PROJECT ABSTRACT

Sensor Data Replay Tool for IoT Testing

A deterministic replay console that streams recorded sensor datasets to firmware and cloud code under test, with transport controls, fault injection and an MQTT publish log.

Project type: Software

Availability: Built to order

Suitable branches: IoT & Embedded, Computer / IT

Technologies: JavaScript, HTML5 Canvas, MQTT, CSV data pipelines, sensor simulation

Price: Request a quotation

Prepared by Projectech · projectech.in — for academic project reference.

1. Introduction / Background

Testing IoT firmware and cloud ingestion code against live sensors is slow and non-deterministic: a freezer fault happens once at 3 a.m., and the developer misses it. Professional hardware-in-the-loop rigs solve this but cost far more than a student budget.

This project takes the recorded-data approach: sensor sessions are captured once as CSV, then replayed deterministically - with original timestamps and gaps preserved - so the device under test cannot tell replay from live sensors. Faults (spikes, dropouts, drift) are injected on demand, and every replayed sample can be published over MQTT exactly as a real node would.

2. Problem Statement

Firmware and cloud developers need repeatable sensor stimuli to verify edge cases: threshold crossings, sensor dropouts, clock jumps and slow calibration drift. Live testing cannot reproduce these on demand, and hand-written mocks rarely preserve real-world noise and timing.

This project provides a replay console: load a recorded CSV dataset, stream it at 0.5x-8x speed with a timeline scrubber, inject scripted faults, and watch the device-under-test respond - with a full MQTT publish log and pass/fail test scenarios to close the loop.

3. Objectives

- Stream recorded sensor CSVs deterministically with timestamps and gaps preserved.
- Provide transport controls: play/pause, speed selection, +/-1 minute seek and timeline scrubbing.
- Support on-demand fault injection: temperature spikes, sensor dropouts and humidity drift.
- Publish replayed samples over MQTT with a visible publish log.
- Ship realistic seeded datasets (cold storage 24 h, motor vibration run, greenhouse week) and automated test scenarios with expected device responses.

4. Proposed Methodology

- Capture or generate sensor sessions as timestamped CSV files with one column per channel plus anomaly flags.
- Parse datasets into an in-memory sample table; the replay engine advances a playhead at the selected speed multiplier.
- Apply fault scripts as overlays on the base data (additive spikes, null dropouts, slow drift) without mutating the source file.
- Render live channel values, a scrolling chart window and the timeline; publish each sample to the configured MQTT topic.
- Execute scripted test scenarios that replay a slice with a fault and check the device/cloud response against the expectation.

5. System Architecture / Working

The tool is a single-file web app. The data layer holds the seeded datasets (1,440-sample cold-storage session, 480-sample vibration run, 2,016-sample greenhouse week) and the CSV import/column-mapping logic. The replay engine is a deterministic scheduler driving the playhead, fault overlays and MQTT publisher.

The presentation layer has three views: the replay console (transport, live values, chart, MQTT log), the datasets view (session table plus CSV preview), and the scenarios view (automated tests with pass/fail/queued states). The device under test subscribes to the same MQTT topic a real sensor node would use.

6. Hardware / Software Requirements

- Software: any modern web browser; an MQTT broker for live publishing (the public test broker test.mosquitto.org is used in the demo, or any local broker such as Mosquitto).
- Datasets: three seeded CSV sessions included; any user CSV with a timestamp column and numeric channels can be imported.
- No sensor hardware is required - that is the point of the tool.

7. Expected Outcomes

- A working replay console with deterministic, timestamp-preserving playback at 0.5x-8x speed.
- Three realistic seeded datasets with tagged anomalies (defrost cycles, bearing-fault vibration, irrigation events).
- Fault injection for spikes, dropouts and drift, with a live MQTT publish log.
- Five automated test scenarios mapping replayed faults to expected device/cloud behaviour.

8. Applications

- Firmware testing for IoT sensor nodes without bench hardware.
- Cloud ingestion and alerting pipeline validation.
- Regression testing of sensor-handling code in CI pipelines.
- Teaching deterministic testing practices in embedded courses.
- Demonstrating edge-case behaviour (dropouts, spikes) in project vivas.

9. Future Scope

- Record mode: capture live MQTT streams into new replayable datasets.
- Scriptable fault timelines with a visual fault-track editor.
- Multi-node replay with synchronised timestamps across a virtual fleet.
- Assertion engine that auto-grades device responses against expectations.
- Export of replay sessions as test fixtures for pytest/CI integration.