AN EDGE-CENTRIC IOT SYSTEM FOR SMART BUILDING ENERGY MANAGEMENT

João Silva, João Alves, Nuno Silva, Nuno Moreira, Abdul Rauf,
Victor Rodriguez, Victor Bongard, Rui Pinto, Gil Gonçalves

Presenter: João Silva
Email: joao.pvd.silva@gmail.com

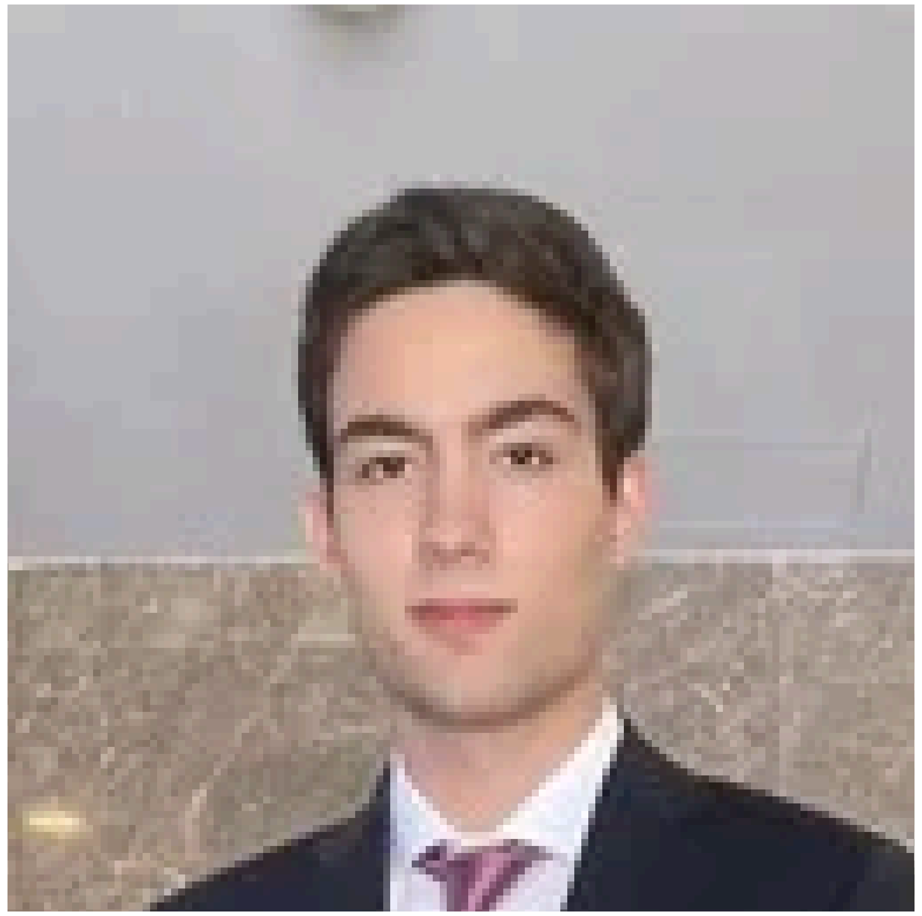
IARIA

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

PRESENTER

João Silva

Master's student at FEUP enrolled in the Cyberphysical and Internet of Things course



Expirience

- Bachelor in Informatics and Computing Engineering
- Currently pursuing Masters in Informatics and Computing Engineering
- Summer Internship at Critical Manufacturing

Involvement

- Developed edge layer components
- Contributed to middleware layer design
- Worked on cloud backend development
- Participated in initial cloud frontend development

Contact

joao.pvd.silva@gmail.com

OUTLINE

- **Context and Problem Definition**
- **Proposed Solution**
- **Arquitecture**
- **Implementation**
  - Frontend
  - Backend
  - Grafana/InfluxDB
  - RaspberryPi
  - ESP32
  - WeatherAPI
  - Occupancy Prediction
  - Hybrid Decision Logic
- **Evaluation and Validation**
- **Impact**
- **Limitations and Future Work**

CONTEXT AND PROBLEM DEFINITION

- Buildings and industrial facilities consume large amounts of energy through HVAC and lighting.
- Traditional BMS rely on static schedules and centralized control.
- This leads to:
    - Limited real-time visibility
    - Energy waste
    - Low resilience
- IoT and CPS technologies enable fine-grained sensing and adaptive control, but many solutions remain cloud-dependent.
- **Goal**: design a resilient, edge-centric IoT system capable of monitoring and controlling environmental conditions locally while supporting cloud supervision.

PROPOSED SOLUTION

ONE-LINER DESCRIPTION

Edge IoT system for room occupancy, lighting, and HVAC control with local resilience and cloud dashboards

CONTEXT

- Industrial plants face rising energy costs and inefficient consumption due to lack of real-time control
- Traditional systems don’t monitor or optimize usage dynamically
- There’s a need for IoT-based visibility and automation to reduce waste and support sustainability

PROBLEM/NEED

- Rising energy costs; lack of real-time visibility/control
- Need to keep operating even if the Pi goes offline
- Want proactive actions (occupancy/weather forecasts) not just monitoring

ARQUITECTURE

The architecture follows a layered IoT/CPS structure to separate physical, communication, processing and business domains. This does:

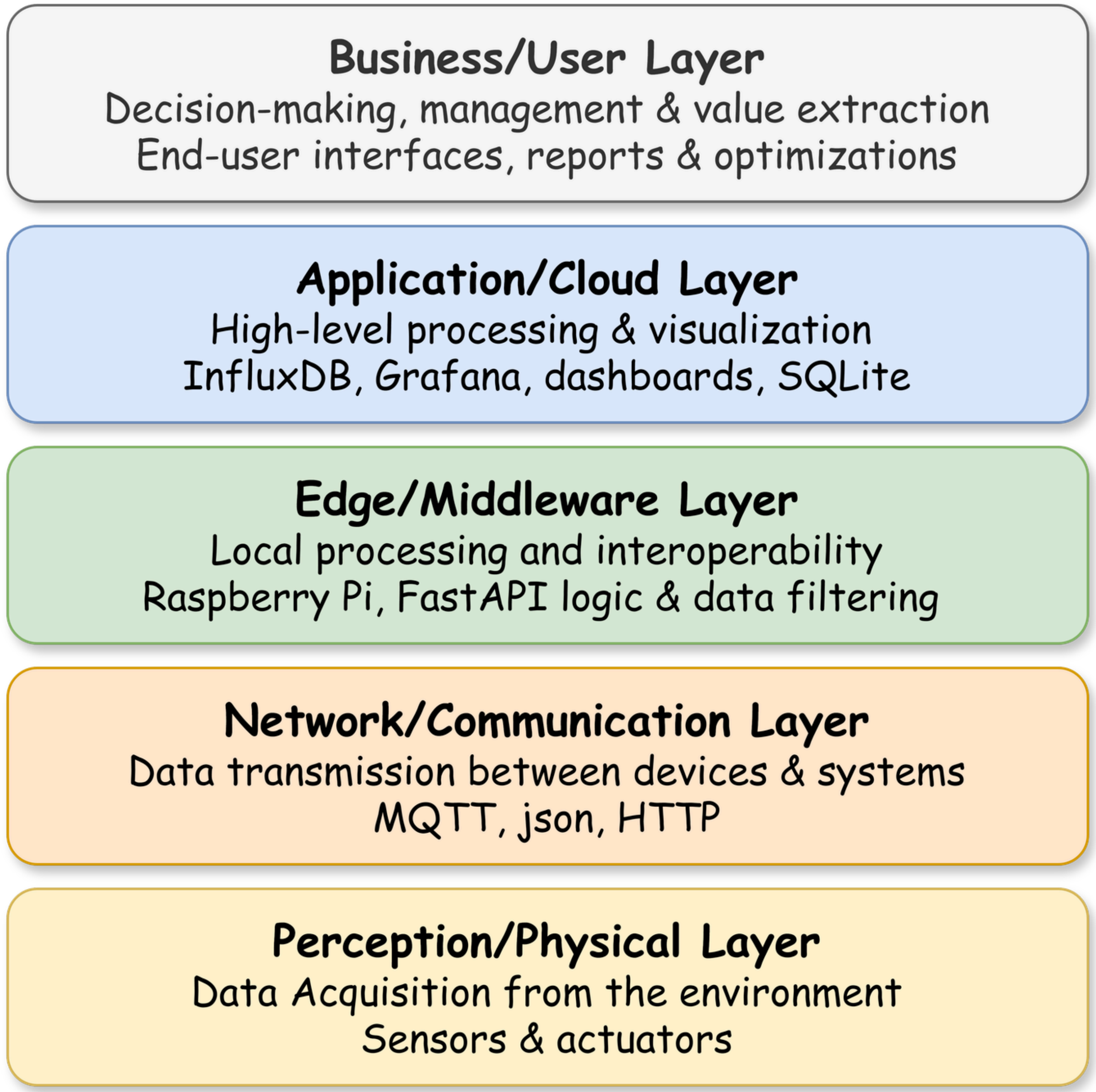
- Promotes modularity and independent scalability per layer
- Enables data flow from physical to digital with clear responsibilities
- Aligns with CPS principles: sensing → computing → actuation → feedback

TECH STACK

- ESP32 -> MQTT -> PI -> FastAPI -> Backend -> FastAPI -> Frontend
- Backend: persists to SQLite, serves API to frontend
- InfluxDB: persists logs in a time-series format
- Grafana reads from Influx; embedded panels in React
- Frontend: Vite/React; mock mode for dev

WHY THIS APPROACH?

- Resilience: Pi keeps MQTT + FastAPI + Forecast AI running locally; data is queued/written when cloud is reachable.
- Forecast: Occupancy/weather model on Pi (local logs) feeds FastAPI for setpoints/automation.
- Scalability: Add more zones/Arduinos by pointing them to the same broker; cloud/backend remains unchanged.



Business/User Layer
Decision-making, management & value extraction
End-user interfaces, reports & optimizations

Application/Cloud Layer
High-level processing & visualization
InfluxDB, Grafana, dashboards, SQLite

Edge/Middleware Layer
Local processing and interoperability
Raspberry Pi, FastAPI logic & data filtering

Network/Communication Layer
Data transmission between devices & systems
MQTT, json, HTTP

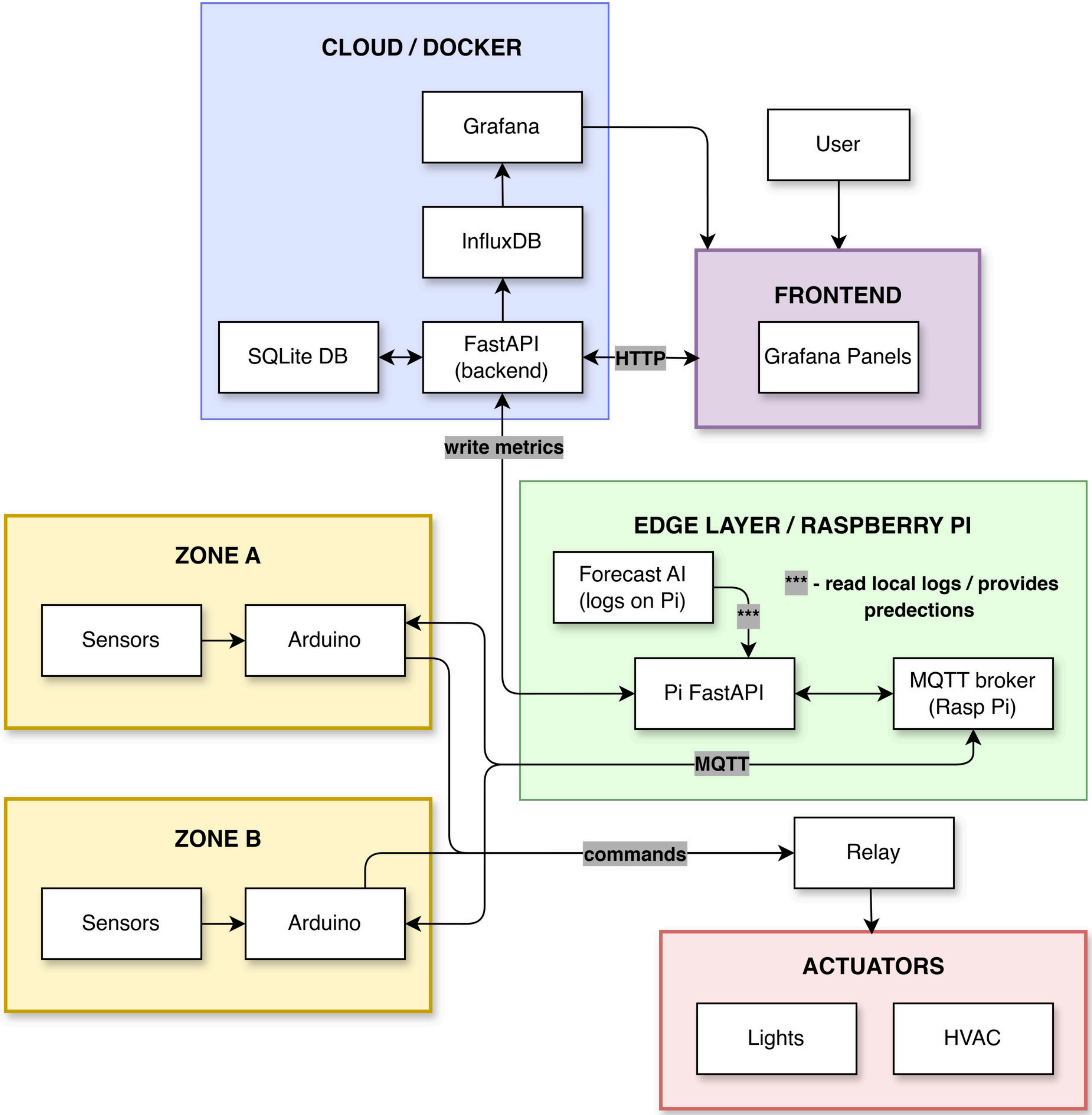
Perception/Physical Layer
Data Acquisition from the environment
Sensors & actuators

ARQUITECTURE

Data path: Sensors/Arduinos → MQTT (Pi) → FastAPI (backend) → InfluxDB/SQLite → Grafana/Frontend; forecasts run on Pi from local logs and feed FastAPI

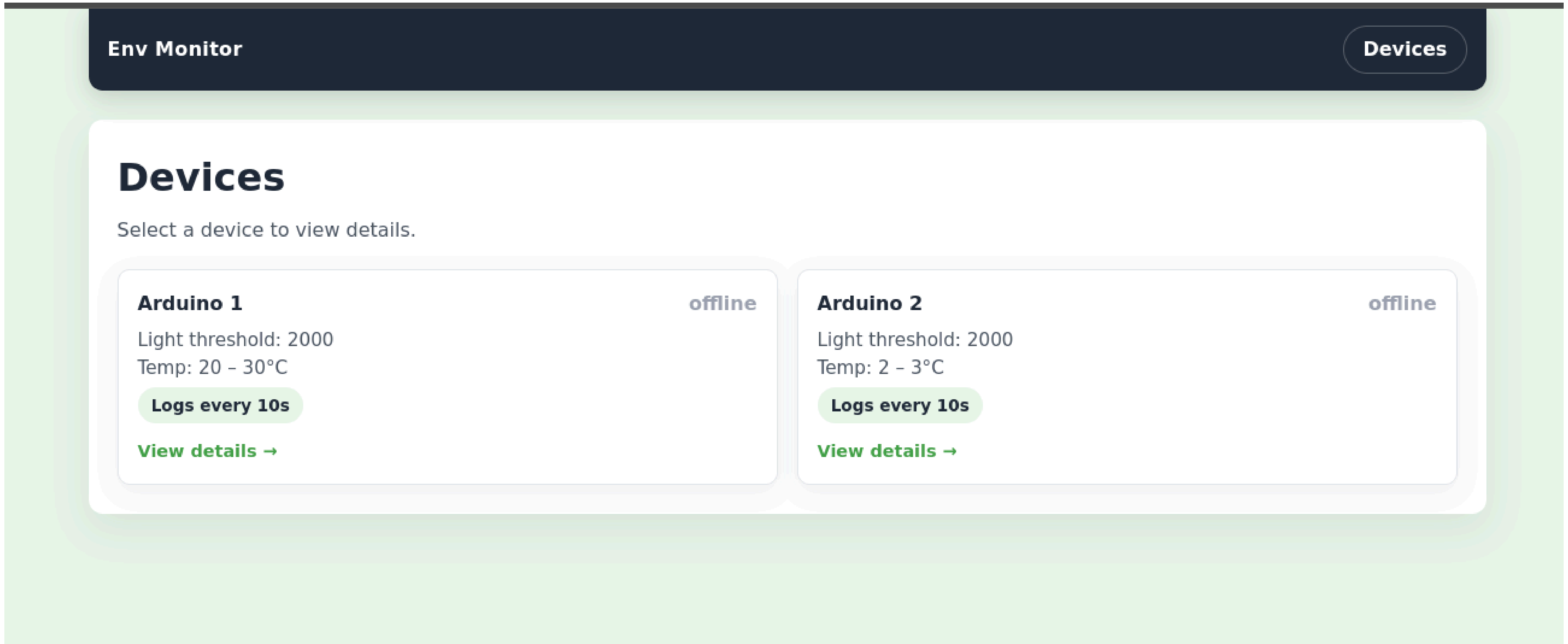
Control path: User → Frontend → FastAPI → Backend → PI → MQTT → Arduinos/relay → Lights/HVAC

Hosting: InfluxDB + Grafana in Docker;
Backend with SQLite + FastAPI;
React frontend embeds Grafana panels;
MQTT+FastAPI+Forecast AI run on the Pi



IMPLEMENTATION (Frontend)

- The realized frontend app enables the user to quickly access information about all of the connected devices.
- The app requests the data from the backend where the last state of every device is stored, alongside their logs



Devices

Select a device to view details.

Arduino 1 offline

Light threshold: 2000

Temp: 20 – 30°C

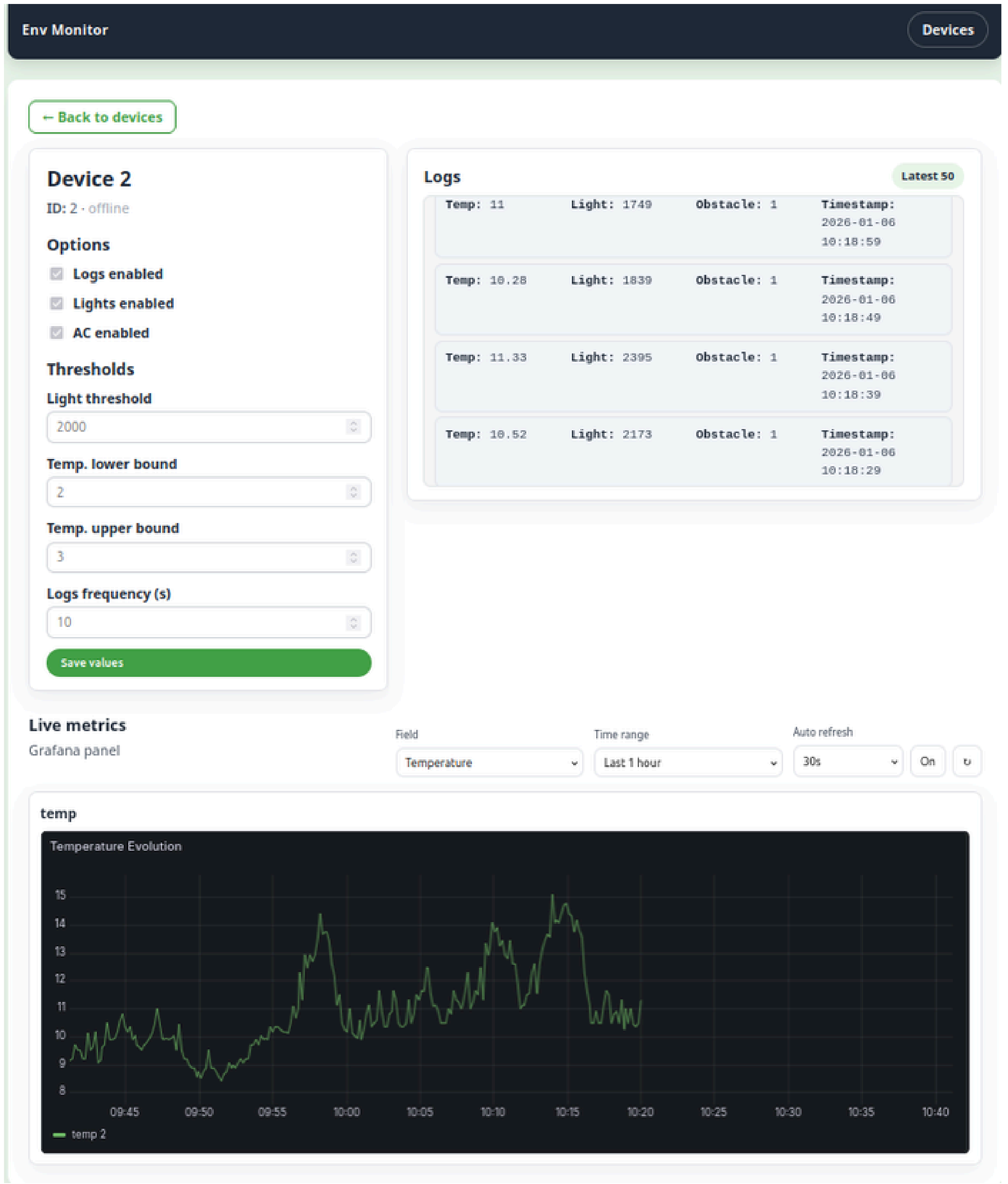
Arduino 2 offline

Light threshold: 2000

Temp: 2 – 3°C

IMPLEMENTATION (Frontend)

- Within the app, the user can view specific information about each device in particular. Including their threshold values, recent logs and multiple graphs with relevant information.
- The values of each arduino can be easily changed through this interface.



Device 2

ID: 2 · offline

Options

☑ Logs enabled

☑ Lights enabled

☑ AC enabled

Thresholds

Light threshold: 2000

Temp. lower bound: 2

Temp. upper bound: 3

Logs frequency (s): 10

Logs — Latest 50

```
Temp: 11      Light: 1749   Obstacle: 1   Timestamp:
                                          2026-01-06
                                          10:18:59
Temp: 10.28   Light: 1839   Obstacle: 1   Timestamp:
                                          2026-01-06
                                          10:18:49
Temp: 11.33   Light: 2395   Obstacle: 1   Timestamp:
                                          2026-01-06
                                          10:18:39
Temp: 10.52   Light: 2173   Obstacle: 1   Timestamp:
                                          2026-01-06
                                          10:18:29
```

Live metrics

Grafana panel

Field: Temperature

Time range: Last 1 hour

Auto refresh: 30s

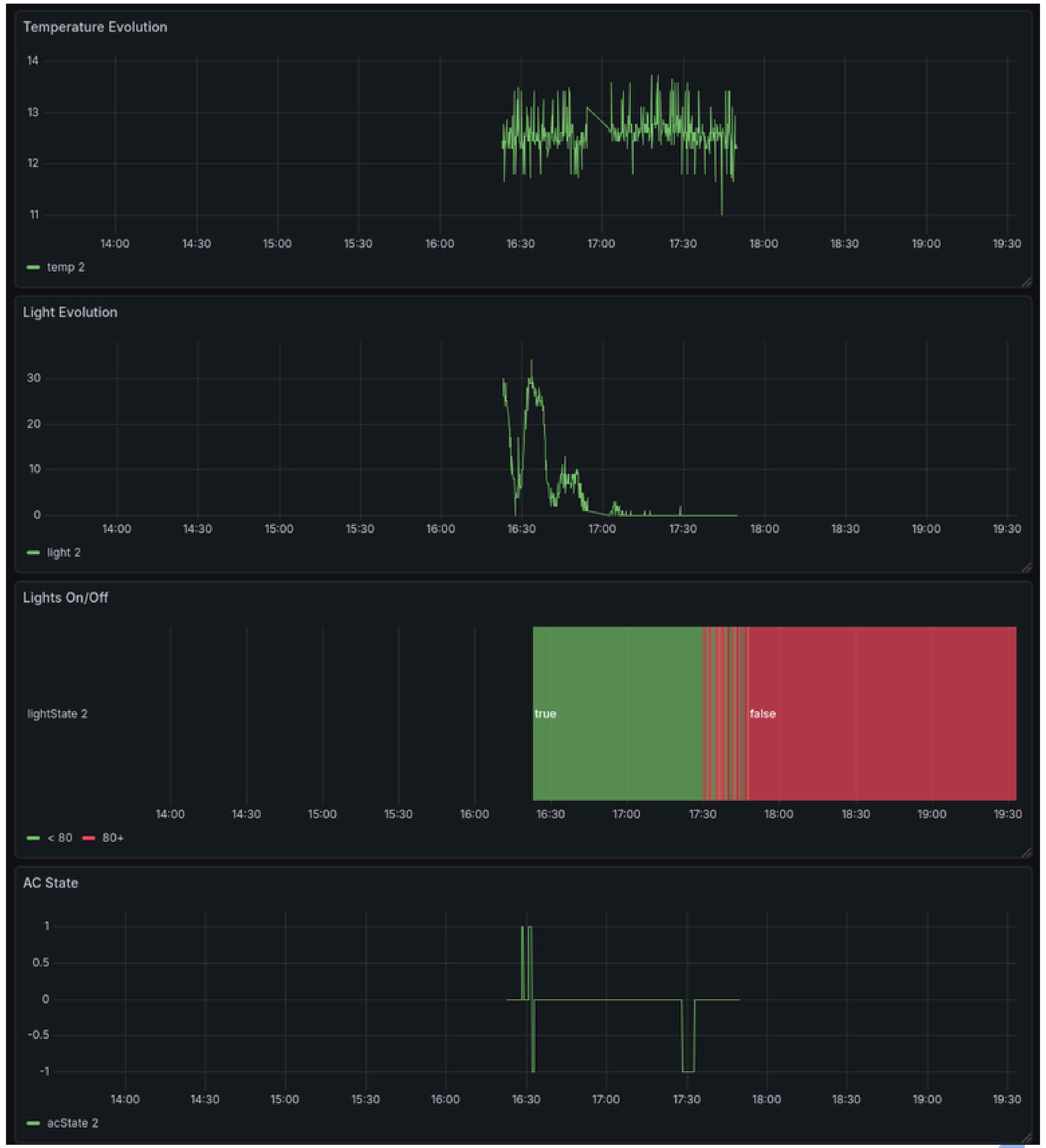
temp

IMPLEMENTATION (Backend)

- FastAPI Python backend that exposes REST endpoints for the web dashboard (frontend) and the Raspberry PI
- Persists devices, commands, and state in SQLite via SQLAlchemy.
- Receives snapshots and logs from the PI, updates device state, and forwards metrics do InfluxDB.
- Queues commands as “pending” and marks them ACKed only after confirmation from the PI (reliability by design).
- Notifies the PI when new commands are available, while keeping an endpoint exposed for the PI periodic polling as a fallback
- Implements a watchdog to mark devices offline if no snapshot is received within a timeout.

IMPLEMENTATION (Grafana/InfluxDB)

- InfluxDB stores time-series metrics sent by the backend (device telemetry, states, logs).
- Backend writes structured points (measurements + tags + fields) to InfluxDB.
- Grafana queries InfluxDB using Flux to visualize device behavior over time.
- Dashboards show real-time and historical trends for each unique device.
- Panels include: temperature, light value, light state, HVAC state



IMPLEMENTATION (Raspberry Pi)

- Raspberry Pi acts as the edge gateway between ESP32 devices and the backend.
- Runs an MQTT broker that ESP32s connect to for telemetry and control.
- Executes a Python manager service that:
    - Subscribes to device MQTT topics (logs, status, config).
    - Publishes commands/configs to devices via MQTT.
    - Assigns id’s to new devices.
- Periodically pulls pending commands from the backend API and ACKs execution back to ensure reliability.
- Forwards snapshots and logs to the backend over HTTP (FastAPI Endpoints).
- Buffers data locally (fallback) so temporary outages don’t lose information.
- Implements an AI Forecasting model to pre-condition zones.

IMPLEMENTATION (ESP32)

- Connects to MQTT broker on the Raspberry Pi over WI-FI.
- Publishes telemetry logs (sensor reading, states, timestamps) at a configurable frequency.
- Subscribes to command/config topics to receive updates (thresholds, enable/disable flags).
- Applies commands locally and immediately, then continues normal operation.
- Persists critical configuration (e.g., device ID) in non-volatile memory.
- Implements robust reconnect logic to handle WI-FI or broker outages without data corruption.

IMPLEMENTATION (Notes)

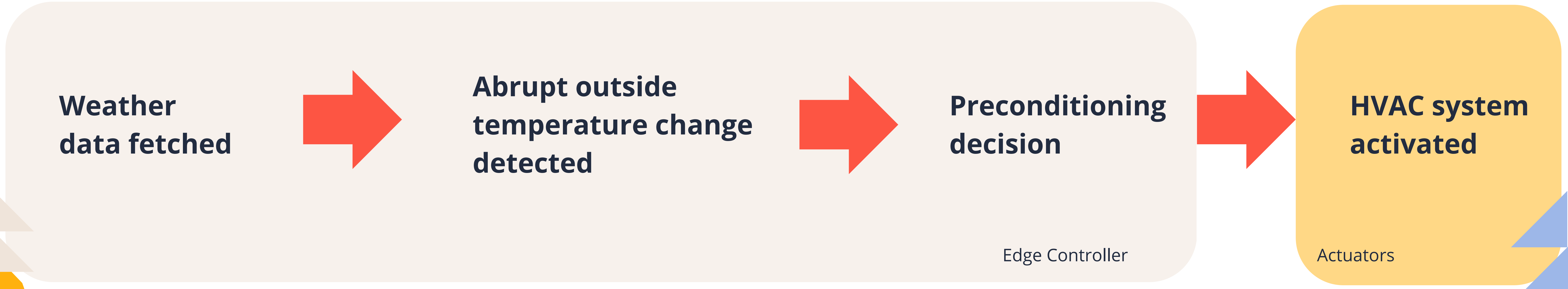
- The system is designed to tolerate partial failures without compromising overall operation. Individual components can fail independently, while the remaining components continue functioning or automatically recover once the failed elements become available again. This approach avoids single points of failure and ensures reliable, self-recovering system behavior.
- ESP32 device onboarding is fully automated. New devices running the system firmware automatically connect to the Raspberry Pi, receive a unique identifier, and are integrated into the existing data flow without manual configuration. This enables a scalable, plug-and-play architecture for adding new devices.

IMPLEMENTATION (WEATHER API)

- We integrated the OpenWeatherMap API to retrieve short term outdoor temperature forecasts

THE GOAL

- Keep the room’s comfort temperature according to the next hours forecast
- Reduce unnecessary HVAC activations



Weather data fetched → **Abrupt outside temperature change detected** → **Preconditioning decision** → **HVAC system activated**

Edge Controller

Actuators

IMPLEMENTATION (OCCUPANCY PREDICTION & PEAK HOURS)

OCCUPANCY PREDICTION

- We used a script in order to generate some synthetic occupancy data in several rooms
- Supervised learning was used in that data
- A logistic regression model was trained to predict the probability of occupancy of a room at a given time
- If the probability of occupancy in a room > 0.6 and the room is not in the comfort temperature threshold then the HVAC system would be activated

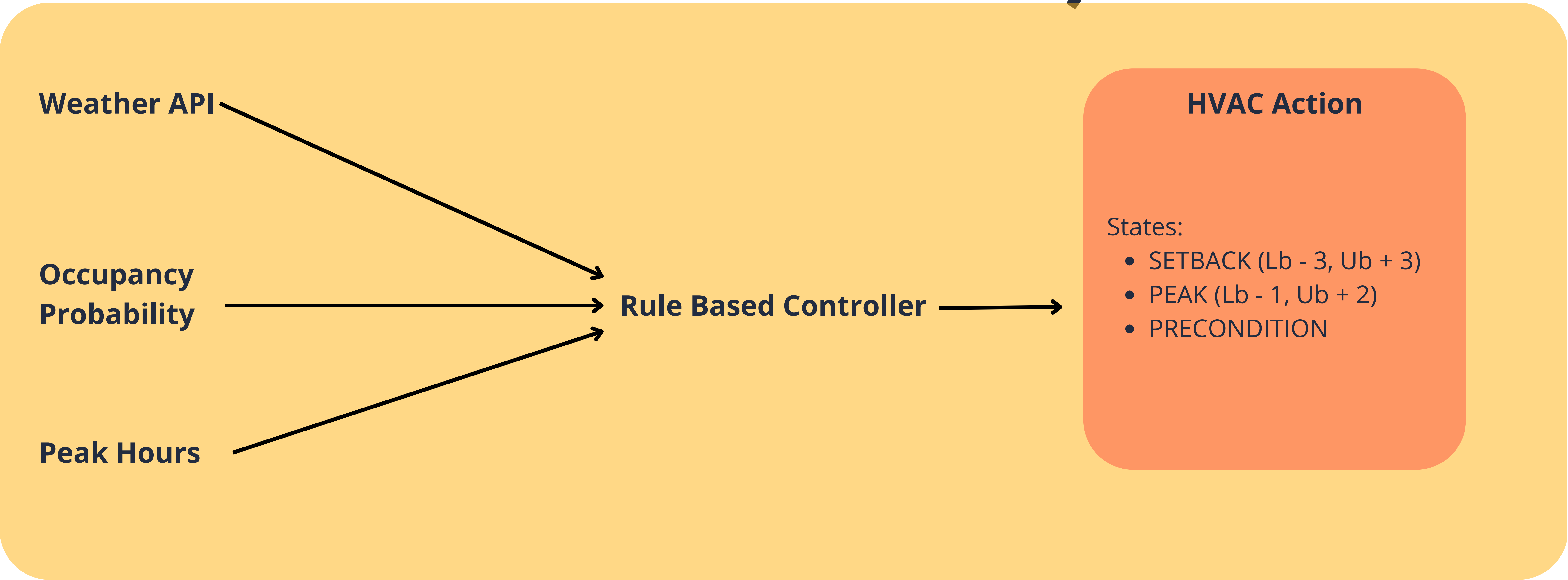
Why Logic Regression?

- Lightweight
- Interpretable
- Edge friendly

PEAK HOURS

- During the time that energy costs the most we also widen the comfort temperature threshold
- Reduces the number of HVAC system activations

IMPLEMENTATION (HYBRID DECISION MAKING LOGIC)



Weather API

Occupancy Probability

Peak Hours

Rule Based Controller

HVAC Action

States:

- SETBACK (Lb - 3, Ub + 3)
- PEAK (Lb - 1, Ub + 2)
- PRECONDITION

EVALUATION SETUP

- Validation focused on system architecture and runtime behavior
- Two scenarios:

Physical deployment

  - 1 Raspberry Pi gateway
  - 2 ESP32 sensor nodes
  - Real telemetry via MQTT

Mixed-data scenario

  - Multiple mocked devices
  - Stress testing of data pipeline

- Tests validated:
  - End-to-end data pipeline
  - Command propagation
  - Dashboard correctness
  - System resilience

OBSERVED RESULTS

- Evaluation Results
  - System successfully maintained data ingestion and visualization across all tests.
- Resilience observations
  - Backend restart recovery ≈ 2 seconds
  - Edge restart recovery in a few seconds
  - Devices automatically reconnect
- Fault tolerance
  - Edge buffers telemetry during backend outages
  - Backend holds unacknowledged commands
  - No data loss between edge ←→ backend
- Scalability validation
  - Tested with up to 10 logical devices
  - MQTT topics handled correctly

IMPACT

- Demonstrates a modular edge-centric architecture for building energy management.
- Key benefits

Energy efficiency

  - Occupancy-aware lights control
  - Weather-based preconditioning

Resilience

  - Local autonomy during backend failures
  - Telemetry buffering and automatic recovery

Scalability

  - Multi-zone deployment through MQTT
  - Plug-and-play onboarding of new devices

Observability

  - Real-time and historical monitoring via Grafana dashboards

- Suitable for small and medium-scale facilities with low deployment overhead.

LIMITATIONS AND FUTURE WORK

Current Limitations

- Predictive models trained with synthetic occupancy data
- No automated sensor calibration or anomaly detection
- Limited validation of energy savings in real environments
- Visualization relies on embedded dashboards, which may introduce authentication constraints

Future Work

- Integrate real occupancy datasets for model training
- Add device-level buffering and backfilling mechanisms
- Implement monitoring, alerting, and anomaly detection
- Improve security and access control
- Support large-scale multi-gateway deployments