Projetos ESP-IDF de alto desempenho

Como usar CI/CD para ganhar agilidade e confiança

Espressif Summit Brazil 2025

Pedro Minatel

Campinas, 5 de agosto de 2025

Agenda

1. Espressif
2. Agenda
3. Processo de desenvolvimento de firmware
4. Por que usar controle de versão?
5. O que é CI/CD?
6. GitHub Actions
7. GitHub Actions - Como criar um workflow
8. Conclusão
9. Obrigado

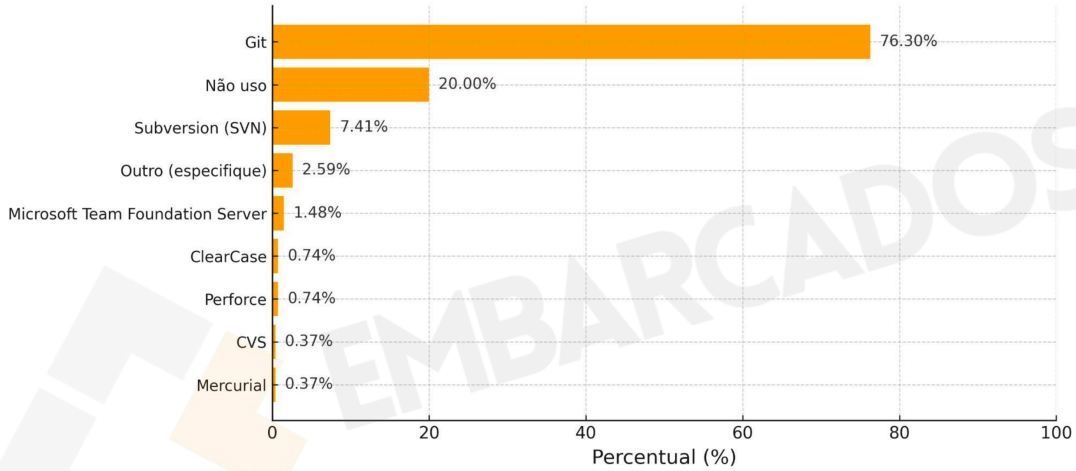
ESPRESSIF

Você utiliza controle de versão?

- Git
- SVN
- CVS
- Etc.

Relatório Portal Embarcados

Qual dos seguintes sistemas de software de controle de versão você usa atualmente?



Também foram citados: BitBucket, Gitlab, Cópias sucessivas do projeto no Google Drive

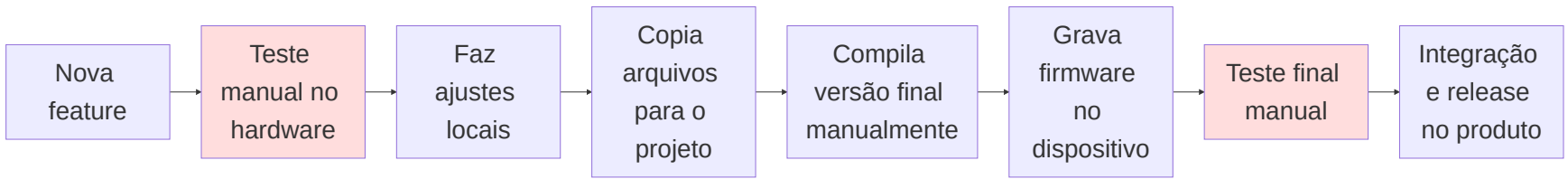
Fonte: Embarcados - Relatório da Pesquisa sobre o Mercado Brasileiro de Sistemas Embarcados e IoT 2025

Processo de desenvolvimento de firmware

ESPRESSIF

Como, geralmente, é o processo de desenvolvimento sem controle de versão?

Processo de desenvolvimento de firmware



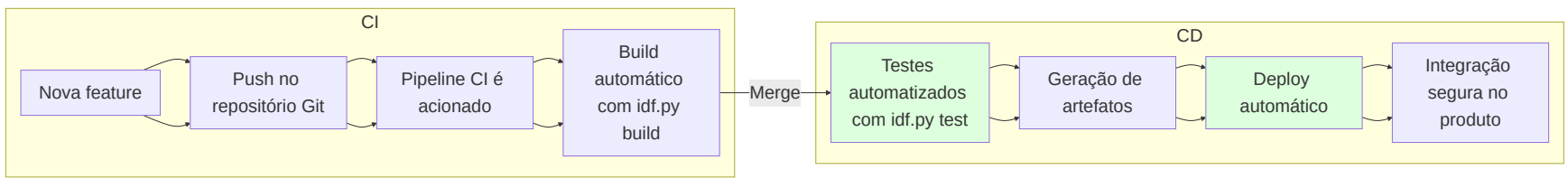
Por que usar controle de versão?

- 🔁 **Histórico completo do projeto**
  - Acompanhe cada mudança feita no código
- 👥 **Trabalho em equipe facilitado**
  - Integração e colaboração assíncrona
- 🧪 **Integração com CI/CD**
  - Autmação de testes, builds e deploys rastreáveis por commit
- 🛠️ **Auditoria e rastreabilidade**
  - Quem mudou o quê, quando e por quê
- 🚀 **Confiança no desenvolvimento**
  - Menos medo de experimentar e mais agilidade com segurança

O que é CI/CD?

- ⚙️ **CI - Continuous Integration**
  - Integração contínua de código em um repositório compartilhado
  - Cada mudança é automaticamente testada e compilada
- 🚀 **CD - Continuous Delivery / Deployment**
  - Entrega contínua: geração automática de artefatos prontos para uso
  - Deploy contínuo: publicação automática em produção (ou dispositivos)
- 🎯 **Objetivo:**
  - Detectar erros rapidamente
  - Entregar versões com frequência, confiança e agilidade

O que é CI/CD?
Processo de desenvolvimento de firmware com CI/CD



O que é CI/CD?
Como integrar CI/CD com o GitHub

- 🧩 O GitHub oferece suporte nativo a CI/CD via **GitHub Actions**
  - Permite rodar scripts automatizados sempre que algo acontece no repositório
- Cada vez que um código é enviado (push) ou um PR é aberto:
  - Um **workflow** pode ser automaticamente executado
  - Etapas como **build**, **testes**, **deploy** podem ser executadas
- Integração com projetos com ESP-IDF:
  - Use containers com o SDK pré-instalado
  - Reproduza builds locais no ambiente de CI
- CI/CD se integra diretamente ao controle de versão

GitHub Actions

O GitHub Actions é gratuito para projetos públicos? **Sim!**

- Projetos públicos têm **execução gratuita e ilimitada** de workflows
- Ideal para comunidades, bibliotecas e projetos open source
- Inclui suporte a:
  - Linux, macOS, Windows
  - Containers e ambientes personalizados (como o ESP-IDF)

⚠️ *Projetos privados têm limites gratuitos menores — depois disso, é pago*

GitHub Actions

Como integrar CI/CD com o GitHub

- A **Espressif mantém Actions oficiais** para facilitar:
  - Build com `idf.py`
  - Testes com Unity (C)
  - Uso de containers com o ESP-IDF
- Ideal para automatizar:
  - ⚙️ Builds confiáveis
  - 🧪 Testes simulados e HIL (Hardware-in-the-Loop)
  - 🚀 Geração de firmwares para produção (artefatos)

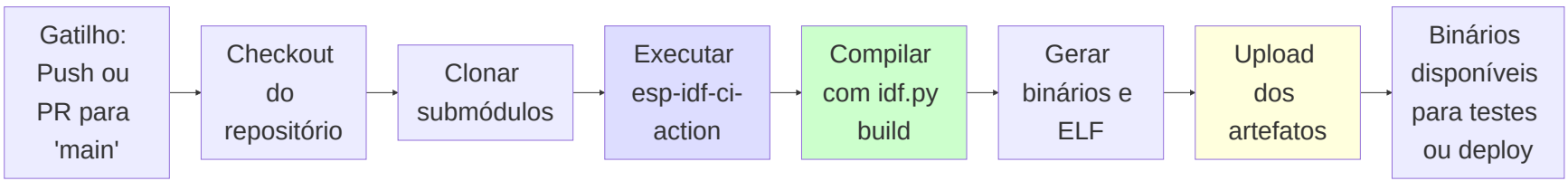
GitHub Actions - Como criar um workflow

O que é um Workflow no GitHub Actions?

- Um **workflow** define o que deve acontecer automaticamente no seu projeto
- Escrito em YAML e armazenado em `.github/workflows/` (dentro da raiz do projeto)
- Pode conter múltiplos **jobs**, que têm múltiplos **steps**
- Executado em resposta a eventos como `push`, `pull_request`, etc.

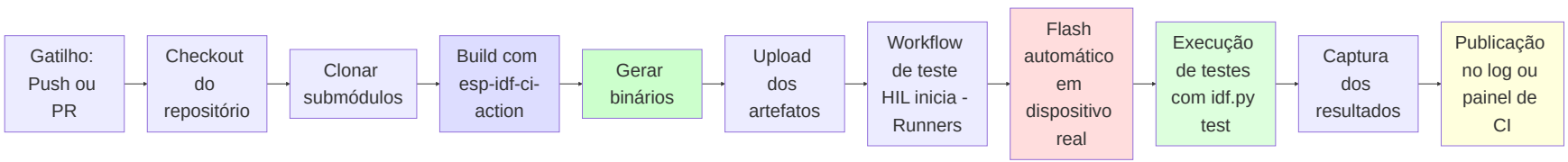
GitHub Actions - Workflow para ESP-IDF

Fluxo de build com GitHub Actions + ESP-IDF



GitHub Actions - Workflow para ESP-IDF

Build + Teste em Hardware (HIL) com GitHub Actions



GitHub Actions - Workflow para ESP-IDF

Workflow para `build` e upload dos binários como `artefatos`

.github/workflows/build.yml

```yaml
name: Build Application Workflow

on:
  pull_request:
    branches:
      - main

jobs:
  build:
    name: Build Application
    ...
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
        with:
          submodules: 'recursive'
      ...
```

GitHub Actions - Workflow para ESP-IDF

Define

.github/workflows/build.yml

```yaml
jobs:
  build:
    name: Build Test App
    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        espidf_target:
          - esp32
          - esp32c3
          - esp32c6
          - esp32h2
          - esp32p4
          - esp32s2
          - esp32s3
```

GitHub Actions - Workflow para ESP-IDF

Workflow para `build` e upload dos binários como `artefatos`

.github/workflows/build.yml

```yaml
steps:
  - name: Checkout repository
    uses: actions/checkout@v4
    with:
      submodules: 'recursive'
  - name: Build Test Application with ESP-IDF
    uses: espressif/esp-idf-ci-action@v1
    with:
      esp_idf_version: "latest"
      target: ${{ matrix.espidf_target }}
      path: 'test_app'
```

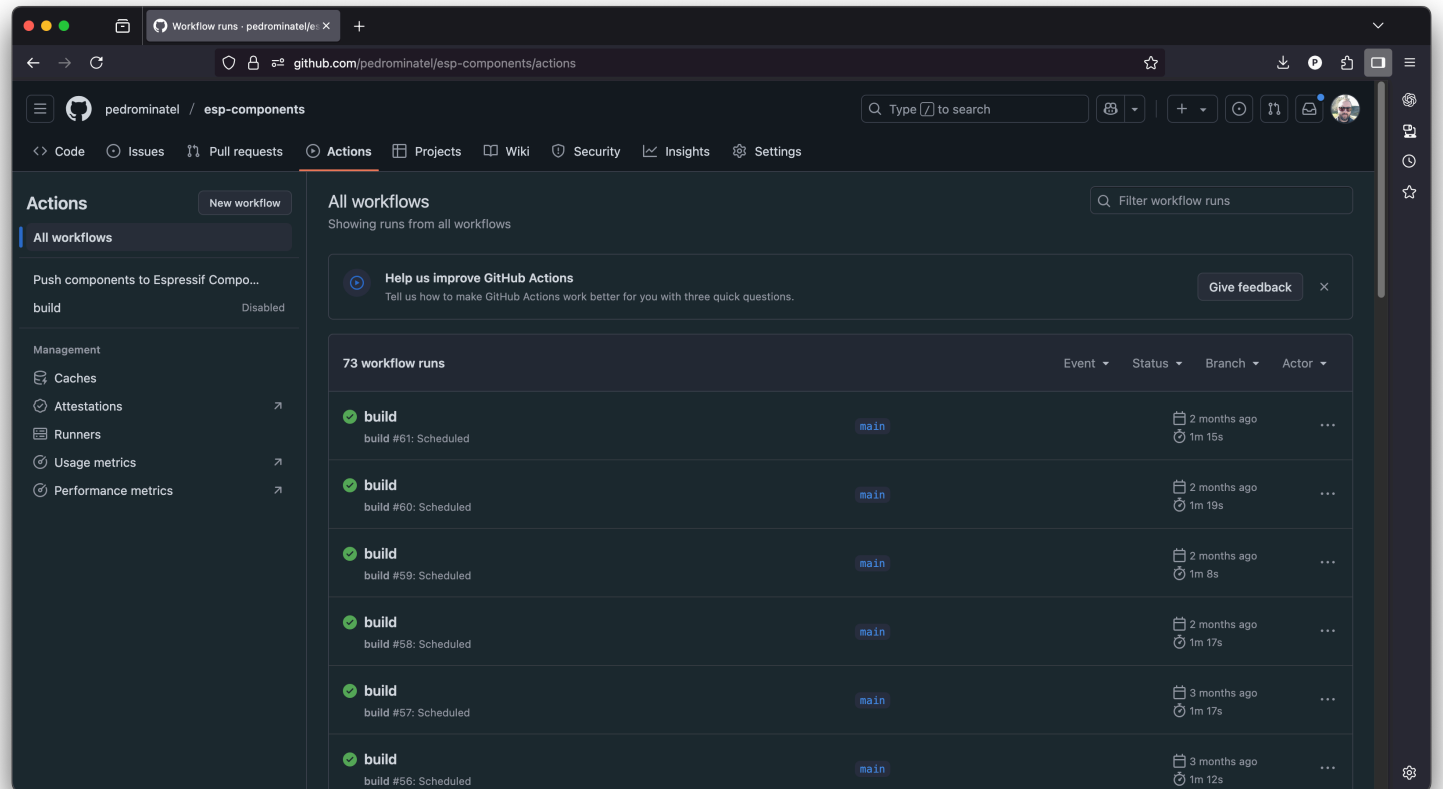
GitHub Actions - Workflow para ESP-IDF

Workflow para `build` e upload dos binários como `artefatos`

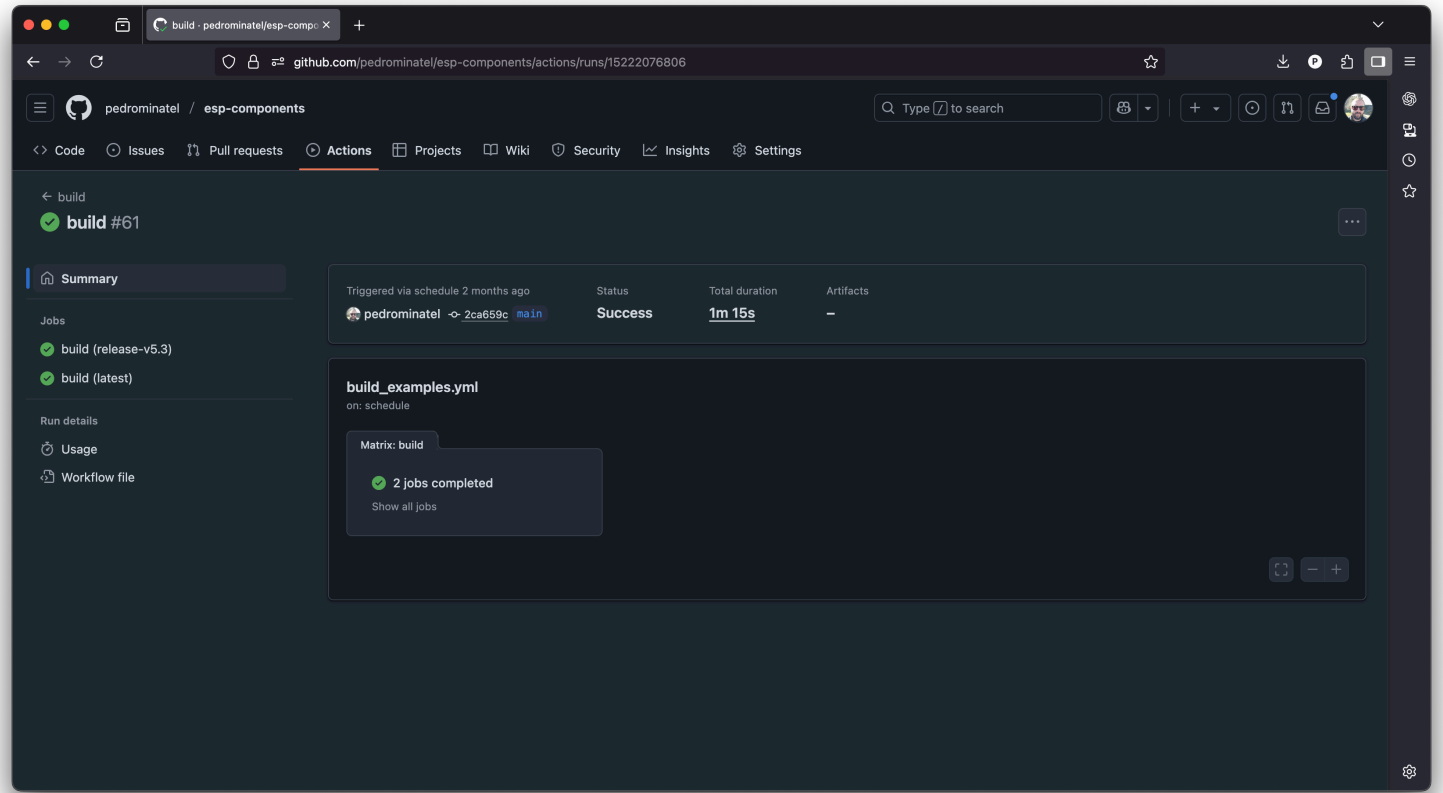
.github/workflows/build.yml

```yaml
      - name: Upload files to artifacts for run-target job
        uses: actions/upload-artifact@v4
        with:
          name: test_app_bin_${{ matrix.espidf_target }}
          path: |
            test_app/build/bootloader/bootloader.bin
            test_app/build/partition_table/partition-table.bin
            test_app/build/example_test_app.bin
            test_app/build/example_test_app.elf
            test_app/build/flasher_args.json
          if-no-files-found: error
```

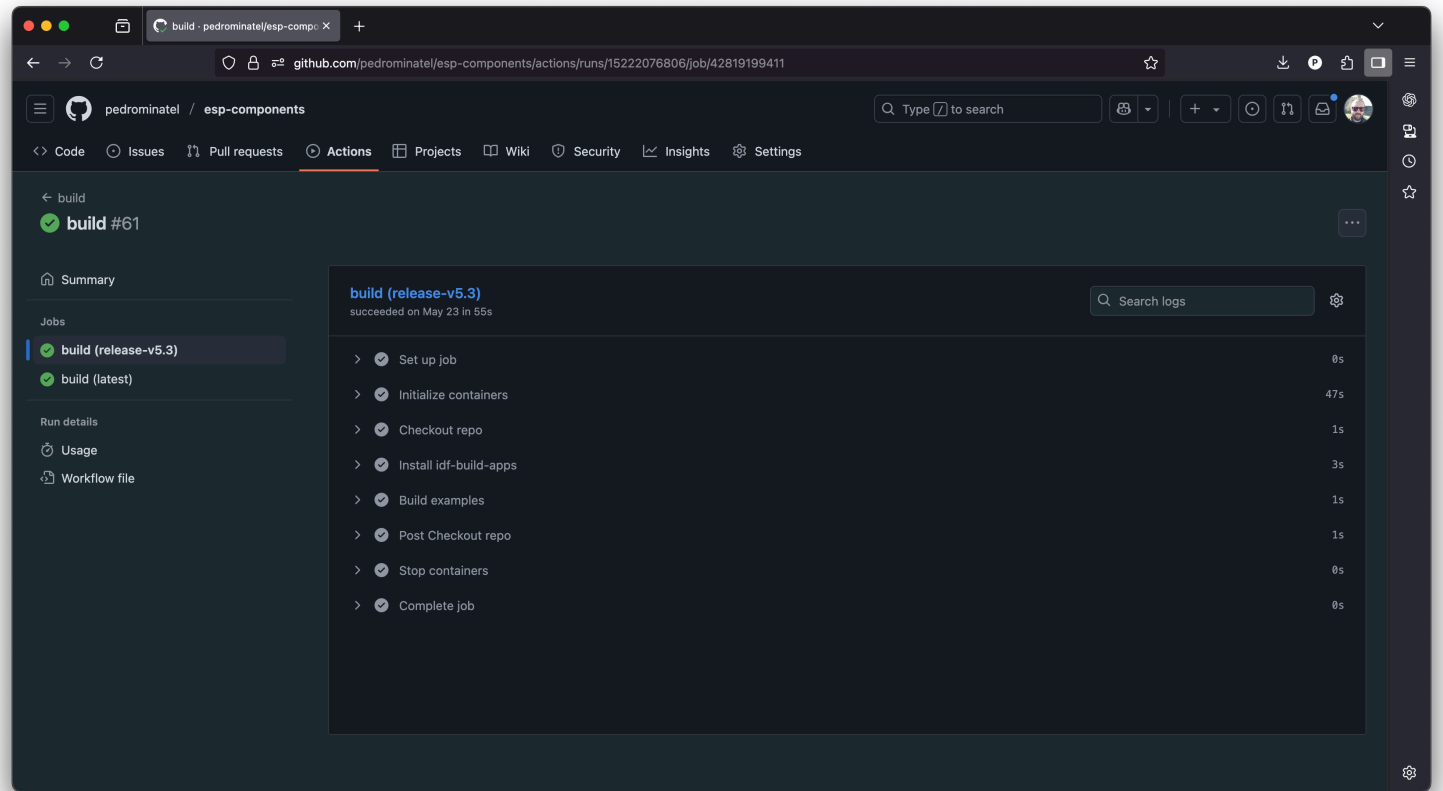
GitHub Actions - Interface no GitHub



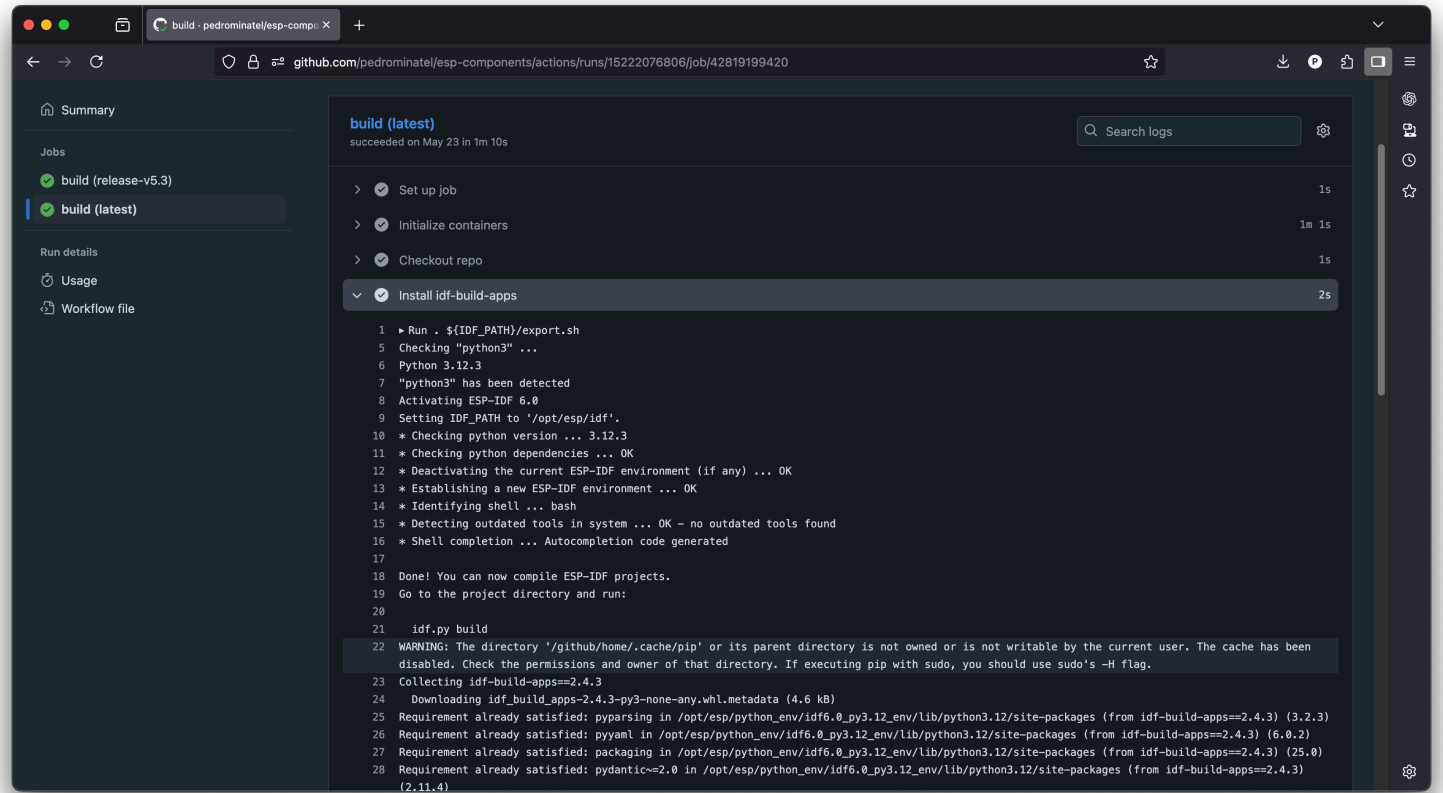
GitHub Actions - Interface no GitHub



GitHub Actions - Interface no GitHub



GitHub Actions - Interface no GitHub



GitHub Actions - Onde usar
Onde podemos usar o GitHub Actions?

- 🛠️ Build automático do projeto
- 🧪 Testes automatizados (HIL)
  - Runners com RaspberryPi
  - Simulação
    - Wokwi
    - QEMU
- 🧩 Testes de componentes e publicação no Registry
- 📦 Geração de binários (com merge de imagens)
- 📡 Binário para OTA
- 🚀 Deploy automatizado

GitHub Actions - O que ganhamos?
Resultados com CI/CD em um projeto ESP-IDF

- ⚡ **Builds mais rápidos e frequentes**
  - Cada PR já vem validado e pronto para merge
- 🧪 **Mais qualidade no código**
  - Bugs são detectados antes de chegarem à produção
- 🚀 **Entrega contínua de firmware**
  - Facilita ciclos de release curtos e confiáveis
- 🤝 **Melhor colaboração em equipe**
  - Todos veem o impacto das mudanças rapidamente
- 🔒 **Mais confiança no que vai para o dispositivo**
  - Tudo é testado, versionado e rastreável

✅ Mais agilidade, menos retrabalho, mais segurança no seu projeto

GitHub Actions - Futuro

Novo action `espressif/build-esp-idf-projects-action`

- Builds em paralelo
- Build seletiva (apenas arquivos modificados)
- Reutilizável em outros repositórios
- Integração com testes automáticos com pytest
- Para projetos maiores com múltiplos apps ou componentes

GitHub: https://github.com/espressif/build-and-test-esp-idf-projects-example

Conclusão

- 🧰 CI/CD **não é só para web** — também é essencial para firmware
- ⚙️ O GitHub Actions permite automação **simples, poderosa e gratuita**
- 🚀 A Espressif fornece ferramentas prontas para ESP-IDF
- ✅ Com CI/CD:
  - Você ganha velocidade sem perder qualidade
  - Reduz erros humanos
  - Releases de firmware com mais confiança

Comece pequeno, automatize o essencial e evolua com o tempo!

Obrigado