GUIDE

Smart Heat your Home with HEATHER

Andy Symons
Version 2.4
29 July 2024

Contents

1 Introduction

1.1 Purpose

Smart heating – the ability to control the temperature of each room in a house separately, each with its own schedule – increases comfort, saves fuel costs, and by reducing consumption is good for the environment.

1.2 Features

A system built with HEATHER will have the following features:

- Thermostats is each room scheduled by local calendar events with a temperature specification
- Switched devices similarly scheduled by local calendar events
- Manual override for thermostats and switched devices by a change on the device itself, the dashboard, or a voice assistant. The set value is retained for a defined period (default 2 hours)
- For thermostats, optional closure sensors cause heating to turn off if a door or window is left open for a defined period (default 3 minutes)
- For thermostats, optional occupancy sensors cause heating to turn off if a room is left unoccupied for a defined period (default 1 hour)
- For thermostats, a warmup period allows unoccupancy to be ignored for a defined period at the start of a calendar event (default 2 hours)
- Away mode: set all thermostats to a specified temperature (default 5C) and all switched devices off when there is no one at home
- For thermostats, the background temperature specifies per room the temperature to use when there is no calendar event -- default 5C but specified separately from the minimum (frost) and away temperatures
- Raises notifications for thermostats, and switched devices, including and zone control switches that do not respond to a new setting, or go offline
- Optionally logs all transactions with thermostats, switched devices and zone control switches
- Device battery life (where applicable) conserved by only transmitting real changes

1.3 Components

This document is a guide to setting up a smart-heated home using Home Assistant with the HEATHER suite, which comprises three blueprints, a code generator, and a packaged script:

- Blueprints
  - **Heating X2**: Schedule Thermostats with Calendars**.** Blueprint for automations that control the temperature for each room (all the thermostats in a room).
  - **Calendar Switch Y2:** Blueprint for automations that control switched devices, for example an immersion heaters or a hot water cylinder.
  - **Zone Switch Z2:** Blueprint for automations that control zone control valves in systems with more than one heating zone, or boilers that require an external heat demand switch.
- The **Code Generator**[1]
  - A series of WORD (.docx) mail merge files that are used to create YAML files

[1] Formerly 'Heating XYZ Code Generator'

    - An example EXCEL (.xls) mail merge data file
- The packaged script for **Logfile Entry**

Which components are required will depend on the nature of the smart heating system being built.

1.4 History

February 2023 – release 1.0: **Heating X**, the first release, was published on the Home Assistant Community Forum and GitHub.

3 February 2024—release 2.0: **Heating X2** was first released after a year of beta testing in two homes and feedback from users of the first release. It has a new name because it is not compatible with Heating X.

24 February – release 2.1

27 February 2024 – release 2.2

16 March 2024 – release 2.3: **Calendar Switch Y2** and **Zone Switch Z2** were added (but called "2" for alignment with Heating X2) to complete the suite in the same style.

29 July 2024 – release 2.4: the **Logfile Entry** script, already introduced in March, was updated to use the new Home Assistant File integration, and these instructions updated accordingly. The name **HEATHER** was adopted for the project that developed the entire suite of components, which is more than just the three blueprints, the suite itself, and this guide. All former Gist components were consolidated into a the single Gihub repository with the single new name HEATHER.

2 Hardware

A typical (UK) heating system installed in the twentieth century will have a gas boiler, a number of radiators and a hot water cylinder. A 'programmer' like the one in the picture usually enables the user to set one or more on/off times every day or on certain days of the week for hot water and heating.

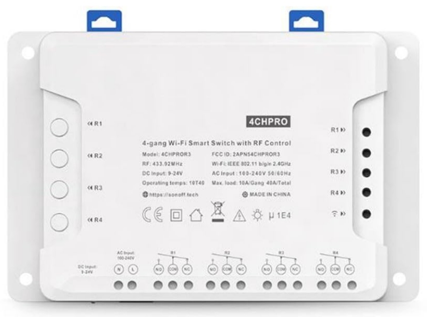
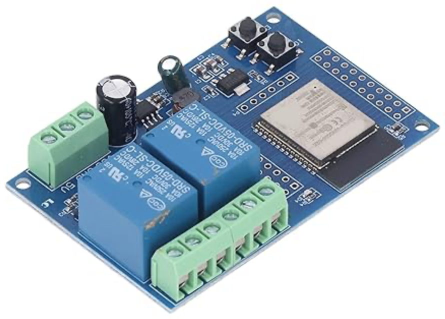
For the heating there will a single wall thermostat. For hot water, there will be a thermostat on the cylinder.

Systems with only hot water and heating have either two zone control valves – called 'S-plan' – or a single 3-position valve – called 'Y-plan'.

Later systems use a combi boiler, which supplies hot water on demand. In this case, there will be just a single valve for the heating. Larger installations may divide the heating into zones, each with its own control valve, thermostat, and programmer.

2.1 Boiler and zone control switches

To connect the heating to a home automation system a smart switch of some kind will be required. There are several available as complete units or boards that can be programmed with ESPHome. Note that if you have a Y-plan plumbing layout, the relay switch has to be two-way, i.e. have a normally closed (NC) terminal for 'heating off' as well as a normally open (NO) terminal for 'heating on'. For S-plan and multi-zone systems, a simple on-off (NO) will suffice.

Electrically, the smart switch takes the place of the old programmer, and the old thermostat circuit should be bypassed. I have sadly not found one that is a plug-compatible physical replacement for the conventional programmer (gap in the market?) so a little DIY wiring will be required. My design is electrically compatible, so that is can be understood by installers, and a future owner can revert to the old system if he/she wishes.

The number of channels needed depends on the number of zones you have – for hot water and heating, two; for multi-zone systems one per zone; or if you have a combi boiler with one heating zone, you just need one switch.

2.2 Thermostatic radiator valves

The other key ingredient will be thermostatic radiator valves (TRV) on all or most of your radiators. Several types are available. It does not matter which one you choose as long as it is compatible with Home Assistant.

Heating X2 is going to provide scheduling, door or window open, and away mode, so you do not need any of these built into the TRV. It will be operating in manual / remote mode rather than (internal) auto mode.

If your radiators already have a mechanical TRV then swapping them should be easy. For older installations, you may need to get a plumber to fit mechanical TRVs first.

You need not necessarily have a TRV on every radiator, but those left without will not be independently controllable from Home Assistant and will only come on at all if another radiator in the zone is on.

Check with your boiler manual whether it is equipped to deal with an all-TRV system: when all the TRVs are shut will it automatically stop pumping and heating? Many will: some require a wiring change to allow the boiler to turn the pump on and off. Others require a by-pass valve or one radiator (e.g. a towel rail) always left on to provide a bypass. If the boiler has a 'heat demand' input, you can connect that to a smart switch and treat it like a zone control valve. Some are already wired such that if any zone control valve is on the boiler comes on.

2.3 Other types of thermostat

When you connect a smart TRV to Home Assistant, it will appear as a 'climate' device. **Heating X2** can control any kind of smart thermostat, so you could connect one for electrical underfloor heating, for example.

You can use Home Assistant to make a Generic Thermostat – an integration that connects a temperature sensor and a switch. You could for example heat a room with an electric heater on a smart plug and a separate temperature sensor. If you want close control of a hot water cylinder you could use a temperature sensor (with a probe on the cylinder) and form a generic thermostat directly with the hot water zone valve and/or an electric immersion heater. If you want to control an entire zone with one thermostat instead of buying lots of TRVs, you could similarly form a generic thermostat between a room temperature sensor and the zone control valve. All of these can be controlled by **Heating X2**.

2.4 Switched heating devices

Some heating devices have a built-in thermostat not accessible to Home Assistant, so only need turning on and off. Examples are a hot water cylinder (heated by hot water from the boiler controlled by a zone valve), an electric immersion heater connected to a smart switch, or underfloor heating with only a smart switch and a manual thermostat rather than a smart thermostat.

2.5 Door and window closure sensors

If you have one or more closure sensors in a room, on doors or windows, you can tell **Heating X2** about them. **Heating X2** will then turn the heating to the minimum (frost setting, default 5C) if a door or window is let open for more than a set period of time (default 3 minutes).

2.6 Occupancy sensors

If you have one or more occupancy sensors in a room, such as motion detectors or mmWave human presence detectors, you can tell **Heating X2** about them. **Heating X2** will then turn the heating to the background setting (default 5C but separate from the frost setting) if the room is left unoccupied for more than a set period of time (default 1 hour).

3 Software Installation

3.1 The Logfile Entry script

All three XYZ blueprints require a script that facilitates the formatting and writing of the log file(s). (More on logging below). It is mandatory, even if you do not want logging, because the test for logging being required is included in the script to declutter the blueprints.

Log File Entry is not a blueprint, just a script. The recommended installation method is as a package:

1. If you do not already have Packages enabled, do so as described in the Home Assistant documentation
2. In */config/packages/* add the *file logfile_entry.yaml*
3. Copy the Logfile Entry code from GitHub
4. Paste the code into */config/packages/logfile_entry.yaml*
5. Reload the configuration

The same script is used by **Heating X2**, **Calendar Switch Y2**, and **Zone Switch Z2**, so you only need do this once.

3.2 The Blueprints

There are two alternative ways to install blueprints

1. Automatically
2. Manually

3.2.1 Automatic install via Home Assistant

- To open your Home Assistant instance and show the blueprint import dialog with the **Heating X2** blueprint pre-filled, click each of the links below in turn and follow the Home Assistant instructions
    - Heating X2
    - Calendar Switch Y2
    - Zone Switch Z2

3.2.2 Manual install

A drawback of the automatic method is that all comments are stripped out. I find the header comment useful to check which version I have; other comment is necessary if changes are required or for debugging. The manual method preserves all the original text and is not particularly difficult!

Using your preferred Home Assistant file editor:

1. Create an empty folder */config/blueprints/automation/AndySymons*
2. Add three empty files to this folder
    a. *heating_x2.yaml*
    b. *calendar_switch_y2.yaml*
    c. *zone_switch_z2.yaml*
3. Go to each of the following Github pages in turn. For each, copy the text of the file and paste it into the corresponding file on your system that you just created
    a. Heating X2
    b. Calendar Switch Y2
    c. Zone Switch Z2

In case any of these links is broken you can find the files by searching for AndySymons in Github then finding the HEATHER repository.

3.3 The Code Generator

Each instance of **Heating X2**, **Calendar Switch Y2**, and **Zone Switch Z2** requires a number of helpers – input texts, input booleans, input numbers and timers. You may also need templates, binary sensor groups, log notification services, the automations themselves and a dashboard.

You *can* define these by hand, but to help set up a large installation, the Code Generator creates all the YAML code you need from a single EXCEL spreadsheet that simply lists your zones, rooms and thermostats.

The Code Generator uses Microsoft Mail Merge, so you will need a system running Microsoft Office (sorry, I could not find a free app for this).

Full details on downloading and using the Code Generator are given in chapter 7 ‘Creating your Heating System Configuration with the Code Generator’ on page 16.

4 Calendars

4.1 Create calendars

For each automation (room or individual device such as water heaters), create a calendar using the Home Assistant Local Calendar integration in the UI (you need Home Assistant 2022.12.1 or later). At the time of writing there is no way to create calendars automatically with YAML.

4.2 Add Events

Add events to the calendars for each period that you want to specify a temperature that is different from the background temperature (see below) or periods when a switched device is turned on.

4.2.1 Summary (title) field

In the summary field put the name you want to appear on the dashboard – e.g., “<room> Daytime setting”, “<room> Night setting”, “<room> Evening boost”. Including the room name is recommended for troubleshooting – it will be easy to see if you put an event in the wrong calendar.

4.2.2 Description field (temperature specification)

For switched devices the description is not used by the system, but you can add comments for your own reference if you wish.

For thermostats, in the description field, *add the required temperature* anywhere between two hashes: e.g., “Set temperature to #21.5# C”. Do not use another hash before the one that introduces the temperature or write anything else between the hashes. The automation will check that there is a valid temperature and report any errors in the reason text. You can add comments for your own reference if you wish outside the hashes.

4.2.3 Event start and end times

Set the event start and end times as usual. Consecutive events are allowed, and the automation will transition directly from one to the other.

Overlapping events are also allowed. If a second event overlaps the first – i.e. starts before the first one ends -- it will take precedence over the first one until it finishes; then the first one becomes current again if it has not finished in the meantime. Any number of overlapping and consecutive events are allowed. In all cases the event that started last will be current. If two or more start at the same time, then one of them is selected at random.

4.2.4 Event repeats

Set the events to repeat daily or on specific days of the week as required.

5 Logging

Since release 2.3, logging of key steps in all automations built with the **Heating X2**, **Calendar Switch Y2**, or **Zone Switch Z2** is available but optional. If you do not require logging, ignore this chapter.

The log file content looks like this (for a room with two TRVs):

```
Heating X2 notifications (Log started: 2024-02-27T13:42:10.558298+00:00)
--------------------------------------------------------------------------------
2024-02-27T13:42:10.558335+00:00 ---
2024-02-27T13:42:10.564862+00:00 [Parlour heating control] Triggered by: set_temperature_change
2024-02-27T13:42:15.695047+00:00 [Parlour heating control] Thermostat ‘Parlour north TRV’ set to 20.0
2024-02-27T13:42:15.703642+00:00 [Parlour heating control] Thermostat ‘Parlour south TRV’ is already set to 20.0
2024-02-27T13:42:15.708166+00:00 [Parlour heating control] New state: Set manually to 20.0.
2024-02-27T13:42:20.695623+00:00 ---
2024-02-27T13:42:20.700644+00:00 [Parlour heating control] Triggered by: set_temperature_change
2024-02-27T13:42:20.742891+00:00 [Parlour heating control] Set temperature ignored during echo block period
2024-02-27T13:42:45.008107+00:00 ---
2024-02-27T13:42:45.010569+00:00 [Parlour heating control] Triggered by: 11choblock_timer_end
2024-02-27T13:42:45.051430+00:00 [Parlour heating control] The thermostat ‘Parlour north TRV’ is set correctly.
2024-02-27T13:42:45.054353+00:00 [Parlour heating control] The thermostat ‘Parlour south TRV’ is set correctly.
2024-02-27T13:42:52.420332+00:00 ---
2024-02-27T13:42:52.425302+00:00 [Parlour heating control] Triggered by: manual_override_end
2024-02-27T13:43:28.606181+00:00 [Parlour heating control] Thermostat ‘Parlour north TRV’ set to 21.0
2024-02-27T13:43:32.987376+00:00 [Parlour heating control] Thermostat ‘Parlour south TRV’ set to 21.0
2024-02-27T13:43:32.995748+00:00 [Parlour heating control] New state: Set to 21.0 by calendar event ‘Parlour Daytime’ until
Tue 27 Feb 2024 at 18:30.
2024-02-27T13:43:33.610287+00:00 ---
2024-02-27T13:43:33.616708+00:00 [Parlour heating control] Triggered by: set_temperature_change
2024-02-27T13:43:33.687466+00:00 [Parlour heating control] Set temperature ignored during echo block period
2024-02-27T13:43:38.188684+00:00 ---
2024-02-27T13:43:38.193401+00:00 [Parlour heating control] Triggered by: set_temperature_change
2024-02-27T13:43:38.234533+00:00 [Parlour heating control] Set temperature ignored during echo block period
2024-02-27T13:44:02.017457+00:00 ---
2024-02-27T13:44:02.021264+00:00 [Parlour heating control] Triggered by: 11choblock_timer_end
2024-02-27T13:44:02.357930+00:00 [Parlour heating control] The thermostat ‘Parlour north TRV’ is set correctly.
2024-02-27T13:44:02.360491+00:00 [Parlour heating control] The thermostat ‘Parlour south TRV’ is set correctly.
```

There is a group of lines for each trigger, preceded by a blank line for readability.

To activate logging, you need to

a. Set up one or more log file notification services (one for each log file)
b. Pass the name of the service to the automation blueprint(s)

You can have one file for all rooms or one per room or any combination. The logfile entries always identify which automation is writing the entry – ‘[Parlour heating control]’ in the sample – so there is no ambiguity if they are all in one file, but you may wish to split them up for ease of management. I use one per room because it is then easy to follow the course of action of a single automation.

5.1 Defining the log notification services

Sadly, Home Assistant no longer supports the creation of notification services in YAML, so this is not available in the Code Generator.

1) First install the File integration if you do not already have it.
2) Decide where you want to put your log files. I use /config/log_files. You should create the folder but do not need to create the individual files; they will be created automatically the first time the service is called.
3) In configuration.yaml insert
4) To create one notification service:
   a) Open the File Integration in the Home Assistant UI (via ‘Settings’ → ‘Integrations’ if it is not already open after installation).
   b) Click ADD ENTRY
   c) Choose ‘set up a notification service’
   d) Specify the file path starting with */config/* and any subfolder you want to use for your log files, then the name of the file. I name the file for the room and use a *.log* extension

because that means the console app will be opened if you open a downloaded copy of the file. You can use .txt (or any other extension) as you prefer. So and example in my case is */config/log_files/butlers_room.log*

e) Leave the timestamp option turned off because the Logfile Entry script adds one in a more readable format. Click SUBMIT.

f) The notification service is created with a cryptic name like '*billiard_room_heating_log [/config/log_files/billiard_room_heating.log]*'. You can change this to a friendly name by clicking the menu list to the right of the entry and selecting rename. In this case '*Billiard room heating log*' might be a more meaningful name.

g) The notification service is created with an entity id like '*notify.file*', (*~file1* etc). To change this you have to go to Settings → Entities and find the entry you just added. Then change the entity name to e.g. '*notify.billiard_room_heating_log*'.

Now repeat for each service required – i.e. one per logfile.

After making these changes do a full system restart. A quick reload is not sufficient in this case.

You can test your notification services using Developer Tools → Services.

5.2 Identify the log notification service to the blueprint

The last input to the blueprint is the *name* of the logfile notification service (not the file itself), i.e. the value of the entity_id in the configuration without the '*notify*' prefix – in the above example it is '*billiard_room_heating_log*'.

Unfortunately, Home Assistant does not at present provide for a selector of services, so this is a plain text field. To avoid errors, I recommend you copy and paste the name from the entity, taking off the '*notify.*' prefix.

6 Creating your Heating System Configuration by Hand

6.1 Creating Helpers by Hand

The blueprints require a number of helpers, which are used as global variables. You can create these by hand, but the simplest method for a large installation is to use the Code Generator (below).

6.1.1 Input_text helpers

- Setting reason (X, Y, and Z) – into which the automation writes the reason for the current setting for display on dashboards. Make the length 255 characters because the default 100 is too short.

6.1.2 Input_number helpers

- Required temperature (X only) – to hold the temperature currently set for the room
- Manual temperature (X only) – to hold the temperature specified by manual override

6.1.3 Input_boolean helpers

- Required state (Y and Z) – to hold the state (on or off) currently set for the room
- Manual state (Y only) – to hold the state (on or off) specified by manual override

6.1.4 Timers

- Echoblock timer (X, Y, and Z) – for use inside the automation to time the response from a thermostat before checking, and to distinguish genuine manual changes of the set temperature from those made by the automation.
- Manual override timer (X and Y) – to hold the timer for a manual intervention
- Warmup timer (X only) – to hold the timer for the event warmup period
- Door or window open timer (X only) – to hold the time until the heating will be turned off following a door or window opening (it is paused when they are all closed)
- Room unoccupancy timer (X only) – to count down the time until the heating will be turned off following the room becoming unoccupied (it is paused when the room is occupied)

6.2 Creating Automations by hand using the Blueprints

Automations can be created by the code generator or by hand. After using the Code Generator, they can still be edited in the UI afterwards. For instructions see below.

6.2.1 Using the Heating X2 blueprint by hand

To create a system by hand you need one **Heating X2** based automation per room. Select Blueprint **Heating X2** and specify all the inputs described above – the ‘climate’ devices (thermostats), calendar, and helpers. They all have to be defined before the blueprint can be invoked to make an automation! You can also adjust the parameters (fixed values for this room), but they all have defaults so you might want to skip this step at first and come back later when tuning the automations.

6.2.1.1 Parameters

- Minimum thermostat temperature – the minimum temperature that your thermostat can be set to, a.k.a. frost setting. Default 5C.

- Maximum thermostat temperature – the maximum temperature that your thermostat can be set to. Default 30C.
- Manual override period – the time period for which a manual intervention will override the schedule. Default 2 hours.
- Warmup period – the period of time from the start of a new event for which room unoccupancy will be ignored. Default 2 hours.
- Door or window open delay time – the time for which a door or window can be open before the heating switches off. Default 3 minutes.
- Room unoccupancy delay time – the time for which no presence in the room is detected before it is consider unoccupied. Default 1 hour.

6.2.2 Using the Calendar Switch Y2 blueprint by hand

For devices that are just switched use the **Calendar Switch Y2** blueprint. You can group devices to control from one calendar if you wish, but there will usually just be one. Select Blueprint **Calendar Switch Y2** and specify all the inputs described above – the switched device(s), calendar, and helpers. Again, they all have to be defined before the blueprint can be invoked to make an automation. You can also adjust the parameter (fixed value for this room), but it has a default so you might want to skip this step at first and come back later when tuning the automations.

6.2.2.1 Parameter

- Manual override period – the time period for which a manual intervention will override the schedule. Default 2 hours.

6.2.3 Using the Zone Switch Z2 blueprint by hand

For zone switching you need one **Zone Switch Z2** automation for each zone.

The inputs are the zone switch, a heat demand sensor, and the helpers. Again, they all have to be defined before the blueprint can be invoked to make an automation.

Heat demand operates on two levels. First create a heat demand binary sensor for each thermostat, which is on when heat is required, then create a binary sensor group for each zone that contains all the thermostats in the zone.

6.2.3.1 Heat demand sensor

To create by hand, use the following example for each thermostat.

```
### Guest bedroom trv set temperature
- sensor:
  - name: Guest bedroom trv set temperature
    unique_id: guest_bedroom_trv_set_temperature
    unit_of_measurement: 'C'
    icon: mdi:thermometer
    state: "{{ state_attr('climate.guest_bedroom_trv', 'temperature') }}"

### Guest bedroom trv measured temperature
- sensor:
  - name: Guest bedroom trv measured temperature
    unique_id: guest_bedroom_trv_smeasured_temperature
    unit_of_measurement: 'C'
    icon: mdi:thermometer
    state: "{{ state_attr('climate.guest_bedroom_trv', 'current_temperature') }}"

### Guest bedroom trv heat demand
- binary_sensor:
  - name: Guest bedroom trv heat demand
    unique_id: guest_bedroom_trv_heat_demand
    icon: mdi:heat-wave
    state: "{{ float(state_attr('climate.guest_bedroom_trv', 'temperature'),0) >
     float(state_attr('climate.guest_bedroom_trv', 'current_temperature'),99) }}" # false if either
     sensor is unavailable
```

6.2.3.2 Heat demand group

This can be created as a binary sensor group in the UI, or using YAML like the following example

```
binary sensor:
### Zone 3 First Floor Heat Demand Group
  - platform: group
    name: "Zone 3 First Floor Heat Demand Group"
    unique_id: zone_3_first_floor_heat_demand_group
    entities:
      - binary_sensor.dressing_room_trv_heat_demand
      - binary_sensor.guest_bedroom_trv_heat_demand
      - binary_sensor.landing_trv_heat_demand
      - binary_sensor.main_bathroom_trv_heat_demand
      - binary_sensor.master_bedroom_west_heat_demand
      - binary_sensor.master_bedroom_east_trv_heat_demand
      - binary_sensor.second_bedroom_north_heat_demand
      - binary_sensor.second_bedroom_south_trv_heat_demand
```

7 Creating your Heating System Configuration with the Code Generator

As seen above, each instance of **Heating X2**, **Calendar Switch Y2**, and **Zone Switch Z2** requires a number of helpers – input texts, input booleans, input numbers and timers. You may also need templates, binary sensor groups, the automations themselves, and dashboard cards.

You *can* define these by hand, but to help set up a large installation, the Code Generator can creates most of the YAML code you need from a single EXCEL spreadsheet that simply lists your zones, rooms and thermostats.

The Code Generator uses Microsoft Mail Merge, so you will need a system running Microsoft Office (sorry, I could not find a free app for this).

If you use the code generator, you should not create the same helpers, timers, or template sensors by hand as that would create duplicates. For automations and the dashboard choose one method – manual or automatic – but not both!

7.1 Download the Code Generator files

To get the Code Generator:

1. Download the Github Code Generator folder. This link takes you directly to a Github folder download tool with the link to the folder. If the link does not work find the folder in the AndySymons/HEATHER repository and copy its link into the directory download tool. Other download options are available.
2. Save the zip file in a temporary location (e.g. your 'Downloads' folder).
3. Unzip it onto a *local* drive. Note that Mail Merge can have difficulties with some cloud drive systems such as iCloud and OneDrive.

Most of these files are temporary; you only need them long enough to set up your system; once you have generated the YAML code you can delete them. the only file specific to your system is the EXCEL DATA file. You might want to archive that afterwards just in case you need to re-install the whole system.

After unzipping, you will have a folder with an example EXCEL workbook 'DATA' for your zone, room and thermostat data, and a number of WORD mail merge files corresponding to the YAML files to be created. Instructions on use follow below.

7.2 Prepare your system

The code generator is going to generate code to paste into separate YAML files for each type of code, so you need to have these files available if they are not already. Separating the files is good practice for larger systems anyway, where putting everything in the *configuration.yaml* file would be unwieldy.

1) Make a full backup
2) Use your preferred Home Assistant code editor (such as File Editor or Studio Code Server) to create the following empty files in the /CONFIG folder, unless you already have them:
   - *binary_sensors.yaml*
   - *input_booleans.yaml*
   - *input_numbers.yaml*
   - *input_texts.yaml*
   - *templates.yaml*
   - *timers.yaml*

3) If you already have some input datetimes, input numbers, input texts, templates, or timers defined in configuration.yaml, move them to the new files you just created; otherwise you will get duplicate keys.
4) In configuration.yaml enter the following text without indentation, unless you already have any of it:
```
binary_sensor: !include binary_sensors.yaml
input_boolean: !include input_booleans.yaml
input_number: !include input_numbers.yaml
input_text: !include input_texts.yaml
template: !include templates.yaml
timer: !include timers.yaml
```
5) In developer tools, check the configuration and reload it.

7.3 Enter your zone, room, and thermostat information

In your code generator folder downloaded earlier, copy the first file *Heating XYX code generator DATA - example.xlsx* to *Code Generator DATA – <your home name>.xlsx*. Then open it in EXCEL. This is going to be the single data source for all the later mail merges, specifying the zones, rooms, and thermostats for your whole heating installation. You should delete the example data in the yellow cells and replace it with your own real data. Add or remove lines as required, but do not touch the grey cells with the formulae in them.

There is a line for each Thermostat, grouped by zone and room:

A. In column A (Zone) enter names for the zones in your system, if any. Thermostats controlling radiators or other devices in the same zone should have identical zone names. If you have no zones, leave the column blank. For thermostats or switched devices that are not in a zone, leave the column blank. Note that if you are using a zone valve as part of a generic thermostat, you should not include it in a zone as well.

B. In column B (Room) enter the room name with normal capitalisation and spacing, e.g. Dining Room, Kitchen, or Annie's Room, as you wish them to appear in dashboards. Letters, numbers, spaces, and apostrophes are allowed. One room corresponds to one Calendar (see above) and one automation. Words like 'Heating' and 'Thermostat' are added later, so do not include them in the room name.

C. Column C (Location) is used when there are multiple devices in the same room to be run from the same calendar and room sensors. Repeat the room name exactly in column A and give the thermostats distinct names for their location within the room in column B. I use North, South, East and West in my example, but it can be any text as long as it is unique for the room. No Location name is needed when there is only one device in the room.

D. Column D (Type) is for the type of thermostat to be used in the name. It will be typically 'TRV' or 'Thermostat', or perhaps 'Generic thermostat' or 'Virtual thermostat' if you created it in Home Assistant.

E. Column E (Domain) can only have one of two values: 'climate' or 'switch'. Devices marked 'climate' will be controlled by Heating X2, devices marked 'switch' will be controlled by Calendar Switch Y2. Note that column D is used in the names and column E is not.

F. Column F (Away) is for the entity id of whatever switch or sensor you are using to determine the 'Away' mode. In my example it is a simple Input Boolean on the dashboard but any binary sensor (including template sensors) or a switch can be used. Typically all rooms will be linked to the same switch or sensor, but you can have more than one. Leave this column blank if you do not want the Away mode facility at all; leave individual fields blank if you want some rooms to be unaffected by Away mode.

G. The blue cells in columns G, H and I give *recommended* (default) names for your zone switches, calendars, and devices. These names are used in the code generator unless you change them. If you used different names when you created the thermostat devices, you should either put the actual name here or change the actual name to match the recommendation.

Once this is done, we are ready for the first code generation. Save and close the workbook for now, but you can always come back if you change your mind.

7.4 General mail merge procedure

All the following steps follow a similar procedure, so the main steps are given here. For further details on how to use mail merge see the Microsoft help pages.

1. Open the one of the files *Code Generator for xxxx.yaml.docx in* WORD.
2. On the first opening, you will be asked to specify the data source. Select *Code Generator DATA – <your home name>.xlsx* (or whatever you named the file in 7.3 above).
3. Agree to the security warning.
4. Accept the default range of 'entire worksheet'.
5. Select the 'Mailings' tab.
6. Click 'Finish and merge' and select 'Edit individual documents'. This opens a new WORD document (typically called *Directory1*, *Directory2* etc.), which contains all the YAML code for xxxx.yaml.
7. You do not need to save this file, just copy the entire contents onto the clipboard
8. In Home Assistant, find the file *xxxx.yaml*.
9. If you have definitions from an earlier build, delete them.
10. Paste the contents of the clipboard, taking care not to overwrite any existing definitions that you still need. It should show as valid YAML code with no errors.
11. Close the *Directory* file without saving it.
12. Close the file *Heating X2 code generator for xxxx.yaml.docx* without saving it.

7.5 Specific YAML generation by mail merge

7.5.1 Generate the input text helper definitions

This step will create the input text helpers needed by all the blueprints. Follow the instructions at 7.4 above with xxxx = *input_texts*.

7.5.2 Generate the input number helper definitions

This step will create the input number helpers needed by Heating X2 (only).

Follow the instructions at 7.4 above with xxxx = *input_numbers*.

7.5.3 Generate the input number boolean definitions

This step will create the input boolean helpers needed by Calendar Switch Y2 and Zone Switch Z2. Follow the instructions at 7.4 above with xxxx = *input_booleans*.

7.5.4 Generate the timer definitions

This step will create the timers as appropriate to all the blueprints. Follow the instructions at 7.4 above with xxxx = *timers*.

7.5.5 Generate the binary sensor definitions

Skip this step if you are not using any zone control.

1. Open the file *Heating X2 code generator for binary_sensors.docx* in WORD.
2. Follow the same instructions as for input numbers, but at step 8 select *binary_sensors.yaml*.

7.5.6 Generate the template sensor definitions

The **Heating X2** blueprint does not require template sensors (since release 2.0), but Zone Switch Z2 does. This step is therefore mandatory if you have zones, otherwise optional, for use in dashboards or other automations.

There are three sensors for each thermostat: set and measured temperatures and heat demand.

The procedure is as for the previous files. Follow the instructions at 7.4 above with xxxx = *templates*.

7.5.7 Generate the automations

If you wish, the code generator can generate the code for *automations.yaml* (a file that you will already have) that invokes the blueprints **Heating X2**, **Calendar Swich Y2** and **Zone Switch Z2**. Due to mail merge restrictions, a separate code generator is required for each type of automation.

Follow the instructions at 7.4 above with each of the three files that is applicable to you:

– *Heating XYX code generator for automations X.yaml.docx*
– *Heating XYX code generator for automations Y.yaml.docx*
– *Heating XYX code generator for automations Z.yaml.docx*

In every case the output goes to ‘automations.yaml’.

7.6 Restart

For the helpers, template sensors and automations to take effect you need to reload the configuration.

1. Check the configuration first in developer tools and address any problems that are reported.
2. If you are using logging then you need to do a full restart; otherwise, a ‘quick reload’ will suffice.
3. You should check the helpers and automations in the UI in case you have any mismatched names.
4. At this point you can add any door and window closure sensors, and any presence sensors to automatons by hand.
5. You might want to adjust the parameters for some rooms.
6. Minor errors in any of the files you generated, such as name-mismatches, can be corrected by hand. If you have a lot of them, you can always go back to the beginning of this chapter and start again.

8 Dashboards

You can build dashboards however you like, but if you are making a new, large installation, you might like an example dashboard to get you started. There are example dashboard cards for thermostats (X), switched devices (Y) and zone control (Z) based on the naming conventions used by the code generator.

I use Mushroom cards and Better thermostat in my examples, so you will need to load those from HACS if you do not already have them.

Due to mail merge restrictions, a separate file is provided for each type of card. Follow the instructions at 7.4 above with each of the three files that is applicable to you:
– *Heating XYX code generator for automations X.yaml.docx*
– *Heating XYX code generator for automations Y.yaml.docx*
– *Heating XYX code generator for automations Z.yaml.docx*

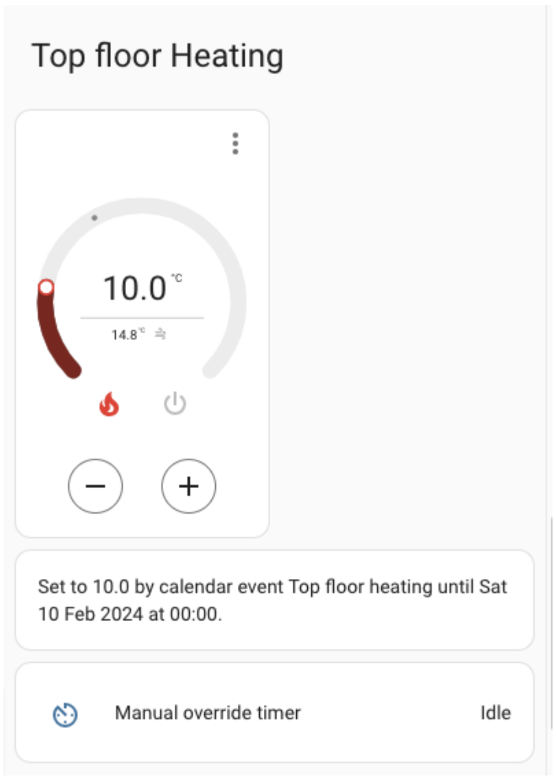
Then, rather than pasting the entire output to a yaml file, the YAML generated by the Code Generator is designed to be pasted into a Lovelace 'Manual' card, one card at a time.

1. Select the code *for one room only* (one page without the comment) and copy it to the clipboard.
2. On your existing dashboard view press 'add card'
3. Select the 'manual' card.
4. Delete the default YAML title
5. Paste the YAML from the clipboard into the card.
6. Correct any error that occur, for example if you used different names for some entities.
7. Click save.
8. Repeat for each of the other rooms.

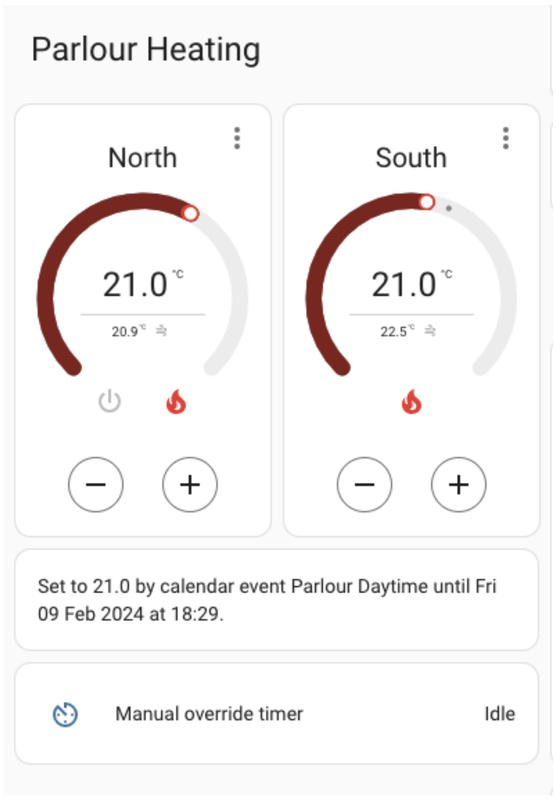
The code is compatible with the UI, so you can make changes using the UI after the initial definition.

8.1 Room thermostat card (X)

My example thermostat card per room is a vertical stack with the name of the room as its title, the thermostats for that room are placed in a grid 2 columns wide, each labelled with its Instance name, so one or an odd number leaves a blank space. Below that are the reason text (as a markdown card because the text is too long for a regular text entity) and the manual countdown timer (which gives you the possibility to reset it).



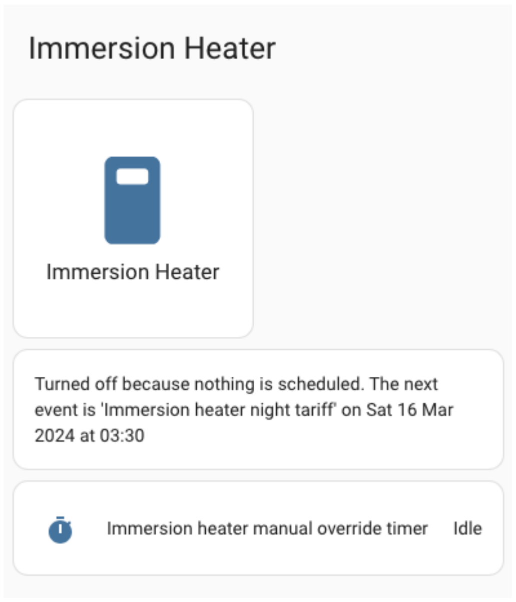
Set to 10.0 by calendar event Top floor heating until Sat 10 Feb 2024 at 00:00.



Set to 21.0 by calendar event Parlour Daytime until Fri 09 Feb 2024 at 18:29.

8.2 Switched device card

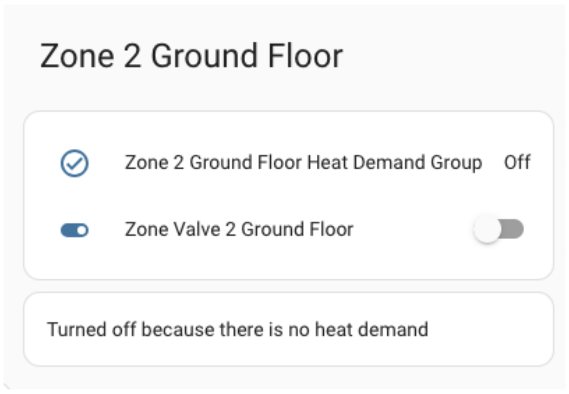
My example for switched devices is similar, with the name of the device as its title, then buttons for each device (though typically only one) for the automation are placed in a grid 2 columns wide, each labelled with its Location name is applicable, so one (or an odd number) leaves a blank space. Below that are the reason text (as a markdown card because the text is too long for a regular text entity) and the manual countdown timer (which gives you the possibility to reset it).



Turned off because nothing is scheduled. The next event is 'Immersion heater night tariff' on Sat 16 Mar 2024 at 03:30

8.3 Zone control card

My example for zone control has the name of the zone as its title, then the zone heat demand sensor and the zone valve switch Below that is the reason text (as a markdown card because the text is too long for a regular text entity).



Zone 2 Ground Floor

Zone 2 Ground Floor Heat Demand Group Off

Zone Valve 2 Ground Floor

Turned off because there is no heat demand

This card should be placed on the same dashboard view as the rooms with thermostats in the zone. It is then easy to check that a demand from any thermostat causes the zone to turn on and it goes off only if no thermostat requires heat.

Note that the dashboard switch is for information only. If it is operated by hand t will be automatically reset to the correct setting.

8.4 Testing tip

While testing, you might like to add some of the other helpers and timers to the cards so that you can better see what is going on. Adding timers gives you the possibility to cancel them.