justifica pela necessidade de soluções robustas em sistemas embarcados que operem de maneira eficiente mesmo em condições adversas, como a ausência de conectividade à internet. A possibilidade de visualizar a hora de diferentes regiões pode ter aplicações relevantes no mundo real, como a sincronização de processos globais que dependem de horários precisos. Setores como logística, telecomunicações e geração de energia solar, que apresenta variações de duração conforme a localização geográfica, podem se beneficiar dessa funcionalidade.

Além disso, o **Omni-Clock** pode ser utilizado como uma ferramenta educacional para estudantes, proporcionando uma abordagem interativa ao estudo dos fusos horários. Esse método dinâmico pode aumentar o interesse dos alunos e facilitar a compreensão desse conceito geográfico essencial.

Outro aspecto relevante é a sua aplicação para viajantes internacionais, que podem precisar ajustar rapidamente seus relógios ao fuso horário do destino sem depender de redes móveis ou internet. Dessa forma, o **Omni-Clock** oferece uma solução prática e eficiente para aqueles que transitam entre diferentes fusos horários com frequência.

Originalidade

O projeto **Omni-Clock** apresenta um conceito inovador ao funcionar como um relógio embarcado que se adapta dinamicamente ao fuso horário desejado, permitindo a sincronização com múltiplos relógios ao redor do mundo sem necessidade de conectividade externa. Embora existam ideias correlatas, não há um projeto idêntico a este.

Exemplos de projetos correlatos:

- **Relógio do Clima**: Um cronômetro que informa o tempo restante para se evitar o pior da crise climática. Criado pelos artistas Gan Golan e Andrew Boyd, teve suas primeiras versões instaladas em Berlim (2019), Nova York (2020) e Seul (2021). Diferente do **Omni-Clock**, este projeto não possui funcionalidades de ajuste dinâmico de fuso horário. (Fonte)

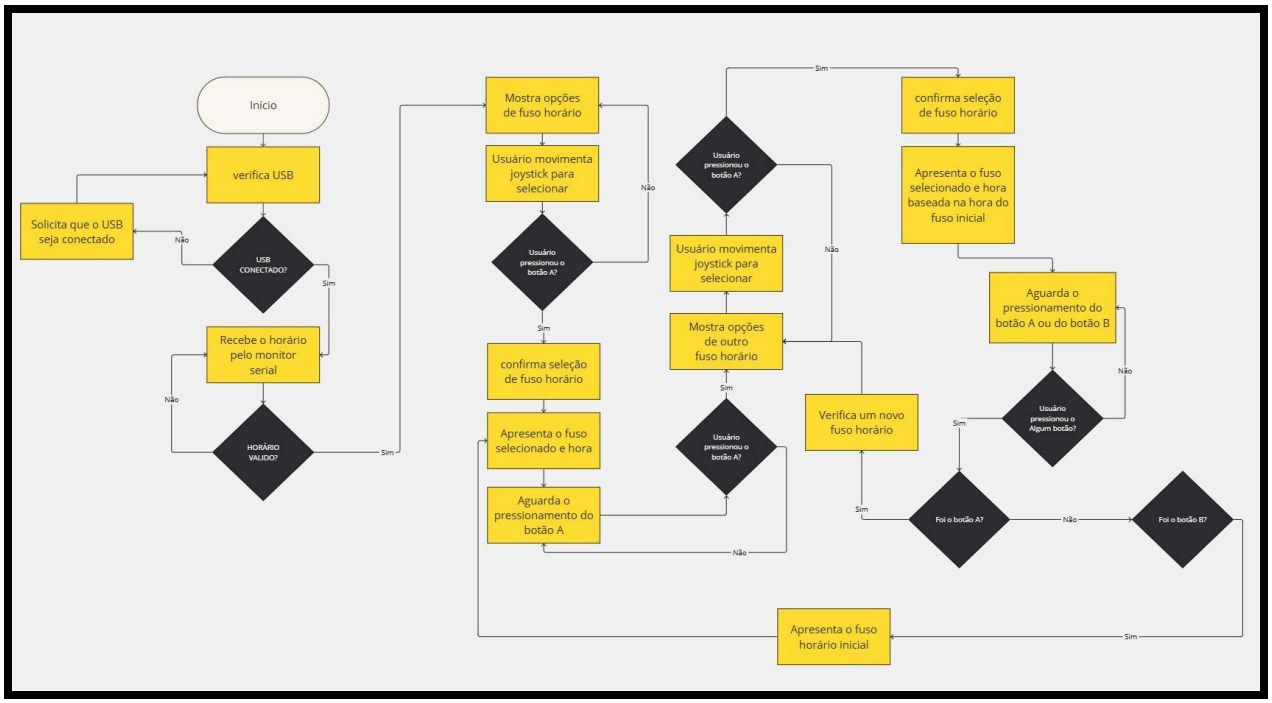
- **Relógios GMT**: Tradicionalmente utilizados por viajantes, os relógios GMT apresentam um ponteiro adicional de 24 horas que permite visualizar um segundo fuso horário. Alguns modelos possuem uma luneta rotativa com uma escala de 24 horas, possibilitando a exibição simultânea de dois fusos horários. No entanto, esses relógios não possuem a capacidade de ajuste automático de

fuso, como o **Omni-Clock**. (Fonte)

Portanto, o **Omni-Clock** se diferencia por sua capacidade de adaptação automática a qualquer fuso horário e sua abordagem digital interativa, unindo elementos de precisão, praticidade e acessibilidade sem depender de conexão com a internet.

b) Especificação do hardware

Diagrama em bloco



Descrição dos Blocos do Sistema

1. **Início (Bloco Inicial)**

   O sistema inicia com a configuração dos periféricos necessários, garantindo que todos os recursos essenciais estejam prontos para o funcionamento. Utilizando a função i2c_adc_gpio_init().

2. **Verificação de Conexão USB (Bloco de Processo – Retângulo)**

   O sistema realiza a verificação da presença de um dispositivo USB conectado. Para isso, utiliza-se a função `stdio_usb_connected()`, que determina se a conexão está estabelecida corretamente.

3. **Solicitação de Conexão USB (Bloco de Processo – Retângulo)**

   Caso não seja detectado nenhum dispositivo USB, o sistema solicita ao usuário que conecte um dispositivo. Para exibir essa mensagem, os pinos GPIO 14 (SDA) e GPIO 15 (SCL) são usados para se comunicar com o display OLED SSD1306, com a interface de comunicação I2C, utilizando funções gráficas já pré-definidas.

4. **Exibição de Opções de Fuso Horário (Bloco de Processo – Retângulo)**

   O sistema exibe um mapa mundial no display OLED, permitindo ao usuário escolher um fuso horário. Para renderizar o mapa, o sistema faz uso da função `ssd1306_draw_world_map(ssd1306_t *ssd, const uint8_t *bitmap)`, utilizando novamente os pinos GPIO 14 e GPIO 15.

5. **Seleção do Fuso Horário pelo Usuário (Bloco de Processo Retângulo)**

   O usuário interage com o joystick para selecionar o fuso horário desejado. A posição da cruz no mapa, que indica a seleção, é gerada com base nas entradas do joystick. A posição (X, Y) da cruz é definida pelos valores lidos pelo ADC do joystick, que são convertidos para as coordenadas do display através das funções

`map_x_to_display(uint16_t input_x)` e `map_y_to_display(uint16_t input_y)`.

6. **Confirmação da Seleção de Fuso Horário (Bloco de Processo – Retângulo)**

   Quando o usuário confirma sua seleção pressionando o botão A, o sistema registra a escolha e altera a interface do display para refletir o fuso horário selecionado. Para detectar a interação com o botão A, o GPIO 5 é configurado como uma entrada com pull-up, e uma interrupção é configurada para detectar a borda de descida do botão. A interrupção é tratada pela rotina de callback `gpio_irq_handler(uint gpio, uint32_t events)`, que altera a variável `tela`, permitindo a modificação da interface do display para a seleção do fuso horário atual.

7. **Apresentação do Fuso Selecionado e Hora Baseada no Fuso Inicial**

   O sistema exibe o fuso horário selecionado e a hora correspondente, calculada a partir de um fuso horário inicial de referência. Para isso, são utilizadas variáveis que armazenam a hora e o fuso horário atual. O display OLED é atualizado com essas informações utilizando a função `ssd1306_draw_string(ssd1306_t *ssd, const char *str, uint8_t x, uint8_t y)`, onde os pinos GPIO 14 (SDA) e GPIO 15 (SCL) são configurados para comunicação com o display.

8. **Recebimento do Horário via Monitor Serial**

   O sistema recebe informações de horário através da comunicação serial. Para iniciar essa comunicação, a função `stdio_init_all()` é utilizada, permitindo a troca de dados via USB com o monitor serial. Os caracteres referentes à hora são recebidos no formato `HH MM`, sendo que `HH` representa a hora (dois caracteres) e `MM` os minutos (dois caracteres).

9. **Apresentação do Fuso Horário e Hora Atualizada**

   Após o recebimento do horário, o sistema exibe o fuso horário selecionado e a hora atualizada. Essas informações são apresentadas no display OLED utilizando a função `ssd1306_draw_string(ssd1306_t *ssd, const char *str, uint8_t x, uint8_t y)`, com as variáveis que armazenam o fuso horário e a hora calculada.

10. **Exibição de Opções de Outro Fuso Horário**

    O sistema exibe novamente o mapa mundial no display OLED, permitindo ao usuário selecionar um novo fuso horário. A interação é realizada por meio do joystick, e a posição da cruz no mapa é gerada com base nas entradas do joystick. A posição (X, Y) da cruz é calculada a partir das leituras do ADC do joystick, que são convertidas em coordenadas de tela através das funções `map_x_to_display(uint16_t input_x)` e `map_y_to_display(uint16_t input_y)`.

11. **Exibição do Fuso Horário Inicial**

    O sistema exibe o fuso horário de referência inicial no display OLED, com a

mesma metodologia utilizada para exibir o fuso horário e a hora atualizada. A configuração dos pinos GPIO 14 e GPIO 15 continua sendo usada para a comunicação com o display, e a função `ssd1306_draw_string()` é aplicada para mostrar o fuso horário inicial.

Blocos de Decisão (Losangos):

- **USB CONECTADO?:** Verifica se o dispositivo USB está conectado. Se sim, o fluxo segue o caminho "Sim"; caso contrário, segue o caminho "Não".

  O sistema realiza a verificação da presença de um dispositivo USB conectado. Para isso, utiliza-se a função `stdio_usb_connected()`, que determina se a conexão está estabelecida corretamente.

- **Usuário pressionou o botão A?:** Verifica se o usuário pressionou o botão A.

  Para detectar a interação com o botão A, o GPIO 5 é configurado como uma entrada com pull-up, e uma interrupção é configurada para detectar a borda de descida do botão. A interrupção é tratada pela rotina de callback `gpio_irq_handler(uint gpio, uint32_t events)`, que altera a variável `tela`, permitindo a modificação da interface do display para a seleção do fuso horário atual.

- **Usuário pressionou o botão B?:** Verifica se o usuário pressionou o botão B.

Para detectar a interação com o botão B, o GPIO 6 é configurado como uma entrada com pull-up, e uma interrupção é configurada para detectar a borda de descida do botão. A interrupção é tratada pela rotina de callback `gpio_irq_handler(uint gpio, uint32_t events)`, que altera a variável `tela`, permitindo a modificação da interface do display para o retorno à tela de apresentação das informações iniciais.

- **HORÁRIO VÁLIDO?:** Verifica se a hora recebida via monitor serial é válida.

  Recebe caracteres por meio da função getchar(); e armazena em variáveis, fazendo uma verificação se os caracteres recebidos são numéricos. Em seguida converte os caracteres para inteiro com a função atoi() e verifica se o valor das horas é superior a 24 e se o dos minutos é superior a 60.

- **Usuário pressionou algum botão?:** Verifica se o usuário interagiu com algum botão.

  usa a função: gpio_set_irq_enabled_with callback() em todos os botões disponíveis e estados em pull up, assim, quando houver alguma borda de descida a função callback irá ver por meio de estruturas condicionais internas, qual foi o pino GPIO que gerou a interrupção.

Descrição da Pinagem Utilizada

A seguir, são apresentados os detalhes da configuração dos pinos GPIO utilizados no sistema:

Entradas Digitais

- GPIO 5 – Botão A (Entrada GPIO)
- GPIO 6 – Botão B (Entrada GPIO)
- GPIO 22 – Push Button do Joystick (Entrada GPIO)

Saídas PWM para LEDs

- GPIO 13 – LED Verde (Saída PWM)
- GPIO 12 – LED Azul (Saída PWM)
- GPIO 13 – LED Vermelho (Saída PWM)

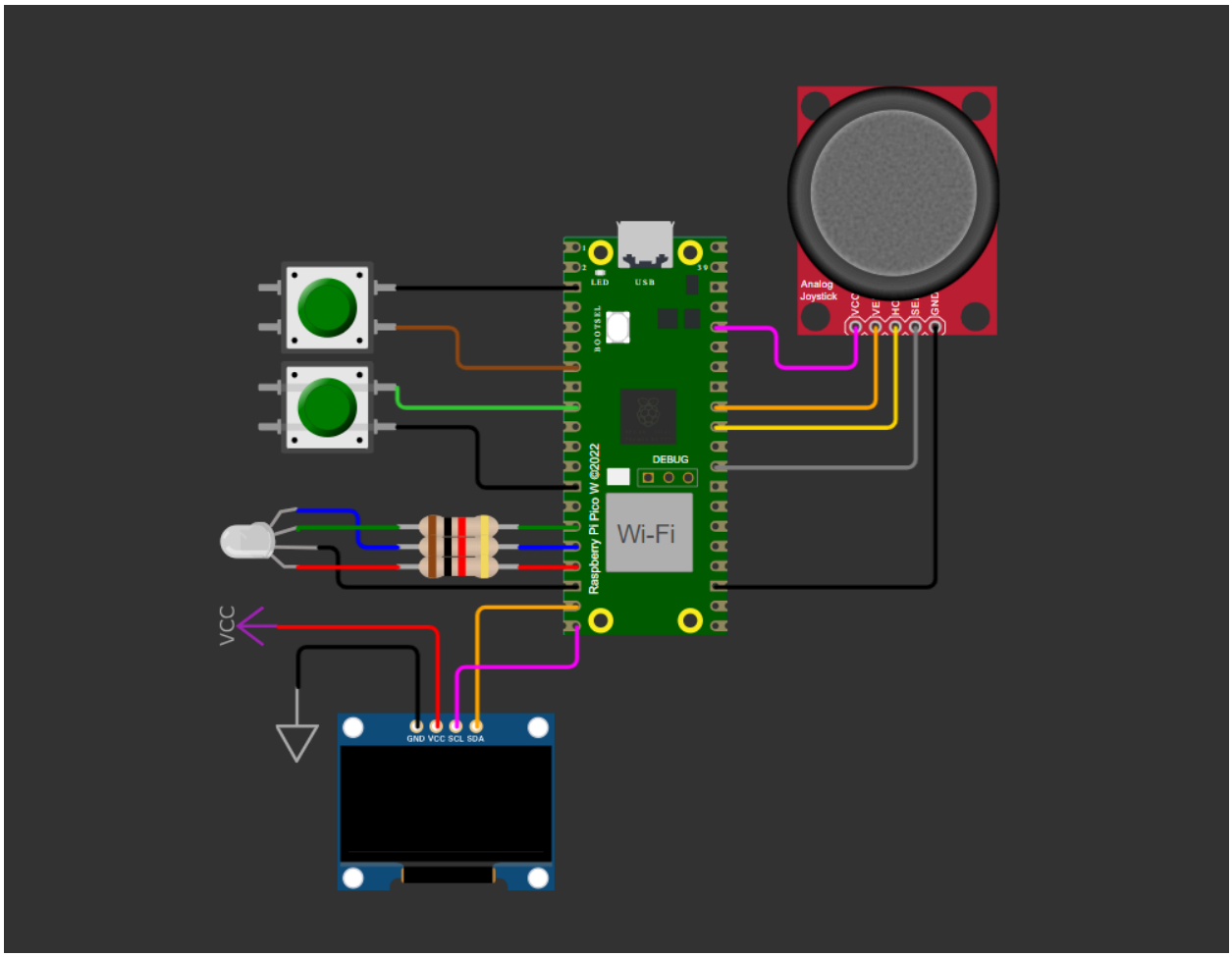
Comunicação I²C com Display OLED

- GPIO 14 – Linha de Dados I²C (SDA)
- GPIO 15 – Linha de Clock I²C (SCL)
- Endereço do Display OLED – `0x3C`

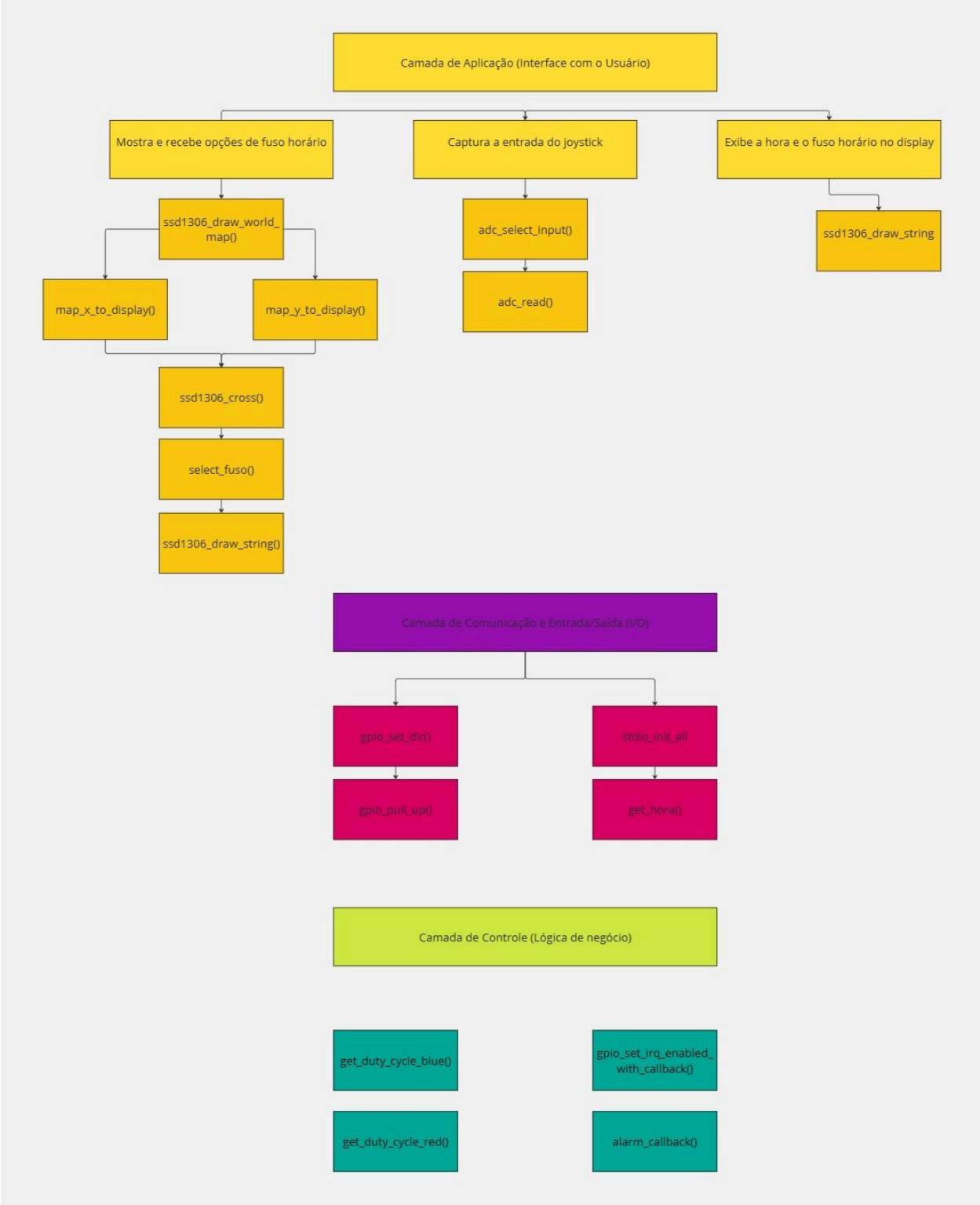
Entradas Analógicas para Joystick

- GPIO 27 – Pino VRX do Joystick (Leitura de eixo X – ADC)
- GPIO 28 – Pino VRY do Joystick (Leitura de eixo Y – ADC)

Circuito completo do hardware



c) Especificação do firmware



- **Descrição das funcionalidades**

Camada de Aplicação (Interface com o Usuário)

1. **Mostra e recebe opções de fuso horário:**

    - `ssd1306_draw_world_map()`: Desenha um mapa mundial na tela SSD1306 para auxiliar na seleção do fuso horário.
    - `map_x_to_display()`: Mapeia a coordenada X do joystick para a posição correspondente no display.
    - `map_y_to_display()`: Mapeia a coordenada Y do joystick para a posição correspondente no display.
    - `ssd1306_cross()`: Desenha uma cruz no display SSD1306 para indicar a posição selecionada.
    - `select_fuso()`: Permite ao usuário selecionar um fuso horário usando o joystick.
    - `ssd1306_draw_string()`: Desenha uma string (texto) no display SSD1306, provavelmente para exibir as opções de fuso horário.

2. Captura a entrada do joystick:

    - `adc_select_input()`: Seleciona a entrada analógica do conversor analógico-digital (ADC) conectada ao joystick.

- `adc_read()`: Lê o valor analógico da entrada do joystick usando o ADC.

3. Exibe a hora e o fuso horário no display:

   - `ssd1306_draw_string()`: Desenha a hora e o fuso horário no display SSD1306.

Camada de Controle (Lógica de Negócio)

1. get_duty_cycle_blue(): Obtém o ciclo de trabalho (duty cycle) para o controle da cor azul, provavelmente de um LED RGB.
2. get_duty_cycle_red(): Obtém o ciclo de trabalho para o controle da cor vermelha, provavelmente de um LED RGB.
3. gpio_set_irq_enabled_with_callback(): Configura uma interrupção (IRQ) em um pino GPIO (General Purpose Input/Output) e associa uma função de callback a essa interrupção.
4. alarm_callback(): Função de callback que é executada quando o alarme é disparado.

Definição das Variáveis

As seguintes variáveis são utilizadas no sistema para controle de hardware, manipulação de dados e configuração geral do programa.

1. Variáveis de Configuração e Controle

- `last_print_time` – Controla o tempo para impressões periódicas (debug).
- `last_button_b_time` – Controla o debounce do botão B (joystick).
- `last_button_a_time` – Controla o debounce do botão A.
- `current_time` – Armazena o tempo atual para controle de eventos.
- `last_frame` – Controla a taxa de atualização do display.
- `fuso_atual` – Armazena o fuso horário atual selecionado.
- `fuso_externo` – Armazena o fuso horário externo selecionado.
- `ssd` – Estrutura para controle do display SSD1306.
- `tela` – Controla qual tela/função está ativa no programa.
- `alarm` – Controla o alarme de tempo.

2. Variáveis de Tempo

- `hora_num2` – Armazena a hora atual.
- `minutos_num2` – Armazena os minutos atuais.
- `hora_char2` – Representação em string da hora atual.
- `minutos_char2` – Representação em string dos minutos atuais.
- `hora_num3` – Armazena a hora do fuso horário externo.
- `hora_char3` – Representação em string da hora do fuso horário externo.
- `buffer` – Buffer de string para formatação de texto.
- `hora_buffer` – Buffer de string para formatação da hora.

3. Variáveis de Hardware

3.1. Controle de LEDs (PWM)

- `RED_PIN` – Pino GPIO para controle do LED vermelho.
- `BLUE_PIN` – Pino GPIO para controle do LED azul.
- `GREEN_PIN` – Pino GPIO para controle do LED verde.
- `MAX_DUTY` – Valor máximo do ciclo de trabalho PWM para controle de brilho dos LEDs.
- `pwm_wrap` – Valor de wrap para o PWM.
- `pwm_slice_red` – Número do slice PWM para o LED vermelho.
- `pwm_slice_blue` – Número do slice PWM para o LED azul.

3.2. Controle do Joystick

- `VRX_PIN` – Pino GPIO para leitura da posição horizontal (X) do joystick.
- `VRY_PIN` – Pino GPIO para leitura da posição vertical (Y) do joystick.
- `PB` – Pino GPIO para leitura do botão de pressão do joystick.

3.3. Controle de Botões

- `button_a` – Pino GPIO para o botão A.
- `button_b` – Pino GPIO para o botão B.

3.4. Comunicação I²C com Display OLED

- `I2C_PORT` – Definição da porta I²C utilizada.
- `I2C_SDA` – Pino GPIO utilizado para a linha de dados I²C (SDA).
- `I2C_SCL` – Pino GPIO utilizado para a linha de clock I²C (SCL).
- `endereco` – Endereço I²C do display OLED (`0x3C`).

• Protocolo de comunicação

Fez-se uso dos seguintes protocolos de comunicação:

I2C para comunicação com o display.

d) Execução do projeto

Pesquisas Realizadas

Para a concepção e implementação do projeto, foram conduzidas pesquisas aprofundadas sobre:

- **Divisão da Terra em fusos horários**, garantindo a correta conversão entre horários de diferentes regiões.
- **Renderização de imagens em displays OLED**, permitindo a exibição do mapa-múndi no display SSD1306.
- **Bibliotecas da Raspberry Pi Pico**, incluindo funcionalidades essenciais como PWM, ADC, temporizadores e interrupções, fundamentais para a interação do sistema com os periféricos.

Escolha do BITDOGLAB

A escolha dos componentes foi baseada na compatibilidade com o microcontrolador **Raspberry Pi Pico** e nas necessidades do projeto:

- **Display OLED SSD1306 (128x64 pixels)** – Para exibição gráfica das informações de horário e mapa-múndi.
- **Joystick analógico** – Para permitir a navegação e seleção de fusos horários.

- **Botões físicos (GPIO)** – Para confirmar seleções e alternar entre telas do sistema.

Definição das Funcionalidades do Software

O software foi projetado para cumprir os seguintes requisitos:

1. Captura da hora inicial via monitor serial.
2. Seleção do fuso horário utilizando o joystick.
3. Cálculo automático da conversão de horário entre fusos.
4. Exibição da hora ajustada no display OLED.
5. Navegação entre telas utilizando botões físicos.

Inicialização e Programação na IDE

O desenvolvimento do firmware foi realizado na **IDE VS Code**, utilizando o **SDK da Raspberry Pi Pico**. O código foi escrito em **C**, com a integração de bibliotecas especializadas para controle de hardware.

Depuração

Diversas técnicas de depuração foram aplicadas para garantir a confiabilidade do sistema:

- **Depuração por `printf`**: Impressão de variáveis no monitor serial para identificar falhas no fluxo do programa e corrigir comportamentos inesperados.

- **Apresentação no Display**: Devido a limitações de hardware da **BitDogLab**, ajustes no software foram necessários para compensar imprecisões, garantindo um funcionamento adequado.
- **Testes de Entrada**: Implementação de verificações para impedir entradas inválidas de horas e minutos, garantindo a robustez da interface com o usuário.
- **Testes de Fusos Horários**: Foram aplicados diversos fusos como referência inicial, permitindo validar a precisão dos cálculos internos e garantir a exibição correta dos horários ajustados.

•Resultados

O **Omni-Clock** foi submetido a múltiplos testes em diferentes cenários, demonstrando sua funcionalidade e capacidade de adaptação a diversos fusos horários. No entanto, algumas limitações foram identificadas, principalmente relacionadas à precisão na seleção do fuso, devido à resolução restrita do display OLED (128x64 pixels) e à sensibilidade limitada do joystick.

Apesar dessas restrições, o projeto foi devidamente depurado e está pronto para uso em aplicações futuras. Sua arquitetura modular permite sua integração em outros sistemas embarcados que necessitem de um relógio adaptável a diferentes fusos horários.

O **Omni-Clock** exemplifica o grande potencial dos sistemas embarcados na automação e simplificação de processos cotidianos, proporcionando maior praticidade para

usuários em qualquer lugar do mundo, independentemente do acesso à internet. Esse projeto destaca a versatilidade dessas tecnologias e abre caminho para futuras inovações na área.

e) Referências

1. https://brasilescola.uol.com.br/geografia/fuso-horario.htm
2. https://www.timeanddate.com
3. **[1] BitDogLab. Manual BitDog Lab, Disponível em:** https://github.com/BitDogLab/BitDogLab/tree/main/doc **Acesso em: 24 fev. 2025.**
4. **[2] RASPBERRY PI LTD. Raspberry Pi Pico W Datasheet: An RP2040-based microcontroller board with wireless. 2025. Disponível em:** https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf**. Acesso em: 24 fev. 2025.**
5. **[3] RASPBERRY PI LTD. RP2040 Datasheet. 2025. Disponível em:** https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf**. Acesso em: 24 fev. 2025.**
6. **[4]RASPBERRY PILTD. Raspberry Pi Pico C/C++SDK. 2025. Disponível em:**

https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf

. Acesso em: 24 fev. 2025.

REPOSITÓRIO

https://github.com/andreirodriguess/Omni_clock

LINK DO VÍDEO DE APRESENTAÇÃO

https://youtu.be/syjVBWtrnEY