

# **CEPEDI - EmbarcaTech**

**Discente: Adimael Santos Da Silva**

## **Relatório do Projeto BitSOS: Transmissão de SOS em Código Morse**

<b>1. Introdução.....</b>	<b>2</b>
<b>2. Objetivo.....</b>	<b>3</b>
<b>3. Materiais e Ferramentas.....</b>	<b>4</b>
<b>4. Desenvolvimento.....</b>	<b>4</b>
4.1. Configuração do Ambiente.....	4
4.2. Desenvolvimento do Firmware.....	4
<b>5. Repositório do projeto.....</b>	<b>6</b>
<b>6. Testes e Resultados.....</b>	<b>6</b>
<b>7. Conclusão.....</b>	<b>8</b>
<b>8. Referências.....</b>	<b>9</b>

## **1. Introdução**

Este relatório documenta o desenvolvimento de um projeto em linguagem C utilizando a plataforma de simulação Wokwi integrada ao Visual Studio Code (VSCode) e a placa de desenvolvimento BitDogLab. O objetivo do projeto foi criar um sistema capaz de transmitir o sinal “SOS” em código Morse, utilizando um LED externo conectado a uma das GPIOs da placa. O projeto foi desenvolvido no VSCode com integração ao simulador Wokwi.

O trabalho visa consolidar os conceitos de programação embarcada e uso de sistemas de simulação, promovendo a compreensão de GPIOs, controle de temporização e modularização de código em sistemas embarcados.

## 2. Objetivo

- 2.1. Configurar o ambiente de desenvolvimento utilizando o Visual Studio Code e integrar o simulador Wokwi.
- 2.2. Desenvolver um firmware que implemente o sinal “SOS” em código Morse com as seguintes características:
  - Um ponto (“.”) representado por um LED aceso por 0,2 segundos.
  - Um traço (“-”) representado por um LED aceso por 0,8 segundos.
  - Intervalo entre pontos e traços: 0,125 segundos.
  - Intervalo entre letras: 0,25 segundos.
  - Intervalo entre ciclos de “SOS”: 3 segundos.
- 2.3. Modularizar o programa utilizando funções.
- 2.4. Testar e simular o funcionamento do sistema no Wokwi.

### 3. Materiais e Ferramentas

- 3.1. Placa de desenvolvimento BitDogLab (Compatível com Raspberry Pi Pico).
- 3.2. LED externo.
- 3.3. Resistor de 1 kΩ.
- 3.4. Cabos de conexão
- 3.5. Computador com:
  - Visual Studio Code instalado
  - Extensão de suporte para C/C++ e Wokwi.
  - SDK do Raspberry Pi Pico.
  - Simulador Wokwi.

### 4. Desenvolvimento

#### 4.1. Configuração do Ambiente

1. Instalação do Visual Studio Code:
  - Baixado e instalado o Visual Studio Code.
  - Adicionadas extensões de suporte a C/C++ e Wokwi
2. Configuração do SDK do Raspberry Pi Pico:
  - Instalado o SDK do Raspberry Pi Pico para compilação do código.
3. Integração com Wokwi:
  - Configurado o simulador Wokwi para simulação do projeto.

#### 4.2. Desenvolvimento do Firmware

O código foi desenvolvido em linguagem C, com as seguintes funções principais:

- **Função `led_on()`**: Controla o LED, acendendo-o por um intervalo específico e depois apagando-o.
- **Função `enviar_sos()`**: Implementa o padrão SOS em código Morse (“...---...”) com os tempos especificados no enunciado.
- **Função `main()`**: Configura o GPIO e entra em um loop infinito para enviar o sinal SOS continuamente.

## Código Fonte:

```
#include <stdio.h>
#include "pico/stdlib.h" //biblioteca SDK padrão do PICO - funcionalidades
para programação básica.

// Define o pino do LED
#define LED_PIN 13

// Define os tempos em milissegundos
#define DOT_TIME 200      // Duração de um ponto (.)
#define DASH_TIME 800    // Duração de um traço (-)
#define GAP_TIME 125     // Intervalo entre pontos e traços
#define LETTER_GAP_TIME 250 // Intervalo entre letras
#define CYCLE_GAP_TIME 3000 // Intervalo entre ciclos de SOS

// Função para acender o LED por um determinado tempo
void led_on(int duration) {
    gpio_put(LED_PIN, true); // Liga o LED
    sleep_ms(duration);      // Espera o tempo especificado
    gpio_put(LED_PIN, false); // Desliga o LED
}

// Função para enviar o sinal SOS
void enviar_sos() {
    // Sinal de SOS: ... --- ...
    printf("Enviando sinal SOS...\n");

    // Enviar três pontos (.)
    for (int i = 0; i < 3; i++) {
        led_on(DOT_TIME);
        printf("."); // Exibe um ponto no terminal
        fflush(stdout); // Garante que o texto seja exibido imediatamente
        sleep_ms(GAP_TIME); // Intervalo entre os pontos
    }

    sleep_ms(LETTER_GAP_TIME); // Intervalo entre letras
    printf(" "); // Espaço entre letras

    // Enviar três traços (-)
    for (int i = 0; i < 3; i++) {
        led_on(DASH_TIME);
        printf("-"); // Exibe um traço no terminal
        fflush(stdout);
        sleep_ms(GAP_TIME); // Intervalo entre os traços
    }
}
```

```

sleep_ms(LETTER_GAP_TIME); // Intervalo entre Letras
printf(" "); // Espaço entre Letras

// Enviar três pontos (.)
for (int i = 0; i < 3; i++) {
    led_on(DOT_TIME);
    printf("."); // Exibe um ponto no terminal
    fflush(stdout);
    sleep_ms(GAP_TIME); // Intervalo entre os pontos
}

printf("\n\n"); // Nova linha após completar o SOS
}

int main() {
    // Inicializa o GPIO do LED
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    // Inicializa o terminal para exibir mensagens
    stdio_init_all();

    while (true) {
        enviar_sos(); // Envia o sinal SOS
        sleep_ms(CYCLE_GAP_TIME); // Espera antes de reiniciar o ciclo
    }

    return 0;
}

```

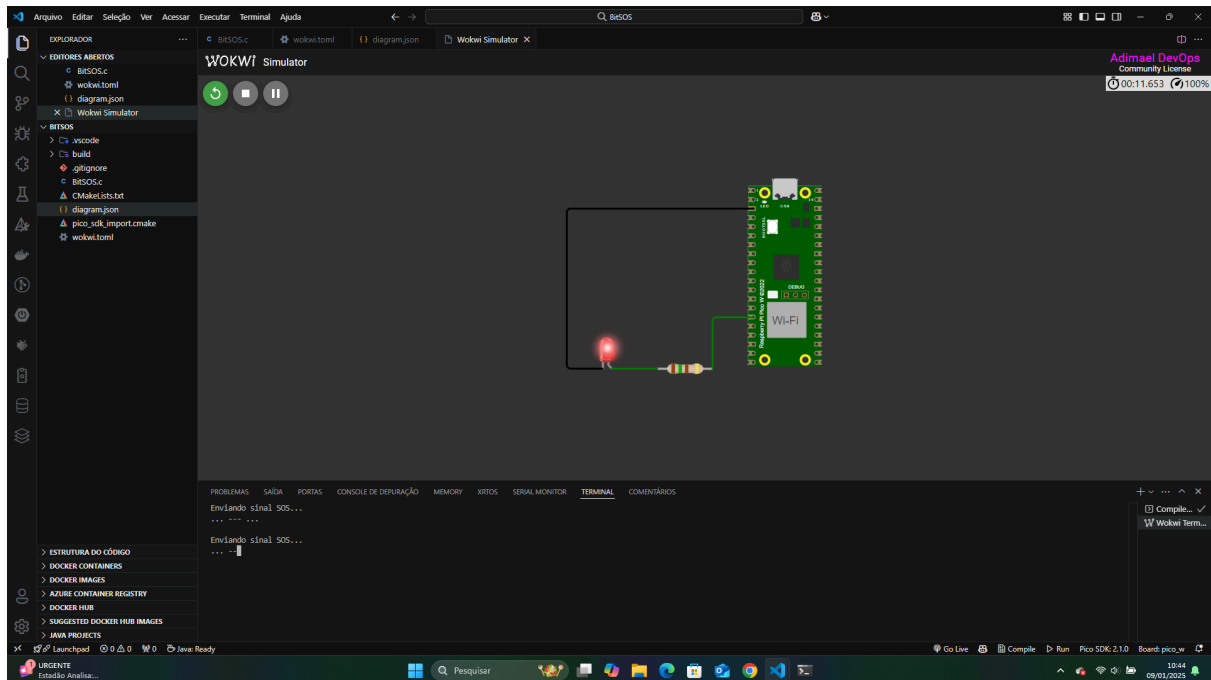
## 5. Repositório do projeto

O código completo do projeto está disponível no GitHub: [Repositório do Projeto](https://github.com/adimael/BitSOS.git) <<https://github.com/adimael/BitSOS.git>>.

## 6. Testes e Resultados

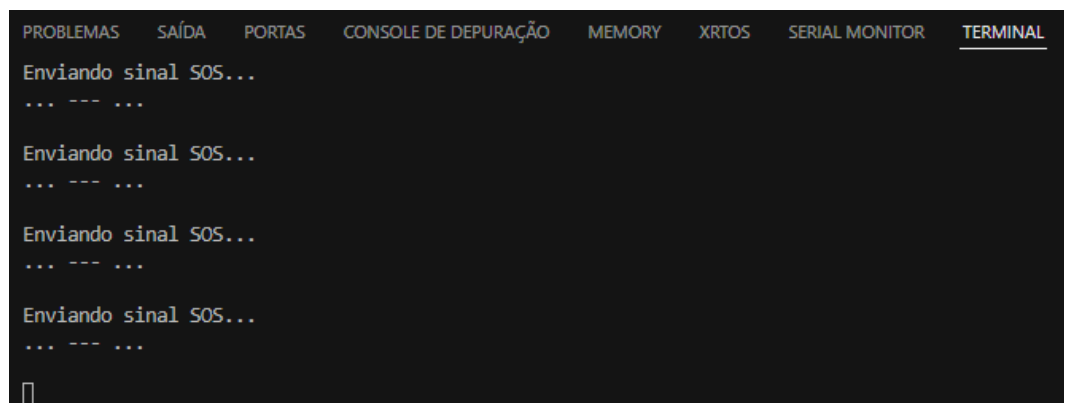
### 6.1. Simulação no Wokwi

- A simulação foi realizada no Wokwi e apresentou o comportamento esperado.
- O LED piscou conforme o padrão SOS: três pontos, três traços, três pontos.



## 6.2. Mensagens no Terminal

- O terminal exibiu o padrão SOS (“...---...”) sincronizado com os pulsos do LED.



## 6.3. Intervalos

- Foram respeitados os tempos especificados no enunciado para pontos, traços, intervalos entre letras e ciclos.



## **7. Conclusão**

O projeto alcançou os objetivos propostos, integrando com sucesso a placa de desenvolvimento BitDogLab ao ambiente de desenvolvimento Visual Studio Code. O sistema implementado foi capaz de transmitir o sinal “SOS” em código Morse, seguindo as especificações estabelecidas.

Este trabalho permitiu consolidar conceitos fundamentais de programação embarcada, como manipulação de GPIOs, controle de temporização e modularização de códigos, além de explorar ferramentas como o Wokwi para simulação.

## 8. Referências

RASPBERRY PI. *Raspberry Pi Pico and Pico W Documentation*. Disponível em: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html>. Acesso em: 09 jan. 2025.

TELECOM UFF. *Tutorial Código Morse*. Disponível em: [https://www.telecom.uff.br/pet/petws/downloads/tutoriais/codigo\\_morse/Tutorial\\_Codigo\\_Morse\\_2019\\_12\\_19.pdf](https://www.telecom.uff.br/pet/petws/downloads/tutoriais/codigo_morse/Tutorial_Codigo_Morse_2019_12_19.pdf). Acesso em: 09 jan. 2025.

WOKWI. *Documentação Wokwi*. Disponível em: <https://docs.wokwi.com/pt-BR/>. Acesso em: 09 jan. 2025.