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

Hardware Architecture Design

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

Contents

Introduction

Within the scope of VHUB Comfort-as-a-Service (CaaS) IoT system project a MVP prototype was designed and realized. This document covers the design of a distributed computer system including determining microcontrollers, sensors, timing, resource use and performance.

Background

A project plan comprising objectives, problem, literature review, scope and risk analysis was compiled [1]. User-stories were obtained based on literature review and allowed to define the requirements for the system [2].

Hardware Architecture

The following architecture was designed to collect sensor data and process it locally based on the system requirements and project objectives. It wires sensors to microcontrollers and one microcomputer, which stream sensor values over MQTT to the central microcomputer for processing.

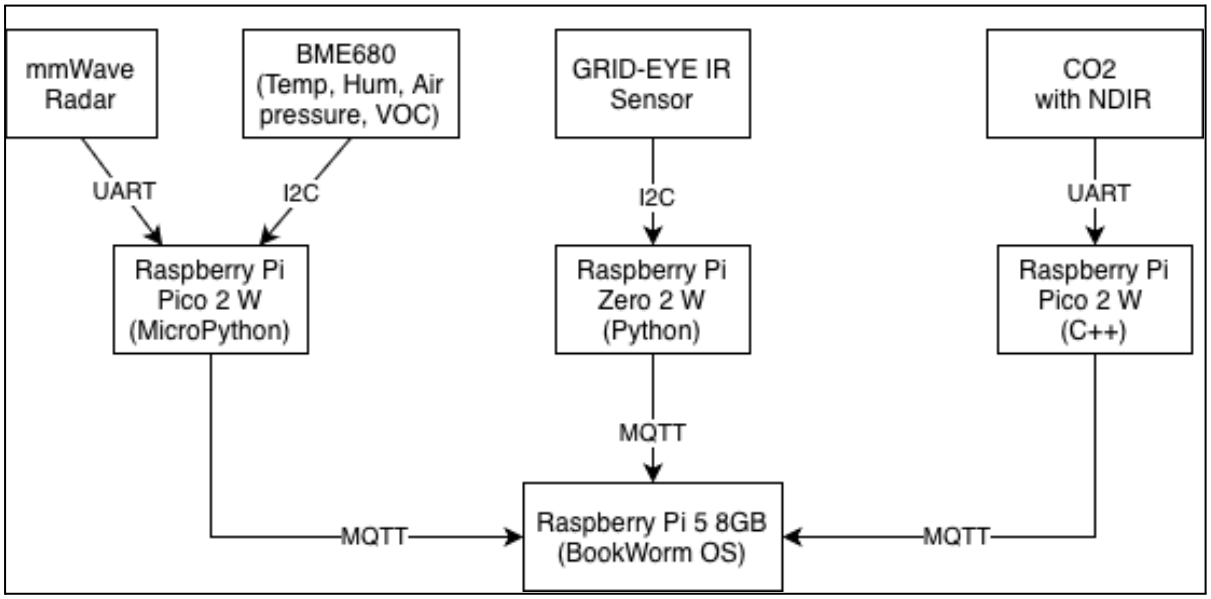

Image 1: hardware architecture diagram

To meet semester time constraints, sensors were split across wireless microcontrollers and microcomputers: one sensor required C++ (others had MicroPython libraries), so two different microcontroller setups were used instead of building a new C++ stack; a Python-only sensor was run on a Raspberry Pi Zero 2 W (Linux + full Python).

Pi Pico 2 W and Pi Zero 2 W (both with Wi-Fi/Bluetooth) communicated with the central computer via MQTT; wireless networking enabled a modular kit where users can attach/detach sensors and relocate them around the office for experiments or personalized setups.

For sensor interfaces, I2C was chosen as the primary hardware bus for its reliability and broad support; UART was used for the sensor that lacked I2C.

Hardware Components

Table 1: hardware components

#	Role	Name	Link	Price
1	Central microcomputer	Raspberry Pi 5 8GB	https://pip-assets.raspberrypi.com/categories/892-raspberry-pi-5/documents/RP-008348-DS-4-raspberry-pi-5-product-brief.pdf?disposition=inline	75€
2	Microcontrollers (2x)	Raspberry Pi Pico 2 w	https://datasheets.raspberrypi.com/pico/pico-2-product-brief.pdf	7€
3	Microcomputer	Raspberry Pi Zero 2 w	https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf	17€
4	mmWave Radar (location & human detection, people counter)	Ai-Thinker Rd-03D 24GHz Radar Sensor Module	https://www.tinytronics.nl/product_files/007073_rd-03d_v1.0.0_specification20240103.pdf	7.5€
5	CO2 Sensor	MemsFrontier MTP40-F CO2 Sensor	https://www.tinytronics.nl/product_files/004660_MTP40-F-CO2-sensor-module-single-channel.pdf	17.25€
6	Temperature, Humidity, Air Pressure, VOC level	BME680 Sensor Module with Level Converter	https://www.tinytronics.nl/en/sensors/air/pressure/bme680-sensor-module-with-level-converter-air-pressure-air-quality-humidity-temperature	10.5€
7	GRID-EYE (Thermal heatmap)	Grove - AMG8833 8x8 Infrared Thermal Temperature Sensor Array	https://www.seeedstudio.com/Grove-Infrared-Temperature-Sensor-Array-AMG8833.html	26€
			Total:	160€

Sensor usage

PMV/PPD Calculation with PyThermalComfort

Sensors are selected to provide the inputs required by the PyThermalComfort library's ISO 7730 PMV/PPD model.

Source: https://pythermalcomfort.readthedocs.io/en/latest/#quick-start

Example

```
# Calculate PMV and PPD using ISO 7730 standard

result = pmv_ppd_iso(
    tdb=25,  # Dry Bulb Temperature in °C
    tr=25,  # Mean Radiant Temperature in °C
    vr=0.1,  # Relative air speed in m/s
    rh=50,  # Relative Humidity in %
    met=1.4,  # Metabolic rate in met
    clo=0.5,  # Clothing insulation in clo
    model="7730-2005"  # Year of the ISO standard
)
```

Implementation

Model: `pythermalcomfort.models.pmv_ppd_iso` with `model="7730-2005"` (ISO PMV/PPD). Returns `pmv` (thermal sensation index) and `ppd` (percent people dissatisfied).

Table 2: PMV/PPD inputs and assumptions

Parameter	Source	Notes
tdb (air temp)	BME680 via MQTT	Live
tr (radiant temp)	Assumed = tdb	Static assumption
rh (humidity)	BME680 via MQTT	Live
vr (air speed)	Hard-coded 0.1 m/s	Still indoor air
met (metabolic rate)	User feedback app	Falls back to 1.2
clo (clothing)	User feedback app	Falls back to 0.7

PMV/PPD computation

PMV balances human heat gains and losses under steady-state conditions. The ISO model solves for skin heat loss via convection, radiation, sweat evaporation, respiration, and diffusion, then maps the heat balance to the PMV scale (-3 cold to +3 hot). PPD is derived from PMV with the ISO exponential relation, meaning even at PMV = 0 at least 5% are dissatisfied.

Sensor Nodes

Table 3: sensors used in the project

Sensor	Purpose
CO_2, VOC, Pressure	Additional context for analysis; fed to LLM alongside PMV/PPD
mmWave Radar	Indoor occupancy tracking (up to 3 targets); 2D coordinate output

AMG8833 GRID-EYE	8×8 thermal heatmap; experimental people/temperature sensing

Air quality + mmWave Node (Pico 2 W)

MicroPython edge node combining radar-based occupancy detection with environmental sensing.

Table 4: air quality & mmWave node data

Aspect	**Details**
MCU	Raspberry Pi Pico 2 W (RP2350 + Wi-Fi)
Firmware	MicroPython
Sensors	RD03D mmWave radar (UART) + BME680 (I²C)
MQTT Topic	sensors/pico/air_mmwave
Update Rate	Up to 10 Hz (when targets detected)

What it measures:

- **Occupancy:** Detects up to 3 people with position (x, y), distance, angle, and speed
- **Environment:** Temperature, humidity, pressure, VOC/gas resistance

CO_2 Node (Pico 2 W)

C++ (PlatformIO) firmware for accurate CO_2 measurement using NDIR technology.

Table 5: CO2 with NDIR node data

Aspect	**Details**
MCU	Raspberry Pi Pico 2 W (RP2350 + Wi-Fi)
Firmware	C++ / Arduino (PlatformIO)
Sensor	MTP40-F NDIR (UART @ 9600 baud)
MQTT Topic	sensors/pico/mtp40f/co2
Update Rate	0.4 Hz (every 2.5 seconds)

Why C++? No MicroPython library exists for the MTP40-F sensor, so this node uses the Arduino ecosystem.

GRID-EYE IR Thermal Map Node (Pi Zero 2 W)

Full Python environment for 8×8 thermal imaging.

Table 6: GRID-EYE node data

Aspect	Details
MCU	Raspberry Pi Zero 2 W (Linux)
Firmware	Python 3
Sensor	AMG8833 GRID-EYE (I²C)
MQTT Topic	sensors/thermal/amg8833
Output	64-pixel (8×8) temperature array

Why Pi Zero? The AMG8833 library requires full Python (not MicroPython), and the thermal array benefits from the extra processing power for potential heatmap analysis.

System Specifications

Air + mmWave Node

Pin Configuration

Table 7: pin configuration for mmWave radar and environmental sensor

Sensor	**Bus**	**Pins**	**Settings**
RD03D Radar	UART0	GP0 (TX), GP1 (RX)	256000 baud
BME680	I²C0	GP5 (SCL), GP4 (SDA)	Default address 0x77

RD03D Radar Protocol

The radar operates in multi-target mode, detecting up to 3 people simultaneously.

Table 8: mmWave radar protocol

Property	**Value**
Frame size	30 bytes
Header	0xAA 0xFF
Tail	0x55 0xCC
Per-target data	x, y (mm), speed (cm/s), distance (mm)

How it works: The firmware waits up to 120 ms for a valid frame, polling the UART every 10 ms. Once a frame arrives, the code computes angle and distance for each target before publishing.

Timing & Performance

Table 9: timing & performance data for mmWave air quality and mmWave radar node

Metric	Value	Explanation
Wi-Fi timeout	~15 s	30 attempts × 0.5 s spacing
Loop cadence	20 ms	Base sleep between iterations
Radar timeout	120 ms	Max wait for valid frame
Publish interval	≥100 ms	Limits rate to 10 Hz max
End-to-end latency	<220 ms	Detection → MQTT publish
Payload size	200–600 bytes	Depends on target count

MQTT Contract

Broker: 192.168.x.x:1883 (no TLS)
Client ID: pico-mmwave-air
Topic: sensors/pico/air_mmwave
QoS: 0 (fire-and-forget)

Payload structure:


```
{
  "timestamp_ms": 123456,
  "radar": {
    "target_count": 2,
    "targets": [
      {"id": 0, "angle": 45, "distance_mm": 1500, "speed_cms": 10, "x_mm": 1060, "y_mm": 1060},
      {"id": 1, "angle": -30, "distance_mm": 2000, "speed_cms": 0, "x_mm": -1000, "y_mm": 1732}
    ]
  },
  "environment": {
    "temperature_c": 23.5,
    "humidity": 45,
    "pressure_hpa": 1013.25,
    "gas_kohm": 50.2
  }
}
```

Error Handling

Table 10: error handling for mmWave and air quality node

Scenario	Behavior
Wi-Fi disconnects	Client cleared, 2 s pause, reconnect on next loop
Sensor read fails	Returns null environment block; radar data still published
MQTT error	Same recovery as Wi-Fi drop

CO_2 Node

Pin Configuration

Table 11: pin configuration for CO2 with NDIR node

Sensor	Bus	Pins	Settings
MTP40-F	UART (Serial1)	GP6 (RX), GP7 (TX)	9600 baud

Timing & Performance

Table 12: timing & performance data for CO2 with NDIR node

Metric	Value	Explanation
Sample interval	2.5 s	CO_2 reading + publish cycle
LED feedback	50 ms strobe	Visual indicator per sample
Wi-Fi timeout	10 s	Connection deadline
Publish rate	~0.4 Hz	One message every 2.5 s
Payload size	<64 bytes	Simple JSON structure

MQTT Contract

Broker: 192.168.50.176:1883 (no TLS)
Client ID: pico2w-mtp40f
Topic: sensors/pico/mtp40f/co2
QoS: 0

Payload structure:

{"co2_ppm": 850}

Configuration Options

These settings can be adjusted in firmware for different deployment scenarios.

Table 13: adjustments and configuration options

Option	Default	How to Adjust
Publish interval	100 ms	Increase PUBLISH_INTERVAL_MS to reduce network load or save power
Quality of Service (QoS) level	0	Use QoS 1 for guaranteed delivery (higher latency)
TLS	Disabled	Enable if deploying outside trusted LAN
BME680 oversampling	Library default	Adjust for accuracy vs. speed tradeoff

Network Deployment Options

Standard setup: All nodes connect to the local Wi-Fi network and publish to an MQTT broker on the Raspberry Pi 5.

Offline mode (no internet required): The Pi 5 can act as a Wi-Fi Access Point:

- Sensor nodes connect directly to the Pi 5's network
- Use Ethernet on Pi 5 for any external connectivity needed
- Ideal for isolated deployments or demonstrations

Security note: TLS and authentication are not configured by default. Deploy only on trusted networks, or enable credentials before production use.

Conclusion

Described hardware architecture meets the requirements for a modular sensor kit at the prototyping stage. Sensors are easily mountable and can be swapped. The connection between different nodes is wireless and doesn't depend on the distance unless wi-fi is there. To avoid the use of the network for sensor nodes, they can connect to Raspberry Pi 5 as an Access Point, but then it is recommended to use Pi 5 without internet or use ethernet. Separate document covers software architecture and implemented perceived thermal comfort sensing and predicting solution.

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