

OTHMAR KYAS

HOW TO SMART HOME



KEY CONCEPT PRESS

How To Smart Home

A Step by Step Guide for Smart Homes & Building Automation A Key Concept Book by Othmar Kyas

5th Edition

How To Smart Home

Published by Key Concept Press www.keyconceptpress.com

ISBN 978-3-944980-12-6

Fifth Edition May 2017

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 © 2017 by KEY CONCEPT PRESS

This eBook was posted by AlenMiler on AvaxHome!

Many New eBooks in my Blog:

<http://avxhome.in/blogs/AlenMiler>

Mirror: <https://avxhome.unblocked.tw/blogs/AlenMiler>

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.

Tutorial Videos and Bonus Material for Download

Bonus material for this book can be downloaded from the book website on <http://www.keyconceptpress.com/how-to-smart-home>. Tutorial Videos can be viewed on www.keyconceptpress.com/tutorials.

Notification on Updates and New Releases

If you want to be notified when an update to this book or a new release from Key Concept Press becomes available sign up [here](#). Our mailing list is exclusively used to keep you and others informed about Key Concept Press. We do not share or sell your information to any third parties
<http://www.keyconceptpress.com/newsletter.html>.

About the Author

Othmar Kyas is an internationally renowned expert in communication technology and strategic marketing. He is author of fourteen books, which have been translated into five 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 dam 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

Tutorial Videos and Bonus Material for Download

Notification on Updates and New Releases

About the Author

1 Read Me

1.1 Who is this Book for?

1.2 What You Will NOT Find

1.3 What You WILL Find

1.4 Safety First!

1.5 Take no Risks

1.6 Formatting Rules

2 The Big Picture

2.1 Smart Buildings and the Internet of Things (IoT)

2.2 The Potential for Energy Conservation

2.3 Safety Management and Assistive Domotics

2.4 Changing the World (a bit) to the Better

2.5 Bibliography

3 Key Concepts

3.1 Devices under Control

3.2 Sensors and Actuators

3.3 Home Automation Network (HAN)

3.4 Controller (Smart Hubs)

3.5 Remote Control Devices

3.6 Cloud Services

3.7 Bibliography

4 Home Automation Network Protocols

4.1 Network Address Translation (NAT)

4.2 Open Ports and Port Forwarding (Port Sharing)

4.3 UPnP

4.4 Dynamic DNS

4.5 HTTP REST

4.6 HTTP Server Push

4.7 Bibliography

5 You Don't Know What You Don't Know - Smarthome Security

5.1 Attacking the HAN

5.2 IoT Search Engines - Shodan and friends

5.3 Bibliography

6 Home & Building Automation: Markets and Trends

6.1 Market Size and Growth

6.2 Smart Devices & Deep Learning Technologies

6.3 Bibliography

7 Smart Homes for the Masses: Google, Apple, Samsung, Amazon and more ...

7.1 Google's Nest Labs and Google Home

7.2 One More Thing ... Apple HomeKit

7.3 Samsung's SmartThings

7.4 Amazon's Echo

8 To Cloud or not to Cloud - This is the Question

8.1 Securing your Project Cloud Account

9 The Project

9.1 Overview

9.2 Equipment and Prerequisites

10 The Home Control Centre: Open Remote

10.1 OpenRemote Overview

10.2 OpenRemote Controller Installation

10.3 Java Installation and Configuration under macOS

10.4 Java Installation and Configuration under Windows

10.5 First Synchronization between Designer and Controller

10.6 The Importance of Directory and File Management

10.7 OpenRemote Professional Designer

10.8 The “Hello World” App

11 A Pretty Smart Sensor: Internet Weather

11.1 OpenRemote Control via HTTP: Retrieving Internet Weather Data

11.2 Designing the App Layout

11.3 Bibliography

12 Integration of Multimedia: iTunes Remote

12.1 Scripting Basics: Shell what?

12.2 Testing it Right - Best Practice for Script Writing

12.3 Script Based iTunes Control in macOS

12.4 Smartphone Remote for iTunes (macOS)

12.5 Script Based MediaPlayer Control (Windows 10)

12.6 Script Based iTunes Control (Windows 10)

12.7 Talk to Me

13 A Little AI: Drools Rules

13.1 Wake me up Early if it Rains: iAlarm

13.2 Remember me: Maintaining State Information

13.3 From Sunrise to Sunset

14 More iDevices

14.1 Denon / Marantz Audio System Control

14.2 Device Control Using Z-Wave

14.3 Bibliography

15 Where are you? Geo-fencing!

15.1 Google Drive

15.2 If This Then That (IFTTT)

15.3 A Geo-Fencing Database on Google Drive

15.4 Google Sheets Manipulation

15.5 Writing a Geo-fencing Sheet Management Script

15.6 A Smartphone App for the Geo-Fence Database

15.7 Further considerations for using a Geo-Fencing Database: Concurrency and Hysteresis

15.8 Google Sheets Integration with OpenRemote

15.9 A Cloud Based Smarthome Control Platform

16 Industry Grade Home Infrastructure Control: KNX

16.1 What is KNX?

16.2 How does KNX Work?

16.3 The KNX Software Infrastructure: ETS

16.4 Which Operating Systems does ETS Support?

16.5 ETS on a Mac

16.6 Other KNX.org Software Tools

16.7 ETS5 Installation

16.8 Importing Vendor Catalogs

16.9 ETS5 Infrastructure Configuration

16.10 ETS5: Adding the Building Infrastructure

16.11 ETS5: Configuring the KNX Elements

16.12 ETS5: Connecting Infrastructure to Controls

16.13 Notes on Configuring KNX Devices

17 KNX Control via OpenRemote Designer

17.1 Background Pictures for the Smartphone and Tablet App

17.2 Configuring KNX Based Heating Mode Control

17.3 Smartphone Based Heating Control

17.4 Drools Based Heating Automation

18 Remote Smarthome Control

19 Cold Start: Launch Automation

19.1 Windows 10 Task Scheduler

19.2 macOS launchd

20 Troubleshooting and Testing

20.1 Preventive Maintenance

20.2 OpenRemote Heartbeat and Watchdog

21 We Proudly Present: Reporting

21.1 A Drools Reporting Rule

22 Appendix

22.1 Upgrading from OpenRemote Designer to Professional Designer (2.1 to 2.6)

22.2 Troubleshooting Strategies

22.3 Problem Symptom - Cause Pairs

22.3 A Brief Introduction to JavaScript and Google Script

23 Bibliography

1 Read Me

1.1 Who is this Book for?

This book explains state of the art smart home, building automation and Internet of Things technologies and demonstrates step by step how to apply them to real world projects. The toolset covered consists of tablets, smartphones, sensor equipped devices, the Internet and the latest wireline and wireless building automation standards. 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 that 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
- ZigBee, a smart home communication standard
- KNX, a smart home communication standard

- Drools, an open source object oriented rule engine
- OpenRemote, an open source Internet of Things software platform
- IFTTT (IF This Than That), a cloud based Internet of Things control service
- Google Sheets, a cloud based spreadsheet service, part of Googles G Suite
- Google Apps Script, a scripting environment based on JavaScript for G Suite products
- macOS *Linux Windows 10 / Java*

Parts of the project integrate consumer electronics devices, such as audio equipment from Denon and Marantz. However, project and instructions are designed, so that they can easily be adapted to other manufacturers. Be aware, however, that equipment, which is more than a few years old, probably will 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.

While not part of the concrete project described in the second part, the book also covers popular smart home solutions 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 integration with the described solution is discussed and explained. The technologies, design and planning approaches, test philosophies and security considerations discussed 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 IoT and does a much deeper dive into the technology basics, than the average smart home customer typically is 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 comprehensive 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. For the implementation of the sample project in this book I have selected a combination of a local home controller and a cloud based solution. The local home computer component is based on the open source platform OpenRemote, the cloud component on cloud services from Google and IFTTT. In combination they deliver a solution providing interoperability with most devices and platforms on the market, while allowing for full customization to needs of residential as well as commercial users.

1.4 Safety First!

For the proposed project security and availability aspects are discussed in detail and according recommendations are being made. 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. And we will explain measures against it. In the sample project described, safety and security plays a key role as well. All step by step instructions take this 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 PC 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. I cannot take any liability for any undesired outcome of the given instructions.

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.

2 The Big Picture

Home automation, at the intersection of the rapidly developing technologies Internet, mobile communication, sensor technologies, self learning software and renewable energies, has changed considerably over the course of the past years. The drivers for this development are

- the dynamic growth of the broader Internet of Things market
- increased capabilities and lower prices of home infrastructure and intelligent devices
- advances in sensoring technologies
- improved usability of mobile and stationary user interfaces driven by omnipresent smartphones and tablets
- growing motivation to conserve resources and use renewable energy
- market entrance of multinational technology and consumer electronics companies

Up to the turn of the millennium, home automation was mainly focused on installing controllable power-outlets or light switches using copper telephone lines or infrared (IR) controls. Technologies developed more than thirty years ago, which from today's perspective are slow, unreliable, and insecure, were at the heart of building control.

The rapid developments in mobile communications have introduced a technological leap forward in home automation since. Wireless networks (3G, LTE, Wi-Fi, LoRa) and smart devices with wireless communication interfaces (Bluetooth, ZigBee, Wi-Fi) are omnipresent and allow the user to take building automation to the next level. Instead of simply switching power outlets on and

off, specific and meaningful functions of consumer electronics, household devices, or infrastructure components can be stirred. As a result, instead of rudimentary functionality, home automation today can deliver capabilities that have a real impact on comfort, security, as well as energy conservation of residential, commercial and industrial buildings.

Equally important as the mobile communication revolution have been the advances in sensoring technologies, while much less in the focus of the public. Comparable with the pace of the digital evolution, sensor capabilites have improved. At the same time size and prices of sensors are at a fraction of what they used to be a few years ago. In addition new generations of digital sensors have replaced analog sensoring technologies, allowing a single sensor to measure a multitude of parameters.

Advances in usability have also contributed to the mainstream adoption of home automation. The smartphone and tablet revolution has finally brought the personal, universal remote control device to the home. Proprietary, stationary panels and control devices are phasing out, being replaced by apps, which are customized to individual use cases. They are easy to operate, to maintain, and to upgrade. With smart phone voice assistants having become mainstream, reliable voice control has also arrived for interaction with smart homes and buildings.

2.1 Smart Buildings and the Internet of Things (IoT)

With improved usability and new capabilities the motivations for installing smart home technologies have become broader as well. The vision of green buildings, capable of significantly reducing energy and water consumption, is finally becoming real. Other use cases are safety management, home automation for the elderly and disabled (assistive domotics) and remote building control. Beyond providing stand alone solutions for individual buildings, smart home technologies have become a central component of larger Internet of Things concepts such as smart cities, smart industry or smart grids.

2 .2 The Potential for Energy Conservation

Looking at the distribution of total energy consumption, the share of the private sector is significant. In the 28 European Union countries (EU-28), in 2014, 25 % of the total energy was consumed by the residential sector, in the US 22 % (Figures 2.1 - 2.4). Thus, energy conservation in homes does move the needle even from a global perspective. And the savings potential for all energy forms used in the private sector is large. Space heating and cooling takes the largest share with between 50 % and 70 % of the total residential energy usage. Water heating takes second place, followed by electric appliances and lighting.

In US households over the past 20 years the share of energy used for space heating has steadily declined, while the share for space cooling has increased. Main factors for this trend are increased adoption of more efficient equipment, better insulation, more efficient windows, and population shifts to warmer climates. Still the potential for energy conservation in the industrialized world is enormous.

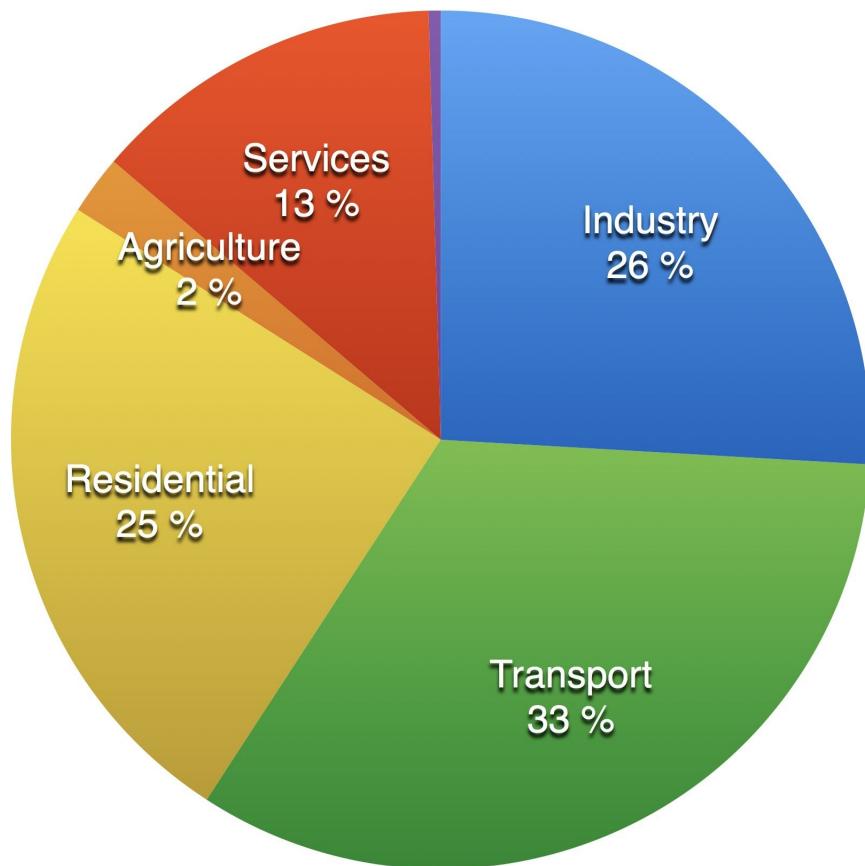


Figure 2.1 2014 EU-28 energy consumption per sector

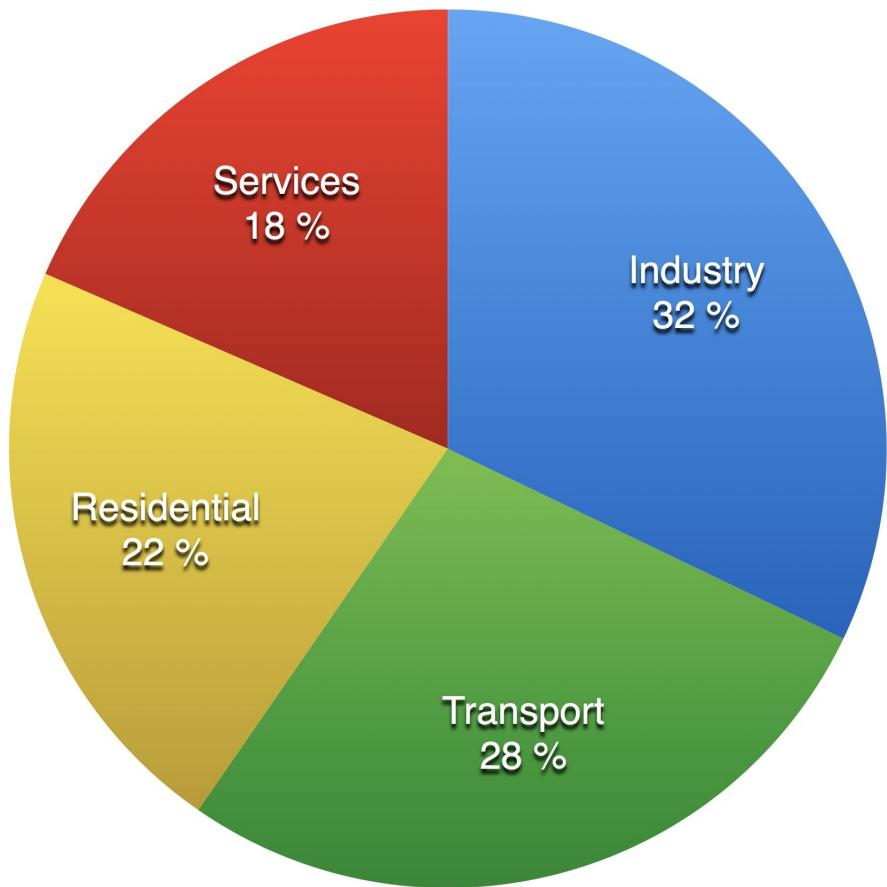


Figure 2.2 2014 US energy consumption per sector.

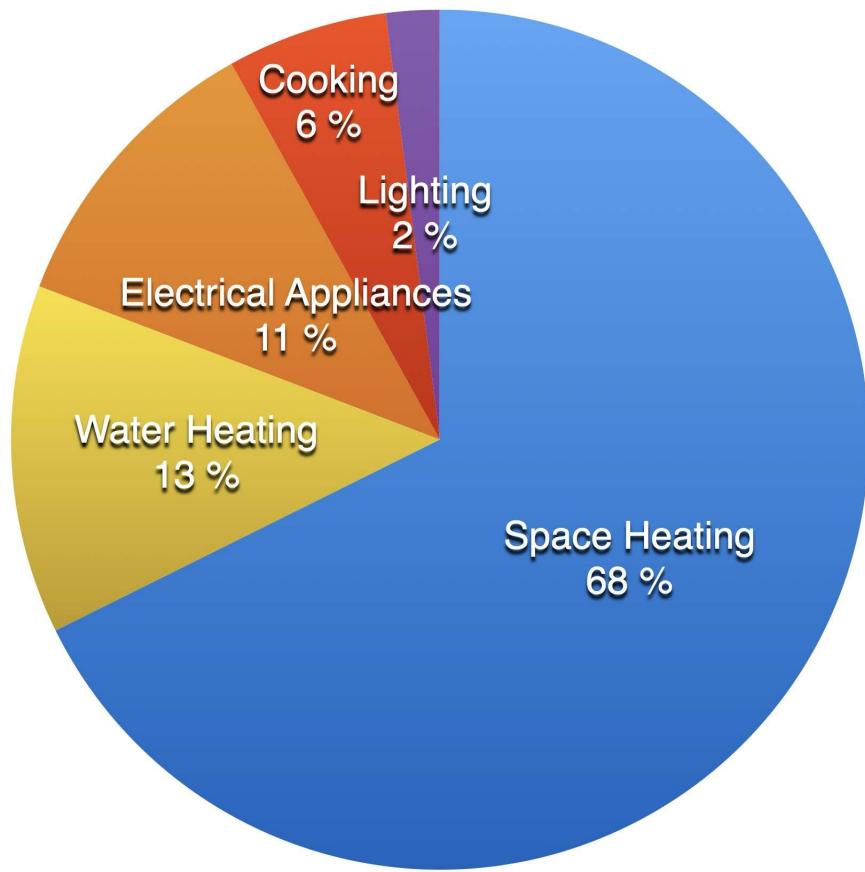


Figure 2.3 2014 Residential end-use energy split EU-28

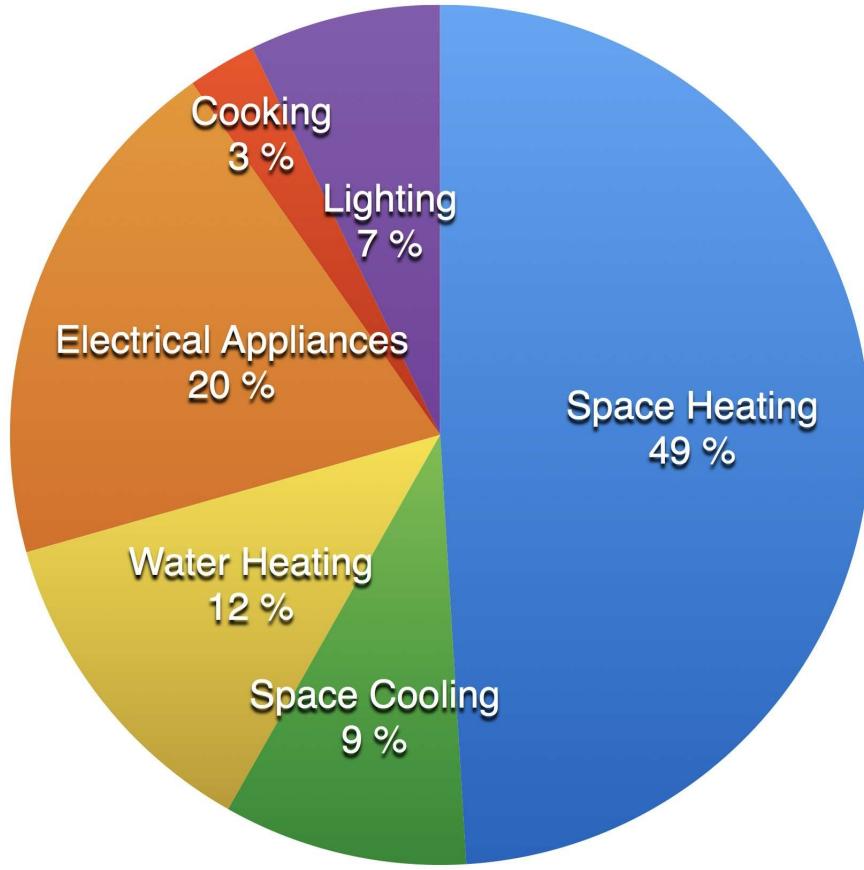


Figure 2.4 2014 Residential end-use energy split US

There are four main approaches for reducing residential energy consumption, all of which should be noted:

- building insulation
- usage of state of the art appliances
- installation of efficient water heating and space heating systems

- installation of smart building automation and control

With the advances in home automation as described above, building automation has become an increasingly attractive choice, providing the opportunity for significant savings with relatively low upfront investment. Smart appliances coordinate their operation with smart meters (home gateways), reducing overall energy consumption and avoiding load peaks. Monitoring current and past power consumption and identifying load profiles provide the basis for intelligent power management with capabilities such as:

- Intelligent heating control by automatically managing room temperature based on time, outside temperature, and presence
- Smart lighting system, managing illumination based on presence detection, sunrise, or sunset timing and room function
- Intelligent, proactive blinds, keeping the interior of the building cool or warm
- Monitoring and management of electricity consumption
- Reducing water consumption through sensor faucets and intelligent plant watering management

2.2.1 Calculating Actual Building Automation Energy Savings

Studies report electricity savings of up to 30 % using automated lighting as well as heating energy savings of 15 % - 20 % using automated heating in residential buildings. But how much is it that you can really conserve by implementing a smart home? Answering this question was the task of a major standardization effort in Europe, which has come up with a comprehensive specification on how to measure and calculate building automation based energy savings: The European standard EN15232: “Energy performance of buildings - Impact of Building Automation, Control and Building Management”. For the first time, EN15232 specifies standardized methods to assess

the impact of Building Automation and Control Systems (BACS) on the energy performance of these different building types:

- offices
- lecture halls
- education
- hospitals
- hotels
- restaurants
- wholesale & retail
- residential

The performance of building automation is categorized in four classes (A-D), A representing the highest performance building automation, D the lowest.

For each building type and each BACS Class, so called BACS Factors are given, with which the thermal and electrical energy savings can be calculated. Table 2.2 shows the description of the four BACS classes and the BACS factors for the different building types. Table 2.1 displays the percentage of thermal savings by installing building automation of efficiency classes A and B in reference to the standard class C.

	Thermal Energy				Electrical Energy			
	D	C	B	A	D	C	B	A
BACS Class								
Offices	1,51	1,00	0,80	0,70	1,10	1,00	0,93	0,87
Lecture hall	1,24	1,00	0,75	0,50	1,06	1,00	0,94	0,89
Education	1,20	1,00	0,88	0,80	1,07	1,00	0,93	0,86
Hospitals	1,31	1,00	0,91	0,86	1,05	1,00	0,98	0,96
Hotels	1,31	1,00	0,85	0,68	1,07	1,00	0,95	0,90
Restaurants	1,23	1,00	0,77	0,68	1,04	1,00	0,96	0,92
Wholesale&Retail	1,56	1,00	0,73	0,60	1,08	1,00	0,95	0,91
Residential	1,10	1,00	0,88	0,81	1,08	1,00	0,93	0,92



Table 2.1 Thermal and electrical savings for various building types using BACS classifications

Class A	High performance BACS Networked room automation with automatic demand control Scheduled maintenance Energy monitoring Sustainable energy optimization
Class B	Advanced BACS Networked room automation without automatic demand control Energy monitoring
Class C	Corresponds to standard BACS Networked building automation of primary plants No electronic room automation,thermostatic valves for radiators No energy monitoring
Class D	Non energy efficient BACS Without networked building automation functions No electronic room automation No energy monitoring

Table 2.2 BACS class definition

2.2.2 Smart Grids need Smart Buildings Smart homes also allow for integration with smart power grids, which are in build out around the globe, driven by renewable energy generation on the rise. Smart meters and smart gateways can only work, if a home control and

automation infrastructure is in place. This infrastructure then can interact with the supply and demand driven electricity cost in smart power grids. Wind and sun based renewable energy generation introduces significant energy level fluctuations in the utilities' power grids. Thus, for example, it can make sense to cool down the freezer two or three degrees below normal operation during times of high wind, so it can stay off longer in times of the day with lower energy supply.

By being able to continuously monitor energy levels and prices in a smart power grid, and by scheduling (delay or early start) high energy processes such as

- heating up the hot water tank
- operating the dish washer and washing machine
- cooling the freezer and refrigerator

smart meters can contribute significant energy savings without impacting the comfort level of residents.

2.3 Safety Management and Assistive Domotics

Another application for state of the art home automation is remote building control and safety management with features such as

- controlling the vacant home (temperature, energy, gas, water, smoke, wind)
- feeding and watching pets
- watering plants indoors and outdoors
- presence simulation to keep out intruders
- assistive living systems (assistive domotics), allowing elderly and handicapped people to stay home safe through reminder systems, medication dispensing devices, blood pressure and pulse monitoring as well as emergency notification

2.4 Changing the World (a bit) to the Better

The standardization of home and building automation based on open Internet technologies, omnipresent smartphones and sensor equipped devices have been the catalyst for many manufacturers to integrate smart control functionality into their products by default. Internet of Things concepts have embraced smart homes and smart buildings as an integral part of their solutions. This has further accelerated the acceptance and deployment of smart home technologies, which have arrived in the mainstream. So (finally) everything is there, that is needed for a truly intelligent home. Following along the explanations and instructions in this book, you will be able to test this claim by translating the described concepts into real world solutions.

Smart home technologies have come a long way. Technological advances, climate change and demographic transitions have redefined what they are and why they are being deployed. Rather than being a futuristic niche for geeks and luxury home owners, smart homes have become an integral part of the life of millions. Capable of helping to solve pressing problems such as global warming or demographic transitions, they have the potential to changing the world (a bit) to the better.

2.5 Bibliography

US Department of Energy. (2016): Total Energy Interactive Table Browser.
[www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?
f=A&start=2013&end=2014&charted=3-6-9-12](http://www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?f=A&start=2013&end=2014&charted=3-6-9-12)

Eurostat European Commission (2016): Final energy consumption, by sector.
ec.europa.eu/eurostat/data/database?node_code=tsdpc320

Enerdata. (2015) Energy Efficiency Trends in Residential in the EU.
www.odyssee-mure.eu/publications/efficiency-by-sector/household/

US Department of Energy. (2016) Annual Energy Outlook 2015.
<https://www.eia.gov/beta/aeo/#/?id=4-AEO2015&cases=ref2015&sourcekey=0>

Baggini, Angelo and Lyn Meany. (2012) Application Note Building Automation and Energy Efficiency: The EN 15232 Standard. http://www.leonardo-energy.org/sites/leonardo-energy/files/Cu0163_AN_Building%20automation_v1_bis.pdf

3 Key Concepts

In the following chapters we will discuss the components and the communication infrastructure which builds the home automation network (HAN). You will understand how the various technologies, protocols and devices work together and how they provide a secure, performant base for home and building automation. Chapter four provides as detailed discussion of the underlying mechanisms when smart devices communicate with each other, with a smart hub, with a cloud service or a remote smartphone. This will allow you to make informed technology decisions and in turn according product choices for your own projects. In chapter five we will look into security risks and breaches as they occur in smart home environments and how to avoid them. A description of common exploits and attacks on smart home networks will complement this section of the book.

From a topology perspective home automation consists of the following building blocks:

- devices under control (DUC)
- sensors and actuators
- the home automation network (HAN)
- the building controller (smart hub)
- remote control devices (tablets, smartphones, panels) and
- cloud services

Figure 3.1 shows an outline of the smart home topology.

3.1 Devices under Control

Devices under control are all components, such as home appliances, consumer electronics, lighting, or window blinds, which are connected to and controlled by the home automation system. An increasing number of things come with such functionality built-in through integrated web-servers, WiFi-, Bluetooth-or Z-Wave-interfaces. Components without built-in control capabilities can often be equipped with adapters in order to integrate them with the smart home infrastructure.

3.2 Sensors and Actuators

Sensors are the eyes and ears of the home network. There are sensors for a wide range of applications such as measuring temperature, humidity, light, liquid, gas, detecting movement or noise.

Actuators are the hands of the home network. They are the means of how the smart home can actually do things in the real world. Depending on the type of interaction required, there are mechanical actuators such as pumps and electrical motors, door locks or electronic actuators such as electric switches and dimmers.

3.3 Home Automation Network (HAN)

The home automation network (HAN) provides the connectivity between devices under control, sensors and actuators on the one hand and the controller (smart hub) along with remote control devices and cloud services on the other hand. From a communication network perspective there are three technology options for home and building automation control networks:

- powerline communication
- wireless communication
- wireline communication

3.3.1 Power Line Communication

The power line communication principle uses existing electric power lines in buildings to transmit carrier wave signals in the frequency range of 20 kHz to 100 MHz. The long dominant, decades old, low speed power line standard X.10, has been finally replaced by HomePlug, which became the IEEE 1901 standard in 2010. AV2.1, the latest version of HomePlug, published in 2014, is able to achieve transmission speeds of up to 500 MBit/s while performing data encryption with a 128-bit AES cipher. The main advantage of power line communication is the low price for its components and that fact, that no additional wiring is required. The downside is, that power line distribution units can significantly impact transmission speeds. In some cases the design of the electric wiring can even prohibit the coverage of parts of the electric power line infrastructure in a building. Major vendors of HomePlug products are TP-Link, D-Link, Devolo, Netgear, GigaFast and Zyxel.

3.3.2 Wireless Communication

Today a large number of wireless communication technologies for building and

home automation systems are available. Transmission speeds and reach of the various technologies depend on transmission frequency and modulation, and range from 20 kBit/s to 250 kBit/s and 60 ft (20 m) to 3000 ft (1000 m) respectively. Other important considerations are power consumption and location accuracy. Technological advances have significantly improved all performance aspects of wireless transmission technologies in recent years. The main drivers for wireless technologies gaining popularity in home automation have been:

- smart devices with wireless connectivity built in becoming widely available at low prices
- new or updated home automation standards increasing safety, throughput and reducing power consumption
- wireless plug and play home automation solutions offered by large vendors and telecommunication companies

While wireless building control for years has been plan B for lower end, post-construction projects, the evolution of wireless technologies along with a steep decline in cost has changed the industry. Today, Z-Wave, ZigBee, BLE (Bluetooth low energy), WiFi and RFID interfaces are available fully integrated in a wide range of controllable power-outlets, light switches, and household appliances. Many audio and video consumer electronic devices come with Bluetooth or WiFi interfaces, ready to stream content from the Internet, and ready to be fully controlled via smartphones. And the technological evolution continues. In 2016 the Wi-Fi Alliance announced 802.11ah (HaLow), a new standard optimized for Home Automation and Internet of Things (IoT) applications. It features a number of benefits compared to current Wi-Fi technologies. Using the 900 MHz band, other than conventional Wi-Fi networks operating in the 2.4 GHz and 5 GHz bands. This allows wireless signals to reach almost twice as far as current Wi-Fi radios, while using less power to broadcast. With this WiFi routers will become far more efficient at killing dead spots. Also mobile phones and WiFi based IoT devices will be able to communicate over greater distances while conserving battery life.

Another new technology player in home automation is the IEEE standard 6LoWPAN, which has been adopted for home automation applications by the industry alliance Thread, which was founded back in 2014. While the IEEE standard 6LoWPAN is based on IEEE 802.15.4 as Z-Wave, it supports IP-addressable smart devices, allowing them to be directly accessed from the Internet. While 802.11ah and 6LoWPAN products are just starting to reach the market, these new standards will change the home automation landscape. But also the established wireless technologies are upgrading their specifications to improve battery life, transmission speeds and safety. Examples are Bluetooth Smart (5.0), Z-Wave Plus and ZigBee 3.0. Devices using energy harvesting technologies such as products based on the EnOcean standard, which operate wireless control links using energy retrieved from the environment through temperature changes, light changes or push button mechanical energy, are also becoming widely available. To sum it up, the technology and standards clutter in wireless home automation networks is here to stay for the foreseeable future. Table 3.1 lists all major open standards used for wireless building automation today. It also includes the proprietary Insteon standard, which combines wireless transmission with powerline based communication. The powerline communication in Insteon networks serves as a backup in case wireless transmission is distorted or unavailable.

	First Released	Range (indoor/outdoor)	Speed	Frequency
Z-Wave	1999	100 m	250 kBit/s	908.42 MHz
Z-Wave Plus	2013	167 m	250 kBit/s	908.42 MHz
ZigBee	2003	30m - 500m	250 kBit/s	2.4 GHz, 868 MHz
ZigBee 3.0	2014	30m - 500m	250 kBit/s	2.4 GHz, 868 MHz
Wireless Hart	2004	50m / 250m	250 kBit/s	2.4 GHz
MiWi	2003	20m / 50m	20 kBit/s, 40 kBit/s, 250 kBit/s	868 MHz 2.4 GHz
EnOcean	2001	30m - 300m	125 kBit/s	902 MHz (EU), 315 MHz
DASH7 (active RFID)	2004	1000m	200 kB/s	433 MHz
.....

THREAD / 6LoWPAN	2015	30m-500m	250 kBit/s	2.4 GHz
Bluetooth LE (4.0)	2014	30m-100m	250 kBit/s	2.4 GHz
Homekit AccessoryProtocol (HAP)	2014	10m	1 MBit/s	2.4 GHz
HaLow	2016	up to 1000m	100 kBit/s - 8 MBit/s	900 MHz
Bluetooth Smart (5.0)	2016	> 100m	125 kBit/s 1 MBit/s 2 MBit/s	2.4 GHz
Insteon	2005	45 m	13 kBit/s	915MHz

Table 3.1 Wireless building automation standards

* LR-WPAN (Low Rate Wireless Personal Area Networks)

** Strongly depends on topology, frequency and distortion of the sensor

3.3.3 Wireline Building Automation

The main open standards for wireline based building automation are KNX, G.hn, LON and HomePNA. KNX is a European (EN50090, 2003) and international (ISO/IEC 14543-3, 2006) standard for home and building automation. The abbreviation KNX stands for Konnex, and replaces the older European standards EIB (European Installation Bus), Batibus (primarily used in France), and EHS (European Home Systems). Today in Europe more than 75% of industrial building automation solutions as well as most upscale residential smart homes are realized using KNX. Over the past years, KNX has started to be adopted in many regions of the world outside of Europe as well.

LON (Local Operating Network), originally introduced in 1990 by Echelon Corporation and an ISO/IEC 14908 standard since 2008, is the building automation solution of choice for large scale automation projects such as airports, stadiums, or street lightning. Contrary to the hierarchical KNX architecture it uses a decentralized approach. In large installations local

information can be processed locally, without being sent to a central control node. This allows for the scalability and redundancy needed in public installations with high availability requirements.

Another wireline transmission standard for the networked home, which has been primarily deployed in North America, is HomePNA. Specified by the Home Phoneline Networking Alliance it's first release was published in 1990. For almost twenty years it was mainly used to transmit IPTV services over telephone wires, since 2006 also over coaxial cables. In an effort to put an end to the multitude of non-interoperable home networking technologies, the ITU-T working group G developed a single transmission standard for home networks, which can be used over any type of wire (telephone, coax, power line and fiber): G.hn (hn - home network). The initial release of G.hn was published in 2010. The industry alliance behind the standard is the HomeGrid Forum (HGF). In 2013 the Home Phoneline Networking Alliance merged with the Home Grid Forum. G.hn provides data transmission rates of between 250 MBit/s and 2 Gbit/s, depending on the physical medium.

3.3.4 Topologies of Home Automation Networks

All three home automation network technology categories— power line, wireless, and wireline - have significantly improved in transmission speed, reliability and interoperability through standardization efforts over the past years. In general, HANs based on power line communication and wireless transmission are dominant in residential home automation due to lower component prices and installation cost. Wireline based control networks are found in the premium residential segment and in control applications for public, commercial and industrial buildings. Since devices with wireless communication capabilities built in have became widely available, the number of wireless HANs has been growing fast, particularly in the residential segment. The single biggest challenge in HANs today however is the mix of different technologies, implementation variants and closed, proprietary solutions, which one has to deal with. Practically all HANs today consist of a multitude of technologies out of historic reasons or because of product constraints. Figure 3.1 provides a

topology overview of how the different technologies in a HAN interconnect with each other. Looking at figure 3.1 you can see, that HAN communication takes place in three building blocks:

- the communication within the HAN
- communication between the HAN and external cloud services
- communication between the external remote control devices and the HAN or external cloud services

Communication within the HAN

The two main communication hubs inside the HAN are the DSL router and the home controller (smart hub). The DSL router on the one hand connects all WiFi and Ethernet based components within the HAN with each other. At the same time it provides connectivity to the Internet. The wireline technologies such as KNX, HomePlug (powerline) or HomePNA (phone wires, coaxial cable) are connected to the router through IP/Ethernet converter units. The wireless technologies other than WiFi (the only wireless standard routers support), connect to the home controller or smart hub. Most smart hub vendors also support wireless communication standards other than WiFi such as Bluetooth, Zigbee and Z-Wave. If the home controller (smart hub) consists of a PC with home automation software installed, connectivity to Bluetooth, Zigbee, Z-Wave or THREAD is provided by appropriate USB interface units. In addition to supporting multiple HAN network technologies, the responsibility of the home controller is to manage smart devices (adding and removing devices) and to execute rules and scenarios. Further (through the DSL router) the home controller provides connectivity to external cloud services and remote control units (smartphones), which are away from home. It is important to understand that WiFi based smart devices (e.g. WiFi controlled wall plugs, lights bulbs, etc.) are directly connected to the router, as if the smart hub was not there. This is the case because smart hubs (at least in residential environments) do not create their own local area network, but use the one maintained and managed by the router. To connect and manage WiFi based smart devices the home controller sends out device discovery messages on to the local area network. Once a device responds,

it can be added to its set of managed smart devices and become part of the controller rule and scenario definitions (Figure 3.1).

Communication between the HAN and external cloud services or external remote control devices

To connect to external cloud services and to allow for remote control from outside of the house, most controllers use the secure HTTP REST protocol. HTTP REST also allows cloud services to communicate to the controller from their end using the HTTP Server Push protocol. An alternative for remote access to the controller from outside is to set up a VPN connection. Through the VPN tunnel a client from outside can be added to the local network, allowing it to act as a local client (Figure 3.1). Both options, HTTP/HTTP REST and VPN are safe and secure, if set up and operated properly. Most major vendors of smart hubs (Samsung SmartThings, Wink Labs, etc.) or cloud centric products (e.g. Google NEST) are using HTTPS REST.

When smart devices are operated stand alone without a home controller, in order to make their setup less complex for the user, some vendors use the UPnP protocol. UPnP (Universal Plug and Play Protocol) allows networked devices to automatically discover each other by exchanging commands. UPnP also allows devices in conjunction with the IGD-PC (Internet Gateway Device Protocol) protocol to automatically open ports on the DSL router, in order to communicate with external devices and services. So, by using UPnP, smart devices can directly communicate with a cloud service to become activated, to install software updates or to allow a remote user to access them (Figure 3.1). While device setup using UPnP is relatively easy, it is also insecure. Multiple major UPnP related security incidents have occurred in the past, with the consequence that leading IT security organizations (e.g. CERT) are recommending not to use UPnP. Beyond its security issues UPnP deployments do not scale well due to their extensive use of multicasts, which leads to the consumption of significant network resources in networks with large number of nodes. A widely used UPnP based solution is the WeMo product line from Belkin. Chapter 5 provides more details on the security issues related to UPnP.

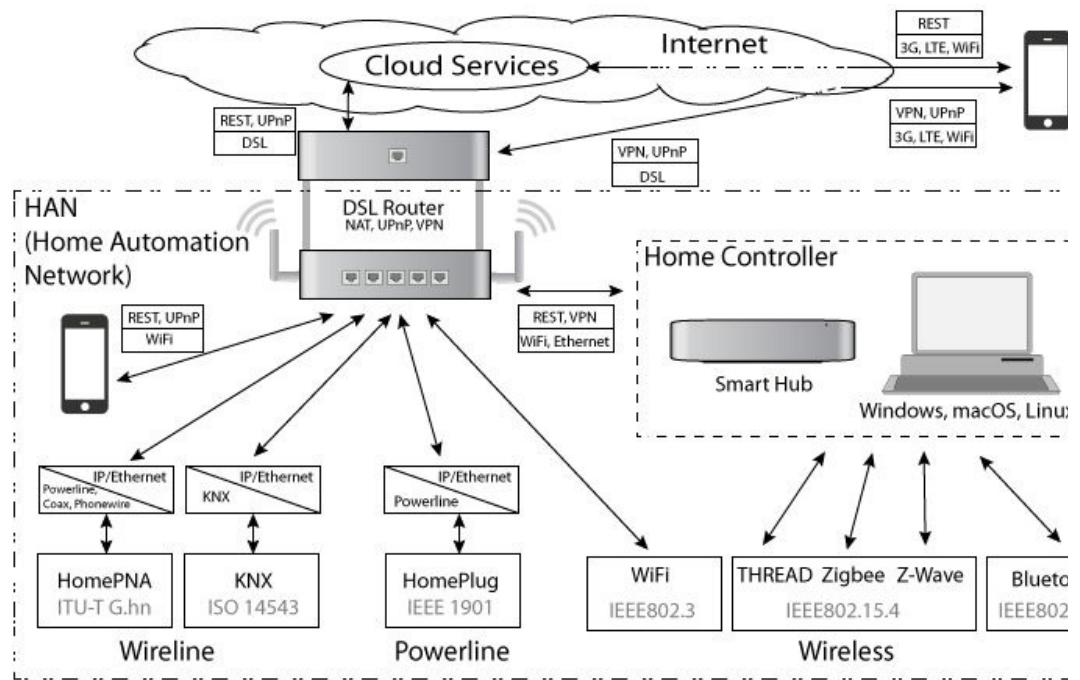


Figure 3.1 Communication infrastructure of the Home Area Network (HAN)

3.4 Controller (Smart Hubs)

The home or building controller, often also referred to as smart hub, is the entity, which acts as the brain of the building automation system. It collects information through sensors and receives commands through remote control devices. It acts based on user commands or a set of predefined rules using actuators or means of communication such as loud speakers, emails, pumps, motors, door locks or telephones. Controllers can be either

- local computer systems with home automation software and interface adapters for HAN communication technologies (Z-Wave, Bluetooth, Zigbee)
- integrated devices with specialized hardware and software (smart hubs) or
- cloud based services

While most smart home solutions are built around a local controller, there are also decentralized approaches, which attempt to avoid a controller in order to reduce complexity. An example is Apple's Homekit solution, which can be operated with or without a controller unit. Chapter seven provides details about the smart home solution from Apple as well as from other major vendors.

3.4.1 The Home Controller

For residential home automation, the controller typically is an always-on standalone Linux Windows macOS computer, running the control application for the house. Typically it comes embedded in form of a vendor branded, easy to set up smart hub, or it is an always on computer, equipped with the necessary HAN communication interfaces and running home automation software (Table 3.3). Higher end residential and industrial buildings use dedicated high availability, redundant controller systems with uninterruptible power supplies (UPS).

The home controller ideally provides connectivity to all smart devices in the building and thus typically needs to support multiple home automation infrastructure protocols (WiFi, Z-Wave, Bluetooth, KNX, etc.). Its tasks are to

- securely register (and deregister) devices
- to interface to remote control devices such as smartphones, tablets or panels
- to interface to cloud services and
- to execute predefined rules.

Some functions such as rule definition and execution or remote control via smart phone from outside of the house are often implemented through a cloud service. The degree of integration with an external cloud service varies from vendor to vendor. Some solutions are implemented very cloud centric with key operational functions executed in the cloud, others use the cloud merely as a remote control and monitoring hub to be accessed while away from home. Solutions without any cloud component have become rare, however they still can be accessed and controlled from away using VPN. A VPN connection is a secure communication tunnel from a PC, a smart phone or a tablet, which can be set up via the public Internet to connect to any private local network, such as the home automation network of a building. As said however, today most home controller vendors offer products which combine local operation with some form of cloud service. As long as the smart phone or tablet is local and connected to the home WiFi network, the control of smart home devices takes place directly via the smart hub. If the control device is away from the building, the communication takes place via the cloud service, which in turn communicates with the smart hub.

Vendor	Smart Hub	Home Infrastructure Connectivity
Samsung smarthings.com	SmartThings Hub	Z-Wave, ZigBee, Bluetooth LE and Wi-Fi; IFTTT support
Wink Labs wink.com	Wink Hub	Z-Wave, ZigBee, Bluetooth LE, Wi-Fi, Kidde, Lutron ClearConnect
Logitech logitech.com	Harmony Home Companion	Wi-Fi and Bluetooth
	Insteon Hub	Bluetooth and Wi-Fi

Insteon insteon.com	Vista Automation Module	Z-Wave, Wi-Fi, and Bluetooth
Honeywell security.honeywell.com		

Table 3.2 Smart Hub Vendors and Products

The alternative to the ready to use plug and play smart hub solutions, as listed in table 3.2, are open source building automation platforms. While requiring some technical expertise and time for configuration and customization, these solutions are more flexible and provide more comprehensive functionality than their commercial pendants. Being non commercial and vendor neutral, a core philosophy of their architecture is to ensure, that none of the data, which is generated through remote control and automation procedures, leaves the controller to be exploited by third parties. The three largest and most popular open source platforms are OpenRemote, openHAB and Home Assistant (Table 3.3).

Platform	Operating Systems	Cloud Service	Application Protocol	Home Infrastructure Connectivity
OpenRemote openremote.com	Windows 10, Linux, macOS, Raspbian	designer.openremote.com (Configuration only)	HTTPS REST	WiFi, Z-Wave, ZigBee, Bluetooth, KNX, EnOcean, Insteon, HTTP, IFTTT, ...
openHAB openhab.org	Windows 10, Linux, macOS, Raspbian	myopenhab.org	HTTPS REST	WiFi, Z-Wave, ZigBee, Bluetooth, KNX, EnOcean, Insteon, HTTP, IFTTT, ...
Home Assistant home-assistant.io	Windows 10, macOS, Ubuntu, Raspbian	—	HTTPS REST, MQTT	WiFi, Z-Wave, ZigBee, Bluetooth, KNX, EnOcean, Insteon, HTTP, IFTTT, ...

Table 3.3 Open Source Smart Hub Platforms

3.4.2 OpenRemote

OpenRemote was started in 2008 by JBoss founder Marc Fleury and Juha Lindfors. It is an open source building control and automation framework with a

large and active user and developer community. The controller software runs on top of Java Virtual Machine (JVM), which provides a wide range of options for selecting the underlaying hardware system such as Linux, macOS, Windows or Raspberry PI. The configuration interface is provided through a cloud service (designer.openremote.com). All configuration changes have to be downloaded from the cloud to the local controller installation, before they take effect. The user interface is provided in form of an iOS / Android app. Given it's long history, OpenRemote supports a large number of building automation platforms, devices and protocols. Besides in residential homes OpenRemote is also used in commercial, industrial and municipality projects.

3.4.3 Home Assistant

Home Asssistant is a vendor neutral, open source home control software platform started by Paulus Schoutsen in 2014. It is written in Python 3 and controlled through a web interface optimized for use on mobile devices. Home Assistant integrates with several hundred different software platforms, services and devices. As application protocols it uses HTTP REST and MQTT. Home Assistant is running on a local server, with all data stored in a local SQLight database. It does not provide a cloud component, but does interface with cloud services such as IFTTT.

3.4.4 openHAB

openHAB, as Home Assistant and OpenRemote, a vendor neutral, open source home control platform, was founded by Kai Kreutzer in 2011. It runs on top of Java Virtual Machine (JVM), which provides a wide range of controller system options such as Linux, macOS, Windows or Raspberry PI. It provides a web based user interface and iOS / Android apps for control and configuration. A cloud service under myopenHAB.org provides additional options for remote control as well as for integration with other cloud services such as IFTTT. By supporting Google Cloud Messaging (GCM) and Apple Push Notifications (APN) it can send push notifications to mobile phone apps.

3.4.5 Cloud (only) Controller Platforms

Along with device automation and control evolving into a large sector stretching across multiple industries under the notion of Internet of Things, IoT cloud based platforms have become available, which also target the home and building automation market. Examples are Particle (<https://www.particle.io>) or IFTTT (<https://ifttt.com/discover>). In particular IFTTT has become popular by providing a simple, intuitive cloud based solution for connecting smart devices and services, which could not interoperate otherwise. The market for cloud only based controller solutions is still in it's infancy, however, with new mobile access technologies optimized for IoT (e.g. 5G mobile communication) it has the potential for significant growth in the years to come.

3.5 Remote Control Devices

One of the main reasons for the increased acceptance of home automation systems in the residential segment is, that with the omnipresence of smart phones and tablets the need for dedicated automation control devices has vanished. Within a few years, literally all home automation systems on the market have introduced smartphone and tablet based control applications. In addition, advances in voice recognition have finally brought voice based control to smart homes. The remote control devices can either directly connect to the home area network (HAN) while they are inside the house, or from outside of the house via a VPN connection, via a cloud service or (not recommended) via open ports and dynamic DNS services.

3.6 Cloud Services

As outlined above, over the past years vendors have started to offer home automation functions such as rule execution or smartphone based remote control through cloud services. The degree of their integration with local home automation solutions varies from vendor to vendor. Some approaches are very cloud centric with key operational functions executed in the cloud, others use the cloud merely as a remote control and monitoring hub to be accessed while away from home. With cloud services along with mobile access technologies (3G, LTE, 5G) becoming performant, reliable and inexpensive, they have been established as a standard component of home automation solutions.

Having understood the topology of home and building automation infrastructures, in the following chapter we can now take a deeper dive into the underlying communication processes.

3.7 Bibliography

Weiping Sun, Munhwan Choi and Sunghyun Choi (2013): 802.ah - A long range 802.11 WLAN. [IEEE 802.ah: A long range 802.11 WLAN](#)

WiFi Alliance (2016): Wi-Fi HaLow. <http://www.wi-fi.org/discover-wi-fi/wi-fi-hallow>

Home Grid Forum (2013): Converging Technologies. [Converging Technologies - Moving from HomePNA to G.hn.](#)

IEEE Computer Society (2011): IEEE Standard for Local and metropolitan area networks Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). <http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf>

International Telecommunication Union (2012): Recommendation ITU-T G.9959 Short range narrow-band digital radio communication transceivers – PHY and MAC layer specifications. <https://www.itu.int/rec/T-REC-G.9959>

EnOcean Alliance (2013): EnOcean Wireless Standard ISO/IEC 14543-3-10. https://www.enocean-alliance.org/en/enocean_standard/

The Thread Group (2017): What is Thread? <http://threadgroup.org/What-is-Thread/Connected-Home>

4 Home Automation Network Protocols

This chapter provides more detail on some of the key home acces network (HAN) communication protocol procedures, especially as they relate to configuration settings on the network router, which are essential to get right. While this discussion might be demanding for the reader, who has not yet dealt with communication protocols yet, it's understanding is of utmost importance and the prerequisite for the ability to plan, design and install a safe and performant smarthome network. Trying to set up or change the configuration parameters of routers, smart hubs and smarthome devices without exactly knowing what you are doing is like playing the lottery: chances to get it right are very small. So unless you know what NAT, UPnP, IGD, or HTTP-REST is and how they function, do not skip this chapter.

4.1 Network Address Translation (NAT)

The way a DSL router connects the devices in a private local area network (e.g. a HAN) to the public Internet is through the process of Network Address Translation (NAT). For all communication with the public Internet the DSL router is being assigned an external IP address, which has to be unique in the entire Internet. (This address by the way is rotated and changed by the Internet service provider every 24 hours or when the router restarts, unless you have a more expensive permanent IP address.) Towards the private local area network the IP addresses do not need to be unique, but can randomly be selected from the below ranges: 10.0.0.0 – 10.255.255.255

172.16.0.0 – 172.31.255.255

192.168.0.0 – 192.168.255.255

The IP addresses of all network packets from local devices, which need to be sent to the public Internet, are replaced by the NAT instance of the router with its external IP address, before they are being sent on to the public Internet. In addition to the source IP address, the router also stores data such as destination address and port number of every packet, so it can easily route reply packets from the public Internet to their correct destination in the local network.

4.2 Open Ports and Port Forwarding (Port Sharing)

Port numbers are 16 bit long identifiers for networked applications, which are managed by the Internet Assigned Numbers Authority (IANA). Common port numbers are:

- HTTP: 80
- File Transfer Protocol: 20/21
- Secure Shell (SSH): 22
- Telnet: 23
- Simple Mail Transfer Protocol (SMTP): 25

Every packet in the network must contain a destination port number, based on which the network stack of a networked device decides to which application to send the packet to, and based on which routers decide, whether or not to process this packet . An "Open Port" is a router configuration setting for a specific port number, which tells the router to process packets, which carry this port as destination port. As an example, if you are running a web sever and a FTP server, you will need ports 80 for web, and 20 / 21 for FTP to be open. A "Closed Port" is a router configuration setting, which tells the router to actively refuse or completely ignore packets, which carry this port as destination port. To summarize, the term "Open Port" (also "Port Forwarding" or "Port Sharing") describes a router setting, where all packets, which are arriving from the public Internet, and which carry a particular port as destination port, are forwarded to a particular IP address in the private network. Popular applications for configuring port forwarding are

- Operating a public HTTP server within a private LAN
- Allowing Secure Shell (SSH) access to a host on the private LAN from the Internet

- Allowing FTP access to a host on a private LAN from the Internet
- Running a publicly available game server within a private LAN
- Accessing home automation devices (surveillance cameras, thermostats) from the Internet

The problem with open ports is, that malicious code can be hidden inside packets sent to any open port, which is why accepting packets on open ports can cause your system to be hacked and infected. Applications which are associated with a port such as a FTP server, a HTTP server or a surveillance camera might get attacked. Smart devices within a HAN, which are reachable via an open port (see next section on UPnP), are of particular concern, since their software is by far less advanced and protected than the one of a FTP or HTTP server. They often respond to arbitrary requests and are an easy target for an attack. Since you normally will not run an HTTP or FTP server from your HAN, on your HAN-DLS router you should not have an open port at all.

4.3 UPnP

The UPnP protocol (Universal Plug and Play) together with the Internet Gateway Device Protocol (IGD) provides the functionality to automatically configure port forwarding on DSL routers. A typical use case in HANs are remote controllable webcams, refrigerators, baby phones, wall plugs or light switches, which use UPnP to connect to an external cloud service. Many routers have UPnP enabled per default, so that a user with a UPnP capable smart device can setup such devices without any changes to the router. UPnP capable smart devices such as products from the Belkin WeMo series then automatically connect to an external cloud service, which is used for software updates or for remote control through a smartphone app. Once this connection is established, the cloud service then can in turn connect to the device inside the HAN. As an example, port numbers Belkin's WeMo devices use are 49153 and 49154. The fact that smart devices using UPnP can open arbitrary ports on DSL routers without administrative rights is a major security risk. In 2013 the US Computer Emergency Readiness Team (CERT) issued a vulnerability note on UPnP (<http://www.kb.cert.org/vuls/id/922681>), in 2016 the German BSI issued a similar warning (https://www.bsi-fuer-buerger.de/BSIFB/DE/Service/Aktuell/Informationen/Artikel/Botnetz_iot_24102) following the massive DDNS attack against the Internet service provider Dyn. I strongly recommend to disable UPnP on your router and not to use any UPnP based products.

4.4 Dynamic DNS

Dynamic DNS resolves the problem, that the IP address of residential Internet/DSL routers is dynamically re-assigned by the Internet service providers (ISP), which causes it to change every 24 hours or every time the router reboots. This is why the router and with it any device inside the private network can only be reached from the outside, if its current public IP address is known. Here the dynamic DNS services come into play. They work as follows: Routers, which need to be accessible from the Internet, periodically send their current IP address to a dynamic DNS server, where it is associated with a dedicated domain name. When this domain name is called, the dynamic DNS server responds with the current IP address of the router. Dynamic DNS services are used to connect from the public Internet or from any other network outside of the private network (e.g. from the office) to home computers, home security systems, webcams or a privately hosted websites located inside the HAN. Dynamic DNS is also (mis)used in combination with remote control software such as SSH, RDP or VNC, since it is much easier than setting up a VPN connection. The most common way to use dynamic DNS in smart home environments is to combine dynamic DNS with port forwarding to access IP surveillance cameras, network digital video recorders, thermostats, refrigerators or wall switches. Unfortunately all of the above applications pose a significant security risk. The reason is, that dynamic DNS service provider domains offer an easy way to identify hosts, which run insecure IoT devices. Many IoT devices have vulnerabilities and their software is commonly not updated in order to fix them. While the usage of dynamic DNS in itself is not inherently harmful, it does add risks to your HAN. In the past several malware campaigns have utilized dynamic DNS services as part of their payload distribution. Using dynamic DNS services is an invitation for potential intruders. They can easily take control of your network if you use dynamic DNS in combination with open ports, as explained in the next chapter.

4.5 HTTP REST

REpresentational State Transfer (REST) is a technology developed based on the Internet standards HTTP and URI, allowing the secure and stateless communication between networked devices. Stateless means, that no client information needs to be stored on the server between two requests, which is why this technology is ideally suited for IoT and smart home applications. Each request from any client contains all the information necessary to service that request. Most RESTful web services are implemented using the OAuth2.0 security standard, which uses session based authentication, either by establishing a session token via a POST command, or by using an API key contained in the POST command. Usernames, passwords, session tokens, and API keys do not appear in the transmitted URLs, as otherwise they could be captured in webserver logs. Many REST based IoT devices and cloud services such as Google NEST or Samsung SmartThings use a REST/OAuth 2.0 based architecture. While it does require intervention by the user before a REST connection between two devices or a device and a service is activated, it is significantly more secure than using open ports or UPnP. Before a device (e.g. a smart thermostat) or an application (e.g. the home controller software) can connect to another application or device using REST/OAuth 2.0, it must obtain an access token, that grants access to the RESTful API. In case of an application, this is done the first time the application is started, in case of a device, during device set up. Once the user (e.g. the owner of a cloud service account) grants the permission to connect, an access token is being sent. This access token then is either manually entered in the device (e.g. NEST thermostat) or, in case of an application, it is stored as a secret key on the host computer of the application. To further increase security, these tokens have limited lifetimes. Once their end of life is reached, new access tokens are sent in response to special refresh tokens.

4.6 HTTP Server Push

HTTP Server push allows to initiate communication from the public Internet (e.g. a cloud service) to a device inside a private network (e.g. HAN) without configuring open ports. This is done by the server (e.g. the cloud service) not terminating a connection after response data has been sent to a client. The server basically leaves the connection open. Now a cloud based rule, which needs to execute a service, or a remote control request from a smart phone (via the cloud service) can be sent immediately. HTTP Server Push was introduced as part of the HTTP 2 standard (RFC 7540) in 2015.

Given the significant risk UPnP, dynamic DNS and open router ports pose, the much safer approach of HTTP REST and HTTP Server Push have gained wide acceptance in the industry. Today the combined use of REST, OAuth 2.0 and HTTP-2 Server Push provides a secure framework for communicating from private networks to the public Internet and vice versa.

4.7 Bibliography

Carnegie Mellon University, Software Engineering Institute (2011):
Vulnerability Note VU#357851 - UPnP requests accepted over router WAN
interfaces. <http://www.kb.cert.org/vuls/id/357851>

Postscapes (2017): IoT Standards and Protocols.
<https://www.postscapes.com/internet-of-things-protocols/>

Dominique Guinard, Iulia Ion, and Simon Mayer (2011): In Search of an Internet
of Things Service Architecture: REST or WS-*?
<http://www.vs.inf.ethz.ch/publ/papers/dguinard-rest-vs-ws.pdf>

5 You Don't Know What You Don't Know - Smarthome Security

In the above chapters we have looked in detail at some of the communication protocols in private local area networks. We have outlined the risks associated with traversing the border from private networks to the public Internet. In particular I recommend to avoid the usage of dynamic DNS services in combination with port forwarding and (or) the usage of UPnP, which automatically configures port forwarding and dynamic DNS without user intervention. To make sure you truly understand the potential threat such configurations pose, in this chapter I want to provide a brief description of

how such attacks work.

5.1 Attacking the HAN

In the first step a potential intruder identifies potential domain names, which could point to IoT or smart home devices, and which are visible from outside the private network. This is done using a bulk DNS guessing tool like subbrute (<https://github.com/TheRook/subbrute>), which is a so called subdomain bruteforcer. It works as follows: When you set up a dynamic DNS service, you are typically given a domain name from your dynamic DNS service provider, which you can amend with a subdomain of your choice. For example, when signing up for a dynamic DNS account with DNSdynamic, you are provided with the domain ns360.info which you for example can amend to myroomlight.ns360.info Feeding subbrute with the domain of the dynamic DNS provider and a dedicated word list, which contains vendor names and descriptions for IoT devices (a starting point for such lists can be found at <https://github.com/danielmiessler/SecLists>), you can quickly identify a large number of potential domains, which point to IoT devices. In the next step you feed this list to a port scanner, which screens the hosts of these domains for open ports. To accelerate the search, the port scan is typically restricted to the ports, which are known to be used by vendors of IP cameras or smart home devices such as 49153 for Belkin WeMo devices or 85 for Swann security cameras. When the port scanner finds an open port, the devices, to which these ports point, will often respond with information about vendor, software version or a login screen. This response is called the fingerprint or the banner of the application. The final step of the attack then is to guess the password (which more often than none is the default password) or to exploit a known vulnerability of the discovered devices' software version, in order to take over control of the device and subsequently potentially the network.

5.2 IoT Search Engines - Shodan and friends

Tools, which automatically identify open ports of Internet connected routers, have been around for many years. A relatively new approach is Shodan, which combines automatic port scanning with a search engine database. The service was introduced in 2009 by John Matherly, and has been optimized to identify devices, which are connected to or which are accessible from the public Internet. Shodan scans ports, which are typically related to routers or IoT devices such as web cams, thermostats, etc.: HTTP/HTTPS 80, 8080, 443, 8443, FTP 21, SSH 22, Telnet 23, SNMP 161, SIP 5060, or RTPS (Real Time Streaming Protocol) 554. If an application or device answers to a request with its fingerprint, Shodan stores this information, and adds it to it's database. Other tools for automated application fingerprinting have been around for years, either for the purpose of web application penetration testing or for use by hackers. Well known tools are httpprint, Xprobe2 or P0f3. There are also online fingerprinting services such as the one from netcraft (http://toolbar.netcraft.com/site_report?url=undefined#last_reboot), which can be used to analyze vulnerabilities from online. Shodan itself can also be used to test the visibility of your private network and it's components by pointing it to the external IP address of your network router. This allows a first assessment about how visible your home area network and it's components are from the outside. Figure 5.1 shows the feed of a private webcam, which has been identified to be accessible from the Internet by Shodan.

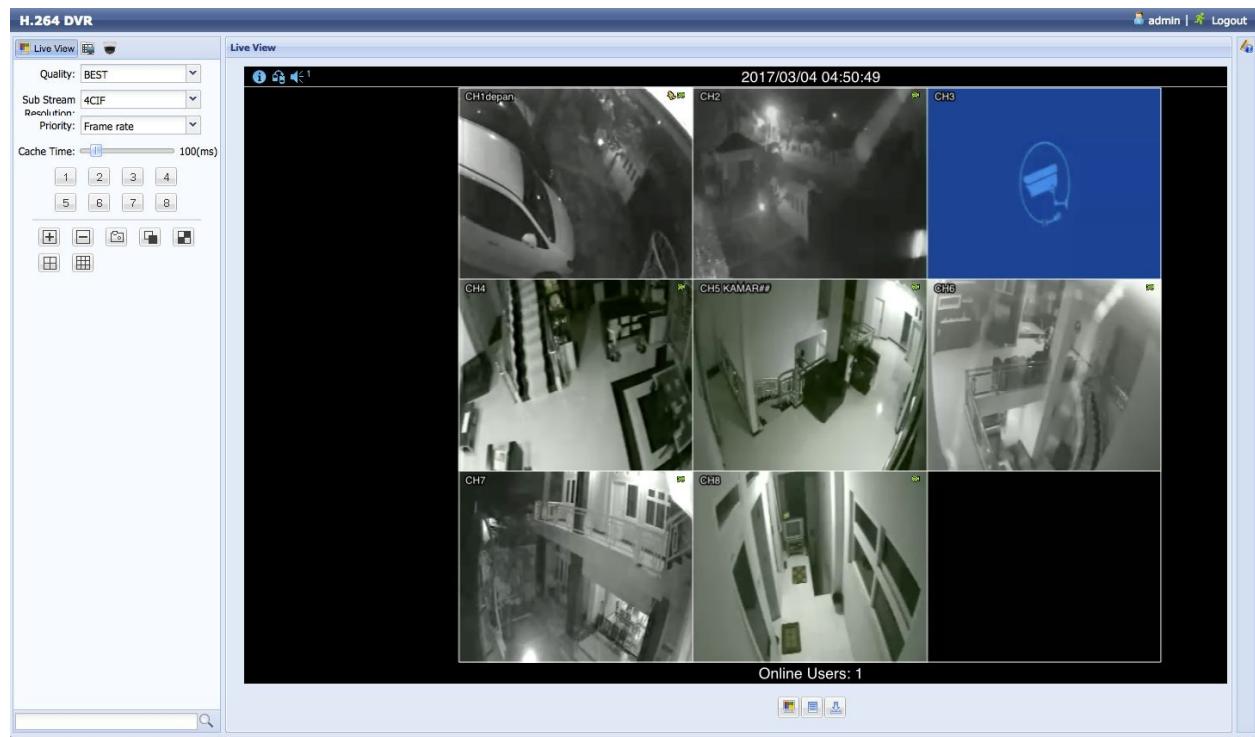


Figure 5.1 Private webcam feed accessible from the Internet as identified by IoT search engine Shodan

5.3 Bibliography

Earlence Fernandes, Kevin Eykholt, *et al.* (2016): Security Analysis of Emerging Smart Home Applications. <https://iotsecurity.eecs.umich.edu>

6 Home & Building Automation: Markets and Trends

The traditional differentiation between expensive, proprietary industrial building control systems and residential smart home automation is blurring. Over the past years these two market segments have changed drastically and are increasingly overlapping. Expensive proprietary solutions have become more open standards based and less expensive. Low end solutions for residential customers have become more sophisticated, are using the same technologies as industrial systems and have moved out of a market niche, becoming mainstream. A development similar to what happened when the

markets for professional and home PCs blended a few decades ago. While the needs for reliability, redundancy and robustness of professional building control systems have led to the development of many proprietary standards, now the pace of the digital evolution has caught up with these requirements. In addition, new functionalities for smart building control are arriving at speeds, which proprietary standards cannot match anymore. Recent examples are

- smart grid and smart meters
- web/IP enabled home appliances
- web/IP enabled consumer electronics
- Internet based information and services such as supply demand based energy prices or weather, traffic and location information.
- smartphone or tablet based home automation control customizable for individual users

- voice based control functionalities
- self-learning algorithms integrated in smart devices to automatically learn usage patterns

6.1 Market Size and Growth

Being one of the fastest growing segments within the IoT industry, the market for smart home products is forecasted to continue its rapid growth in the years to come. While market growth projections vary significantly between various market research reports, all predict a healthy, continuous growth in all regions of the world. Zion Market Research estimates the worldwide residential smart home market, covering the segments

- Smart Kitchen
- Security & Access Control
- Lighting Control
- Home Healthcare and
- HVAC Control

to grow from USD 24.10 billion in 2016 to USD 53.45 billion in 2022. Adding the market segments for commercial and industrial buildings with the segments

- Facilities Management Systems
- Security & Access Control Systems and
- Fire Protection Systems

MarketsandMarkets expects the total market of building control to grow from USD 53.66 billion in 2016 to USD 99.11 billion by 2022 (Figure 6.1).

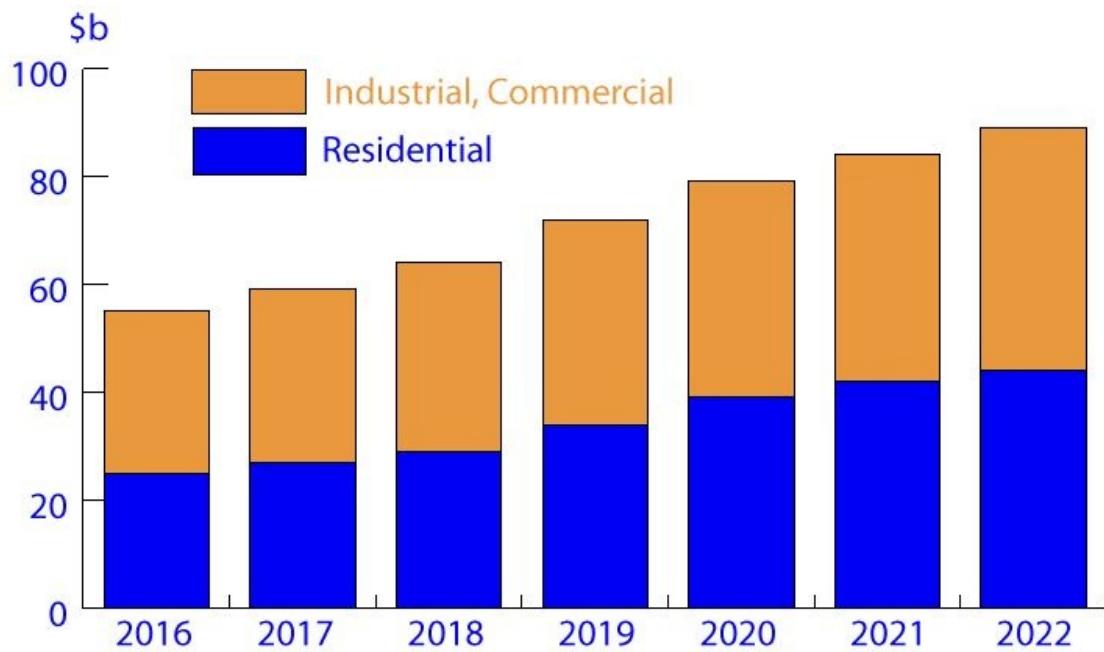


Figure 6.1 Building automation market size projection in billion US\$ 2016 - 2022 (Sources: MarketandMarkets (2017), Zion Market Research (2016), keyconceptpress.com (2017))

6.2 Smart Devices & Deep Learning Technologies

With the competition intensifying in all smart device market segments, products are becoming more sophisticated, easier to use and more secure. The integration of deep learning (or self) learning algorithms is taking this development to the next level. An example for applications of deep learning technologies are refrigerators, which analyse usage and eating patterns in order to predict which tasks should be undertaken and when. Other implementations are found in thermostats and air conditioning control systems, which analyse the behaviour of residents in order to learn which room requires which temperature when. From a product perspective the smart device categories with the highest growth potential and with high adoption rates by residential households are:

- intelligent thermostats
- intelligent security systems
- intelligent refrigerators
- Fitnessband & smart watch and
- intelligent vacuum cleaners

6.3 Bibliography

marketsandmarkets (2017): Building Automation System Market by Communication Technology. Global Forecast to 2022.

<http://www.marketsandmarkets.com/Market-Reports/building-automation-control-systems-market-408.html>

Zion Market Research (2016): Smart Home Market (Smart Kitchen, Security & Access Control, Lighting Control, Home Healthcare, HVAC Control and Others): Global Industry Perspective, Comprehensive Analysis and Forecast, 2016-2022 <https://www.zionmarketresearch.com/sample/smart-home-market>

7 Smart Homes for the Masses: Google, Apple, Samsung, Amazon and more ...

While the smart home market has experienced double digit growth rates in recent years, it were the years 2014 and 2015, by when it truly became mainstream. In 2014 three of the largest consumer product and service companies made bold entries into the smart home market. Apple introduced it's HomeKit architecture, Samsung spent more than 200 million US\$ for home automation startup SmartThings and Google acquired learning thermostat maker Nest Labs for 3.2 billion US\$. In 2015 Amazon announced Echo, a speaker system with an integrated voice-activated digital assistant, which quickly evolved into the corner stone of the company's home automation strategy. In 2016 Google followed with Google Home, it's version of a voice controlled intelligent speaker system for the smart home.

7.1 Google's Nest Labs and Google Home

Nest Labs was founded in 2010 by two former Apple engineers with the focus on self learning thermostats. The key innovation of Nest Labs centered around the fact, that most people do not program their thermostats because it is too complicated. Nest thermostats automatically create a heating or cooling schedule based on the daily routines of the residents. Initially the residents frequently set the target room temperature by turning the Nest thermostat wheel several times a day. Storing these settings the thermostat is capable of building a temperature schedule. Nest thermostats have to be connected to the Internet to receive software updates. Since part of their functionality requires location information determined by postal zip-codes, international deployment outside of the US has been slow initially. At the end of 2014 Nest acquired streaming video camera maker Dropcam and has since integrated its products with Dropcam's surveillance capabilities. Dropcam recordings can be triggered by Nest smoke detector alarms. Dropcam motion alerts are automatically turned on when Nest thermostats are being set to „away“. Third generation Nest devices support three communication standards:

- Wi-Fi: 802.11b/g/n 2.4 GHz, 802.11a/n 5 GHz
- Thread 802.15.4 at 2.4 GHz and
- Bluetooth 4.0 Low Energy (LE)

The WiFi interface is used to connect Nest devices to the Internet (and the Nest cloud service) and to smart home controllers. The Thread interface is used to communicate with other Nest devices, ensuring communication among them even when there is no WiFi network available. And Bluethooth LE is used for setting up and for configuring Nest devices using smartphones or tablets. Nest Lab's Thread protocol (<http://www.threadgroup.org>), is based on the 6LoWPAN standard (IPv6 over IEEE 802.15.4 LR-WPAN). An open source version of the standard was released in 2016 under the name OpenThread. It uses the same transport protocol as ZigBee and WirelessHART, which is why existing 802.15.4 products eventually could be upgraded to the Thread protocol via a

software update.

Google Home, introduced in late 2016, is - similar as Amazon's Alexa - a speaker system with an integrated voice-activated digital assistant and the capability to control smart home devices. The Google Home hardware is controlled by Google Assistant, Google's voice-recognition system, which is also found on Google's Pixel phone. Google Home includes two built-in far-field microphones, that are always listening in to conversations occurring in its neighbourhood, ready to react, when it hears the right trigger words "Hey Google" or "OK Google". The only communication standard Google Home supports is WiFi (802.11b/g/n/ac (2.4GHz/5Ghz)). Thus it can only directly control and connect to devices which either natively support WiFi (e.g. Belkin's WeMO devices), Google Nest devices, or WiFi based smart hubs. In order to connect to smart devices using other wireless standards, a bridge is required. An example are the Zigbee based Philips Hue light bulbs, which are controlled by the Philips Hue hub, which is a Zigbee to WiFi bridge. Using the Google Home apps AutoVoice and Tasker technical savvy users can create custom conversation actions for the Google digital assistant. Google Home interfaces with smart home platforms such as SmartThings, Belkin WeMo, Google Nest or IFTTT.

7.2 One More Thing ... Apple HomeKit

With its HomeKit framework Apple has made a strategic move to enter the smart home market. The large installed base of smartphones and tablets with the powerful voice assistant Siri provide the platform for a basic, easy to use, plug and play type smart home solution. The core of the Apple HomeKit specification consists of the three elements

- home configuration database
- HAP HomeKit Accessory Protocol
- API for HomeKit Apps

As transport protocol Apple has specified IP (LAN, WiFi) and BLE (Low Energy Bluetooth). Using the HomeKit API third party developers can build iOS applications, which discover HomeKit compliant accessories and add them to the home configuration database, access the database and communicate with configured accessories and services. In addition to iOS applications Apple's voice assistant Siri has also access to HomeKit, allowing for voice based smart home control.

Accessories which are not HomeKit compliant can connect to the HomeKit infrastructure through bridging devices (HomeKit Bridges). However this approach is limited to accessories which

- offer no user control
- have no physical access (such as door locks) and which
- use non competing transport layer technologies such as ZigBee or Z-Wave

This basically restricts HomeKit bridging to simple sensors, which do not use WiFi or Bluetooth. Examples for ZigBee to HomeKit bridges are Philips Hue hub or the Focalcrest Z-Wave / Zigbee to HomeKit bridge (<http://www.focalcrest.com/solutions/smart-home-hub/>).

All WiFi or Bluetooth based sensors as well as all smart home components, which offer active user control (e.g. thermostat controllers, light switches, door locks) have to implement the HAP protocol and enter the Apple MFi (Made-for-iPhone/iPad) program. Software bridges to integrate HomeKit with wireline smart home technologies such as KNX or HomePlug are currently not available.

HomeKit devices can either be directly controlled via an Apple iOS device from within the HAN, or from away using an Apple TV (3rd or 4th generation) or an iPad (iOS 10 or higher) as a HomeKit gateway along with an iCloud account. Even without a HomeKit gateway HomeKit devices can be configured to work together or to operate automated following triggers such as turning off lights if the front door is locked, or turning on lights if the smartphone is nearing home. When you use an Apple TV or an iPad as a smart home hub, you have more automation capabilities. Automation rules can be based on location changes, time of the day, device behaviour or sensor message. The home hub can also directly relay commands to WiFi and Bluethooth based home devices from a remote smartphone or tablet. (Bluetooth devices need to be within 25 feet of the home hub).

Compared to other smart home solutions Apple Homekit is a closed ecosystem, which allows hardly any customization. On the contrary PC based smart hubs and integrated smart hubs such as SmartThings, Google Home or Amazon Alexa provide APIs, allowing the integration of third party devices and customizations which are far more complex than what HomeKit offers.

7.3 Samsung's SmartThings

The third large consumer products company in 2014 to make a serious effort towards smart home technologies was Samsung with it's acquisition of US startup SmartThings (<http://www.smartthings.com>). Core of the SmartThing solution is an easy to use smartphone app (iOS, Android), which communicates to the SmartThing Hub, which in turn controls Z-Wave and Zigbee compliant smart home accessories. The SmartThings Hub can directly communicate with the smartphone app as long as it is within its range. In parallel it connects to a cloud service, which serves as the communication hub when communicating with the building from away. For the creation of custom rules SmartThings provides an integrated development environment (<http://docs.smartthings.com/en/latest/tools-and-ide/>). The SmartThings platform interoperates with smart home platforms such as Amazon Echo, Google Home, Google Nest and IFTTT.

7.4 Amazon's Echo

Amazon's Echo was the first smart home controller, which came in form of a smart speaker. Introduced in 2015, it consists of built in speakers, seven microphones, the voice-activated digital assistant Alexa and communication interfaces for Wi-Fi 802.11a/b/g/n and Bluetooth. The Echo (unless deactivated through the Microphone Off button) is always listening in to conversations occurring in its neighbourhood, ready to react, when it hears the trigger words "Alexa", "Echo" or "Amazon". The Amazon Echo interfaces with major smart home device manufacturers and solutions such as Nest, Wink, Belkin WeMo, Philips Hue, and IFTTT. In 2017 Amazon released Tap, a miniature version of Echo with the same capablities, but with much smaller built in speakers and the capability to connect to external speakers. Using the Amazon Voice Service (AVS) API any third party product with a microphone and a speaker can be customized to respond to the Alexa wake command. Using the Alexa Skills Kit Alexa can be customized and extended (<https://developer.amazon.com/alexa-skills-kit>).

8 To Cloud or not to Cloud - This is the Question

In spite of the trends towards open standards, for the realization of smart home projects the variety of wireline and wireless standards in combination with proprietary vendor solutions remains a challenge. Any architecture with the objective to go beyond point solutions will need to be built upon a central, rule based home controller, capable of connecting to devices via multiple technologies. In most homes at least part of the control infrastructure will be based on WiFi and wireline technologies for the foreseeable future. In newly built residential homes as well as in commercial and public buildings for security and reliability reasons wireline technologies will continue to serve as the backbone for the control of key infrastructure elements such as power outlets, lighting and HVAC (heating, ventilation and air conditioning). As we have seen, the various smart home solutions take different approaches at addressing the vast range of different customer needs and use cases. One important fundamental question you need to consider when planning your smarthome infrastructure is, whether or not you want to make use of cloud technology.

In recent years, cloud services for smart home applications have become widely available. This is true at the component level as well as the platform level. At the component level the wave of product developments for the promising IoT market has led to the integration of advanced communication capabilities for intelligent devices, from wall switches, lamps and thermostats to window shades and refrigerators. The latest generations of these devices come with optional cloud capabilities, some, like the Google NEST thermostats, even do require a cloud service for operation. At the platform level an increasing number of solutions have become available as well. Device centric cloud platforms (e.g. Belkin's WeMo cloud) typically are restricted to the products of single vendors and hardly allow for any customization or interoperability with third party products. Smart home service platforms on the other hand typically support multiple vendors through a proprietary application. Integration with new devices

and customization is generally possible via platform APIs. As more and more cloud platforms become available, efforts for standardization and interoperability of cloud services are beginning to show results. A core technology for cloud service interoperability is REST (REpresentational State Transfer). REST, as already discussed in chapter four, a method which provides service client interoperability without the need of storing information on the server between two service requests. REST was originally used to design WWW technologies such as HTTP (Hypertext Transfer Protocol) and URI (Uniform Resource Identifier).

With the above trends a solution architecture, which integrates cloud solutions, is a viable option for residential smarthome solutions as well as for professional building automation projects. Questions, which need to be answered in order to arrive at an informed decision include:

Aspect	Question
Functionality	Are cloud services available for the infrastructure components I am planning to use?
Performance, Scalability, Availability	Are cloud based home automation control platforms available, which provide the performance, the scalability and the availability I require now and in the future?
Security & Privacy	Are cloud services available, which meet the security and privacy requirements I have today and in the future?
Price and Availability	How does pricing for cloud services compare with a traditional, local solution? Will this service continue to be available in the foreseeable future?

Table 8.1 Considerations for cloud platform based building control

Cloud solutions have improved in all aspects over the past years, however whether or not they are a viable option for your project depends on your requirements and the answers to the above questions. Today cloud service providers operate state of the art data centres, which are independently audited based on international standards such as SSAE16 / ISAE 3402 Type II (Service Organization Control), ISO 27017 (Cloud Security) and ISO 27018 (Cloud Privacy). Leading cloud service providers such as Amazon and Google guarantee 99,95 % availability of their cloud service and by default encrypt all

customer data. In addition to the availability of your cloud service you also need to factor in the uptime your Internet Service Provider (ISP) guarantees for your Internet connection.

8.1 Securing your Project Cloud Account

When you choose to implement the cloud based components (Google Docs, IFTTT) of the smarthome project in the second half of the book, I strongly recommend to select a secure account password, or, even better, activate the 2-Step-verification option. 2-Step verification is a simple but powerful way to significantly improve the strength of your authentication. Go to Google Docs and select [*My Account – Sign-in & security*](#). Under the section [*Signing in to Google*](#) you can set up 2-Step verification for your account. I further recommend to run the [*Security Checkup*](#) and the [*Privacy Checkup*](#), which is available at myaccount.google.com. All data in Google Docs accounts are encrypted per default using Perfect Forward Secrecy (PFS), and for all communication to and from Google Docs the usage of HTTPS (Hypertext Transfer Protocol Secure) is enforced.

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 called a development sprint. 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 chapter we will start with installing and configuring the open source smart building and Internet of Things (IoT) platform OpenRemote, which will serve as the home controller for the project. The OpenRemote framework will also allow us to build a customized smartphone and tablet control app in little time with no programming skills required. Later, the OpenRemote controller will also be used to run the automation rules, which we will define during the course of the project.

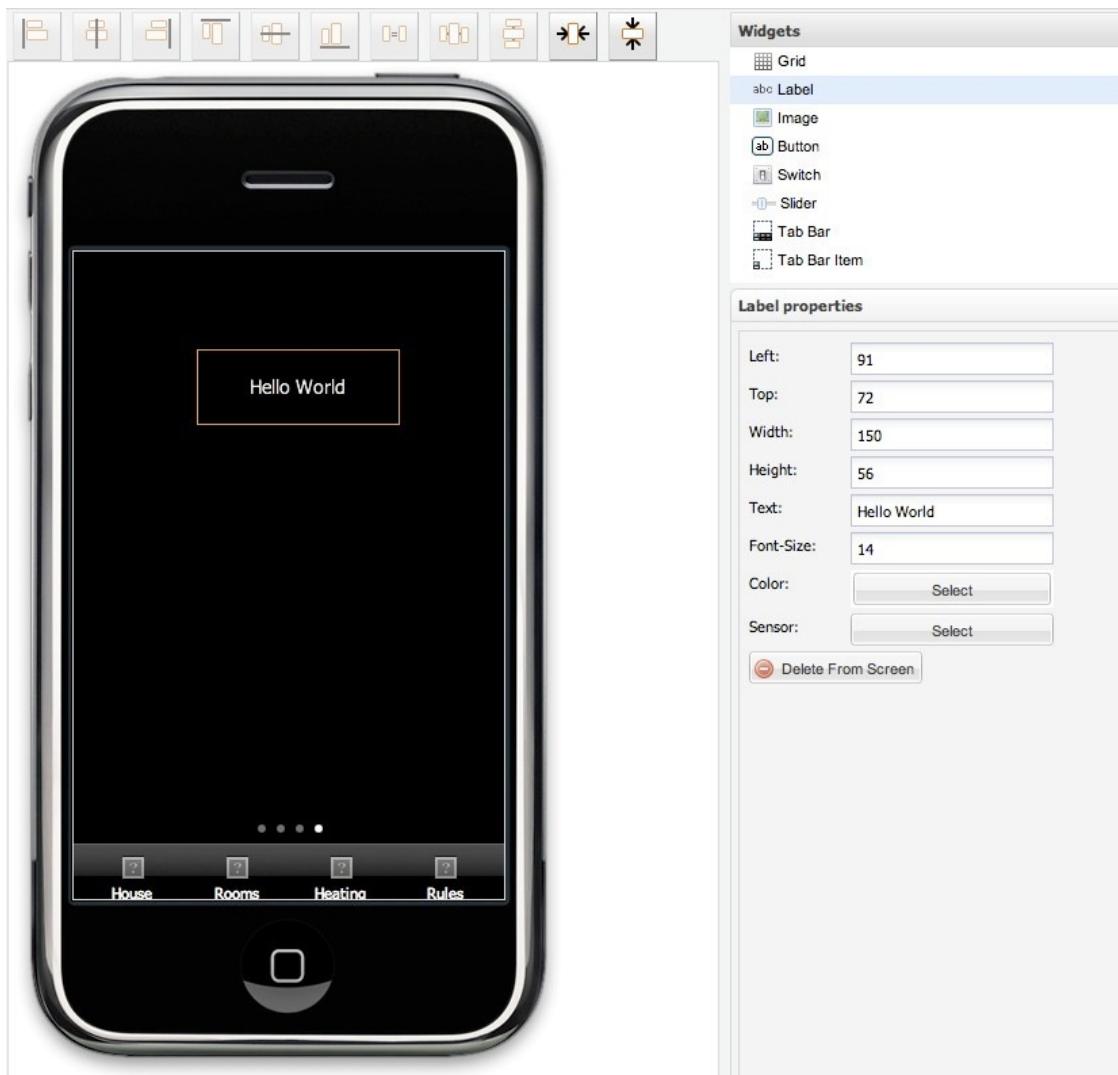


Figure 9.1 Chapter 10: Building a custom smartphone app with OpenRemote

In chapter eleven we will configure our first sensor and connect it to the smart home application. Specifically, we will be polling weather condition and temperature data for a specific location from the popular Weather Underground Internet service, displaying it with our smartphone app (Figure 9.2).

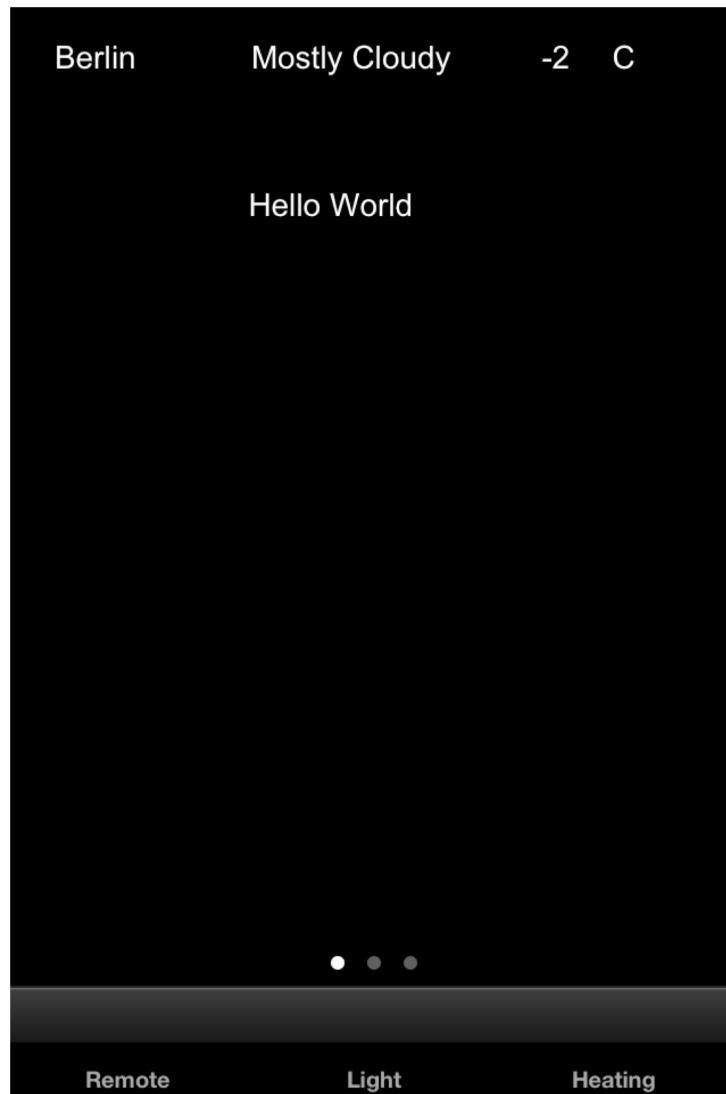


Figure 9.2 Chapter 11: Retrieving and displaying weather information (top of the screen)

In chapter twelve we integrate multimedia control functionality to our smart home project. We will set up our OpenRemote smartphone app to function as a remote control for Apple's multimedia suite iTunes (on both Macs and PCs) and Microsoft's MediaPlayer under Windows 10 (Figure 9.3).

In chapter thirteen we introduce the rule based automation capabilities of

OpenRemote. 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 or snows". 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. Another scenario will be "Welcome Home", which uses a smartphone triggered geofencing feature to start playing the iTunes playlist of choice for the person returning home. We will even give our smart home a voice and have it read reminders and appointments for the day to us via its Hi-Fi stereo.

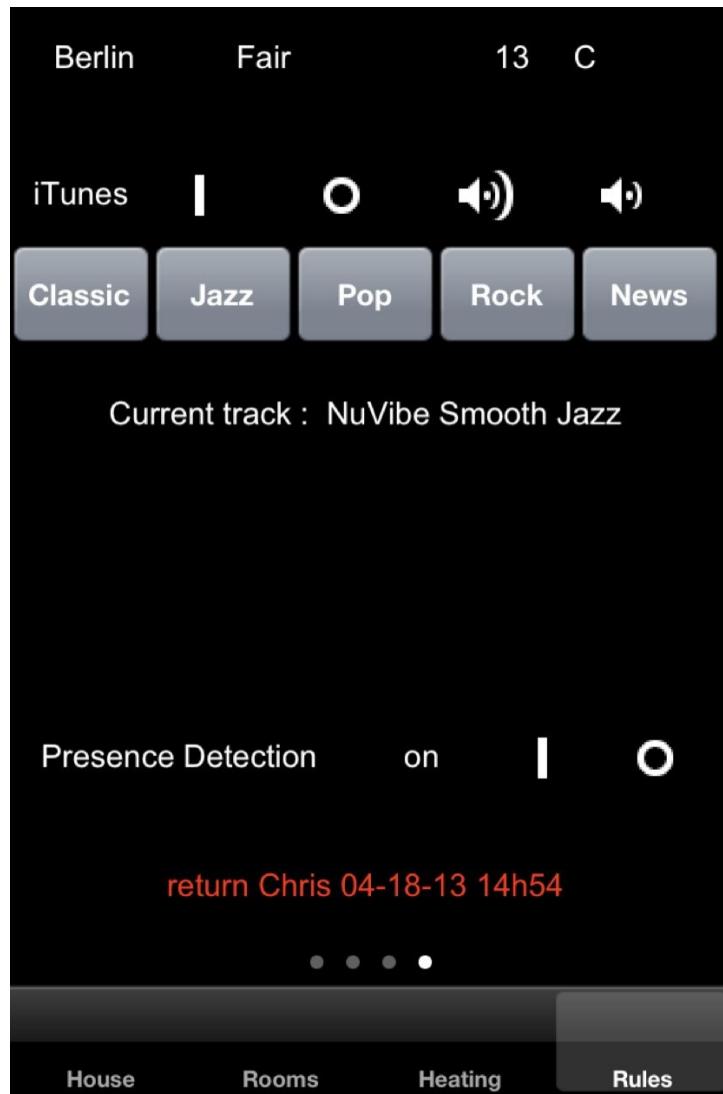


Figure 9.3 Chapters 11,12,13: Adding controls for iTunes and a Hi Fi stereo system

Up to that phase all you need for following and implementing the project is a Mac or PC, a WiFi network and an Internet capable smartphone. For some of the functionalities in the later chapters you will need the components you have chosen to use for your smart home, such as Z-Wave or KNX sensors or actuators. Even if you use different products than described in our project, the majority of the descriptions will still be valid for your individual implementation⁷⁸.

In chapter fourteen, we add wireless light and power outlet control to our project, using the popular Z-Wave standard. In addition, we accomplish the integration of consumer electronics hardware using a Denon audio video receiver (AVR 3313) as an example (Figure 9.3).

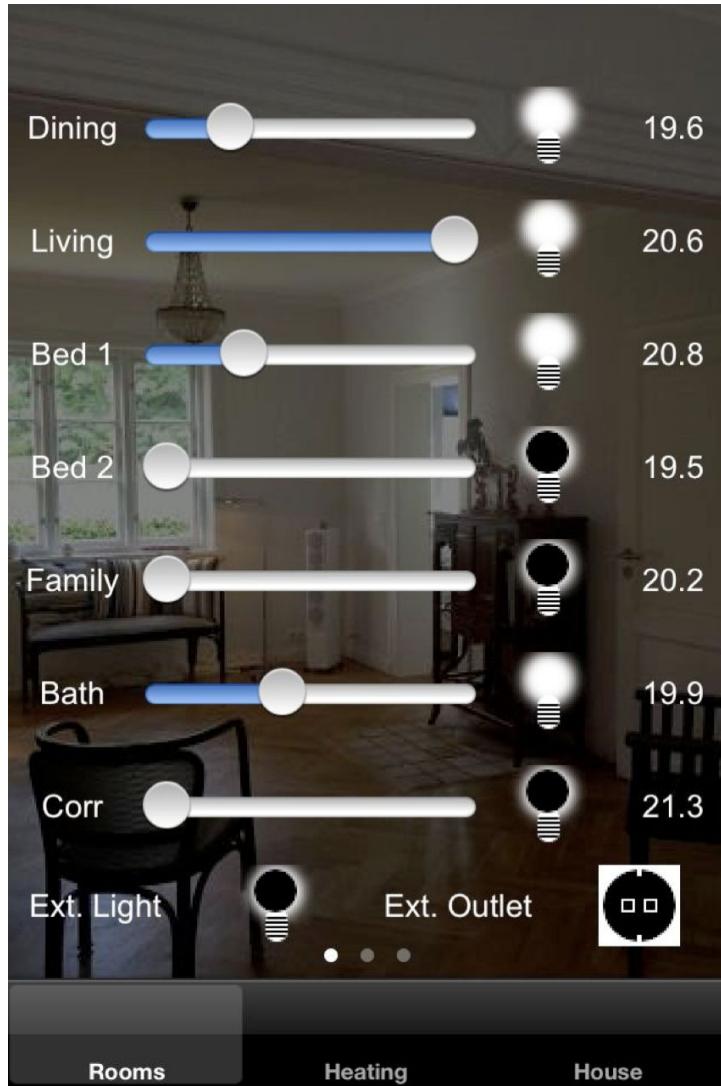


Figure 9.4 Chapter 16: Controlling lights and power outlets

Chapter fifteen adds presence control and geofencing capabilities. We will integrate our solution with the cloud services IFTTT and Google Docs, enabling our home controller to detect iOS and Android smartphones leaving or approaching the building (Figure 9.3, lower part of the screen).

In chapters sixteen and seventeen we integrate KNX based infrastructure components for heating and lighting. In a step-by-step fashion, we explore how

to download, install and configure ETS, the official KNX association control software. Then, we fully integrate the KNX controls with our OpenRemote project (Figure 9.5).

Manufacturer	Name	Description	Product	Order Number	Medium Type	App
Albrecht Jung	Kompakt Rau Kompakt-F 4093KRM	Kompakt Rau Kompakt-F 4093KRM	TP	4093KRM	Kom	
Albrecht Jung	Kompakt Rau Kompakt-F 4093KRM	Kompakt Rau Kompakt-F 4093KRM	TP	4093KRM	Kom	
Merten	Binary in 644592	Binary in 644592	TP	644592	Mult	
Merten	Binary in 644792	Binary in 644792	TP	644792	Mult	
Merten	Binary in 644992	Binary in 644992	TP	644992	Mult	
Merten	Binary in 6398 98	Binary in 6398 98	TP	6398 98	Univ	
Merten	Blind act 649804	Blind act 649804	TP	649804	Shut	
Merten	Blind act 648704	Blind act 648704	TP	648704	Shut	
Merten	Blind/Swi 649912	Blind/Swi 649912	TP	649912	Blind	
Merten	Blind/Swi 649908	Blind/Swi 649908	TP	649908	Blind	
Merten	Bus coup 671299	Bus coup 671299	TP	671299	Dim,	
Merten	Bus coup 671298	Bus coup 671298	TP	671298	Dim,	
Merten	Control un 646991	Control un 646991	TP	646991	Univ	
Merten	Data rail 6806 02	Data rail cc 6806 02	TP	6806 02		
Merten	Dummy Dummy	Dummy Dummy	TP	Dummy	Dum	
Merten	Dummy Dummy	Dummy Dummy	TP	Dummy	Dum	
Merten	EMO valv 6391 18	EMO valve 6391 18	TP	6391 18	Valv	
Merten	INSTABU 6801 29	INSTABUS 6801 29	TP	6801 29	Dalit	
Merten	INSTABU 625199	INSTABUS 625199	TP	625199	Swit	
Merten	INSTABU 625299	INSTABUS 625299	TP	625299	Swit	
Merten	KNX / IP- 680329	KNX / IP-R 680329	TP	680329	KNX	

Figure 9.5 Chapters 16,17: Adding KNX control

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. In chapter eighteen we add the capabilities to access all control functions from remote via the Internet, in chapter nineteen we build the functionality to automatically restore operation after a restart of the controller. Chapters twenty and twentyone cover system test and advanced data reporting.

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

- smartphone / tablet based display of weather and temperature
- 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
- operation of an audio reminder system with text-to-voice conversion of calendar items
- 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
- iOS / Android location based presence detection and geofencing using IFTTT and Google Docs
- integration with other smart home platforms and intelligent devices via IFTTT
- VPN based remote control from outside of the house
- Automated restoration of operation after power outages or planned downtime

Sensoring approaches such as geofencing based on smartphones or weather condition information retrieved from the Internet provide just a glimpse of what state of the art home automation based on open standards is capable of delivering. Using the functionality of our project as a start, a vast variety of variations and add-ons can easily be implemented.

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 macOS or a Windows PC. While the functionality of the project in this book can also be realized with older operating system variants, the step by step descriptions are based on macOS 10.12 (Sierra) and Windows 10.
- 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 OpenRemote, which we use throughout this book supports literally all major building control

... are analogous and soon, supports many an major command control standards. (Table 10.1).

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 Home Control Centre: Open Remote

We will start our project with the installation and configuration of OpenRemote. OpenRemote is a state of the art open source software platform for building automation and device control. It has been used for smart building 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. Still, the platform will allow you to build a fully custom, professional building control system, including a smartphone app, which will serve as the mobile control centre. The “always on” OpenRemote controller, which is supported on macOS, Linux, Windows and other platforms, will run the automation rules for our project.

All OpenRemote components with their full functionality are available free for private use, educational purposes and trials. Commercial users pay a one time fee of 150 US\$. A former differentiation between a free and a pay for version was removed in early 2016. Since then, all users - private, educational and commercial - use the identical software.

10.1 OpenRemote Overview

The OpenRemote platform consists of three software components:

- The OpenRemote Controller, an always-on (24/7) Linux, Windows or macOS server application, which connects the mobile control devices (smartphones, tablets) to building automation systems and devices under control. Control devices can be building infrastructure (light switches, power outlets etc.), consumer electronic devices, or home appliances. The OpenRemote Controller can also run scripts, which are called rules. These rules are automation sequences, which are implemented based on the open Drools event processing language.
- The second component consists of the OpenRemote mobile clients for iOS or Android. Graphical user interface and functionality of the OpenRemote App can be fully customized using the third component of OpenRemote, the OpenRemote Professional Designer.
- OpenRemote Professional Designer is an online, cloud based application, providing a graphical user interface for designing the mobile client interface and implementing the related commands, sensors, and switches. Once user interface and the associated control functions have been designed, the Professional Designer configuration files are synchronized with the local controller installation. The smartphone client application is updated automatically, when connecting to the controller, immediately reflecting changes or updates made within the online Professional Designer project.

OpenRemote supports a large variety of building automation protocol standards. In addition, it provides API's for the customization and extension of its capabilities. The current software release OpenRemote Controller 2.6 supports the following control protocols (Table 10.1).

Control Protocol	Description
1-Wire Protocol	Low data rate communication bus for Maxim Integrated Products.

	www.maximintegrated.com
AMX Controller	AMX Inc. proprietary device control protocol.
DateTime Protocol	Display of date and time, including sunrise/sunset calculation.
Denon Serial AVR Protocol	Protocol to control Denon <i>Marantz audio</i> video/ devices.
Domintell	Protocol for Domintell building control infrastructure.
DSC IT-100 Serial Protocol	Protocol for DSC (Digital Security Controls) systems.
EnOcean	Energy harvesting wireless technology for device control ISO/IEC 14543-3-10. www.enocean.com
GlobalCache	Infrared control devices by specialist GlobalCache. www.globalcache.com
HSC Z-WAVE IP Gateway	Honeywell Z-Wave Gateway.
HTTP	Hypertext Transfer Protocol
HTTP/MJPEG	Motion JPEG Video Streaming
HTTP/REST	Representational State Transfer. Usage of HTTP for stateless client-server communication
Insteon	Home automation system based on power line and radio frequency (RF). www.smartlabsinc.com
Integrated Control Technology (ICT)	Protocol for the Protege Suite, an enterprise level building automation system
ISY-99	Control protocol for Universal Devices home automation solution.
Keene IR Anywhere Protocol	Proprietary protocol to transmit Infrared (IR) Control Commands over IP or RS232
KNX	International standard for industry grade wireline home automation. www.knx.org
Lutron HomeWorks	Protocol for Lutron building control infrastructure.
Open Webnet	Remote control protocol for Legrand My Home systems
panStamp Lagarto	Open source protocol for PanStamp wireless modules. www.panstamp.com
Philips Hue Protocol	HTTP/REST API to communicate with Philips Hue bridge devices
Russound RNET Protocol	Protocol for distributed audio and video solutions from Russound.
Samsung TV Remote Protocol	Protocol used to control Samsung TV systems.
Shell execution protocol	Execution of shell scripts.
Shell execution protocol	Execution of shell scripts.
TCP/IP	Transmission Control Protocol

Telnet	Telnet Protocol
UDP	User Datagram Protocol
Velbus	Protocol for Velbus home automation products www.velbus.eu
Wake-On-Lan Protocol	Protocol activating networked systems in power save mode.
X10	Legacy standard for power line based home automation.
XBMC	Open source media player platform. xbmc.org
xPL,IRTrans, VLC, FreeBox, MythTV	Commercial and OpenSource home automation solutions.
Z-Wave	Wireless communication protocol optimized for home automation. www.z-wavealliance.org

Table 10.1 Communication protocols and automation standards supported by OpenRemote Controller 2.6

With its intuitive user interface, OpenRemote allows for designing a fully customizable building and home control solution without the need to actually write code. This is not to say that home automation is becoming as easy as an off the shelf software installation. With components from different vendors having to play together, there will often be the need for iterations of testing, troubleshooting, and fine-tuning. However, with OpenRemote, we have a powerful platform at hand, which allows for professional results and comprehensive functionality. In addition, the large and helpful OpenRemote user community of building automation professionals and home automation hobbyists provide help and support.

10.2 OpenRemote Controller Installation

Getting OpenRemote downloaded, installed and running should take less than one hour. On www.keyconceptpress.com/tutorials you can also watch the video tutorials for macOS and Windows 10, which demonstrate the installation step by step as described below.

First we need to register for an OpenRemote account. Open <https://designer.openremote.com/login.jsp> and select *Don't have an account?*, then *Order Now*. Fill out the fields for email and address. If you are a private or educational user, or if you want to trial the software as a company, on the right hand side of the window, under *Order Summary*, select *Add Coupon* and enter PRIVUNIV. Then select *APPLY*, which reduces your order total to 0 US\$, then select *SUBMIT YOUR ORDER* (Figure 5.1).

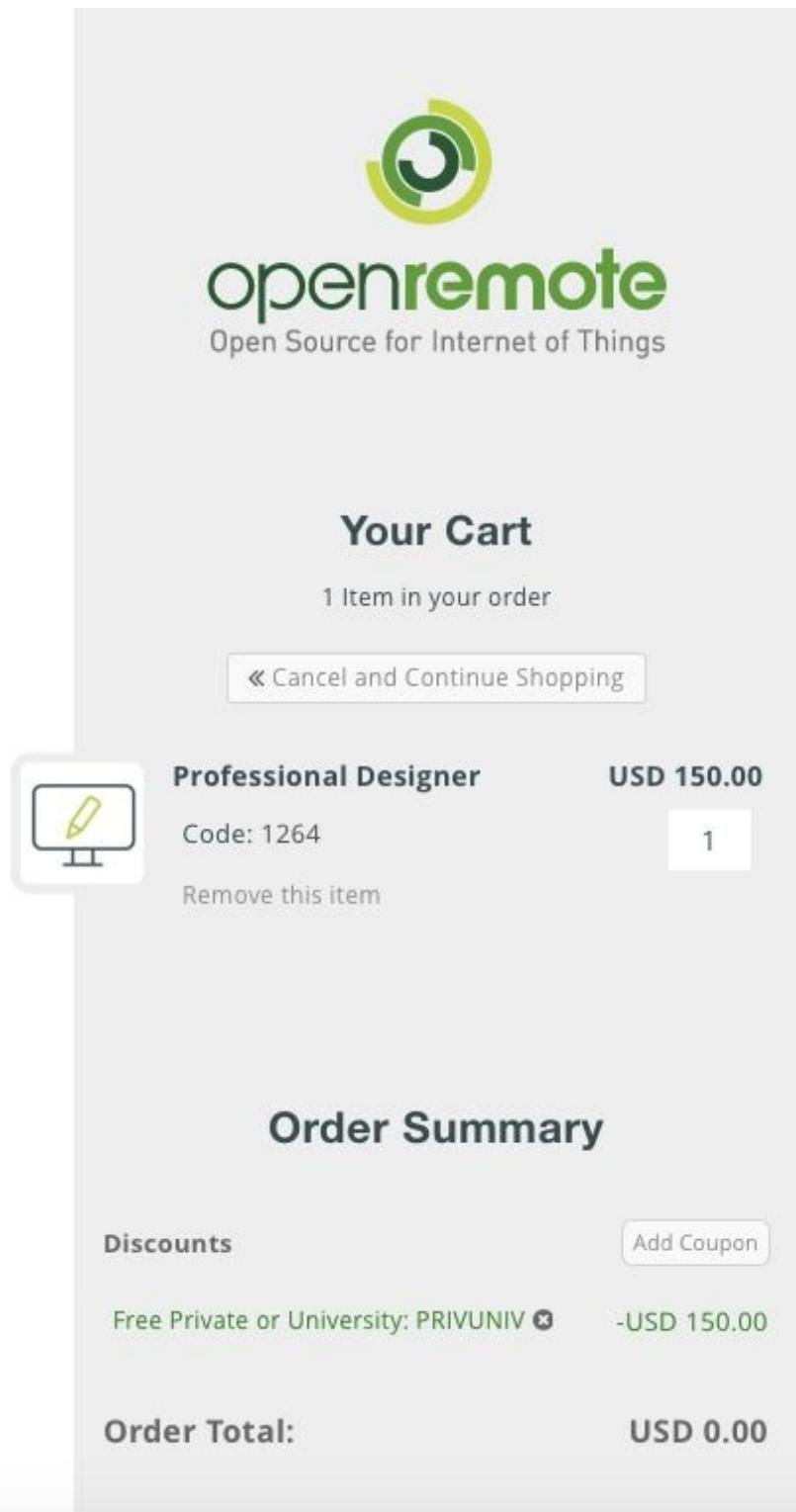


Figure 10.1 Setting up a free account for OpenRemote Professional Designer

With the user name and password, which you will receive in an email after registration, you log in to your new account at <https://designer.openremote.com/login.jsp>. A window with the OpenRemote Professional Designer GUI will open. Now you can install OpenRemote Controller. Select the *Download Resources* button at the upper right corner of the user interface, and a GitHub window with the latest binary and source code files for OpenRemote Controller will open (Figure 10.2). In the section of the most recent release (at the time of this writing 2.6) you go to *Downloads* and select the file OpenRemote_Controller.zip.

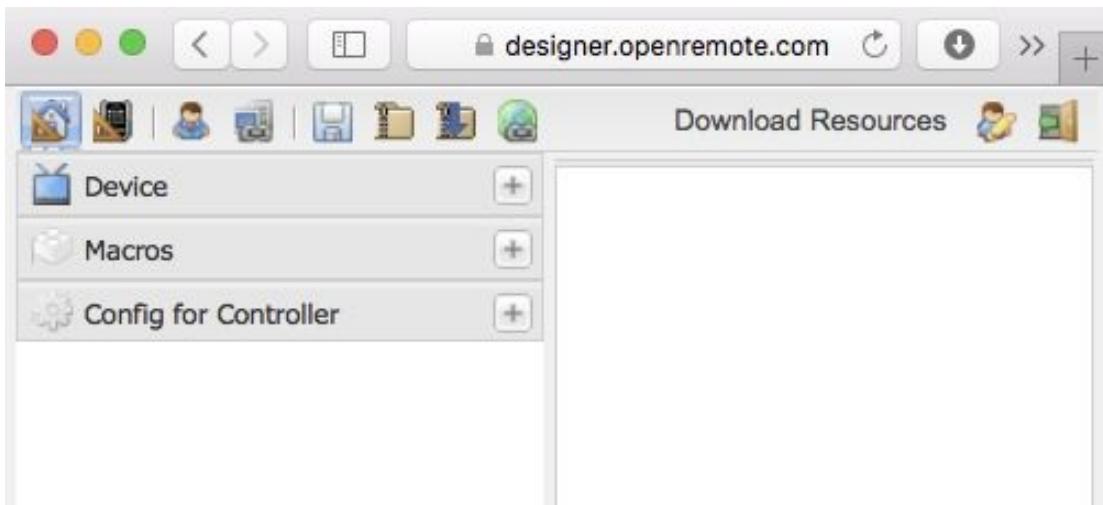


Figure 10.2 The OpenRemote Professional Designer GUI with the Download Resources button

After downloading the compressed file, which has a size of around 30 Mbytes, move it to a local project directory (e.g. shproject) under your home directory and extract it. You will now see the directory tree of the software with the startup files openremote.sh (macOS, Linux) and openremote.bat (MS Windows) in the /bin directory (Figure 10.3).

Under Windows, as well as under macOS, the home directory is the one you

are in when opening a terminal window. To make moving around directories easier, you should rename the top level OpenRemote directory, which is called something like OpenRemote-Controller-2.6.0 to something simpler such as ORC260.

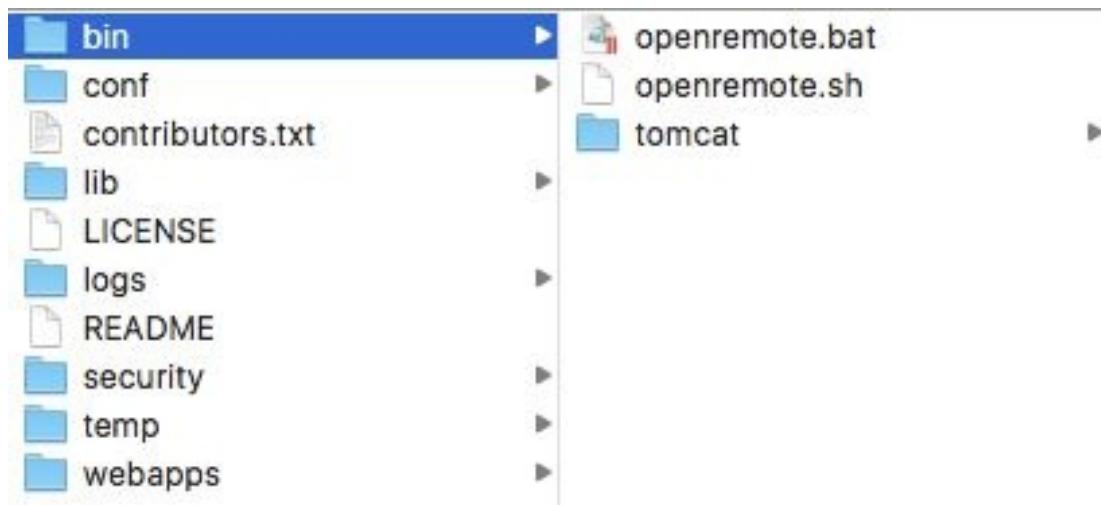


Figure 10.3 The OpenRemote Professional Designer directory tree

Since OpenRemote Controller is a Java application, before you are able to start it for the first time, you need to install Java on your computer. I will cover this in separate sections for macOS and Windows. If you are working under Windows move on to section 10.4 now.

10.3 Java Installation and Configuration under macOS

On a Mac you start by opening a macOS Terminal window by selecting [Applications – Utilities](#). If you have not worked with the terminal application before, there are a few commands you need to know in order to survive at the beginning:

macOS Command	Description
ls -l	display listing with the option - l (l for long), displays the content of the current directory, including hidden files and permissions
ls -a	display listing with the option -a also shows hidden files (e.g. all files starting with a period such as .bash profile are hidden)
pwd	print working directory - displays the path of the current directory
cd ..	change directory followed by a space and two dots - gets us one directory hierarchy up
cd	just typing cd gets you back to your home directory
cd /target	changes to the specified directory
mkdir name	create (make) directory
man mdc	show the manual entry for a command

Table 10.1 Basic macOS Terminal commands

First you need to verify if you have Java installed. To do this use the command
java -version

On a new Mac, you will not find Java installed, because since version 10.8 it is no more part of macOS. In response to the above command, macOS will alert you that you need to download Java in order to execute this command. If you select [more information](#) in the alert window, you will be taken to the Oracle Java website. There you select [JDK Download](#), and after accepting the Oracle License Agreement you can download and install the JDK version for macOS. While you will not conduct Java development work, just installing the Java Runtime Environment (JRE) is not sufficient for the integrated rules environment Drools. So select the JDK (Java Development Kit) package for

installation. After the successful Java installation 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 you have installed is being displayed (Figure 10.4).

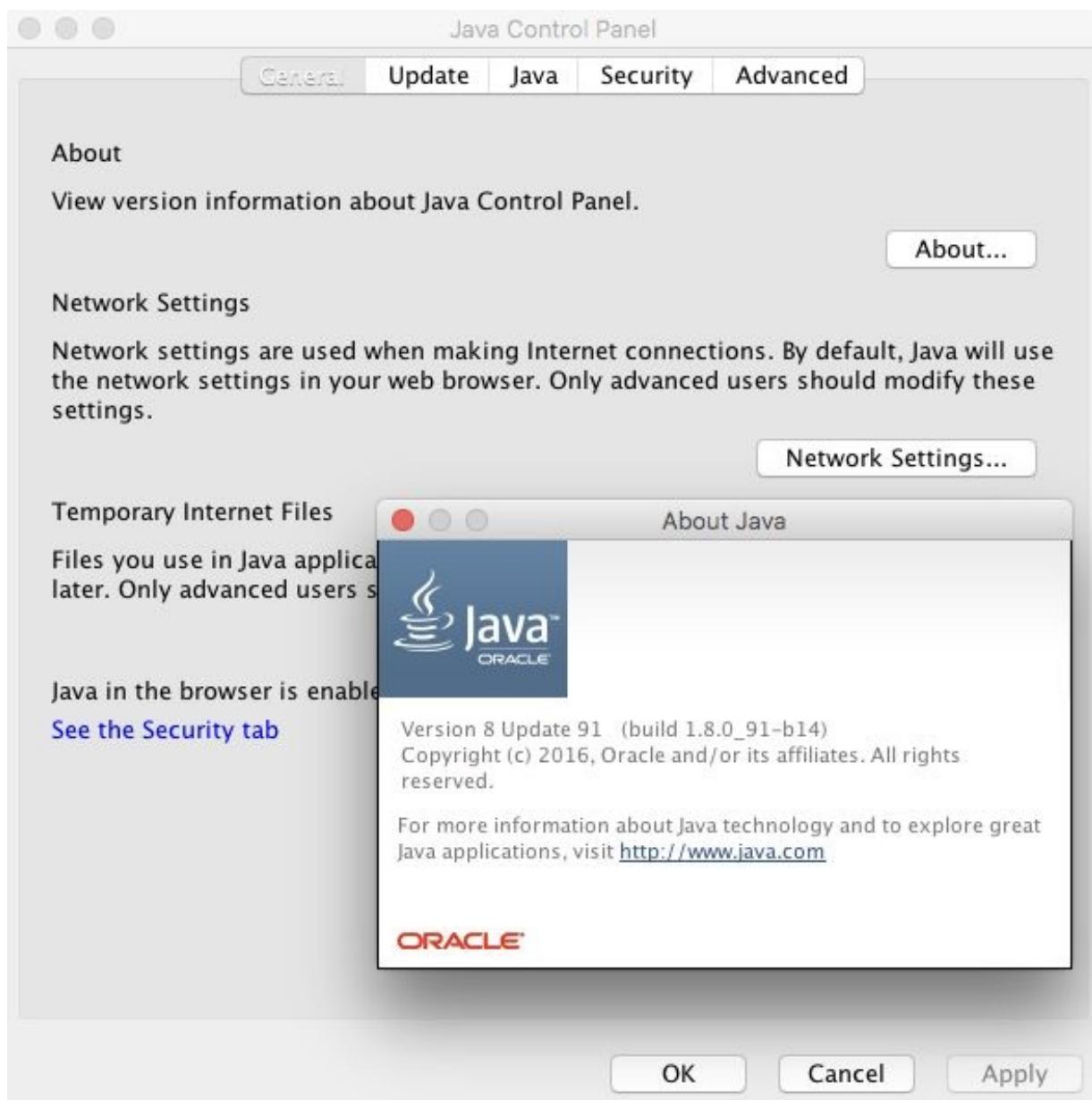


Figure 10.4 The Java Control Panel on macOS

If you are using Safari as your browser, make sure to also enable Java in Safari by selecting *Preferences* and *Security*. Open *Plug-in Settings...* and add Java to the list of approved Plug-ins. You will now be able to verify your installed Java version with Safari using the link
<https://www.java.com/en/download/install.jsp>.

Apple also offers a legacy Java version (Java 1.6 <http://support.apple.com/kb/DL1572>) for download from its support website, which was the last Java version, officially supported by Apple. There are still some older macOS applications out, which do require this legacy version, however in most cases going with the latest Java version from Oracle is the best choice, which today is also being recommended by Apple on its Java [support page](#). Since controller version 2.6 and higher the Open Remote software includes the latest Drools version (6.4) and is fully compatible with the latest Java release, which at the time of this writing is 1.8.

You can also verify your Java installation using the `java -version` command. The JDK will then report the version number of its integrated JRE environment: `java version "1.8.0_25"`

Java(TM) SE Runtime Environment (build 1.8.0_25-b17) Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode) 10.3.1 Setting \$JAVA_HOME on a Mac

There is one more thing we need to do before we can start the OpenRemote controller, which is setting the \$JAVA_HOME variable. \$JAVA_HOME is used by Java programs to find the path of the Java files and needs to contain the full path to the Java installation.

Under macOS and Linux, the list of locations or paths, which a program uses to search for executables, is stored in the \$PATH variable. There is a system wide

and a user specific \$PATH definition. We will just use the user specific \$PATH variable at this point. The user specific \$PATH definitions under macOS are contained in the file .bash_profile file in the user home directory. The (hidden) file .bash_profile might not exist yet in your home directory, which is why you probably will need to create it. To list the hidden files in your home directory use the command ls -a. Then enter the following two commands in the terminal window: touch ~/.bash_profile

open ~/.bash_profile

You can enter the tilde character (~) by typing [alt N](#). The command touch along with a filename creates an empty file and the command open plus a filename opens the specified file in the default text editor, which on a Mac isTextEdit. Now you need to add the line that sets \$JAVA_HOME to contain the directory of your JDK installation. The best way to do this is to use the command java_home, which automatically returns a path suitable for the \$JAVA_HOME environment variable. The command can be found in the directory *usr/libexec/*. If you open a Terminal window, change to the directory *usr/libexec/* and enter the command ./java_home, you will get something like:

LibraryJava/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home With that the entry for the file .bash_profile using java_home looks like the following: export JAVA_HOME=\$(*usr/libexec/java_home*) You enter the above line and save .bash_profile in TextEdit, close the terminal window and open it up again (which forces the system to process the new \$PATH definition), and test your work by entering echo \$PATH and echo \$JAVA_HOME.

\$JAVA_HOME

-bash: *LibraryJava/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home*: is a directory 10.3.2 Starting OpenRemote Controller for the First Time The macOS start script for OpenRemote Controller is openremote.sh in *shProjectORC260/bin/*. When we do a long listing (*ls -l*) of the files in the ORC260/bin/ directory we can see that we do not have the execution right for

the file yet. In case you are new to file permissions: File permissions in macOS and Linux are set in three groups (owner, group, everyone) with three symbols for each group receiving permissions. The symbols can contain:

Permission Code	Description
-	no permission
r	read permission
w	write permission
x	execute permission

Table 10.2 File permissions in macOS

The first letter of the file listing is not a permission, but shows whether the line entry references a file (-) or a directory (d). So the permissions start actually at the second letter of the below listing: ls -l .shProjectORC260/bin

total 48

-rw-r--r--@ 9854 10 Mar 2016 openremote.bat -rw-r--r--@ 12039 10 Mar 2016
openremote.sh We see that we have only read and write rights for openremote.sh.
With the command chmod +x we can set the execution rights: chmod +x
.shProjectORC260/bin/openremote.sh and can now start the OpenRemote controller by
entering: ./openremote.sh run

The dot slash sequence references to the current directory in Unix type systems.
For security reasons it is not part of the \$PATH definition. So for commands to
be executed in the current directory you always have to start with dot slash (./).

In the terminal window, you will now see a lot of text running by, which
eventually will stop, displaying something like below: -----

DEPLOYING NEW CONTROLLER RUNTIME...

Dec 9, 2016 11:03:33 PM org.restlet.engine.connector.HttpClientHelper start
INFO: Starting the internal HTTP client

INFO 2016-12-09 23:03:34,163 :

OpenRemote Z-Wave protocol version : '3.0.0'

INFO 2016-12-09 23:03:34,229 : No rule definitions found in
'UserssmarthomeshProjectORC260/webapps/controller/rules'.

INFO 2016-12-09 23:03:34,229 : Started event processor : Drools Rule Engine
INFO 2016-12-09 23:03:34,413 : Started event processor : RRD4J Data Logger
INFO 2016-12-09 23:03:34,570 : Startup complete.

Your OpenRemote Controller is now up and running for the first time.

10.3.3 Curly or straight, this is the question!

Finally, before you get started with scripting or entering terminal commands, an important hint. Quotation marks in scripts and terminal commands have an important function. However, the quotes used need to be the straight quotation marks, rather than the curly quotation marks, which are used by word processor software. It can happen, that when you copy a command line from a text document or when you edit a shell script or an AppleScript usingTextEdit or another word processor, that the straight quotes are converted to curly quotes, resulting in runtime errors, which at first sight are hard to detect. So always take a close look at the quotation marks. I will repeat this hint several more times throughout the book, since it is a common problem for people not so close to software development, working along a book. Mac users can now move on to section 10.5.

10.4 Java Installation and Configuration under Windows

Also under Windows (XP/7/8/10), before starting OpenRemote, you need to make sure you have a Java Development Kit (JDK) package installed. To find out if you have an installation on your computer, under Windows 10 select *Start – Control Panel – Add or Remove Programs*. If you do not have Java installed, you need to download the JDK package from the Oracle Java website. Next go to the Windows menu for environment variables. Open *System* in *Control Panel*. (or simply type *advanced system* in the search bar). Open the *Advanced* tab, select *Environment Variables*. Select *New* to add a new variable name and value. Enter JAVA_HOME (in capital letters, exactly as shown) as the variable name. For the path definition select *Browse* and locate the path of your JDK installation, which is located just below the Java directory. Be very careful to make no mistakes when setting the environment variable. The controller will not start up if the environment variable does not point to the correct directory. With the command set from a terminal window you can validate the settings of your environment variable: set

```
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_45
```

To open a terminal window under Windows 10 simply type *command* in the search bar. Now you can start up the controller for the first time. In the terminal window change to the directory *shProjectORC260/bin/* and enter *openremote.bat run* You will now see a lot of text running by, which eventually will stop displaying something like: ...

...

```
INFO 2016-12-09 23:03:34,570 : Startup complete.
```

```
INFO 2016-12-09 23:03:34,572 : Controller Definition File Watcher for Default
```

Deployer started.

Dec 9, 2016 11:03:35 PM org.apache.coyote.http11.Http11Protocol start INFO:
Starting Coyote HTTP/1.1 on http-8688

Dec 9, 2016 11:03:35 PM org.apache.catalina.startup.Catalina start INFO:
Server startup in 3829 ms

Your OpenRemote Controller is now up and running for the first time.

10.5 First Synchronization between Designer and Controller

Now you need to validate your installation and synchronize the local controller installation with your online Designer account for the first time. First access the controller web interface by opening the URL <http://controller-machine-IP:8688/controller> in an Internet browser window. For the part of this URL, which says controller-machine-IP, you insert the IP address of the hardware, your controller is running on. If you run the controller on your local machine, the IP address you need to use is simply 127.0.0.1 or localhost, in which case you open the URL <http://localhost:8688/controller>.

The screenshot shows the 'Welcome to OpenRemote Controller' page. At the top, there's a logo for 'openremote' with the tagline 'Open Source for Internet of Things'. Below the logo, the text 'Welcome to OpenRemote Controller' is displayed. A message instructs the user to link the controller to their designer account, providing the MAC address: '60-33-4B-0B-30-18'. It also suggests using 'Controller management' in 'OpenRemote designer' to create a link. A green box at the top indicates 'Sync Complete.' Below this, there are buttons for 'Reload configuration and clear cache' and 'configuration update' with radio buttons for 'online' (selected) and 'offline'. A note states that this requires a 'Designer account'. There are input fields for 'username' (containing 'smarthome.test') and 'password' (containing '*****'). A button labeled 'Sync with Online Designer' is at the bottom, along with a link for users without an account.

Welcome to OpenRemote Controller

Please link this controller to your designer account.
Your mac address to use is: 60-33-4B-0B-30-18

Use "Controller management" in [OpenRemote designer](#) to create link.

Sync Complete.

Reload configuration and clear cache

configuration update : online offline

This requires your **Designer** account.

username : smarthome.test

password : *****

Sync with Online Designer

Don't have OpenRemote Designer account? [Create account now!](#)

Figure 10.5 The OpenRemote Professional Designer controller synchronization window

Now the OpenRemote Professional Designer controller synchronization window will open, showing the MAC address of your controller hardware in the upper half of the window. Ignore the message *Use Controller management in OpenRemote designer to create link*. Controller management will be activated in a later release of OpenRemote Controller and provide auto detection of the controller IP address. For now directly proceed by entering your Professional Designer login details, and select the button *Sync with Online Designer* (Figure 10.5). A few seconds later the synchronization window will show the message *Sync complete* (Figure 10.6).



Figure 10.6 Successful synchronization between an OpenRemote Professional Designer Account and a local controller

10.6 The Importance of Directory and File Management

A last advice before we move on. I recommend to place all custom scripts and files, which you develop throughout this project, in a dedicated directory within the root directory of your project (in our case the directory shProject), rather than in the OpenRemote directory (in our case ORC260). Otherwise you will have to manually move your custom scripts and files from your old to your new OpenRemote installation in case of a software upgrade, which then might become a source of problems. Following this guideline we create the folder shConfig within shProject. It will contain all scripts and configuration files for our smart home project. When we now move our installation or upgrade the OpenRemote software, we just need to make sure, that the folder shConfig remains in the same directory as our OpenRemote folder.

10.7 OpenRemote Professional Designer

We now will take a closer look at our OpenRemote Professional Designer account. We open <https://designer.openremote.com/login.jsp> and log into our Designer account. The Designer GUI consists of two main elements. The Building Modeler and the UI Designer. The Building Modeler (Figure 10.7) is used for defining commands, sensors, switches and sliders, and for selecting and configuring the associated protocols. In the UI Designer, we create the graphical user interface for our smartphone or tablet control app, linking its functional elements to the commands in the Building Modeler. (Figure 10.8).



Figure 10.7 Selecting the Building Modeler

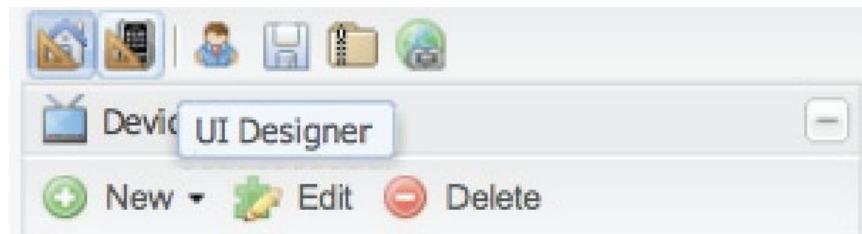


Figure. 10.8 Selecting the UI Designer

10.8 The “Hello World” App

As a first step and in order to understand the workflow of OpenRemote we want to display “Hello World” on our smartphone or tablet. For that in the UI designer window we click on *New –New Panel*, select our mobile device (Android, iPhone, iPad) and enter our project name (e.g. shome), as the name under panel property on the right hand menu bar. The default name for the screen, which is Starting Screen, we rename to Remote. Next, from the right hand menu, we drag an *abc label* element onto the smartphone panel and enter Hello World in it’s text field on the right hand side of the GUI. As the last step, we click on the save button in the upper left corner. Our OpenRemote UI Designer screen should now look similar to the one in Figure 10.9. Make sure you actually hit the save button. Also look for the Designer confirmation messsage, which confirms that all changes have been saved. It pops up in a small window at the bottom right of the GUI after each save command. With no available panel design the controller will not be detected by the OpenRemote smartphoen app, which is the last step in our installation procedure.

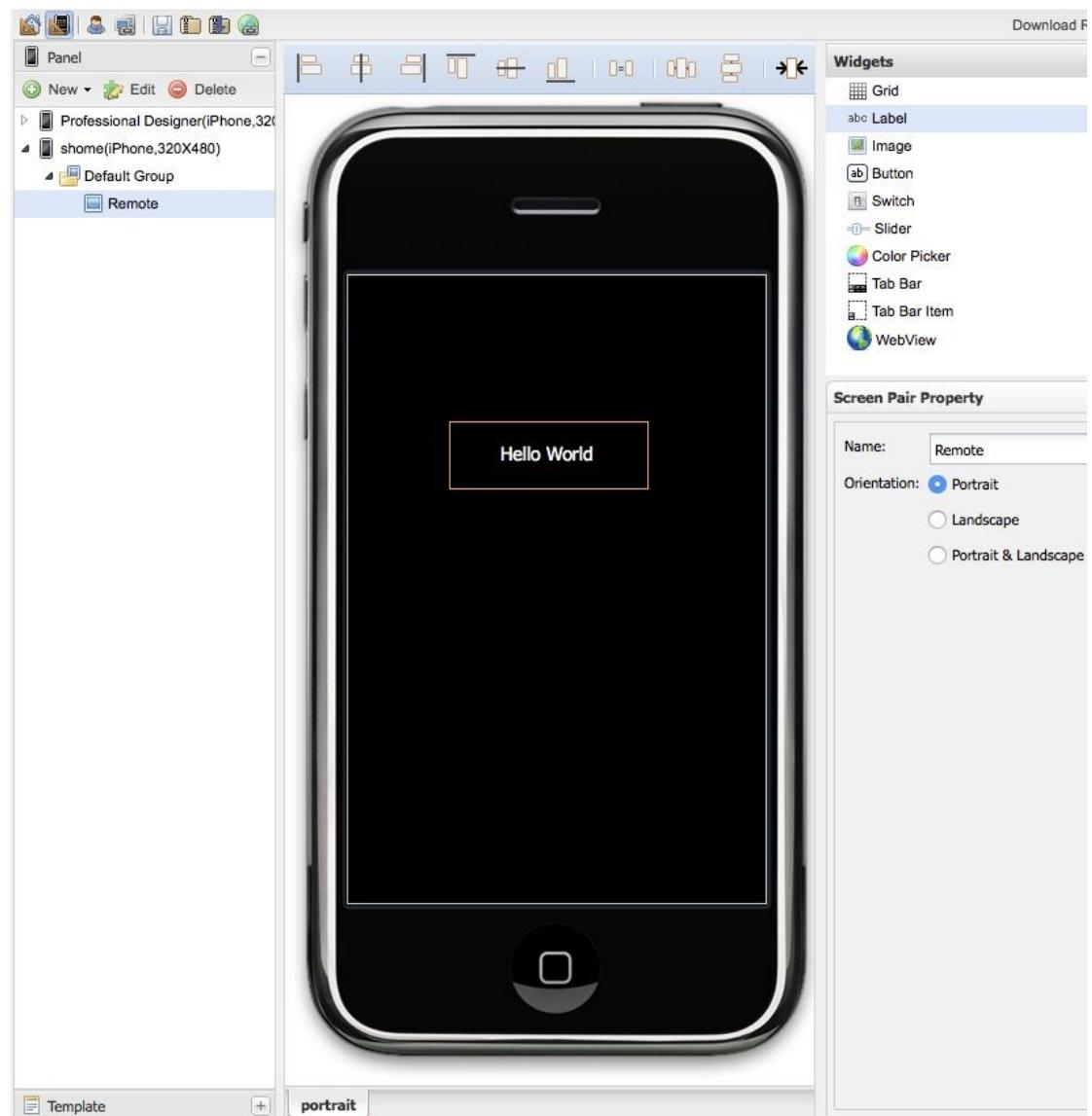


Figure 10.9 First steps with OpenRemote Professional Designer

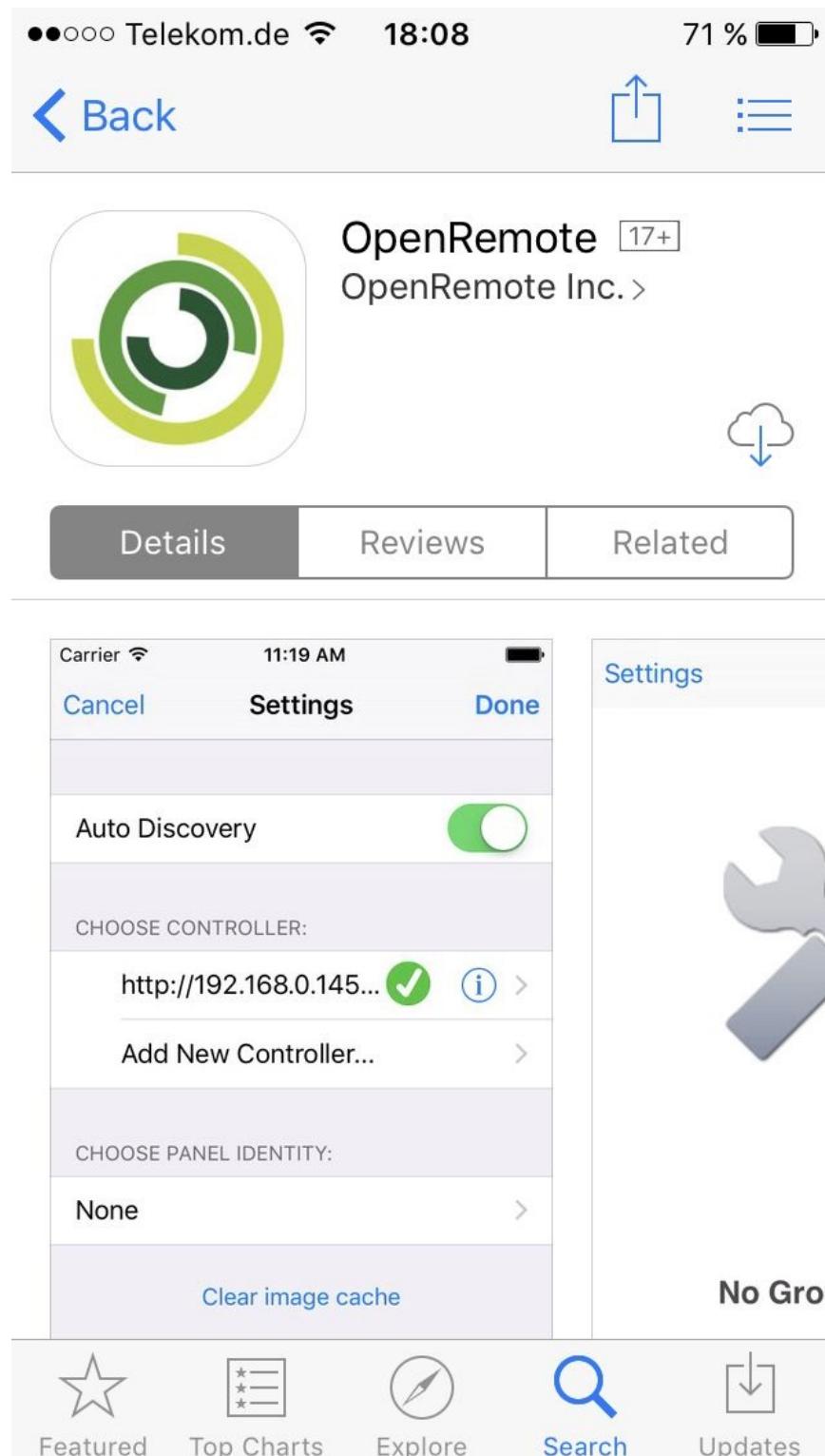


Figure 10.10 Downloading the OpenRemote client app for iOS

Now we download the mobile client app, the OpenRemote Panel, onto our tablet or smartphone. Do not get confused by multiple OpenRemote client versions. You need the one which is described as: “Universal version of the OpenRemote 2 console” (Figure 10.10). When you start the OpenRemote app for the first time, it opens with a configuration screen. Later, access to the configuration screen is gained by shaking the phone (Figure 10.11). For the time being leave Auto Discovery mode deactivated and manually set the IP address of your computer hosting the controller. To find your Mac’s IP address under macOS it is easiest to open the Network Utility application. To find it, simply enter the term Network Utility in the Mac Spotlight search.

Under Windows you open a terminal window and type

```
ipconfig /all
```

To configure the controller IP address you now enter the IP address of your computer followed by 8688/controller in the configuration screen of your OpenRemote app. The entry should look similar to

```
http://192.168.178.41:8688/controller
```

If you activate the AutoDiscovery mode, within a few seconds the app will automatically detect your controller, and you just need to select it. If AutoDiscovery mode does not detect your controller, you might not have built and saved a valid panel design in Professional Designer, or you might not have synchronized your controller with your Designer account yet. The app does not connect to a controller without any panel identity.

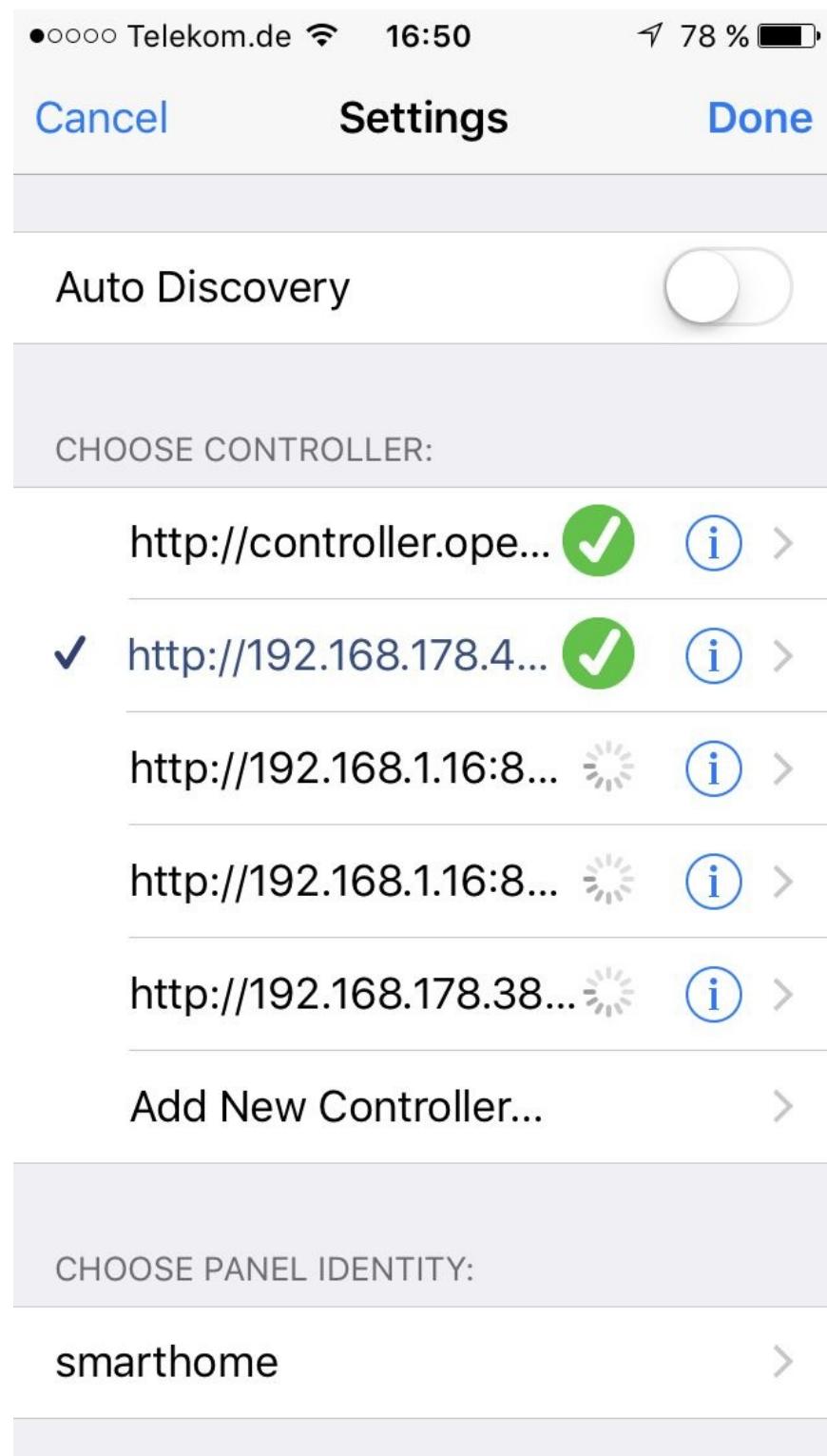


Figure 10.11 The OpenRemote app's configuration screen (shake phone for access)

Next, at the bottom of the configuration screen, select the *Panel Identity* of your Designer layout, which in our case is shome. The app now automatically connects to your local Open Remote Controller and retrieves the application you just designed on your OpenRemote Professional Designer account. Make sure to synchronize your local controller with your OpenRemote Professional Designer account after every change of your Designer project. You should now see the “Hello World” message on your smartphone or tablet app.

In case you see an error message from the client app or the OpenRemote controller such as controller.xml not found or panel.xml not found, don't panic. These files (controller.xml, panel.xml) are only downloaded to your local controller and your smartphone client after the first synchronization with your OpenRemote Professional Designer account. You probably have not synchronized Designer and Controller yet. Make sure to synchronize Designer and Controller again by opening up

<http://localhost:8688/controller>

and clicking on *Sync with Online Designer*. The controller will then connect to your online OpenRemote Professional Designer account and synchronize with the latest changes to your project. The local copy of your design will be updated in the file panel.xml in the directory ..//shProject/ORC260/webapps/controller/. After a restart of your smartphone app, you should finally see “Hello World” on the screen of your smartphone.

Congratulations!

Since you now have successfully installed and configured the home automation control platform for your project, we can get started with the implementation of its first feature

AS A FIRST READING.

11 A Pretty Smart Sensor: Internet Weather

One of the key aspects of OpenRemote is its ability to fully integrate open standard protocols like TCP/IP, HTTP and Telnet. This allows for correlation and interaction of the building infrastructure with devices, that use Internet protocols, such as consumer electronics, home appliances, smartphones or information services. As an example we will design a smart weather sensor for our building automation project, which retrieves weather information from the Internet.

11.1 OpenRemote Control via HTTP: Retrieving Internet Weather Data

Our objective for this section is to retrieve weather and temperature information for a particular geographic location from the popular Weather Underground service (<https://www.wunderground.com>), which is part of The Weather Channel, a subsidiary of IBM. In order to be able to use the API from Weather Underground, we need to register for the free Stratus plan, which allows us to retrieve weather information up to 500 times per day at a maximum rate of 10 calls per minute. The API can be accessed using an HTTP GET command in the following format: GET `http://api.wunderground.com/api/your API key/features/settings/q/query.format` In place of your API key we insert the key ID, which we find after registration by selecting *Weather API – Key Settings*. The format variable in the above statement allows us to choose between json and xml as the format for the weather API response. Since we prefer to work with XML data, we will select XML here.

The query variable contains the location for which we want weather information for in the below nomenclature:

Query Variable	Example
US state/city	CA/San_Francisco
country/city	Germany/Berlin
airport code	KJFK

Finally we replace the variable features with the name for the chosen data set. Below a subset of the offered data sets:

Dataset	Description
astronomy	Returns the moon phase, sunrise and sunset times.
conditions	Returns the current temperature, weather condition, humidity, wind, 'feels like' temperature, barometric pressure, and visibility.

forecast

Returns a summary or the weather for the next 5 days. This includes high and low temperatures, a string text forecast and the conditions.

With that we can build an HTTP command, which retrieves the weather information for Berlin, Germany in XML format:

`http://api.wunderground.com/api/your API`

`key/conditions/q/Germany/Berlin/.xml` (Of course the above link will only work once you have inserted your key ID in place of your API key.) We now go to OpenRemote Designer and create a new device, which we call HTTP. Under this device we define a new command, which we call WeatherBerlin by selecting **New – New command**. As *protocol* we select HTTP, as *URL* we insert our HTTP command for the Weather Underground API from above:

`http://api.wunderground.com/api/your API`

`key/conditions/q/Germany/Berlin/.xml` We specify GET as the *HTTP method* and as *polling interval* 2400, adding the unit s for seconds (2400s). The interval of 40 minutes for weather updates keeps us well below our limit of 500 requests per day. and leaves room for additional screens retrieving weather information under the same account. Do not forget to add the unit s for seconds following the interval value, since the default value for the field is ms, in which case you would quickly exceed your Weather Underground quota.

What is left to do is extracting the temperature and weather values from the data, the Weather Underground site sends back triggered by our HTTP GET command.

For data extraction from a HTTP response OpenRemote offers the usage of XPath and regular expressions. If the data comes back in XML format, as in our case, it is easiest to use an XPath expression to process the data.

For those who have not worked with XML before - it is a really simple, mostly self-explanatory specification for data containers. It is also widely used in the Internet as the data transport format of choice, allowing for easy data exchange between different entities (websites, databases, software, etc.). XPath is nothing more than a structured definition language (similar to regular expressions) to

extract and filter data out of XML data structures. A good tutorial to start with XPath, in case you have not worked with it before, is available at http://www.w3schools.com/xsl/xpath_intro.asp.

To take a look at what we are dealing with, we trigger a response of the weather API by opening the URL we have crafted in a browser: <response>

```
<version>0.1</version>
```

```
<termsOfService>
```

```
http://www.wunderground.com/weather/api/d/terms.html </termsOfService>
```

```
<features>
```

```
<feature>conditions</feature> </features>
```

```
<current_observation>
```

```
<image>
```

```
<url>http://icons.wxug.com/graphics/wu2/logo_130x80.png</url>
<title>Weather Underground</title>
<link>http://www.wunderground.com</link> </image>
```

```
<display_location>
```

<full>Berlin, Germany</full> <city>Berlin</city>

<state/>

<state_name>Germany</state_name> <country>DL</country>

<country_iso3166>DE</country_iso3166> <zip>00000</zip>

<magic>10</magic>

<wmo>10389</wmo>

<latitude>52.51666641</latitude> <longitude>13.39999962</longitude>
<elevation>34.00000000</elevation> </display_location>

<observation_location>

<full>Berlin Tegel,</full>

<city>Berlin Tegel</city>

<state/>

<country>DE</country>

-- -- -- -- --
<country_iso3166>DE</country_iso3166> <latitude>52.56441498</latitude><longitude>13.30881691</longitude> <elevation>121 ft</elevation></observation_location>

<estimated></estimated>

<station_id>EDDT</station_id> <observation_time>Last Updated on April 26, 9:50 PM CEST</observation_time> <observation_time_rfc822>Tue, 26 Apr 2016 21:50:00 +0200</observation_time_rfc822><observation_epoch>1461700200</observation_epoch><local_time_rfc822>Tue, 26 Apr 2016 22:00:45 +0200</local_time_rfc822><local_epoch>1461700845</local_epoch><local_tz_short>CEST</local_tz_short><local_tz_long>Europe/Berlin</local_tz_long><local_tz_offset>+0200</local_tz_offset> <weather>Partly Cloudy</weather><temperature_string>37 F (3 C)</temperature_string> <temp_f>37</temp_f>

<temp_c>3</temp_c>

<relative_humidity>87%</relative_humidity> <wind_string>From the SSE at 7 MPH</wind_string> <wind_dir>SSE</wind_dir>

<wind_degrees>160</wind_degrees> <wind_mph>7</wind_mph>

<wind_gust_mph>0</wind_gust_mph> <wind_kph>11</wind_kph>

<wind_gust_kph>0</wind_gust_kph> <pressure_mb>999</pressure_mb>

<pressure_in>29.50</pressure_in> <pressure_trend>-</pressure_trend>
<dewpoint_string>34 F (1 C)</dewpoint_string>
<dewpoint_f>34</dewpoint_f>

<dewpoint_c>1</dewpoint_c>

<heat_index_string>NA</heat_index_string>
<heat_index_f>NA</heat_index_f> <heat_index_c>NA</heat_index_c>
<windchill_string>32 F (0 C)</windchill_string>
<windchill_f>32</windchill_f> <windchill_c>0</windchill_c>
<feelslike_string>32 F (0 C)</feelslike_string> <feelslike_f>32</feelslike_f>
<feelslike_c>0</feelslike_c> <visibility_mi>6.2</visibility_mi>
<visibility_km>10.0</visibility_km> <solarradiation/>

<UV>0</UV>

<precip_1hr_string>-9999.00 in (-9999.00 mm)</precip_1hr_string>
<precip_1hr_in>-9999.00</precip_1hr_in>
<precip_1hr_metric>-9999.00</precip_1hr_metric> <precip_today_string>0.00
in (0.0 mm)</precip_today_string> <precip_today_in>0.00</precip_today_in>
<precip_today_metric>0.0</precip_today_metric> <icon>partlycloudy</icon>

<icon_url>http://icons.wxug.com/i/c/k/nt_partlycloudy.gif</icon_url>
<forecast_url>

<http://www.wunderground.com/global/stations/10389.html> </forecast_url>

<history_url>

```
http://www.wunderground.com/history/airport/EDDT/2016/4/26/DailyHistory.htm  
</history_url>
```

```
<ob_url>
```

```
http://www.wunderground.com/cgi-bin/findweather/getForecast?  
query=52.56441498,13.30881691
```

```
</ob_url>
```

```
</current_observation>
```

```
</response>
```

While the above response looks threatening at first sight, all we are looking for are the values of the XML elements `<temp_c>` (in our case with the value of 3) and `<weather>` (with the value of partly cloudy). The XPath expression which selects the node `<temp_c>` consists of a double slash (`//`) followed by the node name: `//temp_c`

The command `text()` selects the actual value of a node. With that our XPath expressions to extract the current temperature and weather condition are as follows: `//temp_c/text()`

```
//weather/text()
```

To validate our XPath expressions we copy the XML source and our XPath

expression into an online XPath query checker such as
<http://codebeautify.org/Xpath-Tester>

and see that our assumption was correct. (Figure 11.1)

The screenshot shows the Code Beautify XPath Tester interface. On the left, there's an 'XML Input' section containing XML code for a weather response. On the right, there's a 'Result : Generated XPath' section showing the output of the XPath expression //temp_c/text().

XML Input:

```
<response>
    <version>0.1</version>

    <termsofService>http://www.wunderground.com/weather/api/d/terms.html</termsofService>
        <features>
            <feature>conditions</feature>
        </features>
    <current_observation>
        <image>

        <url>http://icons.wxug.com/graphics/wu2/logo_130x80.png</url>
            <title>Weather Underground</title>
    
```

XPath Expression:

```
//temp_c/text()
```

Result : Generated XPath:

```
1 Text = 4
2
```

Figure 11.1 XPath expression validation with the Codebeautify XPath Tester We can now copy the XPath expressions in the OpenRemote command windows of our two commands, which we call TempBerlin and WeatherBerlin (Figure 11.2). In order to use the commands in our OpenRemote app we need, as a final step, to embed them in a sensor definition. This is easily done by selecting New – New Sensor and adding the command we want to associate with the sensor. We simply name the two sensors after their commands TempBerlin and

WeatherBerlin and select for each of them their respective command.

Edit command

Name: TempBerlin

Protocol: HTTP

HTTP attributes

URL: http://api.wunderground.com/api/77ca1333333333333333333333333333/conditions/q/US/Berlin.json

HTTP Method: GET

Content-Type:

Workload:

Username:

Password:

XPath Expression: //temp_c/text()

RegularExpression:

Polling interval: 2400s

JSONPath Expression:

Submit **Reset**

Figure 11.2 Temperature retrieval from the Weather Underground service using XPath. Alternatively we could have also used a regular expression to filter on one, two or three digits following the string <temp_c> and <weather>. For those who have not worked with regular expressions before: They are a structured specification language used to specify string pattern matches. You find a good reference under

http://en.wikipedia.org/wiki/Wikipedia:AutoWikiBrowser/Regular_expression.

11.2 Designing the App Layout

As the final step, we want to display the sensor output in our smart home app. Before we do that, we need to plan the layout for our GUI. Our plan is to have three screens called *Remote*, *Lighting* and *Heating*. In the UI designer window, we drag a *Tab Bar* item from the widget menu on the right to the bottom of our design and add three *Tab Bar Item* elements to it, which we call Remote, Heating, and Lighting. Then we add two new screens, in addition to the existing Remote screen, which we call Lighting and Heating. We will make use of these two screens later. (Fig. 11.3, Fig . 11.4)

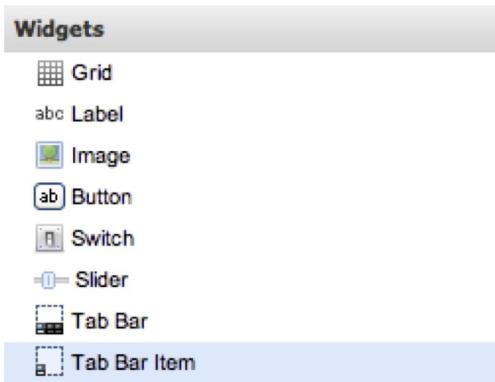


Fig. 11.3 Adding a *Tab Bar* Item to theOpenRemote GUI design

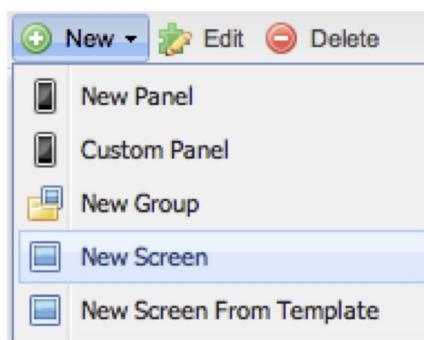


Fig 11.4 Adding a *New Screen* to the Openremote GUI design

To our Remote screen we add a *Grid* element by dragging the Grid symbol from the menu into the design, setting the parameters to:

Row Count: 1, Col Count: 1, Left:0, Top:0, Width:75, Weight:45

This positions a grid with one cell of the size of 75x45 to the upper left corner of our design. We add three additional cells to the right:

Row Count: 1, Col Count: 1, Left:75, Top:0, Width:150, Weight:45

Row Count: 1, Col Count: 1, Left:225, Top:0, Width:30, Weight:45

Row Count: 1, Col Count: 1, Left:255, Top:0, Width:30, Weight:45

We now drag an *abc label* symbol into the upper left cell and enter the location name for our weather report in the *Label Properties* field to the right, which is in our case Berlin. We do the same for the second cell, entering Weather in the text field, but here we also add the sensor WeatherBerlin, which can be selected from the sensor drop down menu in Label properties. In the next field, we enter T for temperature and select the sensor TempBerlin. And the final cell just contains the unit for our temperature display, in our case C for Celsius (Figure 11.5). We save our design by clicking on the *disc* symbol. We now go to the controller window in our web browser and click on *Synch with Online Designer*. After restarting our OpenRemote app on our smartphone we should now see in addition to our Hello World message weather condition and temperature for Berlin. (Figure 11.6).

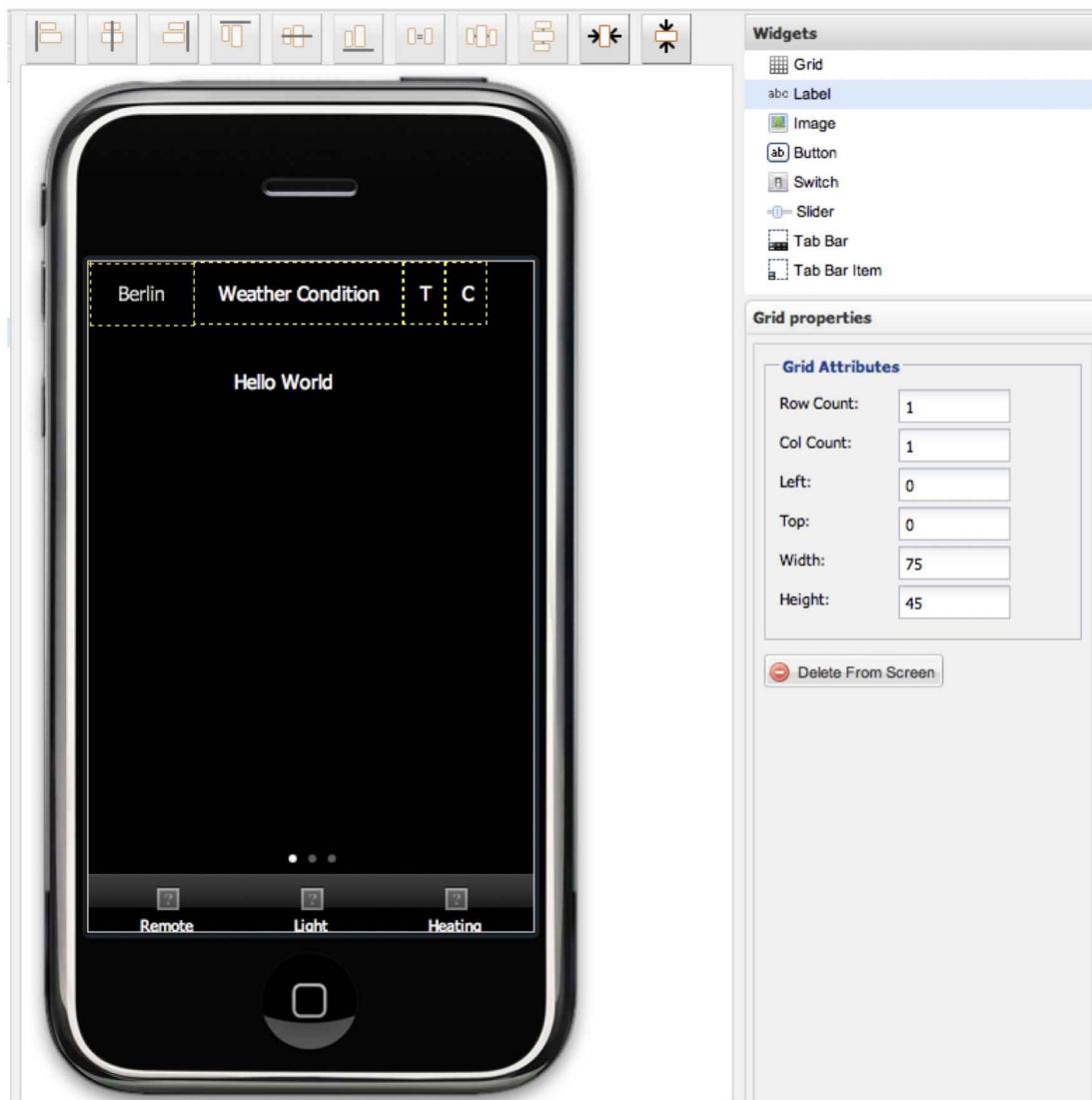


Figure 11.5 The OpenRemote GUI design for the Internet based weather sensor

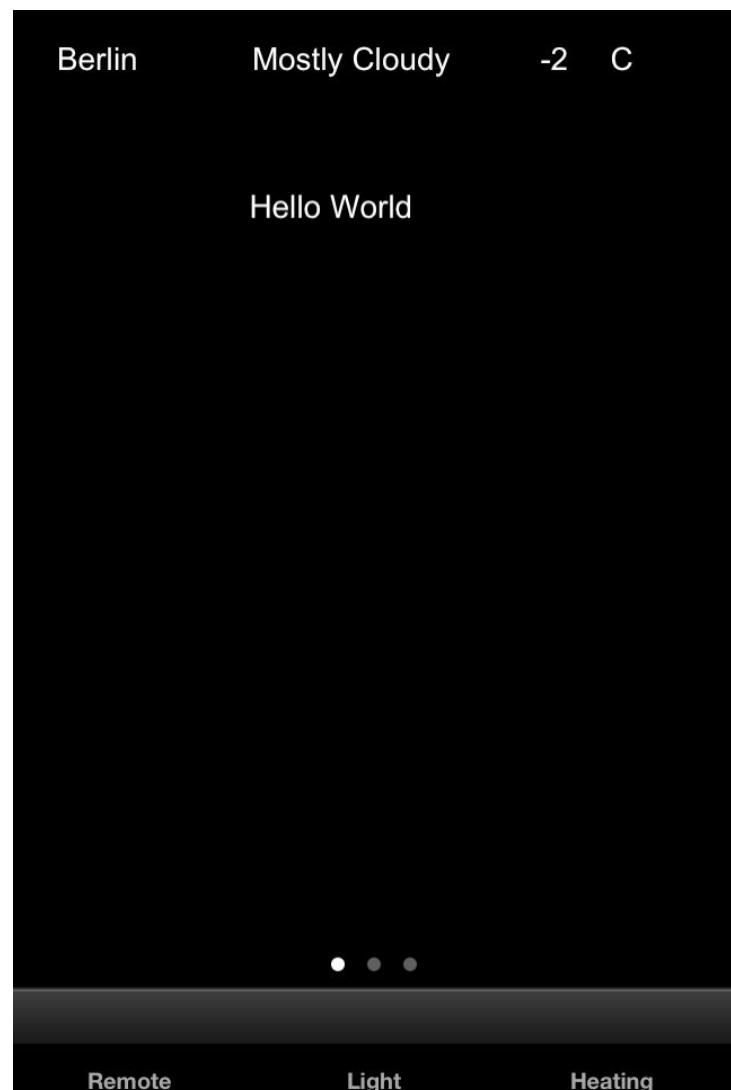


Figure 11.6 The weather sensor display in operation

11.3 Bibliography

Weather Underground - The Weather Channel (2016): Data Features: URL Format. <https://www.wunderground.com/weather/api/d/docs?d=data/index>.

12 Integration of Multimedia: iTunes Remote

The ability to control scripts via OpenRemote also allows us to integrate the multimedia capabilities of computers, such as playing audio and video files, TV programs, and radio stations. In this chapter we will set up our OpenRemote smartphone app to function as a remote control for Apple's multimedia suite iTunes (on both Macs and PCs) and Microsofts MediaPlayer under Windows 10. In chapter thirteen we will then demonstrate how to create automated rules, which allow us to put together the control components, we have created so far into a powerful, rule based smart home control application.

12.1 Scripting Basics: Shell what?

Scripts are short programs consisting of one or more commands in text form. Such programs, which do not need to be compiled in order to be executed, are also called command interpreters. All computer operating systems have scripting environments, to allow users to perform basic functions such as file handling or task scheduling. The command interpreter in Unix type systems is called shell. With Unix Version 7 an implementation called Bourne shell became popular, in later years its open source implementation Bash shell was chosen as the default command line interpreter environment for Linux and macOS. Thus when opening the terminal window on a Mac, you are using its Bash shell environment.

The Windows scripting equivalent to the Unix shell command-line interpreter in Linux and macOS used to be cmd.exe, which was introduced with Windows NT in 1993. While cmd.exe offered a few improvements over its predecessor cmd.com, compared to the UNIX shell its capabilities were still very limited. In 2006 Microsoft released PowerShell, a command line scripting environment with a feature set comparable to Unix shells, using many of the same commands. PowerShell is part of Windows 10, but can also be downloaded for free for prior Windows versions. In 2016 Powershell became OpenSource and since is also available for macOS and Linux.

We will be using PowerShell for the Windows versions of our scripts and the Mac Bash shell environment for macOS based scripts. In addition, under macOS we will also be using AppleScript, an easy to use scripting environment for Macs, which conveniently allows to create automation and control tasks for macOS applications (<https://msdn.microsoft.com/powershell>).

12.2 Testing it Right - Best Practice for Script Writing

While scripts are really short pieces of code, they are very sensitive to syntax errors, and their functioning depends heavily on your system settings and your file structure. This means, that you probably will need to make a few modifications to the sources I describe, in order to get it to run on your system.

So I would advise you NOT to copy and paste the code of the described scripts onto your machine and execute in order to see what's happening.

I recommend to test the scripts line by line (by copying each line, or in case of a loop just the portion with the loop - as long it can run on its own. If needed make necessary modifications, execute the code, and watch if the outcome is as expected. Once the line is tested and works as desired, you add it to the already cleared code and execute the script from there. While doing this, I recommend to add temporary echo commands to the code, which allows you to observe the execution process of the code. I ensure you, that this approach will get you to a working and stable script much faster, than a try and error approach which starts from larger pieces of code.

12.2.1 Curly or straight, this is the question!

Finally, before you get started with scripting, I repeat this important hint for editing scripts. Quotation marks in scripts have an important function. However, the quotes used in scripts need to be the straight quotation marks, rather than the curly quotation marks, which are used by word processor software. It can happen, when you edit a shell script or an AppleScript usingTextEdit or another word processor, that the straight quotes are converted to curly quotes resulting in runtime errors, which at first sight are hard to analyze. So always take a close look at the quotation marks. Curly or straight, this is the question ;-).

~~In case you have never written scripts, or you have not had the chance to work~~
with scripts in a long time, now might be a good time to invest an hour or two to go through one of the many excellent basic tutorials on scripting. To find some of them, search terms you could use with Internet search engines are: shell scripting beginner Apple or shell scripting beginner Windows 10

12.2.2 Scripting under macOS

To create a first test shell script create the directory shconfig under your project directory shProject. For writing our shell script we can use any text editor. On a Mac it is easiest to use TextEdit. Open TextEdit and choose *File – New* and then *Format – Make Plain Text*. Save the file using the extension .sh for shell.

Execution of the script is done by typing ./ if we reside in the same directory as the script, followed by the name of the script, in the Mac terminal window, e.g.:
. /test.sh

Now insert the following content into test.sh, then save the file: #!/bin/sh

```
echo "Hello World"
```

Now open a terminal window, change to the directory of test.sh, which in our case is shProject/shconfig and give the file the required execution permission by typing: chmod a+x test.sh

Now you run the script by typing ./test.sh

If you see the Hello World output in the terminal window, you have successfully run your first script. Be aware, that in case you are not in the directory, where the script is located at, you always have to enter the full path to the script in order to start it. Even if you reside in the directory of the script itself, you need -

for security reasons - to precede the file name with ./ as below: ./test.sh

12.2.3 Shell Scripts for Telnet and HTTP

Shell scripts can also be used to execute HTTP and Telnet commands. With that you can also craft shell scripts, which retrieve HTML pages or run Telnet sessions. On a Mac HTTP commands can be called from the command line using the cURL command, on a Linux system you can use the command wget. As an example the below command retrieves the file test.txt using HTTP GET and stores it in local_test.txt: curl http://example.org/test.txt > local_test.txt As an example, the shell scripts, which switch our DENON audio system to on and off would read as follows: denonOn.sh

```
#!/bin/sh
```

```
curl http://192.168.178.16/goform/formiPhoneAppDirect.xml?Z2IPD
```

denonOFF.sh

```
#!/bin/sh
```

```
curl http://192.168.178.16/goform/formiPhoneAppDirect.xml?PWSTANDBY
```

When creating a shell script for a Telnet session, you need to feed the inputs for the session with a pipe command to the telnet command as follows: #!/bin/sh

```
{
```

```
sleep 3
```

```
echo null|PWSTANDBY
```

```
echo exit
```

```
} | telnet 192.168.178.16
```

If you need simulate an entire login sequence or want to execute other consecutive commands, make sure to insert sleep commands in between to give the telnet protocol the chance to sequentially execute the commands and receive the according responses between the commands. Below a sample structure you can use.

```
#!/bin/sh
```

```
HOST='0.0.0.0'
```

```
USER='User'
```

```
PASSWD='Pass'
```

```
CMD="
```

```
{
```

```
echo open "$HOST"
```

```
sleep 2
```

```
echo "$USER"
```

```
sleep 2
```

```
echo "$PASSWD"
```

```
sleep 2
```

```
echo "$CMD"
```

```
sleep 2
```

```
echo "exit"
```

```
} | telnet
```

12.2.4 Scripting under Windows 10

Under Windows we will use Microsoft's scripting environment PowerShell, which comes in two versions: PowerShell ISE, which provides a graphical user interface with editor and terminal window integrated, and the standard PowerShell in form of a terminal window with the PS prompt for PowerShell. The easiest way to start PowerShell under Windows 10 is to type PowerShell in the Cortana search bar. Windows 10 will then offer two applications: Windows PowerShell and WindowsPowershell ISE. After selecting WindowsPowershell ISE, the application will open. Mac users now move on with the next section, Windows users skip ahead to section 12.5.

12.3 Script Based iTunes Control in macOS

Under macOS writing a script, which involves interaction with applications such as iTunes, is easiest using the powerful scripting language AppleScript. The AppleScript specification is based on the Open Scripting Architecture (OSA). Therefore any AppleScript can be executed from a shell using the `osascript` command. To get started with AppleScript it is best to use the AppleScript Editor application. Go to *Applications – Utilities – AppleScript Editor* to start the AppleScript development application. Enter the following code, replacing Relaxation with a playlist that actually exists on your PC: tell application "iTunes"

```
play user playlist "Relaxation"
```

```
end tell
```

Then select run and iTunes should launch and start to play the selected playlist. (Fig. 12.1)

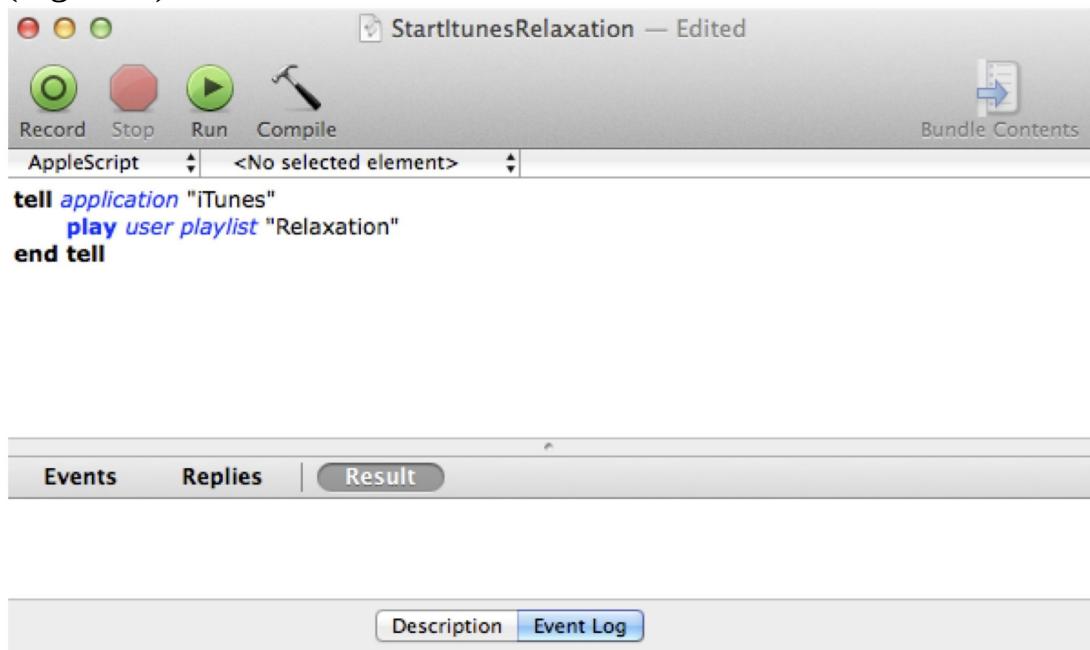


Figure 12.1 Working with the AppleScript Editor If you want to run the above in a shell script, you just need to insert the osascript command in the first line and tell the script interpreter to handle all text following exec osascript as an osascript until EOF is reached: exec osascript << EOF

```
tell app "iTunes"  
  
play user playlist "$1"  
  
end tell  
  
EOF
```

Adding the possibility to set the playlist through an argument as well as an argument error procedure gives you the final version of your script
playItunes.sh: #!/bin/sh

```
#Presence detection using the smartphone DHCP request when booking ton a  
Wi-Fi (WLAN) network if [[ $# -lt 1 || $# -gt 1 ]];then
```

```
echo "$0: Argument error: itunesStart.sh [playlist]"
```

```
exit 2
```

```
fi
```

```
exec osascript << EOF
```

```
tell app "iTunes"
```

```
    play user playlist "$1"
```

```
end tell
```

```
EOF
```

The command

```
playItunes.sh Relaxations
```

now opens iTunes and starts playing the playlist Relaxation.

Prior to having iTunes play a playlist we need to start it and redirect the iTunes output via AirPlay to an output of choice. Since there is no direct AppleScript command for this, we need to simulate the user interaction, which is to click on the drop-down menu *Choose which speakers to use* and select the AirPlay device of choice. To do this we need to make iTunes the active window and then simulate the drop-down click: click (first UI element whose help is "Choose which speakers to use") and the selection of the desired AirPlay device, in our case the Denon AVR-3313. The command keystroke "DENON"

will literally type the letters DENON and by doing this select the AirPlay device

DENON in the drop down window opened by the click command. The command key code 76

operates the return-key and the one second delay makes sure the GUI can follow the speed of AppleScript. Before executing the keystroke command we insert the command line tell application "iTunes" to activate to avoid it being written into another window, which might have opened in the meantime: tell application "iTunes" to activate delay 1

tell application "System Events"

tell window "iTunes" of process "iTunes"

click (first UI element whose help is "Choose which speakers to use.") delay 1

tell application "iTunes" to activate keystroke "DENON"

delay 1

key code 76

end tell

delay 1

end tell

In order to enable the above script to work, you need to ensure that under [System Preferences – Accessibility](#) on your Mac the Terminal application is configured as a controller for other applications. For that go to [System Preferences – Security&Privacy](#), select the pane [Privacy](#) and the menu [Accessibility](#). Then click on the [plus sign](#) and add the Terminal application to the list of apps, which can control your Mac. Checkmark the Terminal.app entry and you are done (see figure 12.2).

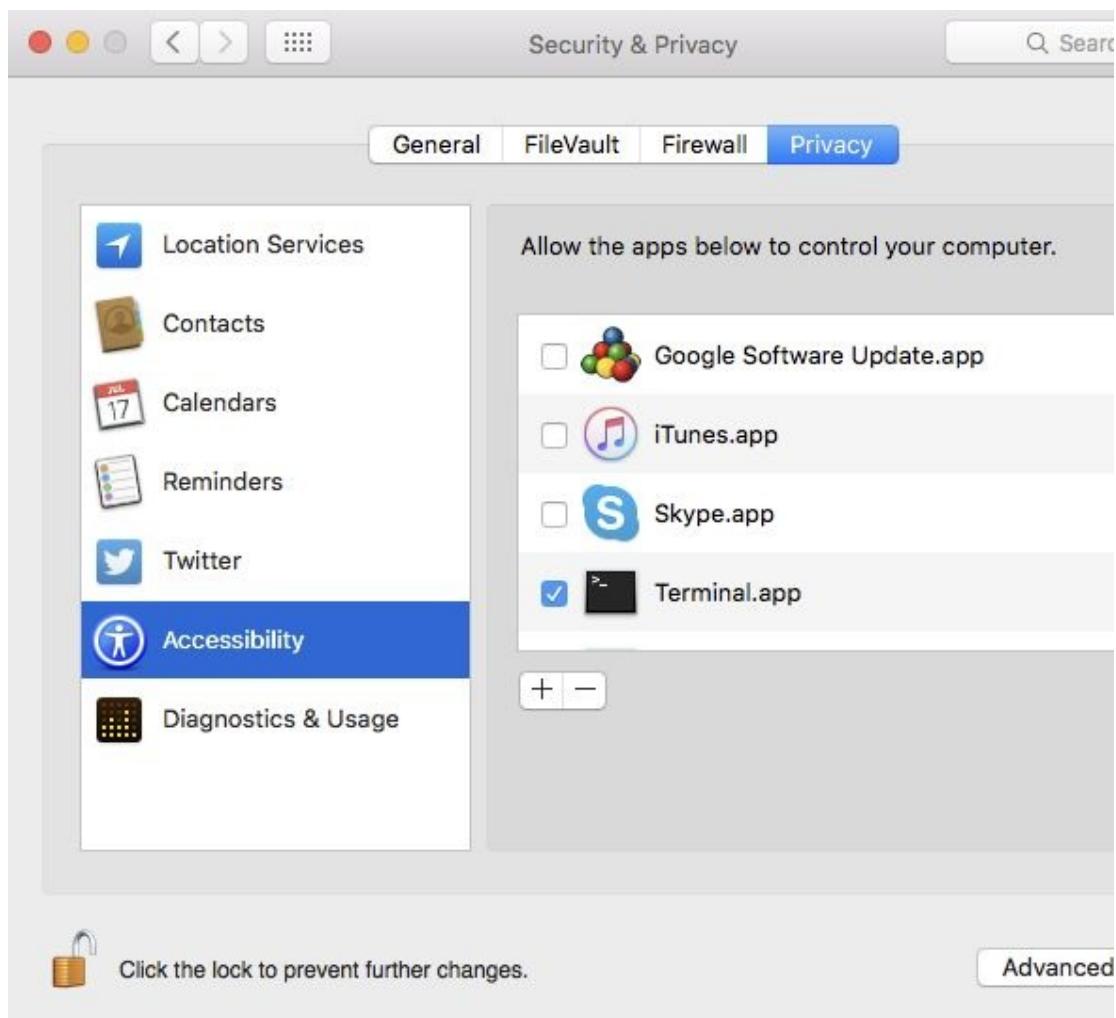


Figure 12.2 Enabling control for the Terminal application on a Mac (macOS 10.11 El Capitan) While this is not the most elegant solution, it demonstrates, how practically any application can be integrated in your smart home

now practically any application can be integrated in your smart home infrastructure. We add the command Z2ON (power on zone 2 speaker) to our DENON audio system with an HTTP command: curl http://192.168.178.16/goform/formiPhoneAppDirect.xml?Z2ON

And finally we want iTunes to start playing the playlist Recentlyplayed, for which we add the following command: tell application "iTunes" to play playlist "Recentlyplayed"

With this our complete script iTunesStart.sh reads now as follows: #!/bin/sh

```
#send command Z2ON to DENON audio system using HTTP GET
```

```
curl http://192.168.178.16/goform/formiPhoneAppDirect.xml?Z2IPD
```

```
sleep 3
```

```
echo sleep 3
```

```
#Start iTunes, set airplay to DENO and start playlist Recentlyplayed exec  
osascript << EOF
```

```
tell application "iTunes" to activate delay 2
```

```
tell application "System Events"
```

```
tell window "iTunes" of process "iTunes"
```

click (first UI element whose help is "Choose which speakers to use.") delay 1

tell application "iTunes" to activate keystroke "DENON"

delay 3

key code 76

tell application "iTunes" to play playlist "Recentlyplayed"

delay 1

end tell

end tell

EOF

To complete our shell based iTunes control capabilities we add the scripts stopiTunes.sh, iTunesVolUp.sh, iTunesVolDown.sh and iTunesTo75.sh to control the volume and close iTunes. (iTunesTo75.sh sets the iTunes volume to 75% of the maximum): siTunesStop.sh

#!/bin/sh

```
#send command PWSTANDBY to DENON audio system using HTTP GET
```

```
curl http://192.168.178.16/goform/formiPhoneAppDirect.xml?PWSTANDBY
```

```
#write off to state variable file iTunes.txt echo off >  
Userssmarthome/shProject/shconfig/itunes.txt #Stop iTunes
```

```
exec osascript << EOF
```

```
tell application "iTunes" to quit
```

```
EOF
```

```
iTunesVolUp.sh
```

```
#!/bin/sh
```

```
#Turn up iTunes volume
```

```
exec osascript << EOF
```

```
tell application "iTunes"
```

```
if it is running then
```

-- -- -- -o -- --

```
if sound volume is less than 100 then set sound volume to (sound volume + 10)
end if
```

```
end if
```

```
end tell
```

```
EOF
```

iTunesVolDown.sh

```
#!/bin/sh
```

```
#Turn down iTunes volume
```

```
exec osascript << EOF
```

```
tell application "iTunes"
```

```
if it is running then
```

```
if sound volume is greater than 0 then set sound volume to (sound volume - 10)
end if
```

end if

end tell

EOF

iTunesVolTo75.sh

#!/bin/sh

#Set iTunes volume to 75

exec osascript << EOF

tell application "iTunes"

if it is running then

set sound volume to 75

end if

end tell

EOF

Later, we will add the above scripts as commands to OpenRemote Designer, using the shell execution protocol. Since we also want to display the iTunes status and the current track iTunes is playing, we need one more AppleScript based shell script, which determines if iTunes is playing, and which retrieves artist and title of the currently played song. If iTunes is in status playing, we want the iTunes status file itunes.txt to contain the value on, otherwise the value off. Later using the file itunes.txt we will build an OpenRemote sensor and an based on it an OpenRemote switch, which will allow us to switch on and off iTunes from our smarthome console. In case iTunes is in playing mode, the script should further retrieve artist name and title and output it as a string using an echo command. With that our script iTunesPlaying.sh reads to: #!/bin/sh

```
#Retrieves status and currently played title from iTunes and displays results  
using echo command #If iTunes status is “playing” it writes “on” to status file  
iTunes.txt, if status is not playing it writes “off” to iTunes.txt state=`osascript -e  
'tell application "iTunes" to player state as string`#echo "iTunes is currently  
$state."; if [ $state = "playing" ]; then
```

```
echo on > /Userssmarthome/shProject/shconfig/itunes.txt artist=`osascript -e 'tell  
application "iTunes" to artist of current track as string` track=`osascript -e 'tell  
application "iTunes" to name of current track as string` echo "Current track  
$artist: $track"
```

```
else
```

```
echo "iTunes is currently $state."
```

```
echo off > Userssmarhome/shProject/shconfig/itunes.txt fi
```

To test we manually start an iTunes playlist and type in a terminal window
./iTunesPlaying.sh

We should now see the artist and song title iTunes is playing as an ouput in the terminal window. Further we should see the value on in the newly created file itunes.txt.

At last we create a simple shell script. which reads out the content of itunes.txt by simply using the cat command. We call the file iTunesStatus.sh and it just contains the two lines: #!/bin/sh

cat Userssmarhome/shProject/shconfig/itunes.txt Since our shell scripts are now working as desired, we can move on to integrate them in our OpenRemote structure. Before we do this however, one last word on shell script writing for OpenRemote.

12.3.1 Shell Script Execution from OpenRemote: /bin When OpenRemote executes a shell script, it does it from its binary directory /ORC/bin, independent of where the shell script actually is stored, which in our case is shProject/shconfig. So if the shell script for example writes to a file in the same directory as its location (e.g. *shconfig*), *make sure you use the full absolute path definition. If you use a relative path definition in our example you would need to go two diretctory levels up (from shProjectORC/bin to shProject) and from there one level down to shconfig*. The according command would would read echo off > ../../shconfig/itunes.txt If someone else needs to maintain or modify your scripts, it is easier and more clear to use the longer absolute path definitions: echo off >

UserssmarthoneshProject/shconfig/itunes.txt as we have done in our examples above.

12.4 Smartphone Remote for iTunes (macOS)

We now can move on to configure the above scripts as commands in OpenRemote Designer. We go to OpenRemote Designer and create a new device called iTunes by selecting [New — New Device](#). We then select the device and open the command editor window: [New — New Command](#). As the name for our first command we enter iTunesStart, for the protocol we select [Shell execution protocol](#). We enter the shell script name iTunesStart.sh including its complete path. (Figure 12.3) The same way we create the command iTunesStop.

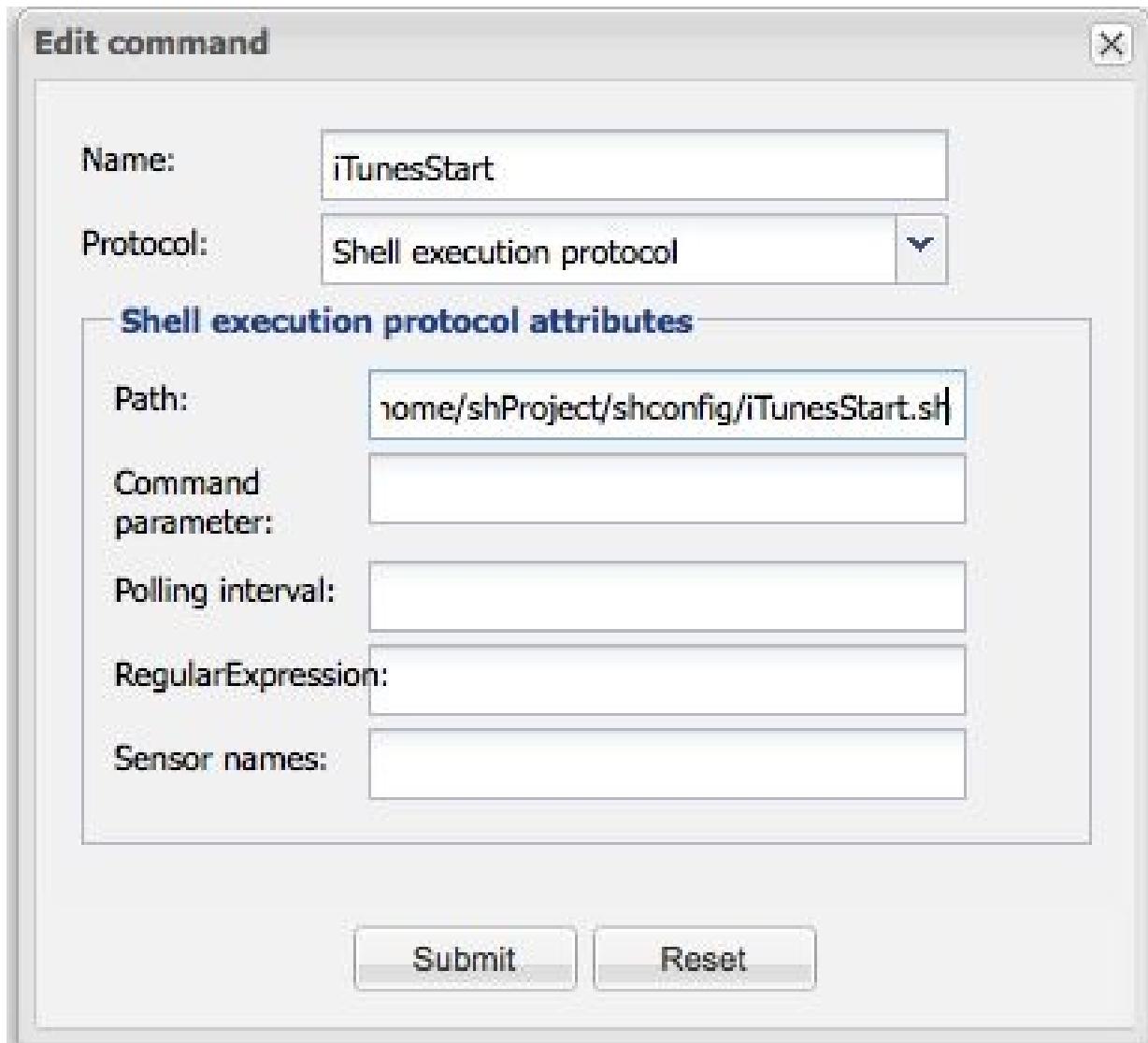


Figure 12.3 Command to start iTunes with audio output device DENON

We repeat the above steps for our shellscript playItunes.sh with the parameters of the playlists we want to control, in our case classic, jazz, rock, pop and news.

Now we create the command iTunesCurrentlyPlaying, which starts the shellscript iTunesCurrentlyPlaying.sh with the *Polling interval* five seconds to output artist and title of the current track as well as the values on and off for our iTunes status file itunes.txt (Figure 12.4) Then we create the OpenRemote sensor iTunesCurrentlyPlaying based on this command. Only the creation of the sensor actually activates the periodic execution of the command every five seconds.

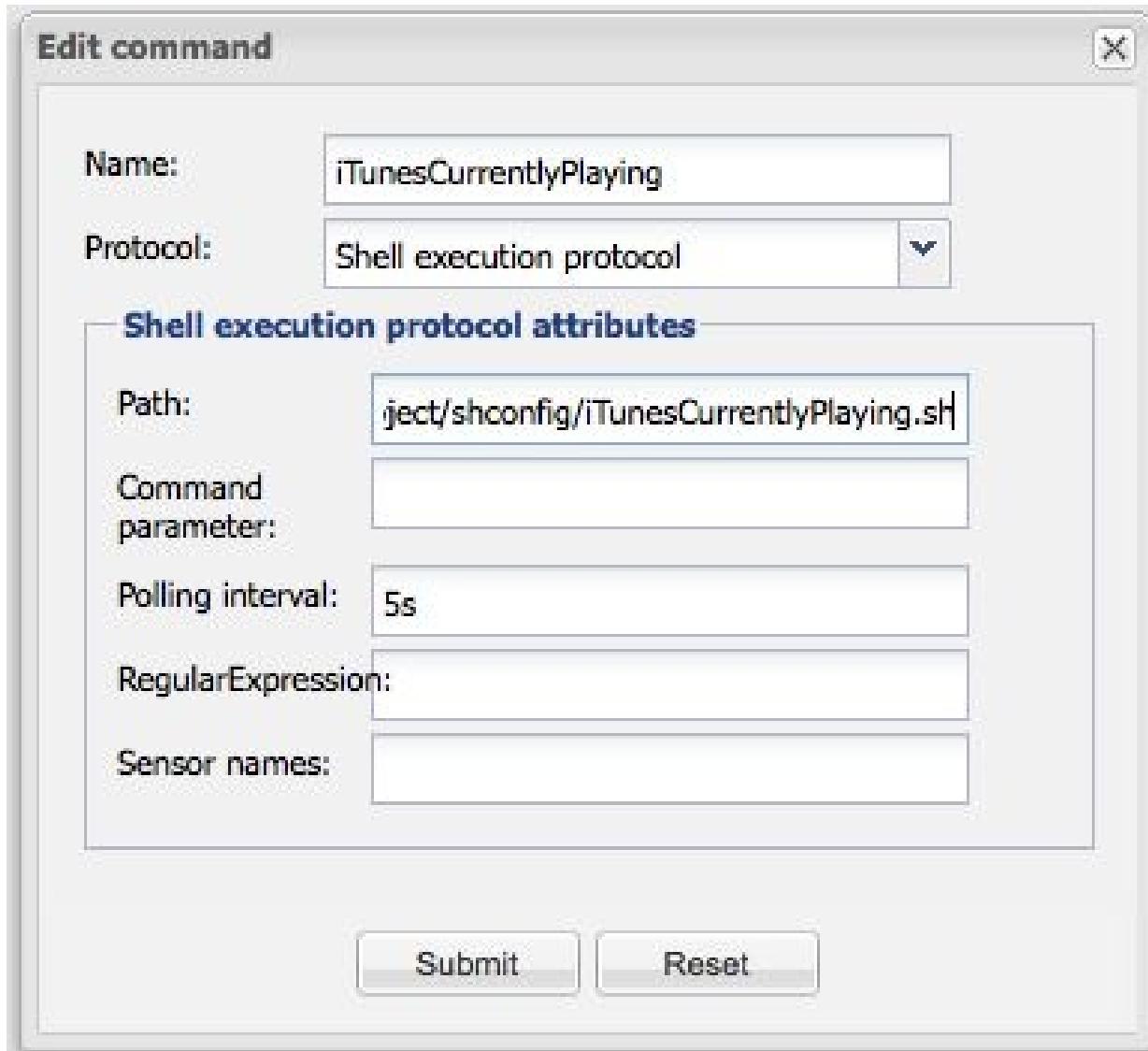


Figure 12.4 The command *iTunesCurrentlyPlaying*

Finally we build a switch command for iTunes. For this task we already have the two commands *iTunesStart* and *iTunesStop*, however are missing the sensor, which displays the status of iTunes. So we first create the command *iTunesStatus* based on our shell script *iTunesStatus.sh*. In the next step we build the sensor *iTunesStatus* and can now create the switch command *iTunes* with *iTunesStart*, *iTunesStop* and *iTunesStatus* (Figure 12.5).

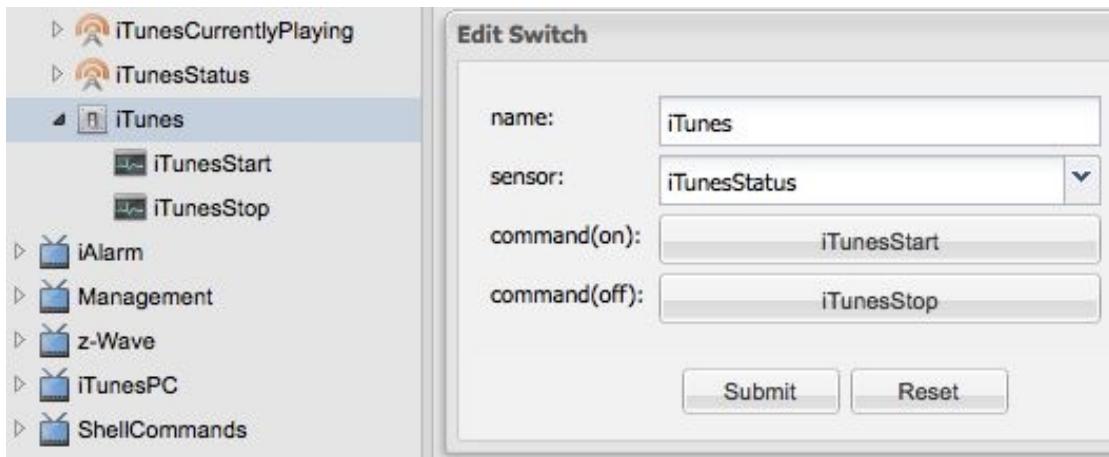


Figure 12.5 Switch command definition for iTunes

With that done, we have all the commands and sensors we need and can move on to the UI Designer section. We create a new screen called iTunes and add a grid with 2 rows, 5 columns and the dimensions 315 x 90 (for an iPhone). We drag in a *label* element for the name of the control sequence (in our case iTunes), a *switch* element for our iTunes switch, which we configure with the iTunes switch command, and seven *buttons*, which we configure with the commands PlayiTunes Classic, Play iTunes Jazz, PlayiTunes Pop, PlayiTunes Rock, PlayiTunes News, volumeupiTunes and volumedowniTunes. Then we add another rectangle below for the display of artist and track information, which we configure with a *label* element, which we assign the sensor command iTunesCurrentlyPlaying. (Figure 12.6). Figure 12.7 shows the resulting GUI of our iTunes remote control on an iPhone screen.

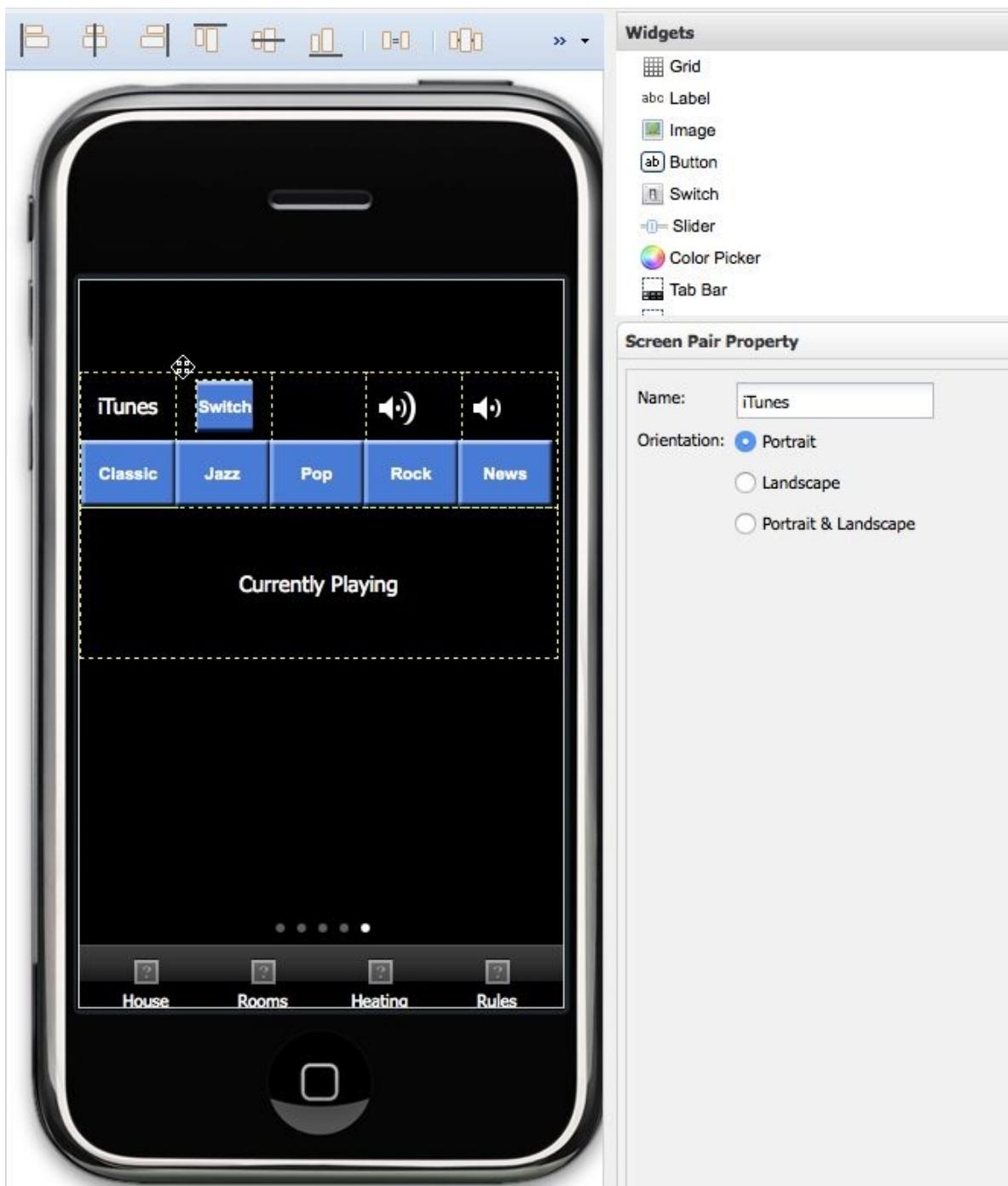


Figure 12.6 Design of iTunes Remote Control GUI with OpenRemote Professional Designer

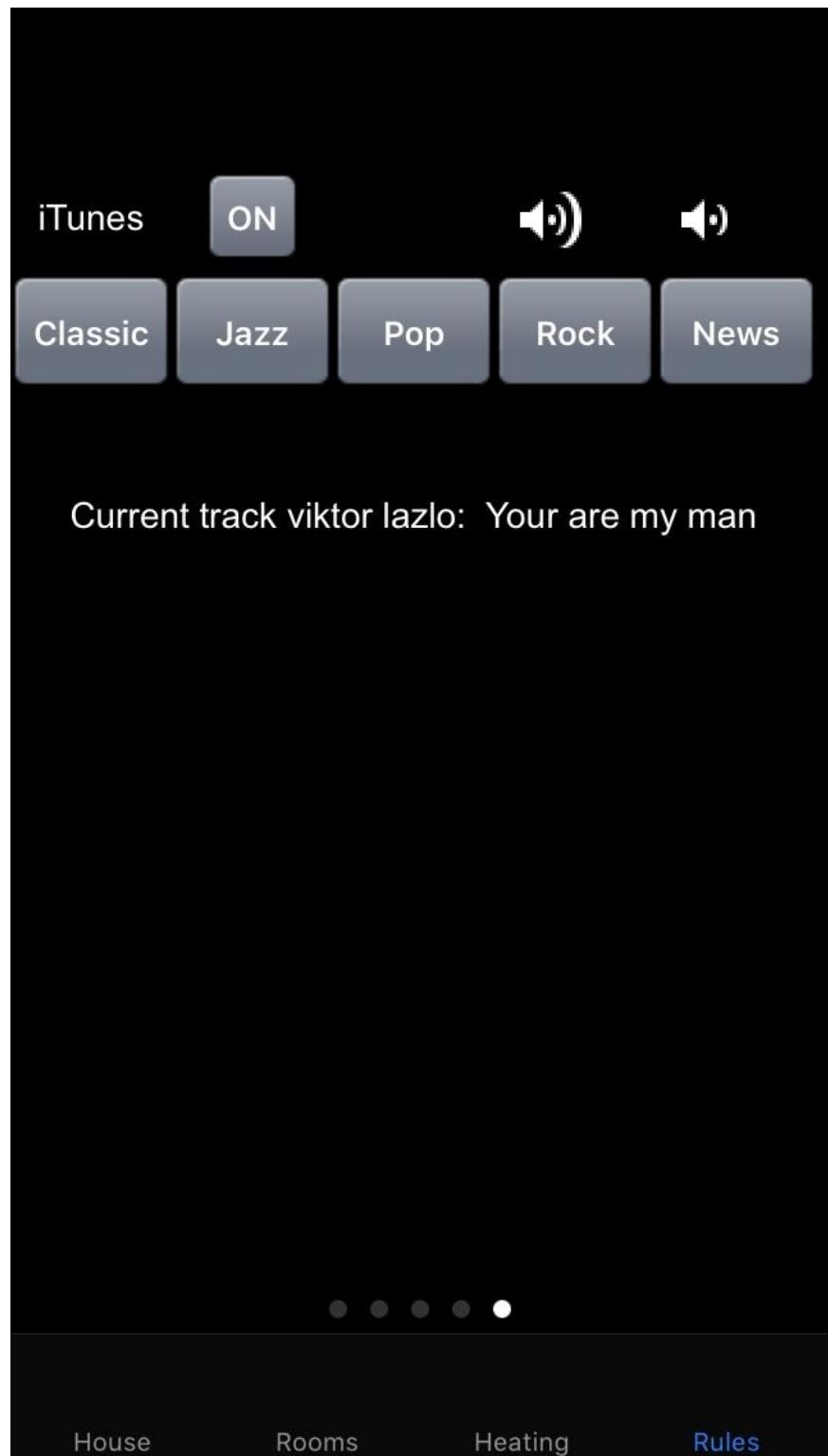


Figure 12.7 iTunes Remote Control GUI on smartphone screen

12.4.1 Subsequent Command Execution: Macros

If several OpenRemote commands shall be executed in sequence, OpenRemote provides an easy to use macro function. To use it, we expand the *Macro* menu and select *New* on the left side of the OpenRemote GUI. As an example we name our first macro iTunesON and can now drag the commands, which we want to be executed into the macro window, in our case the commands AVR 3313 Zone 2 (ON) and startItunes DENON, which starts the DENON AVR 3313 with Zone 2 outputs active and then iTunes, with audio airplay set to DENON. To make sure there are no timing problems between two commands, we insert a delay function in between (Figure 12.8).

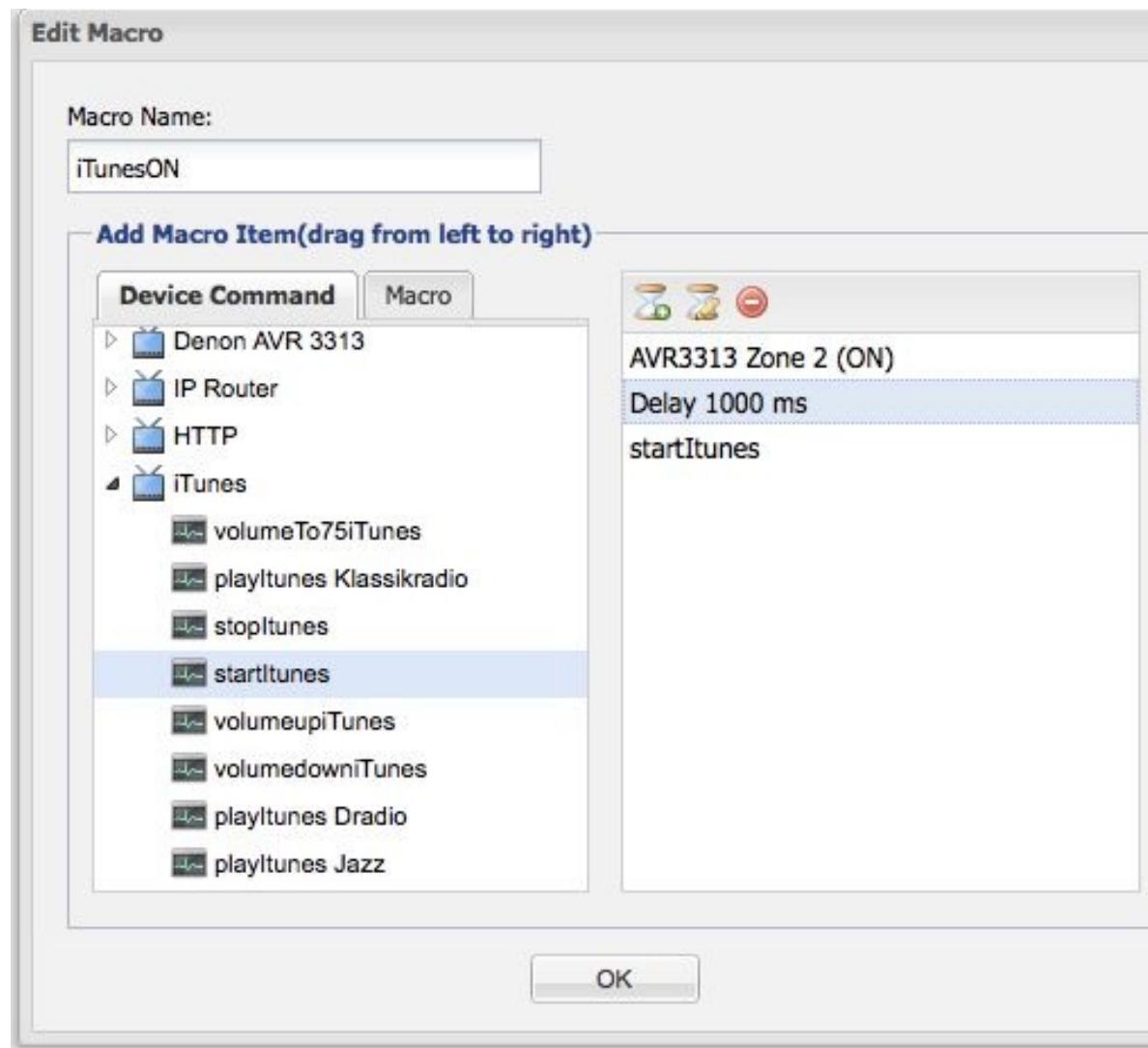


Figure 12.8 Definition of the macro *iTunesON*: Creating sequences of OpenRemote commands

12.5 Script Based MediaPlayer Control (Windows 10)

For playing audio and video files Windows 10 provides Windows MediaPlayer in its most recent version 12, which has not changed since 2009. We will use the Microsoft scripting environment Powershell to craft a simple script, which starts MediaPlayer playing a dedicated playlist. We will then also build a script which controls Apple iTunes for Windows, which is always available for free in its latest version.

12.5.1 Scripting under MS Windows

The WindowsPowershell ISE GUI consists of three windows (Figure 12.9):

- the script editor
 - the terminal output window and
 - the command window, with a listing of the available commands
-

The screenshot shows the Windows PowerShell ISE interface. In the top navigation bar, the title is "Windows PowerShell ISE". Below it is a toolbar with various icons. The main workspace has two tabs: "MediaPlayerPlaylist.ps1" and "GetPowerShellVersion.ps1". The "GetPowerShellVersion.ps1" tab is active, showing the command "1 get-host". The results pane below displays the output of the "get-host" command:

```
PS C:\Users\othmar> get-host

Name          : Windows PowerShell ISE Host
Version       : 5.0.10240.16384
InstanceId    : c391faec-8cd9-4ef8-9bd1-c2213f71270e
UI           : System.Management.Automation.Internal.Host.InternalHostUser
               nterface
CurrentCulture : de-DE
CurrentUICulture : en-US
PrivateData   : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled : True
IsRunspacePushed : False
Runspace      : System.Management.Automation.Runspaces.LocalRunspace
```

To the right of the results pane is a "Commands" pane with a search interface. It shows a list of cmdlets starting with 'A':

- A:
- Add-AppxPackage
- Add-AppxProvis
- Add-AppVolume
- Add-BitLockerKe
- Add-BitsFile
- Add-CertificateE
- Add-Computer
- Add-Content
- Add-DnsClientN
- Add-DtcClusterT
- Add-EtwTracePr
- Add-History
- Add-InitiatorIdTc
- Add-JobTrigger
- Add-KdsRootKey
- Add-Member

Figure 12.9 The PowerShell GUI (Windows 10, PowerShell 5) If you insert the command

```
get-host
```

in the script window and click on the green *run script* arrow, the above code line is being executed, and the current version of PowerShell is being displayed in the terminal window.

As many other Microsoft applications Windows Media Player Microsoft can be controlled using ActiveX controls. A reference for the ActiveX controls available for Windows Media Player can be obtained from the Windows Development center: [https://msdn.microsoft.com/en-us/library/windows/desktop/dd564679\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd564679(v=vs.85).aspx)

For the purpose of inter-process communication under Windows Microsoft has defined the COM (Component Object Model) interface standard. Using the command new-object -com WMPLAYER.OCX we create a new COM object which references to Windows Media Player. For better readability of the code, we assign this object definition to the variable \$player. (In Powershell all variables start with \$). Next we assign the the name of our playlist to the variable \$name. The command \$player.mediacollection.getByName(\$name) now reads out the titles of our playlist, which we store in the variable \$item. If \$item is not empty, using the command \$player.openplayer(\$filename) we have Windows Media Player play the songs of our playlist. With that our simple Powershell script reads as follows: \$player = new-object -com WMPLAYER.OCX

```
$name = "jazz"
```

```
$item = $player.mediacollection.getByName($name) if ($item.Count -gt 0) {
```

```
$filename = $item.item(0).sourceurl
```

```
$player.openplayer($filename)
```

```
} else {
```

```
    "$name' not found."
```

```
}
```

If you copy the above code in the PowerShell editor and select *Run Code*, Windows Media Player will open and start playing the playlist called smarthouse. Obviously you have to replace the playlist name of our example (jazz) with a playlist, which actually exists on your PC (Figure 12.10).

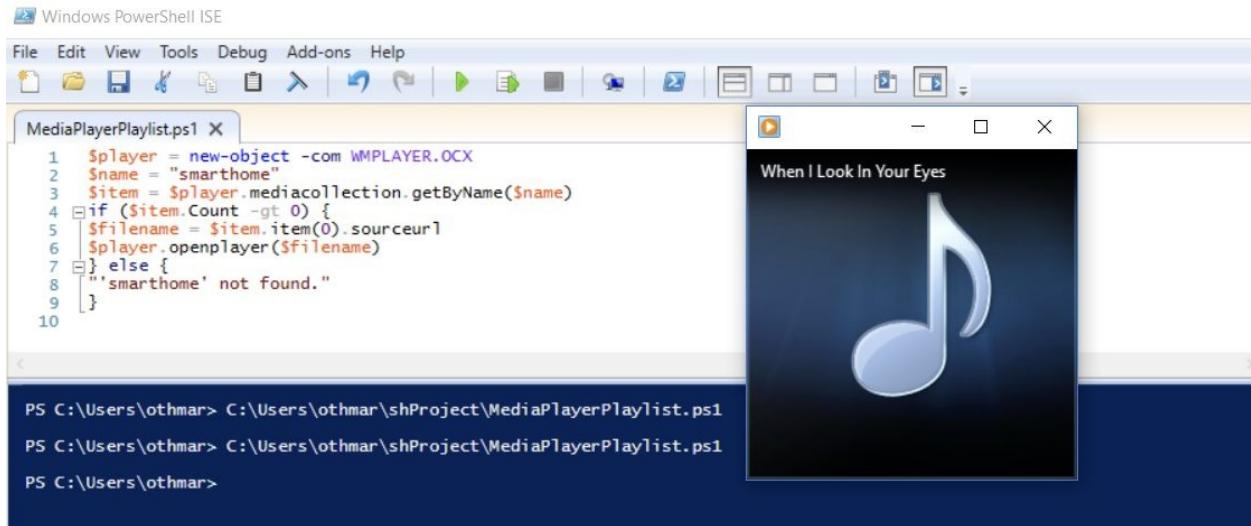


Figure 12.10 Controlling Windows Media Player using PowerShell under Windows 10

The command Write-Host, which displays text or variable values in the terminal window, is handy for troubleshooting scripts: Write-Host "Our playlist:"\$filename The command

Start-Sleep -Seconds 40

stops the execution of the PowerShell script for 40 seconds.

To get a listing of all processes running you can use the Get-Process command. You can close any process with the command get-process <processname> | %{\$_.closemainwindow() } .

To be more flexible in using our script, we add the parameter variable \$args[] in our script. This sets the playlist as a parameter when the script is executed. Further we add a validation at the start of the script which verifies, if the playlist parameter was provided along with the script command. We can now save our script under the name startMediaPlayer.ps1. Its full code is listed below: if
(((\$args.count -lt 1) -or (\$args.count -gt 1)){

```
echo „Argument error: startMediaPlayer.ps1 [Playlist]“
```

```
exit
```

```
}
```

```
$player = new-object -com WMPLAYER.OCX
```

```
$name = $args[0]
```

```
$item = $player.mediacollection.getByName($name) if ($item.Count -gt 0) {
```

```
$filename = $item.item(0).sourceurl
```

```
$player.openplayer($filename)
```

```
} else {
```

```
""$name' not found."
```

```
}
```

By typing

```
.\\startMediaPlayer.ps1 jazz in the PowerShell terminal window, we can now  
start the playlist smarthome. The script to stop Windows Media Player just  
consists of the single line: get-process wmplayer | % { $_.closemainwindow() }
```

We save it under the name stopMediaPlayer.ps1.

Controlling the system audio volume from PowerShell is a little more tricky. In order to use the [Audio]::Volume and [Audio]::Mute properties from PowerShell for PC audio control you need to use the Microsoft Core Audio API. For this copy the below code from Alexandre Jasmin in the PowerShell script window:
Add-Type -TypeDefinition @'

```
using System.Runtime.InteropServices; [Guid("5CDF2C82-841E-4546-9722-  
0CF74078229A"), InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
```

```
interface IAudioEndpointVolume {
```

```
// f(), g(), ... are unused COM method slots. Define these if you care int f(); int  
g(); int h(); int i();
```

```
int SetMasterVolumeLevelScalar(float fLevel, System.Guid  
pguidEventContext); int j();  
  
int GetMasterVolumeLevelScalar(out float pfLevel); int k(); int l(); int m(); int  
n();  
  
int SetMute([MarshalAs(UnmanagedType.Bool)] bool bMute, System.Guid  
pguidEventContext); int GetMute(out bool pbMute);
```

```
}
```

```
[Guid("D666063F-1587-4E43-81F1-B948E807363F"),  
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
```

```
interface IMMDDevice {
```

```
int Activate(ref System.Guid id, int clsCtx, int activationParams, out  
IAudioEndpointVolume aev); }
```

```
[Guid("A95664D2-9614-4F35-A746-DE8DB63617E6"),  
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
```

```
interface IMMDDeviceEnumerator {
```

```
int f(); // Unused
```

```
int GetDefaultAudioEndpoint(int dataFlow, int role, out IMMDevice endpoint);  
}
```

```
[ComImport, Guid("BCDE0395-E52F-467C-8E3D-C4579291692E")] class  
MMDeviceEnumeratorComObject { }
```

```
public class Audio {
```

```
    static IAudioEndpointVolume Vol() {
```

```
        var enumerator = new MMDeviceEnumeratorComObject() as  
IMMDeviceEnumerator; IMMDevice dev = null;
```

```
        Marshal.ThrowExceptionForHR(enumerator.GetDefaultAudioEndpoint(/*eRend  
0, /*eMultimedia*/ 1, out dev)); IAudioEndpointVolume epv = null;
```

```
        var epvid = typeof(IAudioEndpointVolume).GUID;  
        Marshal.ThrowExceptionForHR(dev.Activate(ref epvid, /*CLSID_ALL*/ 23,  
0, out epv)); return epv;
```

```
}
```

```
public static float Volume {  
  
    get {float v = -1;  
        Marshal.ThrowExceptionForHR(Vol().GetMasterVolumeLevelScalar(out v));  
        return v;}  
  
    set {Marshal.ThrowExceptionForHR(Vol().SetMasterVolumeLevelScalar(value,  
        System.Guid.Empty));}  
  
}  
  
public static bool Mute {  
  
    get { bool mute; Marshal.ThrowExceptionForHR(Vol().GetMute(out mute));  
        return mute; }  
  
    set { Marshal.ThrowExceptionForHR(Vol().SetMute(value,  
        System.Guid.Empty)); }  
  
}  
  
}
```

'@

Now you are able to control your PCs volume from PowerShell with the commands [Audio]::Volume (Check current volume) [Audio]::Mute = \$true (Mute speaker) [Audio]::Mute = \$false (Un-mute speaker) [Audio]::Volume = 0.75 (Set volume to 75%) This allows us to create the PowerShell scripts MediaPlayerVolUp.ps1, MediaPlayerVolDown.ps1, MediaPlayerMute.ps1 and MediaPlayerUnmute.ps1. Alexandre Jasmin's code needs to be at the beginning of each of the below scripts, to make them work:
MediaPlayerVolDown.ps1

..... code from abve ...

```
[Audio]::Volume
```

```
$level = [Audio]::Volume
```

```
if ($level -gt 0.1) {
```

```
$level = $level - 0.1
```

```
} else {
```

```
"MinVolume Reached"
```

```
}
```

```
[audio]::Volume = $level # 0.8 = 80%
```

```
[Audio]::Volume
```

```
MediaPlayerVolUp.ps1
```

```
..... code from abve ...
```

```
[Audio]::Volume
```

```
$level = [Audio]::Volume
```

```
if ($level -lt 0.9) {
```

```
$level = $level + 0.1
```

```
MediaPlayerMute.ps1
```

```
..... code from abve ...
```

```
[audio]::Mute = $true # Set to $false to un-mute MediaPlayerMute.ps1
```

..... code from abve ...

[audio]::Mute = \$false # Set to \$true to mute 12.5.2 Troubleshooting Tips

You might have to change the Powershell execution policy for your scripts to execute with the below command: Set-ExecutionPolicy -Scope CurrentUser RemoteSigned In addition keep in mind, that you always need to specify the full path to any script such as: C:\Users\smarthome\shProject\stopMediaPlayer.ps1

If you start a script from inside its storage location (for security reasons) you need to type a leading .\ (dot backslash) followed by the actual script name.

.\stopMediaPlayer.ps1

12.5.3 Starting Powershell Scripts from the Command Line In order to call our Powershell scripts from OpenRemote, we will start them with a terminal command without opening Powershell. This is done by typing powershell.exe followed by the full path to the script along with - if required - script parameters: powershell.exe

C:\Users\smarthome\shProject\startMediaPlayer.ps1 jazz All of the above code examples are part of the bonus material, which can be downloaded for free from <http://www.keyconceptpress.com/how-to-smarthome>. We now can move on to configure our Media Player remote control in OpenRemote Designer.

12.6 Script Based iTunes Control (Windows 10)

As MediaPlayer iTunes for Windows has a COM API as well, which can be used to control the iTunes functions. We start by creating the iTunes object using the Cmdlet New-Object, which can be used to create either COM or .NET objects and store in the variable \$iTunes: \$iTunes = New-Object -ComObject iTunes.Application Next we store the reference to the iTunes library name in \$libraryName and the reference to the selected playlist in \$selectedPlaylist: \$libraryName = \$iTunes.LibraryPlaylist.Name \$selectedPlaylist = \$iTunes.Sources.ItemByName(\$libraryName).Playlists.ItemByName('RadioClassical') Now we can play the playlist using the command: \$selectedPlaylist.PlayFirstTrack() To be more flexible in using our script, we use the parameter variable \$args[] to set the playlist as parameter when calling the script. Further we add a check at the start of the script which verifies whether the playlist parameter was provided with the script command. Our script iTunesStart.ps1 now reads as: if ((\$args.count -lt 1) -or (\$args.count -gt 1)){

```
echo „Argument error: startItunes.ps1 [Playlist]“
```

```
exit
```

```
}
```

```
$iTunes = New-Object -ComObject iTunes.Application $libraryName =  
$iTunes.LibraryPlaylist.Name $selectedPlaylist =  
$iTunes.Sources.ItemByName($libraryName).Playlists.ItemByName($args[0])  
$selectedPlaylist.PlayFirstTrack() As an example we can now start the playlist  
Relaxation by typing .\startItunes.ps1 Relaxation The commands to change the volume,  
to start and to stop iTunes are: $iTunes.Mute = $true
```

```
$iTunes.Mute = $false
```

```
$iTunes.Stop()
```

```
$iTunes.Play()
```

Further commands for even more functions would be: \$iTunes.Pause()

```
$iTunes.PlayPause() (toggle between play and pause)  
$iTunes.CurrentTrack.Name
```

```
$iTunes.NextTrack()
```

\$iTunes.PreviousTrack() A good collection of iTunes Powershell Cmdlets can be found on <http://www.thomasmaurer.ch/projects/powershell-itunes/>.

We can now write the Powershell scripts iTunesStart.ps1, iTunesStop.ps1, iTunesVolUp.ps1 and iTunesVolDown.ps1: iTunesStart.ps1

```
if ((($args.count -lt 1) -or ($args.count -gt 1)) {
```

```
echo „Argument error: startItunes.ps1 [Playlist]“
```

```
exit
```

```
}
```

```
$iTunes = New-Object -ComObject iTunes.Application $libraryName =  
$iTunes.LibraryPlaylist.Name $selectedPlaylist =  
$iTunes.Sources.ItemByName($libraryName).Playlists.ItemByName($args[0])  
$selectedPlaylist.PlayFirstTrack() iTunesStop.ps1
```

```
$iTunes = New-Object -ComObject iTunes.Application $iTunes.Stop()
```

```
iTunesVolUp.ps1
```

```
$iTunes = New-Object -ComObject iTunes.Application $iTunes.SoundVolume =  
$iTunes.SoundVolume + 2
```

```
iTunesVolDown.ps1
```

```
$iTunes = New-Object -ComObject iTunes.Application $iTunes.SoundVolume =  
$iTunes.SoundVolume - 2
```

```
iTunesSetVol.ps1
```

```
$iTunes = New-Object -ComObject iTunes.Application $iTunes.SoundVolume =  
$volume Since we also want to display the track that iTunes is playing after each  
control command, we would like to read out the active iTunes track and store it  
to the html log file playing.html. This file we will regularly poll through a sensor  
and display its content. For this purpose we will use the Powershell cmdlet set-
```

```
content: Set-Content -Value $iTunes.CurrentTrack.Name -Path .\ORC\webapps\controller\playing.html  
Specifying the playlist logfile as a script argument variable $args[0] we can now  
write our script iTunesPlaying.ps1: if (($args.count -lt 1) -or ($args.count -gt 1))  
{
```

```
echo „Argument error: iTunesPlaying.ps1 [Playlist logfile]“
```

```
exit
```

```
}
```

```
$iTunes = New-Object -ComObject iTunes.Application $currentTrack =  
$iTunes.CurrentTrack.Name $playlistlog = $args[0]
```

```
Set-Content -Value $currentTrack -Path .\ORC\webapps\controller\$playlistlog  
To test we start an iTunes playlist and type .\iTunesPlaying.ps1 playing.html in  
the Powershell window. If we now open the URL
```

<http://localhost:8080/controller/playing.html> in a web browser, we should see a display of the current track iTunes is playing. The implementation of the remote control user interface in OpenRemote Designer can now be done analog to the description in section 7.4.

12.7 Talk to Me

An important element of a modern smart home is that the communication between home and user is not restricted to computer GUIs, but also takes place at a human level such as speech. Thus, we will briefly cover how to implement speech output for our project.

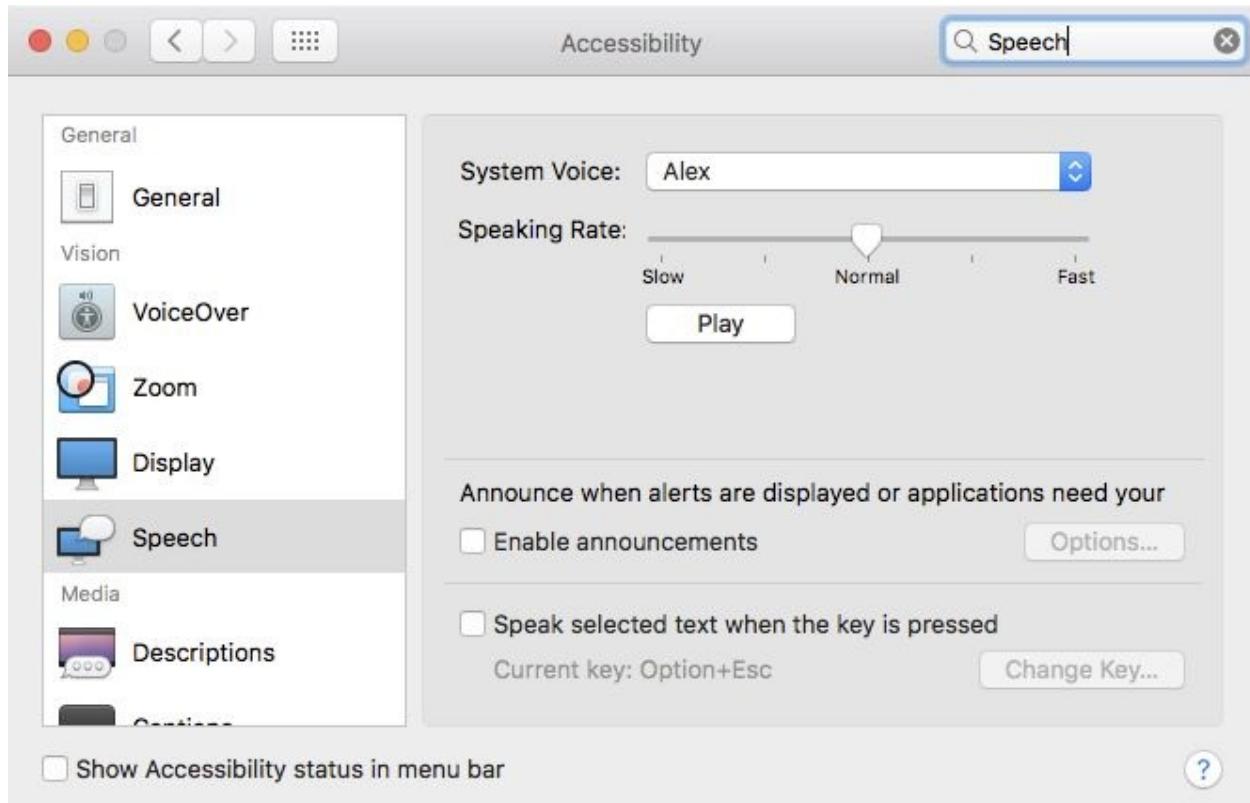
12.7.1 Speech Output Under macOS

Under macOS we can initiate speech output using the command line function say. Typing man say

gets us a display of the command options:

```
say [-v voice] [-r rate] [-o outfile [audio format options]] | -n name:port | -a device]
```

Per default, macOS comes with a number of English voices such as *Alex* or *Victoria*. However, you can easily expand the available voices to additional flavors and languages. Select *System Accessibility — Speech* and you can select between a variety of language and speaker options (Figure 12.11).



*Figure 12.11 macOS speech and language options menu In order to change the audio output for the speech to an external speaker, we could use the AppleScript commands, which change the audio settings in System Preferences. This time however we will use a "hard coded" command line function, which avoids the delay the AppleScript solution would introduce when executing. There are several functions available as free downloads in the Internet. We choose one called Switch AudioSource. (Internet search switchaudio-osx). After its download we unzip the file and copy the function Switch AudioSource to our directory shProject. Typing the command Switch AudioSource gets us a listing of its functions: Switch AudioSource
Please specify audio device.*

Usage: Switch AudioSource [-a] [-c] [-t type] [-n] -s device_name -a : shows all devices

-c : shows current device

-t type : device type (input/output/system). Defaults to output.

-n : cycles the audio device to the next one **-s device_name** : sets the audio device to the given device by name The **-a** option displays the available output options: Switch AudioSource -a

Built-in Input (input)

AirPlay (output)

Built-in Output (output) For our project we want to switch from built-in speakers to AirPlay and back. With the help of Switch AudioSource, we can now easily create the two simple shell scripts outputBuiltIn.sh and outputAirPlay.sh:
outputBuiltIn.sh

```
#!/bin/sh
```

```
#Switch audio output to Built-in Output Switch AudioSource -s "Built-in Output"
```

```
outputAirPlay.sh
```

```
#!/bin/sh
```

```
#Switch audio output to AirPlay
```

Switch AudioSource -s AirPlay As a note to the above: The Switch AudioSource parameter "Built-in Output" in the first script contains a space, which is why you need to put it in quotes. With these two shell scripts, we now create the OpenRemote commands Switch to AirPlay and Switch to Built-in Output. The say command also provides an option (-s) to choose the output, but this way we can create a generic speech shell script, with the output options available as

separate commands. We now create macSpeak.sh, which uses the voice Victoria and the parameter \$1 as the text for speech output: macSpeak.sh

```
#!/bin/sh
```

```
#Speaks the text given as parameter
```

```
if [[ $# -lt 1 || $# -gt 1 ]];then
```

```
echo "$0: Argument error: macSpeak.sh [text]"
```

```
exit 2
```

```
fi
```

```
say -v Victoria $1
```

In order to output the current time we create sayTime.sh. Here we simply use the date command, format its output to hours (%H) and minutes (%M), separated by a colon (this is the format the say command recognizes as time), and store it to a shell variable we call ctime (for current time). This is the format the say command recognizes as a time string.

```
sayTime.sh
```

```
#!/bin/sh
```

#Speaks the current time

```
ctime=$(date +"%H:%M")
```

```
say -v Victoria "It is $ctime"
```

As before we now create the corresponding OpenRemote shell protocol commands Say the Time and Good Morning. For Good Morning we call macSpeak.sh with the parameter Good morning. Time to get up. To test our commands we create a small OpenRemote macro called demo. The macro switches on our Denon AVR Zone 2

Sets the volume to 25dB

```
starts iTunes with the Playlist RadioJazz
```

```
sets the Audio Output to AirPlay
```

has Victoria wish us a good Morning

has Victoria tell us the time

Do not forget to insert delays between the commands, which allow the various commands to execute. In particular you need to allow Victoria to finish her first sentence, before asking her to tell us the time. (Figure 12.12). For the final version of your code, you should verify whether the first say command has

finished its execution before starting the next one.

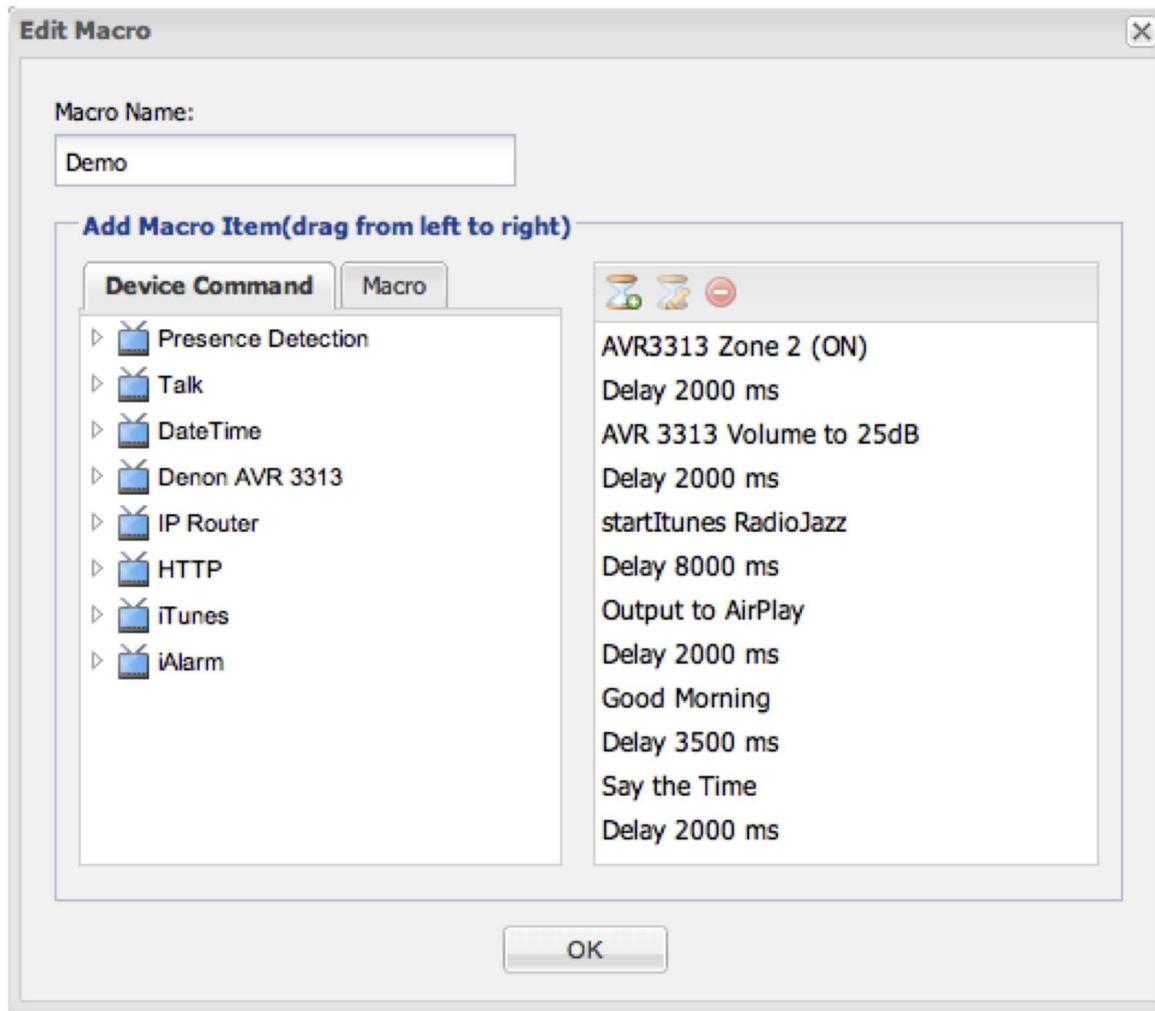


Figure 12.12 OpenRemote demo macro using speech commands In a similar manner you can create a shell script which outputs a text file to the say function. With that you can create voice output for the current weather forecast or the calendar entries of the day. As an example under macOS we can get a print out of the day's iCal events using the free utility icalBuddy from Ali Rantakari. (<http://hasseg.org/icalBuddy/>). With its number of options you can access events and tasks as shown below: USAGE: icalbuddy [options] <command> <command> specifies the general action icalBuddy should take: 'eventsToday' Print events occurring today 'eventsToday+NUM' Print events occurring

'between today and NUM days into the future 'eventsNow' Print events occurring at present time 'eventsFrom:START to:END' Print events occurring between the two specified dates 'uncompletedTasks' Print uncompleted tasks 'undatedUncompletedTasks' Print uncompleted tasks that have no due date 'tasksDueBefore:DATE' Print uncompleted tasks that are due before the given date 'calendars' Print all calendars

'strEncodings' Print all the possible string encodings 'editConfig' Open the configuration file for editing in a GUI editor 'editConfigCLI' Open the configuration file for editing in a CLI editor For voice output of today's events we just need to route the output of icalbuddy to the say command: icalbuddy eventsToday | say

We create a brief shell script sayCalendar.sh and an associated OpenRemote shell protocol command and can now use it as part of a rule or macro.

sayCalendar.sh

```
#!/bin/sh
```

```
#Voice output of todays events
```

```
icalbuddy eventsToday | say -v Victoria
```

12.7.2 Speech Output Under Windows

Under Windows Powershell we can use the object SAPI.SPVoice to turn text into speech output. As a first test we simply type the following commands in the Powershell window: \$Voice = new-object -com SAPI.SpVoice

```
$Voice.Speak( "Hello World!", 1 )
```

If you change the parameter 1 to 5, you can have Powershell read out a text file. Create a text file and have Powershell read it to you: \$Voice = new-object -com SAPI.SpVoice

\$Voice.Speak("C:\Text\smartHome.txt", 5) First we want to create a generic script, that reads out the text provided with the script as parameter. As we have done before, we start with the validation that exactly one parameter was provided when calling the script. Then we define the object SAPI.SPVoice and call \$Voice.Speak with the content of \$args[0]: pcSpeak.ps1

```
if (($args.count -lt 1) -or ($args.count -gt 1)){
```

```
echo "Argument error: psSpeak.ps1 [text for speech output]"
```

```
exit
```

```
}
```

```
$Voice = new-object -com SAPI.SpVoice
```

```
$Voice.Speak( $args[0], 1 )
```

We call the script pcSpeak.ps1 and can call it now with any text: ./pcSpeak.ps1 "Hello my name is John"

As a second script we want to output the current time. Since we just want hours and minutes, we format the Get-Date Cmdlet using the %h and %m options: \$a = Get-Date -format %h %m

With that we can finish our script sayTime.ps1 as follows: sayTime.ps1

```
$cTime = Get-Date -format "%h %m"
```

```
$Voice = new-object -com SAPI.SpVoice
```

```
$Voice.Speak( "It is", 2 )
```

```
$Voice.Speak( "$cTime", 2 )
```

In order to start our Powershell scripts from OpenRemote we will again start them as command line without opening Powershell, which is done by typing powershell.exe followed by the full path to the script and the script parameters: powershell.exe C:\Users\smarthome\shProject\sayTime.ps1

In OpenRemote Designer we can now create the command Say Time, selecting Shell Execution Protocol as the communication protocol. In the Path field, we enter powershell.exe and in the Command parameter field, we enter C:\Users\smarthome\shProject\sayTime.ps1.

As a first text-to-voice command we create the command Good Morning. In the Path field we enter powershell.exe. In the Command parameter field we enter C:\Users\smarthome\shProject\pcSpeak.ps1 followed by "Good Morning, time to get up", separated by a space. To test our commands we create a small OpenRemote macro which starts iTunes followed by the speech output commands Good Morning and sayTime.

To configure the text-to-voice settings under Windows 10 you go to [*Settings — Time & Language — Speech*](#). Per default Windows 10 comes with the two voices Marc Mobile and Zira Mobile. a single voice.

13 A Little AI: Drools Rules

In this chapter we will explain how to set up rules for our OpenRemote controller, which is the always-on home automation component of OpenRemote. So far we control the sensors and applications, which we have developed in the previous chapters, using our smartphone application. We now will control them with a rule engine from a stationary computer, and will be able to implement powerful home automation scenarios such iAlarm or Coming Home. A few years ago, rule based expert systems were hyped up and referred to as artificial intelligence. While this is far from reality, it is still surprising, how well a small set of well thought through rules can perform in defined environments. For our smart home project the rules database is the brain, where the building automation intelligence resides, and where defined actions based on detected events are taken. The rules infrastructure, which OpenRemote uses, is called Drools, an open source object oriented rule engine written in Java. Besides in Java, rules however can also be specified in MVEL, Python or Groovy.

A rule engine is basically nothing but an if/then statement interpreter. Each if/then statement is called a rule. The commercial, productized version of Drools is called JBoss Rules from RedHat. Details can be found under <http://www.jboss.org/drools/>.

The Drools syntax is relatively simple. The rules are stored in files with the extension .drl. In OpenRemote per default rules from the OpenRemote Professional Designer rule editor are stored in the subdirectory/webapps/controller/rules/modeler_rules.drl of the OpenRemote installation.

The basic structure of a rule definition is: rule "name"

attributes

when

LHS

then

RHS

end

LHS stands for left hand side (the if part of an if/then rule), RHS stands for right hand side (the then part of an if/then rule). There are a number of keywords, which are reserved as Drools commands, and which must not be used as names or identifiers. Some of them are: true, false, null, eval, when, then, end, not, or, end A key role in Drools play rule attributes. They are optional and can be used to control the behavior of the rule. Examples for attributes are timer or no-loop. LHS (Left Hand Side) is the conditional part of the rule, RHS (Right Hand Side) contains the commands to be executed in case LHS becomes true. When LHS is left empty, the rule is assumed to be always true. Multiple conditions without a logic operator are treated as and-conditions. Or-conditions are separated with an or operator.

All elements of the basic rule structure described above, including the rule name, are mandatory for the rule to work. Single-line comments are marked using double backslashes //, multi-line comments are marked using /* and */. The full specification for the Drools language can be obtained from <http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch05.html>

13.1 Wake me up Early if it Rains: iAlarm

As our first rule we will be creating an intelligent alarm, which we call „Wake me up early if it rains“, and for which we will use the weather sensor from chapter 6. Our rule shall execute the following actions:

- at 6 a.m. retrieve the current weather from our OpenRemote sensor Weather Condition Berlin
- at 6 a.m. retrieve the current value (on/off) from ialarm.txt
- determine if the weather sensor value contains the text strings rain or snow
- in case the string rain or snow is found and ialarm.txt contains the value on, start iTunes with playlist RadioPop, update the playlist logfile playing.txt and set ialarm.txt back to the value off
- In case it does not rain or snow, the alarm will go off at 6h45 a.m.

13.1.1 Controlling iAlarm via Smartphone

Before we move on to design the rule for our intelligent alarm, we want to set up the capability to switch our iAlarm on and off from our smartphone with the help of the status file ialarm.txt macOS and Linux users create the shell script ialarmswitch.sh, Windows users create the two Powershell scripts turniAlarmOn.ps1 and turniAlarmOff.ps1: ialarmswitch.sh

```
#!/bin/sh
```

```
echo $1 > Userssmarthome/shProject/shconfig/ialarm.txt turniAlarmOn.ps1
```

```
echo "on" > C: echo $1 > Userssmarthome/shProject/ialarm.txt
```

turniAlarmOff.ps1

```
echo "off" > C: echo $1 > Userssmarthome/shProject/ialarm.txt
```

The echo command (under Mac shell as well as Microsoft Powershell) writes a text string to a file in case it exists. If the file does not exist, it creates the file containing the text string. The shell command ialarmswitch.sh needs to be executed with the command parameter on or off.

As discussed in section 7.3.1 be aware that OpenRemote executes shell scripts from its binary directory ORC/bin, independent of where the shell script actually is located at, which in our case is shProject/shconfig. Since we want the log file to also reside in shProject/shconfig, we need to specify the full absolute path for our logfile ialarm.txt, which is *Userssmarthome/shProject/ialarm.txt*. Do also not forget, that you have to enable the execution rights of the new shell script files: chmod +x *.Userssmarthome/shProject/shconfig/ialarmswitch.sh*. In OpenRemote Designer we now create a new device called iAlarm and define the two commands TurniAlarmOn, TurniAlarmOff and iAlarmStatus. On a Mac for the commands TurniAlarmOn and TurniAlarmOff we select *Shell Execution Protocol*, for *Path* the full path to ialarmswitch.sh and as *Command parameter* the values on respectively off (Figure 8.1).

Under Windows we select *Shell Execution Protocol* as well, in the *Path* field we enter powershell.exe and as *Command parameter* we enter the actual path to our Powershell script: C:\Users\smarthome\shProject\shconfig\turniAlarmOff.ps1

C:\Users\smarthome\shProject\shconfig\turniAlarmOn.ps1

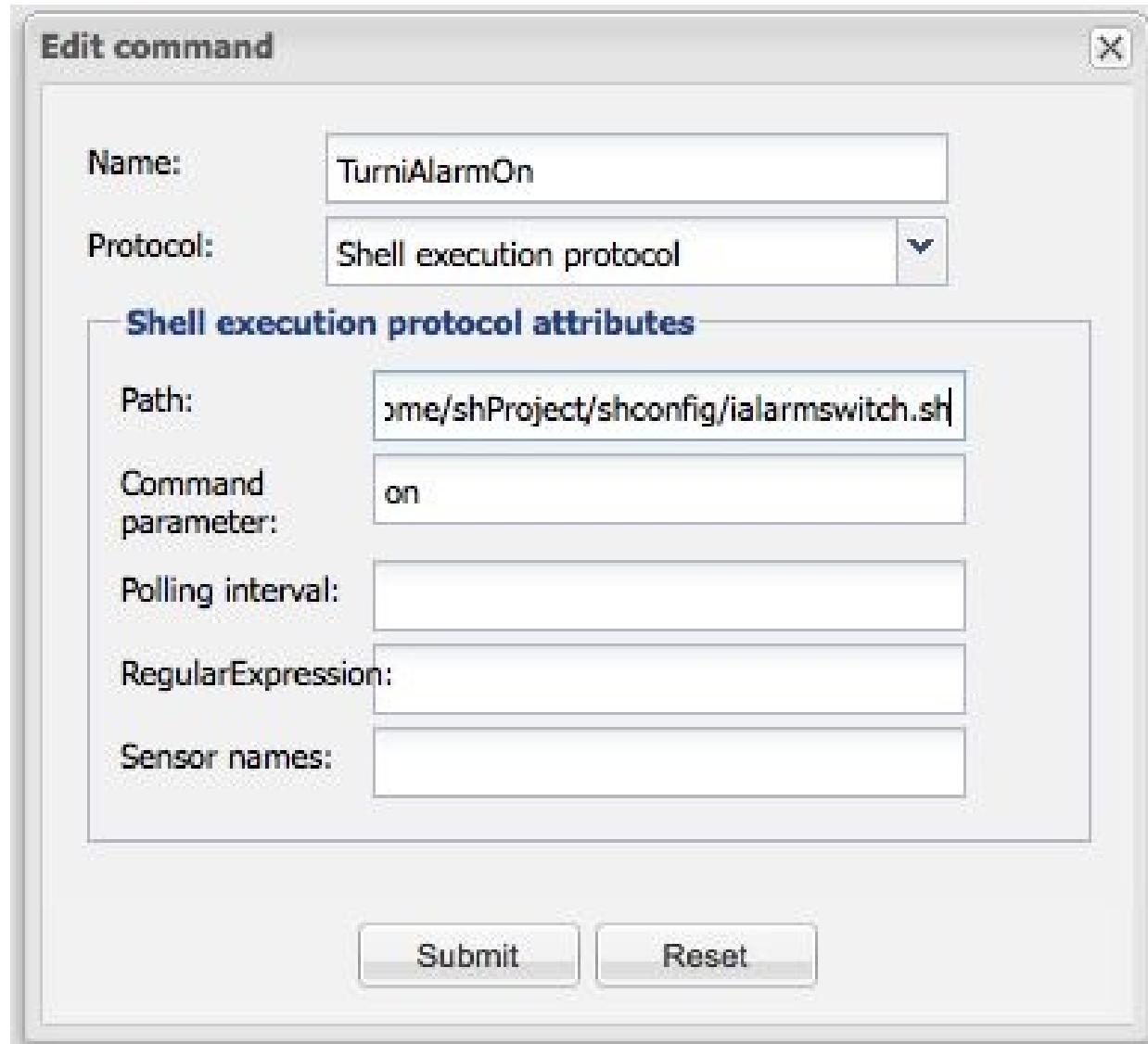


Figure 13.1 OpenRemote command definition for TurniAlarmOn for macOS

For the command iAlarm Status we also select in the field **Protocol** the **Shell execution protocol**. As **Polling interval** we select 3s (three seconds). I recommend always to select the maximum polling interval which fulfills the required functionality. If you end up with large numbers of sensors polling shell scripts, HTTP requests or other entities in small intervals such as one second or below, the overall performance of your system will start being impacted.

well Sthe HTTP protocol, provide the URL to the local OpenRemote web server for the file iAlarmfunction.html in the URL field
<http://localhost:8080/controller/iAlarmfunction.html> and select GET for the field **HTTP Method**. (Figure 13.2).

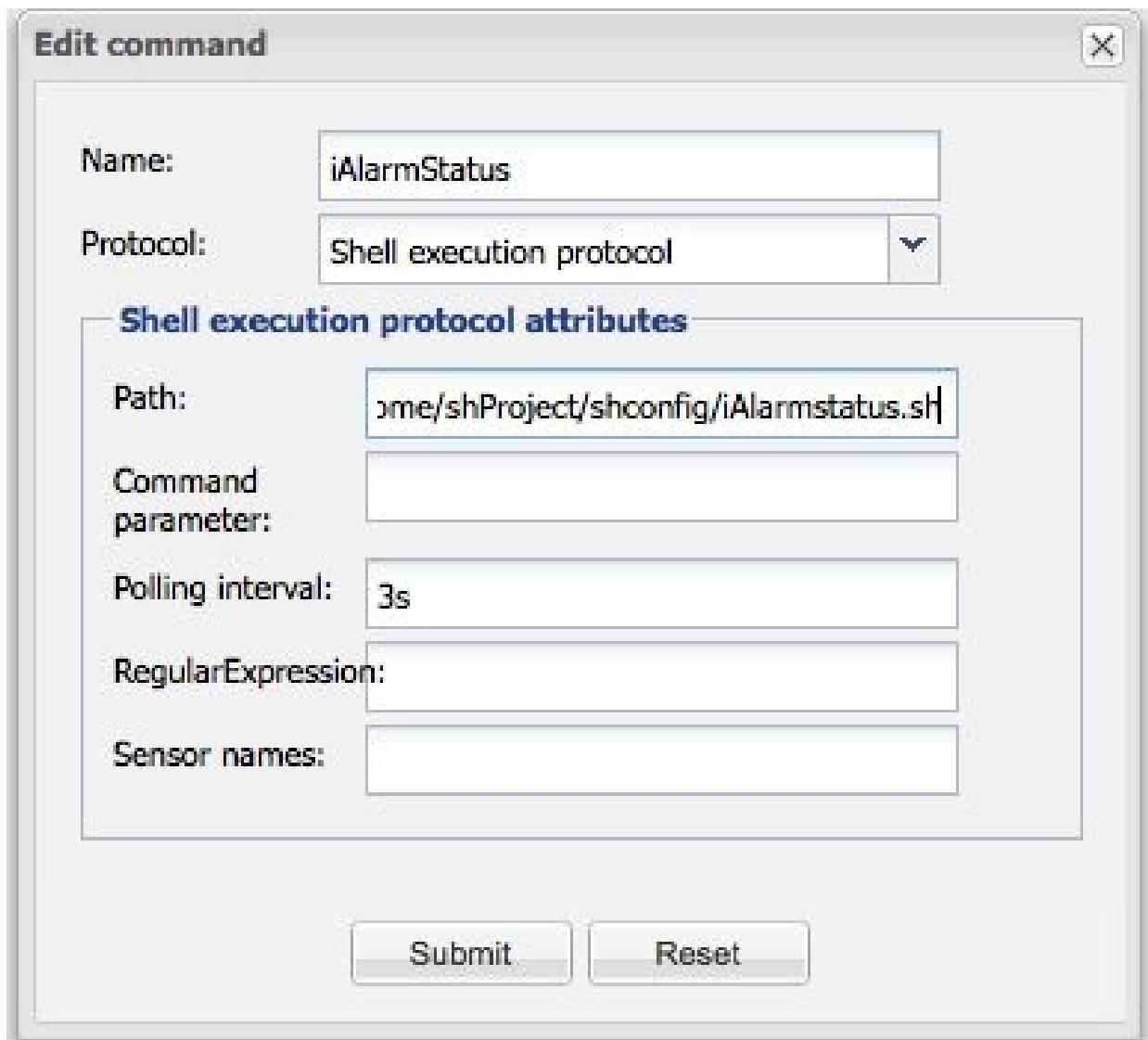


Figure 13.2 OpenRemote command definition for iAlarm Status using the shell execution protocol Now we can create the sensor iAlarm using the command iAlarmStatus we just defined. (Figure 13.3)

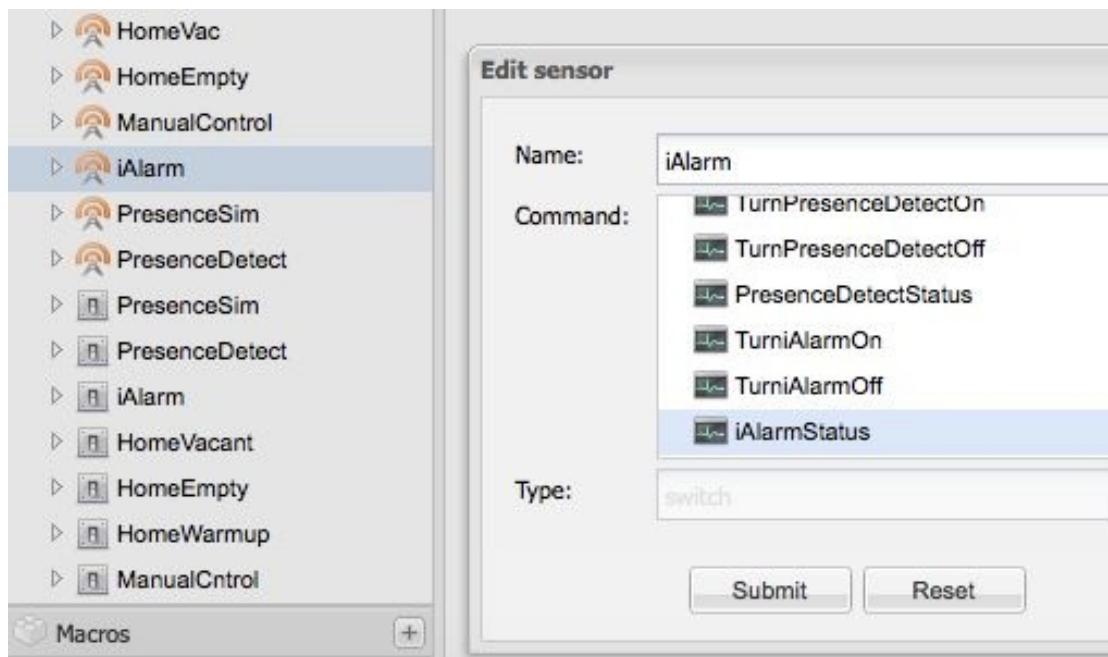


Figure 13.3 OpenRemote Definition for the iAlarm sensor As the last step we define an iAlarm switch, using the iAlarm sensor and the two iAlarm on / off commands. We select [New — New Switch](#) and select the iAlarm sensor for [sensor](#) and TurniAlarmOn and TurniAlarmOff for [command\(on\)](#) and [command\(off\)](#) (Figure 13.4).

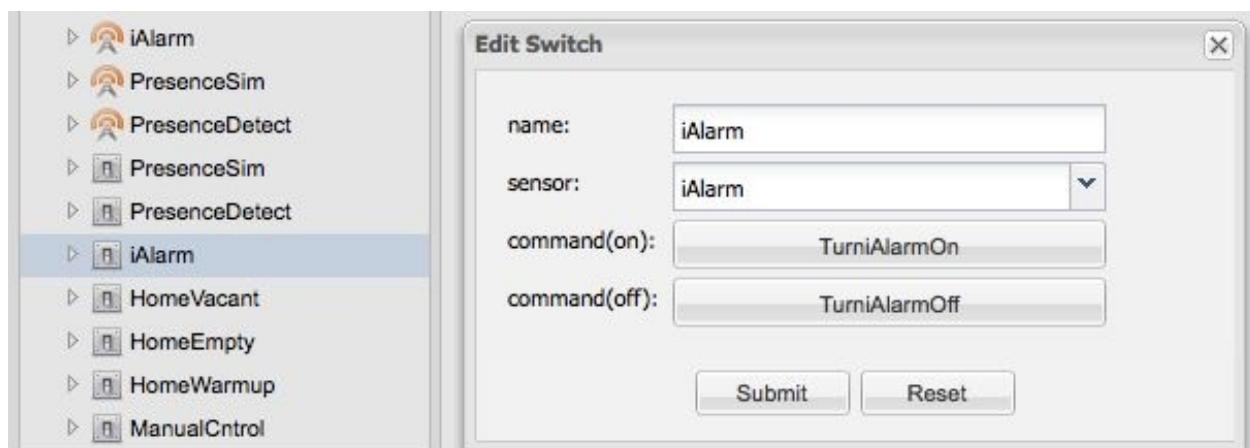


Figure 13.4 OpenRemote Definition for the iAlarm switch Now we add the GUI

controls for the iAlarm switch. We add the necessary grid elements and insert an abc label element for the label and a switch element for the iAlarm switch. As Switch Command we select the switch command iAlarm we just defined. After synchronizing our local controller with the updated design, we can toggle our iAlarm to on and off (Figure 13.5).

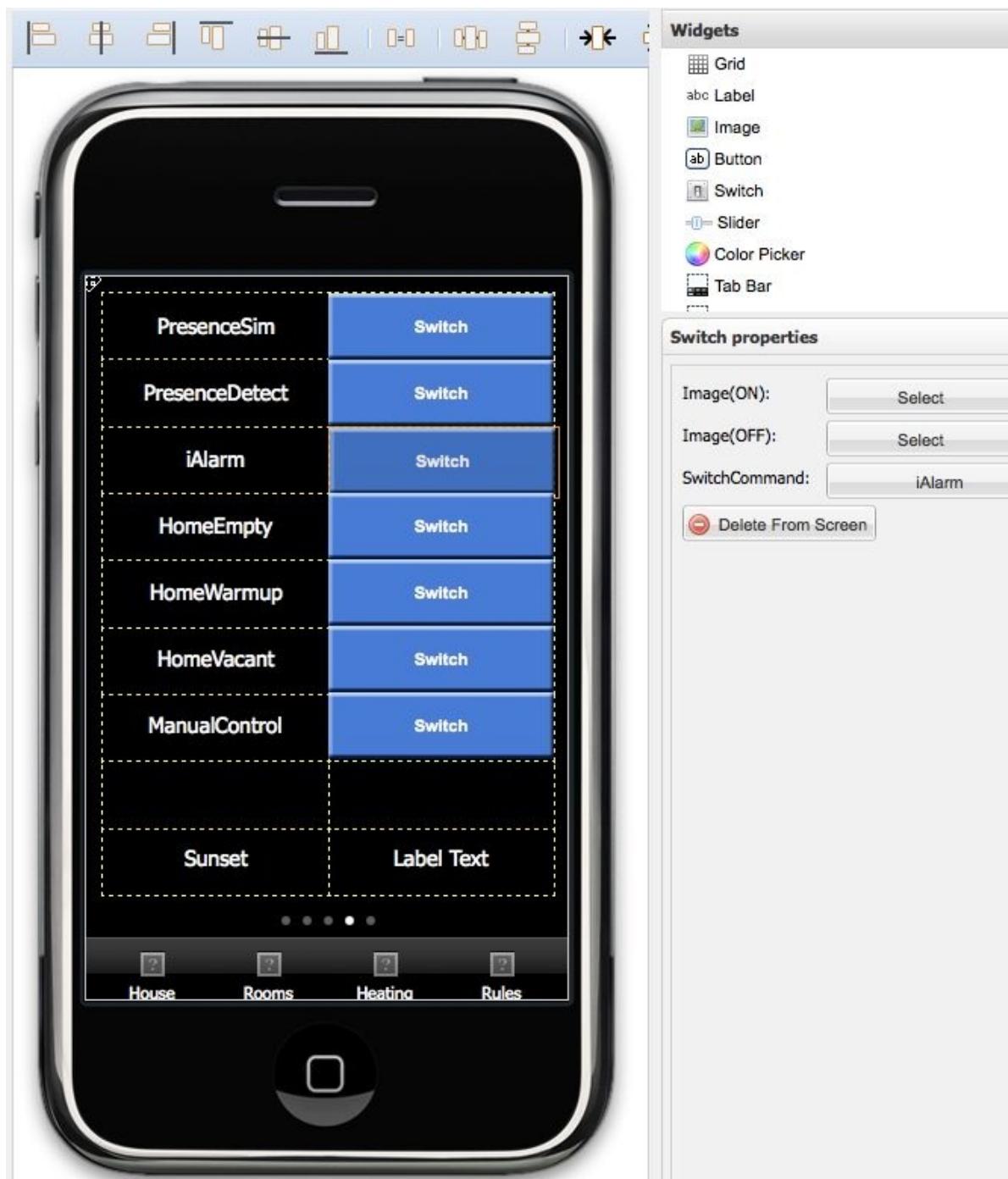


Figure 13.5 OpenRemote GUI design for iAlarm and other smart home controls
13.1.2 The iAlarm Rule Script

We can now begin the design of our iAlarm rules script. We start with the name

we can now design the design of our timer rules script. we start with the main statement followed by a timer, which shall kick off the rule at six a.m.. In Drools the timer uses the format of UNIX cron expressions, which consist of six mandatory fields and the optional year field, each separated by a white space: timer (cron: <seconds> <minutes> <hours> <day-of-month> <month> <day-of-week> [year]) Valid values are

seconds: 0-59 or * for any value

minutes: 0-59 or * for any value

hours: 0-23 or *

day-of-month: 1-31, L,?,*

month: 1-12 or JAN-DEC

day-of-week: 1-7 or SUN-SAT

Year: 1970-2199

The value L stands for last day of month, * stands for any value and ? stands for no value. If any value other than ? is specified in day-of-month, a ? must be chosen for the day-of-week field. Similarly, when a day-of-week value is specified, the day-of-month field must contain ?. Day and month ranges can be specified using the hyphen character (-), minute increments with the backslash character (/). We want our rule to execute every Monday through Friday at 6 a.m. which gets us to the timer expression timer (cron: 0 0 6 ? * MON-FRI)

With the CustomState command we can execute OpenRemote sensors. With the command name : value we write the value of the sensor to the variable name. In the then part of our rule we simply print out the content of our variable name followed by current date and time: (1) //Rule reading out weather sensor at 6a.m. and starting iTunes in case of rain or snow//

(2) rule "Wake me up early if it rains"

(3) timer (cron: 0 35 17 ? * MON-FRI)

(4) when

(5) CustomState(source == "Weather Condition Berlin", name : value) (6) then

(7) System.out.println("name");

(8) Date date = new Date();

(9) System.out.println(date.toString());

(10) end

(1) Single-line Comment

(2) The name of our rule

(3) Timer which stops the script here until the time condition is met (4) Begin of the rule if condition: when

(5) Read out OpenRemote sensor “Weather Condition Berlin”, storage of its content into variable name (6) Begin of the rule then condition: then

(7) Print out the content of variable name

(8) Write current date and time to variable date (9) Print out the content of variable date

(10) End

We now want to test what we have so far. In OpenRemote Designer on the left hand side of the screen we expand *Config for Controller* and select the menu entry *rules*. We are now in the OpenRemote rules editor. In addition to the above script we need to insert the import commands for several OpenRemote scripts and Java packages. We just need to do this once at the beginning of our rules definition file. So insert the following definitions at the very beginning of the file, followed by our first script in the rules editor window (Figure 13.6):

//Package, globals and imports:

```
package org.openremote.controller.protocol
```

```
global org.openremote.controller.statuscache.CommandFacade execute; global  
org.openremote.controller.statuscache.SwitchFacade switches; global  
org.openremote.controller.statuscache.LevelFacade levels; import  
org.openremote.controller.protocol.*;
```

```
import org.openremote.controller.model.event.*; import java.lang.Float;  
  
import java.sql.Timestamp;  
  
import java.util.Date;
```

rule „Wake me up early if it rains“

timer (cron: 0 35 17 ? * MON-FRI)

when

CustomState(source == „Weather Condition Berlin“, name : value) then

```
System.out.println(„name“);
```

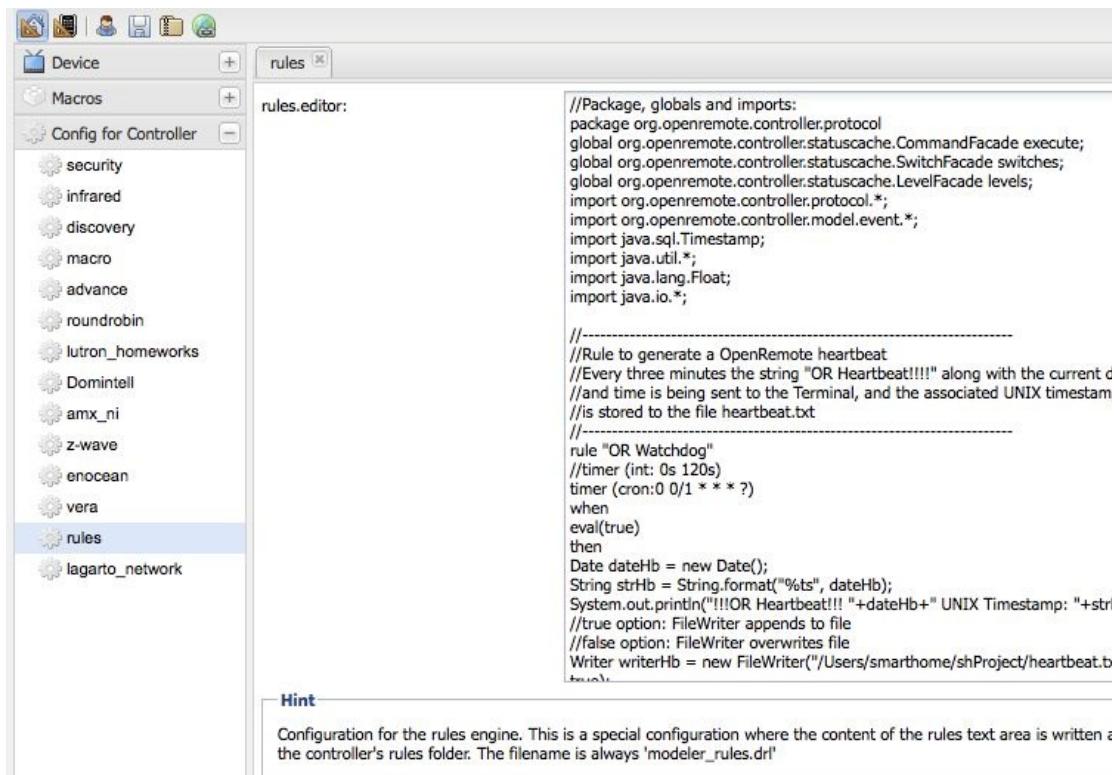
```
Date date = new Date();
```

```
System.out.println(date.toString());
```

end

After inserting the rule definition and the import packages click on the *submit* button twice and you will receive the confirmation *Property saved successfully*. Next we go to the UI Designer window and save our designer project by clicking on the *disc* symbol. We get the message *UI Designer Layout saved at* Then we synchronize our local rules definition file with our Online Designer project by clicking *Sync with Online Designer* in the OpenRemote Controller window. Now our local rules definition file

.../ORC/webapps/controller/rules/modeler_rules.drl is updated. We open it with a text editor to validate that our rule definition actually has loaded. When looking for the source of a problem related to rules definitions it is always a good idea to check the content of your local modeler_rules.drl file, since this is, what actually gets executed. Due to an incomplete synchronization process it can happen, that your local file is actually different than your latest rule definition in OpenRemote Designer. At the very bottom of the OpenRemote GUI you can also monitor the saving progress.



The screenshot shows the OpenRemote Controller interface with the 'rules' configuration selected in the left sidebar. The main area displays the contents of the 'modeler_rules.drl' file:

```

rules.editor:
//Package, globals and imports:
package org.openremote.controller.protocol;
global org.openremote.controller.statuscache.CommandFacade execute;
global org.openremote.controller.statuscache.SwitchFacade switches;
global org.openremote.controller.statuscache.LevelFacade levels;
import org.openremote.controller.protocol.*;
import org.openremote.controller.model.event.*;
import java.sql.Timestamp;
import java.util.*;
import java.lang.Float;
import java.io.*;

//-----
//Rule to generate a OpenRemote heartbeat
//Every three minutes the string "OR Heartbeat!!!!" along with the current date and time is being sent to the Terminal, and the associated UNIX timestamp is stored to the file heartbeat.txt
//-----
rule "OR Watchdog"
//timer (int: 0s 120s)
timer (cron:0 0/1 * * * ?)
when
eval(true)
then
Date dateHb = new Date();
String strHb = String.format("%ts", dateHb);
System.out.println("!!!OR Heartbeat!!! "+dateHb+" UNIX Timestamp: "+strHb);
//true option: FileWriter appends to file
//false option: FileWriter overwrites file
Writer writerHb = new FileWriter("/Users/smarthome/shProject/heartbeat.txt");
writerHb.write(strHb);
writerHb.close();

```

Hint

Configuration for the rules engine. This is a special configuration where the content of the rules text area is written to the controller's rules folder. The filename is always 'modeler_rules.drl'

Figure 13.6 OpenRemote rules definition editor When the OpenRemote controller now loads, observe closely the control messages in the terminal window. At the very beginning you will now see a line similar to INFO 2013-05-09 14:54:13,328 : Initialized event processor : Drools Rule Engine This tells us, that the Drools definition file has been parsed correctly without any error. In case the parser finds an error, this is where you will find the according error messages. In order to test our rule now, we need to change the time in the timer definition of our rule to a time two or three minutes ahead of our current time. Then we synchronize our rules definition again and observe the terminal window of our controller. After startup our rule should print the current weather condition from our weather sensor followed by time and date in the controller window, once the time specified in our timer definition has passed. We should see something like: INFO 2013-05-09 17:35:54,915 : Startup complete.

Partly Cloudy

Thu May 09 17:37:00 CEST 2013

We now know our rule is working and can proceed. First, on the conditional part of our rule, we need to validate if iAlarm is switched on. This is done by determining if the value of our custom sensor iAlarm Status is on:
CustomState(source == "iAlarm Status", value == "on") In general, depending on what type of sensor you have defined (custom, range, level, switch), you can reference your sensors with the four commands custom state

range

level

switch

Valid values for switch sensors are on and off, for level and range sensors any integer, and for custom sensors any arbitrary string. The event value is the value the sensor must report for the rule to be triggered. Examples are Range (source == " Livingroom " , value == " 25 ") Level (source == " brightness " , value == " 15 ") Switch (source == " OutdoorLight " , value == " on ") value != minValue, value != maxValue

When the value of a sensor changes, an event is triggered and sent to the rule engine. You can also store the value of the sensor to a variable, and process it later in the rule. In the below example the value of the level sensor brightness is stored to the variable lamp01: Level (source == " brightness " , lamp01 : value) Since for combining several rule elements the most common condition is a logical and, it is implicit on the LHS side of a Drools rules definition. This means we simply need to insert the above line below our first Event command, and both events are logically connected through an and condition. For our iAlarm rule we further need to conduct a search substring operation for the occurrence of rain or snow in the output of our weather sensor. For this purpose we use the Java command matches for regular expression based string searches. We define the string variable testStr, which we assign the content of our variable name (it holds the value of our weather sensor) and the string variables lookUp1 and lookUp2, which we assign our search terms rain and snow. The regular expression search for our two substrings shall be case insensitive, which is why our regex definitions have to start with (?i): (?i).*rain.*

(?i).*snow.*

The asterisk (*) stands for any number, the dot (.) for any character except new line. With that our Java commands for the substring search of our test string testStr, using the two substrings in variable lookUp1 and lookUp2, read:

testStr.matches("(?i).*"+lookUp1+".*") testStr.matches("(?i).*"+lookUp2+".*") We combine both substring searches in an if command, connecting them with a logical or, which in Java is the double pipe ||: rule "Wake me up early if it rains"

timer (cron: 0 41 18 ? * MON-FRI)

when

```
CustomState(source == "Weather Condition Berlin", name : value)
CustomState(source == "iAlarm Status", value == "on") then
```

```
String testStr = (String) name;
```

```
String lookUp1 = "rain"; String lookUp2 = "snow"; if ((testStr.matches("(?i).*"+lookUp1+".*")) || (testStr.matches("(?i).*"+lookUp2+".*"))){
```

```
System.out.println("iAlarm going off "+name); Date date = new Date();
```

```
System.out.println(date.toString());
```

```
}
```

end

We can now test our rule definition again. In order to get an alarm, we again need to adjust the time in our timer definition to a few minutes ahead of us, and we need to replace one substring search (e.g. snow) by a keyword, which

currently is being displayed by our weather sensor, e.g. cloudy. (And in case you are working on a Saturday or Sunday, adjust the day-of-week statement to MON-SUN). If the above is working we are almost done. In addition to the print statement, which we can leave in as logging information, we just need to add the commands to start iTunes (startItunes RadioPop), to update the playlist log file playing.html (iTunesPlaying) and to switch our alarm function off (Turn iAlarm Off). The command execution uses the format execute.command("OpenRemote command name"); Since OpenRemote macros are not supported to be called from the rules environment, we cannot just call our macro Pop, which we defined to contain the two commands startItunes RadioPop and iTunesPlaying with a five second delay timer in between. (The timer was needed to give iTunes the chance to load the playlist, before updating the playlist log file). Instead we need to call the two commands within our rule and insert a Java program delay by using the Thread.sleep command, which causes our Java thread to suspend execution for a specified period. When using Thread.sleep we also need to deal with the so called InterruptedException, which is why the Java code we need to insert for the delay is exactly as follows: try {

```
Thread.sleep(5000);
```

```
} catch(InterruptedException ex) {
```

```
    Thread.currentThread().interrupt();
```

```
}
```

We can now finish our first rule script, which reads in its final version to: rule “Wake me up early if it rains”

timer (cron: 0 0 6 ? * MON-FRI)

when

```
CustomState(source == "Weather Condition Berlin", name : value)
CustomState(source == "iAlarm Status", value == "on") then
```

```
String testStr = (String) name;
```

```
String lookUp1 = "rain"; String lookUp2 = "snow"; if ((testStr.matches("(?i).*"+lookUp1+".*")) || (testStr.matches("(?i).*"+lookUp2+".*"))) {
```

```
System.out.println("iAlarm going off "+name); Date date = new Date();
```

```
System.out.println(date.toString());
```

```
execute.command("startItunes RadioPop"); execute.command("Turn iAlarm Off"); try {
```

```
Thread.sleep(5000);
```

```
} catch(InterruptedException ex) {
```

```
Thread.currentThread().interrupt();
```

```
}
```

```
execute.command("iTunesPlaying"); }
```

```
end
```

What is left is the second simple rule, which should start iTunes at 6h45a.m., in case the alarm function is still on (which means, iAlarm did not go off at 6.a.m.): rule "Wake me at 6h45"

```
timer (cron: 0 45 6 ? * MON-FRI)
```

```
when
```

```
CustomState(source == "iAlarm Status", value == "on") then
```

```
System.out.println("iAlarm going off"); Date date = new Date();
```

```
System.out.println(date.toString());
```

```
execute.command("startItunes RadioPop"); execute.command("Turn iAlarm Off"); try {
```

```
Thread.sleep(5000);
```

```
    } catch(InterruptedException ex) {  
  
        Thread.currentThread().interrupt();  
  
    }  
  
execute.command("iTunesPlaying"); end
```

Once you have understood the above rules and got them working, you will be able to easily expand them, or build others using a similar structure. Many rules will also be simpler than the ones above, and just execute one or two commands based on a simple on/off condition of a sensor.

13.2 Remember me: Maintaining State Information

The approach we have chosen to switch iAlarm on and off by writing its state information to a configuration file (ialarm.txt), which iAlarm uses to determine whether it should become active or not, can be used in general, to manage state information for our smart home project. Even if the controller software reloads or our smart home server restarts, the information saved in such information state files is being maintained, and the application can continue operation with the same settings as before the incident. Such files can also be used to simulate global variables, which communicate state information across applications. In our project we do this by defining the following additional state information files:

- presencesim.txt
- presencedetect.txt
- homeempty.txt
- homewarmup.txt
- homevacant.txt and
- manualcontrol.txt

With simple shell scripts and the command parameters on / off we will assign the values on or off to these files. With a second set of shell scripts, using the cat command, we will read out the values of each of these files and display them with a sensor on the smart home app. This will then allow us to create on/off switches for our smartphone GUI and Drools rules, which set lights and the heating system of our house to the scenarios of an empty home (empty during the day), a vacant home (vacant for two or more days), warmup (activation of heating without night savings to bring up temperature in a vacant home) and presence simulation (light activity to simulate presence). It further will allow us to deactivate all rules by setting the value of manualcontrol.txt to on. And finally we are able activate and deactivate smart home functions such as iAlarm and

presence detection.

Below the simple shell script, which we store in our /shConfig directory, and which assigns the strings on respectively off to the file homeempty.txt.

homeemptyswitch.sh

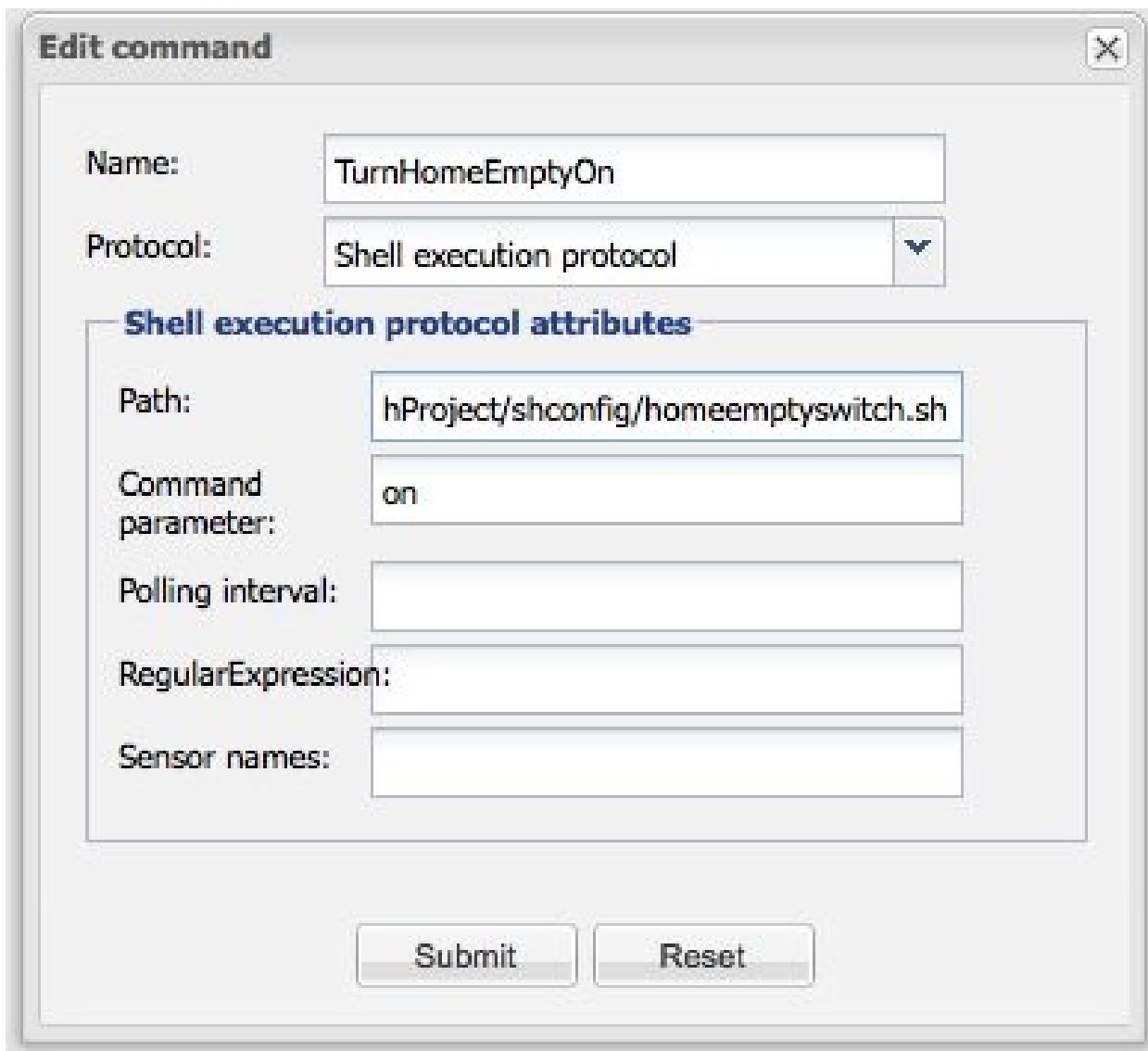
```
#!/bin/sh
```

echo \$1 > *Userssmarthome/shProject/homeempty.txt* And here the shell script homeemptystatus.sh, which reads out the content of homeempty.txt by using the cat command: homeemptystatus.sh

```
#!/bin/sh
```

cat *Userssmarthome/shProject/homeempty.txt* Important! Make sure to insert the file names along with the full absolute path definition as shown in the source code above. OpenRemote executes the shell scripts from its bin directory, independent of where the shell script sources reside. In our case this is shProject/ORC/bin. Since we want our state information files to reside outside of the ORC code directory tree in shProject/shConfig, with the full absolute path *Userssmarthome/shProject/* we ensure the shell process to locate the file it needs to execute.

Now in OpenRemote Professional Designer we create the two commands TurnHomeEmptyOn and TurnHomeEmptyOff, using the shell script homeemptyswitch.sh. For the field *Protocol* we select *Shell execution protocol*, in the field *Path* we enter the full path to the location of our shell scripts in /shConfig, and as *Command parameter* we use on and off (Figure 13.6).



*Figure 13.6 Information state command: Setting homeempty.txt to on Then we create the command HomeEmptyStatus, which reads out the value of homeempty.txt. For that purpose we create the command HomeEmptyStatus using as well the **Shell execution protocol** as **Protocol**, and inserting the full path to our shell script in **Path**. In addition we now need to select a polling interval. In our case we select 5s, which stands for five seconds. When entering a value without specifying a unit, OpenRemote defaults the value to ms. Make sure to select a value of 1 second (1000ms) or above, in order to allow sufficient processing time for your server, considering that you might have several polling*

*commands active concurrently (Figure 13.7). Next we create a sensor command using our command HomeEmptyStatus, choosing type **custom** and entering the custom state items on (Name), on (Value), off (Name) and off (Value) (Figure 13.8). Only then the polling function of our command HomeEmptyStatus is actually activated.*

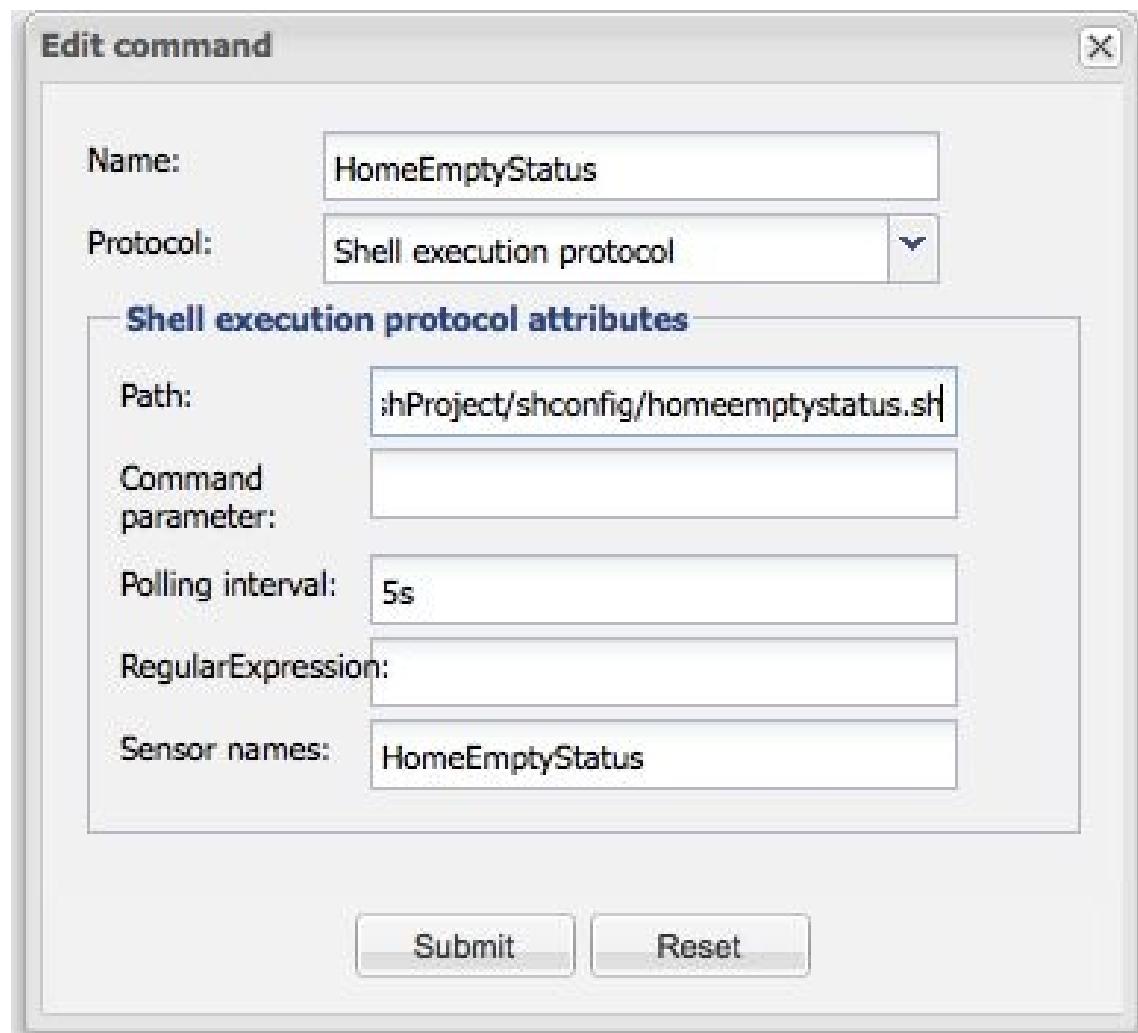


Figure 13.7 Command to periodically read out the value of homeempty.txt

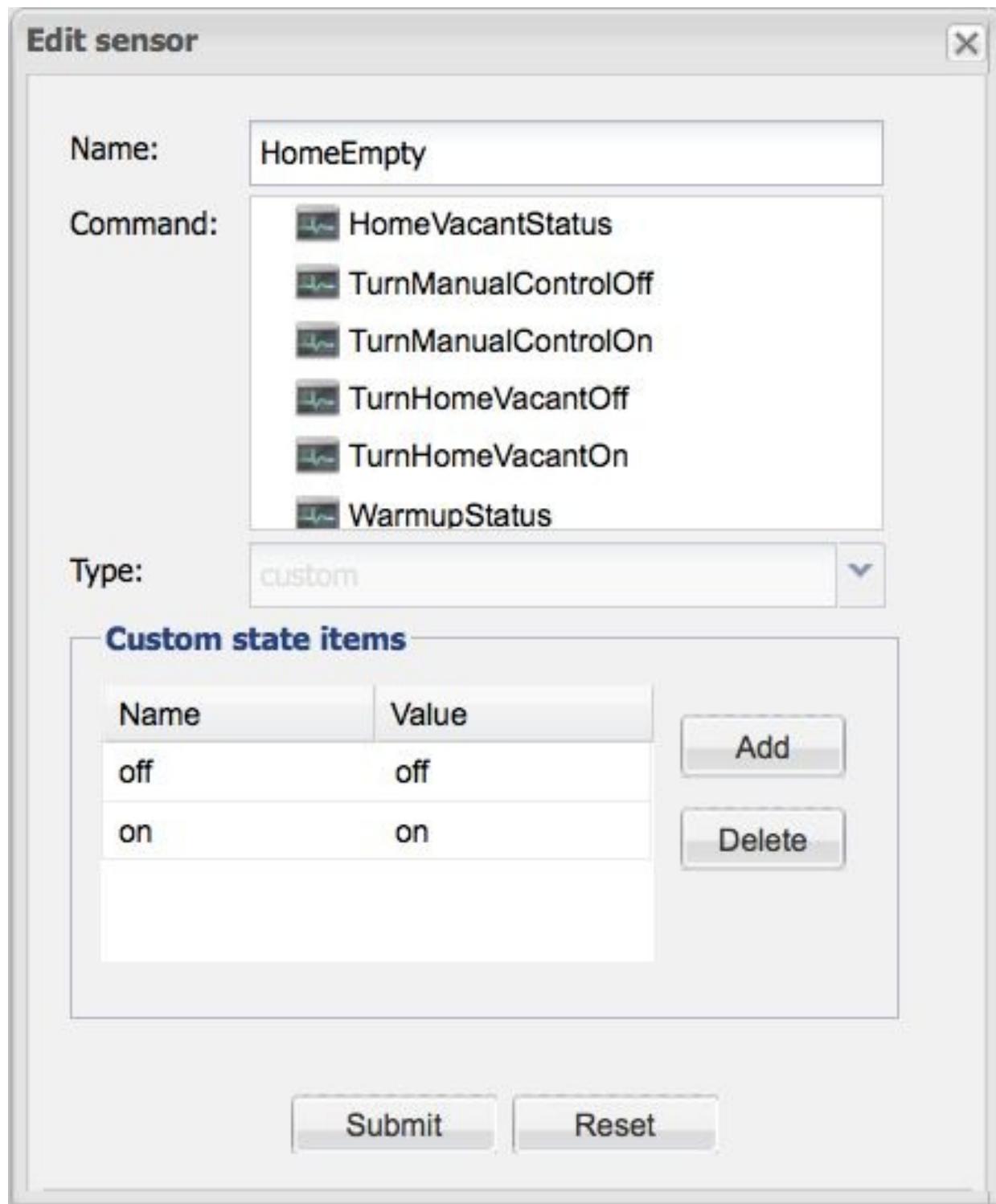


Figure 13.8 Sensor which activates the periodic readout of homeempty.txt
Finally we create a switch command called HomeEmpty using the sensor
HomeEmptyStatus and the two commands TurnHomeEmptyOn and

HomeEmptyStatus and the two commands *TurnHomeEmptyOn* and *TurnHomeEmptyOff*.

We have now created the necessary commands to switch the information state file homeempty.txt to on and off while reading out its current value. We can now, as the final step, create the GUI controls in OpenRemote Professional Designer. First we define a grid with the number of rows and columns, which we plan to use. Then we insert the labels PresenceSim, PresenceDetect, iAlarm, HomeEmpty, HomeWarmup, HomeVacant and ManualControl. Next to each label we insert a switch with the related switch command. In case of the HomeEmpty scenario it is the HomeEmpty switch (see also Figure 8.5). Figure 8.9 shows the resulting user interface on an iPhone.

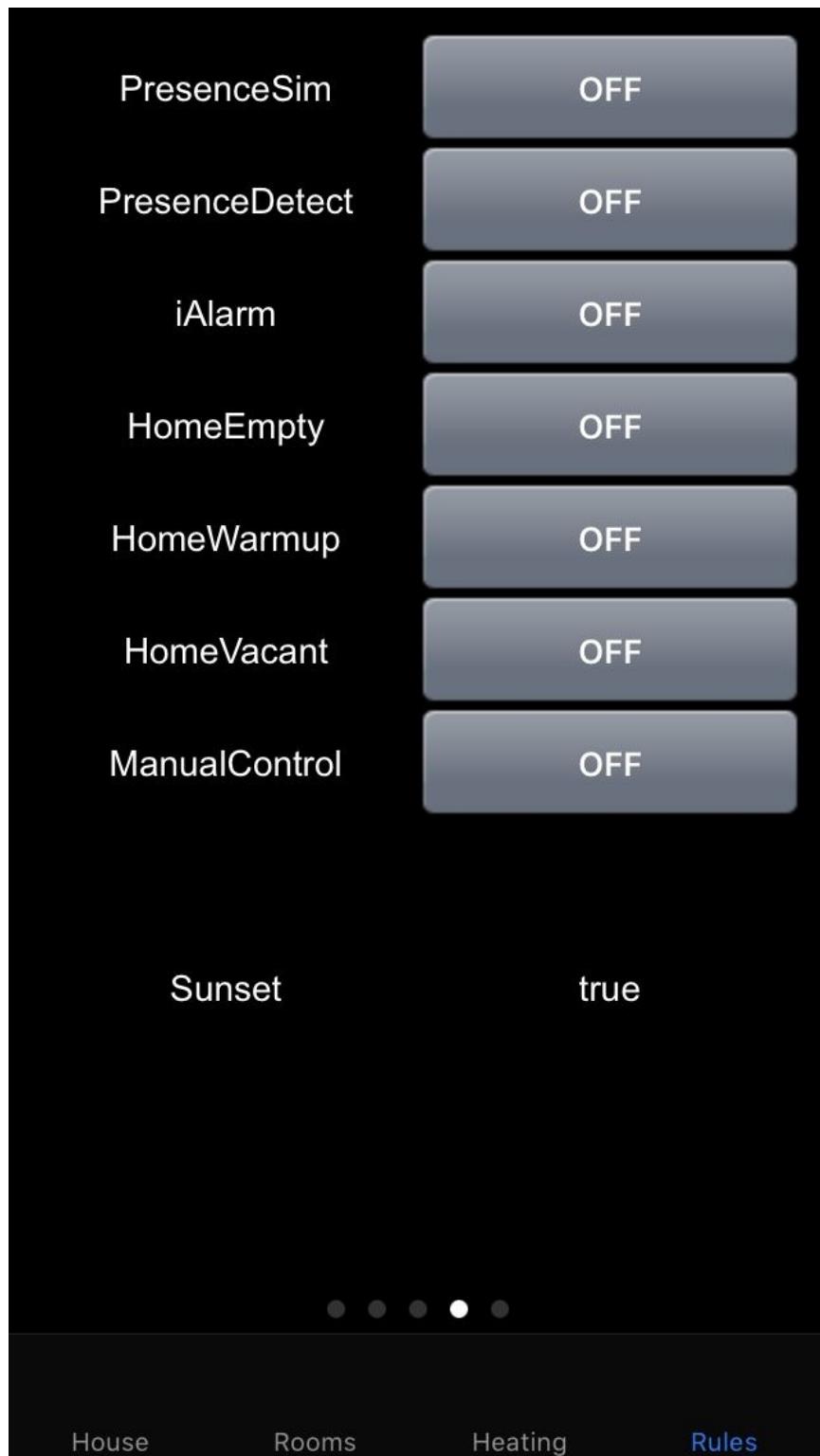


Figure 13.9 GUI controls for persistent information state controls After testing, that our controls actually do change the value of our information state text files,

we can now control the sensors from our Drools rules as desired for the different smart home scenarios.

Reading out the value of a sensor in Drools can be done with command CustomState. As an example the following command line determines if the sensor HomeVacantStatus has the value off.

CustomState(source == "HomeVacantStatus", value == "off") If we now want the rule "switch heating living and dining room to comfort at 6:00 a.m." to only execute if our VacantStatus control is set to off, we simply need to insert this line to the when part of the condition: rule "switch heating living and dining room to comfort at 6:00 a.m."

timer (cron: 0 50 22 ? * MON-SUN)

when

CustomState(source == "HomeVacantStatus", value == "off") then

execute.command("Comfort mode livingroom (ON)"); Below two examples for the rule implementation of the smart home control HomeEmptyStatus and //Rule to switch heating system to night when HomeEmpty is on rule "Heating to Night when HomeEmpty is On"

when

CustomState(source == "HomeEmpty", value == "on") then

```
execute.command("TurnHomeVacantOff");
```

```
execute.command("TurnHeatingOff");
```

```
execute.command("TurnControlDisableOff");
```

```
System.out.println("HomeEmpty ON");
```

```
execute.command("Antifreeze mode familyroom (OFF)");
```

```
execute.command("Comfort mode familyroom (OFF)");
```

```
execute.command("Standby mode familyroom (OFF)");
```

```
execute.command("Night mode familyroom (ON)"); Date date8 = new Date();
```

```
System.out.print(date8.toString());
```

```
System.out.println(" !!!familyroom switched to night!!!!");
```

```
execute.command("Antifreeze mode bedroom1 (OFF)");
```

```
execute.command("Comfort mode bedroom1 (OFF)");
```

```
execute.command("Standby mode bedroom1 (OFF)"); execute.command("Night mode bedroom1 (ON)");
```

```
Date date12 = new Date();
```

```
System.out.print(date12.toString());
```

```
System.out.println(" !!!bedroom1 switched to night!!!!");
```

```
execute.command("Antifreeze mode bedroom2 (OFF)");
```

```
execute.command("Comfort mode bedroom2 (OFF)");
execute.command("Standby mode bedroom2 (OFF)"); execute.command("Night
mode bedroom2 (ON)");
```

```
Date date11 = new Date();
```

```
System.out.print(date11.toString());
```

```
System.out.println(" !!!bedroom2 switched to night!!!");
execute.command("Antifreeze mode bathroom (OFF)");
execute.command("Comfort mode bathroom (OFF)");
execute.command("Standby mode bathroom (OFF)"); execute.command("Night
mode bathroom (ON)");
```

```
Date date15 = new Date();
```

```
System.out.print(date15.toString());
```

```
System.out.println(" !!!bathroom switched to night!!!");
execute.command("Antifreeze mode livingroom (OFF)");
execute.command("Comfort mode livingroom (OFF)");
execute.command("Standby mode livingroom (OFF)");
execute.command("Night mode livingroom (ON)"); Date date9 = new Date();
```

```
System.out.print(date9.toString());
```

```
System.out.println(" !!!livingroom switched to night!!!");
execute.command("Antifreeze mode diningroom (OFF)");
execute.command("Comfort mode diningroom (OFF)");
```

```
execute.command("Comfort mode diningroom (OFF)");  
execute.command("Standby mode diningroom (OFF)");  
execute.command("Night mode diningroom (ON)"); Date date10 = new Date();
```

```
System.out.print(date10.toString());
```

```
System.out.println(" !!!diningroom switched to night!!!"); end
```

```
//Rule to switch heating system to comfort when Warmup is on rule "Heating to  
Comfort when Warmup On"
```

```
when
```

```
CustomState(source == "Warmup", value == "on") then
```

```
execute.command("TurnHomeEmptyOff");
```

```
execute.command("TurnHomeVacantOff");
```

```
execute.command("TurnControlDisableOff");
```

```
System.out.println("HomeWarmup ON");
```

```
execute.command("Antifreeze mode familvroom (OFF)");
```

```
execute.command("Comfort mode familyroom (ON)"); Date date8 = new Date();
```

```
System.out.print(date8.toString());
```

```
System.out.println(" !!!familyroom switched to comfort!!!");  
execute.command("Antifreeze mode bedroom1 (OFF)");  
execute.command("Comfort mode bedroom1 (ON)"); Date date12 = new Date();
```

```
System.out.print(date12.toString());
```

```
System.out.println(" !!!bedroom1 switched to comfort!!!");  
execute.command("Antifreeze mode bedroom2 (OFF)");  
execute.command("Comfort mode bedroom2 (ON)"); Date date11 = new Date();
```

```
System.out.print(date11.toString());
```

```
System.out.println(" !!!bedroom2 switched to comfort!!!");  
execute.command("Antifreeze mode bathroom (OFF)");  
execute.command("Comfort mode bathroom (ON)"); Date date15 = new Date();
```

```
System.out.print(date15.toString());
```

```
System.out.println(" !!!bathroom switched to comfort!!!");  
execute.command("Antifreeze mode livingroom (OFF)");  
execute.command("Comfort mode livingroom (ON)"); Date date9 = new Date();
```

```
System.out.print(date9.toString());
```

```
System.out.println(" !!!livingroom switched to comfort!!!");  
execute.command("Antifreeze mode diningroom (OFF)");  
execute.command("Comfort mode diningroom (ON)"); Date date10 = new  
Date();
```

```
System.out.print(date10.toString());
```

```
System.out.println(" !!!diningroom switched to comfort!!!"); end
```

13.3 From Sunrise to Sunset

In many smart home environments several devices such as outdoor lights or shades need to be triggered by sunset or sunrise. The time of sunset and sunrise change from day to day and are different for every location, and needs to be calculated accordingly. In OpenRemote sunset and sunrise can be easily calculated using the DateTime protocol. Currently the implementation provides the following commands:

- date (returns a date/time as string depending on the given formatter)
- sunrise (returns the sunrise time as string depending on the given formatter)
- sunset (returns the sunset time as string depending on the given formatter)
- minutesUntilSunrise (returns an integer with the minutes until sunrise)
- minutesUntilSunset (returns an integer with the minutes until sunset)
- isDay (returns a boolean string)

- isNight (returns a boolean string) Figures 13.10 and 13.11 show a command which returns true at sunset, and a sensor, which uses this command to sense for sunset.

New command

Name:

Protocol:

DateTime Protocol attributes

Latitude for Sunrise/Sunset calculation:

Longitude for Sunrise/Sunset calculation:

Timezone todo time corrections. If not set, default is taken.:

What should be calculated / returned?:

How should date/time values be formatted?:

Figure 13.10 OpenRemote DateTime Protocol for sunset calculation

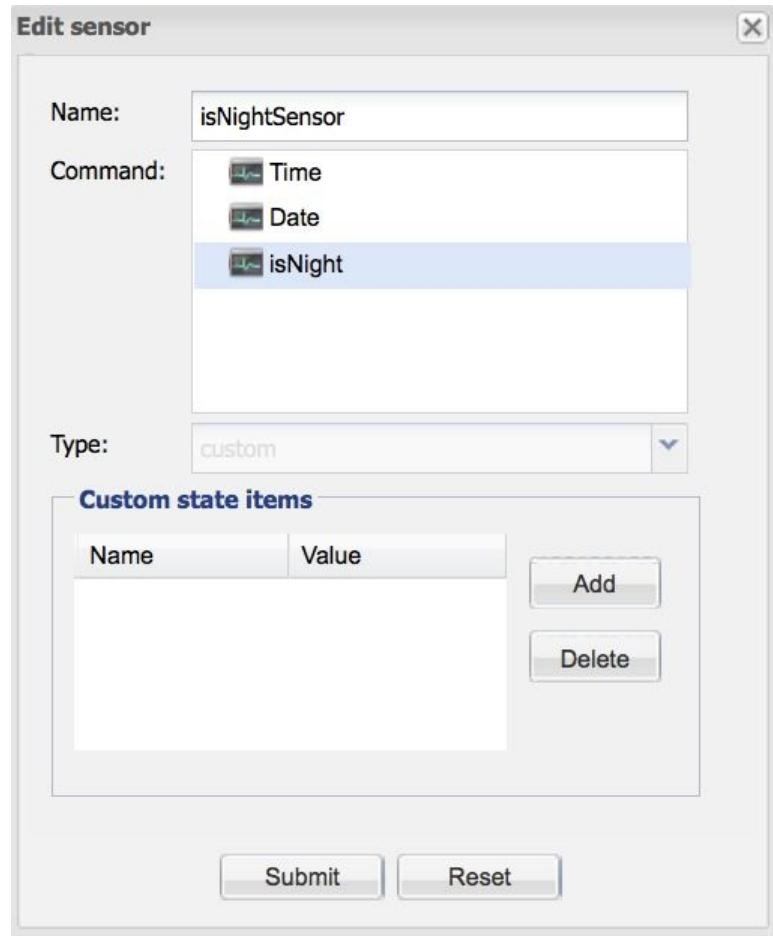


Figure 13.11 OpenRemote DateTime Protocol sensor for monitoring sunset The according rule set for switching on external lights at sunset, and switch them off at 10 p.m could be the following:

rule "Switch on external lights at sunset"

when

```
CustomState(source == "isNightSensor", value == "true") then
```

```
System.out.println("Sunset: Outdoor Light switched ON");
execute.command("Switch external floor and door lights (ON)"); end
```

rule "Switch off external lights at 10p.m."

timer (cron: 0 0 22 ? * MON-SUN)

when

eval(true)

then

```
System.out.println("10p.m.: Outdoor Light switched OFF");
execute.command("Switch external floor and door lights (OFF)"); end
```

14 More iDevices

In this chapter we will now enhance our existing automation scenarios by adding additional components to our smart home control infrastructure. Many new generation home appliances contain integrated Web-or Telnet-server, which allow them to be controlled via HTTP or Telnet commands. Others, which are either simpler or older (or both), such as coffee machines or lamps, need to be connected to smart power-outlets. For the later and for controlling the building infrastructure, we will demonstrate how to use and incorporate Z-Wave components in our project.

14.1 Denon / Marantz Audio System Control

We will start by showing how to integrate a Denon AV receiver into our smart home control system using Telnet commands. The two leading Japanese audio equipment manufacturers Denon and Marantz belong to the same holding company, which is why they use the identical control protocol. So the basic set of commands should work on all recent models from both manufacturers. From the Denon website (see bibliography) one can download the full specification of the DENON AVR control protocol, which allows for full control of all functions. Before we start setting up the configuration of our new device in OpenRemote, we open a terminal window and validate if we can reach our target device using a Telnet connection. We type telnet and at the Telnet prompt open followed by the IP address of our device: telnet> open 192.168.178.16 23

Trying 192.168.178.16...

Connected to denonavr3313.fritz.box.

Escape character is '^]'.

For security reasons the built in Telnet servers of many consumer electronic devices close the connection after every single command they receive. This does not hurt for the one-way remote control scheme, which we are implementing, but is bothering when trying to test more complex things using an online terminal session. On the DENON AVR, we are using, we also have to deal with this behavior, which is why after each command, which we send to the device via Telnet, we need to close the terminal session and open a new one for the next command.

Also make sure to configure your target device (in our case the DENON AVR 3313) with a fixed IP address rather than leaving it with the default setting of

DHCP. If the latter is the case, the moment your router assigns a new IP address to your target device, the settings we will configure in OpenRemote, which contain a static IP address, will not work anymore. So we assign a fixed IP address to our device and test if control via Telnet actually works by sending a command such as Z2ON: telnet> open 192.168.178.16 23

Trying 192.168.178.16...

Connected to denonavr3313.fritz.box.

Escape character is ,^]`.

Z2ON

The DENON AVR should activate zone two and on the front LED you should be able to see the according status display.

Now since Telnet control is working, we can get started with the OpenRemote configuration. We go to OpenRemote Professional Designer and create a new device, which we call **Denon AVR**. We want to create four commands: *switch on*, *switch off*, *volume up* and *volume down*. In our OpenRemote Designer Building Modeler we select **Denon AVR – New – New Command**, and enter the command names, Telnet as the protocol, the default Telnet port number 23, the IP address of our Denon receiver, and, as a first example, the command to switch Zone 2 on, which we look up in the Denon protocol manual to be Z2on. (The Denon 3313, which we use in our example, is capable of sending its audio output to multiple zones. A zone is typically a separate room, with loudspeakers connected to the according zone output. By switching from zone to zone one can control to which room the audio-output is sent). If the target control device provides a Telnet service without a prompt, as is the case with the Denon Telnet daemon, the

OpenRemote implementation requires to prepend each command with null| (null pipeline) without leaving a space to the actual command. To find out if your target device runs Telnet with or without a prompt, just open a Telnet session to the device using a terminal window and look at the output. With that the command for switching on Zone 2 of our Denon AVR, which we enter in the OpenRemote command window reads: null|Z2ON

Figure 14.1 shows the finished command.

Edit command

Name:	AVR3313 Zone 2 (ON)	X
Protocol:	Telnet	▼
Telnet attributes		
IP Address:	192.168.178.16	
Port:	23	
Command:	null Z2ON	
Read Timeout (s):		
Read Regex Filter:		
Read Regex Group:		
Default Read Response:		
Polling interval:		
<input type="button" value="Submit"/> <input type="button" value="Reset"/>		

Figure 14.1 OpenRemote Telnet command for switching Denon AV3313 Zone2 On In the same way we configure the Denon commands Z2OFF Switch Zone2 off Z2UP Zone 2 Volume Up Z2DOWN Zone 2 Volume Down Now we just need to set up the button controls for our four commands in the OpenRemote UI Designer menu and we are done. As a final step we want to add push buttons for four presets, which we have programmed with our favorite Internet radio stations. The according commands for turning on presets 1 through 4 are:

null|NSB01

null|NSB02

null|NSB03

null|NSB04

This gives us another four commands to associate with four button controls. With that we are done with our Denon remote control. (Figure 14.2)

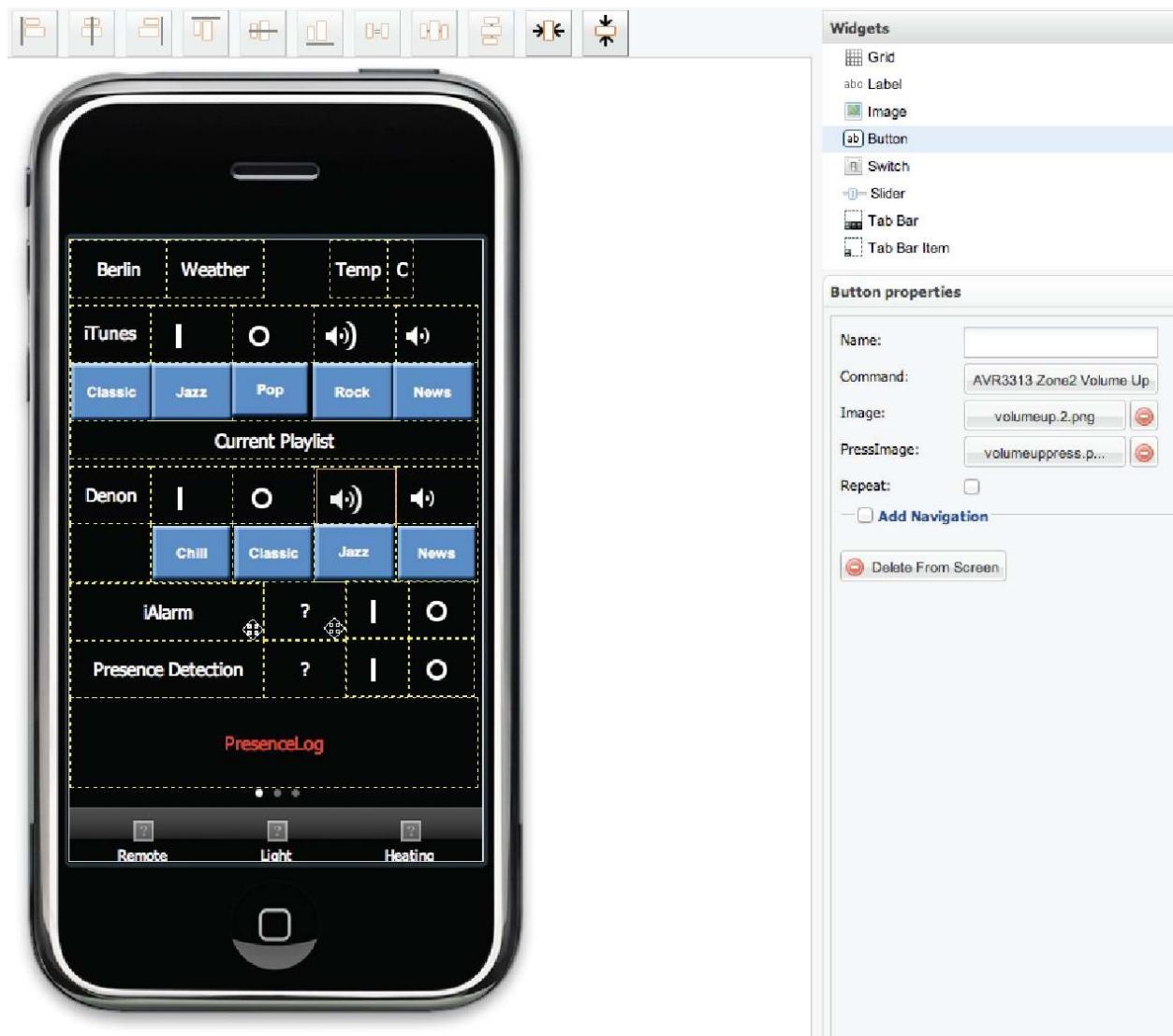


Figure 14.2 The OpenRemote screen layout for Denon AV3313 control Of course we now can easily correlate our Denon commands inside macros with other commands to create single push button scenarios such as

- dinner (switch some lights off, dim other lights, switch on the Denon audio system and select a radio station) or
- leave home (switch music and lights off, change heating operating state to standby, switch on outside lights for three minutes)

Equally we can combine these commands with our iTunes control commands, which now lets us choose in which zone we want to listen to our iTunes playlist. As an example we create a Denon command AVR 3313 Volume to 25 dB, which sets the volume for zone 2 to 25dB. The according Denon protocol command would be null|Z225

In our macro definitions for the iTunes controls (Classic, News, Pop, Rock, Jazz) we now add the OpenRemote commands we have defined AVR3313 Zone 2 (ON) and AVR 3313 Volume to 25 dB, separated by a 2 second delay, to give the first Telnet command the chance to execute before the second is sent out.

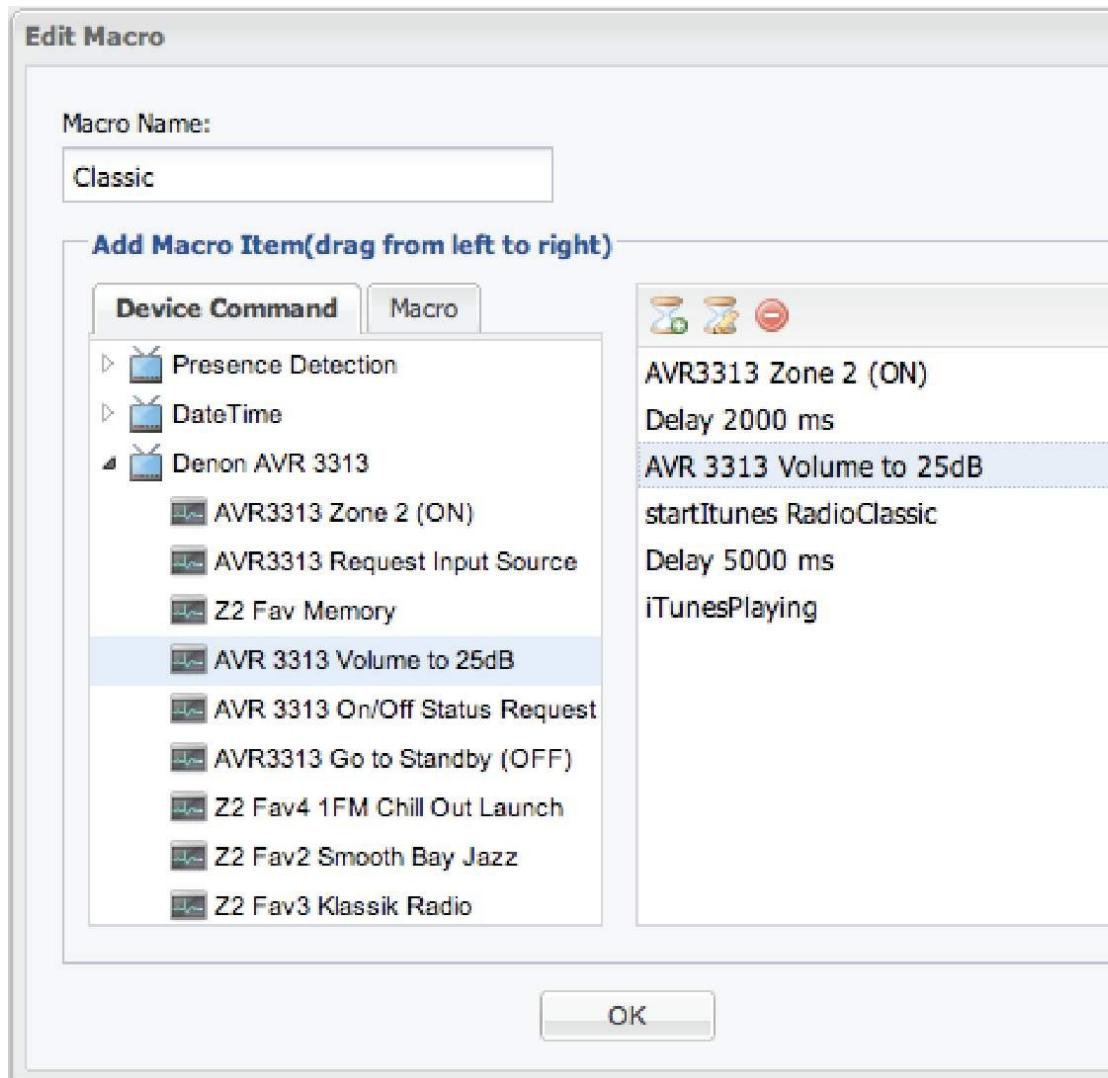


Figure 14.3 Enhancing the iTunes macros with Denon control commands In a similar manner we can update our rule definitions with Denon control commands as needed. As an example we can add the two commands execute.command("AVR3313 Zone 2 (ON)"); execute.command("AVR 3313 Volume to 25dB"); to our rule "Wake me up early if it rains" as shown below, which would ensure, that the iTunes playlist, which is triggered by the alarm condition, is played on the Denon zone 2 at a volume of 25dB. By inserting one command before and one after the Java sleep sequence, we ensure that there is enough time between the two commands to execute: rule "Wake me up early if it rains"

timer (cron: 0 0 6 ? * MON-FRI)

when

```
CustomState(source == „Weather Condition Berlin“, name : value)
CustomState(source == „iAlarm Status“, value == „on“) then
```

```
String testStr = (String) name;
```

```
String lookUp1 = „rain“;
```

```
String lookUp2 = „snow“;
```

```
if ((testStr.matches(,(?i).*"+lookUp1+".*")) || (testStr.matches(,(?i).*"+lookUp2+".*"))){
```

```
System.out.println(„iAlarm going off „,+name); Date date = new Date();

System.out.println(date.toString()); execute.command(„startItunes RadioPop“);
execute.command(„Turn iAlarm Off“); execute.command(„AVR3313 Zone 2
(ON)“); try {

    Thread.sleep(5000);

} catch(InterruptedException ex) {

    Thread.currentThread().interrupt(); }

execute.command(„iTunesPlaying“);

execute.command(„AVR 3313 Volume to 25dB“); }

end
```

We see, that once the basic structure is in place and functioning, it is very easy to expand the functionality of a rule.

14.2 Device Control Using Z-Wave

As outlined in the introductory chapters of this book, for the control of the building infrastructure there are a number of options. One of the most popular and wide spread open standard based solutions is Z-Wave, which we will cover in this section. The Z-Wave communication protocol is also supported by OpenRemote, which is why we can seamlessly integrate any Z-Wave controlled device into our smart home control center. To build a Z-Wave network all you need is a Z-Wave controller, typically in form of a USB-stick, and Z-Wave controllable devices. There are a large number of Z-Wave devices available from various vendors, starting from power-switches, sensors (water, door, energy, light), door-bells or general purpose controllers. Other examples, which demonstrate what is possible with Z-Wave, are LED light-bulbs with built in Z-Wave switches or Z-Wave controllable film, that converts glass from transparent to opaque for energy and privacy needs. (<http://aeotec.com/homeautomation>).

14.2.1 Z-Wave Network Setup

For our project we will use the Aeotec USB stick, currently probably one of the most popular Z-Wave controllers on the market. To install it we download the necessary software, a USB to UART driver (available for macOS, Windows and Linux) from

<http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.asp>

After driver installation under OS-X we can validate the correct installation with the command

```
ls dev | more:
```

```
smarthome$ ls dev | more
```

.....

.....

console

cu.Bluetooth-Modem

cu.Bluetooth-PDA-Sync

cu.SLAB_USBtoUART

If everything is correct we should see a device named cu.SLAB_USBtoUART.

Under Windows the USB stick is assigned a COM port (e.g. COM3) after insertion. Go to [*Device Manager – Ports*](#) to identify the port number, which you will need later for the configuration in OpenRemote. Also be aware, that when you move the stick from one USB-port to another, the COM port might change, and your configuration might not work anymore. (Figure 14.4)

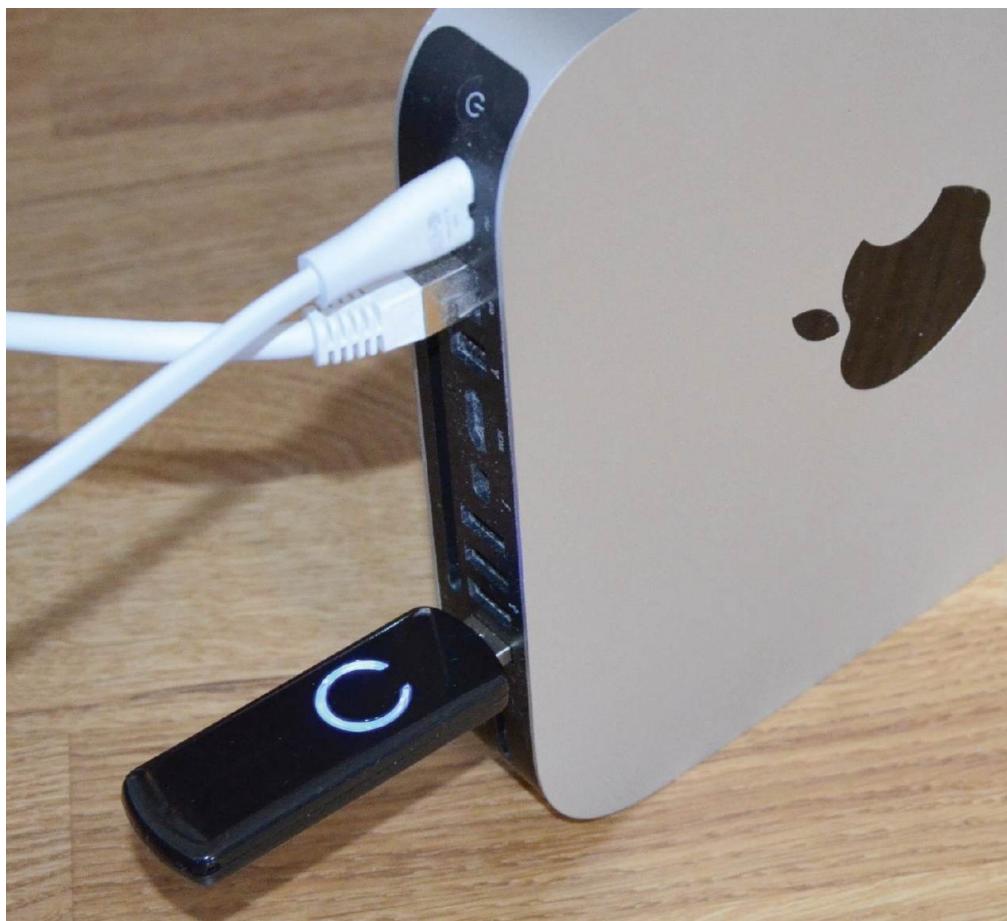


Figure 14.4 The Aeotec Z-Wave Controller USB Stick

14.2.2 Connecting Z-Wave Devices

As a first Z-Wave device we connect an Everspring AN158 to our Aeotec USB stick based Z-Wave network. The AN158 is a combined power-switch and power meter (Figure 14.12). In order to include it in our Z-Wave network, we need to disconnect our Aeotec USB stick and press its control key, which starts flashing slowly. Now we put it next to the plugged in Everspring AN158, which we now also switch to inclusion mode by pressing its control button three times within 1.5 seconds. The Aeotec LED now starts blinking fast, and then stays solid for three-seconds. This is the sign that we have successfully associated our Everspring AN158 with our Z-Wave network. We now plug the Aeotec USB stick back into our controller DC

SUCH BACK INTO OUR COMMUNICATI F C.

14.2.3 Configuring OpenRemote for Z-Wave Operation

Now we log into OpenRemote Professional Designer and open the Z-Wave configuration screen by selecting *Config for Controller – Z-Wave*. For the field *zwave.Comm.Layer* we select *RXTX* from the drop down menu. In the field *zwave.com.Port* under MS Windows we enter the COM port we use for our USB-stick, under macOS we enter the path to the device driver *devtty.SLAB_USBtoUART* we have installed before (Figure 14.5).

The screenshot shows a configuration dialog titled "z-wave". It contains several input fields and a hint section. The fields are:

zwave.pad.host:	localhost
zwave.commLayer:	RXTX
zwave.comPort:	/dev/tty.SLAB_USBtoUART
protocol.zwave.classname:	org.openremote.controller.protocol.zwave.ZWaveCommandBuilder
zwave.pad.port:	7876

Hint
Configuration for the Z-Wave protocol

Buttons at the bottom: **Reset to defaults** and **Submit**.

Figure 14.5 Configuring Z-Wave in OpenRemote

Next we let OpenRemote automatically detect our Z-Wave devices by selecting the *Discovered Devices Wizard* (Figure 14.6).



Figure 14.6 Starting Z-Wave node auto detection: Discovered Devices Wizard

The Discovered Devices Wizard GUI opens, displaying all detected Z-Wave nodes in the network (Figure 14.7). Now we checkmark the first device, select *Devices – Create Selected Devices* and OpenRemote automatically adds the device settings for that device, in our case the Everspring AN158, to the OpenRemote device repository (Figures 14.8, 14.9).

Discovered Devices Wizard

Devices Edit Options

Discovered Devices

Name	Model	Protocol	Type	Used
Device 2	N/A	zwave	Switch	true

Figure 14.7 Discovered Z-Wave nodes

Discovered Devices Wizard

Devices Edit Options

- Show all
- Show only new
- Create selected devices**
- Exit wizard

	Protocol	Type	Used
	zwave	Switch	true

Figure 14.8 Creating devices from discovered Z-Wave modes



Figure 14.9 Automatically configured Z-Wave devices

Now, all what is left to do, is to create a GUI in OpenRemote Designer, which contains the necessary GUI elements: the switch element for switching our AN158 on and off and the sensor, to display the power meter reading of our device (Figure 14.10).

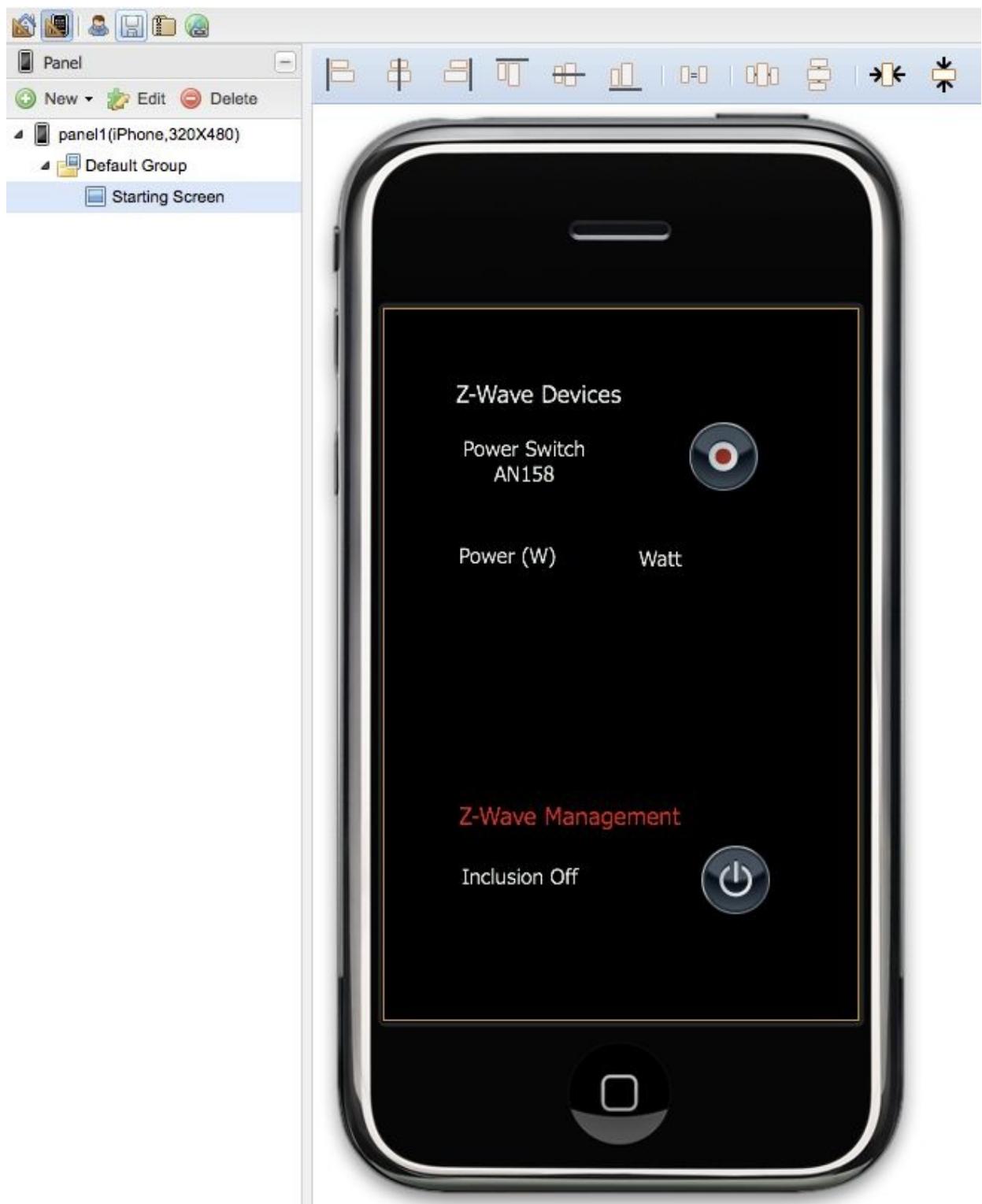


Figure 14.10 Creating the Z-WAVE GUI for the AN158 in OR Designer

We can now control the AN158 power switch from our smartphone app and monitor its power level (Figure 14.11). Using the above procedure we can integrate any Z-Wave item of our building infrastructure. Equally we can integrate any OpenRemote Z-Wave command, sensor or switch into any of our OpenRemote rules.

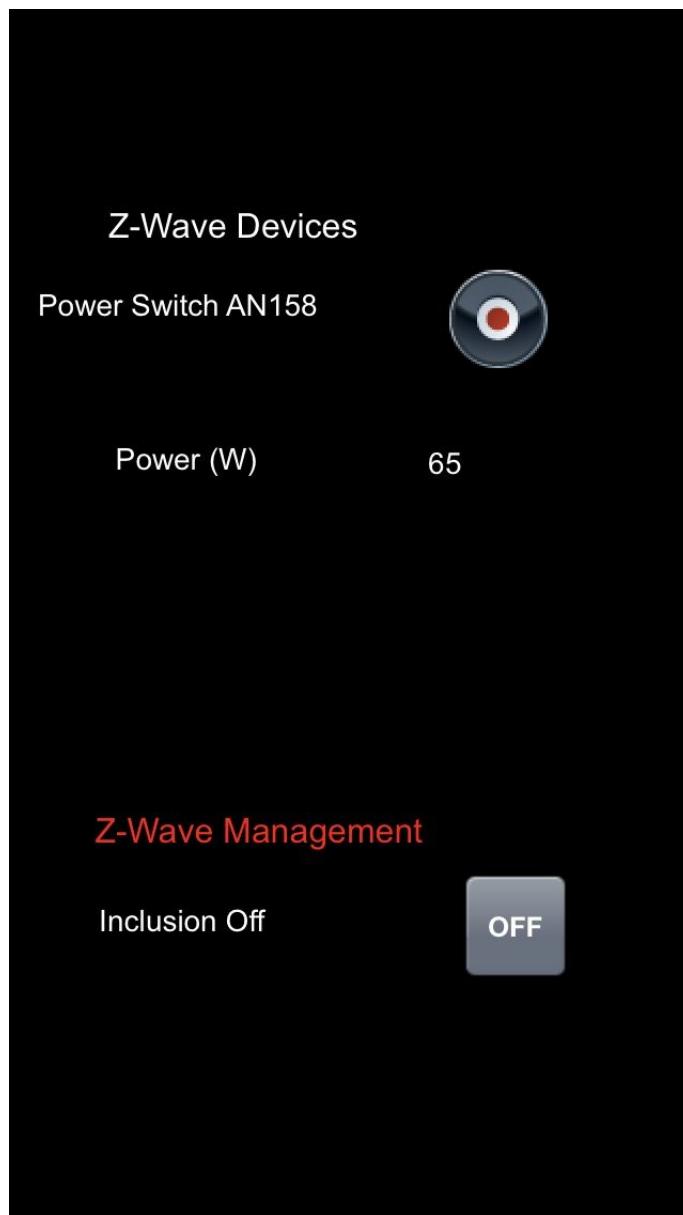


Figure 14.11 The AN158 control GUI on a smartphone

Figure 14.11 The Z-Wave control unit on a smartphone



Figure 14.12 The Z-Wave combined power switch and power meter Everspring AN158

14.3 Bibliography

DD&M Holdings Inc. (2013): DENON AVR control protocol 5.2a.

http://www.denon-media.ru/upload/2014/06/09/avrx1000_e300_protocol_10.0.0_v04.pdf

Christian Paetz, Serguei Polterak (2001): ZWay Manual Z Wave.Me. http://en.z-wave.me/docs/zway_manual_en.pdf

15 Where are you? Geo-fencing!

With the omnipresence of GPS receivers in smartphones, cars, laptops, watches, pet collars, luggage and many other devices, the dimension space has become an integral component of device control and automation. For the smart home knowing the location of its residents opens up an entire new category of applications. Important infrastructure components such as heating, lighting or presence simulation can be coordinated with the location of the users of a building. The combination of a location aware device (e.g. a smartphone) with a second device (e.g. a smart home controller) taking an action if the first device enters or leaves a certain area, is called geo-fencing. Some of the most popular geo-fencing applications in home automation are

- turn up heating (or lift the shades in summer) if a person leaves work in late afternoon
- turn down heating (or lower the shades in summer) if a person leaves home in the morning (with no one left at home)
- turn on lightning if a person arrives at home
- turn off lighting if a person leaves the home (with no one left at home)
- monitoring the location of pets carrying smart collars
- monitoring the location of kids, with notifications if they leave home

- monitoring elderly and handicapped people with notifications if they leave home
- turn on presence simulation if a person leaves home at night (with no one left at home)
- turn on the alarm system if a person leaves home (with no one left at home)

In this chapter we will add geo-fencing capabilities to our project, which will deliver the necessary functionality for all of the above applications. In order to do this, we add two new components to our infrastructure, which are very powerful, easy to use and, best of all, which are free: Google Drive and IFTTT (If This Then That).

15.1 Google Drive

Google Drive is a popular cloud based file storage and application service. It provides cloud based applications for creating documents, spreadsheets, presentations, drawings, and forms. Every user can use all applications and up to 15 GBytes of data for free. We will use a spreadsheet on Google Drive (Google Sheets) as our geo-fencing database. Every time a person leaves or enters a geo-fencing area, date, time, name and location of the event will be stored in a sheet called Presence. A geo-fencing management application will periodically probe, if the sheet Presence contains data. If a geo-fencing entry is detected, the according data is passed on to the OpenRemote home controller to take actions according to its rules. Further the entry will be copied to another sheet called Archive, where it is stored. The original entry in the sheet Presence will be deleted. The management application will be written in PHP using the Google Sheets API for handling the Google Sheets operations. In addition we will create a smartphone app, which will allow us to display the contents of our Archive sheet on a smartphone.

15.2 If This Then That (IFTTT)

For the implementation of the Geofencing trigger we use the IFTTT (If This Then That) service. IFTTT is a popular web based service, which allows to create simple, conditional statements, (applets), which can communicate with a large number of different devices and services. Using IFTTT, with a few clicks you can trigger actions based on events such as phone calls, emails, a new photo taken, reminders, tweets, instagram posts or entering a location. The events connected to these triggers can be equally diverse and span from sending blog posts or emails to opening your garage door or brewing coffee. With the fast growing popularity of IFTTT, the number of products and services, which you can connect to using IFTTT, is enormous. Just to name a few of the bigger names: Apple (iOS), Google (Nest, Android, Drive), Facebook, Twitter, Tesla, BMW, Philips (HUE), Belkin (WeMo), GitBit, GE Appliances or SmartThings. In addition to connecting to the ready to use apps of vendors, with the so called Maker applet you can define a custom action, which will send a web request to a publicly accessible URL. For our geo-fencing applet we will connect an iOS (or Android) location trigger with a custom Maker action, which will store name, location as well as date and time of the geo-fence-trigger to our spreadsheet Presence on Google Sheets. (Figure 15.1), which will serve as the geo-fencing database for our OpenRemote home controller. Based on the entries in the Google sheet, the controller will trigger actions according to the rules in the OpenRemote rule database.

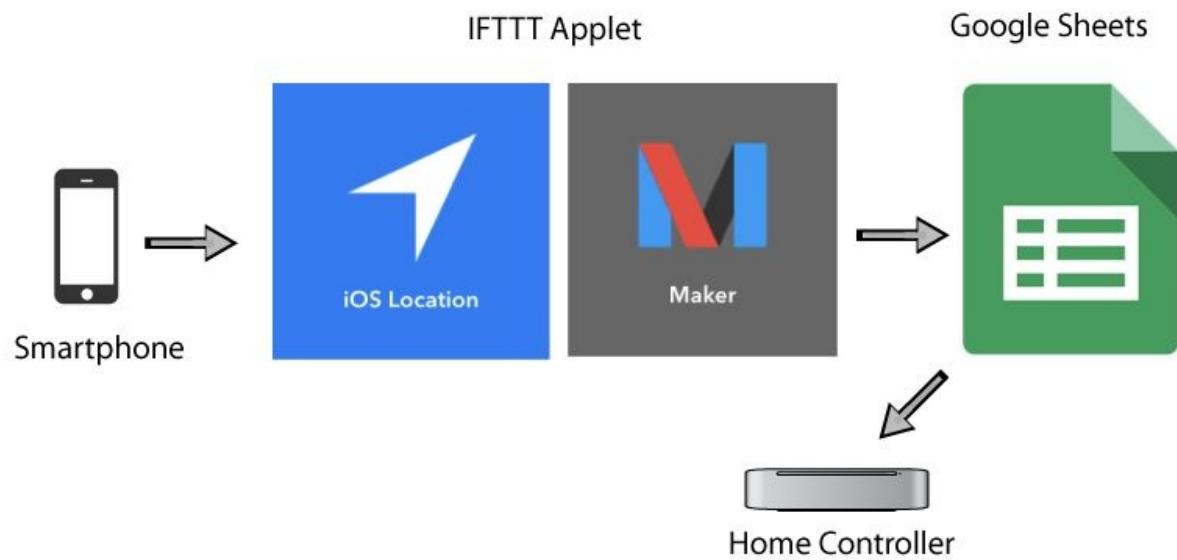


Figure 15.1 The Geofencing application architecture

15.3 A Geo-Fencing Database on Google Drive

As the first step we will set up a Google form and an associated Google sheet. (You could also start with a Google sheet and then create an associated form for data input. The result will be the same). Go to <https://www.google.com/drive/> and set up a free Google Drive account in case you do not have one yet. From the Google Drive home page select *My Drive – More –Google Forms* (Figure 15.2).

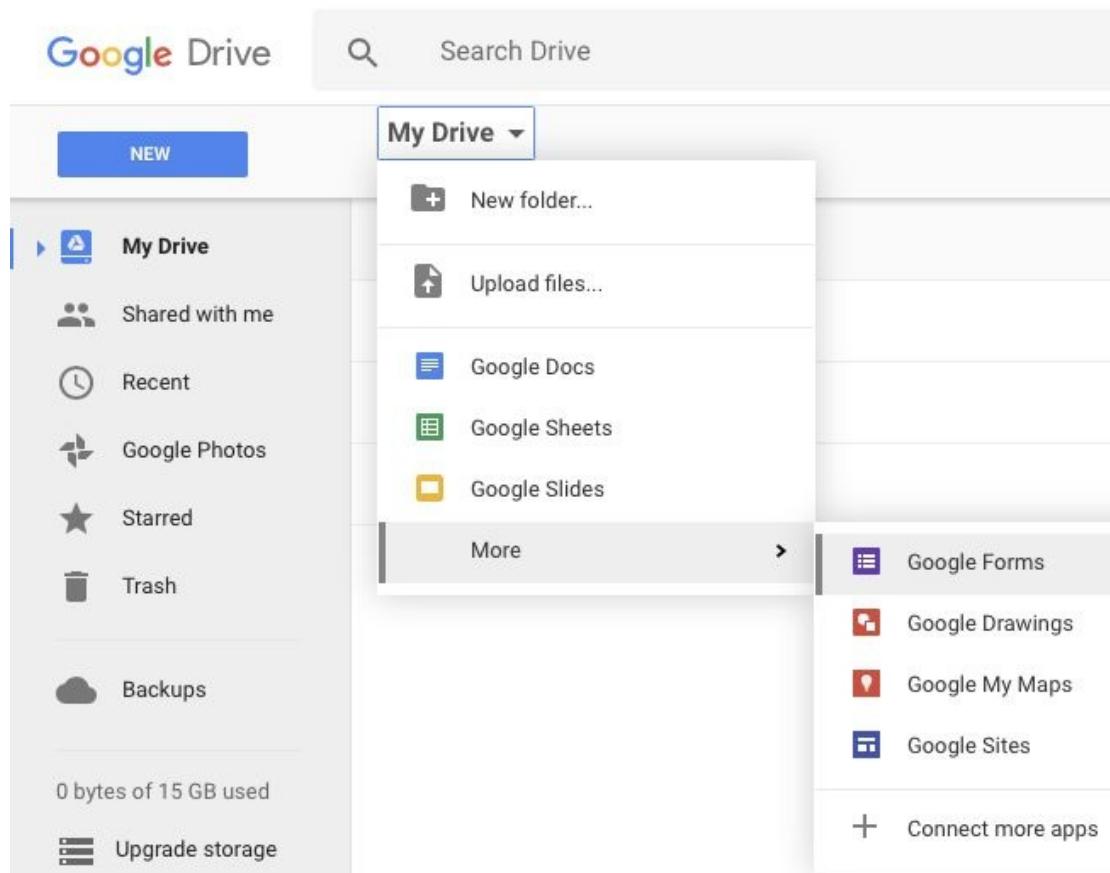


Figure 15.2 Google Drive and Google Forms Create a new form, name the form (e.g. HomePresenceChris), and add two fields. Name the fields Name and Event.

Then select Responses – Select Response Destination, select Create a new spreadsheet and create a linked sheet. We name the new spreadsheet PresenceChris and create a second page (sheet). The first sheet we call

Presence, the second one Archive. Make also sure the time zone setting for the spreadsheet is correct, so that recorded time stamps are displayed correctly. It can be found under File – Spreadsheet settings(Figures 15.3, 15.4, 15.5).

The screenshot shows a Google Form interface. At the top, there are tabs for "QUESTIONS" (underlined in blue) and "RESPONSES" with a count of "2". Below the tabs, the form title is "HomePresenceChris". Under the title, it says "Form description".

The first question is labeled "Name" and is a "Short answer" type. It has a text input field with the placeholder "Short answer text". To the right of the input field are three icons: a square with a minus sign, a trash can, and the word "Required".

The second question is labeled "Event" and is also a "Short answer" type. It has a text input field with the placeholder "Short answer text".

Figure 15.3 Creating a Google form as the basis for the geo-fencing database

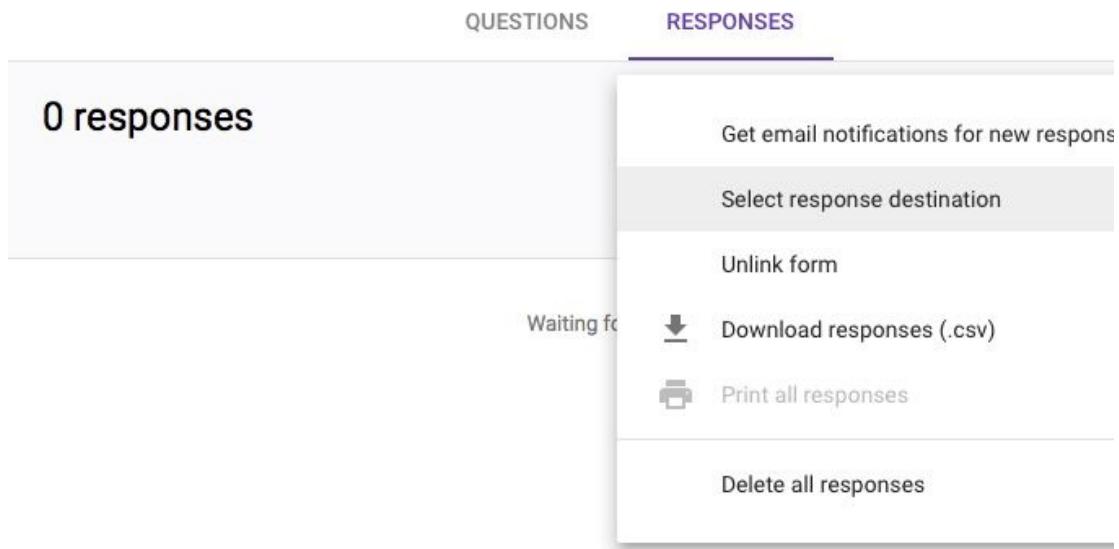


Figure 15.4 Selecting the destination for the form data

Select response destination

Responses are currently being sent to [this spreadsheet](#).

Create a new spreadsheet HomePresenceChris [Learn more](#)

Select existing spreadsheet

[CANCEL](#) [CREATE](#)

Figure 15.5 Creating a new Google sheet based on a Google form Now Select [Get pre-filled link](#), enter the values for your two fields, which you want to add every time the geo-event takes place, in our case the values Chris (name of the tracked person) and GowardRd (to be tracked location). Time and date of exit and entry will be added automatically. Then select [Submit](#) (Figures 15.6, 15.7).

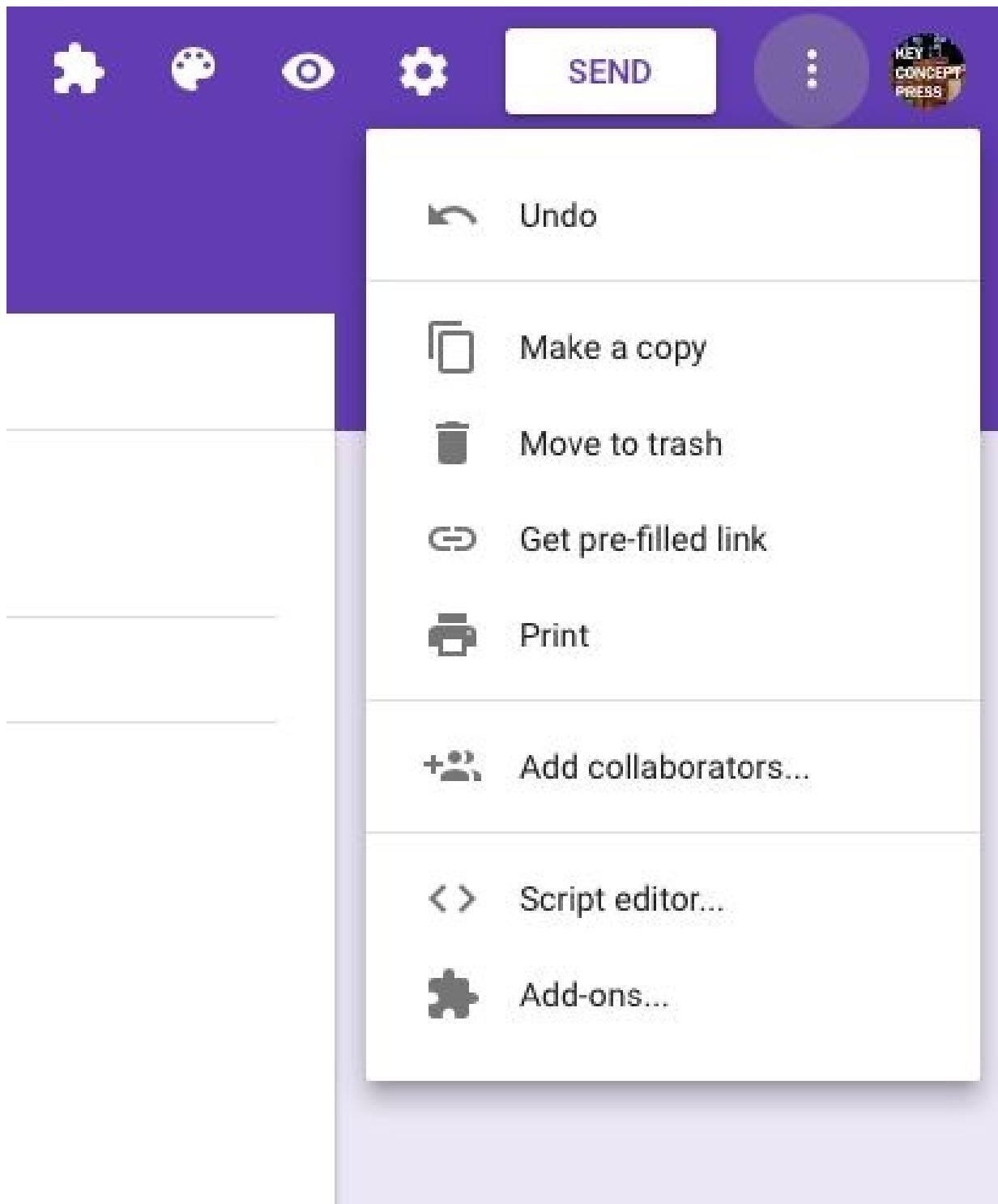


Figure 15.6 Get the pre-filled link for the Google form

Share the link to this form with your pre-filled responses.

https://docs.google.com/forms/d/e/1FAIpQLSdgAlM7_BtYnPSuXL0rxm9J4Bcwrgentry.580126226=Chris&entry.347387655=GowardRd

HomePresenceChris

Name

Chris

Event

GowardRd

SUBMIT

Never submit passwords through Google Forms.

Figure 15.7 Retrieving the URL for the pre-filled Google form In response to the Submit command a URL for the form with our data as pre-filled entries appears at the top of the form user interface (Figure 15.7). Copy the resulting URL and store it in a text document. It should look like the following:

https://docs.google.com/forms/d/e/1FAIpQLSdgAlM7_BtYjPSuXL0rxm9J4Bcwrgentry.580126226=Chris&entry.347387655=GowardRd The strings

entry.580126226 and entry.347387655 are called form identifiers. All you need to do now is to convert this URL to a URL, which enters the two values Chris and GowardRoad into your spreadsheet. You do this by replacing the textstring viewform in the URL with the textstring formResponse. The URL should now look like the following:

*https://docs.google.com/forms/d/e/1FAIpQLSdgAlM7_BtYnPSDXL0rxm9J4Bcwr
entry.580126226=Chris&entry.347387655=GowardRd Every time you now open this URL with an Internet browser, a new row with the entries Chris and GeoTrigger is being added to your Google spreadsheet. In the browser window you will see the confirmation, that the data has been added (Figure 15.8).*

HomePresenceChris

Your response has been recorded.

[Submit another response](#)

*Figure 15.8 Confirmation of successfull data entry in the Internet browser window 15.3 Creating a IFTTT Geo-Fencing Applet Now we will turn the location service of our smartphone in a geo-fencing device and store all geo-fencing-alerts into the Google Sheets spreadsheet, we just set up. Go to <https://ifttt.com> and create a free account, in case you do not have one. Then select **My Applets** and **New Applet**. Click on **this** and enter iOS (or Android) in the search box, then select **iOS Location** (or **Android Location**) and then **You enter or exit an area**. You select the adress and the radius for your geo-fence trigger and select **Create trigger** (Figure 15.9).*



Complete trigger fields

Step 2 of 6

You enter or exit an area

This Trigger fires every time you enter or exit an area you specify.

Locate an area *

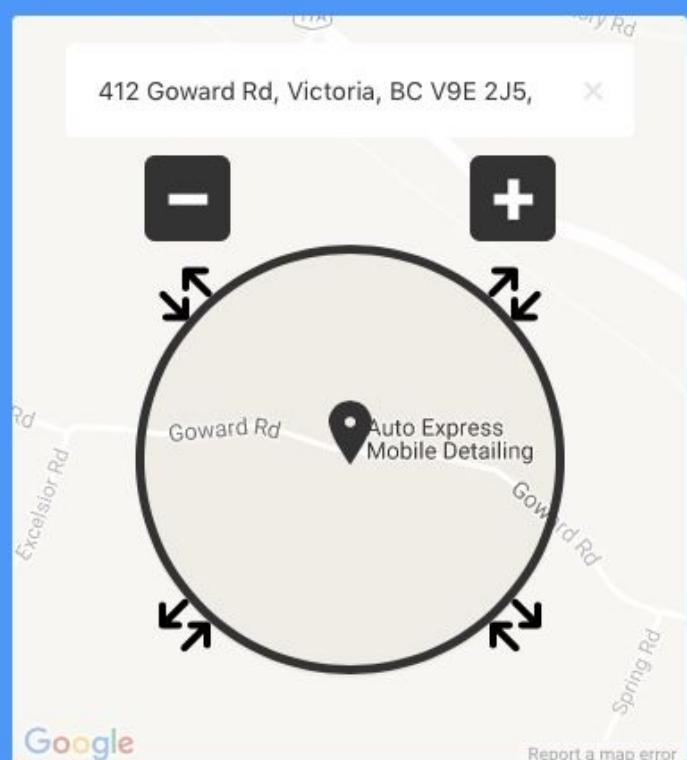


Figure 15.9 The IFTTT iOS Location Applet Now you click on [that](#) and enter

Maker in the search box. Then select **Maker**. Select **Make a web request** and enter the Google Sheets data entry URL from the previous section in the **URL** field. Then select **+Ingredient** and **EnteredORExited** (Figure 15.10). This adds the term `{{EnteredOrExited}}` to the URL, which configures the trigger to become active when exiting or entering the selected area. Then select **GET** in the **Method** field and **application/x-www-form-urlencoded** in the **Content Type** field.

M Make a web request

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

URL

```
https://docs.google.com/form  
s/d/e/1FAIpQLSdgAlM7_BtYn  
PSuXL0rxm9J4BcwrgOCnmG  
6bnMIQE3yL024Pw/viewform  
?  
entry.580126226=Chris&entr  
y.347387655=GeoTrigger|
```

Surround any text with "
<<<" and ">>>" to escape
the content

+ Ingredient



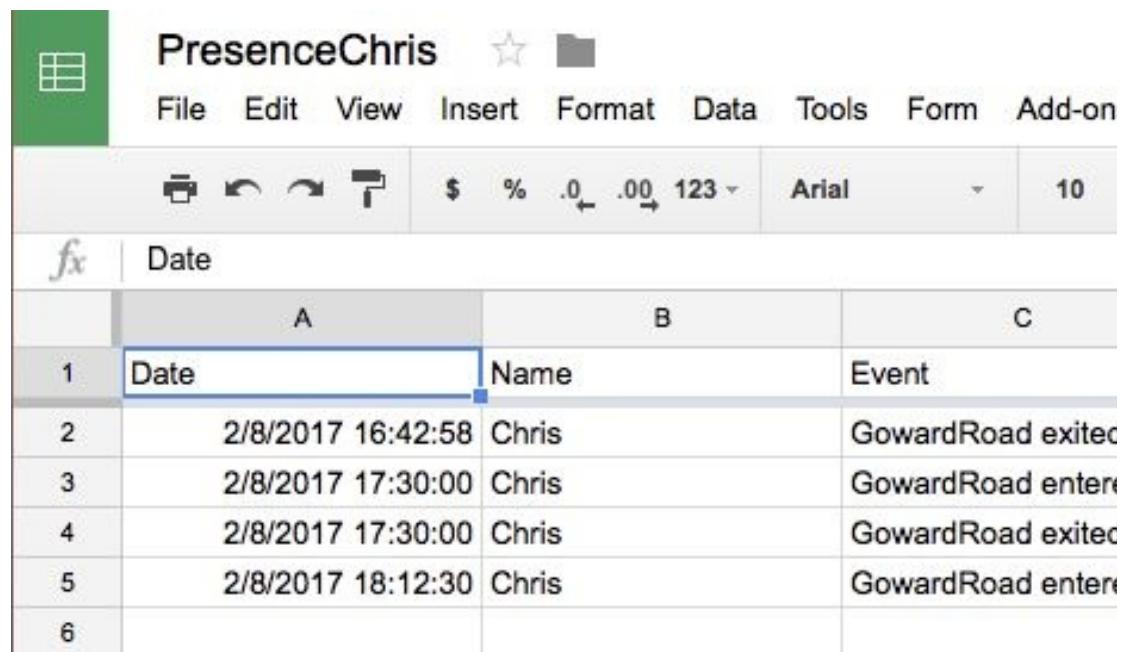
You enter
or exit an
area

Latitude
Longitude
Radius
OccurredAt
Address
LocationMapImageUrl
LocationMapUrl
EnteredOrExited

Content type (optional)

Figure 15.10 Completing the IFTTT Action Field of the Maker Applet Test your work by leaving and entering the geo-fencing area with your smartphone.

According entries should appear in your Google Sheets spreadsheet (Figure 15.11). All which is left to do now is to poll the spreadsheet Presence for new entries and trigger according actions.



The screenshot shows a Google Sheets spreadsheet titled "PresenceChris". The interface includes a green sidebar icon, the title bar, and a toolbar with various icons. The main content is a table with columns labeled A, B, and C. Column A is labeled "Date", column B is labeled "Name", and column C is labeled "Event". The data consists of six rows:

1	Date	Name	Event
2	2/8/2017 16:42:58	Chris	GowardRoad exited
3	2/8/2017 17:30:00	Chris	GowardRoad entered
4	2/8/2017 17:30:00	Chris	GowardRoad exited
5	2/8/2017 18:12:30	Chris	GowardRoad entered
6			

Figure 15.11 The Google sheet Presence, our geo-fencing database

15.4 Google Sheets Manipulation

Since we only want to keep a single entry in our geo-fencing sheet, we add a script, which deletes the entry, after it has been processed by the controller, and which copies the content to our sheet Archive. All spreadsheets and the sheets it contains are identified by two types of ID, the spreadsheet ID and the sheet ID. To retrieve the ID of one sheet open the spreadsheet and select the desired sheet. Then copy the URL from the browser address field. It should look something like the below:

<https://docs.google.com/spreadsheets/d/1X39VbE37fD4IlF6L0Upom8x75m4LEvI>

The spreadsheet ID is the value between the "d" and the "/edit" in the URL. The sheet ID is numeric and is the value of the gid parameter in the URL. In case of the above URL the two IDs are:

[https://docs.google.com/spreadsheets/d/1X39VbE37fD4IlF6L0Upom8x75m4LEvI/edit#gid=sheetId](https://docs.google.com/spreadsheets/d/1X39VbE37fD4IlF6L0Upom8x75m4LEvI/edit#gid=526889915)

spreadsheet ID: 1X39VbE37fD4IlF6L0Upom8x75m4LEvI

sheet ID: 526889915

More information about manipulating Google sheets can be found in the Google Sheets API description under

<https://developers.google.com/sheets/api/guides/concepts>.

15.5 Writing a Geofencing Sheet Management Script

We will now create a PHP command line script, which copies the first line from the geofencing sheet Presence (in case there is an entry), appends it to the sheet Archive and then deletes the entry in Presence. Further we will restrict the maximum number of entries in the Archive sheet to 100.

15.5.1 Setting up our work environment: PHP and Apache

Since PHP code requires a webserver to run, we need to install the PHP programming environment and a webserver, which in our case will be the Apache webserver. The required PHP version for our script is 5.4 or greater (<http://php.net/>) with the PHP command-line interface (CLI) and JSON extensions, as well as the PHP Composer (<https://getcomposer.org>), which is a PHP tool for managing libraries.

Activating PHP and Apache under macOS

On macOS an Apache webserver as well as PHP comes preinstalled and just needs to be activated. To activate PHP under macOS you need to uncomment (by removing the hashtag) the line

```
#LoadModule php5_module libexec/apache2/libphp5.so
```

in the file httpd.conf, which is located in *privateetc/apache2/httpd.conf*. To edit httpd.conf open a terminal window, navigate to the above directory and enter the command

```
sudo pico httpd.conf
```

The command `sudo` is required, since you need to have administrator rights to edit the file, and the text editor `pico` is needed, since you cannot edit the file with standard text editors such as `TextEdit`, for security reasons. More details on the activation of PHP on a Mac can be found at
<http://php.net/manual/en/install.macosx.bundled.php>.

The document root of the Apache server under macOS (and with that the directory for our php scripts), is

Library/WebServer/Documents/

Do not forget to add write permission to this directory for our later development work:

`chmod +w`

To be able to obtain information about our PHP installation we create the plain text file `info.php` with the content `<?php phpinfo(); ?>` using the

`touch info.php`

command. Then we edit the file with the Pico text editor entering

`pico info.php`

The Apache Webserver is started using the command

```
sudo apachectl start
```

(With sudo apachectl the server can be stopped again). We can now check our PHP version and at the same time test our Apache server by entering

```
http://localhost/info.php
```

We can see that both extensions we need, CLI and JSON, are already activated. The version of our Apache server can be obtained using the command

```
httpd -v
```

in the terminal window. The last step in preparing our PHP development environment ist the download and installation of the PHP library management tool PHP Composer. A step by step description can be ontaned from <https://getcomposer.org/download/> .

Installing PHP and Apache under Windows 10

Under Windows 10 both Apache and PHP need to be downloaded and installed. In addition you need to install Microsoft Visual C++ 2012 Runtime, which is required by Apache.

For the Apache server software go to

<http://www.apachelounge.com/download/>

For the Microsoft Visual C++ 2012 Runtime environment go to

<https://support.microsoft.com/en-us/kb/2977003>

For PHP go to

<http://windows.php.net/download/>

One of the many good descriptions for installing and configuring PHP and Apache under Windows 10 can be found here.

<https://www.znetlive.com/blog/how-to-install-apache-php-and-mysql-on-windows-10-machine/>

15.5.2 PHP Scripting Using the Google Sheets API

In order to use the Google API you need to register under

<https://console.developers.google.com/flows/enableapi?apiid=sheets.googleapis.com&pli=1>

This automatically creates a new project called my projects in AP manager. To get a working PHP script along with Google API functions up and running, it is easiest to follow the Google PHP Quickstart documentation at

<https://developers.google.com/sheets/api/quickstart/php>. In six simple steps you set up the sample file quickstart.php on your computer, which retrieves data from a sample Google Sheets spreadsheet. Before successfully starting quickstart.php replace in line 129 of the file Item.php located at

Library/WebServer/Documents/ve in

LibraryWebServer/Documents/vendorgoogle/auth/src/Cache/Item.php

the string Now with your timezone such as UTC or UTC+1, UTC-6 etc.

When you start the script with the command

php quickstart.php

you get the data of the sample sheet as a print out:

php quickstart.php

Name, Major:

1/14/2017 11:23:05,

1/14/2017 11:33:20,

1/14/2017 11:48:46,

1/14/2017 13:28:36,

1/14/2017 15:05:31,

1/15/2017 9:33:58,

1/15/2017 9:33:58,

1/15/2017 9:37:10,

1/15/2017 9:46:36,

1/15/2017 12:31:55,

1/15/2017 13:11:00,

Now just replace the two lines of the sample script, which contains the spreadsheet ID and the data range. To get the spreadsheet ID of your spreadsheet just open the sheet and copy its URL from the browser address field. It should look something like the below:

<https://docs.google.com/spreadsheets/d/1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg>

The spreadsheet ID is the value between the "d" and the "/edit" in the URL. The sheet ID is numeric and is the value of the gid parameter in the URL:

<https://docs.google.com/spreadsheets/d/1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg#gid=526899915>

spreadsheet ID: 1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg

sheet ID: 526899915

If you now want to read out and print the first line of the spreadsheet Presence, the according code in quickstart.php should read as follows:

```
//reads and prints content of Presence sheet
```

```
$spreadsheetId = '1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg';

$range = 'Presence!A2:C2';

$response = $service->spreadsheets_values->get($spreadsheetId, $range);

$values = $response->getValues();

if (count($values) == 0) {

    print "No data found.\n";

} else {

// print "Name, Major:\n";

foreach ($values as $row) {

// Print columns A, B,C,D which correspond to indices 0,1,2,3.

printf("%s, %s, %s, %s\n", $row[0], $row[1], $row[2], $row[3]);

}

}
```

}

As a result you get a print out of the cells A2,B2,C2 of the spreadsheet Presence. If this is working, we can start further manipulations of our spreadsheet. We rename the file from quickstart.php to geofencemgm.php and make the remaining customizations, so it performs the following functions:

- Read Cells A2,B2,C2 from sheet Presence
- If the number of values contained in Presence is unequal to zero append them to the sheet Archive and print them
- Then delete row 2 of the sheet Presence.
- If the number of rows in sheet Archive exceeds 100, delete row 2 in sheet Archive (row one contains column names, row two the oldest data)

To read data out of a spreadsheet you use the command spreadsheets.values.get:

```
$result = $service->spreadsheets_values->get($spreadsheetId, $range);
```

To write data the you use the command spreadsheets.values.update:

```
$result = $service->spreadsheets_values->update($spreadsheetId, $range, $body, $params);
```

To append data to a sheet you use the command `spreadsheets.values.append`:

```
$result = $service->spreadsheets_values->append($spreadsheetId, $range, $body, $params);
```

To delete rows you use the method `spreadsheets.values.batchUpdate` with `deleteDimension` in the request body of this method.

Important! In order to be able to actually write to the spreadsheet you need to change the scopes definition of the Google sample code from `SPREADSHEETS_READONLY` to `SPREADSHEETS`:

```
('SCOPES', implode(' ', array(
```

```
Google_Service_Sheets::SPREADSHEETS)
```

```
));
```

After changing the scope definition, be sure to update the credentials file `sheets.googleapis.com-php-quickstart.json` which is located in the `./credentials` directory! To do this, delete the existing file (`sheets.googleapis.com-php-quickstart.json`), go to the terminal window and run `quickstart.php` again. You will again be asked to open a URL in your browser which in turn will provide you

~~You will be asked to open a URL in your browser, which in turn will provide you with a verification code, which you enter in the terminal window. Then a new credentials file will be created, and you will be able to use the append and update API commands (Figure 15.12).~~

Open the following link in your browser:

[https://accounts.google.com/o/oauth2/auth?
response_type=code&access_type=offline&client_id=720996768109-
1ponc7hnv6kku9hq4qsuqj9288dhfnso.apps.googleusercontent.com&redirect_uri](https://accounts.google.com/o/oauth2/auth?response_type=code&access_type=offline&client_id=720996768109-1ponc7hnv6kku9hq4qsuqj9288dhfnso.apps.googleusercontent.com&redirect_uri)

Enter verification code:

4/zzMkIFTzrfHOyBaokSHAf_eb0QttRw1lUz2sDlCdDK0

Figure 15.12 Dialogue to generate Google Sheets credential file

Below the code listing of the completed file geofencemgm.php (Figure 15.13). Lines 2 through 70 of the code deal with the OAuth 2.0 protocol (IETF RFC6749 standard) for authentication and authorization. Every client computer executing code using Google API calls to access user data on Google Docs needs to be identified and authorized in order to be allowed to do so. For each new client computer this is done by logging into the Google API console and requesting an access token from the Google Authorization server. The token is stored in the user directory ./credentials (Figure 15.12). Typically the access token is short lived (3600 seconds), which is why the client is also assigned a refresh token, which allows the client to get access tokens on demand. This process only needs to be done the first time the scripts runs on a new client computer, which has not been authorized to the Google Authorization server yet. All that is handled by code lines 2 to 70. At line 73 the code starts, which the actual functionality:

Reading cells A2,B2,C2 from sheet Presence (lines 78 - 86)

If the sheet Presence contains values, they are appended to the sheet Archive and printed (87 - 105)

Deleting row 2 of the sheet Presence after copying and printing the data

Deleting row 2 of the sheet Archive, if the total number of rows in Archive exceeds 100

Comments in the code explain every step (Figure 15.13). The file geofencemgm.php can also be contained in the bonus material for this book, which can be downloaded from <http://www.keyconceptpress.com/how-to-smart-home>.

```
1 <?php  
2 require_once __DIR__ . 'vendor/autoload.php';  
3  
4 define('APPLICATION_NAME', 'Google Sheets API PHP Quickstart');  
5 define('CREDENTIALS_PATH', '~/.credentials/sheets.googleapis.com-php-quickstart.json');  
6 define('CLIENT_SECRET_PATH', __DIR__ . '/client_secret.json');  
7 // If modifying these scopes, delete your previously saved credentials  
8 // at ~/.credentials/sheets.googleapis.com-php-quickstart.json  
9 define('SCOPES', implode(' ', array(
```

```
10 // Google_Service_Sheets::SPREADSHEETS_READONLY)

11 Google_Service_Sheets::SPREADSHEETS)

12 ));

13

14 if (php_sapi_name() != 'cli') {

15 throw new Exception('This application must be run on the command line.');

16 }

17

18

19 // Returns an authorized API client.

20 // @return Google_Client the authorized client object

21

22 function getClient() {

23 $client = new Google_Client();

24 $client->setApplicationName(APPLICATION_NAME);

25 $client->setScopes(SCOPES);

26 $client->setAuthConfig(CLIENT_SECRET_PATH);

27 $client->setAccessType('offline');
```

```
28
29 // Load previously authorized credentials from a file.
30 $credentialsPath = expandHomeDirectory(CREDENTIALS_PATH);
31 if (file_exists($credentialsPath)) {
32     $accessToken = json_decode(file_get_contents($credentialsPath), true);
33 } else {
34     // Request authorization from the user.
35     $authUrl = $client->createAuthUrl();
36     printf("Open the following link in your browser:\n%s\n", $authUrl);
37     print 'Enter verification code: ';
38     $authCode = trim(fgets(STDIN));
39
40     // Exchange authorization code for an access token.
41     $accessToken = $client->fetchAccessTokenWithAuthCode($authCode);
42
43     // Store the credentials to disk.
44     if(!file_exists(dirname($credentialsPath))) {
```

```
45     mkdir(dirname($credentialsPath), 0700, true);
46 }
47 file_put_contents($credentialsPath, json_encode($accessToken));
48 printf("Credentials saved to %s\n", $credentialsPath);
49 }
50 $client->setAccessToken($accessToken);
51
52 // Refresh the token if it's expired.
53 if ($client->isAccessTokenExpired()) {
54     $client->fetchAccessTokenWithRefreshToken($client->getRefreshToken());
55     file_put_contents($credentialsPath, json_encode($client->getAccessToken()));
56 }
57 return $client;
58 }
59
60 // Expands the home directory alias '~' to the full path.
61 // @param string $path the path to expand.
62 // @return string the expanded path.
```

63

```
64 function expandHomeDirectory($path) {  
  
65     $homeDirectory = getenv('HOME');  
  
66     if (empty($homeDirectory)) {  
  
67         $homeDirectory = getenv('HOMEDRIVE') . getenv('HOMEPATH');  
  
68     }  
  
69     return str_replace('~', realpath($homeDirectory), $path);  
  
70 }
```

71

72

```
73 // Reads Cells A2,B2,C2 from sheet Presence. If the number of values contained in Presence is  
unequal to zero append them to the sheet Archive, print them
```

```
74 // and delete row 2 of the sheet Presence. If the number of rows in sheet Archive exceeds 100,  
delete row 2 in sheet Archive
```

```
75 // Sheet URL:  
https://docs.google.com/spreadsheetsd1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg
```

```
76 // SpreadsheetID = '1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg';
```

77

```
78 // Get the API client and construct the service object.
```

```

79 $client = getClient();
80 $service = new Google_Service_Sheets($client);

81 // Service object constructed. Define the spreadsheets.values.get method, use the method to
   retrieve the data from Presence and store the result in $values

82 $spreadsheetId = '1X39VbE37fD4IlF9L0Upom8x75m4LEvpWYcPb6Gww5gg';

83 $range = 'Presence!A2:C2';

84 $response = $service->spreadsheets_values->get($spreadsheetId, $range);

85 $values = $response->getValues();

86 //If the sheet Presence contains data, use the method spreadsheets.values.append to append the
   data to the sheet Archive, and print them

87 if (count($values) != 0) {

88     $client = getClient();

89     $service = new Google_Service_Sheets($client);

90     $body = new Google_Service_Sheets_ValueRange(array('values' => $values));

91     $valueInputOption = 'RAW';

92     $range = 'Archive';

93     $params = array('valueInputOption' => $valueInputOption);

94     $result = $service->spreadsheets_values->append($spreadsheetId, $range, $body, $params);

95     //Print the content of cells B2 and C2

```

```
96     $client = getClient();  
  
97     $service = new Google_Service_Sheets($client);  
  
98     $range = 'Presence!B2:C2';  
  
99     $response = $service->spreadsheets_values->get($spreadsheetId, $range);  
  
100    $values = $response->getValues();  
  
101    foreach ($values as $row) {  
  
102        // Print columns A, B which correspond to indices 0,1.  
  
103        printf("%s %s\n", $row[0], $row[1]);  
  
104    }  
  
105 }  
  
106  
  
107 // Delete row 2 from sheet Presence using the method spreadsheets.values.batchUpdate with the  
request content deleteDimension.  
  
108 // Define the variables for the content of the request portion of the method: deleteDomension,  
range, sheetid, dimension, startindex, endindex  
  
109 // Then construct an array with the contents of the request boday ($batchUpdateRequest). Finally  
call the batchUpdate method in a response statement.  
  
110 $client = getClient();  
  
111 $service = new Google_Service_Sheets($client);  
  
112 $sheetid = '526899915';
```

```
113 $ROWS = 'ROWS';

114 $startindex = '1';

115 $endindex = '2';

116 $requests[] = new Google_Service_Sheets_Request(array(
117     'deleteDimension' => array(
118         'range' => array(
119             'sheetId' => $sheetid,
120             'dimension' => $ROWS,
121             'startIndex' => $startindex,
122             'endIndex' => $endindex
123         )));
124 ));

125 $batchUpdateRequest = new
    Google_Service_Sheets_BatchUpdateSpreadsheetRequest(array('requests' => $requests));

126 $response = $service->spreadsheets->batchUpdate($spreadsheetId,$batchUpdateRequest);

127

128 //If the number of rows in sheet Archive exceeds 100, delete row 2 in sheet Archive

129 $client = getClient();
```

```
130 $service = new Google_Service_Sheets($client);

131 // Service object constructed. Define the spreadsheets.values.get method, use the method to
    retrieve the data from Archive and store the result in $values

132 $spreadsheetId = '1X39VbE37fD4IIF9L0Upom8x75m4LEvpWYcPb6Gww5gg';

133 $range = 'Archive';

134 $response = $service->spreadsheets_values->get($spreadsheetId, $range);

135 $values = $response->getValues();

136 //If the sheet Archive contains more than 99 entries delete row 2 (row one contains column
    names, row two the oldest data).

137 if (count($values) > 99) {

138     $client = getClient();

139     $service = new Google_Service_Sheets($client);

140     $sheetid = '1364035202';

141     $ROWS = 'ROWS';

142     $startindex = '1';

143     $endindex = '2';

144     $requests1[] = new Google_Service_Sheets_Request(array(
145         'deleteDimension' => array(
146             'range' => array(
```

```
147     'sheetId' => $sheetid,  
  
148     'dimension' => $ROWS,  
  
149     'startIndex' => $startindex,  
  
150     'endIndex' => $endindex  
  
151     ))  
  
152 );  
  
153     $batchUpdateRequest1 = new  
Google_Service_Sheets_BatchUpdateSpreadsheetRequest(array('requests' => $requests1));  
  
154     $response = $service->spreadsheets->batchUpdate($spreadsheetId,$batchUpdateRequest1);  
  
155 }  
  
156 ?>
```

Figure 15.13 Code for the Google Sheets management script geofencemgm.php

15.6 A Smartphone App for the Geo-Fence Database

The advantage of using Google Documents for our geo-fence database is, that we can utilize the many add-on applications, which come with the platform. As an example we use the add-on AppSheet, which is capable of generating a smartphone app based on our spreadsheet. We go to Google Docs and open our spreadsheet PresenceChris. We select *Add-ons – get Add-ons...*, enter *AppSheet* in the search bar and follow the installation instructions (Figure 15.14). You now can configure which sheets of your spreadsheet are displayed, which columns and whether or not the user shall have read only or read and write access (Figure 15.15). It is even possible to create an access whitelist to control which users can sign in and access your app. You now download the AppSheet app to your smartphone and log in with your Google account credentials. Logging in is only required the first time the app is started. After that, when opening the app on your smartphone, you are directly presented with the content of your spreadsheet, in our case the sheet Archive with the last 100 entries of our geo-fencing database (Figure 15.16).

Add-ons

All ▾ AppSheet



AppSheet
offered by www.appspot.com
Build a custom mobile app from your Google spreadsheet. It is easy, takes only a few minutes, and no coding is involved.



Group Gator
offered by labs.amplifiedit.com
Group Gator extends the power of Google Sheets to simplify the delegation of member management for Google Apps domain groups.



ezShared Contacts
offered by www.cloud34.fr
Manage shared contacts of your Google Apps domain. Add, delete and modify all shared contacts of your domain.

Figure 15.14 AppSheet Add-on for Google Sheets

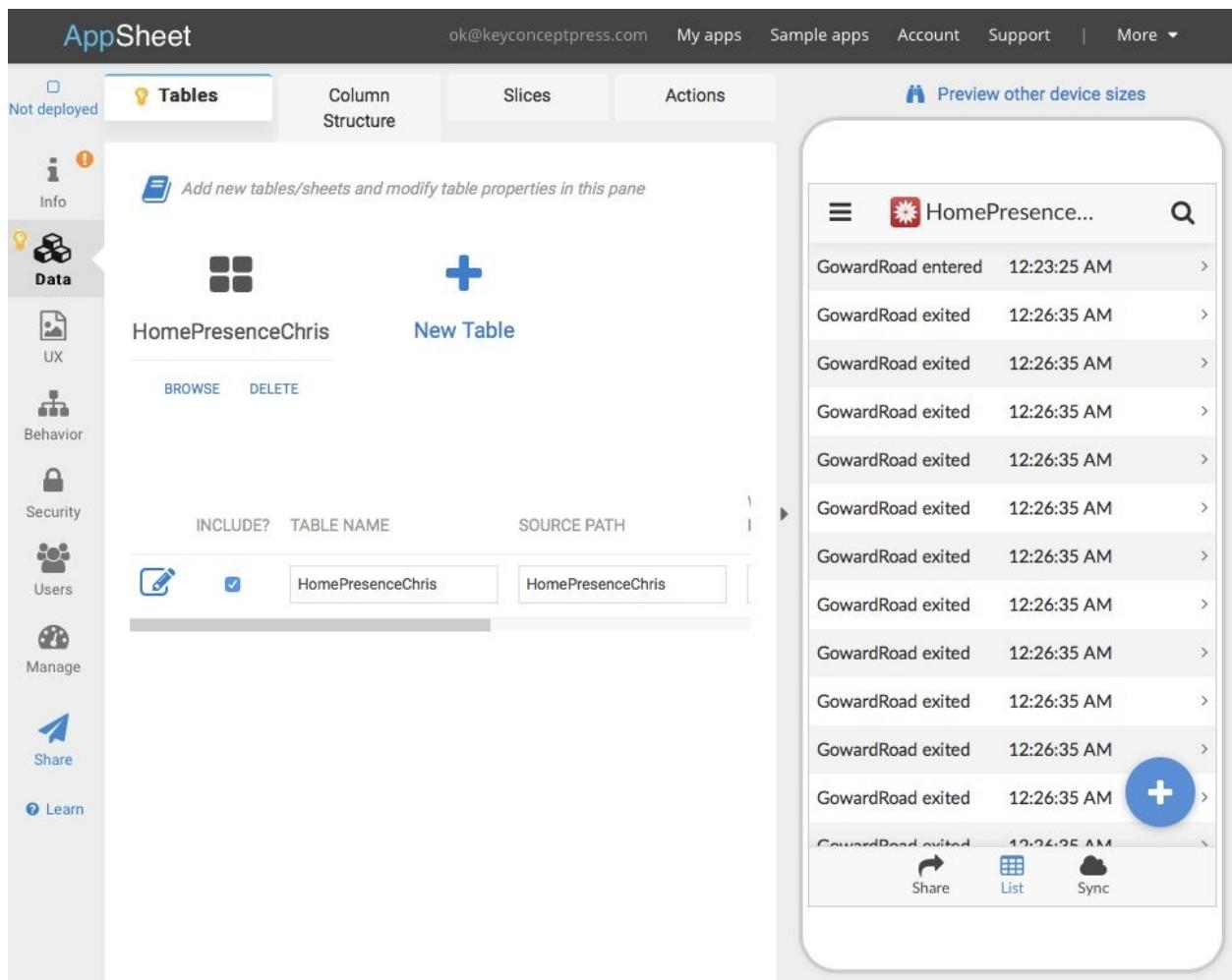


Figure 15.15 Configuring AppSheett

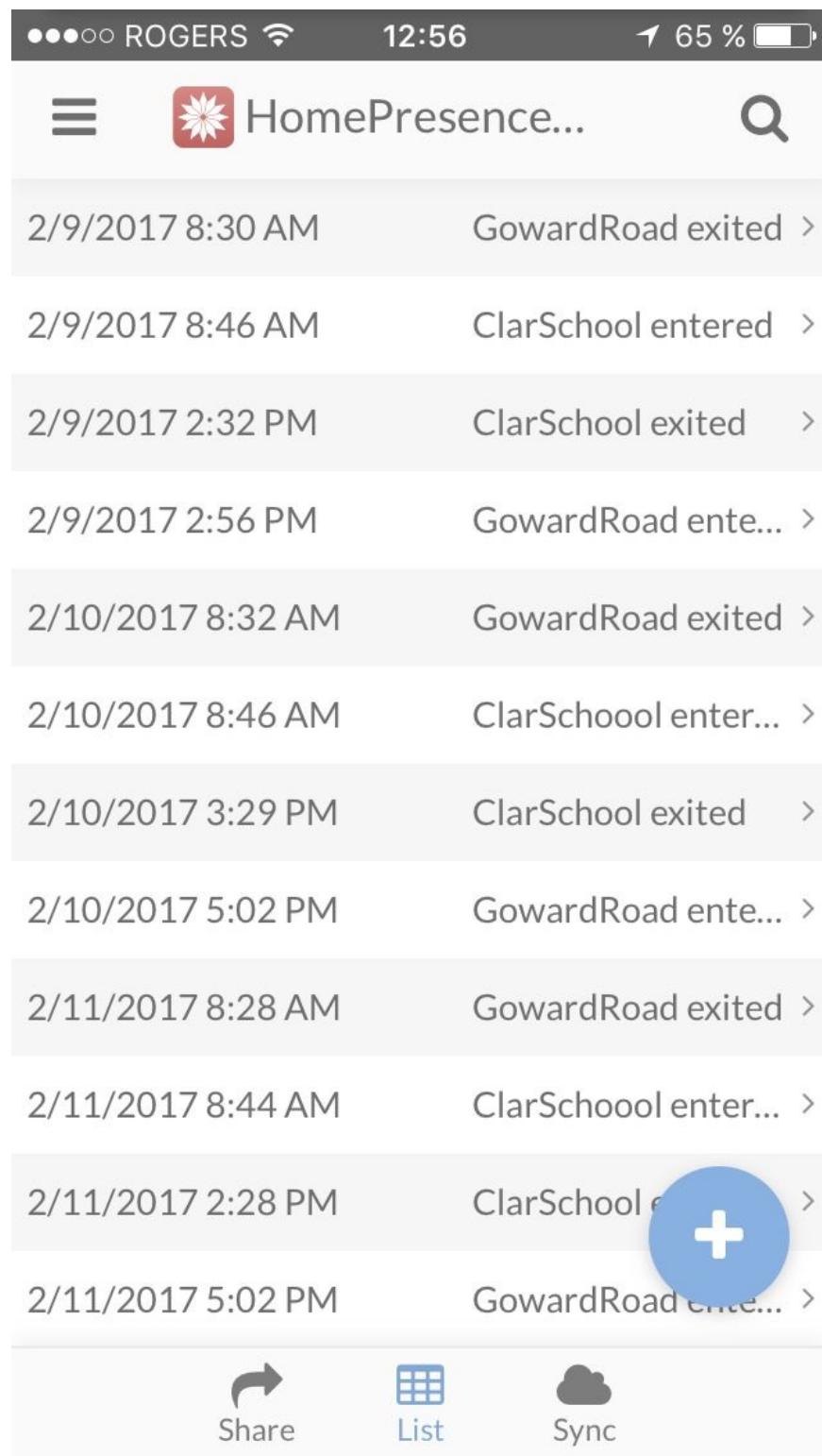


Figure 15.16 Geo-fencing data on a smartphone using AppSheet

15.7 Further considerations for using a Geo-Fencing Database: Concurrency and Hysteresis

If you plan to track multiple devices or persons, it is recommended to create a separate spreadsheet for each of them. Otherwise you would have to deal with concurrency situations, when two or more devices enter or leave a geo-fencing area at the same time. Concurrency control is a prominent subject in database design, but dealing with it is not trivial, and you can easily avoid it with a spreadsheet design, which avoids these scenarios. Defining multiple geo-fencing trackers for a single device, and storing it in one and the same spreadsheet on the other hand is no problem. So if you want to track a child's way to school, you would set up two geo-fence triggers: one for leaving or entering home, and a second one for leaving or entering school. Both trigger events would then be stored in the same sheet of your spreadsheet (Figure 15.16).

A second consideration for geo-fencing triggers relates to the selection of the geo-fence values for entering and exiting. If the geo-fence radius for entering or exiting a location is exactly the same, it can occur, that if the tracked device resides exactly at that radius, that the geo trigger starts to oscillate. This means it fires multiple times, changing its value between entering and exiting. If you design your geo-fence trigger with the IFTTT applet and its built in function [*EnteredORExited*](#), this effect is already being taken care of. However, if you implement separate triggers for entering and exiting an area, make sure to select a different radius for the enter and the exit condition. This so called hysteresis approach is a common design principle in control engineering, and thus of general importance for smart home design. As an example a thermostat controlling the heating system may turn heating on when the temperature drops below a value X, but not turn it off until it rises above a value Y.

15.8 Google Sheets Integration with OpenRemote

Now that our geofencing infrastructure is up and running, we can integrate it with our OpenRemote home controller. For that we create the shell script `geofencestatus.sh`, which runs our php script `geofencemgm.php` and returns the strings `Chris GowardRd exited` or `Chris GowardRd entered`, in case there is an entry: `#!/bin/sh php LibraryWebserver/Documents/geofencemgm.php` Make again sure to specify the full absolute path to the PHP script in the shell script. To detect geofencing events you then create an OpenRemote Command called `presenceChris`, which calls the shellscript `geofencestatus.sh`, and an associated OpenRemote sensor, which polls this command every five seconds. We now have an OpenRemote geofencing sensor, which we can use to trigger actions in our smart home using the Drools rules envrionment as described in chapter thirteen.

15.9 A Cloud Based Smarthome Control Platform

Beyond the above described functionality, you could further enhance the capabilities of the Google Sheets / IFTTT combination and even operate it stand alone for some limited functionality. As an example, using the below URL nomenclature you can send triggers with multiple values to any IFTTT Maker app:

[https://maker.ifttt.com/yourtriggername/with/key/yourkey?
value1=val1&value2=val2&value3=val3](https://maker.ifttt.com/yourtriggername/with/key/yourkey?value1=val1&value2=val2&value3=val3)

You can also easily write a Google Script to use value changes in a Google Sheet to trigger the retrieval of an URL or the sending of an email. As an example, the below Google script sends an email when cells D2 or D3 are changed to contain the value ok:

```
1  /**
2   * add trigger for onedit -
3   * see menu -> Resources -> Current project's triggers
4   */
5  function Initialize() {
6
7  var triggers = ScriptApp.getProjectTriggers();
8
9  for(var i in triggers) {
```

```
10  ScriptApp.deleteTrigger(triggers[i]);  
  
11 }  
  
12  
  
13 ScriptApp.newTrigger("sendNotification")  
  
14 .forSpreadsheet(SpreadsheetApp.getActiveSpreadsheet())  
  
15 .onEdit()  
  
16 .create();  
  
17  
  
18 };  
  
19  
  
20  
  
21 /**  
22 *  
23 */  
  
24  
  
25  
  
26 function sendNotification(e) {
```

```

27
28 if("D2" == e.range.getA1Notation() || "D3" == e.range.getA1Notation()) {
29   if(e.value == "ok") {
30
31
32
33 //Define Notification Details
34 var recipients = "othmar@kyas.net";
35 var subject = "Update"+e.range.getSheet().getName();
36 var body = "cell D2 has been updated";
37
38 //Send the Email
39 MailApp.sendEmail(recipients, subject, body);
40 }
41 }
42 }

```

Figure 15.16 Google Sheet content change trigger script

Chapter 22 contains a brief introduction to JavaScript and it's implementation as Google Scripts, as well a description of the above code. With the above set up our Google Sheets based database could serve as a home controller, which receives data from IFTTT events and which triggers actions through IFTTT commands.

16 Industry Grade Home Infrastructure Control: KNX

16.1 What is KNX?

KNX (short for Konnex) is a European (EN50090, 2003) and international (ISO/IEC 14543-3, 2006) standard for industry grade home and building automation. It specifies how technical building infrastructure such as light, heating, ventilation, shutters, alarm-systems or power-outlets be controlled by switches, sensors, tablets or smart-phones based on a dedicated wireline control infrastructure. By merging three previous standards (EIB, EHS, BatiBus) for building automation KNX achieves compatibility across technologies (light, heating, power-outlets etc.) and vendors. In other words: In an all KNX compliant building any switch can be configured with a few mouse-clicks to control any technology. For example a room-controller from vendor A can be configured to control a heating system from vendor B, dim lights from vendor C and report room temperature to a smart-phone app from vendor D. Since KNX is built upon a dedicated, standardized cable infrastructure it is more expensive (and more reliable) than automation technologies using wireless technologies or the existing electric power lines.

16.2 How does KNX Work?

KNX is build up of the following components:

- A centralized cable infrastructure consisting of :
- all power cabling (light, power-outlets, shutters, etc.) leading to a central control location
- a KNX control cabling infrastructure, connecting all switches and sensors to the same central control location via dedicated, standardized KNX cabling
- KNX actuators, located at the central control location. The actuators conduct the actual control (switching) of the building infrastructure through their connection to the various power cables (dim lights, control heating systems, activate or deactivate wall-outlets, etc.)
- KNX switches and sensors in the building, which are connected through the KNX control cabling infrastructure to the KNX actuators, telling them when to switch which system.
- KNX communication protocol, which is the means of communication between the devices in the KNX network
- KNX-USB module to connect a PC for configuring the KNX infrastructure
- KNX/IP router to integrate the KNX infrastructure with IP other IP devices (smartphones, tablets)

Through this concept, the control layer and the building engineering layer are physically and logically separated. With that any switch can control any device independent of its location. As a result, the operation of the building engineering infrastructure can be controlled in a much more sophisticated and flexible way compared to traditional infrastructure. KNX switches, KNX sensors, KNX devices and KNX actuators communicate using the KNX protocol, which uses IP as transport protocol. Via a KNX-USB module the PC with ETS software is connected to the KNX network. If, as in our project, additional control and monitoring functions via networked devices (smartphones, tablets) are desired, a

KNX/IP router module is needed. The devices - in our example the OpenRemote controller - accesses the IP address of the KNX/IP router and through it is capable of sending KNX commands or reading KNX status messages, as we will see further down.

16.3 The KNX Software Infrastructure: ETS

ETS stands for Engineering Tool Software and is a manufacturer independent software application developed by the KNX-Organization (<http://www.knx.org>). It has been designed to configure KNX based building control installations. There are also vendor specific implementations for KNX configuration and installation, however the KNX ETS tool is widely used in the industry and the quasi-industry standard for KNX installations. Since the KNX API is standardized, any KNX software should work with any KNX certified component. The individual software functions of the KNX components are provided via product libraries, which the vendors typically offer as free downloads from their websites. ETS in its current version 5 runs on the Windows operating systems and is available in three licenses:

- ETS5 Demo: a free test and trial version for very small test projects with up to three KNX devices only
- ETS5 Lite: a low cost license (about 200€) for small to mid-range projects with up to 20 KNX devices
- ETS5: the full professional license for all projects sizes and functions (about 100€)

Previous versions (you might run into) were (ETS1: 1993-1996, ETS2: 1996-2004, ETS3: 2004-2010, ETS4: 2010).

ETS5 was released in 2014 with higher performance (in particular for larger projects) and several usability improvements compared to ETS4. Previous ETS versions can be upgraded to ETS5, which however requires the usage of a hardware dongle.

ETS can be purchased and downloaded directly from the knx.org web site. While ETS is used for KNX configuration, the KNX infrastructure can also be

monitored and controlled via smartphone, tablet or PC. Among other tools our project platform OpenRemote also supports the KNX protocol. After configuring our infrastructure with ETS thus we will integrate the KNX components into our OpenRemote based project.

16.4 Which Operating Systems does ETS Support?

The following Windows environments are supported: Windows XP Professional; SP3 (32 Bit), Windows XP Professional; SP2 (64 Bit), Windows Vista; SP2 (32/64 Bit), Windows 7/8; SP1 (32/64 Bit), Windows Server 2003; SP2 (32/64 Bit) Windows Server 2003 R2 SP2 (32/64 Bit), Windows Server 2008; SP2 (32/64 Bit), Windows Server 2008 R2; SP1 (32/64 Bit) The required computer hardware requirements are: CPU: \geq 2 GHz, RAM: \geq 2 GB, HDD: \geq 20 GB, RES: \geq 1024 x 768

16.5 ETS on a Mac

While the KNX-Forum does not officially support ETS running in virtual environments it does work without problems. I have been running ETS4 and ETS5 on macOS using Parallels Desktop as under Windows 7/8/10 with no issues.

16.6 Other KNX.org Software Tools

There are also some other software tools, which are provided by the KNX organization:

- iETS Server is a gateway application creating an interface between KNX and the IP world using the EIBlib/IP protocol (other than KNXnet/IP). It allows iETS clients to remotely connect to the KNX network for maintenance and troubleshooting purposes. The software dates back from 2003. In the meantime many new tools from a number of vendors (e.g. aycontrol.com) are available, so you probably do not want to go with this.
- Falcon is a network connection library for ETS, and is used by software developers to write KNX device drivers.
- EITT is the KNX Interworking Test Tool used to test the interoperability of KNX devices by manufacturers
- KNX Manufacturer Tool is used by manufacturers of KNX hardware for the creation of the ETS product database entries of their devices.

16.7 ETS5 Installation

The ETS5 installation file can be downloaded (for free) directly from knx.org. Its size is about 500 Mbytes. After decompressing (unzip) the downloaded file you run the resulting .exe file. The ETS install procedure automatically sets up all required components (Figure 16.1).

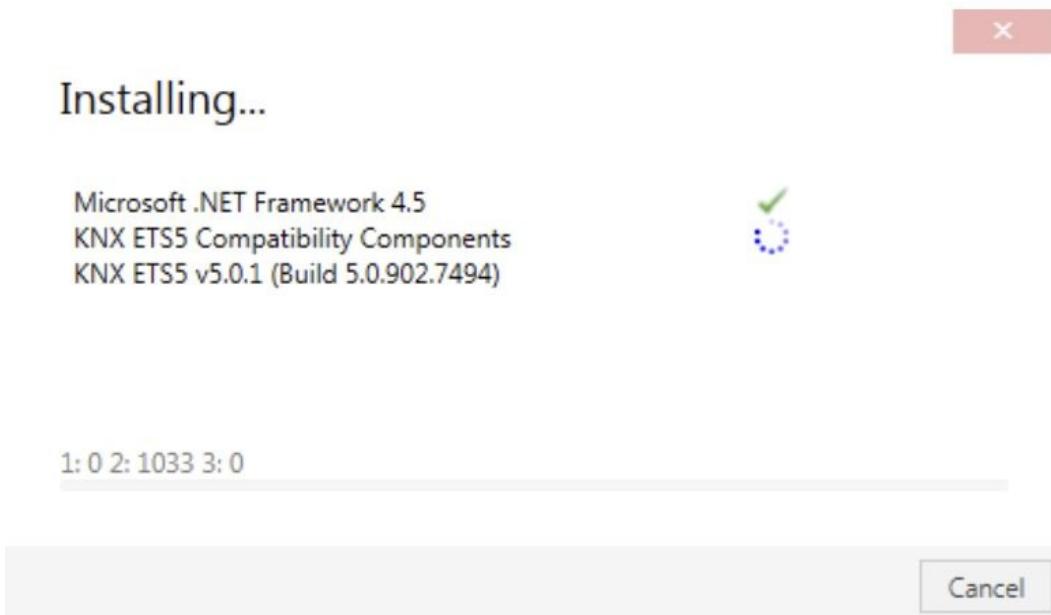


Figure 16.1 Installing ETS5

After successful installation you can immediately run ETS without any license key as a demo version, which restricts the maximum number of KNX devices to three. If you have purchased a Lite or Professional license insert the KNX USB dongle into your computer and wait until Windows finishes installing the required device drivers (Figure 16.2, 16.3).



Figure 16.2 The KNX ETS5 Dongle

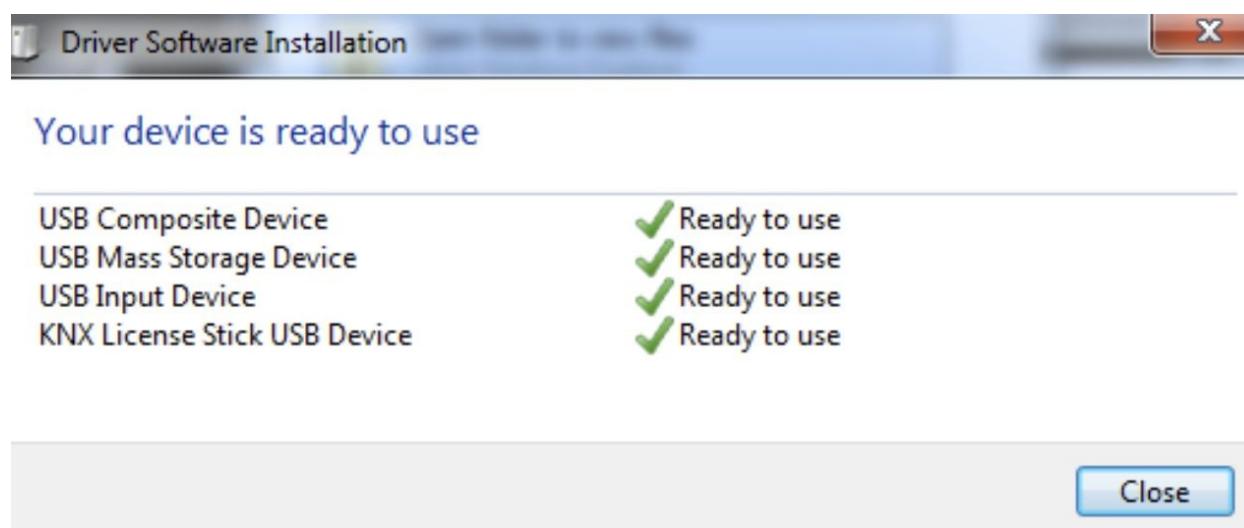


Figure 16.3 Installing the ETS5 dongle

When you start ETS5 it's license status is displayed at the bottom of the ETS5 user interface. Now click on the *Licenses* button at the bottom of the GUI and the licensing window opens. Now copy the dongle ID which is displayed on the left column of the licensing window to the Windows Clipboard. With the ID you can now download the required license file from your account at the KNX.org website. Go to *My Account – My Products*, enter the dongle ID and select Add Key. The product key field now turns into a link, which you can use to download the license file (format *.license) to your computer. Now you can go back to the ETS5 license window and select the green *+ button*. Browse to the license file, select it and click on *Open*. Now your license is activated and displayed at the bottom of the GUI (Figure 16.4).

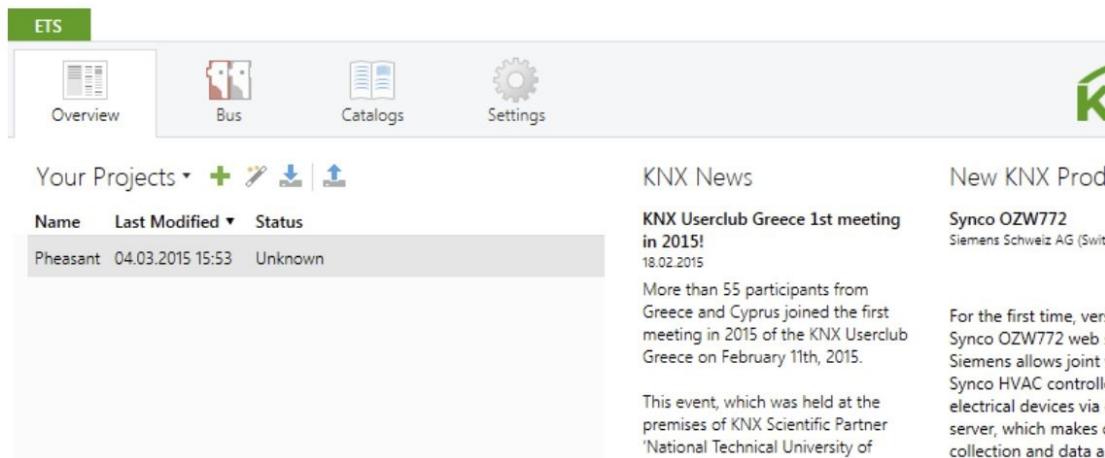


Figure 16.4 The ETS5 main screen

16.8 Importing Vendor Catalogs

For each KNX device you use in your project, you need to download and install its product specific KNX library file (catalog file). Some vendors offer a single file with all product libraries they provide, for others you need to individually locate and download the catalog files for the products you need. Older ETS catalog files have the extensions .vd3 or .vd4, new files (in XML format) carry the extension .knxprod. Once you have downloaded the catalog files you open the *catalog* menu in ETS5, select the vendor files and ETS will either convert them to XML (in case of older files such as vd3 or vd4) or it will directly import them into the ETS catalog (Figure 16.5).

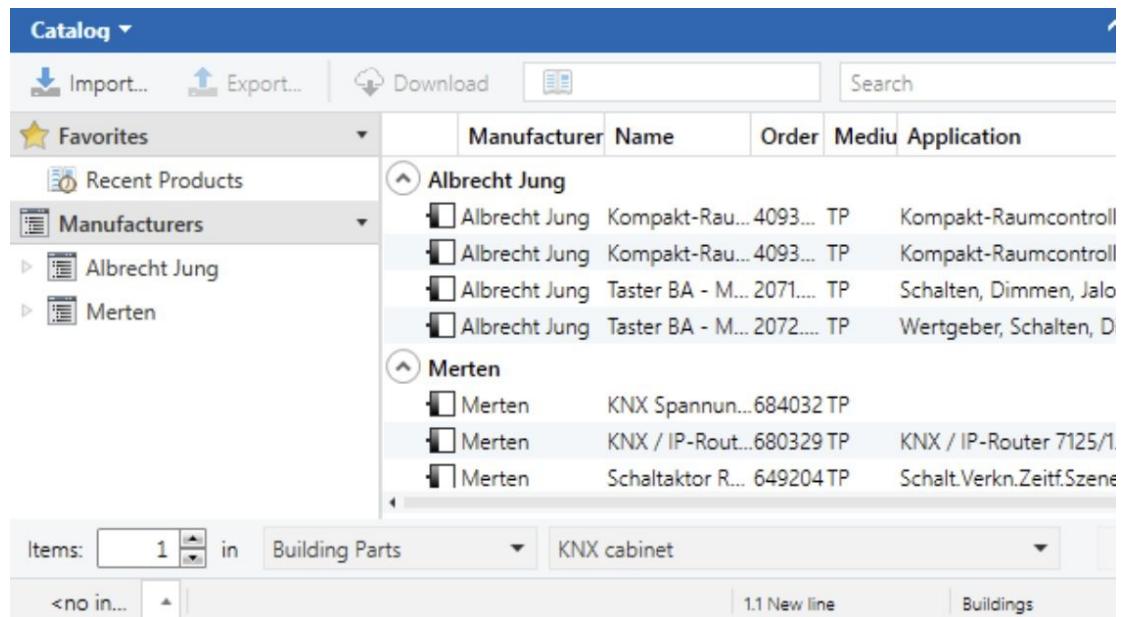


Figure 16.5 Vendor entries in the ETS5 catalog menu

16.9 ETS5 Infrastructure Configuration

In order to set up a KNX infrastructure you need to go through the following five steps:

- create new project and the building topology in ETS
- add the KNX elements to the topology
- create logical functions (with KNX addresses as identifiers)
- connect KNX devices through the logic functions (e.g. a KNX switch with a KNX dim actuator)
- download the configuration to the KNX hardware
- The first step is to create a new project (Figure 16.6).

New project (1) Import Date: 10.03.2015 Last Modified: 10.03.2015 15:15

Details Project Log Project Files

Name: New project (1) Password:

Project Number: BCU Key:

Contract Number: Codepage: US-ASCII

Start Date: Group Address Style: Free
 Two Level
 Three Level

End Date:

Status: Unknown

Comment:

Compatibility: Hide extended group address range for plugins

Figure 16.6 Starting a new project in ETS5

16.10 ETS5: Adding the Building Infrastructure

We start by adding a building, and assigning it a name. Then we add the floors by right clicking on the building entry. We create three levels and call them basement, first floor and second floor. Then we configure (again through menus, which pop up through a right-mouse-click) rooms, the corridor and in the basement the KNX cabinet (Figure 16.7).

The screenshot shows the ETS5 software interface. On the left, there is a tree view under the 'Buildings' tab. It includes 'Dynamic Folders', a folder named 'pheasantroad4' containing 'basement', 'first floor', and 'second floor', and a 'Trades' section. The 'Trades' section is currently selected. On the right, there is a detailed table with columns for Address, Room, Description, and Application Program.

	Address	Room	Description	Application Program
1.1.-	KNX cabinet		KNX Power Supply	
1.1.0	KNX cabinet	IP Router	KNX / IP-Router 7125/1	
1.1.1	KNX cabinet	HK 1	Schalten PWM 2067/0.1	
1.1.2	KNX cabinet	HK 2	Schalten PWM 2067/0.1	
1.1.3	corridor		Switch and dim li...	Schalten, Dimmen, Jalousien
1.1.4	bathroom			Kompakt-Raumcontroller
1.1.5	bedroom1			Kompakt-Raumcontroller
1.1.6	bedroom2			Kompakt-Raumcontroller
1.1.7	corridor	Flurtemp		Kompakt-Raumcontroller
1.1.8	diningroom			Kompakt-Raumcontroller
1.1.9	familyroom			Kompakt-Raumcontroller
1.1.10	livingroom			Kompakt-Raumcontroller
1.1.11	KNX cabinet	DK 1		Universal Dimmen 3244
1.1.12	KNX cabinet	DK 2		Universal Dimmen 3244
1.1.13	livingroom		Next to bedroom1	Schalten, Dimmen, Jalousien
1.1.14	livingroom		Next to bedroom2	Schalten, Dimmen, Jalousien
1.1.15	bedroom2		Schalter Wintergarten	Schalten, Dimmen, Jalousien
1.1.16	KNX cabinet	230V Switch		Schalt.Verkn.Zeitf.Szene
1.1.17	corridor		Hauseingang Ausgang	Wertgeber, Schalten, Dimmen

Figure 16.7 Adding building, floors and rooms in ETS5

16.11 ETS5: Configuring the KNX Elements

Once we are done with the building layout, we can start adding the KNX modules (actuators, switches, sensors) we want to use. For this we need to select each room and add its KNX components by selecting it from the vendor catalog listing in ETS. Once we are done with the rooms we add the KNX devices (actuators, power-supply, IP-Router, etc.), which we have installed in the central KNX cabinet (Figure 16.8).

Buildings			
		Add Delete Download Info Reset Unload	
Buildings	Address	Room	Description
Dynamic Folders	1.1.-	KNX cabinet	KNX Power Supp
pheasantroad4	1.1.0	KNX cabinet	IP Router
basement	1.1.1	KNX cabinet	HK 1
KNX cabinet	1.1.2	KNX cabinet	HK 2
first floor	1.1.11	KNX cabinet	DK 1
bathroom	1.1.12	KNX cabinet	DK 2
1.1.4 Kompakt-Raumc...	1.1.16	KNX cabinet	230V Switch
bedroom1			
bedroom2			

Figure 16.8 Configuring the devices in the KNX cabinet

Depending on the type of element and the individual functionality, for every KNX component we need to set a number of parameters. For example in case of a switch we can configure it to function as a simple light switch, or a more sophisticated dimmer. Figure 16.9 shows the configuration screen of the Tuya

superimposed dimmer. Figure 16.9 shows the configuration screen of the Jungs Compact Room Controller. The switches left and right to the LED display can be used for room temperature control, switches T1 through T14 for light control, shutter control, wall-outlet control, presence detection or scenarios combining multiple controls.

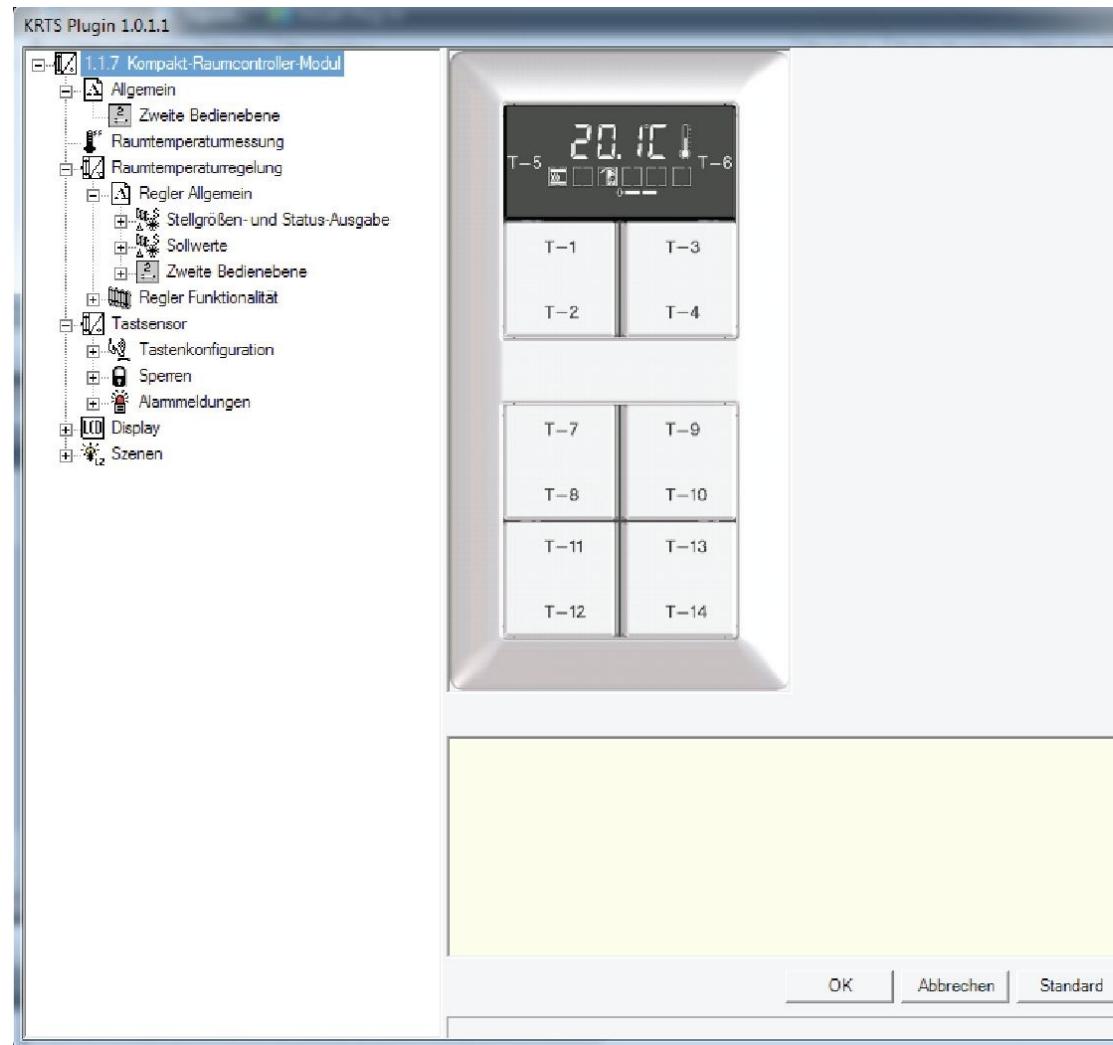


Figure 16.9 Setting parameters for a room controller

The control parameters for the heating system are typically the most complex to understand and define. While you can get started with default parameters in many other cases, for the heating system you need to take the time to understand

what you are doing. Most importantly you have to make the right selection between underfloor heating and central heating, each of which operates at different water temperatures. And you need to set the correct control algorithm - continuous PI or switched PI (PIW). (Some systems also offer a two point PI mode). Switched PI, which is used for underfloor heating systems, just sends one single Bit to the according heating actuator and its connected valves, just sending an open or close command. Continuous PI sends its control message in form of an entire byte, telling the actuator and valve how far to open or close. These systems are often used in larger central heating configurations. To a certain extent ETS prevents basic misconfigurations. For example is it not possible, to assign a one byte heating control command to a heating actuator, which can only process binary (1 bit) valve commands - open or close. ETS will respond with an error message and refuse the connection to be made.

16.12 ETS5: Connecting Infrastructure to Controls

At this point there are three steps left to do, until we can actually start testing our configuration:

the assignment of KNX addresses for each function

the assignment of sensors, switches and actuators to the functions represented by KNX addresses

the actual programming of the KNX hardware by downloading the configuration from the ETS software to the KNX devices

The group addresses are logical addresses, which are assigned to each function in the building, *e.g.* switch on light bathroom or dim light bedroom). For better organization we start by defining three address groups (Switch to the *Group Adresses* menu and select the green + sign), which shall contain the controls for related functions (Figure 16.10):

heating

light

power sockets

For each group we add the individual functions we want to implement in the group address menu. The actual KNX addresses for the functions are selected

automatically by ETS (Figure 16.11). Now for each KNX address (which is equivalent to a function) we need to assign the devices (switches and actuators), which actually realize the functions. This is simply done by dragging the devices from the *Devices* pane into the *Building* window (Figure 16.12).

Group Addresses ▾			
+ Add X Delete - Download ▾ i Info ▾ R Reset U Unload ▾ S Search			
	Middle	Gr	Name
▶ D Dynamic Folders		0	first floor
▶ G 0 heating		1	Heating Mode Change
▶ G 1 light			
▶ G 2 power sockets			

Figure 16.10 Creating Address Groups in ETS

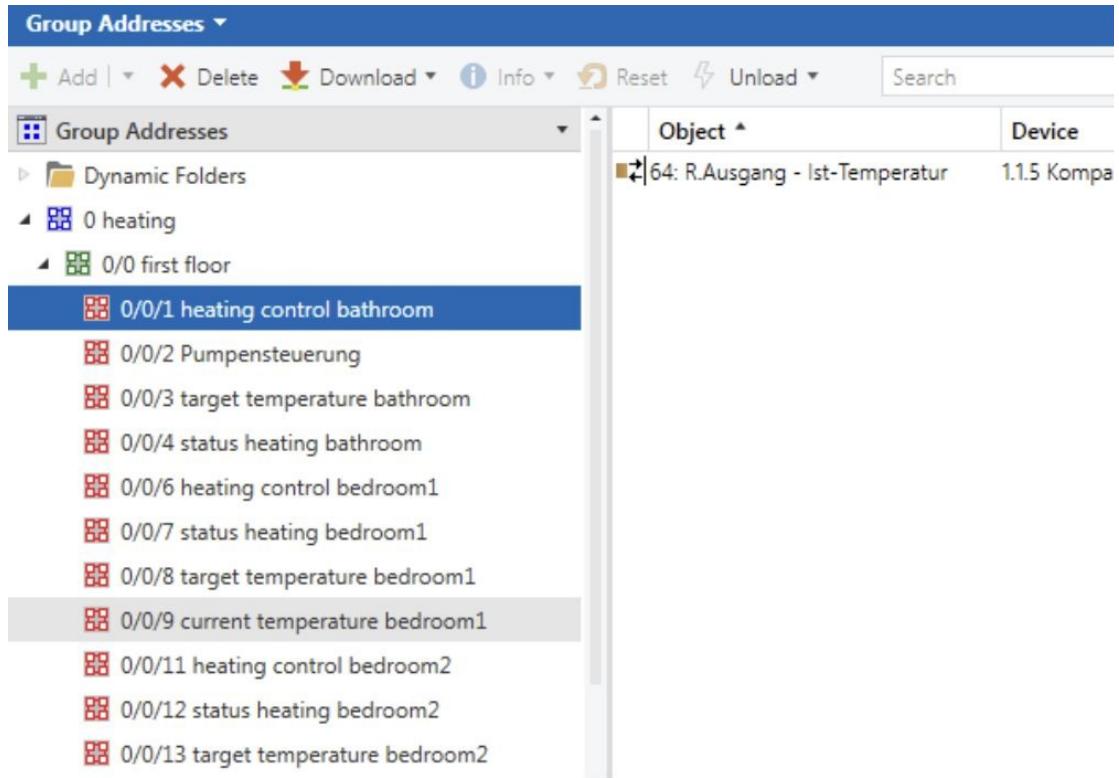


Figure 16.11 Configuring KNX Addresses in ETS

Once we have configured all desired functions, we need to download the configuration from ETS to the individual switches, sensors and actuators. Via an IP Router or a KNX USB module we connect our PC to the KNX network. Then in the *Building* menu we select the first device we want to program, right click it and in the pop-up-menu select *Download All*. Now we need to physically go to the device we want to configure and press the program button. The ETS software will now start download to the target device. Once we are finished with all the devices of a given KNX address we can do the test and switch on our device.

Buildings

Add | Delete | Download | Info | Reset | Search

KNX cabinet

- 1.1.- KNX Power Supply KNX Spannungsv...
- 1.1.0 IP Router KNX / IP-Router REG-K
- 1.1.1 HK 1 1 bedroom 1 2 bedroom 2 3 co...
- 1.1.2 HK 2 1 livingroom Heizungsaktor RE...
- 1.1.11 DK 1 C1 family room C2 bathroom...
- 1.1.12 DK 2 C1 bedroom2 C2 diningroom...
- 1.1.16 230V Switch Channel 1 Aussensteck...

first floor

- bathroom
- 1.1.4 Kompakt-Raumcontroller-Modul

bedroom1

- 1.1.5 Kompakt-Raumcontroller-Modul

bedroom2

corridor

diningroom

familyroom

kitchen

livingroom

second floor

Trades

	Address	Room	Description
1.1.-	KNX cabinet	KNX Power Supply	
1.1.0	KNX cabinet	IP Router	
1.1.1	KNX cabinet	HK 1	
1.1.2	KNX cabinet	HK 2	
1.1.3	corridor	Switch and dim livir	
1.1.4	bathroom		
1.1.5	bedroom1		
1.1.6	bedroom2		
1.1.7	corridor	Flurtemp	
1.1.8	diningroom		
1.1.9	familyroom		
1.1.10	livingroom		
1.1.11	KNX cabinet	DK 1	
1.1.12	KNX cabinet	DK 2	
1.1.13	livingroom	Next to bedroom1	
1.1.14	livingroom	Next to bedroom2	
1.1.15	bedroom2	Schalter Wintergart	
1.1.16	KNX cabinet	230V Switch	
1.1.17	corridor	Hauseingang Ausse	

Devices Parameter Building Parts

Figure 16.12 Assigning devices to KNX addresses

16.13 Notes on Configuring KNX Devices

There are a couple of things which are important to understand when configuring KNX devices:

The parameters which are available for the configuration of KNX objects in the menu (room controllers, heating actuators etc.) often depend on the general settings for that object. For example the important parameter *dimming value status* for reading out the current setting of a dimmer is only available on a dimming actuator, if in *General Settings* the *status value object* is activated.

Similarly for a heating actuator the control parameter expected by the heating actuator needs to be set in the general settings for the object (*1 Bit switching value* or *1 Byte continuous control value*).

Another example how tricky it sometimes can be, to configure complex devices, is the Jung compact controller display. Depending on a button being configured as *push-button* or *rocker*, different configuration values are possible. For example important functions such as change of operating mode (*comfort*, *night*, *stand-by*) or change of *set point temperature* can only be assigned to a button, which is configured as *push-button*!

Another important configuration setting is the read flag. If you want to read out the value of a device, the reading flag has to be set. Often the reading flag is not set by default. (Figure 16.13)

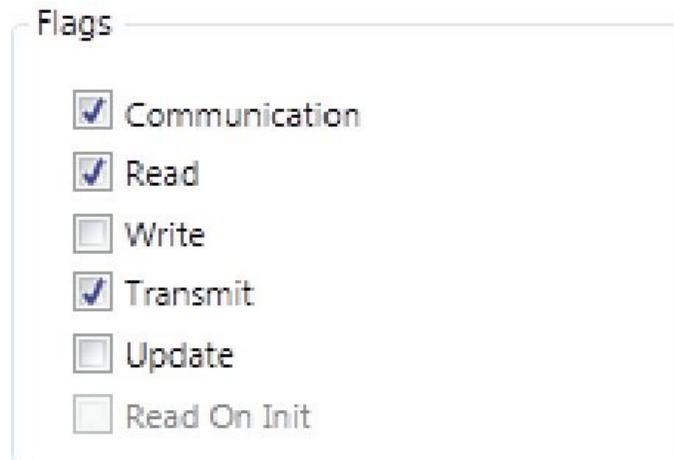


Figure 16.13: Setting the KNX Read Flag in order to be able to retrieve values

To summarize, especially with more complex devices containing multiple objects, one should carefully look at general settings and flags, watching out on possible impacts of their settings on secondary menu options or functionality. Also once done, taking notes and writing a brief documentation is not a bad idea for later reference.

17 KNX Control via OpenRemote Designer

When entering your KNX configuration into OpenRemote Designer the first step is to import the KNX Group Addresses. For this purpose we first create a CSV-export file from our ETS installation. We select the top folder of the Group Addresses in the ETS software, right-click and select *Export Group Addresses* in the pop-up-menu (Figure 17.1)

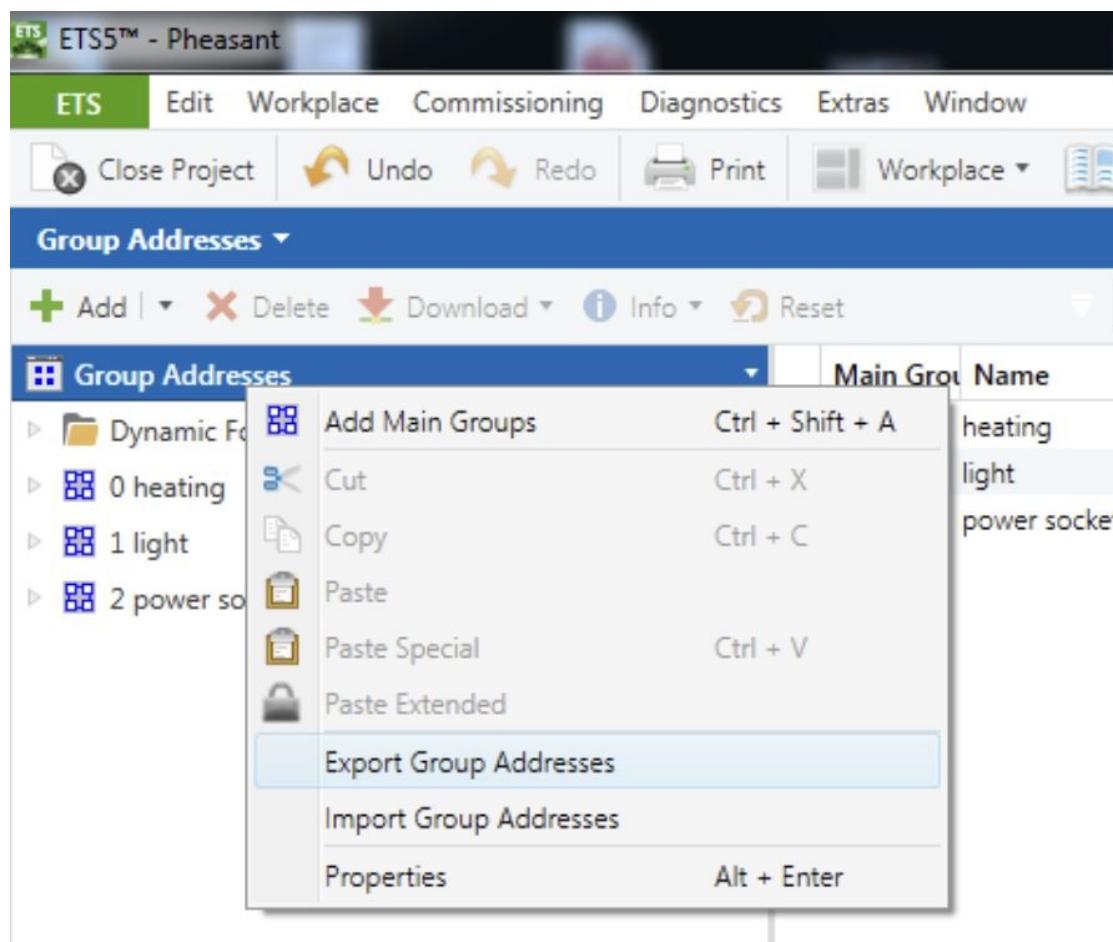


Figure 17.1 Export Group Addresses in ETS5 (1)

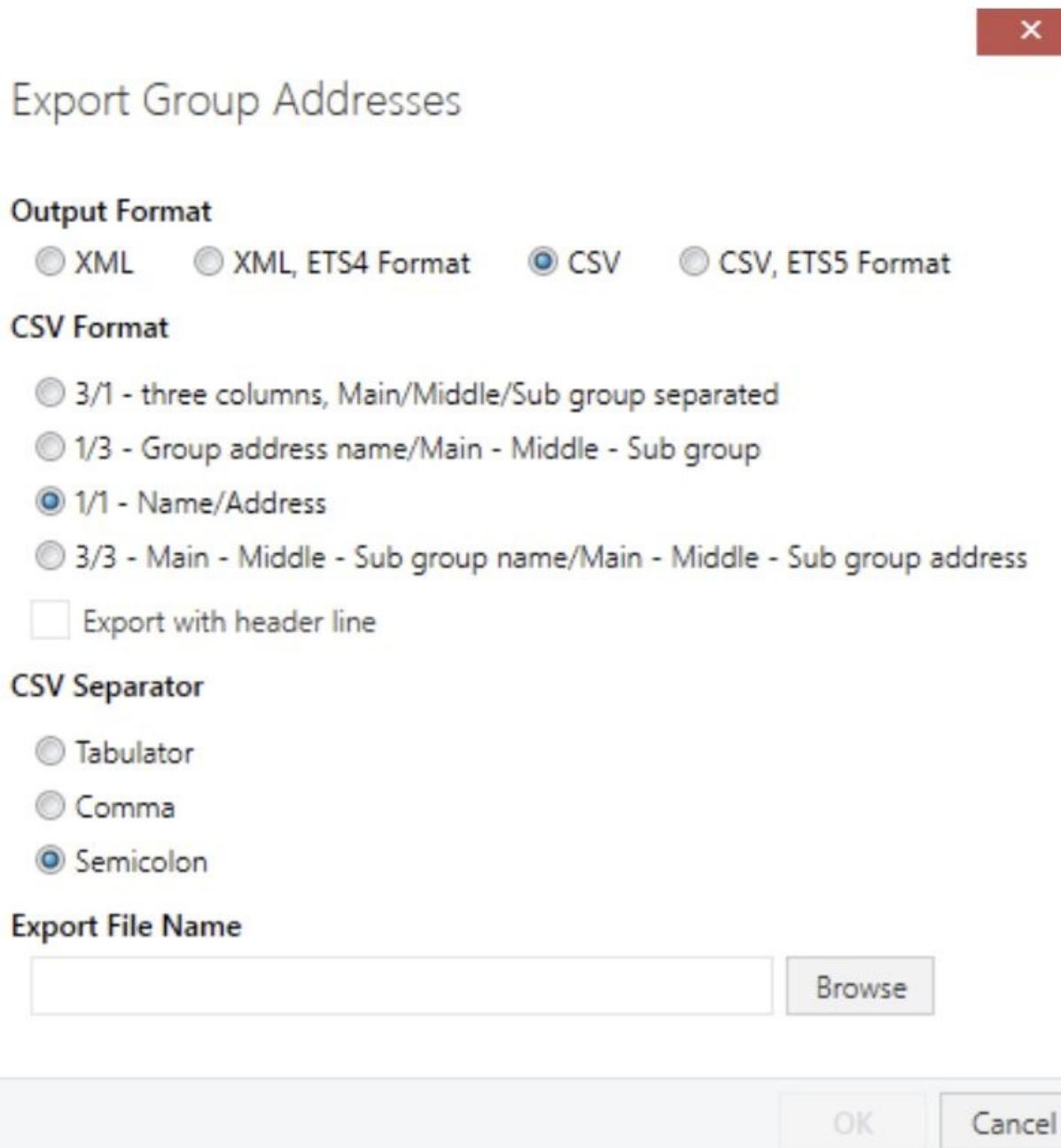


Figure 17.2 Exporting Group Addresses in ETS5 (2)

In the window which appears we select *CSV, 1/1, Semicolon* and click *OK*. (Figure 17.2). This will get us a simple CSV-file. Before importing, we open the CSV-file with a text editor and make sure no other characters appear. Several KNX versions frame the information elements with quotes. If this is the case, we need to delete those until the entries look like the following:

Switch light dining room;1/0/16

Value light dining room;1/0/17

Dim light dining room;1/0/18

Dim value light dining room;1/0/19

Switch light family room;1/0/20

In OpenRemote Designer as the first step we then create a new device selecting *New – New Device* and giving it a name, for example IP KNX Router. (The naming conventions for the devices are just of administrative nature, and do not have an impact on functionality). We mark our device and can now import our ETS Group Addresses by selecting *New – Import* ETS data and importing our CSV file as shown in Figures 17.3 and 17.4.

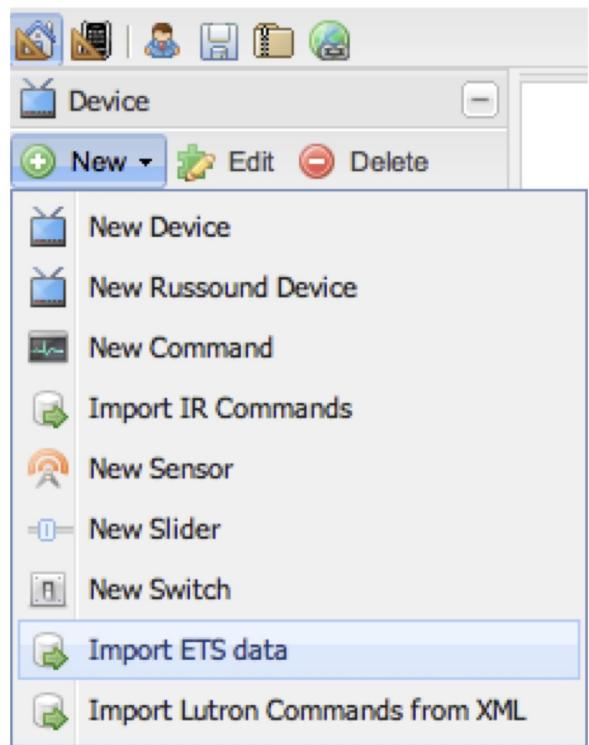


Fig. 17.3 Importing ETS data to OpenRemote

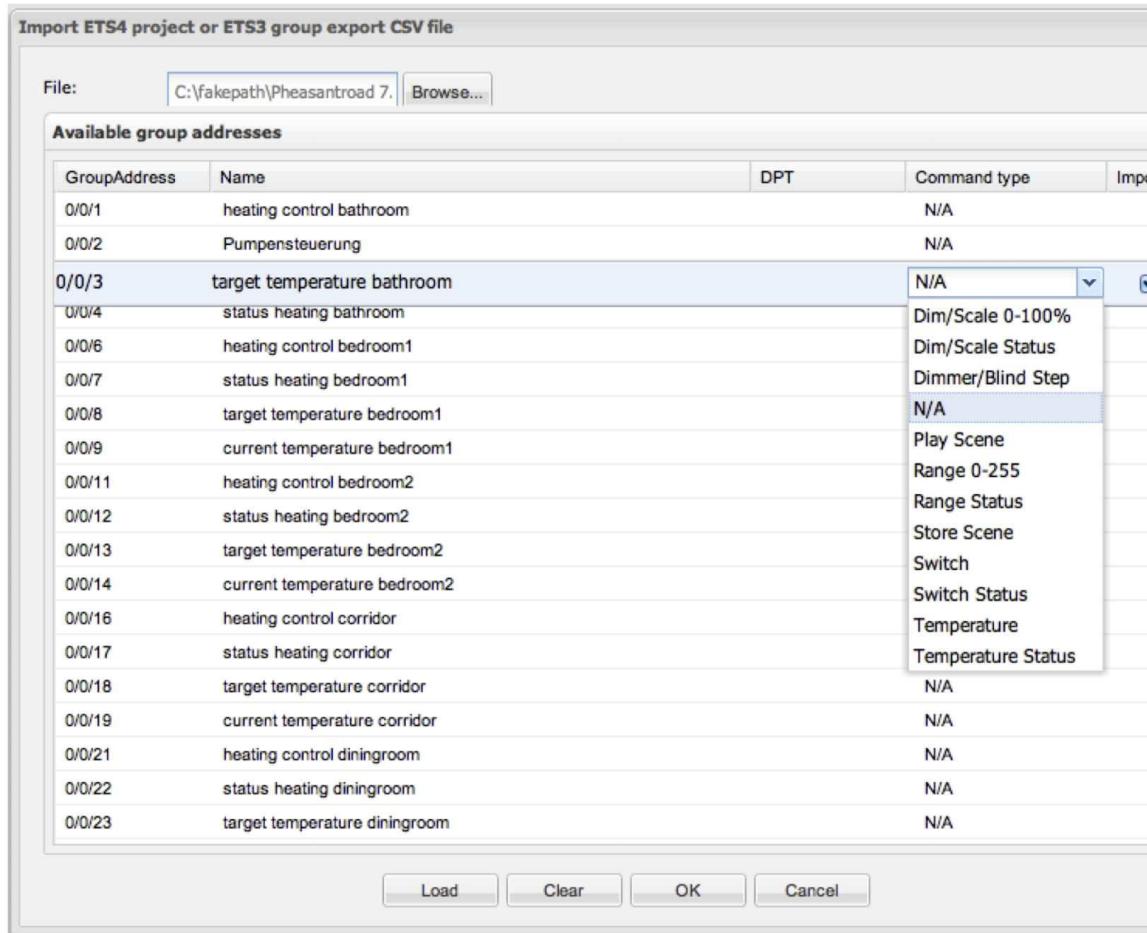


Fig. 17.4 Open Designer Import Dialogue for ETS data

After selecting the **Load** command, the ETS data appears in the Open Remote Designer import window. We now need to select the addresses we want to import and assign a command type for each group address. Selecting the correct command type is essential for the functioning of our application. For Group Addresses, which just report data (temperature, range of values, state of a dimmer or a switch) we need to select the appropriate **Status** command type (Temperature Status, Switch Status, etc.). For Group Addresses, which actually initiate an action (e.g. activate a switch, change the dim value or temperature) we need to select the according command type (e.g. Temperature, Switch or Dim/Scale 0-100%). Once we are done we hit OK and your selected KNX commands appears on the left side of our Open Remote configuration screen

with the correct KNX command and the correct KNX Data Point Type (DPT) selected. (Fig. 17.5)

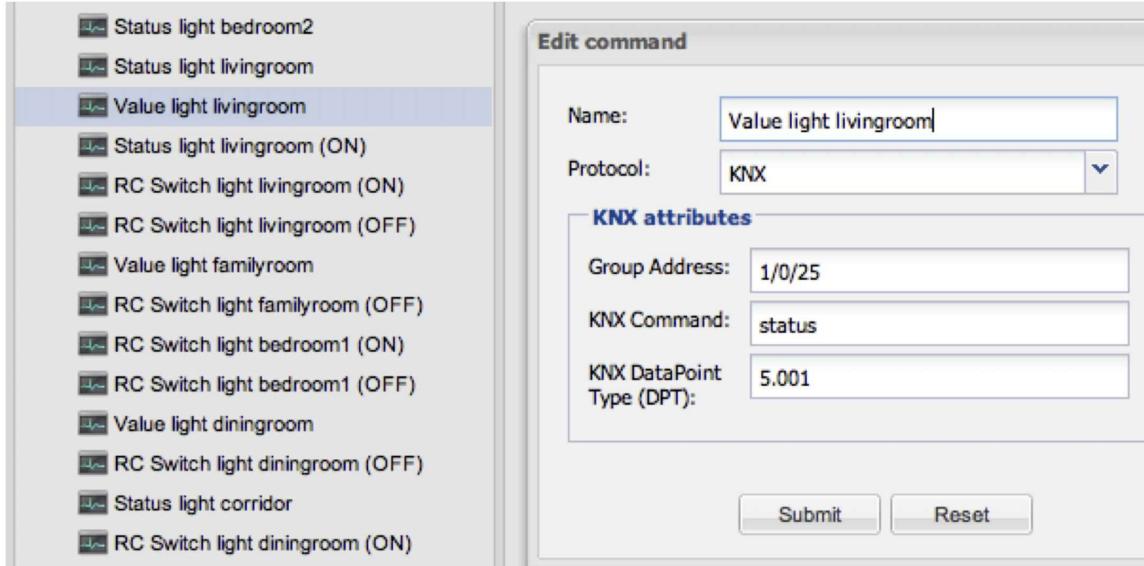


Figure 17.5 Configuring a dim status command: Report the light value from the living room

Alternatively we can also manually add KNX commands by selecting [New – New Command](#) or edit imported data records. For a functioning configuration it is essential to configure the correct DPTs and commands for each entry. Table 17.1 shows a listing with some important DPTs, Figure 17.6 the command editing dialogue in OpenRemote Designer.

Option	Description	
DPT ID	Format	DPT Name
1.001	B1 (one bit Boolean)	
2.001	B2 (two bit Boolean)	DPT_Switch_Control
5.001	U8 unsigned integer 8 bit	DPT_Scaling
6.001	V ₈	DPT_Percent

7.001	U16 (unsigned integer 8 bit)	DPT_Value_2_U_Count
8.001	U16 (unsigned integer 8 bit)	DPT_Value_2_Count
9.001	F16 (floating point 16 bit)	DPT_Value_Temp
10.001		DPT_TimeOfDay
11.001		DPT_Date

Table 17.1 Selected KNX Data Point Types

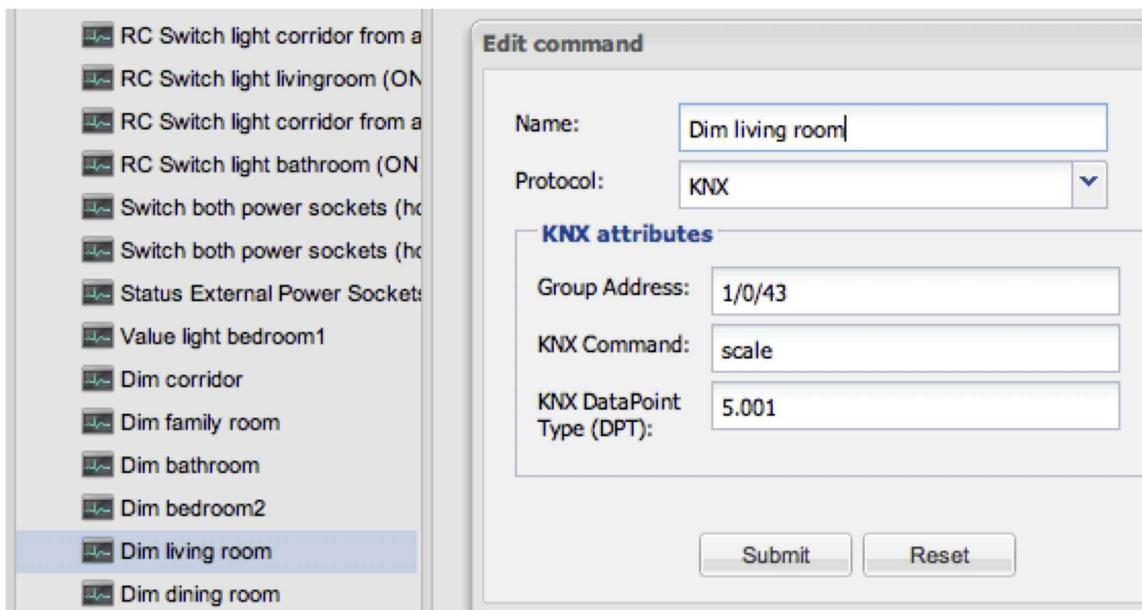


Figure 17.6 Configuring the dim command using “scale”: Dim living room light

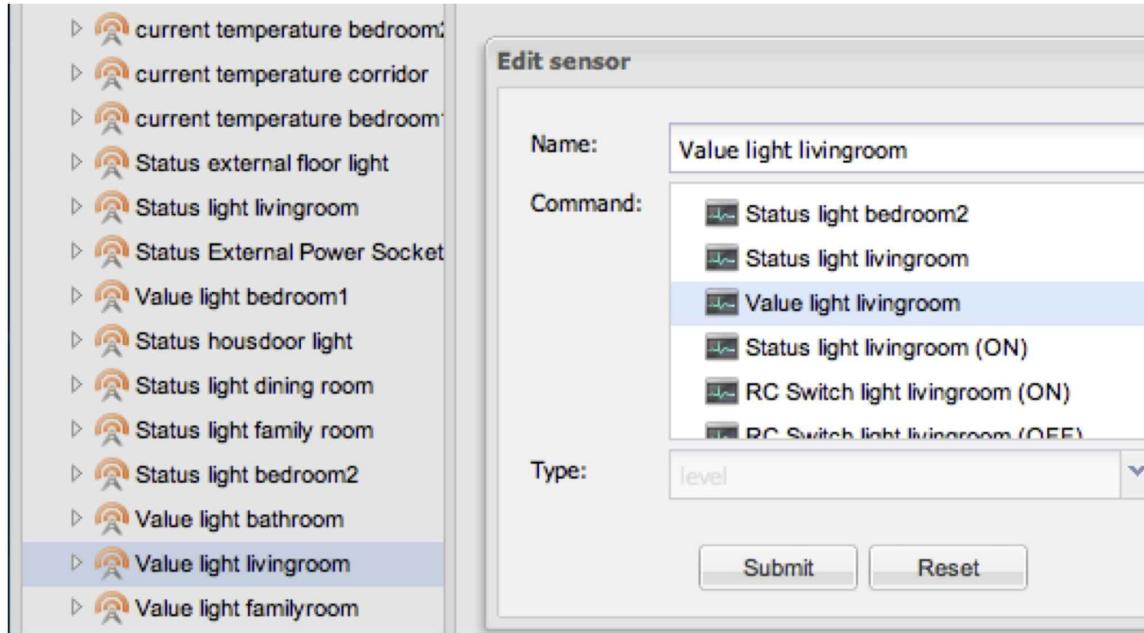


Figure 17.7 Configuring a sensor: Dim sensor living room



Figure 17.8 Configuring a slider: Dim slider living room

Once we are done with the commands, we need to configure sensors for each of our functions (switches, sliders, value displays). We select *New – New Sensor* and select the according status command for it. The final step is then to configure switches and sliders. For a slider to dim light we for example select

New – New Slider and then add both - the according action command as well as the according sensor containing the light brightness value.

In our example we want to configure a slider to dim the light in the living room. We first need the status command reporting the current value of the light in the living room. (Figure 17.5). Second we need the command which actually dims the light. The according scale command is shown in figure 17.6. Third we configure the sensor for our dim slider (Figure 17.7), which uses the status command from figure 17.5. And finally we configure the slider itself (Figure 12.8), consisting of the sensor and the scale command “Dim living room”.

To clarify the KNX ETS side of things at this point as well: The dim status KNX Group Address (in our example 1/0/25, Fig. 17.9) in the ETS software links to the *Status feedback value object/brightness* of the KNX dim actor unit, and has also, as you can see in figure 17.5, a DPT of 5.001. Similarly the dim command through KNX Group Address 1/0/43 (Figure 17.6) links on the KNX side to the *General value object* of the dim actor, which allows the setting of the dim level.

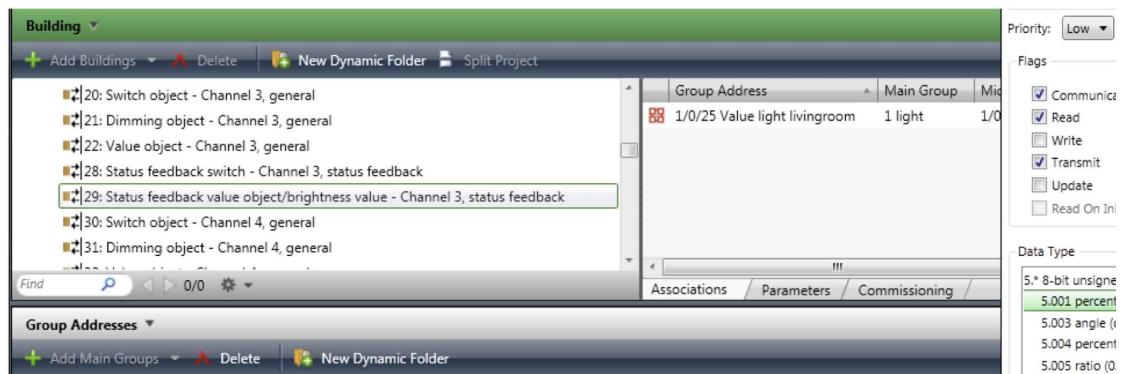


Figure 17.9 KNX Group Address and linkage to the dim actor value brightness object in ETS 4

In the same fashion as explained above we now configure switches or simple sensors for displaying status information. We can now move on and design the

~~controls for displaying status information. We can now move on and design an~~
OpenRemote GUI for our controls as shown in previous chapters.

17.1 Background Pictures for the Smartphone and Tablet App

Besides defining your own GIFs for buttons, sliders and other control element, you can also upload your custom background picture for your OpenRemote based smartphone or tablet app. The background picture in most cases probably needs to be darkened using photo editing software. Figure 17.10 shows an example background screen on an iPhone. If you desire you can even position the grid for the light switch above the position of the lamp. You then take two pictures - one with the lamp switched on and one with the lamp switched off. You assign the two pictures to the switch configuration and can now tab on the lamp in the picture. The lamp will go on, while the picture in the screen changes accordingly.



Figure 17.10 OpenRemote Based iPhone app with custom background screen

17.2 Configuring KNX Based Heating Mode Control

As the final example for OpenRemote to KNX interaction we will implement the classic time triggered operating mode change for a heating system. Typically heating systems are configured in operating states such as:

Command	Description
comfort	comfort temperature for rooms during the day
night	to keep the house well above the freezing point during longer periods of absence
antifreeze	to keep the house well above the freezing point during longer periods of absence
standby	below night temperature level for shorter periods of absence

Traditional heating control systems are often complex to handle. They execute a fixed schedule, set often years ago, when the system was installed. As a result the systems operate very inefficient, more or less dependent on how often the owner is ready to make an adjustment. At the same time heating (or cooling) in most regions of the world represents the by far largest portion of household energy consumption. Thus integrating heating management into the smart building control system will in most cases provide a significantly potential for energy saving.

In our example we will control Jung room controller units, which contain a thermostat, reporting the room temperature, and a management unit, which controls the heating actor, dim lights and control blends. It is one of the more complex units with a wealth of functionality. (Figure 17.11)

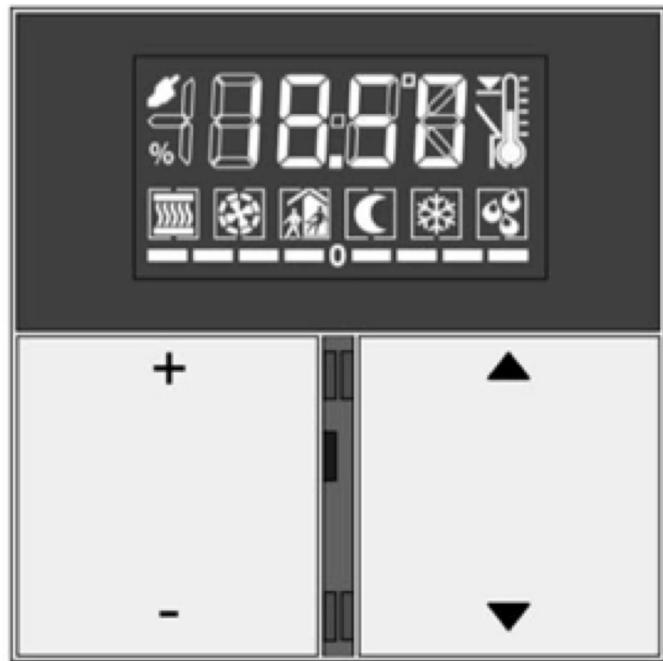


Figure 17.11 The Jung compact room control unit

Besides configuring the manual switches on the room control unit itself, the heating operating state can be configured by either by sending a one byte control value or by setting a combination of four one bit switches, which we will use for our heating control function. In the ETS configuration menu for the room controller we select the four bit control mode, which activates four 1 Bit KNX objects of the type DPT 1.001. We now can link each of the four objects to a KNX Group Address which allows us to access them through OpenRemote (Figure 17.12)

Group Addresses ▾

Add | Delete | Download | Info

- ▲ 0/1 Heating Mode Change
 - 0/1/0 Comfort mode bedroom1
 - 0/1/1 Standby mode bedroom1
 - 0/1/2 Night mode bedroom1
 - 0/1/3 Antifreeze bedroom1
 - 0/1/4 Comfort mode bathroom
 - 0/1/5 Standby mode bathroom
 - 0/1/6 Night mode bathroom
 - 0/1/7 Antifreeze mode bathroom
 - 0/1/8 Comfort mode bedroom2
 - 0/1/9 Standby mode bedroom2
 - 0/1/10 Night mode bedroom2
 - 0/1/11 Antifreeze mode bedroom2
 - 0/1/12 Comfort mode diningroom
 - 0/1/13 Standby mode diningroom

Figure 17.12 KNX Group Address linkage to the four switch objects heating operating mode change

In order to switch to a particular operating state, the four switch objects have to be set as outlined in the matrix in Table 17.2:

Antifreeze	Comfort	Standby	Night	
------------	---------	---------	-------	--

1	X	X	X	Antifreeze
0	1	X	X	Comfort
0	0	1	X	Standby
0	0	0	1	Night

Table 17.2 Control matrix for Jung compact room control unit

Thus for example, in order to switch the heating status to night, we need to set four switches (Antifreeze to 0, Comfort to 0, Standby to 0 and Night to 1). For the other states we just need to set three, two or one switches respectively. (X stands for don't care). We import the newly defined KNX Group Addresses into OpenRemote, assigning the command types "Switch" to each group address. OpenRemote imports each group address automatically into two commands, one extended by (ON) and one by (OFF) (Figure 17.13).

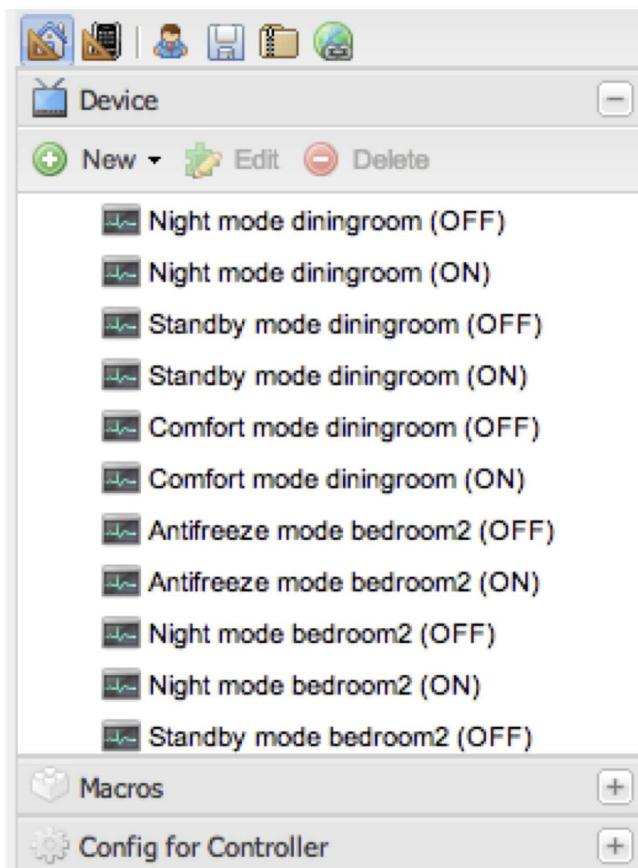


Figure 17.13 Heating state commands after import in OpenRemote

We will now first configure our OpenRemote control panel so we can change the operating mode for each room by pressing a single button. Once the manual control via OpenRemote is working, it will be easy to implement Drools rules, which manage the heating system according to our needs.

17.3 Smartphone Based Heating Control

Since we need to set up two four single commands in order to change the heating state for a room we will use the OpenRemote macro function for implementation. For each room state we will create an according macro. In the Building Modeler we expand the *Macro* menu and select *New*. We give the macro a name (e.g. bedroom2) and simply drag the commands from the left side of the macro window to the right side (Figure 17.14).

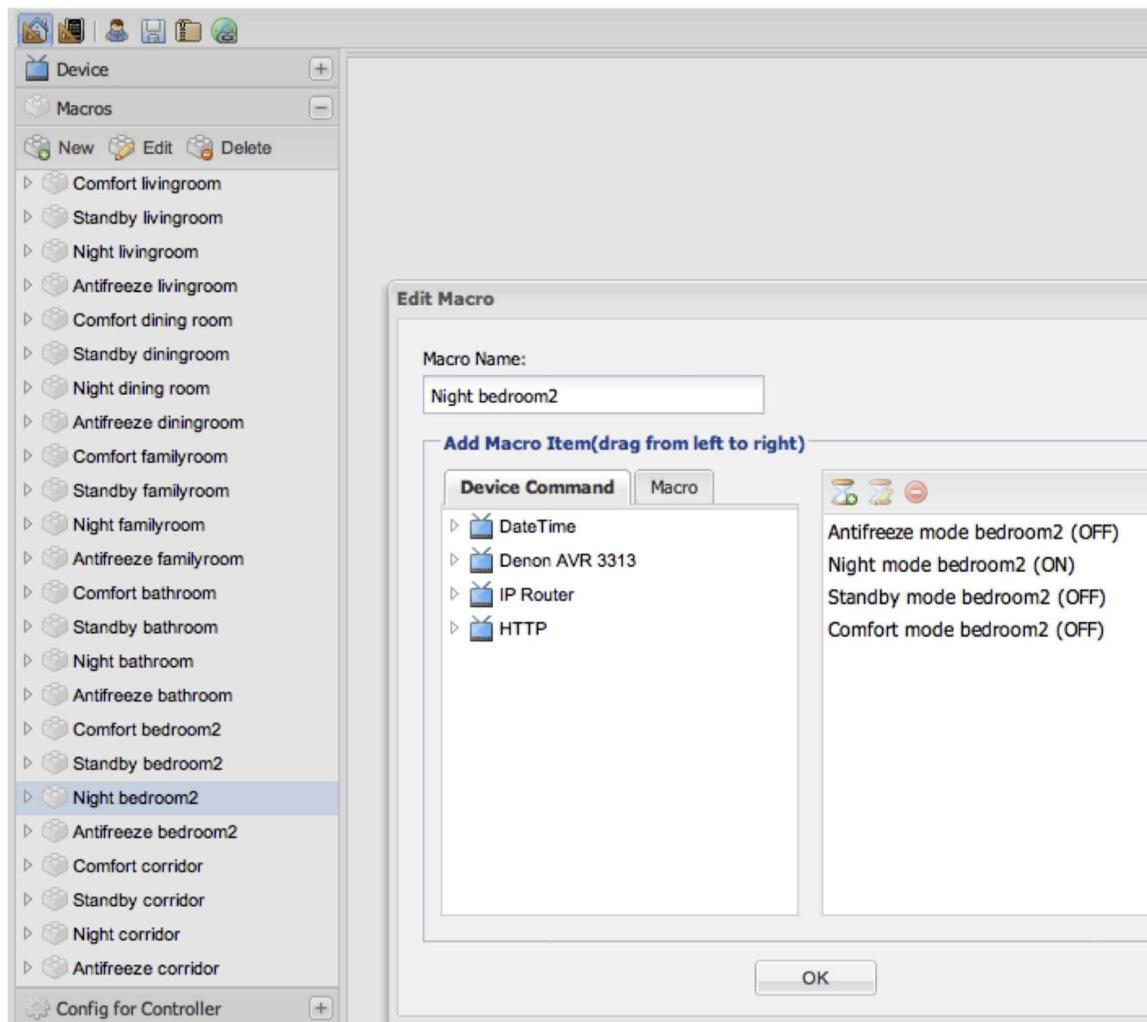


Figure 17.14 Macro for switching bedroom two room controller to night mode

We do this for all rooms and all operating states and can now assign the macros to push buttons on our control panel screen. In addition to the push buttons we also want to display the current and the target temperature on our smartphone app. Both - current and target temperature - are also KNX read objects, which our Jung room controller provides with a DPT of 9.001. We link KNX Group Addresses to each current and target temperature object in ETS and import them assigning the command type “temperature status” in the OpenRemote import dialogue. OpenRemote automatically creates the associated commands as well as sensors, so all which is left to do is configure the screen on our panel. (Figure 17.15)

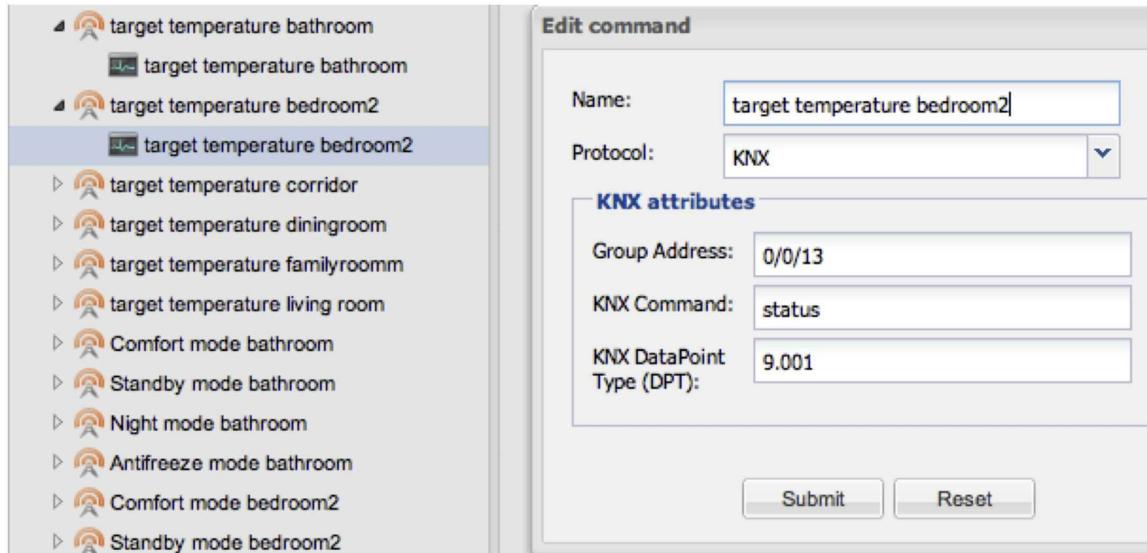


Figure 17.15 Sensors for displaying target and current room temperature

After setting up the layout using several grids we drag four **Button** widgets for each room into the grid and assign the according macros. For the display of the current and the target temperature we use the **Label** widget. We use two for each room and link them with the current and target temperature sensors respectively. We save the design, synchronize our controller and test the functionality on our smartphone or tablet (Figure 17.16). To round off our design we create two

symbol images (one *button image* and one *pressed button image*) for each heating state, choose a background picture and are done. (Figure 17.17).

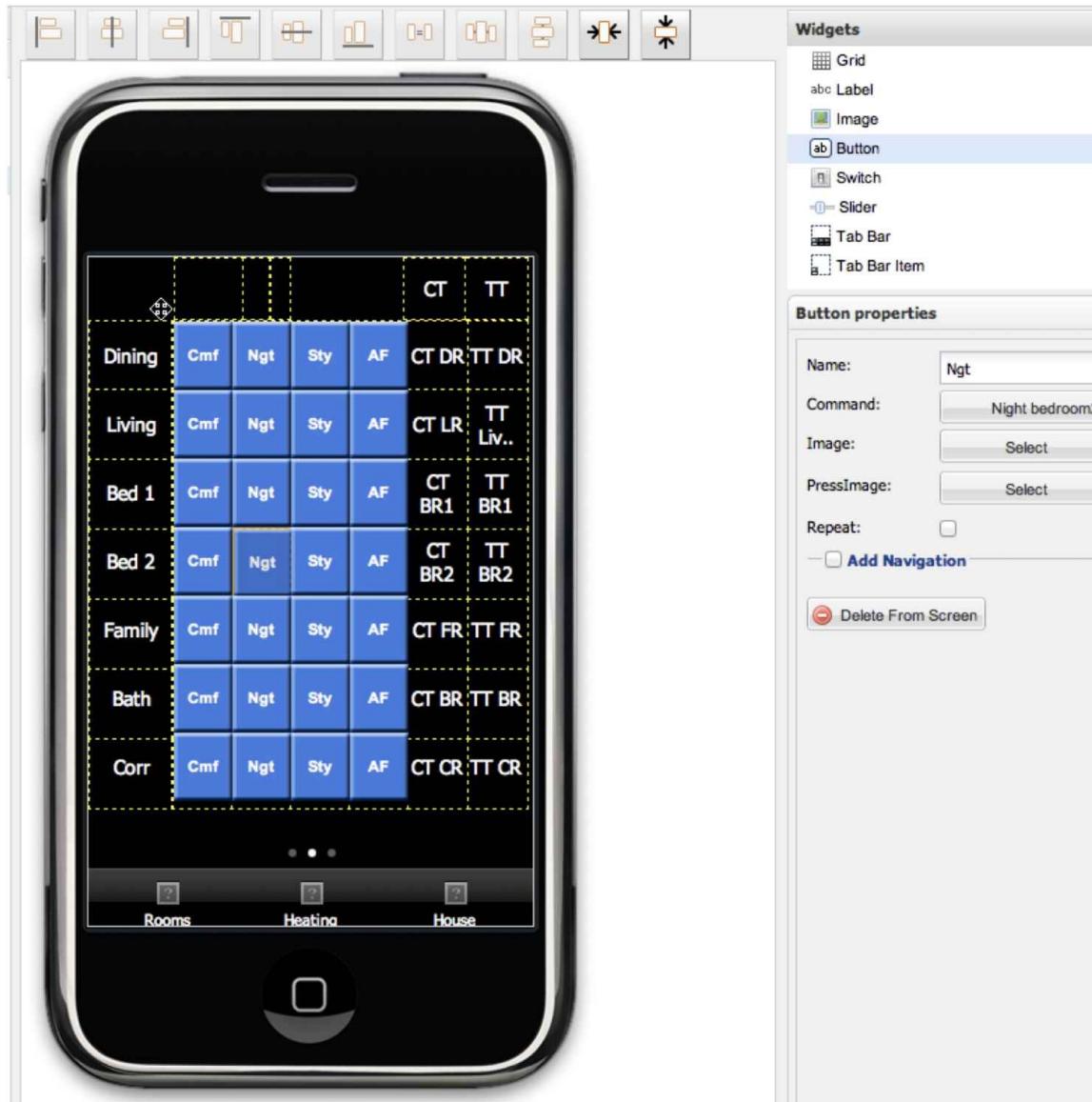


Figure 17.16 Configuration of the smartphone screen for our heating management

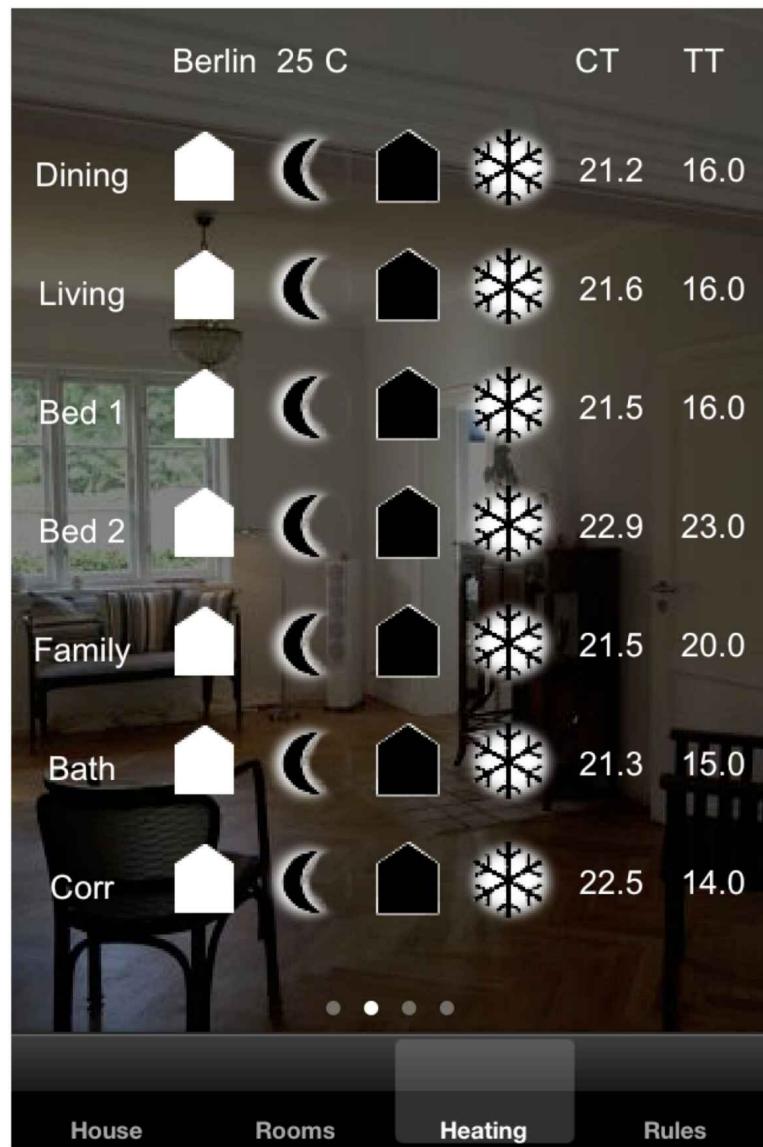


Figure 17.17 OpenRemote smartphone app for heating management

17.4 Drools Based Heating Automation

Finally we want to automate the change of heating operating modes based on rules. Initially we will simply switch to comfort mode at 5:30 a.m. As in our previous example with the time controlled start of iTunes we use the rules attribute `timer` and specify a schedule of Monday through Friday together in the macro. Here we need to list them individually in the rules statement, since it is not possible to use macro statements in rules. In addition we add a logging function to our rule. So after every execution of our command we want a brief status report along with date and time information: at 5:30 a.m.: timer (cron: 0 30 5 ? * MON-FRI)

For switching to comfort mode we need to execute two switch commands (antifreeze (OFF) and comfort (ON)). For the panel based control we put these commands `System.out.println("!!!bedroom1 switched to comfort!!!"); Date date1 = new Date();`

```
System.out.println(date1.toString()); With that the rules code for switching
bedroom1 and bedroom2 to comfort Monday through Friday at 5:30 a.m.
(including package definitions) would look like follows:
package
org.openremote.controller.protocol global
org.openremote.controller.statuscache.CommandFacade execute; global
org.openremote.controller.statuscache.SwitchFacade switches; global
org.openremote.controller.statuscache.LevelFacade levels; import
org.openremote.controller.protocol.*; import
org.openremote.controller.model.event.*; import java.sql.Timestamp;

import java.util.Date;

rule "heating management switch to comfort at 5:30 a.m."
    
```

timer (cron: 0 30 5 ? * MON-FRI)

when

eval(true)

then

```
execute.command("Antifreeze bedroom1 (OFF)"); execute.command("Comfort mode bedroom1 (ON)"); System.out.println("!!!bedroom1 switched to comfort!!!"); Date date1 = new Date();
```

```
System.out.println(date1.toString()); execute.command("Antifreeze bedroom2 (OFF)"); execute.command("Comfort mode bedroom2 (ON)"); System.out.println("!!!bedroom2 switched to comfort!!!"); Date date2 = new Date();
```

```
System.out.println(date2.toString()); end
```

In the controller terminal window you will be able to monitor the activity as reported through our logging function.

18 Remote Smarthome Control

A key element for a smart home is the capability to access all control functions from remote via the Internet. This allows the realization of various important use cases such as

- control and monitoring of the vacant home (temperature, energy, gas, water, smoke, wind)
- feeding and watching pets
- watering plants indoors and outdoors
- checking on elderly and handicapped people to ensure they are safe and well.

From a technical perspective, the above functions require the ability to access the home Wi-Fi network from a smartphone, tablet or notebook via the Internet. In order to do this, we will need to do two things:

- configure our Internet/DSL router to run a Dynamic DNS service (DDNS)
- set up a Virtual Private Network (VPN) connection between the mobile device we plan to use for accessing our smart home Wi-Fi network and our Internet/DSL router at home.

18.1 Configuring a Dynamic DNS Service The Dynamic DNS service resolves the problem, that the IP address for your Internet/DSL router is dynamically assigned by your Internet service provider (ISP) which causes it to change every 24 hours or every time you reboot your router. (This assumes that you do not have Internet access via a

permanently assigned, fixed IP address, which is the case for practically all residential homes.) Your router however can only be reached from the outside, if its current IP address is known. This is the reason for the existence of Dynamic DNS services. Routers, which need to be accessible from the Internet periodically send their current IP address to a DynDNS server, where it is associated with a dedicated domain name. When this domain name is called, the DynDNS server responds with the current IP address of the router. Using this IP address now a connection to the router can be set up. There are a number of free and pay for DDNS service companies. A good listing can be found under <http://www.gnutomorrow.com/best-free-dynamic-dns-services-in-2013/>

So all you need to do is to select a DDNS service, and register. At registration you select a domain name, a user name and a password. Then you are ready to configure your router. You log into your Internet/DSL router and look for a menu item called remote access / Dynamic DNS or something similar. You activate this function and enter your DDNS provider name as well as a domain name, a user name, and a password of your account.

However, most major router manufacturers today provide free DDNS services for their customers. The reason is, that an increasing number of users wants to be able to access their home network from the Internet to remote control lighting, heating systems or other smart home functions. So before signing up to one of the DDNS-services check on the homepage of your router manufacturer if a free DDNS service is provided. Here the links to the free DDNS services of some popular router manufacturers: [D-Link](#), [Netgear](#), [Asus](#), [AVM Fritz!Box](#).

18.2 Configuring a VPN

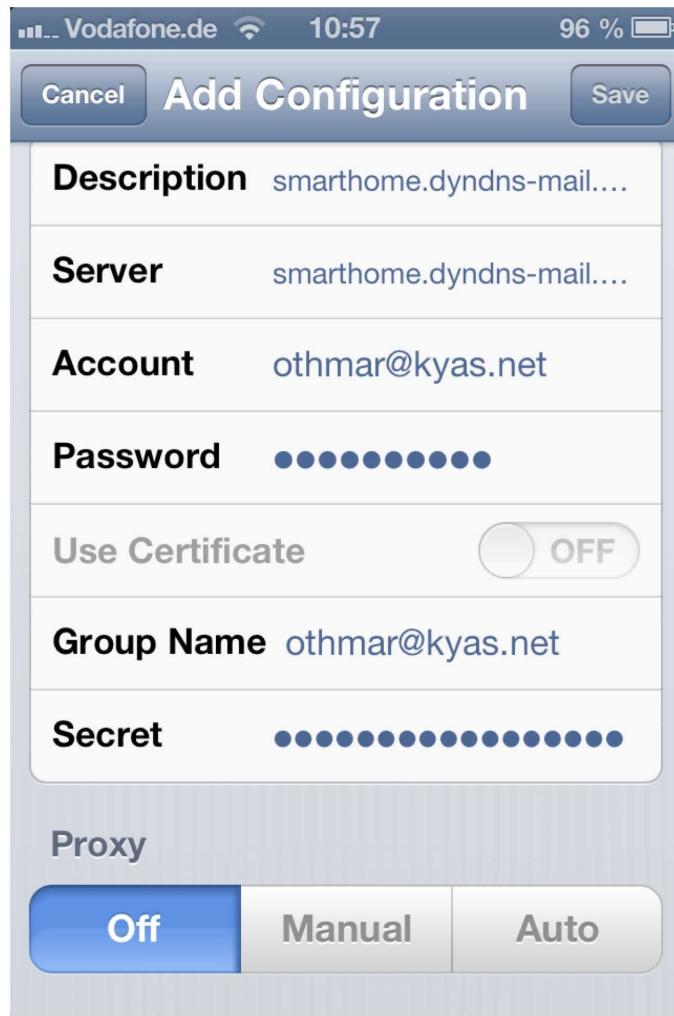
The second step is to configure a VPN connection between the mobile device that you plan to use for remote smart home access and your home network. A VPN is basically a secure (encrypted) point-to-point connection between two

VPN is basically a secure (encrypted) point-to-point connection between two networks or computers. On each end point of the connection, a VPN software agent needs to be installed. Both VPN agents need to be configured, among other things, with a so called "shared secret", which is the key for the encrypted network connection. It allows the two VPN agents to establish a connection (sometimes referred to as a VPN tunnel) with each other. In our case (as with most residential homes), one end of the VPN connection will be the DSL/Internet router of the smart home, the other end a smartphone, tablet or notebook.

It is best to start with your DSL/Internet router and check the manual or the support web site for how to set up a VPN connection. Most vendors provide a small utility that automatically creates the necessary configuration files for the router as well as for the second VPN end point. This is necessary, since you cannot manually generate the encryption keys for the shared secret. For the configuration utility, you will need to have the following information at hand:

- Dynamic DNS (DDNS) domain name
- DDNS user name
- DDNS password
- IP address of your DSL/Internet router (the one inside your Wi-Fi network)
- Subnet address mask of your DSL/Internet router

The utility will then typically generate two configuration files. One is for the DSL/Internet router, which you will use when activating its VPN capability, and a second one is for your smartphone, PC or Mac. As an example, on an iPhone you go to *Settings – General – VPN – Add VPN Configuration*. You then select the communication protocol (typically IPSec) and enter the DDNS credentials along with the VPN shared secret you have received from the VPN utility.



(Figure 18.1)

Figure 18.1 VPN Configuration on an iPhone When you now activate the VPN connection from your smartphone (while it is connected to the Internet, e.g. via 2G or 3G), a secure VPN connection will be established to your Internet/DSL Router. Now from the perspective of the apps on your smartphone or tablet you are connected to your home Wi-Fi network, as if you were at home. You can now start your OpenRemote app and have full access to your smart home control functions.



Figure 18.2 Activating VPN on an iPhone

19 Cold Start: Launch Automation

If the case of the smart home controller reboots (due to a power outage or due to maintenance downtime), we need to ensure, we have a running system in a defined operating state in place once the system is back up. For this purpose we need to have a script in place that automatically restarts the OpenRemote controller after the reboot of the system and documents the restart in a log file. For this purpose, under macOS /Linux we will use crontab under Windows Task Scheduler.

19.1 Windows 10 Task Scheduler

Under Windows scheduling a task to be executed at startup is quite straight forward. We open Task Scheduler by entering Scheduler in the search field next to the Windows 10 start button and then selecting Task Scheduler from the results. Next we click on the *Action* menu and select *Create Basic Task*. We give the task a name, select *Next*, and then *When the computer starts*. Now we select *Start a program – Next*, browse to `openremote.exe`, enter `run` in the *Add arguments* field, and select *Next* again, to finish the creation of the task. (Figure 19.1)

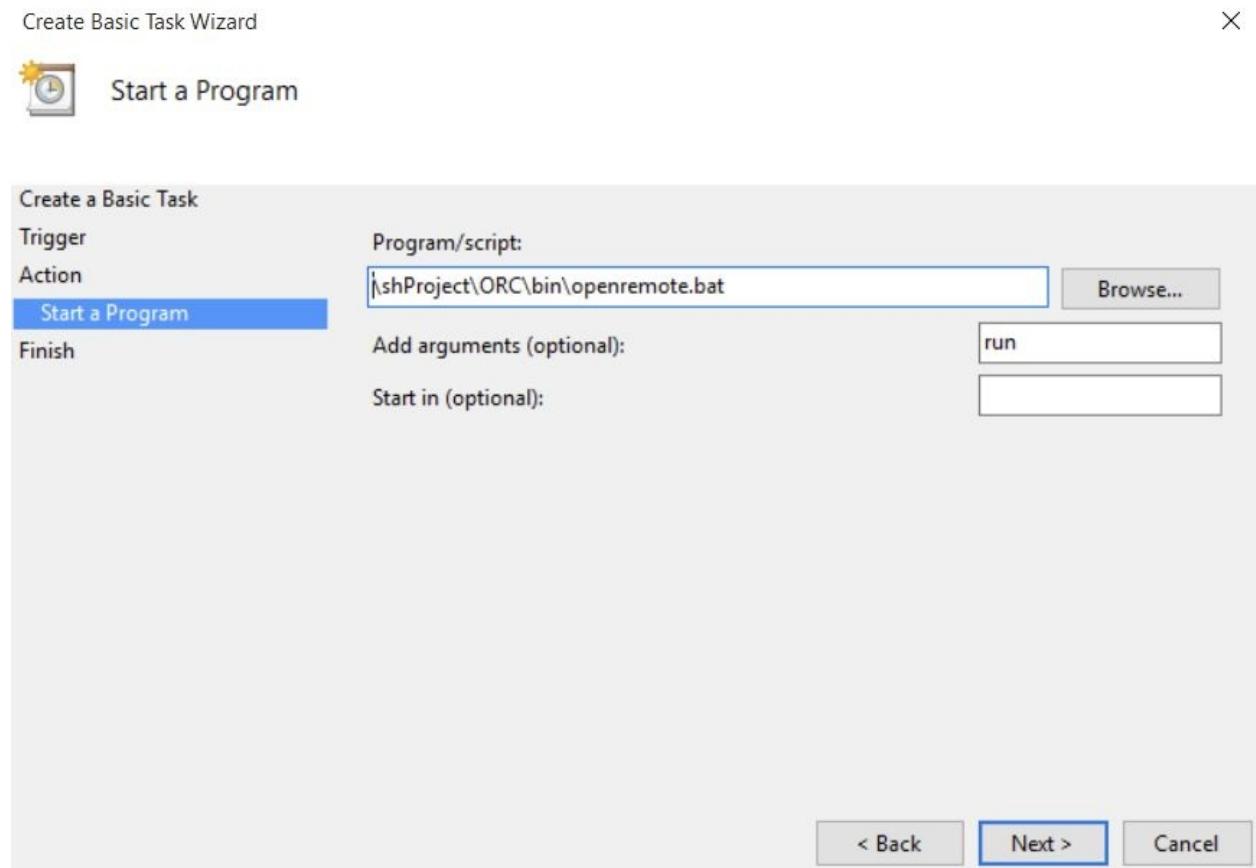


Figure 19.1 Task Scheduler configuration under Windows 10

19.1.1 Sending a Reboot Notification Email

In addition to automatically starting OpenRemote we want our smart home

controller to send out a reboot notification email. While Windows Task Manager can be configured to send an email at system startup, we want to be more flexible. For example we might want our OpenRemote application to trigger sending an email under certain circumstances as well. Therefore we will use the command line email application mailsend, which can be downloaded for free from <https://github.com/muquit/mailsend/releases/>. The mailsend documentation can be found under <https://code.google.com/p/mailsend/wiki/mailsendFAQ>.

In order to send an email with mailsend we open Windows Terminal, change to the mailsend directory and type mailsend.exe followed by the specific options for our mail server. Most mail servers today require authentication via user name and password and encryption using STARTTLS or SSL. If you are not sure about the capabilities of your SMTP mail server type the below command (the example for Gmail): mailsend.exe -info -port 587 -smtp smtp.gmail.com Now you should be able to construct the complete command for sending an email. Below the command for sending an email using the Gmail SMTP server with authentication and STARTTLS encryption: mailsend.exe -to ok@keyconceptpress.com -from info@smarthomeserver.com -starttls -port 587 -auth -smtp smtp.gmail.de -sub "Open Remote Server Notification" +cc +bc -v -user info@smarthomeserver.com -pass "yourpassword" -M "Open Remote Server restarted"

The mailsend options we use in the above example are listed below. A complete listing of all options can be found under <https://github.com/muquit/mailsend>.

Option	Description
-smtp	Hostname/IP address of the SMTP server
-to	email address/es of the recipient
-from	email address of the sender
-ssl	SMTP over SSL
-starttls	Check for STARTTLS and if supported switch to it
-auth	authenticating trying CRAM-MD5, PLAIN, LOGIN, GSSAPI, NTLM, DIGEST-MD5, SCRAM-SHA-1, SCRAM-SHA-256, SCRAM-SHA-512, X-SSPI, X-GSSAPI, X-NTLM, X-DIGEST-MD5, X-SCRAM-SHA-1, X-SCRAM-SHA-256, X-SCRAM-SHA-512, X-X-SSPI, X-X-GSSAPI, X-X-NTLM, X-X-DIGEST-MD5, X-X-SCRAM-SHA-1, X-X-SCRAM-SHA-256, X-X-SCRAM-SHA-512, X-X-X-SSPI, X-X-X-GSSAPI, X-X-X-NTLM, X-X-X-DIGEST-MD5, X-X-X-SCRAM-SHA-1, X-X-X-SCRAM-SHA-256, X-X-X-SCRAM-SHA-512

	authenticating trying: CRAM
-user	password for ESMTP authen
-pass	Content line
-M	

Table 19.1 Essential options for the command line email tool mailsend Once we have validated our email command in Windows Terminal, we open the Windows PowerShell editor, paste in the command (along with the complete path to our mailsend directory) and store it under something like ormail.ps1 (Figure 19.2). Again we test our work by executing the script from within the PowerShell editor.

As the last step we now need to configure Task Scheduler to run ormail.ps1 at startup. We open Task Scheduler, click on the *Action* menu and select *Create Basic Task*. We give the task a name like OR Reboot Notification, select *Next*, and then *When the computer starts*. Now we select *Start a program* and enter powershell.exe in the *program/script* field and C:\Users\smarthome\mailsend\ormail.ps1 in the *Add arguments* field (Figure 19.3). Of course we can now also add the openremote.bat run command to our Powershell script ormail.ps1 and only schedule a single task for startup. Finally, with a few reboots of our system we validate the functionality of our auto reboot and notification email capabilities.

```
Windows PowerShell ISE
File Edit View Debug Help
ormail.ps1
1 C:\Users\othmar\mailsend\mailsend1.16.exe -to ok@keyconceptpress.com -from info@postmaster27.de -starttls -port 58

[C] Open Remote Server restarted

--kXbnm0becBnp47UT--

[C] .

[S] 250 Requested mail action okay, completed: id=0M6UrD-1XSqVr0m0R-00yNDM
[C] QUIT
[S] 221 kundenserver.de Service closing transmission channel
```

Figure 19.2 PowerShell script for sending the reboot notification email

Create Basic Task Wizard



Create a Basic Task

Trigger

Action

Start a Program

Finish

Program/script:

powershell.exe

Add arguments (optional):

C:\Users\smar

Start in (optional):

< Back

Next >

Figure 19.3 Notification email in Task Scheduler (Windows 10)

19.2 macOS launchd

With macOS Tiger Apple introduced launchd as the new service-management framework for its operating system. It replaces older services such as init, crond or watchdogd. Its purpose is to start, stop and manage daemons, applications, processes and scripts. While launchd is open source and also available for Linux, most Linux distributions still use systemd or Upstart for service management. This section focuses on the description of preparing an macOS system for auto starting OpenRemote when rebooting using launchd.

19.2.1 Setting up a Standard User for OpenRemote If you have not done yet, I strongly recommend to set up a dedicated user for your smarthome controller. Running OpenRemote in operation using an administrator account is a significant security risk, and should never be done. If anything goes wrong, may it be an erroneous script, someone else making unauthorized changes, a virus or an attack from outside, under administrator rights the consequences will be much more severe, than under a standard user account. Administrators can create, manage, and delete other users, install and remove software, and change your Mac's settings. For these reasons, server software, which can be accessed from other devices (as in our case using smartphones or tablets) or even from the Internet (in case you set up a VPN connection for OpenRemote) should never be installed using a user account with administrator privileges. Go to *System Preferences – Users and Groups* and click on the plus sign on the left hand side to add a new standard user account (Figures 19.4, 19.5).

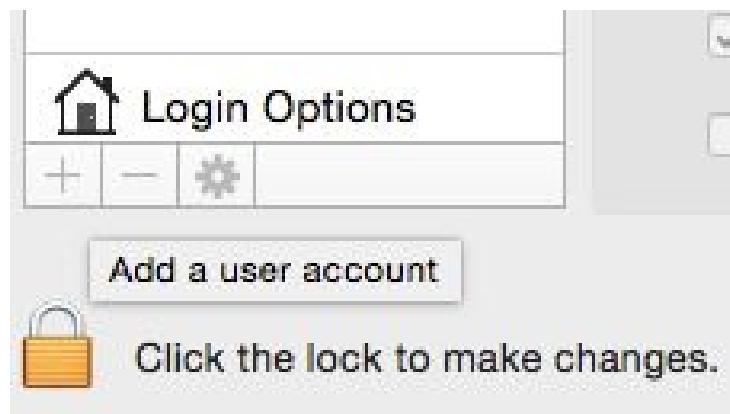


Figure 19.4 Adding a standard user under macOS (step 1)

New Account: Standard

Full Name: smarthome

Account Name: smarthome
This will be used as the name for your home folder.

Password:

Use iCloud password
 Use separate password

.....

.....|

Hint (Recommended)

? Cancel Create User

A detailed screenshot of the "User & Groups" pane in System Preferences. It shows a form for creating a new user account. The "New Account:" dropdown is set to "Standard". The "Full Name:" field contains "smarthome". The "Account Name:" field also contains "smarthome", with a note below stating "This will be used as the name for your home folder". Under "Password:", there are two radio buttons: "Use iCloud password" (unchecked) and "Use separate password" (checked). Below the radio buttons are two password input fields, both showing masked text. A key icon is positioned next to each field. A "Hint (Recommended)" text input field is present. At the bottom are "Cancel" and "Create User" buttons, with the latter being blue and highlighted.

Figure 19.5 Adding a standard user under macOS (step 2) Then select [Login Options](#) on the left side of the menu and configure [Automatic login](#) for your new user, in our case the user account smarthome. When you log in to your new account for the first time, the system will ask you to configure your AppleID, which you must decline. Then set up a dedicated email account for the smarthome controller, if you want to send notification emails as described below. Now, every time the system reboots, the new standard user smarthome will automatically login (Figure 19.5). This is what we need, since among other things our start up sequence will use AppleScript to send a notification email. The usage of AppleScript and the email application requires our smarthome user to be logged in. From a security perspective, this is acceptable, since this user cannot make changes to the system and only has access to OpenRemote. If you want even more security, change the OpenRemote startup script openremote.sh to read only, with write access restricted to the administrator. This prevents the classic attack, which modifies the start up script and then restarts the system in the hope, that during the start up sequence the script would be executed using administrator rights. However, since we will use our standard user to restart OpenRemote, as you will see below, this attack would not be possible in our system setup even without changing access rights.

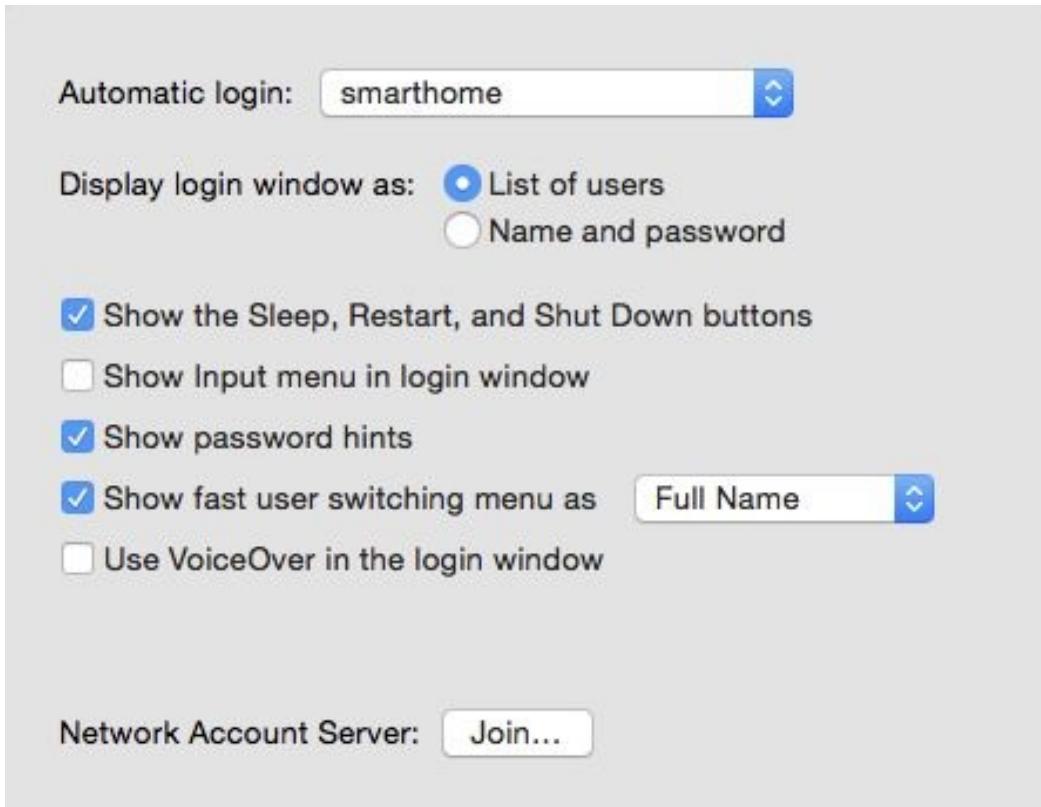


Figure 19.6 Adding a standard user under macOS (step 3) 19.2.2 Daemons and Agents

Once you are done with setting up OpenRemote under a dedicated standard account we can start building the autostart functionality. There are two different types of processes the launchd service is dealing with: agent processes and

daemon processes

An agent is run on behalf of the logged in user. This means it only runs, when the user is logged in. Daemons run on behalf of the root user or any user specified with the User option, independent of the user being logged in or not. The system differentiates three agent and two daemon types. However, you should never have to edit System Agents or System Daemons:

Type	Run on behalf of
User Agents	Currently logged in user
Global Agents	Root or the specified user
Global Daemons	Currently logged in user
System Agent	Root or the specified user

Table 19.2 Agent and daemon types

As our autostart sequence requires a logged in user as explained above, we will use a User Agent for our startup script.

19.2.3 The Startup Shell Script

Before we go about configuring our User Agent, we need to create the shell script, which the agent shall start. If you just want to start OpenRemote the according shell script is simple: `#!/bin/sh`

```
cd Userssmarthome/shProject/ORC/bin ./openremote.sh run
```

We save it under a name such as `orstart.sh` in our `shProject/shconfig` directory, set the execution permission with `chmod +x orstart.sh`

and move on to configuring our user agent. Even though the user agent will not open a Terminal window and thus will start OpenRemote without a GUI, I recommend to use the the startup option `openremote.sh run` rather than the background option `openremote start`, since the first option allows us to track the server output from the default output variables standard output and standard error, as we will see below.

19.2.4 Hint: Clean directory management

I recommend to place all custom scripts and files into the directory shProject/shconfig, rather than in the OpenRemote directory (which in our project we called ORC260) or one of its subdirectories. Otherwise you will have to manually move your custom scripts and files from your old to your new OpenRemote installation in case of a OpenRemote software update.

19.2.5 The Utility LaunchControl

Agents and Daemons consist of XML files with well defined syntax, which are not easy to write for beginners. In order to get things done quickly I strongly recommend to use the *launchd* utility *LaunchControl* from soma-zone. For testing you can download the application with full functionality for free from <http://www.soma-zone.com/LaunchControl/>. The purchase price of less than 10€/US\$ for the application is very fair considering the functionality you get. After downloading and opening LaunchControl you select *User Agent* in the GUI and *File – New*. In the field *Label* you enter the name of your task. Apple recommends to use reverse URL notation, since label names must be unique. So you could use something like *com.coldstart.openremote*. In the left pane you rename the default file name *local.job* to the name you want it to give. I recommend to use the label name as file name as well. The label is the name of our agent, under which it is recognized by *launchd*, the file name is simply the name of the agent definition file. There is really no reason why you should call it differently.

Next you drag the symbol *WorkingDirectory* from the utility pane on the right hand side of the LaunchControl GUI into the main window. Now you should have a total of three configuration blocks in the main window of your agent configuration: Program to run

Working Directory

Run at load

(*Program to run* and *Run at load* are automatically loaded when opening a new configuration file). In *Program to run* we now enter the name of our shell script (*orstart.sh*), in Working Directory the path to it's directory:

UserssshProject/shconfig If you enter a file which does not exist or which does not have execution rights, the entry will remain red. The same is true for the working directory path. Both, file name and directory path, need to appear in green for the agent to be able to execute.

Finally we drag the *StandardErrorPath* symbol into our program window. Two command boxes appear: *Standard Output* and *Standard Error*. They allow us to monitor the standard and error outputs which our OpenRemote startup script generates. Edit the path definitions and the file names according to your needs. We save our daemon definition with *File – Save*.

When we now select *Load* from the upper right corner of the GUI we run our agent for the first time (Figure 19.7).

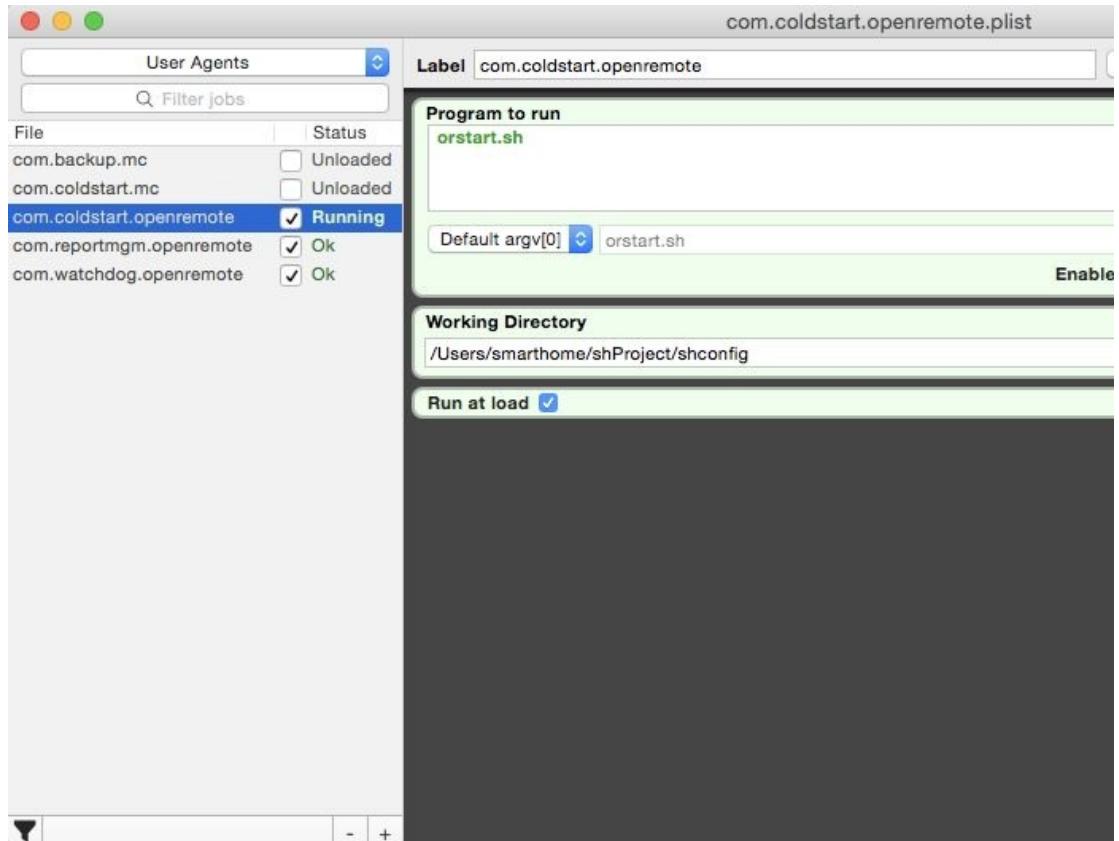


Figure 19.7 OpenRemote startup User Agent using LaunchControl In the activity pane on the left hand side we can see if our agent has successfully loaded. To validate if our startup script has actually started OpenRemote we open a Terminal window and use the ps (process status) command, which displays every active process along with its job number. In order to filter out the process of interest, in our case the OpenRemote process, we pipe the output to a filter using |grep openremote: ps aux|grep openremote

```
1290 0,0 0,0 2432772 640 s000 S+ 4:31pm 0:00.00 grep openremote
In the above case however, OpenRemote is NOT running. The only process that matches openremote is the grep process itself (the process doing the searching).
Below an example of the process state command with OpenRemote running: ps aux|grep openremote
```

```
smarthome    4168 7.6 4.1 8373520 681644 ?? S Tue12PM 318:44.94
LibraryJava/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/bin/java -
Dcatalina....
```

```
smarthome    4187 0.0 9.8 8547572 1636312 ?? S Tue12PM 218:59.03 /
...
```

```
smarthome    8320 0.0 0.0 2424580 384 s001 R+ 8:26AM 0:00.00
grep openremote In case the startup agent is not working as desired, click on the
TRC (Trace) symbol in the Standard Output and the Standard Error window of
LaunchControl. This will get you the standard and error output trace of your
startup script with the necessary information for debugging (Figure 14.8).
```

To stop a process you can use the kill command. With kill you can either specify the process you want to stop by PID (Process ID) which is displayed when using the process listing command ps, or by name. The command kill 571

will stop the process with the PID 571. Alternatively you can monitor and stop processes using the macOS Activity Monitor *Applications – Utilities – Activity Monitor* App.

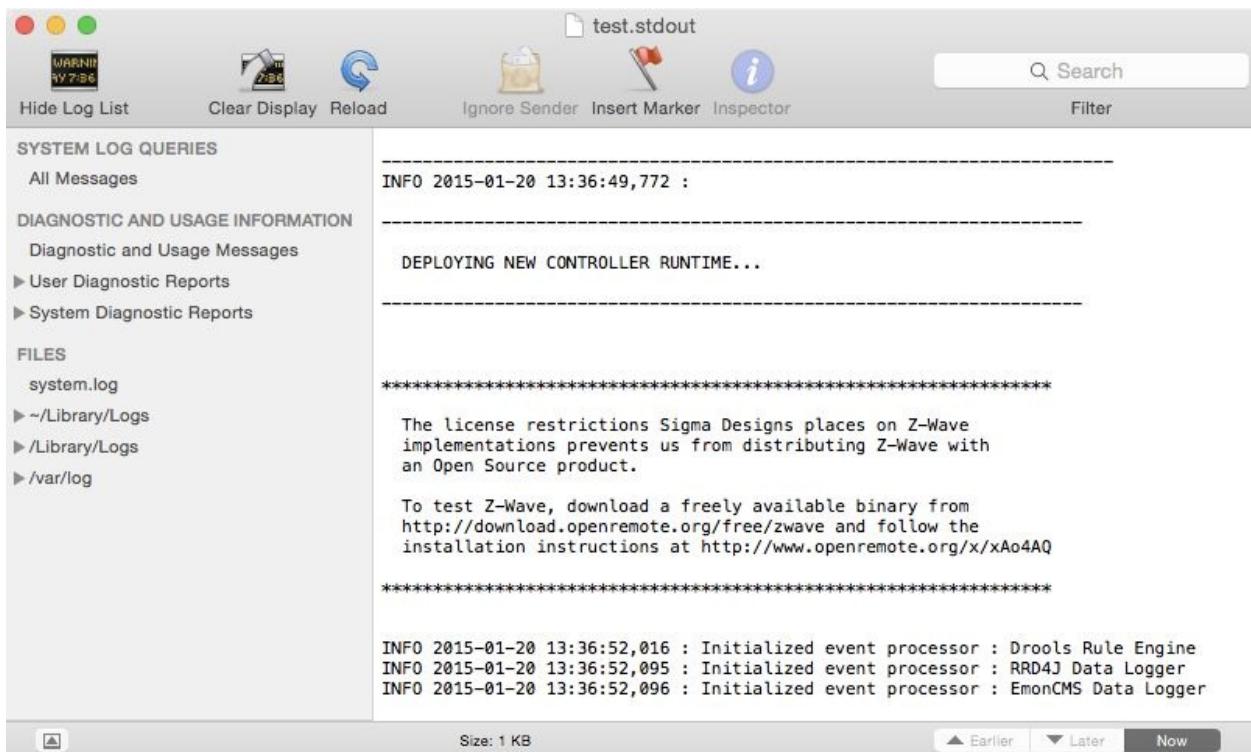


Figure 19.8 Analyzing the startup sequence using LaunchControl 19.2.6 Reboot Notification via E-Mail

Finally we want to send a notification email every time the server is rebooting. Since we want to avoid setting up a SMTP server on our system, which is not a trivial task, we will use the Apple Mail email account of our standard user. Alternatively we could also use the command line mailer application mailsend as described in the Windows section above. Mailsend is also available for macOS. For our project however we will simply control our smart home email account using AppleScript commands, which we will then - in a second step - call from a shell script. As explained above, in order for this approach to work you will need to set the user account of your OpenRemote server to *Automatic Login* (open System Preferences and select *Users&Groups – Login Options*) and you will need to setup an Apple Mail account for this user.

Below the simple AppleScript, which sends out an email with the current date in

its message body. At first the AppleScript variables recipientName, recipientAddress, theSubject, theTime, theText and theContent are set. The command current date generates the current date, which we want to be contained in our notification message. The command tell application “Mail” activates the Apple mail application, and Create the message, Set a recipient, and send create and send the email message. Below the complete AppleScript code: set recipientName to “Smarthome”

```
set recipientAddress to “info@youremail.com”
```

```
set theSubject to „Server restart”
```

```
set theTime to current date
```

```
set theText to “Server restart at “
```

```
set theContent to theText & theTime
```

```
tell application “Mail”
```

```
##Create the message
```

```
set theMessage to make new outgoing message with properties
```

```
{subject:theSubject, content:theContent, visible:true}
```

```
##Set a recipient
```

```
tell theMessage
```

```
make new to recipient with properties {name:recipientName,  
address:recipientAddress}
```

```
##Send the Message
```

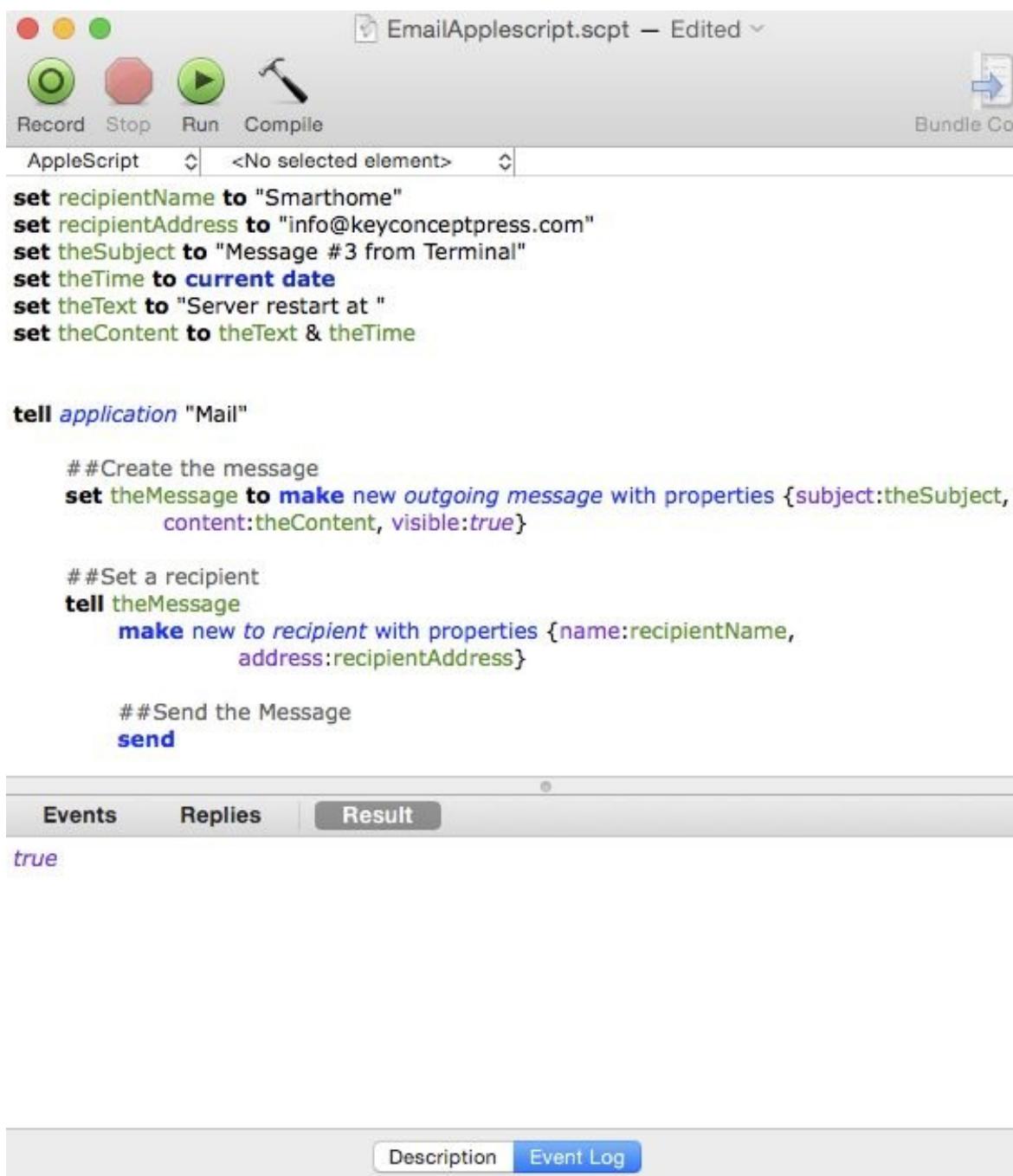
```
send
```

```
end tell
```

```
end tell
```

Now open the AppleScript editor, paste in the above script (inserting your email address for recipientAddress) and hit *Run* to test if the script is working. Almost immediately an email should be sent to the recipient address in the script (Figure

19.9).



The screenshot shows the AppleScript Editor window titled "EmailApplescript.scpt — Edited". The script content is as follows:

```
set recipientName to "Smarthome"
set recipientAddress to "info@keyconceptpress.com"
set theSubject to "Message #3 from Terminal"
set theTime to current date
set theText to "Server restart at "
set theContent to theText & theTime

tell application "Mail"
    # Create the message
    set theMessage to make new outgoing message with properties {subject:theSubject,
        content:theContent, visible:true}

    # Set a recipient
    tell theMessage
        make new to recipient with properties {name:recipientName,
            address:recipientAddress}

    # Send the Message
    send

```

The "Result" tab is selected, showing the output "true". Below the editor is a toolbar with "Description" and "Event Log" buttons.

Figure 19.9 AppleScript for sending an email Now we want to run the above AppleScript within a shell script. For this we just need to insert the osascript

command in the first line and tell the script interpreter to handle all text following this command as AppleScript until EOF is reached: #!/bin/sh

```
exec osascript << EOF
```

```
set recipientName to "Smarthome"
```

```
set recipientAddress to "info@keyconceptpress.com"
```

```
set theSubject to "OpenRemote Server restarted"
```

```
set theTime to current date
```

```
set theText to "OpenRemote Server Beachroad 4, Upville restarted at "
```

```
set theContent to theText & theTime
```

```
tell application "Mail"
```

```
set theMessage to make new outgoing message with properties  
{subject:theSubject, content:theContent, visible:true}
```

```
tell theMessage
```

```
make new to recipient with properties {name:recipientName,  
" . . . . . "
```

```
address:recipientAddress }
```

```
send
```

```
end tell
```

```
end tell
```

```
EOF
```

We save the above code as tmailor.sh (terminalmail openremote). All which is left to do now is to add tmailor.sh to our autostart script orstart.sh, which now reads as follows.

```
#!/bin/sh
```

```
sleep 60
```

```
cd /Users/smarthome/shProject/shconfig ./tmailOR.sh
```

```
cd /Users/smarthome/shProject/ORC/bin
```

```
./openremote.sh run
```

The command sleep 60 introduces a 60 second delay to the start of the script,
~~since we want to make sure the auto login procedure for our smarthome user is~~

Since we want to make sure the auto login procedure for our smartphone user is finished, by the time we start OpenRemote and send the notification email.

19.2.7 Curly or straight, this is the question!

Finally an important hint for editing AppleScript and shell scripts in general, where quotation marks have an important function. Double quotes (in shell scripts, AppleScripts and UNIX in general) are used for quoting phrases as you have seen in some of the above scripts. However, these quotes need to be the straight quotation marks, rather than the curly quotation marks, which are used by word processor software. Now sometimes, when you edit a shell script or an AppleScript usingTextEdit or another word processor, it can happen, that the straight quotes are converted to curly quotes resulting in runtime errors, which at first sight are hard to analyze. So always take a close look at the quotation marks: „Curly or straight, this is the question!“ (Figure 19.10).

```
“  
„curly double quotes  
"  
"straight double quotes"  
`  
`backtick`
```

Figure 19.10 Curly versus straight double quotes and the single back tick

20 Troubleshooting and Testing

In many projects with high dependency on reliable software, the aspect of software test is underestimated. In particular in control engineering, which is what smart home control really is, a systematic and planned test phase is important in order to put a reliable, redundant system in place. In a smart home environment scripts and system configurations interact with people and assets, which can get hurt or damaged if something goes wrong. Thus a wait and see approach is not an option. While a detailed tutorial on test strategies and tactics is beyond the scope of this book, I cannot emphasize enough the importance of this topic. First considerations for test planning already need to be made before the start of the project. Test concepts and code testability have to be part of high level software design. Then for each software module, in parallel to writing the actual code, relevant test cases need to be defined. Systematic, planned test execution in the end ensures a software quality level which meets the requirements of a 24/7 smart home operation.

Once the software is tested and ready for deployment, the so called dress rehearsal tests start. During this phase, the system is tested by end users, who are given a heads up to watch out for funny or faulty behavior. Once the software passes this phase, the first stable release is rolled out. As part of the now beginning release management, the latest tested and stable version is well documented and saved as a backup. A roll back to this version must always be possible at any time.

For troubleshooting problems or unexpected behavior of the smart home control system, besides monitoring processes and log files on the controller, it is always a good idea to be able to monitor the smart home network itself as well. A powerful troubleshooting tool for this purpose is the open source protocol analysis tool Wireshark, which is available from <http://www.wireshark.org> (Figure 20.1).

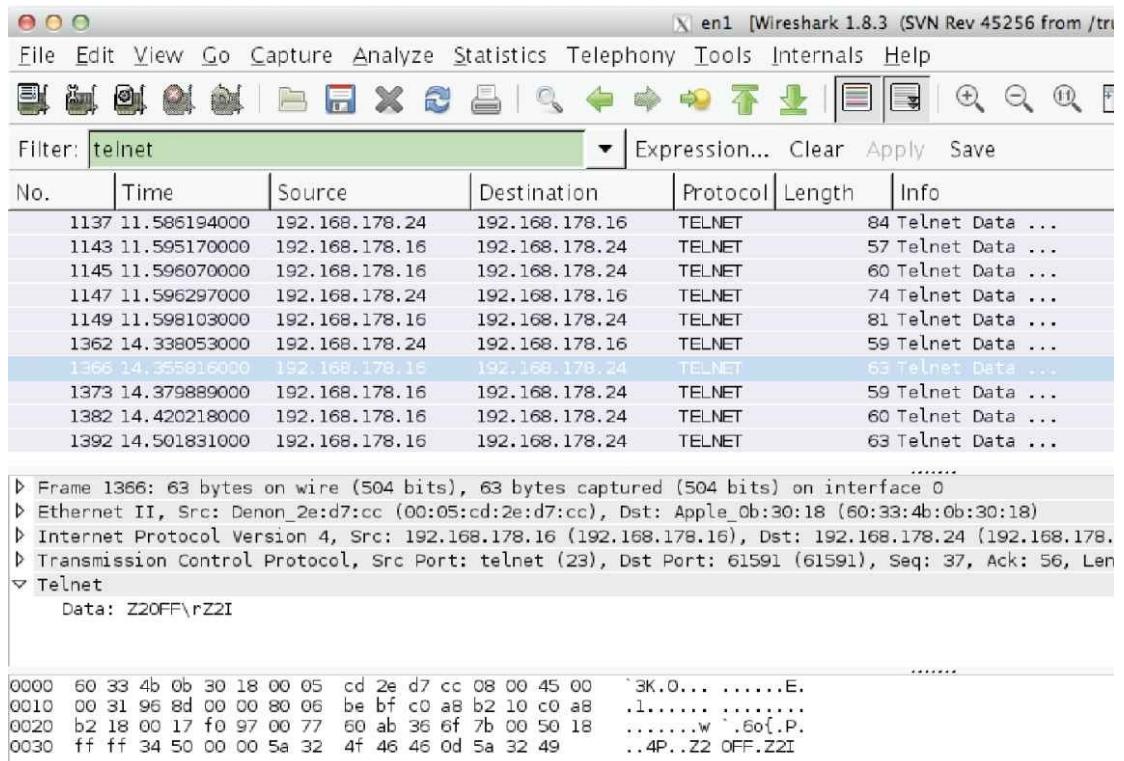


Figure 20.1 Wireshark protocol trace of the Denon 3313 Telnet response to a “Z2?” command

20.1 Preventive Maintenance

Contrary to traditional troubleshooting, the approach of preventive maintenance is to try to identify and repair a problem before users are impacted. Typically such systems consist of a problem detection component and (ideally) a self-healing component.

A common method for problem detection in real time, high availability control systems are watchdog scripts. These are scripts that continuously monitor the key software processes. If a process stops working, they set an alarm. Some watchdog implementations browse the listing of active processes in order to verify whether a process is still alive. There are, however, cases in which a process is still listed as active, while in reality it already has stopped working. Commonly these are referred to as Zombie processes. Thus the proper way to reliably monitor the health of a process is to insert a small routine, which periodically (e.g. every minute) writes a value (e.g. 100) to a file or variable. A separate watchdog script then decrements this value (e.g. every minute) by 10 and compares the resulting value to zero. If the process stalls for several minutes, the variable is not set back to its initial value and eventually reaches zero, so that the watchdog sets an alarm.

In addition to monitoring software execution, watchdogs can also be used to verify whether an action actually reaches the real world, as intended. As an example, a watchdog for a reminder system that announces an important message in a smart home could validate the output of the message by recording the announcement via microphone, and comparing it with the intended output.

20.2 OpenRemote Heartbeat and Watchdog

As an example, we will monitor our key process, the OpenRemote controller, using a heartbeat and a watchdog script. As heartbeat we will use the file heartbeat.txt, which an OpenRemote rule will overwrite every minute, storing the current date and time to it.

The watchdog script watchdog.sh (watchddog.ps1 for Windows) will read out the content of heartbeat.txt every ten minutes. The script will then compare the current time at execution with the time contained in heartbeat.txt. If the delta of the two time stamps is larger than 10 minutes, the watchdog will send an alarm email. The times can of course be modified and adapted to the availability requirements of your smart home project. In our case the maximum downtime of the smart home system until the alarm email is being sent is 10+1=11minutes. (The delay of one minute is being added by the resolution of the heartbeat, the 10 minutes delay by the threshold set for the watchdog).

We start with the OpenRemote heartbeat rule. Since we want to use the Java Input/Output class java.io we need to load java.io.* with the command import java.io.*;

The heartbeat shall occur every minute, which we achieve with the timer expression: timer (cron:30 ?) At the 30th second of every minute the rule will be evaluated. We want to make sure, that the heartbeat I/O operation does not collide with other I/O operations, which might be scheduled to take place at exactly the full hour, which is why we avoid the expression timer (cron:0 ?).

For the output process we use the heartbeat (Hb) variable writerHb, which we associate with the Java class Writer and its subclass FileWriter. With new Date() we assign the current date to the variable dateHb. With the string formatting option %ts we store the current date in form of the UNIX timestamp (number of seconds since Jan 1, 1970) in the variable strHb. The command

writerHb.write(dateHb.toString()); now writes the UNIX timestamp to the text file heartbeat.txt as outlined below: Date dateHb = new Date();

```
String strHb = String.format("%ts", dateHb ); Date dateHbTime = getTime();
```

FileWriter("Userssmarthome/shProject/shconfig/heartbeat.txt",false);
writerHb.write(strHb.toString()); writerHb.close(); The FileWriter option false
instructs the function to generate a new file every time it is called, while the
option true would append the content to the file in case it exists. In case no file
exists, a new file is generated in any case. We want heartbeat.txt to only contain
the current date and time, which is why we use the FileWriter option false. Since
we want heartbeat.txt to be stored in the shProject directory we specify the path
accordingly. The complete code for the heartbeat rule including the necessary
package definitions and library imports now reads as follows: //Package, globals
and imports:

```
package org.openremote.controller.protocol global  
org.openremote.controller.statuscache.CommandFacade execute; global  
org.openremote.controller.statuscache.SwitchFacade switches; global  
org.openremote.controller.statuscache.LevelFacade levels; import  
org.openremote.controller.protocol.*; import  
org.openremote.controller.model.event.*; import java.sql.Timestamp;  
  
import java.util.*;  
  
import java.lang.Float;  
  
import java.io.*;
```

//-----

//Rule to generate a OpenRemote heartbeat //Every minute the string “OR
Heartbeat!!!! current date” is sent to the Terminal //Every minute current UNIX
timestamp is written to the file heartbeat.txt //-----

rule “OR Watchdog”

timer (cron:30 ?) when

eval(true)

then

Date dateHb = new Date();

String strHb = String.format("%ts", dateHb); System.out.println("!!!OR
Heartbeat!!! "+dateHb+" UNIX Timestamp: "+strHb); //true option: FileWriter
appends to file //false option: FileWriter overwrites file
FileWriter("Userssmarthome/shProject/shconfig/heartbeat.txt",false);
writerHb.write(strHb.toString());

```
writerHb.Close();
```

```
end
```

Now we move on to the watchdog script. Under Windows PowerShell the command [int][double]::Parse((Get-Date -UFormat %s)) returns the UNIX timestamp. We store it in the variable \$watchdog_time: \$watchdog_time = [int][double]::Parse((Get-Date -UFormat %s)) With Get-Content we read the timestamp of heartbeat.txt and store it in the variable \$heartbeat_time:
\$heartbeat_time = Get-Content
“C:\Users\smarthome\shProject\shconfig\heartbeat.txt”

The delta between \$watchdog_time and \$heartbeat_time we store in \$deltaHb:
\$deltaHb = \$watchdog_time - \$heartbeat_time Now we add the below if-statement, which sends the server down notification using the command line mailer application mailsend.exe, as explained in section 19.1.1: if (\$deltaHb -gt 600) {

```
mailsend.exe -to info@keyconceptpress.com -from info@smarthomeserver.com  
-starttls -port 587 -auth -smtp smtp.gmail.de -sub “Open Remote Server  
Notification” +cc +bc -v -user info@smarthomeserver.com -pass  
“yourpassword” -M „!!Open Remote Server down!!“
```

```
}
```

The complete watchdog script, which we store as watchdog.ps1 now reads as below: \$watchdog_time = [int][double]::Parse((Get-Date -UFormat %s))
\$heartbeat_time = Get-Content

“C:\Users\smarthome\shProject\shconfig/heartbeat.txt”

```
$deltaHb = $watchdog_time - $heartbeat_time if ($deltaHb -gt 600) {
```

```
C:\Users\smarthome\mailsend\mailsend1.16.exe -to info@keyconceptpress.com  
-from info@postmaster27.de -starttls -port 587 -auth -smtp smtp.1und1.de -sub  
“Open Remote Server Notificaton” +cc +bc -v -user info@postmaster27.de -pass  
“kyas.net” -M “Open Remote Server down!!!“
```

```
}
```

You should be able to use the Drools heartbeat rule pretty much as listed above. When using the code of *watchdog.ps1* just make sure to adapt the file name and path definitions for *heartbeat.txt* and *mailsend1.16.exe* to your environment. Also make sure to download and install a copy of the command line mailer application *mailsend* as described in section 19.1.1.

As the last step under Windows we now need to configure Task Scheduler to run *watchdog.ps1* at startup and at 10 minute intervals. We select *Start – Control Panel – System* and *Security – Administrative Tools – Task Scheduler*. Then we click on the *Action* menu and select *Create Task*. We give the task a name like *ORWatchdog*, and configure under *Actions* to run *watchdog.ps1*. Under *Triggers* set Task Scheduler to start the task at system startup and to repeat it every 10 minutes (Figure 20.2).

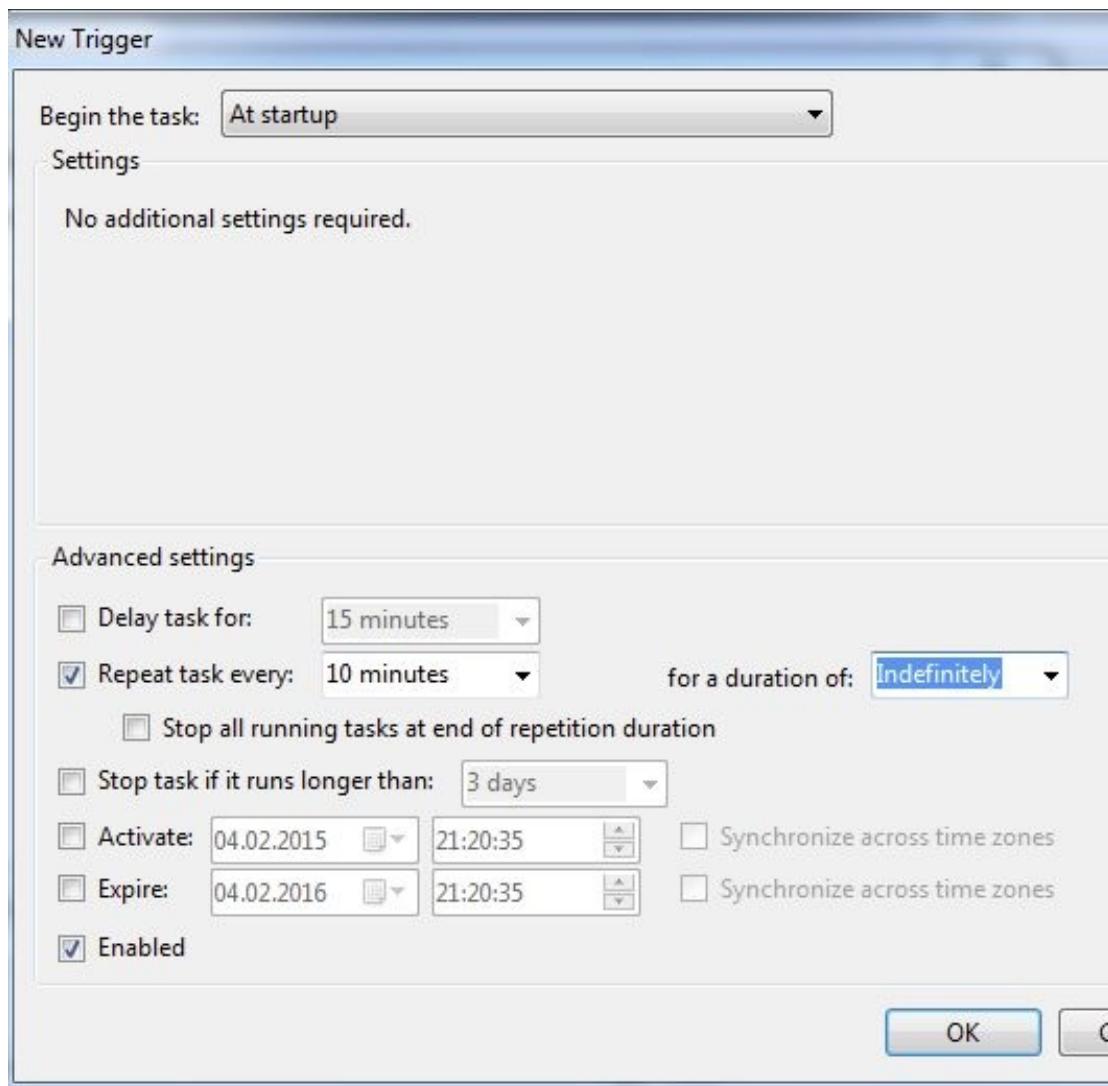


Figure 20.2 Watchdog configuration with Windows Task Scheduler: Run at startup and repeat every 10 minutes The watchdog shell script for macOS, which we also place in the shProject/shconfig directory, looks as follows: #!/bin/sh

```
watchdog_time=$(date +%s);
```

```
heartbeat_time=`cat heartbeat.txt`; deltaHb=$((expr $watchdog_time - $heartbeat_time)); if [ "$deltaHb" -gt "600" ]
```

then

```
./tmailORDown.sh
```

```
fi
```

A few words of explanation to the above code lines: The `date` function provides current date and time. The option `+%s` formats the output of `date` to a Unix timestamp. Using the `cat` command we read the content of `heartbeat.txt` into the variable `heartbeat_time`. Using an `if` statement we compare the delta between `watchdog_time` and `heartbeat_time` to the `watchdog` threshold of 600 seconds. In case `$deltaHb` is larger than 600 seconds the script `tmailORDown.sh` is being executed, which sends out a downtime alert email. (See chapter 14 for a detailed description of the email sending script.) When you are using the script as above make sure to adapt the path statement in the code to your environment in case you use different directory names.

As the last step we configure the macOS launchd agent for our `watchdog` script using LaunchControl is described in more detail in Chapter 14. In LaunchControl we select *User Agent* and *File — New*. In the field *Label* we enter as the name for the task something like `com.watchdog.openremote`. In the left pane we rename the default file name, which is `local.job` to the same name we just chose for the label. Next we drag the symbols *WorkingDirectory* and *StartInterval* from the utility pane on the right hand side of the LaunchControl GUI into the main window. Now we should have a total of four configuration boxes in the main window of our agent configuration: Program to run

Working Directory

Run at load

StartInterval

(*Program to run* and *Run at load* are automatically loaded when opening a new configuration file).

In *Program to run* we now enter the name of our shell script (*watchdog.sh*), in Working Directory the path to its directory: *UserssshProject/shconfig*

If you enter a file name which does not exist or which does not have execution rights, the entry will remain red. The same is true for the working directory path. Both, file name and directory path, need to appear in green for the agent to be able to execute. In *StartInterval* we enter 600. The watchdog script will now be called at startup and from then every ten minutes. We save our agent definition with *File — Save*. When we now select *Load* from the upper right corner of the GUI we run our agent for the first time (Figure 20.3).

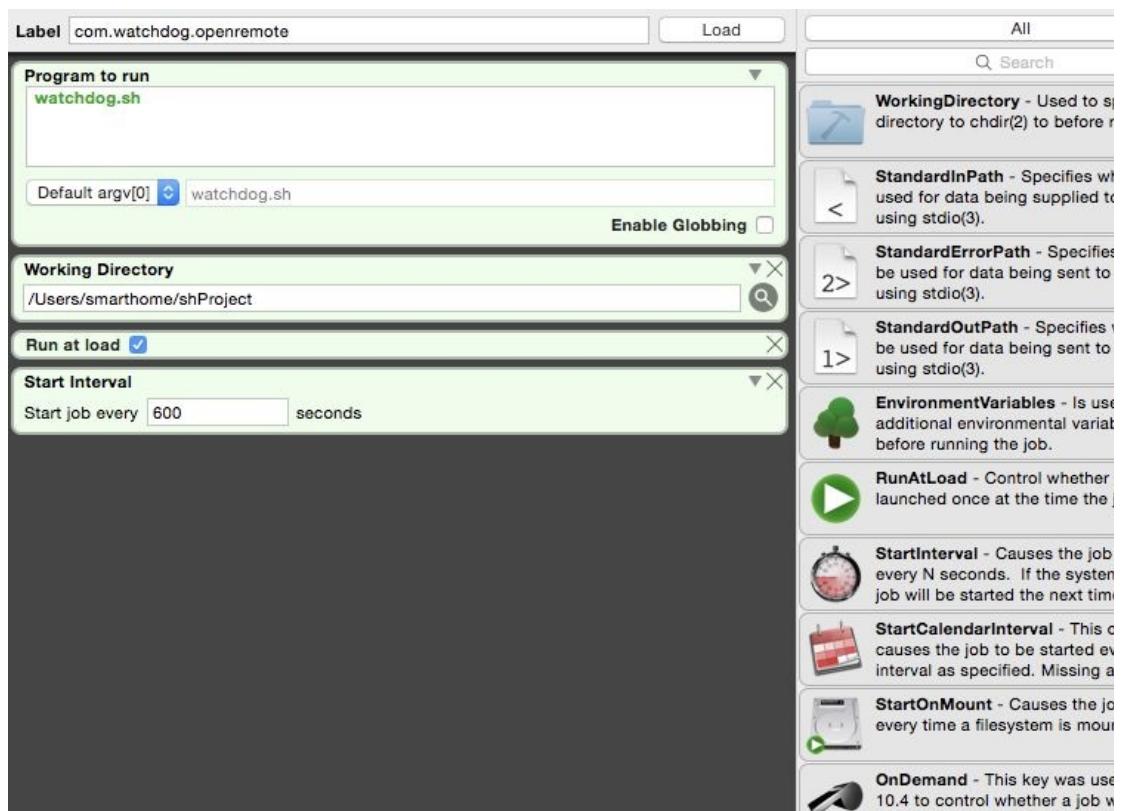


Figure 20.3 Watchdog configuration in macOS with LaunchControl

21 We Proudly Present: Reporting

Equally important as setting up a well tested and safe control infrastructure is the generation of reports on the key operating parameters. Reports are the basis for evaluating the operational parameters of a building and how they play together with its usage as well as the environmental conditions. Most buildings undergo constant changes due to additions, upgrades or changes in usage. Reports can aid in planning and evaluating these changes from the building control perspective. In addition reports play a vital role in monitoring and optimizing the energy consumption of a building. For our project we will create an automated report which hourly records the room temperature for each room, the outside temperature and the weather condition. The data will be stored in a CSV formatted file. Once per day a report for the past 24 hours will be sent out per email to an external email address. In addition all data will be consolidated on a monthly basis and stored accordingly.

21.1 A Drools Reporting Rule

The first step will be to write a Drools rule which stores the data of our room sensors, the outside temperature as well as the weather condition to a CSV file. The code for our corridor temperature sensor reporting rule including the necessary package definitions and library imports reads as follows:

```
//Package, globals and imports: package org.openremote.controller.protocol.global  
org.openremote.controller.statuscache.CommandFacade execute; global  
org.openremote.controller.statuscache.SwitchFacade switches; global  
org.openremote.controller.statuscache.LevelFacade levels; import  
org.openremote.controller.protocol.*; import  
org.openremote.controller.model.event.*; import java.sql.Timestamp;
```

```
import java.util.*;
```

```
import java.lang.Float;
```

```
import java.io.*;
```

```
//-----  
-----
```

```
//Rule to store corridor temperature in CSV format //-----  
-----
```

```
rule "CorridorTempReport"
```

```
timer (cron:0 30 * ?)
```

when

```
$temp7 : Event( source == "CurrentTemperatureCorridor", $tCorr : value ); then
```

```
String.newLine = System.getProperty("line.separator"); String CorrTmp  
= ",CorrTemp,"+$tCorr+newLine; //data output to terminal
```

```
System.out.println(CorrTmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter("Userssmarthome/shProject/reports/tmpreport.txt",true);  
writerRep.write(CorrTmp.toString()); writerRep.close();
```

end

A few words of explanation to the above code lines. Following the rule name we start with the timer command. Since we want the rule to execute at every full hour (second 10, minute 30 of every hour) we specify the following timer expression: timer (cron:10 30 ?)

The when statement in the next line marks the beginning of the rule's LHS. If the sensor exists, its value is stored into a variable. The condition reads: \$temp7 : Event(source == "CurrentTemperatureCorridor", \$tCorr : value); The code line declares a new local variable of type Event called \$temp7. Inside the brackets we have the rule condition, which searches for an entity with the name „CurrentTemperatureCorridor” and which, in case it exists, assigns its value to

the variable \$tCorr. If this condition is true, we have the value of our corridor sensor stored in \$tCorr and the RHS (the then part) of the rule is being executed. By the way, the \$ sign of some of the variables is not a syntax requirement. It is simply used to keep track of various groups and types of variables.

In the RHS side of our rule we first declare the local variable dateRp of type Date, assign it the current date and store it in form of a UNIX timestamp to strRp. Then we define the string newline and assign the system property line.separator to it: String newLine = System.getProperty("line.separator"); At the end of the last sensor data (which in our case is the corridor data) we append this string as a line feed command. Rather than appending /n, which works in some cases (but not in other cases like in Windows environments), this approach achieves a system independent line feed command. Now we write the corridor temperature data followed by a colon and the newLine string to the variable CorrTmp. Using Java FileWriter we write the content of CorrTmp to the file tmpreport.txt.

As the first two entries for each CSV data record we want the current date and the UNIX time stamp, which we add to the first data record, in our case the WeatherConditionBerlin record. Below for the sake of completeness the eight Drools rules for weather condition, outside temperature, bathroom, family room, living room, bedroom1, bedroom1 and corridor: //-----

//Rule to store Weather Condition in CSV format //-----

rule "WeatherConditionReport"

timer (cron:10 30 * ?) when

```
$weather : Event( source == "WeatherConditionBerlin", $WcBerlin : value );  
then
```

```
Date dateRp = new Date();
```

```
String strRp = String.format("%ts", dateRp); String WeatherData =  
dateRp+","+strRp+",WeatherBerlin,"+$Berlin; //data output to terminal
```

```
System.out.println(WeatherData); //data output to file tmpreport.txt  
Writer writerRep = new  
FileWriter("Userssmarthome/shProject/reports/tmreport.txt",true);  
writerRep.write(WeatherData.toString()); writerRep.close();
```

```
end
```

```
//-----
```

```
//Rule to store outside temperature in CSV format //-----
```

```
rule "OutsideTempReport"
```

```
timer (cron:15 30 * ?) when
```

```
$temp1 : Event( source == “Tempberlin”, $TempBerlin : value ); then
```

```
String OutsideTmp =”,TempBerlin,”+$TempBerlin; //data output to terminal
```

```
System.out.println(OutsideTmp); //data output to file tmpreport.txt Writer  
writerRep = new  
FileWriter(“Userssmarthome/shProject/reports/tmreport.txt”,true);  
writerRep.write(OutsideTmp.toString()); writerRep.close();
```

```
end
```

```
//-----
```

```
//Rule to store bathroom temperature in CSV format //-----
```

```
rule “BathTempReport”
```

```
timer (cron:20 30 * ?) when
```

```
$temp2 : Event( source == “CurrentTemperatureBathroom”, $tBath : value );  
then
```

```
String BathTmp =” ,BathTemp,”+$tBath; //data output to terminal
```

```
System.out.println(BathTmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter(“Userssmarthome/shProject/reports/tmpreport.txt”,true);  
writerRep.write(BathTmp.toString()); writerRep.close();
```

```
end
```

```
//-----  
-----
```

```
//Rule to store livingroom temperature in CSV format //-----  
-----
```

```
rule “LivingTempReport”
```

```
timer (cron:25 30 * ?) when
```

```
$temp3 : Event( source == “CurrentTemperatureLivingroom”, $tLiv : value );  
then
```

```
String LivingTmp =”,LivingTemp,”+$tLiv; //data output to terminal
```

```
System.out.println(LivingTmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter(“Userssmarthome/shProject/reports/tmpreport.txt”,true);  
writerRep.write(LivingTmp.toString()); writerRep.close();
```

```
end
```

```
//-----  
-----
```

```
//Rule to store familyroom temperature in CSV format //-----  
-----
```

```
rule “FamilyTempReport”
```

timer (cron:30 30 * ?) when

```
$temp4 : Event( source == "CurrentTemperatureFamilyroom", $tFam : value );  
then
```

```
String FamilyTmp =",FamilyTemp,"+$tFam; //data output to terminal
```

```
System.out.println(FamilyTmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter("Userssmarthome/shProject/reports/tmpreport.txt",true);  
writerRep.write(FamilyTmp.toString()); writerRep.close();
```

end

//-----

```
//Rule to store bedroom1 temperature in CSV format //-----  
-----
```

rule "Bedroom1TempReport"

```
timer (cron:35 30 * ?) when
```

```
$temp5 : Event( source == “CurrentTemperatureBedroom1”, $tBed1 : value );  
then
```

```
String Bed1Tmp =”,Bed1Temp,”+$tBed1; //data output to terminal
```

```
System.out.println(Bed1Tmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter(“Userssmarthome/shProject/reports/tmpreport.txt”,true);  
writerRep.write(Bed1Tmp.toString()); writerRep.close();
```

```
end
```

```
//-----  
-----
```

```
//Rule to store bedroom2 temperature in CSV format //-----  
-----
```

```
rule “Bedroom2TempReport”
```

timer (cron:40 30 * ?) when

```
$temp6 : Event( source == "CurrentTemperatureBedroom2", $tBed2 : value );  
then
```

```
String Bed2Tmp =",Bed2Temp,"+$tBed2; //data output to terminal
```

```
System.out.println(Bed2Tmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter("Userssmarthome/shProject/reports/tmpreport.txt",true);  
writerRep.write(Bed2Tmp.toString()); writerRep.close();
```

end

//-----

//Rule to store corridor temperature in CSV format //-----

rule "CorridorTempReport"

```
rule CurrentTemperatureReport
```

```
timer (cron:45 30 * ?) when
```

```
$temp7 : Event( source == "CurrentTemperatureCorridor", $tCorr : value ); then
```

```
String.newLine = System.getProperty("line.separator"); String CorrTmp  
= ",CorrTemp,"+$tCorr+newLine; //data output to terminal
```

```
System.out.println(CorrTmp);
```

```
//data output to file tmpreport.txt Writer writerRep = new  
FileWriter("Userssmarthome/shProject/reports/tmpreport.txt",true);  
writerRep.write(CorrTmp.toString()); writerRep.close();
```

```
end
```

21.1.1 /O Exception Handling

In case you have not written code with I/O commands before, be aware, that you also should add basic error (exception) handling along with the `FileWriter` commands. Chances are, that events occur during I/O operations, which cause the software to produce errors and terminate. For example, when writing to a file, the operating system may not permit to do so, perhaps because the disk is locked, or there is no space available. You must prepare your code to catch this exception and therefore prevent the program from terminating. In Java to catch an exception the statements that might throw the exception are encapsulated in a `try` block. When we catch an exception we name the class of the exception (in our case `IOException`) and define a variable that will have that type. Within the body of the catch clause we can print out an exception message. For our example

a basic exception handling code could look as follows: try

```
{    output = new PrintWriter(new FileWriter("data.txt")); ...  
}  
  
catch(IOException e)  
  
{    System.out.println(" File could not be created: " + e); }
```

21.1.2 Report File Management

For the handling of the report file generated by the above Drools rule under macOS we create two simple shell scripts. (Under Windows the analog functionality would be implemented using Powershell). One script sends out an email with the CSV file tmpreport.txt attached. The second one renames the file to a date based filename, creates a new, empty file version of tmpreport.txt and removes all report files older than 90 days. We start with the script reportmail.sh, which sends out an email with the file tmpreport.txt as an attachment. As explained in detail in chapter 14 we use an Applescript, which we insert inside a shell script. Compared to the example from chapter 14 we just add the command make new attachment as shown below: make new attachment with properties {file name:"Macintosh HD:Users:smarthome:shProject:reports:tmpreport.txt" as alias}

With that the code for reportmail.sh looks as follows: #!/bin/sh

```
exec osascript << EOF
```

```
set recipientName to "Smarthome"
```

```
set recipientAddress to "info@keyconceptpress.com"
```

```
set theSubject to "Smart Home Report"
```

```
set theTime to current date
```

```
set theText to "OpenRemote Server Beachroad 4, Upville daily report"
```

```
set theContent to theText & theTime tell application "Mail"
```

```
set theMessage to make new outgoing message with properties  
{subject:theSubject, content:theContent, visible:true}
```

```
tell theMessage
```

```
make new to recipient with properties {name:recipientName,  
address:recipientAddress}
```

```
make new attachment with properties {file name:"Macintosh  
HD:Users:smarthome:shProject:reports:tmpreport.txt" as alias}
```

```
send
```

```
end tell
```

```
end tell
```

```
EOF
```

Now we start with the second script reportmgm.sh. At the beginning we call the script reportmail.sh, which sends out the email with the report file as an attachment. Then the command sleep 10s stops the script processing for ten seconds to ensure the email send process finishes before continuing with the remaining script operations. We then define the variables day (containing the current date), filename (containing the date based filename) and path (containing the path to the report directory). The move command (mv) renames the current report file tmpreport.txt to a date based filename, the touch commands creates a new, empty report file. And finally the command find \$path/*.txt -mtime +90 -exec rm {} \

finds all files in our report directory which for the last time were modified more than 90 days ago (-mtime +90) and removes them (rm {}). Following the above description our shell script reportmgm.sh looks as follows:

```
#!/bin/sh
```

```
# Daily OpenRemote report handling script # Variables for the script
```

```
./reportmail.sh
```

```
sleep 10s
```

```
day=$(date +%F)
```

```
filename="$day.txt"
```

```
path="/Users$smarthehome/shProject/reports"
```

```
mv $path/tmpreport.txt $path/$filename touch $path/tmpreport.txt
```

```
# Removal of report files older than 30 days find $path/*.txt -mtime +90 -exec  
rm {} \
```

After thorough testing of our script we can create an OpenRemote command which calls our script reportmgm.sh and write a Drools rule, which calls this command once per day. For a detailed description see chapter 8, where we initiate an iTunes script from Drools. Alternatively we can schedule the start of reportmgm.sh using launchd (macOS) or Task Scheduler (for starting the Powershell variant of the script under Windows) as discussed in chapter nineteen.

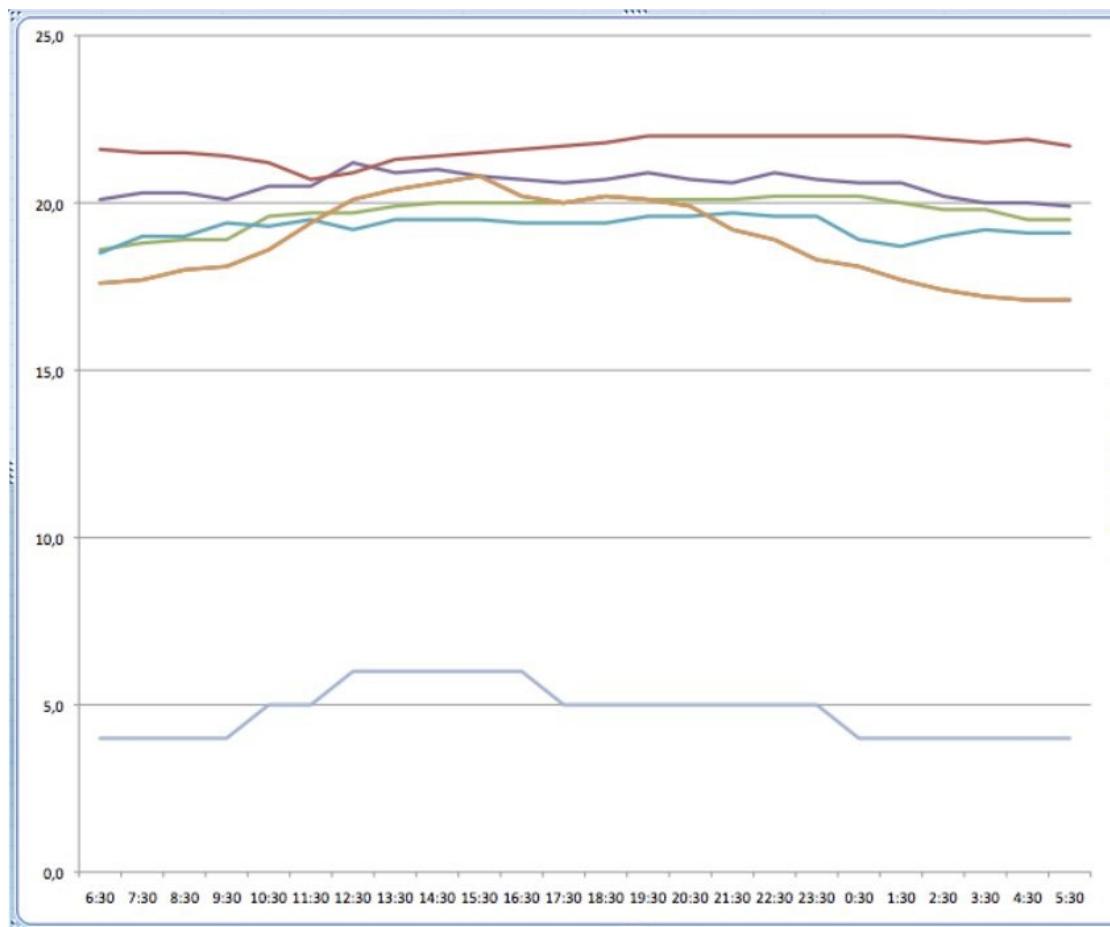


Figure 21.1 The daily smart home temperature report

22 Appendix

22.1 Upgrading from OpenRemote Designer to Professional Designer (2.1 to 2.6)

With the introduction of the new OpenRemote Professional Designer and OpenRemote Controller version 2.6 a number of important capabilities were introduced. Among them new and updated implementations of communication protocols and support for the latest version of Java and Drools. Since key components of the internal OpenRemote Controller structure changed, it does require the projects to reside in the new OpenRemote Professional designer. Projects which have been developed in the previous OpenRemote Designer have to be re-implemented in OpenRemote Professional Designer, if you want to use OpenRemote Controller 2.6. This means you have to manually reenter all devices, commands, sensors and switches in OpenRemote Professional Designer. To make this process as painless as possible, below a few recommendations for how to go about it.

When reentering all of your OpenRemote Designer Devices, Commands, Sensors, Switches and Panels in OpenRemote Professional Designer, it is easiest to open two browsers windows and place them side by side, one containing the old OpenRemote Designer, and one the new OpenRemote Professional Designer. Then start to copy entries one by one from your existing OpenRemote Design project to your new OpenRemote Professional Designer account. When doing this I recommend to port one functional group - *e.g.* all devices, commands, sensors and switches for one panel - after the other. Once a functional group is complete in your new OpenRemote Professional Designer project, you should test if everything works as expected, before you move on to the next functional group (Figure 22.1).

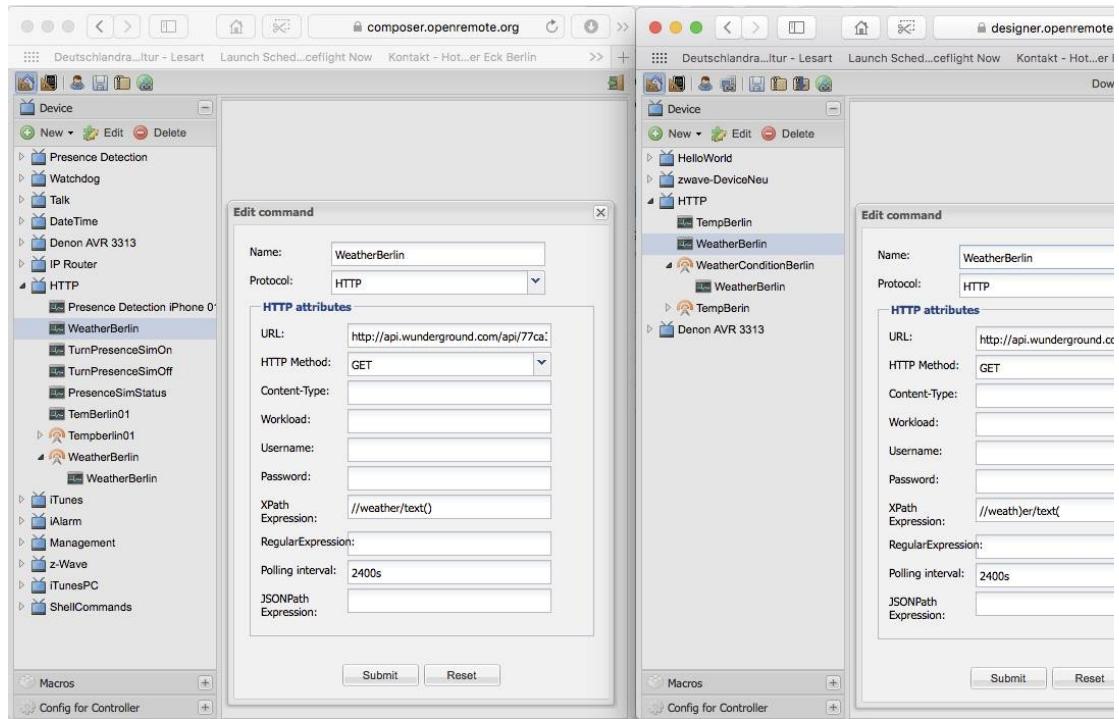


Figure 22.1 Porting a project from OpenRemote Designer to OpenRemote Professional Designer

22.1.1 Porting HTTP Commands

Be careful when porting HTTP commands. The HTML Server directory for the new controller 2.6 is

`http://localhost:8688/controller ...`

and not under

`http://localhost:8080/controller ...`

where it used to be up to controller version 2.1. So make sure you update all HTTP command entries, which reference the internal OpenRemote HTML server directory accordingly.

22.1.2 Porting Scripts and Files

In case you have put scripts or files into the 2.1 controller web sever directory

..`/ORC/webapps/controller/xxx.html`

make sure you do not forget to copy those to the according directory of your new 2.6 controller. The same is the case for any files or scripts you have placed in the

`ORC/bin`

directory. Ideally you have placed all custom scripts in a seperate directory above the OpenRemote software directory, as recommended in this book. (For this purpose we have created the directory `shProject/shconfig`, which is a directory at the same level than `shProject/ORC`). Then you do not have to deal with a mix of your custom files and OpenRemote software components.

22.1.3 Updating \$PATH and \$JAVA_HOME

If you change your directory structure and / or update the Java version on your system, make sure to update your `$PATH` and `$JAVA_HOME` definitions accordingly.

22.2 Troubleshooting Strategies

Unfortunately there are many things which can go wrong when you try to integrate multiple devices from different vendors using multiple programming languages and communication protocols. Below a list of troubleshooting strategies and symptom - root-cause pairs, which however can only be a small part of what you will really need.

- Once you have a working system, make a backup-copy of the system before making any further changes.
- When making any changes, and when you want to determine if you have a clean running system you need to start OpenRemote in developer mode with the run command. Then the startup sequence can be seen in the terminal window. In particular look if the Drools Rule engine initializes without any error, and that the sensors initialize without error messages. If the Drools engine is not initialized, none of your rules will work.
- If you have made changes, and the result is very different than expected, rebooting not only OpenRemote but the entire host makes sense to make sure you are not dealing with a caching problem. Also after changing things like sensor definitions, shell script commands or commands involving other protocols, restarting the host is a good idea, and can correct sporadic errors of commands and sensors.
- To make sure you have the latest panel identity on your OpenRemote smartphone controller, load a test panel identity and then the working panel identity to make sure you actually have loaded the latest version on your control device.
- When developing shell scripts do an extensive testing of the scripts before integrating them into OpenRemote. Also be aware, that OpenRemote executes shell scripts from its ORC/bin directory. If the script shall interact with files in other directories, you need to use the long absolute path definitions to make sure things work correctly. Do not use relative path definitions.
- Make sure the WiFi connection from your controller smartphone or pad to the

OpenRemote server is available and reasonably fast

22.3 Problem Symptom - Cause Pairs

Symptom: Slow GUI, intermittent, delayed sensor updates Possible cause: Large number of sensors with short polling intervals of one second or below. I recommend to always select the maximum polling interval which can fulfill the required functionality. In a smart home environment typical polling intervals are 5, 10, 60 or 1800 seconds. Sensors polling values in intervals below one second should be the absolute exception. If you end up with large numbers of sensors polling shell scripts, HTTP requests or other entities in small intervals such as one second or below, the overall performance of your system will start being impacted.

Symptom: Symptom: Slow GUI, missing sensor updates Possible Cause: Sensor command failing to query its value. Reasons: HTTP request fails due to server outage or requested HTML page no more available. Other reasons are changed or deleted shell script files, missing file execution permission or script commands, which are not working anymore. Restart the OpenRemote Controller from the terminal window in development mode with the run command and look for sensor related error messages in the start up process such as the one below: ERROR [main]: Creating sensor failed. Error : The node '2' is not known by the Z-Wave controller XML Element :
<sensor xmlns="http://www.openremote.org" id="300305"
name="Device 2-OnOffSensor" type="switch"> <include
type="command" ref="59" /> <state name="on" />

<state name="off" />

</sensor>

Symptom: Rule changes not reflected after saving and restart of OpenRemote controller
Possible Cause: OpenRemote rules definition file .../ORC/webapps/controller/rules/modeler_rules.drl failed to update during Designer - Controller synchronization process. Due to an incomplete synchronization process it can happen, that your local file is actually different than your latest rule definition in OpenRemote Designer. Open the local rules definition file modeler_rules.drl with a text editor and validate if it contains the latest changes. If not save all changes in Professional Designer and synchronize Designer and controller again.

Symptom: Rules do not execute

Possible Cause: Syntax error in rules definition. This will probably be the most frequent root cause for problems with Drools. Drools is very syntax sensitive and fails to load if there is the smallest syntax error in the rules definition file. Therefore after any change in your rules definitions monitor the start up process of your OpenRemote controller and make sure Drools is successfully processing your rules definition file, which is done with a line like the following at the very beginning of the start up process: INFO 2013-05-09 14:54:13,328 : Initialized event processor : Drools Rule Engine Even if only one rule is incorrect, Drools will fail to load all of your rules and none of your rule definitions will be available.

Symptom: Shell script is not executing correctly
Possible Cause: The path definitions within the shell script do not reflect the fact, that OpenRemote executes shell scripts from its bin directory, independent of where the shell script sources reside. In our case this is the directory shProject/ORC/bin. If for example the script homeemptyswitch.sh in the directory shProject/shconfig is supposed to change the value of homeempty.txt, which is as well in shProject/shconfig, the path definition needs to read as follows:

```
#!/bin/sh
```

```
echo on > ../../shconfig/homeempty.txt
```

Best is to use full, absolute path definitions instead of relative path definitions. So avoid the shorter but higher risk relative path definitions all together.

Symptom: AutoDetect mode of the OpenRemote app does not recognize the active controller Possible Cause: No panel identity available in Designer or at the local controller. Make sure you actually hit the save button in Designer. Also look for the Designer confirmation message, which confirms that all changes have been saved. It pops up in a small window at the bottom right of the GUI after each save command. Then make sure the controller actually synchronizes with your Designer account. With no available panel design the controller will not be detected by the OpenRemote smartphoen app, which is the last step in our installation procedure.

22.3 A Brief Introduction to JavaScript and Google Script

As you are probably aware of, JavaScript one of the major programming languages in WWW environments. It is used by Web browsers and Web servers to implement any functionality you can think off, and it is supported by all Web browsers by default, without the need for a plug-in. JavaScript is object based. Object oriented programming instead of using variables and functions, is based on data and behavior. In other words you are grouping variables and functions to functional entities called objects. The behavior part of objects, which are the functions inside objects, are called methods. Generic Objects, which can be reused to create custom objects are called a class.

In order to implement a Google sheets based smart home functionality you will need some basic understanding of JavaScript, which is why it is a good idea to go over one of the many good beginner tutorials on JavaScript in the Internet such as <https://www.youtube.com/watch?v=zPHerhks2Vg>. For the most part you will just need to do slight modifications of the code which is provided with the description of the sample project in this book. However it is still a good idea to spend an hour or two to get familiar with JavaScript, since it is such an integral technology of Internet today, and you will most likely need it again sooner or later.

Along with its cloud based spreadsheet service Google provides a JavaScript based programming environment called Google Scripts. By adding a library of classes for Google sheets, it allows to easily manipulate spreadsheets and data. To get started the most important classes are Spradsheet, Sheet and Range. Each of the classes contains a list of methods (functions), which provide the actual functionality such as reading out cell values or writing values into cells. Objects and methods are referred to with the so called dot notation (e.g. objectA.objectB.method) or the bracket notation (e.g. object.value[]). Objects can also contain other objects. The logic becomes clear, when we look at a first example, in which we want to write the value 8 to cell A5 in our spreadsheet. Go

to Google Sheets, open your spreadsheet smarthome and give the first sheet a name such as Lighting. Then select *Tools –Script Editor ...*, rename the generic function myfunction, which you will find in the script editor, to something like insertData and paste in the below code.

```
function insertData() {  
  
var spreadsheet = SpreadsheetApp.getActiveSpreadsheet();  
  
var sheet = spreadsheet.getSheetByName("Lighting");  
  
var cell = sheet.getRange("A5");  
  
cell.setValue(999);  
  
}
```

Click on run and confirm the request for authorization Google sheets requires when running a script for the first time. If you now switch to your spreadsheet, you should see the value 999 in cell A5. In the first line of the code example we assign a reference to the active spreadsheet by calling the method `getActiveSpreadsheet()` of the object `SpreadsheetApp` and assign it to the variable `spreadsheet`:

```
var sheet = spreadsheet.getSheetByName("Lighting");
```

Then we create the variable sheet and assign it the reference to the sheet Lighting:

```
var sheet = spreadsheet.getSheetByName("Lighting");
```

And we create the variable cell and assign it the reference to cell A5:

```
var cell = sheet.getRange("A5");
```

In the last step we use the method set.Value() to assign the value 999 to our variable cell:

```
cell.setValue(999);
```

We could also have implemented the above in a single line, which even better demonstrates the logic behind the object oriented structure of JavaScript:

```
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Rules").getRange("A
```

In practice however our first approach, using multiple variables, is preferable, since it provides us with the flexibility to reuse them for further functionality.

Class	Method	Description
SpreadsheetApp	getActiveSpreadsheet()	Returns the currently active spreadsheet
SpreadsheetApp	openById(id)	Opens the spreadsheet with the given ID

SpreadsheetApp	getSheetByName(name)	Returns a sheet with the given name
Sheet	getRange(row, column)	Returns the range as specified in A1 notation
Sheet	appendRow(rowContents)	Appends a row to the spreadsheet
Range	setValue(value)	Sets the value of the range
Range	clear()	Clears the range of contents, formats, and data-validation rules

Figure 22.2 Important SpreadsheetApp methods

I

23 Bibliography

US Department of Energy. (2016): Total Energy Interactive Table Browser.
[www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?
f=A&start=2013&end=2014&charted=3-6-9-12](http://www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?f=A&start=2013&end=2014&charted=3-6-9-12)

Eur US Department of Energy. (2016): Total Energy Interactive Table Browser.
[www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?
f=A&start=2013&end=2014&charted=3-6-9-12](http://www.eia.gov/beta/MER/index.cfm?tbl=T02.01#/?f=A&start=2013&end=2014&charted=3-6-9-12)

Eurostat European Commission (2016): Final energy consumption, by sector.
ec.europa.eu/eurostat/data/database?node_code=tsdpc320

Enerdata. (2015) Energy Efficiency Trends in Residential in the EU.
www.odyssee-mure.eu/publications/efficiency-by-sector/household/

US Department of Energy. (2016) Annual Energy Outlook 2015.
<https://www.eia.gov/beta/aeo/#/?id=4-AEO2015&cases=ref2015&sourcekey=0>

Baggini, Angelo and Lyn Meany. (2012) Application Note Building Automation and Energy Efficiency: The EN 15232 Standard. http://www.leonardo-energy.org/sites/leonardo-energy/files/Cu0163_AN_Building%20automation_v1_bis.pdf

Weiping Sun, Munhwan Choi and Sunghyun Choi (2013): 802.ah - A long range 802.11 WLAN. [IEEE 802.ah: A long range 802.11 WLAN](#)

WiFi Alliance (2016): Wi-Fi HaLow. <http://www.wi-fi.org/discover-wi-fi/wi-fi-hallow>

Home Grid Forum (2013): Converging Technologies. [Converging Technologies - Moving from HomePNA to G.hn.](#)

IEEE Computer Society (2011): IEEE Standard for Local and metropolitan area networks Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). <http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf>

International Telecommunication Union (2012): Recommendation ITU-T G.9959 Short range narrow-band digital radio communication transceivers – PHY and MAC layer specifications. <https://www.itu.int/rec/T-REC-G.9959>

EnOcean Alliance (2013): EnOcean Wireless Standard ISO/IEC 14543-3-10. https://www.enocean-alliance.org/en/enocean_standard/

The Thread Group (2017): What is Thread? <http://threadgroup.org/What-is-Thread/Connected-Home>

Carnegie Mellon University, Software Engineering Institute (2011): Vulnerability Note VU#357851 - UPnP requests accepted over router WAN interfaces. <http://www.kb.cert.org/vuls/id/357851>

Postscapes (2017): IoT Standards and Protocols. <https://www.postscapes.com/internet-of-things-protocols/>

Dominique Guinard, Iulia Ion, and Simon Mayer (2011): In Search of an Internet of Things Service Architecture: REST or WS-*?

<http://www.vs.inf.ethz.ch/publ/papers/dguinard-rest-vs-ws.pdf>

Earlence Fernandes, Kevin Eykholt, *et al.* (2016): Security Analysis of Emerging Smart Home Applications. <https://iotsecurity.eecs.umich.edu>

marketsandmarkets (2017): Building Automation System Market by Communication Technology. Global Forecast to 2022.

<http://www.marketsandmarkets.com/Market-Reports/building-automation-control-systems-market-408.html>

Zion Market Research (2016): Smart Home Market (Smart Kitchen, Security & Access Control, Lighting Control, Home Healthcare, HVAC Control and Others): Global Industry Perspective, Comprehensive Analysis and Forecast, 2016-2022 <https://www.zionmarketresearch.com/sample/smarthome-market>

Weather Underground - The Weather Channel (2016): Data Features: URL Format. <https://www.wunderground.com/weather/api/d/docs?d=data/index>.

DD&M Holdings Inc. (2013): DENON AVR control protocol 5.2a.

http://www.denon-media.ru/upload/2014/06/09/avr1000_e300_protocol_10.0.0_v04.pdf

Christian Paetz, Serguei Polterak (2001): ZWay Manual Z Wave.Me. http://en.z-wave.me/docs/zway_manual_en.pdf

DD&M Holdings Inc. (2013): DENON AVR control protocol 5.2a.

<http://www.denon->

media.ru/upload/2014/06/09/avrx1000_e300_protocol_10.0.0_v04.pdf

Christian Paetz, Serguei Polterak (2001): ZWay Manual Z Wave.Me. http://en.z-wave.me/docs/zway_manual_en.pdf

This eBook was posted by AlenMiler on AvaxHome!

Many New eBooks in my Blog:

<http://avxhome.in/blogs/AlenMiler>

Mirror: <https://avxhome.unblocked.tw/blogs/AlenMiler>