Análise Detalhada do Programa PIO: Piscar 3 LEDs com Frequências Diferentes

# 1. Introdução

Este documento analisa detalhadamente um projeto desenvolvido para o Raspberry Pi Pico, utilizando o subsistema PIO (Programmable I/O) para piscar três LEDs conectados aos pinos GPIO 11, 12 e 13, com frequências distintas (1Hz, 2Hz e 3Hz).

# 2. Arquivo PIO: blink.pio

O arquivo blink.pio contém um programa PIO escrito em linguagem de montagem específica para o subsistema PIO do chip RP2040. Este programa é genérico e pode ser reutilizado por várias máquinas de estado.

.program blink  
 pull block ; Aguarda um valor ser enviado pela CPU (tempo de delay em ciclos de clock)  
 out y, 32 ; Transfere os 32 bits recebidos para o registrador 'y'  
.wrap\_target  
 mov x, y ; Copia o valor de 'y' para o registrador 'x' (contador)  
 set pins, 1 ; Liga o LED (nível lógico alto no pino configurado)  
lp1:  
 jmp x-- lp1 ; Espera 'x+1' ciclos de clock (delay com LED aceso)  
 mov x, y ; Prepara novo delay  
 set pins, 0 ; Desliga o LED (nível lógico baixo)  
lp2:  
 jmp x-- lp2 ; Espera novamente 'x+1' ciclos (delay com LED apagado)  
.wrap ; Retorna ao início para repetir o processo indefinidamente

Este programa é independente do pino utilizado. O pino é configurado dinamicamente no código C.

# 3. Arquivo C: blink\_pio.c

Este código configura três máquinas de estado para executar o mesmo programa PIO blink, com parâmetros diferentes: cada uma com seu GPIO e frequência.

Trecho principal do código adaptado:

int main() {  
 stdio\_init\_all();  
 PIO pio = pio0;  
 uint offset = pio\_add\_program(pio, &blink\_program);  
  
 uint freqs[] = {1, 2, 3};  
 uint gpios[] = {11, 12, 13};  
  
 for (int i = 0; i < 3; i++) {  
 uint sm = i;  
 blink\_program\_init(pio, sm, offset, gpios[i]);  
 pio\_sm\_set\_enabled(pio, sm, true);  
 uint32\_t delay\_cycles = (125000000 / (2 \* freqs[i])) - 3;  
 pio->txf[sm] = delay\_cycles;  
 }  
  
 while (true) {  
 sleep\_ms(1000);  
 }  
}

# 4. Explicações Detalhadas

• Cada chamada a blink\_program\_init configura uma máquina de estado para usar um pino diferente.  
• A função pio->txf[sm] = ... envia um valor para a FIFO da máquina de estado correspondente.  
• Esse valor representa o tempo de delay desejado, calculado a partir da frequência.  
• A mesma lógica PIO é usada, mas cada máquina trabalha com contexto e parâmetros diferentes.

# 5. Conclusão

O projeto demonstra como reutilizar um mesmo programa PIO para controlar múltiplos pinos de forma independente, utilizando o poder das máquinas de estado do subsistema PIO do RP2040. Essa arquitetura permite criar aplicações de tempo real extremamente precisas sem sobrecarregar a CPU principal.