How to implement Node-red flow heatdemand processing within Home Assistant.

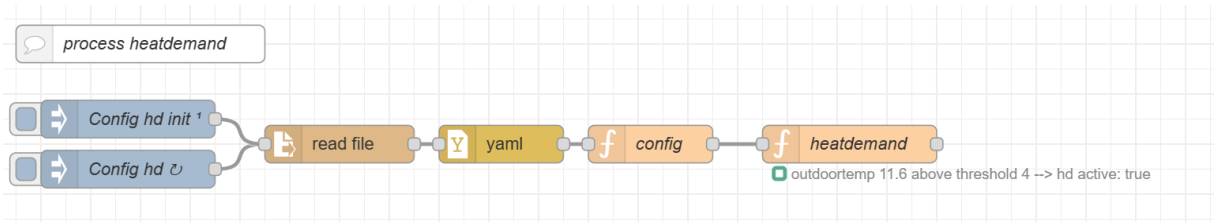
The heat demand flow is able to decide if a heat demand for a boiler / heatpump is there by reading thermostat entities (actual vs. settemp) and using parameters to switch a heating circuit on/off depending on heatdemand.

The following technical prerequisites are needed:

1. **Node-Red addon is installed and active.**
2. **MQTT Broker is installed and discovery prefix is set to standard "homeassistant"**
3. **additional "axios" module is configured within NR as additional npm package**
4. **a longterm api access token is generated in HA**

With these prerequisites the km200 data processing flow consists of:

1. The Node-Red flow for heat demand:



A configuration file hd.yaml has to exist in the config directory of HA.
The following entries within hd.yaml:

1. **server local ha api access**
2. **the longterm access token generated**
3. **outdoor temp entity**
4. **outdoortemp_threshold: hd active if outdoortemp is above threshold**

5. **thermostats per room with entity, settemp and actualtemp**
   deltam: defining minimum delta temp for heatdemand
   hc: heating circuit (hc1 to hc4)
   weight: weight of this thermostat

6. **heatingcircuits**
   hc: hc1 to hc4
   weigthon and weigthoff
   state: for mqtt write
   entity: entity within HA
   on /off: writing values for hc on/off (-1= auto ; 0 = off)
   savesettemp: saving previous settemp for floorheating when overwritten by 0 (off):
   true/false

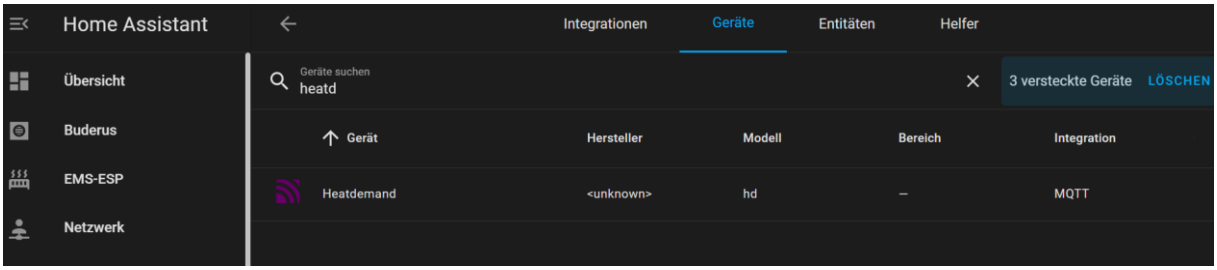
Example hd.yaml:

```yaml
- server: http://localhost:8123/api/
- token: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- outdoortemp_entity: sensor.boiler_outdoortemp
- outdoortemp_threshold: 4

- thermostats:

  - room: WZ
    entity: climate.wohnzimmer_thermostat
    settemp: temperature
    actualtemp: current_temperature
    deltam: 0.25
    hc: hc1
    weight: 3

  - room: WG
    entity: climate.wintergarten_thermostat
    settemp: temperature
    actualtemp: current_temperature
    deltam: 0.25
    hc: hc1
    weight: 3


- heatingcircuits:

  - hc: hc1
    weighton: 3
    weightoff: 2
    state: ems-esp/thermostat/hc1/tempautotemp
    entity: number.thermostat_hc1_tempautotemp
    on: -1
    off: 0
    savesettemp: false

  - hc: hc2
    weighton: 5
    weightoff: 0
    state: ems-esp/thermostat/hc2/tempautotemp
    entity: number.thermostat_hc2_tempautotemp
    on: -1
    off: 0
    savesettemp: true
```

Flow Logic:

Once on Start the heat demand entities are created by using mqtt discovery api calls.

These entities are grouped under the device “Heatdemand” within mqtt integration:



Please note that entities are not automatically deleted when you change names. This has to be done using mqtt explorer or a similar tool.

The heatdemand logic is described by:

For each thermostat actualtemp is compared to settemp. If (settemp-actualtemp) > deltam then there is a heatdemand for this thermostat / climatate entity. The demand is given by the weight.

All demands for all thermostats of one heating circuit (hc1 to hc2) is aggregated and compared to the parameters of the heating circuit.
If sum(weigths) >= weigthon then hc will be switched on using the on value.
Otherwise the hc will be switched off using the value for off.

For floorheating the change of settemp to off will overwrite the former settemp.
For floorheating savesettemp could be be then set to true.
Then the former settemp will be stored and used for comparison of temperatures.

NR Flows:

The following flow can be copied and imported to node-red:

```
[{"id":"d3dedac6e827f993","type":"inject","z":"c07aa589530de634","name":"Config
hd","props":[{"p":"payload"},{"p":"init","v":"false","vt":"bool"}],"repeat":"30","crontab":"","once":true,"onceDelay":"60","topic":"","payload":"/config/hd.yaml","payloadType":"s
tr","x":110,"y":580,"wires":[["069d079ea3a6eaf8"]]},{"id":"069d079ea3a6eaf8","type":"file
in","z":"c07aa589530de634","name":"","filename":"payload","filenameType":"msg","format":"utf8","chunk":false,"sendError":false,"encoding":"none","allProps":false,"x":280,"y"
:560,"wires":[["73ec84ee0202eefd"]]},{"id":"73ec84ee0202eefd","type":"yaml","z":"c07aa589530de634","property":"payload","name":"","x":410,"y":560,"wires":[["e824e1af320cc
be7"]]},{"id":"26e21a0a0c34ca9f","type":"function","z":"c07aa589530de634","name":"heatdemand ","func":"let axios = global.get(\"axios\");\nlet server_mqtt = msg.server +
\"services/mqtt/publish\";\nlet server = msg.server + \"states/\";\nlet bearer = \"Bearer \" + msg.token;\n\nif (msg.init) await init_controls();\nawait heatdemand();\n\nreturn
msg;\n\nasync function init_controls() {\n  try {\n\n    //control_switch(\"hd_active\", \"1\");\n\n    for (let i = 0; i < msg.heatingcircuits.length; i++) {\n      const state =
\"hd_\" + msg.heatingcircuits[i].hc + \"_\";\n      control_state(state + \"weighton\", parseFloat(msg.heatingcircuits[i].weighton));\n      control_state(state + \"weightoff\",
parseFloat(msg.heatingcircuits[i].weightoff));\n      control_state(state + \"weight\", 99);\n      control_state(state + \"state\", msg.heatingcircuits[i].state);\n
control_state(state + \"on\", msg.heatingcircuits[i].on);\n      control_state(state + \"off\", msg.heatingcircuits[i].off);\n      control_state(state + \"status\", \"hc control
status\");\n      if (msg.heatingcircuits[i].savesettemp) await control_state(state + \"savesettemp\", 0);\n    }\n\n    for (let i = 0; i < msg.thermostats.length; i++) {\n      const
state = \"hd_\" + msg.thermostats[i].room + \"_\";\n      let value = 0;\n      try {\n        const state1 = await getstate(msg.thermostats[i].entity);\n      } catch (e) {
mess(\"*** wrong entity: \" + msg.thermostats[i].entity); }\n      control_state(state + \"actualweight\", 0);\n      control_state(state + \"weight\",
parseFloat(msg.thermostats[i].weight));\n      control_state(state + \"deltam\", parseFloat(msg.thermostats[i].deltam));\n    }\n  } catch (e) { }\n}\n\n\nasync function
heatdemand() {\n  let w1 = 0, w2 = 0, w3 = 0, w4 = 0;\n\n  try { if (msg.thermostats.length == 0 || msg.thermostats.length == undefined) return; }\n  catch (e) { return; }\n\n  let
hd = false;\n\n  const outdoortemp = (await getstate(msg.temp_entity)).state;\n  const active = (await getstate(\"switch.hd_active\")).state;\n\n  let status = \"outdoortemp \" +
outdoortemp;\n\n  if (outdoortemp <= msg.temp_threshold) status += \" below threshold \" + msg.temp_threshold;\n  else status += \" above threshold \" +
msg.temp_threshold;\n\n  if (outdoortemp > msg.temp_threshold && (active == 1 || active == \"on\")) hd = true;\n  status += \" --> hd active: \" + hd;\n  node.status({ fill:
\"green\", shape: \"ring\", text: status });\n\n  for (let i = 0; i < msg.thermostats.length; i++) {\n    const state = \"hd_\" + msg.thermostats[i].room + \"_\";\n    let settemp = 0,
acttemp = 0, savetemp = 0;\n\n    const state1 = await getstate(msg.thermostats[i].entity);\n    settemp = parseFloat(state1.attributes[msg.thermostats[i].settemp]);\n
acttemp = parseFloat(state1.attributes[msg.thermostats[i].actualtemp]);\n\n    savetemp = 0;\n    for (let i1 = 0; i1 < msg.heatingcircuits.length; i1++) {\n      if
(msg.thermostats[i].hc == msg.heatingcircuits[i1].hc && msg.heatingcircuits[i1].savesettemp == true) {\n        savetemp = (await getstate(\"sensor.hd_\" + msg.thermostats[i].hc +
\"_savesettemp\")).state;\n        if (savetemp == undefined) {\n          savetemp = 0;\n          await set_state(\"sensor.hd_\" + msg.thermostats[i].hc + \"_savesettemp\",
0);\n        }\n        savetemp = parseFloat(savetemp);\n        if (savetemp > settemp) settemp = savetemp;\n        mess(\"settemp:\" + settemp + \"   savetemp:\" +
savetemp);\n      }\n\n    }\n  \n    const deltam = parseFloat(msg.thermostats[i].deltam);\n    const delta = settemp - acttemp;\n    const weight =
parseInt(msg.thermostats[i].weight);\n    if (msg.thermostats[i].hc == \"hc2\") mess(delta+ \"  \"+deltam);\n    if (delta > deltam) {\n      await set_state(state +
\"actualweight\", weight);\n      if (msg.thermostats[i].hc == \"hc1\") w1 += weight;\n      if (msg.thermostats[i].hc == \"hc2\") w2 += weight;\n      if (msg.thermostats[i].hc
== \"hc3\") w3 += weight;\n      if (msg.thermostats[i].hc == \"hc4\") w4 += weight;\n    }\n    else await set_state(state + \"actualweight\", 0);\n  }\n\n\n  for (let i = 0; i <
msg.heatingcircuits.length; i++) {\n    const hc = msg.heatingcircuits[i].hc;\n    const state = \"hd_\" + hc + \"_\";\n    let w = 99;\n    if (hc == \"hc1\") w = w1;\n    if (hc ==
\"hc2\") w = w2;\n    if (hc == \"hc3\") w = w3;\n    if (hc == \"hc4\") w = w4;\n\n    await set_state(state + \"weight\", w);\n    const von =
parseFloat(msg.heatingcircuits[i].on);\n    const voff = parseFloat(msg.heatingcircuits[i].off);\n    const vs = (await getstate(msg.heatingcircuits[i].entity)).state; // actual state\n\n
if (!hd && vs == voff) {\n      let data = { \"payload\": von, \"topic\": msg.heatingcircuits[i].state, \"retain\": \"True\" };\n      let response = await
postmqtt(JSON.stringify(data));\n    }\n    if (!hd ) await set_state(\"sensor.hd_\" + msg.heatingcircuits[i].hc + \"_savesettemp\", 0)\n\n    if (hd) {\n      //try {\n      if (w
>= msg.heatingcircuits[i].weighton && vs == voff) {\n        await set_state(state + \"status\", true);\n        mess(\"new heat demand for \" + hc + \" -->  on: \");\n        let
data = { \"payload\": von, \"topic\": msg.heatingcircuits[i].state, \"retain\": \"True\" };\n        let response = await postmqtt(JSON.stringify(data));\n      }\n\n      if (w <=
msg.heatingcircuits[i].weightoff && vs != voff) {\n        await set_state(state + \"status\", false);\n        mess(\"no heat demand anymore for \" + hc + \" --> off: \");\n\n
if (msg.heatingcircuits[i].savesettemp) {\n        for (let ii = 0; ii < msg.thermostats.length; ii++) {\n          if (msg.thermostats[ii].hc == hc) {\n            let settemp;\n
//try {\n              const state2 = await getstate(msg.thermostats[ii].entity);\n              settemp = state2.attributes[msg.thermostats[ii].settemp];\n              //} catch
(e) { settemp = -1; }\n              if (settemp != voff) await set_state(state + \"savesettemp\", settemp);\n          }\n        }\n      }\n      let data = {
\"payload\": voff, \"topic\": msg.heatingcircuits[i].state, \"retain\": \"True\" };\n      let response = await postmqtt(JSON.stringify(data));\n      }\n\n      //} catch (e) { }\n
}\n  }\n\n}\n\nasync function control_reset() {  // heat demand control switched off - reset control states for hc's\n  for (let i = 0; i < msg.heatingcircuits.length; i++) {\n    const hc
= msg.heatingcircuits[i].hc;\n    const on = parseInt(msg.heatingcircuits[i].on);\n    //adapter.setState(msg.heatingcircuits[i].state, { ack: false, val: on });\n
//adapter.setState(\"controls.\" + hc + \".status\", { ack: true, val: true });\n  }\n}\n\nfunction mess(text) {\n  msg.payload = text;\n  node.send(msg);\n}\n\n\n\n\nfunction
json(json, variableKeyName) {\n  const variableKeyValue = json[variableKeyName];\n  return variableKeyValue;\n}\n\n\nasync function getstate(state) {\n  //const urls = server +
\"states/\" + state;\n  const urls = server + state;\n  const options = { url: urls, method: \"GET\", headers: { \"Authorization\": bearer, \"content-type\": \"application/json\" } };\n
try { \n    let body = await axios(options); \n    //msg.payload = body.data; node.send(msg);\n    return body.data; \n  }\n  catch (e) { return 0; }\n}\n\n\nasync function
control_switch(state, value) {\n  let field = state;\n  let pl = {\n    \"~\": \"homeassistant/switch/hd/\" + field,\n    \"name\": state,\n    \"uniq_id\": state,\n    \"stat_t\":
\"~/state\",\n    \"cmd_t\": \"~/state\",\n    \"state_class\": \"measurement\",\n    \"step\": 1,\n    \"ic\": \"mdi:heat-wave\",\n    \"payload_off\": \"0\",\n
\"payload_on\": \"1\",\n    \"object_id\": state,\n    \"dev\": {\n      name: \"Heatdemand\",\n      \"mdl\": \"hd\",\n      \"ids\": [\"hd\"]\n    }\n  };\n\n  let topic =
\"homeassistant/switch/hd/\" + state + \"/config\";\n  let data = { \"payload\": JSON.stringify(pl), \"topic\": topic, \"retain\": \"True\" };\n  await
postmqtt(JSON.stringify(data));\n\n  topic = \"homeassistant/switch/hd/\" + state + \"/state\";\n  data = { \"payload\": value, \"topic\": topic, \"retain\": \"True\" };\n  await
postmqtt(JSON.stringify(data));\n}\n\n\nasync function control_state(state, value) {\n  state = state.toLowerCase();\n  let field = state;\n  let pl = {\n    \"~\":
\"homeassistant/sensor/hd/\" + field,\n    \"name\": state,\n    \"uniq_id\": state,\n    \"stat_t\": \"~/state\",\n    \"cmd_t\": \"~/state\",\n    \"state_class\":
\"measurement\",\n    \"object_id\": state,\n    \"dev\": {\n      name: \"Heatdemand\",\n      \"mdl\": \"hd\",\n      \"ids\": [\"hd\"]\n    }\n  };\n  let topic =
\"homeassistant/sensor/hd/\" + state + \"/config\";\n  let data = { \"payload\": JSON.stringify(pl), \"topic\": topic, \"retain\": \"True\" };\n  await
postmqtt(JSON.stringify(data));\n\n  topic = \"homeassistant/sensor/hd/\" + state + \"/state\";\n  data = { \"payload\": value, \"topic\": topic, \"retain\": \"True\" };\n  await
postmqtt(JSON.stringify(data));\n}\n\n\nasync function set_state(state, value) {\n  state = state.toLowerCase();\n  let field = state;\n  let topic = \"homeassistant/sensor/hd/\" +
state + \"/state\";\n  let data = { \"payload\": value, \"topic\": topic, \"retain\": \"True\" };\n  await postmqtt(JSON.stringify(data));\n}\n\n\nasync function postmqtt(data) {\n\n
\n  const urls = server_mqtt;\n\n  let response = await axios({\n    method: 'post',\n    url: urls,\n    data: data,\n    headers: {\n      \"Authorization\": bearer,\n
\"content-type\": \"application/json\",\n    }\n  });\n  return response;\n  \n/*\n  msg.payload = data.payload;\n  msg.topic = data.topic;\n  node.send(msg);\n
return;\n*/\n}","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":690,"y":560,"wires":[[]]},{"id":"e824e1af320ccbe7","type":"function","z":"c07aa589530de634","name"
:"config","func":"for (let i = 0; i< msg.payload.length;i++ ) {\n  let m = msg.payload[i];\n  if (m.server != undefined) msg.server = m.server;\n  if (m.token !=
undefined) msg.token =  m.token;\n  if (m.outdoortemp_entity != undefined) msg.temp_entity = m.outdoortemp_entity;\n  if (m.outdoortemp_threshold != undefined)
msg.temp_threshold = m.outdoortemp_threshold;\n  if (m.thermostats != undefined) msg.thermostats = m.thermostats;\n  if (m.heatingcircuits != undefined)
msg.heatingcircuits = m.heatingcircuits;\n}\n\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":530,"y":560,"wires":[["26e21a0a0c34ca9f"]]},{"id":"3536cf97804ec8d3","type":"comment","z":"c07aa589530de6
34","name":"process heatdemand","info":"","x":120,"y":480,"wires":[]},{"id":"9f309d3b9e467457","type":"inject","z":"c07aa589530de634","name":"Config hd
init","props":[{"p":"payload"},{"p":"init","v":"true","vt":"bool"}],"repeat":"","crontab":"","once":true,"onceDelay":"10","topic":"","payload":"/config/hd.yaml","payloadType":"str
","x":110,"y":540,"wires":[["069d079ea3a6eaf8"]]}]
```