M2025 IoT
Baris Baser & Kai Suematsu

IoT M2025: Raspberry Pi Biometrics

Introduction / Background

We built an end-to-end Raspberry Pi–based biometric logging and dashboard system to correlate indoor CO_2 levels with physiological metrics (heart-rate) for a research demo in our CMU IoT course. By combining a Sensirion SCD-30 CO_2/temperature/humidity sensor with Apple Watch–derived heart-rate exports, we aimed to fuse environmental and biometric data streams, visualize trends in Streamlit, and compute simple health and sleep scores. This system had to run reliably under systemd, handle live SSH debugging, and scale to multi-day logging, all while being easy to operate during a live presentation.

Code: https://github.com/bariswheel/co2biologger
Drive Folder With Biomarker & CO2 Data: IOT_2025
Devops Runbook for Raspberry Pi System End-To-End: Raspberry Pi Biometrics Runbook (7/29/25 last update)

Challenges in the Raspberry Pi Biometrics Workflow

Below is a comprehensive list of all the problems encountered during development.

1. **Background logger resiliency**: Needed to add retry-on-CRC-failure logic to the SCD-30 logger so isolated hardware hiccups don't crash the service.
2. **Live heart-rate tailing**: Uncertainty around how to `tail -f` a continuously appending CSV for heart-rate logs and include that in a runbook.
3. **Pretty-printing JSON/CSV in CLI**: Sought a Linux tool to colorize and format raw Python-style JSON blobs and large CSVs over SSH (jq, bat, csvkit) to make sense of csv files with json key value pairs.
4. **Raw CSV schema confusion**: Discovered the biomarker CSV has only `data,time` columns where `data` holds nested JSON, not discrete heart-rate fields.
5. **CSV field-size limit**: Hit Python's default 128 KB field limit when parsing giant daily CSVs, leading to `_csv.Error: field larger than field limit`.
6. **ColumnIdentifierError in csvcut**: Misunderstood CSV layout and attempted to cut non-existent `metrics`column, causing tooling errors.
7. **Export settings debate**: Verified biomarker automation settings (no aggregation, 1 min cadence) matched the observed CSV behavior.
8. **Service port conflict**: biologger.service repeatedly failed because port 5000 was already bound by a manually launched uvicorn.
9. **Manual vs. systemd testing**: Learned to run `health_ingest.py` manually and free port 5000 before letting systemd restart the service.
10. **ISO timestamp parsing**: Fusion script's `pd.to_datetime(..., utc=True).dt.tz_convert(None)` failed on mixed-format ISO strings lacking timezone offsets.
11. **Sparse HR matches**: Initial lenient `merge_asof` yielded very few `hr_bpm` matches, prompting adjustments to tolerance and merge logic.
12. **Single-day fusion**: fusionv2.py only processed the newest CO_2 file for one date (7-28) and skipped newer raw data (7-29) due to date-matching logic.
13. **Flattening nested JSON**: `flatten_hr` struggled with Python-style dict-strings containing smart quotes and varying key orders, resulting in JSONDecodeErrors.
14. **Indentation and syntax errors**: Repeated fixes to fusionv3.py required full rewrites to resolve indentation, import, and exception-handling issues.
15. **Timezone mismatches**: Merge errors from comparing naive vs. timezone-aware datetimes in CO_2 vs. HR data caused `MergeError` exceptions.
16. **Multiple CO_2 file handling**: Needed to glob and combine all CO_2 JSON logs for a day rather than selecting only the last file.
17. **Streamlit deprecations**: dashboardv2.py used `st.experimental_rerun()` and `st.experimental_get_query_params()`, which were removed and causing crashes.
18. **Dashboard enhancements**: Added separate CO_2 vs. time, HR vs. time, and overlay charts plus metrics and download buttons to the Streamlit app.
19. **Axis zoom control**: Wanted default zoom for the correlation chart (e.g., x starting at 500 ppm, y at 40 bpm) using Altair scales.
20. **Runbook formatting**: Consolidating and formatting dozens of commands into a coherent runbook.

Top 3 Challenges & Analysis

1. Parsing and Flattening Gigantic Biomarker CSVs

The raw biomarker export produces one enormous CSV per day, embedding a Python-style dict-string in a single `data`column. Python's CSV reader repeatedly chokes on the 128 KB default field size, and naive `ast.literal_eval` or JSON loads falter on inconsistent quoting and escape sequences. We solved this by raising `csv.field_size_limit(sys.maxsize)`, copying the live file to a temporary static snapshot, and writing a robust `flatten_hr()` that safely normalizes smart quotes, handles missing keys, and iterates over all date–Avg pairs.

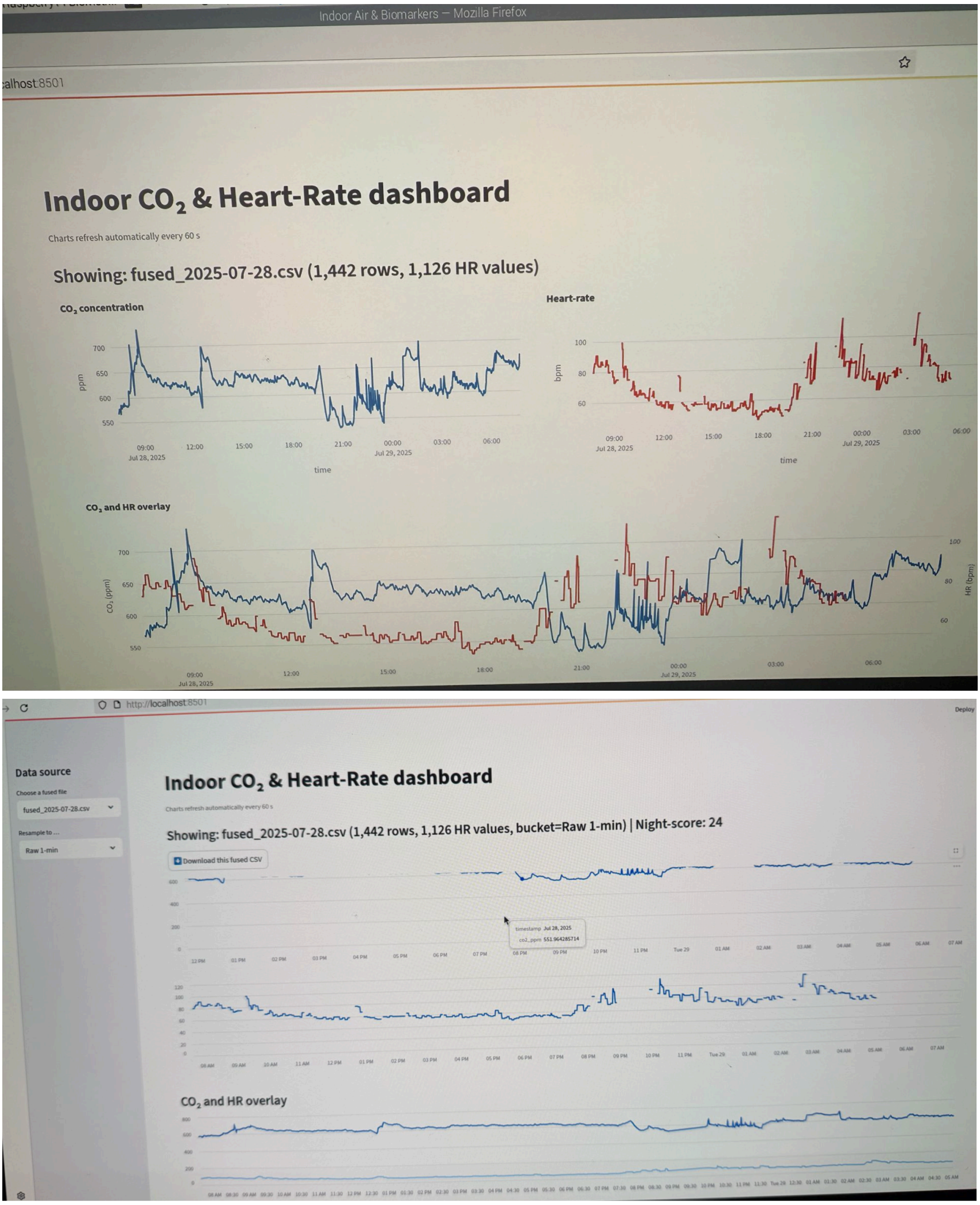
2. Service Startup Conflicts and Manual Testing

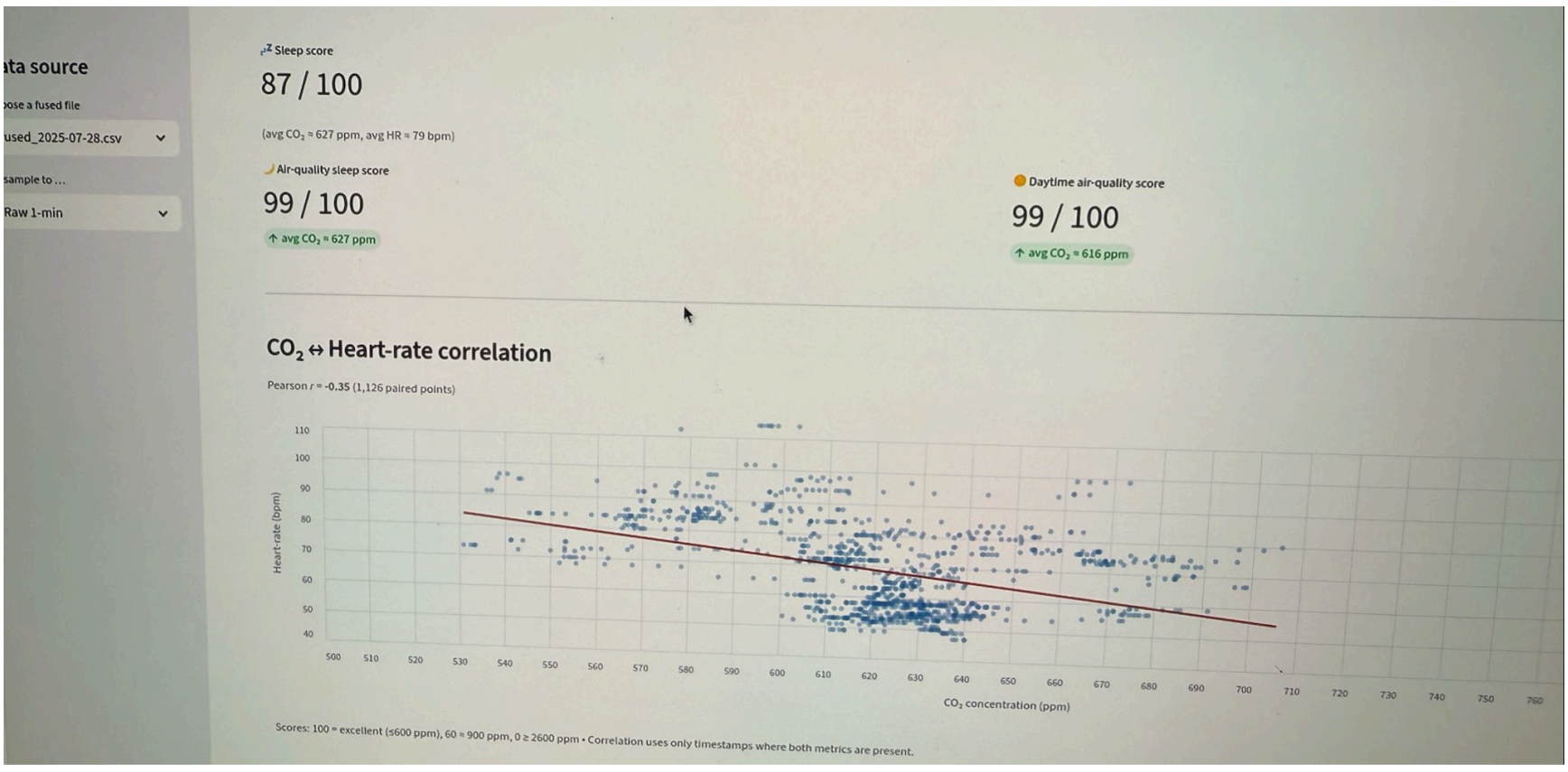
Automating ingestion under systemd initially failed because a stray `uvicorn health_ingest` process was already bound to port 5000. systemd dutifully retried and then gave up. The key lesson was to always manually test your FastAPI app — `uvicorn health_ingest:app --host 0.0.0.0 --port 5000` — and kill any runaway processes (`pkill -f uvicorn`) before flipping back to systemd. We updated our runbook to include explicit `pgrep`/`kill` steps, `lsof -i :5000` checks, and a temporary port flag (`--port 5050`) for safe dev.

3. Multi-day Fusion Logic Bug

The fusion script originally globbed only the newest CO_2 JSON and one raw CSV per run, ignoring subsequent days' data. As a result, running on July 29 still fused only July 28's data. Addressing this required looping over every `bio_YYYY-MM-DD.csv` in `raw/`, matching against all `co2_{YYYY-MM-DD}T*.json` for that date, and writing one fused file per day into `data/fused/`. We updated the glob patterns, sorting, and output paths accordingly, enabling fully automated, multi-day batch fusion.

Dashboard Screenshots and Data Visualizations of Tester Subject





Conclusion

We successfully built a robust end-to-end IoT pipeline on a Raspberry Pi to correlate indoor CO_2 measurements with heart-rate data. Key accomplishments include resilient sensor reading (with CRC retry logic), fully automated multi-day data fusion, and an interactive Streamlit dashboard featuring resampling controls, download capabilities, and sleep/air-quality scoring. We've demonstrated systems thinking in IoT design, using aggregation and bins, offline tolerance, and demonstrated mature engineering.

Future Expansions and Novel Directions

- **Quality Confidence Scoring:** Introduce a dynamic confidence metric that weights data by CO_2 variance, HR availability, and sensor events, giving real-time reliability insights.
- **Event-Driven Alerts:** Add anomaly detection for rapid biomarker drops or CO_2 spikes, shifting from fixed-interval sampling to an event-driven architecture that surfaces critical conditions immediately.
- **Advanced Biometrics Integration:** Expand beyond heart-rate to include noise detection via Raspberry pi microphones, HRV, SpO_2, and stress markers from wearables, deepening physiological context for air-quality correlations.
- **Predictive Modeling:** Apply time-series forecasting (e.g. LSTM) to anticipate CO_2 fluctuations or physiological changes, enabling proactive ventilation control or personalized health prompts.
- **Smart Building Integration:** Expose the fused data via REST APIs to connect with building management systems (BMS), closing the loop for automated HVAC adjustments based on live occupancy health metrics.
- **Spatial Multi-node Analysis:** Deploy multiple Pi nodes across zones combined with wearable localization to map occupant-specific air-quality interactions and identify localized hotspots.

Collectively, these innovations position our system as a compelling foundation for both academic research and commercial productization, pushing the boundaries of environmental and health IoT with a focus on reliability, real-time insights, and actionable intelligence.