🔁Radiator Automation Setup — n8n Integration Guide

> This document explains every connection, node, and data flow between Home Assistant, the AI microservice, and the Telegram notification loop.
> The goal is to let the system **learn automatically** and **send Telegram instructions** whenever radiator settings should change.

🌐Overview

Cycle frequency: every 10 minutes

```
┌───────────────┐    ┌───────────────┐    ┌───────────────┐
┌───────────────────┐
│ Home Assistant│ → │  n8n Flow     │ → │  AI Service  │ → │ Telegram
Webhook │
└───────────────┘    └───────────────┘    └───────────────┘
└───────────────────┘
```

Core goals 1. Collect real-time temperatures & targets from Home Assistant.
2. Collect manual radiator levels from the Telegram Bot database.
3. Fetch current and forecast outdoor temperatures.
4. Train and query the AI model.
5. Let the AI decide and send Telegram notifications automatically.

🧰Prerequisites

Component	Requirement
n8n	Running locally (`http://localhost:5678`)
Home Assistant	REST API access and long-lived access token
AI Service	Reachable at `http://ai_service:8000` (Docker)
Telegram Bot	Already configured and online
SQLite DB	`bot/radiators.db` shared or exposed for n8n read access
Weather	Open-Meteo (no API key) or SMHI API

🉐1. Create the Workflow Skeleton

1. **New Workflow → “Radiator Training and Prediction”**

2. Add **Cron Trigger**
3. **Mode:** Every X minutes
4. **Value:** 10
5. **Time Zone:** Europe/Stockholm

5️⃣2. Collect Sensor Data from Home Assistant

Node Type: HTTP Request
Name: `Fetch Room Temps`

- **Method:** GET
- **URL:** `http://homeassistant.local:8123/api/states`
- **Headers:**

```
Authorization: Bearer <YOUR_HA_TOKEN>
Content-Type: application/json
```

- **Output:** JSON

Example filters (in *Set* node right after this one):

Room	Temperature Sensor	Target Helper
Bedroom	`sensor.bedroom_temp`	`input_number.bedroom_target`
Living Room	`sensor.livingroom_temp`	`input_number.livingroom_target`
Office	`sensor.office_temp`	`input_number.office_target`
Bathroom	`sensor.bathroom_temp`	`input_number.bathroom_target`
Hallway	`sensor.hallway_temp`	`input_number.hallway_target`

Extract each sensor's `.state` value and store them in workflow variables:

```
$json["bedroom_temp"] = items.filter(i => i.entity_id=="sensor.bedroom_temp")
[0].state
...
```

🔢3. Get Current Radiator Levels

If you can mount the bot's `radiators.db`: 1. Add an **Execute Command** node.
2. Command:

```
sqlite3 /workspace/bot/radiators.db "SELECT room, level FROM radiators;"
```

3. Use **Split Out** option to parse results to JSON:

```
const lines = $input.item.json.stdout.split('\n');
return lines.filter(Boolean).map(l=>{
  const [room,level]=l.split('|');
  return {room,level:Number(level)};
});
```

Alternatively, expose an HTTP endpoint in the bot later and call that.

🔴4. Fetch Outdoor Temp and Forecast

Node Type: HTTP Request
Name: `Fetch Forecast`

- **Method:** GET
- **URL:**

  ```
  https://api.open-meteo.com/v1/forecast?
  latitude=57.1&longitude=12.25&hourly=temperature_2m&forecast_days=1
  ```

- **Output:** JSON

Use **Function** node to extract:

```
const temps = $json.hourly.temperature_2m;
return {
  outdoor_temp: temps[0],
  forecast_temp: temps[3] || temps[0]
};
```

---

5. Prepare Training Payloads

Add a **Merge** node combining: - HA data - Radiator DB data - Forecast data

Then a **Function** node “Build Training Payloads”:

```
const rooms = ["Bedroom","Living Room","Office","Bathroom","Hallway"];
const forecast = $json.forecast_temp;
const outside = $json.outdoor_temp;

return rooms.map(room => ({
  json: {
    room,
```

```
    current_temp: parseFloat($json[room.toLowerCase().replace(' ','_')
+'_temp']),
    target_temp: parseFloat($json[room.toLowerCase().replace(' ','_')
+'_target']),
    radiator_level: $json[room+"_level"],
    outdoor_temp: outside,
    forecast_temp: forecast,
    timestamp: new Date().toISOString()
  }
}));
```

---

6. Train the AI

Node Type: HTTP Request
Name: `Train AI`

- **Method:** POST
- **URL:** `http://ai_service:8000/train`
- **Body Content Type:** JSON → Use Input Data

This sends one request per room with current state.

🆚7. Predict and Let AI Decide

Node Type: HTTP Request
Name: `Predict AI`

- **Method:** POST
- **URL:** `http://ai_service:8000/predict`
- **Body:** same payload as /train

AI Service behavior - Calculates predicted next temperature for each possible radiator level. - Chooses level with minimum error to target. - If recommendation differs ≥ 1 level → sends Telegram message via `TELEGRAM_WEBHOOK`.

No extra n8n node is needed for notifications — the AI handles it internally.

🔄8. Optional – Log Results to Google Sheets or InfluxDB

Add nodes if you want to store all predictions for later visualization in Grafana/Home Assistant.

Columns:

```
timestamp, room, current_temp, target_temp, outdoor_temp, forecast_temp,
current_level, recommended_level, delta_predicted
```

⛎9. Scheduling & Error Handling

- Add a **Wait + Retry** loop if HA or forecast API fails.
- Add a **No Change** gate to skip unnecessary /train calls when no temp delta since last reading.
- Enable workflow **Active = true** in n8n sidebar.

---

10. Security & Permissions

Component	Credential	Note
HA API	Long-lived token	read only
AI Service	internal network only	port 8000
Telegram Webhook	stored in `.env`	not public
SQLite DB	read-only mount	0600 permissions

🛅11. Home Assistant Entity Naming (Recommended)

Room	Temp Sensor	Target Helper	Optional Entity
Bedroom	`sensor.bedroom_temperature`	`input_number.bedroom_target`	`input_boolean.be`
Living Room	`sensor.livingroom_temperature`	`input_number.livingroom_target`	—
Office	`sensor.office_temperature`	`input_number.office_target`	—
Bathroom	`sensor.bathroom_temperature`	`input_number.bathroom_target`	—
Hallway	`sensor.hallway_temperature`	`input_number.hallway_target`	—

🛑12. Workflow Test Procedure

1. Run the **AI Service** and **Bot** via `docker-compose up -d`.
2. Open n8n → run workflow manually once.
3. Check AI service logs → should show `trained=True`.
4. Adjust a radiator level in Telegram → new training sample arrives.
5. After a few hours of data, AI starts sending “Set radiator to X” messages.

🟡13. Expected Telegram Messages

Example 1

```
🏠 Bedroom: set radiator to 5
   now 19.2 °C → target 20 °C
🏍️ outside 3 °C, forecast 5 °C
```

Example 2

```
🏠 Living Room: maintain level 4
   stable at 21.1 °C / target 21 °C
```

🈂️14. Long-Term Data Improvements

- Log each `/train` sample in InfluxDB for graphing.
- Recalculate per-room feature importances weekly.
- Optionally export models nightly for backup.

✅Summary

Step	Description	Responsible Node
1	Schedule every 10 min	Cron Trigger
2	Fetch HA temps/targets	HTTP Request
3	Fetch radiator levels	Execute Command / SQLite
4	Fetch forecast	HTTP Request
5	Assemble payload	Function node
6	Train model	HTTP Request (train)
7	Predict adjustment	HTTP Request (predict)
8	AI sends Telegram message	AI Service
9	(Opt.) Store logs	Sheets / Influx nodes

With this workflow running, your Home Assistant data, manual radiator feedback, and outdoor forecast continuously train the AI model.
Every few hours the system becomes smarter — adapting to weather, time of day, and room dynamics — and will automatically message you only when a radiator needs adjusting.