

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

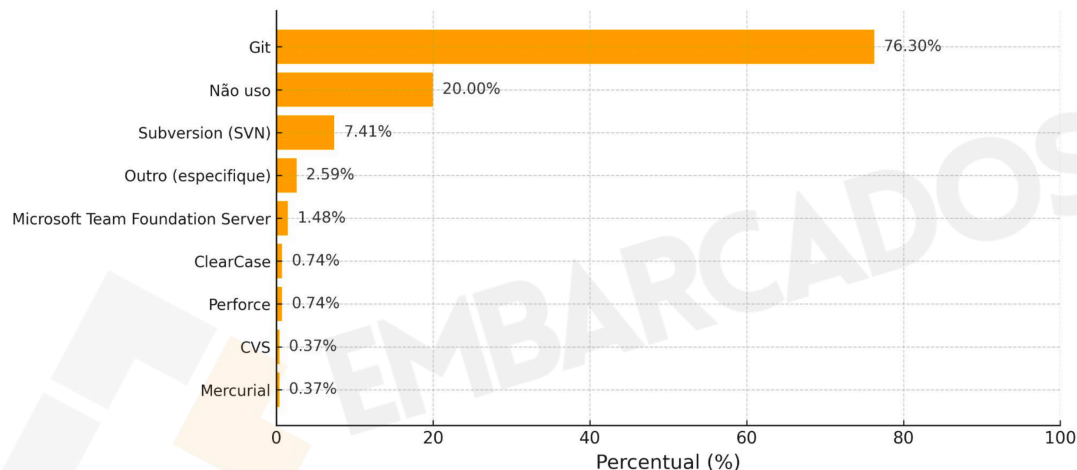
Pergunta

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?

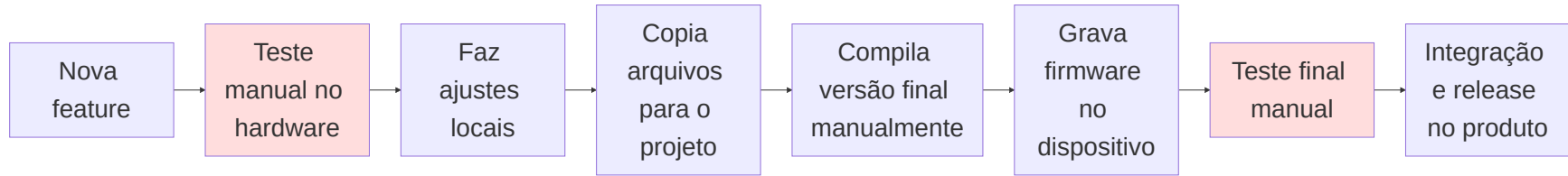


Também foram citados: BitBucket, Gitlab, Cópias sucessivas do projeto no Google Drive

Processo de desenvolvimento de firmware

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

Processo de desenvolvimento de firmware



Por que usar controle de versão?

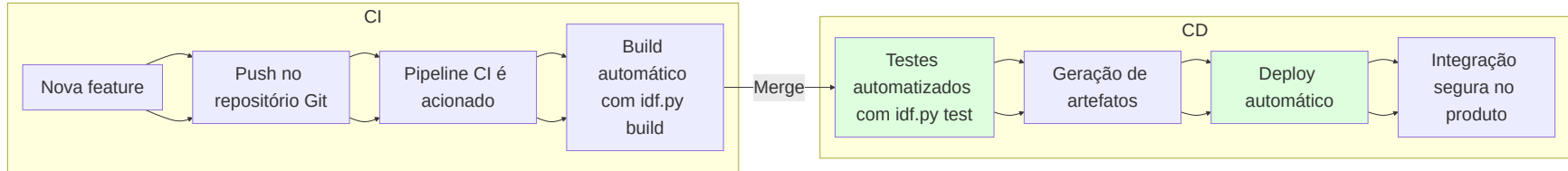
-  Histórico completo do projeto
 - Acompanhe cada mudança feita no código
-  Trabalho em equipe facilitado
 - Integração e colaboração assíncrona
-  Integração com CI/CD
 - Automação de testes, builds e deploys rastreáveis por commit
-  Auditoria e rastreabilidade
 - Quem mudou o quê, quando e por quê
-  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)

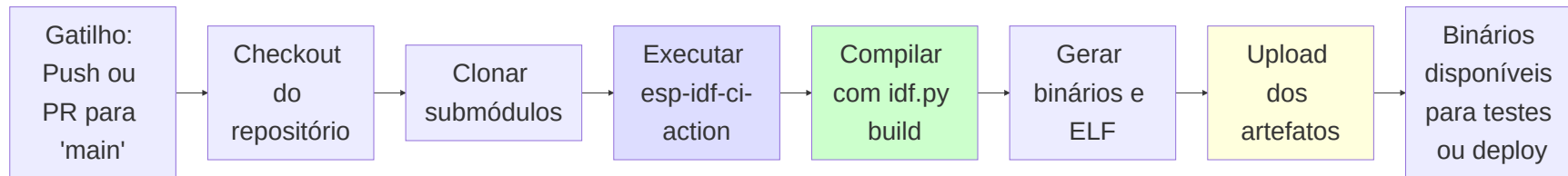
GitHub Actions - Como criar um workflow

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.

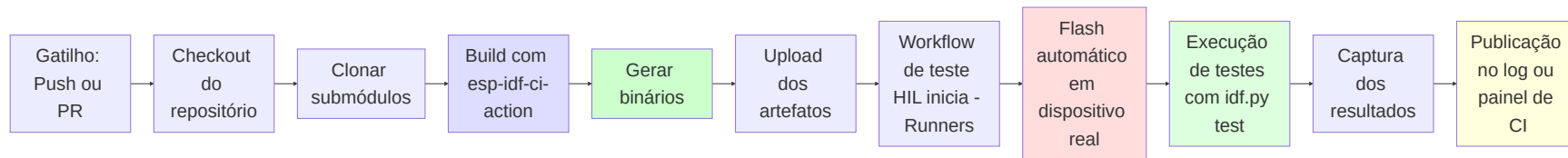
GitHub Actions - Workflow para ESP-IDF

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`

```
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

```
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`

```
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'
```

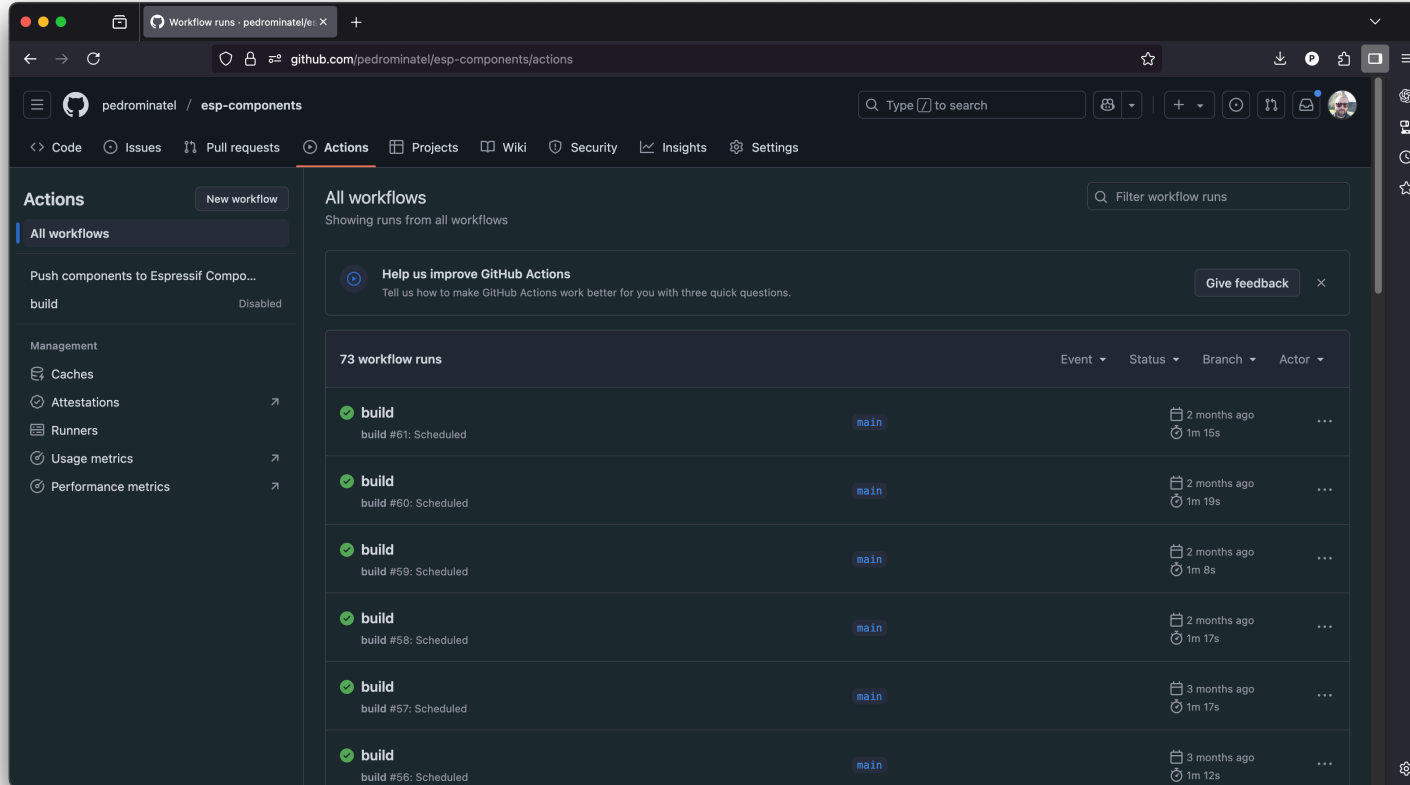
GitHub Actions - Workflow para ESP-IDF

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

`.github/workflows/build.yml`

```
- 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

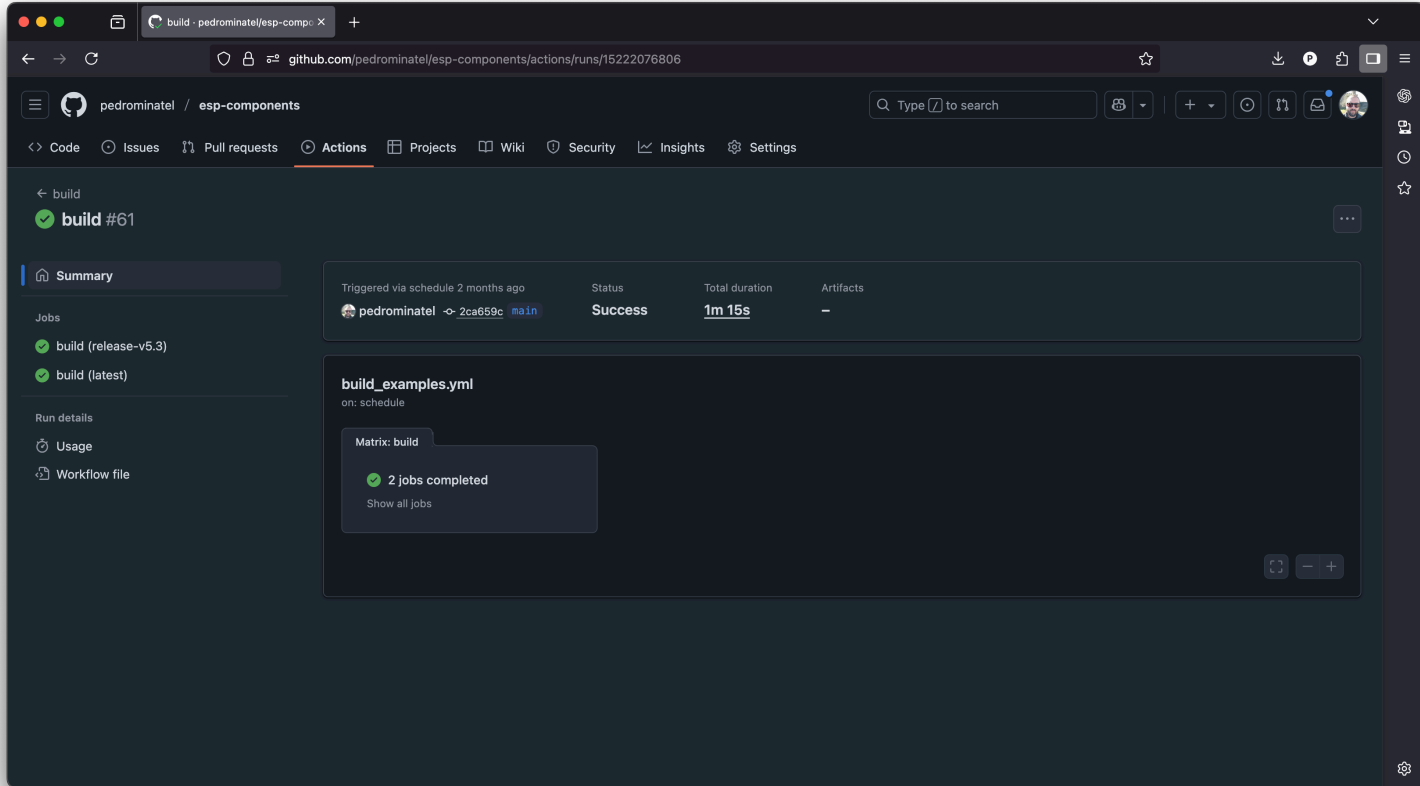


The screenshot shows the GitHub Actions interface for the repository 'pedrominate/esp-components'. The top navigation bar includes links for Code, Issues, Pull requests, Actions (selected), Projects, Wiki, Security, Insights, and Settings. The left sidebar shows the 'Actions' section with a 'New workflow' button and a list of workflow runs. The main content area displays 'All workflows' with a search bar and a list of workflow runs. A 'Help us improve GitHub Actions' banner is visible at the top of the workflow runs list.

Workflow runs table:

Event	Status	Branch	Actor
build #61: Scheduled	Success	main	2 months ago 1m 15s
build #60: Scheduled	Success	main	2 months ago 1m 19s
build #59: Scheduled	Success	main	2 months ago 1m 8s
build #58: Scheduled	Success	main	2 months ago 1m 17s
build #57: Scheduled	Success	main	3 months ago 1m 17s
build #56: Scheduled	Success	main	3 months ago 1m 12s

GitHub Actions - Interface no GitHub



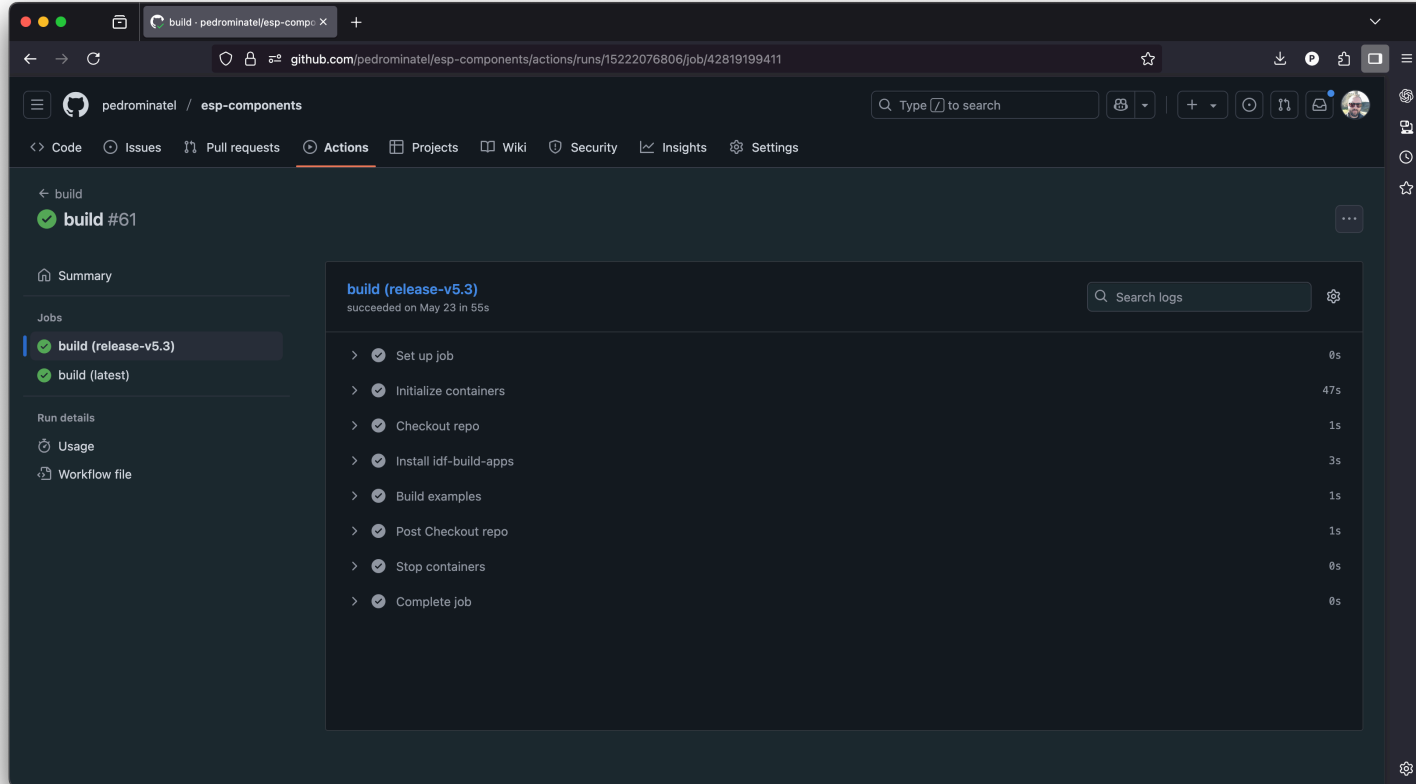
The screenshot shows the GitHub Actions interface for a workflow named "build" in the repository "pedrominate/esp-components". The workflow is currently running, indicated by a green checkmark and the label "build #61".

The interface displays the following information:

- Summary:** A section on the left with tabs for Summary, Jobs, Run details, Usage, and Workflow file. The Jobs tab is selected, showing a list of jobs: "build (release-v5.3)" and "build (latest)".
- Jobs:** A list of jobs that have been executed. The first job, "build (release-v5.3)", is shown with a green checkmark and the label "build #61".
- Run details:** A section on the right showing the details of the selected job. It includes the workflow file "build_examples.yml" and the matrix "build".
- Matrix:** A table showing the matrix configuration for the workflow. It indicates that 2 jobs completed successfully.

The interface also includes a search bar at the top, a navigation menu on the left, and a sidebar on the right with various icons for repository management.

GitHub Actions - Interface no GitHub



The screenshot shows the GitHub Actions interface for the 'build' workflow in the 'esp-components' repository. The workflow is currently running, and the job 'build (release-v5.3)' has succeeded on May 23 in 55s.

Workflow Summary:

- Workflow: build
- Job: build #61
- Summary: build (release-v5.3) succeeded on May 23 in 55s

Jobs:

- build (release-v5.3)
- build (latest)

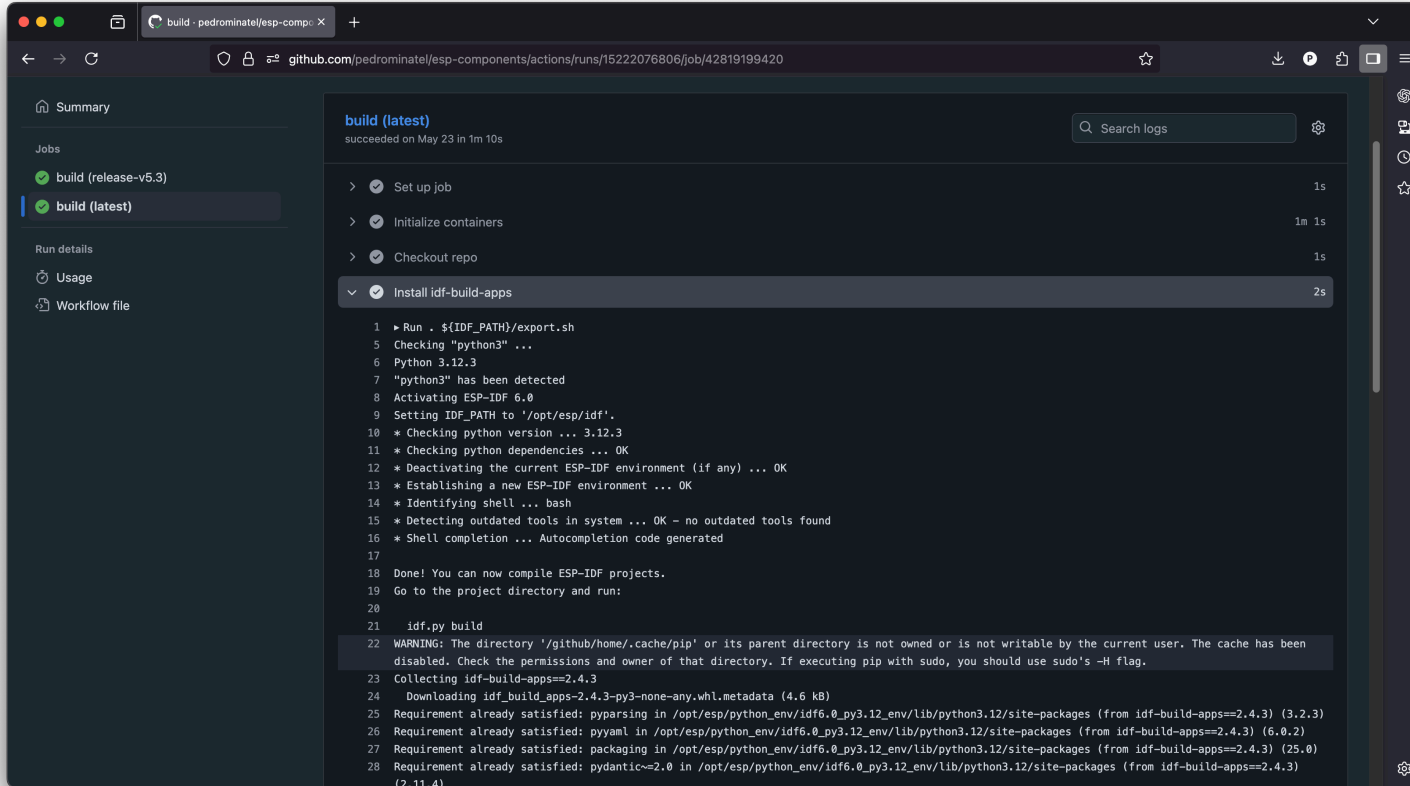
Run details:

- Usage
- Workflow file

Job Steps:

Step	Duration
Set up job	0s
Initialize containers	47s
Checkout repo	1s
Install idf-build-apps	3s
Build examples	1s
Post Checkout repo	1s
Stop containers	0s
Complete job	0s

GitHub Actions - Interface no GitHub



The screenshot displays the GitHub Actions interface for a repository named 'pedrominate/esp-comp'. The main view shows the 'build (latest)' job, which succeeded on May 23 in 1m 10s. The job steps are listed on the right, with 'Install idf-build-apps' being the current step, taking 2s. The left sidebar shows the 'Summary' tab, 'Jobs' list, 'Run details', 'Usage', and 'Workflow file'.

build (latest)
succeeded on May 23 in 1m 10s

Search logs

- Set up job 1s
- Initialize containers 1m 1s
- Checkout repo 1s
- Install idf-build-apps 2s**

```
1 ▶ Run . ${IDF_PATH}/export.sh
5 Checking "python3" ...
6 Python 3.12.3
7 "python3" has been detected
8 Activating ESP-IDF 6.0
9 Setting IDF_PATH to '/opt/esp/idf'.
10 * Checking python version ... 3.12.3
11 * Checking python dependencies ... OK
12 * Deactivating the current ESP-IDF environment (if any) ... OK
13 * Establishing a new ESP-IDF environment ... OK
14 * Identifying shell ... bash
15 * Detecting outdated tools in system ... OK - no outdated tools found
16 * Shell completion ... Autocompletion code generated
17
18 Done! You can now compile ESP-IDF projects.
19 Go to the project directory and run:
20
21 idf.py build
22 WARNING: The directory '/github/home/.cache/pip' or its parent directory is not owned or is not writable by the current user. The cache has been
23 disabled. Check the permissions and owner of that directory. If executing pip with sudo, you should use sudo's -H flag.
24 Collecting idf-build-apps==2.4.3
25 Downloading idf_build_apps-2.4.3-py3-none-any.whl.metadata (4.6 kB)
26 Requirement already satisfied: pyparsing in /opt/esp/python_env/idf6.0_py3.12_env/lib/python3.12/site-packages (from idf-build-apps==2.4.3) (3.2.3)
27 Requirement already satisfied: pyyaml in /opt/esp/python_env/idf6.0_py3.12_env/lib/python3.12/site-packages (from idf-build-apps==2.4.3) (6.0.2)
28 Requirement already satisfied: packaging in /opt/esp/python_env/idf6.0_py3.12_env/lib/python3.12/site-packages (from idf-build-apps==2.4.3) (25.0)
29 Requirement already satisfied: pydantic~=2.0 in /opt/esp/python_env/idf6.0_py3.12_env/lib/python3.12/site-packages (from idf-build-apps==2.4.3)
(2.11.4)
```

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