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

Management & Control Report

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

Contents

Contents **1**
Introduction **1**
Background 1
Key Points 2
Implementation **2**
Version Management 2
Release Management 3
Teamwork Support 3
Automated Testing 4
C++ Firmware (PlatformIO) 4
Python/Streamlit Dashboard 4
MicroPython Firmware 5
Conclusion **5**

Introduction

This document describes the implementation of project management and control infrastructure for the Thermal Grace system. It covers version management, release automation, teamwork support, and automated testing for both hardware and software systems. It details the development environment and project management infrastructure. It ensures that all code changes are validated, releases are automated, and contributors have clear guidelines to follow.

Background

Table 0: previous work

Document	**Link**
Project Plan	Vitality HUB Perceived Thermal Comfort | Project Plan.pdf

Requirements	Vitality HUB Perceived Thermal Comfort | Requirements.pdf
Design	1. Software Architecture Design v2.pdf 2. Hardware architecture technical specification.pdf
Realization	Thermal Grace Realization Report.pdf

Key Points

What was implemented.

Table 1: hardware M&C summary

Requirement	**Implementation**
Version management	VERSION file + semantic versioning
Release management	Automated GitHub Releases via Actions
Teamwork support	CONTRIBUTING.md + PR template
Automated testing (HW)	C++ build validation in CI
Automated testing (SW)	Python tests + linting in CI

Table 1.1: software M&C summary

Requirement	**Implementation**
Development environment	requirements.txt + requirements.py
Automated build	GitHub Actions for all firmware types
Automated test infrastructure	pytest, flake8, ruff in CI

Implementation

Version Management

Table 2: version management

Item	**Description**
VERSION file	Tracks current version (0.1.0-alpha) in repo root
Semantic Versioning	Format: MAJOR.MINOR.PATCH[-label]

Why: Provides a single source of truth for the project version, enabling consistent versioning across documentation, releases, and code.

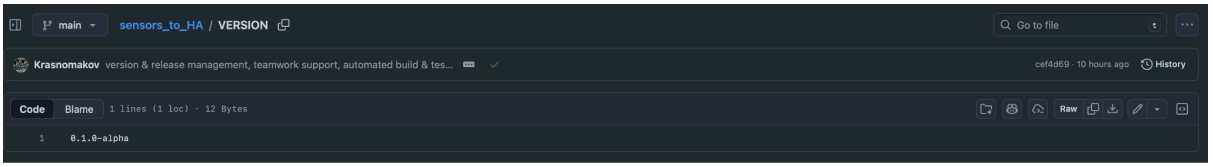

Image 1: version file

Release Management

Table 3: release management

Item	Description
.github/workflows/release.yml	GitHub Action triggered on v* tags
Auto-generated release notes	Uses GitHub's native release note generation
Explicit permissions	contents: write granted for release creation

How it works: When a tag like v1.0.0 is pushed, the workflow automatically creates a GitHub Release with generated release notes from commit history.

Verification: Successfully tested with tag v0.1.0-test — release was created automatically.

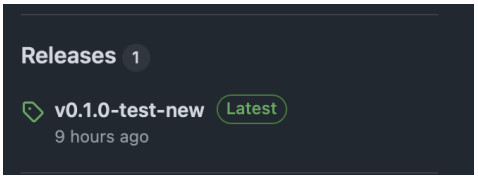

Image 2: release badge in the repo

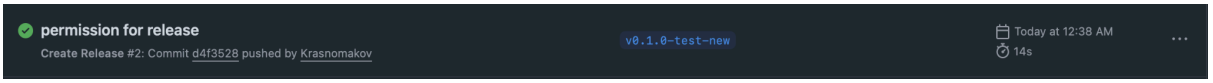

Image 3: successfully completed release workflow

Teamwork Support

Table 4: contributing and pull request template

Item	Description
CONTRIBUTING.md	Guidelines for contributing (forking, branching, testing, PRs)
.github/PULL_REQUEST_TEMPLATE.md	Standard checklist for all pull requests

Why: Ensures consistent contribution practices and facilitates code review by providing structure to pull requests.

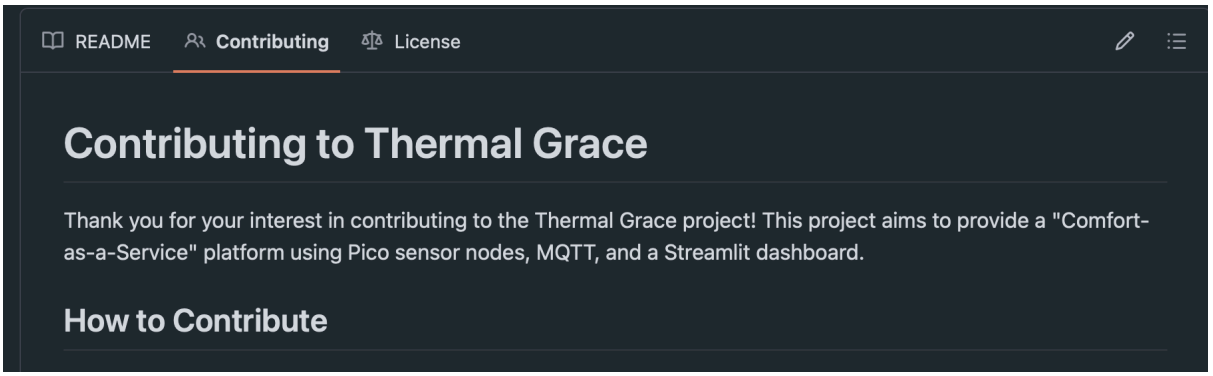

Image 4: Contributing file

Automated Testing

Automated build and test pipelines were created for all major components of the system:

C++ Firmware (PlatformIO)

Table 5: C++ firmware automated test

Aspect	Details
Workflow	.github/workflows/cpp_build.yml
Projects	MTP40-F_CO2_sensor/MTP40-F, MTP40-F_CO2_sensor/MTP40F_MQTT
Validation	Compilation check via pio run

Image 5: successfully completed workflow for C++ firmware building & testing

Python/Streamlit Dashboard

Table 6: Python tests

Aspect	Details
Workflow	.github/workflows/python_tests.yml
Testing	pytest for thermal comfort model and core logic tests
Test Files	test.py, test_multi_user.py, tests/test_logic.py

Linting	flake8 style and syntax checks

Test Coverage
Unit tests verify:

- Comfort calculation (estimate_met, estimate_clo, compute_comfort)
- MQTT payload parsing (parse_env_from_payload)
- State-to-CSV mapping (snapshot_to_row)
- LLM prompt building (build_prompt_from_snapshot, build_multi_user_prompt)

Image 6: successfully completed Python tests

MicroPython Firmware

Table 7: MicroPython firmware tests

Aspect	Details
Workflow	.github/workflows/micropython_lint.yml
Linting	ruff for syntax and import checks
Directories	air_quality_mmWave_mqtt, bme680_air_quality_pi_pico_2_w, mmWave_pico_2_w, AMG_8833_Grid_eye

Fixes Applied
Resolved 24 linting errors including:

- Unused imports removed
- Multiple imports on one line split
- Unnecessary semicolons removed
- Star imports replaced with explicit imports

Image 7: successfully completed MicroPython firmware tests

Conclusion

The recent infrastructure updates to the Thermal Grace repository represent a significant step forward in the project's maturity. By introducing automated testing and standardized versioning, we have shifted from a collection of experimental scripts to a robust, reliable system.

These changes were made to ensure that every component—from the low-level firmware to the data analysis tools—is automatically verified for correctness. For the development of the project, this means:

- **Enhanced Stability:** New features can be added with the assurance that they won't break existing functionality.
- **Transparent Progress:** Automated release tracking provides a clear history of how the system has evolved.
- **Seamless Collaboration:** With clear contribution guidelines and automated safety nets, new developers can join the project and start contributing effectively from day one.

As Thermal Grace continues to evolve, these foundational improvements ensure it remains a trustable and professional tool for monitoring indoor environmental comfort.