Spickzettel: GitHub für gemischte Projekte (Code, Hardware, Dokumentation)

Ziel

Ein Projekt mit **Code, Schaltplänen und Dokumentation** sinnvoll in einem GitHub-Repository verwalten.

Empfohlene Struktur im GitHub-Repo

```
projektname/
├── src/              # Quellcode
├── hardware/         # Schaltpläne, Layouts, Gerberdaten
├── docs/             # Doku: Markdown, Bilder, PDFs (generiert oder
versioniert)
├── test/             # Tests, Simulationen
├── .github/          # Actions-Workflows, PR-Vorlagen
├── .gitignore
├── .gitattributes
├── README.md
└── LICENSE
```

GitHub-spezifische Features sinnvoll nutzen

README.md

- Erste Anlaufstelle für Projektbeschreibung
- Verlinkung zu Unterordnern, Dokumentation, Hardware

GitHub Issues

- Dokumentation von Bugs, ToDos, Doku-Wünschen, Hardwareänderungen
- Labels nutzen: `bug`, `doc`, `hw`, `question`, `discussion`

Pull Requests

- Trennung von Feature-Änderungen in `src/`, `hardware/` oder `docs/`
- Reviewer gezielt nach Fachbereich zuweisen (z. B. Code vs. Layout)

Branch Protection

- Verhindert direkte Änderungen auf `main`
- Aktivieren für `main`, `release/*`, ggf. `hardware/main`

GitHub Actions

- Automatischer PDF-Build für Doku (LaTeX, Markdown → PDF)
- Automatisches Hochladen von Build-/Exportdateien als **Release Asset**

Git LFS

- Große Dateien wie `.sch`, `.kicad_pcb`, `.pdf`, `.zip` via `git lfs track` verwalten
- Achtung: LFS-Limits auf GitHub beachten (1 GB Speicher, 1 GB Transfer frei)

Empfohlene Arbeitsweise

Branching

- `main`: stabil, veröffentlichte Versionen
- `dev`: für Entwicklung, Tests
- `hw/feature-*`: für Hardwareänderungen
- `doc/update-*`: für Dokumentation

Commit-Stil

- Commit-Messages klar trennen: `feat(code):`, `fix(hw):`, `docs(readme):`

Releases

- Für veröffentlichte Versionen `Release` auf GitHub anlegen
- Gerber-Dateien, PDFs, Executables als Release-Asset anhängen

---

Best Practices

- `.gitattributes` + `.gitignore` sauber führen (insb. für `.pdf`, `.sch`, `.zip`, etc.)
- CI-Prozesse per GitHub Actions automatisieren (z. B. PDF-Doku)
- Diskussionen zu Hardware/Doku über GitHub Issues führen
- Repository-Beschreibung & Topics pflegen (sichtbar in GitHub-Suche)
- Optional: Wiki oder Pages für veröffentlichte Doku nutzen

Ein GitHub-Repo ist ideal für interdisziplinäre Projekte – mit klarer Struktur und gezielter Nutzung der GitHub-Features bleibt alles nachvollziehbar und koordiniert.