Gooee HVAC — Developer Guide (README_DEV.md)

Overview

Gooee HVAC is a unified engineering platform combining: - **Heat-loss modelling** (U-values, constructions, glazing, infiltration) - **Hydronic network simulation** (pipes, circuits, emitters) - **Balancing & pump sizing** - **Commissioning reports** - **Full project save/load** in a single JSON format - **Integrated GUI** built in PySide6 (Qt)

This document provides developers with: - Project architecture - File structure - Coding conventions - Serialization structure - Extensibility guidance

---

Directory Structure

```
HVAC/
├── core/
│   ├── heatloss/
│   │   ├── heatloss_elements.py
│   │   ├── construction_presets.py
│   │   ├── window_presets.py
│   │   ├── heatloss_json.py
│   │   ├── infiltration_model.py
│   │   └── building_wizard_cli.py
│   │
│   ├── hydronics/
│   │   ├── hydronic_network.py
│   │   ├── emitter_model.py
│   │   ├── friction_models.py
│   │   ├── balancing_engine.py
│   │   ├── pump_sizing.py
│   │   ├── commissioning_report.py
│   │   └── network_serialization.py
│   │
│   ├── project_io.py
│
├── gui/
│   ├── gui_gooee_main.py
│   ├── gui_room_editor.py
│   ├── gui_window_editor.py
│   ├── gui_construction_editor.py
│   ├── gui_uvalue_wizard.py
│   ├── gui_heatloss_wizard.py
```

```
│   ├── gui_hydronics_panel.py
│   ├── gui_balancing_panel.py
│   ├── gui_pump_panel.py
│   ├── gui_pipe_editor.py
│   ├── nodeview_building.py
│   └── nodeview_controller.py
```

Architecture Summary

1. Heat-Loss Subsystem

- **Room elements** are composed of multiple `Construction` layers.
- **Windows** support multi-layer glazing systems.
- **Heat-loss calculation**: U-value × area × ΔT.
- **Infiltration modelling**: ACH-based, includes user-defined settings.
- **Serialization**: handled by `heatloss_json.py`.

2. Hydronic Subsystem

- **Nodes**: connection points in hydraulic topology.
- **Pipes**: length, diameter, roughness, loss coefficient.
- **Emitters**: radiators, UFH loops, etc.
- **Circuits**: ordered sets of pipes + emitters.
- **Balancing**: auto-calculates valve ΔP to equalize circuits.
- **Pump sizing**: determines design head, flow, and power.
- **Commissioning output**: text + CSV reports.
- **Serialization**: full network reconstruction via `network_serialization.py`.

3. Project Save/Load

- Stored as unified JSON with keys:

```
project = {
  "metadata": {},
  "building": { ... },
  "hydronics": { ... }
}
```

- Used by GUI (File → Save / Open).

4. GUI Architecture

- Built using **PySide6 (Qt)**.
- Main window: `gui_gooee_main.py`.
- Uses **QTabWidget** for toolkit integration.

- Editors are embedded widgets with signals returned to the main controller.
- NodeView handles tree selection → editor updates.

Serialization Format Overview

Heat-Loss Structure

```
{
  "levels": [
    {
      "name": "Ground Floor",
      "rooms": [
        {
          "name": "Living Room",
          "elements": [...],
          "windows": [...]
        }
      ]
    }
  ]
}
```

Hydronic Network Structure

```
{
  "nodes": { id: {"label": "..."}, ... },
  "pipes": { id: {...}, ... },
  "emitters": { id: {...}, ... },
  "circuits": { id: {"pipe_ids": [...], "emitter_ids": [...]}}
}
```

Coding Standards

- All modules use **PEP8** (within reason for engineering math).
- Classes use dataclasses where suitable.
- GUI uses clear MVC pattern:
- **Model**: core/heatloss, core/hydronics
- **View**: editors, tables
- **Controller**: `nodeview_controller.py`
- Serialization is always bi-directional:
- `*_to_dict()`
- `*_from_dict()`

Extensibility Guidance

Adding New Heat-Loss Types

- Create new class extending base element.
- Add to `heatloss_elements.py`.
- Extend GUI editor or presets as required.
- Update `heatloss_json.py` to serialize new type.

Adding New Emitters / Hydronic Components

- Define new Emitter class.
- Add to `emitter_model.py`.
- Update `emitter_to_dict()` and `emitter_from_dict()`.
- Add GUI path if needed.

Adding New Reports

- Add function to `commissioning_report.py`.
- Optional: attach to a new GUI tab.

Adding New Project Sections

- Modify `project_io.py` to include new subsystem.
- Update main window load/save handlers.

Developer Workflow

1. Create or open project.
2. Build heat-loss structure.
3. Build hydronic circuits.
4. Balance circuits and choose pump.
5. Generate commissioning report.
6. Save project for later.

Future Roadmap

- DXF export for layouts.
- NodeView2 graphical hydronics diagram.
- Air-side modules (fans, ducts, AHU).
- Renewable systems (heat pumps, solar thermal).
- Full PDF export (reportlab).
- Theme engine (dark/light modes).

- Plugin architecture.

Developer Contact Notes

This document is intended for internal development and contributors. An operator-facing guide should be written separately (README_USER.md).

End of file.