**🔑 Git Push – Sprint-rutine (solo)**

git status # 1) sjekk om noe er endret

git add -A # 2) legg til alt

git commit -m "[S4] ..." # 3) kort logg (hoppes over hvis clean)

git push # 4) dytt til sprint-branchen

**Sprint 4 – Kalibrering (🎯)**

**Kontekstblokk**

* **Fra S2 (Vær & profiler):** Værklient med caching og profilsettings er på plass; CLI integrert. DoD oppnådd med ≥95 % cache-hit, stabile tall (±1–2W).
* **Fra S3 (Fysikkmotor):** Kraftmodell (gravitasjon, rulling, aero, akselerasjon) ferdig; smoothing og golden test i CI. Output deterministisk (±1–2W).
* **Avhengigheter:** Kalibrering (S4) bygger videre på physics-output (sample-watt, NP/avg, smoothing).
* **Neste steg (Frontlog):** Indoor-pipeline (S5), Reports & observabilitet (S6).

**Sprintbeskrivelse**

**Oppgaver:**

1. Kalibreringsprosedyre (5–8 min, 3–6 % bakke).
2. Fit CdA/Crr fra data (uten powermeter).
3. Lagre pr sykkel/profil; bruk globalt i beregninger.
4. Test med syntetiske segmenter med kjent CdA/Crr (nytt).

**DoD (må alle bestås):**

* Reproducible fit på testdata.
* MAE ≤10 % på kalibreringssegment mot powermeter-aktivitet.
* Flagget “Kalibrert: Ja/Nei” i output.
* ✅ Må være reproduserbar med smoothing aktivert.
* ✅ Avhengighet til S2→S3: bruke physics-output (sample-watt, NP/avg, smoothing) som baseline.

**Estimert tid:** 12–18h

**Arbeidsmetode (for Copilot)**

**0) Lag en plan først (obligatorisk):**

* Presenter **5–7 trinn** i en punktliste *før* du skriver kode.
  + For **hvert trinn**: Filer + hvorfor (1–2 setn), Forventet output (tester/CLI), Risiko/avhengigheter (kort).
* **Vent på “OK PLAN”** før du starter Trinn 1.

**1) Kjør trinnvis etter godkjent plan:**  
For hvert trinn:  
a) **Vis diff** (filnavn + essensielle endringer).  
b) **Kjør tester** (cargo test -q / pytest -q) og rapporter kort.  
c) **CLI/Output**: vis forventet felt/eksempel (om relevant).  
d) **Vent på “OK <trinn>”** før neste trinn.

**2) Stop-regler:**

* Stopp og be om avklaring hvis endringen påvirker **>3 filer**, **public API/CLI**, eller krever **nye avhengigheter**.
* Ved testfeil: **minimer scope**, fiks enkleste feil først, kjør tester igjen.

**3) Debug-partner:**

* **Bruk Chat** som debugging-partner ved feil eller uklarheter. Del feilmelding og kort forklaring på hva du prøvde.

**4) Ferdigkriterium:**

* Alle DoD verifisert med tester/CLI og deterministisk output.

**Svarformat (mal) Copilot skal bruke:**

* **PLAN (5–7 trinn)** → vent på “OK PLAN”.
* **TRINN 1**: diff → tester → kort output → vent på “OK 1”.
* **TRINN 2** … (gjenta til ferdig).

**Sprint-delta (mini-kontekst)**

* Fysikkmotor (S3) er levert og stabil (golden ±1–2W).
* Kalibrering bygger videre på physics-output.
* Resultatet markerer første steg mot bruker-tilpassede profiler.

**Tekniske rammer (kontrakt)**

* Input fra compute\_power (samples, profile, weather).
* Kalibreringssegment: 5–8 min, bakke 3–6 % stigning.
* Fit-metode: bruk differanse mellom forventet (physics) og observert hastighet/høyde for å estimere CdA/Crr.
* Output: oppdatert Profile med felt calibrated: bool, samt lagring pr sykkel/profil.
* CLI skal vise felt calibrated: "Ja/Nei".

**Endringsgrenser (halvfast default)**

* **Tillatt:** core/src/calibration.rs (ny), core/src/lib.rs, cli/\*.py, tests/\*\*.
* **Ikke rør:** Strava-API, CI-workflow utenom testutvidelse, fysikkmotor (S3).

**Tester / kommandoer**

* pytest -q
* cargo test -q
* python -m cli.analyze --dry-run --activity <id>

**Verifikasjonsplan (DoD → test)**

1. Golden test med syntetiske segmenter (kjente CdA/Crr) → fit gjenskaper verdier.
2. Reell test: kalibreringssegment gir MAE ≤10 % mot powermeter.
3. CLI dry-run viser “Kalibrert: Ja/Nei”.
4. cargo test verifiserer kalibreringsalgoritmen.
5. pytest verifiserer CLI-integrasjon.
6. Output deterministisk ved gjentatt kjøring (samme input → samme fit-resultat).

**Forventet output**

* core/src/calibration.rs – ny modul for CdA/Crr-fit.
* core/src/lib.rs – eksponere kalibreringsfunksjon.
* cli/analyze.py – legge til felt “Kalibrert: Ja/Nei” i output.
* tests/ – syntetiske tester (kjent CdA/Crr), pytest på CLI-output.

**Frontlog (kommende oppgaver)**

* S5: Indoor-pipeline (🧪).
* S6: CLI/Reports & observabilitet (📈).
* S7: QA & hardening (🛡️).

✅ Sprint: [ID – Navn]

Branch: [navn]

Commits: [hash – message]

Endrede filer: [paths]

Tester: [pytest + cargo test resultater]

Observasjoner: [korte punkter]

Status: [Ferdig / Delvis ferdig / Feil gjenstår]

**Sluttføring av Workflow (ChatGPT v2)**

1. **Sluttrapport:** lim inn Copilot-rapport i template.
2. **Delta-sammendrag:** skriv 2–3 linjer etter komprimeringsmal:
   * Hva ble gjort.
   * Teststatus.
   * Observasjon.
3. **Evaluering & Oppdatering:** oppdater DoD og Frontlog basert på nye funn/innsikter.
   * Klassifiser funn som A (DoD), B (Frontlog), C (Observasjon).
   * Marker ferdige sprinter, flytt avhengigheter.
   * Juster Sprintplan om nødvendig.
4. **Synk Masterplan:** oppdater statuslinje eller legg til ny rad.
5. **Oppdater Sprint Log:** kort notat med status og carry-over.
6. **Template neste sprint:** sjekk om “halvfast default” trenger justering og fyll ut template basert på neste sprint.