

SMART HOME SAFETY - 2.0



Luca Pattavina - 262449

Course: Software Engineering for Autonomous Systems
University of L'Aquila
A.Y. 2021-2022
Prof. Davide Di Ruscio

Introduction

Home Safety measures can be classified into two types, external safety measures - such as protecting the home from thieves and external disasters, internal safety measures - such as fire accidents, gas leakage and water leakage. Autonomous systems can be used to automatically monitor and take measures to handle such accidents, and in our project we are focussing on building a Smart Home Internal Safety System using the technologies and methodology used to build efficient autonomous systems.

Objective

In this project, we are focusing on implementing the safety aspect in a smart home automation system. We are trying to detect and solve a few kinds of internal safety threats (unlike the external safety threats like thieves and intruders) that can occur in a basic home environment.

The possible safety threats that we are trying to solve in this project are :

1. Gas Leakage or excess smoke in the house :

While one of the most commonly used resources in a home environment is the LPG gas, there are high chances for a gas leakage which can end in a harmful explosion. A lot of accidents have always been reported on this aspect. Also, excess smoke inhalation causes severe health issues. So, Through this project, we try to prevent this scenario by implementing a proper gas/smoke leakage detection and explosion prevention system.

2. Fire Accident in the house :

Fire is the most widespread cause of death by accident in a home scenario, it affects thousands of residents each year, resulting in injury and loss of life. Accidents involving fire can occur due to electric short circuits or even from a stove flame etc. Through this project, we try to prevent this scenario by implementing a fire detection and fire stopping system.

3. Water Leakage in the house :

Water damage claims are on the rise due to faulty plumbing materials, failed connections, human error, and burst pipes. So, to avoid any property damage or wastage of water, through this project we try to implement a water leakage detection and handling system.

4. Wastage of Resources :

LPG gas and water are resources that are not very efficiently used in a home environment. Sometimes we forget to turn off the stove knob, and sometimes we forget to turn off the water taps as well. So, one of the goals of this project is to prevent wastage of gas and water as well.

5. Air Conditioning

Even though we are talking about safety, comfort has a big role in the home automation systems, avoiding continuous climate adjustments and high consumptions. The goal for this topic, once defined the initial settings, is to automatize this aspect of our smart home.

Technology Specifications:

Our system will consist of the following devices,

1. **Gas Sensors** - They are used in gas leakage detecting equipment in home and industry, are suitable for detecting LPG, natural gas, town gas, cooking fumes and cigarette smoke.
2. **Fire Detector Sensors** - A fire detector is a sensor designed to detect and respond to the presence of a flame or fire.
3. **Temperature Sensors** - Temperature sensors are used to help in fire detection, because fire is a combination of flame, high temperature and smoke.
4. **Flow liquid meter sensors** - Flow liquid meter sensor is one of the sensors that is used for this monitoring process. This sensor uses the Hall Effect sensor inside it to measure the water flow rate and is placed on a pipe that has a diameter equal to the diameter of the sensor.
5. **Floor Water Detection** - This sensor monitors water level in the ground. It's used to prevent not only water leakage, but also damages to mine or my neighbour's water line.
6. **AC and Heater Sensor** - These sensors, one for each device, are used to enable and disable the latter according to our air conditioning plan.
7. **Window and Door Sensor** - In order to have a full description of our home, knowing the state of our main ways of air flowing is a must, and these sensors provide us the possibility to open and close these ways.
8. **Presence Sensor** - Implementing safety and comfort smart home is meaningless if we have no idea of who and where is someone in the house.

Scenarios:

1. **Turning off the gas knobs when it is just left on for a set long time or when the system detects Gas leakage :** Gas Sensor detects that a gas stove was left on for a setted period of time or the percentage of gas is high in atmosphere of the kitchen, then the control unit will instruct the actuator to close the gas knob.
2. **Starting Home Fire Sprinklers when Fire detected at home :** If the system detects fire it will set sprinklers on in that room. If the room it's empty it will close the door to prevent fire spreading in the house.
3. **Turning off the water taps when it is just left on for a set long time :** If the flow liquid meter sensors sense that there is continuous water flow in the pipeline for a setted period of time then our control unit will instruct the actuator to switch off the tap to save water.
4. **Turning off water taps or eventually General Water Pipe:** if the system, using the WaterDetectionSensors detects an increasing water level in the floor, it'll initially close the water taps (like the previous scenario) but if the following measure is even higher it directly closes the General Water Pipe.
- ~~5. **Opening of windows when the temperature is high :** If the sensed values from the temperature sensor is high then the control unit will instruct the actuator to open all the windows of that room.~~
6. **Air Conditioning:** the final user will have a dashboard to manage his temperature preferences for each room. When the actual temperature exceeds from the ones set by the user, the system will use the heater or the AC in the room to restabilize the comfort temperature. Moreover, there's a second dashboard for managing the Presence Mode preferences: if this mode is set on, the room with someone inside will have different temperature settings.

Diagrams

The following diagrams explain the functionality of operations and implementation of the system:

Component Diagram

The proposed framework is based on the MAPE-K model. **Model**, **Analyzer**, **Planner** and **Executer** are 4 different classes that, following this order, receive the information from the MQTT Broker, analyses and routes the data contained in the MQTT message to our little **knowledge**, the SensorValues component, and to the Planner. The latter works with the message received and compares this information with the state of the system written in the knowledge, and decides whether to execute or not the action that the system can perform.

As we can see, we have five more components to describe. Let's start from understanding their colours, the purple one are contained in our Eclipse project, the green ones, instead, are outside.

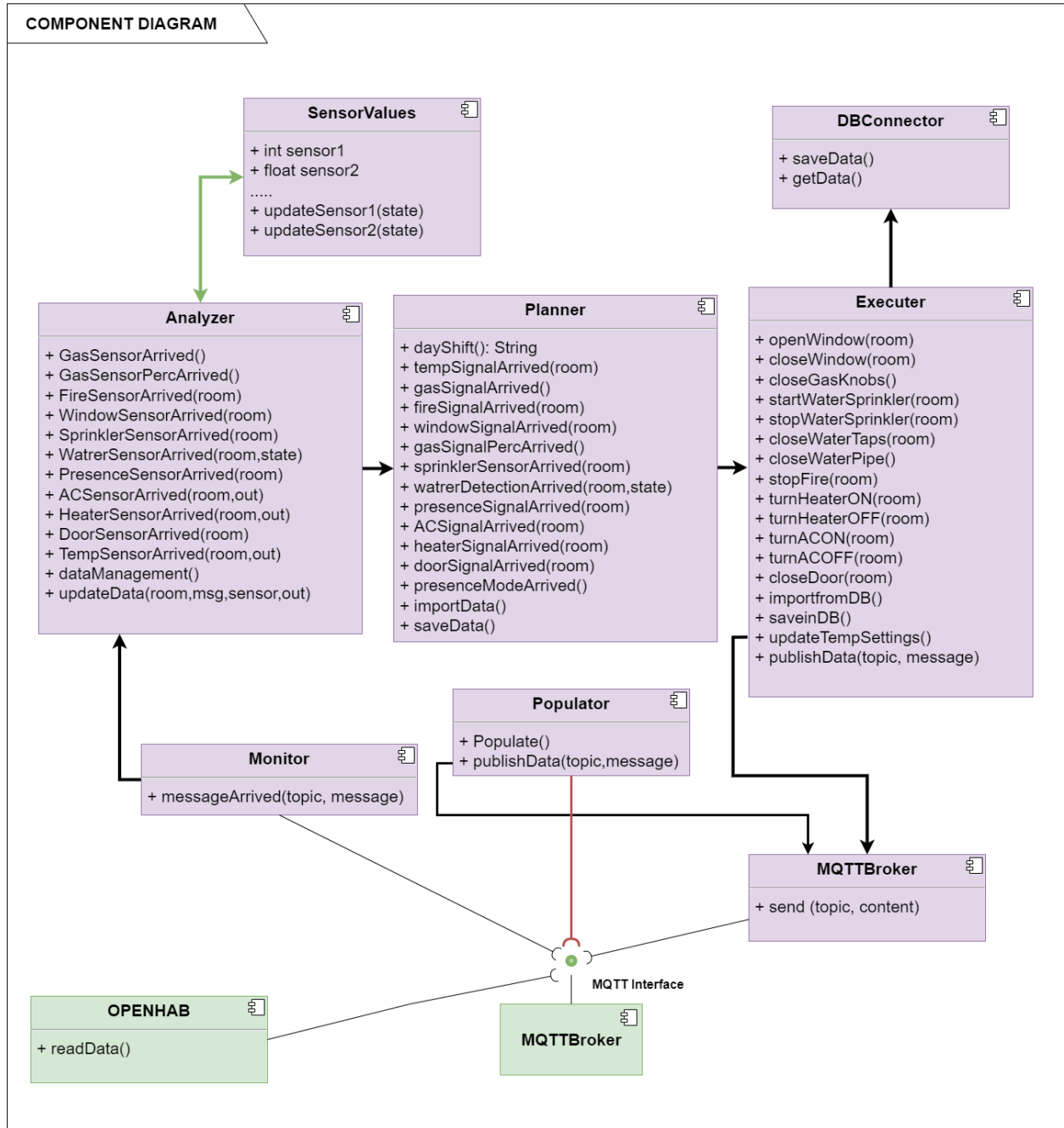
The green **MQTT Broker** is more of a "abstract" component, that reminds ourselves that a MQTT Broker must run in our machine in order to let our messages flow.

OpenHab is our UI, of course linked as a Client in our MQTT interface, in order to receive and display all the changes of our system.

Coming back to our Eclipse code, we have the last three components:

1. **MQTT Broker**, used by our Executer to send the data.
2. **Populator**, that I personally use to simulate the first start of the system, simulating the first data sent by the sensors.
3. **DBConnector**. In this updated version, I added some kind of persistency, giving the user the possibility to save the data relative to his temperature preferences and therefore to restore his last saved data.

SMART HOME SAFETY

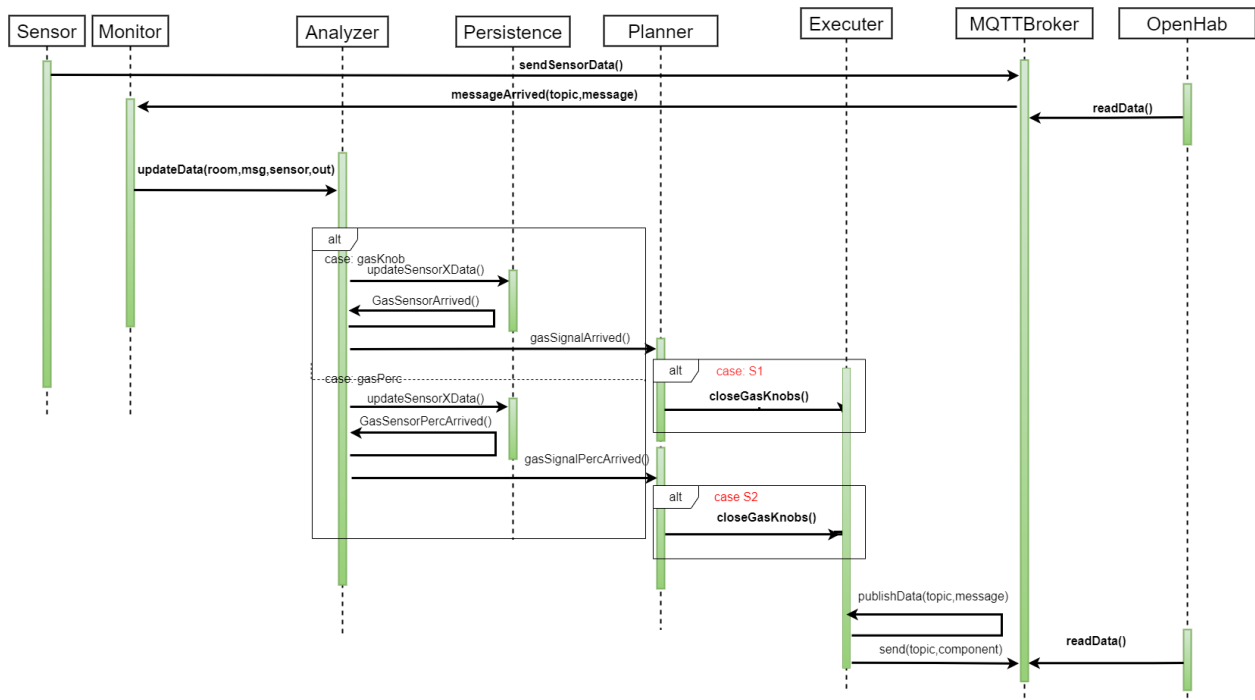


SMART HOME SAFETY

Sequence Diagrams

Scenario 1 - Gas leakage and overconsumption

The following sequence diagram shows how our system works with the data given from the gas sensors. We have two types of gas sensor, one that gives the state of the gas knob (open or closed) and one that sends the quantity of the gas in the air. If the gas is open for more than a given threshold (**case S1**), in our case 15 minutes by default, modifiable in the OpenHab UI, it closes the gas. Also, if the gas it's on and the gas percentage in the room's air reaches 0.8 ‰ (**case S2**) the system will automatically close the gas knob in the kitchen.

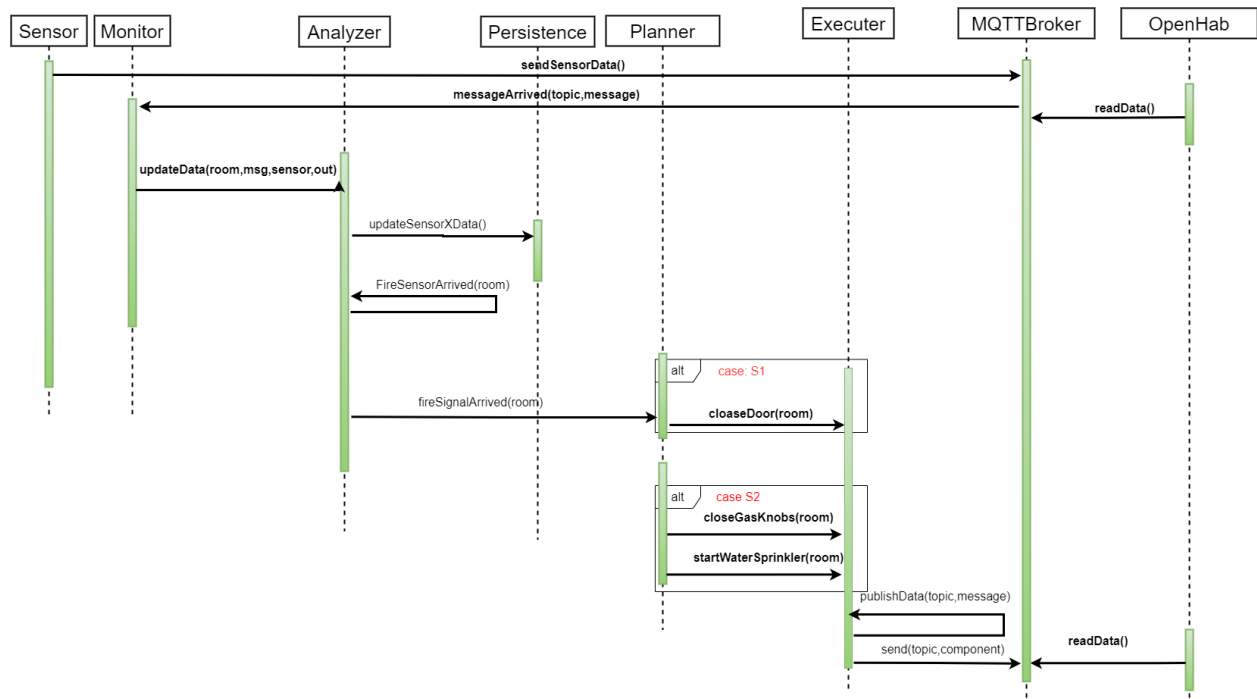


SMART HOME SAFETY

Scenario 2 - Fire detection

The following sequence diagram shows how our system works when fire is detected at home. Once the signal is sensed, after checking that the fire state is "1" (case S2) the system will automatically start the room water sprinklers to shut down the fire. If the room it's empty (case S1), at the moment of the detection, the door will automatically close.

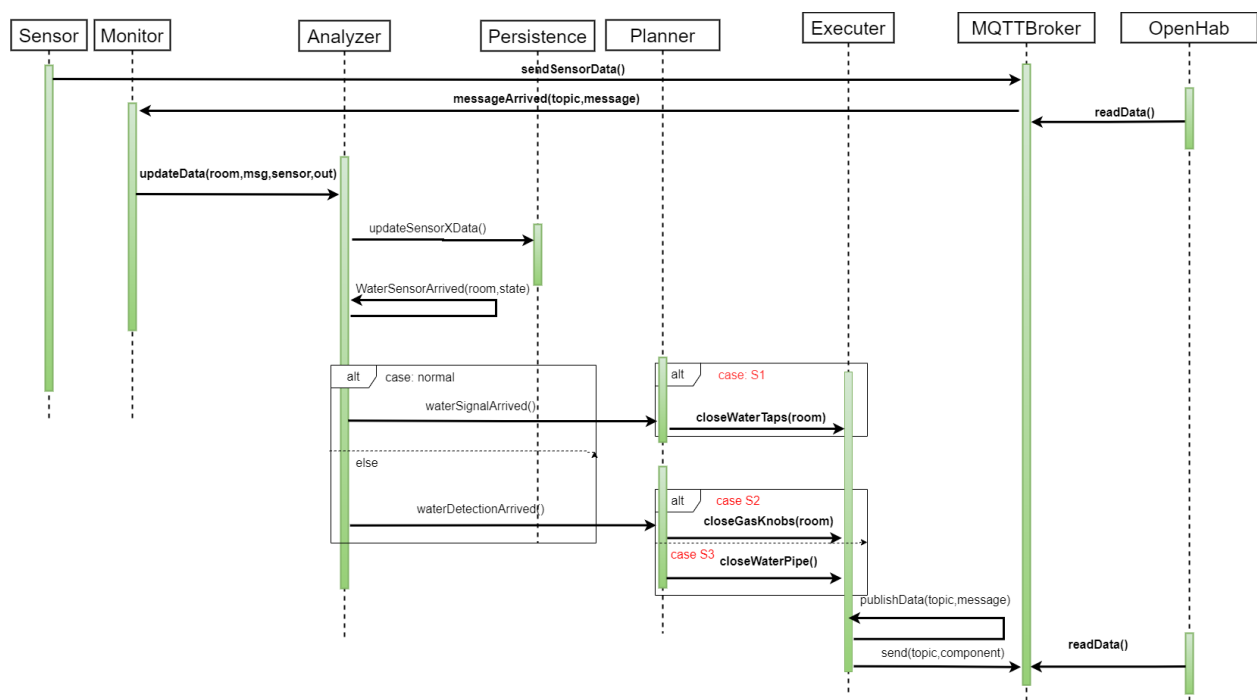
After Some time(i.e. 10 seconds in our project) water sprinklers will automatically stop, and the fire sensor is automatically reset to 0. This is not shown in the sequence.



SMART HOME SAFETY

Scenario 3 and 4 - Water leakage and overconsumption

This sequence diagram shows how our system works when there is continuous flow of water. Based on a selected time_lapse (default 15 minutes in our system), once the water signal is sensed, it waits for this time lapse and, if the water is still on (case S1) the system will automatically close the water in the specified room. In case of water leakage, detected by our WaterDetectionSensors, we'll initially close the local water tap (case s2) (kitchen or bathroom), then, if this level keeps raising, (case S3) we'll close our main water pipe.



Scenario 6 - Air Conditioning

Here, we analyse the steps that our system performs to control and maintain the temperature set by the owner. These temperatures are set in our UI, send via MQTT, read and stored in our knowledge.

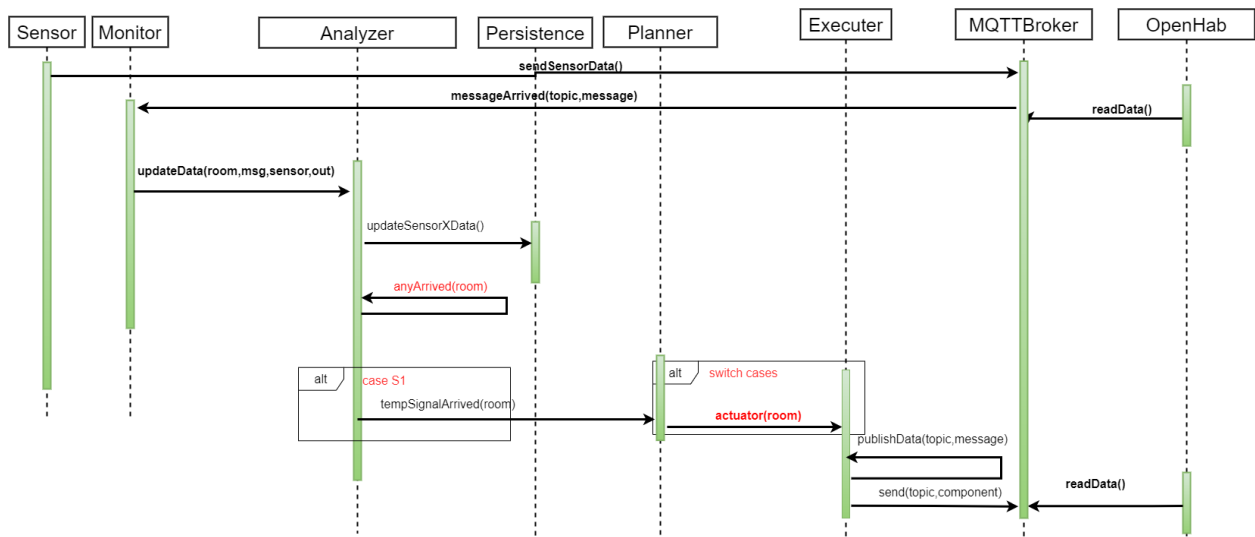
Every time one of these sensors is activated (**anyArrived(room)**):

- roomTempSensor
- roomHeaterSensor
- roomACSensor
- roomSetterMin/Max temperature
- roomPresenceTemperature
- presenceMode

the checking routine for the scenario starts.

We initially check if the message has been sent by our system, (**case S1**) or it's directly from the sensors in our home. In this case, we send the topic message to our planner component that evaluates all the possible temperature adjustments based on the user preferences and the actual state of the system's sensors, like those listed before (**switch cases**).

In our system, we decided to prevent pointless consumptions, so, e.g. if we have to modify the temperature in a room using heater or AC, the window will automatically close, and if the comfort temperature is reached, the system prevents the usage of both heater and AC.

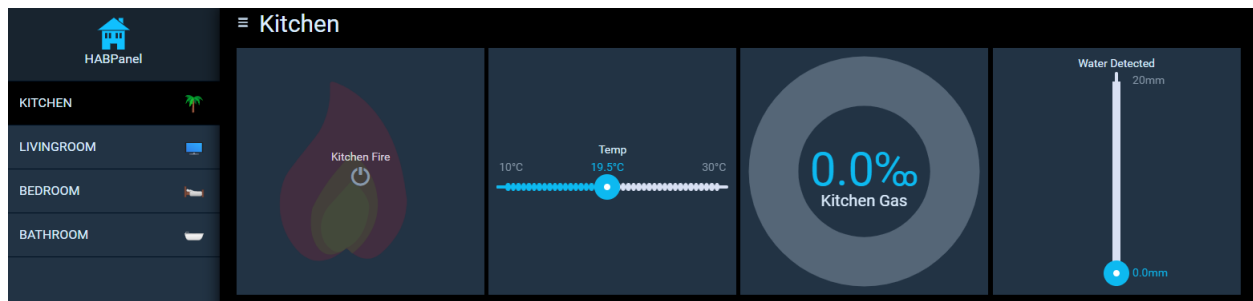


The Application

Starting from the simulation part, we have to split that in two different parts: the starting simulation and running simulation.

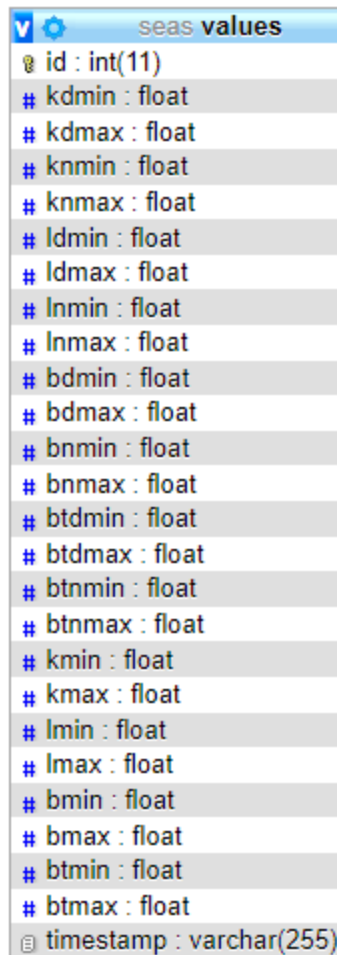
The initial simulation is handled by the **Populator** component, that initializes all the sensor with a define state or value.

The running simulation, the one who triggers our system, is made by the **HAB Panel**, a specific panel created inside our OpenHab application, that makes us modify the state of those sensors that can trigger our application.



SMART HOME SAFETY

The main application is developed in **Java** and its functionality is described by the component and all the sequence diagrams in the above section. In these new version of application, we talked about a DBConnector component in order to have persistence in our data. We used an SQL database with a single table, containing the sensor values that we want to store and a timestamp of the storage action.



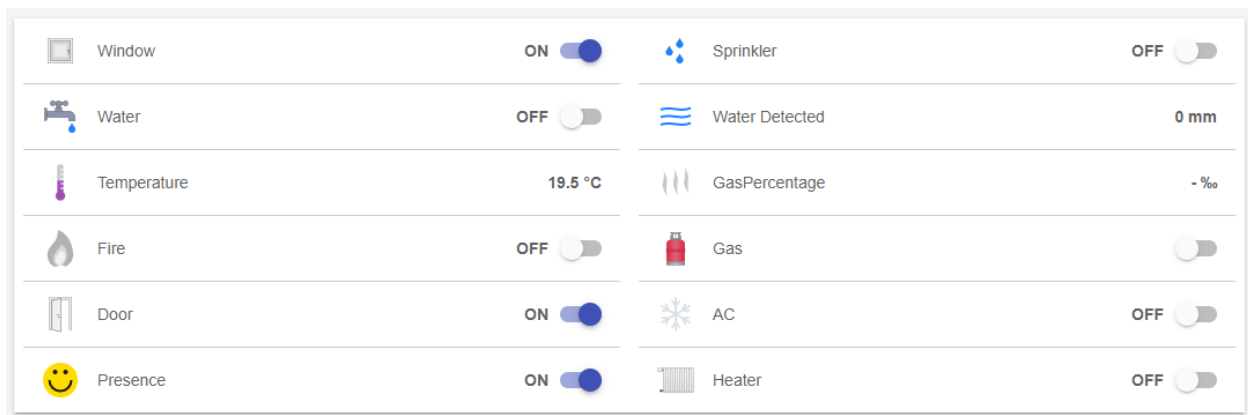
The image shows a screenshot of a database table definition for a table named 'seas values'. The table has a primary key 'id' of type 'int(11)'. It contains 27 other columns, all of type 'float', and a 'timestamp' column of type 'varchar(255)'. The columns are listed in a single column, with each row starting with a '#' symbol. The columns are: kadmin, kadmin, knmin, knmax, ladmin, ladmin, lnmin, lnmax, badmin, badmin, bnmin, bnmax, btdmin, btdmax, btnmin, btnmax, kmin, kmax, lmin, lmax, bmin, bmax, btmin, btmax, and timestamp.

id : int(11)
kadmin : float
kadmin : float
knmin : float
knmax : float
ladmin : float
ladmin : float
lnmin : float
lnmax : float
badmin : float
badmin : float
bnmin : float
bnmax : float
btdmin : float
btdmax : float
btnmin : float
btnmax : float
kmin : float
kmax : float
lmin : float
lmax : float
bmin : float
bmax : float
btmin : float
btmax : float
timestamp : varchar(255)

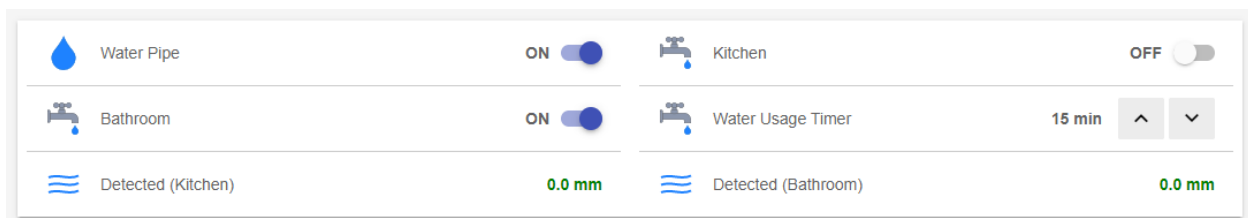
SMART HOME SAFETY

For the UI part we decided to use **OpenHab**, deciding thought to program and set up his environment all by the **code** and not using the visual console to set up all the different components. We coded sensor by sensor, room by room, linking them in our Java application through MQTT topics.

This is the UI that monitors the kitchen of our smart home.




This is the UI that monitors the kitchen of our smart home.





SMART HOME SAFETY


This is the UI section for the presence mode temperature settings.

Kitchen



 Min

17.0 °C


 

 Max



21.5 °C


 

Bathroom



 Min

17.0 °C


 

 Max



18.0 °C


 

Bedroom



 Min

16.5 °C


 

 Max



18.5 °C


 

Livingroom



 Min

15.5 °C


 


 Max

18.0 °C

Activate

 Presence Mode

- 

14