Anexo A1 – Piscar LED e Ler Botão com Superloop

# Descrição

Este exemplo demonstra a estrutura básica de um sistema embarcado com superloop, onde tarefas são executadas em sequência fixa e contínua, sem interrupções nem temporizadores.

# Objetivo

- Piscar um LED (GPIO 12) com um período fixo  
- Verificar continuamente o estado de um botão (GPIO 5)  
- Exibir o status do botão no terminal via USB

# Requisitos

- Placa BitDogLab (RP2040W)  
- SDK oficial do RP2040 instalado no VSCode  
- LED conectado ao GPIO 12  
- Botão conectado ao GPIO 5 com resistor de pull-up

# Código-fonte (main.c)

#include <stdio.h>  
#include "pico/stdlib.h"  
  
#define LED\_PIN 12  
#define BUTTON\_PIN 5  
  
int main() {  
 stdio\_init\_all();  
  
 gpio\_init(LED\_PIN);  
 gpio\_set\_dir(LED\_PIN, GPIO\_OUT);  
  
 gpio\_init(BUTTON\_PIN);  
 gpio\_set\_dir(BUTTON\_PIN, GPIO\_IN);  
 gpio\_pull\_up(BUTTON\_PIN);  
  
 while (true) {  
 gpio\_put(LED\_PIN, 1);  
 sleep\_ms(500);  
 gpio\_put(LED\_PIN, 0);  
 sleep\_ms(500);  
  
 bool pressionado = !gpio\_get(BUTTON\_PIN);  
 if (pressionado) {  
 printf("Botão pressionado!\n");  
 } else {  
 printf("Botão não pressionado\n");  
 }  
 }  
 return 0;  
}

# CMakeLists.txt (mínimo funcional)

cmake\_minimum\_required(VERSION 3.13)  
include(pico\_sdk\_import.cmake)  
  
project(superloop\_led\_button C CXX ASM)  
pico\_sdk\_init()  
  
add\_executable(superloop  
 main.c  
)  
  
target\_link\_libraries(superloop pico\_stdlib)  
pico\_enable\_stdio\_usb(superloop 1)  
pico\_enable\_stdio\_uart(superloop 0)  
  
pico\_add\_extra\_outputs(superloop)

# Observações Práticas

- A leitura do botão ocorre após o ciclo do LED, o que gera latência de resposta.  
- Modelo adequado para sistemas simples e previsíveis.  
- Pouco indicado para sistemas que exigem resposta imediata a eventos externos.