Projektiarkkitehtuuri — luonnos

Tämä dokumentti kuvaa Production_Tester -projektin korkean tason arkkitehtuurin. Se on luonnos — kommentoi, niin täydennän ja muokkaan.

Yleiskatsaus

Järjestelmä on moduulirakenteinen Python-sovellus, jonka pääosat ovat käyttöliittymä, testauksen ohjauslogiikka, simulaattori ja laiterajapinnat. Projektin kansiorakenne avainosineen:

- `Code/` — ydintoiminnallisuudet: `spec_loader.py`, `test_runner.py`, `test_worker.py`, `ui_logic.py`
- `UI/` — graafinen käyttöliittymä: `main_window.py`, `start_button.py`, `oscilloscope_widget.py` jne.
- `Code/simulator/` — simulaattorin komponentit: `simulator.py`, `measurement_circuit.py`, `noise_model.py` jne.
- `Code/hardware/` — laite-rajapinnat ja kaupalliset mittalaitteet
- `Robot/` — Robot Framework -avainsanat ja testit

Komponentit ja vastuut

- Käyttöliittymä (`UI/`)
- Vastaa käyttäjän vuorovaikutuksesta, testien käynnistyksestä ja tulosten näyttämisestä.
- Kommunikoi sovelluslogiikan kanssa `ui_logic.py`-rajapinnan kautta.

- Sovelluslogiikka (`Code/`)
- `spec_loader.py`: lataa testit ja raja-arvot (Spec/limits.json).
- `test_runner.py`: orkestroi testin ajon, kutsuu `test_worker`-prosessia tai säiettä ja kerää tulokset.
- `test_worker.py`: suorittaa yksittäiset mittaukset ja kommunikoi laitteiden tai simulaattorin kanssa.

- Simulaattori (`Code/simulator/`)
- Mahdollistaa laitteiden ja signaalien simuloinnin kehitystä ja CI:tä varten.

- Laiterajapinnat (`Code/hardware/` ja `Code/hardware/interfaces/`)
- Määrittelevät abstraktion kaupallisille mittalaitteille ja tuotteille.

- Robot & testiautomaatio (`Robot/`)
- Avainsanat ja testit Robot Frameworkille; käyttö integraatio- ja UI-testeihin.

Tiedonkulku

1. Käyttäjä käynnistää testin UI:sta (`UI/main_window.py`).
2. UI kutsuu sovelluslogiikkaa (`ui_logic.py` → `test_runner.py`).
3. `test_runner` lataa speksit `spec_loader.py` ja käynnistää workerit (`test_worker.py`).
4. Workerit käyttävät laite-rajapintaa; kehityksen/CI:n aikana ne voivat käyttää simulaattoria (`Code/simulator/`).
5. Tulokset palautuvat `test_runner`-tasolle, jossa ne tallennetaan/raportoidaan (esim. `output.xml`, `report.html`).

Rajapinnat ja laajennettavuus

- Laiteohjainrajapinnat on toteutettu erillisinä luokkina moduulissa `Code/hardware/interfaces/`, mikä mahdollistaa uusien laitteiden lisäämisen ilman ydinkoodin muutoksia.
- Simulaattori seuraa samaa rajapintaa kuin fyysiset laitteet, joten testit voidaan ajaa kumpaankin.

Riippuvuudet ja ajonaikaympäristö

- Python 3.x; riippuvuudet listattu `requirements.txt`.
- Robot Framework -testit löytyvät `Robot/`-kansiosta.

Ei-funktionaaliset vaatimukset (luonnos)

- Suorituskyky: testien rinnakkainen ajo `test_worker`-instansseilla.
- Robustius: selkeät virheenkäsittelypolut laite- ja simulaatioyhteyksissä.
- Testattavuus: simulaattori mahdollistaa deterministisen testiajon CI:ssä.

Deployment ja kehityskäytännöt

- Sovellus voidaan asentaa Python-ympäristöön projektin `requirements.txt` avulla.
- UI voidaan käynnistää paikallisesti `UI/run_ui_test.py` -ohjelmalla (katso README:t ja Robot-testaus `Robot/`).

Seuraavat askeleet

1. Käy läpi tämä luonnos ja kerro puuttuuko tietoja tai haluatko laajempia kaavioita.
2. Voin generoida PDF:n ja renderöidä PlantUML-kaavion valmiiksi, jos haluat (tarvitaan `pandoc`/LaTeX ja `plantuml`).