EMBARCATECH

PROJETO FINAL: SIMULAÇÃO DE SISTEMA DE MONITORAMENTO DE TEMPERATURA NO TRANSPORTE DE FRUTAS

Gabriel Cosmo Oliveira

Apresentação do Projeto:

Este projeto simula um sistema de monitoramento de temperatura para o transporte de frutas, utilizando o microcontrolador RP2040 na plataforma de educação BitDogLab e seus periféricos. Ele emula a variação de temperatura dentro de uma câmara fria e simula um sistema de alarme e controle ambiental.

Objetivos do Projeto:

O objetivo principal deste projeto é simular o monitoramento e controle de temperatura em uma câmara fria para o transporte de frutas, garantindo que as condições ideais para preservação sejam mantidas. O projeto visa ilustrar a variação de temperatura, a detecção de condições ideais e a atuação de um sistema exaustor para renovar o ar e controlar a qualidade do ambiente. Além disso, ele oferece ao usuário a possibilidade de ajustar os parâmetros do intervalo de temperatura ideal através de um sistema de configuração.

Descrição do Funcionamento:

O funcionamento do projeto é dividido em várias funcionalidades interativas:

1. **Leitura de Temperatura Simulada**: A temperatura é simulada utilizando o eixo vertical do joystick da BitDogLab, cujos movimentos ajustam o valor da

temperatura exibido no display de LEDs SSD1306. Em um sistema real, a variação da temperatura seria proveniente da leitura de um sensor, que também seria feita através de um conversor analógico digital como o que foi implementado. E além de exibir a temperatura em um display, o sistema poderia, com o intuito de realizar análises posteriores, armazenar esses valores em algum dispositivo físico, ou mesmo em algum servidor na nuvem utilizando protocolos de comunicação sem fio.

2. **Indicação de Status de Temperatura**: O LED RGB é utilizado para indicar se a temperatura está dentro de uma faixa ideal (verde), um pouco fora da faixa ideal (amarelo) ou muito fora da faixa ideal (vermelho), funcionando como um alarme visual, além disso, um dos buzzers presentes na placa também é utilizado para emitir um sinal sonoro cuja intensidade também depende da discrepância entre a temperatura medida e a temperatura ideal. No sistema real, esses tipos de alerta poderiam ser utilizados para chamar a atenção do condutor do veículo que transporta as frutas, podendo estar presentes em um painel de caminhão, por exemplo. Os valores lidos poderiam ser usados para controlar a temperatura de maneira manual ou mesmo automática, acionando sistemas refrigeradores ou aquecedores.
3. **Ativação de Exaustor Simulado**: A cada 30 segundos, uma rotina de interrupção ativa uma matriz de LEDs WS2812, criando padrões visuais de dígitos que variam de 9 até 0 (uma contagem). O intuito dessa etapa é simular o funcionamento de um exaustor, cuja função seria renovar o ar da câmara fria, uma vez que, muitas frutas produzem gás etileno naturalmente durante o seu processo de amadurecimento, o que faz com que elas amadureçam ainda mais rápido, e durante um transporte de longas distâncias isso pode não ser adequado. Além disso, o controle via timer é apenas uma solução, mas existem outras, essa ativação poderia ser feita manualmente, por exemplo, ou mesmo utilizando algum sensor que detectasse o nível de concentração desse gás no ambiente, o tempo de 30 segundos escolhido também serve apenas para exemplificar o funcionamento.
4. **Configuração de Temperatura**: O projeto permite que o usuário, ao pressionar um botão, acesse uma rotina de configuração onde é possível informar novos limites de temperatura mínima e máxima. Essas configurações são feitas via comunicação UART USB, com validação dos dados inseridos. Em um projeto real, essas configurações poderiam ser feitas através de um aplicativo de celular, ou de alguma central de controle, desde que fosse

implementada a comunicação sem fio do microcontrolador com algum servidor de dados.

Justificativa:

A execução deste projeto se justifica pela necessidade crescente de sistemas de monitoramento ambiental para o transporte e conservação de alimentos, como frutas. Controlar a temperatura de forma eficiente é essencial para evitar o desperdício e preservar a qualidade dos produtos.

A simulação proposta neste projeto pode ser uma base para o desenvolvimento de sistemas reais de monitoramento e alarme, aplicáveis em setores de transporte de produtos perecíveis.

Além disso, um problema também decorrente desse transporte é a possível falta de cuidado do condutor com as medidas de segurança necessárias no processo, o que faz com que a automação dessas atividades seja de fundamental importância.

Especificação do Hardware:

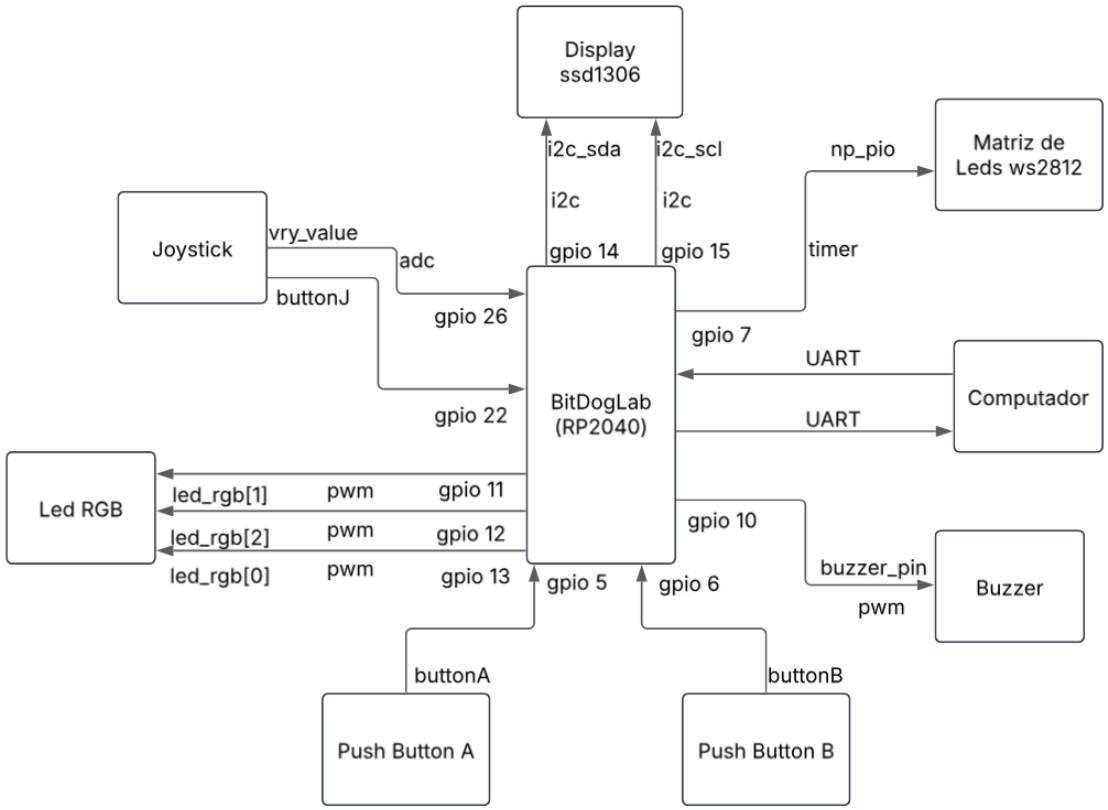
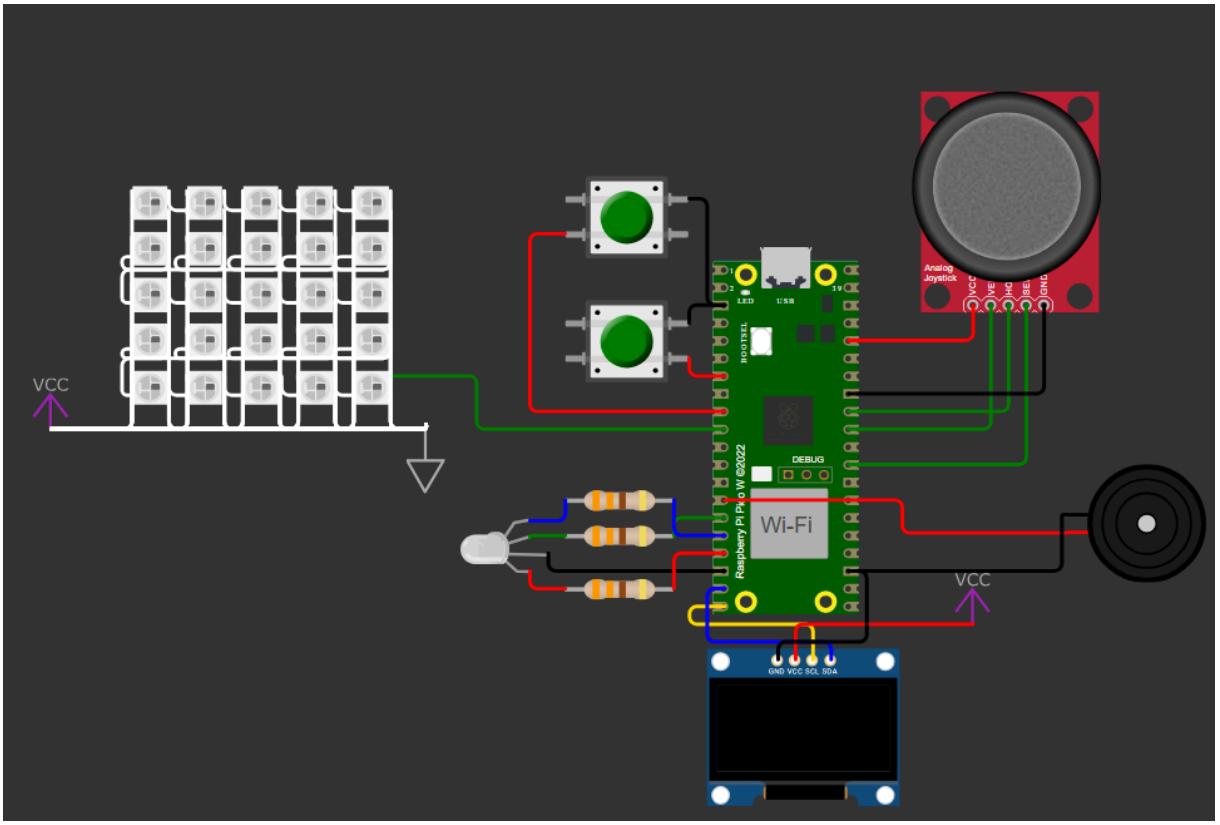


Diagrama de blocos do hardware utilizado

1. **Bloco do Microcontrolador (RP2040):** Esse é o coração de todo o projeto, é o microcontrolador que será responsável por enviar, receber e fazer o processamento de todos os dados das demais partes. Vale ressaltar que ele está sendo utilizado na plataforma do Raspberry Pi Pico W, integrado à placa BitDogLab. Portanto, detalhes mais gerais, como a disposição dos chips de memória, módulo Wi-Fi ou coisas do gênero, foram propositalmente omitidos.
2. **Bloco do Joystick:** Esse é o bloco responsável por realizar o controle da temperatura que será posteriormente exibida no display. Ele está acoplado aos pinos 22, 26 e 27 do microcontrolador. No entanto, apenas os pinos 22 e 26 foram utilizados (botão e eixo Y). Para realizar esse controle, foi configurado o canal 0 do conversor analógico-digital, utilizando comandos como *adc_select_input()* e *adc_read()*. O botão ligado ao pino 22 é utilizado para atribuir um valor específico à variável que armazena o valor de temperatura. Esse valor é a média do intervalo ideal. A leitura dos valores provenientes do movimento do joystick ainda é usada para controlar o sinal de PWM que é enviado ao LED RGB.
3. **Bloco do Botão A:** É usado para ativar uma rotina de configuração do firmware utilizando a UART e o USB da placa. Essa rotina será detalhada em um tópico posterior. O botão é conectado à GPIO 5 e foi configurado como entrada com um resistor de pull-up. Toda a configuração é feita na função *init_buttons()*. O bouncing é tratado via software nos três botões utilizados.
4. **Bloco do Botão B:** Esse botão está conectado à GPIO 6 e é utilizado apenas para colocar o microcontrolador em modo BOOTSEL. Sua configuração é a mesma do botão A, ou seja, entrada com resistor de pull-up. Como citado anteriormente, o bouncing também é tratado aqui. Vale ressaltar que tanto o botão A como o botão B utilizam interrupção na realização das suas ações, ambos no evento de borda de descida.
5. **Bloco do LED RGB:** O LED é controlado via PWM e tem o intuito de servir como alerta visual da variação de temperatura. Os seus canais vermelho, verde e azul estão conectados às GPIOs 13, 11 e 12, respectivamente. Eles são configurados como saída na função

init_rgb(). O ciclo de trabalho associado ao sinal de PWM que os controla é alterado dinamicamente conforme o movimento do joystick nas funções *yellow_led_blink()* e *red_led_blink()*, e também dentro do loop principal. O wrap máximo é de 10.000 e não tem divisor de clock configurado, gerando um período de aproximadamente 80 µs.

6. **Bloco do Buzzer:** O buzzer utilizado é o que está conectado à GPIO 10. Ele é utilizado para gerar um sinal sonoro simples, com o intuito de servir como alarme. Ele é controlado via PWM, com frequências de 1.000 e 2.000 hertz e ciclos de trabalho de 10% e 30%, respectivamente. Ele é ativado dentro das funções *yellow_led_blink()* e *red_led_blink()* e configurado na função *play_buzzer()*.
7. **Bloco da Matriz WS2812:** A matriz de LEDs é conectada à GPIO 7. O seu uso está atrelado a uma rotina de interrupção gerada via timer. Essa rotina é ativada a cada 30 segundos, gerando uma animação de contagem regressiva na matriz. Durante sua execução, a interação com o joystick fica bloqueada. Sua configuração é feita na função *npInit()*, e o seu controle pode ser feito através de comandos como *print_frame()*, *npClear()*, *npWrite()*, etc.
8. **Bloco do Display SSD1306:** O display é usado para exibir os dados de temperatura. Ele é controlado via protocolo I2C, a linha de clock é conectada na GPIO 15 e a linha de dados na GPIO 14, com endereço 0x3C. Ele é configurado na função *init_display()* com um baud rate de 400.000. Ele possui 128x64 pixels de resolução e é controlado através de comandos como *ssd1306_draw_string()*, *ssd1306_fill()*, *ssd1306_rect()*, *ssd1306_send_data()*, etc.
9. **Bloco do Computador:** Esse bloco representa o computador que irá interagir com a placa utilizando USB. Durante essa interação, o microcontrolador imprime dados na saída serial referentes à temperatura e, ao ativar a rotina de configuração pressionando o botão A, ele solicita a entrada de dados, que pode ser feita com softwares como o PuTTY. Essa entrada de dados é “semi-bloqueante”, pois, caso o usuário não insira a informação solicitada em um intervalo de 10 segundos, a rotina é finalizada. Além disso, os dados inseridos são validados antes de serem confirmados.



Circuito utilizado na simulação

A imagem acima mostra o circuito utilizado no projeto, o circuito pode conter simplificações no que diz às interfaces de acoplamento entre os componentes (drivers), podendo não representar com total fidelidade o circuito da placa, no entanto, é totalmente funcional, o circuito exato da placa pode ser encontrado em sua documentação da versão mais recente.

Descrição das funcionalidades:

O firmware desenvolvido no projeto tem como principal funcionalidade a simulação da medição de temperatura realizada por um sensor. Nesse caso, a variação de temperatura é simulada utilizando o joystick da placa, conforme mencionado anteriormente. Isso é feito com o conversor analógico-digital. A função responsável por isso é a *get_temperature()*, que realiza o cálculo da temperatura com base na leitura do canal 0 do conversor e retorna isso como um *float*. Além disso, ela também converte o *float* obtido do cálculo para uma *string* através da função *sprintf()*. Essa *string* gerada será efetivamente impressa posteriormente no display.

Como a resolução do ADC do RP2040 é de 12 bits, ou seja, possui 4096 valores, o centro ideal do joystick seria no valor 2047, referente à medição de uma tensão de 1,65 V. No entanto, os joysticks resistivos presentes nas placas não têm uma precisão muito alta. Por este motivo, uma leitura inicial do ADC é feita para determinar onde é o centro do joystick. Isso é crucial na realização dos cálculos pela função que determina a temperatura. Portanto, é importante que o programa seja iniciado com o joystick parado na posição central.

Enquanto a variação é simulada, o firmware monitora constantemente se a temperatura está dentro de um intervalo considerado seguro através de uma condicional no laço de repetição principal. Caso a temperatura se distancie um pouco do intervalo, um alarme visual e sonoro é acionado. Isso é feito utilizando um PWM para controlar a intensidade do LED RGB, fazendo com que ele pisque em amarelo, e do sinal sonoro do buzzer. A função que faz esse controle é a *yellow_led_blink()*. Caso a temperatura se distancie muito do intervalo ideal, o LED vermelho é acionado com uma intensidade maior e o buzzer também emite um sinal mais intenso. A função que realiza essa ação é a *red_led_blink()*.

Os ciclos de trabalho dos PWMs que controlam os LEDs são incrementados e decrementados de acordo com o tempo e com variáveis que definem o passo. Essas variáveis são chamadas *stepRed* e *stepYellow*. Além disso, o *wrap* do PWM também é levado em consideração no cálculo, com as variáveis de passo sendo frações do *wrap* total, que é 10000. Quanto aos ciclos do buzzer, eles são determinados de maneira estática dentro das funções que controlam os LEDs. Quando a função chamada é a que controla o LED amarelo, o buzzer irá soar com uma frequência de 1 kHz e ciclo de trabalho de 0,1. Quando a função chamada for a *red_led_blink()*, o LED soará na frequência de 3 kHz com ciclo de trabalho em 0,3.

A temperatura lida e convertida do ADC é impressa no display de LEDs SSD1306. Para isso, foram utilizadas as funções já demonstradas no decorrer do curso e uma função chamada *draw_temperature()*. O valor de temperatura é obtido como um ponto flutuante, mas, para facilitar a impressão, ele é convertido em uma *string* com a função *sprintf()*, como citado anteriormente. Dessa forma, cada caractere que compõe o valor de temperatura será impresso individualmente através da função *ssd1306_draw_char()*. Essa função utiliza uma matriz de símbolos desenhados no formato 8x8 pixels, disponível no arquivo *font.h*.

A leitura e impressão possuem um pequeno atraso que proporciona maior estabilidade ao programa, mas que também faz com que o valor mostrado no

display nem sempre seja exato quando o joystick está no centro. A falta de precisão do joystick, que induz ao uso de uma zona morta, também contribui para isso. Portanto, para resolver esse problema e atualizar a leitura, basta clicar no botão central do joystick. Será então impressa a temperatura média do intervalo de segurança.

O firmware implementa uma rotina de interrupção com *timer* de repetição que foi configurada para ser ativada a cada 30 segundos. Dentro dessa rotina, uma variável booleana de *flag* é atualizada e utilizada dentro do *loop* principal para chamar uma função que realiza a animação de contagem na matriz de LEDs. Durante a animação, o padrão da matriz varia entre os dígitos 9 até 0 com intervalos de 500 milissegundos entre eles, durante um total de 5 segundos. Essa rotina é bloqueante, ou seja, o controle de temperatura via joystick não pode ser feito enquanto ela está sendo executada.

Além disso, o botão A é utilizado para realizar uma entrada de dados via comunicação serial. Essa inserção pode ser realizada pelo teclado com o auxílio de um programa como o *PuTTY*. Ao pressionar o botão, uma interrupção é ativada e outra variável de alerta ou *flag* é acionada. A rotina é tratada dentro do *loop* principal com o auxílio de uma função chamada *temp_config()*. Nesta função, é solicitado que o usuário insira dois valores de temperatura que serão utilizados como limiar mínimo e máximo de um novo intervalo de segurança para a temperatura.

Durante a inserção, os valores são validados para verificar a sua coerência. Portanto, valores fora do intervalo total de medição não podem ser digitados. Além disso, essa rotina é *semi-bloqueante*, pois, embora a inserção de dados bloqueie a execução do programa, o usuário tem um limite de tempo máximo para fazê-lo. Originalmente, esse limite foi configurado para 5 segundos em cada inserção, mas pode ser alterado mudando a macro *TIMEOUT_US*.

Por fim, o botão B foi configurado para colocar a placa no modo *BOOTSEL*. Ele não simula nenhuma função do sistema real, apenas facilita o uso da placa. Como mencionado anteriormente, todos os botões utilizados possuem o tratamento de *bouncing* implementado via software.

Conclusão:

O projeto demonstrou com sucesso a viabilidade de um sistema de monitoramento de temperatura para o transporte de frutas, utilizando o

microcontrolador RP2040. A simulação desenvolvida permitiu a ilustração de diversos aspectos essenciais para um sistema real, como a leitura e exibição da temperatura, alertas visuais e sonoros, ativação de mecanismos de controle ambiental e a possibilidade de configuração dinâmica dos parâmetros de segurança.

A implementação do joystick como fonte de entrada para a temperatura simulada permitiu um controle interativo, enquanto o LED RGB e o buzzer possibilitaram a ilustração de um sistema de sinalização para condições indesejadas. A matriz de LEDs WS2812 foi utilizada para simular a atuação de um exaustor, evidenciando a problemática do controle de gases no ambiente de armazenamento. A integração do sistema com a comunicação UART também adicionou um elemento de configuração essencial, permitindo a adaptação do sistema a diferentes cenários de uso.

Apesar das limitações inerentes a uma simulação, o projeto apresenta um grande potencial de expansão. Em uma implementação real, sensores de temperatura e de concentração de gases poderiam substituir a entrada via joystick, e sistemas de refrigeração e ventilação poderiam ser controlados automaticamente com base nos valores medidos. Ademais, a integração com tecnologias de comunicação sem fio permitiria o monitoramento remoto, aprimorando ainda mais a eficiência do firmware.

No geral, a proposta cumpriu seu papel didático, permitindo um melhor entendimento do funcionamento de sistemas embarcados aplicados ao controle ambiental. Futuras evoluções do projeto poderiam torná-lo mais próximo de uma solução prática para o setor de transporte de produtos perecíveis.

Referências:

Silva, L. M., Costa, T. P., & Pereira, R. S. (2020). *Uso de Sensores para Controle de Temperatura em Armazenamento de Alimentos*. Food Control and Safety Journal, 31(6), 482-495. https://doi.org/10.1016/j.foodsaf.2020.04.014

Martins, P. M., & Silva, A. G. (2019). *Tecnologias de Monitoramento Ambiental para o Transporte de Produtos Perecíveis*. Journal of Food Technology, 56(2), 90-102. https://doi.org/10.1016/j.jfoodtech.2019.01.012

Santos, J. F., Silva, F. M., & Costa, R. S. (2021). *Sistema de Monitoramento e Controle de Temperatura em Câmaras Frias para Armazenamento de Alimentos*. Revista Brasileira de Engenharia de Produção, 34(4), 45-58. https://doi.org/10.1590/rbeprod.2021.0176

Almeida, J. P., & Fernandes, R. D. (2017). *Aplicações de Microcontroladores no Monitoramento de Temperatura em Processos Industriais*. Journal of Industrial Electronics, 42(1), 14-22. https://doi.org/10.1109/jie.2017.0456