

SOS – Código Morse

Programa residência em TIC 37

Patrick de Sousa Queiroz

11 de janeiro de 2025



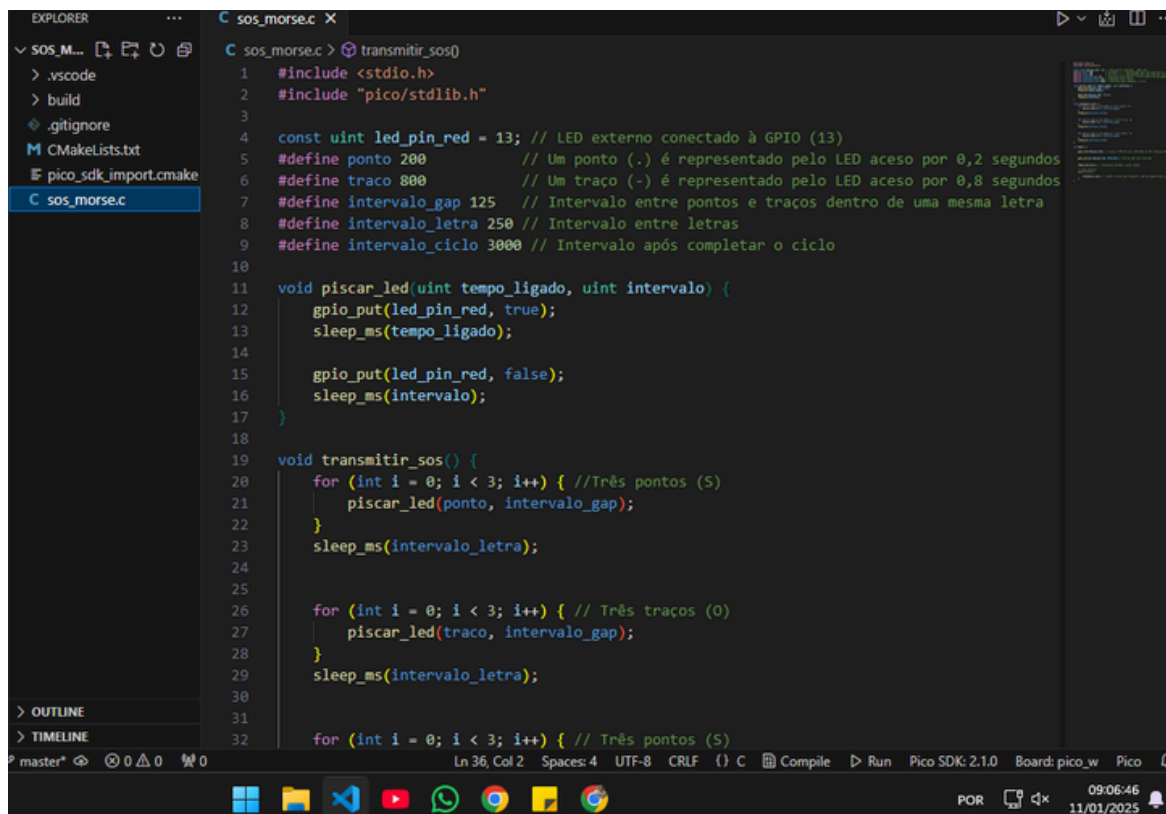
Pra garantir que os conceitos iniciais de programação em sistemas embarcados usando C e a placa BitDogLab, fossem fixados, fiz dois códigos: um no VSCode e outro no Wokwi.

O primeiro foi desenvolvido pra rodar no ambiente de desenvolvimento real, usando a placa BitDogLab, enquanto o segundo é uma simulação online, no Wokwi.

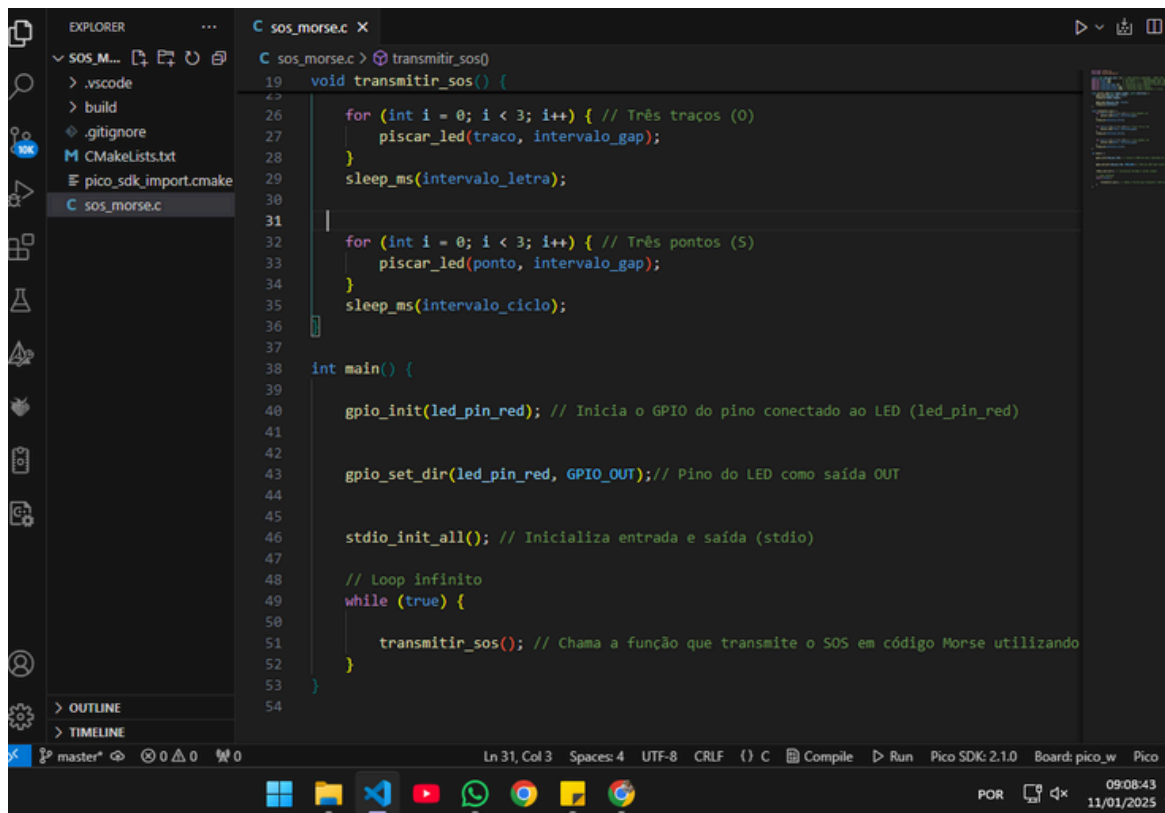
Explicarei detalhadamente como cada um funciona e quais são as diferenças no processo de execução, destacando as configurações e ajustes feitos pra garantir que ambos cumpram o mesmo objetivo do projeto.

Comecei o projeto criando um novo projeto C/C++ na extensão do Raspberry Pi, onde adicionei as bibliotecas `pico/stdlib.h` e `stdio.h` para que tudo funcionasse direitinho. Essas bibliotecas são essenciais para o código rodar e se comunicar com a placa.

Pra deixar o código mais organizado, criei duas funções que ajudaram a separar as tarefas, tornando tudo mais fácil de entender e editar. Isso ajudou bastante a manter o projeto mais limpo e fácil de modificar.



```
1 #include <stdio.h>
2 #include "pico/stdlib.h"
3
4 const uint led_pin_red = 13; // LED externo conectado à GPIO (13)
5 #define ponto 200 // Um ponto (.) é representado pelo LED aceso por 0,2 segundos
6 #define traco 800 // Um traço (-) é representado pelo LED aceso por 0,8 segundos
7 #define intervalo_gap 125 // Intervalo entre pontos e traços dentro de uma mesma letra
8 #define intervalo_letra 250 // Intervalo entre letras
9 #define intervalo_ciclo 3000 // Intervalo após completar o ciclo
10
11 void piscar_led(uint tempo_ligado, uint intervalo) {
12     gpio_put(led_pin_red, true);
13     sleep_ms(tempo_ligado);
14
15     gpio_put(led_pin_red, false);
16     sleep_ms(intervalo);
17 }
18
19 void transmitir_sos() {
20     for (int i = 0; i < 3; i++) { // Três pontos (S)
21         piscar_led(ponto, intervalo_gap);
22     }
23     sleep_ms(intervalo_letra);
24
25     for (int i = 0; i < 3; i++) { // Três traços (O)
26         piscar_led(traco, intervalo_gap);
27     }
28     sleep_ms(intervalo_letra);
29
30     for (int i = 0; i < 3; i++) { // Três pontos (S)
```



```
19 void transmitir_sos() {
20     for (int i = 0; i < 3; i++) { // Três traços (0)
21         piscar_led(traco, intervalo_gap);
22     }
23     sleep_ms(intervalo_letra);
24
25     for (int i = 0; i < 3; i++) { // Três pontos (S)
26         piscar_led(ponto, intervalo_gap);
27     }
28     sleep_ms(intervalo_ciclo);
29
30     int main() {
31         gpio_init(led_pin_red); // Inicia o GPIO do pino conectado ao LED (led_pin_red)
32
33         gpio_set_dir(led_pin_red, GPIO_OUT); // Pino do LED como saída OUT
34
35         stdio_init_all(); // Inicializa entrada e saída (stdio)
36
37         // Loop infinito
38         while (true) {
39             transmitir_sos(); // Chama a função que transmite o SOS em código Morse utilizando
40         }
41     }
42 }
```

Depois de testar tudo, compilei o código e enviei pra placa BitDogLab. Quando a placa rodou o código, pude ver o LED piscando em código Morse, como era o objetivo. Abaixo, tem uma foto que mostra o LED piscando de acordo com o código Morse.



Esse processo mostrou que o código, a placa e o Morse estão funcionando bem juntos. O projeto foi concluído com sucesso e agora o LED transmite a mensagem em código Morse certinho.

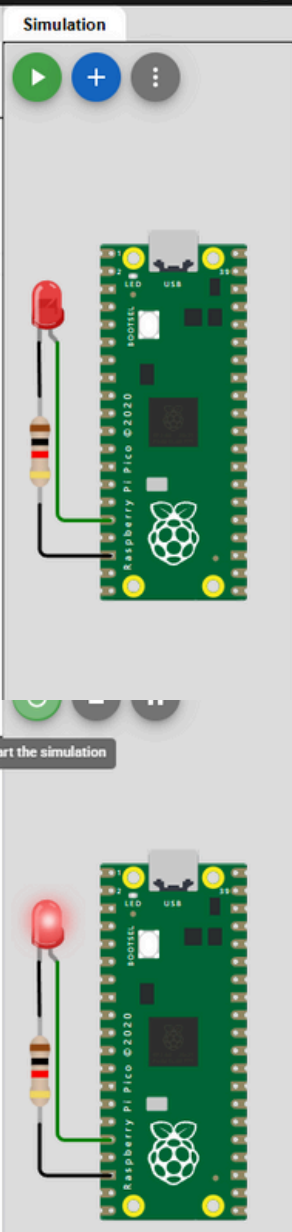
No Wokwi, o processo foi bem parecido com o realizado na placa física. Comecei criando o projeto em C/C++ e usando as mesmas bibliotecas essenciais: `hardware/clocks.h` para ajustar o clock e `pico/stdlib.h` para controlar os recursos da placa.

Defini o pino do LED como GPIO 12, já que é onde o LED externo estava conectado, os tempos e intervalos do código Morse. Criei duas funções: uma para controlar o piscar do LED e outra para transmitir a mensagem "SOS" em código Morse.

Na função que transmite o "SOS", o LED pisca três pontos (S), seguidos por três traços (O) e mais três pontos (S), com os intervalos necessários entre cada parte. A imagem abaixo mostra o comportamento do LED piscando conforme o código Morse, como esperado.

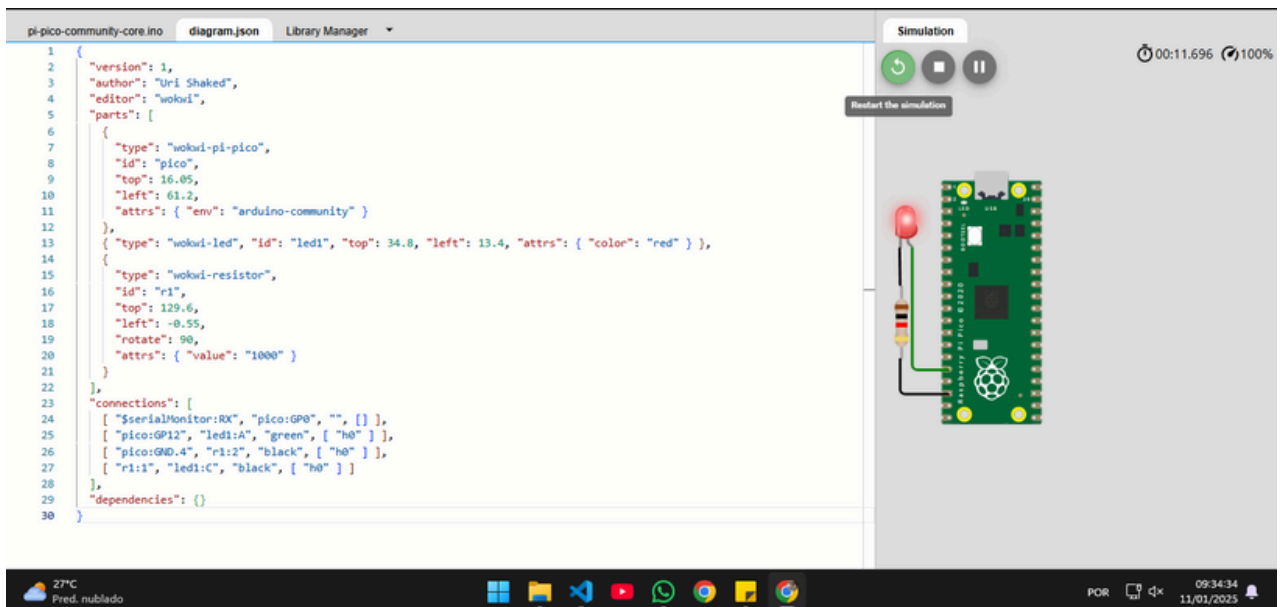
```
pi-pico-community-core.ino  diagram.json  Library Manager  Simulation

1  #include "hardware/clocks.h" // Biblioteca para ajustar o clock
2  #include "pico/stdlib.h"
3
4  const uint led_pin_red = 12; // LED externo conectado à GPIO (12)
5  #define ponto 200           // Um ponto (.) é representado pelo LED aceso por 0,2 segundos
6  #define traco 800           // Um traço (-) é representado pelo LED aceso por 0,8 segundos
7  #define intervalo_gap 125   // Intervalo entre pontos e traços dentro de uma mesma letra
8  #define intervalo_letra 250 // Intervalo entre letras
9  #define intervalo_ciclo 3000 // Intervalo após completar o ciclo
10
11 // Função para controlar o LED
12 void piscar_led(uint tempo_ligado, uint intervalo) {
13     gpio_put(led_pin_red, true); // Liga o LED
14     sleep_ms(tempo_ligado);       // Mantém ligado por "tempo_ligado" ms
15
16     gpio_put(led_pin_red, false); // Desliga o LED
17     sleep_ms(intervalo);          // Pausa por "intervalo" ms
18 }
19
20 // Função para transmitir "SOS" em código Morse
21 void transmitir_sos() {
22     // Três pontos (S)
23     for (int i = 0; i < 3; i++) {
24         piscar_led(ponto, intervalo_gap);
25     }
26     sleep_ms(intervalo_letra);
27
28     // Três traços (O)
29     for (int i = 0; i < 3; i++) {
30         piscar_led(traco, intervalo_gap);
31     }
32     sleep_ms(intervalo_letra);
33
34     // Três pontos (S)
35     for (int i = 0; i < 3; i++) {
36         piscar_led(ponto, intervalo_gap);
37     }
38     sleep_ms(intervalo_ciclo);
39 }
40
41 int main() {
42     set_sys_clock_khz(125000, true); // Configura o clock do sistema para 125 MHz
43
44     gpio_init(led_pin_red);           // Inicializa o GPIO para o pino do LED
45     gpio_set_dir(led_pin_red, GPIO_OUT); // Configura o pino do LED como saída
46
47     // Loop infinito
48     while (true) {
49         transmitir_sos(); // Transmite o SOS continuamente
50     }
51 }
```

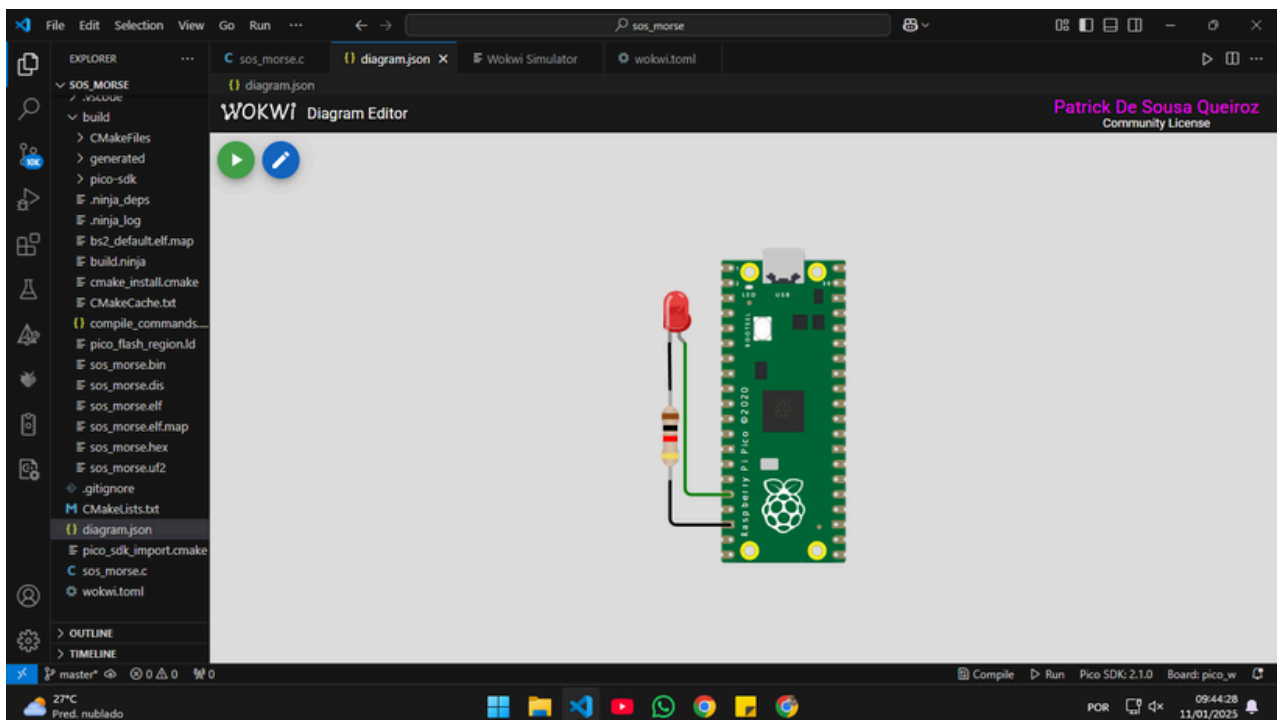


Esse processo confirmou que o projeto funciona bem tanto no ambiente físico quanto na simulação online, com o LED transmitindo corretamente o código Morse.

O diagrama foi montado diretamente no Wokwi e, depois de rodar a simulação, deu tudo certo, confirmando que o funcionamento estava como esperado.



Wokwi no VSCode



Links:

GitHub: https://github.com/queirozPatrick/sos_morse
Wokwi: <https://wokwi.com/projects/419783616669222913>
LinkedIn: <https://www.linkedin.com/in/patricksq/>