

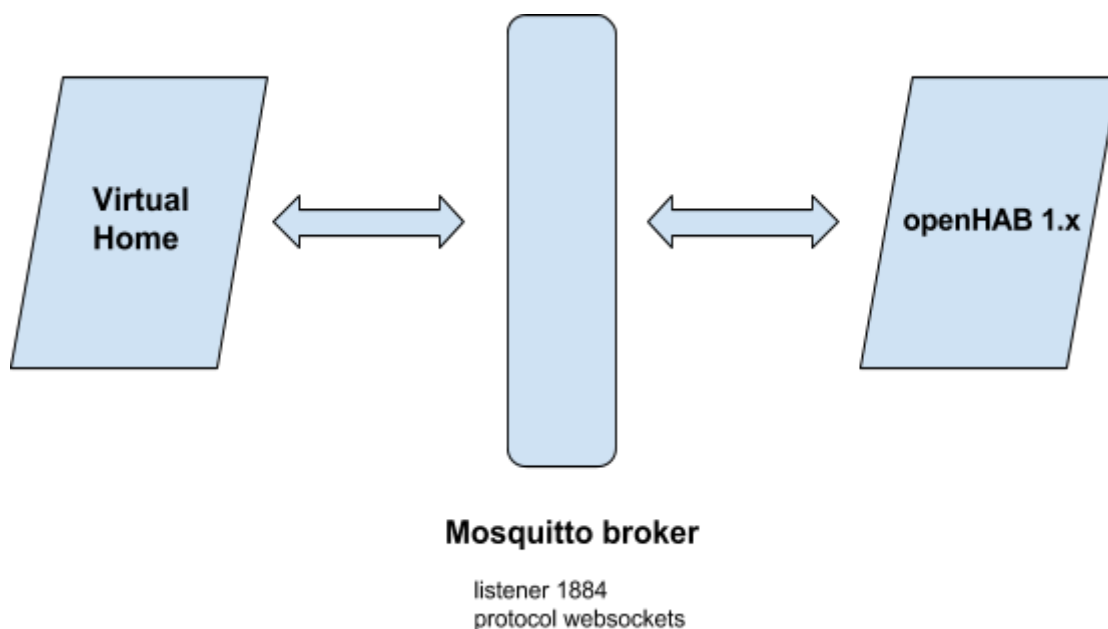
Software Engineering for Autonomous Systems - Smarties

Smarties is an Autonomous System project for a virtual home, which manages different actuators as sensors and effectors in order to reduce some energy consumptions, increase house security and take multiple decision according to its configuration. The idea of all these functionalities is give to the end user the full control of his/her house, since Empty House mode, Fresh House mode, Warm House mode, etc, can be activated at any moment.

Smarties is divided into two components such as openHAB Framework and Virtual Home that connect each other using a broker called Mosquitto, which is an Open Source message broker that implements the MQTT protocol versions 3.1 and 3.1.1. For more information about this broker, visit [Mosquitto.org](https://mosquitto.org).

Mosquitto

The steps to install and configure Mosquitto broker using Ubuntu 16.04.3 LTS as an operative system are shown in the github repository <https://github.com/carloselpapa10/smarties/tree/master/readme>. After the installation, we must activate the Websocket protocol on the port 1884 in order to establish the communication between Virtual Home and openHAB - this step is also explained in the repository.



Virtual Home

We decided to draw a virtual house as an interface with a series of items such as lights, motion and temperature sensors, windows and doors using HTML5 and CSS3. However, we used the Paho Javascript Client, which is an MQTT browser-based client library that uses WebSockets to connect to the Mosquitto broker.

Tests were successfully done using Google Chrome as a browser.



Specifications:

First of all we draw on paper the design and we gave programming names for all the items that are going to be controlled by the system then.

The house is divided by rooms and each room has motion sensor, lights, doors and windows with different amount from each other depending on the type of room it is.

The interface is responsible for showing the status of the items. However, we can turn ON/OFF an item at any moment and it will publish the change through MQTT protocol.

1. The lights on/off
2. The temperature if its normal or if it is high it will change the color of the temperature image.
3. Motion sensors are activated if they detect any movement.
4. Windows can be closed/opened.
5. Doors can be closed/opened.

The browsing page connects instantly and subscribes to all the topics related to the home as soon as you turn on the connection collects the data and status from the publisher and shows the results on the specified items instantly without any need of refreshing the page or a page refresher every certain amount of time

OpenHAB Framework

OpenHAB (Open Home Automation Bus) is an open source, technology agnostic home automation platform which runs as a center of your smart home. For further information visit [OpenHAB web site](#). We have used OpenHAB version 1.8.



Specifications:

- **Add ons:**

A brief description for each bundle used in Smarties below.

Org.openhab.action.mail-1.10.0: This bundle allows us to send an email using the command SendMail. We used this bundle when the presence and fire alarms are activated. The full description of the bundle is obtained [here](#).

Org.openhab.binding.mqtt-1.10.0: This is an essential bundle that enables the MQTT protocol to communicate with Virtual Home. Indeed, if this bundle is not found for openHAB, Smarties will not work at all. The full description is at [Mqtt Repository](#).

Org.openhab.persistence.mysql-1.10.0: This bundle allows us to store each item interactions to the database in such a way we can perform some machine learning algorithms using these data. The full description of this bundle is provided by this [page](#).

Org.openhab.binding.ntp-1.10.0: This bundle is used to query an NTP server you configure for the current time, within tens of milliseconds of accuracy. Items in openHAB will receive these updates. The full description is [here](#).

- **Configuration:**

We have used openHAB Designer to build the configuration of this project - It can be downloaded from the openHAB web site. The files created are my_demo.items, my_demo.rules, mydemo_sitemaps, and mysql.persist.

my_demo.items file: We created this file to define items and groups used in the Virtual Home as Temperatures, Motions, Lights, Windows and Doors.

```
Switch alarm_presence
Switch alarm_fire

Group Lights_Random
Group Lights[
Group Motions
Group Windows
Group Temperatures
Group Doors

Switch powerSaveMode <energy>
Switch safetyHouse <present>
Switch keepHouseFresh <settings>
Switch keepHouseWarm <settings>

Switch light_1 <light> (Lights) {mqtt=">[mybroker:
Switch light_2 <light> (Lights) {mqtt=">[mybroker:
Switch light_3 <light> (Lights) {mqtt=">[mybroker:
Switch light_4 <light> (Lights) {mqtt=">[mybroker:
Switch light_5 <light> (Lights) {mqtt=">[mybroker:

Switch motion_1 <siren> (Motions) {mqtt=">[mybroke
Switch motion_2 <siren> (Motions) {mqtt=">[mybroke
Switch motion_3 <siren> (Motions) {mqtt=">[mybroke
Switch motion_4 <siren> (Motions) {mqtt=">[mybroke
Switch motion_5 <siren> (Motions) {mqtt=">[mybroke
```

my_demo.rules file: Once items have been defined, we added a series of conditions or rules related with the High Level Goals of the project which is explained in the next section. A rule can be activated for an interaction of an item - turn ON/OFF a light or

for a certain time using a cron expression (more details about create a cron expression [here](#)).

```

/* SAFETY HOUSE HIGH LEVEL GOAL */
rule "Send email when detected a presence"
when
    Item Motions changed
then
    if(safetyHouse.state == ON){
        if(Motions.members.filter(i | i.state == ON).size >0 )
        {
            //Motions.members.forEach(i | say(i))
            if (alarm_presence.state != ON) {
                alarm_presence.postUpdate(ON)
            }
        }
    }
end

rule "Presence alarm_presence"
when
    Item alarm_presence received update ON
then
    // turn on all lights
    sendCommand(Lights,ON)

    // send an email
    sendMail("c.avendano10@gmail.com", "Detected a presence", "
end

```

mydemo_sitemaps: Here we draw the Smarties sitemap, where the items are shown and where the end user can interact with the system. Our sitemap contains General, Livingroom, Kitchen, Beedroom, Bathroom and Bed sections.

```

sitemap home label="Smarties" {
    Frame label="General" {
        Switch item=powerSaveMode label="Save Energy Consumption"
        Switch item=safetyHouse label="Empty House Mode"
        Switch item=keepHouseFresh label="Fresh House Mode"
        Switch item=keepHouseWarm label="Warm House Mode"
    }

    Frame label="LivingRoom" {
        Switch item=light_1 label="Light"
        Switch item=motion_1 label="Motion"
        Text item=temp_1 valuecolor=[>50="red",>25="orange",>15="g
        Switch item=door_1 label="Door"
    }

    Frame label="Kitchen" {
        Switch item=light_2 label="Light"
        Switch item=motion_2 label="Motion"
        Text item=temp_2 valuecolor=[>50="red",>25="orange",>15="g
        Switch item>window_1 label="Window 1"
        Switch item>window_2 label="Window 2"
    }
}

```

Openhab.cfg: This file contains the main openHAB configuration. Here, we specified the broker configuration as well as email details, persistence, etc.

- **High Level Goals:**

The High Level goals are implemented to give the owner of the house the opportunity to easily handle his smart home configuration to achieve certain effects as Saving energy, ensuring safety, keeping the house warm and the air inside fresh.

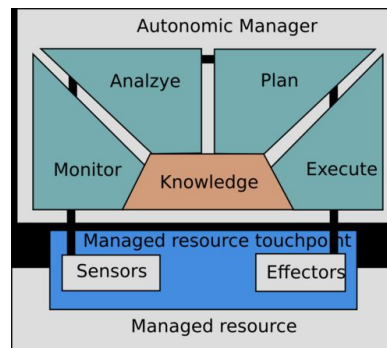
Those high level goals can be easily activated or deactivated

1. **Save energy consumption:** If this goal is activated, lights will automatically turn off after 1 minute of no motion detected in the respective room.
2. **Empty house mode:** When activating this mode, first all windows and doors will be closed, Motion sensors are set to the initial state of off and lights will be turned off. Following that the email alarm is activated, which will inform the owner by mail in case of any changes in doors, windows and motion detectors attached with a picture of the intruder.
3. **Fresh house mode:** As soon as the mode is activated and the system is not in empty house mode. All the windows will be opened hourly for 5 minutes. (For demonstration Matters every minute for 10 seconds).
4. **Warm house mode:** If active, all the windows will be closed in case the temperature in the house falls below a temperature configured in the variable tempMin.

Other general rules:

- Turn on the lights in rooms where motion is detected.
- Turn on the motion when a door is opened.
- Send a mail to the owner if the temperature in any room is above the variable tempMax warning about a possible fire.
- Activate safe house mode when no changes in motion sensors were stored in the database within the last 5 minutes.

The Mape-K loop



1. **Effectors:** Lights, Windows, Doors.
2. **Sensors:** Motions, Temperatures.
3. **Monitoring:** Information regarding states of Effectors and Sensors is gathered. Specific states (also called symptoms) activate Analysis via Rules.
4. **Analysis:** As soon as a Rule is triggered by monitoring Data it will analyze current states or past states from the Database and execute changes in Effectors or Sensors or change high Level goals
5. **Plan:** In our System "Smarties" past data is analyzed to establish if there are people home to ensure the safety of the house by activating the high level goal Empty House mode if needed.
6. **Execute:** If the analysis requests change, they will be executed according to the rules and plans defined.

Interaction with Smarties Database (Knowledge Base)

Our Knowledge Base stores all changes of effectors, sensors and plans in a MySQL Database to allow them being analyzed for establishing the need of changes in our high level goals. However, our Knowledge Base also stores the status items every one minute according to the specs in the persistence file. Next picture shows the configuration in openHAB.

```
mysql.persist
// persistence strategies have a name and a definition and are referred to in the "Items" section
Strategies {
    everyMinute : "0 * * * * ?"
    default = everyChange
}

Items {
    Lights*, Motions*, Windows*, Temperatures*, powerSaveMode* : strategy = everyChange, restoreOnStartup, everyMinute
}
```

Smarties Spring Boot

Smarties Spring Boot is an application that provides different Web Services Rest Protocol to the Virtual Home. The main service is called `noOneAtHome` and can be requested introducing the url <http://localhost:8090/noOneAtHome> once ***smarties-springboot.jar*** be executed. This functionality checks the last **N** minutes motion data into the database and then if there is not changes at all and Safety House mode is OFF, so this will turn ON immediately and an email will be sent to the owner. However, this functionality works between 6:00 am and 10:00 pm at day.

Smarties Spring Boot application provides a properties file in which we can configure our environment as we require it ranging from database url to port server number. Next picture shows the application.properties file.

```

1 spring.datasource.url=jdbc:mysql://localhost/openhab
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
5
6 server.port=8090
7
8 smarties.config.minutes = 5
9 smarties.config.hourMAX = 22
10 smarties.config.hourMIN = 6

```

Virtual Home application will be requesting every minute to `noOneAtHome` service. If nobody is at home during N amount of time it will get a true value, otherwise false. This value will be published on a topic named *myhouse/noOneAtHome* and an openHAB variable will get this value to be evaluated for a specific rule. Next picture shows the Smarties Spring Boot `noOneAtHome` service.

```

//@CrossOrigin(origins = "http://localhost:8090")
@RequestMapping(value = "/noOneAtHome", method = RequestMethod.GET, headers = "Accept=application/json")
public boolean motionActivity() {

    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Calendar cal = Calendar.getInstance();
    cal.add(Calendar.MINUTE, - minutes);

    String time = dateFormat.format(cal.getTime());

    if(cal.getTime().getHours() > hourMin && cal.getTime().getHours() < hourMax ) {

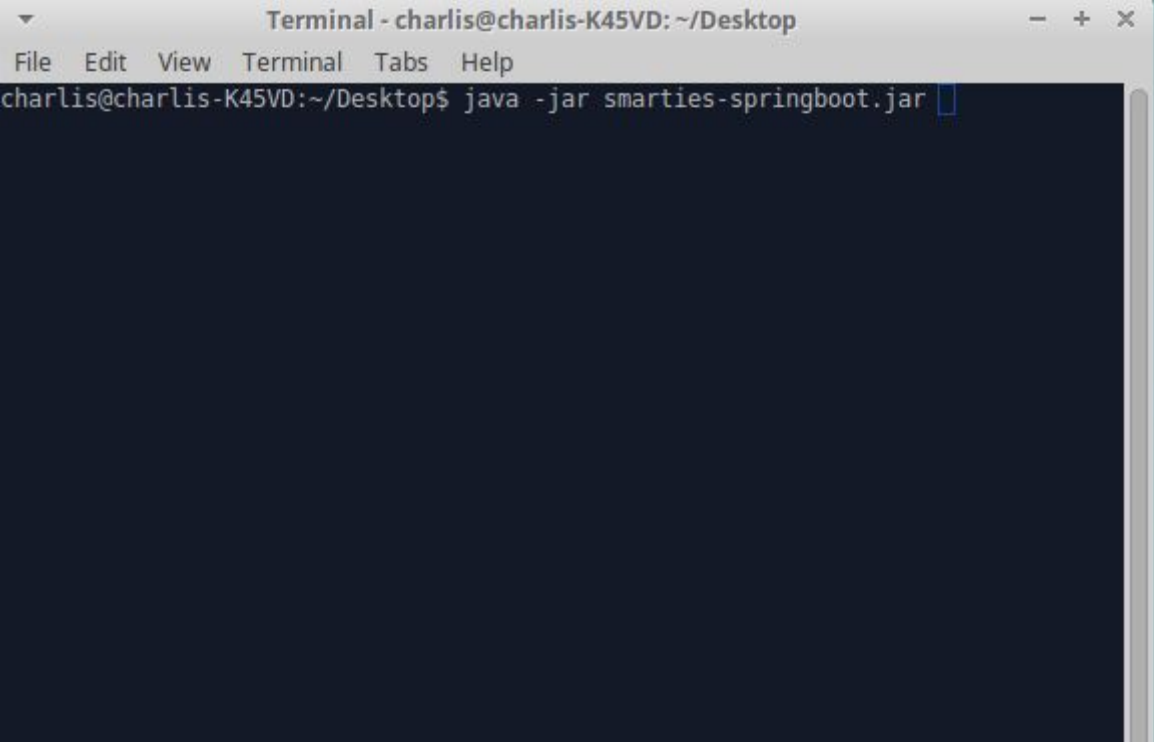
        if(this.smartiesMapper.checkMotion1Activity(time) == 0 &&
           this.smartiesMapper.checkMotion2Activity(time) == 0 &&
           this.smartiesMapper.checkMotion3Activity(time) == 0 &&
           this.smartiesMapper.checkMotion4Activity(time) == 0 &&
           this.smartiesMapper.checkMotion5Activity(time) == 0) {

            return true;
        }
    }

    return false;
}

```


Finally, to run this application we can open the console, finding the jar and inserting **java -jar smarties-springboot.jar**. After that, we can test the service typing on a browser the url <http://localhost:8090/noOneAtHome>.



```
Terminal - charlis@charlis-K45VD: ~/Desktop
File Edit View Terminal Tabs Help
charlis@charlis-K45VD:~/Desktop$ java -jar smarties-springboot.jar
```

