Smart Heat your Home with Heating X2

Guide

Andy Symons
Version 2.2
27 February 2024

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.

This document is a guide to setting up a smart-heated home using Home Assistant, the **Heating X2** blueprint, the **Heating X2** Code Generator (v2.1) and the **Zone X2** blueprint. The result will be a system with following features:

- Temperature of each room scheduled with a local calendar
- Manual override by a change on the thermostat, dashboard, or voice assistant. The set value is retained for a defined period (default 2 hours)
- Optional closure sensors cause heating to turn off if a door or window is left open for a defined period (default 3 minutes)
- Optional occupancy sensors cause heating to turn off if a room is left unoccupied for a defined period (default 1 hour)
- Warmup period: unoccupancy is ignored for a defined period at the start of a calendar event (default 2 hours)
- Away mode: set all rooms to a specified temperature (default 5C) when there is no one at home
- Background temperature: specify per room the temperature to use when there is no calendar event (default 5C but specified separately from the frost and away temperatures)
- Raises notifications for thermostats that do not respond to a new setting, go offline, or come back online
- Thermostat battery life conserved by only transmitting real changes

1.2 History

Heating X, the first release, was published on the Home Assistant Community Forum and GitHub in February 2023. The current release **Heating X2** (Heating X, release 2) is based on a year of beta testing Heating X in two homes with several rooms each, and feedback from users of the first release.

Heating X2 is published separately from Heating X because it is not 'plug compatible' – the inputs have changed so it is not just a matter of changing the code. Release 1 users should consider upgrading, but a new install will be required. Note however, that the release 2 code generator now includes the generation of the automations. The new blueprint has a different name **Heating X2**, so can co-exist with Heating X.

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 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.

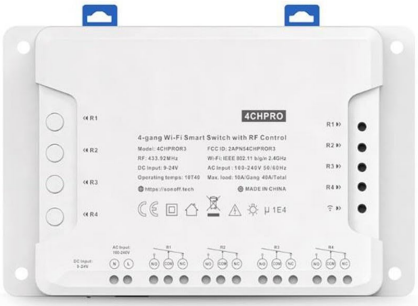
For the heating there will a single wall thermostat. For hot water, there will be a thermostat on the cylinder.

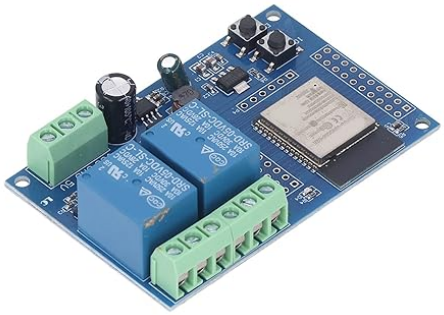
Systems with hot water and heating have either two zone control valves – called 'S-plan' – or a single 3-position valve – called 'Y-plan'.

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 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.

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. In other words, when all the TRVs are shut will it automatically stop pumping and heating? Many will. Some require a wiring change e.g. to allow the boiler to turn the pump on and off. Some require a by-pass valve just in case. Some plumbers leave one radiator always 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.

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 also 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 added to **Heating X2**.

2.4 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).

2.5 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 Download the Heating X2 Blueprint

3.1.1 Automatic install via Home Assistant

1. To open your Home Assistant instance and show the blueprint import dialog with the **Heating X2** blueprint pre-filled, click this link
2. Follow the Home Assistant instructions

3.1.2 Manual install

1. Using your preferred Home Assistant file editor create an empty file */config/blueprints/automation/AndySymons/heating_x2.yaml*
2. Go to the Github page
3. Copy the YAML code to the clipboard
4. Paste it into the new file

3.2 Download the Code Generator

Each instance of **Heating X2** requires a number of helpers (input texts, input numbers and timers). You *can* define these by hand, but to help set up a large installation, there is a code generator that creates all the YAML code you need for helpers, templates, automations and dashboards from a single EXCEL spreadsheet that 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).

To get the Code Generator:

1. Go to the Github Code Generator repository
2. Download the ZIP file
3. Unzip it on a local drive because Mail Merge can have difficulties with cloud drive systems such as iCloud and OneDrive.

The 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.

After unzipping, you will have a folder with an EXCEL workbook for your room and thermostat data, and a number of WORD mail merge files corresponding to the YAML files to be created:

1. Heating X2 code generator DATA.xlsx
2. Heating X2 code generator for input_numbers.docx
3. Heating X2 code generator for input_texts.docx
4. Heating X2 code generator for timers.docx
5. Heating X2 code generator for templates.docx
6. Heating X2 code generator for heating control automations.docx
7. Heating X2 code generator for binary_sensors.docx
8. Heating X2 code generator for zone control automations.docx
9. (Dashboards, choose one)
   - 9a. Heating X2 code generator for dashboard view.docx
   - 9b. Heating X2 code generator for dashboard cards.docx

3.3 Download the Zone X2 Blueprint

You only need this if your system requires zone control (see above).

3.3.1 Automatic install via Home Assistant

1. To open your Home Assistant instance and show the blueprint import dialog with a specific blueprint pre-filled, click this link
2. Follow the Home Assistant instructions

3.3.2 Manual install

1. Using your preferred Home Assistant file editor create an empty file */config/blueprints/automation/AndySymons/zone_x2.yaml*
2. Go to the Github Gist
3. Copy the YAML code to the clipboard
4. Paste it into the new file

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).

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)

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.

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 Helpers

The blueprint requires 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).

If you cannot or do not want to use the Code Generator, you can create the helpers by hand for each automation (room) that you are going to create with the blueprint.

5.1 Input_text helpers

- Setting reason – into which the automation writes the reason for the current setting for display on dashboards. Make the length 255 characters.

5.2 Input_number helpers

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

5.3 Timers

- Warmup timer – to hold the timer for the event warmup period
- Manual override timer – to hold the timer for a manual intervention
- Door or window open timer – 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 – 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)
- Echoblock timer – 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.

6 Using the blueprint by hand

Automatons 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.

To create by hand you need to create one for each calendar – i.e. per room thermostats, and for other devices that are to be controlled by a calendar. Select Blueprint **Heating X2** and specify all the inputs described above – the devices, 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.

6.1 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.

7 Logging

Since release 2.2 logging of key steps in automations built with the **Heating X2** blueprint 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: echoblock_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: echoblock_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. Messages in the log files are always prefixed with the name of the automation that wrote them -- '[Parlour heating control]' in the sample.

7.1 Defining the log notification services

If you want a lot of files, e.g. one per room, then you may prefer to use the Code Generator – see below.

To create one by hand, add the following to your *configurations.yaml* file:

```yaml
notify:
### Heating log
  - name: heating_log
    platform: file
    filename: heating_log.txt
    timestamp: true
```

Of course 'heating_log' is an example: you can use any name you like.

If you already have a 'notify:' key, do not duplicate it, put this code under it.

You do not have to create the `heating_log.txt` file, it will be created automatically the first time the automation runs.

If you prefer to have your log file(s) in a subfolder, you can write -- e.g. for a folder called ‘log_files’:

```
notify:
### Heating log
  - name: heating_log
    platform: file
    filename: log_files/heating_log.txt
    timestamp: true
```

You DO have to create the folder by hand.

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

7.2 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 name attribute in the configuration – in the above examples it is ‘heating_log’.

Unfortunately, Home Assistant does not at present provide for a selector of services, so this is a plain text field. It is recommended you copy and paste the name from the configuration. Do not add any prefix (like ‘notify.’).

8 Using the Code Generator

If you use the code generator, you should not create any of the 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!

8.1 Prepare your system

The code generator is going to generate code to paste into separate YAML files for each type of helper and the templates, 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_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_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.

8.2 Enter your zone, room, and thermostat information

In your code generator folder downloaded earlier, open the first file ‘Heating X2 code generator DATA.xlsx’ in EXCEL. This is the data source for all the later mail merges, specifying the rooms and thermostats of the whole heating installation. It initially contains details of a fictitious installation by way of example. You should delete the data in the yellow cells and replace it with your own. 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.

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 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.

In column C (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.

Column C (Thermostat) is used when there are multiple thermostats for 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 Thermostat name is needed when there is only one thermostat in the room.

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.

Column E (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.

The blue cells in columns F, G and H give *recommended* (default) names for your zone switches, calendars, and thermostat devices. These names are used as such 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.

8.3 Generate the input number helper definitions

1. Open the file 'Heating X2 code generator for input_numbers.docx' in WORD.
2. On the first opening, you will be asked to specify the data source. Select (your edited copy of) 'Heating X2 code generator DATA.xlsx'.
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 'Letters1', 'Letters2' etc.), which contains all the YAML code for input_ numbers.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 'input_datetimes.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 'Letters' file without saving it.
12. Close the 'Heating X2 code generator for input_numbers.docx' without saving it.

8.4 Generate the input text helper definitions

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

8.5 Generate the timer definitions

1. Open the file 'Heating X2 code generator for timers.docx' in WORD.

2. Follow the same instructions as for input numbers, but at step 8 select ‘timers.yaml’.

8.6 Generate the template sensor definitions

The **Heating X2** blueprint does not require template sensors (since release 2.0), but the code is available in case you want them for use in dashboards or other automations. There are sensors for set and measured temperatures and for heat demand, which are used for zone control (see below).

The procedure is as for the previous files.

1. Open the file ‘Heating X2 code generator for templates.docx’ in WORD.
2. Follow the same instructions as for input numbers, but at step 8 use ‘templates.yaml’.

If you regenerate Release 1 template sensors, you will find that duplicate entities appear in the system (as seen in the entity settings) because I previously used random numbers for the unique ids. You therefore need to go to the Home Assistant settings entities list and

a) delete by hand all the versions that have the error ‘no longer provided by …’
b) edit the entity name in the new ones to be the same as the one the deleted, i.e. without "_2" appended.

The good news is that I learned from this mistake and the code generator now uses fixed unique ids based on the entity id, so next time there should be no duplication: new entities will replace the old ones.

8.7 Generate the heating control automations

If you wish, the code generator can generate the code for *automations.yaml* (a file that you will already have) that invokes the blueprint **Heating X2**, once for each room, with the correct thermostats and helpers (as generated above), and the default parameters. Using the code generator saves time to get started if you have a lot of rooms by ensuring that the right helpers and timers are linked to the right automations, which is easy to get wrong in the UI! The new automations can still be edited in the UI afterwards.

You can skip this step altogether if you prefer to generate the automations by hand (see above).

For automatic generation:

1. Open the file ‘Heating X2 code generator for heating control automations.docx’ in WORD.
2. Follow the same instructions as for input numbers, but at step 8 use ‘automations.yaml’. Take particular care not to disturb existing automations unless you are replacing them.

8.8 Generate log file definitions

The Code Generator provides the YAML code to create one log file per room, stored in a sub-folder of /CONFIG/ called ‘log_files’. Other configurations are possible but would need to be created by hand as described above.

1. You must create the ‘log_files’ folder by hand as a subfolder of /CONFIG/.
2. Open the file ‘Heating X2 code generator for logging service definitions.docx’ in WORD.
3. Follow the same instructions as for input numbers, but at step 8 select ‘configurations.yaml’.

8.9 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'.

8.10 Generate the zone control automations

Skip this step if you are not using any zone control, or if you prefer to generate the automations by hand – which is very simple in this case: you just need to identify a heat demand group (binary sensor) and the matching zone control switch.

For automatic generation:

1. Open the file 'Heating X2 code generator for zone control automations.docx' in WORD.
2. Follow the same instructions as for input numbers, but at step 8 use 'automations.yaml' (for the second time). Take particular care not to disturb existing automations unless you are replacing them.

8.11 Restart

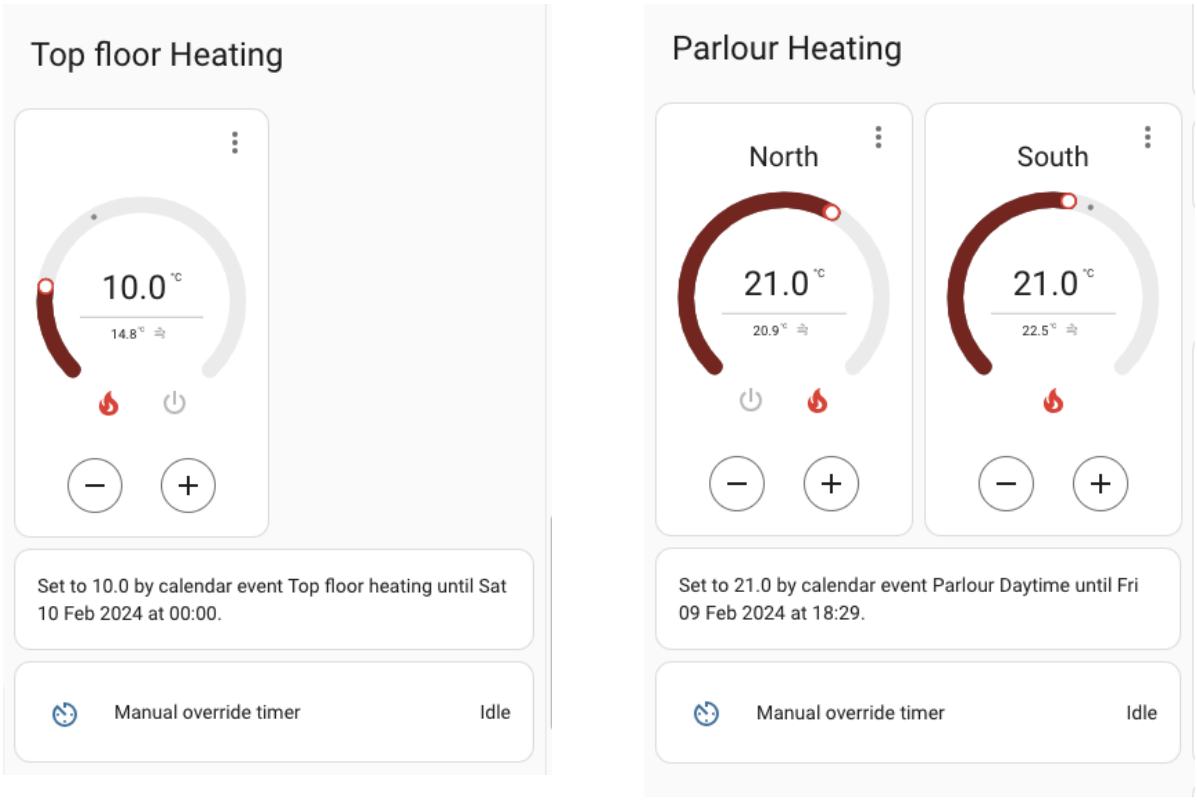
For the helpers, template sensors and automations to take effect you need to reload the configuration.

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.

9 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.

My example 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).



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

To build your started dashboard, you can choose either to add a complete view with cards for all rooms or add one card at a time to your own view.

9.1 Create a whole dashboard view

To create the YAML for an entire view that will display all the room cards in the same order you listed them in your DATA sheet:

1. Open 'Heating X2 code generator for dashboard **view**.docx'
2. Use mail merge as in the previous steps.
3. Copy the entire file to the clipboard.
4. Use the dashboard editor UI to create an empty View and name it e.g. Heating
5. Select the 'Edit raw configuration' option in the dashboard editor menu (top right).

6. Find the empty view you just created -- probably at the end of the file. It will look something like:
```
  - title: Heating
    path: heating
    icon: mdi:heat-wave
    subview: false
    badges: []
    cards: []
```
7. Remove the last pair of square brackets, which indicates that there are no cards
8. Paste the YAML code starting on the next line
9. Save, exit the raw editor, and press DONE
10. You should now see the whole view with a card for each room in the order listed in the DATA file and a thermostat sub-card (half width) for each thermostat in the room.

9.2 Create dashboard cards one room at a time

If you prefer to add one card at a time to your own layout, the second option generates exactly the same code, except that the dashes and indents are adjusted to suit pasting into a card.

1. Open 'Heating X2 code generator for dashboard **cards**.docx'
2. Use mail merge as in the previous steps.
3. Select the code *for one room only* (one page without the comment) and copy it to the clipboard.
4. On your existing dashboard view press 'add card'
5. Select the 'manual' card.
6. Delete the default YAML title
7. Paste the YAML from the clipboard into the card.
8. Click save.
9. Repeat from step 3 for each of the other rooms.

9.3 Testing tip

While testing, you might like to add some of the 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.