aKite
RETAIL REVOLUTION

Smart HVAC System:

Energy Optimization and Advanced Control

Detailed, with code

Author:
Derek GUSATTO

Supervisor:
Wladimiro BEDIN

November 29, 2024

Contents

1 Introduction

The idea is to develop a system that optimizes the energy costs of a space (e.g. a store), of the various strategies taken into consideration (Reinforcement Learning, Bonsai Project and others from summer 2023) the one that prevailed in terms of results and possibilities it is Home Assistant, with an interface to Amazon Alexa. Considering that the system wants to be extended to multiple spaces (e.g. a chain store) it was decided to use Azure for the management of multiple spaces.

1.1 Access Istruction

The Home Assistant dashboard is available from this site To get log in credentials or a new profile you need to ask the Administrator (Derek Gusatto, editor of this document, send an email).

2 Technologies

2.1 Home Assistant

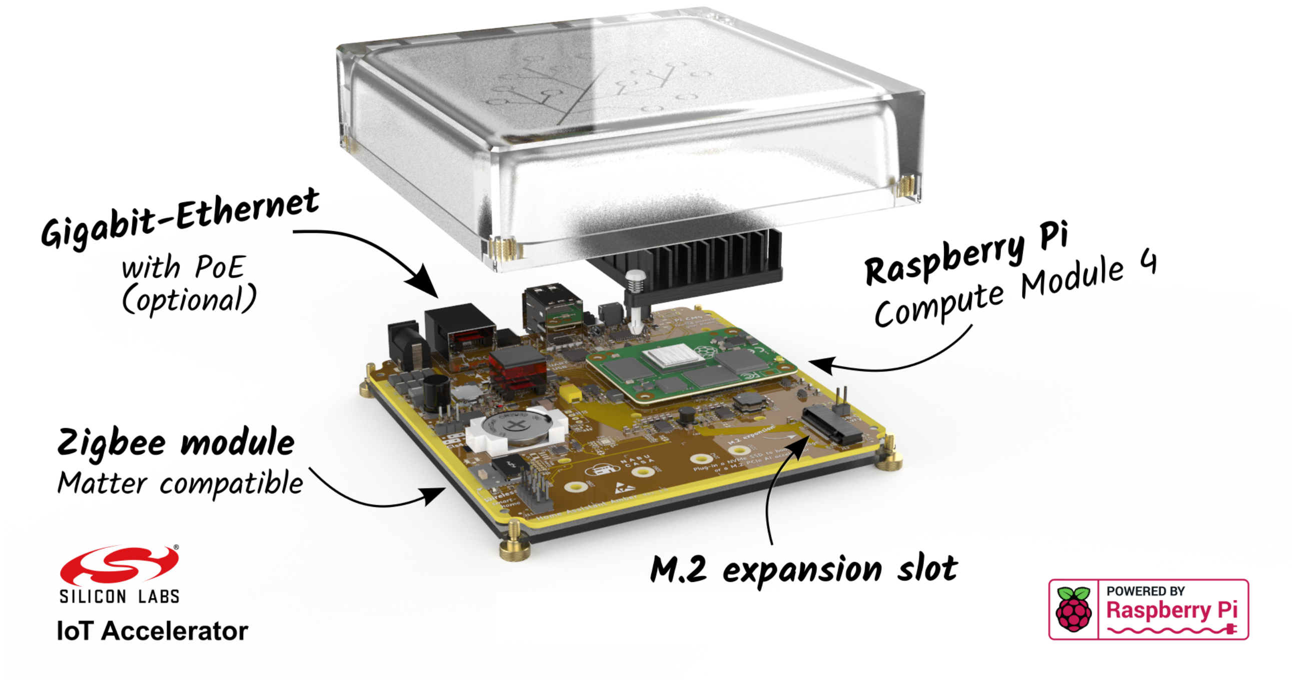
Home Assistant is an open-source home automation platform designed to provide users with a seamless and customizable experience in managing and automating smart devices in their homes. Home Assistant is developed and maintained by a global community of contributors, ensuring continuous improvements and innovations. Its open-source nature allows users to customize and extend the platform to meet specific needs.

Supporting a vast array of smart devices and protocols, including Zigbee, Z-Wave, and MQTT, Home Assistant provides broad compatibility that allows users to integrate and control devices from different manufacturers within a single platform. Users can create powerful automations and scripts to perform complex tasks based on conditions such as time of day, device status, or user location. This capability enables personalized and responsive smart home environments.

Home Assistant empowers users to create intelligent and responsive living environments tailored to their unique needs. With its focus on privacy, flexibility, and broad compatibility, it stands as a powerful solution for anyone looking to take control of their smart home ecosystem.

2.1.1 Yellow Box

We've chosen a Yellow Box, the intermediate level hardware component by Home Assistant because it is extensible and comes with all the ingredients you need to help you build a robust smart home. As others Home Assistant Boxes, it needs a *Raspberry Pi 4 installation*.



2.1.2 Nabu Casa

Nabu Casa is a subscription-based cloud service that seamlessly integrates with Home Assistant to provide secure remote access and enhanced functionality without the need for complex network configurations. It offers a straightforward solution for users who want to control their smart home devices from anywhere in the world. Nabu Casa Cloud facilitates easy integration with popular voice assistants such as Amazon Alexa and Google Assistant, allowing for voice control of Home Assistant-enabled devices. Additionally, it ensures user privacy by handling remote access without exposing sensitive data or requiring port forwarding. By supporting the development of Home Assistant, Nabu Casa Cloud contributes to the continuous improvement and innovation of the platform while offering users a reliable and convenient way to expand their smart home capabilities.

2.2 ESP Home

ESPHome is an open-source platform designed to simplify the integration of ESP8266 and ESP32 microcontrollers with home automation systems, particularly Home Assistant. It provides a user-friendly framework for creating custom firmware that runs on these devices, enabling them to communicate seamlessly with your smart home setup. ESPHome allows users to define sensor readings, control outputs, and configure Wi-Fi connections through an intuitive YAML configuration file, eliminating the need for complex coding. By leveraging ESPHome, users can integrate a wide range of devices into their home automation system, from temperature sensors and light switches to more specialized components, all while maintaining compatibility with Home Assistant. ESPHome not only streamlines the process of developing and deploying firmware but also enhances the flexibility and scalability of home automation projects, making it an invaluable tool for creating a fully customized smart home environment.

2.3 Amazon Alexa

Wanting to take advantage of users' familiarity with Amazon Alexa and its intuitive interface, we chose to interface the devices connected to Home Assistant also with Alexa. This is possible thanks to the skills already made available by Home Assisitant and the Nabu Casa cloud account. It's possible to design a new skill from scratch too.

2.4 Microsoft Azure

Azure is Microsoft's cloud computing platform offering a wide range of services, including computing, analytics, storage, and networking. It's designed to help organizations build, manage, and deploy applications on a global network of data centers.

For Internet of Things (IoT), Azure provides a robust suite of tools and services enabling the development, deployment, and management of IoT solutions. Key capabilities include:

- **Azure IoT Hub**: A central messaging hub to connect, monitor, and control IoT devices securely and at scale. It facilitates the bi-directional communication between IoT devices and the cloud, enabling the ingestion of sensor data from a wide array of connected devices. IoT Hub supports protocols like MQTT, AMQP, and HTTPS, making it versatile for various use cases. It can integrate seamlessly with services like **Azure Digital Twins** to model and simulate complex environments based on real-time sensor datasets. This capability allows businesses to create dynamic digital replicas of physical assets, providing actionable insights and enabling predictive analytics.
- **Azure Digital Twins**: Enables the creation of digital replicas of physical environments to analyze systems and optimize operations;
- **Azure IoT Central**: A managed IoT application platform that simplifies the deployment of IoT applications without extensive coding;
- **Azure Sphere**: Provides secure hardware, operating system, and cloud services for building connected devices;
- **Azure Machine Learning and AI**: Allows predictive analytics and real-time decision-making for IoT applications;
- **Edge Computing**: Services like Azure IoT Edge bring cloud intelligence and analytics to the edge, enabling offline operations and reduced latency;
- **Data Integration**: Tools like Azure Time Series Insights and Azure Synapse Analytics make it easy to visualize and analyze IoT data.

2.5 Project Haystack

The Project Haystack is an open-source initiative designed to standardize semantic data models for smart buildings, IoT devices, and related applications. The project addresses the challenge of managing and making sense of the vast amounts of data generated by connected devices in modern infrastructures.

Core Concepts and Goals

- **Semantic Tagging**: Project Haystack uses a uniform tagging methodology to add meaningful, standardized labels to data points, such as sensors, devices, or control systems. This semantic tagging ensures interoperability between different systems;
- **Interoperability**: By creating a common vocabulary, Project Haystack enables disparate systems to work seamlessly together, reducing integration complexity and costs.
- **Open-Source Model**: The initiative provides an open, community-driven platform for developing and sharing standardized tagging libraries, tools, and resources;
- **Application Focus**: The standard is particularly beneficial for industries such as smart buildings, energy management, HVAC systems, and industrial IoT, where managing large-scale, diverse data streams is critical.

Key Features

- **Tagging Framework**: Provides an extensible system for defining metadata tags for any kind of device or data point;
- **Data Models**: Offers pre-defined models for common systems like lighting, HVAC, and energy meters.
- **RESTful APIs**: Supports communication protocols for exchanging tagged data between applications;
- **Community Contributions**: A rich library of community-created tools, libraries, and resources is available to enhance the adoption of Haystack standards.

Advantages

- Reduces the cost and complexity of integrating IoT and building automation systems;
- Promotes greater system interoperability;
- Provides a scalable and flexible framework adaptable to future needs.

3 Components

3.1 Microcontrollers

- **Raspberry Pi 4** (in the Home Assistant Yellow Box);
- **ESP32**: with ESP-Home framework installed manages sensors and actuators with digital communications, 1-Wire protocol, UART, and others. The ESP communicate with Home Assitent via Wi-Fi.

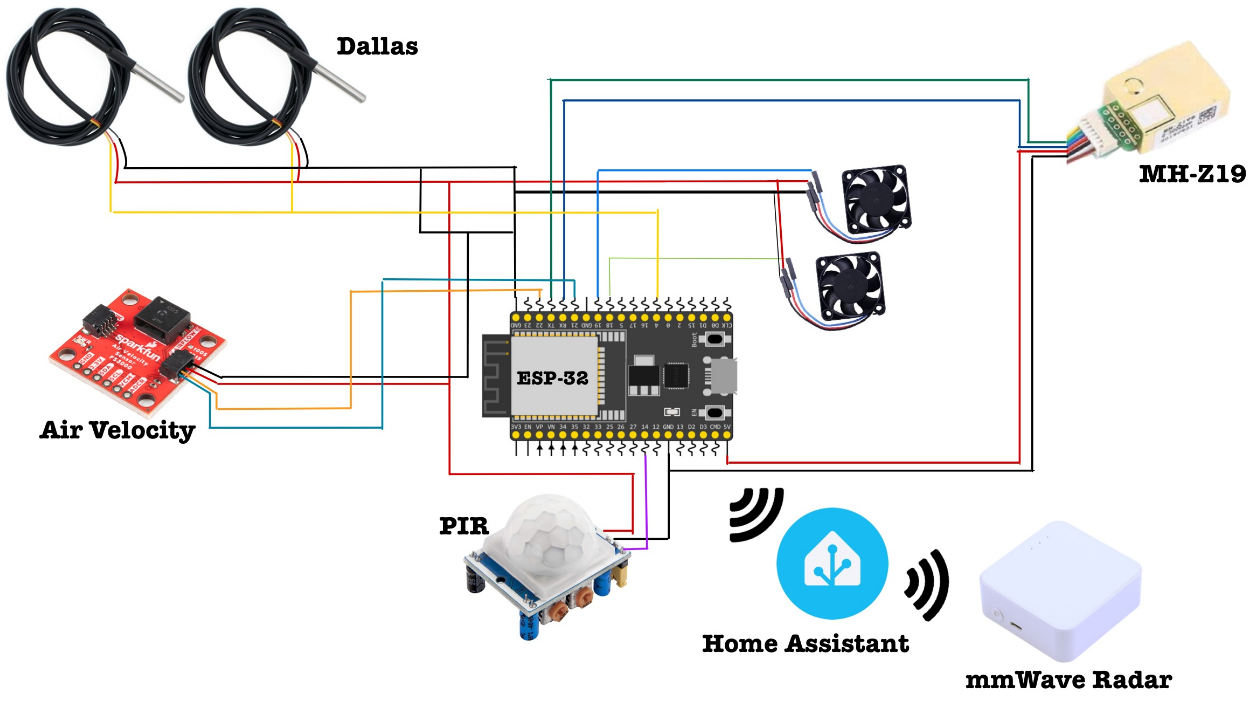
3.2 Sensors

- **Dallas**: Dallas Temperature Sensors (DS18B20) are precise and reliable, over a wide range, furthermore they feature 1-Wire for minimal wiring, through which it is connected to the ESP;
- **MH-Z19**: MH-Z19 is a compact and reliable carbon dioxide (CO_2) sensor designed for indoor air quality monitoring. It provides accurate CO_2 concentration measurements through its non-dispersive infrared (NDIR) technology and offers easy integration with ESP via UART;
- **HC-SR501**: The HC-SR501 is a widely used passive infrared (PIR) motion sensor designed to detect motion by sensing changes in infrared radiation levels. The HC-SR501 is valued for its adjustable sensitivity and time delay settings, making it versatile for various motion detection needs. It is connectet to a digital pin of the ESP;
- **mmWave Radar**: mmWave Radar is a human detection sensor. The kit by Seeed Studios is an advanced sensor designed for precise motion detection and distance measurement using millimeter-wave technology. Operating at frequencies around 60 GHz, this radar provides high-resolution sensing capabilities that surpass those of traditional infrared or ultrasonic sensors. The sensor is connected to Home Assistant via ESP Home, like the other ESP to which the other sensors are connected;
- **Air Velocity Sensor**: The SparkFun Air Velocity Sensor Breakout - FS3000-1005 (Qwiic) is an advanced air velocity sensor designed for precise airflow measurements. The sensor offers a simple interface using the Qwiic connection system, making it easy to integrate with ESP modules. Its compact design and low power consumption make it ideal for applications like HVAC monitoring, environmental studies, and air quality analysis.

3.3 Actuators

- **PWM fans**: Pulse Width Modulation fans, are widely used in electronic projects and home automation systems for their efficient and precise speed control. By varying the duty cycle of the PWM signal, these fans can adjust their rotational speed to match specific cooling needs, providing enhanced performance and energy efficiency. Their integration into a home automation system allows for dynamic control based on temperature readings, ensuring optimal cooling and energy savings.

3.4 Wiring schemes



4 YAML Configurations

4.1 1-Wire Bus

```
one_wire:
  - platform: gpio
    pin: GPIO04
```

Sets up a 1-Wire bus on GPIO pin 4 for communicating with Dallas temperature sensors.

4.2 Sensor Configuration

```
sensor:
  - platform: dallas_temp
    address:  0x467f5de81d64ff28
    name: "temperature #1"
    id: temperature_sensor1
    update_interval: 60s

  - platform: dallas_temp
    address:  0x950ad8f11d64ff28
    name: "temperature #2"
    id: temperature_sensor2
    update_interval: 60s

  - platform: mhz19
    id: mh_z19
    co2:
      name: MH-Z19 CO2 Value
      id: mh_z19_co2

  - platform: fs3000
    name: "Air Speed"
    model: 1005
    update_interval: 60s
    unit_of_measurement: m/s
```

This section configures two Dallas temperature sensors, an MH-Z19 CO2 sensor and the Air Speed sensor.

4.3 Binary Sensor Configuration

```
binary_sensor:
  - platform: gpio
    pin: GPIO14
    name: "PIR Sensor"
    device_class: motion
```

Sets up a PIR motion sensor connected to GPIO pin 14.

4.4 Output and Fan Configuration

```
output:
  - platform: ledc
    id: fan_speed
    pin: GPIO19
    frequency: 25000 Hz
    min_power: 0.01
    max_power: 1.0

  - platform: ledc
    id: fan_speed18
    pin: GPIO18
    frequency: 25000 Hz
    min_power: 0.01
    max_power: 1.0
```

Defines outputs using the LEDC platform for controlling fan speed on GPIO pins 19 and 18.

```
fan:
  - platform: speed
    output: fan_speed
    name: "Fan 19"
    id: fan_toggle

  - platform: speed
    output: fan_speed18
    name: "Fan 18"
    id: fan_toggle18
```

Configures two fans using the defined outputs for speed control.

4.5 UART Configuration

```
uart:
  tx_pin: GPIO01
  rx_pin: GPIO03
  baud_rate: 9600
```

Sets up UART communication with specified TX and RX pins and a baud rate of 9600.

4.6 I2C Configuration

```
i2c:
  sda: GPIO21
  scl: GPIO22
  scan: true
  id: bus_i2c
```

Sets up I2C communication with specified SDA and SCL pins.

5 Automations & Helpers

5.1 Automations

Home Assistant automations enable users to create complex interactions between their smart home devices, allowing a more intelligent and responsive environment. These automations are typically defined in the `automations.yaml` file, which is part of Home Assistant's configuration files. In this file, users can define triggers, conditions, and actions that dictate how devices should behave under certain circumstances. Triggers are events that initiate an automation, such as a time of day or a sensor reaching a specific state. Conditions specify additional requirements that must be met for the automation to proceed, while actions define what the automation will do, such as turning on lights or sending notifications. The most important automations that have been designed are those for sending notifications in the case of sensors that do not respond or report problems. These automations, called "*Error - [Name] Sensor*" have a structure like this:

```
if sensor.state == unavailable or sensor.state == unknown
        SEND NOTIFICATION
```

At the yaml code level we therefore have (for the MH-Z19 CO_2 sensor):

```
- id: "1726068594861"
  alias: Error - CO2
  description: ""
  triggers:
    - trigger: state
      entity_id:
        - sensor.sensors_network_mh_z19_co2_value
      to: unavailable
    - trigger: state
      entity_id:
        - sensor.sensors_network_mh_z19_co2_value
      to: unknown
  condition: []
  action:
    - action: notify.notify
      metadata: {}
      data:
        message: CO2
        title: Error with a sensor
  mode: single
```

5.2 Helpers

Home Assistant helpers provide users with tools to manage and enhance their smart home setups by adding layers of customization and logic that go beyond basic device interactions. These helpers are defined within Home Assistant's interface or in configuration files, and they enable the creation of virtual entities or variables that can be referenced in automations and scripts. Common types of helpers include input booleans, input numbers, and input selects, each offering a unique functionality to streamline control and monitoring.
For instance, input booleans act as simple on/off switches that can be used to enable or disable specific automations. Input numbers allow users to set numerical values for dynamic adjustments, such as specifying a desired temperature range. Input selects let users create drop-down menus with predefined options, providing a flexible way to switch between multiple states or modes.
The helpers were used in two ways:

- to keep track of the date and time of the last presence detection of the mmWave Radar sensor;
- to keep track of the date and time of the last temperature changes detected by each of the two dallas sensors;

5.3 Algorithmic data

Automations and helpers were combined to obtain three ”dummy” sensors, using sensor templates in `configuration.yaml`, which are updated by automations based on helpers. These sensors are used for the automations needed for climate management, so they are just logical data to be used in algorithms. These are:

- **Ore dall'ultima presenza**: reports the difference in hours between now and the time the helper last detected presence;
- **Tempo dall'ultimo cambiamento Temp #1**: reports the difference in seconds between now and the time the helper last detected temperature change from the first dallas sensor;
- **Tempo dall'ultimo cambiamento Temp #2**: reports the difference in seconds between now and the time the helper last detected temperature change from the second dallas sensor.

6 Climate Controls

This YAML configuration sets up climate control using various strategies for managing heating and cooling systems. The configuration contains active and commented-out (inactive) components, reflecting different attempts at implementing climate control mechanisms. The active components currently utilize thermostat-based controllers for separate cooling and heating, while previous configurations explored PID, Bang Bang, and combined thermostat controllers. The PID was great for the ability to continuously adjusting the temperature, but became confusing and self-defeating when using more than one PID concurrently. The Bang-Bang was useful but - as for having only one thermostat - the problem is having two set points and a confusing interface that requires / permit the user to choose between cooling and heating. So the best options results to be having two thermostats, one for cooling and one for heating, and using one of them at time, deciding a priori which of the two to use without asking or showing anything to the user.

6.1 YAML Configuration

```
#COOLER
- platform: thermostat
  name: "Thermostat Climate Cooler"
  id: thermostat_cooler
  sensor: temperature_sensor1
  min_cooling_off_time: 300s
  min_cooling_run_time: 300s
  min_idle_time: 30s
  cool_action:
    - fan.turn_on: fan_toggle18
  idle_action:
    - fan.turn_off: fan_toggle18
  default_preset: Home
  preset:
    - name: Home
      default_target_temperature_high: 23 °C
```

```
# HEATER
 - platform: thermostat
   name: "Thermostat Climate Heater"
   id: thermostat_heater
   sensor: temperature_sensor1
   min_heating_off_time: 300s
   min_heating_run_time: 300s
   min_idle_time: 30s
   heat_action:
     - fan.turn_on: fan_toggle
   idle_action:
     - fan.turn_off: fan_toggle
   default_preset: Home
   preset:
     - name: Home
       default_target_temperature_low: 23 °C
```

6.2 Automations

The procedure will therefore be as follows for activating the thermostats:

```
if last_presence_detected > 36h  //CLOSED
  STOP EVERYTHING
else
    if last_presence_detected > 1h //NOBODY AT THE MOMENT
        if inside_temperature > (idle_temp - 2)
        and inside_temperature < (idle_temp + 2)
      DO NOTHING
    else
            if inside_temperature < (idle_temp - 2)
        HEAT IF
      else if inside_temperature > (idle_temp + 2)
        COOL IF
  else
    if inside_temperature < idle_temp
      HEAT
    else if inside_temperature > idle_temp
      COOL
```

COOL IF and HEAT IF refer to the following logics:

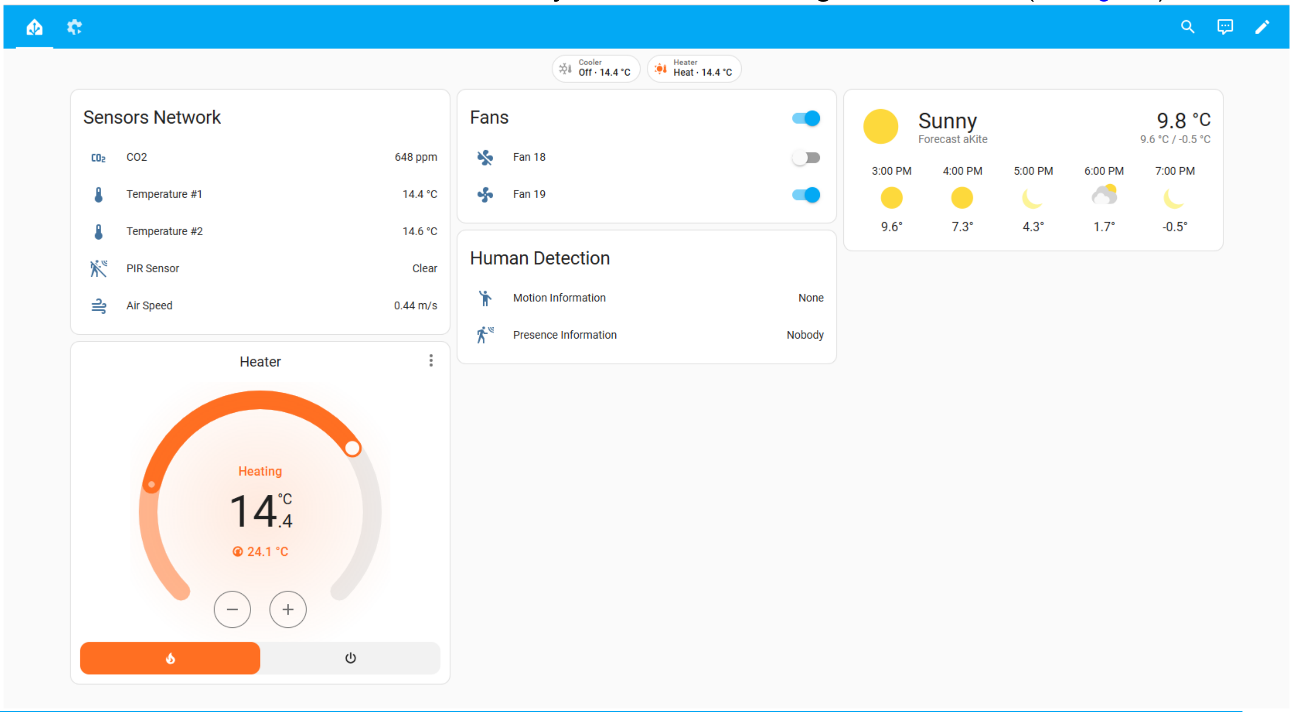
```
if outside_temperature > inside_temperature
        if inside_temperature > idle_temperature
            COOL
        else
            DO NOTHING AND WAIT
    else if outisde_temperature <= inside_temperature
        if inside_temperature < idle_temperature
            HEAT
        else
            DO NOTHING AND WAIT
```

These logics are all applied through automations.

7 Personalised dashboard

In Home Assistant it is more comfortable and convenient to create a new dashboard from scratch rather than trying to customize the default one (*Overview*). This came in handy when wanting to simplify the interface for the user, for example by hiding some controls and showing the thermostats (heater / cooler) only when in operation.

So in practice in the personalised dashboard has been added some cards: one for the sensors connected to the ESP, one for the fans, one for the presence sensor, a forecast card and one for each thermostat. For the thermostats cards has been set a visibility rule bases on the entity state: the card is visible only if the thermostat is in working. In the personalised dashboard there is futhermore a secondary window for the algorithmic data (see §6.3)



Cooler Off · 14.4 °C
Heater Heat · 14.4 °C

Sensors Network

CO2	648 ppm
Temperature #1	14.4 °C
Temperature #2	14.6 °C
PIR Sensor	Clear
Air Speed	0.44 m/s

Fans

Fan 18	off
Fan 19	on

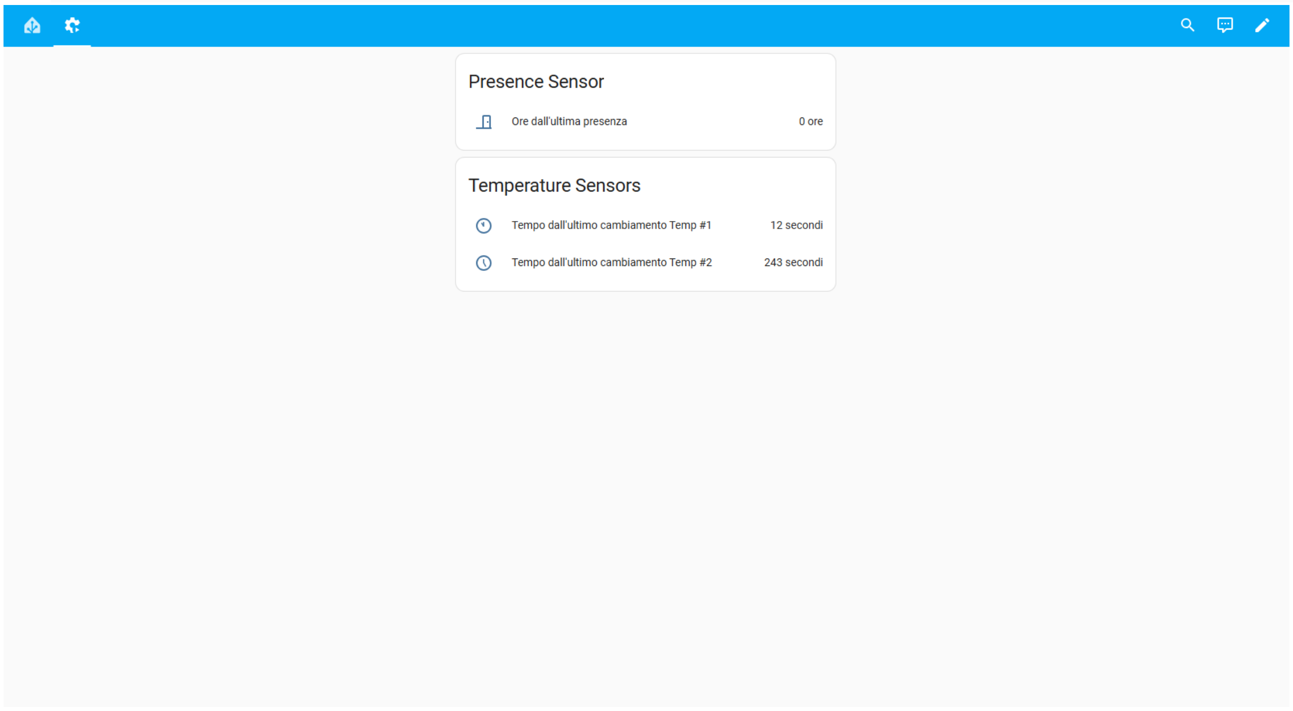
Sunny
Forecast aKite
9.8 °C
9.6 °C / -0.5 °C

3:00 PM	4:00 PM	5:00 PM	6:00 PM	7:00 PM
9.6°	7.3°	4.3°	1.7°	-0.5°

Human Detection

Motion Information	None
Presence Information	Nobody

Heater
Heating
14.4 °C
24.1 °C



Presence Sensor

Ore dall'ultima presenza	0 ore

Temperature Sensors

Tempo dall'ultimo cambiamento Temp #1	12 secondi
Tempo dall'ultimo cambiamento Temp #2	243 secondi

The **problem** with Home Assistant custom dashboards is that they cannot be set as the default dashboard, not even by the system administrator. The dashboard that will open automatically for each user is the full overview dashboard (the default one at birth) and each user will then have to set the custom dashboard as their default view.

8 HayStack & Azure

For the management of a chain of shops, or in any case a set of non-neighboring premises, we chose to use Azure. For communication between the various Home Assistant systems and Azure we chose to use messages in HayStack format to have structured data and, having a standard, remain open to future possibilities.

8.1 HayStack

As regards the HayStack format, there is a Python script which, taking the sensors connected to HomeAssistant, saves them in a format compatible with HayStack.
Subsequently you will need to write a similar script to send updates (e.g. new temperature detection of a sensor).
Furthermore, it will be necessary to evaluate whether a reconversion from HayStack to standard HomeAssistant is needed, whether messages will also be sent from Azure to Home Assistant.

8.2 Azure

For Azure, however, an Azure IoT Hub resource was created (called akite-io-hub). On the HomeAssistant side, however, a script has been created to send messages to Azure.
The next step is to take the received HayStack message and use it to add / update the sensors on the Azure side.

9 Problems & Open Questions

1. **Various ESPHome & HomeAssistant failures**: while the Home Assistant team is aware of some failures in the ecosystem, whether due to updates or for no reason at all, it doesn't look like we're working on fixing them.;

2. **Heartbeat**: as in §5.1 Automations there are already some notifications useful to promptly report when an error is not working as it should. Neverless there should be an heartbeat too. A heartbeat is a tool whereby an "I'm ok" message is periodically sent from the instruments to an external receiver, in order to promptly detect a total failure. Unfortunately Home Assistant doesn't provide an heartbeat method nativaly, so it should be found an alternative;

3. **Various Dallas' calibrations**: it has been discovered that Dallas temperature sensors have very different calibrations, leading to simultaneous readings in the same environment with differences of even a few degrees. Some data can be found in the file: dallas sensors data;

4. **Random air velocity sensor data**: this sensors have 5% of accuracy. This could lead to really bad measurements, and with various peaks (high and low). Others have had the same problem. We tried to contain the problem and solve the problem with a ***Kalman filter***, but it still does not solve enough.

10 Other references

- BS18B20
- HC-SR50
- MH-Z19
- mmWave Radar - Human Detection Sensor by Seeed Studio
- SparkFun Air Velocity Sensor