# Watchdog e recuperação de falhas

Reiniciar só quando precisa, e saber por que - manual prático




| Placa | ESP32 ou ESP32-S3 |
|---|---|
| Framework | Arduino Core, via PlatformIO |
| Bibliotecas | esp_task_wdt e esp_system, ambas do core |
| Prazo padrão | 15 segundos, ajustável |
| Nível | Intermediário |
| Tutorial online | iiot.esp32.tec.br/tutoriais/watchdog-e-recuperacao-de-falhas |

Firmware escrito para o tutorial e **não validado em bancada**. Teste no seu ambiente antes de usar em instalação que importe - e teste especialmente o caminho de falha, com o comando travar, não só a operação normal.

# 1. O que watchdog é, e o que não é

## Não é reiniciar de vez em quando

Existe uma prática que aparece em muito projeto embarcado e que não é watchdog: agendar um reinício a cada seis horas, na esperanca de que isso limpe o que estiver acumulando.

Isso é varrer o problema para debaixo do tapete. Se há vazamento de memória, o reinício periódico esconde a causa e um dia deixa de bastar. E, pior, o dispositivo reinicia quando está funcionando bem - no meio de uma leitura, de uma gravação, de uma transmissão.

Watchdog é o oposto: um temporizador que **só age quando o firmware para de dar sinal de vida**. Enquanto tudo anda, ele nunca dispara. Quando algo trava, reinicia em segundos.

## Os dois watchdogs do ESP32

| Mecanismo | O que vigia | Você configura? |
|---|---|---|
| Task Watchdog (TWDT) | tarefas registradas, que precisam dar sinal periódico | sim, é o do dia a dia |
| Interrupt Watchdog | se as interrupções estão sendo atendidas | não, dispara sozinho |

> No Arduino, a tarefa loopTask que executa seu setup e loop vem registrada no Task Watchdog em muitas versões do core. É por isso que um laço bloqueante longo as vezes produz reinício sem que ninguém tenha configurado watchdog nenhum.

# 2. Configurar e alimentar

```
esp_task_wdt_config_t cfg = {
.timeout_ms = WDT_SEGUNDOS * 1000,
.idle_core_mask = 0,
.trigger_panic = true
};
esp_task_wdt_reconfigure(&cfg);
esp_task_wdt_add(NULL);
```

O add(NULL) registra a tarefa atual. O trigger_panic faz o sistema reiniciar em vez de apenas registrar o alerta, que é o que você quer em campo.

> Em versões antigas do core a chamada era esp_task_wdt_init com dois argumentos. Se o compilador reclamar da struct, é esse o caso.

## O erro que anula o mecanismo

Há uma forma de ter watchdog configurado e proteção nenhuma: **alimentar em lugar errado**.

Se você espalhar chamadas de reset dentro das rotinas demoradas - dentro do laço que espera resposta Modbus, dentro do que aguarda conexão MQTT - o watchdog nunca vai disparar, porque você está garantindo o sinal de vida mesmo com o firmware travado.

A regra: alimente num ponto só, no laço principal, e não dentro do que pode travar. O firmware deste pacote tem exatamente uma chamada de reset.

> Corolário incomodo: se o seu firmware precisa de resets espalhados para não reiniciar, o problema não é o watchdog. É que existe uma operação bloqueante longa demais, e ela precisa virar não-bloqueante.

## Escolher o prazo

Prazo curto demais reinicia por engano. Longo demais deixa o dispositivo travado por tempo demais. Meça o pior caso do seu laço e use cerca de cinco vezes esse valor.

Nunca aperte até o limite: um watchdog que dispara em operação normal ensina a equipe a ignorar reinícios, e aí ele deixou de servir.

# 3. Por que reiniciou

Este é o ponto onde a maioria das implementações para, e é o que separa firmware que se recupera de firmware que só reinicia. Um dispositivo que reinicia sozinho e não diz por que é um problema de diagnóstico permanente.

| Motivo | O que significa | Onde procurar |
|---|---|---|
| ESP_RST_POWERON | energia aplicada | partida normal, nada a fazer |
| ESP_RST_SW | reinício por software | esperado, chamado pelo próprio código |
| ESP_RST_PANIC | exceção no firmware | acesso invalido de memória, divisão por zero |
| ESP_RST_TASK_WDT | uma tarefa travou | operação bloqueante no laço vigiado |
| ESP_RST_INT_WDT | interrupção presa | ISR longa ou interrupções desabilitadas |
| ESP_RST_BROWNOUT | tensão caiu do mínimo | FONTE, cabo ou pico de consumo - não é código |
| ESP_RST_EXT | reset pelo pino | botão ou circuito externo |

**Brownout merece atenção especial.** É o sintoma clássico de fonte subdimensionada ou cabo USB ruim, e é confundido com travamento de firmware com uma frequência impressionante. Se o motivo diz brownout, pare de olhar o código e meça a tensão.

Imprima o motivo no boot, e melhor ainda: publique junto da telemetria. A primeira vez que um dispositivo em campo avisar brownout, você economiza um dia de investigação em software para um problema que era da fonte.

# 4. Contador e modo seguro

## Contar reinícios sem perder na energia

Saber o motivo do último reset ajuda. Saber que houve nove reinícios na última hora muda o diagnóstico completamente. A memória RTC sobrevive a reinício por software e por watchdog, e é o lugar certo para esse contador.

```
RTC_DATA_ATTR static uint32_t contadorResets = 0;
RTC_DATA_ATTR static uint32_t marcaValidade = 0;
static const uint32_t MARCA = 0xC0FFEE01;
```

A marca de validade existe porque a memória RTC não é zerada de forma confiável na primeira energização - sem ela o contador começa com lixo.

A RTC **se perde ao desligar a energia**, e isso é característica útil, não limitação: um contador que zera na queda de energia distingue naturalmente reinício por falha de reinício por manutenção.

## Não entrar em ciclo de reinício

Suponha que o firmware trave sempre que tenta falar com um sensor que queimou. O watchdog reinicia. O firmware sobe, tenta o sensor, trava, reinicia. Em ciclo, para sempre, sem nunca ficar tempo suficiente no ar para receber um comando ou uma atualização OTA.

O watchdog fez seu trabalho e o resultado é pior que o travamento original, porque agora nem para diagnosticar da.

A saída é usar o contador para **degradar em vez de insistir**: depois de repetidas falhas, desligue a parte suspeita e mantenha apenas o essencial - rede, configuração e OTA. O dispositivo passa a funcionar mal, mas funciona, e permanece alcançável. Você conserta remotamente em vez de ir até lá.

# 5. Roteiro de teste

Configurar watchdog é fácil. Confiar nele exige provocar a falha de propósito.

| Teste | Como fazer | Resultado esperado |
|---|---|---|
| Partida normal | energizar a placa | motivo: energia aplicada; contador zerado |
| Reset manual | apertar o botão de reset | motivo: reset externo; contador NÃO sobe, porque não é falha |
| Travamento | enviar 'travar' pelo monitor | reinício em ~15 s com motivo TASK WATCHDOG |
| Contador | provocar três travamentos seguidos | contador chega a três |
| Queda de energia | desligar e ligar a alimentação | contador volta a zero; motivo: energia aplicada |
| Modo seguro | provocar travamentos até passar do limite | anuncia modo seguro e mantém a rede de pé |
| Alcance no modo seguro | acessar o dispositivo pela rede | responde; se não responder, está desligando coisa demais |

# 6. Solução de problemas

| Sintoma | Causa provável | O que fazer |
|---|---|---|
| Reinicia e não sei por que | motivo do reset não é lido | implemente esp_reset_reason antes de qualquer outra coisa |
| Motivo diz brownout | alimentação insuficiente | não é firmware: meça a tensão, troque fonte ou cabo |
| Watchdog nunca dispara | resets espalhados nas rotinas | alimente num ponto só, no laço principal |
| Dispara em operação normal | prazo curto para o pior caso | meça o laço e use cinco vezes o pior caso, ou remova o bloqueio |
| Entrou em ciclo de reinício | falha permanente sem degradação | implemente o modo seguro; sem ele exige acesso físico |
| Contador começa com lixo | falta a marca de validade | a RTC não é zerada de forma confiável na primeira energização |
| Erro de compilação na struct | versão antiga do core | use esp_task_wdt_init(segundos, panic) |

# 7. Checklist final

| | Item |
|---|---|
| [ ] | Task Watchdog registrado com esp_task_wdt_add(NULL) |
| [ ] | trigger_panic ligado, para reiniciar de verdade |
| [ ] | Exatamente UMA chamada de reset, no laço principal |
| [ ] | Nenhum reset dentro de rotina que pode travar |
| [ ] | Prazo em torno de cinco vezes o pior caso medido |
| [ ] | Motivo do reset lido e impresso no boot |
| [ ] | Motivo publicado junto da telemetria, se houver |
| [ ] | Aviso específico para brownout |
| [ ] | Contador em memória RTC com marca de validade |
| [ ] | Só motivos de falha incrementam o contador |
| [ ] | Modo seguro implementado, mantendo rede e OTA |
| [ ] | Teste de travamento proposital: reiniciou como TASK WDT |
| [ ] | Teste de energia: contador voltou a zero |
| [ ] | Teste de modo seguro: dispositivo continuou alcançável |