Projeto Final - EmbarcaTech
Alarme utilizando a Placa BitDogLab

Gisleno R. de Alencar Silva Júnior[1]

[1] Instituto Federal do Ceará (IFCE)

gislenojr@alu.ufc.br

Palavras-chave: Segurança, Automação, Internet das Coisas (IoT), Monitoramento, Alarme.

1. Introdução

A Segurança residencial e industrial é uma das principais preocupações da sociedade, sistemas de alarme são amplamente utilizados para monitorar acessos não autorizados e prevenir invasões. Com base nisso, durante a capacitação do Embarcatech veio a ideia de usar a placa BitDogLab de forma que pudesse ter funções de monitoramento e de resposta para com o usuario em caso de ”invasões”em uma maquete de casa.

Este projeto apresenta o desenvolvimento de um sistema de alarme baseado na Raspberry Pi Pico W (BitDogLab). O sistema utiliza sensores magnéticos para detectar abertura de portas ou janelas numa maquete, os buzzers contidos na placa estão programados para alertar usuários sobre uma invasão e o display OLED da placa para exibir mensagens de status. O controle do alarme é realizado por botões físicos A e B e pelo botão do joystick integrado, permitindo ativar e desativar o sistema conforme necessário.

2. Objetivos

O objetivo deste projeto é desenvolver um sistema de alarme seguro e modularizado, utilizando a Raspberry Pi Pico W [Raspberry Pi Ltd. 2024], capaz de:

- Detectar aberturas de portas e janelas em uma maquete por meio de sensores magnéticos.
- Acionar um alarme sonoro e visual quando um evento for detectado.
- Exibir mensagens no display OLED para indicar o status do sistema.
- Permitir que o usuário ative ou desative o sistema de alarme por meio do botão no joystick.
- Modularizar o código para facilitar manutenção e futuras expansões.

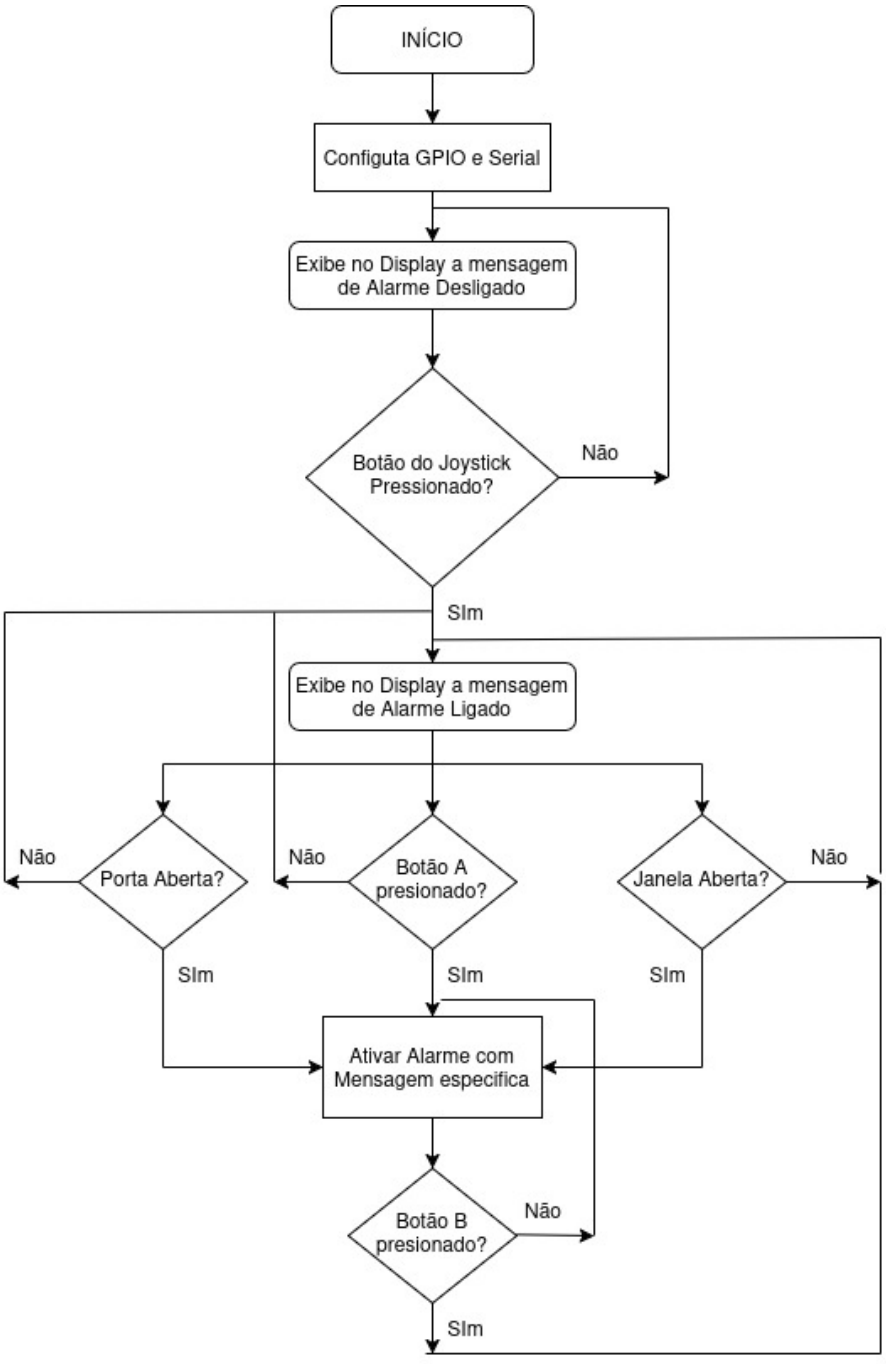
3. Justificativa

O uso de sistemas de alarme é essencial para garantir a segurança de residências e estabelecimentos comerciais. Muitas soluções disponíveis no mercado são caras e dependem de infraestrutura proprietária.

A Raspberry Pi Pico W foi escolhida devido às suas características:

Baixo consumo de energia, ideal para sistemas embarcados. Wi-Fi integrado, permitindo futuras integrações com a Internet das Coisas (IoT) implementação futura substituindo o display por uma aplicação web. A modularização do código permite que o sistema seja expandido no futuro, por exemplo, adicionando conectividade remota para monitoramento via celular, tornando assim um uso de (Iot) residencial.

4. Fluxograma



5. Código

O código foi desenvolvido em **C**, utilizando a biblioteca **pico/stdlib.h** para comunicação com os periféricos e gerenciamento de entrada/saída (GPIOs). Além disso, outras bibliotecas essenciais foram empregadas para garantir a funcionalidade do sistema.

A **hardware/gpio.h** foi utilizada para a configuração e manipulação dos pinos GPIO, permitindo a leitura dos sensores magnéticos e o controle dos botões. A **hardware/pwm.h** possibilitou a geração de sinais PWM para o acionamento dos buzzers, garantindo um alerta sonoro eficiente. A biblioteca **hardware/i2c.h** foi fundamental para a

comunicação com o **display OLED via protocolo I²C**,[Basics] possibilitando a exibição do status do alarme. Já a **hardware/pio.h** permitiu a utilização eficiente dos periféricos programáveis para o controle da matriz de LEDs WS2812B.

A modularização do código garante[Igino et al. 2023] **flexibilidade e facilidade de manutenção**, separando as funcionalidades de **display, buzzer e matriz de LEDs**. Além disso, foram utilizadas as bibliotecas **ws2818b.pio.h** e **ssd1306.h**, tomando como base os códigos implementados no **GitHub da BitDogLab**. Nas mentorias foi perguntado sobre o uso dessas implementações e permitido desde que devidamente referenciadas no projeto. Dessa forma, segue a referência: [BitDogLab 2024]. Segue o mpa de GPIOS, tive como base o Banco de Informações de Hardware (BIH) [(BIH) 2024].

Componente	Descrição	GPIO
Botão A	Ativa o alarme manualmente	GPIO5
Botão B	Desativa o alarme	GPIO6
Botão do Joystick	Liga/desliga o alarme	GPIO22
Sensor Magnético 1	Detecta abertura de porta/janela	GPIO16
Sensor Magnético 2	Detecta abertura de porta/janela	GPIO18
Buzzer A	Alerta sonoro do alarme	GPIO21
Buzzer B	Alerta sonoro do alarme	GPIO10
Matriz de LEDs WS2812B	Indicação visual do alarme	GPIO7
Display OLED (I²C SDA)	Comunicação com o display	GPIO14
Display OLED (I²C SCL)	Comunicação com o display	GPIO15

5.1. Codigo principal do sistema de alarme

As outras entidades como buzzer.h, display.h e led-matrix, estão numa pasta chamada ProjetoFinal-Embarcatech-VersãoFinal e foram zipadas junto a este pdf, mas deixo aqui a codificação principal onde está a main, tendo em vista que o codigo está modularizado.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "buzzer.h"
#include "display.h"
#include "led_matrix.h"

#define BUTTON_A 5
#define BUTTON_B 6
#define JOYSTICK_BUTTON 22
#define SENSOR_PIN_1 16
#define SENSOR_PIN_2 18

// Variaveis globais para controle do alarme
bool alert_active = false;   // Indica se o alarme esta tocando
bool alarm_enabled = false;  // Indica se o sistema de alarme esta
    ativado

int main() {
    stdio_init_all();
    init_display();
    init_led_matrix();
```

```
21      init_buzzer();
22
23      gpio_init(BUTTON_A);
24      gpio_init(BUTTON_B);
25      gpio_init(JOYSTICK_BUTTON);
26      gpio_set_dir(BUTTON_A, GPIO_IN);
27      gpio_set_dir(BUTTON_B, GPIO_IN);
28      gpio_set_dir(JOYSTICK_BUTTON, GPIO_IN);
29      gpio_pull_up(BUTTON_A);
30      gpio_pull_up(BUTTON_B);
31      gpio_pull_up(JOYSTICK_BUTTON);
32
33      gpio_init(SENSOR_PIN_1);
34      gpio_set_dir(SENSOR_PIN_1, GPIO_IN);
35      gpio_pull_up(SENSOR_PIN_1);
36
37      gpio_init(SENSOR_PIN_2);
38      gpio_set_dir(SENSOR_PIN_2, GPIO_IN);
39      gpio_pull_up(SENSOR_PIN_2);
40
41     // Assim que o codigo sobe, esta mensagem aparece no Display
42      display_message("ALERTA", "Alarme", "DESLIGADO");
43
44      while (true) {
45          // Alterna entre ligar/desligar o alarme ao pressionar o botao
                do joystick
46          if (gpio_get(JOYSTICK_BUTTON) == 0) {
47              sleep_ms(300);                    // Debounce para evitar
                    leituras rapidas
48              alarm_enabled = !alarm_enabled;  // Alterna o estado do
                    alarme
49              if (alarm_enabled) {
50                  display_message("ALERTA", "Alarme", "LIGADO");
51              } else {
52                  display_message("ALERTA", "Alarme", "DESLIGADO");
53              }
54              while (gpio_get(JOYSTICK_BUTTON) == 0);  // Aguarda soltar
                    o botao
55          }
56
57          // Se o alarme NAO estiver ativado, ignora os sensores e o
                Botao A
58          if (!alarm_enabled) {
59              continue;
60          }
61
62          // Ativa o alarme ao apertar o botao A ou se um sensor detectar
                 algo
63          if (gpio_get(BUTTON_A) == 0 || gpio_get(SENSOR_PIN_1) == 1 ||
                gpio_get(SENSOR_PIN_2) == 1) {
64              alert_active = true; // Ativa o alarme
65
66              if (gpio_get(BUTTON_A) == 0) {
67                  display_message("AVISO", "Alarme␣ativado", "manualmente
                        ");
68              } else if (gpio_get(SENSOR_PIN_1) == 1) {
```

```
                display_message("AVISO", "Invasor", "na porta");
            } else if (gpio_get(SENSOR_PIN_2) == 1) {
                display_message("AVISO", "Invasor", "na janela");
            }
        }

        // Enquanto o alarme estiver ativo, os LEDs do triangulo acendem e os buzzers fazem som ao mesmo tempo
        while (alert_active) {
            draw_triangle_pattern();
            play_buzzers(); // Ativa os dois buzzers ao mesmo tempo

            sleep_ms(500);  // Tempo do efeito de alerta

            npClear();      // Apaga LEDs
            stop_buzzers();

            sleep_ms(500);  // Tempo do efeito de alerta

            // Se o botao B for pressionado, desativa o alarme
            if (gpio_get(BUTTON_B) == 0) {
                alert_active = false;
                clear_display();  // Limpa a tela
                npClear();        // Apaga os LEDs
                stop_buzzers();   // Desativa os dois buzzers ao mesmo tempo
                display_message("ALERTA", "Alarme", "LIGADO"); // Exibe que o alarme ainda esta ligado
                break;                                         // Sai do loop while(alert_active)
            }
        }
    }
}
```

indent:

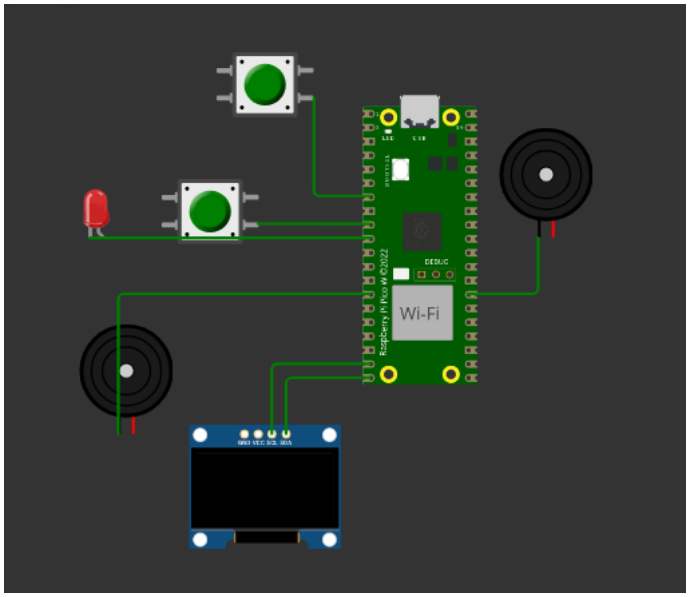
6. Esquemático

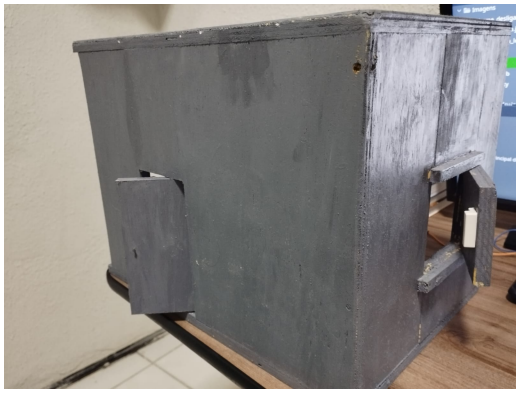
Tendo em vista que o projeto foi desenvolvido diretamente na placa **BitDogLab (Raspberry Pi Pico W)** e não em um simulador, o esquemático gerado no **Wokwi** [Wokwi]possui algumas limitações. Dessa forma, ele serve apenas como uma **representação base** para o que de fato foi implementado fisicamente.

Apesar das limitações do simulador, o esquemático permite visualizar um forma de referência para o projeto.O esquemático do projeto encontra-se no simulador wokwi, imagem abaixo:

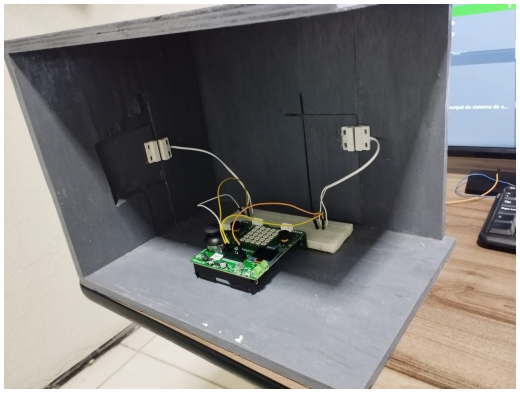
Para representar corretamente a implementação física do sistema de alarme no simulador Wokwi, os componentes foram conectados aos pinos GPIO corretos da **Raspberry Pi Pico W**. Abaixo estão os ajustes realizados:

- **LED Indicador:** O único LED presente no esquema foi conectado ao pino **GP7**, representando a matriz de LEDs WS2812B utilizada no projeto real.
- **Botões:** Os botões físicos foram mapeados corretamente:
  - **Botão A:** Conectado ao pino **GP5**, responsável por ativar o alarme manualmente.
  - **Botão B:** Conectado ao pino **GP6**, utilizado para desativar o alarme.
- **Display OLED:** Configurado para comunicação via protocolo **I²C**, com as conexões:
  - **SDA** no pino **GP14**
  - **SCL** no pino **GP15**
- **Buzzers:** Os buzzers responsáveis pelo alerta sonoro foram conectados corretamente:
  - **Buzzer A:** Conectado ao pino **GP21**.
  - **Buzzer B:** Conectado ao pino **GP10**.





(a) Maquete (visão da frente).



(b) Maquete (visão por dentro).

7. Resultados

Após a implementação e testes do sistema de alarme, os seguintes resultados foram observados:

- O sistema inicia corretamente com o alarme desativado. (Mostrando no Display como segue a imagem)

(a) Mensagem inicial no display.

(b) Alarme Ativado.

- A ativação e desativação do sistema funcionam conforme esperado, utilizando o joystick.
- Quando o sistema está ativado, sensores magnéticos detectam abertura de portas/-janelas, acionando o alarme sonoro e visual.
- Os LEDs da matriz exibem um padrão triangular piscante enquanto o alarme está ativo.
- O botão B desativa o alarme imediatamente, interrompendo o som e apagando os LEDs.
- As mensagens no display OLED são atualizadas corretamente para indicar o estado do sistema.

Para acessar vídeos demonstrando o funcionamento do sistema, acesse o seguinte link:

https://drive.google.com/drive/u/1/folders/1We53Q2xu9bDqHoA7yiC89tpYiGL0W03O.

Referências

Basics, C. How to set up a keypad on an arduino.

(BIH) (2024). Banco de informações de hardware (bih). Disponível em: https://docs.google.com/document/d/13-68OqiU7ISE8U2KPRUXT2ISeBl3WPhXjGDFH52eWlU/edit?tab=t.0.

BitDogLab (2024). Bitdoglab - repositório oficial de códigos. GitHub Repository. Disponível em: https://github.com/BitDogLab/BitDogLab-C.

Igino, W. P. et al. (2023). Ensino de sistemas embarcados baseado em projeto: exemplo aplicado à robótica.

Raspberry Pi Ltd. (2024). Raspberry pi pico w datasheet: An rp2040-based microcontroller board with wireless.

Wokwi. Wokwi 7 segment display.