



TECHNICAL UNIVERSITY OF CRETE
DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

Cloud-based virtual building management system

Papadimitriou Nikolaos

Chania 2017

SUPERVISOR

KONSTANTINOS KALAITZAKIS

THESIS COMMITTEE

KOLOKOTSA DIONYSIA

KOUTROULIS EFTICHIOS

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου για τη συμπαράστασή τους καθ' όλη τη διάρκεια των σπουδών, τον κ. Καλαϊτζάκη που μου έδωσε τη δυνατότητα να ασχοληθώ με ένα πολύ ενδιαφέρον θέμα, τα μέλη του EMBER LAB για τη φιλοξενία, τη παρέα και τα ξενύχτια που είχαμε μαζί καθώς και τους φίλους μου για τις ωραίες στιγμές που ζω στα Χανιά

ABSTRACT

Energy management and Building Management System (BMS) are playing an important role in our days. The need for lowering energy consumption and saving money is urgent in our times. In addition the emergence of Internet of things (IOT) for facilitating people's life has arisen the need for computer supervising. In this thesis we deal with a cloud based virtual management system for the Ember's laboratory.

The Ember laboratory has light sensor, CO₂ sensor, Humidity sensor (RH) , Thermistor, contact sensor and presence sensor and by using its current solution, it manages their data and extracts information using fuzzy logic algorithms. The need behind this thesis was the absence of a hardware/protocol agnostic platform that could accept all kinds of sensors regardless their communication protocol, an open source protocol that is able to be expanded to satisfy new requirements and also has an active community that could help in finding solutions.

In our thesis we used the openHAB platform. It is an open source platform that can support a wide range of devices' protocols, has a great functionality and an online community. We had to configure openHAB to accept connection with the sensors, rewrite all algorithms that were created by the EMBER laboratory staff in the accepted programming language and make a graphical interface for an easier interaction and graphs for better understanding.

For making a universal access to our platform we installed it on university's hosting server and created a web application that can manage it. The application was created keeping in mind that we want it to be expandable, easy to use, adjustable to all accessing devices and secure.

This thesis works properly and can be expanded to manage more rooms, use more technologies for better management and energy save by using forecast and also be installed in embedded systems.

Contents

1.	Introduction and state of the art	8
1.1	BMS and energy management	8
1.2	Review of existing solutions	12
1.2.1	Virtual Operator®	12
1.2.2	Honeywell	13
1.2.3	SmartStruxure solution	13
1.2.4	Beckhoff Building Automation	14
1.2.5	Openremote	14
1.2.6	Alerton	15
1.2.7	OpenHAB	15
1.3	Purpose of this thesis	15
2.	EMBER's Infrastructure	18
2.1	Sensors and Controller	19
2.2	Current solution	25
2.3	Fuzzy logic	27
3.	OpenHAB	29
3.1	Introduction	30
3.2	Analyzing its repository	32
3.3	HABmin and openHAB designer	58
4.	Configuring OpenHAB runtime	62
4.1	Adding new devices	62
4.2	Creating rules	65
4.2.1	PMV	65
4.2.2	Fuzzy rules	71
4.2.3	Rule HVAC set-point change	76
4.2.4	Rule Fuzzy Ventilation	81
4.2.5	Rule Fuzzy Lights	86
4.3	Sitemap	90
4.4	Security	92
5.	Web Site	93
5.1	Setting up the server	93

5.2	Web content	96
5.2.1	Test_input.php	100
5.2.2	Config.php	100
5.2.3	session.php	100
5.2.4	index.php	101
5.2.5	login_page.php	102
5.2.6	Mainmenu.php	103
5.2.7	openhab_page.php	105
5.2.8	habmin_page.php	107
5.2.9	management_page.php	109
5.2.10	addauser.php	117
5.2.11	changepassword.php	118
5.2.12	deluser.php	118
5.2.13	logout.php	118
5.3	Auto-starting OpenHAB	118
5.4	Self signed	119
6.	Results and discussions	121
7.	Conclusion and future prospects	122
7.1	Future system development	122
8.	References	123

1. Introduction and state of the art

The first section of this thesis contains an introduction to basic concepts of a building's management system, a presentation of some of the most known companies dealing with energy management as well as their products and finally the aim of this thesis and the necessities it has to fulfill.

Energy management has become for the last few years a rapidly development sector. This is because of the awareness of ecological balance disturbance and the impact in our everyday life and the cost saving. Many companies are dealing with this sector but there are not providing background for research purposes. That is the reason that in this thesis we decided to search for a universal platform that can accept every known sensor regardless its communication protocol. It will execute all existing algorithms created for energy management by the laboratory's staff member. This platform would be hosted on a cloud server rather than a physical server running in a lab in order to save energy.

1.1 BMS and energy management

Buildings are one of the fastest growing energy consuming sectors. According to EIA (U.S Energy Information Administration) Energy consumed in the buildings sector consists of residential and commercial end users and accounts for 20.1% of the total delivered energy

consumed worldwide. In the *International Energy Outlook 2016* (IEO2016) Reference case, delivered energy consumption in buildings worldwide increases by an average of 1.5%/year from 2012 to 2040 [11]. In order to reduce energy consumption and carbon footprint of buildings, computer based control systems are installed which control and monitor the building's mechanical and electrical equipment such as ventilation, lighting, power systems, fire systems, and security systems. These systems are called **Building Management System (BMS)** otherwise known as a **Building Automation System (BAS)**. The main objectives of BMS as International Energy Agency (IEA 1997) identifies are: a) to provide a healthy and pleasant indoor climate b) to ensure the safety of the user and the owner and c) to ensure economical running of the building in respect of both energy and personnel. Other benefits that a BMS offers are

- Possibility of individual room control
- Effective monitoring and targeting of energy consumption
- Ease of information availability
- Computerized maintenance scheduling

A BMS consists of software and hardware; the software program, usually configured in a hierarchical manner, can be proprietary, using such protocols as C-Bus, Profibus, and so on. There are also produced BMSs that integrate using Internet protocols and open standards such as DeviceNet, SOAP, XML, BACnet, LonWorks and Modbus. The program's aim is to provide a graphical user interface so that a user can easily manage and supervise his equipment and export conclusions about the current building's economic state and work environment. As a result he would be able to create, via the software, the appropriate rules for improving situation, better performance. Concerning the hardware BMS has Sensors, Controls and Controller. *Sensor* is an object whose purpose is to detect events or changes in its environment and sends the information to the controller. It can be analog or digital. Analog inputs are used to read a variable measurement. Examples are temperature, humidity and pressure sensors which could be thermistor, 4–20 mA, 0–10 volt or platinum resistance thermometer (resistance temperature detector), or wireless sensors. A digital input indicates if a device is turned on or not - however it was detected. Some examples of an inherently digital input would be a 24 V DC/AC signal, current switch, an air flow switch, or a volta-free relay contact (dry contact). Digital inputs could also be pulse type inputs counting the frequency of pulses over a given period of time. An

example is a turbine flow meter transmitting rotation data as a frequency of pulses to an input. *Controls* are outputs and can be analog or digital. Analog outputs which control the speed or position of a device, such as a variable frequency drive, an I-P (current to pneumatics) transducer, or a valve or damper actuator. An example is a hot water valve opening up 25% to maintain a setpoint. Digital outputs are used to open and close relays and switches as well as drive a load upon command. An example would be to turn on the parking lot lights when a photocell indicates it is dark outside. *Controller* are essentially small, purpose-built computers with input and output capabilities that come in a range of sizes and capabilities to control devices commonly found in buildings, and to control sub-networks of controllers. Inputs allow a controller to read temperature, humidity, pressure, current flow, air flow, and other essential factors. The outputs allow the controller to send command and control signals to slave devices, and to other parts of the system. Inputs and outputs can be either digital or analog. Digital outputs are also sometimes called discrete depending on manufacturer. Controllers used for building automation can be grouped in three categories: programmable logic controllers (PLCs), system/network controllers, and terminal unit controllers. However an additional device can also exist in order to integrate third-party systems (e.g. a stand-alone AC system) into a central building automation system. Terminal unit controllers usually are suited for control of lighting and/or simpler devices such as a package rooftop unit, heat pump, VAV box, fan coil, etc. The installer typically selects one of the available pre-programmed personalities best suited to the device to be controlled, and does not have to create new control logic[\[2\]](#) .

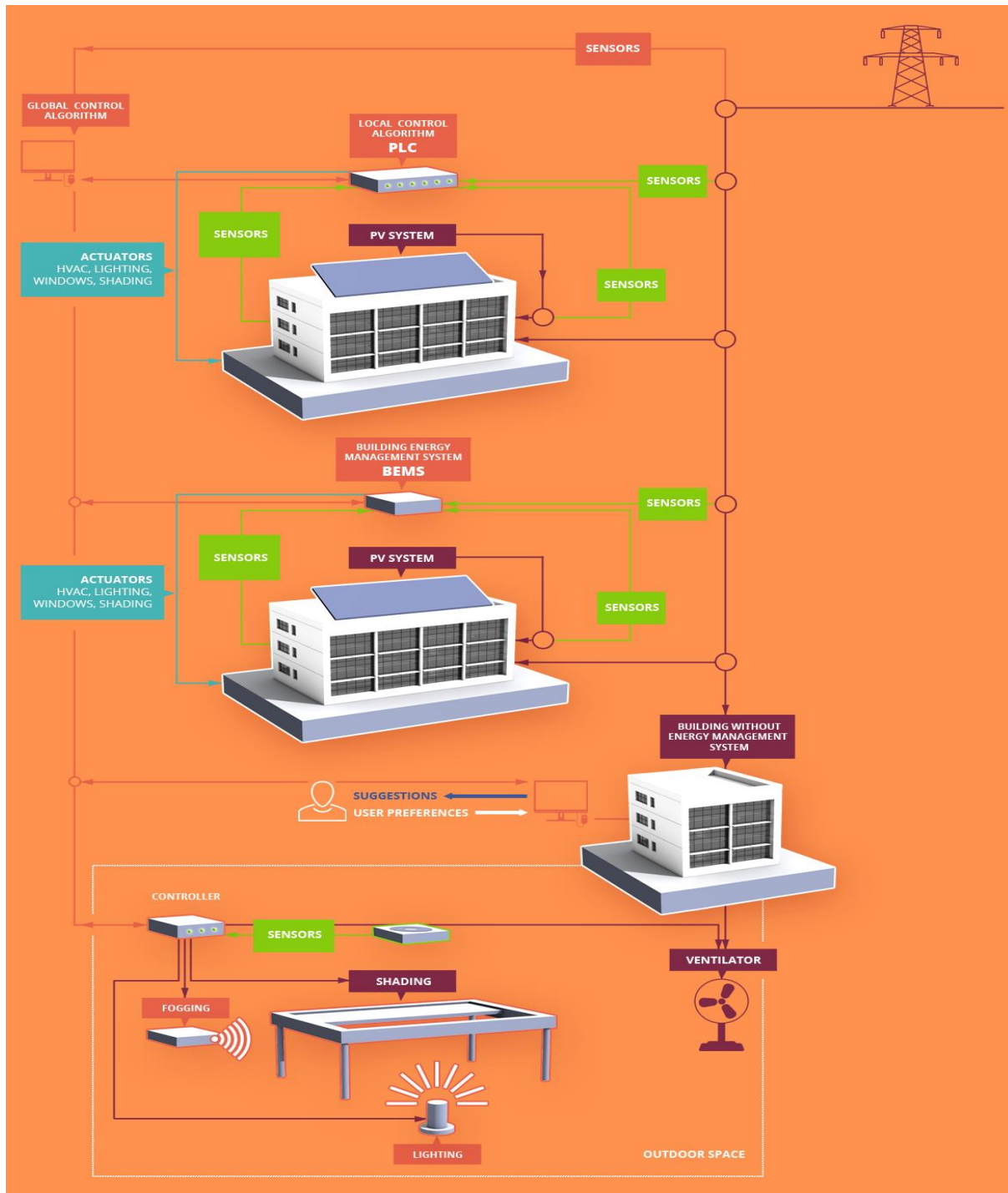


Figure 1.1 CampIt source: <http://www.campit.gr/>

1.2 Review of existing solutions

As an innovative field of technology, lots of companies have entered in automation software development. A customer would be able to choose between a great amount of products that could satisfy his requirements. There are many upon payment products that are fully customized for his needs as long as free ones that the customer should modify according to his preferences. In this section, a list of the best known products will be presented. Products are not fully described because companies don't reveal all product qualifications or prices. Rather they suggest getting in contact with them so they can propose a solution that fits your preferences.

1.2.1 Virtual Operator®

VEMS is a part of Matex Controls Company and it is established in Poland.



Figure 1.2 VEMS logo

Virtual Operator® is their solution into building management

System software. Using artificial intelligence technology

Virtual Operator connects to and monitors different types of building management systems. It monitors any changes occurring in building management systems and takes actions as required. A dedicated algorithm repeats the examination cycle no less than 5 times per minute. It integrates with any type of BMS hardware regardless of its manufacturer. Product is not free of charge. [3]

1.2.2 Honeywell

Honeywell is a company that offers a scalable range of building management and intelligent building solutions and services — from individual subsystems to fully integrated intelligent buildings. Their product supports all major global



Figure 1.3 Honeywell logo

industry open standards as long as it offers Energy management dashboards and automated control algorithms. Honeywell is mainly targeting to big scale building like hospitals, airports, commercial buildings etc. Product is not free of charge. [\[4\]](#)

1.2.3 SmartStruxure solution

SmartStruxure solution is Schneider Electric's proposal on building management. According to



Figure 1.4 Schneider Electric logo

customer's needs, Schneider Electric has a great variety of products to satisfy these needs. As it comes to hardware, Schneider electric has automation servers that are factory programmed with StruxureWare Building Operation software and work as server and controller. There are also modules that can connect with the servers for supplying them with power or increasing the number of analog or digital I/O. Customer is also able to buy BACnet or LON controllers for an existing facility that has stand-alone controller and server. For the software side Schneider electric offers solution for servers that collects site-wide data for aggregation and archiving, pc workstations which is a fully featured environment for operating and administering all aspects of the software, as long as mobile and tablet applications.

Products are not free of charge.[\[5\]](#)

1.2.4 Beckhoff Building Automation



Figure 1.5 Beckhoff logo

medium-sized, internationally acting company located in Verl, Westphalia, Germany. Beckhoff offers a single control platform for all building systems and a modular range of software and hardware components which can provide security in every phase from engineering through to unplanned changes or enhancement. Their software product, called TwinCAT Building Automation promises maximum flexibility, custom-fit scalability and reliable functional integrity. Products are not free of charge. [6]

1.2.5 Openremote

OpenRemote is an open source project, started back in 2009. Its ambition was to overcome the challenges of integration between many different protocols and solutions available for home



Figure 1.6 Openremote logo

automation. Its products offer solutions apart from home automation to several other application domains, ranging from building integration, to healthcare, hospitality, entertainment, and public spaces. Openremote project offers also Openremote Designer, a tool for designing the graphic user interface according to your needs, cloud services and consultancy during all the integration process. They also host an online community. Openremote products are free for private use, educational purpose, and trials. [7]

1.2.6 Alerton



Figure 1.7 Alerton logo

Alerton's solution on building management systems is called BACtalk® system and it has already being installed into several application domain like Healthcare, Airports, Colleges and Universities etc. It is based on Bacnet standard. Their controller AZW-5000

provides communication with wireless sensors which can be self-healing ensuring reliability. Products are not free of charge. [\[8\]](#)

1.2.7 OpenHAB



Figure 1.8 OpenHAB logo

OpenHAB is an open source project on home automation. It started in 2010 and aims at providing a universal integration platform for all things around home automation. The project splits into two parts, the openHAB runtime which is a java framework that deals with the communication between the sensors and the server and visualize the results, and the openHAB designer which is an application for configuring the openHAB runtime. openHAB offers an online community and cloud hosting. Products are free of charge. [\[9\]](#)

1.3 Purpose of this thesis

This thesis was scheduled as a way to satisfy the necessity of an online platform that can communicate with all sensors, regardless of the existence of different communication protocols.

A platform being hardware/protocol agnostic, gives user the opportunity to be able to choose between a greater amount of products and choose wisely one that is value for its money, otherwise customer would be forced to buy from a specific vendor cause products and services that are made from different companies are not created to talk to each other. In economic words, this is called market fragmentation. The definition given from businessdictionary.com is: “Market fragmentation is the emergence of new segments (in a previously homogeneous market) which have their own distinct needs, requirements, and preferences. These fragments reduce the effectiveness of mass marketing techniques and erode brand loyalty”. [\[10\]](#) So thesis aimed at lowering the total cost but also creating a universal platform. This means that our solution is meant be used in existing facilities irrespectively of the current hardware. In addition there is the need of storing the sensors data and use them as input in smart algorithms and fuzzy functions for extracting further knowledge about BMS and energy management. The project is hosted on a cloud server so that there is no necessity of running a physical server in the laboratory thus cost is dropped. Web hosting provides mutable resources on demand on Infrastructure as a service (IaaS) layer. Therefore laboratory user can change any of the fundamental computing resources like processing, storage, networks according to the contemporary occasion via internet and not being demanded of buying extra hard drive or process unit (cpu).

In web development one of the major requirements is safety, not only for whom could access data but also for whom could make variations on hardware or software configurations. So our web application should have authentication but also ssl certification for protection of the privacy and integrity of the exchanged data.

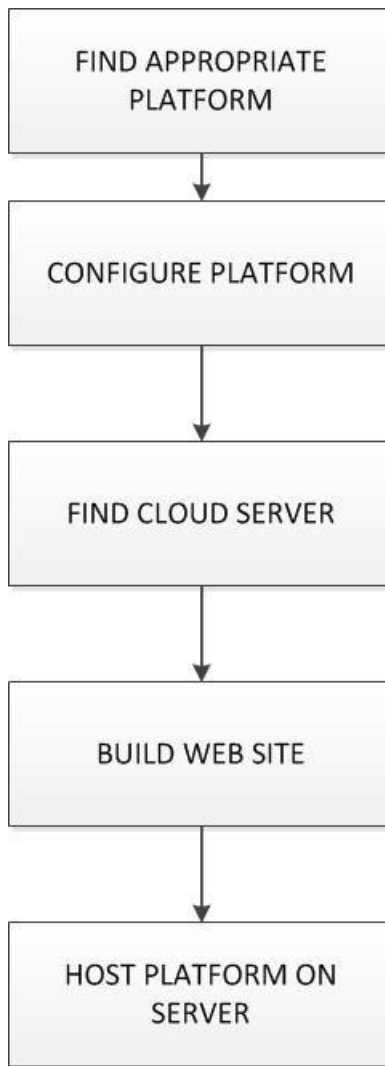


Figure 1.9 Steps followed for creating this thesis

2. EMBER's Infrastructure

EMBER lab, abbreviation for Energy Management in the Built Environment Research lab is a research lab residing in School of Environmental Engineering (ENVENG) in the Technical University of Crete (TUC). EMBER is focusing on applied research in the following topics

- Energy efficiency in buildings and built environment.
- Indoor Environmental Quality and Energy Efficiency.
- Thermal Comfort, Visual Comfort and Indoor Air Quality.
- Performance indicators.
- Green Buildings.
- Zero carbon emissions buildings.
- Urban environment and Climate Change.
- Urban heat island studies and urban heat island mitigation techniques.
- Energy Management Systems.
- Monitoring and Control of indoor environmental conditions.
- Design and integration of smart systems in buildings and urban environment.

In this section we are describing the EMBER's infrastructure (sensors and controller) that was used for this thesis and we analyze the solution currently in use. In the last part we explain what

fuzzy logic is, as it is a type of logic which plays an important role in the lab's decision making rules.

2.1 Sensors and Controller

EMBER laboratory has six (6) sensors which communicate with a controller. The sensors are light sensor, CO₂ sensor, Humidity sensor (RH), Thermistor, contact sensor and presence sensor.



Figure 2.1 Presence sensor



Figure 2.2 light sensor ,CO2 sensor, Humidity sensor (RH) , Thermistor embedded

The controller is Schneider Electric UN2-0I4-00. It enables the control, monitoring, and management of entire sites via StruxureWare Building Expert. It has 6 inputs, 6 outputs but also supports wireless connection. It can communicate via ZigBee [11]. ,Modbus [12] , BACnet ® , EcoStruxure ® and oBIX ® [13]. Controller supports hosting Building Expert web site because of the embedded web server [14].



Figure 2.3 Schneider electric controller

BACnet is the protocol used for the communication between controller and server. BACnet is the acronym for Building Automation and Control NETWORKS and as its name indicates, it allows communication of building automation and control systems for applications such as heating, ventilating, and air-conditioning control (HVAC), lighting control, access control, and fire detection systems and their associated equipment. BACnet protocol began in June, 1987, in

Nashville, Tennessee. It is developed under the auspices of the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) and it is an American national standard, a European standard, a national standard in more than 30 countries, and an ISO global standard.

So as a data communication protocol, BACnet is simply an agreed-upon set of rules that apply to a computer's hardware and software. The rules apply to all aspects of data communication like electrical signaling, addressing, error checking etc. BACnet works with objects. As object we define s a logical representation used in the BACnet protocol like a physical device (device objects) , a temperature input (analog input), a relay control (binary output) etc. By using objects we have a standard set of communication rules - a common “language” - so that each device "looks the same" on the wire rather than having every of each “speak” a different language. Each object has a set of properties, such as the object's name, type, and present value, that describe its behavior or govern its operation. Every property has a name and a value. In the next figure we can see properties of a light sensor.

Analog Input

Save

Import

Export

Description: LightSensor

Name: LightSensor (AI1)

Node: N0057AA

Object BACnet Id: AI1

Input Type: Volts

Value: 267.51

Units: Luxes

Configuration: lightSensorConfiguration (AIC1)

Calibration

Gain: 1

Offset: 0

Figure 2.4 Ember's light sensor and its properties source

In the table below, there are cited all standard object types

Binary Input	Multi-state Input	File
Binary Output	Multi-state Output	Program
Binary Value	Multi-state Value	Schedule
Analog Input	Loop	Trend Log
Analog Output	Calendar	Group
Analog Value	Notification Class	Event
Averaging	Command	Enrollment
LifeSafetyZone	LifeSafetyPoint	

Table 2.1 Object types

Vendors are also able to create their own custom objects.

Communications between building automation and control devices is based on a “Client-Server” model. Information exchange between objects is provided by services. . Services are used to perform reads, writes, and I/O. The object that provides the service is a server and the object that requests the service is the client. Most objects can be both a server and a client, depending on the system's needs. Next table shows all possible services and their description.

SERVICE	DESCRIPTION
AddListElement	Adds one or more items to a property that is a list.
RemoveListElement	Removes one or more items from a property that is a list.
CreateObject	Used to create a new instance of an object in the serving device.
DeleteObject	Used to delete a particular object in the serving device.
ReadProperty	Returns a value of one property of one object.
ReadPropertyConditional	Returns the values of multiple properties in multiple objects.
ReadPropertyMultiple	Returns the values of multiple properties of multiple objects.
WriteProperty	Writes a value to one property of one object.

WritePropertyMultiple	Writes values to multiple properties of multiple objects.
-----------------------	---

Table 2.2 Services between building automation software and control device

In the next figure we can see the BACnet architecture next to the four layers from the Open Systems Interconnection model (OSI model) [15].

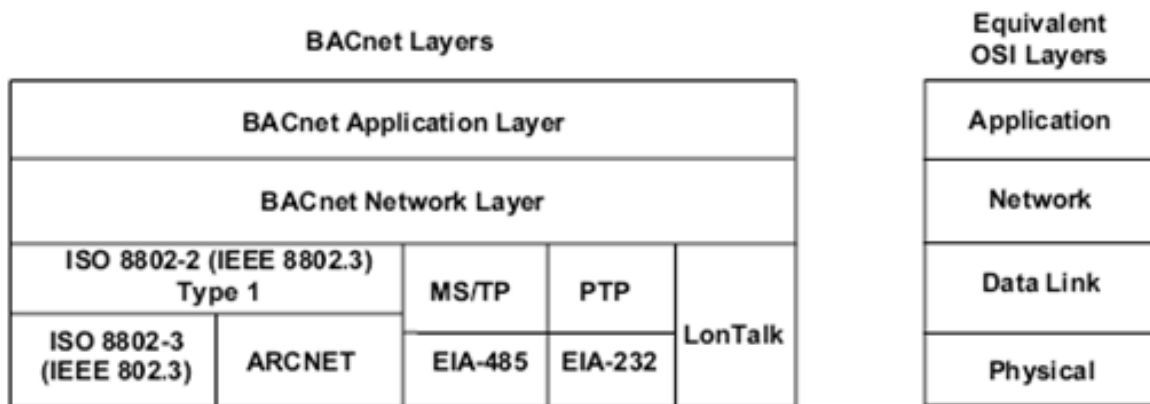


Figure 2.5 BACnet layers source: <http://www.chipkin.com/bacnet-architecture/> BACnet Architecture

A great issue BACnet is dealing with is the lack of certification. Security in the form of authentication would negate the whole reason behind BACnet; that is, an open protocol that is accessible by all makers. As a result users would end up in the quandary of getting stuck with one manufacturer or dealer, or replace the works. Thus security lies in user's regulation of his network.

2.2 Current solution

EMBER laboratory has chosen SmartStruxure™ Lite solution by Schneider-electric. As stated in section [1.2.3](#) it is a building management system for small- to medium-sized buildings. SmartStruxure™ Lite solution includes all sensors and controller described in section [2.1](#) and web application hosted in controller's embedded web server.

The web application is called Building Expert 2.15.1. The application offers exploration of all devices and energy meters, their status but also exploration of the existing scripts. Scripts are written in Lua[16], a lightweight embeddable scripting language. The scripts were written in the context of CampIT, a project for energy management in university campuses [17]. Some of the scripts are

- PMV function
- Lights controlled with fuzzy logic
- HVAC setpoint calculated with fuzzy logic etc

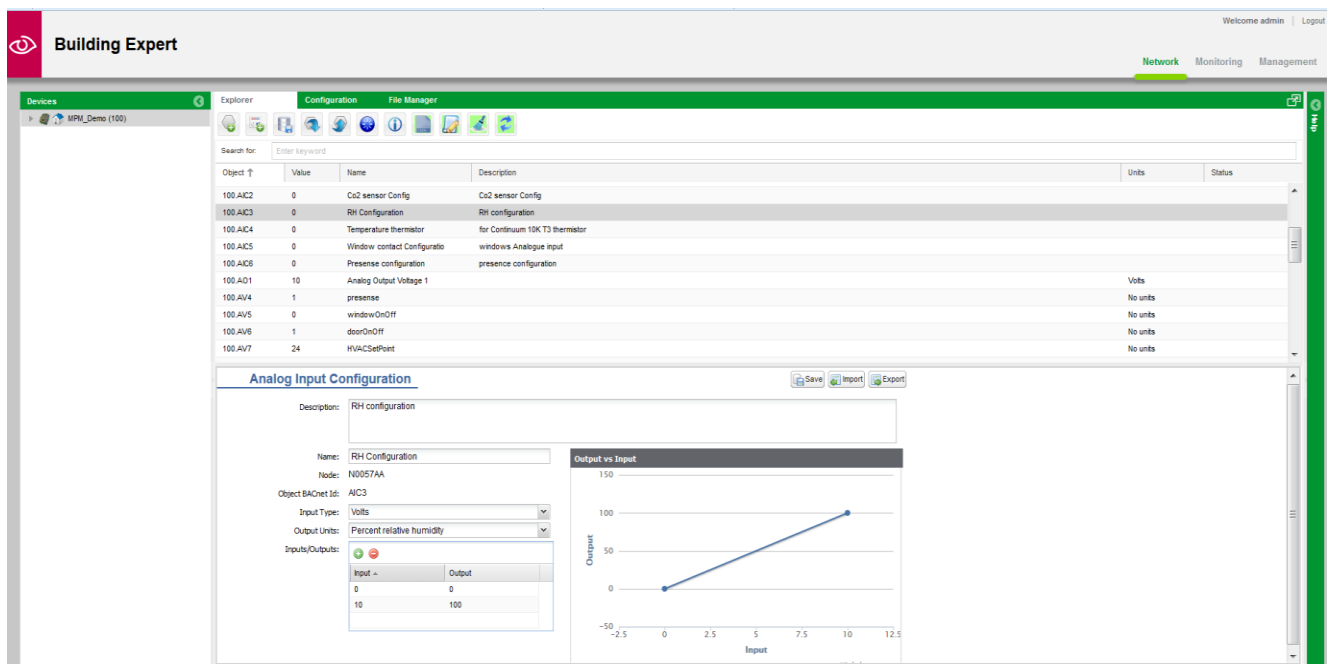


Figure 2.6 Screenshot from Building Expert software

In the application there is also the ability of configuring and managing files and devices as well as importing and exporting databases. This ability is restricted by the role a user has. There are three roles: Admin, Maintenance and user. Each one of these roles has its own permissions. Admin, shortening for Administrator, is the most significant role that has access to all of the application's features. Admin can also configure permissions of each role and can add new users.

2.3 Fuzzy logic

Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based.

It is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. The idea of fuzzy logic was advanced by Dr. Lotfi Zadeh of the University of California at Berkeley in the 1960s while working on a way of how computers could understand human language as it cannot be translated into the absolute terms of 0 or 1. In some way, fuzzy logic mimics the way human brain works in decision making as many real world decisions contain high levels of uncertainty which needs to be taken into account. For example we can estimate that weather is hot without any precise numbers or estimate the size of a house without measurement.

Fuzzy logic relies on fuzzy sets and fuzzy rules. Fuzzy sets are group of related items which belong to the set to different degrees. An example for making clear this is considering the days comprising weekend. Most people agree that Saturday and Sunday belong but what about Friday? It feels like belonging to weekends but also feels like being excluded. This vagueness is what fuzzy is examining and is based on individual perceptions and cultural background.

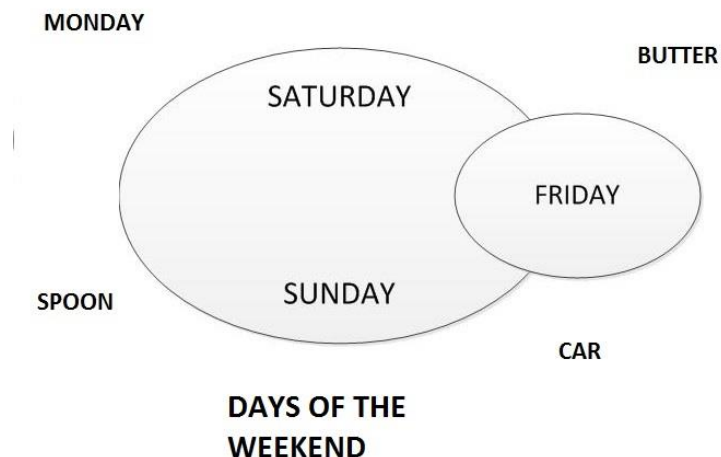


Figure 2.7 Fuzzy logic example

The curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1 is called Membership function (MF). Some of the most are shown in figure below

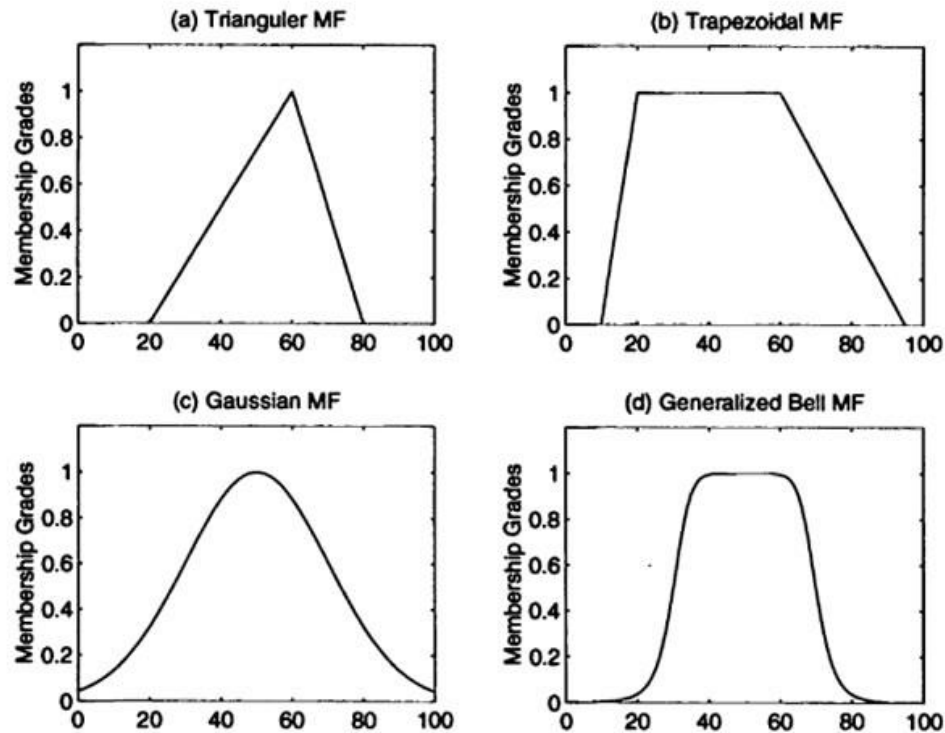


Figure 2.8 Membership functions source: <http://researchhubs.com/post/engineering/fuzzy-system/fuzzy-membership-function.html>

Fuzzy rules are rules that combine knowledge we get from fuzzy sets to make decisions. They are defined in the form: **IF** x is A **THEN** y is B where x and y are linguistic variables.

The utilization of linguistic variables, fuzzy control rules, and approximate reasoning provides a means to incorporate human expert experience in designing a fuzzy logic controller (FCL). In the figure below we can see a typical architecture of FCL

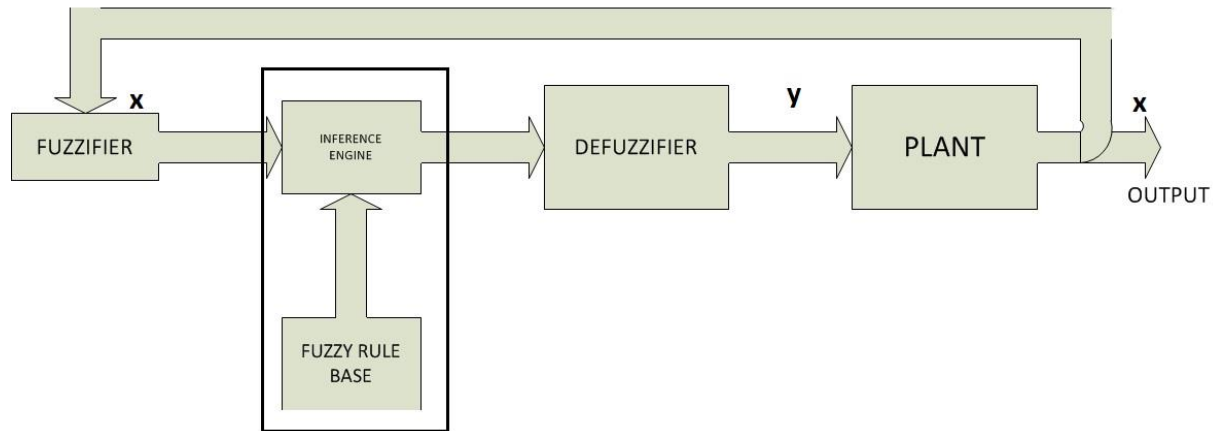


Figure 2.9 FCL architecture

As we can see a FCL comprises of **Fuzzifier**, **Fuzzy rule base**, **Inference engine** and a **Defuzzifier**. The fuzzifier transforms data into linguistic variables (fuzzy sets).

The fuzzy rule base stores the empirical knowledge of the operation of the process of the domain experts. The Inference engine is an engine simulating human decision making by performing approximate reasoning to achieve a desired control strategy. The defuzzifier is used to form a nonfuzzy decision or control action from the Inference engine [18].

3. OpenHAB

OpenHAB is the chosen platform for this thesis. Criteria for selecting a platform from the existing solutions that were presented in section [1.2](#) were

- Free of charge
- Open-source

Platforms that meet the criteria were OpenHAB and Open remote. OpenHAB's advantage over Openremote is having a bigger and more active community. It is also strictly free rather than the payment options that Openremote has.

In this sections, OpenHAB is presented and analyzed for its components. In the last subsection, there is a reference made to HABmin and OpenHAB designer, tools for configuring Openhab runtime.

3.1 Introduction



Figure 3.1 OpenHAB logo

The open Home Automation Bus (openHAB) is an open source, technology-agnostic home automation platform. As a project its initial release was in 2010 and it is still active in development. It is written in Java with an OSGi architecture which allows adding and removing functionality during runtime without stopping the services. The implementation of OSGI is called equinox [19]. As a pure Java solution, OpenHAB is designed to run on a variety of operating systems (Linux, Windows, OS X) and hardware configurations from PC's to micro-computers or any other device as long as the system is capable of running a java virtual machine (JVM) [20].

Being hardware/protocol agnostic, openHAB allows users to integrate and connect a variety of devices from classical home automation systems, such as KNX, Z-Wave, Insteon, EnOcean, to new Internet of Things (IoT) gadgets and devices, such as Sonos[21], Nest Labs[22], Philips Hue[23] and custom built microprocessor nodes and sensors.

In OpenHAB's core, as a web server, there is Jetty which is a Java HTTP (Web) server and Java Servlet container [24]. Jetty allows user to interact with OpenHAB via a web browser or smartphone application. There is a great variety of user interfaces (UI) to choose from apart from the Classic UI which is involved in the original distribution. Some of the most common UIs are

- Classic UI — classic Web interface
- Android Client — HABDroid is an open source Android user interface for openHAB
- Greent UI
- Apple iOS Client
- CometVisu
- Habmin

Through the interface user can control OpenHAB by creating automation rules or supervise the whole automation system [25]. There is also RESTful APIs that allow communication with other systems by sending data, status updates permitting access to items and their states [26].

As a project openHAB splits into two parts:

- openHAB Runtime
- openHAB Designer

OpenHAB Runtime is the package that will run on the server and it provides all the platform's functionality from connecting to sensors to showing the results on the screen. More information on [3.2](#)

OpenHAB designer is the configuration tool for the openhab-runtime. It is a user friendly editor ,eclipse-based [27] that helps with the project management. More information on [3.3](#)

OpenHAB can be downloaded from the official OpenHAB site

<http://www.openhab.org/downloads.html>

Or from Github

<https://github.com/openhab>

3.2 Analyzing its repository

After downloading and extracting the .zip file of the openHAB runtime user can see the repository shown in next figure.














Name	Date modified	Type	Size
 addons	5/22/2016 9:24 PM	File folder	
 configurations	5/22/2016 9:24 PM	File folder	
 contexts	5/22/2016 9:24 PM	File folder	
 etc	5/22/2016 9:24 PM	File folder	
 server	5/22/2016 9:44 PM	File folder	
 sounds	5/22/2016 9:24 PM	File folder	
 webapps	5/22/2016 9:24 PM	File folder	
 LICENSE	5/22/2016 9:24 PM	TXT File	11 KB
 README	5/22/2016 9:24 PM	TXT File	1 KB
 start	5/22/2016 9:24 PM	Windows Batch File	1 KB
 start	5/22/2016 9:24 PM	Shell Script	1 KB
 start_debug	5/22/2016 9:24 PM	Windows Batch File	2 KB
 start_debug	5/22/2016 9:24 PM	Shell Script	2 KB

Figure 3.2 OpenHAB runtime repository

In the ‘addons’ folder user can add modules that can extend openHAB’s functionality. These modules are called bindings and are .jar files. Each different protocol or device needs its own module. So if user wants to include a device which uses KNX protocol has to add in the ‘addon’ folder the appropriate module. Apart for communication with devices and sensors , bindings offer getting time from NTP servers, server monitoring, getting data from the internet and many others. User can download all available and official addons from openHAB's download page [28] or download separately from [openHAB’ github page \[29\]](#).

Some of the most known bindings are:

- [KNX binding](#): Bind your existing KNX hardware through serial or IP interfaces [30]
- [One Wire Binding](#): Bind your 1-wire sensors [31]
- [Bluetooth Binding](#): Bind your items to the bluetooth neighbourhood [32]
- [Wake-on-LAN binding](#): Send WoL packets when using a switch in openHAB [33]
- [Network Health Binding](#): Monitor your server availability [34]
- [Exec Binding](#): Execute any command on the host [35]
- [Serial Binding](#): Access devices connected via USB/RS-232 [35]
- [Http Binding](#): Get data from the internet or send out requests [36]
- [Fritz Box Binding](#): React on call notifications from your !FritzBox [37]
- [Ntp Binding](#): Get the precise time from NTP servers [38]
- [Mpd Binding](#): Control music playback on Music Player Deamon (MPD) clients [39]
- [VDR binding](#): Control your VDR and get notifications on your TV screen [40]
- [Asterisk Binding](#): React on call notifications from your Asterisk server [41]
- FreeTTS: Add Text-to-Speech support using FreeTTS. Not required on Mac as openHAB uses native TTS support there.[<https://github.com/openhab/openhab1-addons/wiki/Addons>]

The configuration of the bindings takes place in the ‘configurations’ folder and especially in the ‘items’ sub-folder and the .cfg file. Users can also create their own bindings if there isn’t something that can satisfy their demands. For further information on creating bindings user can find github contribution page [42].

The ‘contexts’ folder is used by the Jetty server and in ‘etc’ folder are files that configure Jetty.

The 'server' folder contains the core of the runtime as it has all files consisting equinox, OSGI implementation and Jetty server. In 'sounds' folder are stored music files (.mp3,wav) that are used in our system and in 'webapps' folder are stored images used in our web application.

'LICENSE.txt' contains legal terms of using openHAB and 'README.txt' contains a link that redirects user to openHAB's page in github for accessing setup and configuration instructions.

'start' and 'start_debug' that appear two times are text files containing a sequence commands for starting the services in release and debug mode respectively. The main difference of release and debug mode is that in the debug mode, logs will be shown on the terminal.

'start' and 'start_debug' appear two times because one is windows batch file and contains MSDOS shell commands or Windows command for Windows operating system or DOS and the other is Shell script which is for Unix based systems.[43]

Folder 'configuration' has all the necessary files for configuring the openHAB runtime. Next picture shows the contents of the 'configuration' folder.

Name	Date modified	Type	Size
items	5/22/2016 9:24 PM	File folder	
persistence	5/22/2016 9:24 PM	File folder	
rules	5/22/2016 9:24 PM	File folder	
scripts	5/22/2016 9:24 PM	File folder	
sitemaps	5/22/2016 9:24 PM	File folder	
transform	5/22/2016 9:24 PM	File folder	
logback	5/22/2016 9:24 PM	XML Document	3 KB
logback_debug	5/22/2016 9:24 PM	XML Document	3 KB
openhab_default.cfg	5/22/2016 9:24 PM	CFG File	83 KB
users.cfg	5/22/2016 9:24 PM	CFG File	1 KB

Figure 3.3 Configuration folder content

items

In ‘items’ folder are stored the ITEMS files (.items). Items are objects that can be read from or written to in order to interact with them. They can also be bounded to items for reading the status or for updating them e.g. lights.

This syntax is a Xtext DSL grammar [44] and it has this format

```
itemtype itemname ["labeltext"] [<iconname>] [(group1, group2, ...)]
[{{bindingconfig}}]
```

Item type

The itemtype defines which kind of values can be stored in that item and which commands can be send to it. Next table shows the provided item types in openHAB 1.8

Item Type	Description	Accepted Data Types	Accepted Command Types
Call	This type can be used for items that are dealing with telephony functionality.	Call	-
Color	Can be used for color values, e.g. for LED lights	OnOff, Percent, HSB	OnOff, IncreaseDecrease, Percent, HSB
Contact	Can be used for sensors that return an "open" or "close" as a state. This is useful for doors, windows, etc.	OpenClosed	-
DateTime	Stores a timestamp including a valid time zone.	DateTime	DateTime
Dimmer	Accepts percent values to set the dimmed state. Can also be used as a switch by accepting	OnOff, Percent	OnOff, IncreaseDecrease, Percent

	ON/OFF commands (though this only mimics a Switch by sending 0% and 100% for ON/OFF. See 'Dimmers vs Switches' note below		
--	---	--	--

Group	Item to nest other items / collect them in groups	The accepted data types of a group item is the same as of the underlying base item. If none is defined, the intersection of all sets of accepted data types of all group members is used instead.	The accepted command types of a group item is the same as of the underlying base item. If none is defined, the intersection of all sets of accepted command types of all group members is used instead.
--------------	---	---	---

Location	Can be used to store GPS related informations, addresses, etc. by latitude, longitude and altitude	Point	Point
-----------------	---	-------	-------

Number	Has a decimal value and is usually used for all kinds of sensors,	Decimal	Decimal
---------------	---	---------	---------

	<p>like temperature, brightness, wind, etc.</p> <p>It can also be used as a counter or as any other thing that can be expressed as a number.</p>		
Rollershutter	<p>Allows the control of roller shutters, i.e. moving them up, down, stopping or setting it to close to a certain percentage.</p>	UpDown, Percent	UpDown, StopMove, Percent
String	<p>Can be used for any kind of string or textual representation of a DateTime.</p>	String, DateTime	String
Switch	<p>Represents a normal switch that can be ON or OFF. Useful for normal lights, presence detection, etc.</p>	OnOff	OnOff

Table 3.1 Item types for openHAB 1.8 source: <https://github.com/openhab/openhab1-addons/wiki/Explanation-of-Items>

Item name

The item name is the unique name of the item which is used e.g. in the sitemap definition or rule definition files to access the specific item.

Label text

The label text has two purposes. First, this text is used to display a description of the specific item (for example, in the sitemap). Second, it can be used to format or transform output from the item (for example, making DateTime output more readable). Another possibility in label texts is the transformation. With transformation we can translate a status into another language which is easily readable. For making a transformation you have to create a `.map` file in

```
${openhab_home}/configurations/transform
```

Icon names

The icon name is used to reference a png image file from folder

```
${openhab_home}/webapps/images/
```

User can add any .png file as long as it has a size of 32x32 pixel.

Groups

Item can be categorized into groups which are declared in the beginning of the file. Definition is

```
Group Namegroup
```

Groups are supported in sitemaps, rules, functions, the `openhab.cfg` and more places.

Binding config

The binding config defines from where the item gets its values and where a given value/command should be sent. The binding module is stored into the 'addons' folder. Every binding defines what values must be given in the bindingconfig string. That can be the id of a sensor, an ip or mac address or anything else according to what requirements has every binding module [45].

Example:

```
Number LightsensorAll1 "Lightsensor [%.2f LUX]" <energy> (GF_Corridor)
{bacnet="100:0:1"}
```

- of type Number
- with name LightsensorAll1
- formatting its output in format xx.yy LUX
- displaying icon energy
- belonging to group GF_Corridor
- bound to openHAB binding bacnet

Persistence

‘persistence’ folder contains all definition files which have the extension ‘*.persist’.

Persistence support stores item states over time (a time series). OpenHAB is not restricted to a single data store. Different stores can co-exist and be configured independently. Persistence services that are supported are

Name	Queryable	Link	Notes
Cosm		cosm.com	write item states to Cosm/Xively site
db4o	X	db4o.com	a lightweight, 100% pure Java object database
Amazon DynamoDB	X	amazon.com	fully managed NoSQL database service from Amazon
Exec			persist by executing commands in the underlying OS
IFTTT		IFTTT	if-this-then-that cloud service. See my.openHAB
InfluxDB	X	influxdata.com	open-source distributed time series database with no external dependencies

JDBC	X	wikipedia	Java Database Connectivity support for MySQL and several other JDBC-enabled databases (one table per item)
JPA	X	wikipedia	Java Persistence API support for Derby and many other JDBC-enabled databases (one table "historic_item" where all item states are stored as strings)
Logging		logback.qos.ch	write item states to log files with a flexible format
MapDB	X	mapdb.org	only saves the last item state; useful for restoreOnStartup strategy
MongoDB		mongodb.com	NoSQL document-oriented database

MQTT		wikipedia	stores item states to an MQTT broker
my.openHAB		my.openHAB.org	send item states to openHAB's cloud service, and use openHAB items in IFTTT recipes -> outdated as of 31st Jan 2017
openHAB Cloud Connector		myopenHAB.org	send item states to openHAB's cloud service, and use openHAB items in IFTTT recipes
MySQL	X	mysql.com	one SQL table per item
RRD4J	X	RRD4J	Java version of the powerful round-robin database RRDtool . Numeric states only.
Sen.se		open.sen.se	send item states to the Sen.Se web site

SiteWhere		SiteWhere	send item states to a SiteWhere server instance running locally or in the cloud
Xively		cosm.com	See Cosm

Table 3.2 source: [https://github.com/openhab/openhab1-addons/wiki/Persistencepersistence services](https://github.com/openhab/openhab1-addons/wiki/Persistencepersistence%20services)

Configuration of .persist files

.persist files have the the following syntax

```
Strategies {

    <strategyName1> : "<cronexpression1>"

    <strategyName2> : "<cronexpression2>"

    ...

    default = <strategyNameX>, <strategyNameY>
}

Items {

    <itemlist1> [-> "<alias1>"] : [strategy = <strategy1>, <strategy2>, ...]

    <itemlist2> [-> "<alias2>"] : [strategy = <strategyX>, <strategyY>, ...]

    ...

}
```

In the Strategies section we declare the storing strategy. It can be a cron expression [46]

Or any of the next default strategies

- `everyChange`: persist the state whenever its state has changed
- `everyUpdate`: persist the state whenever its state has been updated, even if it did not change
- `restoreOnStartup`: If the state is undefined at startup, the last persisted value is loaded and the item is initialized with this state. This is very handy for "virtual" items that do not have any binding to real hardware, like "Presence" or similar.

In the items section we define the items and the strategy from the above section that should be applied on it [47].

An example of a `.persist` file is following:

```
// persistence strategies have a name and a definition and are referred to in the  
"Items" section  
  
Strategies {  
  
    everyHour : "0 0 * * * ?"  
  
    everyDay  : "0 0 0 * * ?"  
  
  
    // if no strategy is specified for an item entry below, the default list will be  
used  
  
    default = everyChange  
  
}  
  
Items {  
  
    // persist all items once a day and on every change and restore them from the db  
at startup  
  
    * : strategy = everyChange, everyDay, restoreOnStartup
```

```
// additionally, persist all temperature and weather values every hour

Temperature*, Weather* : strategy = everyHour

}
```

We can call a stored value of an item in rules. This means that the persistence extensions are available like methods on all items. Example:

```
Temperature.historicState(now.minusDays(1))
```

Rules & Scripts

Rules and scripts are used for automating processes. Rules are placed in the ‘rules’ folder and have the extension `.rules` and scripts are placed in the ‘scripts’ folder and have the extension `.scripts`.

The differences between rules and scripts are

- You cannot use the "import" statement within a script. Importing a library can be used only in rules
- You cannot reference script-level variables from within a closure block. This means that there are not any global variables.

Scripts are written in Xtend language[48]. They are way to create reusable components for your rules. If you need to use the same code from time to time in different rules, you can create a script and then reuse it in different parts of your logic [49].

Rules are written in Xbase [50] which shares a many details with Xtext.

The Rules’ structure is

```
[Imports]

[Variable Declarations]
```

```
[Rules]
```

In `imports` section is the declaration of libraries that are gonna be referenced for creating the rules, example:

```
import org.openhab.core.library.types.*
```

`variable Declarations` section is the declaration of variables that are going to be used in rules, example:

```
// a variable with an initial value. Note that the variable type is automatically inferred
```

```
var counter = 0
```

```
// a read-only value, again the type is automatically inferred
```

```
val msg = "This is a message"
```

And `Rules` section has a list of rules drafted in the following structure

```
rule "rule name"
```

```
when
```

```
<TRIGGER_CONDITION1> or
```

```
<TRIGGER_CONDITION2> or
```

```
<TRIGGER_CONDITION3>
```

```
...
```

```
then

    <EXECUTION_BLOCK>

end
```

For triggering rules, there are three categories:

- **Item** (-Event)-based triggers: They react on events on the openHAB event bus, i.e. commands and status updates for items
- **Time**-based triggers: They react at special times, e.g. at midnight, every hour, etc.
- **System**-based triggers: They react on certain system statuses.

and are declared after “when”.

In `EXECUTION_BLOCK`, is written all the rule’s logic. It may contain Xbase coding, calling scripts or Actions.

Actions are predefined Java methods that are automatically statically imported and can be used within Scripts and Rules to execute openHAB specific operations like sending the given command for the specified item to the event bus [51].

Calling scripts is done within rules through the action

```
callScript("<scriptname>")
```

Example of a rule

```
// increase the counter at midnight

rule "Increase counter"

when

    Time cron "0 0 0 * * ?"

then
```



```
        counter = counter + 1

    end
```

Rules can also support logging and Concurrency Guard [52].

Sitemaps

Sitemaps are a declarative UI definition. With a few lines it is possible to define the structure and the content of your UI screens. This is a definition of "site map" from WhatIs.com [53].:

A site map is a visual or textually organized model of a Web site's content that allows the users to navigate through the site to find the information they are looking for, just as a traditional geographical map helps people find places they are looking for in the real world. A site map is a kind of interactive table of contents, in which each listed item links directly to its counterpart sections of the Web site. Site maps perform the same service that the layout maps in large shopping malls perform: without them, it is possible to explore a complex site by trial and error, but if you want to be sure to find what you're looking for, the most efficient way to do that is to consult

They are files with `.sitemap` extension stored in 'sitemaps' folder and are written in Xtext DSL grammar. Sitemaps have the following structure

```
[sitemap]{

    [Frame]{

        [additional sitemap elements]}

    *

    *

    *

}
```

[sitemap]: is the declaration of sitemap's name.

[Frame]: is separating the UI into areas

[additional sitemap elements]: includes all elements that can be used in sitemap files.

Element	Description
<u>Colorpicker</u>	Allows the user to choose a color from a color wheel
<u>Chart</u>	Adds a time-series chart object for displaying logged data
<u>Frame</u>	Area with either various other sitemap elements or further nested frames
<u>Group</u>	Renders all elements of a given group defined in an items definition file
<u>Image</u>	Renders an image
<u>List</u>	Not supported by any user interface
<u>Selection</u>	Gives access to a new page where the user can choose among values defined in the mappings parameter
<u>Setpoint</u>	Shows a value and allows the user to modify it. Optionally, it is possible to specify maximum

	and minimum allowed values, as well as a step
<u>Slider</u>	Renders a slider
<u>Switch</u>	Renders a switch item
<u>Text</u>	Renders a text element
<u>Video</u>	Displays a video
<u>Webview</u>	Display a webpage

Table 3.3 Elements used in sitemaps source:<https://github.com/openhab/openhab1-addons/wiki/Explanation-of-Sitemaps>

Sitemaps also support dynamic changes according to item's state like changing colors or icon when lights are on, off [54].

Example

```
sitemap demo label="Main Menu" {

    Frame label="Date" {

        Text item=Date

    }

    Frame label="Demo" {

        Text label="Group Demo" icon="1stfloor" {

            Switch item=Lights mappings=[OFF="All Off"]

            Group item=Heating

            Text item=Temperature valuecolor=[>25="orange",>15="green",<=15="blue"]

        }

    }

}
```

```

    }

    Text label="Multimedia" icon="video" {

        Selection item=Radio_Station mappings=[0=off, 1=HR3, 2=SWR3, 3=FFH,
4=Charivari]

        Slider item=Volume

    }

}

}

```

transform

‘transform’ folder contains files that can translate data to a more human readable form. Two of the most used transformations are map and scale files [55]. Map files with `.map` extension are used to transform a string into another easily understood.

Example

```

0=closed

1=open

undefined=unknown

```

Scale files which have `.scale` extension are used to assign a range of values in a variable.

Example

```

-=undefined

[-40,20]=no significant

[20,29]=comfortable

```

```
[29,38]=some discomfort  
  
[38,45]=avoid exertion  
  
[45,54]=dangerous  
  
[54,100]=heat stroke imminent
```

Other supported transformations are

- RegEx transformation
- XPath transformation
- XSLT transformation
- JSONPath transformation

logback.xml & logalback_debug.xml

These .xml files are used for the logging configuration.

Openhab.cfg

The .cfg file is used to define all basic settings, such as IP addresses, mail server, folder locations ,persistence etc. It contains all the settings of our bindings. Each binding has its own configurations [56].

Example

```
##### NeoHub Binding  
#####  
  
#  
  
# Refresh interval in milliseconds (optional, defaults to 60000ms)  
  
#neohub:refresh=60000
```

```
# Set the NeoHub network address

#neohub:hostname=

# Set the port number for the NeoHub interface (optional, defaults to 4242)

#neohub:port=4242

##### Open Energy Monitor Binding
#####

#

# UDP port of the Open Energy Monitor devices (optional, defaults to 9997)

#openenergymonitor:udpPort=9997

# For testing purposes

#openenergymonitor:simulate=true
```

User.cfg

In user.cfg you can declare user , its password and its role. It is a feature not supported yet

In the following figure, there is a schematic representation of the dependencies between files and the contents of folders. The last are represented by their folders.

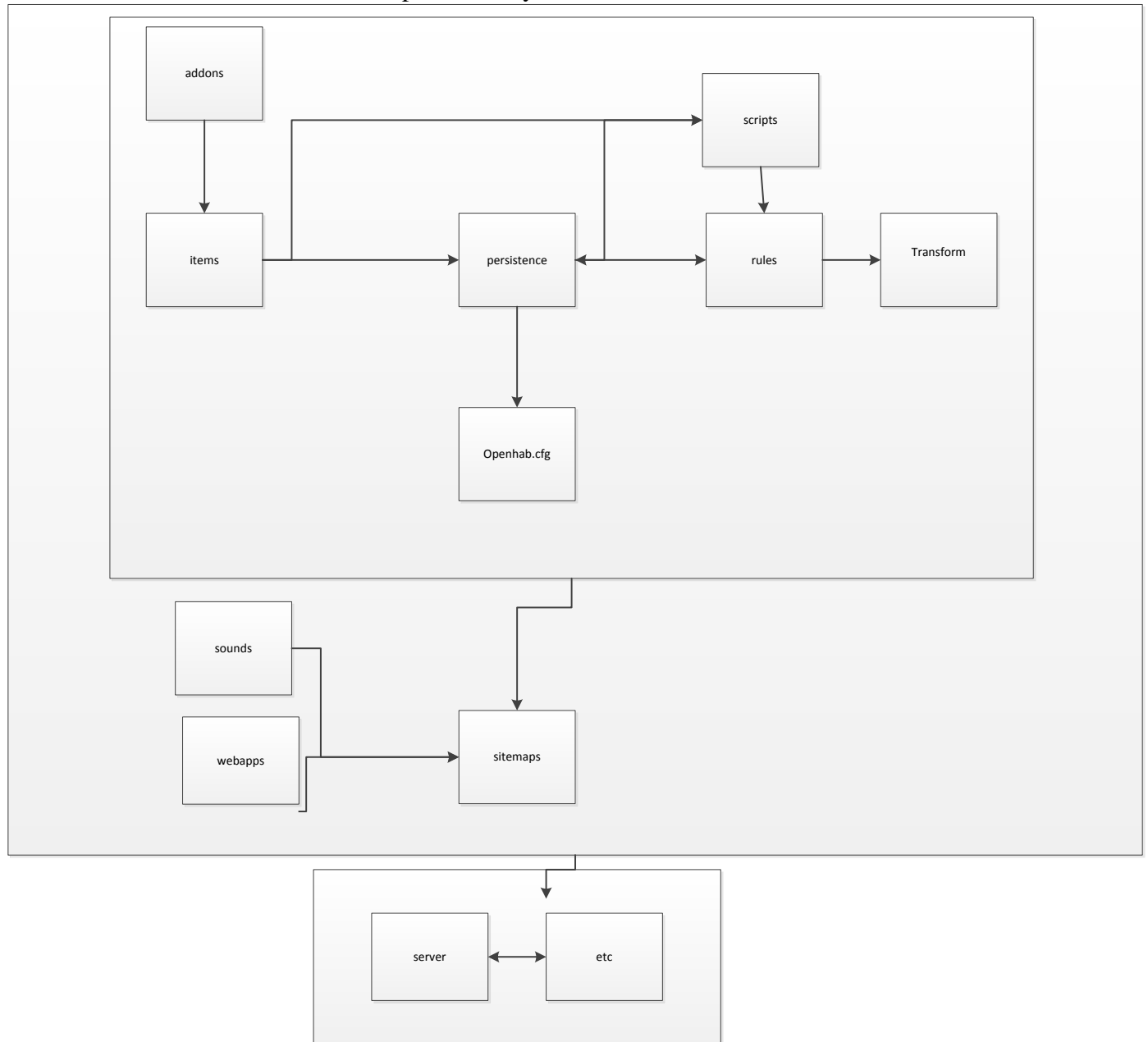


Figure 3.4 Dependencies between files

openHAB Architecture Overview

As was mentioned, openHAB runtime is a set of OSGi bundles deployed on an OSGi framework (Equinox). Next figure shows graphically the architecture [57].

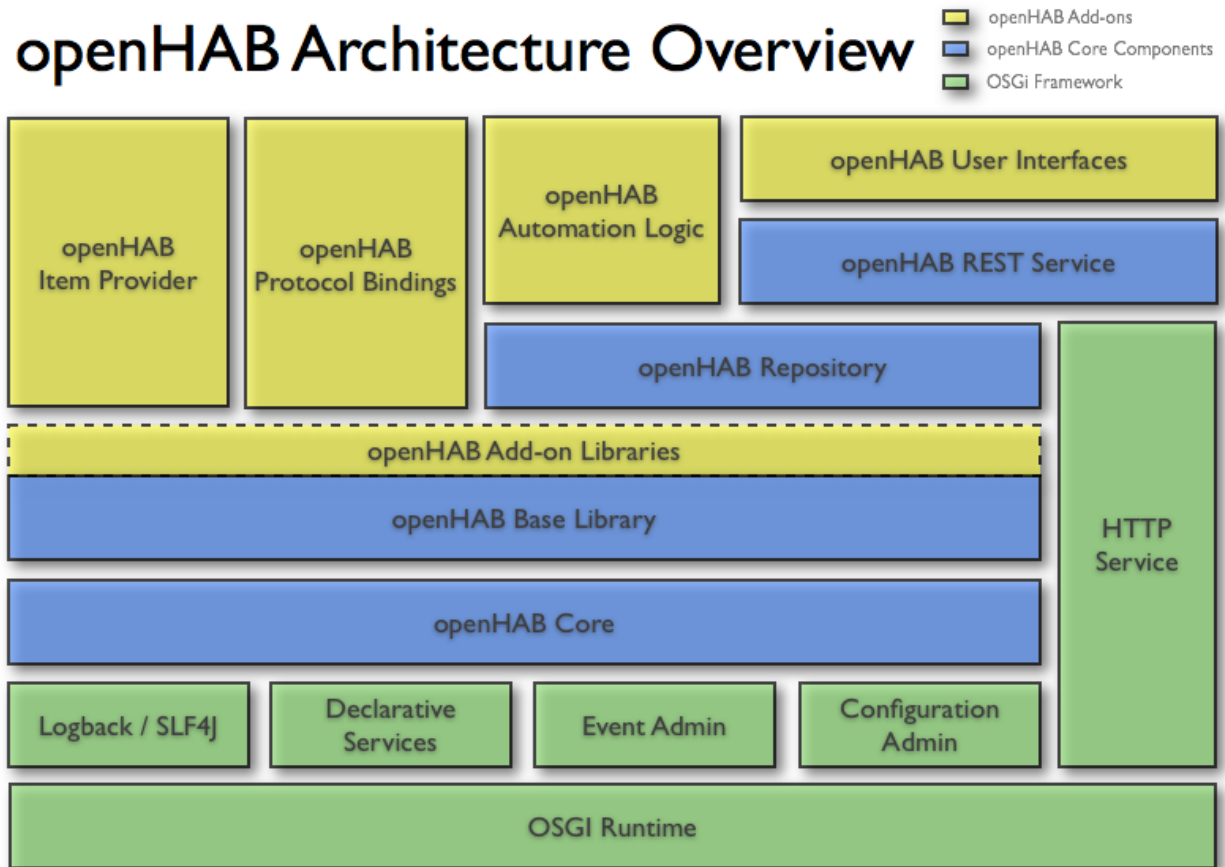


Figure 3.5 OpenHAB architecture overview source: <http://www.openhab.org>

Internal communication

openHAB has two different internal communication channels:

1. an asynchronous event bus
2. a stateful repository, which can be queried

The event bus

The event bus is the service that bundles use to inform other bundles about events or be informed by other bundles for events. Events are distinguished into Commands which trigger an action or a state change and Status updates which inform about a status change of some item/device.

By communicating via the bus, there is low coupling between the bundles. The service that is used for scheduling is called OSGi EventAdmin [58].

OpenHAB is meant to act as an integration hub between devices and as a mediator between different protocols. This means that openHAB is not supposed to run on each connected devices but on a central server.

Apart from stateless services, communication is also supported by Item repository. Item repository is connected with Event bus and keeps track of the current status of all items and it is the accessed by automation rules for creating rules according to current states of devices.

The following diagram shows how these communication channels are used:

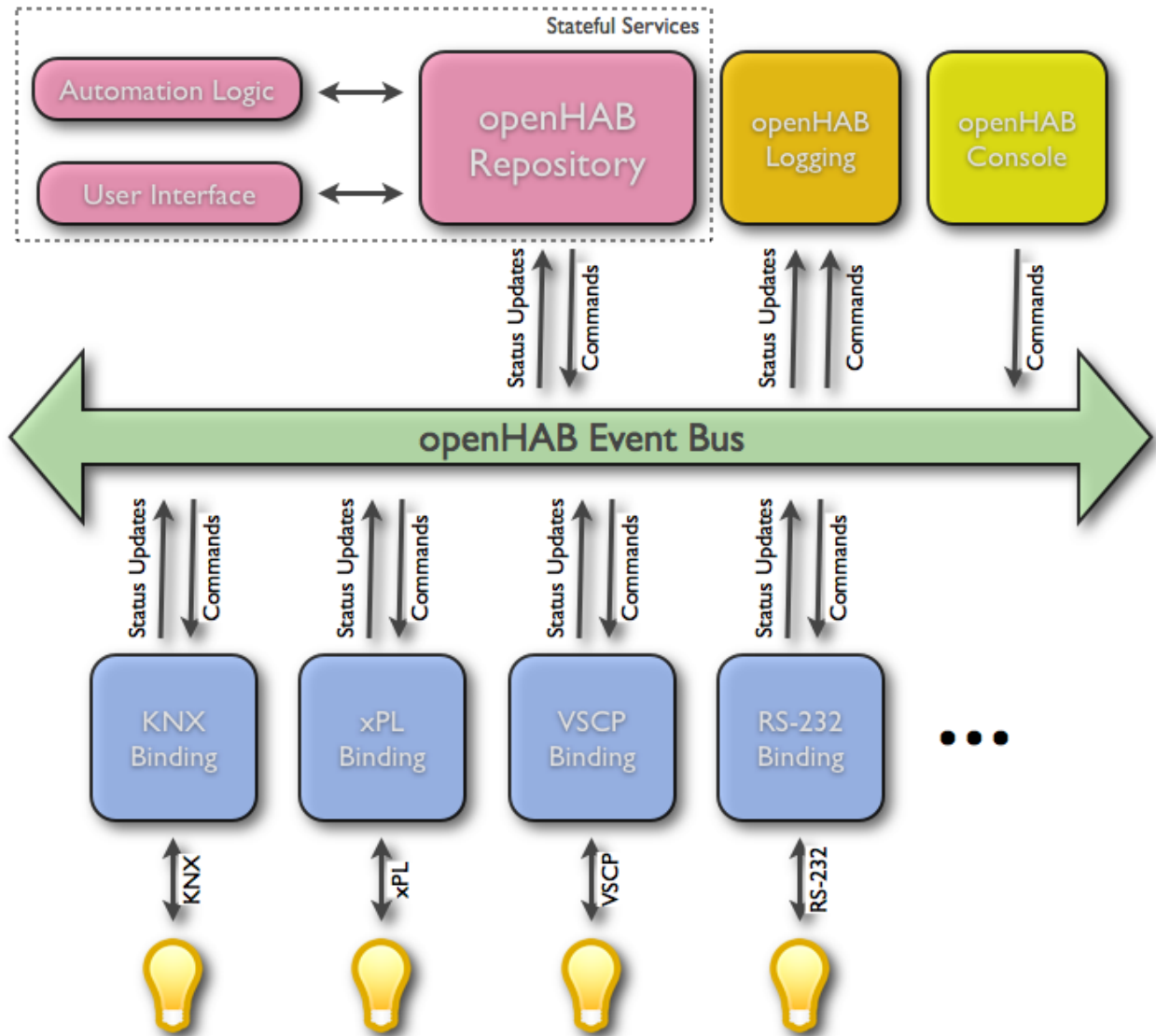


Figure 3.6 Services across the openHAB event bus

3.3 HABmin and openHAB designer

In this subsection we are dealing with two tools used for facilitating the configuration of Openhab. These tools are HABmin and OpenHAB designer.

HABmin

HABmin is an open-source web administration console for openHAB. It is not just a UI for overview of the devices' states but also for configuring bindings, items , sitemaps etc. User is also able to create new rules via a drag and drop process which facilitates the whole procedure of generating new rules. This feature is called rule designer and it is shown in next figure.

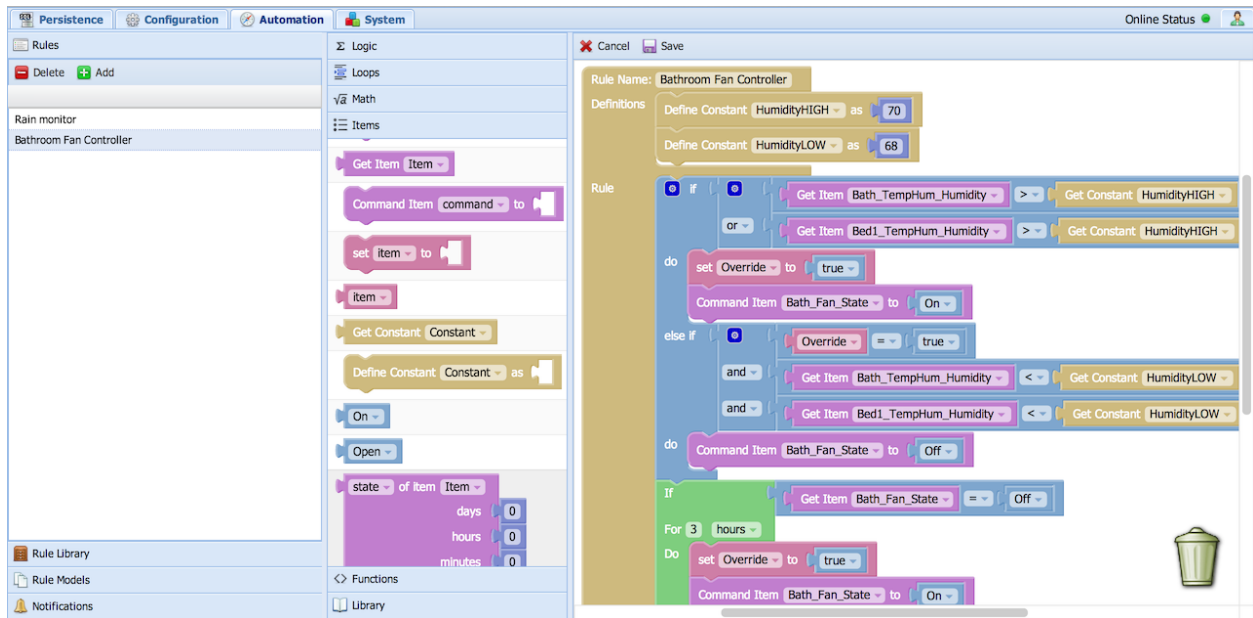


Figure 3.7 Screenshot from HABmin console source:<https://github.com/cdjackson/HABmin/wiki/Rule-Designer:-Overview>

Other features provided by HABmin are viewing log files and OSGi binding status , querying and graphing data from persistence stores. Following screenshots, show some of the functionalities

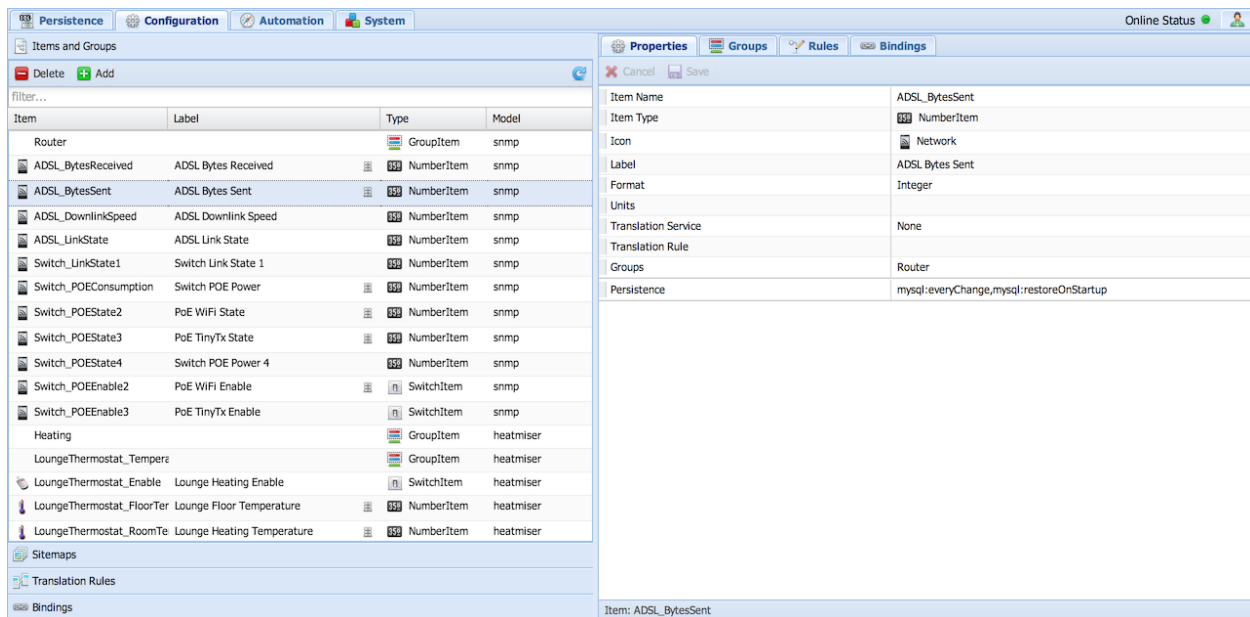


Figure 3.8 item configuration in HABmin source: <https://github.com/cdjackson/HABmin>

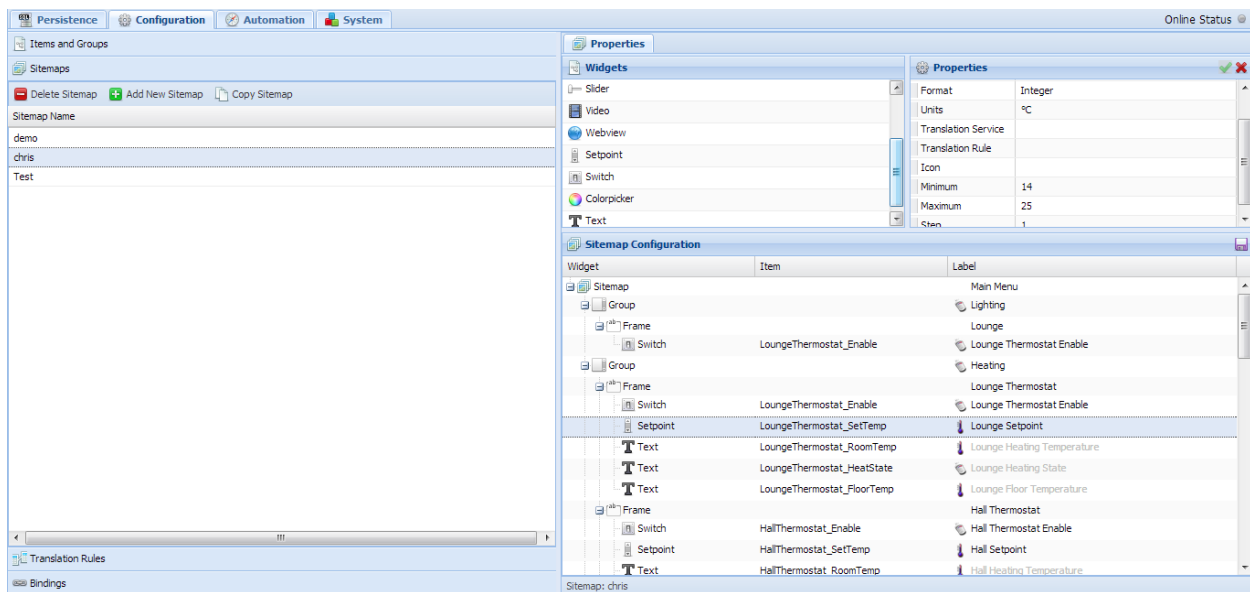


Figure 3.9 Configuring sitemap in HABmin source: <https://github.com/cdjackson/HABmin>

HABmin works using rest-api technology and can be accessed by pointing to the address

<http://localhost:8080/habmin/index.html>

It is downloaded from HABmin's page in github [59] and it is placed in the openHAB repository according to the instructions given in the site .

HABmin was not used through this thesis. The causes of non-usage are that it does not support creating complex rules. Drag and drop feature is restricted to standard bundles and cannot be extended. Also by creating a rule HABmin changes the structure of the `.rules` file making it difficult to read. In addition existing REST interface does not support most of the functionality required by *HABmin demanding some extension in rest-apis* . Finally there is a need to be close to the code so it will be easier to understand what is happening during logic and syntax errors but also how openHAB as a platform works.

HABmin was installed for users who want to make a change quickly and without having the need to learn from the beginning how changes are made in coding level as UI offers easily perceived processes.

OpenHAB designer

The openHAB Designer is an Eclipse RCP application for configuring the openHAB runtime [60]. It has Xtext-based editors to offer a highly user-friendly way of editing configuration. It supports also syntax checking, auto completion, highlighting and content assist. It can be downloaded from the official openHAB web page [61].

OpenHAB designer was not used during this thesis because it threw some syntax errors which were not, and this was irritating for the coding process. It also doesn't support changing background color in order to be more restful for the eyes.

4. Configuring OpenHAB runtime

In this section of the thesis there is an extensive description on how to configure openHAB runtime so it is able to satisfy all our requirements. OpenHAB runtime version 1.8.3 was used for this thesis and was downloaded from the official openHAB site [62]. Firstly we describe how to add our sensors in openHAB and making it able to read their values. By reclaiming these values, we create our rules. Then we built the user interface making it user-friendly and add access authentication to our application.

4.1 Adding new devices

As stated in section [2.1](#) and [2.2](#) communication is made on BACnet protocol. In order to be able to see if we are able to access the sensors we used YABE (Yet Another Bacnet Explorer). Yabe is a graphical explorer for browsing BACnet devices. It was downloaded from tge Sourceforge site [63]. This tool was used to just to be sure that controller is up and running and to easily find the unique ID of each device. As stated in [3.2](#) for establishing a communication we have to use bindings. Binding for bacnet protocol was downloaded from github web page [64]. This binding is not an official OpenHAB binding because of a restriction in GPLv3 license [65]. As a result it is made by an active community member but not hosted in the official OpenHAB server. The .jar file, which is the binding, downloaded from this link was dropped into the ‘addons’ folder which is a floder that can extend openHAB’s functionality. Now for configuring OpenHab runtime to be able to read the sensors data we want firstly to modify the ‘openhab.cfg’ file. In this file we set up the refresh rate in which we read values from our sensors and the hostname. Below we can see the code used for configuration:

```
##### Bacnet Binding #####  
  
# Refresh interval in milliseconds
```

```
bacnet:refresh=60000

# Set the bacnet network address

bacnet:hostname=54321
```

We have no need to set up port number or password and that is because when Bacnet binding starts up, it sends out a broadcast looking for all devices. Devices must be in the same subnet and as the BACnet protocol doesn't provide security in the form of authentication there is no password given.

After modifying the 'openhab.cfg' file, we have to create a .items file in the 'items' folder. In this file we define the devices that we want to access. For each of the device we have to set three mandatory keys. These are deviceID, objectType and objectID.

- `Device ID` is the instance ID (integer) of the device as configured on your local network (it is *not* the IP address of the device)
- `object Type` is one of the following:
 - analog Input = 0
 - analog Output = 1
 - analog Value = 2
 - binary Input = 3
 - binary Output = 4
 - binary Value = 5
- `object ID` is the instance ID (integer) of the object you want to tie to this openHAB item

The declaration of the sensors is stated below:

```
Number LightsensorA11 "Lightsensor [%.2f LUX]" <energy> (LabEmber)
{bacnet="100:0:1"}

Number CO2_Sensor "CO2_Sensor [%.3f]" <energy> (LabEmber) {
bacnet="100:0:2" }
```

```

Number RH "RH [%.3f]" <energy> (LabEmber,PMVgraph) { bacnet="100:0:3" }

Number Presense "presense [%.3f]" <energy> (LabEmber) { bacnet="100:0:6" }

Number Temperature_thermistor "Thermistor[%.3f]"<energy>
(LabEmber,PMVgraph,Ventgraph,HVACgraph){ bacnet="100:0:4" }

Number windowOnOff "windowOnOff " <energy> (LabEmber) {bacnet="100:2:8"}

Number doorOnOff "doorOnOff [%.3f]" <energy> (LabEmber) {bacnet="100:2:9"}

```

All items are defined as Number since it holds decimal value. Next we define the name of the variable that holds the decimal value and is going to be used in our rules (see 4.2). After that we declare the text that is displayed as a description used in sitemaps but also the format of the value (see 4.3). For all our items and variables, the `<energy>` icon was used. All items were also placed in `LabEmber` group. This is used for displaying all items in the same page in the sitemap. Items that belong apart from `LabEmber` and in other group is because we wanted to create graphs (see 4.3).

Finally we declare the mandatory keys of the BACnet devices. Firstly is the `deviceId`. This ID is the same for all as they communicate with the same controller which is configured to have `id=100`. Light, CO₂, RH, Presence and temperature sensors are `analogInput(0)`. That is because they do not give a binary value (0 or 1) but a value that ranges according to the characteristics the manufacturer has set. Finally the last number is the id each object has which is unique.

By doing the above steps we imported the sensors in the openHAB runtime. We are not able yet to see their value unless we create a sitemap (see 3.4).

During the process of adding our devices I dealt with two problems. The first one was that openHAB is not able to read sensor data if YABE is running. No solution found for this problem so as stated in the beginning of this section I used it only to confirm that controller is up and running and to easily find the unique ID of each device. The second problem I faced was that the creator of the BACnet binding has had his manual written wrong. The mistake was in the values which the mandatory keys should have. In the 'readme' it only stated that for a device the declaration of mandatory keys should be


```
{bacnet="device=701105,type=binaryValue,object=3"}
```

and not

```
{ bacnet="701105:5:3" }
```

The answer was found in a KNX forum [66]. After being able to read the BACnet devices values, I posted the proper way of configuring them in the thread that I had created asking for help in the openHAB forum [67].

4.2 Creating rules

4.2.1 PMV

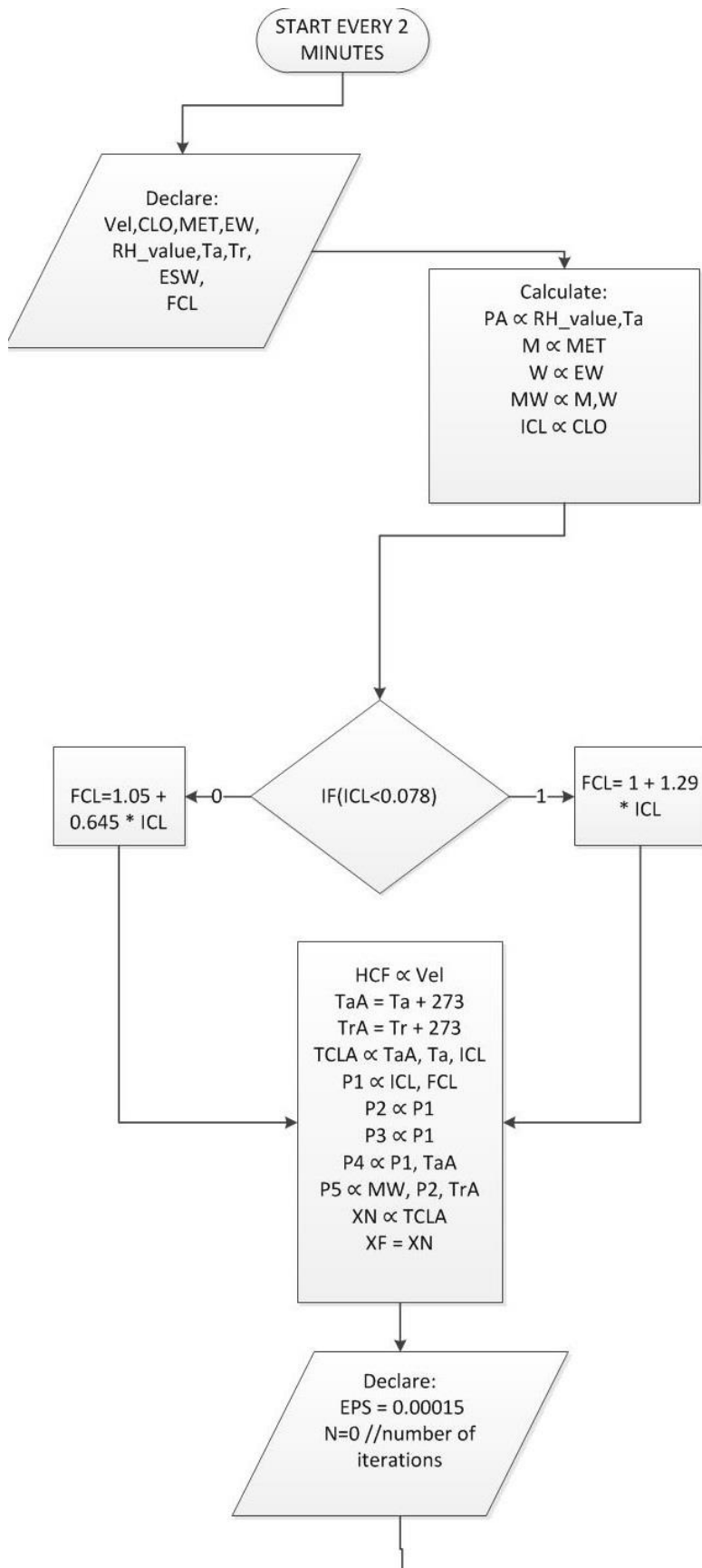
PMV (Predicted Mean Vote) is a model which is used to express the human perception of thermal comfort. It was developed by P.O Fanger using heat balance equations and empirical studies about skin temperature. The PMV index is taken as the mean response of the collected personal votes from a large group of people on a thermal sensations scale from -3 (cold) to +3 (hot). Zero is the ideal value, representing thermal neutrality. A body is in thermal equilibrium if the heat it generates equals the heat that it loses to its environment and the comfort zone is defined by the combinations of six parameters for which the PMV is within the recommended limits ($-0.5 < \text{PMV} < +0.5$). These six values are physical activity [met], value of thermal insulation through clothing [clo], air temperature, mean radiant temperature, air velocity, and relative humidity [68].

PMV code was given in .lua format. As openHAB is able to run xbase code, PMV code was translated into xbase language. Firstly PMV was added as a Number in the `.items` file.

```
Number PMV "PMV [%.3f]" <energy> (LabEmber,PMVgraph,Ventgraph,HVACgraph)
```

Apart from LabEmber, PMV value also belongs in other groups because we want its value to be included into our graphs. In order to make these graphs we have to store the PMV value into a database. The database that was chosen is rrd4j because this is appropriate for graphs. So there was created a `.rrd4j` file into 'persistence' folder which included that PMV value will be stored every minute.

As it comes to the algorithm, it was decided that it will run every two (2) minutes. In the figure below, we can see the algorithm's flowchart



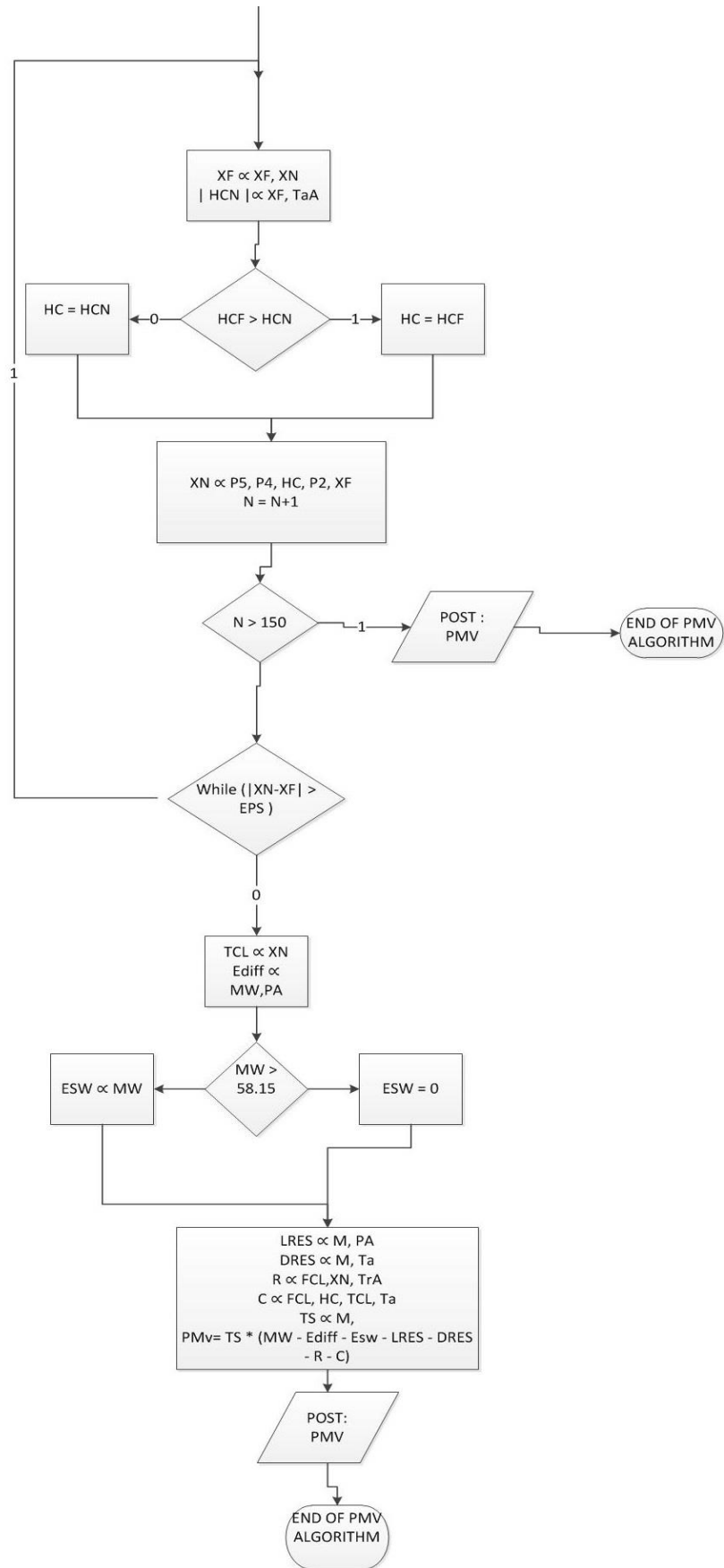


Figure 4.1 PMV algorithm flowchart

In order to understand the algorithm we have to explain the meaning of each of the variables.

The most important are:

Ta : Air Temperature, [deg.C]

Tr: Mean Radiant Temperature, @ [deg.C]

Vel : Relative Air Velocity, [m/s]

RH : Relative Humidity, [%]

CLO : Clothing, [clo]

MET : Metabolic Rate, [met]

EW : External Work, [met] (=normally around 0)

PA : Water Vapor Pressure, [Pa]

We initialize these variables with values that are appropriate to the existing laboratory's conditions. *RH*, *Ta* and *Tr* are initialized with thermistor's and humidity sensor's data.

Next step is to calculate the metabolic rate. Variables that have to be calculated are:

M : Metabolic Rate, [W/m²]

W: External Work, [W/m²]

MW: internal heat production in the human body

All temperature values are converted into kelvin (K) and

HCF : convective heat transfer coefficient by forced convection

Then we deal with clothing and surface temperature of clothing by iteration. We calculate

ICL : thermal insulation of the Clothing, [m²K/W]

FCL : clothing area factor

TCLA : first guess for surface temperature of clothing

P1,P2,P3,P4,P5 : calculation terms

Afterwards we start a while-loop so that we give value in XN, HC terms for calculating other variables or break the process as it is not giving an appropriate value for PMV.

In the last part of the algorithm we calculate terms that are going to be used for the PMV formula and are related with heat loss . These are:

Ediff : heat loss through skin

ESW : heat loss by sweating

LRES : latent respiration heat loss

DRES : dry respiration heat loss

R : heat loss by radiation

C : heat loss by convection

Finally we calculate PMV. In the figure below we show a simplified flowchart for the PMV algorithm.

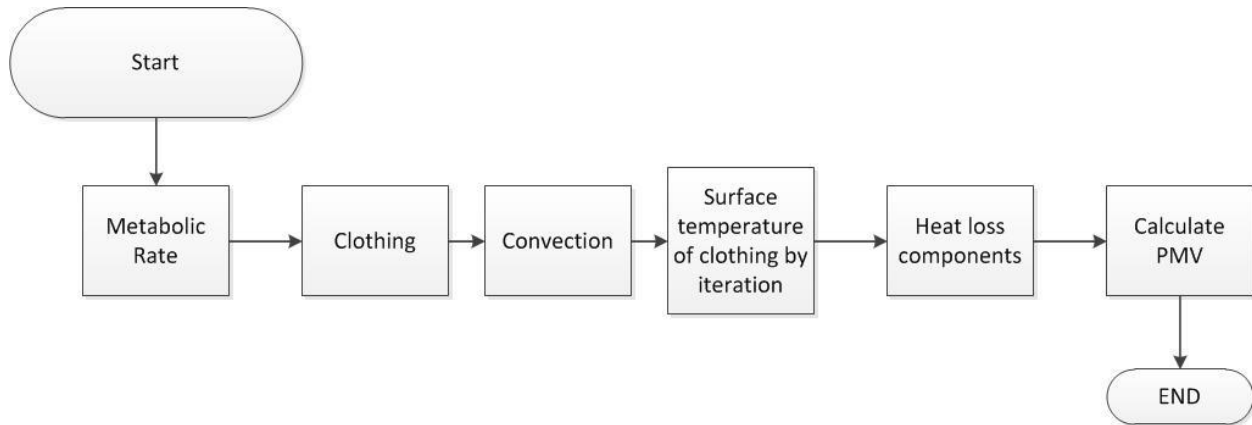


Figure 4.2 Simplified PMV flowchart

The way temperature and humidity affects the PMV value can be seen in the graph below

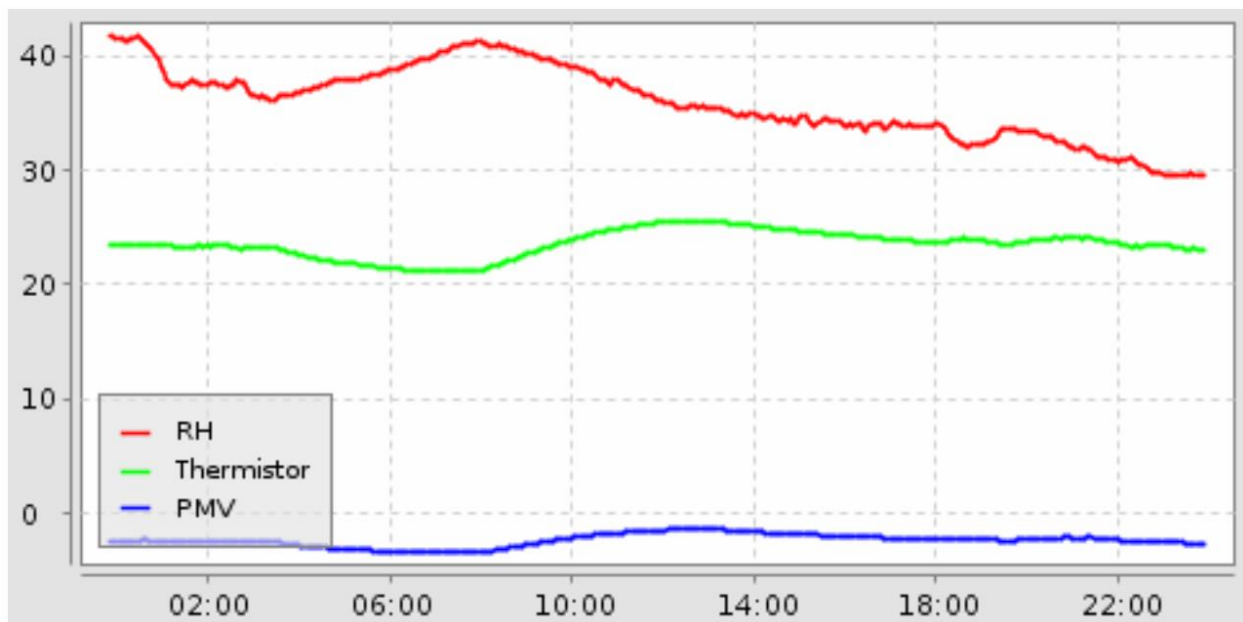


Figure 4.3 RH, Thermistor, PMV common graph

4.2.2 Fuzzy rules

In this subsection we deal with rules that use fuzzy logic to decide what action they should do or to extract conclusions. OpeHAB does not support by default the use of fuzzy logic. For making it able to use it we have to extend its functionality by adding a binding in the ‘addons’ folder and

making the appropriate configurations. The binding which is a .jar file was downloaded from github web site [69].

Then we create a .fcl file. In this file we define the input-output variables the fuzzifier-defuzzifier and the rules. The .fcl file's structure is shown below

```
FUNCTION_BLOCK functionName

VAR_INPUT
...
END_VAR

VAR_OUTPUT
...
END_VAR

FUZZIFY inputVariable
...
END_FUZZIFY

DEFUZZIFY outputVariable
...
END_DEFUZZIFY

RULEBLOCK No1
...
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Firstly we define the name that the function block should have. Next we define the input and output variables and their data types in the corresponding sub-blocks. Example


```

VAR_INPUT

    PMVvalue: REAL;

    Thermistor: REAL;

END_VAR

VAR_OUTPUT

    ventilation: REAL;

END_VAR

```

Next we define how each input variables is fuzzified is defined in `FUZZIFY` block. In each block we define one or more TERMS . Each term is composed by a name and a membership function.

Example

```

FUZZIFY PMVvalue

    TERM NE := trape -9.350 -5.150 -2.500 -1.650;

    TERM ZERO := trape -1.350 -0.400 0.400 1.350;

    TERM PO := trape 1.650 2.850 3.150 4.350;

END_FUZZIFY

DEFUZZIFY ventilation

    TERM NE := trape -150.000 -100.000 -80.000 -60.000;

    TERM ZERO := trape -20.000 -10.000 10.000 20.000;

    TERM PO := trape 60.000 80.000 100.000 150.000;

    METHOD : COG;

    DEFAULT := 0;

END_DEFUZZIFY

```

In the Defuzzifier we also state the method used. In our example is 'Center Of Gravity' and o use '0' as default value, i.e. the value use when no rule activates this variable.

Finally we define the rules in the RULEBLOCK. Example

```
RULEBLOCK setMechVen

AND : MIN;    // Use 'min' for 'and'

ACT : MIN;    // Use 'min' activation method

ACCU : MAX; // Use 'max' accumulation method

    RULE 1 : if PMVvalue IS NE AND Tout IS VCOLD THEN Mechanicalventilationchange IS
NE;

    RULE 2 : if PMVvalue IS NE AND Tout IS MCOLD THEN Mechanicalventilationchange IS
SNE;

    RULE 3 : if PMVvalue IS NE AND Tout IS OK THEN Mechanicalventilationchange IS SP0;

    RULE 4 : if PMVvalue IS NE AND Tout IS MHOT THEN Mechanicalventilationchange IS PO;

    RULE 5 : if PMVvalue IS NE AND Tout IS VHOT THEN Mechanicalventilationchange IS PO;

END_RULEBLOCK
```

Use 'min' for 'and' (implicitly uses 'max' for 'or' to fulfill DeMorgan's Law).

```
AND : MIN;
```

Use 'min' activation method

```
ACT : MIN;
```

Use 'maximum' as accumulation method.

```
ACCU : MAX;
```

That is how we configure our fuzzy controller in all our rules. In order to make openHAB able to locate our .fcl file we have to add some code lines in our .cfg file. These lines are stating the path that openHAB has to follow to find the .fcl file and the input and output parameters. The name of the `FUNCTION_BLOCK` is also stated. Example

```
##### jFuzzyLogic action #####  
  
jfuzzylogic:PMVController.FclFilePath=C:\\Users\\ember\\distribution183runtime\\fuzzy  
logic.fcl  
  
jfuzzylogic:PMVController.InParamList=PMVvalue,Tout  
  
jfuzzylogic:PMVController.OutParam=HVACSETPOINTCHANGE
```

The rules that were implemented were created by the EMBER lab staff [<http://www.ember.tuc.gr/index.php?id=3470>] for the CAMP IT project [<http://www.campit.gr/>]. The fuzzy controller was created in .fis format which is a format for Matlab [70]. So there was the need of converting the .fis file into a .fcl file. An easier way was found during our second rule. There is a tool called **jfuzzylite** [<http://www.fuzzylite.com/>] which is a free and open-source fuzzy logic control library programmed in Java and can easily design and operate fuzzy logic controllers. The tool is also able to convert .fis files into .fcl. The resulting archive file has some minor differences in structure and reserved words from the desirable one.

In our rules in the .rules file we use the command `doFuzzyLogic` in order to call the binding which takes as parameters the name of the `FUNCTION_BLOCK` and a list with the input variables. It returns a list with all output variables.

During my involvement with fuzzy rules I faced difficulties. After having created the first rule and tried to execute it, I found out that it was giving bad results. I got in contact with the creator of the binding who assured me that as far as he has checked the binding was working correctly. The binding is using jFUZZYLogic [71] which is an open source fuzzy logic library, written in java, implementing industry standards to simplify fuzzy systems developments. I started checking if running this library in a java machine with the parameters of our rules was giving appropriate results. The results was alright so I got in contact with the creator of the binding again, informing him about my work and sending a project to convince him that there was a

mistake in the binding. After a few days he informed me that the .jar file that was uploaded in the github page was an older version which was not working properly [72].

4.2.3 Rule HVAC set-point change

In the first fuzzy rule we had to calculate the HVAC set point. This set point is used for determining the desirable room temperature. Firstly we create a new variable in the .items file for holding the HVAC set point value.

```
Number HVACSETPOINTCHANGE "HVACSetPoint [%.3f]" <energy> (LabEmber,HVACgraph)
{bacnet="100:2:10"}
```

It is type `Number` and is determined as `analog Value` in the bacnet definition. There is also a strategy created in the `rrd4j.persist` file in the “persistence” folder for saving the HVAC set point values in the database so we will be able to create a graph out of these. In the .cfg file we add the path to the fuzzy controller so that the openHAB can find it.

The rule is running every time the PMV value changes. We use the PMV and temperature values as input parameters in our fuzzy controller which returns the HVAC set point. In the next figure we can see the rule’s flowchart.

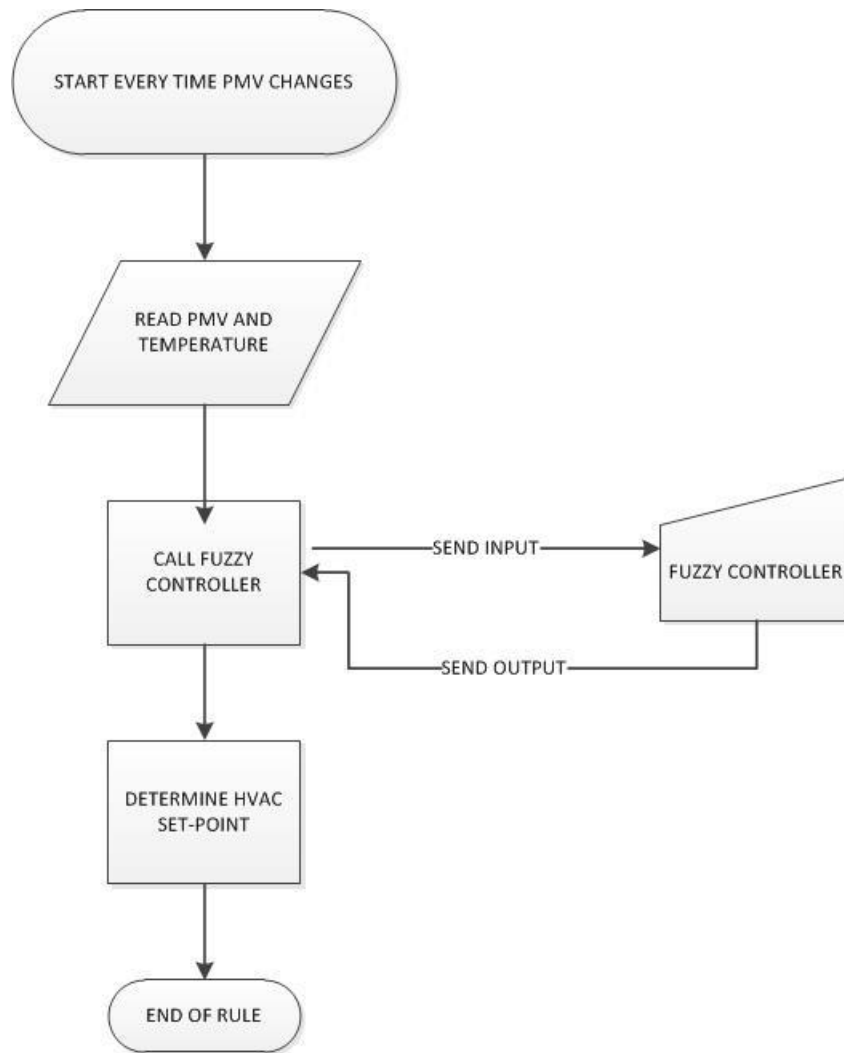


Figure 4.4 HVAC set point algorithm's flowchart

We had to convert the given pre packed .fis into a .fcl file for being able to be read by openHAB runtime. The file was created following the structure that was given in the jFuzzyLogic site which were described in section [4.2.2 Fuzzy rules](#). In the next figures, there are shown all fuzzy controller's membership functions of each input and output parameter.

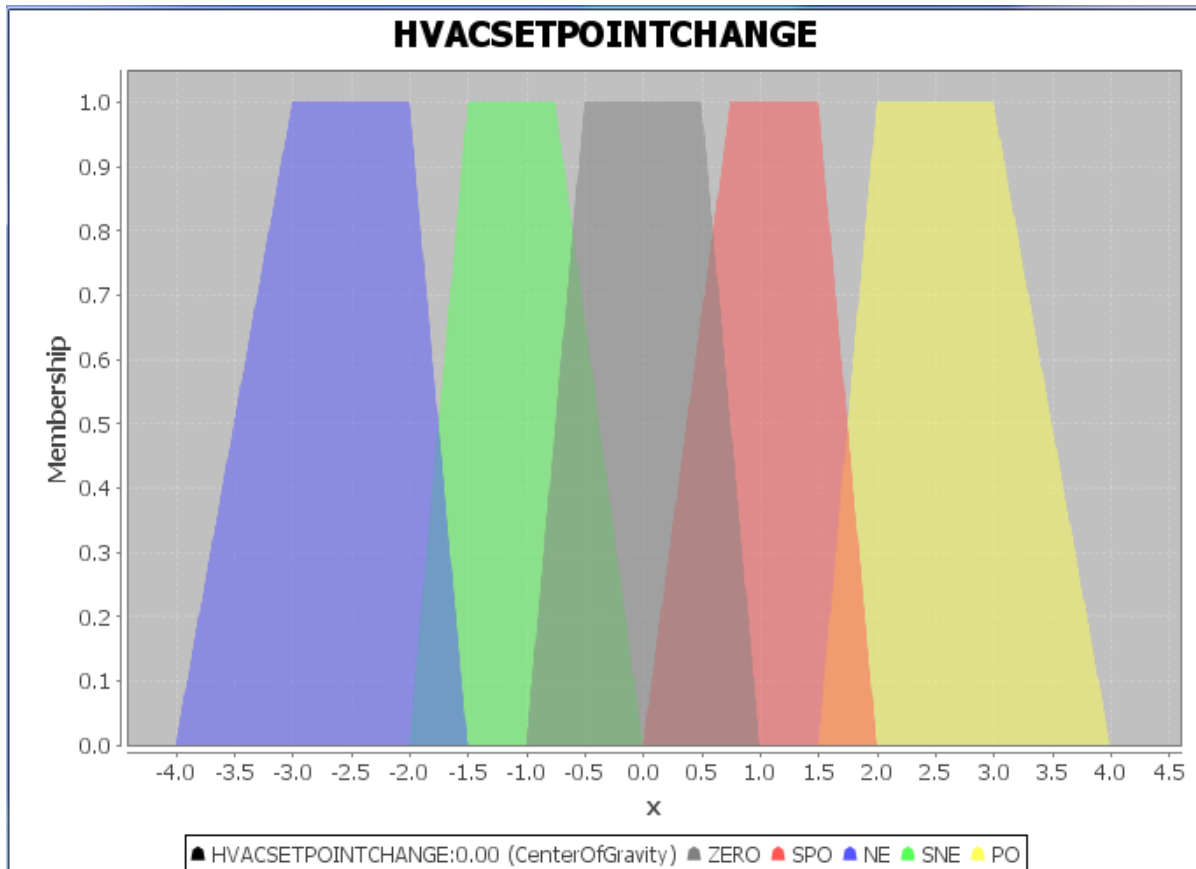


Figure 4.5 HVAC set point membership function, fuzzy controller's output

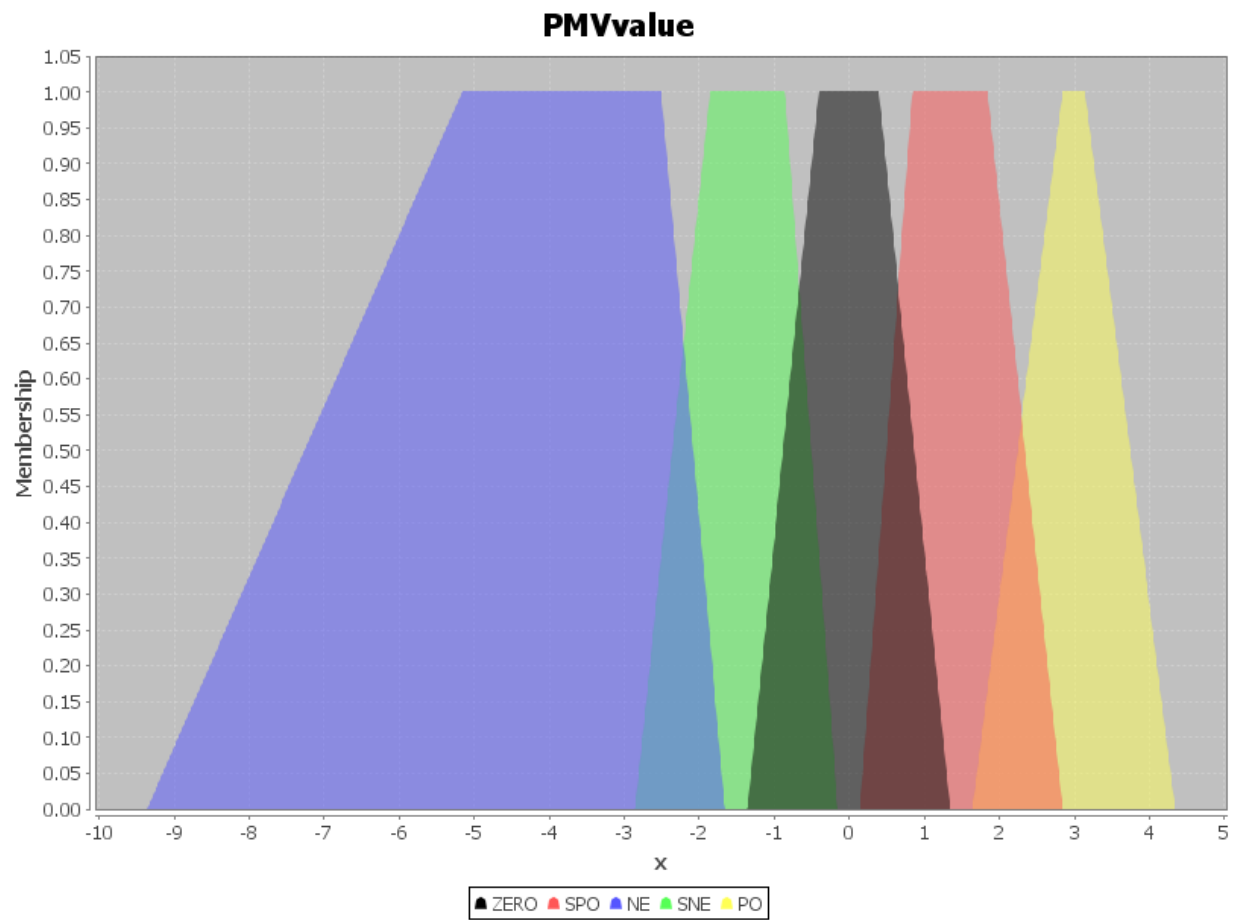


Figure 4.6 Input PMV's membership functions

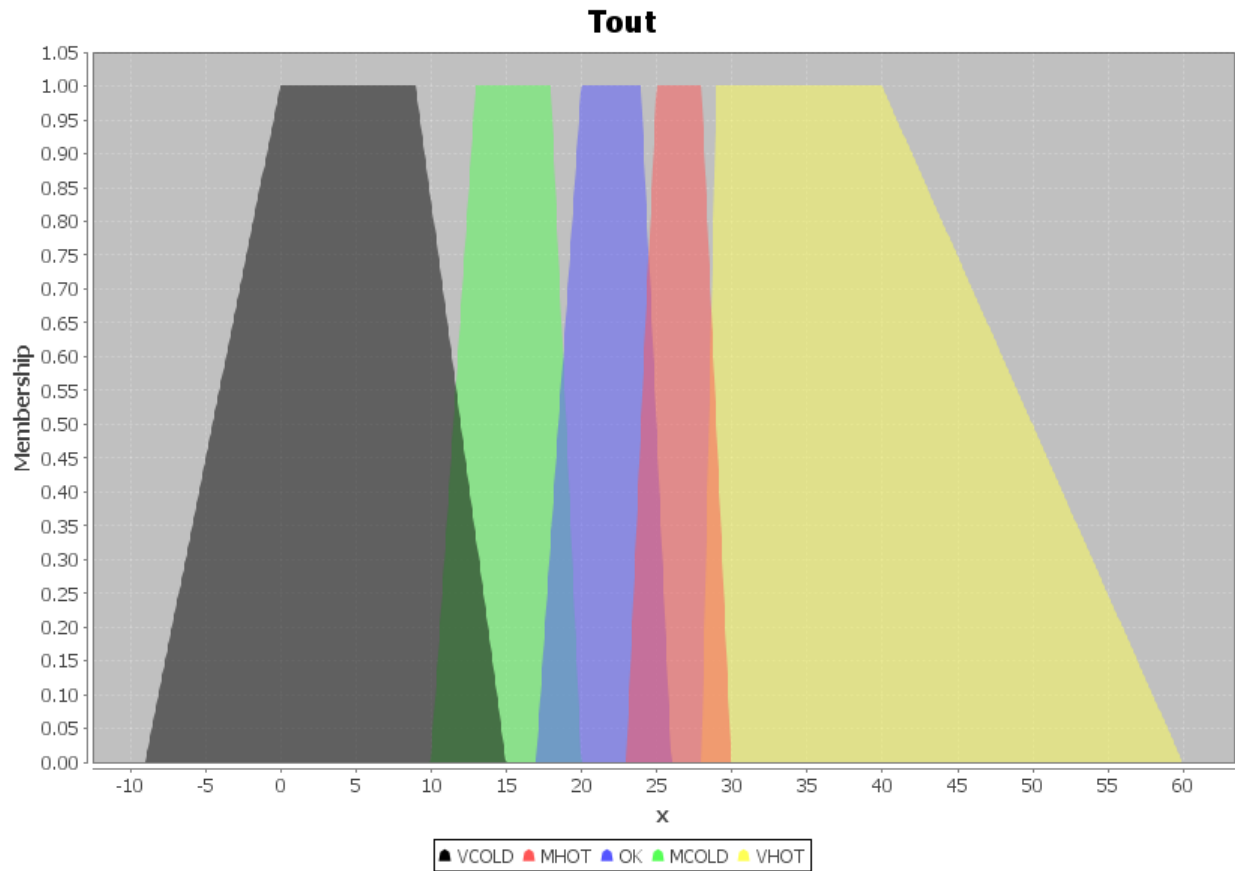


Figure 4.4.7 Input temperature's membership function

In the next graph, we can see how the HVAC set point is affected by temperature and PMV value.

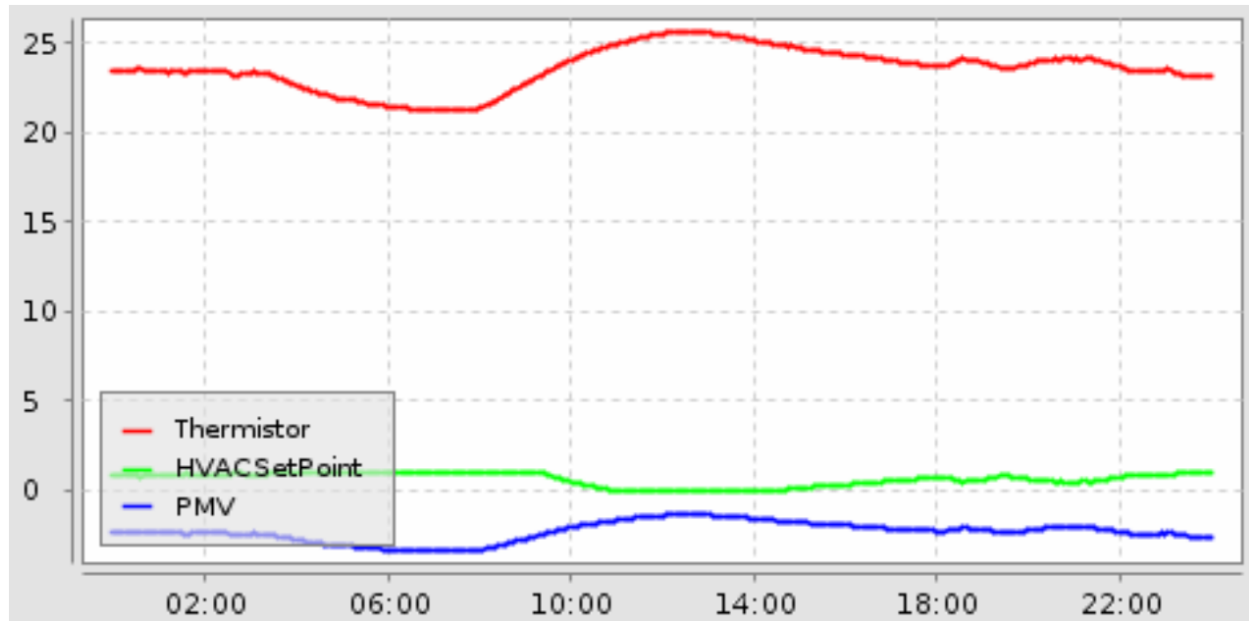


Figure 4.8 HVAC set point graph

4.2.4 Rule Fuzzy Ventilation

The rule which is presented in this section is responsible for controlling the ventilation and the amount of air entering the room via the fans. It is affected by the PMV and temperature values. Firstly we create a new item in .items file

```
Number Mechanicalventilationchange "Ventilation [%.3f]" <energy>
(LabEmber,Ventgraph) {bacnet="100:2:12"}
```

It is type `Number` and is determined as `analog Value` in the bacnet definition. There is also a strategy created in the `rrd4j.persist` file in the “persistence” folder for saving the HVAC set point values in the database so we will be able to create a graph out of these.

In the `.cfg` file we add the path to the fuzzy controller so that the openHAB can find it.

The rule is running every time the PMV value changes. We use the PMV and temperature values as input parameters in our fuzzy controller which returns the ventilation value. In the next figure we can see the rule’s flowchart.

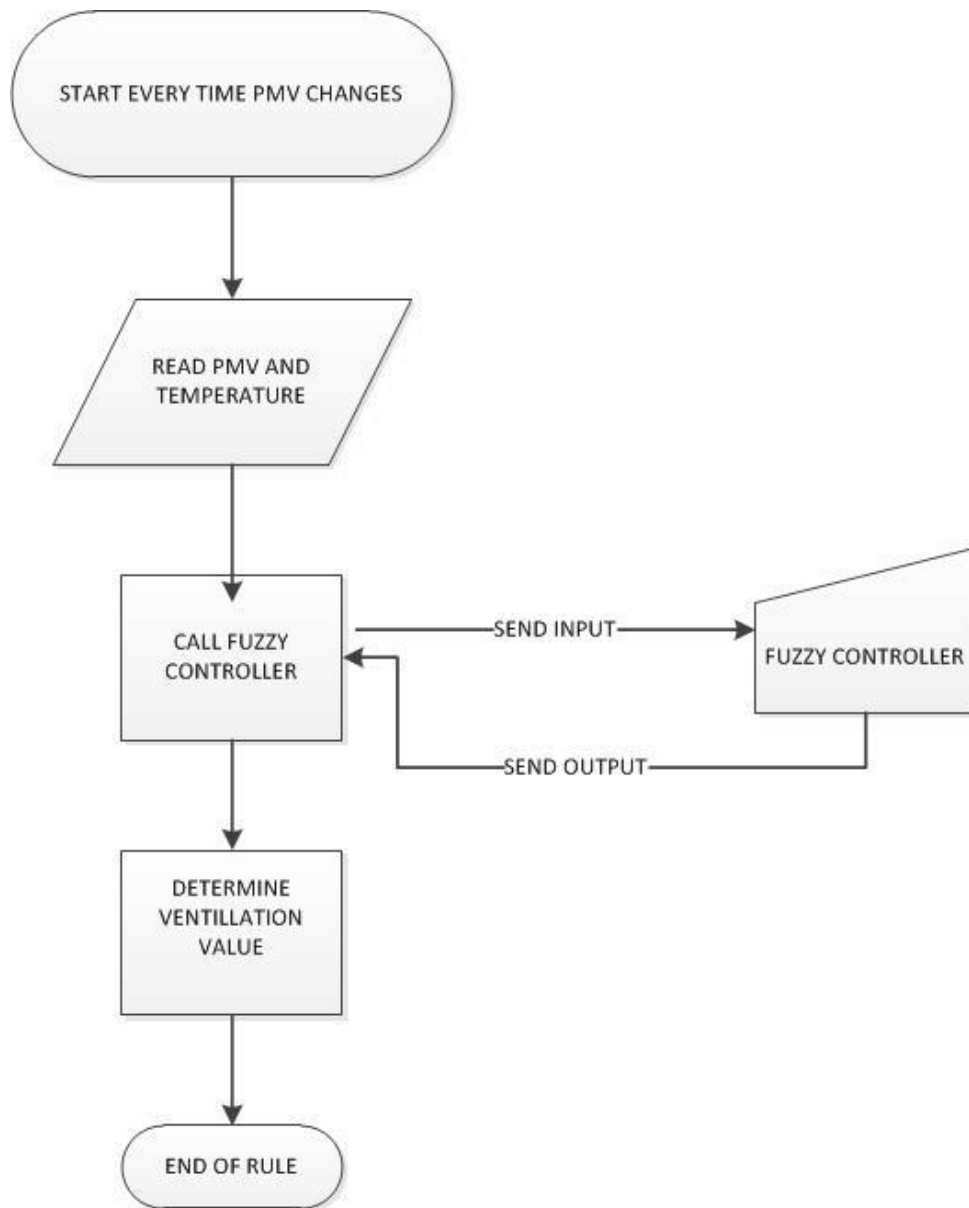


Figure 4.9 Ventilation algorithm's flowchart

As it comes to the .fcl file it was created with the tool **jfuzzylite** which can convert the pre packed .fis file to .fcl. In the next figures, there are shown all fuzzy controller's membership functions of each input and output parameter.

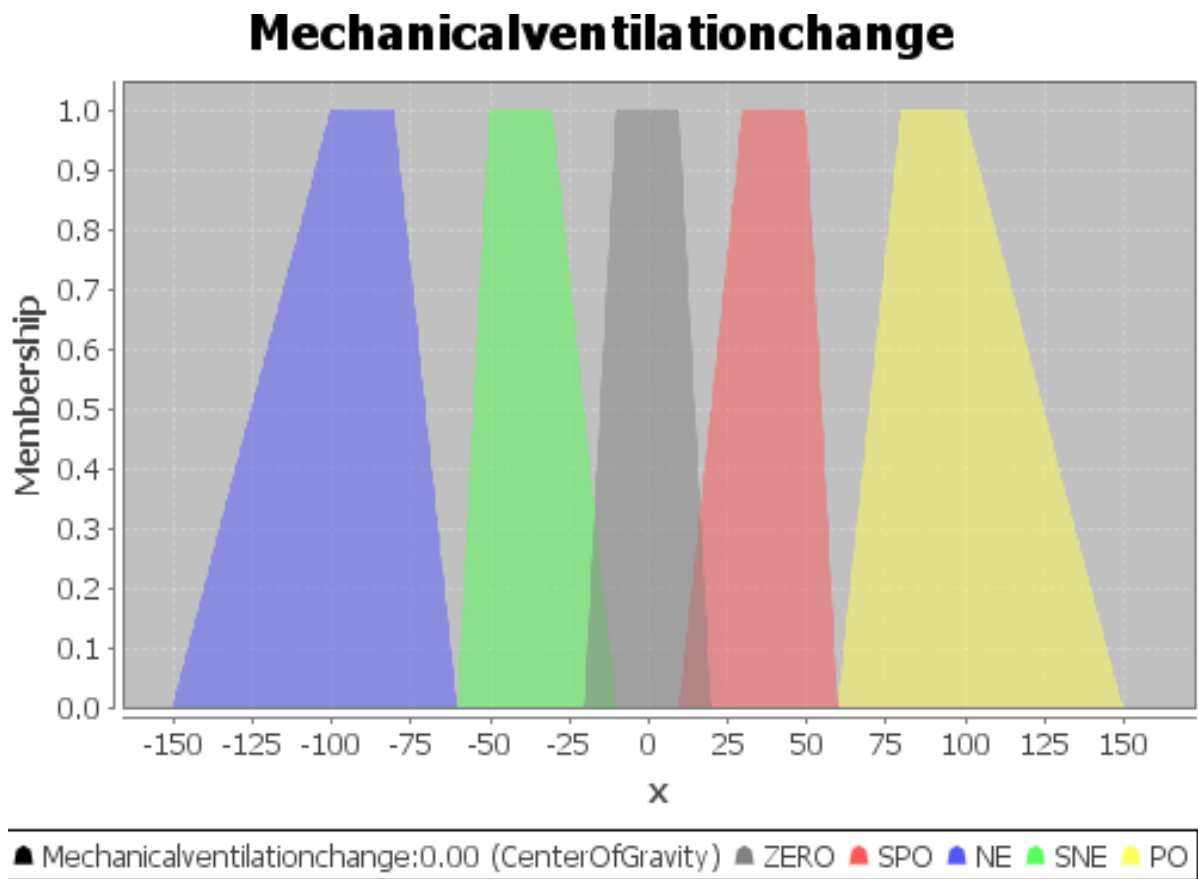


Figure 4.10 Ventilation's membership function, fuzzy controller's output

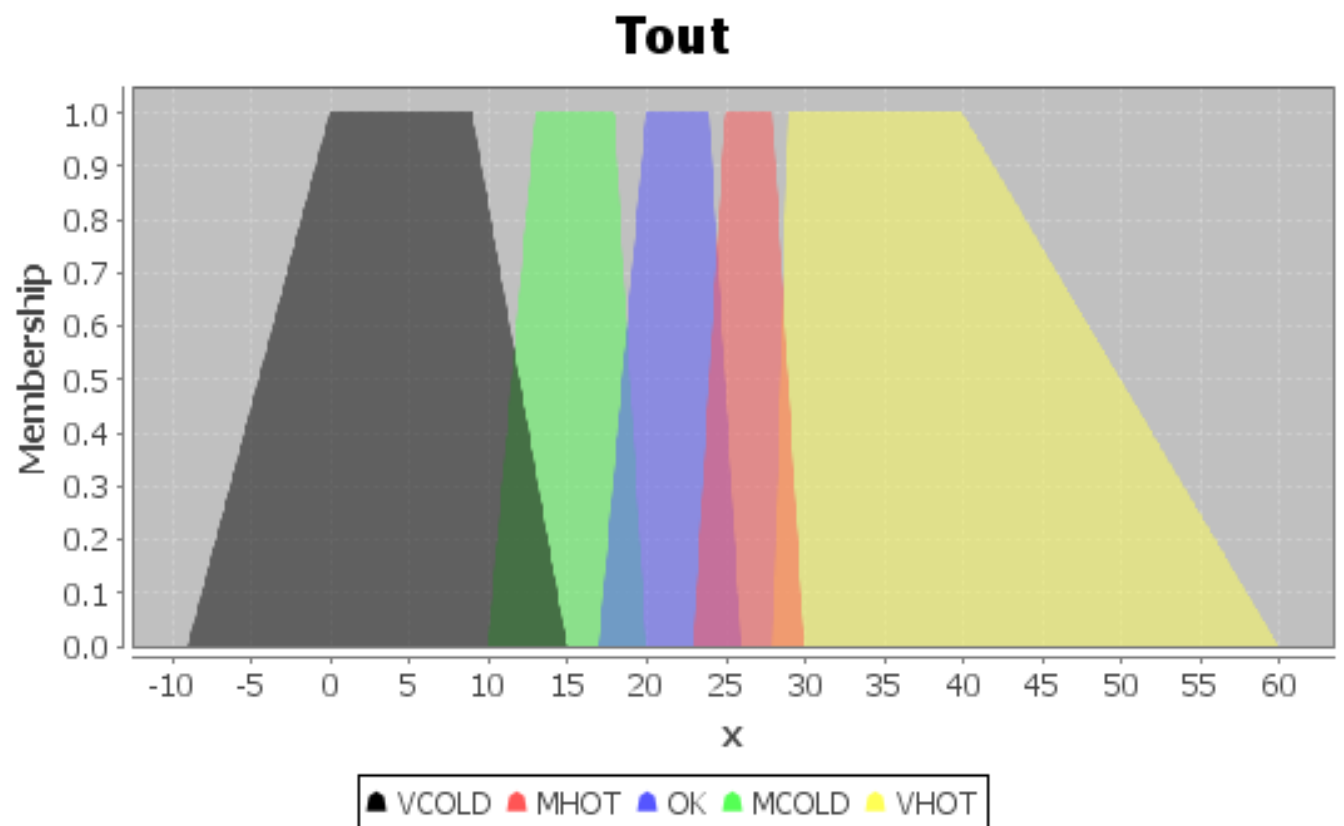


Figure 4.11 Input temperature membership

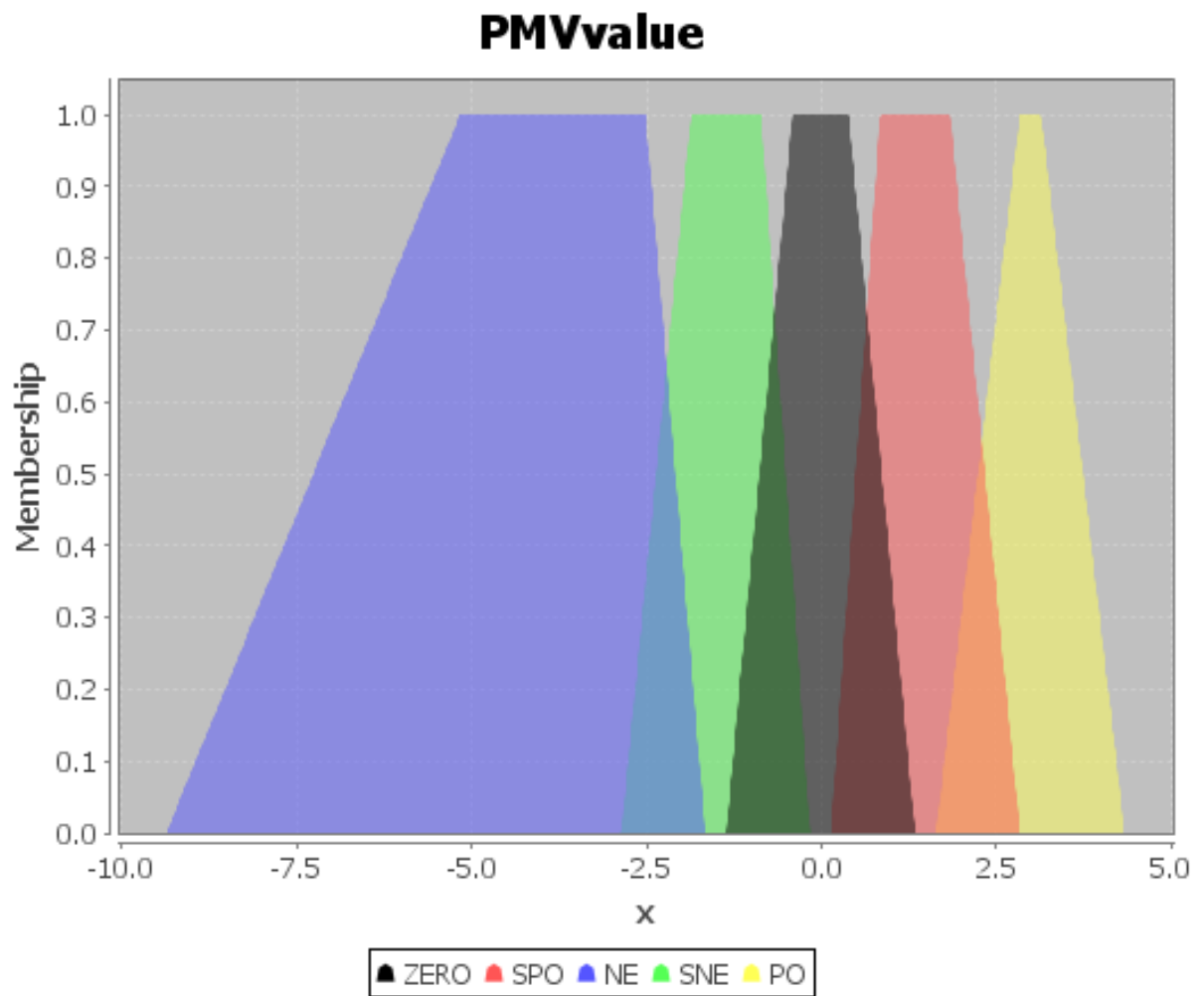


Figure 4.12 PMV's membership function, fuzzy controller's input

In the next graph, we can see how the Ventilation set point is affected by temperature and PMV value.

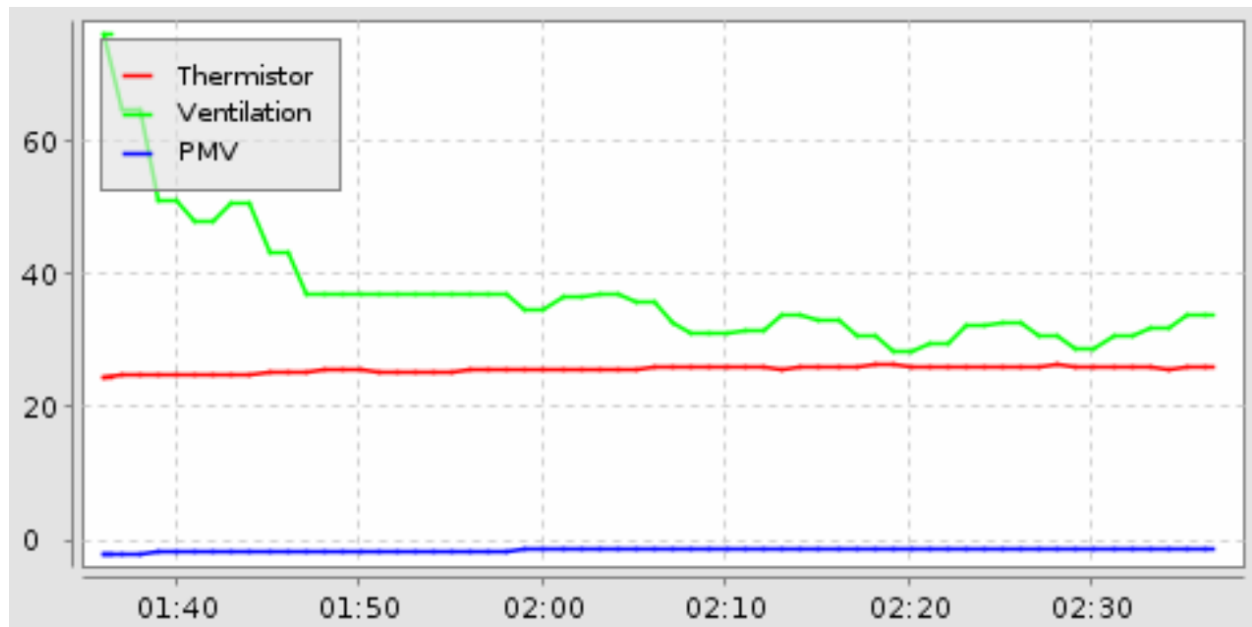


Figure 4.13 Graph of Ventilation and PMV-Temperature

4.2.5 Rule Fuzzy Lights

In this rule we want to control light considering the presence of human and light level. As a result if there is enough daylight lights won't open. Otherwise our fuzzy control takes in consideration if there is any person in the room and the level of daylight to determine if and how many lights should be opened. Our rule was an experiment and it doesn't control in reality lights. Instead it controls two LEDs lights. The control of the LEDs is done using two relays.

So firstly we added to relays in our physical control and two LEDs, one for each relay and were powered by a 9Volt battery. In order to make openHAB able to control the relays we had to add them as switches in the .items file.

```
Switch Relay_2 "Relay2" <energy> (LabEmber) { bacnet="100:4:2" }
```

Furthermore in the .items file another variable was also added a variable for holding the light level which we initialize via another rule when openHAB starts.

```
Number lightlevel "lightlevel [%.3f]"
```

The initialization is done with zero value and it helps so that the algorithm can be executed.

In the .cfg file we add the path to the fuzzy controller so that the openHAB can find it.

The rule runs every minute. Firstly it reads the value of the Presence sensor and the light sensor value. Then the fuzzy controller is called. As an input value, we send a calibrated value of the light sensor. The variable that is checked for determining if LEDs will open is the sum of the fuzzy controller's current and previous output.

In the next figure we can see our algorithm's flowchart

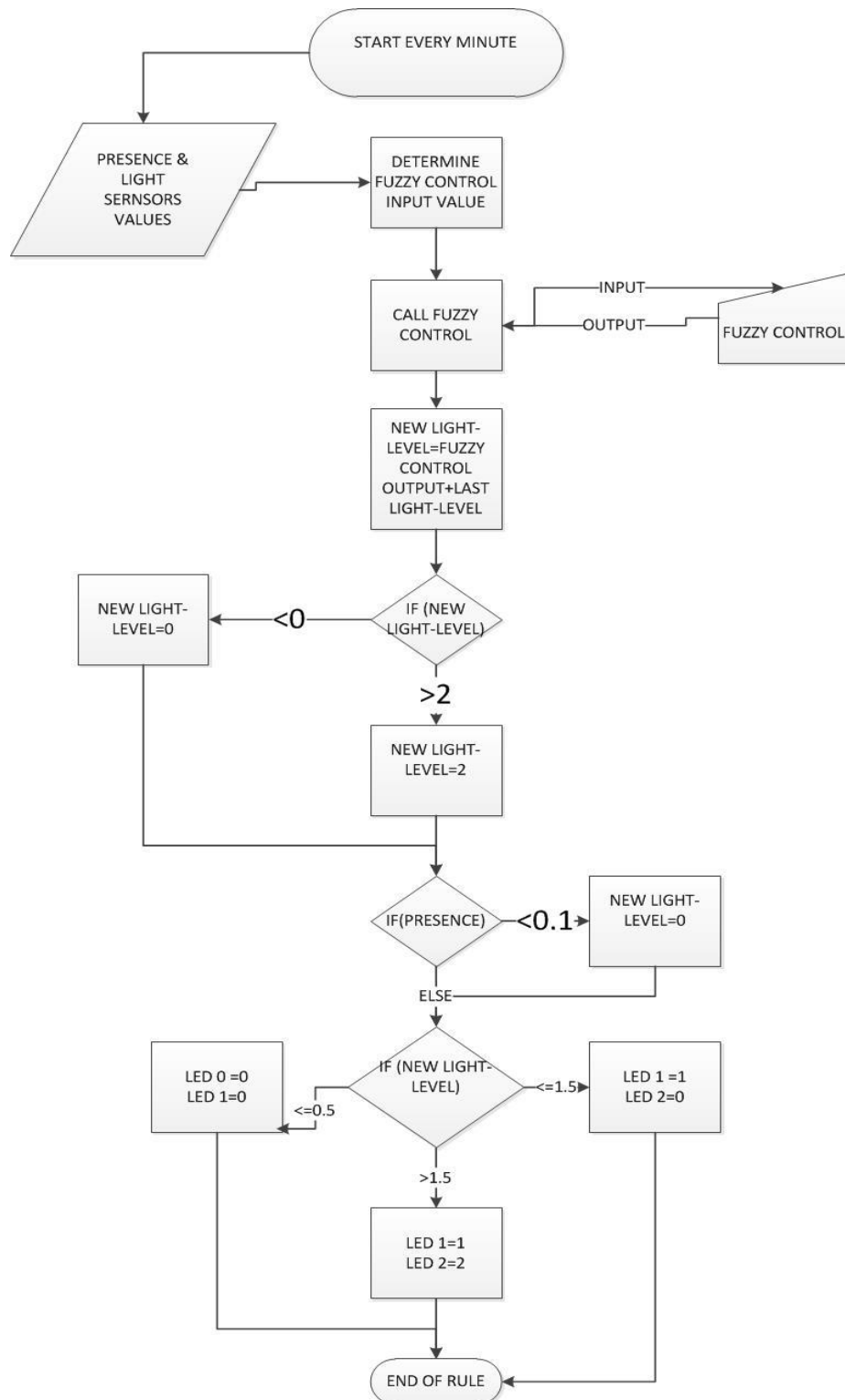


Figure 4.14 Fully lights algorithm's flowchart

As it comes to the .fcl file it was created with the tool **jfuzzylite** which can convert the pre packed .fis file to .fcl. In the next figures, there are shown all fuzzy controller's membership functions of each input and output parameter.

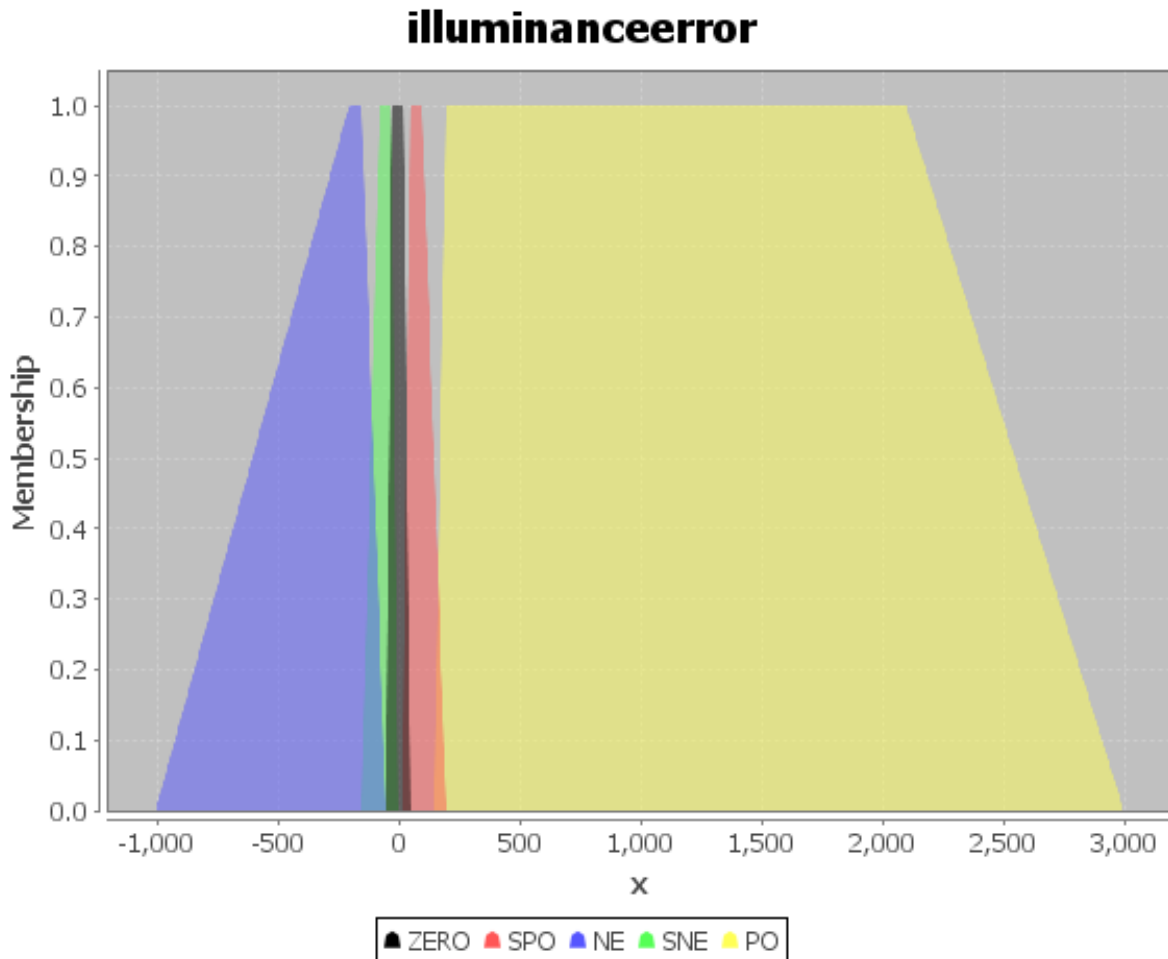


Figure 4.15 Input variable's membership function

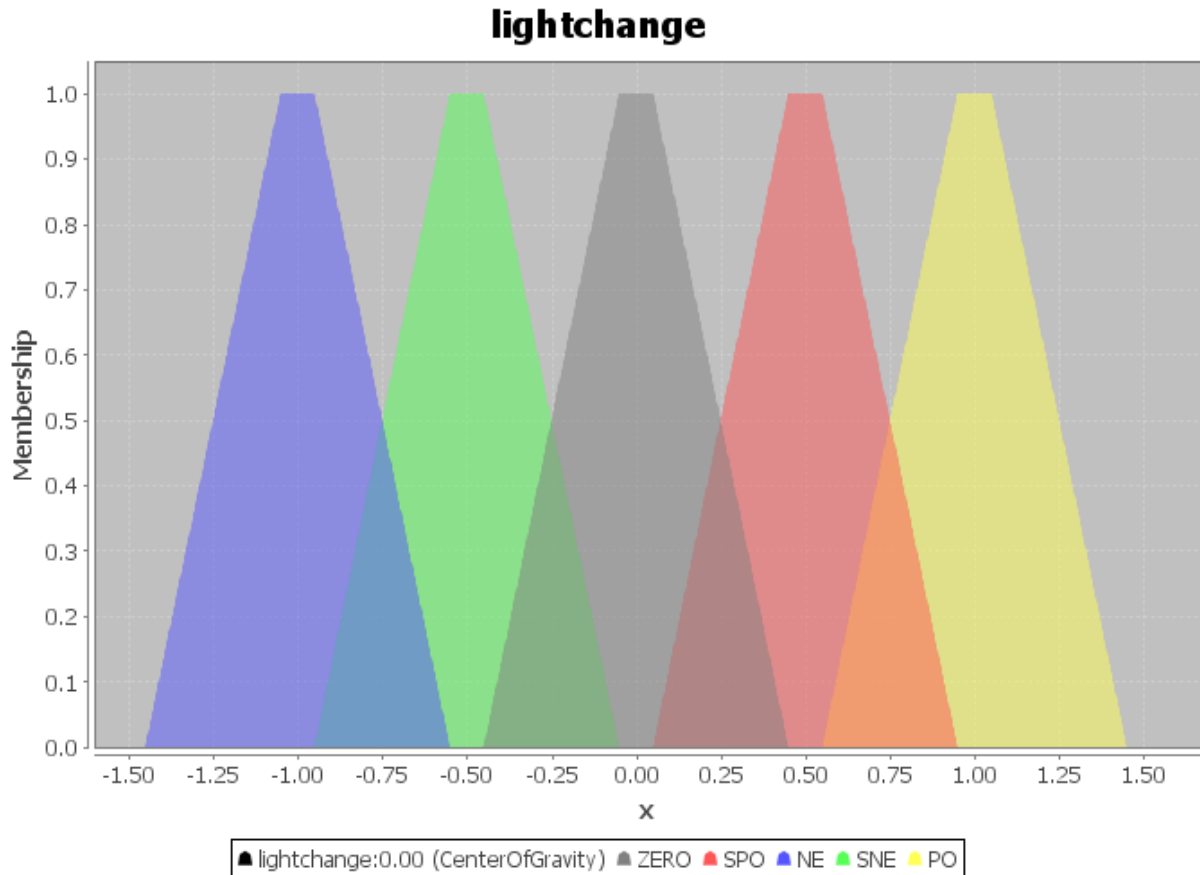


Figure 4.16 Output variable's membership function

4.3 Sitemap

The sitemap is the user interface of the openHAB runtime. It is created as a .sitemap file into the 'Sitemaps' folder. The decided structure was chosen so there is ability to expand the current project. In the first page, user can choose either the building or the graphs. If user chooses one of the graphs, he will be redirected to the particular graph. Otherwise he will be redirected in a page where he can choose the room he wants to check. In this thesis, there was only one room. By choosing this room, user is able to see sensor data, send action to relays and variables' values.

In the figure below, we can see the chosen structure and how it can be expanded

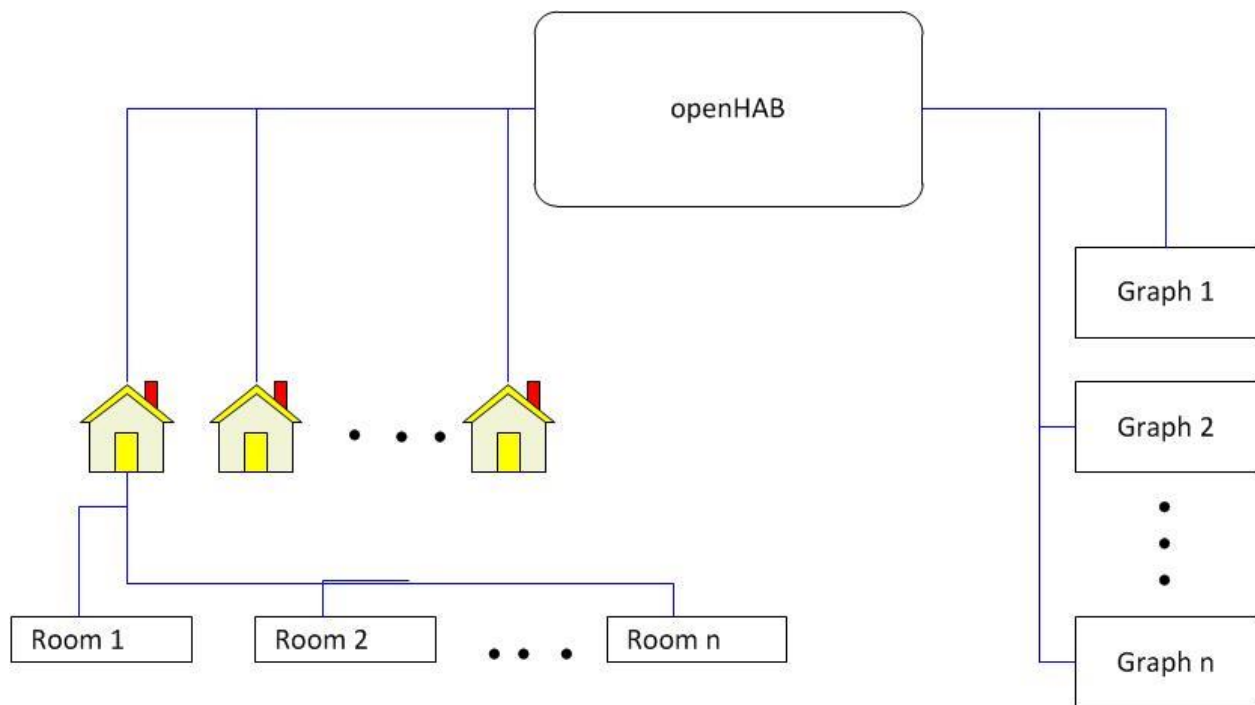


Figure 4.17 Sitemap structure

The next figure shows the first page from which the user can choose between the graphs or the building.

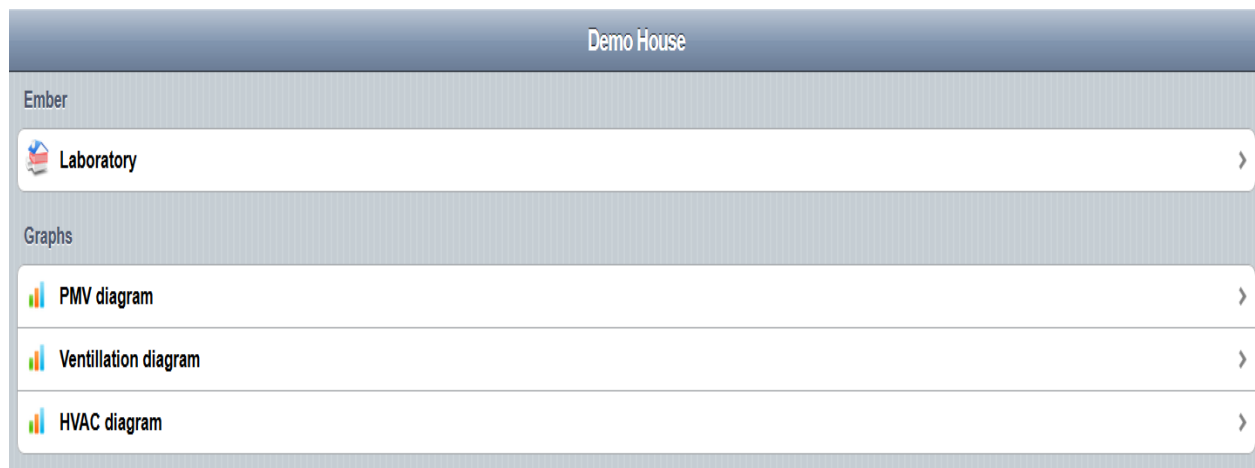


Figure 4.18 Screenshot from sitemap's first page

In the figure below, all data from the EMBER laboratory are shown at a random time.

Laboratory		Home
⚠	Lightsensor	209.07 LUX
⚠	C02_Sensor	693.525
⚠	RH	32.029
⚠	presense	0.985
⚠	Thermistor	27.301
⚠	Relay1	<input checked="" type="checkbox"/>
⚠	Relay2	<input checked="" type="checkbox"/>
⚠	presense analog value	24.000
⚠	windowOnOff	
⚠	doorOnOff	0.000
⚠	HVACSetPoint	0.000
⚠	HVACOnOff	<input type="checkbox"/> 0
⚠	Ventilation	16.021
⚠	PMV	-0.511

Figure 4.19 Sensor data

4.4 Security

One of the major requirements of our system is security. Access must be restricted to people who do not belong to the EMBER lab. So authentication was added as a protections measure. Here, we should mention again that BaCnet does not provide any security (see Sensors and Controller for further information) . So adding authentication is important. The username and password is determined in the `user.cfg` file which is stored in the ‘configuration’ folder. By the time username and password were added, each time openHAB runtime starts, user has to give his credential in a pop-up window, in order to be able to access its features. There is currently no support for different roles in openHAB.

5. Web Site

This section is going to present all aspects concerning the web phase of this thesis. It starts by presenting the configuration of a web server so it is able to manage web content. After that we are seeing all files that constitute the graphic interface and the functionality of our application. In the last part we will analyze the way to make openHAB restarts every time server shuts down and the way we authorize our application to deal with SSL technology.

5.1 Setting up the server

Setting up the server demands finding the physical server, hardware and installing on it the application server as long as other necessary modules.

We got in contact with the Directorate of Telecommunications, Networking and Computing Infrastructure of our university which was able to provide the entire Infrastructure as a Service (IaaS). IaaS is all computer resources complemented by storage and networking capabilities. We were offered a virtual machine (VM) with the following features:

- Operating system : Ubuntu Server 14.04 LTS (x64)
- Processing power: 1 core
- RAM: 4GB
- HD: 150GB

Our given IP:147.27.33.120

Concerning the configuration of the VM, which will permit the hosting of web sites, there is the necessity of installing “LAMP”. A “LAMP” stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the **L**inux operating system, with the **A**pache web server. The site data is stored in a **M**ySQL database, and dynamic content is processed by **P**HP. We received the virtual machine with pre-installed Ubuntu Server 14.04 so there was the need of

installing all the rest of the “LAMP” stack. The installation is done using terminal commands. In order to connect to the VM and use its terminal, we used **PuTTY** [73]. **PuTTY** is a client program for the SSH, Telnet and Rlogin network protocols. These protocols are all used to run a remote session on a computer, over a network. **PuTTY** implements the client end of that session: the end at which the session is displayed, rather than the end at which it runs. For installing every layer of the “LAMP” stack we used **apt-get**. apt-get is the command line package management tool supplied with the Debian package apt. APT is a free software user interface that works with core libraries to handle the installation and removal of software searches. It searches its cached list of packages and lists the dependencies that must be installed or updated automatically so to ensure system stability [74].

So firstly we had to install Apache. Apache is a web server which is currently the most popular in the world. It is a program that serves content using the HTTP protocol and is needed for making a web application work. It waits for requests from web browsers (also known as clients) and responds by sending the required data back. These data are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content. This client-server interaction is the hallmark of how the web works [75].

Next step was to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where our site can store information. It uses SQL which is the programming language for the management of data e.g. execute queries against a database/retrieve/insert/update records from database [76].

Final step was the installation of PHP. It is a server-side scripting language used for making dynamic and interactive Web pages.

For convenience we also installed phpMyAdmin which is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. So it provides all facilities of SQL via a user interface [77].

Now the server is able to host our web site. Before creating the site we had to transfer our openHAB runtime from our laboratory pc to the web server. The transfer was done with FileZilla. FileZilla is an FTP client which is designed to facilitate the transfer of files between two computers over the internet using the FTP protocol [78]. In order to connect to our server via FileZilla we have to configure the connection by giving server's IP address , username and password for authentication and the port that is open for FTP. The next figure is a snapshot of FileZilla

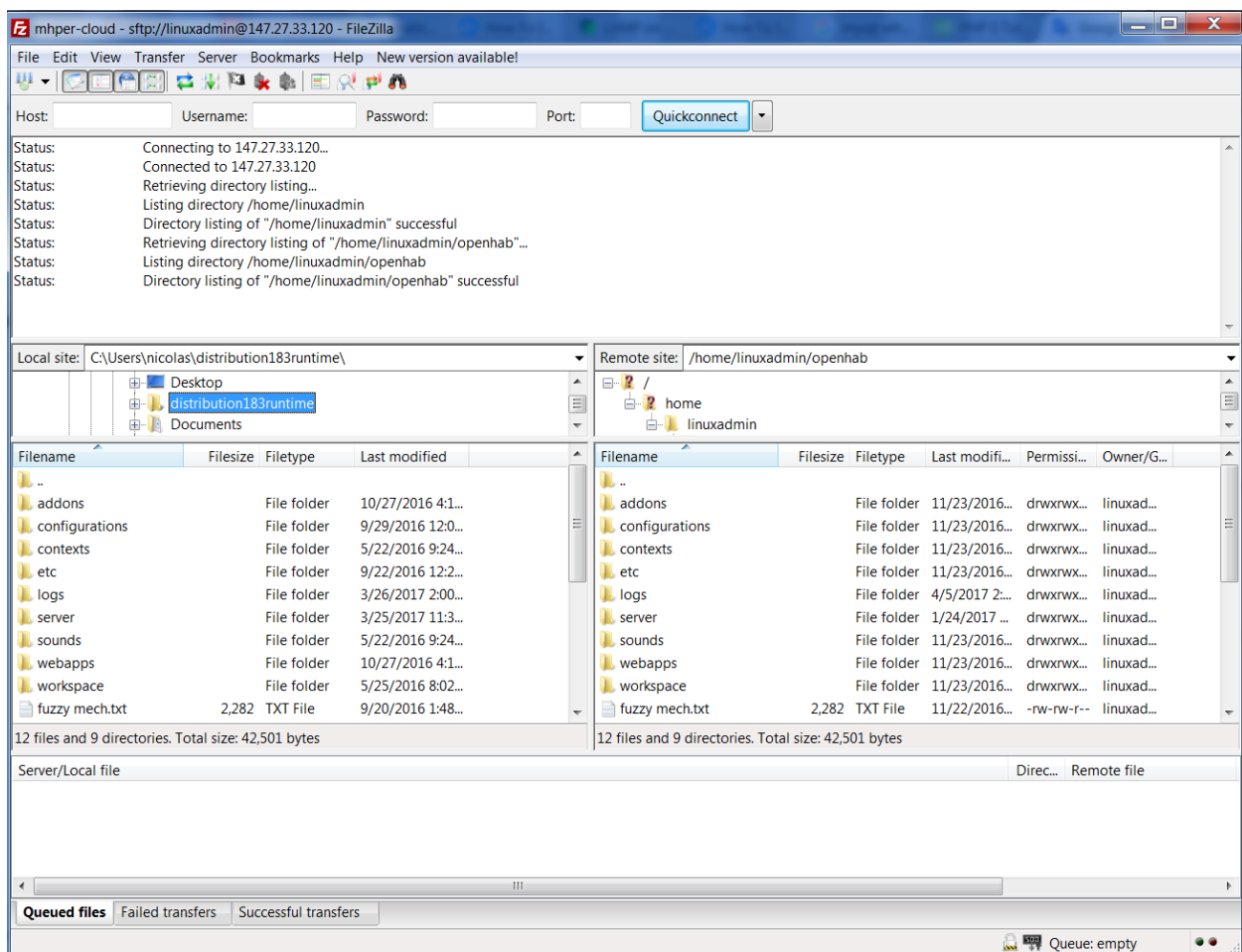


Figure 5.1 Filezilla screenshot

On the left side of the screen is the laboratory's computer repository and on the right is the server's repository.

The execution of the openHAB runtime is done via the same commands that were described in Analyzing its repository . This time the commands are transmitted to the server with Putty. By pointing our browser to the server's IP, we would be able to see the sitemap of the openHAB and access its contents.

5.2 Web content

The site was built taking into consideration security, ease of use, user control, efficiency and expandability. It was created using HTML, CSS, JavaScript, BOOTSTRAP and PHP technologies.

HTML (Hypertext Markup Language) is a markup language for describing the structure of Web pages using ordinary text [79].

CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language. In our occasion, it describes how HTML elements should be displayed [80].

JavaScript is a programming language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. Ok, not everything, but it is amazing what you can achieve with a few lines of JavaScript code [81].

BOOTSTRAP is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites. It speeds up the process of creating a site while maintaining consistency and standards. It also automates the mobility of a site as it enables the adaption of the HTML elements to positions according to the screen inches of the devices accessing the site. Last but not least, BOOTSTRAP uses a grid system as a design pattern which makes it easy to be expanded even by other creator than the creator of the site [83].

PHP It is a server-side scripting language used for making dynamic and interactive Web pages. It handles databases and server files, collects form data and encrypts them.

The site was built as a way for a user to visit the openHAB runtime server in a more structured way. Access to the site is provided via user authentication. Each site user has a role. It can be

administrator or plain user (user). Administrator has full privileges on site's qualifications while user is restricted. These privileges are about configuring openHAB server and site's database. Thus, next page after authentication is the home page where all privileges are exhibited depending on visitor's role. Administrator can access openHAB runtime by clicking on openHAB icon but also he can access HABmin, which is a graphical tool for configuring openHAB server (for more information see HABmin and openHAB designer). There is also the management option. Management is about site's database information. In this section administrator can add or delete user, but also change his current password into a new one.

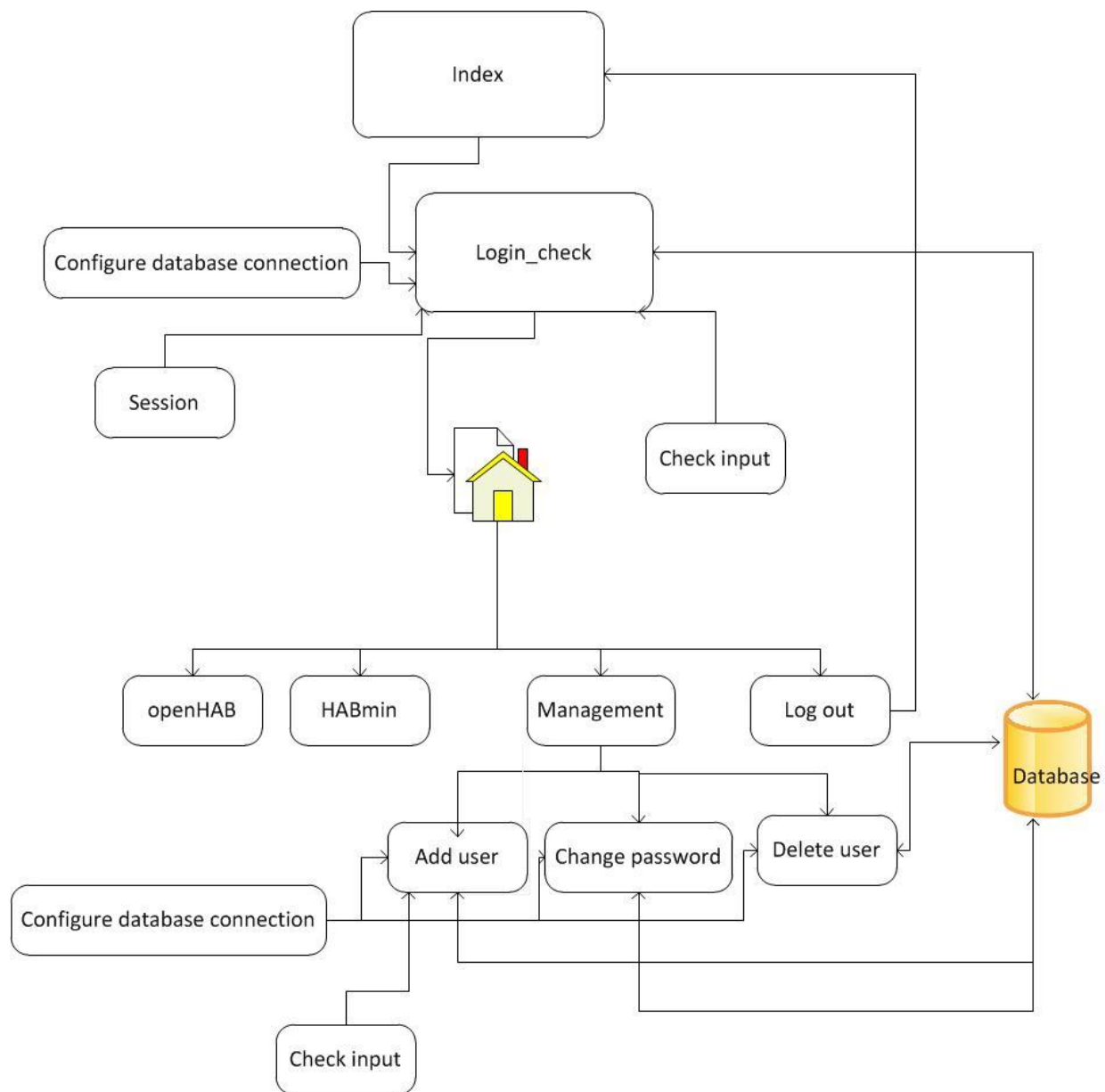


Figure 5.2 Sitemap as seen by administrator

On the contrary, home page provides less possibilities for user. Firstly he can only access the openHAB runtime. This is done because the runtime has an oversight role on openHAB's sensor data and data extracted from rules. So user is not able to change any of the rules or add any new sensors. Management options are also restricted. User cannot affect any other user or administrator. He can only change his password.

Finally there is a log out option available for both administrator and admin which can end up browser and server communication.

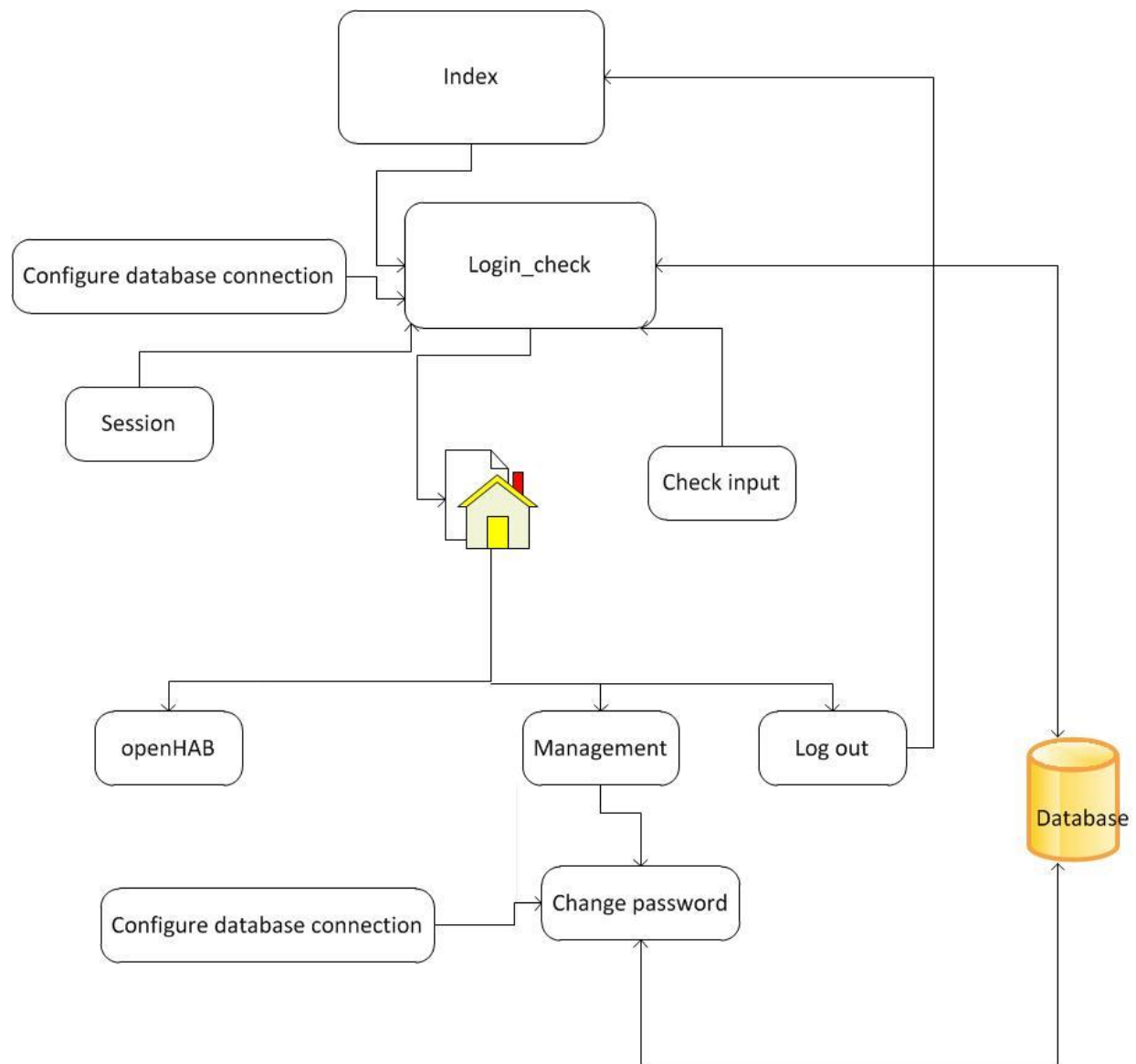


Figure 5.3 Sitemap as seen by user

The web site consists of multiple .php files. These files cooperate together in order to provide the above mentioned functionality. These .php files are

5.2.1 Test_input.php

It contains a function for processing PHP form data. It is included by any PHP file which manages form data. As a result it is a file that cannot be accessed by a visitor and it is used for security purposes. The function contains three sub-functions

- `trim($data)` will strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP `trim()` function).
- `stripslashes($data)` will remove backslashes (`'\'`) from the user input data (with the PHP `stripslashes()` function).
- `htmlspecialchars($data)` converts special characters to HTML entities. (If the user inputs `<` and `>`, `htmlspecialchars()` will translate it to `<` and `>`). This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

5.2.2 Config.php

Config.php is a file that makes the connection to the database. It defines all necessary variables for establishing the connection. These variables are server's name, database name, username and password used for accessing the database. Finally it is included in every .php file that needs to access the database for retrieving, deleting or modifying data.

5.2.3 session.php

This file checks if a user has logged in the application so he is privileged to access the provided content. If user has not logged in, it is forbidden to view any data of the application and he is redirected to the login page. This is done because a user would be able to access all openHAB's features if he knew the proper URL, bypassing the authentication.

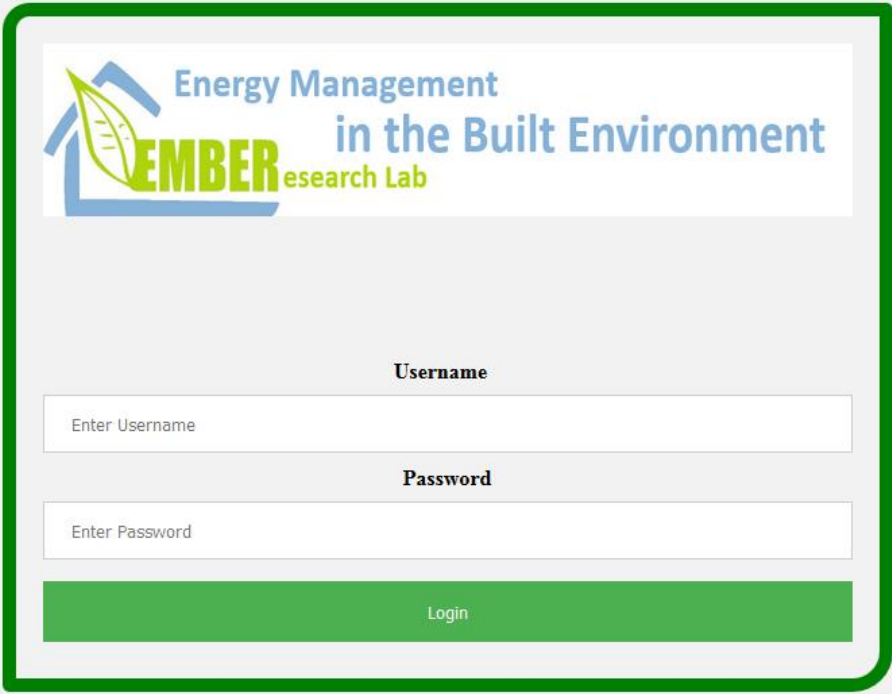
The session.php connects to the database and makes a query on the username. If the username exists it is returned otherwise user is redirected to login page.

The file is included in every page which has restricted content.

5.2.4 index.php

Index.php is the first page a user encounters. The page includes a log in form which asks for user credentials in order to access the web application. When user enters his credentials and hits the 'login' button index.php file sends the form data for processing to a file called login_page.php . The form data is sent with the HTTP POST method.

There are currently two methods for sending data, **GET** and **POST**. Data sent via GET method are visible to everyone as they are displayed in the URL. GET may be used for sending non-sensitive data. On the other hand POST method encrypts the data which are embedded within the body of the HTTP request.



The image shows a login form for a web application. At the top, there is a logo for 'Energy Management in the Built Environment' with 'research Lab' underneath. Below the logo, the text 'Energy Management in the Built Environment' is displayed in a large, blue, sans-serif font. Underneath this, the words 'research Lab' are written in a smaller, green, sans-serif font. The form consists of two input fields: one for 'Username' and one for 'Password'. Each field has a placeholder text 'Enter Username' and 'Enter Password' respectively. Below the input fields is a green button labeled 'Login'.

Figure 5.4 Login page

In order to save time from redirecting to the login_page for the proccession of the data and for making our site able to display error messages on the same page of the form we include the login_page.php into our file and also use `$_SERVER["PHP_SELF"]` variable. So, the `$_SERVER["PHP_SELF"]` sends the submitted form data to the page itself, instead of jumping to a different page.

After the procession of the input data user will be redirected to the main menu (redirection is done in the login_page.php) or an error message will be displayed on screen.

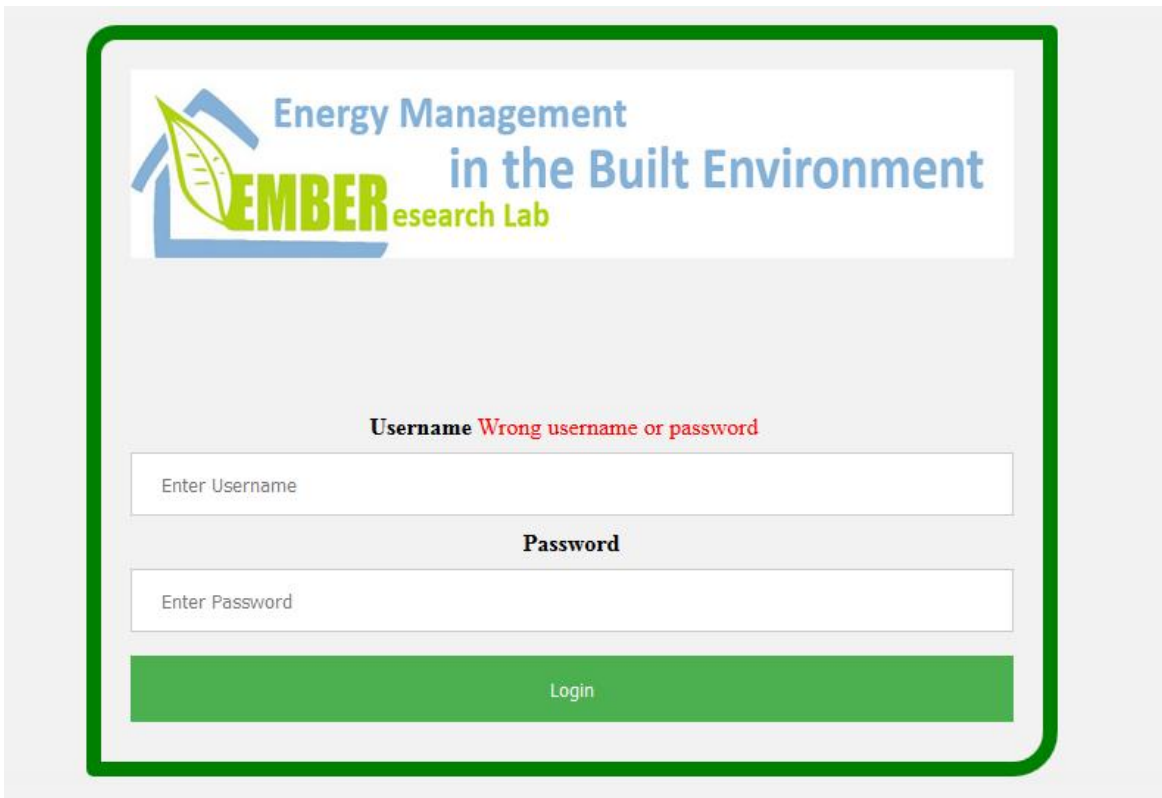
The image shows a web browser window displaying a login page. At the top, there is a logo for 'Energy Management in the Built Environment' with 'EMBER research Lab' underneath. The logo features a stylized house and leaf. Below the header, a red error message reads 'Username Wrong username or password'. The login form has two text input fields: 'Enter Username' and 'Enter Password'. Below these fields is a green button labeled 'Login'. The entire form is enclosed in a green rounded rectangle.

Figure 5.5 Login page with error message

5.2.5 login_page.php

This is the file included in the index.php and is responsible for processing the data of the authentication form. Firstly it includes testinput.php and config.php . With config.php it starts a connection with the database in order to query for existence of username and password. If connection cannot be established it outputs an error message. Afterwards it checks if there is a POST submission and username and password is not null. Then we use the method of testinput.php in order to check username and password so that there is no risk of exploitation. After that we can query our database and find out if user exists and what privileges does he have. There are two roles user can have, plain user and administrator each of which has different privileges. If user doesn't exist, it outputs error message.

5.2.6 Mainmenu.php

This is the main page of our site. There are listed all options a user has according to his role. The page was created using BOOTSTRAP framework. In order to use BOOTSTRAP we have to import it in the head segment of HTML. For security reasons we also include session.php which checks if the user has made a log in.s

In the figure below we can see a screenshot of our web page on administrator mode.

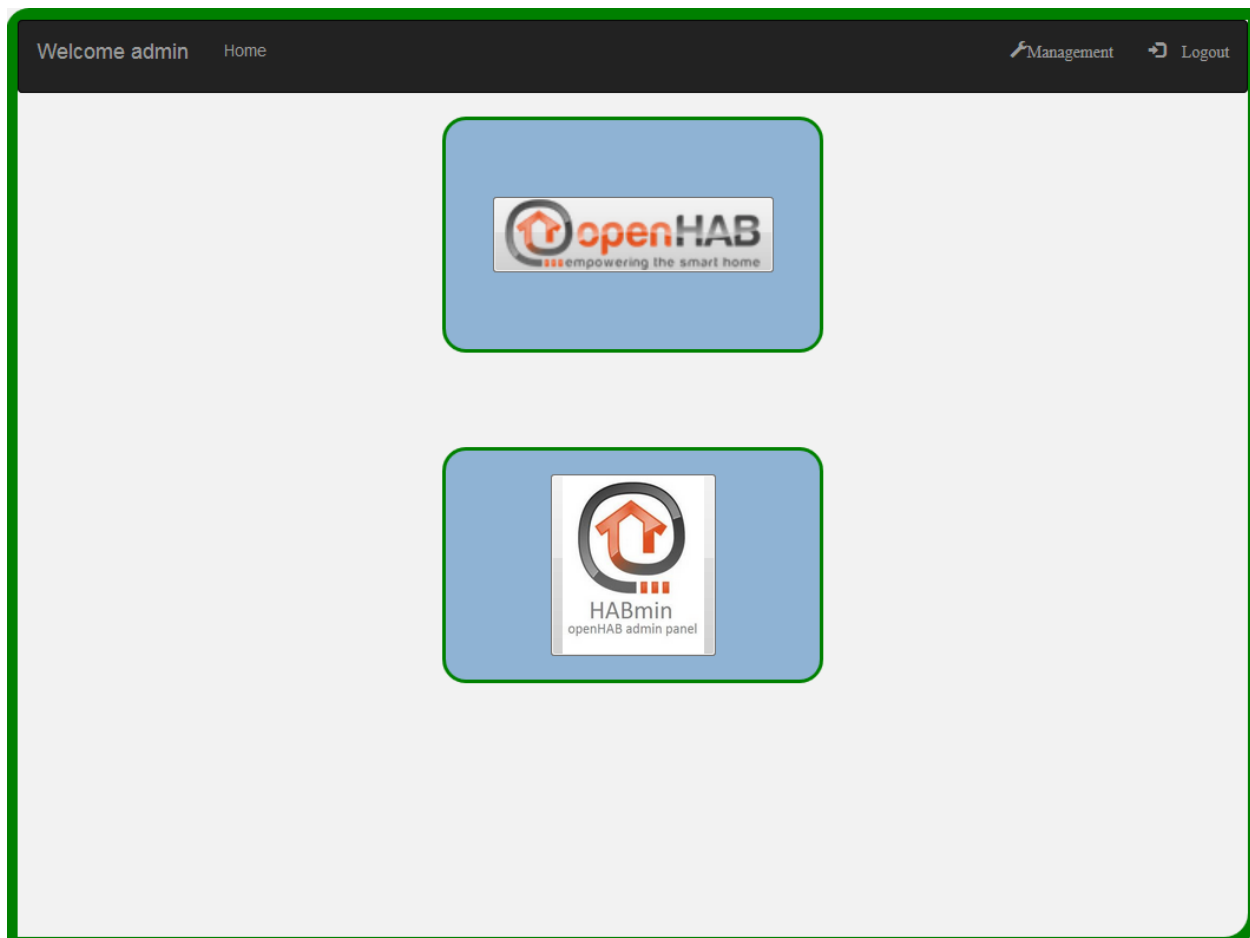


Figure 5.6 Web application's main page as seen by administrator

On the upper part there is a navigation bar. From left to right, we can firstly see a welcome message with the name of the user. The name is saved in our session page. Next we can see a

link button (“Home”) to the same page which is used as a path. On the right side we can see a link to the management page where user can add/delete user and change password. Finally there is a log out button for exiting from the web application, destroy the session and return to the index.php page. The navigation bar was created using the BOOTSTRAP class " `navbar navbar-inverse` " .

Then we use the class "`container cont`" in order to create a division which will contain the appropriate options. By using classes "`row`" and "`optionbox col-sm-4 col-sm-push-4` " we make a grid into our container which will help us place our options in a stylish way, be more adaptive to the different screen sizes of devices used to access the page but also make it easily expandable for future options.

For the first option we create a button with the openHAB trademark which is a reference to `openhab_page.php`. There the user will be able to access the sensors’ data. Below the openHAB image there is the HABmin image. This belongs to the next row of our grid. Because only administrators have access to HABmin as it gives the ability to create new rules, modify or delete older rules or sensors, we make a control so that if the user is not an administrator, button won’t be shown.

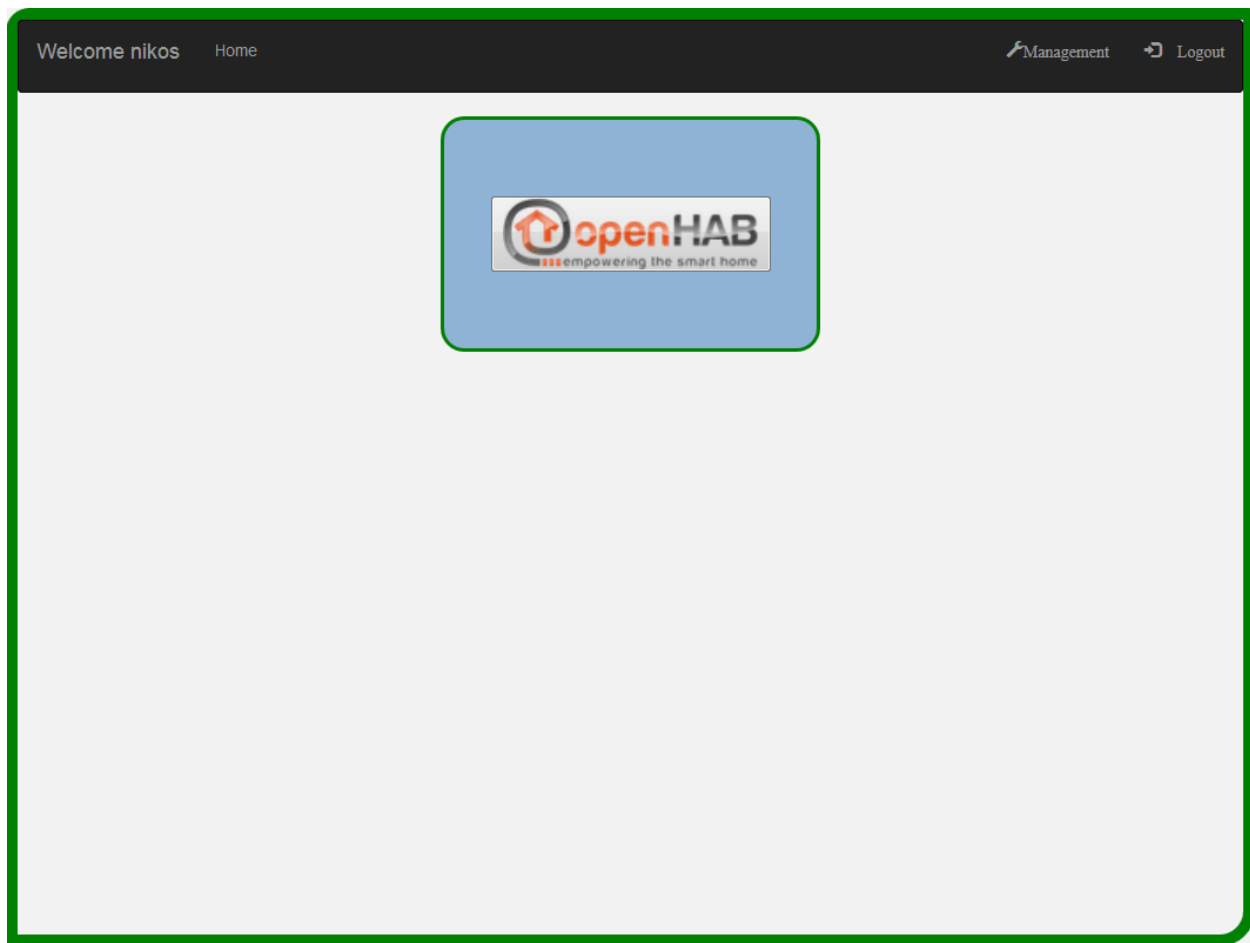


Figure 5.7 Web application's main page as seen by user

5.2.7 openhab_page.php

This page has the access to the openHAB runtime features. When we access the page for the first time, a pop up window will come up asking for username and password. This is asked not from our web application but from the openHAB runtime. So user must insert the username and password for openHAB which may be different from the ones given in our login page.

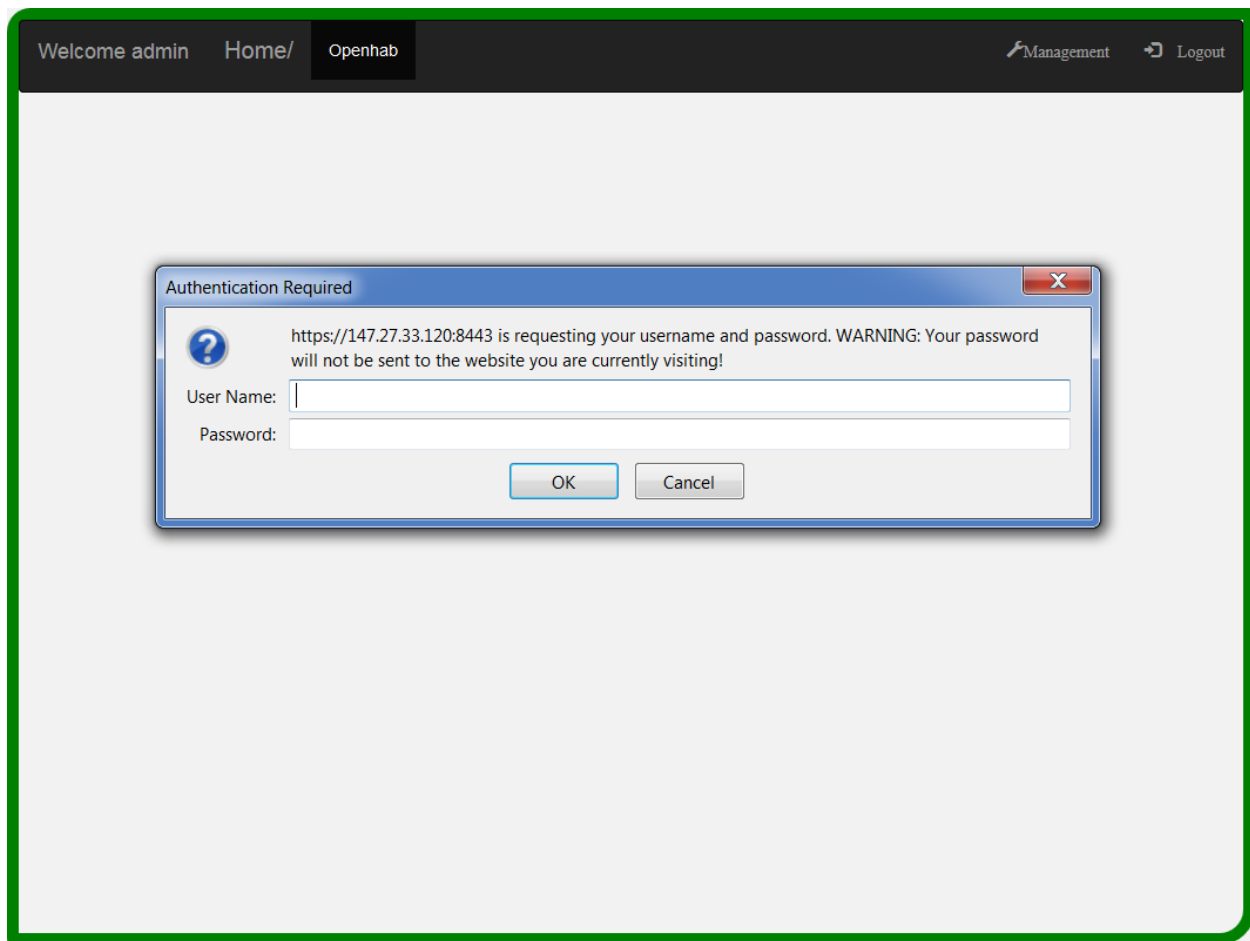


Figure 5.8 Pop up window asking for credentials in order to access openHAB

After the authentication, user will be able to see and access the openHAB runtime through out web application.

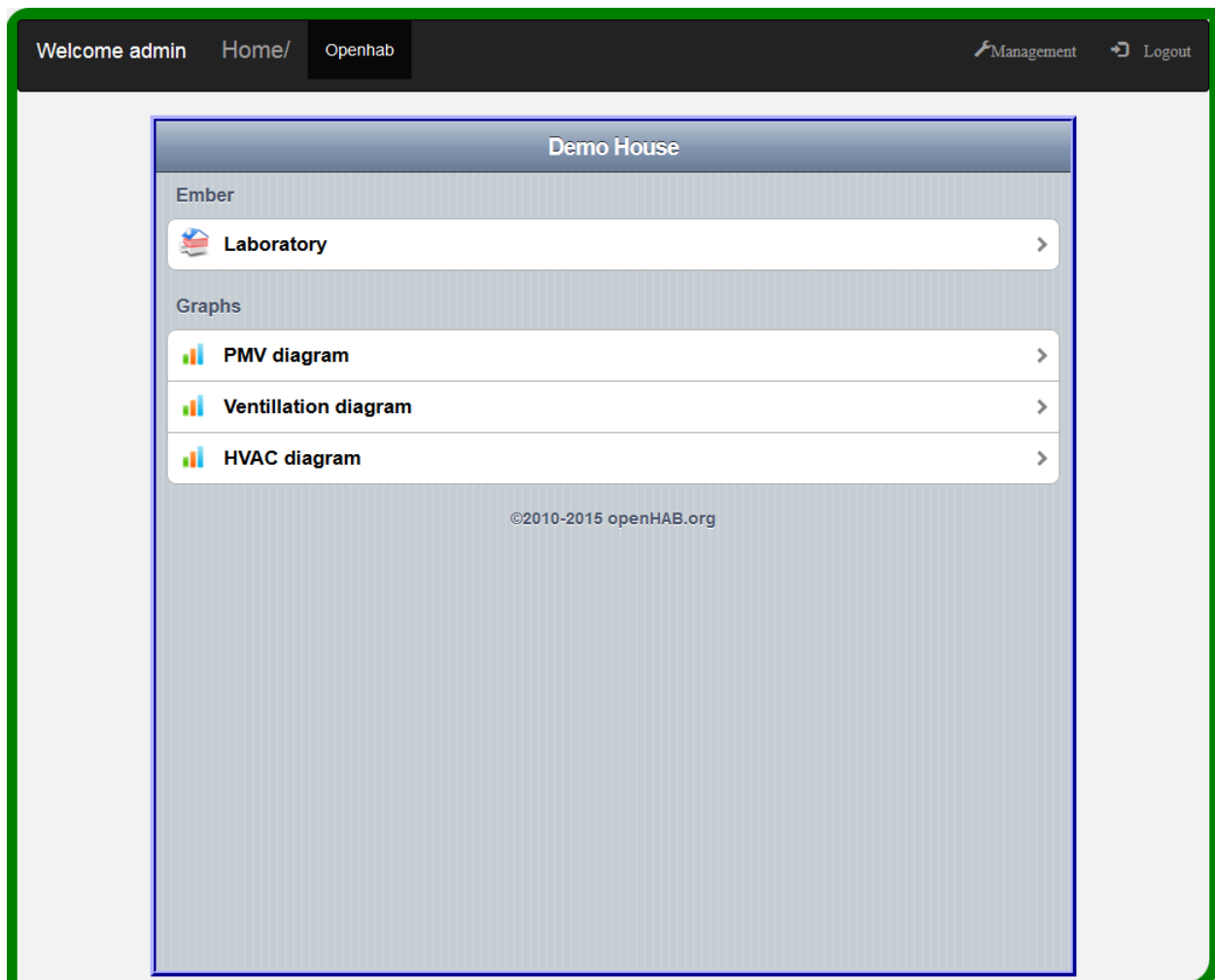


Figure 5.9 OpenHAB runtime's main page as seen in the web application

The way the page is structured is the same as in `mainmenu.php`. There is a navigation bar on the top that has the same options. The only difference is that path is extended up to the site that we currently are. The container is also separated as a grid but in our occasion it has only one row with one element holding a reference to the openHAB runtime. User can access the application the same way as was described in Sitemap. For security reasons `openhab_page.php` includes `session.php`.

5.2.8 habmin_page.php

This page is restricted to users that do not have administrative responsibilities. It has the same concept of the previous page, but instead of openHAB runtime user can see HABmin. HABmin

is a web administration console for openHAB. For further information on HABmin you can see [HABmin](#) and [openHAB designer](#). There is a navigation bar with a path that ends up to the current page. The container has one row with an element holding a reference to the HABmin page. For security reasons `habmin_page.php` includes `session.php`.

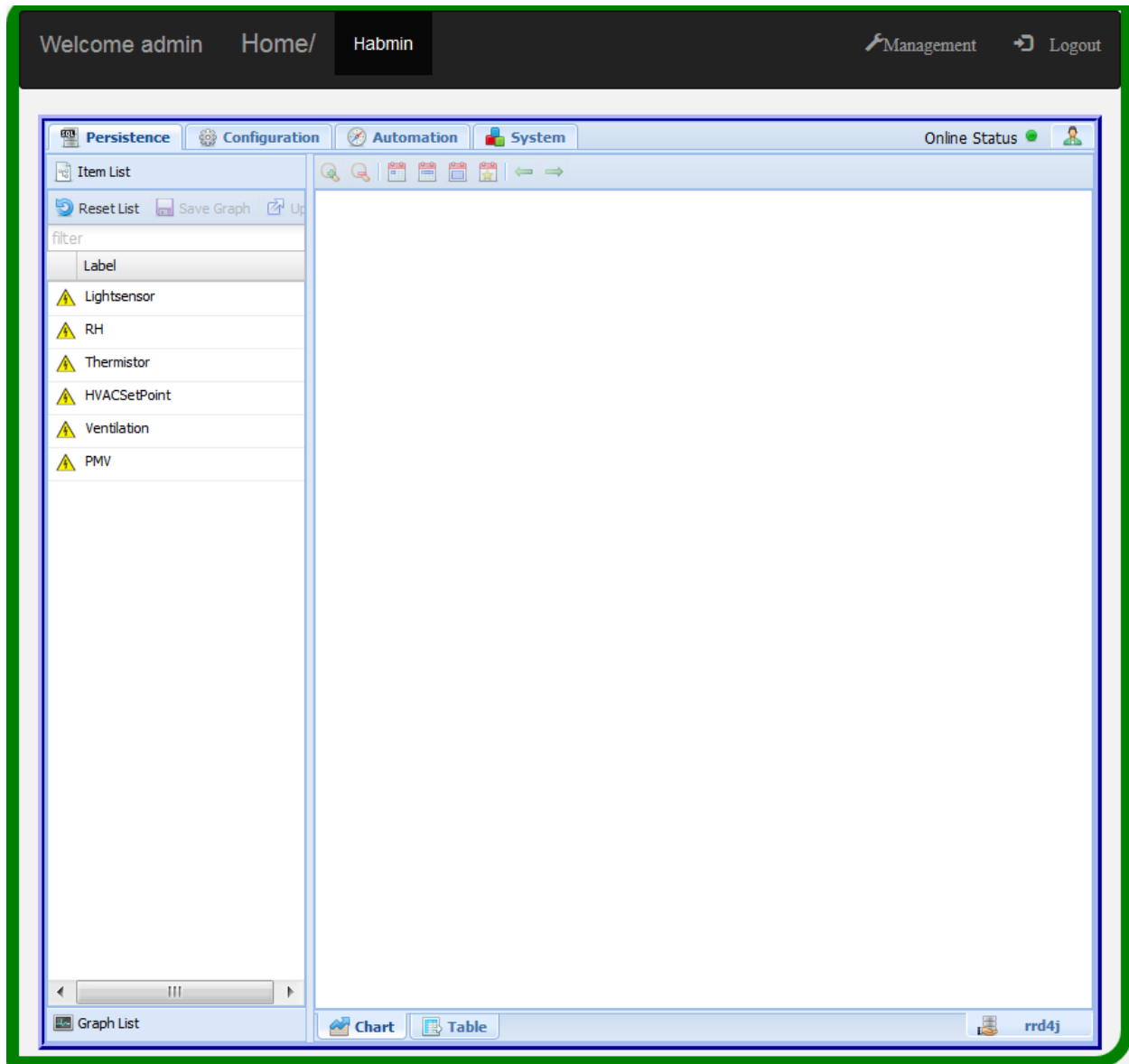


Figure 5.10 HABmin in the Web application

5.2.9 management_page.php

Management_page.php is the page responsible for managing personal data and adding/deleting users. Each user according to his role has different privileges. Administrator can change his password and add/delete users. Plain user can only change his password.

The page consist of the navigation bar that is found in every page and a division. The division has dynamic tabs. Tabs are used to separate content into different panes where each pane is viewable one at a time. Firstly we have to create the menu/tabs which In HTML, is defined as an unordered list ``. To make the tabs toggleable, we add the `data-toggle="tab"` attribute to each link. Then add a `.tab-pane` class with a unique ID for every tab and wrap them inside a `<div>` element with class `.tab-content`. In order to make the tabs fade in and out when clicking on them we add the `.fade` class.

For the administrator mode, the first option is the import of new user in the database.

Welcome admin Home/ Management Management Logout

+ Add User Change Password X Delete user

New user's credentials

Username:

Password:

Role: ☒ Administrator ☐ User

Figure 5.11 Creation of new user

We use a form in which user can add the credentials of the new member. There is also the option of giving a role to the new member. This is done using radio buttons. The import of the new member is completed in `addauser.php` file. Data are sent there using POST method. A pop up message comes up to inform if the process has been completed.

The screenshot shows a web application interface for user management. At the top, a dark navigation bar contains the text "Welcome admin", "Home/", and "Management" (which is highlighted). On the right side of the navigation bar are links for "Management" and "Logout". Below the navigation bar, there are three buttons: "Add User" (with a plus and user icon), "Change Password" (with a lock icon), and "Delete user" (with a minus and user icon). A green success message banner states "User has been added successfully". Below this, the section "New user's credentials" contains a form with three fields: "Username:" with a text input containing "Enter Username", "Password:" with a text input containing "Enter password", and "Role:" with two radio buttons, "Administrator" (selected) and "User". A "Submit" button is located at the bottom of the form.

Figure 5.12 Feedback on insertion on new user

Next option is change of password.

Welcome admin Home/ Management Management Logout

+ Add User Change Password × Delete user

Enter your credentials

Current password:

New password:

Re-type new password:

Figure 5.13 Change of password form

There is a form asking for current password, for security reasons, new password and retyping new password so the user is sure that he has given the desired password. User is informed about the progress of the process via an alert message. The message is triggered by `changepassword.php` file. There is an if statement that checks if there is any alert message. If statement is true, a message is showed up for two seconds. The alert message is initialized in `changepassword.php`.

The screenshot shows a web application interface with a dark header bar. On the left, it says "Welcome admin" and has links for "Home/" and "Management". On the right, there are links for "Management" and "Logout". Below the header, there are three buttons: "Add User" (with a plus icon), "Change Password" (with a lock icon), and "Delete user" (with a minus icon). A green message box states "Password has been changed successfully". Below this, a section titled "Enter your credentials" contains three input fields: "Current password:", "New password:", and "Re-type new password:". A "Submit" button is at the bottom.

Welcome admin Home/ Management Management Logout

+ Add User Change Password × Delete user

Password has been changed successfully

Enter your credentials

Current password:

New password:

Re-type new password:

Figure 5.14 Feedback message on change on password

Finally there is the delete user option.

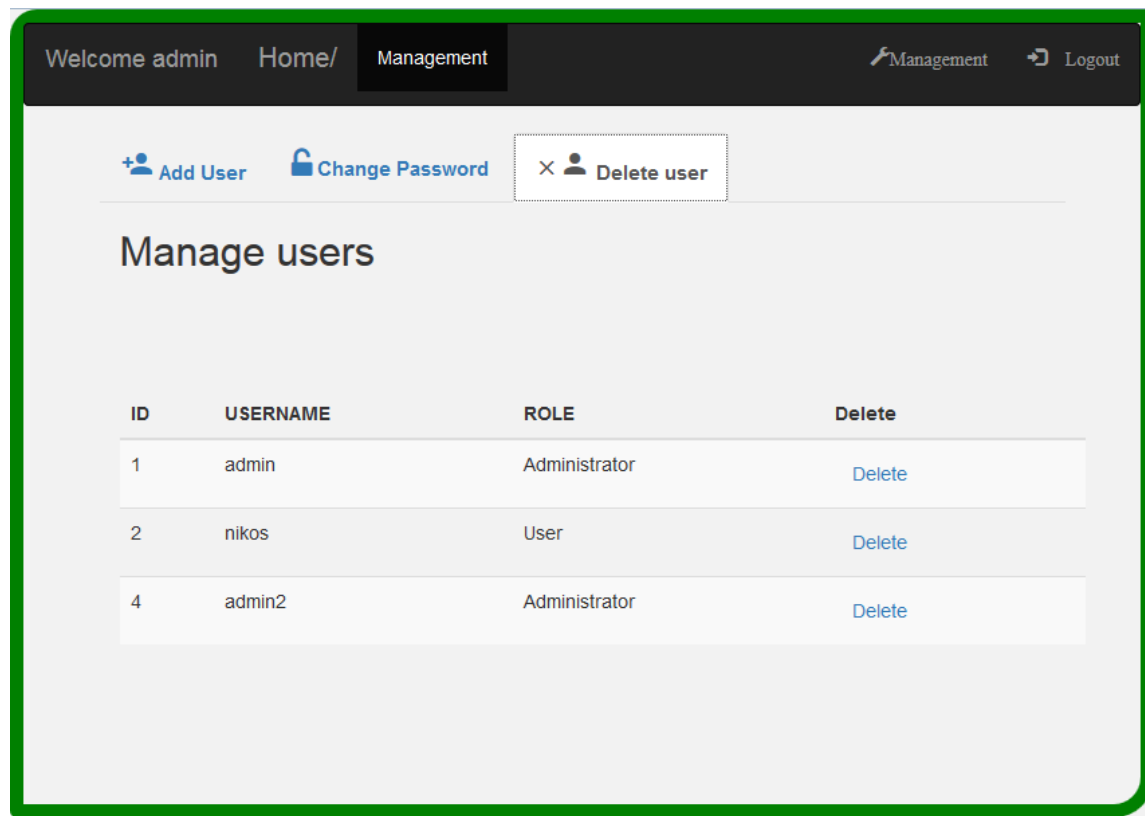


Figure 5.15 Screenshot of delete user option

We create a table which contains all users, with their roles and a button for deleting them as we can see in the above picture. The table is created by connecting to the database and importing all users from there. When delete button is clicked a dialog box shows up asking for confirmation. This is done so that the user is sure about his action.

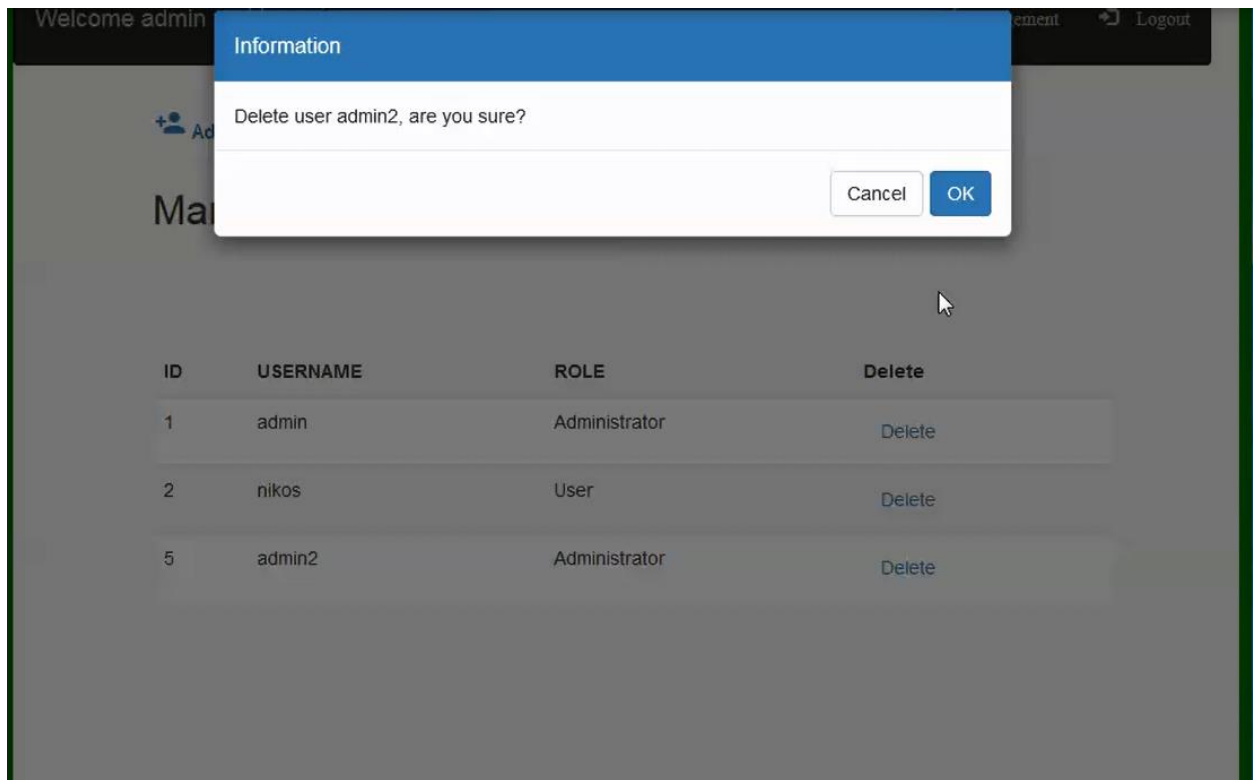


Figure 5.16 Confirmation pop up message on deleting a user

The process of deletion is completed in `deluser.php` file. The progress of the process is informed to the user by an alert message that shows up for two seconds and which is initialized in `deluser.php` file.

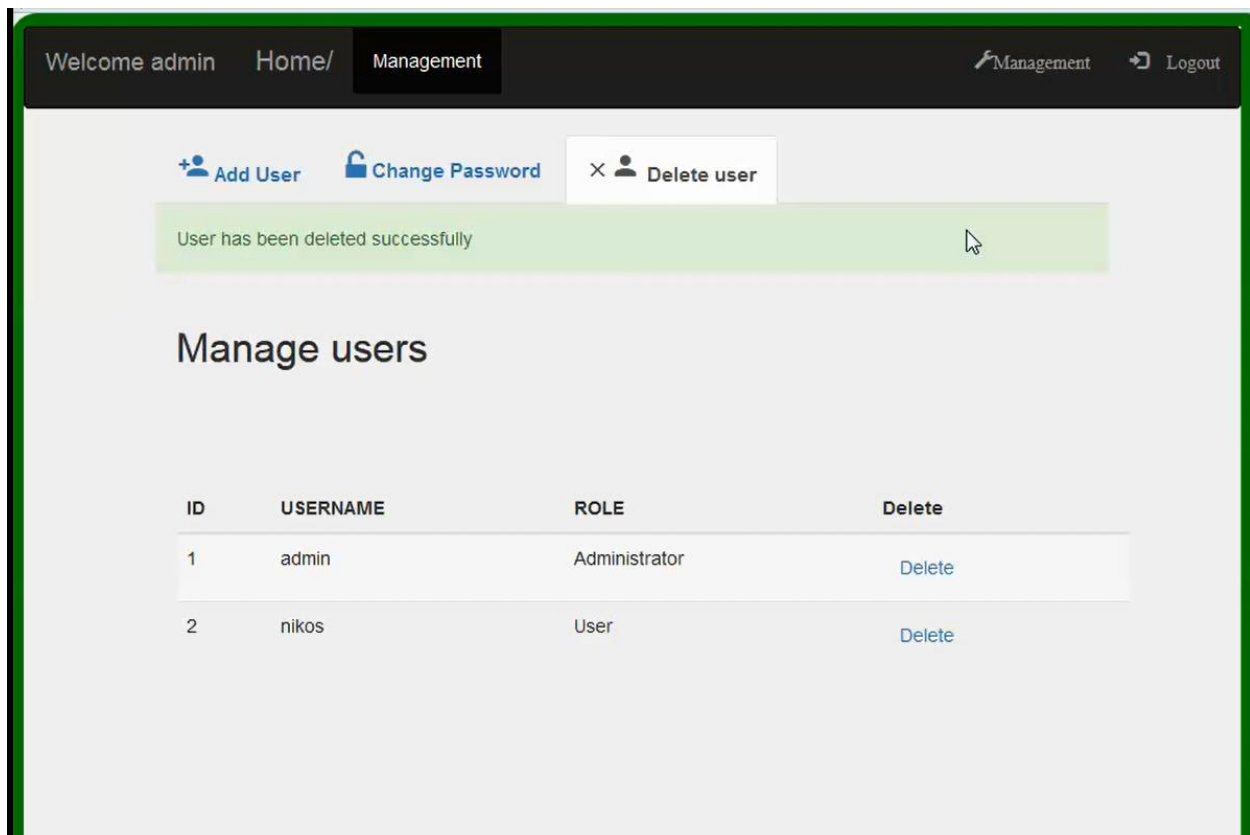


Figure 5.17 Feedback message on deleting a user

As it comes to the plain user, as said before, this kind of user can only change his password.

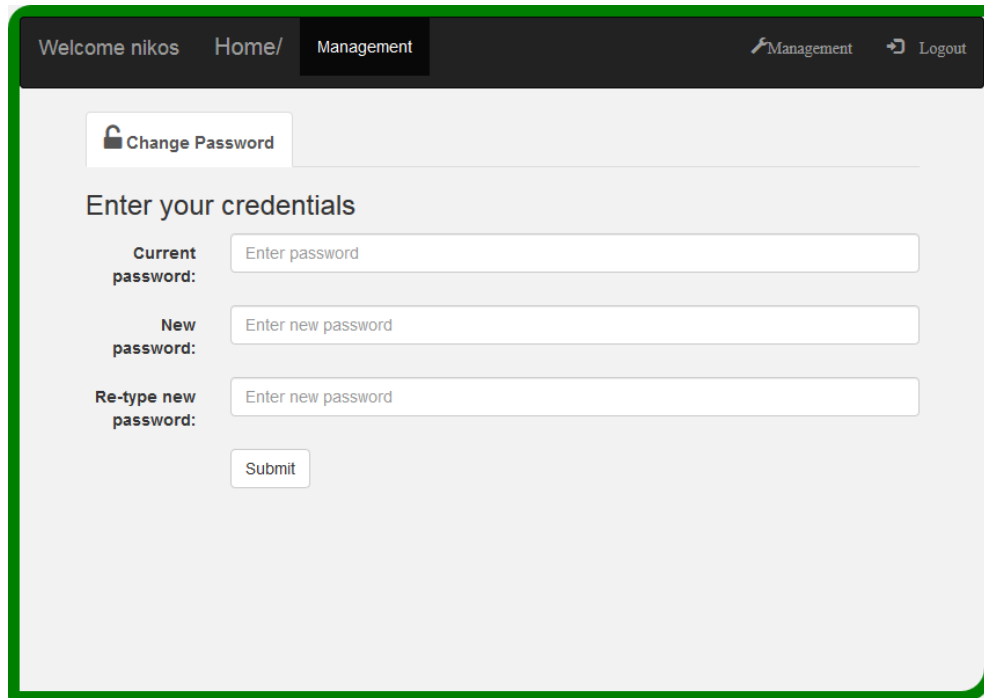
The image shows a web application interface for a 'Management' page. At the top, there is a dark navigation bar with 'Welcome nikos', 'Home/', and 'Management' (highlighted). On the right of the bar are links for 'Management' and 'Logout'. Below the bar, a 'Change Password' button with a lock icon is visible. The main section is titled 'Enter your credentials' and contains three input fields: 'Current password:', 'New password:', and 'Re-type new password:'. Each field has a placeholder text 'Enter password' or 'Enter new password'. A 'Submit' button is located at the bottom of the form.

Figure 5.18 Management page as seen by a user

This is done by checking the session file where it is saved the role that the user has. So for every option of the management page there is a checking of the user's role before if the option should be displayed or not.

5.2.10 addauser.php

This file is aiming for adding a new user into the database. It includes "`testinput.php`" and "`config.php`". The first one is included for preventing a malicious user from running penetrating code, and the second for making a connection to the database. Firstly it checks if there is a POST method that has run and then if all form's input fields from `management_page.php` are filled. If not all fields are filled then there will be an alert message informing the user that he has to fill them all before completing the process of adding the user. Then file connects to the database and make a query about the existence of the username. If the username exists, there cannot be a new user having the same username with an older member. Otherwise the new user is added and a message is created to inform the user about the completion of the process. During the creation of a new user a unique id is given to him.

5.2.11 changepassword.php

This file's aim is to change user's password. Firstly it checks if all input fields are filled and if the new password has been entered correctly both times. Then it checks if the current password given is the one corresponding to the user that has asked for his password to be changed. This is done by making a query to the database using the username that has been saved in the session file as long as the password that has been written in the form's field. If any problem comes up until this time an error message is created and displayed on the screen. Otherwise, there is a new connection to the database that updates the password to the new one. Finally a message is created that informs the user that the process has been finished successfully.

5.2.12 deluser.php

The deletion of a user is done in this file. During the POST method from the managementpage.php the user's id is sent to the deluser.php file. Then a connection to the database is established and a command for deleting the person with the id is sent. Finally a message is created for informing the user about the result of his action.

5.2.13 logout.php

User is redirected to this file when he clicks the log out button. In this file the session is destroyed and redirects the user to the login page (index.php).

5.3 Auto-starting OpenHAB

A problem that emerges in application that runs on a server is what happens when server shuts down because of power outage or for maintenance tasks. Administrator would have to check periodically if everything is running so there won't be great loss in data. Because it is difficult for a person to check all the time, we made openHAB restart every time the server restarts. This is done by creating a new file in /etc/init.d/openhab in which we write bash code for making Ubuntu server run the OpenHAB runtime every time it starts. The code that we used was found in github [83].

5.4 Self signed

Security was a priority for our application. As it contains sensitive data of a research experiment and its audience target is researchers who will not share private data. One form of cryptography for protecting our data is the public-key cryptography. In this technology a public key and a private key are used. The system works by encrypting information using the public key. The information can then only be decrypted using the private key. A common use for public-key cryptography is encrypting application traffic using a Secure Socket Layer (SSL) or Transport Layer Security (TLS) connection. We encrypt traffic using a protocol that does not itself provide encryption. O we use Certificates. A Certificate is a method used to distribute a public key and other information about a server and the organization which is responsible for it. Certificates can be digitally signed by a Certificate authority (CA) or being self-signed. We chose to use self-signed certificate as we do not have domain name which is obligatory for having a certificate from a CA. The process of creating a certificate is as follows:

Step 1

We enable the SSL module in the Apache package. This is done via the command

```
sudo a2enmod ssl
```

and now our server can handle SSL. We restart the server in order to make changes applied.

Step 2

In this step we create the Self-Signed SSL Certificate. Firstly we create a directory where the certificate will be installed

```
sudo mkdir /etc/apache2/ssl
```

and then we create the Certificate and the key

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

openssl: basic command for managing certificates, keys, etc.

req: This specifies a subcommand for X.509 certificate signing request (CSR) management. X.509 is a public key infrastructure standard that SSL adheres to for its key and certificate management.

-x509: It specifies that we want to make a self-signed certificate.

-nodes: This option is about making Apache starts automatically without asking for a passphrase for securing our key file every time.

-days 365: The number of days that the certificate will be valid.

-newkey rsa:2048 : This option will create the certificate request and a new private key at the same time. This is necessary since we didn't create a private key in advance. The option rsa:2048 defines that the key will be 2048 bits long.

-keyout : It defines the output file for the private key.

-out : It defines the output file for the certificate.

After the command above there is a number of questions that have to be answered about the organization's name, its address, its email, etc

Step 3

Here we configure the Apache server to use SSL. This is done by modifying the `default-ssl.conf` file which contains some default SSL configurations. In this file we inform Apache about the server name, domain name and the path for finding the certificate and the key. The editing of the file was done via the **nano** text editor which is an editor using the command line interface.

Step 4

We activate the SSL virtual host by typing

```
sudo a2ensite default-ssl.conf
```

and then restarting our server.

These are all steps followed for enabling the SSL on our web application. During the process we used the client program **PuTTY** in order to connect to our server and give the commands.

Tutorial for the configuration was found in

<https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04>

6. Results and discussions

This thesis can be split into two parts, the configuration of the openHAB runtime and the web site. Both of them work properly, without any known issues up until now. Results coming from openHAB runtime are matching with the laboratory's current solution concluding that algorithms are giving proper results. The web site has been tested both in administrator and plain user mode. During every action database was checked in order to confirm good functionality.

During this thesis there were problems that we had to deal with. The BACnet binding that was used had wrong initialization instructions. For proper use see [Adding new devices](#). Yabe, an explorer for browsing BACnet devices, cannot run alongside openHAB cause openHAB cannot connect to the sensor. The binding that was used for the fuzzy logic was not properly working. By contacting to the creator of the binding we managed to find a solution. For fuzzy rules you can see [Fuzzy rules](#).

7. Conclusion and future prospects

This project was designed keeping in mind that it is going to be expanded. As a result in this chapter we are discussing about some expansion cases that includes up rising software technologies, devices with ability to connect to a network and embedded systems.

7.1 Future system development

In the building where the laboratory is, there are also rooms that belongs to the same sub-network. OpenHAB runtime could also manage the controllers and as a result sensors of these rooms and extract more conclusions about energy efficiency. The project could be scaled up to the whole building making it possible for controllers to communicate between each other. For rooms that do not belong to the same sub network there would be limitation because of the BACnet protocol that it is used. In order to solve this problem we could create a dedicated server for that sub network which will send the sensor data to our application and they will be displayed there. These could be easily done as application's structure is designed to be expandable.

One innovative technology in smart buildings, is the appliance of neural networks. Neural network is a collection of connected simple units which communicate. This system is patterned after the operation of neurons in the human brain and could be used for prediction control and self-adaptiveness. Prediction control could also be implied using older data that are saved. By checking these data we can foresee changes in the environment and being able to decide our actions so to prevent loss of energy. A more integrated procedure would be using apart from older data, the thermal model of the building. A thermal model is a process showing the heat transfer phenomena in and around a building. It takes into consideration heat produced by solar radiation, people, electrical equipment etc. building materials, building's architecture and losses in order to draw conclusions about energy management.

Internet of things (IOT) is the new trend in computer technology. From everyday objects to heart monitoring implants, electric clams in coastal waters, automobiles with built-in sensors and much more, can now connect on the internet and be controlled remotely, give useful information about a person, communicate to each other etc. As a result it, project should also include an IOT in order to extract information on how openHAB deals with such devices, how it reclaims their data, the way it could make them communicate and much more.

In our days microcomputers and embedded systems are having a great share of the market. Microcomputers are small single board computers which are cheaper and have low energy consumption. Implementation of openHAB into these devices would be very interesting because of their great abilities alongside with their energy save operation. A known microcomputer, Raspberry pi [91] has been used in many applications in bibliography and it is also recommended by the openHAB foundation. It could be used in our case as an alternative solution to the current cloud based project. Due to his low consumption it could be used as a server running in the laboratory. It could have a static IP and user could connect to it and use openHAB the same way he uses it now.

8. References

[1] <http://www.eia.gov/outlooks/ieo/buildings.cfm>

[2] https://en.wikipedia.org/wiki/Building_automation#Automation_system

KMC Controls. *"Understanding Building Automation and Control Systems"*. Archived from [the original](#) on 19 May 2013. Retrieved 27 March 2013.

"CEDIA Find: Cool Automation Integrates Smart Air Conditioners with Third-Party Control Systems". CEPro. Retrieved 16 Jun 2015.

Dragoicea, M.; Bucur, L.; Patrascu, M. (2013). *"A Service Oriented Simulation Architecture for Intelligent Building Management"*. *Proceedings of the 4th International Conference on*

Exploring Service Science 1.3. LNBIP 143: 14–28. doi:10.1007/978-3-642-36356-6 2.

<http://poet.lbl.gov/diagworkshop/proceedings/norford.htm>

http://www.iss.uni-stuttgart.de/forschung/veroeffentlichungen/streubel_upec12.pdf

<http://www.hindawi.com/journals/acisc/2012/742461/>

"Lighting control saves money and makes sense" (PDF). Daintree Networks. Retrieved 2009-06-19.

"About VAV". SimplyVAV. Retrieved 5 October 2015.

[US Dept. of Energy, Pacific Northwest National Laboratory, Building Re-Tuning Training Guide: AHU Discharge-Air Temperature Control](#)

[TAYLOR ENGINEERING, Resetting Setpoints Using Trim & Respond Logic](#)

[TRANE, Engineers Newsletter, Energy-Saving Control Strategies For Rooftop VAV Systems, Supply-Air-Temperature Reset. \(Page 2, Column 2, Paragraph 1\) Volume 35–4, ADM-APN022-EN \(October 2006\)](#)

"Building Automation System Clawson Michigan Clawson Manor". Retrieved January 3, 2016.

Patrascu, M.; Dragoicea, M. (2014). "Integrating Services and Agents for Control and Monitoring: Managing Emergencies in Smart Buildings". Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics. Studies in Computational Intelligence Volume 544: 209–224. doi:10.1007/978-3-319-04735-5 14.

Intelligence, Critical (12 April 2014). "European researchers explore the possibility of BACnet botnets". Retrieved 4 September 2016.

Khera, Mandeep (1 September 2016). "Is IoT Security a Ticking Time Bomb?". /securityintelligence.com. Retrieved 4 September 2016.

Dickson, Ben (16 August 2016). "How to prevent your IoT devices from being forced into botnet bondage". techcrunch.com. Retrieved 4 September 2016.

Wendzel, Steffen (1 May 2016). *"How to increase the security of smart buildings?"*. *Communications of the ACM*. **59** (5): 47–49. doi:[10.1145/2828636](https://doi.org/10.1145/2828636). Retrieved 4 September 2016.

Granzer, Wolfgang; Praus, Fritz; Kastner, Wolfgang (1 November 2010). *"Security in Building Automation Systems"*. *IEEE Transactions on Industrial Electronics*. **57** (11): 3622–3630. doi:[10.1109/TIE.2009.2036033](https://doi.org/10.1109/TIE.2009.2036033). Retrieved 4 September 2016.

[3] <http://vems.pl/>

[4] <https://buildingsolutions.honeywell.com/en-US/solutions/hvacbuildingmanagement/Pages/default.aspx>

[5] http://www.ops-ecat.schneider-electric.com/ecatalogue/browse.do?cat_id=BU_BAU_62111_L0&conf=livesite&el_typ=node&nod_id=0000000010&prev_nod_id=0000000011&scp_id=Z000

<http://www.schneider-electric.com/b2b/en/solutions/system/s2/buildings-systems-small-and-medium-building-systems.jsp>

[6] https://download.beckhoff.com/download/document/Catalog/Building_Automation_Specialist_Engineers.pdf

<http://www.beckhoff.com/>

[7] <http://www.openremote.com/>

[8] <https://alerton.com/en-us/pages/default.aspx>

[9] <http://www.openhab.org/>

[10] <http://www.businessdictionary.com/definition/market-fragmentation.html>

[11] <http://www.zigbee.org/what-is-zigbee/>

[12] <http://www.simplymodbus.ca/FAQ.htm>

[13] <http://www.obix.org>

[14] <https://ecobuilding.schneider-electric.com/documents/10807/251162/MPM-UN+Multi-Purpose+Manager+-+Specification+Sheet/a45c304a-c3f9-4151-9f8d-0d5b1a77a958>

[15] <https://ecobuilding.schneider-electric.com/documents/10807/251162/MPM-VA+VAV+Manager+-+Specification+Sheet/9f9c4aee-18eb-4afd-a764-8c5c9e9502d1>

[16]

[17] <http://www.bacnet.org/>

[18] <https://www.ashrae.org/>

- [19] <https://www.lua.org>
- [20] <http://www.campit.gr/>
- [21] <http://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html>
- [22] <http://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html>
- [23] http://www.eenets.com/Files/Download/chapter_5.pdf
- [24] <http://www-rohan.sdsu.edu/doc/matlab/toolbox/fuzzy/fuzzytu3.html>
- [25] <http://researchhubs.com/post/engineering/fuzzy-system/fuzzy-membership-function.html>
- [26] <http://www.eclipse.org/equinox>
- [27] <http://docs.oracle.com/javase/specs/jvms/se7/html/index.html>
- [28] <http://www.sonos.com/en-us/home>
- [29] <https://nest.com/>
- [30] <http://www2.meethue.com/en-us>
- [31] <https://eclipse.org/jetty/>
- [32] <https://github.com/openhab/openhab1-addons/wiki/Feature-Overview>
- [33] <https://github.com/openhab/openhab1-addons/wiki/REST-API>
- [34] <http://www.eclipse.org/smarthome/index.html>
- [35] <http://www.eclipse.org/smarthome/index.html>
- [36] <https://github.com/openhab/openhab1-addons/wiki/Addons>
- [37] <https://github.com/openhab/openhab1-addons/wiki/KNX-Binding>
- [38] [One Wire Binding https://github.com/openhab/openhab1-addons/wiki/Addons](https://github.com/openhab/openhab1-addons/wiki/Addons)
- [39] [Bluetooth Binding https://github.com/openhab/openhab1-addons/wiki/Addons](https://github.com/openhab/openhab1-addons/wiki/Addons)
- [40] [Wake-on-LAN bindinghttps://github.com/openhab/openhab1-addons/wiki/Addons](https://github.com/openhab/openhab1-addons/wiki/Addons)
- [41] [Network Health Bindinghttps://github.com/openhab/openhab1-addons/wiki/Addons](https://github.com/openhab/openhab1-addons/wiki/Addons)
- [42] [Exec Bindinghttps://github.com/openhab/openhab1-addons/wiki/Addons](https://github.com/openhab/openhab1-addons/wiki/Addons)
- [43] [Http Binding https://en.wikipedia.org/wiki/Batch_file](https://en.wikipedia.org/wiki/Batch_file)
- https://en.wikipedia.org/wiki/Shell_script
- [44] [Fritz Box Bindinghttps://github.com/openhab/openhab1-addons/wiki/Explanation-of-Items](https://github.com/openhab/openhab1-addons/wiki/Explanation-of-Items)
- [45] [Ntp Bindinghttps://github.com/openhab/openhab1-addons/wiki/Persistence](https://github.com/openhab/openhab1-addons/wiki/Persistence)
- [46] [Mpd Bindinghttps://github.com/openhab/openhab1-addons/wiki/Scripts](https://github.com/openhab/openhab1-addons/wiki/Scripts)
- [47] [VDR bindinghttps://github.com/openhab/openhab1-addons/wiki/Actions](https://github.com/openhab/openhab1-addons/wiki/Actions)
- [48] <https://github.com/openhab/openhab1-addons/wiki/Asterisk-Binding>
- [49] <https://github.com/openhab/openhab-distro/blob/master/CONTRIBUTING.md>
- [50] https://en.wikipedia.org/wiki/Batch_file
- https://en.wikipedia.org/wiki/Shell_script

- [51] https://eclipse.org/Xtext/documentation/301_grammarlanguage.html
- [52] <https://github.com/openhab/openhab1-addons/wiki/Explanation-of-Items>
- [53] https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm
- [54] <https://github.com/openhab/openhab1-addons/wiki/Persistence>
- [55] http://www.eclipse.org/xtend/documentation/203_xtend_expressions.html
- [56] <https://github.com/openhab/openhab1-addons/wiki/Scripts>
- [57] <http://www.eclipse.org/Xtext/#xbase>
- [58] <https://github.com/openhab/openhab1-addons/wiki/Actions>
- [59] <https://github.com/openhab/openhab1-addons/wiki/Rules>
- [60] <http://searchsoa.techtarget.com/definition/site-map>
- [61] <https://github.com/openhab/openhab1-addons/wiki/Explanation-of-Sitemaps>
- [62] <https://github.com/openhab/openhab1-addons/wiki/Transformations>
- [63] <https://github.com/openhab/openhab1-addons/wiki/Configuring-the-openHAB-runtime>
- [64] <https://github.com/openhab/openhab1-addons/wiki>
- [65] <https://osgi.org/javadoc/r4v42/org/osgi/service/event/EventAdmin.html>,

<http://felix.apache.org/documentation/subprojects/apache-felix-event-admin.html>
- [66] <https://github.com/cdjackson/HABmin>
- [67] https://wiki.eclipse.org/Rich_Client_Platform
- [68] <http://www.openhab.org/downloads.html>
- [69] <http://www.openhab.org/downloads.html>
- [70] <https://sourceforge.net/projects/yetanotherbacnetexplorer>
- [71] <https://github.com/arrizer/BACNet-openHAB-binding>
- [72] <https://www.gnu.org/licenses/quick-guide-gplv3.en.html>
- [73] <https://knx-user-forum.de/forum/supportforen/openhab/38994-%E2%88%9A-openhab-und-bacnet>
- [74] <https://community.openhab.org/t/bacnet-binding-and-how-to-configure-the-cfg-file/8858/4>

- [75] [Fanger, P Ole \(1970\). Thermal Comfort: Analysis and applications in environmental engineering. McGraw-Hill.](#)
- [76] [https://github.com/ansz74/org.openhab.action.jfuzzylogic.](https://github.com/ansz74/org.openhab.action.jfuzzylogic)
- [77] <https://www.mathworks.com/products/matlab.html>
- [78] <http://jfuzzylogic.sourceforge.net/html/manual.html#details>
- [79] <https://community.openhab.org/t/fuzzy-logic-new-add-on/13182>
- [80] <http://www.putty.org/>
- [81] <https://wiki.debian.org/Apt> <http://www.computerhope.com/unix/apt-get.htm>
- [82] <http://www.webdevelopersnotes.com/what-is-web-server>
<http://whatis.techtarget.com/definition/Web-server>
- [83] https://www.w3schools.com/sql/sql_intro.asp
<https://www.thesitewizard.com/faqs/what-is-mysql-database.shtml>
- [84] <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04#step-2-install-mysql>
<https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-14-04>
- [85] <https://filezilla-project.org>
- [86] <https://www.w3.org/html>
- [87] <https://www.w3.org/Style/CSS/#specs>
- [88] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
- [89] <https://getbootstrap.com/>
- [90] <https://github.com/openhab/openhab/wiki/Samples-Tricks>
- [91] <https://www.raspberrypi.org/>

