**Aluno**: Moisés Nascimento Santana.

**Professor:** Otávio Alcântara de Lima Jr.

**Enunciado**

Para assegurar a assimilação dos conceitos iniciais de programação em sistemas embarcados utilizando a linguagem C e do uso da placa de desenvolvimento BitDogLab, propõe-se a seguinte tarefa.

**Objetivos**

* Adotar o **Visual Studio Code** como ambiente de desenvolvimento.
* Configurar e integrar o **Wokwi** ao Visual Studio Code.
* Utilizar a **placa de desenvolvimento BitDogLab**.

**Descrição do Projeto**:

Neste projeto, um LED externo conectado à GPIO (11, 12 ou 13) será utilizado para transmitir o sinal SOS, em código Morse, através de pulsos luminosos contínuos.

O comportamento do LED deve seguir as seguintes regras:

* Um ponto (.) é representado pelo LED aceso por 0,2 segundos.
* Um traço (-) é representado pelo LED aceso por 0,8 segundos.
* O intervalo entre pontos e traços dentro de uma mesma letra é de 0,125 segundos
* (tempo GAP).
* O intervalo entre letras deve ser de 0,25 segundos.
* Após completar o sinal SOS (três pontos, três traços, três pontos), o ciclo é reiniciado após um intervalo de 3 segundos.

**Requisitos do Projeto**:

1. O projeto deve ser simulado no Wokwi Web, bem como no Wokwi integrado ao Visual Studio Code.

2. O firmware deve estar apto a rodar na Placa de Desenvolvimento BitDogLab.

3. O programa deve obrigatoriamente utilizar pelo menos duas funções para modularizar o código.

**Relatório do Projeto: Transmissão de SOS com LED**

**1. Introdução**

Este projeto foi desenvolvido para transmitir o sinal SOS em código Morse utilizando um LED externo conectado à placa de desenvolvimento BitDogLab. O firmware foi implementado em C e utiliza as bibliotecas da Raspberry Pi Pico SDK para o controle do GPIO. A simulação foi realizada no ambiente Wokwi e no Wokwi integrado ao Visual Studio Code.

**2. Objetivos**

* Implementar a transmissão do sinal SOS (três pontos, três traços, três pontos) em um LED, seguindo o padrão do código Morse.
* Modularizar o código utilizando pelo menos duas funções.
* Garantir a compatibilidade com a placa de desenvolvimento BitDogLab.
* Simular o projeto no Wokwi e no Visual Studio Code.
* Respeitar os tempos e intervalos especificados no comportamento do LED.

**3. Metodologia**

**3.1. Regras Implementadas**

Um ponto (.) é representado pelo LED aceso por 0,2 segundos.

Um traço (-) é representado pelo LED aceso por 0,8 segundos.

O intervalo entre pontos e traços dentro de uma mesma letra é de 0,125 segundos.

O intervalo entre letras é de 0,25 segundos.

Após completar o ciclo SOS, há um intervalo de 3 segundos antes do reinício.

**3.2. Modularização**

O código foi dividido em funções para facilitar a leitura, a manutenção e a reutilização:

* piscar\_led(int duracao): Liga o LED por um tempo especificado e, em seguida, o desliga.
* transmitir\_letra(const char \*padrao): Transmite uma sequência de pontos e traços de uma letra em código Morse.

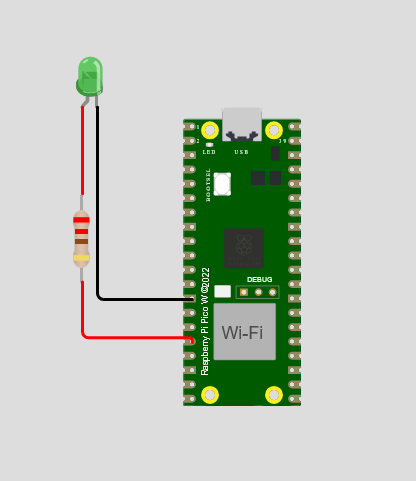
**3.3. Configuração do Hardware**

O LED foi utilizado foi o conectado ao GPIO12 da placa Bitdoglab.

**3.4. Simulação no Wokwi**

O ambiente de simulação Wokwi foi configurado conforme o esquema elétrico descrito abaixo:

* Um LED (cor: verde) conectado ao GPIO12 da placa.
* Um resistor de 220 ohms entre o LED e o pino GPIO.



*diagram.json*

{

"version": 1,

"author": "Moisés Santana",

"editor": "wokwi",

"parts": [

{ "type": "board-pi-pico-w", "id": "pico", "top": 0, "left": 0, "attrs": {} },

{

"type": "wokwi-led",

"id": "led1",

"top": -42,

"left": -82.2,

"attrs": { "color": "limegreen", "flip": "1" }

},

{

"type": "wokwi-resistor",

"id": "r1",

"top": 81.6,

"left": -96.55,

"rotate": 90,

"attrs": { "value": "220" }

}

],

"connections": [

[ "pico:GP0", "$serialMonitor:RX", "", [] ],

[ "pico:GP1", "$serialMonitor:TX", "", [] ],

[ "led1:C", "pico:GND.3", "black", [ "v0" ] ],

[ "led1:A", "r1:1", "red", [ "v0" ] ],

[ "r1:2", "pico:GP12", "red", [ "v37.2", "h73.25" ] ]

],

"dependencies": {}

}

**3.5. Código Desenvolvido**

O código desenvolvido em C utiliza a biblioteca padrão do Raspberry Pi Pico SDK para manipulação do GPIO e realizar temporizações. A lógica central do programa é baseada na transmissão do sinal SOS em código Morse utilizando um LED conectado ao GPIO12 da placa BitDogLab. Essa transmissão segue um padrão bem definido, onde cada ponto e traço do código Morse é representado por um tempo específico em que o LED permanece aceso, com intervalos adequados entre os sinais.

O programa é estruturado de maneira modular, utilizando duas funções principais. A função piscar\_led é responsável por controlar o LED, ligando-o por um período de tempo definido (duração) e desligando-o em seguida. Essa função encapsula a lógica básica de acender e apagar o LED, tornando-a reutilizável em outras partes do código.

Já a função transmitir\_letra cuida de transmitir uma sequência de sinais correspondente a uma letra em código Morse. Ela utiliza um loop para iterar pelos caracteres de uma string de padrão Morse, identificando se cada caractere é um ponto ou traço. Dependendo do caractere, a função chama piscar\_led com a duração apropriada. Entre cada sinal em uma mesma letra, há uma pequena pausa controlada por um sleep\_ms que respeita o intervalo necessário para o código Morse.

O programa principal, no main, é responsável por configurar o GPIO, definindo o pino do LED como saída. Em seguida, um loop infinito transmite repetidamente o sinal SOS, utilizando a função transmitir\_letra para cada letra do código Morse. Entre cada letra e após o ciclo completo, são introduzidas pausas para garantir os intervalos corretos.

#include "pico/stdlib.h"

// Definições de macros

#define PINO\_LED 12 // Pino conectado ao LED

#define TEMPO\_PONTO 200 // Duração de um ponto (em milissegundos)

#define TEMPO\_TRACO 800 // Duração de um traço (em milissegundos)

#define TEMPO\_GAP 125 // Intervalo entre pontos e traços (em milissegundos)

#define TEMPO\_ENTRE\_LETRAS 250 // Intervalo entre letras (em milissegundos)

#define TEMPO\_ENTRE\_CICLOS 3000 // Intervalo entre ciclos completos (em milissegundos)

// Função para ligar o LED por um tempo especificado

void piscar\_led(int duracao) {

gpio\_put(PINO\_LED, 1); // Liga o LED

sleep\_ms(duracao); // Aguarda o tempo especificado

gpio\_put(PINO\_LED, 0); // Desliga o LED

}

// Função para transmitir uma letra em código Morse

void transmitir\_letra(const char \*padrao) {

while (\*padrao) {

if (\*padrao == '.') {

piscar\_led(TEMPO\_PONTO); // Transmite um ponto

} else if (\*padrao == '-') {

piscar\_led(TEMPO\_TRACO); // Transmite um traço

}

padrao++;

if (\*padrao) sleep\_ms(TEMPO\_GAP); // Intervalo entre sinais na mesma letra

}

}

int main() {

gpio\_init(PINO\_LED);

gpio\_set\_dir(PINO\_LED, GPIO\_OUT);

while (true) {

// Transmite SOS

transmitir\_letra("..."); // S: três pontos

sleep\_ms(TEMPO\_ENTRE\_LETRAS);

transmitir\_letra("---"); // O: três traços

sleep\_ms(TEMPO\_ENTRE\_LETRAS);

transmitir\_letra("..."); // S: três pontos

// Intervalo entre ciclos

sleep\_ms(TEMPO\_ENTRE\_CICLOS);

}

return 0;

}

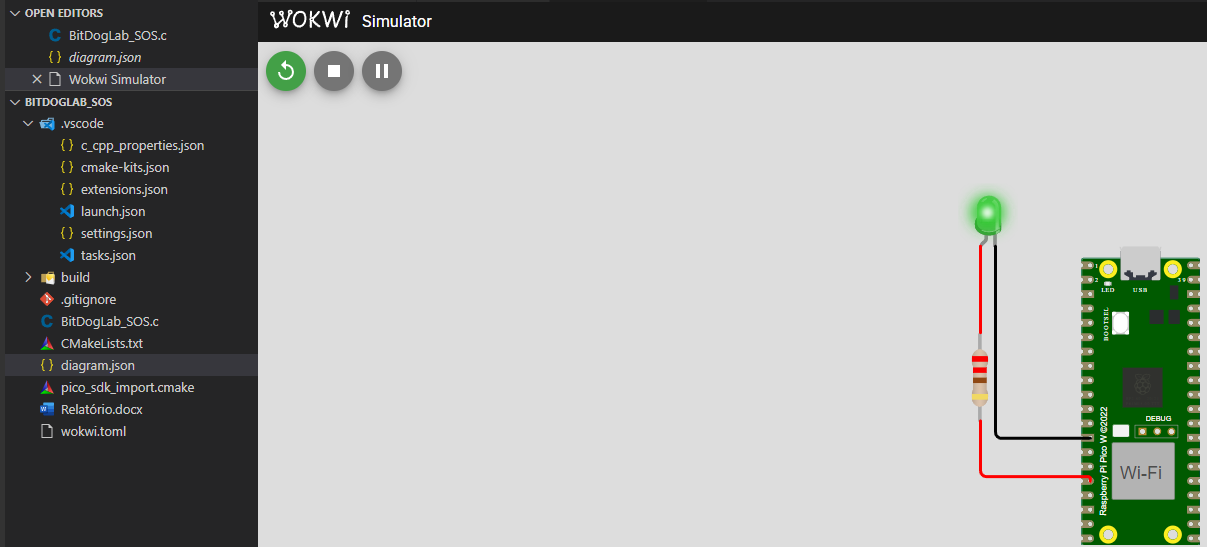
**4. Resultados**

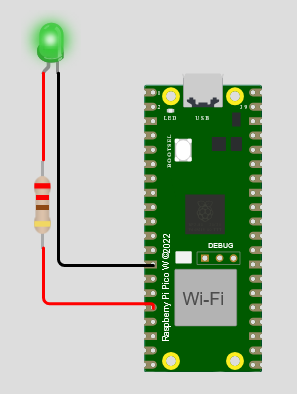
**4.1. Comportamento do Sistema**

* O LED transmite o sinal SOS corretamente em código Morse.
* Os tempos e intervalos foram respeitados conforme especificado.
* O ciclo é reiniciado após um intervalo de 3 segundos.



**4.2. Simulação no Wokwi**

A simulação no Wokwi foi realizada com sucesso, confirmando o funcionamento correto do firmware e do circuito elétrico. A simulação no Visual Studio Code também demonstrou a execução adequada do código.



**4.3. Modularização**

O uso de funções para modularizar o código facilitou a reutilização e a organização, atendendo aos requisitos do projeto.

**5. Conclusão**

O projeto foi concluído com êxito, cumprindo plenamente os requisitos funcionais e técnicos estabelecidos. A transmissão do sinal SOS foi implementada de maneira precisa e contínua, com o comportamento do LED obedecendo fielmente aos tempos e intervalos definidos. O firmware desenvolvido demonstrou compatibilidade total com a placa BitDogLab, permitindo sua execução tanto em hardware real quanto em ambiente de simulação no Wokwi. Além disso, a modularização do código contribuiu significativamente para a organização do programa, facilitando sua manutenção e ampliando sua escalabilidade para futuras adaptações.

**6. Próximos Passos**

O projeto pode ser expandido para permitir a escolha de diferentes padrões de mensagens, oferecendo mais flexibilidade para o envio de códigos Morse personalizados. Uma funcionalidade adicional interessante seria integrar o controle do LED por meio de comunicação serial, permitindo que comandos sejam enviados diretamente ao dispositivo para ajustar o padrão de sinalização.

Outra melhoria seria a utilização de interrupções para gerenciar o acionamento do LED, o que tornaria o código mais eficiente e responsivo. Além disso, a implementação do uso do display OLED no sistema possibilitaria a exibição em tempo real dos códigos Morse transmitidos, proporcionando um recurso visual útil para acompanhamento e validação das mensagens enviadas.