Automatic Greenhouse Ventilation

Author: Alexander Staub (23553)

1. Introduction

This project implements an automated greenhouse ventilation system using Home Assistant. The project is personal: my grandparents have a `Schrebergarten` with a greenhouse where they grow vegetables such as cucumbers and tomatoes. During warm summer periods, the greenhouse climate can drift away from crop-friendly conditions, which makes ventilation a recurring manual task that is often needed every evening.

Greenhouses are designed to provide plants with more optimally controlled microclimate conditions. However, if temperature and humidity are not managed properly, crop quality and yield can be negatively affected, which is why greenhouse automation and control focus strongly on microclimate parameters such as air temperature and relative humidity. [1]

Because manual ventilation is repetitive and not always possible on time, an automation approach can help keep conditions closer to target ranges and reduce day-to-day workload.

1.2 Project goal

The main goal is to automate the ventilation process so that the greenhouse can be ventilated reliably based on sensor readings and control rules. The intended outcome is a system that:

- monitors temperature and humidity,
- triggers ventilation when needed (especially during warm periods),
- reduces the need for daily manual intervention,
- provides a practical stepping stone for later upgrades and extensions.

1.3 Target climate ranges (definition of success)

- Target air temperature: 15–27 °C
- Target relative humidity: 50%-70% a moderate range that supports growth while reducing the risk of condensation, fungal disease and mold

These ranges are used as engineering targets for automation rules and later evaluation in testing. They are not strict biological limits, but a sensible operating window for typical warm-season greenhouse crops such as tomatoes and cucumbers, where excessive heat and persistently high humidity increase disease pressure. [1][2]

1.4 Constraints and assumptions

- Greenhouse volume: ~15 m^3
- No active heating/cooling: the current setup focuses on ventilation only (no active heater/

cooler installed)
- Offline operation: must work without an internet connection
- Home Assistant: monitoring and automation are implemented using Home Assistant

2. System Overview

2.1 High-level architecture

The core system consists of two temperature/humidity sensors and one switching actuator:

- Sensors: two DHT11 sensors (temperature + relative humidity)
  - 1× inside the greenhouse
  - 1× outside the greenhouse
- Actuator: ventilation fans connected to a relay module
- Controller: an ESP32 running ESPHome
- Automation platform: Home Assistant receives the sensor readings and controls the relay based on defined conditions.

Data flow is: `DHT11 (inside/outside) → ESP32 (ESPHome) → Home Assistant → ESP32 (ESPHome) → Relay → Fans`.

2.2 Control concept

Because the current scope uses fans only (no heating/dehumidifier), the system can influence the greenhouse climate mainly by exchanging inside air with outside air. The automation therefore compares inside conditions against outside conditions and only ventilates when outside air is expected to improve the inside climate.

Although the DHT11 sensors only measure air temperature and relative humidity, these values can be used to calculate the dew point. Dew point is helpful because it is a direct indicator of how close the air is to saturation and therefore to condensation on cold surfaces (e.g., glass), which can promote fungal diseases. In addition, dew point reflects the air's moisture state more robustly than relative humidity alone, which is temperature-dependent and can appear "high" even when the absolute moisture content is not. This makes dew point a useful metric when deciding whether exchanging inside air with outside air is likely to reduce condensation risk and improve the greenhouse climate.

$$\gamma(T, RH) = \frac{aT}{b+T} + \ln\left(\frac{RH}{100}\right)$$

$$T_d = \frac{b\gamma(T, RH)}{a - \gamma(T, RH)}$$

$$a = 17.62, \quad b = 243.12$$

Fig. 1

Definitions:

- T_in, T_out = inside/outside temperature
- RH_in, RH_out = inside/outside relative humidity
- DP_in, DP_out = inside/outside dew point (calculated)

1. Extreme temperature control (highest priority, ignores humidity): Goal: prevent heat stress fast.

Start fans if:

- T_in ≥ 29 °C AND T_out < T_in − 0.5%

No humidity/dew point checks here. This is “save the plants” mode.

2. Normal temperature control (humidity-safe cooling) Goal: cool only when it won’t significantly worsen moisture/mold risk.

Start fans if:

- 27 °C < T_in < 29 °C
- AND T_out < T_in − 0.5%
- AND DP_out ≤ DP_in + 1%

This allows cooling even if outside dew point is slightly higher, but blocks cases where outside air is much wetter.

3. Humidity-driven ventilation (dehumidification, can run even if outside is warmer) Goal: keep RH_in < 70% and reduce mold pressure.

Start fans if:

- RH_in > 70%
- AND T_out ≤ 27 °C
- AND outside air is truly drier: DP_out < DP_in − 0.7%

If outside dew point is lower, exchanging air will reduce moisture even if outside is warmer.

If one of these three conditions is true for 1 minute the fans start, if they are false fo 2 minutes the fans turn off.

3. Components

3.1 Sensors

The system uses two identical DHT11 temperature and relative humidity sensors (one inside, one outside). A DHT11 module variant was selected (instead of the bare sensor) because it includes the necessary supporting components (e.g., pull-up resistor) and is therefore easy to connect in a plug-and-play manner to the ESP32. [C1]

3.2 Actuators

3.2.1 Fans

Ventilation is implemented with four 12 V / 92 mm brushless DC axial fans (dual ball bearing) to exchange air between the greenhouse and the outside environment. The selected fan type is specified with an airflow of 54.8 CFM per fan (manufacturer specification). [C2]

Based on the greenhouse volume of 15 m³ and the theoretical total airflow:

- Total airflow (theoretical): 4 × 54.8 CFM = 219.2 CFM
- 219.2 CFM ≈ 372.6 m³/h
- Air exchange time (theoretical): 15 m³ / 372.6 m³/h ≈ 0.0403 h ≈ 2.4 minutes per air change

This calculation is an idealized estimate. Real-world air exchange will typically be lower due to pressure losses, airflow short-circuiting (intake to exhaust), insect mesh, bends, and imperfect sealing.

3.2.2 Relay module

Fan power switching is done via a 4-channel 5 V relay module. Only two relay channels are used in this project: one for the intake fan group and one for the exhaust fan group. [C3]

3.3 Compute & networking

3.3.1 Home Assistant host

Home Assistant runs on Home Assistant Green, which is designed as a dedicated, low-power Home Assistant hub with preinstalled Home Assistant OS. [C4]

3.3.2 Microcontroller

Sensor readings and relay control are handled by an Olimex ESP32-POE-ISO running ESPHome. This board integrates Ethernet + PoE support and provides galvanic isolation on the PoE side (useful for robustness in longer cable runs and electrically “noisy” environments). [C5]

3.3.3 Local network

To ensure operation without an internet connection, a small local WLAN + DHCP setup is used:

- TP-Link TL-WR840N (N300) as local Wi-Fi router for Home Assistant UI access and DHCP server to enable communication between the Home Assistant Green and the ESP32. [C6]
- TP-Link TL-POE160S PoE+ injector to deliver both network and power over a single Ethernet cable to the greenhouse side. [C7]

- REVODATA Gigabit PoE splitter (12 V / 2 A) to separate data and provide 12V DC in the greenhouse. [C8]

3.4 Power conversion

Inside the greenhouse-side electronics, a DC-DC buck converter is used to derive 5V for the ESP32/relay electronics from the available DC supply. [C9]

4. Implementation

4.1 Physical setup & wiring

The physical implementation is split into two levels of detail: a top-down overview of the garden layout and a detailed networking, wiring and placement description.

4.1.1 Top-down system overview

In the garden house, the central infrastructure components are installed:

- Home Assistant Green
- Local Wi-Fi router (DHCP + UI access)
- PoE injector

From this location, a single Ethernet cable is routed to the greenhouse. This cable provides the network connection and the power delivery (via PoE) required for the greenhouse-side electronics:

- 4x 12V Fans
- 12V → 5V Buck Converter
- ESP32
- Relay
- 2x DHT11 Sensors

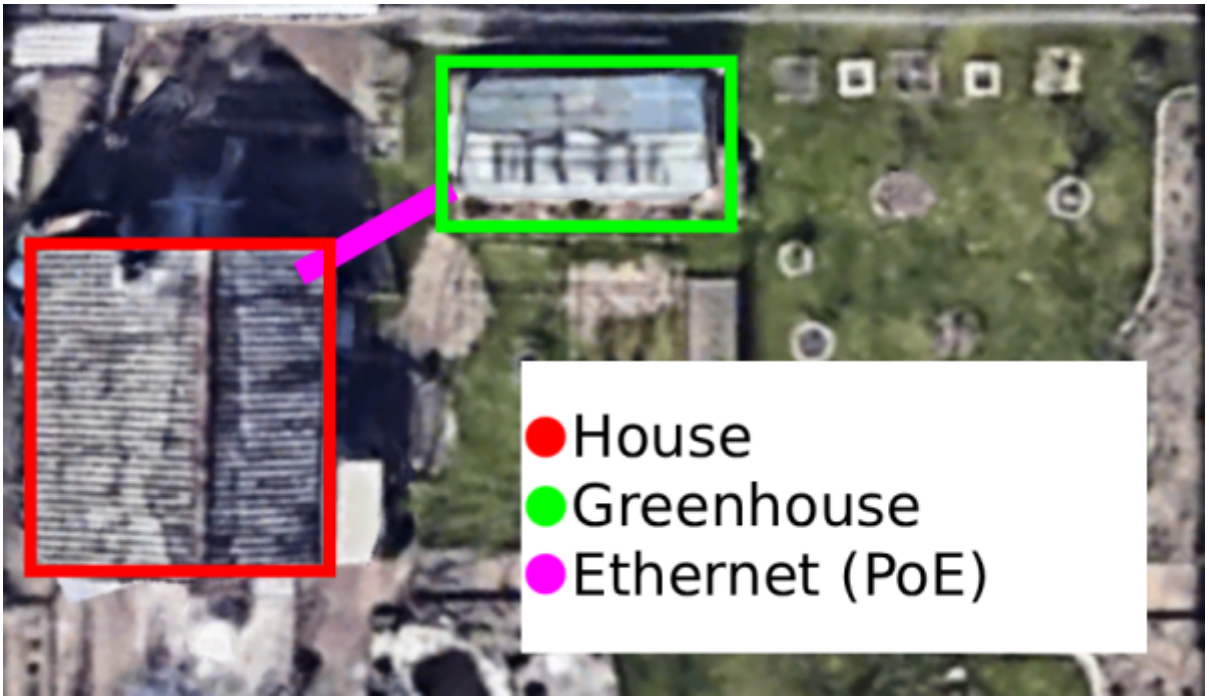

Fig. 2

4.1.2 Network and power wiring

Inside the garden house:

- Router
  - Home Assistant Green (LAN connection)
  - PoE injector (LAN input)

Ethernet cable from the PoE injector to the PoE Splitter in the greenhouse. The PoE splitter separates network and power:

- Data (Ethernet)
  - ESP32 (LAN connection)
- Power (12V DC, 2A)
  - Fan power circuit (12V; seperated in intake and exhaust groups using two relay channels)
  - 12V → 5V buck converter
    - ESP32 (5V input)
      - ESP32 (3.3V output)
        - DHT11 Sensors
    - Relay controller

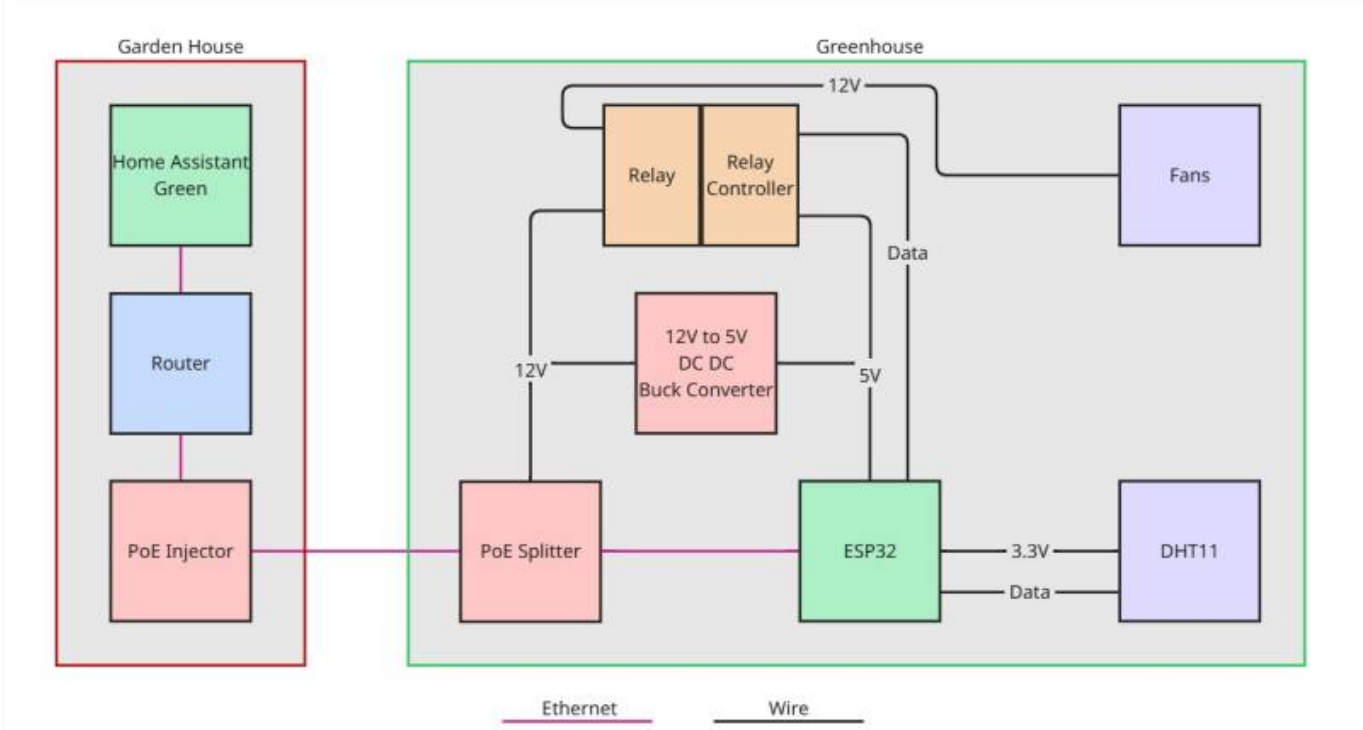

Fig. 3

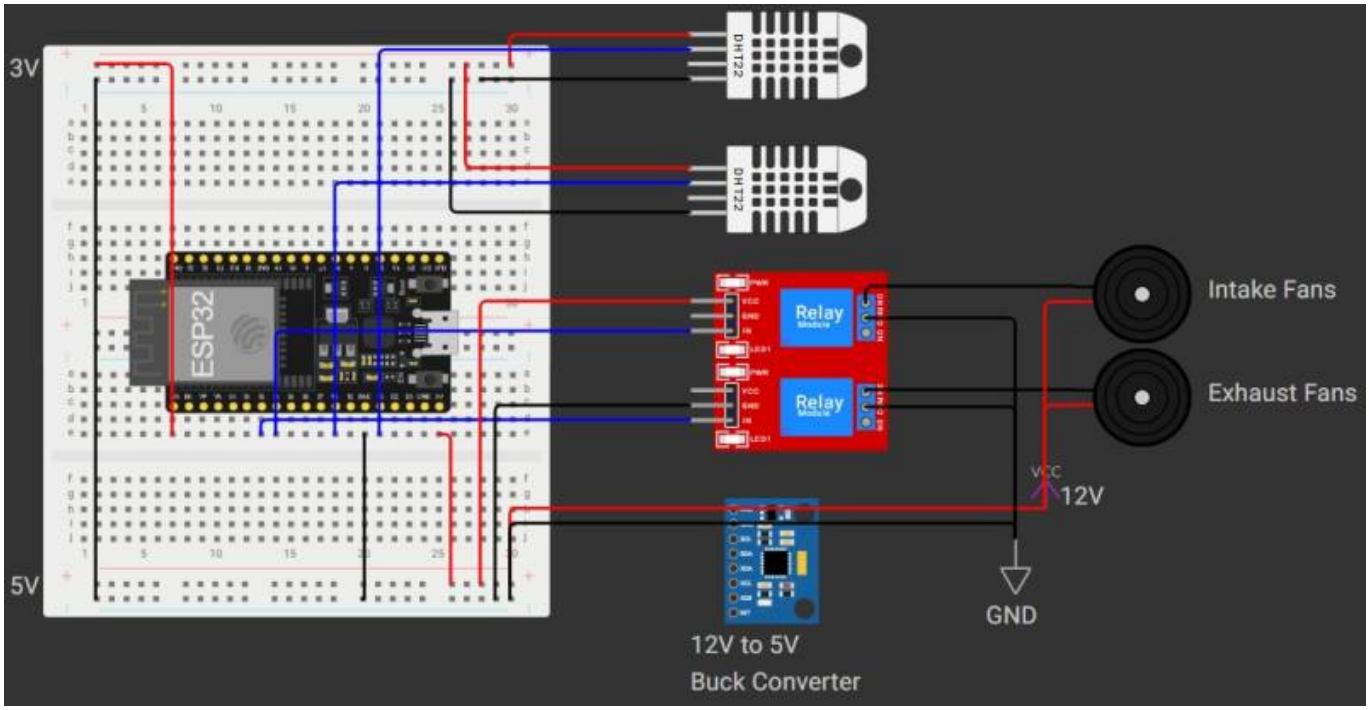

Fig. 4

4.1.3 Power budget

The greenhouse-side system is poweder via the PoE Splitter with a rated budget of 12V 2A. Measured current per fan:

- 0.31 A with open airflow (normal / low restriction)
- 0.35 A with bad airflow (higher restriction)
- 0.46 A startup peak

With 4 fans total, this results in:

- Normal airflow: 4 × 0.31 A = 1.24 A
- Bad airflow: 4 × 0.35 A = 1.40 A
- Startup peak: 4 × 0.46 A = 1.84 A

Measured 5 V rail current (ESP32 + relay + DHT sensors):

- 0.10 A idle
- 0.25 A under load

Converted to an equivalent 12 V input current assuming 80% buck converter efficiency:

- Idle: (5 V × 0.10 A) / (12 V × 0.8) = 0.052 A
- Load: (5 V × 0.25 A) / (12 V × 0.8) = 0.130 A

Worst-case steady-state (bad airflow + electronics under load):

- Total ≈ 1.40 A + 0.13 A = 1.53 A
- Margin to 2.00 A ≈ 0.47 A (about 24%)

Worst-case startup (all fans starting at once + electronics under load):

- Total ≈ 1.84 A + 0.13 A = 1.97 A
- Margin to 2.00 A ≈ 0.03 A (about 1.5%)

Because starting all four fans simultaneously would bring the total current draw very close to the splitter’s limit, a staggered startup procedure is used so that the fans do not reach their peak inrush current at the same time.

4.2 Enclosure

All greenhouse-side electronics (PoE splitter, ESP32, buck converter, breadboard/power distribution, and relay module) are installed in a dedicated enclosure.

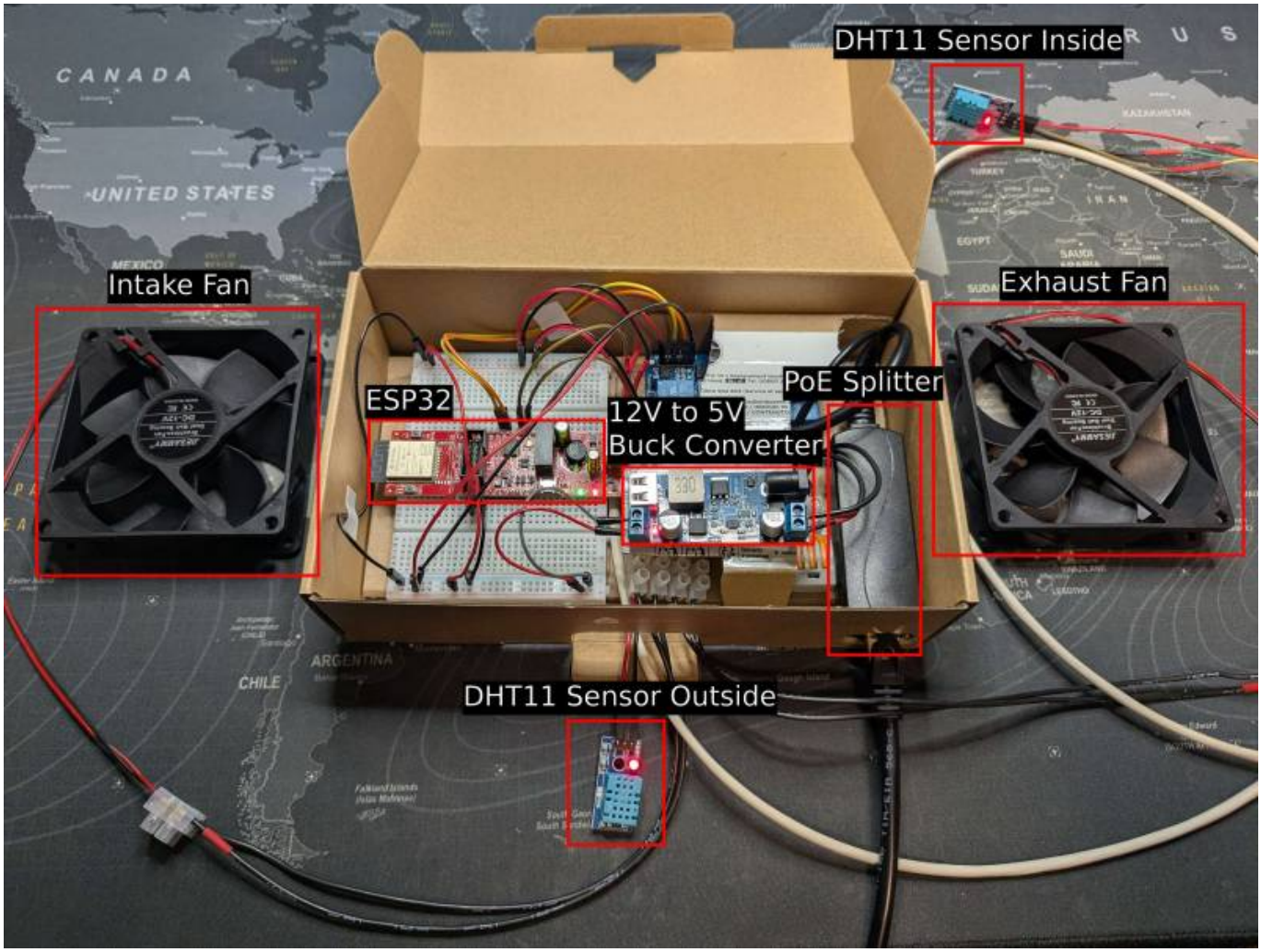
DHT11 Sensor Inside
Intake Fan
Exhaust Fan
ESP32
PoE Splitter
12V to 5V
Buck Converter
DHT11 Sensor Outside
CANADA
UNITED STATES
MEXICO
CHILE
ARGENTINA

Fig. 5

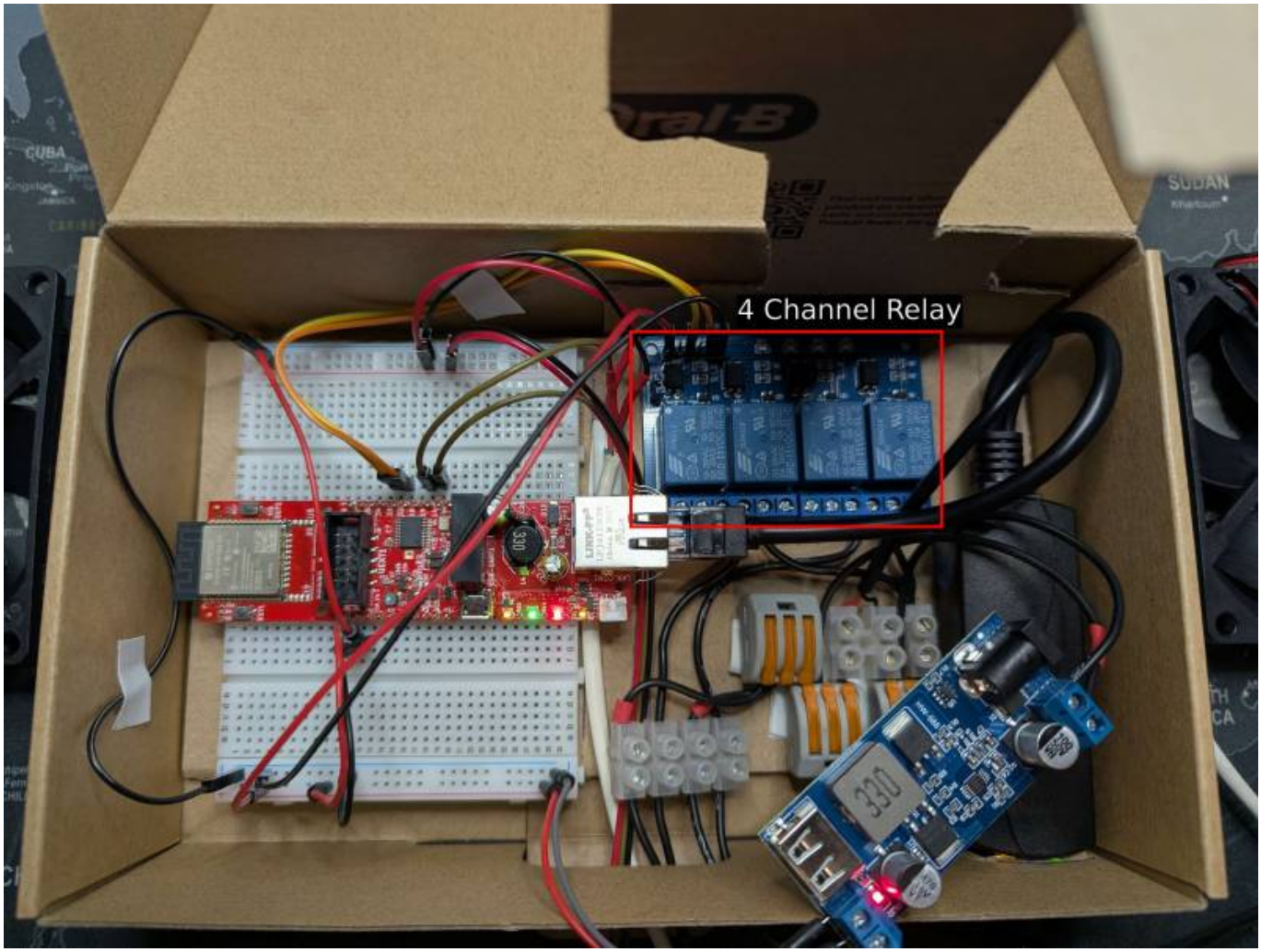

Fig. 6

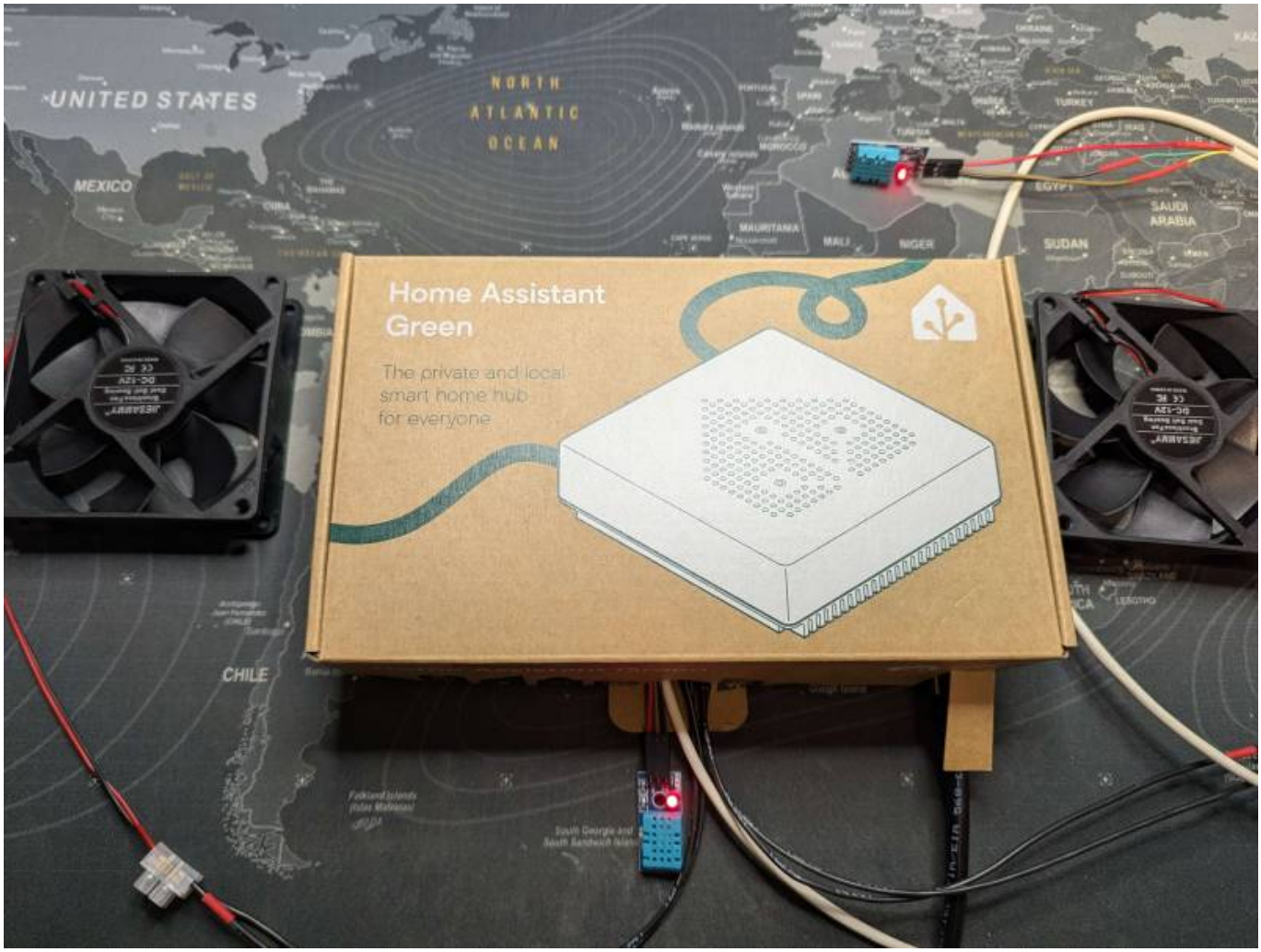

Fig. 7

4.3 Enclosure, Sensor and Fan placement

The enclosure is mounted on the inside of the greenhouse door, which keeps cable runs short to the door-mounted fans and simplifies installation and maintenance.

Sensor placement is designed to capture both the internal climate and the incoming outside conditions:

- Outside sensor
  - Mounted outside, towards the bottom area of the greenhouse (near air intake level)
- Inside sensor
  - Mounted inside, in the middle-to-upper part of the greenhouse

Ventilation placement follows a classic “bottom intake / top exhaust” strategy:

- Intake fans
  - Installed at the bottom of the door (draw in cooler outside air)
- Exhaust fans
  - Installed near the top (remove warm, humid air that accumulates above)

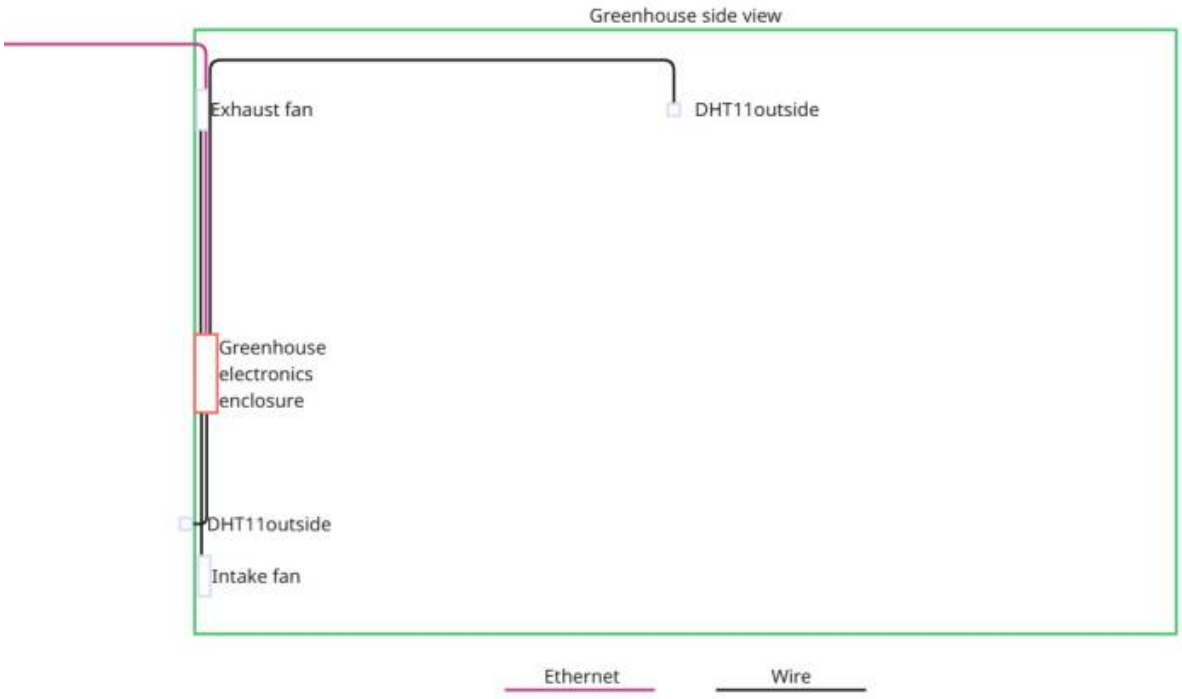

Fig. 8

The electronics enclosure is installed next to the greenhouse door (rather than on the door itself) and remains stationary. The Ethernet feed and the DHT11 sensors are therefore routed to this fixed mounting point, while only the fan wiring requires sufficient slack to accommodate the door's movement without putting strain on the connectors or cables.

4.4 Software setup

For the initial setup and for installing updates, a temporary internet connection is required. After installation, the system is designed to run locally without requiring internet access for normal operation.

4.4.1 Home Assistant version

The project was implemented with the following Home Assistant versions:

- Home Assistant Core: 2026.2.3
- Home Assistant Supervisor: 2026.02.2

4.4.2 Installing ESPHome (ESPHome Device Builder)

ESPHome was installed following the official guide [S1], alternatively by installing the add-on directly in Home Assistant:

- Home Assistant
    - Settings → Apps → Install App
        - ESPHome Device Builder

To flash the ESP32 with ESPHome for the first time, the ESP32 is connected via USB to the Home Assistant Green device. In ESPHome Device Builder a new device configuration is created and then installed:

- ESPHome Device Builder
  - New Device
    - Continue
      - Empty Configuration
        - Name device
          - EDIT (add yaml config)
            - SAVE
              - Install

The initial installation step can require internet access because ESPHome may download required build packages and platform dependencies.

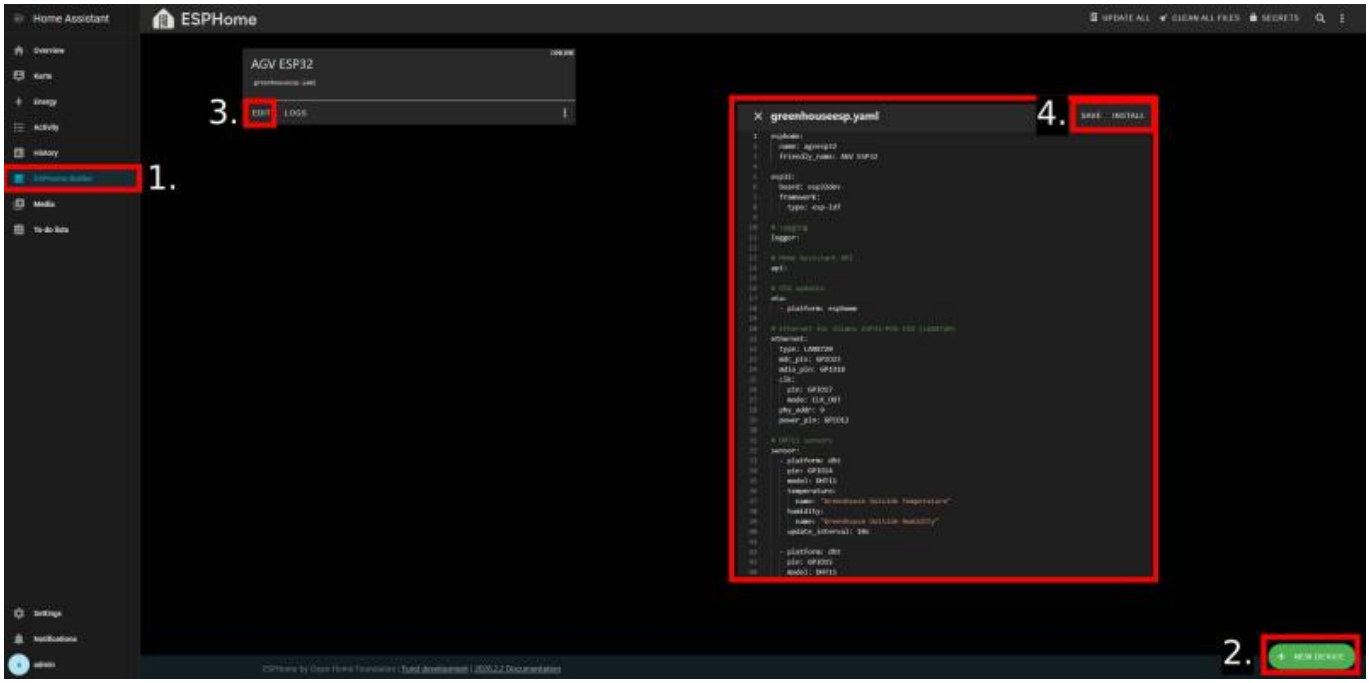

Fig. 9

4.4.3 ESPHome configuration (YAML)

The following ESPHome configuration was used to integrate the Ethernet-enabled ESP32 controller, two DHT11 sensors, and the relay outputs.

greenhouseesp.yaml

```yaml
esphome:
  name: agvesp32
  friendly_name: AGV ESP32

esp32:
  board: esp32dev
  framework:
    type: esp-idf
```

```
# Logging
logger:

# Home Assistant API
api:

# OTA updates
ota:
  - platform: esphome

# Ethernet for Olimex ESP32-POE-ISO (LAN8720)
ethernet:
  type: LAN8720
  mdc_pin: GPIO23
  mdio_pin: GPIO18
  clk:
    pin: GPIO17
    mode: CLK_OUT
  phy_addr: 0
  power_pin: GPIO12

# DHT11 sensors
sensor:
  - platform: dht
    pin: GPIO14
    model: DHT11
    temperature:
      name: "Greenhouse Outside Temperature"
    humidity:
      name: "Greenhouse Outside Humidity"
    update_interval: 10s

  - platform: dht
    pin: GPIO15
    model: DHT11
    temperature:
      name: "Greenhouse Inside Temperature"
    humidity:
      name: "Greenhouse Inside Humidity"
    update_interval: 10s

# Relay outputs
switch:
  - platform: gpio
    pin: GPIO32
    inverted: true
    id: relay_1
    name: "Greenhouse Relay Intake"
    restore_mode: RESTORE_DEFAULT_OFF
```

```
  - platform: gpio
    pin: GPIO33
    inverted: true
    id: relay_2
    name: "Greenhouse Relay Exhaust"
    restore_mode: RESTORE_DEFAULT_OFF

  - platform: template
    name: "Greenhouse Ventilation"
    id: ventilation_sequence
    lambda: |-
      return id(relay_1).state && id(relay_2).state;
    turn_on_action:
      - switch.turn_on: relay_1
      - delay: 1s
      - switch.turn_on: relay_2
    turn_off_action:
      - switch.turn_off: relay_2
      - switch.turn_off: relay_1
```

ESPHome provides many ready-to-use components which can be configured declaratively and are documented on the ESPHome website. [S2]

- logger
  - Enables runtime logs for debugging and monitoring during development.
- api
  - Connects the device to Home Assistant using the native ESPHome API.
- ota
  - Enables over-the-air firmware updates after the initial USB flash.
- ethernet (LAN8720)
  - Configures the Olimex ESP32-POE-ISO Ethernet interface for reliable wired connectivity.
- sensor: platform dht (model DHT11)
  - Reads temperature and relative humidity from each DHT11 sensor at a defined update interval.
- switch: platform gpio
  - Controls relay channels through ESP32 GPIO pins (inverted logic is used to match the relay board behavior).
- switch: platform template
  - Provides a single logical “ventilation” switch that turns intake and exhaust on/off in a controlled sequence (intake first, then exhaust after a short delay).

4.5 Home Assistant entities, dashboard, and automations

In the Home Assistant overview, compatible devices can be discovered by scanning the local network. If the ESPHome firmware (YAML configuration) was installed successfully on the ESP32, a device with the configured name AGV ESP32 should appear and can be added to Home Assistant.

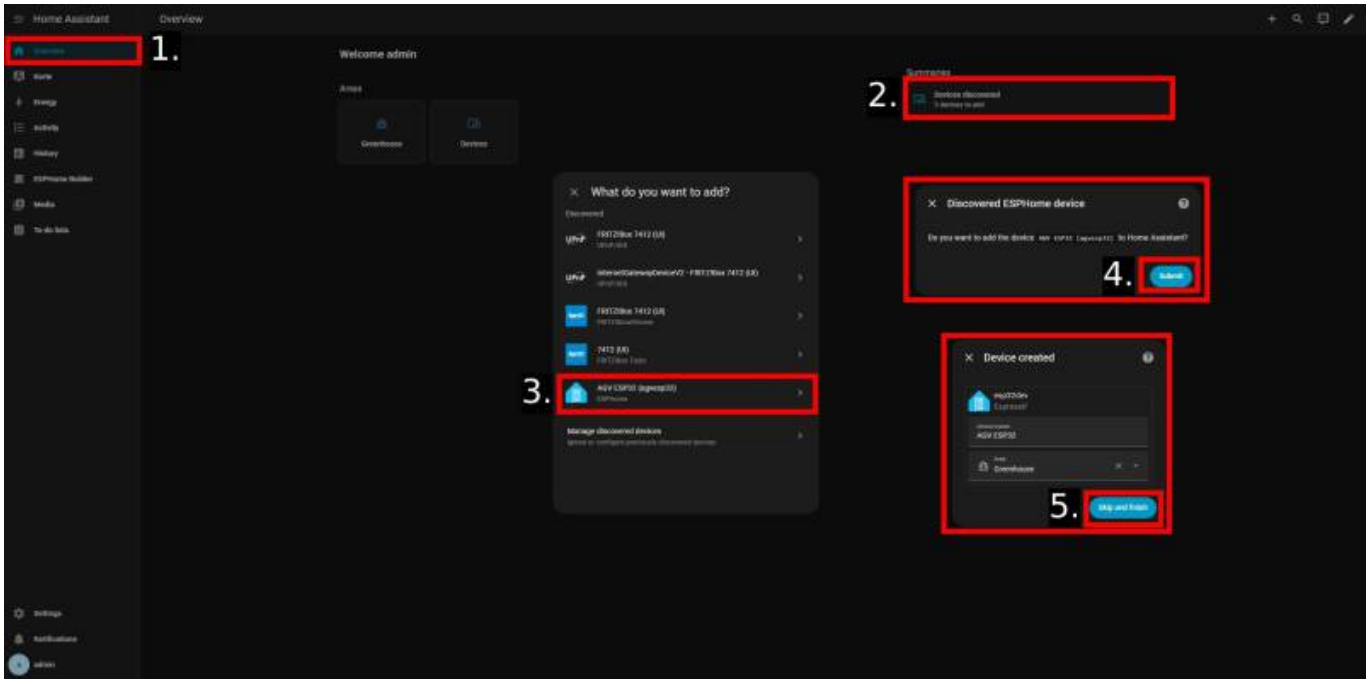

Fig. 10

4.5.1 Entities and dashboard

After pairing the ESPHome device with Home Assistant, the following entities become available (example naming as configured in ESPHome):

- Greenhouse Outside Temperature
- Greenhouse Outside Humidity
- Greenhouse Inside Temperature
- Greenhouse Inside Humidity
- Greenhouse Relay Intake
- Greenhouse Relay Exhaust
- Greenhouse Ventilation

These entities are visualized on a dedicated dashboard to monitor current conditions and to manually override ventilation if required.

To add automatic calculations of the dew points for inside and outside Templates are used. [S3]

Following the quick link in the Template Documentation or under `Settings → Devices & services → Helpers → + Create helper → Template → Sensor` we fill in Name, State (shown below), Unit of measurement (°C), Device class (Temperature), State class (Measurement) and select our ESPHome as device. This adds a sensor we can access with `sensor.<Name>` that calculates the dew point.

inside_dew_point.txt

```
{% set T =
states('sensor.agv_esp32_greenhouse_inside_temperature')|float(none) %}
          {% set RH =
states('sensor.agv_esp32_greenhouse_inside_humidity')|float(none) %}
          {% if T is not none and RH is not none and RH > 0 %}
            {% set a = 17.62 %}
```

```
            {% set b = 243.12 %}
            {% set gamma = (a * T) / (b + T) + log(RH / 100) %}
            {{ ((b * gamma) / (a - gamma)) | round(2) }}
          {% else %}
            {{ none }}
          {% endif %}
```

A second template is used for checking if the control condition for venting are met. Instead of a `Sensor` this is a `Binary Sensor` with state:

should_vent.txt

```
{% set tin  = states('sensor.agv_esp32_greenhouse_inside_temperature')
| float(none) %}
          {% set tout =
states('sensor.agv_esp32_greenhouse_outside_temperature') | float(none)
%}
          {% set hin  =
states('sensor.agv_esp32_greenhouse_inside_humidity') | float(none) %}
          {% set dpin  = states('sensor.greenhouse_dew_point_inside') |
float(none) %}
          {% set dpout = states('sensor.greenhouse_dew_point_outside')
| float(none) %}

          {% if None in [tin, tout, hin, dpin, dpout] %}
            false
          {% else %}
            {% set dT = 0.5 %}
            {% set dp_max = 1.0 %}
            {% set dp_margin = 0.7 %}

            {% set extreme_temp = (tin >= 29 and tout < (tin - dT)) %}
            {% set normal_temp  = (tin > 27 and tin < 29 and tout <
(tin - dT) and dpout <= (dpin + dp_max)) %}
            {% set humidity     = (hin > 70 and tout <= 27 and dpout <
(dpin - dp_margin)) %}

            {{ extreme_temp or normal_temp or humidity }}
          {% endif %}
```

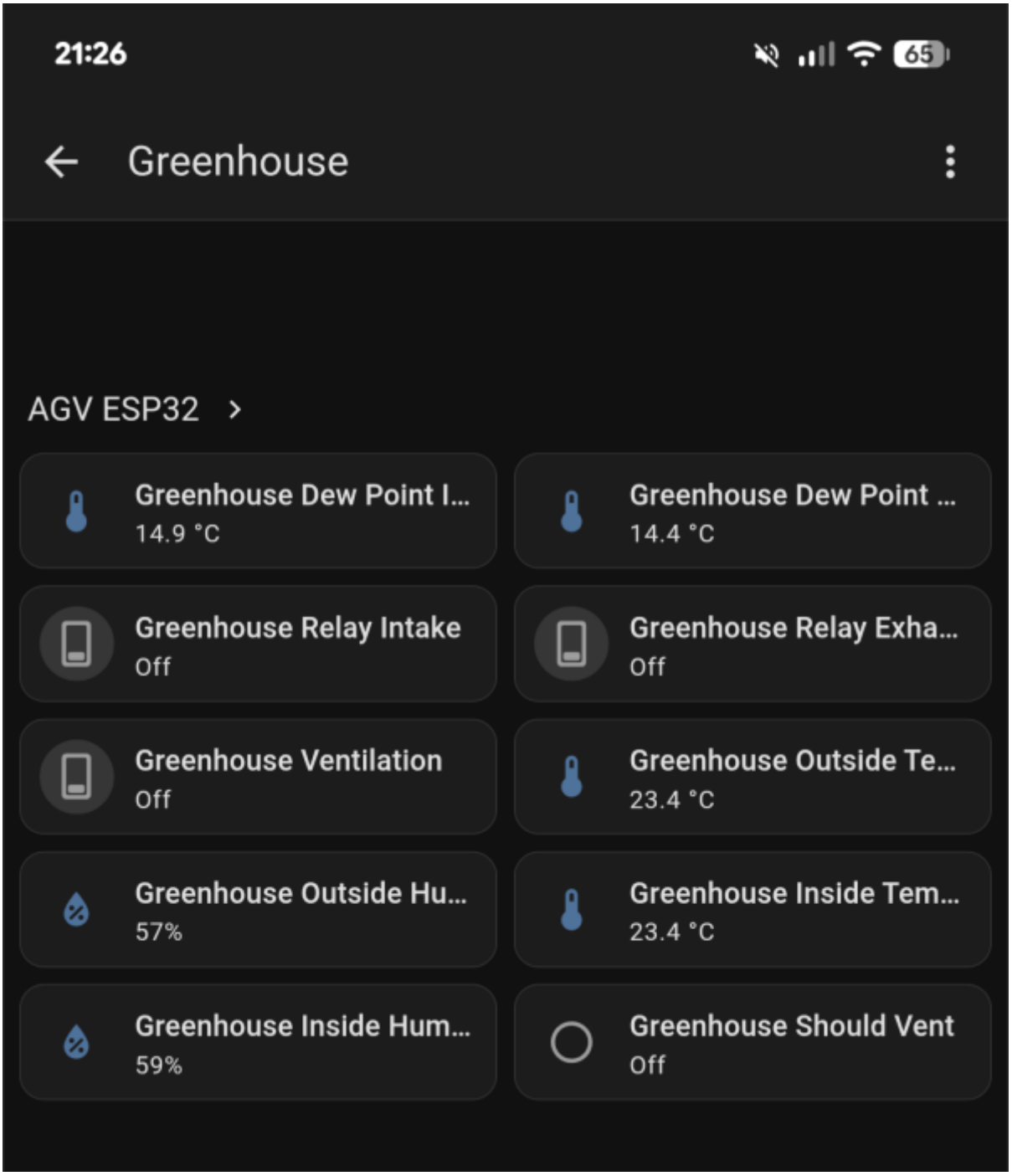

Fig. 11

4.5.2 Automations

The ventilation control logic is implemented using Home Assistant automations using the template created in the previous step.

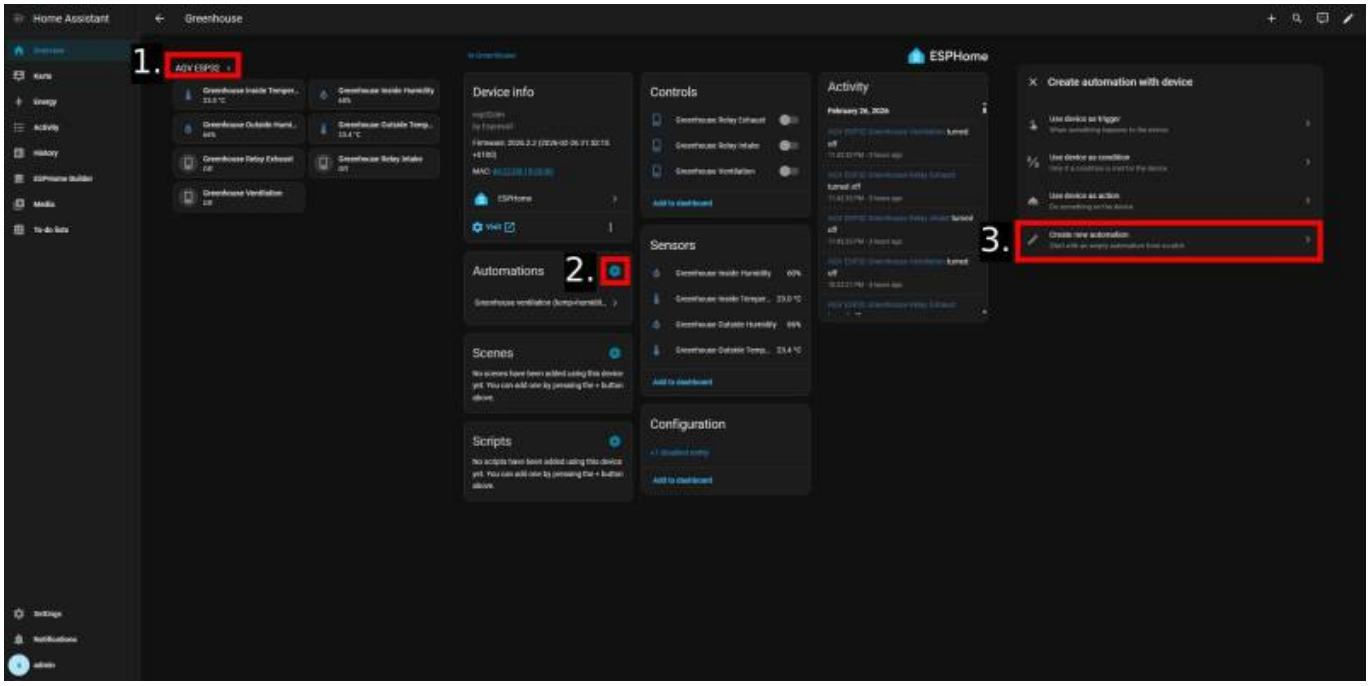

Fig. 12

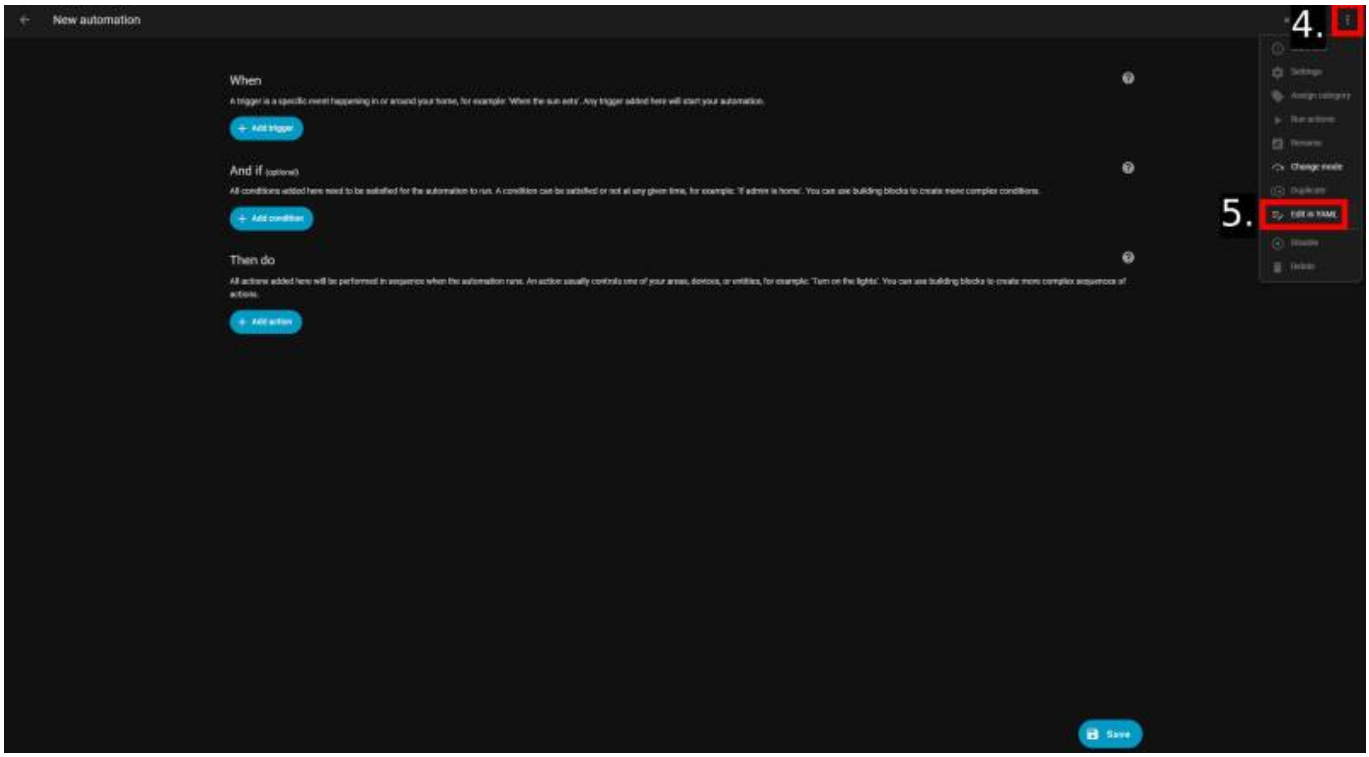

Fig. 13

The following YAML file specifies the criteria that Home Assistant uses to determine when to activate the ventilation system based on 2.2 Control concept.

automation.yaml

```
alias: Greenhouse Ventilation
description: >-
  Vent switch follows binary_sensor.greenhouse_should_vent with anti-
flip-flop
  delays.
```

```
triggers:
  - entity_id:
      - binary_sensor.greenhouse_should_vent
    to:
      - "on"
    for:
      hours: 0
      minutes: 1
      seconds: 0
    trigger: state
  - entity_id:
      - binary_sensor.greenhouse_should_vent
    to:
      - "off"
    for:
      hours: 0
      minutes: 2
      seconds: 0
    trigger: state
actions:
  - target:
      entity_id: switch.agv_esp32_greenhouse_ventilation
    action: switch.turn_{{ trigger.to_state.state }}
mode: single
```

To prevent the ventilation system from switching on or off due to brief sensor spikes or small fluctuations around the thresholds, the “on” conditions must be met for one minute and the “off” conditions for two minutes.

5. Testing & Validation

5.1 Test plan

At the time of writing, the ventilation system could not be tested in the real greenhouse environment because a new greenhouse is currently under construction. Testing was therefore performed at home. While this limits realism, it enables a more controlled environment and repeatable test scenarios. The bathroom was chosen as the test environment because its volume of approximately 8 m³ is suitable for controlled ventilation experiments using one intake and one exhaust fan. This two-fan setup provides a comparable air exchange rate to the greenhouse design, because halving both the room volume (15 m³ → ~8 m³) and the number of fans (4 → 2) keeps the ventilation capacity per volume in a similar range under ideal conditions. In addition, temperature can be increased in a predictable way using a 2kW heater, and relative humidity can be raised quickly (e.g., through shower steam). This makes it possible to evaluate the system by observing how temperature and humidity change when the fans are enabled.

Planned scenarios include:

- Sensor sanity check

- Temperatur rise scenario
    - stop heating when vent starts turn fans off (baseline)
    - stop heating when vent starts keep fans on
    - keep heating when vent starts keep fans on
- Humidity rise scenario

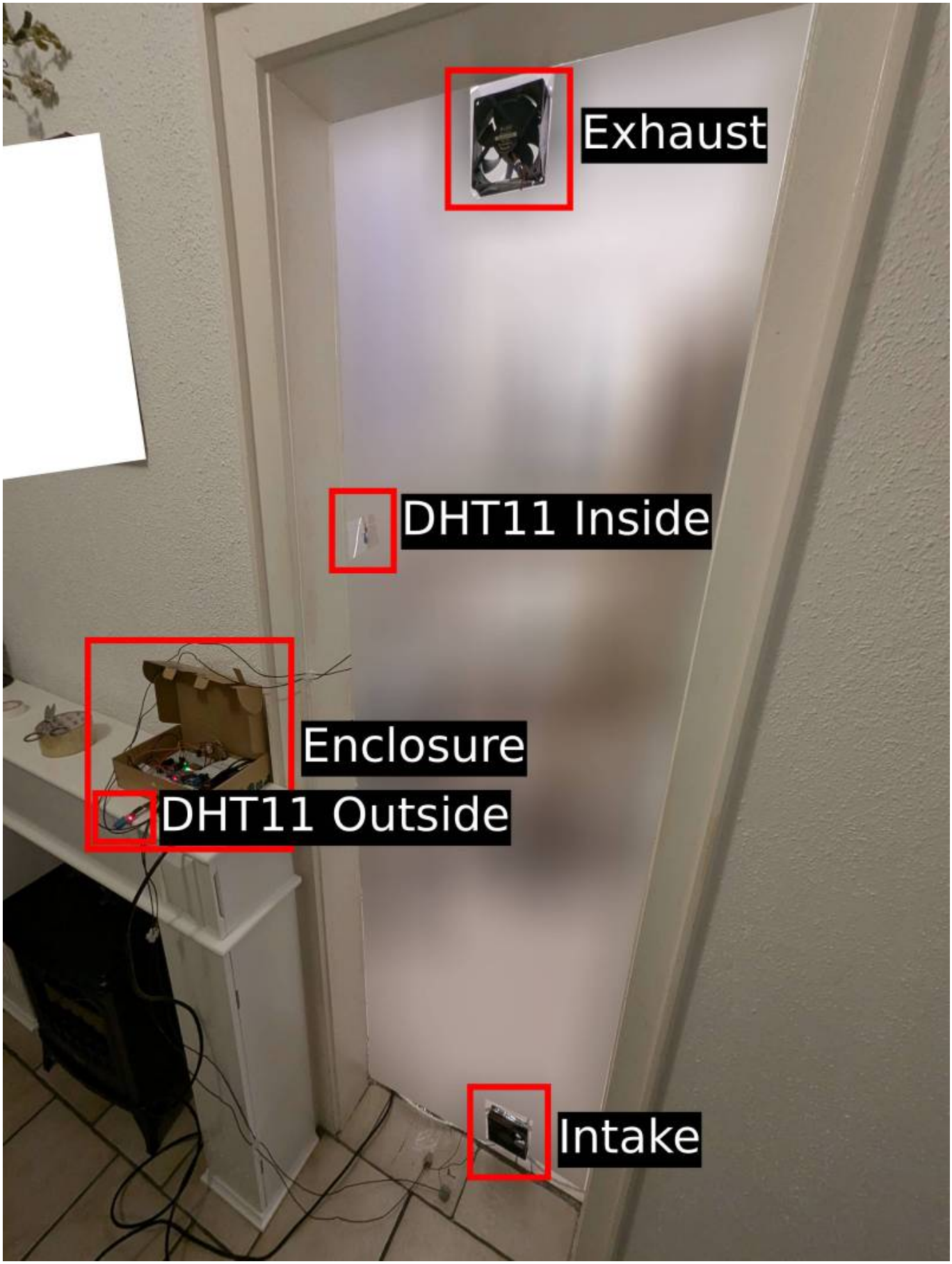

Fig. 14

5.2 Data collection and logging

Measurements (inside/outside temperature and humidity) and actuator states (intake/exhaust/ventilation) are recorded using the ESPHome Logs. This provides a unified view of sensor time series and fan switching events and supports manual notes during testing.

5.3 Observations

Sensor sanity check:

When placed in the same environment, the DHT11 sensors produced highly consistent readings, staying within ±0.1 °C for temperature and within ±1% RH for relative humidity.

Temperature rise scenario 1

In this scenario, the room was heated until the ventilation condition triggered (including the configured 1-minute stability delay). At that point, both the heater and the fans were turned off. The objective was to measure how long it takes for the indoor temperature to drop back below the “should vent” threshold without active ventilation (baseline cooling).

Each run started from an initial indoor temperature of approximately 23.8 °C, and the test was repeated four times.

Measured time to cool out of the “should vent” condition (heater OFF, fans OFF):

- Run 1: 3:50 min
- Run 2: 4:00 min
- Run 3: 4:20 min
- Run 4: 4:20 min

The average time across all runs was 4:07 min, with a slight increasing trend over the repetitions (later runs took longer).

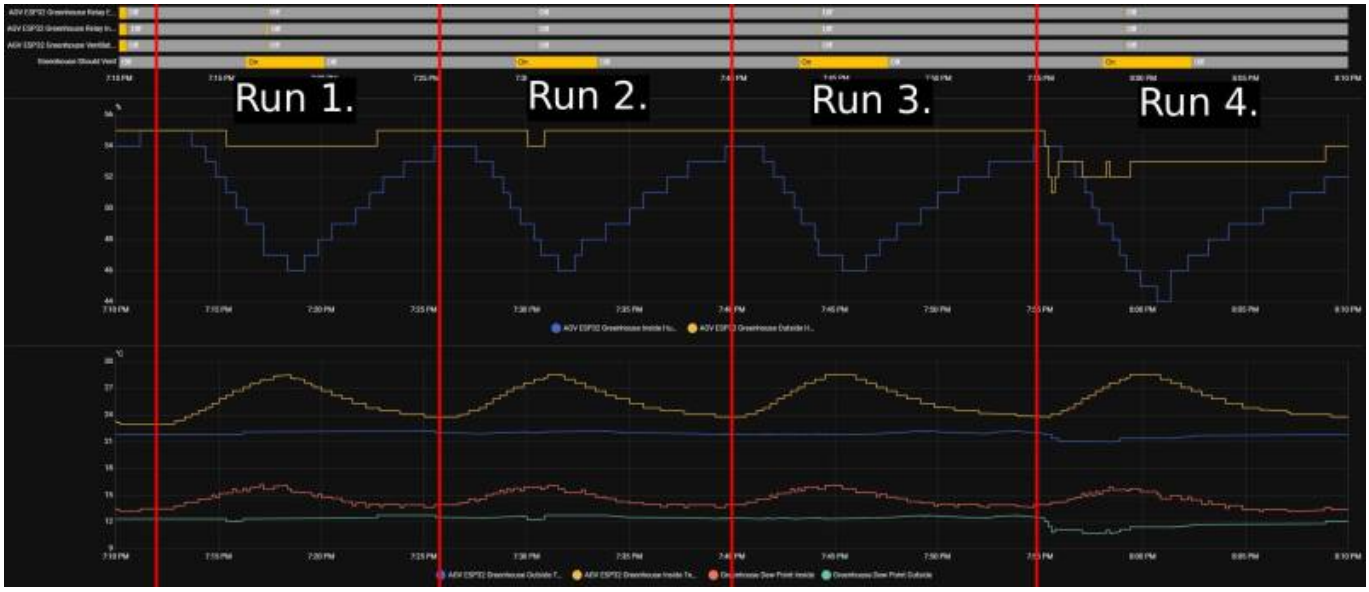

Fig. 15

Temperature rise scenario 2

This scenario follows the same procedure as Scenario 1: the room is heated until the ventilation condition triggers (including the 1-minute stability delay). Once triggered, the heater is turned off. In contrast to Scenario 1, the fans remain on to actively cool the room back below the “should vent” threshold.

The test was repeated three times.

Measured time to cool out of the “should vent” condition (heater OFF, fans ON):

- Run 1: 3:50 min
- Run 2: 3:50 min
- Run 3: 3:50 min

All three runs produced the same result, giving an average time of 3:50 min for this scenario.

Fig. 16

Temperature rise scenario 3

With the 2000 W heater kept on, the fans were not able to cool the room back below the “should vent” threshold, but they likely slowed down the temperature increase.

Humidity rise scenario:

In this scenario, humidity was increased using hot water/steam until the automation triggered ventilation at 70% RH. Indoor humidity peaked at 88% RH and was then reduced to 63% RH while the fans were running. The fans stopped automatically after the humidity dropped back below the threshold (Fig 17. - 1.). After the fans stopped, indoor humidity began to rise again (Fig 17. - 2.). When the fans were started manually, humidity continued to decrease until the inside and outside humidity levels were approximately equal. During the final part of the test, the outside humidity increased (Fig 17. - 3.).

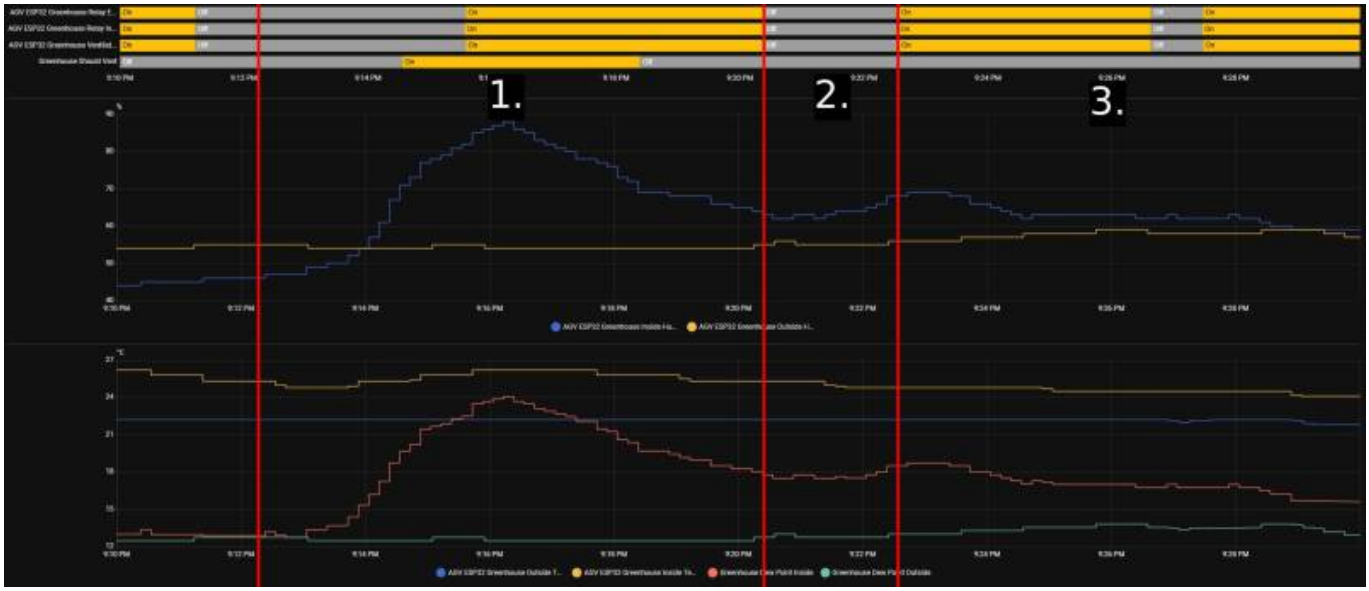

Fig. 17

6. Discussion

The system successfully measured indoor/outdoor temperature and humidity and controlled ventilation automatically via Home Assistant and ESPHome. In the controlled bathroom tests, the automation triggered reliably at the defined thresholds and ventilation reduced both temperature and humidity when outside conditions were more favorable than inside.

In the temperature baseline test (heater OFF, fans OFF), cooling back below the ventilation threshold took 4:07 min on average. With ventilation enabled after switching the heater off (heater OFF, fans ON), cooling took 3:50 min across all runs. With a 2000 W heater kept ON, ventilation could not restore temperature below the threshold, but it likely slowed the temperature increase. The baseline cooldown duration increased over repeated trials (from 3:50 to 4:20), which is plausibly explained by heat storage in the room’s surfaces (walls, tiles, furniture) and their gradual release after the heater was switched off. The ventilation-enabled cooling test was performed after the baseline series, meaning it started under less favorable conditions; despite this, the cooldown time with fans remained lower (3:50), supporting a real cooling effect from ventilation in this setup.

In the humidity test, ventilation started automatically at 70% RH. Indoor humidity peaked at 88% RH and was reduced to 63% RH while the fans were running. After the fans stopped, indoor humidity rose again even though no additional moisture was added, suggesting that without continued airflow some humid air pockets can remain and moisture on surfaces can re-equilibrate with the air. If humidity had climbed above 70% RH again, the automation would have restarted ventilation. For this test, the fans were started manually instead, and indoor humidity continued to decrease until it approached equilibrium with the outside sensor reading. Towards the end, the measured outside humidity also increased slightly, which is most likely an artifact of the indoor test setup with limited fresh-air exchange rather than a realistic outdoor effect.

7. Conclusion and outlook

This project demonstrates a working prototype of a Home Assistant–based greenhouse ventilation system using low-cost sensors and an ESP32 controller. The implementation covers the complete

chain from sensing (inside/outside temperature and humidity) to automated actuation (relay-controlled intake/exhaust fans) and provides a practical foundation for further improvements. Initial indoor validation showed that the automation triggers reliably and that ventilation can measurably improve temperature and humidity conditions when outside air is more favorable than inside air.

Outlook

The next step is the deployment and long-term validation in the real greenhouse once construction is completed. Beyond installation, several extensions could improve functionality, robustness, and autonomy:

- Additional sensing
  - Soil moisture sensing
  - Light and CO_2 sensing for improved climate control decisions
- Extended automation features
  - Automatic watering based on soil moisture and schedules
  - Artificial lighting for plant support during low-light periods
- Active climate control (beyond fans)
  - Heater and/or active cooling
  - Humidifier / dehumidifier system
- Off-grid power system
  - Battery/solar-based power supply for independent operation
  - Power budgeting, electrical safety (e.g., fusing), and autonomy targets (e.g., nights or multiple cloudy days)

8. References / Sources

- [1] Shamshiri, Redmond & Jones, James & Thorp, Kelly & Ahmad, Desa & Che Man, Hasfalina & Taheri, Sima. (2018). Review of optimum temperature, humidity, and vapour pressure deficit for microclimate evaluation and control in greenhouse cultivation of tomato: a review. International Agrophysics. 32. 287-302. 10.1515/intag-2017-0005.
- [2] https://kaufbauer.de/blog/beitrag/gewaechshaus-temperatur-beleuchtung-feuchtigkeit-einstellen.html
- [C1] DHT11 module (plug-and-play): https://www.amazon.de/dp/B07TSF94KD
- [C2] 12V 92mm fans (54.8 CFM spec): https://www.amazon.de/dp/B0FJ8D4Z3W
- [C3] 4-channel 5V relay module: https://www.amazon.de/dp/B01M8G4Y7Z
- [C4] Home Assistant Green: https://www.home-assistant.io/green/
- [C5] Olimex ESP32-POE-ISO: https://www.olimex.com/Products/IoT/ESP32/ESP32-POE-ISO/open-source-hardware
- [C6] TP-Link TL-WR840N router: https://www.amazon.de/dp/B00HNFP4HO
- [C7] TP-Link TL-POE160S PoE+ injector: https://www.amazon.de/dp/B08R3ZHZ78
- [C8] REVODATA PoE splitter 12V/2A: https://www.amazon.de/dp/B08HS4NT13
- [C9] 12V→5V buck converter: https://www.amazon.de/dp/B0BCV6GRH5
- [S1] ESPHome Getting Started (Home Assistant add-on): https://esphome.io/guides/getting_started_hassio/
- [S2] ESPHome DHT sensor component documentation: https://esphome.io/components/sensor/dht/

- [S3] Home Assistant Template documentation: https://www.home-assistant.io/integrations/template/