Atualização OTA com rollback

Se a versão nova falhar, ela volta sozinha - manual prático



Placa	ESP32 ou ESP32-S3 com pelo menos 4 MB de flash
Framework	Arduino Core para o fluxo; ESP-IDF para rollback real
Bibliotecas	Update e HTTPClient, ambas do core
Modelo	pull: o dispositivo consulta e baixa
Nível	Intermediário
Tutorial online	iiot.esp32.tec.br/tutoriais/atualizacao-ota-no-esp32

Leia antes de prometer rollback. O estado pendente de verificação só existe se o bootloader foi compilado com CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE=y. No Arduino o bootloader vem pré-compilado e essa opção não está ligada por padrão: as chamadas não dão erro, mas também não protegem nada. Detalhes na seção 4.

1. Por que duas partições

O ESP32 não sobrescreve o firmware em execução - seria impossível, já que o código que roda é o mesmo que seria apagado. Então existem dois espaços de aplicação: o firmware roda de um, grava o novo no outro, e só então marca qual usar no próximo boot. Quem guarda essa marcação é uma terceira partição, pequena, chamada otadata.

Isso traz uma consequência excelente: **queda de energia no meio da atualização é segura**. O que está sendo escrito é o slot inativo; o ativo continua intacto e o otadata ainda aponta para ele. A placa reinicia como se nada tivesse acontecido.

É uma incômoda: você perde metade da flash. Numa placa de 4 MB, cada slot fica com cerca de 1,9 MB. É o preço, e não há como fugir dele.

A tabela de partições

```
# Name, Type, SubType, Offset, Size, Flags
nvs, data, nvs, 0x9000, 0x5000,
otadata, data, ota, 0xe000, 0x2000,
app0, app, ota_0, 0x10000, 0x300000,
app1, app, ota_1, , 0x300000,
spiffs, data, spiffs, , 0x900000,
```

O otadata ocupa 8 KB e registra qual slot é o válido. O app0 e o app1 são os dois espaços de aplicação. Repare que app1 e spiffs não declaram deslocamento: o gerador calcula a partir do tamanho do anterior.

> Aponte a tabela com board_build.partitions no platformio.ini. Sem isso o projeto usa a tabela padrão, de partição única, e a atualização falha na primeira tentativa.

2. Push ou pull

Modelo	Como funciona	Quando usar
Push	o dispositivo expõe um endpoint e você envia o binário	bancada e manutenção pontual, com acesso a rede local do equipamento
Pull	o dispositivo consulta um servidor e baixa se houver novidade	produto: atualizar cinquenta gateways vira publicar um arquivo

Este manual implementa o pull, que é o caso mais difícil e o mais útil. Por push, atualizar cinquenta equipamentos significa cinquenta operações manuais.

A gravação em si é a parte fácil

```
if (!Update.begin(tamanho)) {
Serial.printf("Espaco insuficiente: %s", Update.errorString());
return false;
}
size_t escritos = Update.writeStream(origem);
if (escritos != tamanho) {
Update.abort();
return false;
}
if (!Update.end(true)) {
return false;
}
```

O begin escolhe sozinho o slot inativo e verifica se o binário cabe. O writeStream grava direto do fluxo HTTP, sem buffer intermediário do tamanho do firmware - o que seria impossível, já que o binário é maior que a RAM. É o end(true) valida a imagem e atualiza o otadata.

Se qualquer etapa falhar, nada muda: o otadata continua apontando para o slot que está rodando.

3. O firmware que sobe é não funciona

Aqui está o risco de verdade, e a razão deste manual existir.

O caso que todo mundo imagina é a atualização falhar no meio. Esse é o caso fácil: já vimos que é seguro por construção.

O caso perigoso é outro. A atualização **conclui com sucesso**, o binário é válido, a placa reinicia no firmware novo, e aí descobre-se que ele tem um bug que impede conectar no Wi-Fi. Ou trava no setup. Ou consome memória até reiniciar em ciclo.

Do ponto de vista do OTA, deu tudo certo. Do ponto de vista prático, o dispositivo está inacessível e você perdeu justamente a capacidade de enviar a correção. É a visita técnica que o OTA existia para evitar, agora com o agravante de que você mesmo causou.

> Nenhuma verificação de integridade detecta isso. O arquivo está íntegro; o comportamento é que está errado.

Verificações que valem a pena, mesmo assim

Verificação	O que pega
Tamanho anunciado pelo servidor	servidor que não informa tamanho: sem ele não há como detectar truncamento
Primeiro byte igual a 0xE9	página de erro HTML gravada como se fosse firmware
Bytes gravados contra anunciados	download interrompido em rede ruim
SHA-256 publicado junto	corrupção silenciosa; custa segundos e elimina a classe inteira

4. Rollback: o estado pendente

A solução é não confiar no firmware novo de imediato. Com rollback habilitado, uma imagem recem-gravada entra em estado **pendente de verificação**. O bootloader a executa, mas com uma condição: se a placa reiniciar sem que o firmware tenha declarado que está funcionando, o bootloader volta sozinho para a imagem anterior.

```
if (estado != ESP_OTA_IMG_PENDING_VERIFY) return;
if (autodiagnosticoOk()) {
esp_ota_mark_app_valid_cancel_rollback();
} else {
esp_ota_mark_app_invalid_rollback_and_reboot();
}
```

O que conta como autodiagnóstico é decisão de projeto, e é onde mora o valor. Conectar no Wi-Fi em até sessenta segundos costuma ser o mínimo. Se o dispositivo depende de MQTT ou de um barramento Modbus, exija também isso: de nada adianta confirmar uma imagem que entra na rede e não fala com o equipamento.

Chame a confirmação depois de o diagnóstico ter tido tempo de rodar, no laço principal e não no setup. Confirmar cedo demais anula o mecanismo inteiro.

O detalhe que invalida tudo

O estado pendente **só existe se o bootloader foi compilado com rollback habilitado**. É opção do bootloader, não do seu código:

```
CONFIG_BOOTLOADER_APP_ROLLBACK_ENABLE=y
```

Em ESP-IDF você acrescenta ao sdkconfig e recompila. Em Arduino o bootloader vem pré-compilado e a opção não está ligada por padrão: chamar as funções não da erro, mas também não protege nada, porque nenhuma imagem entra em estado pendente.

> É a mesma fronteira do NAPT: o Arduino serve para o caminho simples, e o produto que precisa de garantia migra para ESP-IDF. Vale saber disso antes de prometer rollback a um cliente.

5. Passo a passo

Passo 1 - Preparar as partições

Crie o partitions.csv com a tabela da seção 1, ajustando os tamanhos a sua flash, e aponte no platformio.ini. Compile e confira o relatório de tamanho: erro de imagem grande demais significa slots pequenos para o seu firmware.

Passo 2 - Gravar a primeira versão pelo cabo

Não existe OTA sem um firmware já rodando. Grave uma versão com o número bem visível no log de boot, porque é assim que você confirma a troca depois.

```
pio run -t upload
pio device monitor -b 115200
```

Passo 3 - Publicar o binário novo

Altere o número da versão, compile sem gravar e publique o firmware.bin junto de um versão.txt com o número novo. Para teste, o servidor embutido do Python resolve:

```
python -m http.server 8000
```

Passo 4 - Acompanhar a atualização

Observe o monitor serial: tamanho anunciado, progresso da gravação e confirmação antes do reinício. Se o download falhar, teste a URL pelo navegador de um computador na mesma rede - erro de caminho é bem mais comum que erro de código.

Passo 5 - Confirmar a troca

Depois do reinício, o log de boot deve mostrar o número de versão novo. Se mostrar o antigo, a gravação não concluiu ou o otadata não foi atualizado.

Passo 6 - Testar a falha de propósito

Este é o passo que quase ninguém faz, e é o que dá confiança no mecanismo. Compile uma versão **deliberadamente quebrada**, com a senha do Wi-Fi errada por exemplo, e atualize para ela.

Com rollback ativo, a placa deve tentar, falhar no autodiagnóstico e voltar sozinha a versão anterior. Se ficar presa na versão quebrada, o rollback não está funcionando - e é muito melhor descobrir isso na bancada do que num painel trancado.

6. Solução de problemas

Sintoma	Causa provável	O que fazer
Erro de partição não encontrada	tabela padrão em uso	aponte o partitions.csv no platformio.ini
Sketch too big após habilitar OTA	metade da flash virou o segundo slot	use a tabela min_spiffs em placas de 4 MB, ou uma placa maior
Conclui mas volta na versão antiga	otadata não foi atualizado	o end(true) falhou; leia o Update.errorString()
Download interrompido em rede ruim	sem comparação de tamanho	confira bytes gravados contra anunciados e implemente repetição
Gravou HTML como firmware	servidor devolveu página de erro	a checagem do primeiro byte 0xE9 pega antes de gravar
Chamei rollback e nada acontece	bootloader sem a opção	no Arduino padrão o rollback não está disponível; migre para ESP-IDF
Ciclo de reinício após atualizar	imagem ruim sem rollback	recuperação pelo cabo; é o motivo de testar a falha antes de ir a campo

7. Checklist final

	Item
[ ]	partitions.csv criado com otadata, app0 e app1
[ ]	board_build.partitions apontado no platformio.ini
[ ]	Firmware cabe no slot, sem erro de imagem grande demais
[ ]	Número de versão visível no log de boot
[ ]	Primeira gravação feita pelo cabo
[ ]	firmware.bin e versão.txt publicados no servidor
[ ]	URL testada pelo navegador antes de culpar o código
[ ]	Verificação do primeiro byte 0xE9 ativa
[ ]	Comparação de bytes gravados contra anunciados
[ ]	Autodiagnóstico ajustado ao que o dispositivo precisa fazer
[ ]	Confirmação chamada no laço, não no setup
[ ]	Teste de sucesso: log mostrou a versão nova
[ ]	Teste de falha proposital: a placa voltou sozinha