Contador de produção com ESP32

Sem perder pulso: o debounce é um teto de produção



Placa	ESP32 DevKit (ESP32-WROOM-32)
Framework	Arduino Core, via PlatformIO
Entrada	Sensor indutivo PNP 24 V, isolado por PC817
Transporte	MQTT 3.1.1 (PubSubClient)
Nível	Intermediário
Tutorial online	iiot.esp32.tec.br/tutoriais/contador-de-producao-com-esp32

O firmware deste pacote foi escrito a partir do funcionamento documentado do ESP32, do PC817 e de sensores indutivos PNP, e revisado linha a linha. **Ele não foi montado em bancada com sensor e linha reais.** Os números de teto vêm de aritmética, não de medição - e estão no texto justamente para você conferir em vez de acreditar. O teste da seção 7 existe para que você produza a sua própria evidência.

1. O contador que conta menos

Um contador de produção é o projeto que todo mundo acha simples: um sensor, uma interrupção, um número que sobe. Ele é simples até o dia em que o supervisor compara o número do painel com a contagem do expedidor e encontra uma diferença de centenas de peças - sempre para menos, sempre nos turnos de maior ritmo.

O motivo quase nunca é o sensor. É uma linha copiada de um tutorial de botão: um debounce de 50 ms, colocado para tratar um problema que o sensor indutivo não tem, e que impõe um teto de produção que ninguém calculou.

2. O sensor indutivo não trepica

Trepidação é fenômeno de **contato mecânico**. Uma chave ou um relé fecham fisicamente dois metais, e o contato quica por alguns milissegundos antes de estabilizar. Por isso todo tutorial de botão começa com debounce.

Um sensor indutivo PNP não tem contato. Ele é um oscilador com um transistor de saída: quando o alvo metálico entra no campo, a saída satura. Não há quique. O sinal que ele entrega já é limpo.

Isso não significa que o sinal chegue limpo no ESP32. O que aparece na entrada é **acoplamento elétrico** do inversor e dos contatores, não trepidação - e a diferença importa porque o remédio é outro: cabo blindado, par trançado, separação física dos condutores de potência e isolação galvânica. Software não resolve acoplamento; só esconde o sintoma até o dia em que esconder também custa peças.

3. A janela de debounce é um teto

Um debounce por tempo ignora qualquer pulso que chegue antes de a janela expirar. É o que você quer contra trepidação, e exatamente o que você não quer contra produção rápida: o pulso legítimo da peça seguinte também é ignorado.

O teto em peças por minuto é 60.000.000 dividido pela janela em microssegundos:

Janela	Pulsos/s	Teto de produção
50 ms	20	1.200 peças/min
20 ms	50	3.000 peças/min
12 ms	83	5.000 peças/min
10 ms	100	6.000 peças/min
5 ms	200	12.000 peças/min
2 ms	500	30.000 peças/min

O requisito deste projeto é 5.000 peças por minuto, ou seja, uma peça a cada 12 ms. Um debounce de 50 ms não recusa metade das peças: ele recusa **quatro de cada cinco**, e o painel mostra 1.200 com toda a confiança do mundo. Nenhum erro aparece no log, porque do ponto de vista do firmware ele fez exatamente o que mandaram.

A regra prática: a janela precisa ser bem menor que o intervalo mínimo entre peças. Para 12 ms de intervalo, 2 ms é confortável - filtra transiente de chaveamento, que dura microssegundos, e ainda deixa margem de seis vezes.

Existe um teto que o seu código não muda: a **frequência de comutação** do próprio sensor, que vem na folha de dados. Um M12 comum fica entre 500 Hz e 1 kHz. Confira esse número antes de prometer qualquer coisa.

4. Isolação: 24 V não entra no ESP32

A saída do sensor PNP entrega os 24 V da alimentação de comando quando ativa. O GPIO do ESP32 tolera 3,3 V. Ligar um no outro destrói a entrada, e em muitos casos leva o regulador junto.

O optoacoplador adapta o nível e **separa os terras**. A segunda parte importa mais do que parece: o terra de 24 V do painel e o terra do ESP32 costumam estar em pontos elétricos diferentes, e ligá-los cria um caminho por onde o ruído do inversor passa direto para a sua eletrônica.

O cálculo do resistor, e por que 1 k queima

A corrente direta recomendada do PC817 em operação contínua fica em torno de 10 mA, e a queda no LED é de aproximadamente 1,2 V. O resistor absorve o resto:

```
R = (24 V - 1,2 V) / 10 mA = 2.280 ohms -> comercial 2,2 kOhm
P = I^2 x R = (0,0104)^2 x 2200 = 0,24 W -> resistor de 1/2 W
```

Muito exemplo na internet usa 1 kOhm, que era o valor certo para 5 V. Em 24 V ele entrega:

```
I = (24 - 1,2) / 1000 = 22,8 mA
P = (0,0228)^2 x 1000 = 0,52 W
```

Vinte e dois miliampères não queimam o LED de imediato, mas aceleram a degradação do ganho do optoacoplador - o componente perde sensibilidade ao longo de meses, até começar a falhar pulso de forma intermitente. Já o resistor de 1/4 W, que é o que quase todo mundo tem na gaveta, está dissipando o dobro do que aguenta e queima. Use **2,2 kOhm de 1/2 W**.

Do lado do ESP32, o emissor do fototransistor vai ao GND e o coletor ao GPIO com pull-up. Quando o sensor ativa, o fototransistor conduz e puxa o pino para baixo: a lógica fica invertida de propósito, e a interrupção dispara na borda de descida.

5. A rotina de interrupção

A interrupção roda com o resto do sistema parado. Tudo o que você fizer dentro dela é tempo em que a rede, o watchdog e a própria contagem ficam esperando. Três regras evitam a maioria dos travamentos:

IRAM_ATTR é obrigatório. Sem ele a rotina pode estar numa região de flash que fica indisponível durante uma escrita, e a interrupção que chegar nesse instante derruba a placa.

Variável compartilhada é volatile. Senão o compilador guarda o valor em registrador e o laço principal lê um número que nunca muda.

Nada de Serial, alocação, delay ou publicação dentro dela. Guarde o dado e saia; quem imprime e publica é o laço principal.

```
void IRAM_ATTR aoDetectarPeca() {
const int64_t agora = esp_timer_get_time();
portENTER_CRITICAL_ISR(&trava);
if (agora - ultimoPulsoUs >= janelaUs) {
ultimoPulsoUs = agora;
contagem++;
} else {
rejeitados++;
}
portEXIT_CRITICAL_ISR(&trava);
}
```

Repare no contador de **rejeitados**. Ele transforma um teto silencioso em defeito visível: se a linha acelera e esse número sobe junto, a janela está grande demais. Sem ele, o firmware perde peça sem nunca dizer que perdeu.

6. Conte acumulado, nunca incremento

A tentação é publicar quantas peças passaram desde o último envio. É errado, e o erro só aparece no dia em que a rede oscila.

Se a mensagem que dizia doze peças se perde, essas doze peças **desaparecem para sempre**. Não há como o outro lado saber que faltou alguma coisa: ele recebeu uma sequência de números plausíveis, sem buraco visível.

Publicar o total acumulado resolve sozinho. Se uma mensagem se perde, a seguinte já traz o total correto, e o sistema de gestão calcula a diferença entre duas leituras consecutivas. O dado se cura sem intervenção.

Isso exige duas coisas do outro lado. A primeira é tratar o **reinício**: quando o ESP32 reinicia, o total volta a zero, e uma diferença negativa deve ser lida como reinício, não como produção negativa. O firmware publica um identificador de sessão de boot, persistido em NVS, para que essa distinção seja explícita.

A segunda é **nunca aceitar comando de zerar contador com retenção**. Um comando retido é reentregue a cada reconexão do cliente e zera a produção do turno sozinho, no meio da tarde. Vale a mesma regra da integração com SCADA: estado se retém, comando nunca.

A carga publicada:

```
{"ts":1755990000,"count_total":184523,"rate_ppm":4870,
"boot_id":7,"rejected":0,"quality":192}
```

O campo **rejected** viaja junto de propósito. Em operação normal ele fica em zero e ninguém olha para ele. No dia em que a linha acelera além do previsto, ele sobe antes de a diferença aparecer no inventário - e é isso que transforma um erro silencioso em um alarme.

7. Montagem e validação

Passo 1. Alimente o sensor com os 24 V do painel e meça a saída com multímetro, com o alvo dentro e fora do campo. Só depois ligue qualquer coisa no ESP32. Metade dos problemas de contagem termina aqui, em sensor mal posicionado.

Passo 2. Monte a entrada isolada conforme a seção 4 e confirme que os dois terras não se tocam em nenhum ponto.

Passo 3. Antes de ligar a interrupção, leia o pino em laço e imprima o nível. Passe o alvo com a mão e confirme que alterna. Se não alternar, o problema é elétrico e nenhum ajuste de firmware corrige.

Passo 4. Passe exatamente 20 peças, devagar, e compare. Precisa bater exatamente. Contando a mais, há ruído entrando; contando a menos, o pulso é curto demais ou a janela já está grande demais.

Passo 5 - o teste que quase ninguém faz. Reproduza o ritmo máximo previsto. Se a bancada não permite, gere pulsos com um segundo ESP32 ou um gerador de sinal, com um número conhecido. Para 5.000 peças/min: 83 pulsos por segundo durante um minuto, total esperado 5.000, com rejected em zero. Esse teste separa um contador que funciona de um que parece funcionar.

Passo 6. Com a contagem em andamento, reinicie a placa. O total volta a zero e o boot_id precisa mudar. Confirme que o outro lado interpreta isso como reinício.

O monitor serial aceita `status`, `zerar` e `janela N`. O comando de janela existe para você **provar o teto na bancada**: ponha 50000, gere pulsos rápidos e veja rejected subir enquanto o total trava.

8. Solução de problemas

Sintoma	Causa provável	O que fazer
Conta menos do que a linha produz	Janela grande demais para o ritmo	Olhe o contador de rejeitados: se acompanha a produção, é isso. Reduza a janela pela tabela da seção 3.
Conta a mais, e mais ainda quando o inversor liga	Ruído acoplado no cabo	Não aumente o debounce como primeira reação: isso troca erro visível por teto invisível. Cabo blindado, par trançado, afastamento da potência, aterramento.
Contagem congela depois de horas	Trabalho pesado na interrupção ou falta de IRAM_ATTR	Reduza a rotina ao mínimo e confirme o atributo. Verifique bloqueio longo no laço principal.
O total volta a zero sozinho	Comando de reset retido, ou reinício não tratado	Confira o boot_id e a política de retenção do tópico de comando.
Não bate com o expedido, mas rejected está em zero	O problema está antes do ESP32	Alvo passando fora do alcance, peça que não aciona por posição, ou ponto de contagem diferente do ponto de expedição.

9. Checklist antes de entregar

[ ]	Frequência de comutação do sensor conferida na folha de dados
[ ]	Janela de debounce calculada a partir do ritmo máximo, não copiada
[ ]	Teto da janela escolhida anotado e comunicado a quem opera
[ ]	Resistor de 2,2 kOhm de 1/2 W, conferido no multímetro
[ ]	Terras de 24 V e do ESP32 isolados, sem ponto comum
[ ]	Saída do sensor medida antes de ligar a placa
[ ]	Rotina de interrupção com IRAM_ATTR e sem Serial

[ ]	Frequência de comutação do sensor conferida na folha de dados
[ ]	Variáveis compartilhadas declaradas volatile
[ ]	Contagem de 20 peças na mão bateu exatamente
[ ]	Teste de ritmo máximo executado, com rejected em zero
[ ]	Telemetria publicando total acumulado, não incremento
[ ]	boot_id muda a cada reinício e o outro lado trata isso
[ ]	Nenhum tópico de comando com flag de retenção
[ ]	NTP sincronizado antes da primeira publicação
[ ]	rejected monitorado no painel, com alarme se sair de zero