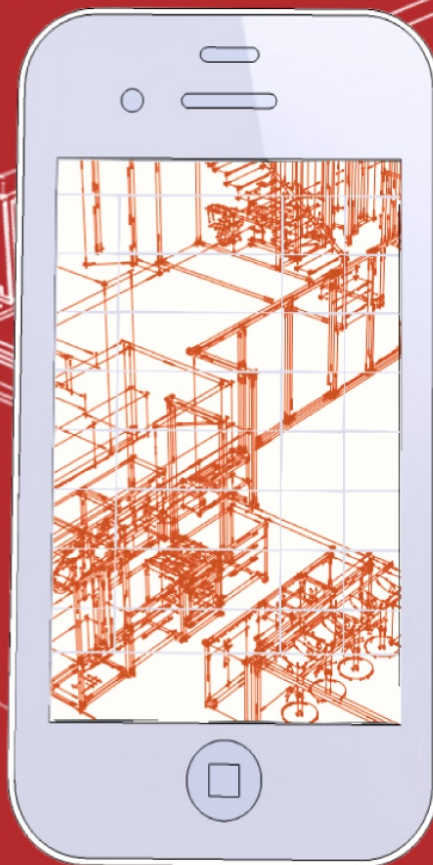


**OTHMAR KYAS**

# **HOW TO SMART HOME**



**KEY CONCEPT PRESS**

# How To Smart Home

With openHAB

A Key Concept Book by Othmar Kyas

*1st Edition, April 2020*

How To Smart Home

Published by Key Concept Press

[www.keyconceptpress.com](http://www.keyconceptpress.com)

ISBN 978-3-944980-19-5

First Edition April 2020

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

Copyright © 2020 by KEY CONCEPT PRESS

## Disclaimer

Every effort has been made to make this book as accurate as possible. However, there may be typographical and or content errors. Therefore, this book should serve only as a general guide and not as the ultimate source of subject information. This book contains information that might be dated and is intended only to educate and entertain. The author and publisher shall have no liability or responsibility to any person or entity regarding any loss or damage incurred, or alleged to have incurred, directly or indirectly, by the information contained in this book. References to websites in the book are provided for informational purposes only and do not constitute endorsement of any products or services provided by these websites. Further the provided links are subject to change, expire, or be redirected without any notice.



## About the Author

Othmar Kyas is an internationally renowned expert in communication technology and strategic marketing. He is author of sixteen books, which have been published internationally in several languages.

Why don't we shut the whole house off for a few days and take a vacation?" "You mean you want to fry my eggs for me?" "Yes." She nodded. "And darn my socks?" "Yes." A frantic, watery-eyed nodding. "And sweep the house?" "Yes, yes - oh, yes!" "But I thought that's why we bought this house, so we wouldn't have to do anything?" „That's just it. I feel like I don't belong here. The house is wife and mother now, and nursemaid.

Ray Bradbury, „The Veldt“, September 1950

Disclaimer .....	3
About the Author .....	4
<b>1. Read Me .....</b>	<b>9</b>
1.1. Who is this Book for? .....	9
1.2. What You Will NOT Find .....	10
1.3. What You WILL Find .....	10
1.4. Safety First! .....	11
1.5. Take no Risks.....	11
1.6. Formatting Rules.....	11
<b>2. The Big Picture .....</b>	<b>12</b>
2.1. Smart Buildings and the Internet of Things (IoT) .....	13
2.2. The Potential for Energy Conservation.....	13
2.3. Safety Management and Assistive Domotics .....	17
2.4. Changing the World (a bit) to the Better .....	17
2.5. Bibliography .....	18
<b>3. Key Concepts .....</b>	<b>19</b>
3.1. Devices under Control.....	20
3.2. Sensors and Actuators.....	21
3.3. Home Automation Network (HAN).....	21
3.4. Controller (Smart Hubs).....	27
3.5. Remote Control Devices .....	30
3.6. Cloud Services.....	30
3.7. Bibliography .....	31
<b>4. Home Automation Network Protocols .....</b>	<b>32</b>
4.1. Network Address Translation (NAT).....	32
4.2. Open Ports and Port Forwarding (Port Sharing).....	32
4.3. UPnP .....	33
4.4. Dynamic DNS .....	34
4.5. HTTP REST.....	34
4.6. HTTP Server Push.....	35
4.7. Bibliography.....	36
<b>5. You Don't Know What You Don't Know - Smarthome Security .....</b>	<b>37</b>
5.1. Attacking the HAN .....	37

5.2.IoT Search Engines - Shodan and friends .....	38
5.3.Bibliography .....	41
<b>6. Home &amp; Building Automation: Markets and Trends ...</b>	<b>42</b>
6.1.Market Size and Growth .....	42
6.2.Smart Devices & Deep Learning Technologies .....	43
6.3.Bibliography.....	45
<b>7. Smart Homes for the Masses: Google, Apple, Samsung, Amazon and more ...</b>	<b>46</b>
7.1.Google's Nest Labs and Google Home .....	46
7.2.One More Thing ... Apple HomeKit.....	47
7.3.Samsung's SmartThings .....	48
7.4.Amazon's Echo.....	48
<b>8. To Cloud or not to Cloud - This is the Question.....</b>	<b>50</b>
<b>9. The Project.....</b>	<b>52</b>
9.1.Overview .....	52
9.2.Equipment and Prerequisites .....	55
<b>10.The Smart Home Control Center openHAB .....</b>	<b>57</b>
10.1.openHAB Overview .....	57
10.2.openHAB Installation: macOS and Windows .....	59
10.3.openHABian Installation: Raspberry Pi.....	60
10.4.First steps with openHAB.....	62
10.5.First steps with openHAB .....	67
10.6.Items.....	73
10.7.Where are You? Presence Detection with openHAB .....	76
10.8.openHAB's Memory and Intelligence: Persistence & Rules.....	77
10.9.OpenHAB Upgrades .....	84
10.10.OpenHAB Backup .....	84
10.11.Migration from openHAB to openHABian .....	85
<b>11. openHAB GUI Design .....</b>	<b>86</b>
11.1.Homebuilder .....	88
<b>12. Integration of Multimedia .....</b>	<b>92</b>
12.1.Setting up Kodi .....	92
12.2.Integrating Kodi with openHAB .....	93
12.3.Controlling HiFi Components: DenonMarantz.....	97

12.4.Creating Command Sequences (Scenes) .....	101
<b>13. A Little AI: openHAB Rules .....</b>	<b>102</b>
13.1.Testing it Right - Best Practice for Script Writing .....	103
13.2.Pretty smart: A weather based alarm.....	104
13.3.A final rule with fixed timing .....	112
<b>14. More iDevices.....</b>	<b>116</b>
14.1.Device Control Using Z-Wave .....	116
14.2.Device Control Using KNX.....	122
14.3.Configuring a KNX Heating Control System .....	125
14.4.Bibliography .....	133
<b>15. Remote Smarthome Control .....</b>	<b>134</b>
<b>16. Troubleshooting and Testing .....</b>	<b>138</b>
16.1.Troubleshooting openHAB .....	139
16.2.Preventive Maintenance.....	140
<b>17. Appendix: Industry Grade Home Infrastructure Control: KNX .....</b>	<b>141</b>
17.1.What is KNX?.....	141
17.2.How does KNX Work? .....	141
17.3.The KNX Software Infrastructure: ETS .....	142
17.4.ETS on a Mac .....	143
17.5.ETS5 Installation .....	143
17.6.Importing Vendor Catalogs .....	145
17.7.ETS5 Infrastructure Configuration.....	145
17.8.ETS5: Adding the Building Infrastructure.....	146
17.9.ETS5: Configuring the KNX Elements .....	147
17.10.ETS5: Connecting Infrastructure to Controls .....	149
17.11.Notes on Configuring KNX Devices .....	152
<b>18. Bibliography.....</b>	<b>154</b>

# 1. Read Me

This book discusses state of the art smart home, building automation and Internet of Things strategies and explains in detail the underlying technologies. In the second part it walks the reader through the implementation of a typical real world home automation project using the popular and powerful open source platform openHAB.

## 1.1. Who is this Book for?

You will be introduced to technology basics, planning and design principles, security and privacy considerations as well as implementation details and testing philosophies. Expecting no specific know-how upfront, the book is suited for both - the professional consultant as well as the technology loving hobbyist.

After explaining the big picture and the key concepts of state of the art home and building automation, the book will walk you through the implementation of a concrete building automation and control project in a step-by-step manner. At the end of each project phase you should have a real, working solution on your desk, which can be further customized and expanded as desired. No programming skills are required as prerequisite. Scripts are explained line by line, configuration settings step by step. Of course, if you have never written a short automation script or configured a DSL router, at some point your learning curve will be steeper than the one of others. However, everything you learn will be based on open standard technologies, which you will be able to utilize in any other IT related project.

Technologies and platforms which are used in the project are:

- Wi-Fi / WLAN
- Telnet, HTTP, TCP/IP
- Z-Wave, a smart home communication standard
- KNX, a smart home communication standard
- openHAB (open Home Automation Bus), an open source building automation and Internet of Things software platform
- macOS / Linux / openHabian/ Windows 10 / Java

Parts of the project integrate consumer electronics devices, such as audio equipment from Denon or Marantz. However, project and instructions are designed, so that that they can easily be adapted to other manufacturers. Be aware, however, that equipment, which is more than a few years old, might lack the required interfaces for smart home integration at the level which is being covered in this book, such as built in WLAN, Bluetooth, web server components, or "Wake-on-LAN" functionality.

In addition to the professional grade, fully customizable openHAB based building control solution, the book also covers off the shelf smart home platforms such as Apple's HomeKit, Google's NEST, Google Home, Samsung's SmartThings or Amazon's Echo. While they are not in the centre of the book, their technologies, design, strengths and weaknesses are discussed. Also integration options with the described solution are outlined and explained.

In general, the technologies, design and planning approaches, test philosophies and security considerations discussed in this book do apply to any smart home and building automation solution.

## 1.2. What You Will NOT Find

This book is not a cookbook for simple plug and play type home automation solutions, which various vendors are offering based on closed and proprietary solutions with limited functionality. It looks at the much broader market of smart homes, building automation and Internet of Things (IoT) and does a much deeper dive into the technology basics than the average smart home customer might be interested in. Plug and play type solutions and how to integrate them with professional level building automation are covered however, but it is the smaller part of a much broader and deeper discussion of the topic.

## 1.3. What You WILL Find

The objective of this book is to explain and demonstrate how to build a fully customizable smart home and building automation solution, which is capable of integrating devices and platforms from different vendors, connecting them through meaningful and useful rules. The outcome is a professional level, real world smart home solution, which improves quality of life while saving energy and time. For the implementation of the sample project in this book I have selected one of the most comprehensive and advanced open source building control platforms available today, openHAB. Step by step you will be walked through the installation and configuration of a typical smart home project, which you can adapt to your local requirements as you go. For the initial phases of the project - display of outside temperature, display of astronomical data for sun and moon, control of your music library, control of consumer electronic equipment and presence detection based on smartphones accessing the local WiFi, you will not even need any equipment but a computer, a smartphone and Internet access. For the later part of the project, where full infrastructure control comes into play, you will need either add-ons to legacy lighting, room heating, power outlet components or smart home infrastructure with control capabilities built in. You will learn which technology and product options are available, to help you making informed decisions at this point.

Leading up to the project in the first chapters you will learn the key concepts of building automation, the main components and their functionality, involved network protocols as well as security issues. Markets and trends for building automation, different architectural approaches (e.g. cloud based versus in-house based) as well as the plug and play solutions from large vendors such as Samsung, Apple, Amazon or Google are being discussed. At the same time as stretching from the

big picture of smart homes to the implementation of a concrete working solution, the book tries to be as brief as possible, in order to make best use of your time.

## 1.4. Safety First!

For the proposed project security and availability aspects are discussed in detail and according recommendations are being made. In chapter five we will take a closer look at what is behind the frequent headline news about smart home door locks being hacked, or smart home cloud accounts being compromised. In addition according measures against potential attacks are being outlined. In the sample project described, safety and security plays a key role as well. All step by step instructions take privacy and operational security into account and are designed accordingly. In addition, fundamental technology design and operations principles for a secure and high availability building control infrastructure are being discussed, providing value beyond the project covered in this book.

## 1.5. Take no Risks

A last word of caution before we get started. Be careful when following the step-by-step instructions. Almost no two PC systems, consumer electronic devices, or other electronic gear are alike. Even when they appear to be, they actually might differ in hardware due to different production runs, in firmware, in software or in configuration. If something goes wrong, you might need to reinstall the operating system on your computer platform and you could lose all your data. So set up a dedicated user for testing or experimentation or even better use a spare computer system, unless you are absolutely sure what you are doing. As an example, using a single-board computer platform such as Raspberry Pi - which among other platforms is also covered in this book - costs less than 100 € and saves space and energy. So budget constraints are no more in the way for a professional and save approach today.

## 1.6. Formatting Rules

For better readability, the following formatting rules are used throughout the book:

- *Monospace*  
Computer output, code, commands, user input
- **LARGE CAPS**  
Communication Protocols (DHCP, IP, KNX, etc.)
- *Italic – medium blue*  
sequence of GUI commands separated by en dash (–)
- Medium blue underlined  
Web addresses (URLs)

For the projects in this book I have created the user account `smarthome` under macOS and Windows 10. The prompts in terminal window screenshots as well as in terminal print-outs read accordingly.



# 9. The Project

## 9.1. Overview

Complex functionality in information technology can be explained best using an incremental approach, starting from simple “hello-world” type of functionality to sophisticated features in sequential steps, each of which can be tested and demonstrated individually. In software engineering terms, this approach is referred to as development sprints. Sprints are relatively small coding modules, which need to be designed in such a way, that they can be demonstrated independent of other development elements, once their implementation is finished. Such sprint demonstrations are formal project milestones, which are conducted in front of the entire engineering team. The advantage of the sprint approach is the continuous validation of functionality. Problems are recognized early and remain manageable. The permanent monitoring of increasing functionality avoids surprises and keeps the fun factor high. In following this philosophy, most design phases of our project are independent from each other and work stand alone. However, some of the initial components build upon each other, so it does make sense to follow the sequence up to chapter thirteen.

In the next two chapters we will start with installing and configuring the open source smart building and Internet of Things (IoT) platform openHAB, which will serve as the home controller for the project. The openHAB framework will also allow us to build a customized smartphone and tablet control app in little time with no programming skills required. Later, openHAB will also be used to run the automation rules, which we will define during the course of the project. As a first example we will be retrieving weather condition and temperature data displaying it with our smartphone app (Figure 9.1).

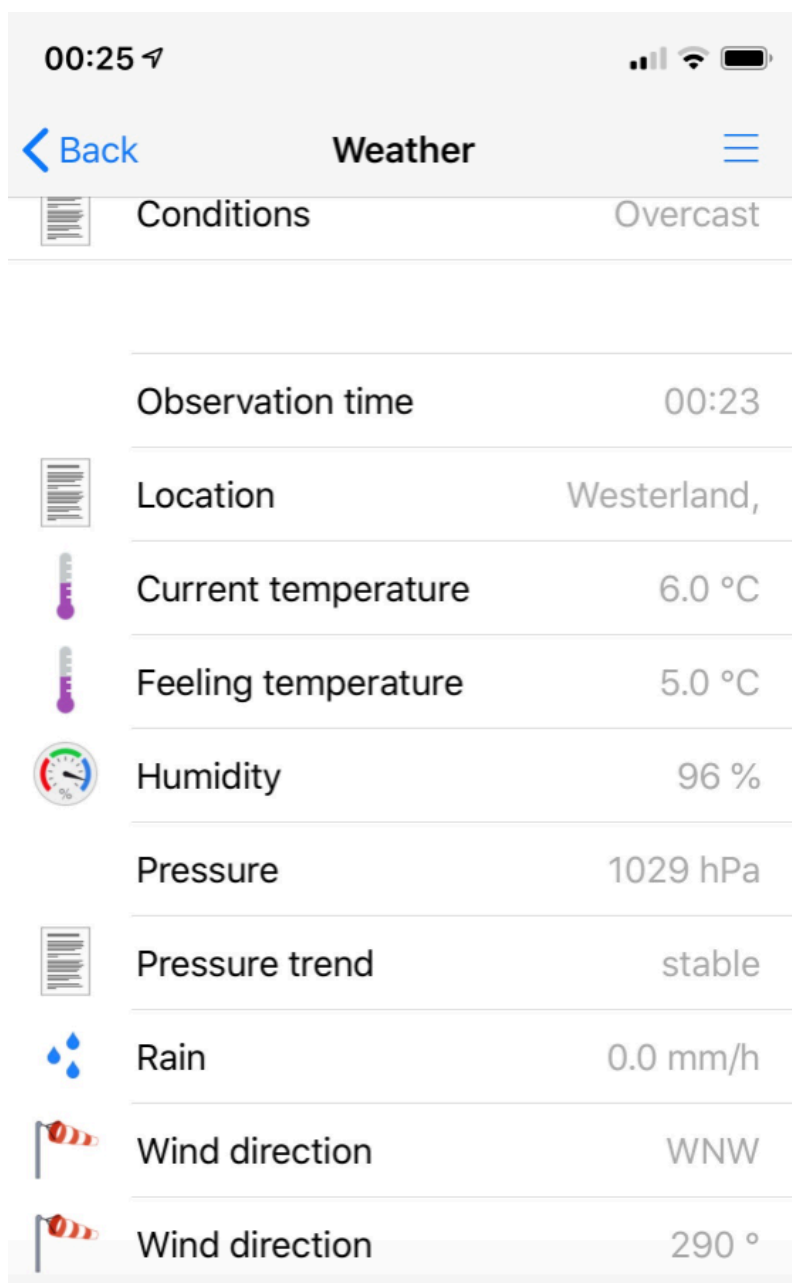


Figure 9.1 Display of weather information using the openHAB smartphone app

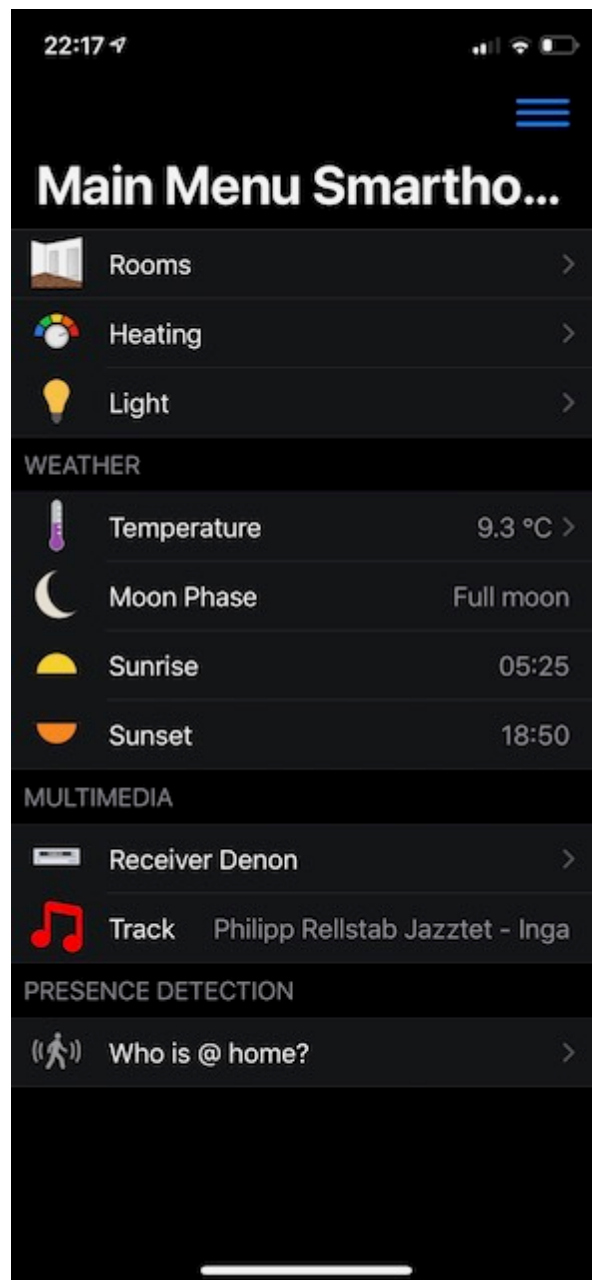
In chapter twelve we integrate multimedia control functionality to our smart home project. We will set up openHAB to integrate and control the popular open source Kodi media center application. In chapter thirteen we introduce the rule based automation capabilities of openHAB. We will use the components, we have built so far, to put together the intelligent wake up scenario: “Wake me up early in case it rains”. The idea is to start a morning wake up scenario 45 minutes earlier than normal, in case of nightly rain or snowfall, to avoid the potential traffic jam. For that, we will use our Internet weather sensor to poll the weather conditions during the night, and, once a wake up condition is met, our scenario will start playing music. Up to that phase all you need for following and implementing the project is a Mac, a PC or a Raspberry Pi, a WiFi network and an Internet capable smartphone. In chapter fourteen we add wireless light and power outlet control to our project, using the standards Z-Wave and KNX.

In the last part of the book we extend our project with functionality, which enables it to reliably function in a real environment and in daily operation. We add the capabilities to access all control functions from remote via the Internet and to automatically restore operation after a restart of the controller.

With that done, our smart home control system will be capable of:

- smartphone / tablet based display of weather, temperature and astronomical data
- smartphone / tablet based control of lights, heating, power-outlets, consumer electronics devices
- smartphone / tablet based control for scenarios such as Good Morning, Welcome, Good Night, Leaving Home
- rule based scenario execution, triggered by time, date, weather condition or temperature
- rules for scenarios such as
  - empty home (empty during the day)
  - vacant home (vacant for two or more days)
  - warmup (activation of heating without night savings to bring up temperature in a vacant home)
  - presence simulation (light activity to simulate presence)
  - deactivation of all rules
- integration with other smart home platforms
- smart home remote control from outside of the house
- Automated restoration of operation after power outages or planned downtime

Using the functionality of our project as a start, a vast variety of variations and add-ons can easily be implemented.



*Figure 9.2 Main menu of the smartphone app based smarthome control screen for our project*

## 9.2. Equipment and Prerequisites

In general you will find that in order to implement smart home controls with functions beyond switching power outlets and lights, you need relatively new equipment. This is true for your WiFi (WLAN) router, for the appliances and consumer electronic devices you want to control as well for the mobile clients (smartphones and tablets) you plan to use. Fortunately, prices for all of the above have gone down over the past years. Therefore, in many cases, you might rather want to upgrade the equipment you have to the latest generation, than giving up functionality or spending a lot of effort trying to integrate legacy equipment. Of course, there are always also good reasons not to upgrade and to keep existing equipment. Everyone will have to make that decision on an individual base.

In order to be able to follow the project in this book, you will need the following, obviously depending on which functionality you plan to implement:

- a home network with Internet access and a WiFi/DSL router
- an iOS or Android powered smartphone or tablet
- a RaspberryPI, macOS or a Windows based computer system
- Z-Wave components (power-outlets, lighting, etc.) in case you plan to use the Z-Wave protocol
- KNX components (power-outlets, lighting, etc.) in case you plan to use the KNX protocol
- consumer electronic devices with LAN / WiFi capability built in

Alternative to Z-Wave or KNX, the usage of other building control standards for the project described in the book is also possible, although not described in detail. The smart building and Internet of Things platform openHAB, which we use throughout this book, supports literally all major building control standards.

In addition to the above equipment, some familiarity with computer and network technology is recommended. You do not have to be able to actually write code. However, if you have never heard of IP, Telnet or HTTP, and if you have never edited a batch file (.bat) or a shell script (.sh), you will probably have to go through a steeper learning curve than others. On the other hand, with the thousands of good Internet tutorials just a mouse click away, there is nothing you cannot learn within a few hours.

;-)

# 10. The Smart Home Control Center openHAB

We start our project with the installation and configuration of openHAB (open Home Automation Bus), one of the leading open source software platforms for building automation and device control. The platform has been used for building automation and Internet of Things (IoT) projects in residential and commercial applications around the world and is supported by a large and active user community. Setting up and configuring the software is a bit more complex than clicking a few buttons, however, you do not need programming skills. As a result the platform allows for building a fully custom, professional building automation system, including smartphone apps and cloud service for state of the art, user friendly control.

## 10.1. openHAB Overview

The openHAB platform consists of three major components:

- The openHAB controller, an always-on (24/7) Linux (Ubuntu, Raspbian, ...), Windows or macOS server application, which connects control devices (smartphones, tablets, panels) to smart objects and other building automation systems. Smart objects can be building infrastructure (light switches, power outlets etc.), consumer electronic devices, home appliances or web services. The openHAB controller can also run automation sequences called rules.
- The second component consists of the openHAB clients. They interact with the smart objects under control via the openHAB controller. Clients can either be a web browser, which connects to the openHAB server or iOS and Android apps for operation on mobile phones or tablets.
- The third component is the free myopenHAB cloud service (operated by the openHAB Foundation), which allows to access your openHAB server from away securely without having the need of a complex VPN setup.

Through more than 250 add-on software interface modules, which in openHAB are called bindings, the platform can interact with home automation devices from all major vendors:

*Air Quality, Discovery, Binding Configuration, Thing Configuration, Channels, Full Example, AirVisual Node, AKM868, Alarm Decoder, AllPlay, Amazon Dash Button, Amazon Echo Control, Anel NET-PwrCtrl, Asterisk, Astro, Atlona, Autelis Pool Control, AVM FRITZ!, Belkin Wemo, BenQ Projector, BigAssFan, Bluetooth, Bosch Indego, Bose SoundTouch, Bticino, CalDAV Command, CalDAV Personal, Cardio2e, Chamberlain MyQ, Chromecast, Cm11a (X10 controller), ComfoAir, ConfigAdmin, CoolMasterNet, CUPS, D-Link Smart Home, Daikin, Davis, DD-WRT, Denon, Denon / Marantz, digitalSTROM, DIYonXBee, DMX, DSC Alarm, DSMR, eBUS, Ecobee, EcoTouch, ekey, Energenie, EnOcean, Enphase Energy, Epson Projector, Exec, Expire, Fatek PLC, Feed, Feican, FHT, Folding@home, Freebox, FreeSWITCH, Fritz AHA, Fritz!Box, Fritzbox (using TR064 protocol), Fronius, Frontier Silicon Radio, FS Internet Radio, FS20, FTP Upload, Garadget, Gardena, Global Cache IR, GlobalCache, GPIO, GPSTracker, GROHE ONDUS, HAI/Leviton Omni and Lumina, HDanywhere, Heatmiser, Helios, HMS, Homematic, Horizon mediabox, HTTP, Hunter Douglas PowerView, Hyperion, iCloud, IEC 62056-21 Meter, IHC / ELKO, innogy SmartHome, Insteon Hub, Insteon PLM, Intertechno, IPP, IPX800, IRtrans, ISY, Jeelink, jointSPACE, Keba, KM200, KNX, Kodi, Kostal Inverter, Koubachi, LaMetric, LCN, Leap*

*Motion, LG TV, LG TV control using serial protocol, LG webOS, LIFX, LightwaveRF, LIRC, Log Reader, Logitech Harmony Hub, Logitech Squeezebox, Loxone, Lutron, MailControl, MAX!, MAX!Cube, MAX!CUL, MCP23017, MCP23017, MCP3424, Meteostick, Miele@home, Milight/Easybulb/Limitless, Minecraft, MiOS Bridge, Mochad X10, Modbus, MPD, MQTT, Mystrom Eco Power, Neato, NEEO, NeoHub, Nest, Netatmo, Network, Network Health, Network UPS Tools, Nibe Heatpump, NibeUplink, Niko Home Control, Nikobus, Novelan/Luxtronic Heat Pump, NTP, Oceanic, OneBusAway, OneWire, OneWire GPIO, Onkyo, Open Energy Monitor, Open UV, OpenPaths, OpenSprinkler, Orvibo, OwnTracks (formerly MQTTitude), OWServer, Panasonic TV, panStamp, Pentair Pool, PHC, Philips Hue, Picnet Sapp, Piface, pilight, Pioneer AVR, PLCBus, PLCLogo, Plex, Plugwise, PowerDog Local API, Powermax, Primare, Pulseaudio, Raspberry Pi RC Switch, RegoHeatPump, RFXCOM, RME, Robonect, Rotel Amplifier, Russound, RWE SmartHome, Sager Weathercaster, Sallegra, Samsung Air Conditioner, Samsung TV, Satel Integra Alarm System, Seneye, senseBox, Serial, Serial Button, Silvercrest Wifi Plug, SleepIQ, SMA Energy Meter, Smarthomatic, SNMP, Solar-Log, SolarEdge, Somfy Tahoma, Somfy URTSI II, Sonance, Sonos, Souliss, Stiebel Eltron LWZ, Swegon Ventilation, Synop Analyzer, Systeminfo, TACmi, tado, Tankerkönig, TCP & UDP, Tellstick, Tesla, TinkerForge, TiVo, Toon, TP-Link Smart Home, Tråderfri, UCProjects.eu Relay Board, UPB, ValloxMV, Velbus, Velleman k8055 USB IO Board, Velux, Video Disk Recorder (VDR), Vitotronic, WAGO, Wake-on-LAN, Weather, WeatherUnderground, WiFi LED, Windcentrale, Withings, WR3223 ventilation controller, Xiaomi Mi Smart Home, xPL, YahooWeather, Yamaha Receiver, Yeelight, Z-Way, Zibase, ZigBee, Zoneminder, ZWave*

The openHAB controller software is structured in five major building blocks, which together connect, monitor and control every smart object of a smart building: These are: Things, Channels, Bindings, Items and Sitemaps.

- Things are the smart objects under control such as home appliances, consumer electronics, lighting, or window blinds. Anything which is connected to and controlled by the home automation system. While often hardware based, Things can also be software entities such as web-services, which provides air quality or weather information.
- Bindings are the software interfaces, which connect the openHAB controller software to the various smart objects. Some bindings, when installed, are capable of automatically identifying connected components they can interact with. Other bindings do not.
- Channels represent the functions of a device. For a light, which can change color, which can be dimmed and which can be switched on and off, you will have three channels to control each functionality. You can compare channels to TCP/IP ports . Each port reaches a different service on a server. In openHAB each channel reaches a different functionality of a Thing.
- Items are the representation of the Things on the openHAB control panel, and the element with which the user interacts. There are single items and group items, the later of which integrate multiple single items.
- Sitemaps build the graphical control interface by arranging and grouping items.

### 10.1.1. openHAB Configuration

All configuration data in openHAB is stored in a JSON database. While an increasing amount of configuration can be done using the graphical PaperUI, advanced configuration tasks such as defining rules and the design of custom control interfaces require the use of text based configuration files. (These files are parsed and end up in the same JSON database, however via editing the text files you have full control over their content). I recommend to get some basic functionality such as switching a light on and off working using the PaperUI, but then, before

implementing a larger project, use the text based configuration files, which we will describe in the following chapters. With Homebuilder openHAB provides a tool, which allows to quickly generate the text based configuration files .items and .sitemap for an entire infrastructure with floors, rooms, lights, power plugs per room, heating, using a well thought out, structured nomenclature. All which is left to do then is to configure the Things in the .things file and add the channel information in the .item configuration file, which then connects the Items to the Things. What sounds cryptic if you hear it for the first time, is actually a very structured approach you will appreciate, once you have started working with it. Since the openHAB documentation provides configuration templates for almost any device and function, typically you will just need to edit proven templates adapting them to your local requirements.

Figure 10.1 shows the openHAB architecture and how Bindings, Things, Channels, Items and Sitemaps interact with each other.

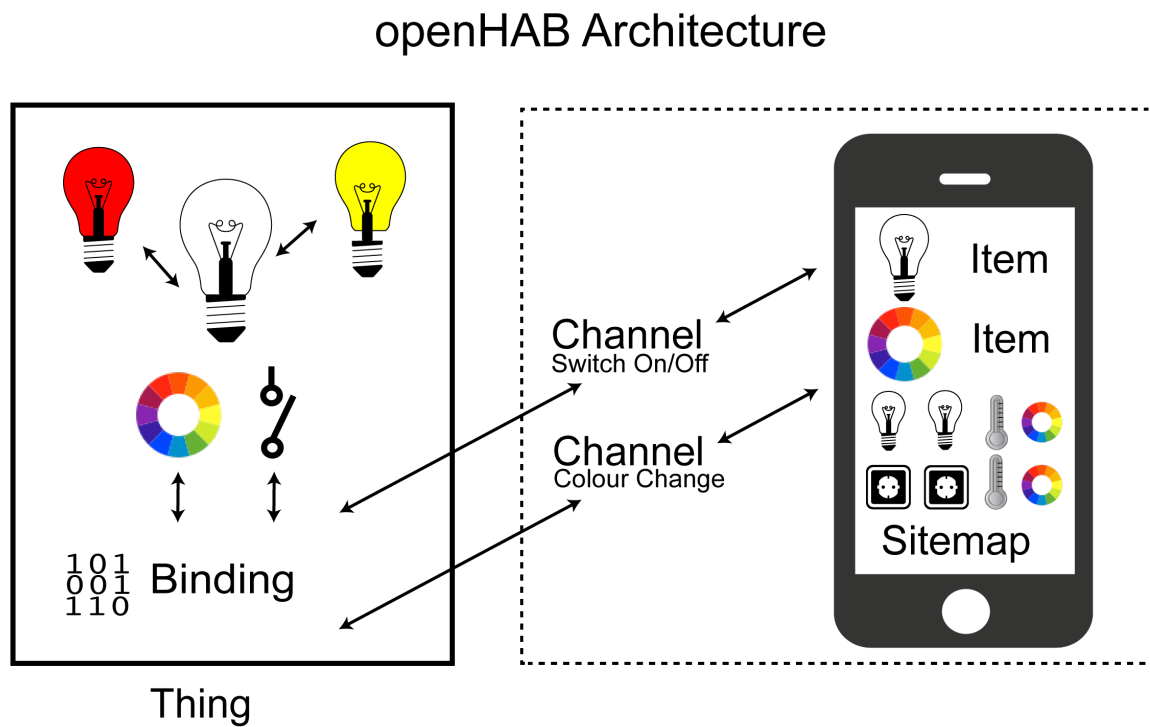


Figure 10.1 The openHAB architecture: Bindings, Things, Channels, Items and Sitemaps

## 10.2. openHAB Installation: macOS and Windows

As outlined above, openHAB is written in Java and requires a Java platform for operation. While it also runs under Java 9 and 10, the officially recommended Java platform version is Java 8 revision 101 or higher. So as the first step download Java 8 from [java.com](http://java.com) or [azul.com](http://azul.com) and install it on your system. Then verify your installation on a Mac by opening a terminal window and typing

```
java -version
```



Alternatively (on a Mac) go to System Preferences, where you will see a Java icon, the Java Control Panel. If you select *General* and *About*, the Java version number of your installation is being displayed.

Under Windows click the *Start* button and scroll through the applications and programs listed until you see the Java folder. Click on the Java folder, then *About Java* to see the Java version number.

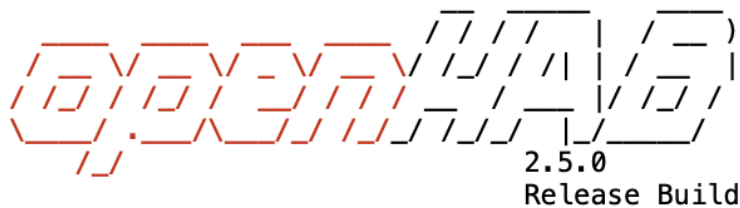
After installing Java 8 go to [openhab.org](http://openhab.org), download openHAB 2.x and install it. Then start the openHAB controller for the first time. On a Mac open the terminal window, navigate to the openHAB folder and type `./start.sh`. Under Windows execute `C:\openHAB2\start.bat` at the command prompt. Wait until the openHAB controller has finished its startup sequence and you can see the *openhab* prompt in the terminal window (Figure 10.2). While you will not interact with openHAB using the terminal window on a daily basis, you should be aware of the following commands:

<tab> for a list of available commands

<cmd> help for help on a specific command.

<ctrl-d> or type `system:shutdown` or `logout` to shutdown openHAB

Launching the openHAB runtime...



Hit '<tab>' for a list of available commands

and '[cmd] --help' for help on a specific command.

Hit '<ctrl-d>' or type '`system:shutdown`' or '`logout`' to shutdown openHAB.

*openhab>*

*Figure 10.2 Start of the openHAB controller*

### 10.3. openHABian Installation: Raspberry Pi

The popular Raspberry Pi is probably the best all-round single-board computer platform available today. Its 2019 edition 4 with a BCM2711B0 quad-core 1.5 GHz CPU, dual display support via micro HDMI, Gigabit Ethernet and USB 3.0 has even reached a performance level, which allows to use it as a basic desktop computer. For the purpose of a home controller, in which case it is used as an embedded system with headless setup (no keyboard and monitor connected), also earlier Raspberry Pi versions will provide the required computing and interface performance. While many operating systems such as Ubuntu, Windows 10 IoT Core, or FreeBSD support Raspberry Pi, its official operating system is Raspbian, a Debian-based Linux distribution. Operating system and applications are stored on a MicroSD card, which is inserted into the according slot on the board.

Switching between two complete different setups therefore is as easy as swapping MicroSD cards. MicroSD cards are used as boot medium due to their very low power consumption of only around 60 - 300 mW. If needed, you can also use external hard drives or SSDs, as long as the shared power drain across all USB ports does not exceed 1.2 A (Pi 4 and Pi 3B+). Given the 5 V USB bus this corresponds to a maximum power drain across all USB ports of 6 W. Since the power consumption of a typical SSD lays in the range of 2 W to 4 Ws, it is advised to use an active USB hub for the connection of larger storage devices. However, with large (up to 1 TB) and fast (designed for 4k streaming) MicroSD cards available, in most cases a MicroSD card will be sufficient for all your needs.

While you can also manually install Raspbian and openHAB on a Raspberry, it is more convenient to download [openHABian](#), an image which contains Raspbian Lite, openHAB and all necessary tools in their newest version. Raspbian Lite is a scaled down, minimal version of the fully blown Raspbian operating system without a graphical user interface. However since openHAB comes with a web server, which provides the graphical openHAB GUI, as far as openHAB is concerned, the openHAB GUI on a Raspberry looks exactly like the one on a Mac or a Windows computer. And since in most cases you will run the openHAB Raspberry unit as dedicated home server, there is no real need to add the fully blown desktop software onto it.

After downloading the openHABian image you use [Etcher](#), which is available for macOS, Windows and Linux, to flash it onto your MicroSD card. Now insert the card in your Raspberry Pi, connect power and Ethernet and wait between 30 and 45 minutes for openHABian to finish its setup. Entering the Raspberries domain name `openh` in your browser allows you to you monitor the installation progress, which largely depends on the performance of your Raspberry Pi model and the throughput of your Internet connection (Figure 10.3).

## openHABian Installation Status

the log will be refreshed automatically every 10 seconds

```
2019-07-10_01:21:43_BST [openhABian] Starting the openHABian initial setup.
2019-07-10_01:21:44_BST [openhABian] Storing configuration... OK
2019-07-10_01:21:45_BST [openhABian] Starting webserver with installation log... OK
2020-01-10_23:05:38_GMT [openhABian] Changing default username and password... OK
2020-01-10_23:06:19_GMT [openhABian] Setting up Ethernet connection... OK
2020-01-10_23:06:19_GMT [openhABian] Ensuring network connectivity... OK
2020-01-10_23:06:19_GMT [openhABian] Waiting for dpkg/apt to get ready... OK
2020-01-10_23:07:22_GMT [openhABian] Updating repositories and upgrading installed packages...
```

*Figure 10.3 Initial Startup of openHABian*

Once the update and configuration process is complete, you connect to the openHAB GUI on your Raspberry by entering

`http://openh:8080`

in your browser. To restart openHAB and for other administrative tasks you will however also need a terminal connection to openHABian. For that you open a terminal window and enter the command

```
ssh openhabian@your.IP.address.here
```

The commands for the most common administrative tasks are listed with

```
openhab-cli -help
```

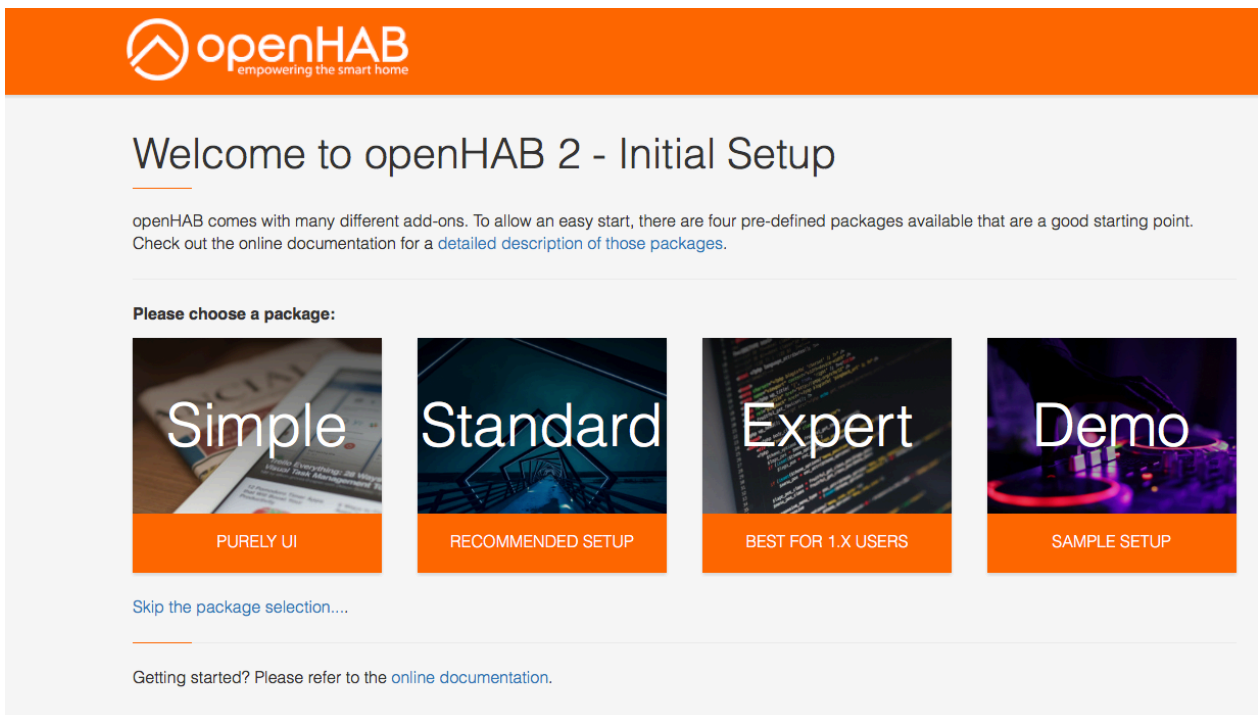
To start the built-in configuration tool enter

```
sudo openhabian-config
```

The actual configuration files are located in the directory `/etc/openhab2/`, however typically you will mount the directory from your working computer (see section 10.4.1) rather than manipulating files via a terminal connection.

## 10.4. First steps with openHAB

Now open a web browser and point it to `http://localhost:8080`. Then select DEMO. This will install a demo smart home configuration, which will allow you to understand the functionality of openHAB. This option also generates demo configuration text files, which can be used as templates later(Figure 10.4).



*Figure 10.4 Initial Startup of openHAB*

You will now see the openHAB start screen with the four menu items Homebuilder, Basic UI, Paper UI and HAB Panel. Now open Basic UI and Paper UI (right-click and open both links in new tabs, so you can switch between the two and the start screen. The Basic UI displays the demo

sitemap, which is the control GUI of the demo configuration (Figure 10.5). Play around with the GUI to get a first feeling on how a openHAB project looks like.

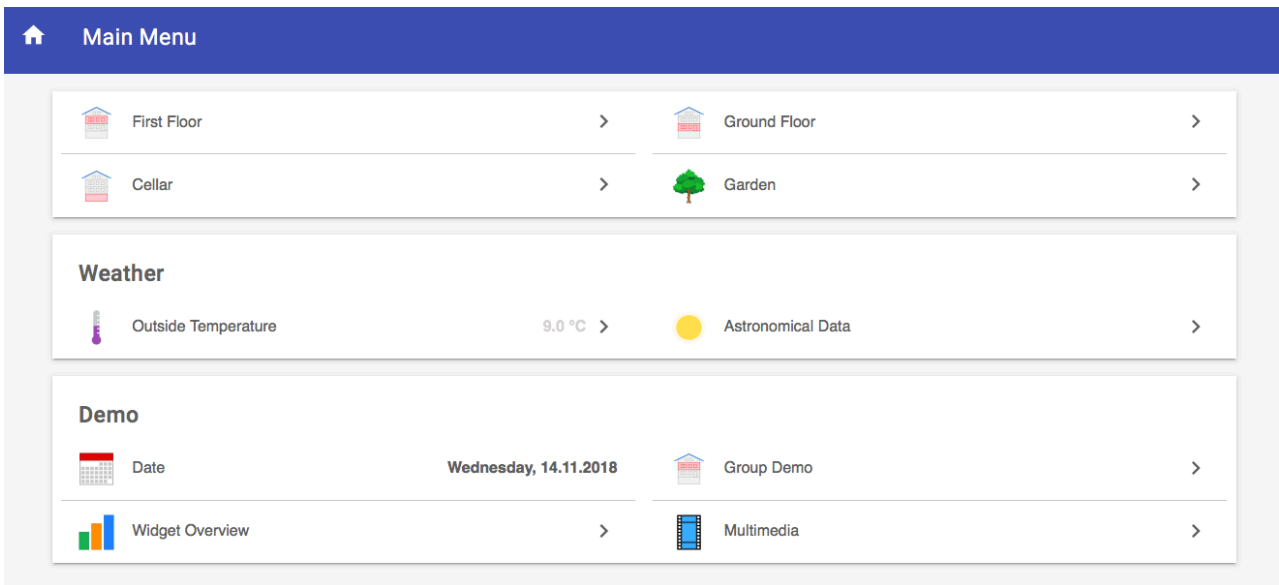


Figure 10.5 The openHAB demo sitemap

Now go to Paper UI, select *Configuration* and take a look at *bindings – things and items*. Click on a Thing to see its Channels, then click on one of the Channels to see the associated Item. This gives you a first feeling how Things, Channels and Items are linked together (10.6).

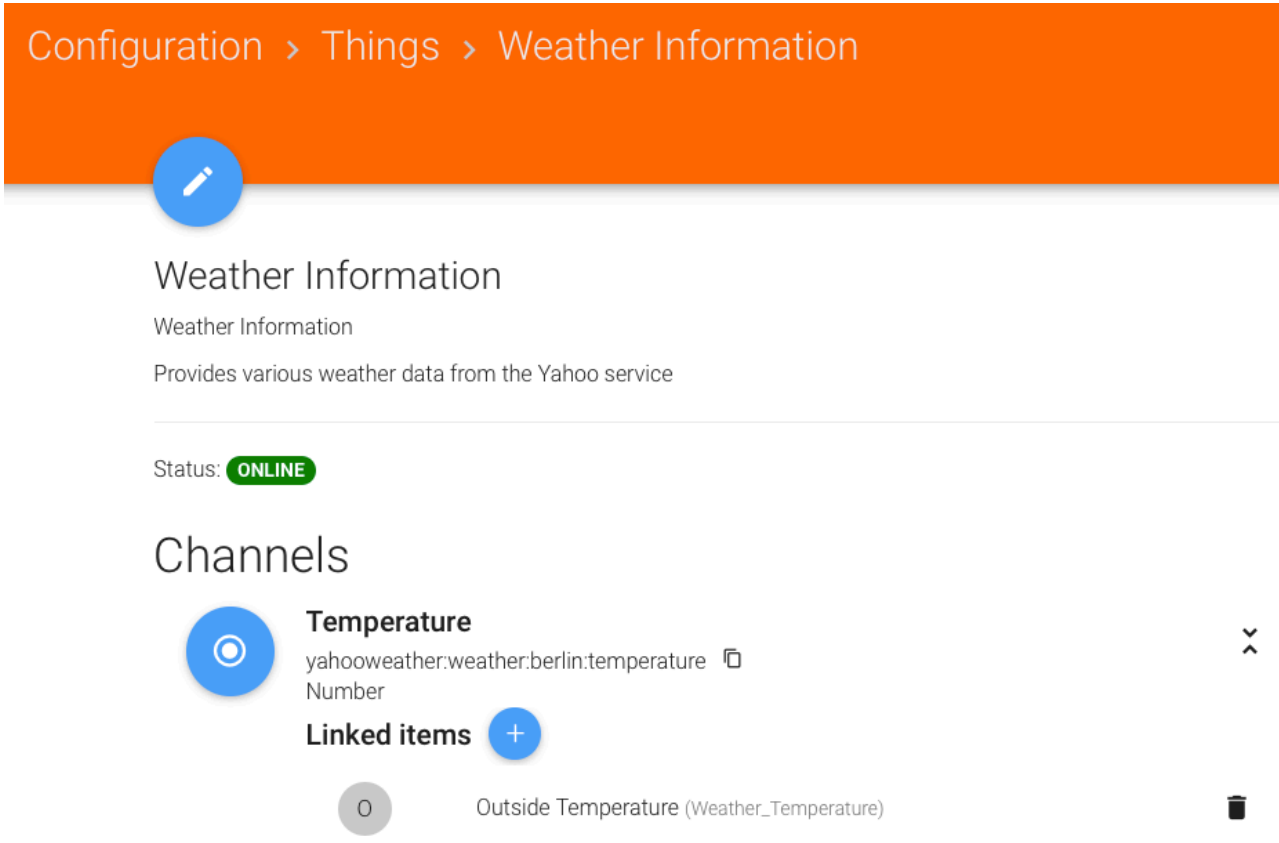


Figure 10.6 Things, Channels and Items in Paper UI

In Paper UI then select the *Control* menu. You will see a display of all Items, similar as in the Basic UI, however not nicely grouped, and without chart information. Control is meant to be a simple, basic GUI for Items, ignoring sitemap configurations or rules.

Before we get started with configuring our own openHAB project, we will do one more thing: we will install the free Visual Studio Code (VS Code) editor, which provides dedicated openHAB extensions. This will make it very easy for us to edit the text based openHAB configuration files. Go to <https://code.visualstudio.com/download>, download the version for your operating system and install it. Now open VS Code and select *preferences – extensions*. Then enter `openhab`. The `openhab` extension will be listed and you can select and install it. Then configure the openHAB settings for Visual Studio Code by selecting the VSC settings icon at the bottom left of the VSC GUI and then *Settings – Extensions – openHABConfiguration* (Figure 10.7.1). In the Host section select *Edit in settings.json* and enter

```
"openhab.host" "your.IP.address.here"
```

or in case you work with openHABian on a Raspberry you can leave the default value

```
"openhab.host": "openhabianpi"
```

In case you run openHAB on the same computer you are running VSC you simply enter `localhost` (Figure 10.7.2). Figure 10.7.3 shows the VSC workspace configured for openHAB with the integrated terminal window on the lower right side of the GUI.

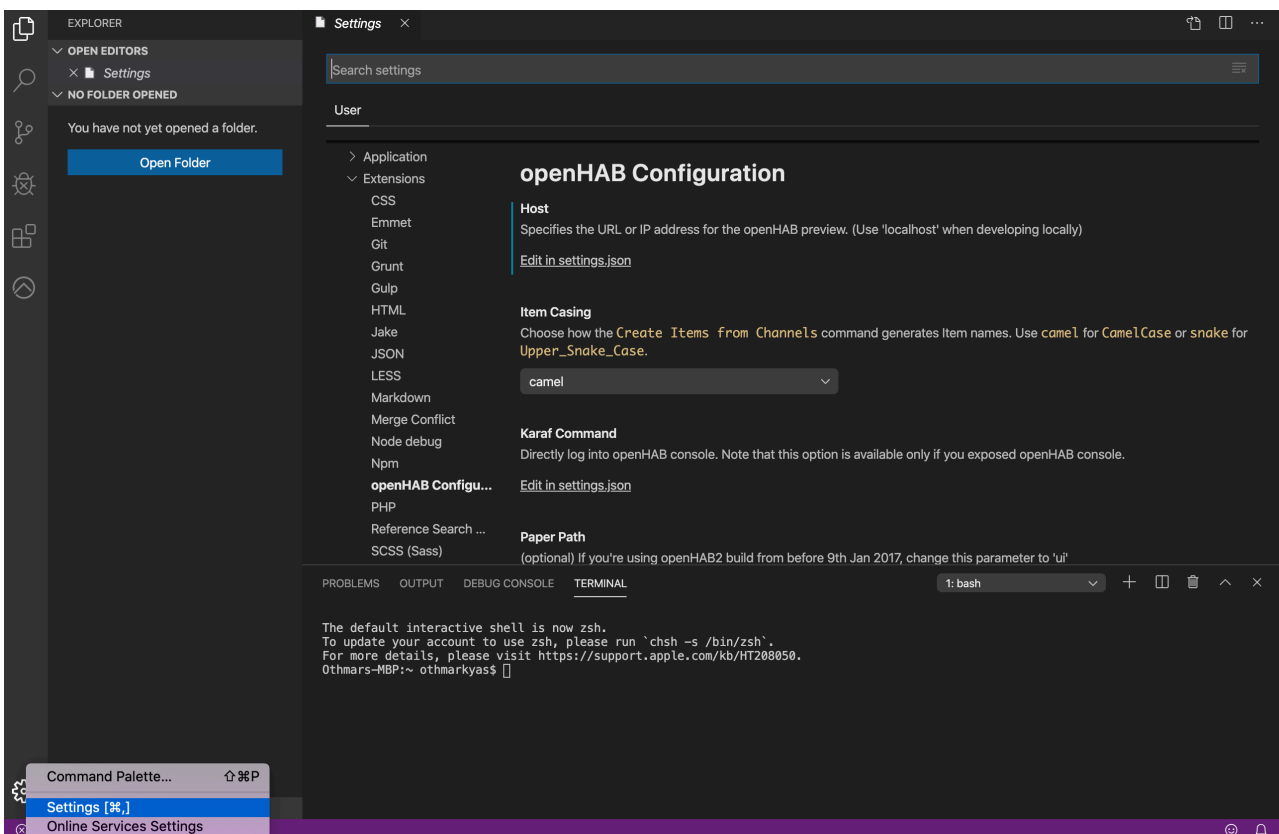


Figure 10.7.1 Configuring the openHAB extensions in Visual Studio Code

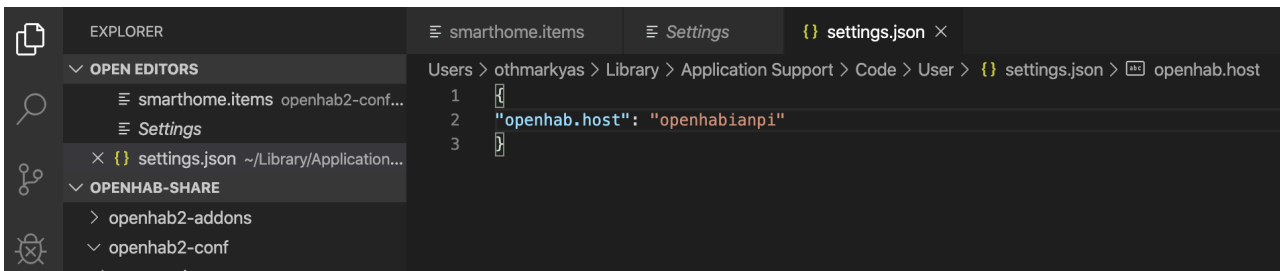


Figure 10.7.2 Setting up the openHAB host address in Visual Studio Code

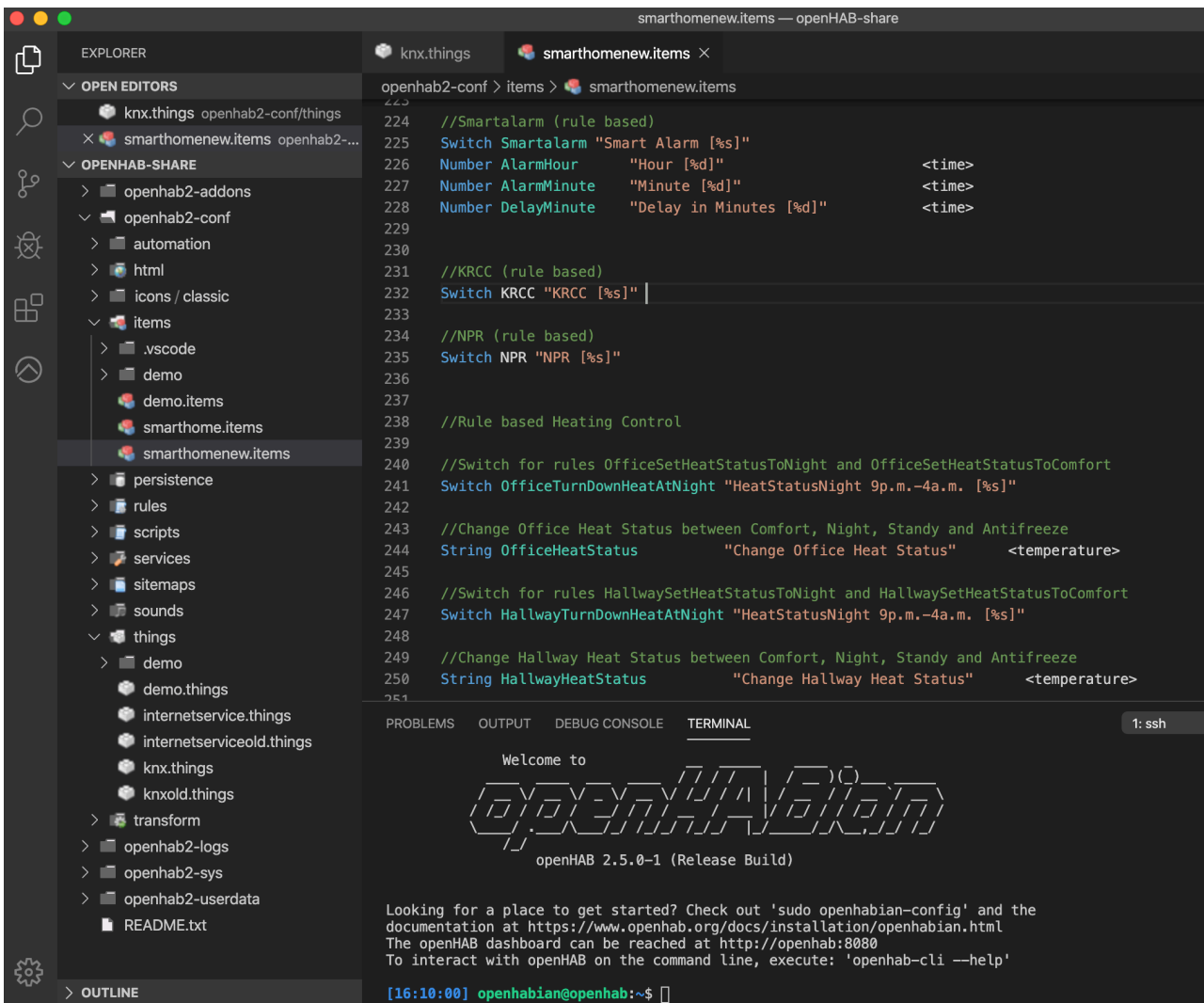


Figure 10.7.3 The Visual Studio Code workspace for openHAB with terminal window at the bottom

### 10.4.1. Setting permissions for openHABian configuration files

One important note on permission management for openHABian on Raspberry Pi . At installation openHABian is installed with the two users openhabian and openhab:

- openhabian is the administrative user with access to the operating system. It has the default password openhabian, which can be changed using the openHAB configuration tool (sudo openhabian-config).

- openhab on the other hand is a user with no password and limited permissions. When openHAB is running, it communicates with Linux as the user openhab.

Per default, as you can see below, the permissions for openHABian configuration files are set as follows:

```
-rw-r--r-- 1 openhab openhab 242 Mar  1 20:16 demo.things
```

The owner of `demo.things` is `openhab` and has read and write access. The group `openhab` (`openhabian` is part of the user group `openhab` as you can see using the command `id openhab`) and also everybody else has read access only.

Thus, when you access openHABian from remote as the user `openhabian`, you need to change the rights for all configuration files (`xxx.things`, `xxx.sitemap`, etc.) in order to be able to edit these files as user `openhabian`. Otherwise you will receive the message `Insufficient permission rights` when you try to save a configuration file, which you have opened and changed using VS Code. With the following command you can add write access to the group `openhab`:

```
sudo chmod -v 664 demo.things
```

The resulting permission configuration is now

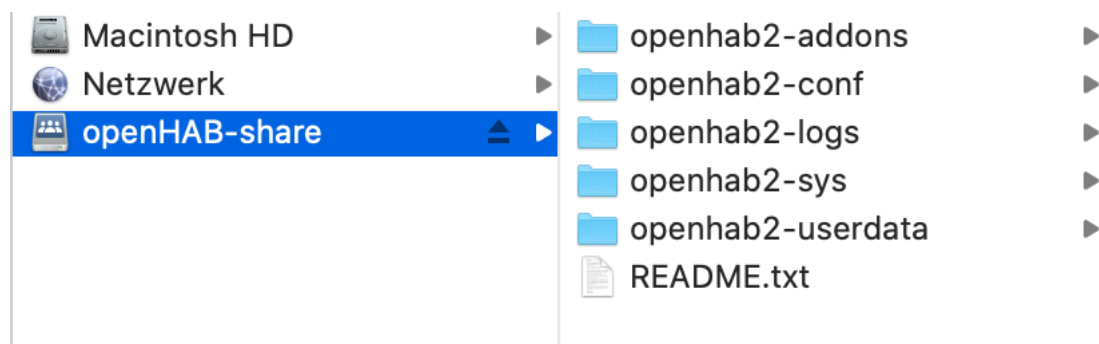
```
-rw-rw-r-- 1 openhab openhab 242 Mar  1 20:16 demo.things
```

and you can edit the file from remote as user `openhabian`.

#### 10.4.2. Remote access to openHAB configuration files

Typically you will have openHAB installed on a dedicated, remote computer system, which is only running openHAB. For this reason you will need to connect to the file system (mount the file system) from your working computer. If you are using a Raspberry as your openHAB platform, the openHABian image has already set up a network share, which directly can be mounted from your local computer. On a Mac simply open Finder (Figure 10.8) and select

```
<Go><Connect to Server> here.your.IP.address  
user: openhabian  
password: openhabian
```



*Figure 10.8 Mounting the openHABian share folder*



The new device *openHAB-share* now provides access to the five openHAB folders openhab2-addons, openhab2-conf, openhab2-logs openhab2-sys and openhab2-user data. In openhab2-conf you will find the configuration files, which you can now edit using any editor such as Visual Studio Code. When you change configuration files or add bindings openHAB will typically automatically refresh the according configuration settings accordingly within a few seconds, which you can follow by looking at the log file openhab.log. In rare cases, if the refresh does not work, you can restart openHAB from with the following command:

```
sudo systemctl restart openhab2.service
```

### 10.4.3. Monitoring openHAB behavior: events.log and openhab.log

To monitor the behavior of openHAB and the impact of system changes you should always look at the two default log files openhab.log and events.log when making changes. You can do this either by selecting the openHAB LogViewer in the browser view (Figure 10.9), or from a terminal window by entering the command

```
tail -f /var/log/openhab2/openhab.log -f /var/log/openhab2/events.log
```

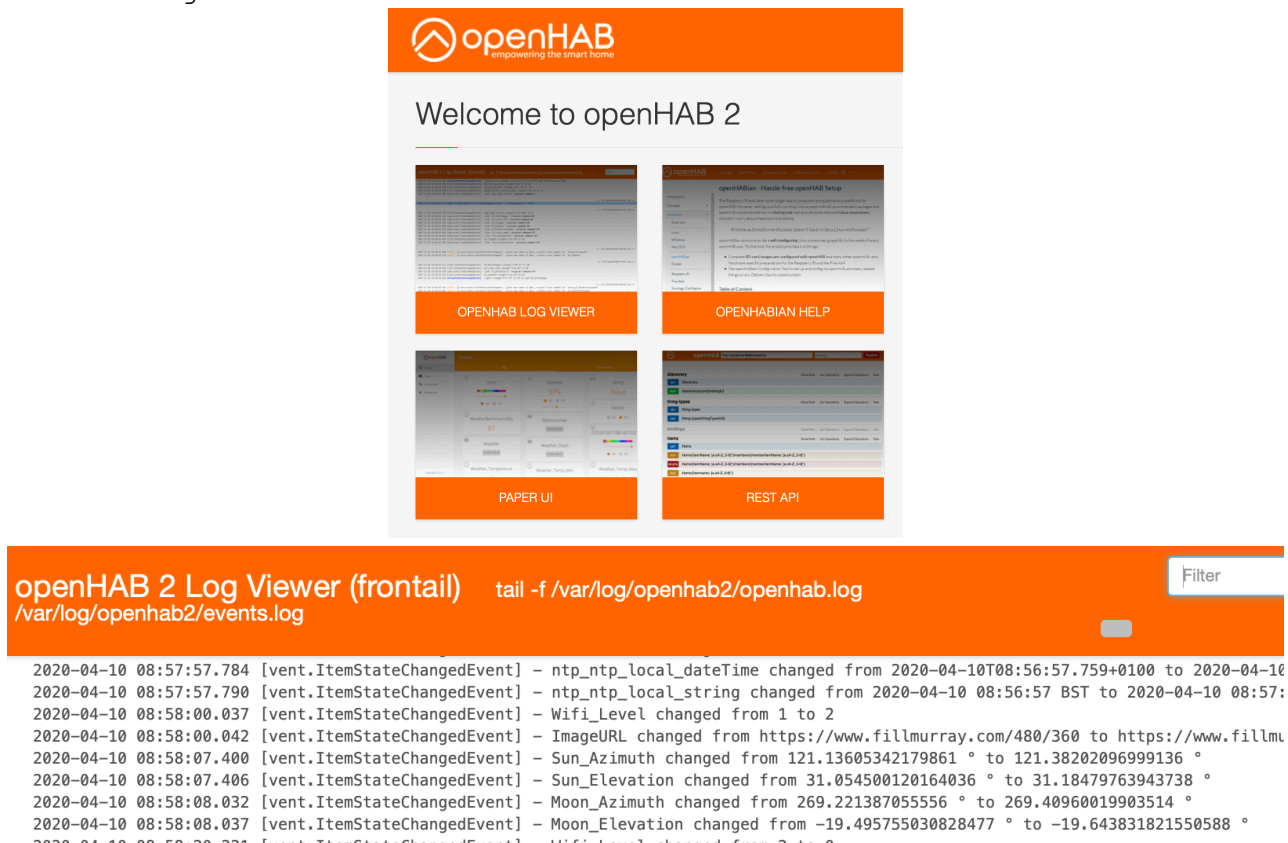


Figure 10.9 The openHAB Log Viewer in Browser View

## 10.5. First steps with openHAB

As mentioned above, we will configure openHAB using configuration text files. If you look into the openHAB directory, you will find the folder conf with the subdirectories html, icons, items, persistence, rules, scripts, services, sitemaps, sounds, things, and transform. To remove the demo configuration create a demo subdirectory in each of the folders /items, /rules, /scripts, /



sitemaps, /rules, /persistence and /things, and move the demo configuration files (demo.things, demo.items, etc.) in the according demo folder. This disables the demo configuration, but keeps the files for later reference (Figure 10.10).

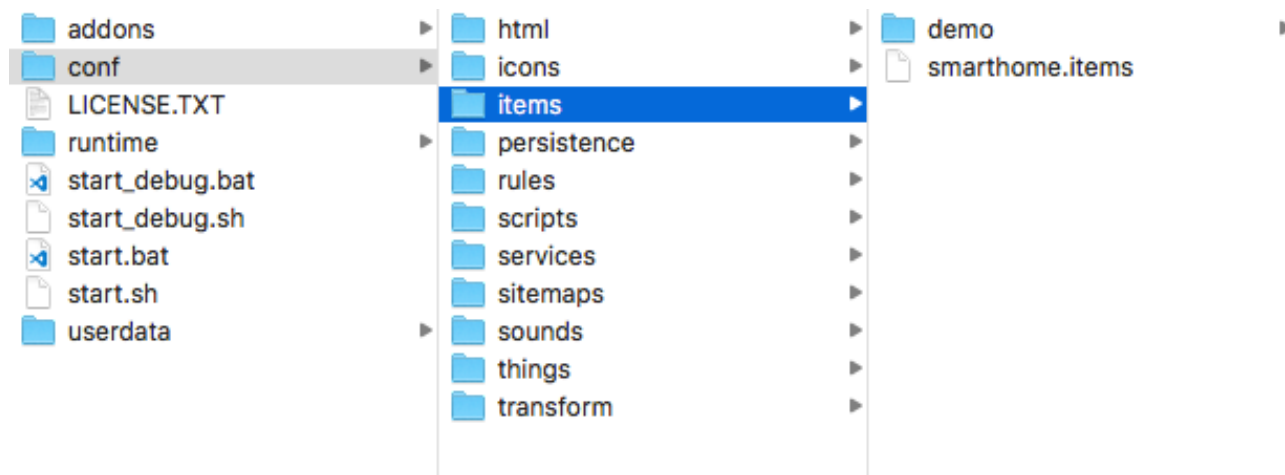


Figure 10.10 The openHAB configuration folder

Initially we will just deal with the two configuration files `.items` and `.things`. The configuration information for bindings and things are contained in the `.thing` configuration file, the information for channels and items in the `.item` configuration file. (This is why there is no `.bindings` and no `.channels` file). In the course of this chapter we will in addition deal with the configuration files `.persistence`, `.rules` and `.sitemap`, but for the moment we concentrate at getting first functionality working.

We create two empty text files (or rename copies of the demo files) called `smarthome.items` and `smarthome.things` in the folders `/items` and `/things`. (If you work on a Raspbian Pi make sure to set the right permissions for the files as described above in the openHABian section). In general you can have multiple or single `.items` and `.things` files, depending on the complexity of a project. Most of the time you will just need to modify configuration templates, which is why configuring openHAB using text files is much easier, than you think by now. Very quickly you will realize, that this approach is much faster, especially when you need to enter more than a few components. You will then just copy text lines and modify a few entries, which is much more efficient then sequentially clicking through multiple GUIs again and again.

As the first step we will configure a few bindings, which we will use for our project. Since bindings are add on software interface modules, you need to reboot openHAB after specifying the bindings you need. The more than 250 bindings for openHAB come with the standard installation, so you do not need to worry about downloading bindings. You just need to specify the bindings openHAB shall load during startup by editing the configuration file `/services/addons.cfg`. With the VS Code file explorer go to [conf – services](#) and open `addons.cfg`. Uncomment the binding entry and add the string

```
astro,hue,ipp,knx,zwave,ntp,openweathermap,kodi,denonmarantz
```

to the line where the bindings are configured. (Figure 10.11):

```
#legacy = true
# A comma-separated list of bindings to install (e.g. "binding =
sonos, knx, zwave")
binding =
astro, hue, ipp, knx, zwave, ntp, astro, openweathermap, kodi, denonmarantz
```

*Figure 10.11 Configuring bindings in /services/addons.cfg*

Then save the changes in VS Code and shutdown the openHAB server by entering `logout` in the terminal window and then restarting it again. On a Raspbian Pi to restart enter

```
sudo systemctl restart openhab2.service
```

Now open the PaperUI and go to [Configuration](#) and [Bindings](#). You should see a display of the bindings you added in the `addons.cfg` file (Figure 10.12).

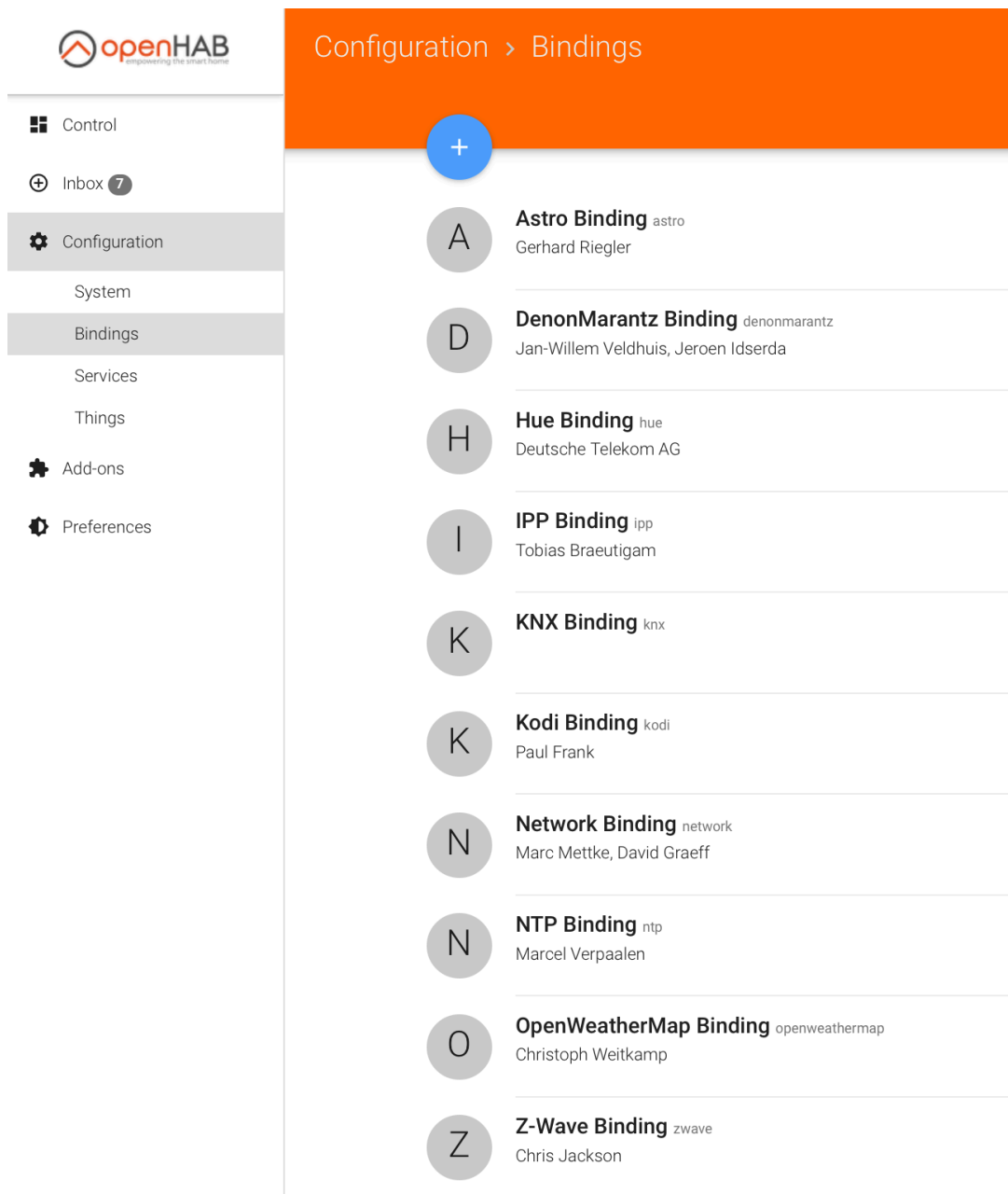


Figure 10.12 Configuring bindings in `/services/addons.cfg`

As the first component for our project we will add the astronomical data for sun and moon from the Astro binding. Simply add the `.things` and `.items` templates from the Astro binding documentation (<https://www.openhab.org/addons/bindings/astro/>) to the configuration files `smarthome.things` and `smarthome.items`. Retrieve the geolocation coordinates for your location using google maps (enter your address, right click on the location marker and select *Whats here?*). Enter the data in the configuration file, latitude first, then longitude, the altitude value is optional.

`smarthome.things:`

```
astro:sun:home [ geolocation="52.5200066,13.4049540", interval=60 ]
astro:moon:home [ geolocation="52.5200066,13.4049540", interval=60 ]
```

`smarthome.items:`

```

Number Sun_Elevation          "Sun Elevation"          <sun>
{ channel = "astro:sun:home:position#elevation" }

Number Sun_Azimuth           "Sun Azimuth"          <sun>
{ channel = "astro:sun:home:position#azimuth" }

DateTime Sunrise_Time        "Sunrise [%1$tH:%1$tM]"
<sunrise> { channel = "astro:sun:home:rise#start" }

DateTime Sunset_Time         "Sunset [%1$tH:%1$tM]"  <sunset>
{ channel = "astro:sun:home:set#start" }

Number Moon_Elevation         "Moon Elevation"       <moon>
{ channel = "astro:moon:home:position#elevation" }

Number Moon_Azimuth          "Moon Azimuth"         <moon>
{ channel = "astro:moon:home:position#azimuth" }

String Moon_Phase            "Moon Phase"           <moon>
{ channel = "astro:moon:home:phase#name" }

```

After saving the changes, the display of the Paper UI should now look like the following (Figure 10.13):

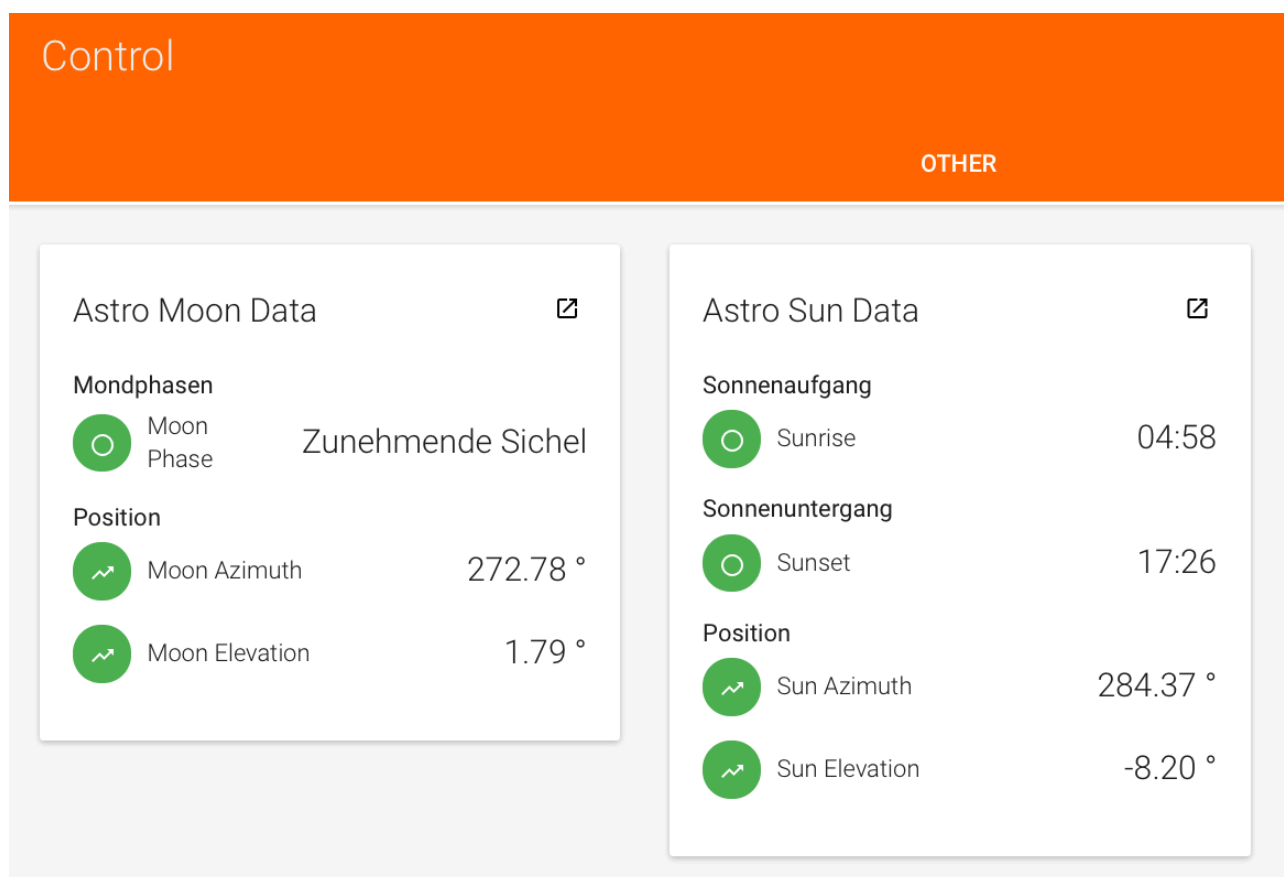


Figure 10.13 Display of Astro items in the Paper UI

Next we will add the display of outside temperature and weather. For this we will use the OpenWeatherMap binding, which we have already installed in our previous step. As a weather

service provider we will use [openweathermap.org](https://openweathermap.org). In order to use their free service we go to their website, register and retrieve an API key. In addition we need the geographic coordinates of the weather station, for which we want to receive weather data. For this we simply go to

<https://openweathermap.org/find?q=>

and enter the desired city name to receive the according coordinates. With these data we can insert the openweathermap bridge code in `smarthome.things` definition file:

```
Bridge openweathermap:weather-api:api "OpenWeatherMap
Account" [apikey="AAA", refreshInterval=30, language="de"] {

    Thing weather-and-forecast local "Local Weather And
Forecast" [location="latitude,longitude", forecastHours=0,
forecastDays=7]

    Thing weather-and-forecast miami "Weather And Forecast In
Miami" [location="25.782403,-80.264563", forecastHours=24,
forecastDays=0]

}
```

Next we add the following item configurations to our `smarthome.items` file, which creates the items `berlinCurrentTemperature` and `berlinStationName`:

```
Number:Temperature berlinCurrentTemperature "Current temperature [%1f
%unit%]" <temperature> { channel="openweathermap:weather-and-
forecast:api:berlin:current#temperature" }
String berlinStationName "Name [%s]" { channel="openweathermap:weather-
and-forecast:api:berlin:station#name" }
```

And finally we add the display configuration for our `smarthome.sitemap` configuration file, which displays Temperature, Humidity along with the according openHAB icons.

```
Frame label="Weather"
{
Text item=wykStationName label="Station"
Text item=wykCurrentTemperature label="Temperature"
}
```

Now, a few seconds after saving our changes we should be able to see the below OpenWeatherMap entries in the Paper UI view (Figures 10.15, 10.16).

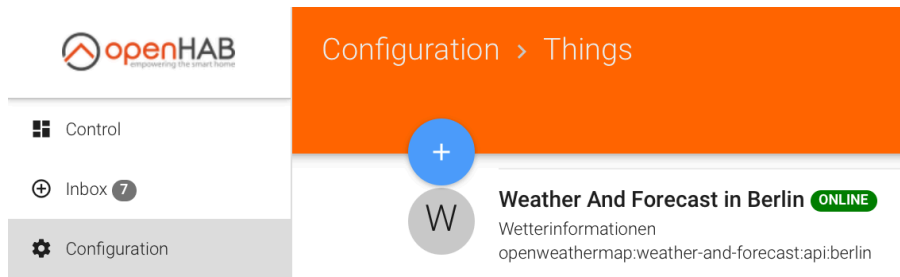


Figure 10.14 OpenWeatherMap Thing in openHAB PaperUI

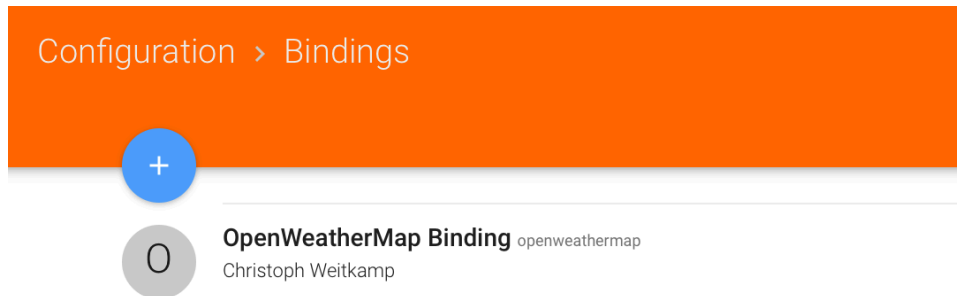


Figure 10.15 OpenWeatherMap Binding in openHAB PaperUI

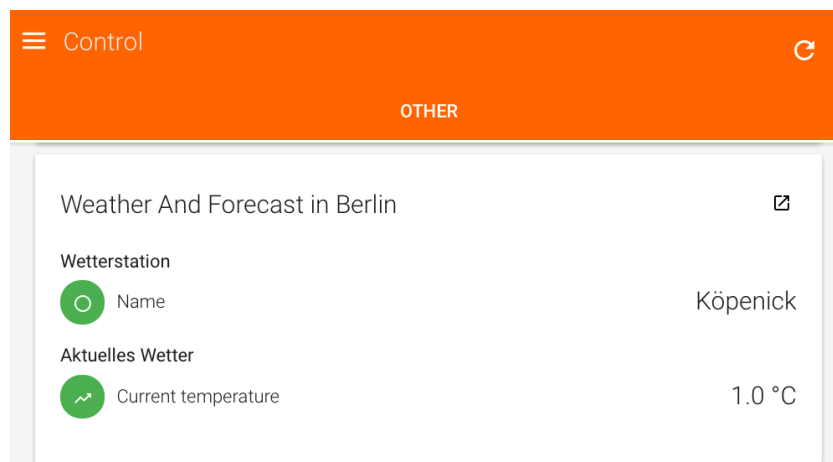


Figure 10.16 OpenWeatherMap Control GUI display in the Paper UI

## 10.6. Items

Before we continue with our project a few words on the syntax of items. There are thirteen types of items, each representing a specific type of information they can store, display and act upon (Table 10.1). With items of the type Group you can build hierarchical items structures (See below). Items of the type Number are used for dimensions such as temperature or speed (Table 10.2). This allows for identifying matching channels and for providing default measurement units. The placeholder %unit% is used to configure the format of the measurement unit. The number value itself can be formatted using the java formatter class syntax. The nomenclature is to use square brackets, a

leading % and the according Java formatting syntax. As an example, using the formatting character f (floating point) the expression

```
"Outside Temperature [%.2f °C]"
```

displays the item label Outside Temperature with the temperature value in °C with two decimal places. In most cases there are openHAB templates with meaningful formatting available, so for most standard situations you can simply go with the formatting proposed in the template.

In general items are defined using the following syntax:

```
itemtype itemname "labeltext [stateformat]" <iconname> (group1,  
group2, ...) ["tag1", "tag2", ...] {channel definition}
```

The first field itemtype specifies one of the thirteen item types (Table 10.2), the field itemname contains the name for the item, which needs to be unique across the entire project. Since building automation projects typically contain large numbers of items, the usage of a systematic, hierarchical naming convention is essential. openHAB recommends a naming convention using a combination of underscores and camel case such as: Livingroom\_CeilingLight\_Color. Frequent names such as ground floor, first floor, bathroom or living room can be abbreviated to GR, FF, LR or BR. As we will see in the next chapter, using the openHAB Homebuilder tool you will be able to automatically generate the configuration text files for .sitemap and .items for you entire infrastructure using this proposed naming convention.

The field labeltext contains the textual description, which the item will display on the sitemap GUI. Along with the textual description this field also allows to specify the format of the measurement value (see above) as well as the state of the connected device in square brackets.

The field iconname defines the icon, which shall be used for displaying the item on the sitemap GUI. openHAB comes with more than 100 predefined icons. Simply look up the name of the item you would like to use at <https://www.openhab.org/docs/configuration/iconsets/classic/#icons> and insert it in the icon name field. If you want to use your own icon, just copy the according .png files into /openHAB-share/openhab2-conf/icons/classic.

The group field contains one or multiple item groups an item is part of. Similar to the hierarchical item naming convention, items themselves can and should be organized in groups. Item groups themselves can also be part of other item groups. The below example shows the definition of the group item Home, which is being displayed using the icon *house* and the label text Smarthome. The group item Bathroom is being displayed via the text Bathroom and the icon bath, while being part of the item group Home:

```
Group      Home                "Smarthome"      <house>  
Group      Bathroom           "Bathroom"       <bath>  
(Home)
```

The Tag field allows to further characterize an item beyond its Type. The final field in curly brackets is the important channel field. It contains the channel definition, which links the item to a Thing. Below for reference again the item definition used to display the current temperature from before, which we now better understand. The item is of type Number with item name Temperature\_OWM, the item label is Temperature and the temperature unit is displayed with one decimal place. In curly brackets we have the channel definition.

```
Number Temperature_OWM "Temperature [%].1f
°C" {weather="locationId=home, type=temperature, property=current"}
```

Besides item Type and Name all fields of an item are optional. Therefore items can be defined and displayed in a sitemap without a link to a Thing. This also means that you can focus on defining the layout of a project using .sitemap and .item configuration files and add the configuration for Things and the channels definition in the .items file later.

Item Type	Description	Command Types
Color	Color information (RGB)	OnOff, IncreaseDecrease, Percent, HSB
Contact	Item storing status of e.g. door/window contacts	OpenClose
DateTime	Stores date and time	-
Dimmer	Item carrying a percentage value for dimmers	OnOff, IncreaseDecrease, Percent
Group	Item to nest other Items / collect them in Groups	-
Image	Holds the binary data of an image	-
Location	Stores GPS coordinates	Point
Number	Stores values in number format, takes an optional dimension suffix	Decimal
Number:<dimension>	like Number, additional dimension information for unit support	Quantity
Player	Allows to control players (e.g. audio players)	PlayPause, NextPrevious, RewindFastforward
Rollershutter	Typically used for blinds	UpDown, StopMove, Percent
String	Stores texts	String
Switch	Typically used for lights (on/off)	OnOff

Table 10.1 Item types

Dimension	default unit metric	default unit imperial
<b>Length</b>	Meter (m)	Inch (in)
<b>Temperature</b>	Celsius (°C)	Fahrenheit (°F)
<b>Pressure</b>	Hectopascal (hPa)	Inch of mercury (inHg)
<b>Speed</b>	Kilometers per hour (km/h)	Miles per hour (mph)
<b>Intensity</b>	Irradiance (W/m <sup>2</sup> )	Irradiance (W/m <sup>2</sup> )
<b>Dimensionless</b>	Abstract unit one (one)	Abstract unit one (one)



Angle	Degree (°)	Degree (°)
-------	------------	------------

Table 10.2 The dimensions of the item type Number

## 10.7. Where are You? Presence Detection with openHAB

Before we move on we will add another important software module to our project, the so called network binding. This software add-on identifies all networked devices on the local network and turns them into Things, which can be used as triggers for rules. Among other things we will then be able to trigger on smartphones connected to local Wi-Fi network. This will allow us to use our Wi-Fi home network itself as a sensor for presence detection. Compared to traditional motion and light detector based presence monitoring, this approach has the advantage that it detects not only that someone is present, but also who it is.

One important condition for our presence detection to function is, that all tracked devices are always assigned the same IP address when they come online. Most routers do this automatically and only assign a new IP address to a device, if the maximum number of IP addresses in the network has been reached. However, to ensure this is always the case you can configure devices with a DHCP priority to ensure, they always appear under the same IP address.

To install the network binding we open the file `/services/addons.cfg` and add `network` to the list of bindings we have installed so far. Then we configure the binding by creating the file `network.cfg` in the directory `/services/` with the following content:

```
binding.network:allowSystemPings=true
binding.network:allowDHCPlisten=false
binding.network:arpPingToolPath=arping
binding.network:cacheDeviceStateTimeInMS=2000
```

(If you want to use arping, which in addition to system pings and the DHCP listen function allows you to ping MAC addresses, you need to install the arping tool in the openHAB directory: <http://www.habets.pp.se/synscan/programs.php?prog=arping>)

Then we restart openHAB. According to the template of the network binding we now add the following lines to our `.things` and `.items` files:

```
Thing network:pingdevice:ChristophersiPhone
[ hostname="192.168.178.31" ]

Switch Christophers_iPhone
{ channel="network:pingdevice:Christophers_iPhone:online" }

DateTime LastSeen_Christopher
{ channel="network:pingdevice:Christophers_iPhone:lastseen" }
```

In our .sitemap file we add the new frame Presence Detection with the text item Christophers\_iPhone:

```
Frame label="Presence Detection"
{
    Text label="Presence Detection" icon="motion" {
        Text item=Christophers_iPhone icon="boy_1" label="Christophers
iPhone [%s]"
        Text item=LastSeen_Christopher icon="time" label="Christopher
LastSeen [%1$tH:%1$tM]"
        Text item=Othmars_iPhone icon="man_1" label="Othmars iPhone
[%s]"
    }
}
```

In our control centre we now can see if Christophers iPhone is connected to our home network (Figure 10.17).

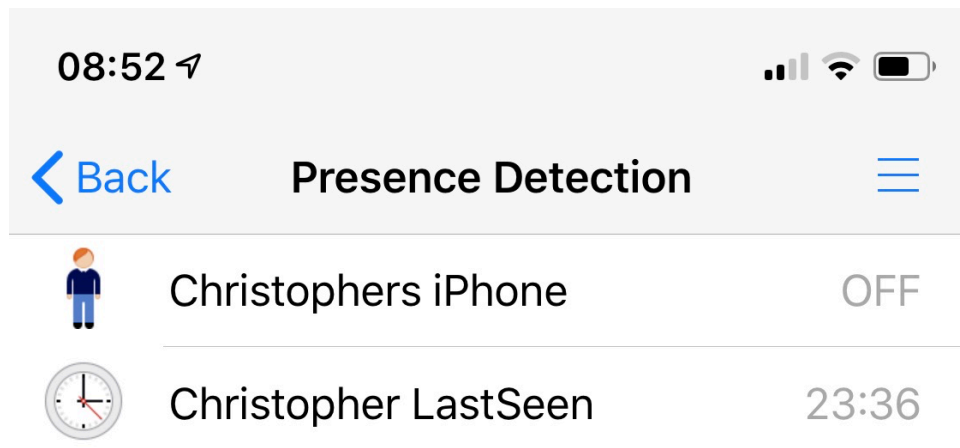


Figure 10.17 GUI for Presence Detection using the openHAB network binding

## 10.8. openHAB's Memory and Intelligence: Persistence & Rules

One crucial functionality of a building automation systems is its ability to store values. This is needed for creating charts, for data analysis, for rules which require historic values and for reloading of values in case of a system restart. In openHAB this functionality is called Persistence. To implement data persistence for openHAB you have the choice among a number of add-ons. For our project we will use the round-robin database RRD4j. The advantage of a round-robin database is, that it does not grow in size. It has a fixed allocated size. Once this size is reached, the oldest

values are overwritten by the newest ones. If you have selected DEMO after the first openHAB start and you are working with the demo configuration, the RRD4j add-on is already installed, and you do not need to follow the below installation instruction. Otherwise open Paper UI, go to [Add-ons](#), select the Persistence pane, scroll down to the RRD4j persistence entry, then select and install it. (Persistence add-ons need to be installed using the Paper UI.) Now go to [Configuration – System](#) and scroll down to the Persistence section. Select the only option, which should now be RRD4j. Now for all persistence operations per default the RRD4j database is being used. In /conf/services you will now see a new text file called rrd4j.cfg, which is being used to configure the database. For the time being we go with the default values of the database. For the item- and event-related persistence configuration the file persistence/rrd4j.persist is used.

To understand the usage of persistence and rules, we will enhance the display of our OpenWeatherMap data with daily minimum and maximum temperatures and with a temperature chart. The chart will display the temperature data on hourly, daily and weekly basis. For the calculation of the daily minimum and maximum temperatures in addition to the persistence data we will create simple rules, which will be stored in the file smarthome.rules, where all rules reside. In addition to smarthome.things, which we do not need to touch at this point, we will then have the following five files

rrd4j.persist

smarthome.rules

smarthome.items and

smarthome.sitemap

Restricting our project at this point to the display of weather and astronomical data keeps the complexity of the configuration files low, while demonstrating the essential functionalities you will need during your project. Below the content for the four files is listed. When you copy the content in the according configuration files, you will get a display in Basic UI as shown in figure 10.20.

As you can see, the items Weather\_Temp\_Max, Weather\_Temp\_Min and Weather\_LastUpdate in smarthome.items do not have channels assigned. Their values are provided by the rules "Set daily max and min temperature" and "Records last weather update time" in smarthome.rules. The persistence configuration in rrd4j.persist records the data of all items within the item group Weather\_Chart and all items with an icon name starting with Temperature. We now understand how to retrieve, process and display data with openHAB and can move on to add new data sources to our project in the next chapter.

rrd4j.persist

```
Strategies {  
  
    everyMinute : "0 * * * * ?"  
  
}
```

```
Items {  
  
  // let's only store temperature values in rrd  
  
    Temperature*, Weather_Chart* : strategy = everyMinute,  
  restoreOnStartup  
  
}
```

### smarthome.rules

```
// Sets max and min temperatures for the day  
  
rule "Set daily max and min temperature"  
  
when  
  
    Item Weather_Temperature changed or  
  
    Time cron "0 0 0 * * ?" or  
  
    System started  
  
then  
  
    val max = Weather_Temperature.maximumSince(now.withTimeAtStartOfDay)  
  
    val min = Weather_Temperature.minimumSince(now.withTimeAtStartOfDay)  
  
    if (max != null && min != null) {  
  
        postUpdate(Weather_Temp_Max, max.state)  
  
        postUpdate(Weather_Temp_Min, min.state)  
  
    }  
  
end  
  
  
// Creates an item that stores the last update time of this item  
  
rule "Records last weather update time"  
  
when  
  
    Item Weather_Temperature received update  
  
then
```

```

    postUpdate(Weather_LastUpdate, new DateTimeType())

end

smarthome.items

Group Weather_Chart

Number:Temperature Weather_Temperature "Temperature [%.1f °C]"
<temperature> (Weather, Weather_Chart)
{ channel="openweathermap:weather-and-forecast:api:wyk:current#temperature" }

Number Weather_Temp_Max "Todays Maximum [%.1f °C]"
<temperature> (Weather, Weather_Chart)

Number Weather_Temp_Min "Todays Minimum [%.1f °C]"
<temperature> (Weather, Weather_Chart)

Number Weather_Chart_Period "Chart Period"

DateTime Weather_LastUpdate "Last Update [%1$ta %1$tR]" <clock>

Number Sun_Elevation "Sun Elevation" <sun>
{ channel = "astro:sun:home:position#elevation" }

Number Sun_Azimuth "Sun Azimuth" <sun>
{ channel = "astro:sun:home:position#azimuth" }

DateTime Sunrise_Time "Sunrise [%1$tH:%1$tM]"
<sunrise> { channel = "astro:sun:home:rise#start" }

DateTime Sunset_Time "Sunset [%1$tH:%1$tM]" <sunset>
{ channel = "astro:sun:home:set#start" }

Number Moon_Elevation "Moon Elevation" <moon>
{ channel = "astro:moon:home:position#elevation" }

Number Moon_Azimuth "Moon Azimuth" <moon>
{ channel = "astro:moon:home:position#azimuth" }

String Moon_Phase "Moon Phase" <moon>
{ channel = "astro:moon:home:phase#name" }

```

smarthome.sitemap

```
sitemap smarthome label="Main Menu Smarthome"

{

    Frame label="Weather"

    {

        Text item=Weather_Temperature
valuecolor=[Weather_LastUpdate=="NULL"="lightgray",Weather_LastUpdate>90
="lightgray",>25="orange",>15="green",>5="orange",<=5="blue"]

        {

            Frame

            {

                Text item=Weather_Temp_Max
valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]

                Text item=Weather_Temp_Min
valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]

                Text item=Weather_LastUpdate
visibility=[Weather_LastUpdate>30]
valuecolor=[Weather_LastUpdate>120="orange",
Weather_LastUpdate>300="red"]

            }

            Frame

            {

                Switch item=Weather_Chart_Period label="Chart Period"
icon="chart" mappings=[0="Hour", 1="Day", 2="Week"]

                Chart item=Weather_Chart period=h refresh=600000
visibility=[Weather_Chart_Period==0, Weather_Chart_Period=="NULL"]

                Chart item=Weather_Chart period=D refresh=3600000
visibility=[Weather_Chart_Period==1]
```

```

        Chart item=Weather_Chart period=W refresh=3600000
visibility=[Weather_Chart_Period==2]

    }

}

Text label="Astronomical Data" icon="sun"

{

    Text item=Sun_Elevation

    Text item=Sun_Azimuth

    Text item=Sunrise_Time

    Text item=Sunset_Time

    Text item=Moon_Elevation

    Text item=Moon_Azimuth

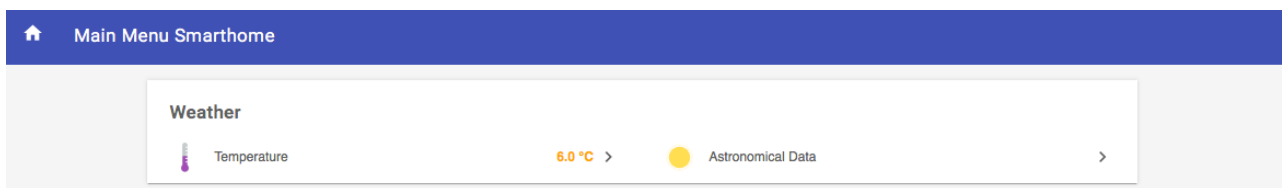
    Text item=Moon_Phase

}

}

}

```



*Figure 10.18 The main menu for weather and astronomical data in the BasicUI*

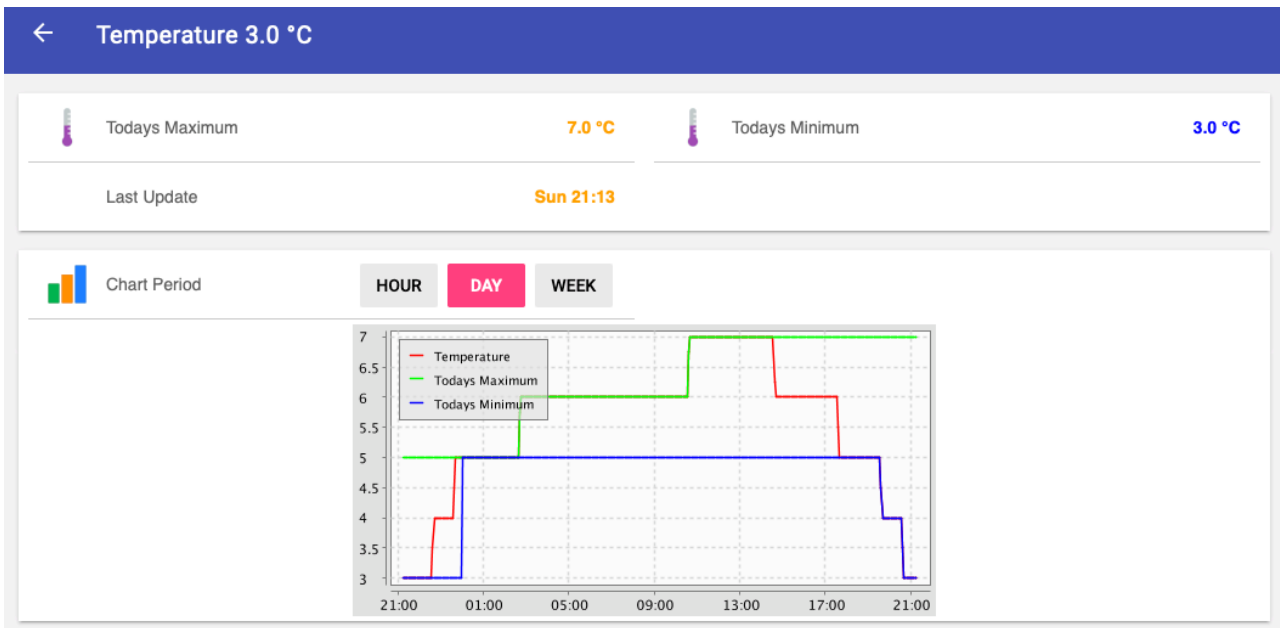


Figure 10.19 Display of processed weather data in BasicUI








 Sun Elevation	-49.17 °	 Sun Azimuth	314.34 °
 Sunrise	07:31	 Sunset	16:10
 Moon Elevation	15.89 °	 Moon Azimuth	222.87 °
 Moon Phase	Waxing gibbous		

Figure 10.20 Display of astronomical data in BasicUI



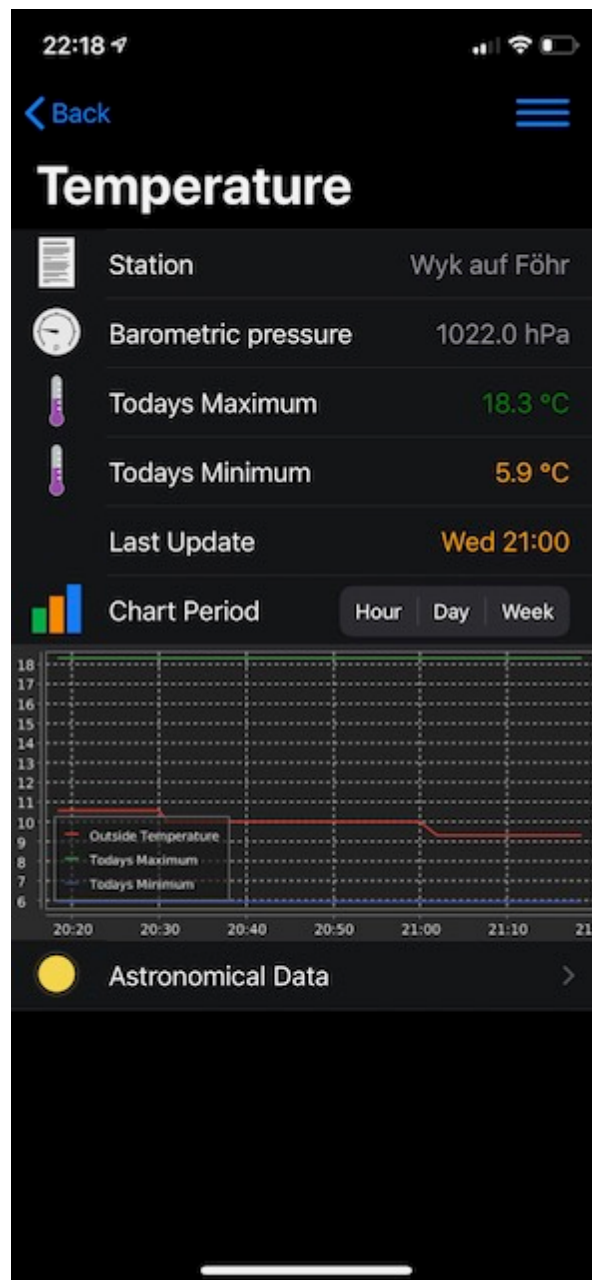


Figure 10.21 Smartphone display of openHAB weather data

## 10.9. OpenHAB Upgrades

UnderMacOS and Linux updates can simply be initiated with the command `update` in the runtime directory:

```
sudo runtime/bin/update
```

For Windows an update guide can be found [here](#).

## 10.10. OpenHAB Backup

Once your system has reached a stable state, you should start to create backups, so you always have a working fallback solution available. The backup and restore command files are located in

the directory `/usr/share/openhab2/runtime/bin/`. With the following commands you can initiate a backup and a restore of your openHAB installation:

```
cd /usr/share/openhab2/runtime/bin
```

```
sudo ./backup yourbackupfilename
```

```
sudo ./restore yourbackupfilename.zip
```

## 10.11. Migration from openHAB to openHABian

When migrating an openHAB installation from one platform to another (e.g. a PC/Mac based openHAB installation to Raspberry Pi based openHABian) it is advised to start with a clean, new openHAB respective openHABian installation and manually copy the configuration files (\*.things, \*.sitemap, etc.) to the according directories. When using a backup-file from an old hardware configuration and restore it on a new one, you might run into problems with some configuration leftovers of your old system interfering with the new one. So the best practice is to start with a fresh install, select demo-configuration, which auto installs persistence, and then manually copy the configuration files to the fresh install.