

Projeto Final de Sistema Embarcado para o EmbarcaTech

Discente: Gabriel Cavalcanti Coelho

Matrícula: TIC370101612

Escopo do Projeto

Título

Sistema de Controle para Conforto Térmico

Apresentação do Projeto

A cidade onde resido, Petrolina-PE, é conhecida por seu clima quente e seco. Durante as manhãs e noites, a temperatura é agradável, mas, à tarde, o calor se torna bem forte, com um ar seco que pode causar sangramentos no nariz de quem não está acostumado. Entre julho e agosto, o clima é mais ameno e confortável, enquanto, entre dezembro e janeiro, as temperaturas são bem altas.

Diante disso, idealizei e desenvolvi este projeto, um sistema de monitoramento de temperatura e umidade para controlar automaticamente o ventilador e o umidificador de um climatizador. Com base nas leituras dos sensores e nos parâmetros definidos pelo usuário, o sistema ajusta a velocidade do ventilador e aciona o umidificador conforme necessário. A interface exibe em tempo real as informações coletadas e o status dos dispositivos, incluindo um rosto que indica o nível de conforto térmico. Além disso, uma matriz de LEDs sinaliza o funcionamento do sistema e alerta sobre a necessidade de reabastecimento do umidificador. Tudo isso, pensando em proporcionar um ambiente mais confortável e agradável, promovendo bem-estar e melhorando a disposição das pessoas.

Objetivos do projeto

1. Monitorar continuamente a temperatura e a umidade do ambiente;
2. Controlar automaticamente a velocidade do ventilador em três níveis conforme a temperatura;
3. Acionar o umidificador quando a umidade estiver abaixo de um valor crítico;
4. Alertar o usuário sobre a falta de água no umidificador por meio de indicadores visuais e sonoros;
5. Exibir no display os valores medidos e o estado do sistema, incluindo um ícone facial para representar o nível de conforto térmico;

6. Permitir a configuração personalizada dos limites de temperatura e de umidade.

Descrição do funcionamento

O sistema utiliza o joystick da BitDogLab para simular sensores analógicos, o eixo X representa o sensor de umidade e o eixo Y representa o sensor de temperatura. O LED RGB representa o estado dos dispositivos, com o LED vermelho simulando as velocidades do ventilador por meio de PWM e o LED azul indicando o funcionamento do umidificador. O display exibe a temperatura, umidade, estado do ventilador e do umidificador, além de um rosto que muda a sua expressão conforme as condições simuladas. A matriz de LEDs indica a tela atual, que pode ser alternada pelo usuário entre a tela principal, a configuração das temperaturas, a configuração da umidade e a calibração do joystick, além de alertar sobre a necessidade de reabastecimento do umidificador.

O sistema conta com o botão do joystick para alternar entre as telas, o botão A para interagir com as telas de configurações e o botão B para simular o estado do sensor de nível do umidificador, que indica se está com pouca água ou não.

Justificativa

O controle adequado da temperatura e da umidade é essencial para o conforto e a saúde em diversos ambientes, especialmente em locais secos ou quentes. Sistemas automatizados de climatização contribuem para o bem-estar e a produtividade dos usuários, além de otimizar o consumo energético ao operar apenas quando necessário.

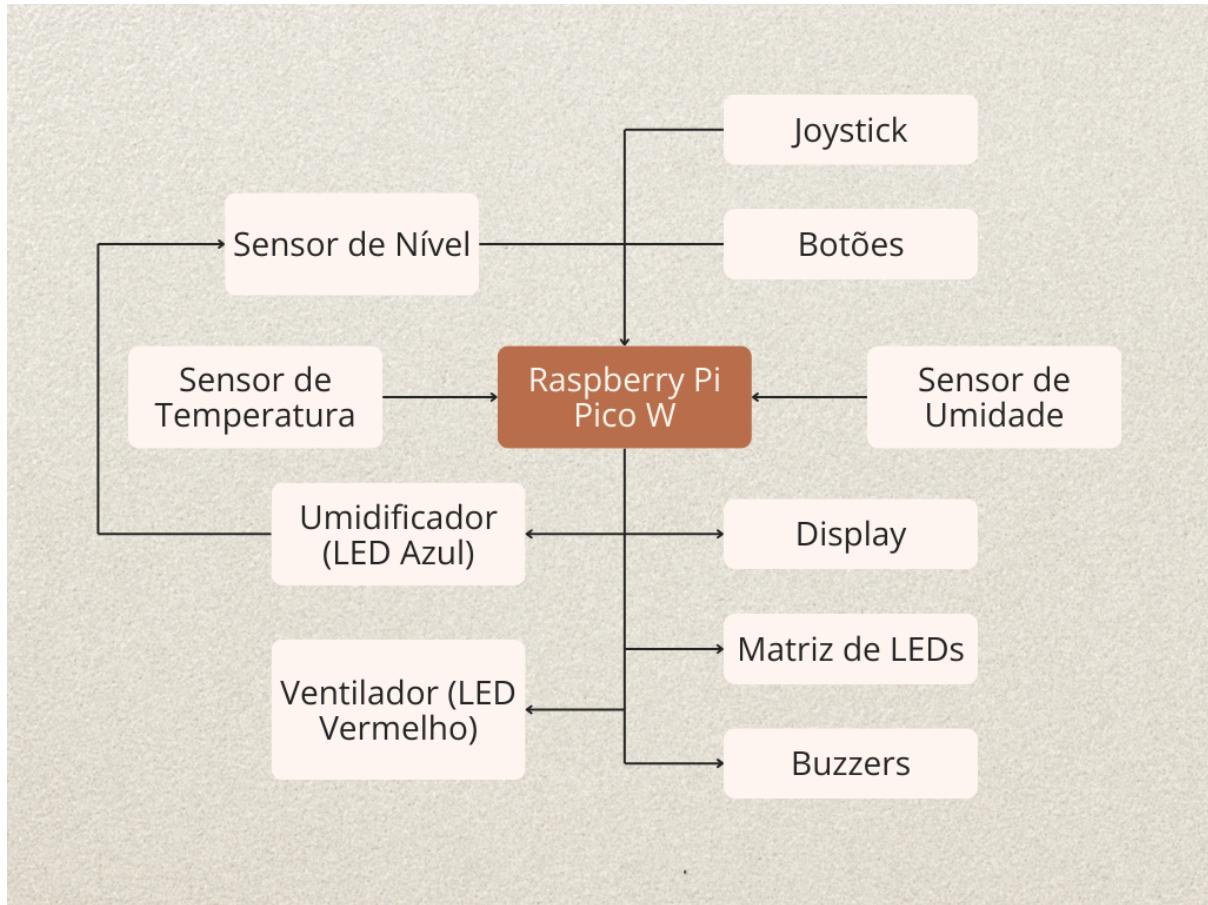
Este projeto simula um sistema desse tipo, também permitindo a validação dos conceitos estudados como PWM, comunicação I2C, controle de GPIOs, PIO e conversor analógico-digital (ADC).

Originalidade

Embora existam diversos projetos correlatos que utilizam sensores de temperatura e umidade, a maioria deles se limita a exibir os valores coletados ou a realizar ações simples de controle. O diferencial que pensei para este projeto está na sua interface interativa, que permite ao usuário personalizar os limites de temperatura e umidade de acordo com as suas preferências, além de monitorar visualmente o status do sistema por meio de ícones faciais e alertas visuais e sonoros, tornando a experiência do usuário mais dinâmica e agradável.

Especificação do Hardware

Diagrama em Bloco



Função dos Blocos

- **Raspberry Pi Pico W:** Faz as leituras dos sensores e entradas do usuário, processa as informações e controla os atuadores conforme as configurações;
- **Sensores de Temperatura e Umidade:** Captam os dados ambientais e enviam ao microcontrolador;
- **Ventilador:** Acionado de forma a variar a sua velocidade conforme a temperatura;
- **Umidificador:** Ligado quando a umidade estiver abaixo do valor crítico definido;
- **Sensor de Nível:** Detecta quando a água do umidificador estiver baixa e envia sinal ao microcontrolador para ativação dos alertas.
- **Display:** Exibe informações sobre temperatura, umidade e estado do sistema;
- **Matriz de LEDs:** Indica status e alertas visuais;
- **Buzzers:** Emite alerta sonoro quando necessário;
- **Joystick:** Utilizado para alterar as telas e configurar os valores de temperatura e umidade desejados;

- **Botão:** Permite interagir com as telas do sistema.

Configuração dos Blocos

- **Raspberry Pi Pico W:** Programado para monitoramento contínuo dos sensores e atualização contínua dos atuadores e da interface com o usuário;
- **Sensores:** Configurados como entradas analógicas no microcontrolador;
- **Ventilador:** Acionado via PWM pelo microcontrolador;
- **Umidificador:** Acionado via saída digital do microcontrolador (O LED azul foi acionado via PWM para gerar uma luz mais fraca);
- **Sensor de Nível:** Entrada digital;
- **Display:** Comunicação via I2C;
- **Matriz de LEDs:** Acionada através da PIO do microcontrolador;
- **Buzzers:** Acionados via PWM pelo microcontrolador;
- **Joystick:** Configurado como entradas analógicas no microcontrolador;
- **Botões:** Entradas digitais com pull-up.

Comandos e Registros Utilizados

PWM (Pulse Width Modulation) utilizado para controlar a velocidade do ventilador, simbolizado através do LED vermelho, variando entre três níveis de intensidade. Também aciona os buzzers e o LED azul em uma intensidade mais baixa:

```
gpio_set_function(uint gpio, GPIO_FUNC_PWM);
slice = pwm_gpio_to_slice_num(uint gpio);
pwm_set_clkdiv(slice, float divider);
pwm_set_wrap(slice, uint_16t wrap);
pwm_set_enabled(slice, true);
pwm_set_gpio_level(uint gpio, uint_16t level);
```

I2C utilizado para comunicação com o display:

```
i2c_init(I2C_PORT, 400 * 1000); // Inicializa o I2C com frequência de 400 kHz
// Configura os pinos SDA e SCL como I2C e habilita pull-ups
gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
gpio_pull_up(I2C_SDA);
gpio_pull_up(I2C_SCL);
// Inicializa e configura o display
ssd1306_init(ssd, WIDTH, HEIGHT, false, ADDRESS, I2C_PORT);
ssd1306_config(ssd);
ssd1306_send_data(ssd);
// Limpa o display
```

```
ssd1306_fill(ssd, false);
ssd1306_send_data(ssd);
```

PIO (Programmable Input/Output) utilizado para o controle da matriz de LEDs:

```
pio_add_program(pio0, &ws2812_program);
sm = pio_claim_unused_sm(np_pio, false);
ws2812_program_init(np_pio, sm, offset, pin, 800000.f);
pio_sm_put_blocking(np_pio, sm, dados);
```

ADC (Analog to Digital Converter) utilizado para ler os valores dos sensores de temperatura e umidade:

```
adc_gpio_init(uint gpio);
adc_select_input(uint input);
adc_read();
```

GPIO (General-Purpose Input/Output) utilizado para botões e sensores:

```
gpio_init(uint gpio);
gpio_set_dir(uint gpio, GPIO_IN);
gpio_pull_up(uint gpio);
gpio_get(uint gpio);
```

Descrição da Pinagem Usada

- **Sensores de Temperatura e Umidade:**

JSK_Y (26) → Simulação do sensor de temperatura.
JSK_X (27) → Simulação do sensor de umidade.

- **Ventilador e Umidificador:**

RED_LED (13) → Simulação do ventilador via PWM.
BLUE_LED (12) → Simulação do umidificador via PWM.

- **Sensor de Nível:**

BUTTON_B (6) → Simula o sinal do sensor de nível.

- **Display:**

I2C_SDA (14) e I2C_SCL (15) → Comunicação I2C.

- **Botão do joystick e Botão A:**

JSK_SEL (22) → Botão do joystick para alternar telas.

BUTTON_A (5) → Botão para configurar temperatura e umidade.

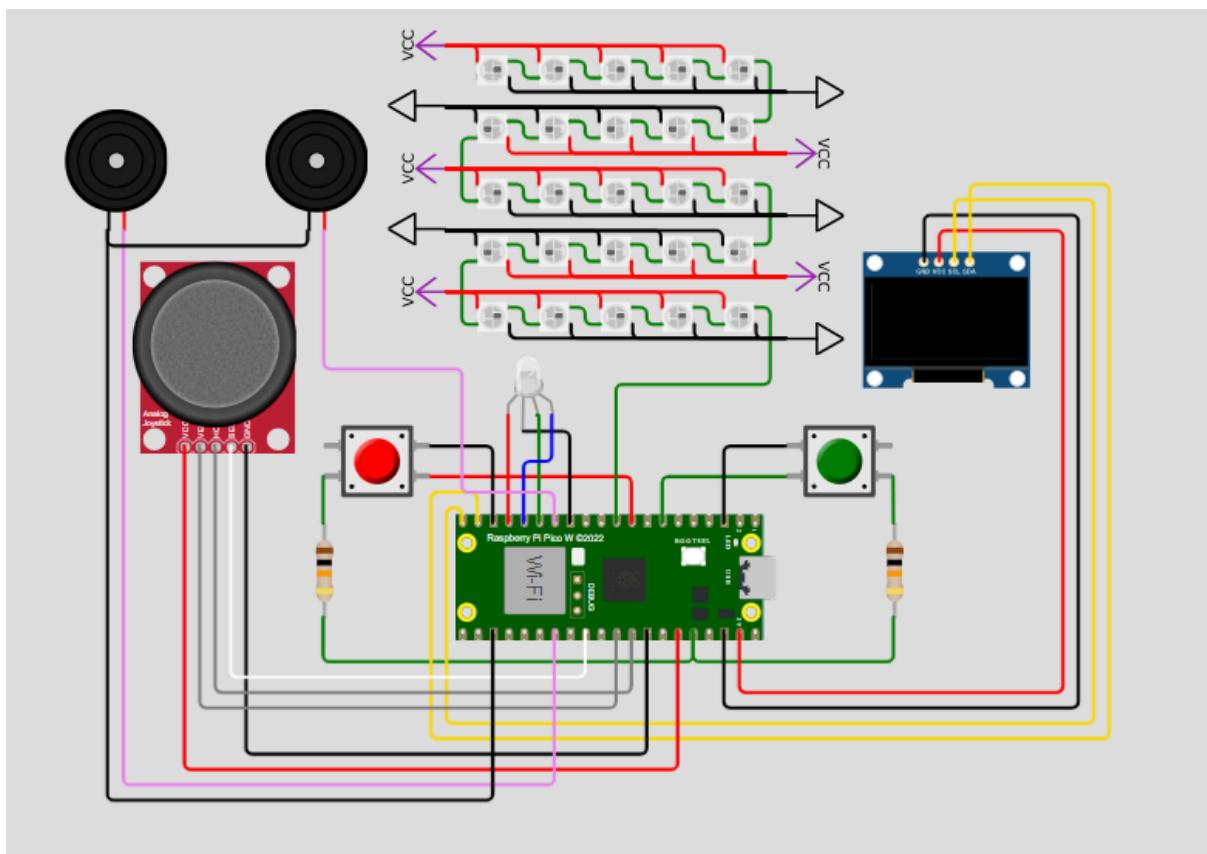
- **Matriz de LEDs:**

MATRIX_PIN (7) → Controle da matriz WS2812 via PIO.

- **Buzzers:**

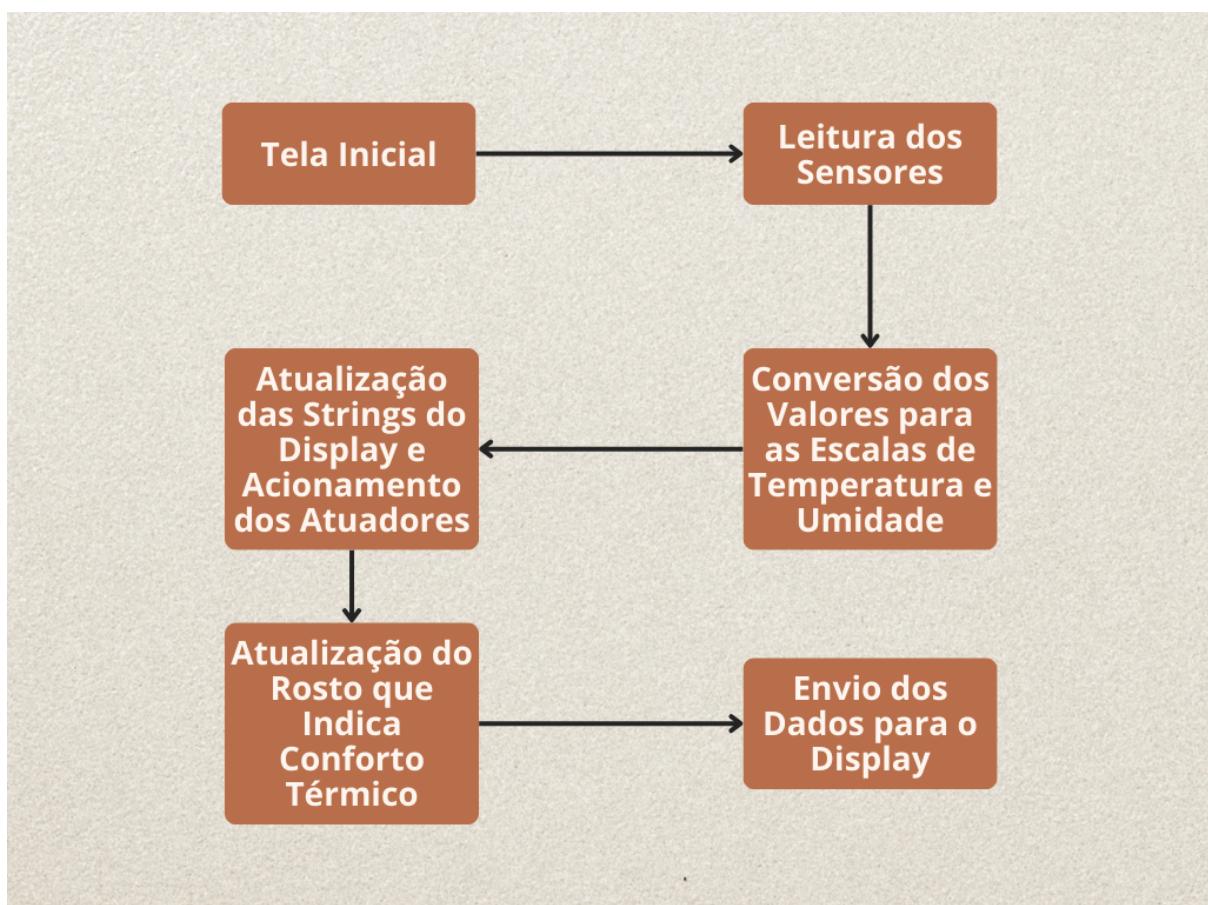
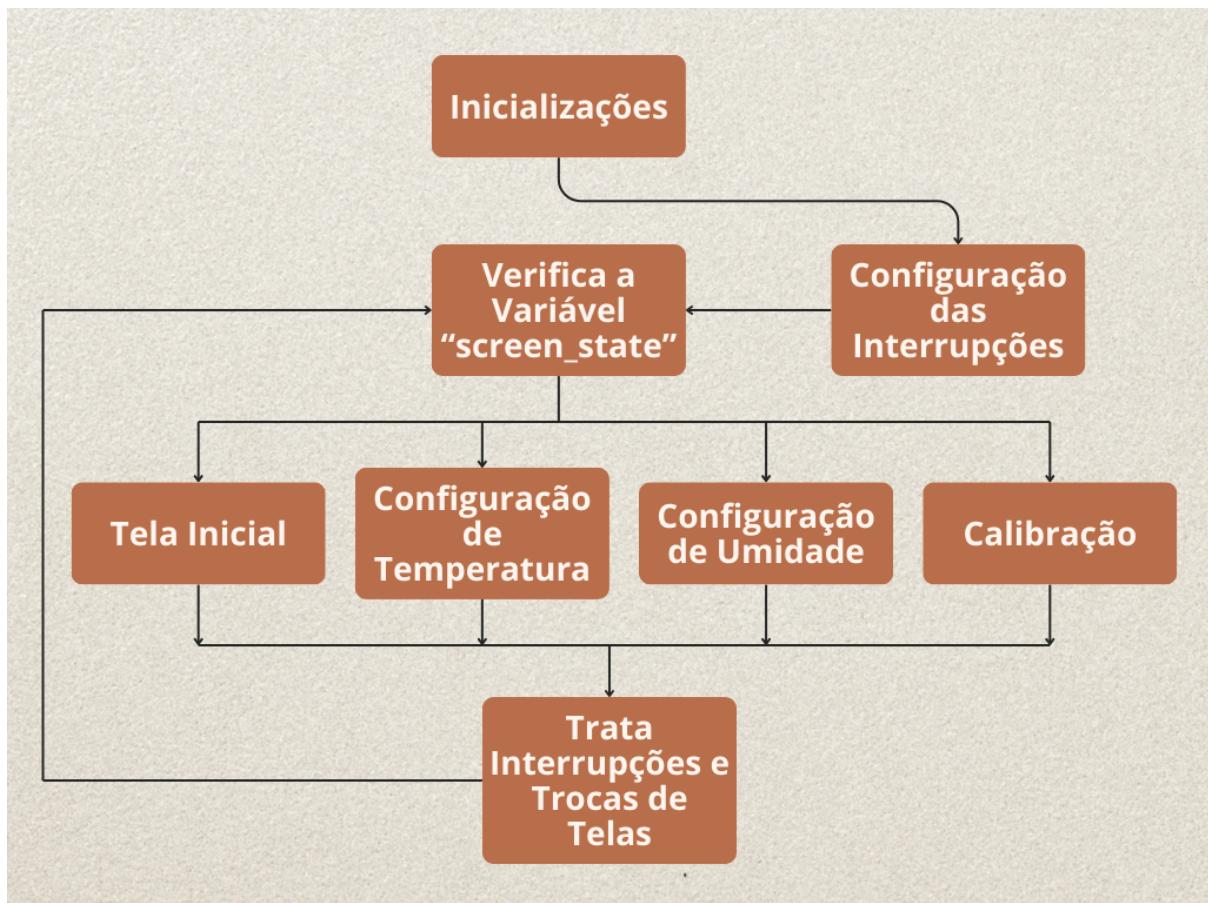
BUZZER_A (21) e BUZZER_B (10) → Emissão de alertas sonoros.

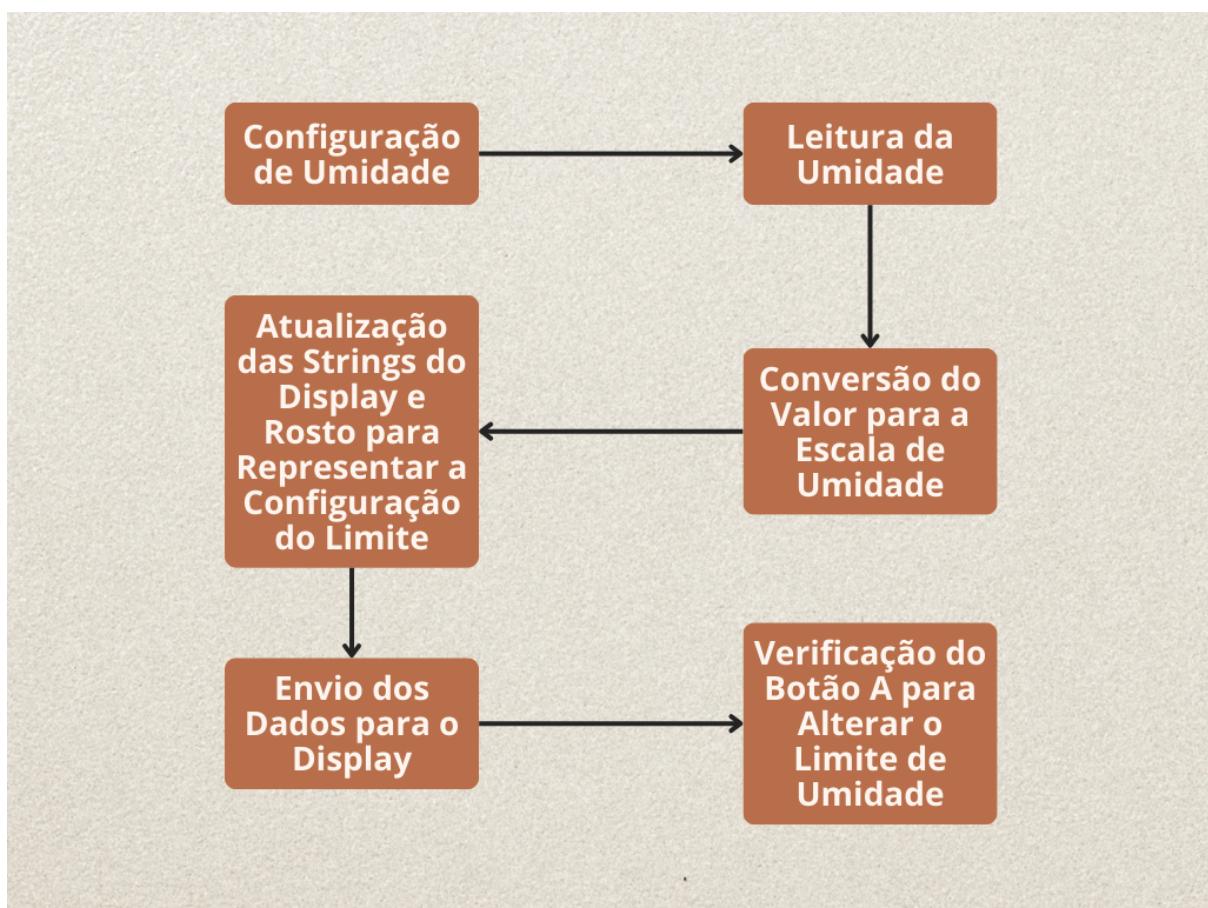
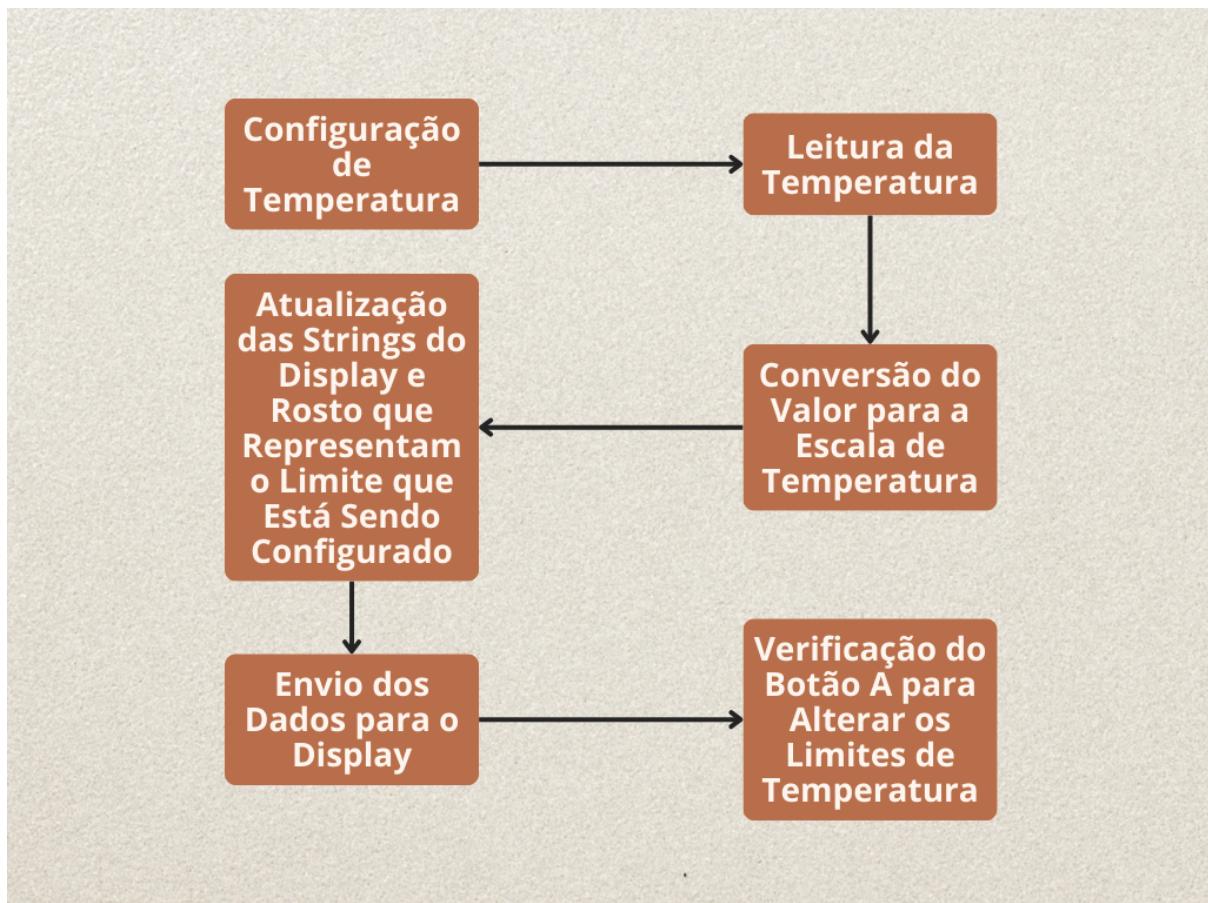
Circuito completo do hardware

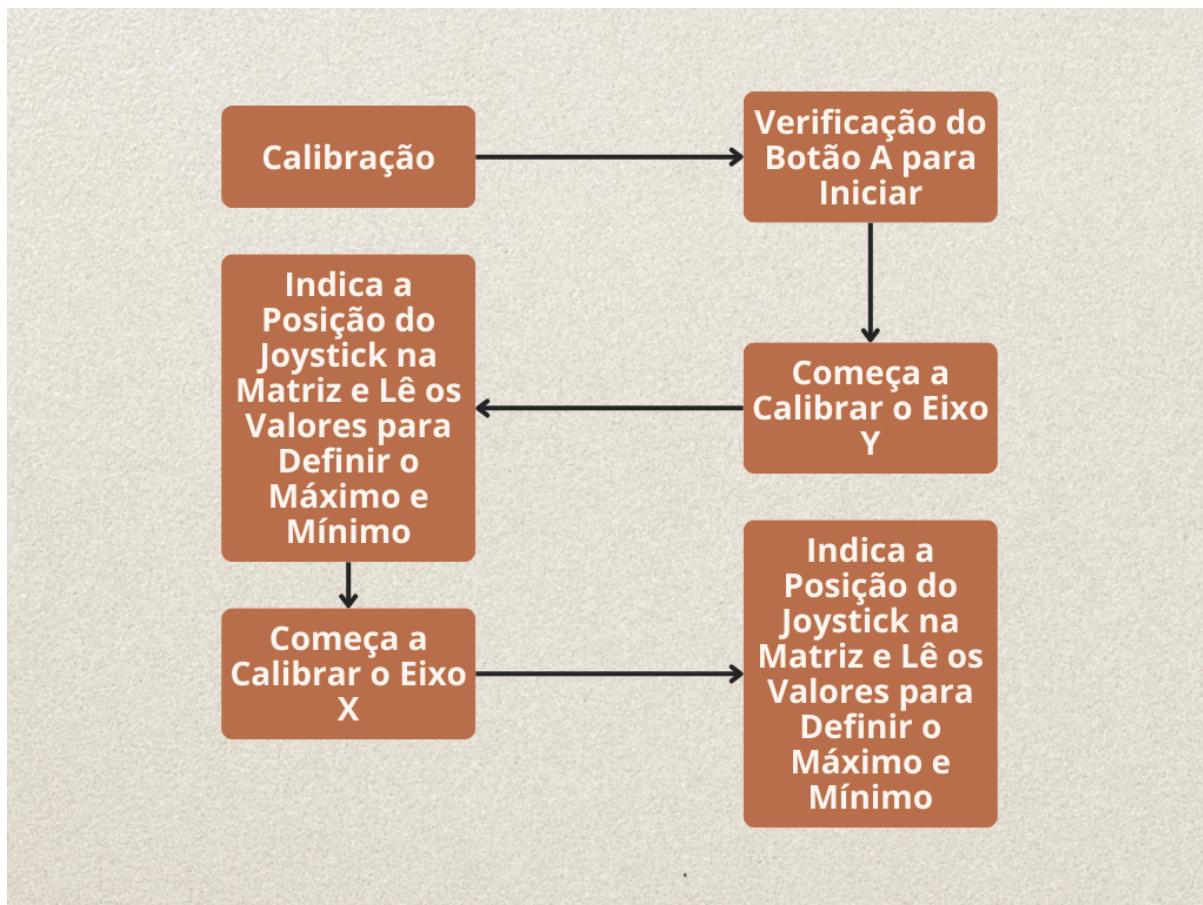


Especificação do Firmware

Blocos Funcionais







Descrição das funcionalidades

- **Leitura dos Sensores:** Utiliza ADC para capturar valores do joystick simulando temperatura e umidade;
- **Processamento dos Dados:** Conversão dos valores lidos para uma escala de temperatura (-15 a 50°C) e umidade (0 a 100%);
- **Controle dos Atuadores:**
 - Ventilador simulado pelo LED vermelho com PWM;
 - Umidificador simulado pelo LED azul com PWM;
 - Buzzers com PWM.
- **Exibição na Matriz de LEDs:** Mostra avisos e a tela atual;
- **Exibição no Display:** Mostra temperatura, umidade, status do sistema e ícones faciais representando o conforto térmico;
- **Configuração pelo Usuário:** Ajuste de limites de temperatura e umidade utilizando botões e joystick;
- **Calibração do joystick:** Indica a posição e lê os eixos para ajustar o processamento para uma simulação mais precisa.

Definição das variáveis

- **Temperatura e Umidade:** `y_scaled`, `x_scaled` armazenam valores convertidos;
- **Estado do Sistema:** `screen_state`, `fan_low`, `fan_medium`, `fan_high`, `humidifier_on`, `face_fan` e `face_humidifier` definem os estados do ventilador e umidificador;
- **Display:** `string1`, `string2`, `string3`, `string4` contém os caracteres que serão enviados ao display;
- **Interação:** `change_screen`, `contador` controlam a navegação entre telas e ajustes;
- **Flags:** `flag_b` e `switch_b` indicam as interrupções e simulam o sinal do sensor de nível.

Fluxograma

O fluxograma geral do software segue a sequência:

1. Leitura dos sensores;
2. Processamento dos valores;
3. Ajuste do ventilador e umidificador;
4. Atualização do display;
5. Monitoramento dos botões para configuração e calibração;
6. Acionamento de alertas, se necessário;
7. Repetição contínua do loop principal.

Inicialização

O processo de inicialização configura os periféricos e módulos necessários:

- Inicializa o display via I2C;
- Configura PWM para controle do ventilador e umidificador;
- Ativa o ADC para leitura dos sensores (joystick);
- Configura GPIOs para botões e buzzers;
- Inicializa a matriz de LEDs via PIO.

Configuração dos Registros

- **PWM:** Ajuste da velocidade do ventilador e acionamento do umidificador;
- **I2C:** Comunicação com o display OLED;
- **GPIOs:** Leitura dos botões e sensores;
- **PIO:** Controle da matriz de LEDs;
- **ADC:** Conversão dos sinais analógicos dos sensores.

Estrutura e Formato dos Dados

- **uint8_t**: Utilizado para armazenamento de pequenos valores inteiros, como status de dispositivos e configurações de LEDs;
- **uint16_t**: Utilizado para armazenar valores dos sensores analógicos e medidas calibradas do joystick;
- **int**: Empregado para cálculos matemáticos, escalas e controle de estados do sistema;
- **bool**: Variáveis booleanas são usadas para flags de controle de eventos e estados de componentes;
- **char[]**: Strings utilizadas para exibir mensagens no display;
- **float**: Utilizado para cálculos que exigem precisão decimal, como a conversão dos valores analógicos.

Protocolo de Comunicação

- O sistema utiliza **I2C** (Inter-Integrated Circuit) para comunicação entre o Raspberry Pi Pico W e o display. O I2C é um protocolo serial síncrono que permite comunicação entre múltiplos dispositivos através de um barramento de dois fios (SDA para dados e SCL para clock).

Formato do Pacote de Dados

As mensagens no protocolo I2C são organizadas em dois tipos de quadro:

1. Quadro de Endereço;
2. Quadro de Dados.

A comunicação ocorre da seguinte forma:

- Início (Start Condition);
- Endereçamento;
- Transferência de Dados:
 - Escrita;
 - Leitura.
- Confirmação (ACK/NACK);
- Término (Stop Condition);

Execução do Projeto

Metodologia

O desenvolvimento deste projeto iniciou-se com a fase de levantamento de ideias, onde considerei diferentes possibilidades antes de definir a proposta final. Um dos critérios fundamentais que eu defini foi a utilização de todos os periféricos da placa BitDogLab, pois queria explorar todos os conhecimentos adquiridos ao longo do curso.

Então, realizei uma série de pesquisas sobre projetos em diversas áreas, buscando inspiração e procurando soluções que pudessem ser adaptadas e aplicadas ao contexto do projeto. Durante esse processo, recorri também ao fórum do Moodle, revisitando uma postagem que havia feito anteriormente. A partir dela, surgiu a ideia de utilizar o joystick da BitDogLab para simular sensores de temperatura e umidade.

Com essa base definida, busquei inspiração no meu cotidiano e nos problemas ao meu redor para conseguir uma aplicação prática e relevante, e, após conseguir pensar na minha ideia, foquei em montar uma interface intuitiva para o usuário, explorando todos os recursos disponíveis, como a matriz de LEDs, o display e os buzzers.

A partir dessas definições, dei início à implementação, organizando as funcionalidades do hardware e software de forma estruturada para garantir a eficiência e a usabilidade do sistema. Conseguí aproveitar bastante dos códigos que eu já tinha feito nas atividades do curso para acelerar a montagem do firmware e, depois de finalizar o código, comecei a etapa de testes, em que detectei e corrigi alguns bugs para, então, melhorar e entregar um código robusto.

Testes de Validação

Para realizar a validação do firmware, fiquei um bom tempo utilizando o sistema e mexendo nos botões e no joystick para verificar o seu funcionamento, com isso, obtive que:

- Durante os testes de validação, foi identificado um bug na etapa de calibração do joystick que, ao pressioná-lo no meio do processo, fazia com que o contador da tela atual fosse incrementado. Para solucionar esse problema, as interrupções foram temporariamente desabilitadas durante o processo de calibração, garantindo um funcionamento mais estável.
- Após essa correção, o sistema foi testado repetidamente para garantir a execução correta do fluxo do código. Foram realizadas diversas tentativas de quebrar o funcionamento normal do firmware, pressionando todos os botões nos momentos mais diversos na tentativa de encontrar algum outro bug. O sistema se mostrou robusto e confiável diante dessas tentativas.

Discussão dos Resultados

O desenvolvimento do Sistema de Controle para Conforto Térmico me serviu de grande aprendizado, me permitindo testar e validar conceitos essenciais dos

sistemas embarcados. Durante a execução do projeto, foi possível implementar o controle automático do ventilador e do umidificador simulando os sensores com o joystick, bem como implementar uma interface com o usuário utilizando o display e a matriz de LEDs e os buzzers.

Os testes realizados mostraram que o sistema responde corretamente às variações nos valores simulados de temperatura e umidade, ajustando automaticamente o funcionamento do ventilador e do umidificador conforme os parâmetros definidos pelo usuário. A interface gráfica foi um diferencial, proporcionando uma experiência intuitiva e interativa.

Os desafios mais relevantes encontrados foram implementar a mudança de “telas” do programa e desenhar a interface e os rostinhos, já que eram coisas que eu ainda não tinha feito durante o curso. O firmware foi exaustivamente testado, incluindo tentativas de forçar falhas pressionando botões em momentos não previstos. O sistema mostrou-se robusto, sem falhas identificadas, o que confirma a eficácia da abordagem implementada.

O projeto atingiu seus objetivos principais, validando conceitos como PWM, comunicação I2C, controle de GPIOs, PIO e ADC. Além disso, demonstrou como um sistema embarcado pode ser projetado para oferecer uma interface amigável e funcional ao usuário.

Com os resultados obtidos, acredito que a proposta é funcional e, com algumas alterações, pode ser expandida para aplicações reais, incorporando sensores físicos e atuadores de climatização para controle em ambientes internos. O projeto como um todo se mostrou muito bom para o meu aprendizado e para a validação dos conhecimentos adquiridos, além de simular como seria a implementação e comportamento esperados em um sistema real.

Referências

ABDULLAH, Rina et al. Design an automatic temperature control system for smart tudungsaji using Arduino microcontroller. ARPN Journal of Engineering and Applied Sciences, v. 11, n. 16, p. 9578-9581, 2016.

AMOO, A. L. et al. Design and implementation of a room temperature control system: Microcontroller-based. In: 2014 IEEE student conference on Research and development. IEEE, 2014. p. 1-6.

BANSAL, Hitu; MATHEW, Lini; GUPTA, Ashish. Controlling of temperature and humidity for an infant incubator using microcontroller. Int. J. Adv. Res. Electr. Electron. Instrum. Eng, v. 4, n. 06, p. 4975-4982, 2015.