



A Web-Based Data Acquisition System for Characterization of Particle Detectors

Kevin Mota, on behalf of the LFNP team

Laboratório de Física Nuclear e Partículas (LFNP) — Universidade do Estado do Rio de Janeiro (UERJ)
kevin.amarilo@cern.ch



Abstract

We present a modular data acquisition (DAQ) system developed for the automated characterization of particle detectors at the Nuclear and Particle Physics Laboratory (LFNP). The system integrates control of CAEN high-voltage power supplies with a finite-state machine (FSM) engine to orchestrate scans across multiple detector channels. The architecture comprises three services: a web-based frontend, a REST API backend with a relational database, and an asynchronous DAQ engine. Each scan runs as an independent FSM instance. Scans can be started, stopped, paused, and resumed remotely, and multiple scans can run concurrently on independent detector channels. The system is deployed via Docker containers, being reproducible across many systems. The DAQ system is currently validated in offline testing and being prepared for integration with CAEN hardware.

1. Motivation

Characterizing particle detectors (e.g. SiPMs, PMTs, gaseous detectors) requires sweeping high voltage across many channels while recording currents and signals from the detector at each operating point. Done by hand, this is slow, error-prone, and hard to reproduce across sessions and users.

Web-based DAQ and control systems have gained significant traction in particle physics. Frameworks such as MIDAS [1] provide comprehensive web management for experiment-wide data acquisition, while tools like WebDCS [2] demonstrate the efficacy of browser-based control systems for detector monitoring. Building upon these concepts, LFNP needed a lightweight, dedicated system that:

- runs unattended, multi-hour scans reliably;
- drives several detector channels *concurrently* and independently;
- is controllable and monitorable remotely, from any device on the lab network;
- keeps a full, timestamped record of every run for later analysis;
- is easy to deploy identically on every lab workstation.

2. System Architecture

The system follows a three-tier, service-oriented design (Fig. 1), fully containerized with Docker.

- **Frontend** — Svelte 5 / SvelteKit single-page app for configuring scans, watching live data, and browsing past runs.
- **Backend API** — FastAPI service with a PostgreSQL database (SQLModel ORM); owns users, runs, and device configuration, issues JWT-authenticated REST endpoints, and proxies control commands to the DAQ engine.
- **DAQ Engine** — an asynchronous FastAPI service that owns the hardware: it hosts the FSM instances, the scan logic, the CAEN hardware drivers, and a WebSocket broadcaster that streams live data straight to the frontend.

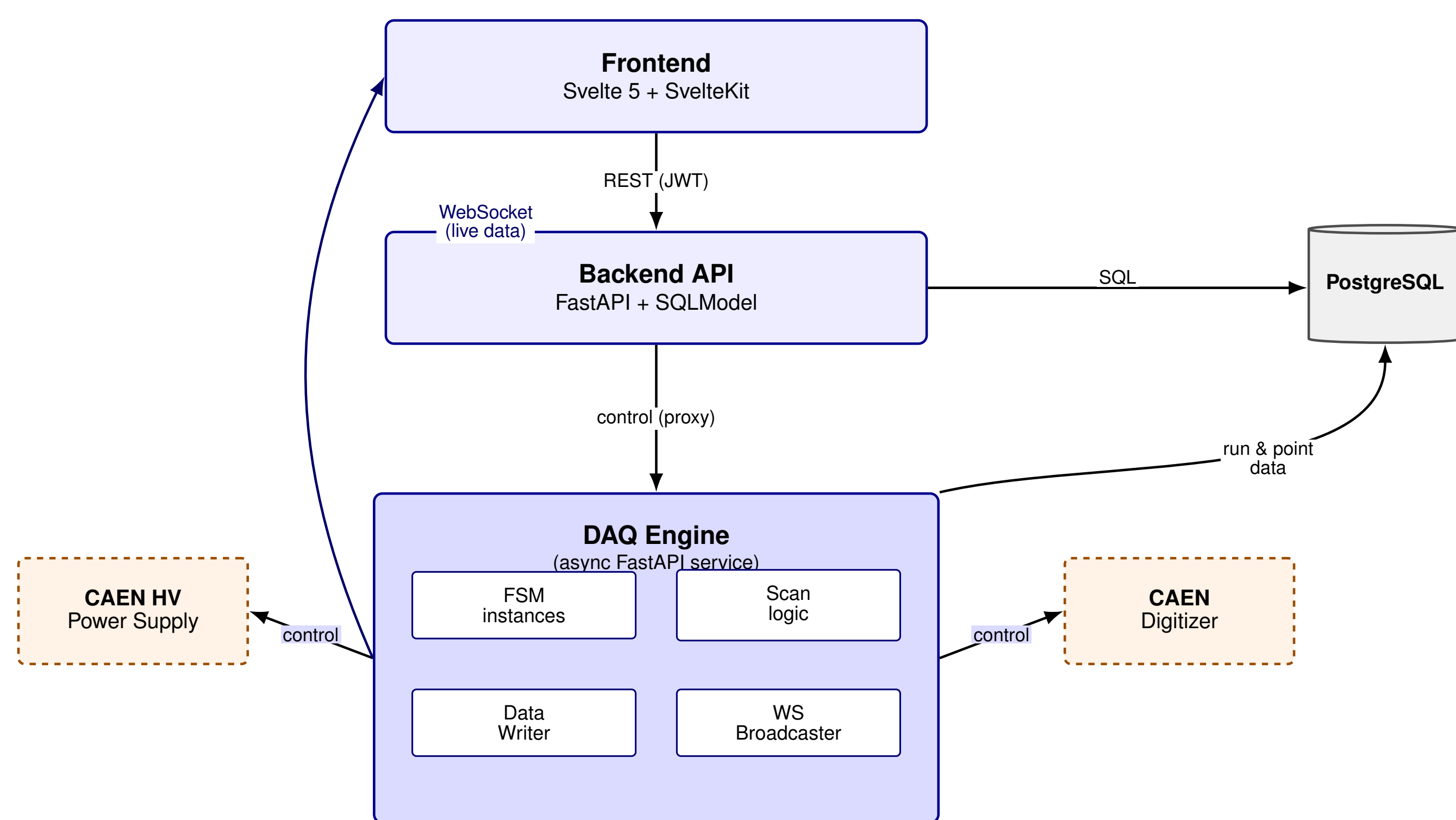


Figure 1: Three-tier architecture: frontend, backend/DB, and the asynchronous DAQ engine, which talks directly to the hardware and streams live data over WebSocket.

Runs, points, and settings are persisted in PostgreSQL; raw run data is written to a shared volume so it survives container restarts.

3. Scan Engine: A Finite-State Machine

Every scan is executed by its own DAQFSM instance (Fig. 2), they can progress independently and concurrently on different channels without interfering with each other. The FSM walks through CONFIGURING → WAITING → RECORDING for every voltage point in the scan, logging every transition to a per-run log file. The user have access to commands that affects the progress of the scan: `pause()` halts progress without discarding the run; `resume()` redoes the current point from scratch to guarantee clean data. `stop()` returns the FSM to HALTED at any point, ramping the supply back down safely. This design lets the operator control a run remotely at any time.

The FSM architecture is highly versatile and can be adapted to alternative operational workflows, such as long-term stability runs at fixed operating points and automated gain calibrations.

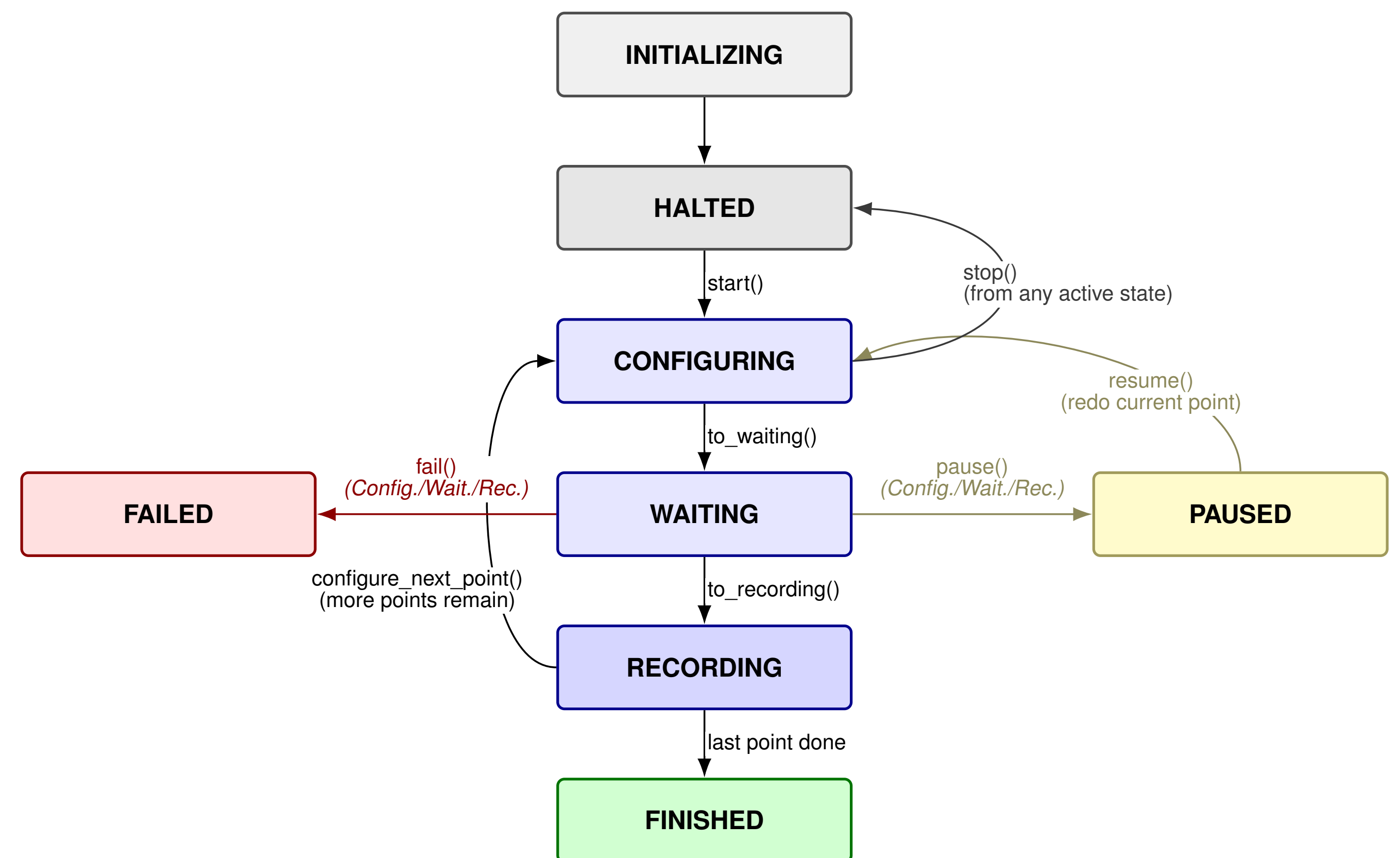


Figure 2: DAQ finite-state machine. A scan cycles through configuring, waiting, and recording for each voltage point; it can be paused/resumed or stopped remotely at any time, and transitions to FAILED on error.

4. Status & Validation

The FSM engine, REST/WebSocket stack, and scan orchestration logic have been validated for use with CAEN HV power supplies and digitizers (DT5742 and DT5743) on the bench at LFNP.

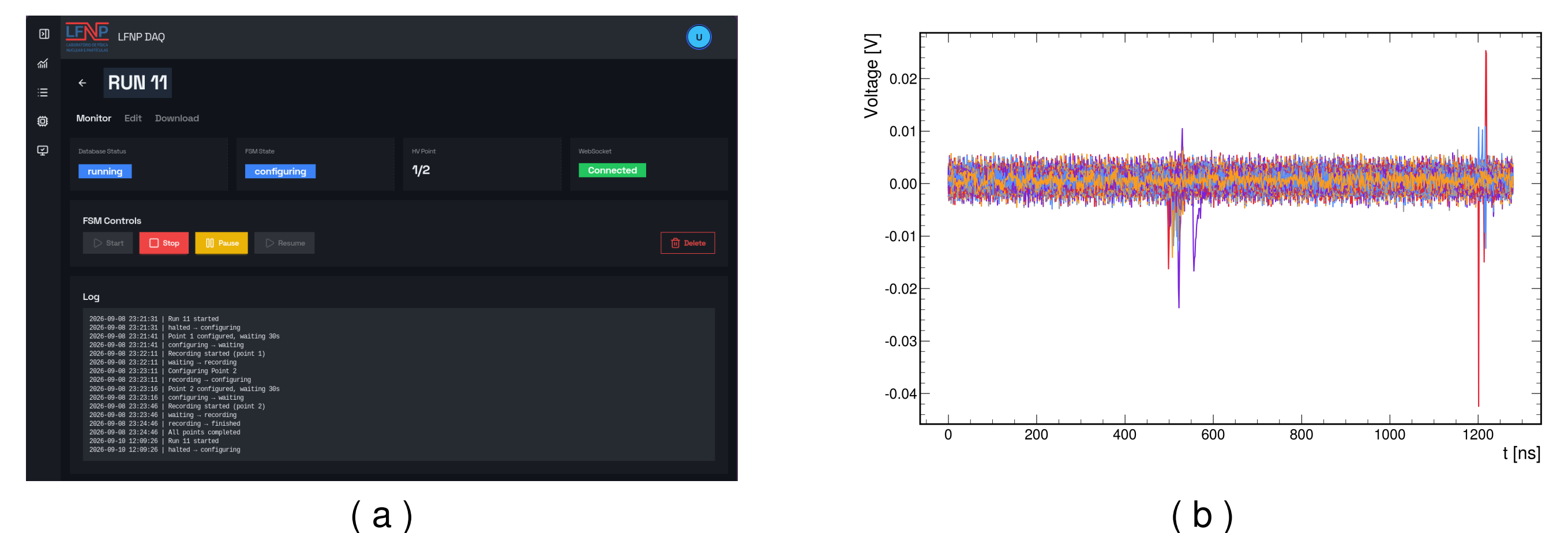


Figure 3: Results of the validation of the system using a RPC detector in LFNP lab. (a) Run detail page with important information for monitoring and controls for the FSM; (b) Data recorded from a RPC detector using a DT5743 digitizer.

The whole stack (frontend, backend, database, and DAQ engine) is defined in a single docker-compose file, so the exact same environment can be brought up with one command on any lab machine, a laptop, or a dedicated acquisition PC. Hardware access (USB digitizers, HV supplies) is passed through to the DAQ container, while configuration is driven entirely by environment variables. The stack is simple to redeploy the system after a lab PC is reinstalled or replaced.

5. Conclusions & Future Work

- A modular, three-service architecture with a per-run FSM gives LFNP a DAQ system that is remotely controllable, supports concurrent multi-channel scans, and keeps a full audit trail of every run.
- Containerized deployment makes the system reproducible across the lab's heterogeneous set of workstations.
- **Next steps:** add other lab's hardware (such as CAMAC and VME QDCs and TDCs) to the system, create a better ways to visualize the live data and automatically run analysis scripts.

References

- [1] MIDAS (Maximum Integrated Data Acquisition System). https://daq00.triumf.ca/MidasWiki/index.php/Main_Page
- [2] WebDCS: A web-based Detector Control System framework. *Proceedings of Science (PoS)*, RPC2024. <https://pos.sissa.it/476/952/>
- [3] K. Mota, *lfnp_daq* source code repository. https://github.com/kevimota/lfnp_daq
- [4] FastAPI documentation. <https://fastapi.tiangolo.com>
- [5] Svelte 5 / SvelteKit documentation. <https://svelte.dev>
- [6] CAEN S.p.A., *caen-libs* Python bindings for CAEN digitizers and power supplies. <https://www.caen.it>

Thanks to all colleagues involved in this work and to

