Sistema Automatizado de Notificação e Devolução de Produtos Frios em Supermercados

Késia Laís dos Santos Souza

IFMA – Monte Castelo
2025

Sistema Automatizado de Notificação e Devolução de Produtos Frios em Supermercados

Késia Laís dos Santos Souza

Projeto final para o Programa de Capacitação Profissional em Sistemas Embarcados. - EmbarcaTech.

Monitoria: 10

IFMA – Monte Castelo
2025

RESUMO

O monitoramento eficiente de produtos frios em supermercados é essencial para garantir a qualidade e segurança alimentar. Este projeto apresenta o desenvolvimento do **Alerta Frio**, um sistema embarcado, utilizando o microcontrolador Raspberry Pi Pico W e periféricos integrados da placa BitDogLab, que facilita a devolução de produtos frios. O sistema detecta quando um cliente desiste de um item da seção de frios e notifica a equipe da frente de caixa para que o produto seja recolhido rapidamente, evitando deterioração e perda. Utiliza botões, LEDs e um display OLED para fornecer notificações visuais sobre o status dos produtos frios, garantindo uma comunicação clara e rápida entre os operadores de caixa e a equipe responsável pelo recolhimento.

Palavras – chave: Sistema embarcado, Supermercados, BitDogLab, Raspberry Pi Pico w, Comunicação eficiente, Qualidade e segurança alimentar

1. Sistema automatizado: Alerta Frio

Link wokwi: https://wokwi.com/projects/422889631360049153

1.1 Descrição

O projeto visa desenvolver um sistema automatizado para auxiliar na devolução eficiente de produtos frios em supermercados. Quando um cliente desiste de um item da seção de frios, o operador pode notificar a equipe da frente de caixa para que o produto seja recolhido rapidamente, evitando que o item se deteriore e cause perda.

O sistema utiliza o microcontrolador Raspberry Pi Pico W, em conjunto com os periféricos integrados da BitDogLab os botões, LEDs e um display OLED para notificações visuais. Ao pressionar um botão, o sistema alterna os LEDs e exibe mensagens no display, indicando o status do produto frio (por exemplo, "Frios no Caixa 2" e "Recolhido caixa 2").

Este projeto é uma solução prática e eficiente para otimizar o processo de devolução de produtos frios, garantindo que os itens sejam manipulados de forma rápida e adequada, mantendo a qualidade e segurança alimentar.

1.2 Objetivo Geral

Desenvolver um sistema automatizado para auxiliar na devolução eficiente de produtos frios em supermercados, utilizando o microcontrolador Raspberry Pi Pico W em conjunto com botões, LEDs e um display OLED para notificações visuais, garantindo a comunicação clara e rápida entre os operadores de caixa e a equipe responsável pelo recolhimento dos produtos

1.3 Objetivos Específicos:

- **Reduzir Desperdício de Produtos Frios:** Implementar uma solução que permita a rápida devolução de produtos frios à seção apropriada, evitando a deterioração e perdas financeiras.
- **Melhorar a Eficiência Operacional:** Facilitar a comunicação entre os operadores de caixa e a equipe de recolhimento através de um sistema de notificações visuais, melhorando a eficiência no processo de devolução.
- **Fornecer Notificações Claras e Específicas:** Utilizar botões e LEDs para indicar a necessidade de recolhimento de produtos frios, exibindo informações detalhadas no display OLED sobre o caixa específico e o status da devolução.
- **Fornecer Notificações Claras e Específicas:** Utilizar botões e LEDs para indicar a necessidade de recolhimento de produtos frios, exibindo informações detalhadas no display OLED sobre o caixa específico e o status da devolução.
- **Automatizar o Processo de Devolução:** Desenvolver um sistema que permita aos operadores de caixa notificar a equipe de recolhimento de maneira simples e eficaz, e acompanhar o status da devolução em tempo real.
- **Assegurar a Qualidade e Segurança Alimentar:** Garantir que os produtos frios sejam manipulados de forma rápida e adequada, mantendo a qualidade e a segurança alimentar dos itens devolvidos.

1.4 Justificativa

A execução do projeto de automação para devolução de produtos frios se justifica pela necessidade de otimizar o processo de devolução em supermercados, reduzindo desperdícios e melhorando a eficiência operacional. Produtos frios têm uma vida útil limitada e, quando não são devolvidos rapidamente, podem se deteriorar, resultando em perdas financeiras e impacto ambiental.

1.5 Originalidade

Em supermercados de grande porte, já existe um sistema semelhante, quando o operador precisa de algo, ele aperta um botão no caixa que acende uma luz. No entanto, quem está na frente de caixa (líderes, apoios, prevenção etc.) não sabe do que se trata essa notificação.

O projeto que desenvolvi é original porque implementa mais botões e LEDs/luzes que notificam especificamente sobre produtos frios no caixa. A informação aparecerá no monitor ou em um dispositivo que ficará com os responsáveis pelo recolhimento, indicando em qual caixa está o item. Após recolherem os frios do caixa, o operador aciona outro botão para informar que o processo já foi concluído.

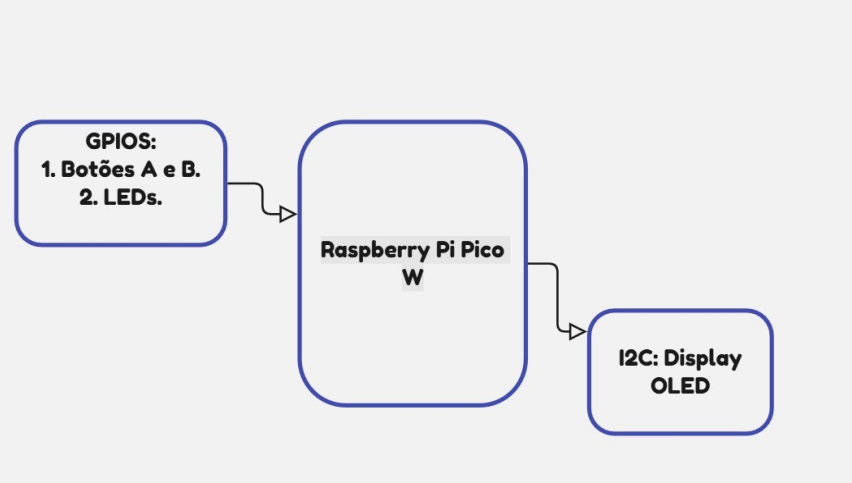
Essa abordagem fornece uma comunicação mais clara e eficiente, garantindo que os itens sejam rapidamente identificados e devolvidos à seção correta, evitando perdas e melhorando o atendimento ao cliente.

Caixa de supermercado, destaque para a lâmpada próxima a numeração do caixa.

2. Especificação do hardware

O projeto Alerta Frio foi desenvolvido utilizando a arquitetura da placa BitDogLab de microcontrolador da Raspberry Pi.

2.1 Diagrama em bloco



Função de cada bloco

Raspberry Pi Pico W:

- Atuar como o controlador principal, gerenciando as entradas dos botões, o estado dos LEDs, e a comunicação I2C com o display OLED.

GPIOS:

Botões A e B:

- **Botão A:** Notifica que há produtos frios no caixa que precisam ser recolhidos.
- **Botão B:** Informa que os produtos frios foram recolhidos.

LEDs:

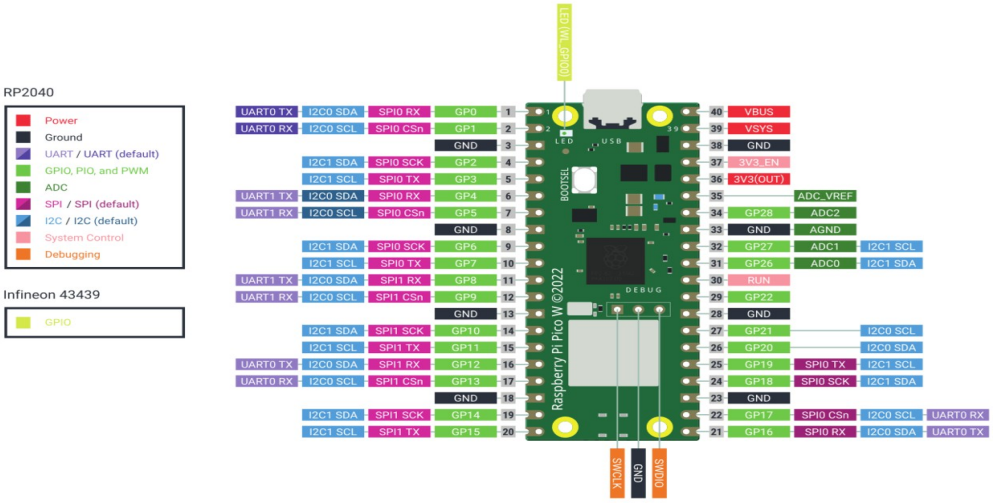
- **LED Vermelho:** Indica que há produtos frios a serem recolhidos.
- **LED Verde:** Indica que os produtos frios foram recolhidos.

I2C: Display OLED:

- Mostrar mensagens detalhadas sobre a necessidade de recolhimento de produtos frios, especificando o caixa e o status da devolução.

2.2 Configuração de cada bloco

- **Raspberry Pi Pico W:** Configuração dos pinos para botões, LEDs e I2C.



- **Botões:** Configurados como entradas com *pull-up* interno.

```
// Configuração dos botões
gpio_init(BOTAO_A);
gpio_set_dir(BOTAO_A, GPIO_IN);
gpio_pull_up(BOTAO_A);
gpio_init(BOTAO_B);
gpio_set_dir(BOTAO_B, GPIO_IN);
gpio_pull_up(BOTAO_B);
```

- **LEDs:** Configurados como saídas.

```
// Configuração dos LEDs
gpio_init(LED_R);
gpio_set_dir(LED_R, GPIO_OUT);
gpio_init(LED_G);
gpio_set_dir(LED_G, GPIO_OUT);
```

- **Display OLED:** Inicialização do display e configuração de área de renderização.

```
// Preparar área de renderização para o display
struct render_area frame_area = {
    .start_column = 0,
    .end_column = ssd1306_width - 1,
    .start_page = 0,
    .end_page = ssd1306_n_pages - 1
};
```

2.3 Comandos e Registros Utilizados

Inicialização do Display OLED:

- ***ssd1306_init()*:** Inicializa o display OLED.
- ***calculate_render_area_buffer_length():*** Calcula o comprimento do buffer da área de renderização.
- ***render_on_display():*** Renderiza a imagem no display.

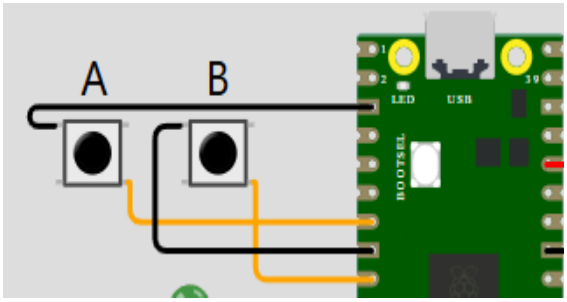
GPIO para Botões e LEDs:

- ***gpio_init():*** Inicializa o pino GPIO.
- ***gpio_set_dir():*** Define a direção do pino GPIO (entrada ou saída).
- ***gpio_pull_up():*** Habilita o *pull-up* interno para o pino GPIO.
- ***gpio_put():*** Define o estado do pino GPIO (alto ou baixo).
- ***gpio_get():*** Obtém o estado do pino GPIO.

2.4 Pinagem

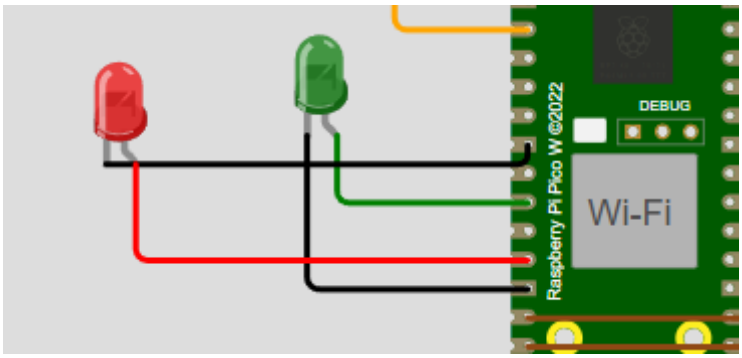
Botões:

- **BOTAO_A (Pino 5):** Entrada para o botão A.
- **BOTAO_B (Pino 6):** Entrada para o botão B.



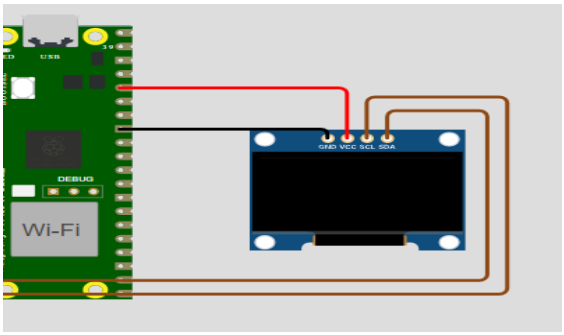
LEDs:

- **LED_R (Pino 13):** Saída para o LED vermelho.
- **LED_G (Pino 11):** Saída para o LED verde.



I2C para Display OLED:

- **I2C_SDA (Pino 14):** Linha de dados I2C.
- **I2C_SCL (Pino 15):** Linha de clock I2C.

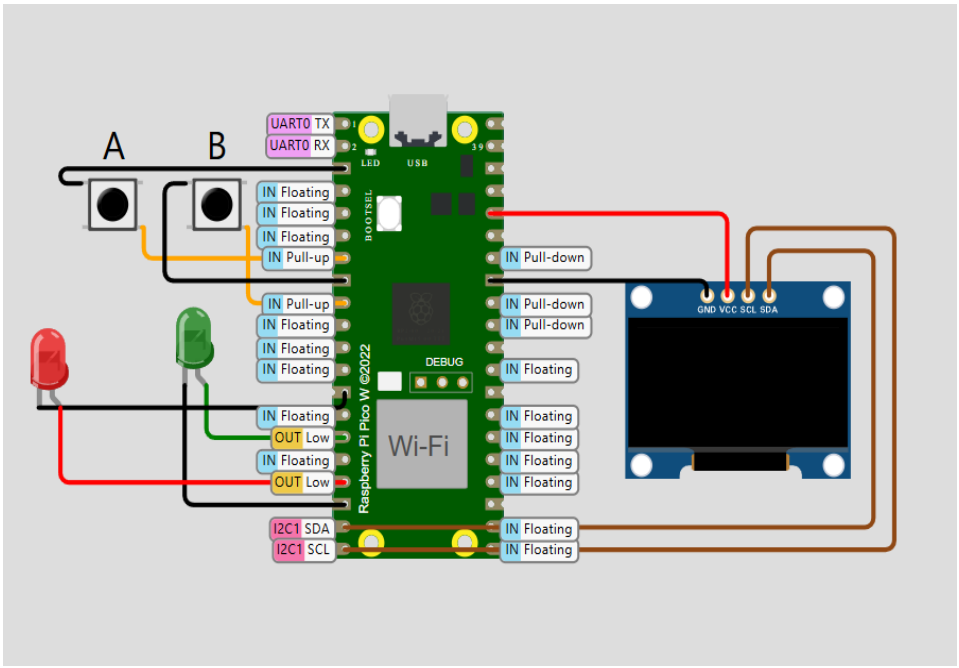


Pinos de alimentação e GND:

VSYS: Alimentação do sistema

GND: Terra do sistema

2.5 Circuito Completo do Hardware

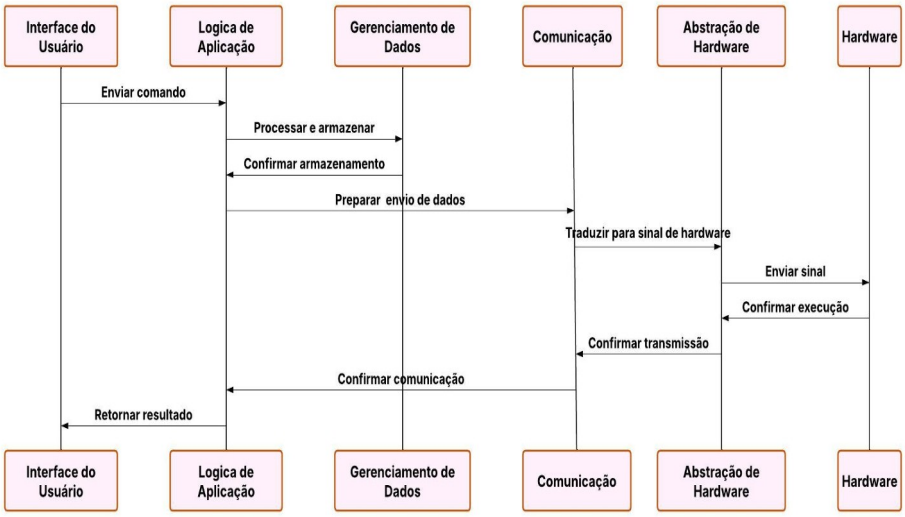


Conexões:

- **Pino 5:** Conectado ao Botão A.
- **Pino 6:** Conectado ao Botão B.
- **Pino 13:** Conectado ao LED Vermelho.
- **Pino 11:** Conectado ao LED Verde.
- **Pino 14 (I2C_SDA):** Conectado ao pino SDA do display OLED.
- **Pino 15 (I2C_SCL):** Conectado ao pino SCL do display OLED.

3.Especificação do firmware

3. 1 Diagrama em blocos do software



3.2 Descrição das Funcionalidades

Interface do Usuário

Função: Permitir que o operador interaja com o sistema por meio de botões e LEDs.

Exemplo: Botão A para notificar que um item de frios foi deixado e precisa ser recolhido, Botão B para confirmar que o item foi recolhido.

Lógica de Aplicação

Função: Processar a lógica de notificação e exibição de mensagens.

Exemplo: Acender o LED vermelho e exibir a mensagem “Frios caixa 2” no display OLED quando o Botão A é pressionado.

Gerenciamento de Dados

Função: Armazenar e recuperar dados relevantes, como o estado dos botões e LEDs.

Exemplo: Monitorar continuamente o estado dos botões e atualizar as saídas conforme necessário.

Abstração de Hardware

Função: Abstrair as interações com os componentes de hardware.

Exemplo: Inicialização e controle dos LEDs e do display OLED.

Hardware

Função: Componentes físicos utilizados no sistema.

Exemplo: Raspberry Pi Pico W, LEDs, botões, display OLED.

3.3 Definição das Variáveis

```
while (true) {
    // Verificar se o botão A está pressionado
    if (!gpio_get(BOTAO_A)) {
     // Ligar o LED do vermelho
        gpio_put(LED_R, 1);
        gpio_put(LED_G,0);

    // Exibir mensagem no display OLED
        ssd1306_draw_string(ssd, 0, 0, "Frios no Caixa 2");
        render_on_display(ssd, &frame_area);
    }

    // Verificar se o botão B está pressionado
    if (!gpio_get(BOTAO_B)) {
        // Ligar o LED verde
        gpio_put(LED_G, 1);
        gpio_put(LED_R,0);

    // Exibir mensagem no display OLED
        ssd1306_draw_string(ssd, 0, 1, " Recolhido 2");
        render_on_display(ssd, &frame_area);
    }
```

- **Variáveis dos Botões**

BOTAO_A: Armazena o estado do botão A (pressionado ou não).

BOTAO_B: Armazena o estado do botão B (pressionado ou não).

- **Variáveis dos LEDs**

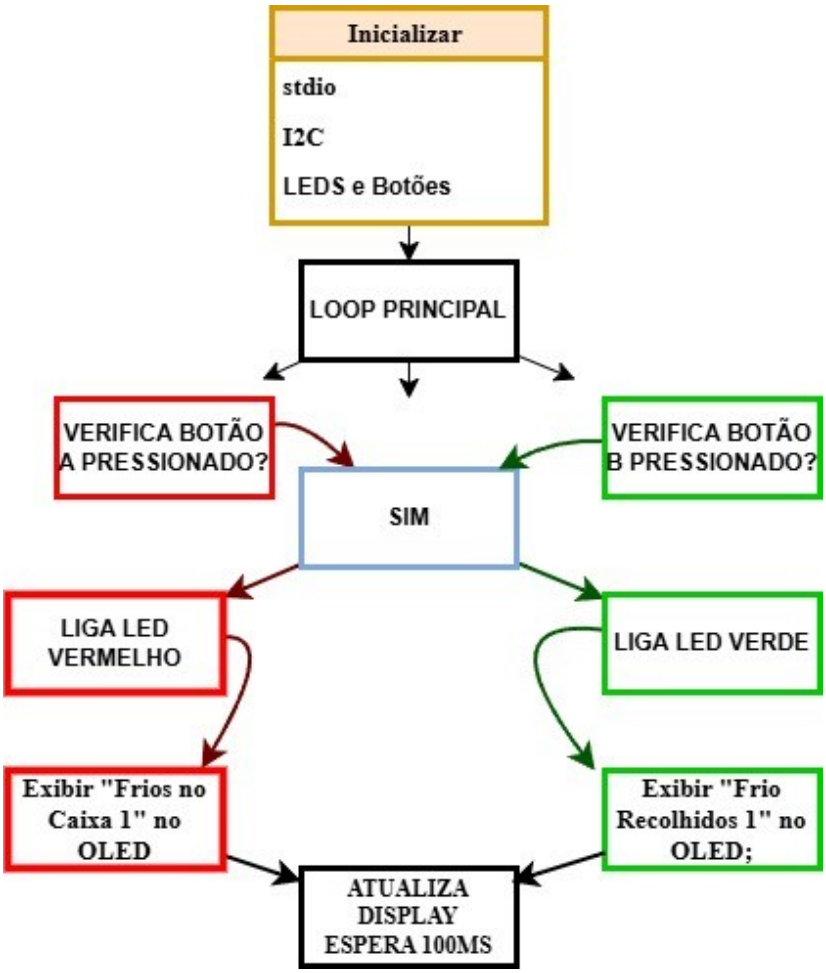
LED_R: Controle do LED vermelho (ligado ou desligado).

LED_G: Controle do LED verde (ligado ou desligado).

- **Variáveis do Display OLED**

ssd[]:memória temporária de dados para o display OLED.

3.4 Fluxograma do software



3.5 Inicialização

Processo de Inicialização do Software:

Configuração do I2C: Inicializa o protocolo I2C para comunicação com o display OLED.

Inicialização dos LEDs: Configura os pinos dos LEDs como saída.

Inicialização dos Botões: Configura os pinos dos botões como entrada com resistores *pull-up.*

Inicialização do Display OLED: Configura o display OLED, preparando a área de renderização.

3.6 Configurações dos Registros

Funções de Configuração dos Registros

Configuração do I2C:

```
// Inicialização do i2c
i2c_init(i2c1, ssd1306_i2c_clock * 1000);
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);
```

Configuração dos LEDs:

```
// Configuração dos LEDs
gpio_init(LED_R);
gpio_set_dir(LED_R, GPIO_OUT);
gpio_init(LED_G);
gpio_set_dir(LED_G, GPIO_OUT);
```

Configuração dos Botões:

```
// Configuração dos botões
gpio_init(BOTAO_A);
gpio_set_dir(BOTAO_A, GPIO_IN);
gpio_pull_up(BOTAO_A);
gpio_init(BOTAO_B);
gpio_set_dir(BOTAO_B, GPIO_IN);
gpio_pull_up(BOTAO_B);
```

Inicialização do Display OLED:

```
// Processo de inicialização completo do OLED SSD1306
ssd1306_init();

// Preparar área de renderização para o display (ssd1306_width pixels por ssd1306_n_pages páginas)
struct render_area frame_area = {
    .start_column = 0,
    .end_column = ssd1306_width - 1,
    .start_page = 0,
    .end_page = ssd1306_n_pages - 1
};

calculate_render_area_buffer_length(&frame_area);

// Zerar o display inteiro
uint8_t ssd[ssd1306_buffer_length];
memset(ssd, 0, ssd1306_buffer_length);
render_on_display(ssd, &frame_area);
```

3.7 Estrutura e Formato dos Dados

Dados Específicos Usados no Software:

Buffer do Display OLED (ssd[]): Um array que armazena os dados a serem renderizados no display OLED.

Estados dos Botões:

BOTAO_A: Armazena o estado do botão A (pressionado ou não).

BOTAO_B: Armazena o estado do botão B (pressionado ou não).

Controle dos LEDs:

LED_R: Controle do LED vermelho (ligado ou desligado).

LED_G: Controle do LED verde (ligado ou desligado).

3.8 Organização da Memória

Endereços de Memória Usados

Variáveis do GPIO: Registradores dos pinos GPIO para LEDs e botões.

Buffer do Display OLED (ssd[]): Área de memória dedicada para os dados do display OLED.

4. Metodologia do Projeto “Alerta Frio”

O código de execução do “Alerta Frio” encontra-se integralmente disponivel no repositorio GitHub: kesiasouza/ProjetoAlertaFrio

Pesquisas Realizadas: Realizar pesquisas é essencial para entender o que já foi feito e quais tecnologias estão disponíveis. No caso do “Alerta Frio”, a pesquisa envolveu estudar sistemas de devolução de produtos em supermercados, analisando diferentes métodos existentes e tecnologias utilizadas. Foram analisados componentes de hardware disponíveis, como microcontroladores, display OLED, LEDs e botões, além de identificar os protocolos de comunicação mais adequados para o projeto. Também foi feita uma análise sobre a eficiência operacional para melhorar a operação de caixas de supermercados.

Escolha do Hardware: Com base na pesquisa, foram selecionados os seguintes componentes de hardware: Raspberry Pi Pico W; Display OLED, para exibir mensagens e notificações ao operador; LEDs, como indicadores visuais para diferentes estados do sistema; Botões, para permitir a interação com o operador; e o protocolo de comunicação I2C para o display OLED, utilizando componentes da placa BitDogLab.

Definição das Funcionalidades do Software: O software deve ler o estado dos botões, monitorando continuamente se estão pressionados; controlar LEDs, acendendo ou apagando conforme o estado dos botões; exibir mensagens

no display OLED, mostrando notificações relevantes ao operador; e executar a lógica de negócios, processando e tomando ações com base nas entradas dos botões.

Inicialização da IDE: A IDE escolhida para o desenvolvimento foi o Visual Studio Code (VS Code). As etapas de inicialização incluíram baixar e instalar o VS Code, adicionar suporte ao Raspberry Pi Pico W na IDE, e instalar extensões necessárias, como a extensão C/C++ para VS Code e ferramentas para Raspberry Pi Pico.

Programação no VS Code: A programação foi realizada no VS Code, seguindo estas etapas: configurar os pinos do hardware, definindo os pinos para LED RGB, botões e I2C; inicializar os componentes, configurando e inicializando LED RGB, botões e display OLED; implementar a lógica do loop principal, monitorando os botões e atualizando LED RGB e display conforme necessário; testar e ajustar o código, realizando testes contínuos e ajustes conforme necessário.

Depuração: A depuração envolveu testes unitários para testar individualmente cada componente do sistema, como LED RGB, botões e display OLED; testes de integração, testando o sistema completo para garantir que todos os componentes funcionam juntos corretamente; correção de erros, identificando e corrigindo erros encontrados durante os testes; e otimização do código, melhorando a eficiência e reduzindo o uso de recursos. O envio do firmware para a placa é feito colocando a Raspberry Pi Pico W em modo BOOTSEL com o RESET, e logo após seleciona “RUN” na extensão da Raspberry Pi Pico no VS Code.

4.1 Testes de validação

Para garantir que o projeto “Alerta Frio” funcionasse corretamente e atendesse às expectativas, foram realizados diversos testes de validação. A seguir, descrevo os principais testes realizados e os resultados obtidos:

Link: Teste de Validação e Resultado final - Google Drive

Teste Inicial dos Componentes:

Objetivo: Verificar o funcionamento individual de cada componente do projeto.

Método: Testar cada componente separadamente, como o display OLED, botões e o LED RGB.

Resultado: Inicialmente, o display OLED não funcionou corretamente devido à falta das bibliotecas corretas. Após identificar o problema, realizei pesquisas e consegui as configurações corretas. Após isso, o display OLED passou a exibir textos conforme esperado.

Teste de Exibição do Display OLED:

Objetivo: Garantir que o display OLED exiba as mensagens corretamente.

Método: Testar a exibição de diferentes mensagens no display OLED.

Resultado: Durante os testes, observei que os números no texto estavam sendo exibidos duplicados. Após identificar o problema e ajustar o código, consegui resolver essa questão, e o display passou a exibir corretamente.

Teste dos Botões:

Objetivo: Garantir que os botões A e B funcionem corretamente.

Método: Pressionar os botões A e B e observar a resposta do sistema, verificando se os LEDs acendem/apagam e se as mensagens são exibidas no display OLED.

Resultado: Após a configuração correta, os botões responderam conforme esperado, acionando os LEDs e exibindo mensagens no display OLED.

Teste do LED RGB:

Objetivo: Verificar o funcionamento do LED RGB.

Método: Testar o LED RGB individualmente, verificando se ele acende nas cores vermelho, verde e azul conforme a configuração.

Resultado: O LED RGB funcionou corretamente, acendendo nas cores esperadas.

Teste Completo do Projeto:

Objetivo: Garantir que todos os componentes funcionem juntos corretamente.

Método: Montar o projeto completo e testar a interação entre os componentes, verificando se os botões, LEDs e display OLED funcionam em conjunto conforme a lógica do projeto.

Resultado: O teste completo foi bem-sucedido, com todos os componentes funcionando corretamente e o sistema operando conforme o esperado.

4.2 Discussão dos Resultados

Os resultados dos testes de validação mostraram que o projeto “Alerta Frio” é confiável e atende aos requisitos definidos. A seguir, uma análise detalhada dos resultados:

Funcionamento Correto: Todos os componentes de hardware (LEDs, botões, display OLED) funcionaram conforme o esperado, respondendo corretamente aos comandos.

Exibição Clara: As mensagens foram exibidas corretamente no display OLED, garantindo que o operador possa ver claramente as notificações.

Responsividade: Os botões responderam prontamente quando pressionados, acionando os LEDs e exibindo mensagens no display sem atrasos perceptíveis.

Estabilidade: O sistema manteve-se estável e funcional durante todos os testes de robustez, sem falhas ou comportamento inesperado.

Conclusão

Com base nos testes realizados e na análise dos resultados, podemos concluir que o projeto “Alerta Frio” é confiável e eficiente. Ele atende aos objetivos propostos e melhora a eficiência da operação de devolução de produtos frios em supermercados. A experiência profissional como operadora de caixa foi fundamental para identificar um problema real e criar uma solução prática e inovadora.

Referências:

- **Experiência Profissional:**
  Trabalhei como operadora de caixa em uma rede atacadista, onde observei a necessidade de um sistema para a devolução eficiente de produtos frios. Essa experiência me ajudou a identificar um problema real e a pensar em uma solução prática.

- **Documentação do Raspberry Pi Pico W:** https://www.raspberrypi.com/documentation/

- **Bibliotecas para Display OLED SSD1306:** https://github.com/BitDogLab/BitDogLab-C/tree/main/display_oled/inc