Thermal Grace - VHUB
Comfort-as-a-Service IoT System

Software Architecture Design

Artur Kraskov
Semester 7
ICT & OL, Delta
Fontys 2025-2026

Contents

Introduction

Within the scope of VHUB Comfort-as-a-Service system development a prototype was realized. The current document reports the software architecture design.

Background

Software design is made to satisfy requirements and fit in the context of user stories. Previously the project plan and the requirements document were compiled [1-2]. Hardware architecture was designed [3].

Software Architecture

Requirements summary

The system must be able to accumulate data from various wireless sensors. It must collect user feedback. Weather data must be fetched via an API. It must compute PMV/PPD values and query LLM for analysis and advice. New sensors or access points must be easily integrable and removable - the algorithm must still be capable of providing an accurate analysis, sensing and predicting perceived thermal comfort.

Stakeholders and Needs

Stakeholder group A — Researchers

- **Needs:** validity/traceability of measurements, reproducible computations, export/logging, interpretability, modularity, adaptability.
- **Concerns:** data quality, model transparency, auditability.

Stakeholder group B — Office users/visitors/workers

- **Needs:** easy feedback input, understandable comfort guidance, minimal friction.
- **Concerns:** privacy, data use, clarity, trust.

C4 diagram

Level 1 - Context

Shows the Thermal Grace system in context with external actors: sensor nodes, weather API, LLM API, and users.

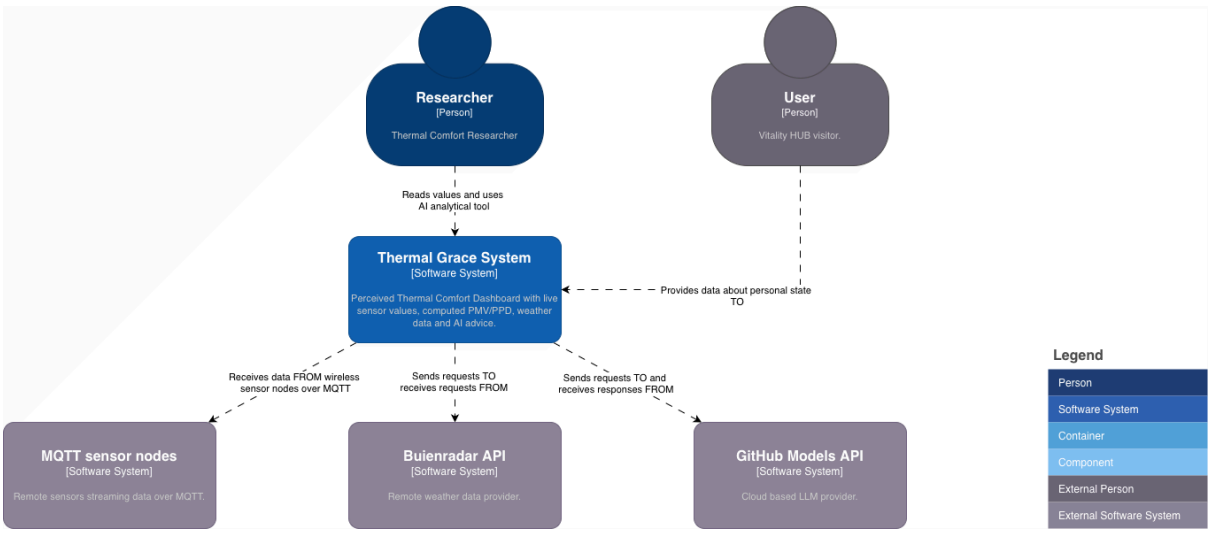

Image 1: C4 diagram level 1 (Context)
High resolution: https://drive.google.com/file/d/1KD9IUZkiTKQc9INoakWdf351lLHdJIYY

Level 2 - Containers

Shows the main containers: MQTT broker, Streamlit dashboard, feedback app, and data storage.

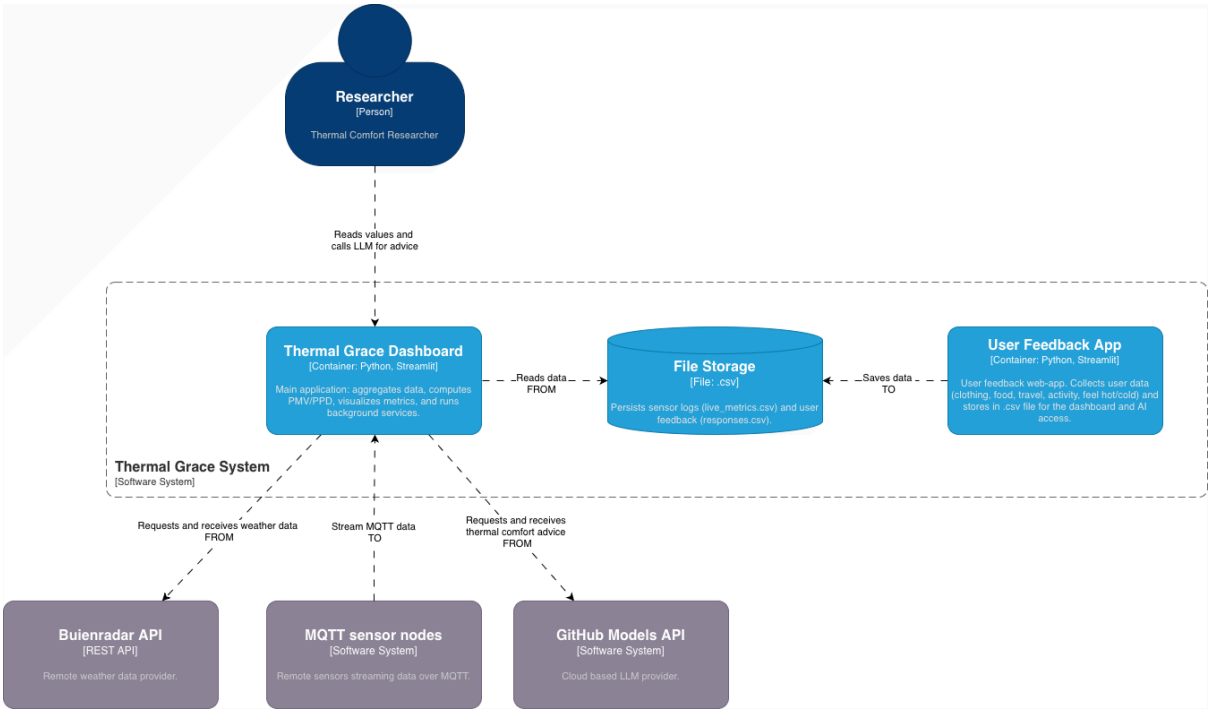

Image 2: C4 diagram level 2 (Containers)
High resolution: https://drive.google.com/file/d/1ZVjtGnIlWeJ-tD-7tcnDyWlDyyZe6cdQ

Level 3 - Components

Shows internal components of the Streamlit dashboard: MQTT receiver, comfort calculator, LLM integration, and UI modules.

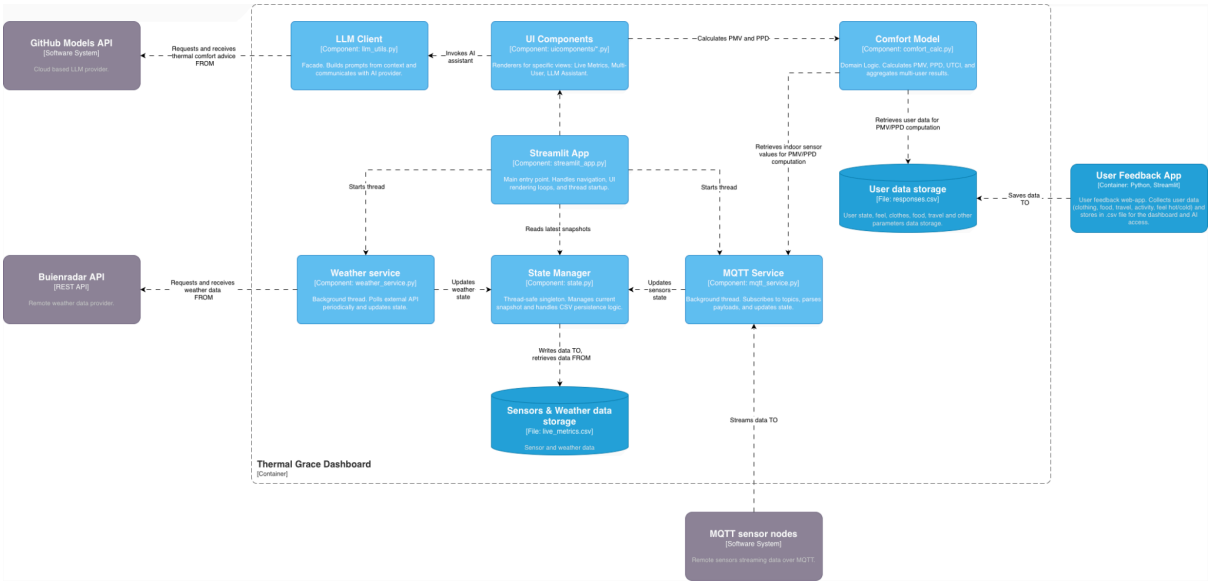

Image 3: C4 diagram level 3 (Components)
High resolution: https://drive.google.com/file/d/1mlP2k4rSQP9jo3I0DwF56UoDrmGmeA2P

Table 1: Key Implementation Details

Feature	**How It Works**
Sensor onboarding	New sensors publish to MQTT topics; dashboard subscribes and parses automatically
Comfort metrics	PyThermalComfort computes PMV/PPD and UTCI from sensor + user inputs
Data logging	All readings saved to .csv files for long-term analytics
Multi-user tracking	Each feedback entry has a UID; system aggregates today's responses for group-level insights

The current implementation meets the requirements by allowing for the expansion of API integrations or data points through modular utility functions. New sensors, such as the RD03D mmWave Radar and MTP40F NDIR CO2 nodes, are easily integrable via the MQTT receiver which handles diverse topics and payloads (up to 3 radar targets). Thermal comfort metrics (PMV/PPD) and UTCI are computed using the `PyThermalComfort` module.

All environmental and occupancy data are logged into `.csv` files for long-term analytics and potential AI model training. A key feature is the Adaptive Multi-User Tracking, which aggregates feedback from multiple occupants today to provide group-level comfort metrics and personalized LLM recommendations. Users provide feedback on their current sensation, activity, and clothing without sharing sensitive personal information; each entry is tracked via a unique identifier (UID).

System Inventory

Existing Systems (External Dependencies)

Table 2: existing systems used in the project

System	Role	Protocol
Mosquitto MQTT Broker	Message transport for sensor data	MQTT
Buienradar API	Dutch weather data provider	REST/HTTP
GitHub Models API	LLM for comfort analysis	REST/HTTP
PyThermalComfort	ISO 7730 PMV/PPD calculation	Python library

Custom-Built Components

Table 3: new and custom systems built for the project

Component	Description
Thermal Grace Dashboard	Streamlit app with live metrics, comfort display, LLM chat
MQTT Ingest Logic	Topic parsing, JSON validation, state snapshot updates
User Feedback App	Streamlit survey form with CSV storage
Sensor Firmware	MicroPython/C++ code for Pico 2 W nodes

Quality Attributes

Security

Threat Model

Table 4: threats, risks and mitigations

Threat	Risk	Mitigation
Spoofed sensor data	Attacker publishes fake MQTT messages	Client ID enforcement; LAN-only broker; JSON validation
API token leakage	Exposed credentials for LLM/weather APIs	Tokens in secrets.toml (git-ignored); never logged

User data exposure	Feedback CSV contains personal context	Minimal PII; file permissions; local storage only
Dashboard hijacking	Unauthorized UI access	LAN-only; optional reverse proxy with auth

Security Controls

Table 5: relevant security controls and implementations

Area	Implementation
MQTT	Broker binds to 192.168.50.x; unique client IDs per sensor; payload validation
Credentials	Stored in .streamlit/secrets.toml; loaded via st.secrets
Rate Limiting	LLM calls gated by "Ask Assistant" button
Logging	Secrets redacted; provenance (topic + timestamp) recorded

Production note: TLS is disabled on the MQTT broker. Enable TLS and authentication for deployments outside trusted networks.

Scalability

Growth Scenarios

Table 6: growth vectors, current state & scaling paths

Growth Vector	**Current State**	**Scaling Path**
More sensors	3 nodes, 3 topics	Add MQTT topics — no code changes needed
Higher frequency	~10 Hz radar, 0.4 Hz CO_2	MQTT handles thousands of msg/sec
More users	Single dashboard	Multiple Streamlit instances + load balancer
Larger datasets	CSV files (KB–MB)	Migrate to InfluxDB/TimescaleDB

Current Optimizations

Table 7: optimizations and benefits

Optimization	**Benefit**
Last-N row fetching	Dashboard reads only recent rows, avoiding full-file scans
LLM response caching	Avoids redundant API calls for same context
UID-based aggregation	Filters unique users before building LLM prompt

Constraints & Assumptions

Table 8: constraints, impact and workaround

Constraint	**Impact**	**Workaround**
Single-host deployment	All services on one Raspberry Pi 5	Sufficient for MVP; scale-out needs DB separation
Internet dependency	Weather + LLM APIs require connectivity	Graceful degradation; comfort calc works offline
Wi-Fi reliability	Data lost if network drops	Sensors auto-reconnect; no local buffering yet
Hardcoded assumptions	Air speed = 0.1 m/s; met/clo fallbacks	Acceptable for office; user feedback overrides

Technology Choices

Table 9: technologies, choices and rationale

Technology	**Choice**	**Why**
Messaging	MQTT (Mosquitto)	Lightweight pub/sub; perfect for IoT; decouples sensors from consumers
Dashboard	Streamlit	Python-native; fast iteration; great for research prototypes
Storage	CSV files	Human-readable; easy export; transparent debugging
Comfort Model	PyThermalComfort	ISO 7730 compliant; open-source; well-documented

LLM	GitHub Models API	Free tier; no GPU needed; simple REST integration
Weather	Buienradar API	Free; Dutch data; simple JSON responses

Trade-offs

Table 10: decisions, benefits and trade-offs

Decision	**Benefit**	**Trade-off**
CSV over database	Simple, portable	Not for high-frequency queries
External LLM	No infrastructure	Requires internet; rate limits
Streamlit over React	Faster development	Less UI flexibility
No MQTT TLS	Simpler setup	Must stay on trusted LAN

Conclusion

The described software architecture design represents the finalized prototype for Vitality HUB's Perceived Thermal Comfort IoT system. The design is modular, reproducible, and utilizes standard frameworks to ensure extensibility. With the integration of the C4 architecture and the implementation of multi-user adaptive logic, the system fulfills the project goals for real-time comfort monitoring and AI-driven recommendations. Future research should leverage the collected historical data to further refine the personalized comfort models.

Table 11: outcomes

Goal	**Achievement**
Real-time monitoring	MQTT streams data; dashboard shows live metrics
Thermal comfort	PMV/PPD computed from sensors + user input
AI recommendations	LLM analyzes conditions and provides advice
Multi-user support	UID-based feedback enables group insights
Modularity	New sensors integrate via MQTT — no core changes
Research-friendly	CSV logging enables data export and analysis

Future Directions

- Personalized models: Train comfort predictions from historical feedback
- Database migration: Move to InfluxDB for time-series performance

- Mobile app: Replace desktop form with mobile-friendly feedback
- Predictive comfort: Forecast conditions based on weather trends

Reference

1. Kraskov A., Kaszuba B., (2025), Vitality HUB Perceived Thermal Comfort | Project Plan, Thermal Comfort-as-a-Service (CaaS) IoT system, FHICT Delta, [Online], Available: https://drive.google.com/file/d/1BXHXlD4_SPatpJrU1mDwF0ZKnESl2W_U
2. Kraskov A., Kaszuba B., (2025), Vitality HUB Perceived Thermal Comfort | Requirements, Thermal Comfort-as-a-Service (CaaS) IoT system, FHICT Delta, [Online], Available: https://drive.google.com/file/d/1NooyuNM97BFz3jYOX2aPSRkbFZ7f5yJF
3. Kraskov A., (2025), Thermal Grace - VHUB Comfort-as-a-Service IoT System, Hardware Architecture Design, FHICT Delta, [Online], Available: https://drive.google.com/file/d/1sWcylxS85DdrUbDKQ6F5CqS8aSHxkFRN