Jeff's heating control algorithm

Author Jeff Thompson

Contact Domoticz forum - @jefft

Github repo github.com/jefft4/Heating-control-algorithm

Version 2.4; June 2021

User Guide

This is a pretty long document; I hope that means it's pretty comprehensive and you can find what you need here. If not, please ask on the Domoticz forum!

Features

- Handles any heat source and heater type that can be represented in Domoticz as the required simple devices
- Works with boilers / heat sources where the temperature can be controlled and those where it's fixed; uses both temperature modulation and time modulation
- Works with any valve type (best used with TRVs of some kind)
- Predicts and pre-heats to meet next set temperature (e.g., warm rooms ready for your arrival home, no cold bedrooms when you wake up), allowing for heat loss
- Allows for heat source warm-up time
- Any number of zones; handles both hot water and room heating
- Zones configurable - indoor and outdoor temperature sensors, setpoint, two valves/controls, heating and heat loss rates, preferred and maximum pre-heating periods
- Heat source configurable – warm-up rate, maximum and minimum temperatures, minimum on time

Setting up

To get things up and working, you need to copy the code into a new event in Domoticz, set up the triggers, and then create a heat source, one or more heating zones and their associated sensors and valves.

The code – heating algo.

- Clone the code from the Github repo using git tools or by downloading the Zip file. There's one file of code for the main algorithm, one for the 'door open' handler, and a couple of documents.
- Create a new event in Domoticz, type dzVents/minimal, delete all the pre-populated code, then copy and paste all of the heating algo code into the event and save it. Same for the open door handler, if you want that.
- I recommend setting your new events 'OFF' at the top of the editor, until you've set up the devices and variables that support them.

There are some things that you need to change in the algorithm dzVents code to make it run in your system:

Triggers:

- Near the end of the file, in the main event code block, are the dzVents triggers for the event.
- Set the 'timer' clause to suit you; this time should match the 'cycleTime' value in the global user variable (see below). Normally, 15 minutes is a good choice but slow-reacting systems like oil-fired boilers may need a longer cycle.
- Edit the 'devices' list to include all the things that should trigger the algorithm. Generally, this is your heating zone setpoint devices. It may be wise to set up only a couple of these initially while you get things working.
- Look at the code in the main event block for the boiler master switch and the service switch. If you want these, adapt the code and create devices as necessary to make it work for you. If you don't want this functionality, remove these two if…end blocks and the two device trigger lines. *If you leave the code and don't create the two devices, it will fail!*

- Decide whether you want debug-level logging; in a live system, I wouldn't, but it is helpful for seeing the calculations and decisions that the algo makes while you're setting up. Comment (--) or uncomment the relevant 'DEBUG_INFO' line in function lognote, right at the top of the code.

Once you've created the event and adapted the triggers to suit your system, you'll need to set up some devices and user variables to define and drive the system. Read on!

The semaphore devices

You will need to create two dummy on/off switches called 'Heating algorithm pre-lock' and 'Heating algorithm post-lock'. These are used by the main event block to avoid the algorithm running repeatedly when it doesn't need to.

Timing and triggers

You can have anything trigger the algorithm in dzVents, of course, but the triggers are worth considering, as you can cause some strange effects with them!

- In most cases, timing the event to run every 15 minutes is good. If you want more complex time conditions, check the dzVents documentation; it can do just about anything.
- In general, don't use temperature sensors as triggers. They may report much more frequently than the heating algo needs to be run. Imagine 5 sensors, each reporting every 5 minutes: that could mean the algo is run on average every 1 minute… madness!
- Setpoints are the best triggers, especially if they have a manual override facility, so that users see the effect of their manual input right away. If you don't include them as triggers, all changes will be processed by the algo's next scheduled run.
- If you have other manual controls, those will need to interact with the algo via setpoints, with the exception of any master on/off type switches which can be handled in the main event block.

Heat source

There's just one of these. It's defined by a single user variable.

Devices:

- A setpoint device for its output temperature; if your source isn't controllable, the device can be a dummy setpoint but it still needs to be there
- A temperature sensor for its output ('flow') temperature and one for its 'return' temperature. If there's only one temperature (e.g. electric heating) then these can be one device. If you don't have measurement of one of these temperatures, create a dummy device and fix its temperature at a sensible value. *More on this in the Questions section.*

User variable: 'Heating control global'

- JSON string, enclosed in {}

"cycleTime"	Integer, minutes	Normal period for reviewing heating situation ** Must match the dzVents time trigger clause! Also check heaterMinOnTime – cycleTime shouldn't be shorter and ideally should be at least double the on-time.

“heaterMinTemp”	Integer, degC	Minimum output temperature of the heat source
“heaterMaxTemp”	Integer, degC	Maximum output temperature of the heat source
“heaterMinOnTime”	Integer, minutes	Minimum on-time of the heat source (to prevent short cycling/damage); refer to your heat source documentation. Typical values are 5~6 mins for gas boilers and 15 mins for oil. For electric heat, this could potentially be very short.
“idxFlowTemp”	Integer	Domoticz idx of the flow temperature sensor
“idxReturnTemp”	Integer	Domoticz idx of the return temperature sensor
“warmUpRate”	Integer, degC/min	How fast the heat source output warms up
“flowSet”	Integer	Domoticz idx of the setpoint device for the heat source’s output temperature

Zones

A heating zone is a distinct part of the building for which you want to control the temperature and ideally, to which you can control the supply of heat independently of other zones.
Zones are defined by a user variable each. The name must begin ‘Heating zone ‘; the rest of the name is purely descriptive.

Devices:

- A setpoint device (which you can use for scheduled temperatures and manual override). This can be a smart TRV’s setpoint – the algorithm has a lockout to prevent loops when it changes the setpoint.
- A temperature sensor device. This can be a smart TRV’s sensor device.
- Associated valve device(s) – see ‘Valves’ below.

User variable: ‘Heating zone X’

- JSON string, enclosed in {}

“sensor”	Integer	Domoticz idx of the temperature sensor for the zone
“age”	Integer, minutes	Maximum age of a sensor reading to be considered valid
“setpoint”	Integer	Domoticz idx of the zone’s setpoint device
“valve”	String	Local zone valve ID for this zone – see Valves below
“mvalve”	String	Master valve ID for this zone – see Valves below
“incRate”	Float	1000 * degC rise/minute/degC delta, rate of heat input to the zone (delta = source temp – room temp)
“decRate”	Float	1000 * degC rise/minute/degC delta, rate of heat loss from the zone (delta = room temp – outdoor temp)
“outSensor”	Integer	ID of the zone’s outdoor temperature sensor – see Sensors below
“prefPHT”	Integer, minutes	Preferred pre-heating time ahead of setpoint rise
“maxPHT”	Integer, minutes	Maximum pre-heating time ahead of setpoint rise

“minDT”	Integer, degC	Minimum useful difference between source and room temperatures
“hyst”	Float, degC	Hysteresis – how far we let the temperature vary from the setpoint before we do something about it
“sw”	Integer	Domoticz idx of the master control for this zone – if this switch is off, the zone will be ignored

Either valve or mvalve must be valid; the other is then optional.

Valves

A valve controls the supply of heat to a zone. The algorithm treats all valves as being either on or off, but will work with valves that need other control approaches like % open or heat mode.

Valves are defined by a user variable which describes the actions to turn them on and off. Valve ID ‘Y’ as used in a zone will be defined by user variable ‘Heating valve Y’.

Devices:

- Any of:
  - On/off switch
  - Percentage switch (dimmer)
  - Thermostat mode device
  - Thermostat setpoint device

User variable ‘Heating valve Y’:

- JSON string, enclosed in {}

“idxOnOff”	Integer	Domoticz idx of zone on/off switch
“idxPercent”	Integer	Domoticz idx of zone percentage switch (typically valve open %)
“idxMode”	Integer	Domoticz idx of zone thermostat mode device
“strModeOn”	String	String value to set the mode device to heat
“strModeOff”	String	String value to set the mode device off (or to Eco mode if preferred)
“idxSetpoint”	Integer	Domoticz idx of zone setpoint device

- At least one of the four idx values must be valid. If it’s idxMode then at least one of the two strings should also be present. A missing mode string will result in no command being sent. All the options present will be processed each time the algorithm changes the valve state, in the order listed.

Master valve vs local valve

Each zone can have two valves. There is only one difference in how these are handled: the master valves are considered to be those which can activate the heat source (‘call for heat’) and so the algo will ensure that in each heating cycle, at least one master valve in the system is on for at least the specified minimum on-time ‘heaterMinOnTime’. This is to protect boilers against being cycled on and off more rapidly than is good for them. So, if your valve controls a boiler, make it the mvalve; if it’s truly just a zone valve or TRV, it can be either valve or mvalve.

Master switch

Sometimes it's useful to be able to disable part, or all, of the heating system – for example, for servicing or if you're going away and don't leave it on. Each zone has a 'sw' attribute, which points to a domoticz switch device. If that switch is not on, the zone is ignored by the algorithm. I use the a single heating master switch device, linked to all the zones.

Sensors

Zone temperature sensors are simple domoticz temperature (or TH/THB) devices – anything that will return a result when dzVents asks for device(x).temperature. These are defined in the zone by their Domoticz idx value.

Zone outdoor temperature sensors can include future predictions of the temperature, taken from a weather forecast API or other source. These are defined in the zone by a string ID – e.g. 'OUT1', which references the user variable 'Heating sensor OUT1'.

Devices

- For zone (indoor) sensor, any temperature device.
- For outdoor sensor, any temperature device for current temperature plus 4 temperature devices for the next 4 hourly predicted temperatures.
- If prediction is not available or not required, then the predicted temperature sensors can be the same device as the current temperature sensor. Similarly, if you have fewer than 4 prediction values, just duplicate the idx values to fill the list.

User variable 'Heating sensor OUT1':

- JSON string, enclosed in {}

"idxNow"	Integer	Domoticz idx of the current temperature sensor
"idx1"	Integer	Domoticz idx of the temperature sensor for 1 hour in the future
"idx2"	Integer	Domoticz idx of the temperature sensor for 2 hours in the future
"idx3"	Integer	Domoticz idx of the temperature sensor for 3 hours in the future
"idx4"	Integer	Domoticz idx of the temperature sensor for 4 hours in the future
"default"	Float	Default value to use if the sensor data isn't up-to-date/sensible

Tuning the algorithm's behaviour

The algorithm is driven by a number of parameters defined in the user variables:

- incRate and decRate for each zone define how quickly that zone heats and cools
- warmUpRate (global) defines how long it takes the heat source to warm up
- minDT, hyst and the other heater parameters also have some effect but are less system-specific.

Setting the heating rates

To calculate the incRate and decRate, use real data. Make an initial estimate, then observe your system's behaviour over a period of time and use what you observe to fine-tune it.

For incRate, try to look at situations where most/all zones are heating constantly for a period of time (the algorithm assumes that your heat source is big enough to heat all zones together). Divide the degC temperature increase for a zone by the time taken to get that increase (in minutes) and then divide by the degC average difference between the heat source temperature and the zone temperature. Finally, multiply by 1000 – we do this just to make it more friendly in the user variable and because domoticz user variables are limited to a rather mean 200 characters!

For decRate, it's a similar approach. The best period to sample is probably just after your heating switches off for the night, through until the lowest zone temperature is reached or the heating switches on again. Same calculation as for incRate, but use the decrease in zone temperature rather than the increase (the result needs to be a positive number).

If you want a starter for ten; my heating rates vary between 0.3 for a room with a slightly undersized radiator and 1.2 for the hot water cylinder. Upstairs increase rates are higher, thanks to the laws of thermodynamics – downstairs zones heat upstairs to an extent whether I like it or not. My decrease rates range from 0.44 for a room with an uninsulated north wall and 8sq.m of glass to 0.2 for the main upstairs zone.

Other tweaks

If you find that your rooms are getting a little too cold before the heating kicks in, reduce the hysteresis value. Conversely, if you find that your heating is running a lot for short periods and possibly over-heating you then letting you cool off (it has to obey minimum on periods and temperatures), then try increasing the hysteresis.

If you want a suggestion, I'd set hysteresis to 0.5 for room zones and perhaps 3 or even 5 for stored hot water. My minDT is 10 degrees – that is, radiators need to be 10degC above the room temperature before they have much effect.

Questions

How does the algorithm behave with TRVs?

The algorithm will work with both dumb and smart TRVs, and a mixture of the two. I personally have a handful of smart TRVs in tricky rooms and dumb ones everywhere else.

Open window detection can be enabled – it causes the valves either to close or to reduce the set temperature. If the valve closes without dropping the set temperature, the algo won't know and so will continue to try to heat (possibly more so as the temperature will have dropped). If you want to be smarter and more robust with open door/window detection, consider having some custom code that's triggered by door & window sensors and reduces zone setpoints temporarily… I've published mine alongside the heating algo, as a starter.

TRVs can be opened and closed by one or more of several approaches; that's why I have four options in the valve definition. For example, my Eurotronic Spirit Z-Wave TRVs can be controlled by setting the mode + setpoint, or by setting the mode to 'Manufacturer specific' and then setting the valve %. Find out what works for your valves and use it.

The algorithm does not try to make use of **partial (%) valve opening**. That might be something to consider in a future enhancement; it's potentially useful as a substitute if you can't control the heat source temperature. For now, modulating the on-time achieves pretty much the same result and the algo does that automatically. Not forcing the valve % also allows the valve to apply its own logic for gradually closing the valve as the temperature approaches the target, which is sometimes useful in avoiding overshoot if the sensors are lagging.

Undershoot: TRVs will gradually close as they get close to the set temperature to avoid overshooting. Combined with the algo reducing the heat input for the same reason, that may lead to the zone slightly undershooting the target temperature or achieving it a little late.

If that happens too much with dumb TRVs, then you might consider setting the valve a little higher than your highest target temperature.

Smart TRVs should be told what to do by the algorithm; if they are closing as they approach the target and causing an undershoot then check whether you can control them by a different approach (e.g., having the algorithm set valve % rather than, or in addition to, setting the mode or septoint). I contorl mine by the setpoint and mode, leaving them free to decide the right valve position.

Potential conflict with the algo.

Through my testing, I've found that 'smart' TRVs are rarely all that smart. They have a schedule, a temperature sensor and a valve. The logic connecting those three things doesn't generally attempt prediction, allow for warm-up times, read the weather or anything else; those advanced calculations are the job of central controllers in systems like Tado and Bosch.

If you set a temperature schedule on a smart TRV and find that it 'fights' with this algorithm during pre-heating periods (the valve still thinks that the temperature should be low, while the algorithm is trying to warm up ready for the coming increase), then you need to separate the schedule from the valve. Delete the valve's schedule and use a dummy Domoticz setpoint device instead, with the algo controlling the valve's temperature via the valve definition 'setpoint' element.

What's the impact on the algorithm if my heaters don't have TRVs?

A zone with no TRV will be treated in the same way as a zone with a TRV; both maintaining a set temperature and pre-heating will work. However, there is a greater chance of overshoot; that is, the room temperature exceeding the set level a little after the system stops heating. TRVs will close themselves (or be closed by the algo) as the temperature gets close to target, avoiding any significant overshoot; a zone without a TRV can't do this, so will have more residual heat in the heaters and will continue to warm the room for a while.

It should be possible to include a calculation for the potential over-run in zones that need this. I might add it if there are enough use cases that need it. Wet heating systems without TRVs are increasingly rare, but there may be electric, UFH and other systems that don't have the equivalent.

How often do my temperature sensors need to report?

More often is generally better than less often, but don't go mad and swamp your system with data! The key is to pick up changes in the room temperature quickly, to avoid the situation where the temperature overshoots because the sensors haven't caught up. I have mine set to report every 5 minutes with the algo running every 15 minutes and it seems to work pretty smoothly.

Some sensors can't be told to report periodically; for example, my TRVs report only changes in temperature, so I have them set to report on a 0.1degC change to minimize the lag. The accuracy and reporting behaviour of the sensors is also relevant to the hysteresis for the zone: don't set hysteresis smaller than the smallest change your sensors will detect and report; it's unlikely to work well.

How long should I set the 'age' parameter for my sensors?

This depends on your sensors; it should be long enough that one missed report doesn't cause the sensor to be ignored (which will cause the algo not to heat the zone) but not so long that you could be using a temperature that's horribly wrong. I'd go with 2~3 reporting intervals, if those are fairly short.

The one troublemaker here is sensors that only report when the temperature changes, like my TRVs. I have the age for those set to 2 hours; it's way longer than I'd like but it does mean that if the temperature really is very stable, we're not likely to ignore the zone and miss the need to warm it up for the next setpoint. *Need to find a smarter way of testing whether a temperature is still valid… any ideas?*

What do I use for the heat source set temperatures and sensors, if I don't have / can't control those?

If you don't have control of the source temperature (very common, as the flow temperature of most boilers is set by a knob on the front panel, unless you have an OpenTherm or EMS gateway device connecting them to Domoticz), then find out what the supply temperature really is and set both max and min supply temperatures to that value. Let it warm up fully before measuring temperature and measure directly on the flow pipe from the boiler/source to the heaters.

If you can't feed Domoticz with the flow and return temperatures, then set up a dummy temperature sensor device and force its value to the flow temperature. The return temperature isn't currently used in the algorithm, so just point it to the same device.

I have an idea to use the return temperature in the future to improve the warm-up calculation; if I get around to that, it'll come with new advice.

I set my Bosch EMS boiler's flow temperature but a few minutes later, it resets to the value on the control panel!

Yes, it does. Welcome to the wonders of EMS! Unfortunately, EMS boilers seem to revert to the panel setting if they've not had a contrary command in the last few minutes. I get around this in my system with an event that runs every 2 minutes and re-asserts my view of what the flow temperature ought to be. Happy to share that little piece of code if you need it, just ask me!

How do I handle water heating?

If your heat source also heats your hot water, as long as there are devices that the algorithm can treat as the valve and sensor for the hot water 'zone', then it's just another zone; one with much higher target temperatures! I define my hot water with the indoor temperature from a sensor on the cylinder and the outdoor temperature from the garage sensor (that's where the cylinder is located) – all five outdoor idx's point to the same sensor, as there's no prediction).

How do I use the local valve and the master valve?

The two are treated the same, except for the heaterMinOnTime check for master valves (see Valves section above). At least one of the two must be present in each zone definition.

If you need to activate only one device (valve) to heat a zone, then define just that device. If it is a device that causes the heat source to switch on/off, then it must be defined as the master 'mvalve'. If not, then I would still use 'mvalve' but it doesn't actually matter.

If your system needs to activate more than one device to heat the zone, you'll need to use both the master and the local. For example, my system is an old-style 'S-plan' system with a master valve each for heating and hot water, both of which 'call for heat' to activate the boiler. I also have some smart TRVs in specific zones. So, I have the 'mvalve' for all my heating zones set to the S-plan heating valve and the 'valve' set to the local TRV where there is one or omitted otherwise.

What if I don't have some of the sensors?

A zone must have a zone (indoor) temperature sensor and an outdoor sensor. If you don't have an outdoor sensor, then set up a weather forecast API in Domoticz and take the current & future temperatures from that. Or at worst, dummy it with a fixed value – e.g. the standard UK average of 15degC.

Do sensors and/or valves need to be unique to a zone?

No.

Multiple zones often share something like a S-plan heating valve. Typically, this is the 'master' valve for the zones.

The outdoor temperature sensor is likely to be shared by most/all zones as there's only one outdoors (okay, if you want to be really clever and account for solar heating as the sun moves around, I guess you'll want more than one!)

Indoor sensors are generally unique to one zone, although setting up multiple zones with a shared indoor sensor could be a solution if you have more valves than sensors.

How do I implement an 'away' or 'holiday' mode?

The smart way to do this is via the setpoints. Away and holiday modes generally set the heating to a low temperature to prevent the system freezing. Domoticz provides timer plans that enable you to switch your entire Domoticz system over to a different set of schedules (a word to the wise, create

the timer plans and select one before you start creating schedules; the default configuration is lost when you create a new plan); alternatively, just edit the setpoint schedules to add some date-specific values covering your holiday and temporarily disable the normal schedule lines.

If you need the system to be completely off, then I would create a master switch like my 'Boiler master', or use the 'sw' attribute of each zone.

Why use the 'sw' zone attribute and a master switch in the code?

The master switches ('Boiler master' and 'Boiler manual/service') disable the entire system – the algorithm exits as soon as it detects one of those isn't on.
The zone 'sw' facility is useful when used with multiple zones and multiple switches; for example if we create one switch for hot water and another for heating, then we can disable the heating in summer while keeping the hot water active, or leave the heating active during a vacation but not the hot water.

Can the algorithm handle more than one heat source?

Not in a single system. You'd have to adapt the code so that each zone specifies its heat source and there's a global data chunk for each source.

Troubleshooting

I'm sure this section will grow as people use the algorithm!

The key points

(1) Check your definitions carefully. Don't add ten of them all at once; add enough to drive the source and one zone, test, then add more if all's well. The JSON strings are case-sensitive.
(2) Check the log for warnings / errors.
(3) If the behaviour isn't as expected, edit the function 'logNote' to uncomment 'DEBUG_INFO = true' and comment 'DEBUG_INFO = false'. The algorithm will then write a ton of detail to the domoticz log, from which you should be able to follow its thought process and calculations. *NB! The web UI's log page sometimes doesn't keep up with the volume of records written by the algorithm in a short time; look at the domoticz.log file directly to be sure you see everything.*
(4) Ask on the forum. Other people will often be the best source of ideas and experience. I'll also try to monitor the thread and respond as quickly as I can.

The algorithm appears to be running several times within a very short period

You'll need to look at what's triggering it. It's possible that you may just have several things in your system changing, in which case you can either live with it or adjust how your system is scheduled/set up.

If you have a very large number of setpoints changing at the same time, a system that is very slow to respond to setpoint change commands, or a slow processor running Domoticz, then it's possible that there is a long queue of setpoint change events and the algorithm's lockout periods aren't long enough to prevent it being triggered several times. If this is the case, you can adjust the '.switchOff().afterSecs(xx)' clauses for whichever of the pre-lock and post-lock needs to be longer. If the triggers are scheduled changes, it's the pre-lock; if the algo's own changes to setpoints are causing re-triggering immediately after it runs, it's the post-lock that needs to be longer.

Example systems – how they could work

1. My system: Wet CH & HW, S-plan plumbing, Worcester-Bosch Greenstar gas system boiler with temperature control and reporting via a EMS-ESP module. Mostly dumb TRVs, some smart ones in the tricky rooms. Also a smart on/off control for the kitchen heaters.

   Heat source (Heating control global): {"cycleTime": 15, "heaterMinTemp": 35, "heaterMaxTemp": 75, "heaterMinOntime": 5, "warmUpRate": 4}

   Zones: Downstairs general ('Hall'), Upstairs general ('Landing'), Kitchen, Lounge, Study, Bedroom 3, Bedroom 1, Hot water. Typical incRate

   Heating zone lounge: {"sensor": 1730, "age": 15, "setpoint": 2157, "mvalve": "CH", "valve": "LL", "incRate": 0.4, "decRate": 0.44, "outSensor": "OD", "prefPHT": 30, "maxPHT": 90, "minDT": 10, "hyst":0.3} – Note the low hysteresis; people sitting still and inactive are likely to feel smaller temperature changes.

   Heating zone landing (this steals heat from downstairs, hence the higher incRate): {"sensor": 507, "age": 15, "setpoint": 1718, "mvalve": "CH", "incRate": 1.2, "decRate": 0.2, "outSensor": "OD", "prefPHT": 30, "maxPHT": 90, "minDT": 10, "hyst": 0.5}

   Heating zone hot water: {"sensor": 1150, "age": 15, "setpoint": 1717, "mvalve": "HW", "incRate": 1.2, "decRate": 0.25, "outSensor": "GG", "prefPHT": 30, "maxPHT": 60, "minDT": 10, "hyst": 3.0} – heats up much faster than any room; uses the garage temperature as its 'outdoors'.

   Indoor sensors: Hall, Landing, Kitchen, Lounge, Hot water are DIY sensors (ESP8266+DHT22); Study, Bed 3, Bed 1 use the sensors on their TRVs.

   Outdoor sensors: real sensor for current temperature, weather forecast API providing 4 x forecast temperatures for heating zones. Hot water cylinder uses current garage temperature as its 'outdoors' for both current and forecast.

   Heating sensor OD (outdoors): {"idxNow": 1562, "idx1": 1826, "idx2": 1833, "idx3": 1840, "idx4": 1847} – the idx1-4 are TempHumBaro Hour 1-4 from OpenWeatherMap in Domoticz.

   Valves: Master HW, Master CH – the S-plan valves. Zone local valves in Study, Lounge, Kitchen, Bed 1, Bed 3. Other zones have no local valve.

   Heating valve LL (the lounge): {"idxMode": 2214, "strModeOn": "Heat", "strModeOff": "Off", "idxSetpoint": 2215} – Eurotronic Z-Wave TRV; set the mode and the setpoint to get action.

   Heating valve CH (the heating master/S-plan): {"idxOnOff": 1708} – a simple Z-Wave relay.

2. Underfloor heating; each zone has its own sensor and valve; source temperatures are lower and warm-up is slower
   Heat source: {"cycleTime": 15, "heaterMinTemp": 30, "heaterMaxTemp": 50, "heaterMinOntime": 5, "warmUpRate": 2}

   Zones: one per physical heating zone
   Indoor sensors: one per physical heating zone
   Outdoor sensors: one real, 4 from weather forecast

Valves: each zone has master valve set to the boiler control (on/off) and local valve set to the relevant physical zone valve

3. Wet CH & HW, smart TRV in every room, no real outdoor sensor
   Heat source: same as example 1
   Zones: one per TRV/room
   Indoor sensors: TRV sensor devices
   Outdoor sensors: all from forecast API
   Valves: TRV valve devices