



Get Ready for Activity – Ambient Day Scheduling with Dementia

Field tested software components

Deliverable Name: D2.4 – Field tested software components

Deliverable Date: 31.01.2020

Classification: Report / public

Authors: Quirino Nardin, Walter Ritter, Tom Ulmer, Sandro Emmenegger, Beat Sauter,

Document Version: V2.0

Project University of Applied Sciences Vorarlberg (FHV), Austria
Coordinator:

Project Partners: Bartenbach GmbH
 Fachhochschule St. Gallen
 Apollis – Institut für Sozialforschung und Demoskopie O.H.G.
 Intefox GmbH
 Altersheim Stiftung Griesfeld
 EMT – energy management team AG
 CURAVIVA Schweiz
 Tirol Kliniken GmbH – Hall

The project GREAT no AAL-2016-023 is funded through the AAL program of the EU



Preface

This document forms part of the Research Project “Get Ready for Activity – Ambient Day Scheduling with Dementia (GREAT)” funded by the AAL 2016 “Living well with dementia” funding program as project number AAL-2016-023. The GREAT project will produce the following Deliverables:

- D1.1 Medical, psychological, and technological framework
- D2.1 Applicable hardware components
- D2.2 Applicable software components
- D2.3 Field tested hardware components
- D2.4 Field tested software components
- D3.1 Implementation report
- D3.2 Field test report
- D4.1 Communication strategy
- D4.2 Stakeholder management report
- D5.1 Report on market analysis
- D5.2 Dissemination plan
- D5.3 Final business plan

The GREAT project and its objectives are documented at the project website <https://uct-web.labs.fhv.at>. More information on GREAT and its results can also be obtained from the project consortium:

Prof. Dr. Guido Kempter (project manager), University of Applied Sciences Vorarlberg (FHV), Phone: + 43 5572 792 7300, Email: guido.kempter@fhv.at

Hermann Atz, Institute for Social Research and Opinion Polling OHG (APOLLIS), Phone: +39 0471 970115, Email: hermann.atz@apollis.it

Mag. Wilfried Pohl, Bartenbach GmbH, Phone: +43-512-3338-66, Email: wilfried.pohl@bartenbach.com

Quirino Nardin, Intefox GmbH, Phone: +43 699 1900 8889, Email: info@intefox.com

Dr. Marksteiner Josef, Tirol Kliniken Hall, Phone: +43 (0)50504 33000, Email: josef.marksteiner@tirol-kliniken.at

Mag. Tom Ulmer, University of Applied Sciences St. Gallen (FHS), Phone: +41 71 226 17 41, Email: tom.ulmer@fhsg.ch

Beat Sauter, energy management team ag (emt), Phone: +41 71 660 02 86, Email: beat.sauter@emt.ch

Anna Jörger, CURAVIVA Schweiz, Phone: +43 (0)31 385 33 45, Email: a.joerger@curaviva.ch

Cornelia Ebner, Stiftung Griesfeld, ÖBPB – APSP, Phone: +39 (0) 471 82 63 43, Email: cornelia.ebner@griesfeld.it

Content

1.	Great Prototype Software Components.....	7
1.1.	System Overview	7
1.2.	System Architecture.....	8
2.	Controller	9
2.1.	Intefox Middleware.....	9
2.2.	Wireless Access Point, Router.....	11
2.3.	OpenVPN Client	15
3.	Intefox - Middleware	16
3.1.	Architecture and Components (fox.core)	16
3.2.	Configurator Software.....	16
3.2.1.	Online Bundle Manager.....	17
3.3.	Logging and Data Access Interface	18
3.3.1.	Event Logger Configuration	19
3.3.2.	Log settings (Selecting the events to be logged).....	21
3.3.3.	Data Access Interface.....	22
3.3.4.	Database Structure	26
3.4.	Controller & Visualization interface	30
3.4.1.	LiveCycle of a Visualization Client	30
3.4.2.	Login	30
3.4.3.	Request Object Descriptions.....	31
3.4.4.	Request Structure.....	34
3.4.5.	LongPoll Request.....	36
3.4.6.	Send Commands.....	37
4.	Light	38
4.1.	Biodynamic Light Extension Bundle	38
4.1.1.	Light Curve Object Description	40
4.1.2.	Protocol Data Exchange between Light and Controller	44
4.2.	DATA: Controller to Light.....	44
4.3.	Statusresponse from Light to Controller.....	45
4.4.	Teach In.....	45
4.5.	Definition Factory Default.....	46
4.6.	Luminaire.....	46
5.	Sound Module	48

5.1.	Image Preparation.....	49
5.1.1.	Basics	49
5.2.	Shairport-Support (for AirPlay functionality, optional)	50
5.3.	WLAN Setup.....	51
5.4.	Music Module Extension Bundle.....	52
5.5.	Sound Module Protocol Description	56
5.5.1.	Commands from Client to Server:.....	57
5.5.2.	Commands from Server to Client:.....	60
5.6.	Relevance/Reuse-Potential outside GREAT	61
5.7.	Sound Concept.....	62
6.	Scent Module.....	66
6.1.	Image Preparation.....	67
6.2.	Scent Module Extension Bundle	67
6.3.	Scent Module Protocol Description	70
6.3.1.	Commands from Client to Server:.....	70
6.3.2.	Commands from Server to Client:.....	71
6.4.	Relevance/Reuse-Potential outside GREAT	72
7.	Enocean Repeater	73
8.	Sensors	75
8.1.	PIR Sensor	75
8.2.	Biovotion Everion Sensor	75
8.3.	Stress Detection	76
8.3.1.	Architecture	77
8.4.	Gathering Vital Data (test phase 1)	78
8.4.1.	Setup.....	79
8.4.2.	Pairing Malfunction Tips	82
8.4.3.	Run Data Gathering.....	82
8.4.4.	High Stress Triggers	84
8.4.5.	Integration into the GREAT System.....	84
9.	User-Interface for Manual Control.....	86
10.	GREAT User-Interface	88
11.	GREAT Manager	94
11.1.	Curve Editor.....	95
11.2.	Schedule Editor.....	96
11.3.	Playlist editor.....	100
11.4.	Data Download Area.....	101

12. Automated Control System	102
12.1. Measuring stress.....	103
12.2. Machine learning	103
12.3. Challenges and coping strategies.....	106
12.4. Implementation	108
12.4.1. Feature engineering.....	108
12.4.2. Processing pipeline of the learning system	110
12.5. Schedule based operation.....	113
12.6. Rule-Based Recommendations.....	114
12.6.1. Processing Rule.....	114
12.6.2. Continuous Adjustment of Variables	115
12.6.3. Block diagram and parameters of the control system.....	116
12.7. Summary and further research	117
13. Setup Procedure of the Final GREAT Product.....	119
14. References	124
15. List of Figures	125
16. List of Tables	127

1. Great Prototype Software Components

1.1. System Overview

The GREAT system should be usable in widely varying environments. Therefore, a highly modular approach has been chosen. Components like light, sound, and scent modules can be used individually or in combination with one another. The system also gathers data from motion detectors and physiology sensors worn by caregivers to detect potential activity/relaxation levels of persons in a room. The system can be controlled via a mobile app manually as well as dedicated hardware buttons, that can be added to the system based on local requirements (see Figure 1 for an overview).

One important principle of the GREAT system is that users must always be in full control of the system, meaning that they will always be able to start/stop actions manually.

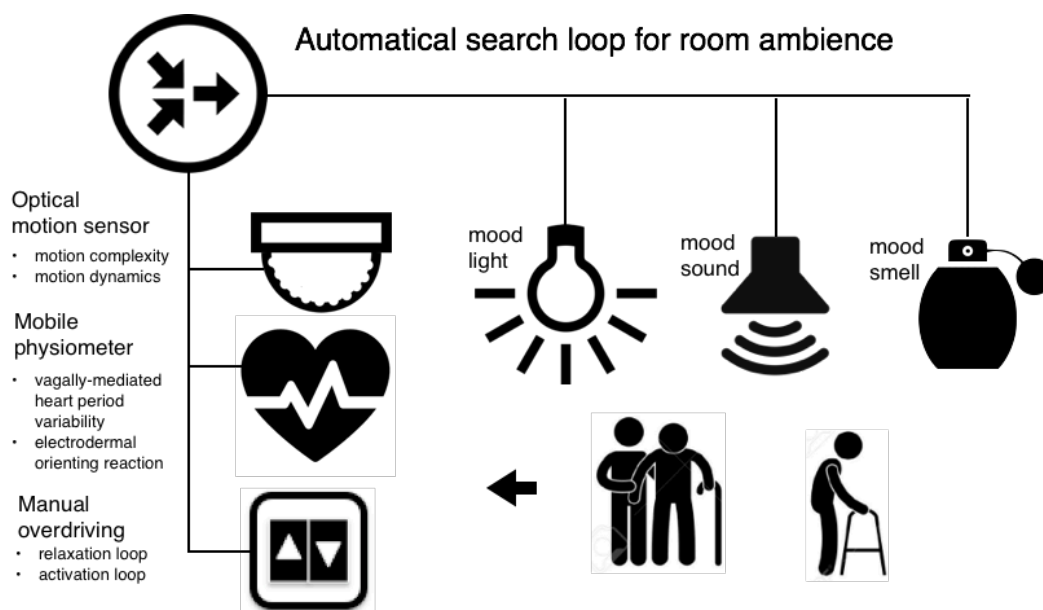


Figure 1: GREAT Components Overview, Source: GREAT consortium.

In a first phase, the system gathers data from motion detectors and physiology sensors as foundation for an analysis of typical patterns. Caregivers are also able to give their current impression of the patient's state (e.g. whether they are very relaxed up to highly activated). During this time caregivers can manually trigger activation- or relaxation-cycles. For learning purposes, the system logs every activity. In a second phase, the system recommends to caregivers the triggering of activation-/relaxation-cycles, when it detects certain situations. The actual triggering of these cycles however is still up to the caregivers. By the end of the field tests, the system should have gathered enough data to trigger activation/relaxation cycles automatically.

1.2. System Architecture

The GREAT system is comprised of a main controller, light-, scent-, and sound-modules, sensors for room-based motion activity and physiological data capturing for selected caregivers, as well as a cloud based storage and configuration layer (see Figure 2).

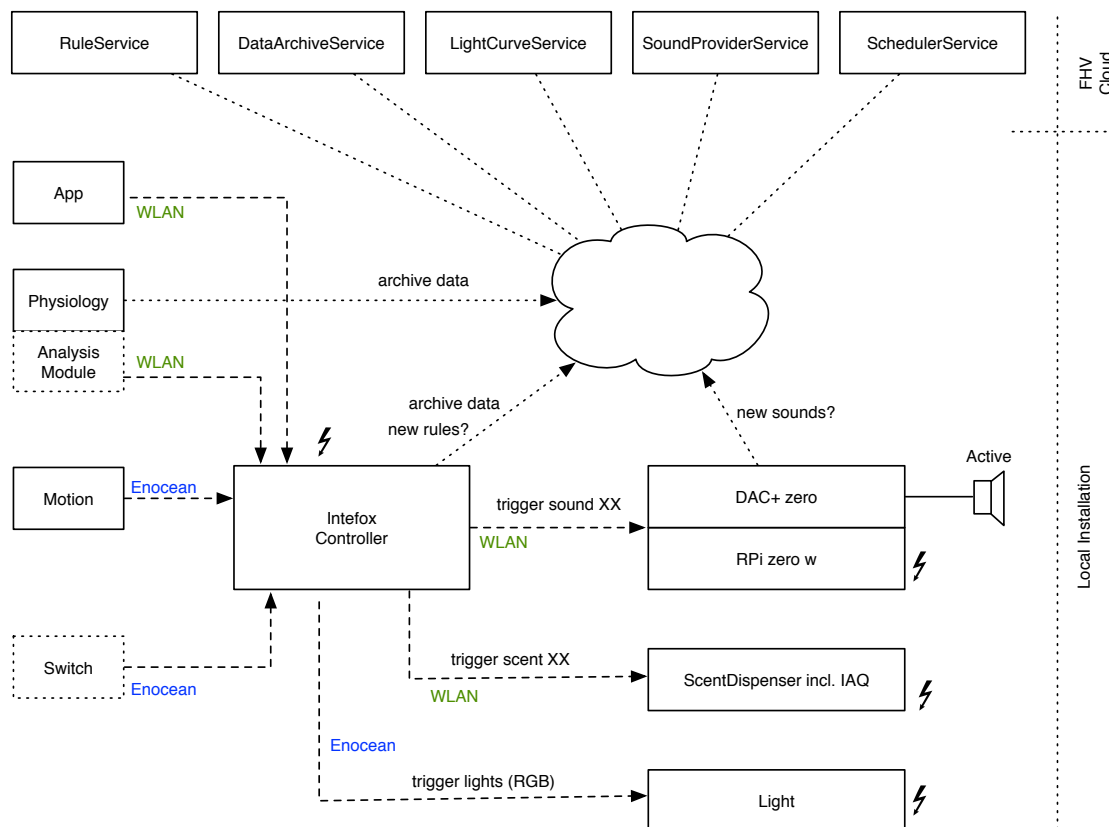


Figure 2: GREAT Distributed System Overview, Source: GREAT consortium.

The main component of the GREAT system is the local controller, that acts as a coordinator among the different other components. It runs an already available middleware solution for smart buildings control (provided by Intefox) with built-in support for a wide range of common building automation protocols (e.g. DALI, EnOcean, KNX,...). The middleware system is also highly extensible to allow for integration of new components, either based on standard protocols or application specific ones.

Multiple GREAT systems can be used at the same time in multiple locations, each of them can be tailored to local requirements.

The controller and the modules are connected over wireless links (EnOcean, WLAN, optionally Bluetooth LE). A WLAN is provided by the GREAT controller itself to allow for efficient connection of components. For internet connection, an Ethernet port with

publicly accessible Internet is required. Alternatively, also a USB WLAN adapter can be used to connect the system to an existing WLAN.

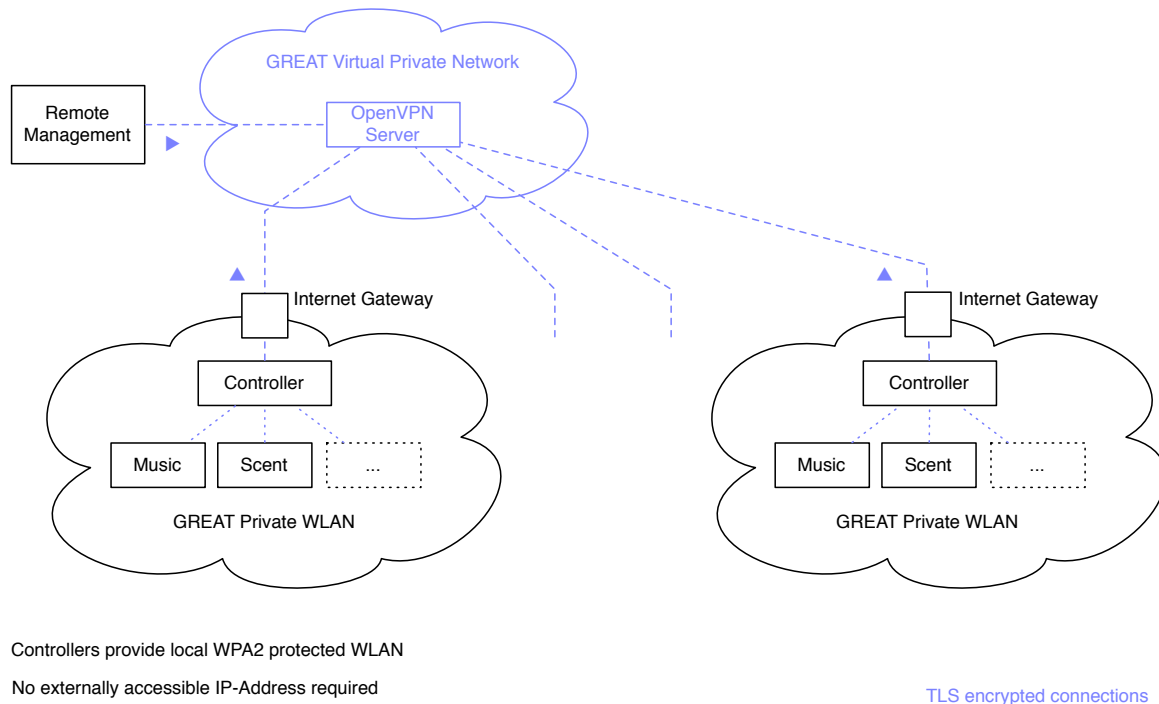


Figure 3: Distributed GREAT installations connected over VPN.

All communication from the controller to the Internet is encrypted. To allow for remote administration of the system, individual controllers are connected into a virtual private network, thus not requiring an externally accessible IP-address (see Figure 3).

2. Controller

The software stack of the GREAT controller is based on the open source Raspbian Jessie Linux distribution for Raspberry PI single board computers. The three main software modules of the controller are:

- Intefox Middleware stack for building automation (fox.core)
- Wireless Access Point functionality including routing
- OpenVPN Client for remote management

2.1. Intefox Middleware

The main purpose of the middleware is to hide the details of the individual component communications und provide unified access to individual objects/components. In this way, e.g. additional lights could be easily added into the GREAT system, even if they used some different building automation standard

like KNX or DALI. At the GREAT software layer, they would be treated the same. In addition to this hardware abstraction layer it also provides a broad list of features that allow for rapid prototyping (drag & drop configurations), ready-made mobile apps, and event logging functionalities.

The basic architecture of the Intefox middleware software is based on an OSGi (originally Open Services Gateway initiative) software layer (see Figure 4).

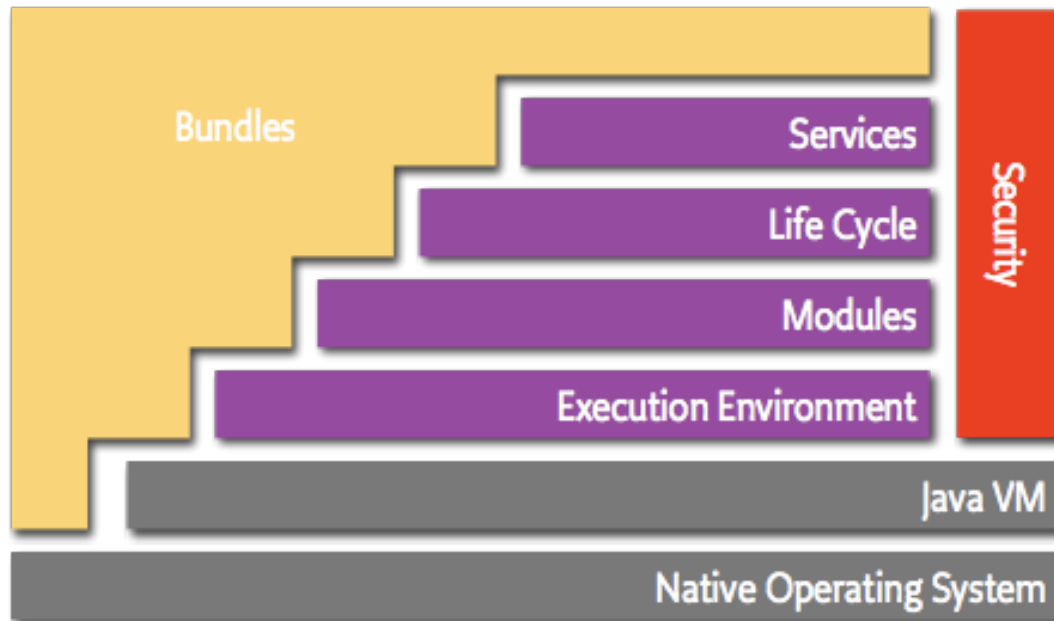


Figure 4: OSGi Architecture Diagram, Source: OSGi Alliance, <https://www.osgi.org/developer/architecture>

By using the OSGi dynamic component architecture, the functionality of the controller-software can be extended even at runtime. The Intefox system allows for easy extension of the system by means of so-called bundles. These bundles typically feature inputs, where they listen for incoming events, and outputs, where they can send events. So, for example in GREAT, a new bundle for the sound component was created, that manages encrypted communication with the sound component. It provides a set of inputs that can be used for controlling the sound module, and a set of outputs that provide other components with information (e.g. the status of the sound component).

In the GREAT setup, the Intefox middleware software runs on a Raspbian Linux operating system, but could be run on other Linux-/macOS- or Windows-based systems too, if a Java Runtime environment is available.

The configuration of the controller is edited using a graphical configurator software based on the open source Eclipse rich client platform (RCP) that is available for multiple platforms (e.g. Linux, macOS, or Windows) (see Figure 5). Connections between inputs and outputs of modules can be made via drag & drop.

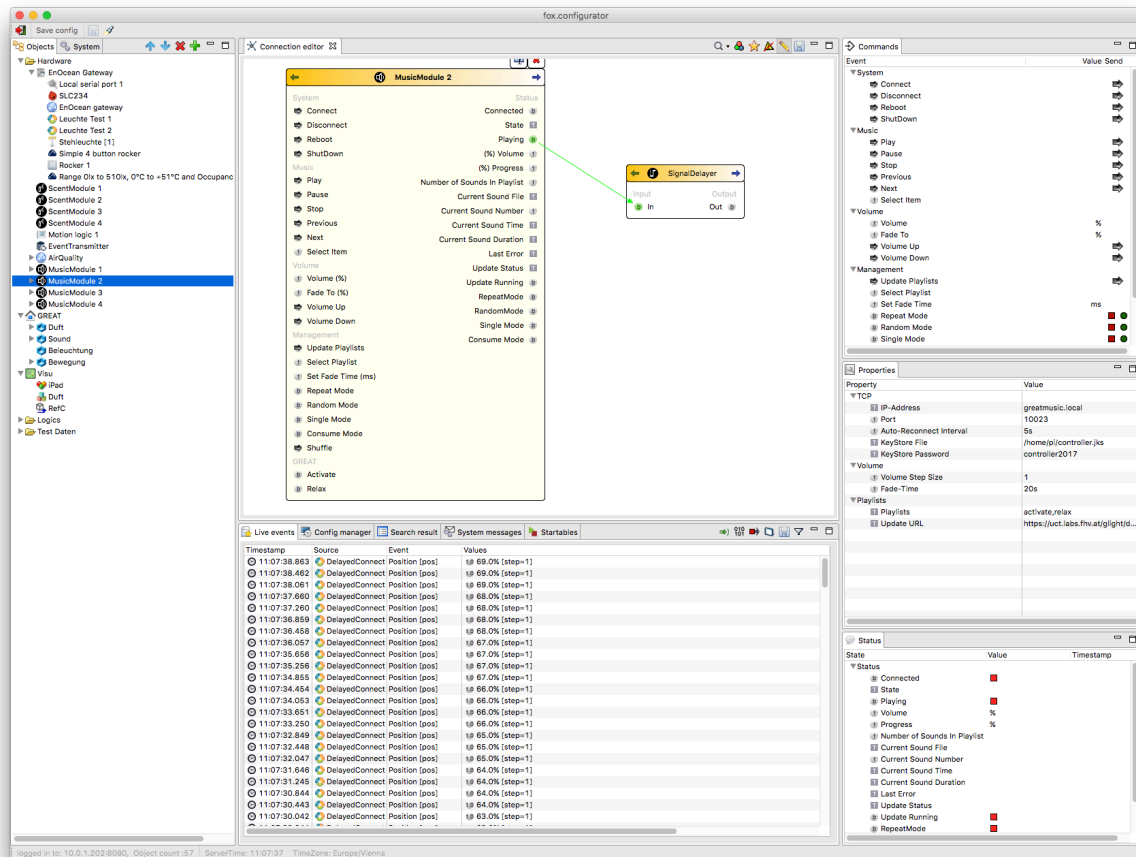


Figure 5: Screenshot of the Intefox configuration software showing connections between individual elements.

See the chapter Intefox - Middleware for a more detailed description of relevant components of the middleware system.

2.2. Wireless Access Point, Router

Each GREAT controller provides its own GREAT wireless network, to allow for easy connection of WLAN based components or mobile app based remote controls, independent of the local availability of a WLAN. The GREAT WLAN is an encrypted WPA2 network for security reasons.

The wireless access point functionality provided by the GREAT controller is built using the open source packages **hostapd** and **dnsmasq**. These packages must be installed using apt-get, as they are not part of the standard installation of Raspbian.

The hostapd provides the functionality for the actual access point, while dnsmasq is a light weight DHCP and DNS server and therefore hands out IP addresses to connected clients.

In the GREAT setup, we use the built in WiFi interface wlan0 as basis for our access point. To avoid that this interface is being used otherwise, it needs to be denied in the /etc/dhcpd.conf above any other interface lines.

```
denyinterfaces wlan0
```

The actual interface definition then takes place in the `/etc/network/interfaces` file:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 172.24.1.1
    netmask 255.255.255.0
    network 172.24.1.0
    broadcast 172.24.1.255
```

This defines a static IP address of the access point interface of 172.24.1.1. In a next step the `hostapd` is configured to provide a GREAT wireless network. Configuration of this network happens in `/etc/hostapd/hostapd.conf`

The typical config for GREAT looks like

```
# This is the name of the WiFi interface we configured above
interface=wlan0

# Use the nl80211 driver with the brcmfmac driver
driver=nl80211

# This is the name of the network
ssid=GREAT

# Use the 2.4GHz band
hw_mode=g

# Use channel 6
channel=6

# Enable 802.11n
ieee80211n=1

# Enable WMM
wmm_enabled=1

# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]

# Accept all MAC addresses
macaddr_acl=0

# Use WPA authentication
auth_algs=1

# Require clients to know the network name
ignore_broadcast_ssid=0

# Use WPA2
wpa=2

# Use a pre-shared key
wpa_key_mgmt=WPA-PSK

# The network passphrase
wpa_passphrase=*****

# Use AES, instead of TKIP
rsn_pairwise=CCMP
```


This basically sets up a WPA2 wireless network on the 2.4 GHz band with a network SSID of GREAT.

In the `/etc/dnsmasq.conf` file the details of the DHCP part of dnsmasq are configured. Specifically the DHCP range, name servers, interfaces and listen addresses are defined here. The specific settings used in GREAT are:

```
domain-needed
bogus-priv
server=8.8.8.8
interface=wlan0
listen-address=172.24.1.1
bind-interfaces
dhcp-range=172.24.1.50,172.24.1.150,12h
```

as well as mappings of hostnames for up to 5 sound- and scent-components.

```
dhcp-host=greatmusic,172.24.1.2,infinite
dhcp-host=greatscent,172.24.1.3,infinite
dhcp-host=greatmusic2,172.24.1.4,infinite
dhcp-host=greatscent2,172.24.1.5,infinite
dhcp-host=greatmusic3,172.24.1.6,infinite
dhcp-host=greatscent3,172.24.1.7,infinite
dhcp-host=greatmusic4,172.24.1.8,infinite
dhcp-host=greatscent4,172.24.1.9,infinite
dhcp-host=greatmusic5,172.24.1.10,infinite
dhcp-host=greatscent5,172.24.1.11,infinite
...
```

This allows for creating port forwarding rules to specific components for remote management tasks.

A final configuration step involves the iptables routing software. Here traffic from the `eth0` interface is forwarded to the `wlan0` interface. The routing information is loaded from a persistence file in the `rc.local` phase.

Excerpt of the iptables-save persisted file that is loaded on startup:

```
# Generated by iptables-save
*filter
:INPUT ACCEPT [4823:503443]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [3401:1416793]
-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i wlan0 -o eth0 -j ACCEPT
COMMIT
# Completed on Fri Aug 18 09:09:28 2017
# Generated by iptables-save v1.4.21 on Fri Aug 18 09:09:28 2017
```

```

*nat
:PREROUTING ACCEPT [19:2901]
:INPUT ACCEPT [16:2709]
:OUTPUT ACCEPT [4:316]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -m tcp --dport 33101 -j DNAT --to-destination 172.24.1.2:22
-A PREROUTING -p tcp -m tcp --dport 33102 -j DNAT --to-destination 172.24.1.3:22
-A PREROUTING -p tcp -m tcp --dport 33103 -j DNAT --to-destination 172.24.1.4:22
-A PREROUTING -p tcp -m tcp --dport 33104 -j DNAT --to-destination 172.24.1.5:22
-A PREROUTING -p tcp -m tcp --dport 33105 -j DNAT --to-destination 172.24.1.6:22
-A PREROUTING -p tcp -m tcp --dport 33106 -j DNAT --to-destination 172.24.1.7:22
-A PREROUTING -p tcp -m tcp --dport 33107 -j DNAT --to-destination 172.24.1.8:22
-A PREROUTING -p tcp -m tcp --dport 33108 -j DNAT --to-destination 172.24.1.9:22
-A PREROUTING -p tcp -m tcp --dport 33109 -j DNAT --to-destination 172.24.1.10:22
-A PREROUTING -p tcp -m tcp --dport 33110 -j DNAT --to-destination 172.24.1.11:22
-A POSTROUTING -o eth0 -j MASQUERADE
-A POSTROUTING -p tcp -m tcp --dport 22 -j MASQUERADE
COMMIT
# Completed

```

These rules also include network address translation to allow for direct reachability of the sound- and scent components over SSH for remote maintenance. Note that this will not be part of the final GREAT product – there, only the main controller will be directly accessible from the maintenance VPN for reducing potential for security issues.

Since in some situations there are no wired network connections available, the GREAT controller also supports connecting to an existing WLAN via a USB WLAN stick mounted as interface wlan1. When a wlan1 interface is becoming available, a wlan1_up script adds rules to iptables to route from wlan1 to wlan0 and vice versa. When the stick is removed and the wlan1 interfaces is down, a wlan1_down script, removes the rules dynamically again.

Connections via WLAN tend to be prone to disconnecting. Therefore, a script checks every 5 minutes via cron, if the WLAN connection is still functional and if not, tries re-establishes the connection to the interface.

```

WLAN=wlan1
wlanExists=`ifconfig | grep ${WLAN}`
if [ $? -eq 0 ]
then
    router=`ip route | awk '/default/ {print $3;exit;}'`
    ping -I ${WLAN} -c2 $router > /dev/null
    #echo $router

    if [ $? != 0 ]
    then
        ifdown --force ${WLAN}
        /bin/kill -9 `pidof wpa_supplicant`
        ifup --force ${WLAN}
    fi
fi

```

The onsite WLAN credentials can be easily set using a wpa_supplicant.conf file, that is placed into the root of the boot partition of the Raspberry PI microSD Card. The system then moves this file into the proper place automatically (/etc/wpa_supplicant/) and uses these credentials. Typically, this text file includes the information:

```
country=AT
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="networkName"
    psk="networkPassword"
    key_mgmt=WPA-PSK
}
```

Instead of directly supplying a password, also the actual key can be supplied. This key for a given SSID can be generated using the `wpa_passphrase` command on the Raspberry.

2.3. OpenVPN Client

Since the GREAT prototypes will be used in various places in Austria, Italy and Switzerland, it's important to be able to update the systems via a remote connection. However, often it's not possible to get an externally reachable IP address at the institutions. To avoid the need of an externally reachable IP address, the GREAT controller connects itself into a remote management virtual private network hosted by the FHV (see Figure 3) automatically when a network connection becomes available. On the client, this is achieved by installing the `openvpn`-package and passing a client specific configuration (typically an `ovpn`-file, but this needs to be renamed to `conf`) to the service. It is then enabled by calling for example:

```
sudo systemctl enable openvpn@clientConfig1
```

The virtual private network is implemented using the open source VPN server OpenVPN. It is configured to apply TLS encryption to connections. Each GREAT controller and maintainer computer have their own certificates. Only clients with a valid certificate can connect to this network. The VPN network is configured to use TCP Port 443 for communication to avoid firewall issues on location.

In case of a remote management task, the maintainer connects a computer to the VPN and can then access GREAT controllers using their hostnames via SSH, as long as the controllers are connected to the Internet. The main controllers listen for SSH connections on port 33100. Submodules can then either be reached via SSH connections originating from the controller, or via port forwarding on the controller directly from the maintainer's computer.

3. Intefox - Middleware

3.1. Architecture and Components (fox.core)

The fox.core server runs on a Java runtime and is using the OSGi standard to load and unload bundles during runtime and provides RESTful interfaces for configuration, control and visualization clients.

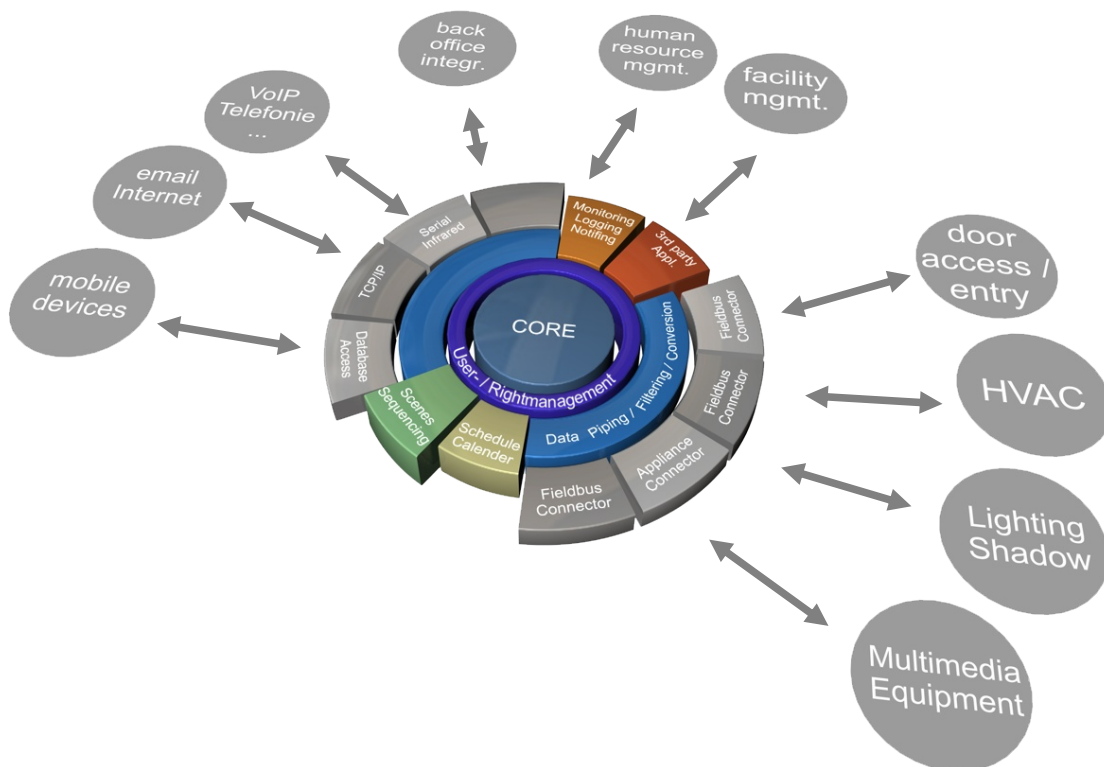


Figure 6: Basic architecture of the foxcore server

Figure 6 shows the basic architecture of the fox.core server, illustrating the layered model to provide abstraction for different technologies and interfaces.

3.2. Configurator Software

The fox.configurator is used to configure and manage fox.core servers. The connection is established via TCP/IP and can therefore be used to either connect locally or remotely. Figure 7 shows the example of inserting a new light object into the system configuration.

Basic features:

- Managing bundles and updates
- Creating and managing objects
- Bundle activation (licensing)
- Managing configurations
- Server diagnostics
- User management
- Timer and Schedulers management
- Managing Cloud services (AutoBackup, Database, PushNotification, Alexa, etc)
- Enabling control and visulization interface
- Commissioning
- Live events

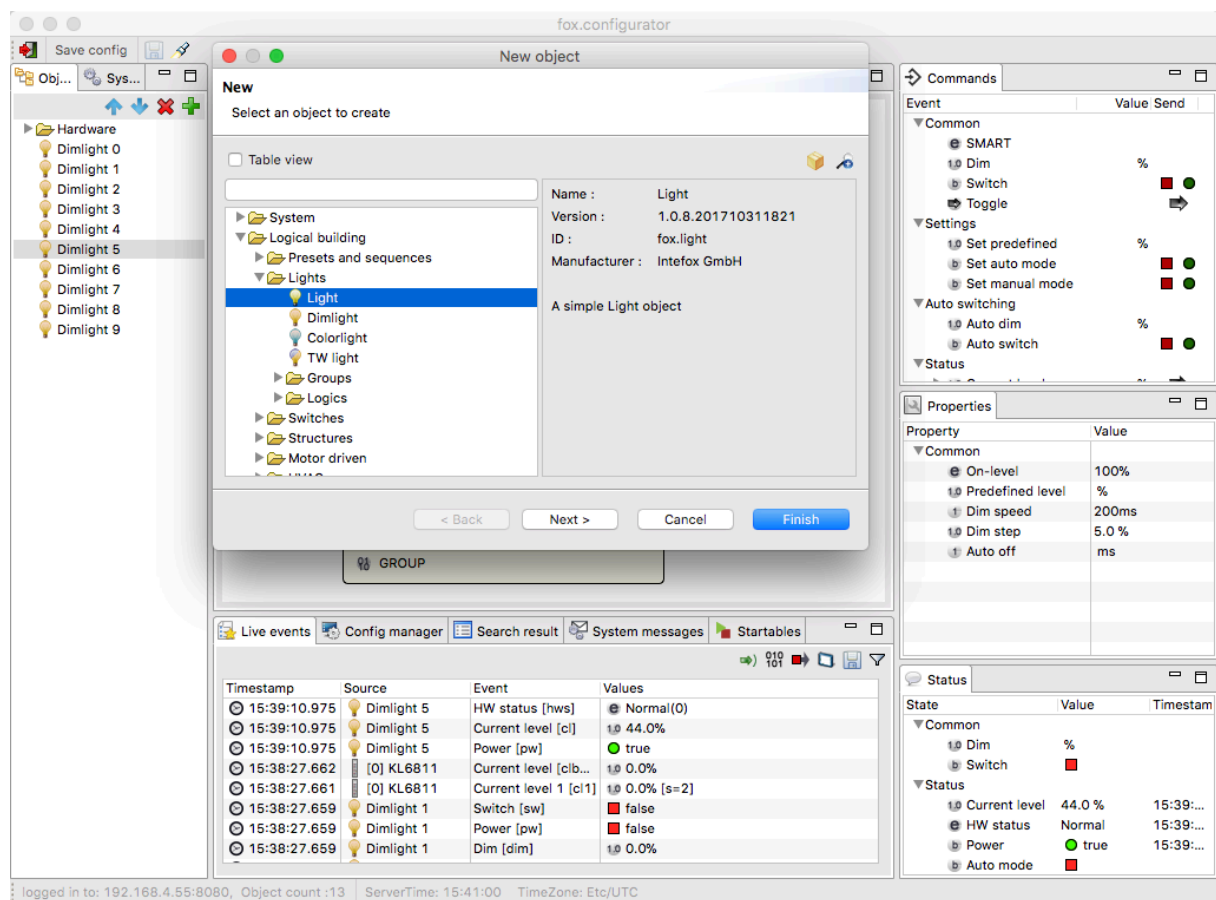


Figure 7: fox.configurator, example of adding a new light object

3.2.1. Online Bundle Manager

The online bundle manager is fully integrated into the configurator software.

The bundles are managed through an online sharing platform where it allows a software developer to upload and share the bundle to others. In that way, it makes it very easy for everyone to install and update bundles on the controller. Figure 8 shows the bundle selection screen to extend the functionality of the core system.

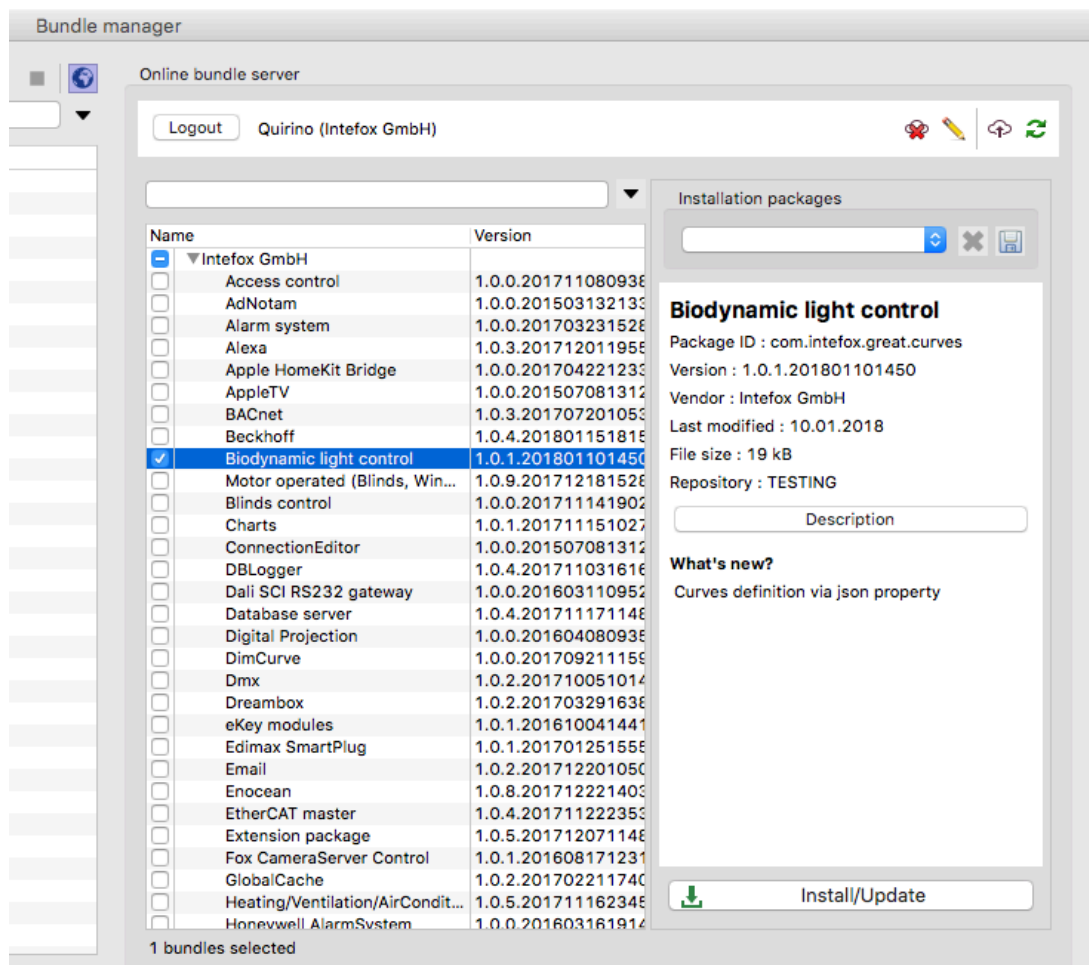


Figure 8: Online bundle manager

3.3. Logging and Data Access Interface

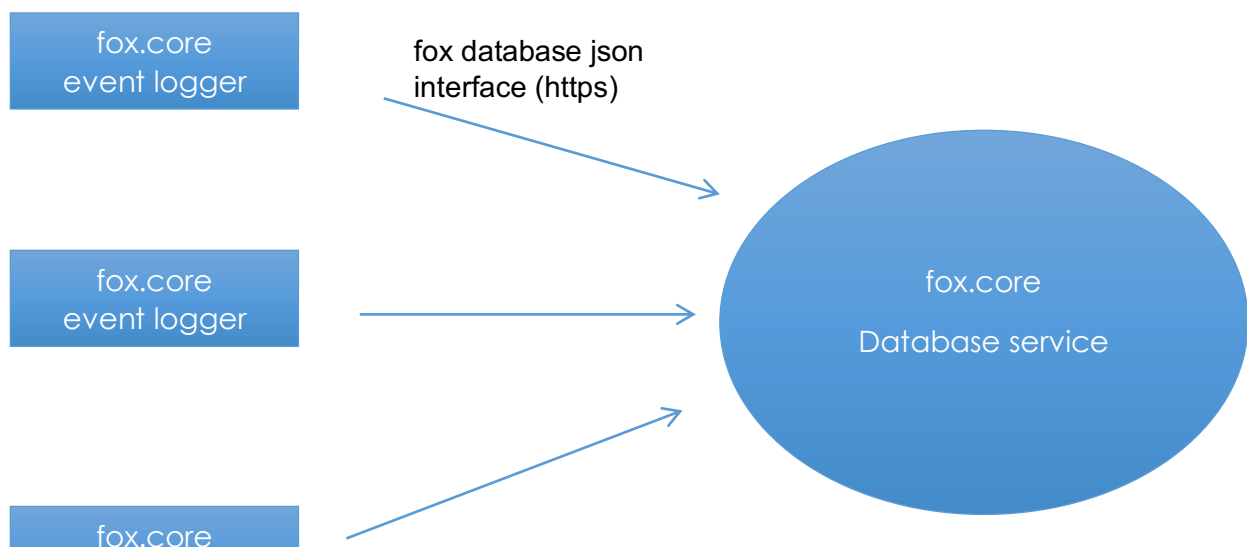


Figure 9: Basic structure of the event logging architecture

Figure 9 illustrates the basic event logging architecture that consists of a database service backend, as well as event logger objects.

Database Service

The database service is available as a fox.core bundle and is designed to either run within the same local network or some hosted server on a fox.core based system.

Currently PostgreSQL is supported as database backend. Further database types might be added, as the need for it arises though.

Event Logger

An event logger processes defined events and sends them to the database service. It also takes care of buffering events locally, if the remote database service is not available at the time. It is available as a fox.core bundle and is designed to run on a fox.core based system.

Multiple instances are allowed on the same controller as well as on different controllers to send the data to the same database service.

Through the event logger it is also possible to compare the logged values from a local controller with values from other controllers live in charts views and reports if they are being logged to the same database.

3.3.1. Event Logger Configuration

Creating the Event logger

First, an event logger object needs to be created. Therefore, select the 'Database loggers' container within the System tree and select 'Add object'. Figure 10 and Figure 11 illustrate the process.

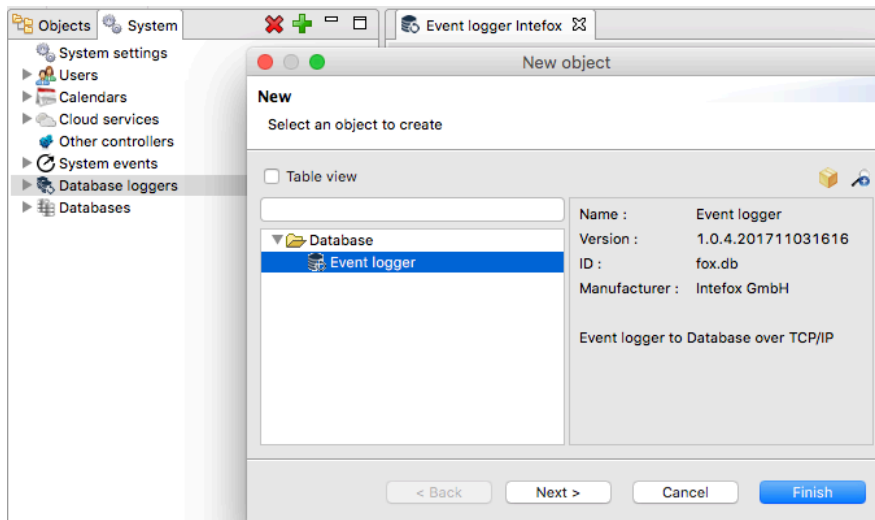


Figure 10: Select the created event logger and edit the properties (URL and Context)

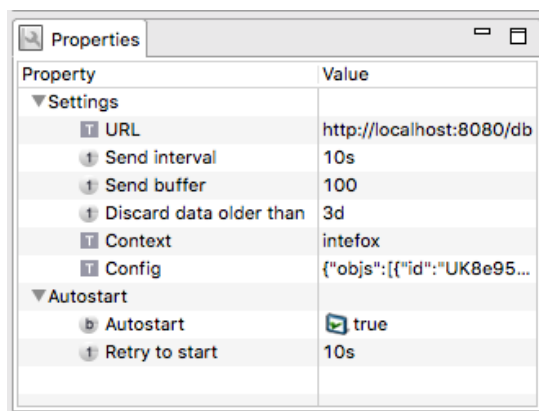


Figure 11: Properties of the event logger service

URL:

If the database runs on the same controller, the URL would be like this:

<https://localhost:8080/db>

Context:

the database name set in the database service configuration. (e.g. myDBname)

Send interval:

Defines the minimum interval to be used to send the queued data to the database service.

Send Buffer:

Defines the buffer size for queueing the data before sending to the database service.

If the queue is full, the data will be sent immediately.

Discard data older than

Defines, how long the data will be kept in the queue if the data could not be sent due to any communication error with the database service.

The event logger uses the interface (described in section 'interface') for the data exchange.

3.3.2. Log settings (Selecting the events to be logged)

With the event logger configuration editor (by double clicking the 'event logger object') the events can be selected to be logged.

There are 2 ways to select an event:

- Type: all objects of this type will be logged with the defined settings
- Object: single objects will be logged with the defined settings

Figure 12 shows the configuration of an eventlogger to log the temperature output of all temperature sensors in the system, even if they might be added at a later point in time.

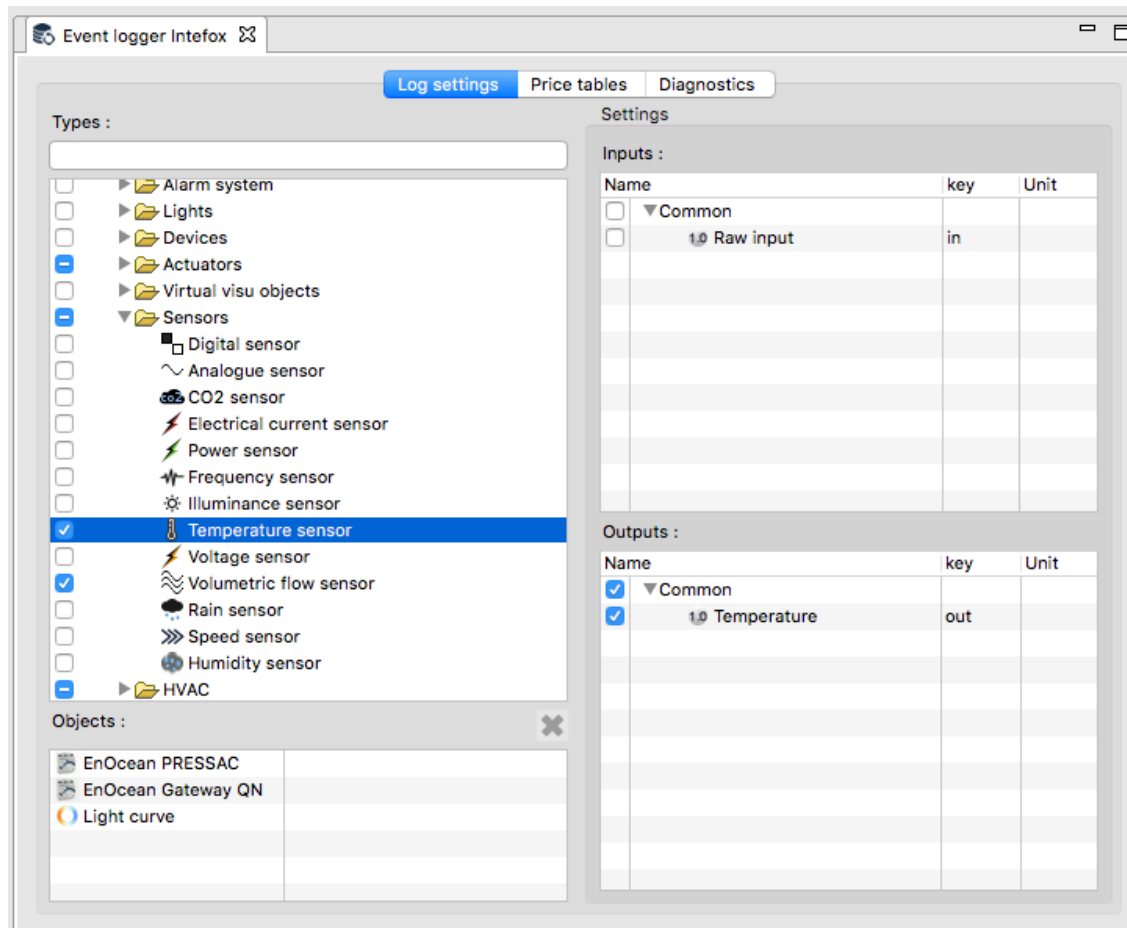


Figure 12: In this example, the 'Temperature' output of all 'Temperature sensors' will be logged, even if they are being created later.

3.3.3. Data Access Interface

The database service offers a REST-based interface to store and retrieve data as well as to read the logging configuration. Parameters are encoded using json format.

format: <https://localhost:8080/db/<cmd>?params...>

cmd: data

writes Event data to the Database

the data is being sent as post data in json format

Description with example data:

<https://localhost:8080/db/data>

< POST DATA AS JSON>

cmd: readconfig

retrieves the current foxEvents table with objectIds in json format

parameters:

id	database id
coreid	if not set, ALL events of all cores will be returned
statistics	includes the current amount of data logged of each event
space	includes the space being used by the DB and also the free space

cmd: getdata

request data in json format

parameter:

id	database id
eid	comma separated id's of foxevents (known when previously read from config) (or)
objId	id of FoxNode object
key	the in- or output key depending on dirout
dirout	0 = input key, 1 = output key

from epoch format (milliseconds since 1.1.1970)

to epoch format (milliseconds since 1.1.1970)

dur duration in milliseconds (note: use either 'to' or 'dur')

(or)

range -->possible values: **keyword**[,back,count]

(back = keyword units back, count = range in keyword units,

e.g. month,6,3 = starting from 6 month's back with a range of 3 months

Possible keywords:

thishour

lasthour

today

yesterday

thisweek

lastweek

thismonth

lastmonth

thisyear

lastyear

Extended range-format:

<type-unit>,<starting point relative to now in type units>,<length in type-units>

hour,back,count

day,back,count

week,back,count

month,back,count

year,back,count

(see examples below)

maxpoints the maximum number of points returned per eventId. Default value = 100

debug if debug parameter is present, the returned values will include additional information of

the requested eventId's

units a string based unit map

➔ example value: K=°C,m/s=km/h

- will return all temperature values as '°C' and all velocity values as 'km/h'

unituser the id of a user object. If the user exists, the unit map will be taken from the user properties

rangeinfo if rangeinfo parameter is present, the returned values will include the requested 'from' and 'to' timestamp. This is helpful if the request is done by a range keyword

examples:

<https://localhost:8080/db/getdata?id=qnhome&eid=3&range=lastweek&debug>

--> returns a json with data of object with eventId=3 of last week, also includes additional debug informations

<https://localhost:8080/db/getdata?id=qnhome&eid=3,5&range=month,6,3>

--> return a json with data of objects with eventId = 3 and 5 from 6 month back with a range of 3 months

<https://localhost:8080/db/getdata?id=qnhome&objId=7jgbd7kx&key=out&dirout=1&from=149898498000&duration=360000>

type line (default), sum, day, week, month, year

type sum:

retrieves the total value only plus price if a pricetable is assigned

example: (analog values)

sum (value per hour): 345 kWh

price (through price table) 123,23 EUR

example: (digital values)

count (impulses), true count (boolean)

type day:

retrieves the total value of 1 day and 24 hour values

type week

retrieves the total value of 1 week and 7 day values

type month

retrieves the total value of 1 month and 28-31 day values

type year

retrieves the total value of 1 year and 12 month values

3.3.4. Database Structure

The fox.core database keeps track of existing configurations and state changes of registered fox.core-systems. The table structure outlined below is optimized for efficient access to event values even in large installations.

The foxcores table keeps basic information about controllers like their name or unique id (see Table 1 for details).

The foxobjects table links individual objects to a specific core object and keeps path information of the object tree and type.

The foxevents table links events to the originating objects and keeps information about event-names, ids and value types.

The vals<eventID> table keep track of concrete value changes of specific events (see Table 2). The value storage is splitted up into individual tables for each event for more efficient access to specific event-values in large installations. For analog value types, vals-tables are also generated on the fly for aggregation intervals of 1 minute / 15 minutes / 4 hours / 1 day (see Table 3).

The foxwatch table keeps information about monitoring conditions and valid range checks for individual foxevents. See Figure 13 for an overview of the relations among the tables.

Table 1: Configuration tables

foxcores	Type	Description
id	integer	primary key
coreld	varchar	fox.core id, bound to the configuration on the controller
name	varchar	the given name of the fox.core configuration
host	varchar	the local published ip of the fox.core configuration

foxevents	Type	Description
id	integer	primary key
oid	integer	the object id, referred to 'foxobjects' table
dirout	boolean	event direction: true = output key, false = input key
key	varchar	the object key
type	integer	0 = trigger, 1 = boolean, 2 = integer, 3 = double, 4 = text, 5 = enumeration (integer), 6 = bytes, 7 = long, 8 = float, 9 = tristate (integer, 0=false, 1=true, 2=undefined)
unit	varchar	the unit of the event (null = no unit)

agg_ts	bigint	the timestamp of the last aggregated values
removed	boolean	true = the event has been deleted
wid	integer	Event watchdog, referred to the 'foxwatch' table
pid	integer	assigned pricetable, referred to the 'pricetables' table
first_ts	bigint	timestamp of the oldest entry
last_ts	bigint	timestamp of the newest entry
count	integer	total count

foxobjects	Type	Description
id	integer	primary key
objid	varchar	object id of the object in the configuration of the controller
name	varchar	the name of the object in the fox.core configuraton
objtype	varchar	the type of the object in the fox.core configuraton
cid	integer	the fox.core id (controller), referred to 'foxcores' table
pathids	varchar	the object id's of the full path, including the path position
path	varchar	the path names of the full path (URL encoded)
pathtypes	varchar	the object types of the full path
pos	integer	the position of the object in the fox.core configuraton tree

foxwatch	Type	Description
id	integer	primary key
name	varchar	the given name of the event watch definition
upperlimit	double	the upper limit of the value
lowerlimit	double	the lower limit of the value
maxinterval	integer	the minimum time in seconds within a value must be received

Table 2: RAW value tables

vals<eventid>	Type	Description
id	integer	primary key
ts	bigint	the timestamp in milliseconds (epoch format)
val	double	the value in the unit as defined in the 'foxevents' table

The raw value tables hold all values received from events.

To better deal with potentially big amounts of data, each event description will create its own table with the name 'vals' followed by the event id.

Table 3: Aggregated value tables

vals<eventid>_1min		
vals<eventid>_15min		
vals<eventid>_4hr		
vals<eventid>_1day		
id	integer	primary key
ts	bigint	the timestamp of the aggregated value in ms
vph	double	the value per hour within the duration (dur)
dur	bigint	the duration of the calculated value (vph)

Aggregated value tables are being created to speed up the longterm requests. Values will be calculated and filled in in realtime after a raw event packet has been received.

Additionally, it is possible, to recalculate the aggregated values from the RAW value table. (this is necessary for instance after data has been imported)

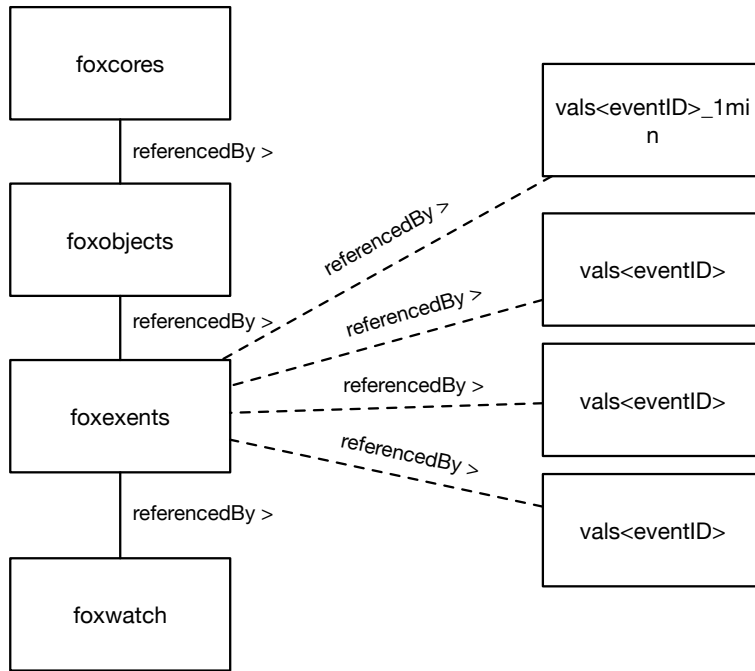


Figure 13: Entity relationship diagram for the logging data backend

3.4. Controller & Visualization interface

The fox.core system provides an HTTP based interface for connecting visualization clients to the system so they can control certain features. The following chapters describe this protocol in detail. For concrete example of a visualization client that uses this protocol, see the chapter on the GREAT User-Interface.

3.4.1. LiveCycle of a Visualization Client

The typical initialization stages of a visualization client are:

- 1) Login (retrieves last version id to check if a reload of descriptions is necessary)
- 2) getDescriptions
- 3) getStructure (mixed with requested Style)
- 4) poll (receive Status updates)
- 5) Send commands asynchronously until the client is terminated

HTTP request, format:

`https://host:port/json/session_id/key`

The return format is JSON

3.4.2. Login

key: login

Table 4: Login Parameters

user	the user name for the visu user
pwd	the password for the visu user note: the login procedure will be changed in future to a more secure method
style	optional: customized style properties for each single object
appid	optional – see PushNotification support
devtoken	optional – see PushNotification support
id	optional – see PushNotification support



Figure 14: Example login request

3.4.3. Request Object Descriptions

key: desc

Retrieves a description of each object type which is represented in the current visu configuration for the user logged in. Additionally, the object type and the current values of each single object will be added at the end.

The JSON reply is splitted in 2 parts:

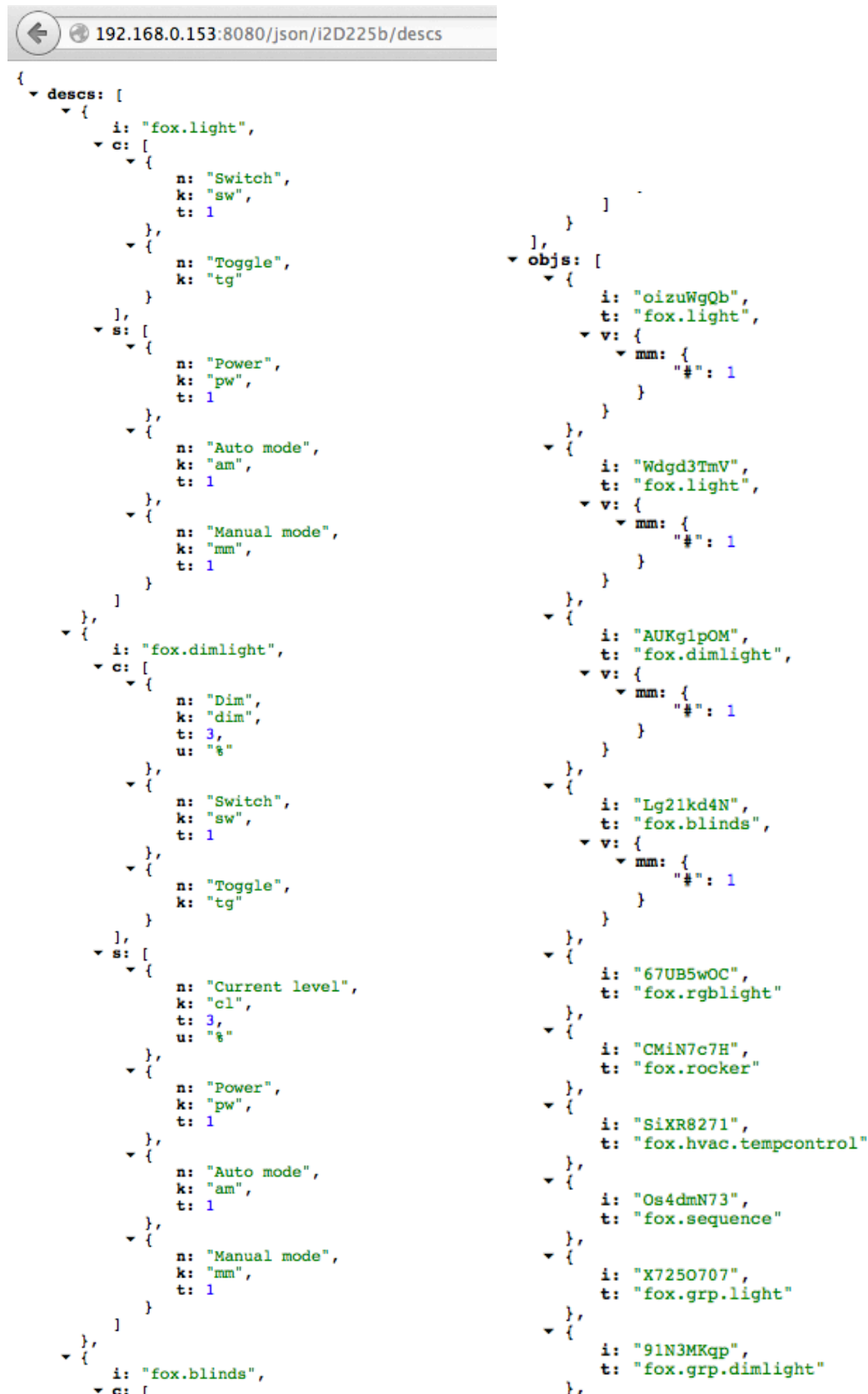
- desc
 - o i id of object type (e.g. fox.light, fox.blinds)
 - o c commands to be send to the core
 - o s status to be received from the core
 - n display name of the command or status
 - k key of the command or status to be used for identification
 - t value type (see table below)
 - u unit of each value (e.g. %, ms, °C)
- obj
 - o i id of the object (unique id per object)
 - o t object type (this value refers to the id (i) in desc (see Table 5))
 - o v current values of the object (see Table 6)

Table 5: Request object types (t)

none	Trigger (no value)
1	Boolean (0 = false, 1 = true)
2	Integer (signed value, uses internaly 4 bytes)
3	Double (floating point value, uses internally 8 bytes)
4	String (Text)
5	Enumeration (Integer value)
6	Bytes (starting with 0x, example: 0xa1bb4c83)
7	Long (signed value, uses internally 8 bytes)
8	Float (floating point value, uses internally 4 bytes)
9	Tristate (0 = false, 1 = true, 2 – undefined)

Table 6: Request values (v)

#	The ,#' key refers to the main value of the command or status
<key>	If a command or status has more values, sub values are identicated by a key word



```

{
  "descs": [
    {
      "i": "fox.light",
      "c": [
        {
          "n": "Switch",
          "k": "sw",
          "t": 1
        },
        {
          "n": "Toggle",
          "k": "tg"
        }
      ],
      "s": [
        {
          "n": "Power",
          "k": "pw",
          "t": 1
        },
        {
          "n": "Auto mode",
          "k": "am",
          "t": 1
        },
        {
          "n": "Manual mode",
          "k": "mm",
          "t": 1
        }
      ]
    },
    {
      "i": "fox.dimlight",
      "c": [
        {
          "n": "Dim",
          "k": "dim",
          "t": 3,
          "u": "g"
        },
        {
          "n": "Switch",
          "k": "sw",
          "t": 1
        },
        {
          "n": "Toggle",
          "k": "tg"
        }
      ],
      "s": [
        {
          "n": "Current level",
          "k": "cl",
          "t": 3,
          "u": "g"
        },
        {
          "n": "Power",
          "k": "pw",
          "t": 1
        },
        {
          "n": "Auto mode",
          "k": "am",
          "t": 1
        },
        {
          "n": "Manual mode",
          "k": "mm",
          "t": 1
        }
      ]
    },
    {
      "i": "fox.blinds",
      "c": [

```

```

],
      "s": [
        {
          "i": "oizuWgQb",
          "t": "fox.light",
          "v": {
            "mm": {
              "#": 1
            }
          }
        },
        {
          "i": "Wdgd3TmV",
          "t": "fox.light",
          "v": {
            "mm": {
              "#": 1
            }
          }
        },
        {
          "i": "AUKg1pOM",
          "t": "fox.dimlight",
          "v": {
            "mm": {
              "#": 1
            }
          }
        },
        {
          "i": "Lg21kd4N",
          "t": "fox.blinds",
          "v": {
            "mm": {
              "#": 1
            }
          }
        },
        {
          "i": "67UB5wOC",
          "t": "fox.rgblight"
        },
        {
          "i": "CMiN7c7H",
          "t": "fox.rocker"
        },
        {
          "i": "SiXR8271",
          "t": "fox.hvac.tempcontrol"
        },
        {
          "i": "Os4dmN73",
          "t": "fox.sequence"
        },
        {
          "i": "X7250707",
          "t": "fox.grp.light"
        },
        {
          "i": "91N3MKqp",
          "t": "fox.grp.dimlight"
        }
      ]
    }
  ]
}

```

Figure 15: Example description request

3.4.4. Request Structure

key: structure

No parameters.

Retrieves the hierarchical structure of the visualization configured for the user logged in.

JSON key description:

- contains the structure in hierarchical order
 - o p page id – a page can either be a ,page' object or a container object (room, floor etc).
 - o r page id – this is a ,reference container object' (building, floor, room)
 - which might contain additional pages
 - o c category id – used to manage style properties in categories
 - l listed objects in category
 - t category type
 - i short id, refers to ,c' properties in ,page' or container objects
 - o o contains additional objects or pages
 - o n display name for the object or page
 - o i object id to refer to in previous request ,objs'
 - o s style properties (customizable)

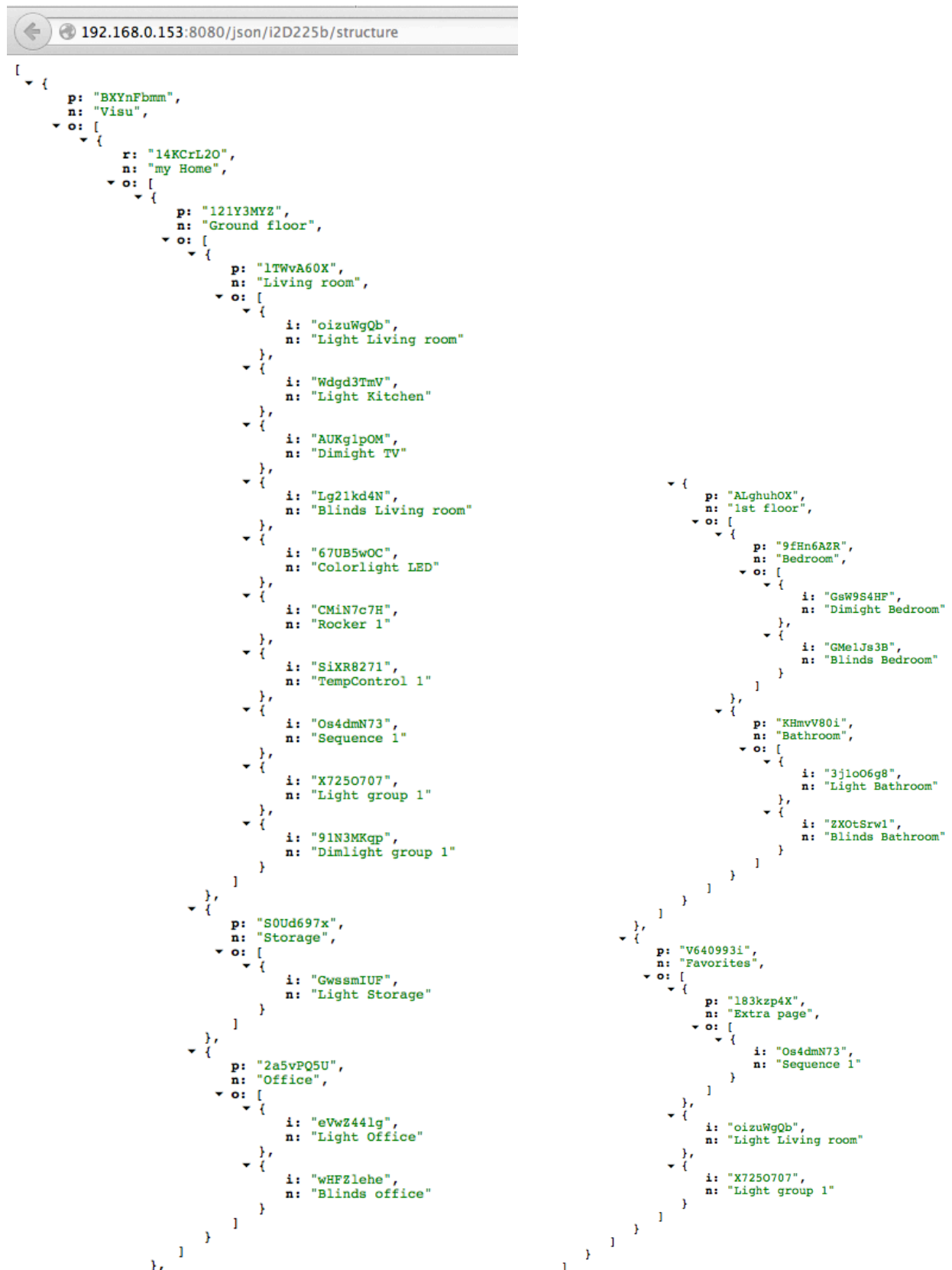


Figure 16: Example structure request

3.4.5. LongPoll Request

key: poll

No parameters

Retrieves value changes.

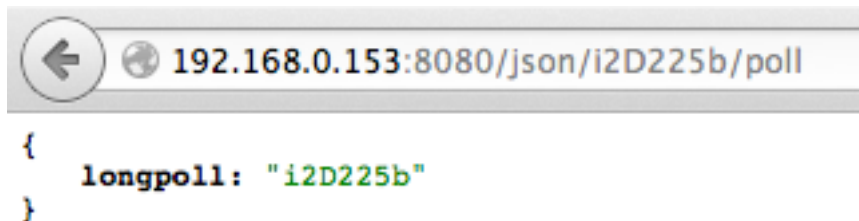


Figure 17: Example long poll request with no changes

If no value has changed, the core will respond after 10 seconds with a longpoll message (see Figure 17).

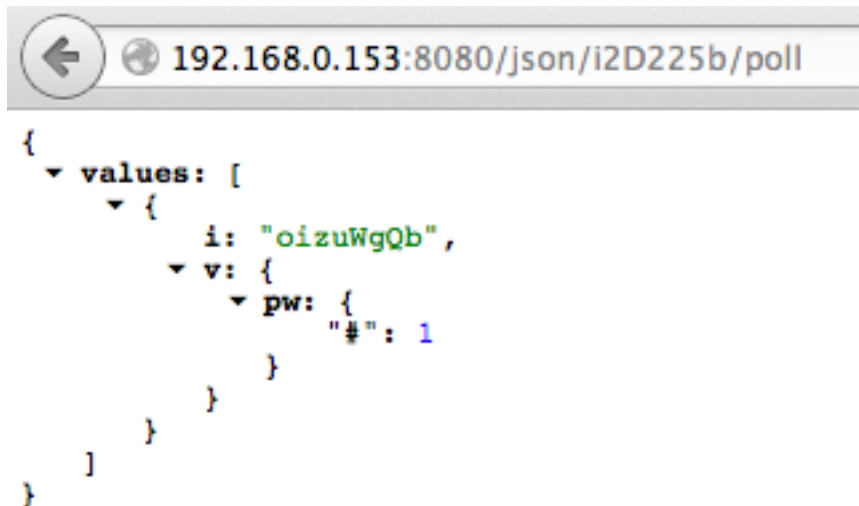


Figure 18: Example long poll request with changes

When a value got changed, the core will respond immediately with the changed values (see Figure 18)

JSON key description:

- values contains the values in order of occurrence
 - o i id of the object which has changed values
 - o v the value – contains the status key and it's new values

3.4.6. Send Commands

key: cmd

Table 7: Command (cmd) parameters

key	the key of the command to be send
objIds	the id(s) of the objects where the command shall execute if more than 1 id is used, the id's must be seperated by semicolon (;)
value	the new value . the value should be UTF-8 encoded
subKeys	if subValues are submitted, seperated by semicolon (;)
subValues	add subvalues in the same order as subKeys, seperated by semicolon(;) The values should be UTF-8 encoded



Figure 19: Example cmd request to switch two lights on

Figure 19 shows an example to switch on two lights.

4. Light

4.1. Biodynamic Light Extension Bundle

The biodynamic light objects regulate the light intensity and color temperature during a day period with the possibilities of different interventions (see Figure 20, Figure 21, Figure 22).

The light curve objects send values to 3 separate output channels to set the following values calculated by a predefined curve in json format:

- dim level (0-100%)
- Color temperature in Kelvin
- fadetime in seconds (0 - 180 sec)

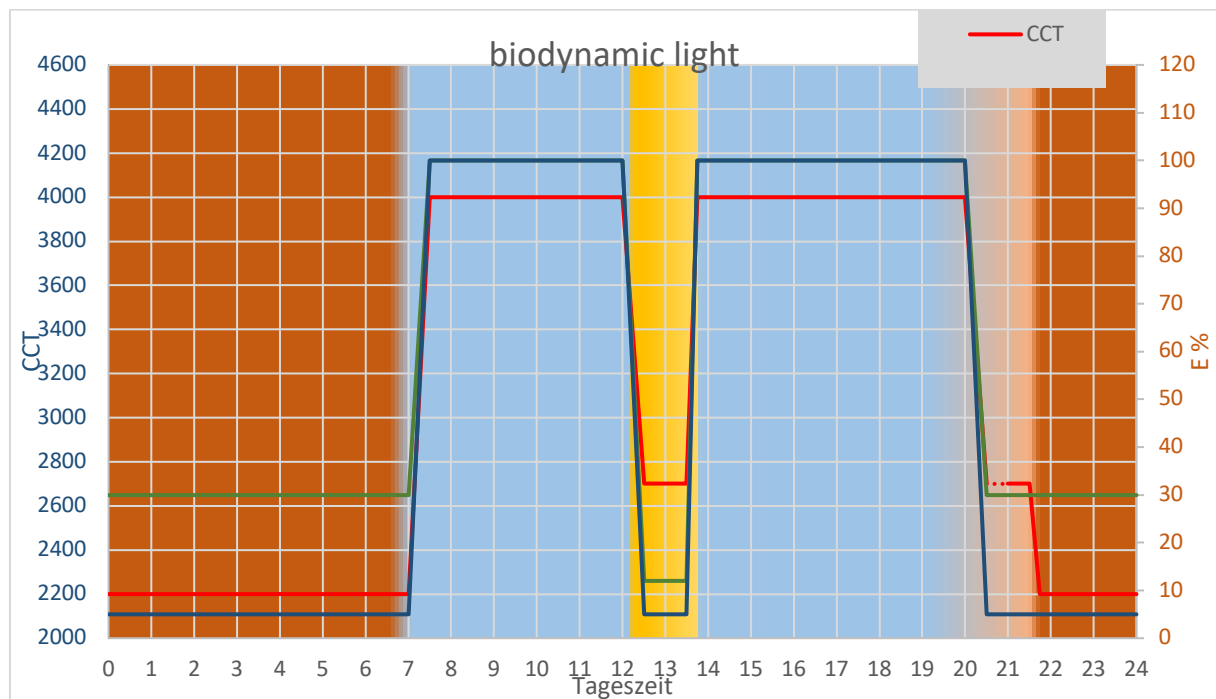


Figure 20: Example biodynamic light definition

Interventions:

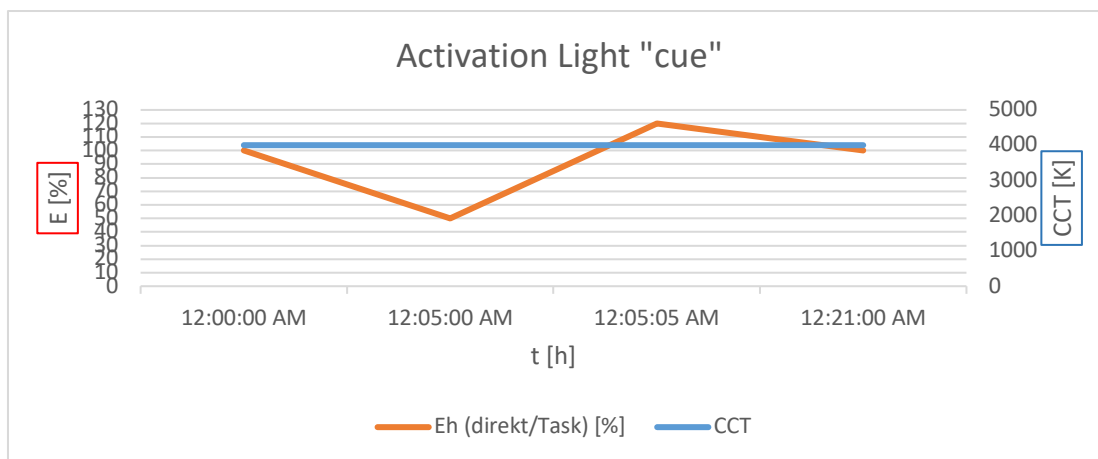


Figure 21: Example: Activation Light 'cue' definition

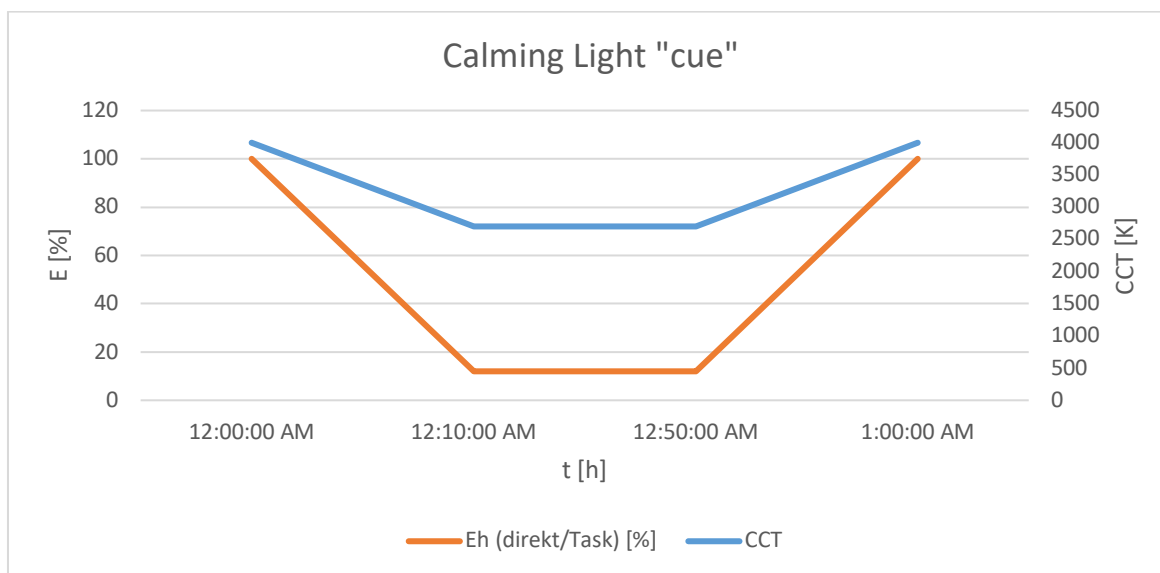


Figure 22: Example: Calming light 'cue' definition

4.1.1. Light Curve Object Description

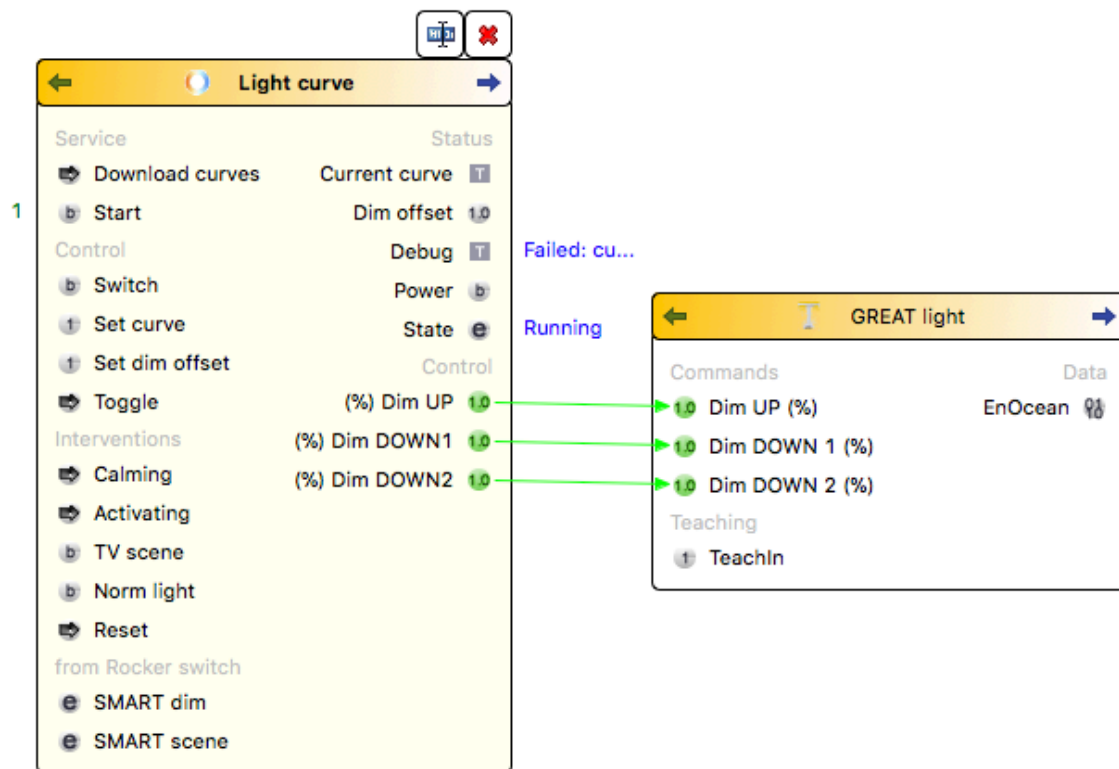


Figure 23: Configuration of the Light curve object in conjunction with the light device

Commands:

Service:

Download curves:

Downloads a new curve configuration from an external server

Start

Starts/Stops the service

Manual control:

Switch

To switch the light on/off manually (e.g. by an external switch)

Set curve

Sets the current curve to use for the calculation of the light values

Set dim offset (+/- 0-100%)

adds a dim offset to the current light values

Toggle

To switch the light on/off manually (e.g. by an external switch)

Send message

Sends a text message to the user clients.

id	some id. the id will be returned in the response
type	OK or YES/NO message
dur	sets the duration of the message to be shown
txtfalse	sets the text to be shown for the 'false' button
txttrue	sets the text for be shown for the 'true' button

Interventions:

Calming

Sets a calming intervention (1 hr)

Activating

Sets the activating interventions (20 min)

TV scene

Starts the TV scene (1 hr)

Norm light

Switches to the Norm light curve

Reset

Switches back to the biodynamic light control

from Rocker switch (Manual control via external switches)

SMART dim

Dims the light up/down

SMART scene

Calls predefined scenes

User interface (commands which are send directly by the client app as a response)

Send feedback

After an intervention has been completed, the user will be ask for a feedback. The feedback contains a number for good/bad and the possibility to enter an additional text.

Call intervention

Requests the server to start a predefined intervention. Possible values are:

Reset (switches back to biodynamic curve)

Activate, Relax, TV scene, Norm light, Off

Cancel (cancels an ongoing intervention. (on Activate and Relax only)

Send response

Sends a response to a previous asked question.

Possible values are: YES, NO, OK

Status:

Current curve

Shows the name of the current selected curve

Dim offset

Shows the current dim offset added to the current output values

Debug

Shows some detailed debug information

Power

Shows if the light is on or off

State

Shows if the service has started (Running, Stopped, Failed)

Control

3 independent control channels to set the value on the connected lights

Properties:

Config

curve in json format

Curve server url

The URL where the curve is being fetched from during automatic update

e.g.

<https://uct.labs.fhv.at/glight/greatcurves/getCurveData.php?key=greatDemo>

Curve server update

Off	Automatic updates are disabled
Every hour	Updates the curve every full hour
Midnight	Updates the curve every day at midnight

Simulation mode

Off	Normal operation
24h in 1 min	Simulates 24h in 1 min (1440 times faster)
24h in 5 min	Simulates 24h in 5 min (288 times faster)
24h in 30 min	Simulates 24h in 30 min (58 times faster)

Scale % max

defines the highest % value set within the curves config.

Example: in an 'Activating Intervention' it we like to overrule the regular 100% value and activate 120% for a short time. In that case, the output values will be scaled to 0-100% for all calculated values and therefore the regular 100% would be scaled to 83.3% to be sent to the hardware.

Send delay

Send delay in milliseconds between the cmds. when multiple commands are sent at the same time

if not set or 0 means NO pause between commands

Autostart

Defines wheter this object will be started (true) after System startup or not (false)

Retry to start

Time to wait after the startup failed to restart the object.

General settings

Max feedback time	Sets the time on how long the system waits for a feedback after an invention has been executed
Show norm light button	Sets whether the User client shall show the normlight button or not
Show TV scene button	Sets whether the User client shall show the TV scene button or not
Start on Interface cmd	Sets whether the object shall be start on any cmd from user client or not
Has light module	Sets whether the user client shall include the light module or not

Has sound module	Sets whether the user client shall include the sound module or not
Has scent module	Sets whether the user client shall include the scent module or not
After intervention	Sets the action after the intervention has been finished Possible values are: Biodynamic, Off, Previous state
On cancel intervention	Sets the action after the intervention has been cancelled Possible values are: Biodynamic, Off, Previous state
Send interval	Sets the interval of the light changes sent to the light modules

4.1.2. Protocol Data Exchange between Light and Controller

The controller acts as master to the lights. All settings and commands will be sent by the controller.

4.2. DATA: Controller to Light

The light sets the requested color and brightness values in the requested fade time by its own with predefined correction values.

Basis: EnOcean 4BS Telegram: A5-38-09 / modified

Byte	Description	Bit pos	Function	Values
0x03	OPTION FLAG FARBTEMPERATUR	0.7 0.6...0.0	Reserve, Future Use Color Temperature	0 0 -> 2000K 1 -> 2050K (INC 50K) 127 -> 8350K (1)
0x02	BRIGHTNESS	0.7...0.0	Definition Brightness	0 (OFF) 1 -> Minimalwert 255 -> Maximalwert
0x01	FADETIME	0.7...0.0	Definition Fade Time	0...180 -> 1Sec 181 -> 4Min 255 -> 75Min
0x00	FUNCTION (Receiver)	0.7...0.4	UPLIGHT DOWNLIGHT FLÄCHE DOWNLIGHT SPOT	7 6 5
	LEARN BIT	0.3	Data telegram Learn Controller	1 Default 0
	SEND STATUS	0.2	No Status Send Status	1 0 Default

STORE FINAL VALUE	0.1	Yes	1 Default
		No	0
SERVICE MODE FLAG	0.0	Service function	1
		Normal operation	0 Default

(1) Received color temperature values will be adapted to the defined light color. (Range 2200K..5000K)

4.3. Statusresponse from Light to Controller

Ist Statusflag in FUNCTION is at *Send Status*, a Status response with the current values will be sent 5.5 sec after the last command.

Byte	Description	Bit pos	Function	Values
0x03	OPTION FLAG	0.7	Reserve, Future Use	0
	CURRENT VALUE	0.6...0.0	Current Color Temperature	0 -> 2000K 1 -> 2050K (INC 50K) 127 -> 8350K (1)
0x02	CURRENT VALUE BRIGHTNESS	0.7...0.0	Current brightness	0 (OFF) 1 -> min value 255 -> max value
0x01	ERROR FLAG	0.7	Error driver (auto reset)	0 -> No error 1 -> Error
	TEMPERATURE VALUE	0.6..0.0	Temperature of the light	0 -> 0°C 100 -> 100°C Max 101..124 reserved, Errorspecific. 125 -> Error Temp. Sensor 126 -> No Temp. Sensor 127 -> Temp. Measure in process, Initial value (2)
0x00	FUNCTION (Absender)	0.7...0.4	UPLIGHT	7
			DOWNLIGHT FLÄCHE	6
			DOWNLIGHT SPOT	5
	LEARN BIT	0.3	Data telegram	1
			Learn Controller	0
	SEND STATUS	0.2	No Status	1
			Send Status	0
	STORE FINAL VALUE	0.1	Yes	1
			No	0
	SERVICE MODE FLAG	0.0	Service function	1
			Normal operation	0

(2) Not all light elements have integrated temperature sensors. Values from 101 to 125 will be used for Status and error specifications.

4.4. Teach In

After pressing a button or by manually activating the teachIn function on the light, the light will switch to the teachIn mode. This will be shown through a flashing light.

The TeachIn mode will be activated with a maximum duration of 1 minute. After that the light will be switched back to normal operation. A reactivating of the TeachIn mode will can be switched immediately.

If the Controller sends a Telegram with the TeachIn bit set within this time, the light will be taughted in.

Maximum 5 controllers can be taughted in. A long press of the button of at least 5 seconds will delete all taughtedIn devices.

4.5. Definition Factory Default

In factory default state, the following functions and values are predefined:

- Each controllr can control the light
- The light is preset to 15% light and 3000K for up-Lights and down-lights.

Timing of commands

The minimum time between commands shall be at least 250ms.

Dimming control

The new values must be at least 100K or 5 brightness points alter to the previous value.

After receiving of the last value and the 'status flag' has been set, a status response will be sent to the controller after 5.5 sec.

4.6. Luminaire

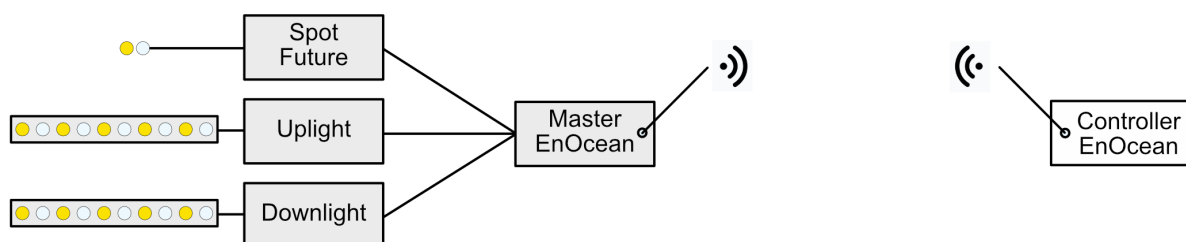


Figure 24: Luminaire schematic

The advanced logical setup of the luminaire gives further possibilities for future applications and modifications.

Master:

The master microcontroller system communicates with the system controller and sets the "slaves" to the specified brightness and color values.

Slave - Uplight, Downlight system:

The slave has its own microcontroller, which sets each LED driver to the appropriate value and controls temperature of the high power LED strips.

5. Sound Module

The sound module offers sound playback for the GREAT system. It is built on top of existing open source software with a minimal abstraction layer to safely connect the sound module to the GREAT system. Compared to the sound module used in the functional tests (see D2.2), the sound module has been extended to support individualized playlists, allow for a more finegrained volume control over time, and simplified triggering of activation / relaxation stimuli.

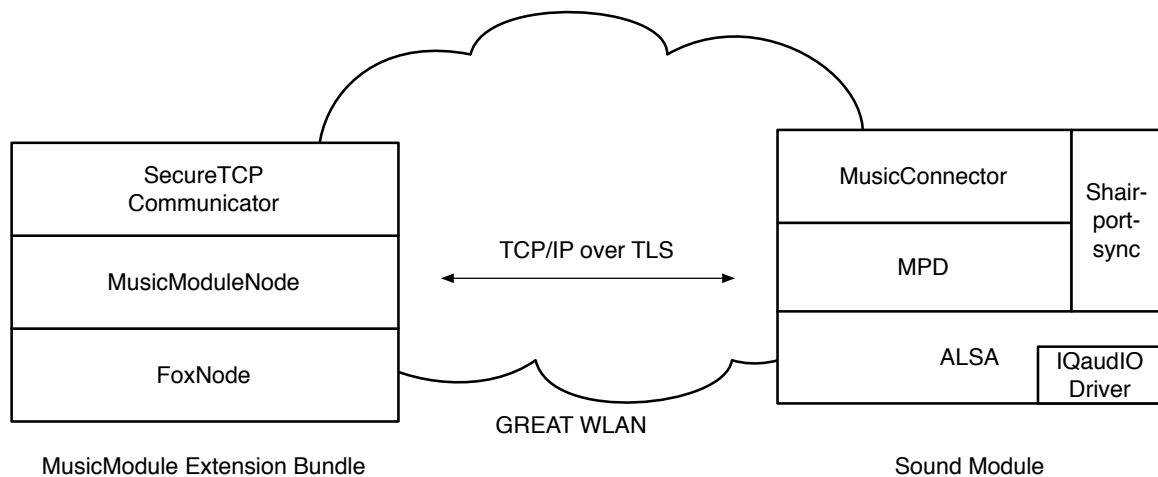


Figure 25: Communication between the Intefox music module extension and the sound module.

The software stack of the sound module is based on the Raspbian Stretch Lite Linux distribution for Raspberry Pi. For sound playback, the open source music player daemon (MPD) is used in combination with the mpc client tool for control (see Figure 25).

A thin abstraction layer on top offers secured communication with the GREAT system and remote playlist updating.

In addition to the playlist-based sound playback offered by mpd, also AirPlay music streaming is supported using the open source shairport-sync package. This allows for using the GREAT sound module also as independent wireless speakers.

The sound module is connected to the GREAT system via WLAN. The connection to the network is monitored and if the network connection is lost (typical reasons could be power outage at the main controller, wireless signal interferences...), reconnection attempts will be made periodically until they succeed. This also allows for automatic connection of the module during installation and after network dropouts.

Availability of the sound module inside the GREAT network is announced via the open source avahi-daemon package.

5.1. Image Preparation

5.1.1. Basics

The basis for the software stack is the Raspbian Stretch Lite distribution. Once this is installed, the other software packages used by the sound module must be installed.

First the driver for the Pi-DACZero sound card needs to be activated. This is done using a dtoverlay-entry in the /boot/config.txt file:

```
dtoverlay=iqaudio-dacplus
```

Optionally the onboard audio can be disabled by commenting out the dtparam for audio-on:

```
#dtparam=audio=on
```

Setting up the music player functionality involves installing the mpd and mpc packages:

```
sudo apt-get install mpd mpc
```

In the mpd.conf file, the user group should be adjusted to the audio group

```
group "audio"
```

and the audio output needs to be adjusted to use the hardware output devices provided by the IQaudIO driver

```
audio_output {
    type          "alsa"
    name          "My ALSA Device"
    device        "hw:0,0" # optional
    mixer_type     "hardware" # optional
    mixer_device   "hw:0" # optional
    mixer_control  "Digital" # optional
    mixer_index    "0" # optional
}
```

Other than that, the default settings are fine.

For the update functionality, the python requests package is required:

```
sudo apt-get install python-requests
```

Also, a sounds directory and a tmp directory need to be created that are used for music storage or temporary files during updates.

```
mkdir /home/pi/sounds
mkdir /home/pi/tmp
```

For the TLS connections to work, the required certificates and key files need to be copied to /home/pi/certs on the PI:

```
ca-chain.cert.pem, great_music.cert, great_music_nopw.key
```

Finally, the files musicModuleServer.py, ThreadedPlaylistUpdater.py and startMusicConnector.sh need to be copied to the system and an entry to the crontab needs to be made to automatically start the music connector on startup:

```
@reboot sh /home/pi/startMusicConnector.sh
```

5.2. Shairport-Support (for AirPlay functionality, optional)

To build and install the shairport-sync package, follow the instructions, given by the author at <https://github.com/mikebrady/shairport-sync>:

Install necessary packages to build the shairport-sync daemon:

```
sudo apt-get install build-essential git xsltoman
sudo apt-get install autoconf automake libtool libdaemon-dev libpopt-dev libconfig-dev
sudo apt-get install libasound2-dev
sudo apt-get install avahi-daemon libavahi-client-dev
sudo apt-get install libssl-dev
sudo apt-get install libsoxr-dev
```

Then checkout the latest version of the shairport-sync source code, configure, make and make install it:

```
git clone https://github.com/mikebrady/shairport-sync.git
cd shairport-sync
autoreconf -i -f
./configure --sysconfdir=/etc --with-alsa --with-avahi --with-ssl=openssl --with-metadata --with-soxr --with-systemd
make
sudo make install
```

Then create a shairport-sync directory in /var/run and change the owner to shairport-sync:shairport.sync

```
sudo mkdir /var/run/shairport-sync
sudo chown shairport-sync:shairport-sync /var/run/shairport-sync
```

Then create a shairport-sync.conf file in /etc/tmpfiles.d to allow shairport to create temporary files. The content should be:

```
d /var/run/shairport-sync 0755 shairport-sync shairport-sync -
```

Finally adjust the shairport-sync.service file to run as a daemon by editing /lib/systemd/system/shairport-sync.service and add a '-d' to the ExecStart and adding a Type = forking line.

```
ExecStart=/usr/local/bin/shairport-sync -d
Type=forking
```

In /etc/shairport-sync.conf the name of the speakers (in general section) and quality parameters can be set. If no name is set, the hostname will be used.

Finally enable the service:

```
sudo systemctl enable shairport-sync
```

5.3. WLAN Setup

Each sound module is prepared to connect to the GREAT wireless network. This is done via the wpa_supplicant.conf file.

```
network={
    ssid="GREAT"
    psk=87b61f54914c527c67f87766167db5f9626c31a5c7a0d9e30cfe5024be6fa1ec
    key_mgmt=WPA-PSK
}
```

Instead of supplying the password in clear text, the wpa_passphrase tool can be used to generate the psk. For this the ssid and password need to be supplied as parameters.

The interfaces configuration file /etc/network/interfaces sets up the wlan0 interface to use the wpa_supplicant.conf file.

The interface definition of the interfaces file in /etc/network/ contains:

```
source-directory /etc/network/interfaces
auto lo
```

```
iface lo inet loopback
```

```
iface eth0 inet manual
```

```
allow-hotplug wlan0
```

```
iface wlan0 inet manual
```

```
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

To watch for network disconnects, a wlanConnector script is run periodically to check if the gateway is still available. If it's not available anymore, the wlan0 interface is shut down and re-enabled to try to reconnect again:

```
#!/bin/bash

# ping the router, no need to hit google for this.
SERVER=172.24.1.1

#specify wlan interface
WLANINTERFACE=wlan0

# Only send two pings, sending output to /dev/null
ping -I ${WLANINTERFACE} -c2 ${SERVER} > /dev/null

# If the return code from ping ($?) is not 0 (meaning there was an error)
if [ $? != 0 ]
then
# Restart the wireless interface
/sbin/ifdown --force ${WLANINTERFACE}
/sbin/ifup ${WLANINTERFACE}
fi
```

The wlanConnector script is run every two minutes as a cron job with the following definition in crontab:

```
*/2 * * * * /home/pi/wlanConnector.sh
```

5.4. Music Module Extension Bundle

The music module system bundle provides functionality of the sound module inside the Intefox automation system. It communicates with the sound module over a TLS secured TCP connection.

The music module bundle is implemented using the OSGi based plugin architecture provided by the Intefox fox.core system.

The music module bundle provides a range of input-connections that can be connected to outputs of other event sources. It also provides output events that can be connected to inputs of other components (see Figure 26). In the GREAT context, the output events are mainly used for logging the system state. See Table 8 for a description of the input events, Table 9 for a description of the output events and Table 10 for a description of the parameters.

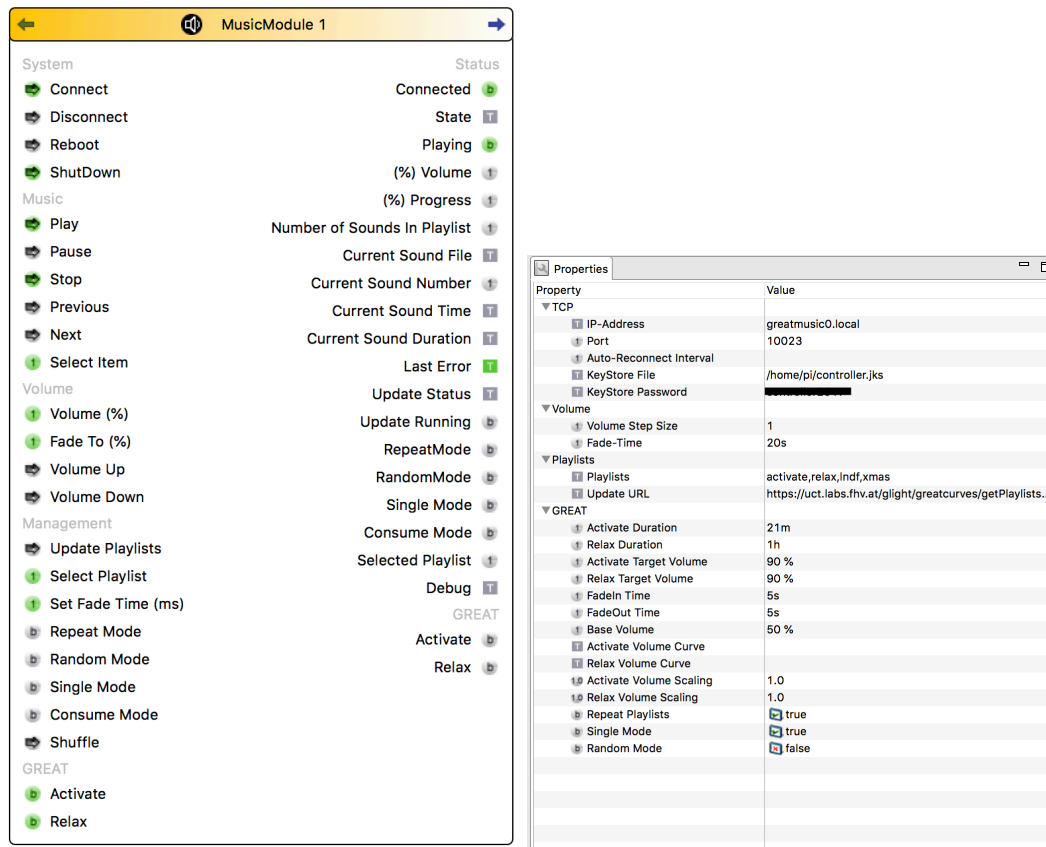


Figure 26: The available connections for the music module extension for the Intefox system and its parameter settings.

Table 8: Description of input events of the music module extension.

Input	Description
Connect	Tries to connect the node to the module at the address given in the URL parameter.
Disconnect	Disconnects from the module and closes all connections
Reboot	Causes the Sound module device to reboot.
ShutDown	Causes the Sound module device to safely shut down.
Play	Starts playback of the current song
Pause	Pauses playback of the current song
Stop	Stops playback of the current song
Previous	Jump to the previous song in the playlist
Next	Jump to the next song in the playlist

Select Item	Selects the song at the given number in the playlist (e.g. the first song would be 1)
Volume	Immediately sets the volume to the given percentage level (volume control is logarithmic)
Fade To	Fades to the given percentage level within the Fade Time that is provided in the parameters.
Volume Up	Increases the volume by the step size defined in the parameters
Volume Down	Decreases the volume by the step size defined in the parameters
Update Playlists	Causes the sound module to update its playlists. The playlist definition is downloaded from the URL that is defined in the parameters
Select Playlist	Selects the playlist at the given index (zero means the first playlist defined in the Playlists parameter settings)
Set Fade Time	Sets the fade time to the given time in ms
Repeat Mode	Turns the repeat mode on/off.
Random Mode	Turns the random mode on/off (plays tracks of the playlist in a random order)
Single Mode	Turns the single mode on/off (only plays a single track of a playlist).
Consume Mode	Turns the consume mode on/off (when random mode, consumes already played tracks so they aren't played twice).
Shuffle	Shuffles the order of the current playlist.
Activate	Selects and starts the activate playlist on True, stops playback on False
Relax	Selects and starts the relax playlist on True, stops playback on False

Table 9: Description of output events of the music module extension.

Output	Description
Connected	True if connected to the sound module, False if not
State	Possible values: Playing, Paused, Stopped, Finished
Playing	True if currently playing, False otherwise
Volume	Current volume level in %
Progress	Progress of the current sound playback in %
Number of Sounds in Playlist	Number of sounds in current playlist
Current Sound File	The currently selected file for playback
Current Sound Number	The number of the currently selected file inside the currently selected playlist
Current Sound Time	The current time of the currently playing sound
Current Sound Duration	The total time of the currently playing sound
Last Error	The last received error message
Update Status	The status of the update process.
Update Running	True if the update is currently running, False otherwise

Repeat Mode	Status of the Repeat Mode
Random Mode	Status of the Random Mode
Single Mode	Status of the Single Mode
Consume Mode	Status of the Consume Mode
Selected Playlist	Index of the selected playlist (zero based)
Debug	Debug text output (only used for development)
Activate	Boolean indicating if an activation session is running
Relax	Boolean indicating if a relaxation session is running

Table 10: Description of parameters of the music module extension.

Parameter	Description
IP-Address	IP-Address or link local name of the sound module
Port	The port on which the sound module listens (default 10023)
Auto-Reconnect-Interval	Interval in seconds for automatic reconnection if the connection gets lost. 0 means automatic reconnect is disabled.
KeyStore File	The path to the keystore file containing the certificate for connection to the sound module.
KeyStore Password	The password for the keystore file
Volume Step Size	The step size for volume up and volume down commands
Fade Time	The time in seconds to transition to the new volume set by the Fade To command
Playlists	Comma separated list of predefined playlist names for the mapping between Index (zero based) and playlist name. 'activate' and 'relax' playlists are mandatory.
Update URL	The URL that provides the playlist information for the sound module. This URL is queried when the Update command is issued.
Activate Duration	The duration of the activate session. If the activate playlist lasts longer, it is stopped after this time.
Relax Duration	The duration of the relax session. If the relax playlis lasts longer, it is stopped after this time.
Activate Target Volume	Percentage of the volume for activate sessions.
Relax Target Volume	Percentage of the volume for relax sessions.
FadeIn Time	Time it takes from start to the defined volume level
FadeOut Time	Time it takes to fade out from the defined volume level to the base volume at the end of a session.
Base Volume	Percentage of the volume defined as starting/endpoint
Activate Volume Curve	Comma separated list of time:volume pairs. Allows for individualized volume progression during a playlist. This setting is optional. If present, it overrules target volume settings.

Relax Volume Curve	Comma separated list of time:volume pairs. Allows for individualized volume progression during a playlist. This setting is optional. If present, it overrules target volume settings.
Activate Volume Scaling	Factor to scale incoming volume values during activation sessions
Relax Volume Scaling	Factor to scale incoming volume values during relaxation sessions
Repeat Playlists	Default settings if a playlist should be repeated (might be adjusted according to the downloaded playlist)
Single Mode	Default setting if a playlist should be put to single mode (might be adjusted according to the downloaded playlist)
Random Mode	Default setting if a playlist should be put to random mode (might be adjusted according to the downloaded playlist)

5.5. Sound Module Protocol Description

The communication between the music module extension for the Intefox system and the sound module hardware is based on TLS secured TCP/IP streams. Messages are exchanged in a text-based form and its commands are based on the open source mpc tool for controlling the mpd music player daemon that is used for the actual sound playback. A separate musicConnector acts as a wrapper to mpc/mpd that provides a secured communication layer to the Intefox extension and communicates with the mpd via the mpc tool.

musicModuleExtension | < - > | musicConnector -> mpc -> mpd

The music connector receives commands from the music module extension via an TLS secured TCP/IP stream. It parses the commands and sends them on to the mpc tool that controls the mpd.

The musicConnector queries the mpc tool for the status of the mpd and sends the status messages back to the music module extension on the Intefox system.

The music connector acts as a server where the music modules extensions connects to. Only one client can connect to the server (a music module extension on the Intefox system has exclusive access).

The music module extension and the sound module must authenticate themselves using a TLS-Certificate. Only communication between verified peers is allowed.

Messages are sent using a text-based stream. If multiple parameters are sent along, they are delimited by spaces or tabs. Each message is terminated by a newline character.

cmd [param]><NL>

If the command is accepted by the musicConnector, a simple ack message is sent back. If a command cannot be handled, a nack message is sent back.

5.5.1. Commands from Client to Server:

nop

This is a keep alive message with no other purpose.

play <itemNumber>

Plays the item at the specified number. This invokes the mpc play command. The item number parameter is 1 based (1 means the first item).

stop

Stops the playback. This invokes the mpc stop command.

next

Switches to the next track in the playlist. This invokes the mpc next command.

prev

Moves to the previous track in the playlist. This invokes the mpc prev command.

pause

Pauses playback of the current track. This invokes the mpc pause command.

shuffle

Shuffles the current playlist. This invokes the mpc shuffle command.

random on | off

Sets the random playback mode of the mpd. This invokes the mpc random command with the supplied argument.

single on | off

Sets the single playback mode of the mpd. This invokes the mpc single command with the supplied argument.

repeat on | off

Sets the repeat mode of the mpd. This invokes the mpc repeat command with the supplied argument.

consume on|off

Sets the consume mode of the mpd. This invokes the mpc consume command with the supplied argument.

volume <targetVolume>

Sets the playback volume of mpd. This invokes the mpc volume command. If a fade operation is ongoing at the moment, it is terminated, and the volume is immediately set to the targetVolume.

fadeTo <targetVolume> <fadeTime>

Fades the volume from the current level to the target volume level within the specified fadeTime (in milliseconds). This repeatedly invokes the mpc volume command with intermediate steps until the target volume is reached.

playlist <playlistName>

Switches to the specified playlist. This invokes the mpc clear command followed by the mpc load command with the specified playlist name. Only if playback was active at the time, the playback for the new playlist will be started automatically. In this case the mpc play command will be invoked for the first item.

startPlaylist <playlistName>

Switches to the specified playlist and starts playback. This invokes the mpc clear command followed by the mpc load command with the specified playlist name followed by the mpc play command for the first item.

update url

Invokes an asynchronous update of the music playlists. The server at the URL is expected to return a playlist description in json format. Then, for each listed item a mpd compatible m3u playlist file is generated and the respective files are downloaded from the sources specified in the json description, if they are not already present on the system.

Core-JSON-format expected as return of the update command:

```
{
  <playlistName1> = {
    [
      "soundFile"="<destination path of sound>",
      "sourceURL"="<download url for sound>"
    ],...
  },
  ...
}
```

```

<playlistName2> = {
    [
        "soundFile"="<destination path of sound>",
        "sourceURL"="<download url for sound>"
    ],...
},...
}

```

An example URL for this call is i.e.:

<https://uct.labs.fhv.at/glight/greatcurves/getPlaylists.php?zoneKey=uctZone&playlistsOnly=true>

The reply contains playlists with filename and source URL infos (in the GREAT context playlist names are supposed to be "activate" and "relax"). Each playlist contains an array of items, where each item is specified by the soundFile property that specifies the target location on the music module system, and a sourceURL property that specifies where the sound is available for download from, in case it doesn't already exist on the system.

If the flag playlistsOnly in the URL is omitted, a more detailed version of the playlists, including meta-info is used. This fully featured playlist-info is only used by the music module extension, but not by the music module itself. The music module extension automatically appends the playlistsOnly flag to the URL passed to the music module and uses the contained meta-data to set the parameters accordingly. A typical example of such a fully featured playlist including meta-info is shown below:

```

{
  "activate": [
    [
      "Sound011_activate_70kHz_herzschlag.wav",
      "https://uct.labs.fhv.at/greatsounds/Sound011_activate_70kHz_herzschlag.wav"
    ],
    [
      "Sound008_activate_30kHz_herzschlag.wav",
      ""
    ]
  ],
  "relax": [
    [
      "Sound007_relax_30kHz_stetig.wav",
      ""
    ],
    [
      "Sound009_relax_30kHz_atem.wav",
      ""
    ]
  ]
}

```

```

    ]
  ],
  "dateDefined": "2019-07-11 11:46:38",
  "firstRequestedDate": "2020-01-31 14:34:50",
  "firstRequestedFrom": "193.170.2.20",
  "lastRequestedDate": "2020-01-31 14:34:50",
  "lastRequestedFrom": "193.170.2.20",
  "playlistID": "58",
  "zoneName": null,
  "activateDuration": 1260,
  "relaxDuration": 3600,
  "singleMode": true,
  "randomMode": false,
  "repeatMode": true,
  "activateVolumeScaling": 1,
  "relaxVolumeScaling": 1
}

```

The major differences to the playlistOnly format are the keys-value pairs for the meta-info.

Status info of the asynchronous update process on the music module are sent to the music module extension while the process is running.

system shutdown|restart

Shuts down or restarts the music module.

5.5.2. Commands from Server to Client:

Note, in contrast to the client-server messages, parameters in server-client messages are delimited by tab, as params can potentially contain spaces in their values.

status finished|playing|stopped|paused

The current playback status. If finished, the playlist has been played to the end. If playing the playlist is currently playing, if stopped, the playlist is currently stopped, if paused, the playlist is currently paused.

volume <currentLevel>

The current volume level in %.

repeat on | off

The current repeat setting of mpd.

random on | off

The current random setting of mpd.

single on | off

The current single setting of mpd.

consume on | off

The current consume setting of mpd.

currentSound <soundName>

The name of the currently playing sound.

updater <statusMessage>

The current status of the playlist update process (e.g. downloading of sound files).
The status message is a tab separated list of more infos.

update <message>

error <error message>

ack

The command was received and executed

nack

The command was received but not understood. No action has been taken.

5.6. Relevance/Reuse-Potential outside GREAT

The developed wrapper to mpd/mpc and the extension for the Intefox system can be used in any scenarios that involve mpd-based music playback systems that need to be controlled from home automation systems.

Outside Intefox powered smart buildings the music module can also be easily integrated into other smart building middleware systems, due to its lightweight and open protocol.

The music module can be used independently of any home automation system as an AirPlay speaker system. The music connector automatically handles AirPlay sessions.

One major challenge during the GREAT project was to identify and develop sounds that not only support the intended activation or relaxation of persons within reach but are also accepted by the persons. The sound concept used for this is described in the next chapter.

5.7. Sound Concept

Hearing is a physical phenomenon, provided the affected possess appropriate receptors. Processing these stimuli cognitively creates an experience of hearing with specific characteristics. These are directly measurable as characteristics of perception as opposed to the original stimuli.

Sounds from natural contexts form the basis of the auditory stimuli used. An advantage of sounds from natural environments over musical stimuli is that music can have very divergent effects on different individuals due to different individual preferences, emotional conditioning and other subjective factors. Certainly, this risk of uncontrollable association and effect based on individual factors also exists for sounds from nature, since these impressions of different persons or groups of persons could also be linked with different references. However, the extent of emotional semantics of sounds such as the sound of the sea does not seem to be as pronounced as the potential of much music: These natural sounds originate from an environment in which, together with complementary visual olfactory stimuli, they initially provide information about the environment. In their origin they have, at least initially, a predominantly informative character. Music, on the other hand, does not initially have a purely informative character in its origin, but is designed to have a certain effect or a certain content, it takes place in an environment in a broader sense, culturally, spatially, temporally, which does not formulate an informative claim but one based on subjective reception (be it aesthetic, emotional or otherwise). Accordingly, music and its effects are strongly influenced by subjective influences.

When designing auditory stimuli for use with a previously indeterminable number of different persons, the greatest possible acceptance should therefore be achieved by keeping the risk of widely divergent reception and thus widely divergent impact as low as possible. As the use of these stimuli is ultimately also designed to be applicable within the framework of a modular, multisensory approach or mode of action, i.e. in conjunction with visual and olfactory stimuli, it is necessary to include cross-modal design possibilities in the production of the auditory stimuli.

This means, apart from the basically positive effect of certain natural sounds, to consider the design possibilities of the parameters that can be treated independently of the special semantic content, in order to promote the greatest possible compatibility in the sense of increasing or supporting the effect of the other

stimuli in the combined multisensory interaction. There are connections between auditory and visual or olfactory perception.

We looked at several adaptable aspects of sounds and evaluated their valence, receptibility and direction (meaning relaxing or activating) as illustrated in Figure 27.

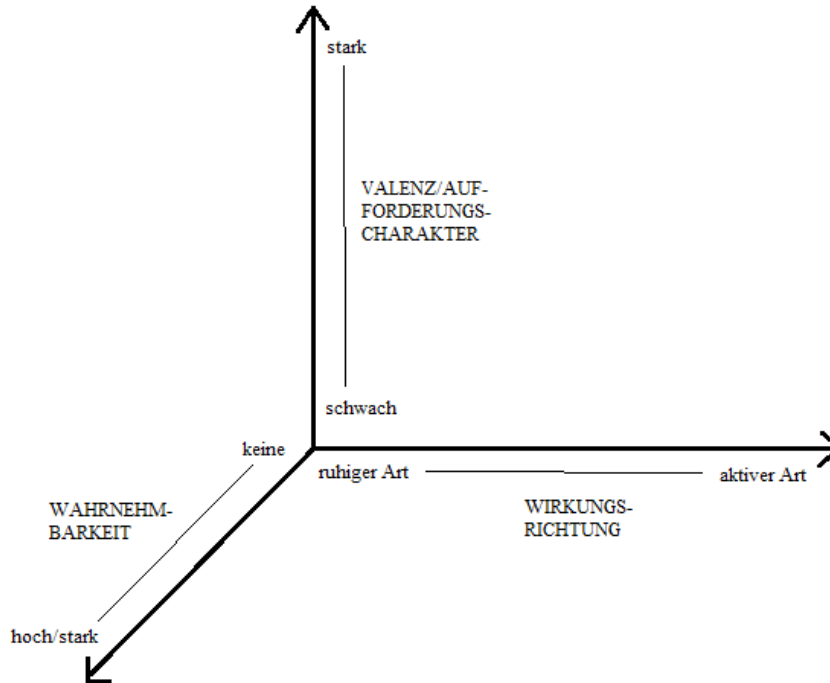


Figure 27: Dimensions of sound characteristics regarding receptibility, valence and direction

Volume

When recording natural sounds and trying to model the volume to a different place sounds can be too quiet or too loud for comfort. Therefore, a logarithmic scale should be applied to create satisfyingly loud experiences.

Valence

Changing the volume of a sound leads to a directly proportional change in its valence.

Receptibility

The general Volume influences the receptibility immensely, as it is the most important feature: it doesn't matter what other features a sound has, if no one can hear it. Volume, therefore, is a basic requirement for reception of sounds.

Direction

Volume has a supporting role concerning this aspect, as it does not possess any qualitative functions cooperating with other features.

Duration

Meaning the temporal extension of an individual sound as a whole. The duration, structure of its parts will be discussed more closely in the chapter shape. There is a

difference between stationary (continuing or repeating) and non-stationary (only appear once) sounds. Stationary ones can be set to a desired duration.

Valence

A stationary, long sound could – by maintaining – continuously augment other features influencing valence. If these other features are very directing (concerning relaxation or activation), a long duration could first lead to an increase by persistence. On the other hand, it could also lead to a decrease as there might be a habituation effect. For non-stationary sounds, the valence is probably connected to other features, very short duration could negate any effects of high valence.

Receptibility

Apart from being too short, the duration should not play a role.

Direction

Duration should not affect, whether a sound is perceived as relaxing or activating.

Contextualization

Every environment has its own acoustic information. Adding another sound can either change its context or leave it unchanged. There should be a balance between interrupting, continuing and creating atmospheres.

Valence

Small differences between previous and new context might remove valence completely. Increasing the variation might create a new context but does not necessarily improve valence.

Receptibility

To actively perceive a sound, it must be discernable in the given environment. A big difference between sound and context means good receptibility.

Direction

Contextualization defines the direction and strength of the stimuli by creating contrasts and subjectively perceived changes in their categorization between relaxing and activation.

Timbre

It describes not only static states of the frequency spectrum but also processes which appear between our varying frequency spectrums.

Valence

A source of sound which gets louder and changes from dull to bright is closing in on us – and is therefore, evolutionary seen, of high valence. If it changes in the opposite direction the “threat” is leaving and can be ignored. The human ear is also tuned in to certain frequencies and is more attentive when they appear.

Receptibility

Human speech usually ranges between 200 and 5000 Hz, our ears are hearing most efficiently between 2000 and 5000 Hz. Therefore, we receive sounds in that range best.

Direction

Bright sounds tend to be perceived as activating, dull sounds as relaxing.

Description of the interventions

To use auditive stimuli with maximum acceptance it's necessary to minimize the risk of varying reception and effect between different participants. Natural sounds and music can both lead to negative associations, depending on their previous experience and connected memories. After our initial approach of using natural sounds and some surprising associations with them, we created abstract sounds that are described below.

Activating sound

The activating scenario provides a 20-minute course. The sound (blowing of wind through a forest) has a very low volume at the beginning of the course and continues to increase until it reaches its maximum after 5 minutes. This level is held for a period of 15 minutes until it falls to a medium level at the end of the course to discharge the activating phase.

The pitch increases over the entire course by 200 percent. The timbre brightens when the threshold of a low-pass filter is raised. The low-pass filter with a slope of 18dB / oct starts with a high cut at 100 hertz and opens completely for the first 5 minutes.

Calming sound

The calming scenario envisages a progression over a period of 60 minutes. The used audio sample (= the used, digitally stored sound), sea noise, starts with a low volume and increases it over 2 minutes to a medium volume. This level will decrease for 8 minutes to remain at a low volume for another 40 minutes. Then, to discharge the calming scenario, the volume level increases again over 10 minutes until it finally reaches the initial level.

The tone color of the sample is rather dull at the beginning, as a low pass filter with a slope of 12 dB / octave performs a lowcut at 4000 Hz.

Later, the threshold of the lowcut drops to 200 Hz. In the end, the threshold is increased to 20000 Hz over a period of 10 minutes. Further, over the course of 15 minutes, the spatial information of the sound has been changed so that by using a reverb with a reverberation time of 4.2 seconds, the dry / wet ratio changes from 100/0 to 0/100 so as the sound seems to move away.

6. Scent Module

The scent module is a remote controllable 2-channel scent dispenser for GREAT. It offers a simple TCP/IP based remote interface for easy integration into home automation systems. Compared to the scent module used in the functional tests (see D2.2), the scent module has been extended in multiple ways:

- Added support for warning messages depending on actuation count (e.g. for bottle replacement)
- Added support for customized trigger sequences for activation/relaxation
- Simplified triggering of activation/relaxation stimuli
- Added support for environmental sensor data of the integrated BME680 sensor.

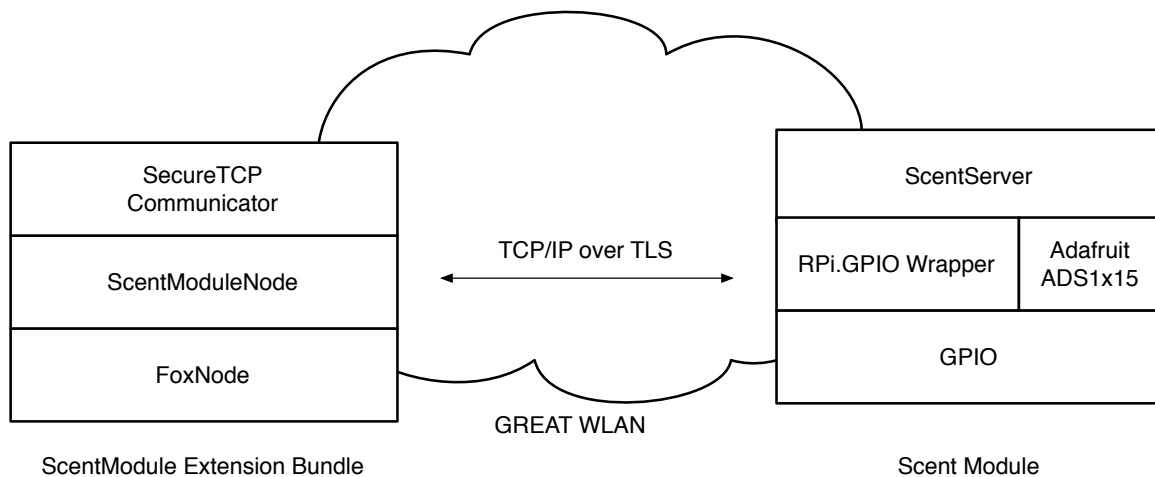


Figure 28: Integration of the scent module into the GREAT controller system based on Intefox.

The software stack of the sound module is based on the Raspbian Stretch Lite Linux distribution for Raspberry Pi which runs on a Raspberry Pi Zero W hardware. For controlling the scent pump-drive motors, the GPIOs of the Raspberry are used which are controlled via the Python based GPIO wrapper RPi.GPIO (see Figure 28). The current that's flowing through the motors is measured via a shunt and an I2C based analog digital converter (see hardware description). To sample and access these values, the Adafruit_ADS1x15 library is used. A simple peak detection algorithm on these motor current values is then used to identify the end of a pump cycle.

The scent module is connected to the GREAT system via WLAN. The connection to the network is monitored and if the network connection is lost (typical reasons could be power outage at the main controller, wireless signal interferences...), reconnection attempts will be made periodically until they succeed. This allows for automatic connection of the module during installation or network dropouts.

Availability of the scent module inside the GREAT network is announced via the open source avahi-daemon package.

6.1. Image Preparation

The basis for the software stack is the Raspbian Stretch Lite distribution. Once this is set up, the following packages need to be installed:

```
sudo apt-get install git build-essential python-dev
sudo apt-get install python-pip
sudo pip install adafruit-ads1x15
sudo apt-get install python-smbus
```

Each scent module should get a unique and meaningful hostname. In the GREAT context, this should be in the format greatscentXX, where xx is a continuous number. The hostname is adjusted by editing the files /etc/hostname and /etc/hosts by replacing the default "raspberrypi" with the new name.

For the TLS connections to work, the required certificates and key files need to be copied to /home/pi/certs on the PI:

ca-chain.cert.pem, great_scent.cert, great_scent_nopw.key

The WLAN needs to be set up in a similar way as described for the sound module, so the scent module connects itself to the GREAT network and attempts to reconnect itself if the network connection is dropped.

Finally, the files scentServer.py, ThreadedPowerReader.py and startScentServer.sh need to be copied to the system and an entry to the crontab needs to be made to automatically start the scent server on startup:

```
@reboot sh /home/pi/startScentServer.sh >> /home/pi/scent.log 2>&1
```

6.2. Scent Module Extension Bundle

The scent module extension bundle provides functionality of the scent module inside the Intefox automation system. It communicates with the scent module over a TLS secured TCP connection.

The scent module bundle is implemented using the OSGi based plugin architecture provided by the Intefox fox.core system.

The scent module bundle provides a range of input-connections that can be connected to outputs of other event sources. It also provides output events that can be connected to inputs of other components. In the GREAT context, the output events are mainly used for logging the system state or triggering messages for bottle replacement. See Table 11 for a description of the input events, Table 12 for a description of the output events and Table 13 for a description of the parameters.

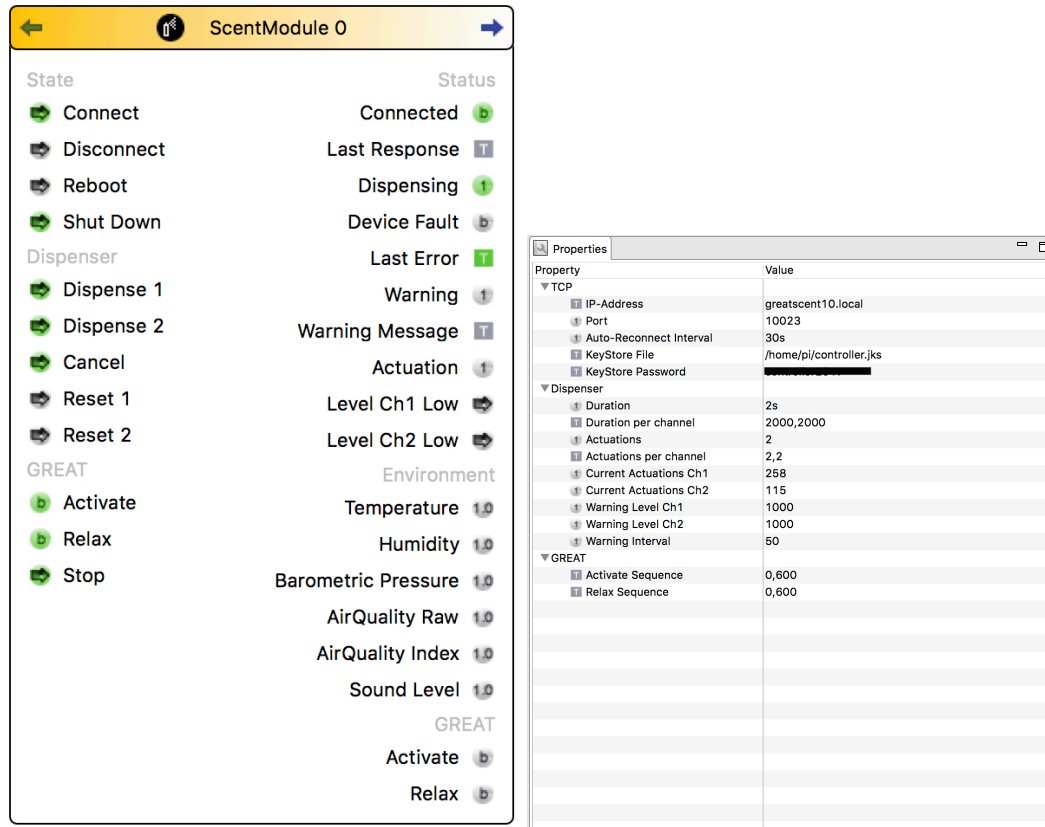


Figure 29: Input and output events (left) and parameter settings (right) of the scent module extension.

Table 11: Description of input events of the scent module extension.

Input	Description
Connect	Tries to connect the node to the module at the address given in the URL parameter.
Disconnect	Disconnects from the module and closes all connections
Reboot	Causes the Sound module device to reboot.
ShutDown	Causes the Sound module device to safely shut down.
Dispense 1	Dispenses scent in slot 1
Dispense 2	Dispenses scent in slot 2
Cancel	Stops scent dispensing immediately
Reset 1	Resets the dispense counter for slot 1
Reset 2	Resets the dispense counter for slot 2
Activate	Switch to turn on/off activation session (only one session can be active at the same time)
Relax	Switch to turo on/off relaxation session (only one session can be active at the same time)
Stop	Stop the currently running session

Table 12: Description of output events of the scent module extension.

Output	Description
Connected	True if connected to the scent module, False if not
Last Response	The last response sent from the scent module
Dispensing	Dispensing state of the scent module: 0...ready, 1...dispensing slot 1, 2...dispensing slot 2
Device Fault	True if the driver module reports a fault state, False otherwise
Last Error	The last error that occurred.
Warning	The last warning code that was received.
Warning Message	The last warning message that was received.
Actuation	Reports a pump cycle on the respective slot.
Level Ch1 Low	Indicator that the level for bottle in slot 1 might be low
Level Ch2 Low	Indicator that the level for bottle in slot 2 might be low
Temperature	The current temperature in the room
Humidity	The current humidity in the room
Barometric Pressure	The current barometric pressure in the room
AirQuality Raw	Indicator for AirQuality (the higher the value, the better the air quality and the lower VOC concentration)
AirQuality Index	Air Quality Index (experimental): the higher the index, the better the air quality.
Sound Level	Reserved for future implementation of a sound level sensor.
Activate	Boolean indicating if an activation session is running
Relax	Boolean indicating if a relaxation session is running

Table 13: Description of parameters of the scent module extension.

Parameter	Description
IP-Address	IP-Address or link local name of the scent module
Port	The port on which the scent module listens (default 10023)
Auto-Reconnect-Interval	Interval in seconds for automatic reconnection if the connection gets lost. 0 means automatic reconnect is disabled.
KeyStore File	The path to the keystore file containing the certificate for connection to the scent module.
KeyStore Password	The password for the keystore file
Duration	The max. duration of a dispense action (overridden by the Duration per channel setting)
Duration per channel	CSV list of max duration in ms, e.g. 3000,1000 would mean 3 seconds for slot 1, 1 second for slot 2
Actuations	Number of actuations (pump cycles) per dispense action (overridden by the Actuations per channel setting)

Actuations per channel	CSV list of number of actuations per channel, e.g. 2,3 would mean 2 pumps for slot 1, and 3 pumps for slot 2
Current Actuations 1	Current number of actuations for slot 1 since the last reset
Current Actuations 2	Current number of actuations for slot 2 since the last reset
Warning Level Ch1	The number of actuations after which a warning should be triggered for slot 1
Warning Level Ch2	The number of actuations after which a warning should be triggered for slot 2
Warning Interval	Defines, after how many actuations the warning should be repeated
Activate Sequence	Comma separated list of milliseconds after which an actuation should be triggered
Relax Sequence	Comma separated list of milliseconds after which an actuation should be triggered

Being able to define different times for delayed repeated triggerings of actuations allows for better fitting of the scent dispensing to room characteristics (e.g. different air flow in rooms).

6.3. Scent Module Protocol Description

The communication between the scent module extension for the Intefox system and the scent module hardware is based on TLS secured TCP/IP streams. Every message received from the client is confirmed by a response from the server running on the scent module hardware. The server also sends status messages to the client.

The scent module extension and the scent module server must authenticate themselves using a TLS-Certificate. Only communication between verified peers is allowed.

Messages are sent using a text-based stream. If multiple parameters are sent along, they are delimited by spaces or tabs. Each message is terminated by a newline character.

<cmd [param]><NL>

If the command is accepted by the scent module server, a simple ack message is sent back. If a command cannot be handled, a nack message is sent back.

6.3.1. Commands from Client to Server:

nop

This is a keep alive message with no other purpose. If no nop message is received within 1 minute, the TCP connection is assumed to be lost.

dispense <channel> <duration> [<actuations>]

Triggers a dispense operation for the channel given (1 means first slot, 2 means second slot) with the specified duration in ms. Optionally also an actuation count (how many pumps should be applied) can be passed. In this case, the duration parameter sets the maximum time until the operation should be stopped even if the actuation count is not reached (e.g. in cases where there is no bottle, or the motor is blocked).

cancel

Immediately stops the current operation.

system shutdown | restart

Shuts down or restarts the music module.

6.3.2. Commands from Server to Client:

Note, in contrast to the client-server messages, parameters in server-client messages are delimited by tab, as params can potentially contain spaces in their values.

ack [<status> <channel>]

The command was received and executed. If the command related to a dispense operation, the status and the channel involved are included.

Available states:

done: the action has completed for channel channel

nack

The command was received but not understood. No action has been taken.

input fault <val>

Signals whether the fault flag is active or inactive (val=1: fault flag set, val=0: fault flag reset)

warning <type> <message>

Signals a warning to the client. The following warning types exist:

- 1: No motor connected?
- 2: No bottle inserted?
- 3: Motor blocked?
- 4: Device busy (e.g. when already dispensing on another channel)

event <type> <data>

Signals an event of type with the specified data.

Available types:

pumped	indicates a pump event on a channel
data	the channel number for which the pump action occurred
iaq	air quality measures
data	tab separated list of air quality measures: temperature,humidity,airPressure,resistance,iaqIndex
soundLevel	sound level measure
data	sound level value

6.4. Relevance/Reuse-Potential outside GREAT

The developed scent module can be easily used within Intefox powered smart buildings, but also allows for easy integration into various other smart building middleware systems, due to its lightweight and open protocol.

7. EnOcean Repeater

During adaptations to the field test setups at the testing places in Hall and Neumarkt, reception issues of EnOcean telegrams between the controller and the luminaires have been discovered. To work around the limited range of EnOcean telegrams, a special purpose repeater was developed, that besides the standard EnOcean repeating functionality also offered more customization possibilities regarding which telegrams should be repeated, as well as a logging functionality of received telegrams and their signal strength, to assist in finding the best placements of the repeater. This allowed for analysis of the situations when a luminaire did not handle a switching command correctly.

The software stack of the repeater module is based on the default Raspbian Stretch Lite Linux distribution for Raspberry Pi and a simple bridge script to forward EnOcean telegrams via a network connection. The system is configured as ReadOnly to be resistant against sporadic power outages.

The bridge script running on the Raspberry Pi is invoked automatically after reboot. It initializes the TCM310 module and activates the EnOcean repeater functionality (by default level 1) of the module. It then listens for EnOcean telegrams and forwards them via UDP to the configured receiver. A simple UDP receiver at the other end will then receive the raw EnOcean ESP3 telegram information seen by the repeater device (see Figure 30 for an illustration).

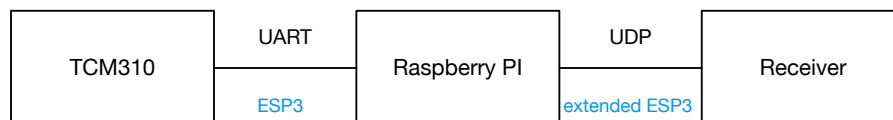


Figure 30: Forwarding of TCP310 EnOcean Telegrams via UDP

In addition to the raw ESP3 telegram the extended ESP3 telegram submitted to the UDP receiver includes a time stamp as well as the most important info extracted from the ESP3 telegram like signal strength, sender and receiver information in human readable format.

[<prefix>]<Time> <RORG> <Sender> <Destination> <Signal> <Data> <RAW ESP3 Telegram>

Example - Data:

Time	RORG	Sender	Destination	Signal	Data	Packet
2020-01-31 10:58:30.559	a5	ffb66493	050e5291	65	28d5007a	a528d5007affb664938002050e529141001a
2020-01-31 10:58:30.592	a5	ffb66481	050d3f70	65	28d5007a	a528d5007affb664818002050d3f704100c5
2020-01-31 10:58:30.616	a5	ffb66480	050e5291	67	28d5007a	a528d5007affb664808001050e5291430058

A detailed list of possible parameters for the forwarding script can be seen in Table 14.

Table 14: Description of parameters for the repeater script.

Parameter	Description
serialPort	The UART port to which the TCM310 module is attached to
ipAddress	The target IP address or link-local name of the receiver of the telegrams
udpPort	The UDP port on which the receiver is listening for telegrams (default 34505)
prefix	The prefix to be prepended in front of the raw telegram data (e.g. to identify the repeater)
repeaterMode	Defines whether the repeater should operate as level 1 or level 2 EnOcean repeater
signalLevel	Only telegrams with a signal level weaker than this threshold should be repeated

This repeater was built as a helper tool for setting up the GREAT system in demanding environments (long distances, thick walls between the controller and luminaires). However, it could also be useful in a context outside of GREAT, whenever EnOcean installations should be debugged or monitored.

8. Sensors

8.1. PIR Sensor

PIR (passive infrared) sensors are used to detect motion. They return a simple boolean value, indicating whether the sensor detects moving persons or not. Changes to this state are communicated via an EnOcean interface to the Intefox controller and are logged for further analysis. See Table 15 for the specification of the sensor used.

Motion detection: Standard Thermokon PIR EnOcean, battery powered

Manual switches: Standard EnOcean rockers

Table 15: Thermokon "EasySense" SR-MDS BAT specification

Vendor	Thermokon Sensortechnik GmbH (Germany)
Series	EasySense
Type	SR-MDS BAT
Technical design	Wireless
Wireless technology	EnOcean ISO/IEC 14543-3-10
Radio frequency	868,3 MHz
Functions	Motion detection and brightness measurement
Motion detection	Passive infrared
Detection area	360°; 105° conical (ceiling installation)
Detection radius (2,5 m room height)	3,25 m
Power source	3 x Battery 3,6V 1/2 AA LS14250
Brightness (Accuracy)	0-510 Lux (+/- 30 Lux)
Sleep time interval (modified)	1s – 1000s (for GREAT set to 1 second)

8.2. Biovotion Everion Sensor

The Everion device is worn by one caregiver at each field test location. The device is worn at the upper arm and can measure either raw data and/or vital parameters. The raw data mode produces a huge amount of data and is used only for research and development purposes. The vital parameters are calculated by functions on the sensor device itself and cause less data and traffic to handle. Currently the following vitals can be measured and streamed with a frequency of 1Hz:

- Heart rate*
- Blood Oxygenation or SPO2*
- Skin temperature*
- Skin blood perfusion*
- Steps / Motion*
- Blood pulse wave
- Heart Rate Variability

- Activity
- Respiration Rate
- Energy Expenditure
- Electrodermal activity/ galvanic skin response
- Barometric pressure

*) clinical grade

Since we are mainly interested in stress detection, the company did implement a stream to gather the IBI (Inter-Beat-Interval in ms) as shown Figure 31. The IBI is used to calculate the HRV (Heart Rate Variability) as the HRV itself an important parameter to detect stress.



Figure 31: Heart Inter-Beat-Interval

8.3. Stress Detection

For the stress index calculation, we rely on results from a past research project where individual calibrations were made during a learning phase. These individual adaptations were based on personal feedbacks of the test persons. These are missing in this project and we had to analyse how the quality was influenced.

There are no standard values or threshold recommendations for some physiological parameters (esp. heart rate variability HRV). The approach from earlier projects to identify individual ranges between low and high stress based on personal user feedback was not possible within the GREAT setup. We therefore needed to develop algorithms to identify stress thresholds based on historical data und on smart combinations of additional data based on analytical approaches and data mining approaches (e.g. plausibility calculations, combination physiological data with PIR data, time of the day, accelerometer activity profiles etc.).

An additional challenge in this project was to find the shortest timeframe with enough data to allow a qualitatively good detection of stress. Figure 32 depicts stress timeframes of 1 hour during the day. In our feedback loop we continuously measure and calculate the relative stress of the caregiver:

$$\text{Relative highstress within timeframe} = \frac{\# \text{ stress index values} > 2}{\# \text{ stress index values}}$$

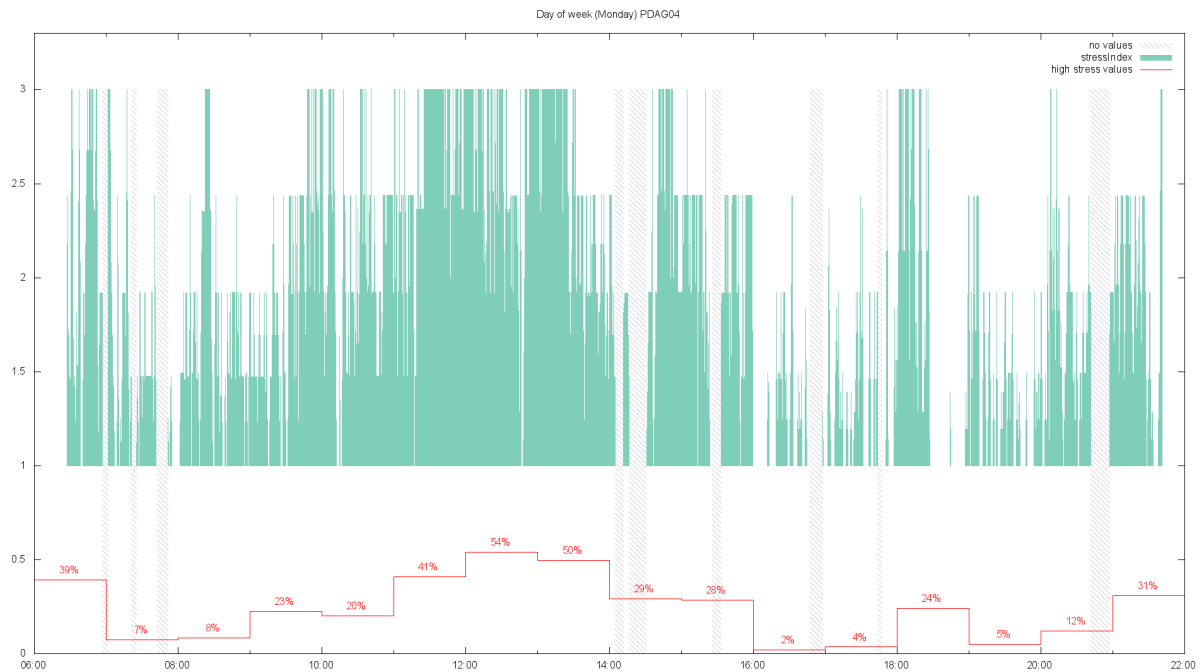


Figure 32: Stress index values and 1-hour phases

8.3.1. Architecture

Customers and users of the Everion device use an App (Android or iOS) to receive and analyze vital data. For our purpose however, we must further process the data and can therefore not use this software directly. For our purpose, we use software provided by Biovotion for research and development tasks. One is the VSM tool to configure the sensor and another one is the Streamer tool to continuously stream data. Unfortunately, both tools are only available for Windows. To keep costs low and to fit into the hardware family used within the GREAT system we tried several setups with the Raspberry Pi as the hardware device receive streamed data from the Everion sensor. The most successful option was to use an Android version (Emteria) for the Raspberry Pi and to adapt the Android app of Biovotion. The source code was provided by Biovotion and first adaption were successful. But for a complete adapted software version to many development tasks were left. Therefore, we did decide to freeze these efforts and continue with the Intel Compute Stick version and Windows 10 home as shown in Figure 33.

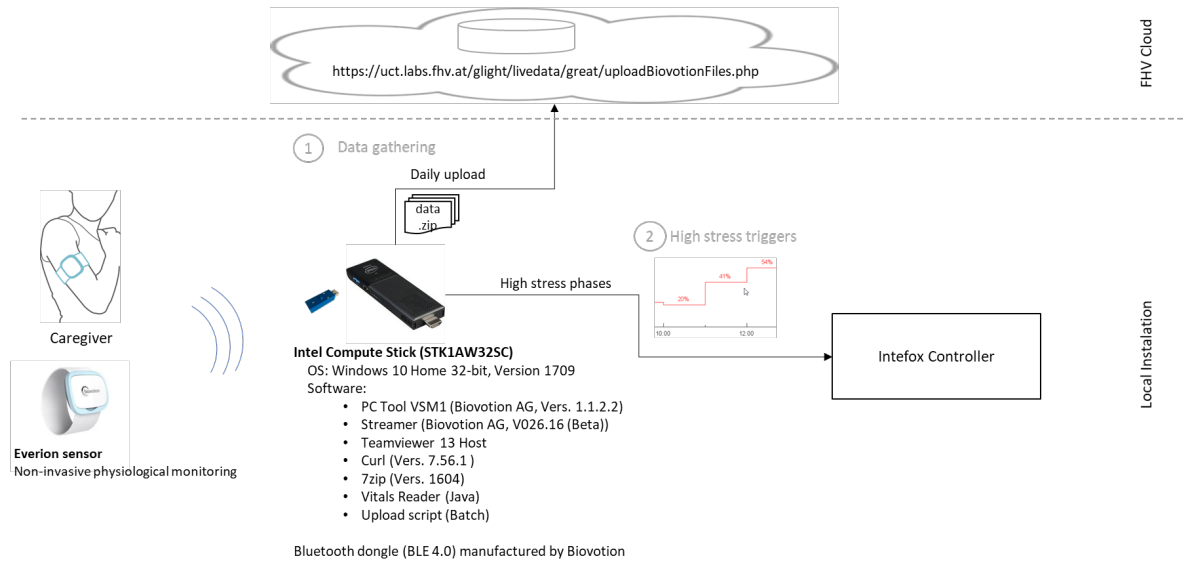


Figure 33: Everion sensor setup

8.4. Gathering Vital Data (test phase 1)

In the test phase, we gathered vital data (1) for further analysis.

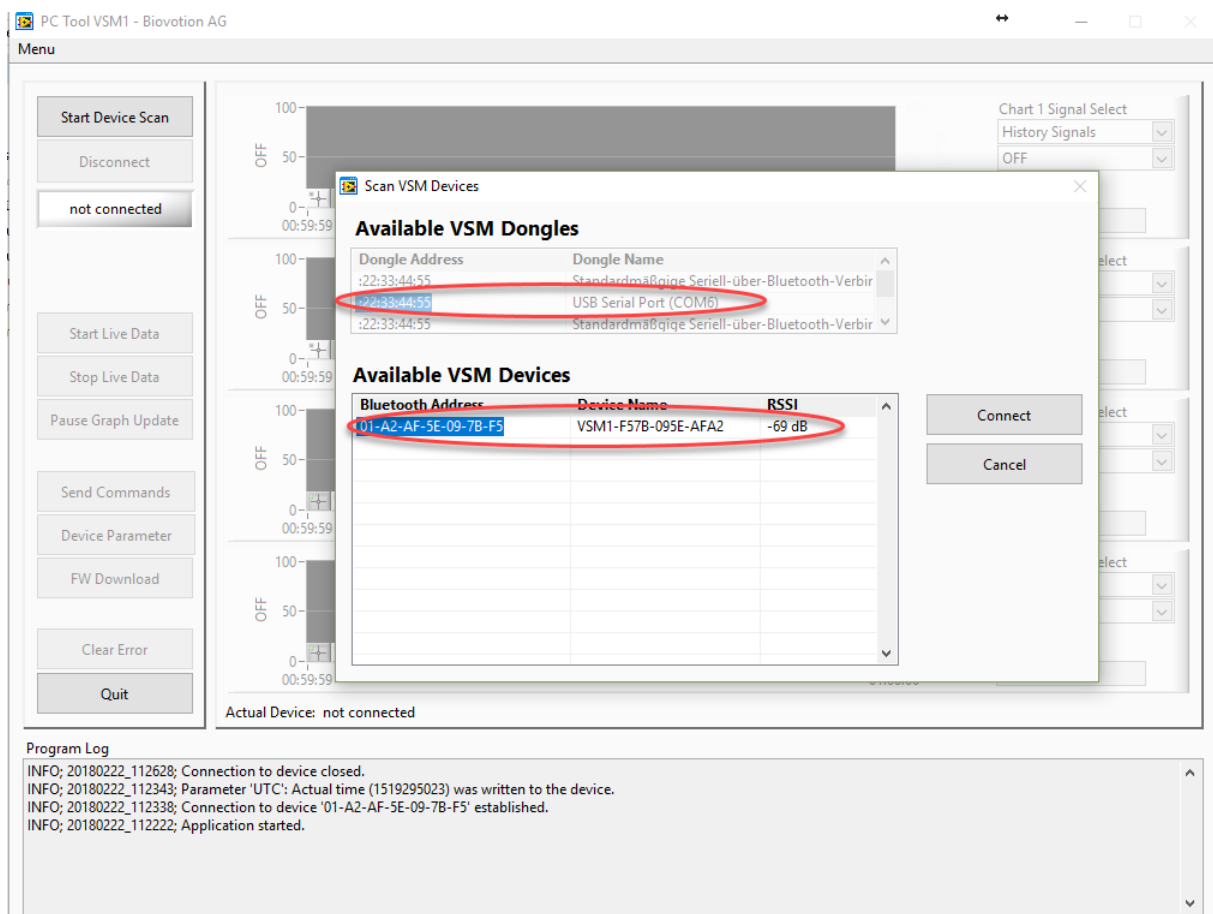


Figure 34: PC Tool VSM1 pairing with sensor

8.4.1. Setup

Follow these steps to initialize the Everion for vital parameter measurements and streaming.

1. Put the Everion device on the charger
2. Connect the Bluetooth dongle from Biovotion with the PC
3. Start the PC Tool VSM1
4. Start scanning the COM port belonging to the dongle
5. The device with its MAC address and the connection signal strength should appear in the list as shown in Figure 34. Select the sensor and connect.
6. Under the menu item *Device Parameter*, the sensors configuration can be changed or just read. The most important parameter is the algo mode. The following modes are available:

0=VITAL MODE (Light Skin Mode)

- 1=VITAL_CAPPED_MODE (SPO2)
- 2=HR_ONLY_MODE (Dark Skin Mode)
- 4=RAW_DATA_VITAL_MODE
- 5=RAW_DATA_HR_ONLY_MODE
- 6=SELF_TEST_MODE
- 7=RAW_DATA_OFF_MODE
- 8=RAW_DATA_FAST (64 Hz)
- 9=MIXED_VITAL_RAW
- 10=VITAL_MODE_AUTO_DATA
- 11=GREEN_ONLY_MODE (Battery Saving Mode)
- 12=RAW_DATA_FIX_CURRENT
- 13=SHORT_SELF_TEST_MODE
- 14=MIXED_VITAL_RAW_SILENT

Depending on the algo mode the memory on the device is restructured. Therefore, this might take around a minute. The modes differ regarding the data measured and collected. For our use case we use the mode 0 (vital mode resp. light skin mode).

Galvanic skin response (GSR_ON) shall be set to 1.

The Table 16 lists the vitals measured and streamed in algo mode 0.

7. Close PC Tool VSM1
8. Open Streamer tool and set the configuration as depicted in
9. Close the tool or run the streamer if you want to start the measuring

Table 16 Vitals of light skin algo mode

Stream	Len	Type	Counter	Timestamp									
		0	1-4	5-8	9	10	11	12	13	14	15	16	17
Algo1	19	9		T(u)	HR(u)	HRQ(u)	SpO2(u)	SpO2Q(u)	PI	Act. Class	Act.	Act. Class Q	Step (u)
Algo2	15	10	C(u)	T(u)	HRV(u)	HRV Q (u)	RR (u)	RR Q (u)	Energy (u)	Energy Q (u)			
Raw board	19	17	C(u)	T(u)	Impedance low byte (u)	Impedance high byte (u)	Local Temp	Local Temp	Obj Temp	Obj Temp	Bar. Temp	Bar. Temp	m bar
IPI DB	10+ N*2	22	C(u)	T(u)									

Table 17: Value specifications

Name	Def	Data Type	Bytes	min (received)	max (received)	[offset; conversion factor]	Description
Heart Rate	value quality	uint8 uint8	1 1	30 0	240 100	1 1	[Bpm] [%]
SpO2	value quality	uint8 uint8	1 1	60 0	100 100	1 1	[%] [%]
Blood Pulse Wave	value quality	uint8 uint8	1 1	0 0	255 100	1/50 1	without units [%]
Perfusion Index	value quality	uint8 uint8	1 1	0 0	255 100	1/50 1	[% swing] [%]
Activity/Motion	value	uint8	1	0	255	100/255	without units
Activity Classification	value quality	uint8 uint8	1 1	0 0	255 100	enum 1	undefined 0 resting 1 walking_flat 2 running_flat 3 biking_flat 4 walking_up 5

							running_up 6 biking_up 7 rowing 8 other 9 biking 10 running 11 walking 12
Steps	value	uint8	1	0	255	1	[steps/second]
Energy Expenditure	value	uint8	1	0	255	2	[cal/s]
	quality	uint8	1	0	100	1	not impl
HRV	value	uint8	1	0	255	1	[ms] (rMSSD)
	quality	uint8	1	0	100	1	[%]
Respiration Rate	value	uint8	1	0	255	1	[Bpm]
	quality	uint8	1	0	100	1	[%]
GSR-Sensor	value [Impedance]	uint16	2	0	65535	1/3000	ampl[kOhm]
Inter Pulse Interval (IPI)	value	uint16					
		B12-15		0	15	100/15	Quality value [%]
		B0-11		0	4095	1	Time in [ms]

The Inter Pulse Interval (IPI) value is a 2-byte value whereas the first 4 bits provide the quality and the remaining 12 bits the interval in milliseconds.

Example:

Entry in CSV data file:

```
22,240840,2017/12/08 22:53:31,,62333,
22,240841,2017/12/08 22:53:31,,62382,
22,240842,2017/12/08 22:53:31,,58281,
```

Leads to the following IBI values

```
62333 : IPI_Q 15 / IPI 893 mS
62382 : IPI_Q 15 / IPI 942 mS
58281 : IPI_Q 14 / IPI 937 mS
```

The sum over all IPI values of one day should always be 86400 seconds. The timestamp is set only once for each page therefore the counter should be considered for the order.

Table 18: Quality value specification

0	No IPI values detected, the time will be filled artificially to reach the 86400 seconds over a day.
1-7	Quality not sufficient (equal to the <50 quality values of vitals)
8-15	Quality is good

8.4.2. Pairing Malfunction Tips

If the device does not appear in a list of the PC Tool VSM1 or the Streamer, then close the software, remove the dongle and plug it in again. This causes a **reset of the dongle**.

If connection problems receive, it might be due to a pairing problem. A pairing issue arises also, if the device was connected to a smartphone and shall be connected to the PC or visversa. In all this case **unpair** the device with the following steps (all data will be deleted):



1. Put the device on charger (If it is on the charger already, remove it, wait until the device vibrates twice and put the device on the charger again)
2. Wait, until the blue led is off (now you have 30 seconds to proceed with step 3)
3. Press the button until the device vibrates (as on the picture below)
4. Remove the finger
5. First, a double vibrate indicates that the unpairing was successful. (If there is no double vibration repeat the steps 1-4), then data deletion is executed. It will take approx. 40 seconds until your able to reconnect with a mobile device.

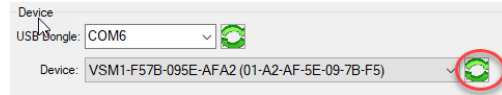


8.4.3. Run Data Gathering

The measuring of vital data can be started with the Streamer tool:



1. Start the tool and open the settings.
2. Pair (connect) it with the sensor by selecting the correct COM port and pressing the green arrows of the device until the device MAC address appears and is selectable.



3. Save settings
4. Then start measuring



5. Check if the pairing and streaming start was successful



6. Every time the device is worn at the upper arm, the measurement continues and the tool streams the values into a file in the users documents and VSM data folder.

Data file example:

```
9,137277,2018/02/22 15:36:49,,75,82,60,0,10,1,2,98,0,65,
10,137277,2018/02/22 15:36:49,,56,53,16,83,16,100,
17,137277,2018/02/22 15:36:49,,231,219,3028,2978,2950,9285,
22,219684,2018/02/22 15:33:48,,62315,
```

The filename follows the pattern as shown in Figure 35. The device id follows the pattern VSM1-<MAC-Adress>.

7. Create a Windows task to run the upload batch file every evening. The files of the day will be zipped and uploaded for further analysis to the cloud server of FHV.

uplodNewFiles.bat

```
forfiles /s /m "*.txt" /d %date% /c "cmd /c 7z a -tzip @FNAME.zip @PATH"
for /r %%f in (*.zip) do (
    curl -v -F "greatUserID=great" -F "greatPassword=<password>" -F
    "upfile=@%%f"
    https://uct.labs.fhv.at/glight/livedata/great/uploadBiovationFiles.php
)
del /s *.zip
```

Precondition: 7zip tool and curl installed.

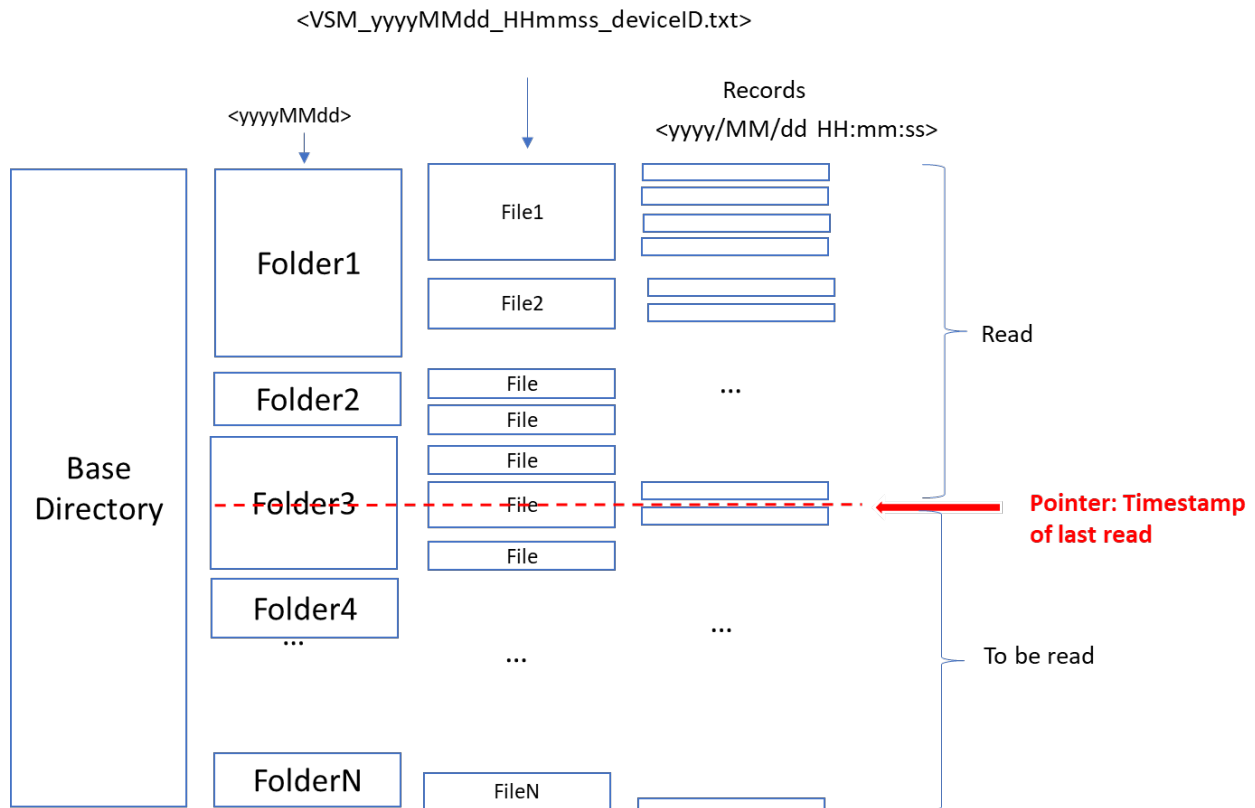


Figure 35: Data file structure

8.4.4. High Stress Triggers

The collected biodata was analyzed together with the systems log files and feedbacks given by the caregivers. We assume to derive a pattern for the stress detection and possible triggers to provide a suggestion for interventions to the caregivers.

The stress events or triggers are then sent to the intefox fox.core controller for further dispatchment. To allow for a continuous measurement and stress calculation, a small Java program has been created, that constantly reads vitals from files created by the Biovotion Streamer and forwards information to the fox.core system.

8.4.5. Integration into the GREAT System

To directly pass information from the Biovotion system into the GREAT system, a system extension for the fox.core has been developed, that receives data from the Biovotion system via a TCP stream. Heart rate statistics currently supported are heart rate average and average/maximum/minimum/standard deviation of RMSSD as a measure for heart rate variability. Each data transmission also includes a timestamp. Information is encoded in key-value-pairs delimited by commas. Also stress events can be forwarded to the GREAT system using the same interface.

The GREAT system extension takes data from the incoming TCP stream and makes the individual channels available as events for connections inside the GREAT system as well as for logging (see Figure 36).

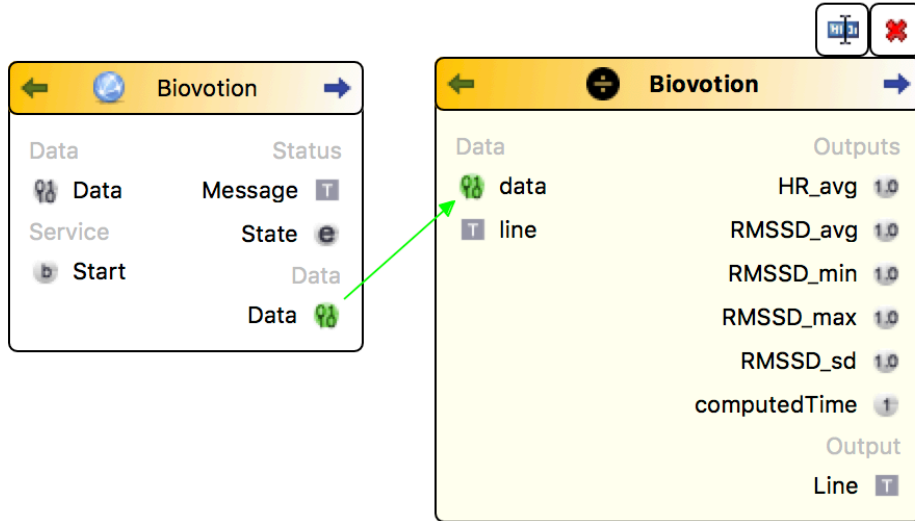


Figure 36: Integration of Biovotion HRV statistics into the GREAT system.

9. User-Interface for Manual Control

For the functional testing period a remote-control tool was created based on the existing Intefox mobile app that's available for iOS and Andorid devices. The configuration was created to allow for an easy control of the separate light/scent/sound-modules.

Figure 37 shows the main menu (left), the scent- (middle), and sound-module (right) offering control for starting an activation or relaxation session.

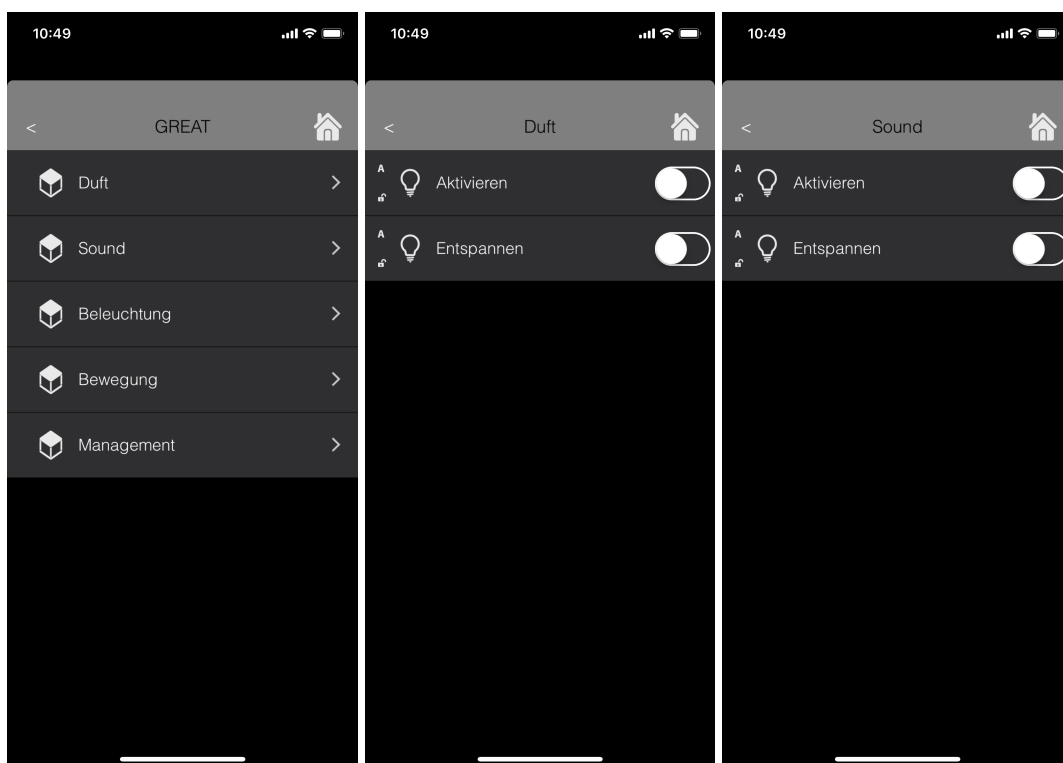


Figure 37: Screenshots of the main menu, the scent-, and sound-module offering control for starting an activation or relaxation session.

Figure 38 shows the control page for the light. When the light is switched on without further action, a biodynamic light is applied throughout the day. The interface allows to choose activation or relaxation interventions, as well as predefined light scenes for quickly applying a norm light or a scene fitting for watching TV. The second screen shows the status of the PIR sensor, delivering brightness values (in lux) and motion status.

This interface hosted by the Intefox app is only intended for manual control and test setups. See chapter GREAT User-Interface for the simplified interface used by actual users during the field test phase.

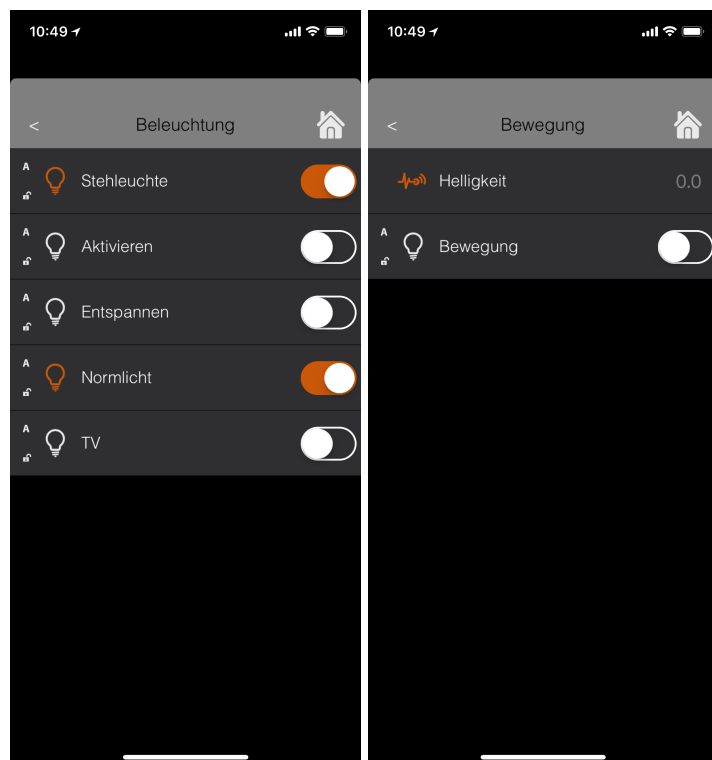


Figure 38: Screenshots of the light control page and the status view for the motion detector.

10. GREAT User-Interface

One of the main requirements for the user interface for the system was that the GREAT system should be easy to use by care giving personal or even elderly users of the system directly. This requirement could not be fulfilled with the standard Intefox mobile app. Therefore, a dedicated user interface for triggering activation and relaxation interventions has been developed.

The basic idea was to reduce the number of elements to the absolute minimum to make it clear where functionality is located and to guide through the processes of starting or modifying a specific intervention. During the project phase, several versions of the user interface have been developed in an iterative human centered design process.

The user client was implemented as a cross platform web application based on the Vaadin framework. It features a responsive design to allow for flexible use on a broad range of devices of different sizes. During the GREAT field tests, tablets have been used as main controlling device.

The start screen of the user client shows all available GREAT areas with buttons for launching activation or relaxation sessions. Depending on the configuration, also interface elements for additional functionality (like a special scene for norm light or tv watching) can be shown. See Figure 39 for an illustration of the maximum configuration and Figure 40 (left side) for the start-screen setup used during the GREAT field tests.



Figure 39: GREAT user client main screen with all elements enabled.

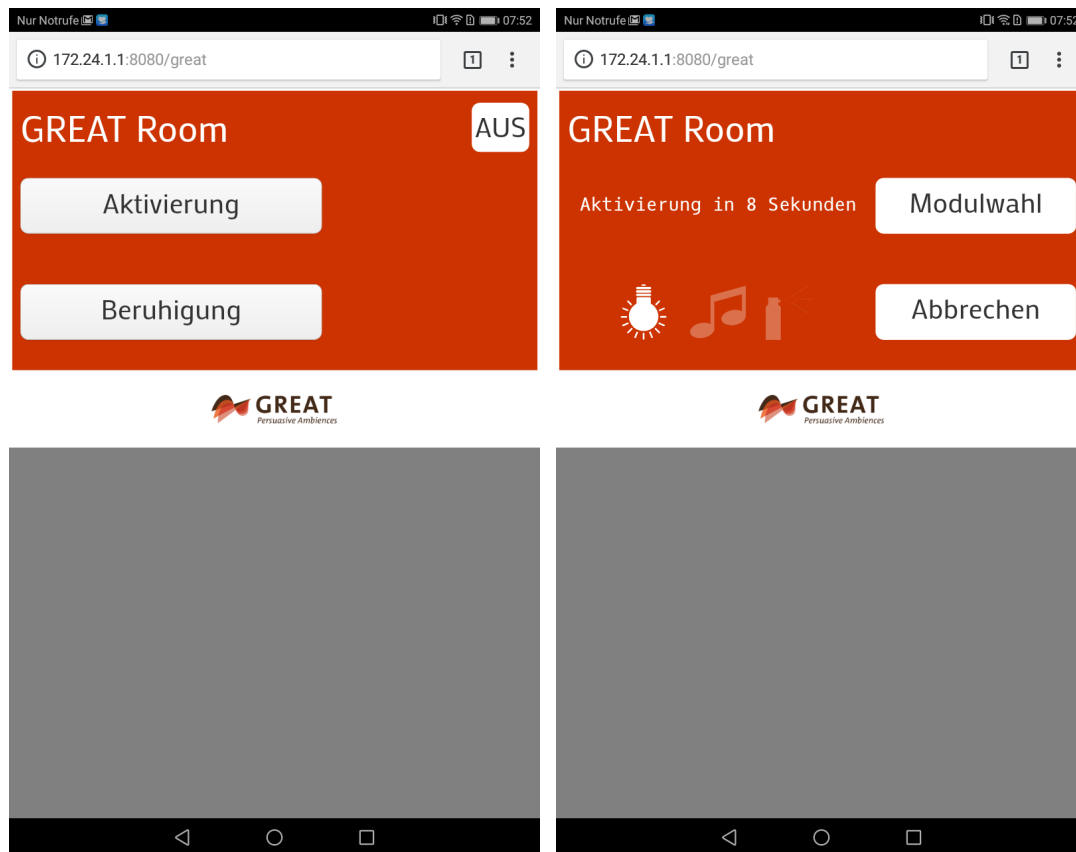


Figure 40: GREAT user client: starting an intervention

After an intervention is selected, it is possible to select or deselect available light/scent/sound modules individually for the intended intervention. If no action is taken, the default setting will be used automatically after a timeout. In this way, only one button touch is required to start an intervention with the default set of modules. Figure 41 shows the customization screen on the left and the status feedback screen on the right, where it can be seen, which modules are active and for how long the currently active intervention will run. Note that there is also always an option to stop any ongoing intervention preliminarily by the user at any time.

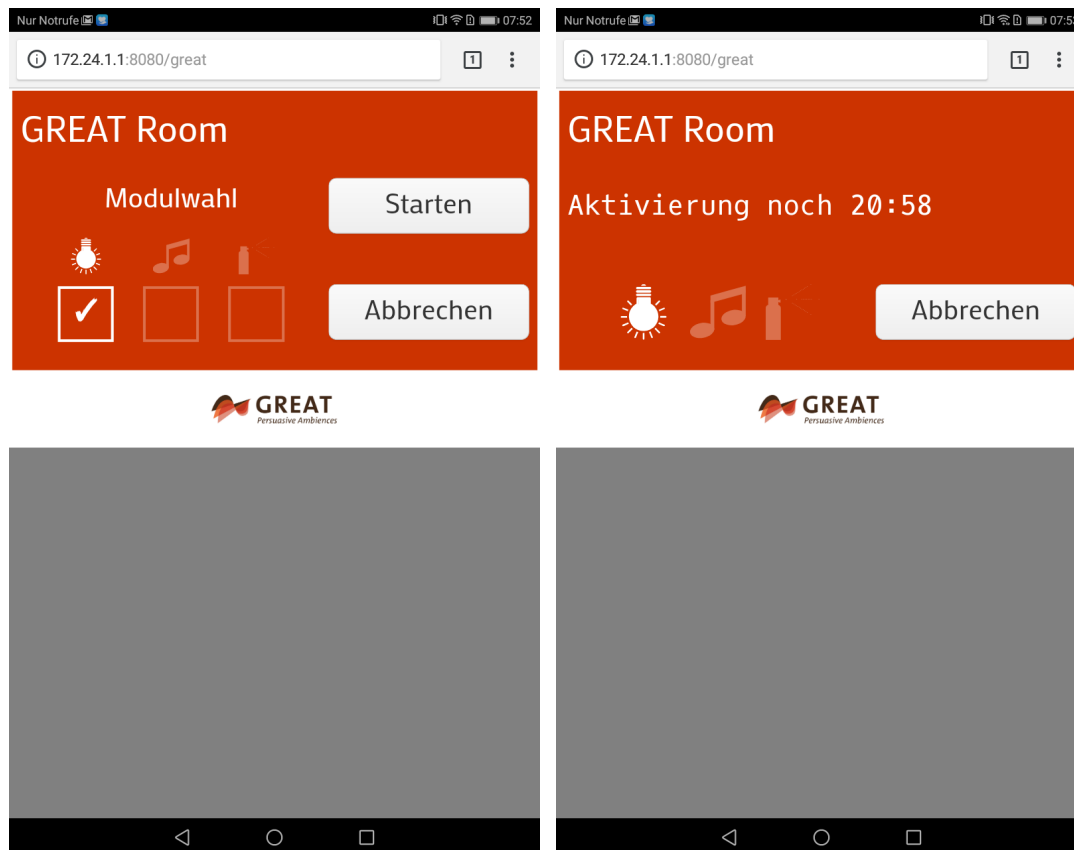


Figure 41: GREAT User client with customization options before the intervention (left,) and status feedback and cancel option during an ongoing intervention (right)

After the intervention has completed, users get the possibility to respond with a feedback, whether they think the intervention helped or not. It also provides an additional text input field to submit further information or comments (see Figure 42).

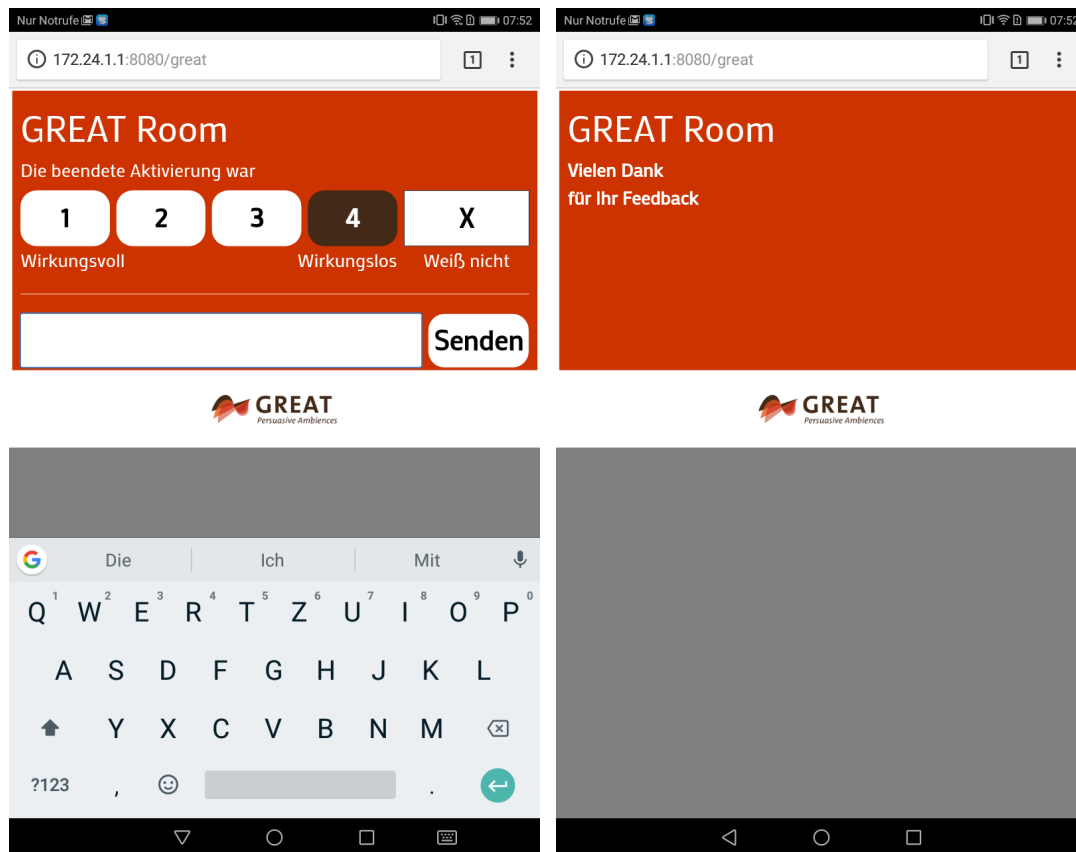


Figure 42: GREAT User client: after an intervention (asking for feedback)

Configuration - Setup (fox.configurator)

The configuration of the user interface can be done via the fox.configurator software. Note that for the final GREAT product, the configuration is envisioned to be created automatically based on which modules are installed (see chapter Setup Procedure of the Final GREAT Product for the envisioned setup process of the final product).

The manual steps involved in setting up a web-based visualization client:

As a first step a user for the great user client and a visualization page needs to be added. The added user can then be assigned to the visualization page created (see Figure 43).

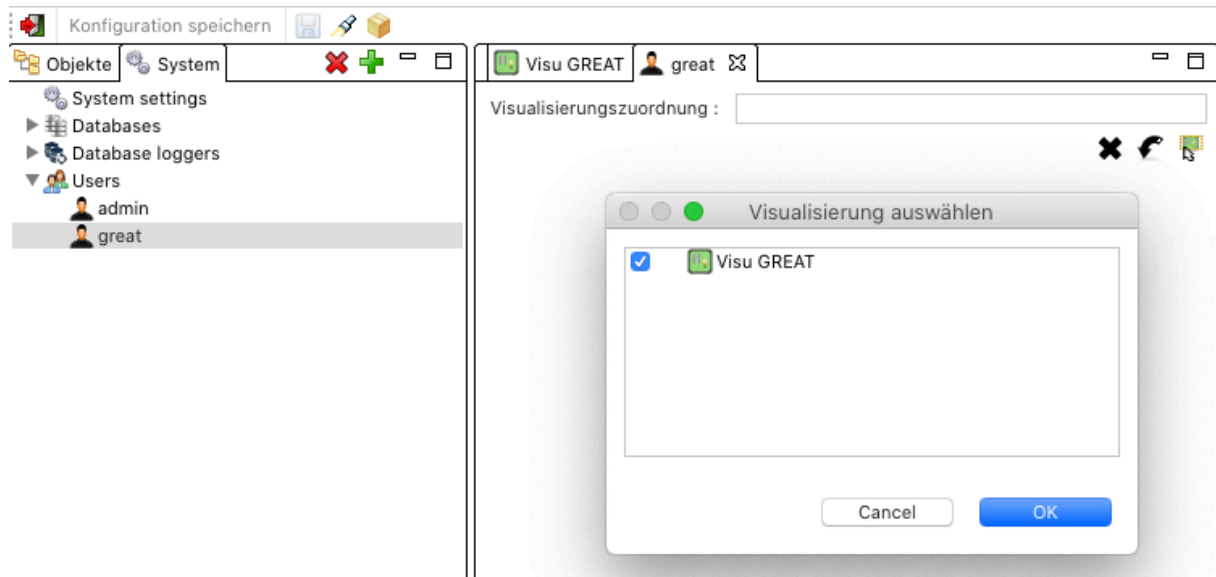


Figure 43: Assigning a user to the GREAT user client

In a next step available GREAT room objects can be added to the visualization page by dragging and dropping them onto the visualization icon. All room objects that are added in this way, will then be shown in the visualization (see Figure 44).

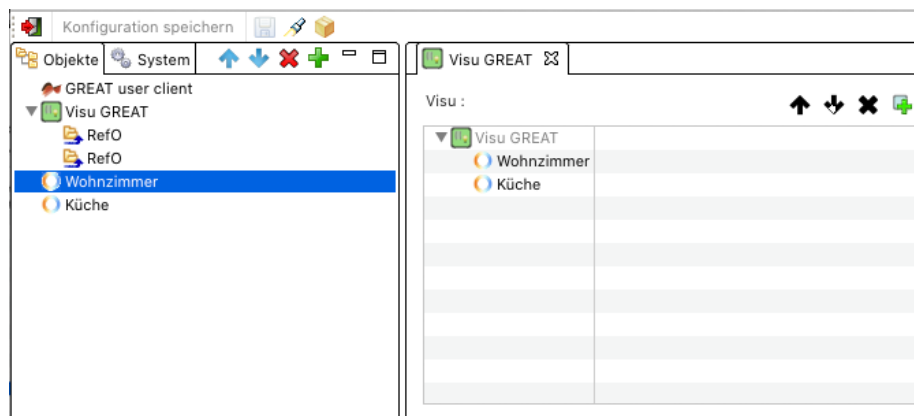


Figure 44: Assigning a Light curve object to the GREAT user client

Then a GREAT user client object needs to be added to the configuration. The previously created user can then be assigned to this object. By specifying a path over which the client can be reached, multiple instances of visualizations could be created for different users (see Figure 45).

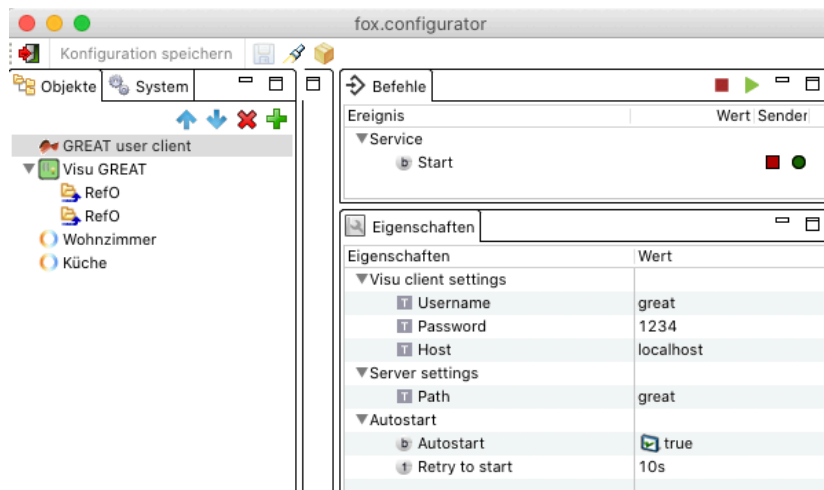


Figure 45: GREAT user client settings

In the example shown above, the user client would be accessible in the web browser by the address:

<https://<ip-of-controller>:8080/great>

This bookmark can then be saved to the homescreen of Android or iOS devices, resulting in a GREAT Icon on the homescreen by which the application can be launched easily, without having to type in the address repeatedly. The application will then also be shown in fullscreen, without any potentially disturbing browser controls.

Finally, to customize available options visible in the user client, modules and special light scenes can be activated or deactivated in the settings for the GREAT room object (see Figure 46).

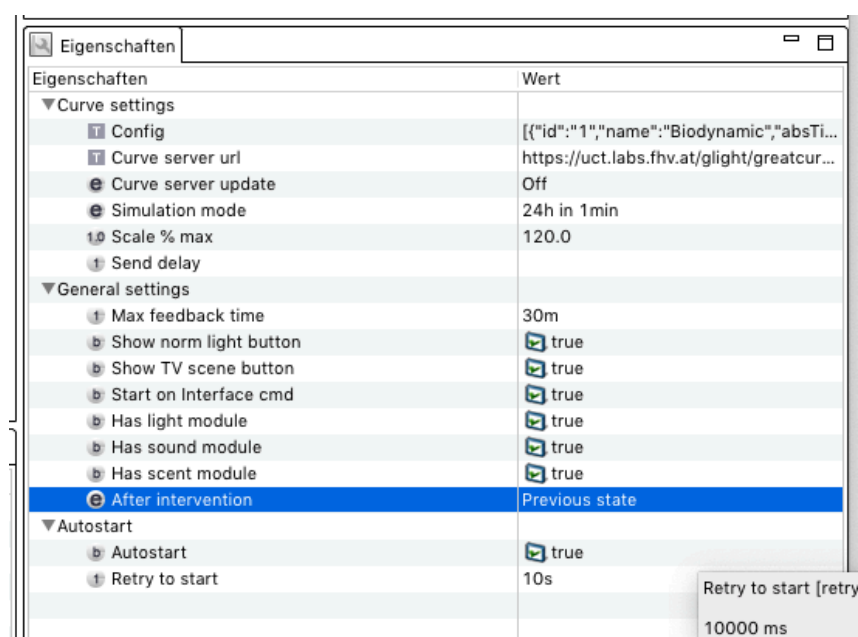


Figure 46: GREAT user client room settings of enable/disable modules and buttons

11. GREAT Manager

The GREAT Manager is a web-based user interface for expert users, that allows for customizing stimuli used by the GREAT system for activation and relaxation. The configurations are stored by a backend system that communicates with the local GREAT system via REST APIs.

The system is comprised of a Curve-Editor, a Playlist-Editor, a Schedule Editor and a Data-Download area that allows for offline analysis of field test data (see Figure 47 for an overview).

The GREAT Manager features a fine-grained permission control system (see Figure 49). This allows for limiting access for users to certain installations – meaning users can only edit configurations of installations officially managed by them. Also, the offered features can be limited to certain aspects on a per user basis (e.g. a user could only be allowed to adjust his playlists but wouldn't be able to edit any schedules).

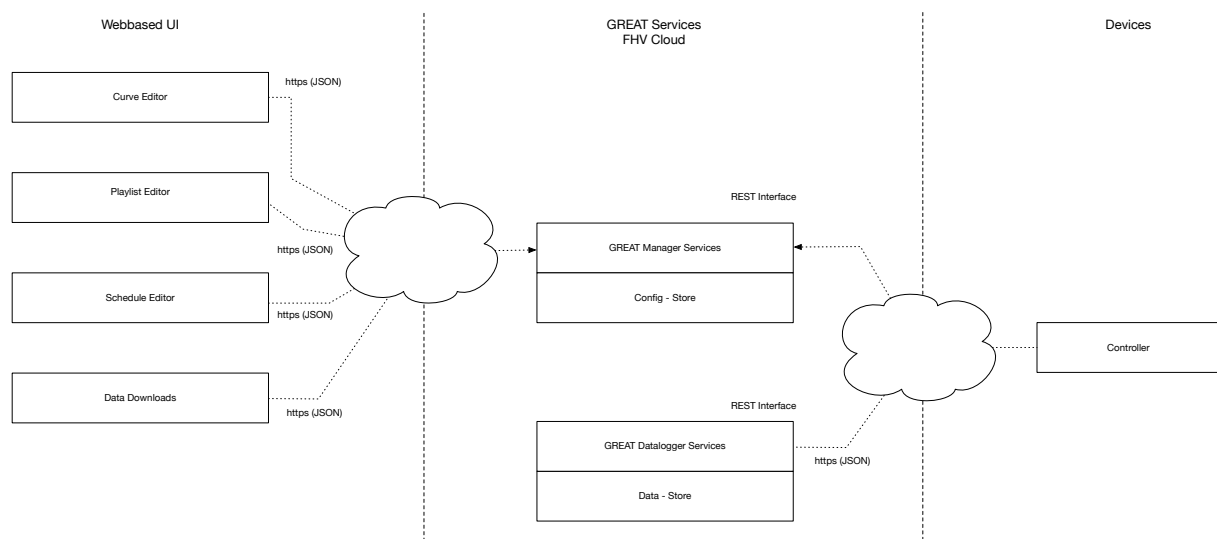


Figure 47: The GREAT Manager system structure consisting of a web-based front end for administration, the backend services for configuration/data persistence and the local GREAT devices communicating over a REST API with the backend.

The GREAT management web interface is built on top of an open source Apache / PHP / MySQL software stack. The GREAT management backend stores configuration data in the data model shown in Figure 48.

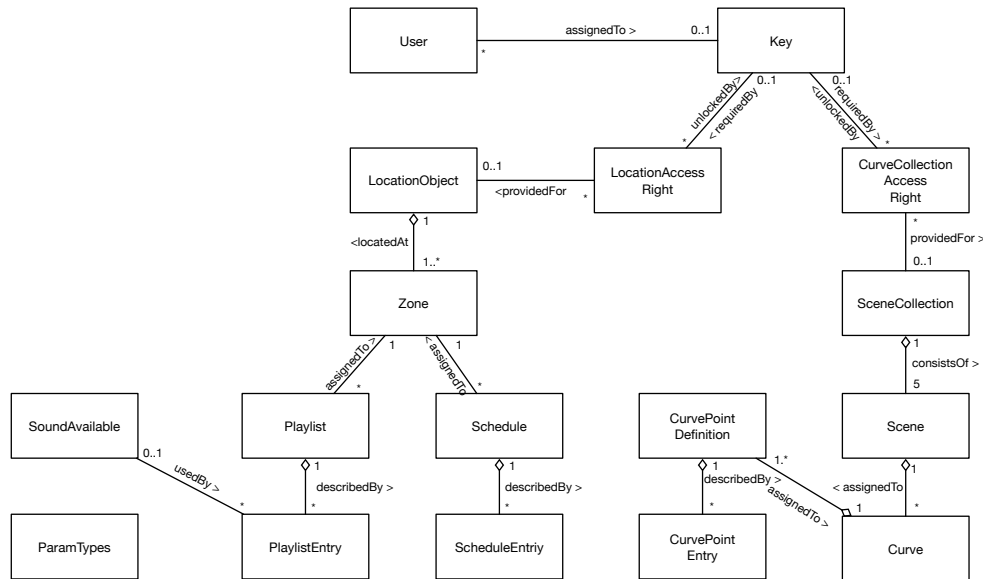


Figure 48: Data model of the GREAT Manager backend

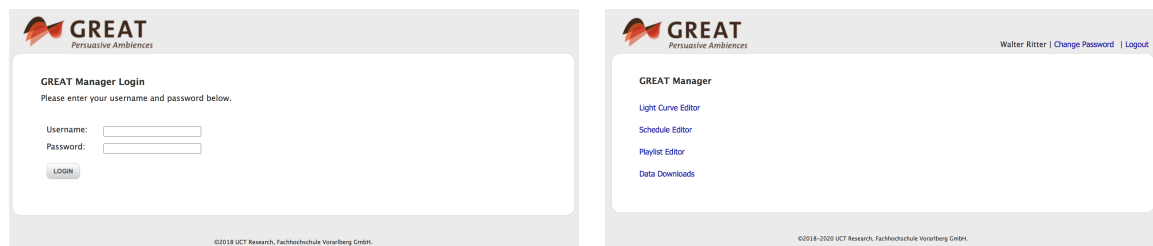


Figure 49: Login and main menu of the GREAT Manager interface

The GREAT Manager (see Figure 49 for the basic functionality) also keeps track of the history of configurations. This history is mainly used as input for analysis of field test data but can also be used to easily reactivate previous configurations again. The editors also allow for defining new configurations that will be activated at a specific date in future.

11.1. Curve Editor

The curve editor allows for customization of the light curves used for activation and relaxation sessions, as well as those for biodynamic lighting and static scenes like TV light or Norm light.

Light curves are defined by a list of values of color temperature, brightness direct level, and brightness indirect level at specific points in time. While the biodynamic curve is defined over a period of 24 hours, activation or relaxation curves are defined for specific durations like for example 21 minutes (activation) or one hour (relaxation).

The values of the curve can either be manipulated by dragging curve points inside the diagram, or by entering values into the table. To define a curve that should be

activated on a specific date in future, an effective date can also be specified for the curve (see Figure 50 for the lightcurve editor interface).

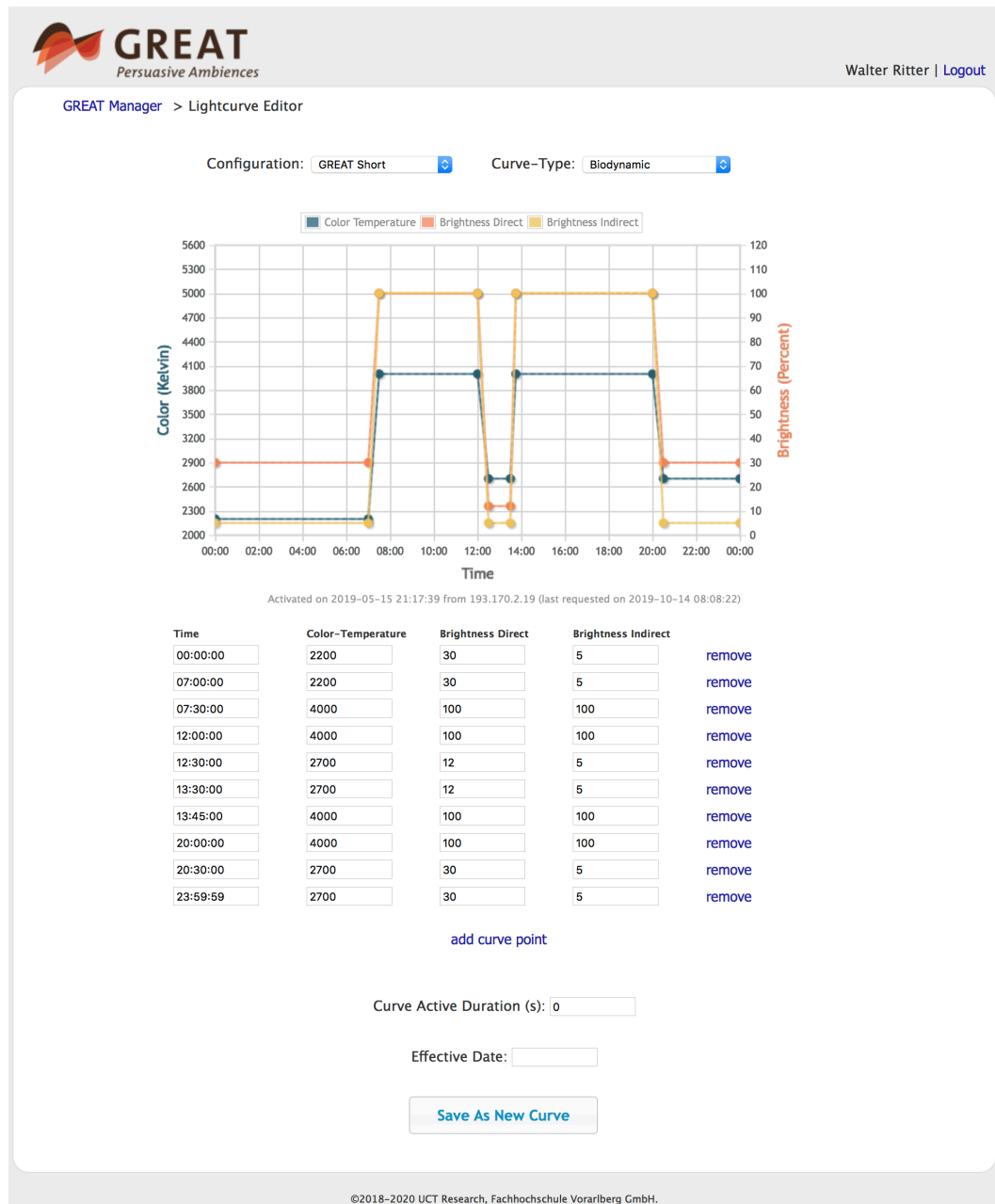


Figure 50: The lightcurve editor screen allows for customizing light curves

Note that a curve can only be saved as a new curve. All previous versions are retained by the system for documentation purposes of the field test configurations.

11.2. Schedule Editor

The schedule editor allows for defining time-schedules when certain interventions should or may be triggered automatically by the system. Either interventions are directly triggered at defined points in time, or a time frame is configured within which

the system is allowed to perform fully automatic interventions (e.g. based on a learned motion activity profile).

GREAT
Persuasive Ambiances

Walter Ritter | [Logout](#)

GREAT Manager > Schedule Editor

Location: (A) Hall Klinik Zone: Aufenthaltsbereich

Time	Days	Light	Sound	Scent	Condition	Brightness	
12:00:00	All	Relax	Relax	Relax	Always	0	remove
14:00:00	All	Activate	Activate	Activate	Always	0	remove

[add schedule time](#)

Effective Date: 2019-07-01

[Save As New Schedule](#)

Schedule-History: 3 (first requested on 2019-07-09 10:37:11) [load as starting point](#)
Activated on 2019-07-09 10:37:11 from 193.170.2.19 (last requested on 2019-07-09 10:37:11)

©2018–2020 UCT Research, Fachhochschule Vorarlberg GmbH.

Figure 51: Schedule editor for creating schedules when certain interventions should be triggered.

Schedule entries can also be attached to a certain condition. Currently the following conditions are supported:

- Always
- Never (entry is deactivated)
- If presence (triggered only if presence is detected at that time)
- If no presence (triggered only if no presence is detected at that time)
- On next presence (triggered the next time presence is detected)
- On next no-presence (triggered the next time when no presence is detected anymore)
- If recommended (triggered when the system deduces that it would be appropriate)

Schedules defined in this way are delivered by the schedule service to the local GREAT controller (see Figure 2). On the local controller, a developed OSGi based Schedule extension for the Intefox system is responsible for requesting and processing the schedule.

From the system perspective, the scheduler object is an alternative input for the remote-control input. Figure 52 shows available connections for the scheduler extension and its parameter settings. Table 19 lists the meaning of the inputs, Table 20 the meaning of the outputs and Table 21 the meaning of the parameters for the extension.

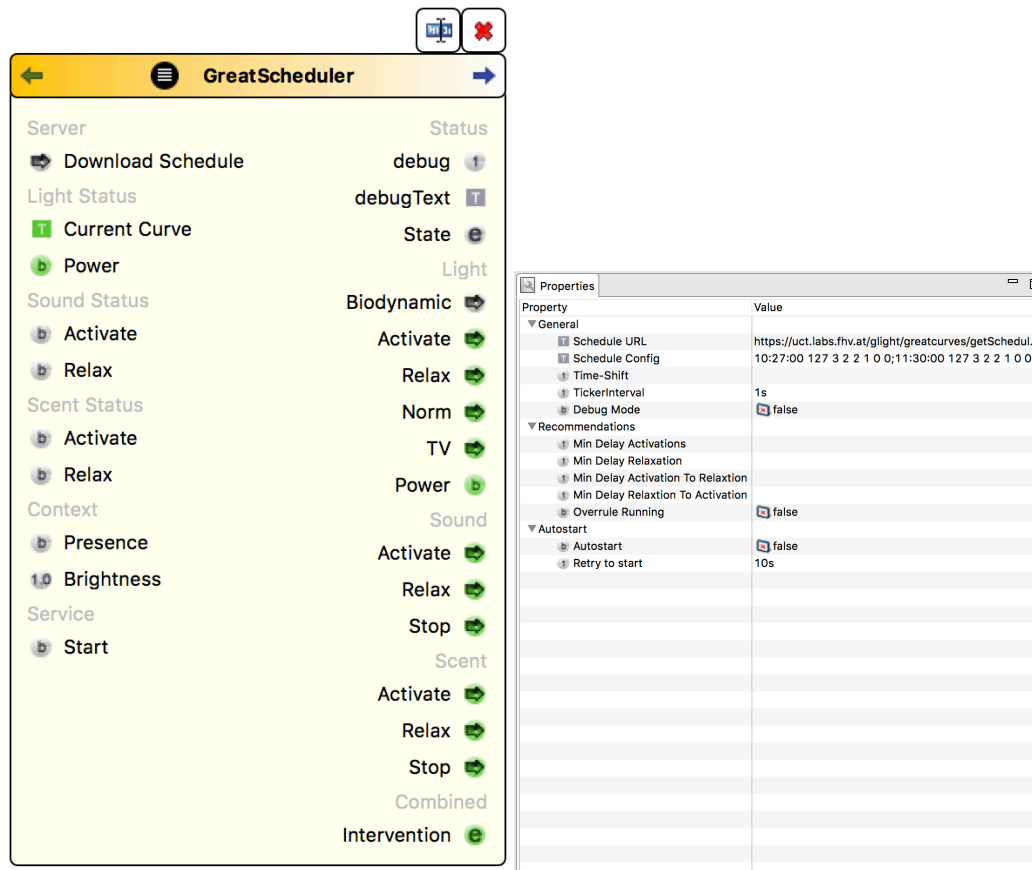


Figure 52: Inputs/Outputs and properties of the GREAT-Scheduler extension

Table 19: Inputs of the scheduler extension

Input	Description
Download Schedule	Tries to download the most recent schedule at the configured URL
Current Curve	The currently active light curve of the light
Power	The current power status of the light
Activate	The current state of Sound activation
Relax	The current State of Sound relaxation
Activate	The current state of Scent activation
Relax	The current state of Scent relaxation
Presence	The current presence state in the room
Brightness	The current brightness in the room
Start	Activate/deactivate the scheduler service (true/false)

Table 20: Outputs of the scheduler extension

Output	Description
--------	-------------

debug	Current state code for debugging
debugText	Debug text (only for development)
debugText	The currently active light curve of the light
State	Indicates if the scheduler is running
Light:	
Biodynamic	Indicates whether the biodynamic curve should be active
Activate	Indicates whether the activation curve should be active
Relax	Indicates whether the relaxation curve should be active
Norm	Indicates whether the Norm light scene should be active
TV	Indicates whether the TV light scene should be active
Power	Indicates whether the light should be turned on
Sound:	
Activate	Trigger to start the activation session on the sound module
Relax	Trigger to start the relaxation session on the sound module
Stop	Trigger to stop the current session on the sound module
Scent:	
Activate	Trigger to start the activation session on the scent module
Relax	Trigger to start the relaxation session on the scent module
Stop	Trigger to stop the scent current session on the scent module
Intervention	Combined intervention including light/sound/scent to connect to the GREAT node

Table 21: Parameters of the scheduler extension

Property	Description
Schedule URL	The URL from where to fetch the schedule
Schedule Config	The currently active schedule In a simple text format that can be copy/pasted.
Time Shift	Time shift in seconds if the whole schedule should be shifted by a specific amount of time
TickerInterval	The time interval within which scheduler actions should be checked (default: 1s)
Debug Mode	Flag to enable debug mode (default: false)
Min Delay Activations	The delay between activation sessions
Min Delay	The delay between relaxation actions
Min Delay Activation To Relaxation	The minimum delay between activation and relaxation sessions
Min Delay Relaxation To Activation	The minimum delay between activation and relaxation sessions
Override Running	Flag to indicate if the scheduler is allowed to override currently running sessions
Autostart	Flag to indicate if the scheduler should be automatically started
Retry to start	If the start of the scheduler fails, the time for retry interval

11.3. Playlist editor

The playlist editor allows for customization of playlists that should be used for activation and relaxation sessions. It offers officially supported sounds for selection, but also allows to specify custom sounds referenced by a URL (see Figure 53).

The screenshot displays the 'GREAT Manager > Playlist Editor' interface. At the top, the 'GREAT Persuasive Ambiences' logo is on the left, and 'Walter Ritter | Logout' is on the right. Below the header, the 'Location' is set to '(A) Hall Klinik' and the 'Zone' is 'Aufenthaltsbereich'. The interface is divided into two main sections: 'Activate-Playlist' and 'Relax-Playlist'. Each section has a 'Built-In' column with dropdown menus and a 'Custom URL' column with text input fields. In the 'Activate-Playlist' section, two 'GREAT Activate 30kHz Herz' items are listed, each with a 'remove' link. In the 'Relax-Playlist' section, two 'GREAT Relax 30kHz Atem' items are listed, each with a 'remove' link. Below these sections is the 'Playback Mode' section with three checkboxes: 'Random Order' (unchecked), 'Repeat' (checked), and 'Single Track' (checked). At the bottom, there is an 'Effective Date' field and a 'Save As New Playlist' button. A 'Playlist-History' section at the very bottom shows '3 (first requested on 2019-07-11 14:29:43)' and a 'load as starting point' link. The footer contains the copyright notice: '©2018-2020 UCT Research, Fachhochschule Vorarlberg GmbH'.

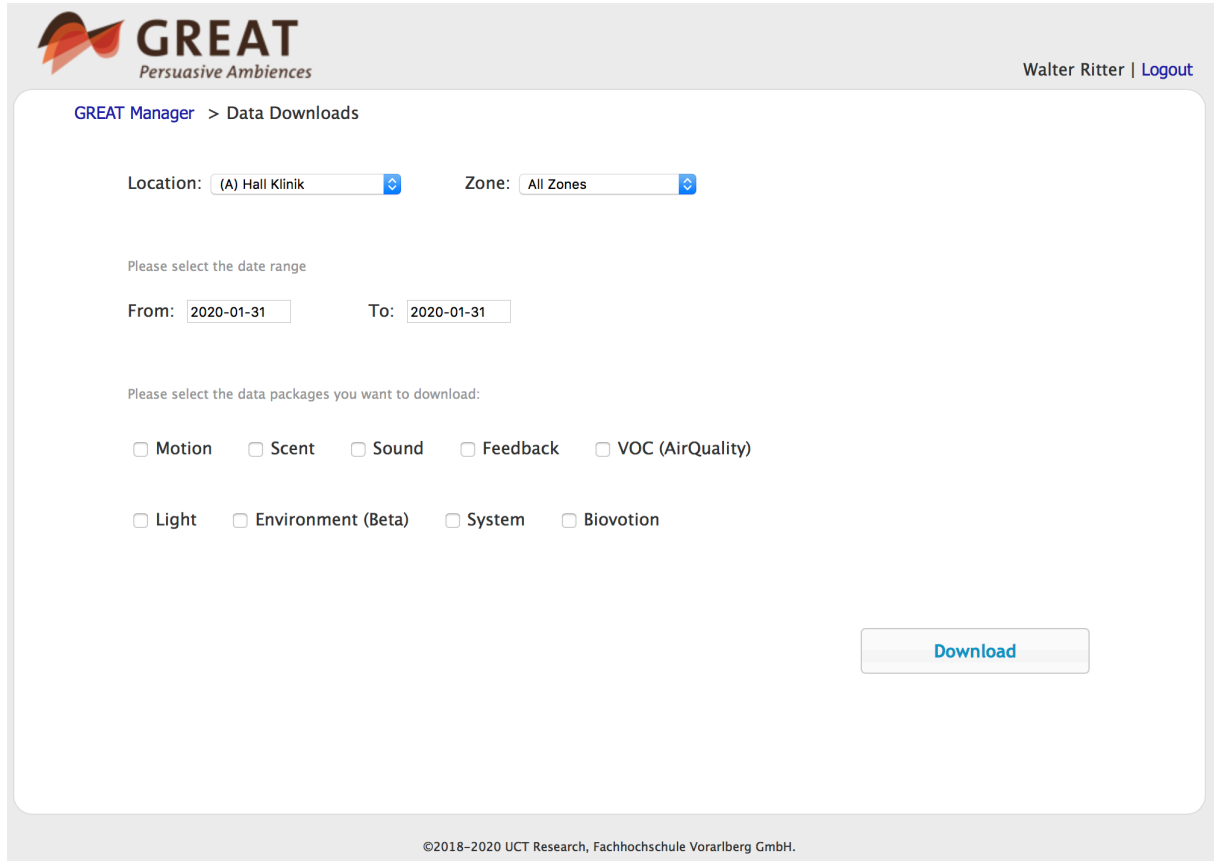
Figure 53: Playlist editor for customizing sound playlists for activation and relaxation sessions.

Also, the playback mode can be set in the interface (random order, repeating, single track repeat).

The sound module extension requests the most recent playlist either by an explicit command, or automatically at specified intervals. It then forwards the request to the sound module, which then downloads the playlists containing the sound references (see Sound Module Protocol Description). It then checks if all sounds of the playlists are already on the sound module. If not, sounds will be downloaded in the background to the sound module.

11.4. Data Download Area

The data download area is an interface to download specific data packages from the field test locations for offline analysis. It allows to specify the location, the zone, and timeframe to limit the size and focus of the download (see Figure 54).



GREAT
Persuasive Ambiances

Walter Ritter | [Logout](#)

[GREAT Manager](#) > Data Downloads

Location: (A) Hall Klinik Zone: All Zones

Please select the date range

From: 2020-01-31 To: 2020-01-31

Please select the data packages you want to download:

☐ Motion ☐ Scent ☐ Sound ☐ Feedback ☐ VOC (AirQuality)

☐ Light ☐ Environment (Beta) ☐ System ☐ Biovation

[Download](#)

©2018–2020 UCT Research, Fachhochschule Vorarlberg GmbH.

Figure 54: Data download area for downloading packages for offline analysis.

For now, the data download service is intended for field test use only. It provides a standardized way for accessing data for offline analysis. Every download is logged by the system.

12. Automated Control System

According to the concept of mood transfer, moods not only influence an individual person but may affect the behaviour of a social group as a whole. In the literature, we also find the terms “emotional contagion” or “group affect” (see e.g. [1, 2, 3, 4, 5]). The terms emotion, affect, and mood are often used interchangeably, but they refer to different states of feeling. Emotions are specific feelings that arise in response to a particular stimulus. Moods are more enduring global states. Affect is the general valence (positive or negative) of that state and accounts for much of the variance in the state [6]. According to the Diagnostic and Statistical Manual of Mental Disorders [7] criteria, depression, for instance, is persistent low mood or loss of interest in activities once enjoyed.

Group emotion researchers such as [3], [5] or more recently [8] often examine the explicit and implicit mechanisms through which group emotion is shared. These may include emotional contagion, vicarious affect, behavioural entrainment and interaction synchrony, i.e. the tendency for group members to automatically adjust their behaviour to synchronize with other members' behaviour.

In our project, we have originally found the concept of mood to be most appropriate for two reasons: On the one hand, affect tends to be examined as an individual-level phenomenon, even though some authors also have focused on group affect [9]. On the other hand, the concept of emotion contagion is associated with the disease metaphor which suggests that simply being in contact with another person puts you in a similar condition. Besides, we have been influenced by a study conducted by [10], which examined the effects of 3 behavioural interventions on the mood in nursing home residents with Alzheimer's disease.

As mentioned before, a mood change may trigger either activation or relaxation on the part of PwD, our target population. In the project we induce mood changes by adjusting the room ambiance in terms of light, sound and scent. As far as sound is concerned, we use natural sounds such as the twittering of birds (for activation) and the rippling of water to achieve a soothing effect. With regard to scent we use mixtures from a professional vendor (Primavera Life GmbH). For activation the ingredients include orange, lime and lemon, whereas for relaxation, rose is the main ingredient. However, the care facilities may use their own scents or preferred mixtures.

Our luminaire contains modules for direct lighting (task light) and indirect lighting (ambient room lighting) with about 110 watt light power each. Colour temperatures can be changed seamlessly from 2200 K to 5700 K with a luminous flux of about 10200 lm which results in about 1000 lux for the illuminated area. Whenever an activation or relaxation intervention is triggered, the light temperature as well as the brightness gradually changes according to the pre-defined lighting curves.

12.1. Measuring stress

Stress measurement methods can be divided into two categories: psychological questionnaires and physiological measurements. The most accurate result will probably come from combining a reliable physical measuring method with self-assessment. Physiological stress may be measured by monitoring heart rate variability, breath frequency, blood pressure, and by measuring different stress hormones such as cortisol which is found in blood, urine, saliva, and hair.

Saliva is considered one of the best biomarkers of stress and sampling is noninvasive (as opposed to venipuncture or blood tests). During stress the adrenal cortex releases stress hormones within seconds. It takes about one to two minutes until the shift in hormone levels (especially cortisol) can be measured in saliva [11, 12]. However, at least for the time being, one still needs a laboratory for the measurements which is why we have opted for using heart-rate variability and activity parameters instead.

Heart-rate variability is a well-established measure for agitation, i.e. a feeling of restlessness or extreme arousal. Arousal is crucial for motivating certain behaviours but PwD often have problems with self-regulating arousal. Instead of measuring agitation levels of PwD, we measure them indirectly via the stress levels of the caregivers. It goes without saying that it would be more accurate and reliable to take measurements directly from the patients. However, due to ethical considerations associated with highly vulnerable persons who in most cases are not able to give their informed consent, this has not been possible.

Based on the mood transfer concept we intended to use the activation level of the whole group (measured by optical motion sensor) and the activation level of single carers (measured by mobile physiometer) as mood indicators. For measuring the activation levels within a particular area, we used passive infrared sensors (PIR) with low latency and high sampling rate. For the measurement of HRV and additional physiological parameters we used the upper arm vital sign monitor Everion from Biovotion.

12.2. Machine learning

We set out to develop algorithms to detect situations when an intervention may be useful. This means the system had to learn from interventions triggered by carer givers. Whereas a care giver may assess and judge a situation based on his or her experience and expertise and use a host of different criteria the system has to rely on a reduced set of parameters, i.e. the stress levels from caregiver, activation levels or actions invoked previously by care givers.

Our machine learning approach consisted of two main phases: a) feature engineering and b) setting up the machine learning pipeline (classifier training, validation and selection) as illustrated in Figure 55. The following paragraphs describe the general approach. Chapter 12.3 then lists some of the challenges to the envisioned approach before chapter 12.4 describes the concrete implementation in more details.

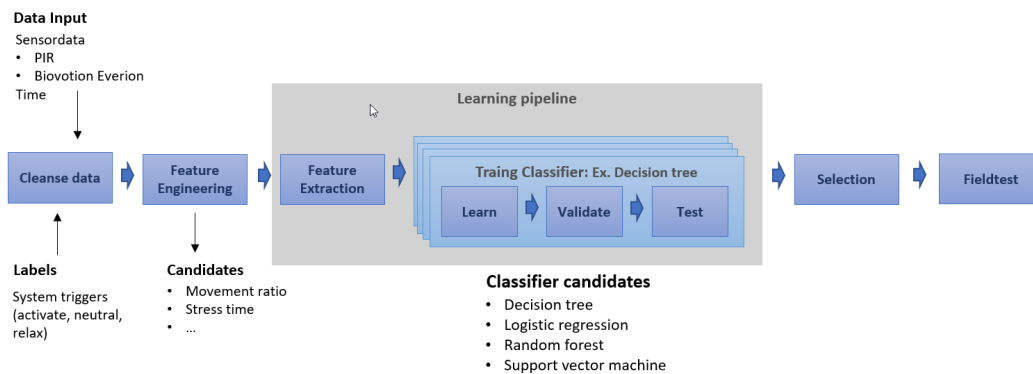


Figure 55: Machine learning approach with classifier variants

First, we labeled data captured from the various sensors with «intervention» and «no intervention» to create training data for our supervised machine learning approach. The machine learning process itself starts with a data cleansing phase where we check for any outliers or inconsistent data points. After this preparatory phase, the feature engineering phase can begin [13].

a) Feature engineering phase

Based on the raw inputs provided by the sensors (PIR and Everion), relevant features needed to be engineered. Possible candidates were: HRV signal before and after contact of the carer with PwD (within a pre-determined time window), time of data capture, normalized stress level of carer, activity level of PwD. The list of features is not pre-defined. Rather, finding or designing appropriate features is an integrated part of the research effort of this project and the first outcome of the machine learning process.

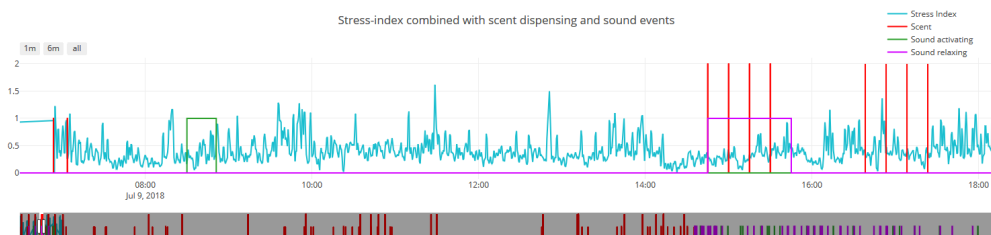


Figure 56: Stress Index and intervention events (scent and sound)

Figure 56 depicts an example of recorded data. The normalized stress index derived from the heart rate variabilities of the carers and the motion samples are feature candidates for learning. The manually triggered system interventions with activating or relaxing scent and sound are labels for learning corresponding predictions which should be translated into recommendations about interventions. The view samples of interventions point out the problem of the small amount of labeled data available for learning, which might lead to poor learning performance.

b) Setup of the learning pipeline

The machine learning pipeline consists of the software implementation of the feature extraction, classifier learning, validation and testing. The extracted features are used to train a classifier to predict intervention situations based on the input features. Additionally, a classifier may suggest different actions a carer could pursue given the value of the input features, i.e. to trigger the system to activate the PwD, to relax the PwD or to do nothing.

For the machine learning based classification, several candidate methods, e.g. logistic regression, decision tree-based classification or support vector machines exist. The final choice of classifier is determined by their performance on the gathered field test data and largely depends on the number of features and data points that are available. Performance is compared according to classical evaluation criteria like true positive rate, false positive rate, precision, recall and F1 scores.

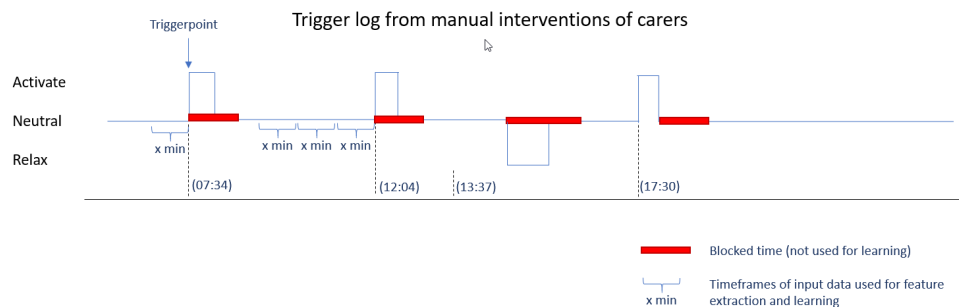


Figure 57: System trigger points for learning the trigger situation

Figure 57 depicts an example of trigger points. Relevant segments of input data for learning are the ones before an activation or relaxation event. Some segments may also relate to situations, when no system interaction is required. The data right after an intervention will not be used for learning (blocked time frame) because of the influence of the intervention itself. The length of the segments is a parameter which also has to be evaluated in different learning settings.

Once the system has learned to identify situations calling for intervention, it sends recommendations to the carers via the GREAT UI. The carer can then follow the recommendation, which results in its reinforcement, or he or she can just ignore it, which results in its weakening or abandoning. The number of high-quality recommendations should increase over time as a result of this training approach.

12.3. Challenges and coping strategies

In the course of implementing our approach we have expected and encountered a number of challenges for which we have been trying to find solutions. These include the following:

Very limited set of parameters and test cases.

Due to the vulnerability of the group and the resulting difficulty of obtaining ethical approval we have decided to measure the stress level of the actual target population, i.e. PwD, indirectly via the caregivers. This caused a significant increase in the noise of our measurements since other factors that are completely unrelated to the presence or interaction with PwD might induce stress for caregivers such as financial or family issues.

Also, it takes a much longer time to reach a significant number of actual test cases than originally expected. In 10 out of the 12 locations, we observed only a single intervention situation per day on average. We expected to have a few hundred interventions per location by the end of the field trials, but this was not the case. The low number of cases effectively limits the possible quality of any classification algorithm. Additionally, the big asymmetry between intervention cases and non-intervention cases makes it less appropriate to use neural network-based approaches.

Care givers seem to be too occupied with their daily work to also keep in mind that they could use an additional system to help improve the mood of all, especially if the effects are not immediately recognizable.

Coping strategy:

Instead of a classical supervised learning approach for the intervention recommendation, the final GREAT product might be more successful to apply an anomaly-detection based algorithm (unsupervised learning).

We might get more useful outcomes if we could use additional parameters and/or different approaches, e.g. automated image-based analysis of activities in the room, however, this imposes a bi gprivacy concern.

Problems to detect apathy.

Room sensors cannot differ between a person doing nothing and no person in the room. The measured room activity can also overlay a patient's activity. If the potentially passive PwD is within a room of active individuals, the measured activity level is misleading.

Coping strategy: Currently, we try to focus on those PwD in our measurements who tend to move in areas with just a few or no other patients. To improve the identification of passivity we could use motion sensors with higher resolution, which

can even detect smallest movements, or image-processing based solutions as mentioned above.

Lack of thresholds for stress.

Stress levels are influenced by a vast number of factors. The body's response to stressful situations varies strongly between individuals. We implemented and extended machine learning algorithms from one of our previous projects, SmartCoping [19, 20], which integrated an approach to calibrate individual stress during a learning phase.

Coping strategy: To compensate for the missing learning phase in this project, we adapted the algorithm to calculate individual stress levels from low stress and high stress levels detected during baseline measurements.

Difficulties related to attributing stress to particular causes.

Another challenge is to distinguish between stress induced by patient situations or moods and stress induced by patient-unrelated causes or circumstances. The latter could be issues concerning a person's private life like relationship problems, illness or pain, money worries etc. as well as stressful situations at the workplace unrelated to the patient, e.g. a conflict with one's colleagues or the management.

Besides, it is difficult to distinguish between physically induced stress and mental stress. If a carer has to accomplish physically demanding tasks like carrying heavy loads or climbing upstairs, the HRV changes.

Coping strategy: It is possible to identify some of those situations, esp. those related to physical effort, by using the accelerometer integrated into the sensor.

With regard to stress caused by patient-unrelated causes, we have to rely on the observations of our study nurse made in the course of dementia-care mapping conducted during the field tests. The study nurse also conducts so-called "situational conversations" immediately after interventions to obtain the subjective assessment of the caregivers. By combining these methods, we should be able to ascertain the causes of stress in particular situations. However, this data will not be available in a live system and can only be used as "pseudo-intervention" input for learning.

Acceptance problems.

The body sensors worn at the upper arm were found to be uncomfortable by some of our test persons who complained about skin irritation from wearing the sensors several hours every day. Others reported itchiness due to sweating or skin pressure. The need to put the sensor into a cradle every day has not caused any problems, since the battery lasted a lot longer than a normal day shift.

During the field tests, the reluctance to wear the sensors led to very little overlapping data-point where also interventions were triggered.

Coping strategy:

The body-worn sensor would deliver valuable information about stress levels of the wearer when worn, however, as experience showed during the field tests, we cannot rely on obtaining data from them as caregiving personal is reluctant to wear them. For the learning algorithm this means that we need to mainly rely on room sensor and intervention data as input.

For the final GREAT product, a possible approach would be to use wearable devices that also provide directly useful features/information for the wearers. Maybe a dedicated SmartWatch that also offered some kind of status info, reminders and feedback opportunities would be more accepted.

12.4. Implementation

Within GREAT machine learning approaches are used on two levels: the stress level identification based on HRV measures from the Biovotion sensor, and the recommendation engine for interventions for care givers.

Since the stress level identification algorithm has been derived in a previous project by a project-partner and adapted for the limitations within GREAT as outlined above, this description details the approach of the intervention recommendation engine for GREAT based on room sensor and intervention usage data.

12.4.1. Feature engineering

PIR sensors used for detecting motion typically output an on/off signal, showing whether it currently detected motion or not. The PIR sensors used in GREAT transmit their binary state over air by the EnOcean protocol to the main controller. The controller then adds a timestamp to the motion event and logs it in the GREAT database. In addition to PIR sensors, also air quality data is obtained by Bosch BME 680 IAQ sensors integrated into the scent dispensers. This data is also logged with a timestamp by the controller. Motion and air quality measures could both be an indicator for changes in activity levels and thus form a basis for intervention recommendation. Note that the air-quality sensor has been added due to concerns that motion by itself might not be a sufficient indicator for recommending helpful interventions.

To get better characteristics of motion based on the binary signal of the motion detectors, we analyse the signal over a time frame of 5 minutes. Within this timeframe we derive two important measures as an indicator of motion activity: The relative period of motion within the timeframe (called integral), and the count of fluctuations of the signal within the period (called cnt). While the first one shows the overall (and possibly) continuous motion activity within the period, the second one captures dynamic characteristics (e.g. if it was a steady flow of motion, or highly interrupted).

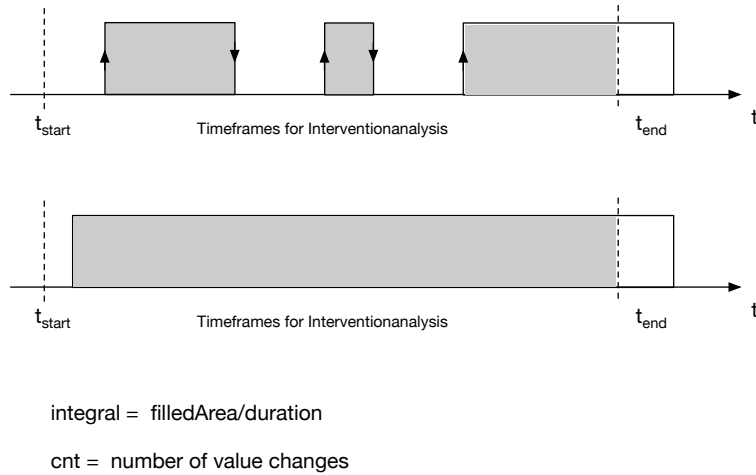


Figure 58: Binary motion data characterization within a time frame.

The same approach can also be used for analog data like air quality, where the integral parameter reflects the relative area under the curve, and the cnt parameter the stability of the signal. In addition to this, also standard deviation and arithmetic average are calculated. To also take dynamics of these values over time into account, we keep values of previous timeslots as a reference, which allows us to calculate the difference of values between time frames.

To get an idea of the usefulness of our derived parameters we visualized data in scatter plots to show relationships between two parameters and the intervention state. Figure 59 shows a visualization of possible relationships between the different calculated parameters and manually triggered intervention events. Blue points mark time slots without interventions, whereas orange points show manually triggered interventions.

As triggered interventions might also have a time and date specific component to them, as can be seen in Figure 59, we also included those in the feature vector. A typical feature vector for classification that contains time, motion, airquality and optionally HRV information looks like the following example, where n is the number of previously observed timeslots that should also be considered:

```
featureVector = [
    dayNum, relSecond,
    activityt0, activityt-1, activityt-2,..., activityt-n,
    activityDifft0-t1, activityDifft1-t2,..., activityDifftn-1-tn,
    airQualityt0, airQualityt-1, airQualityt-2,..., airQualityt-n,
    airQualityDifft0-t1, airQualityDifft1-t2,..., airQualityDifftn-1-tn,
    hrvt0, hrvt-1, hrvt-2,..., hrvt-n,
    hrvDifft0, hrvDifft-1, hrvDifft-2,..., hrvDifft-n,
]
```

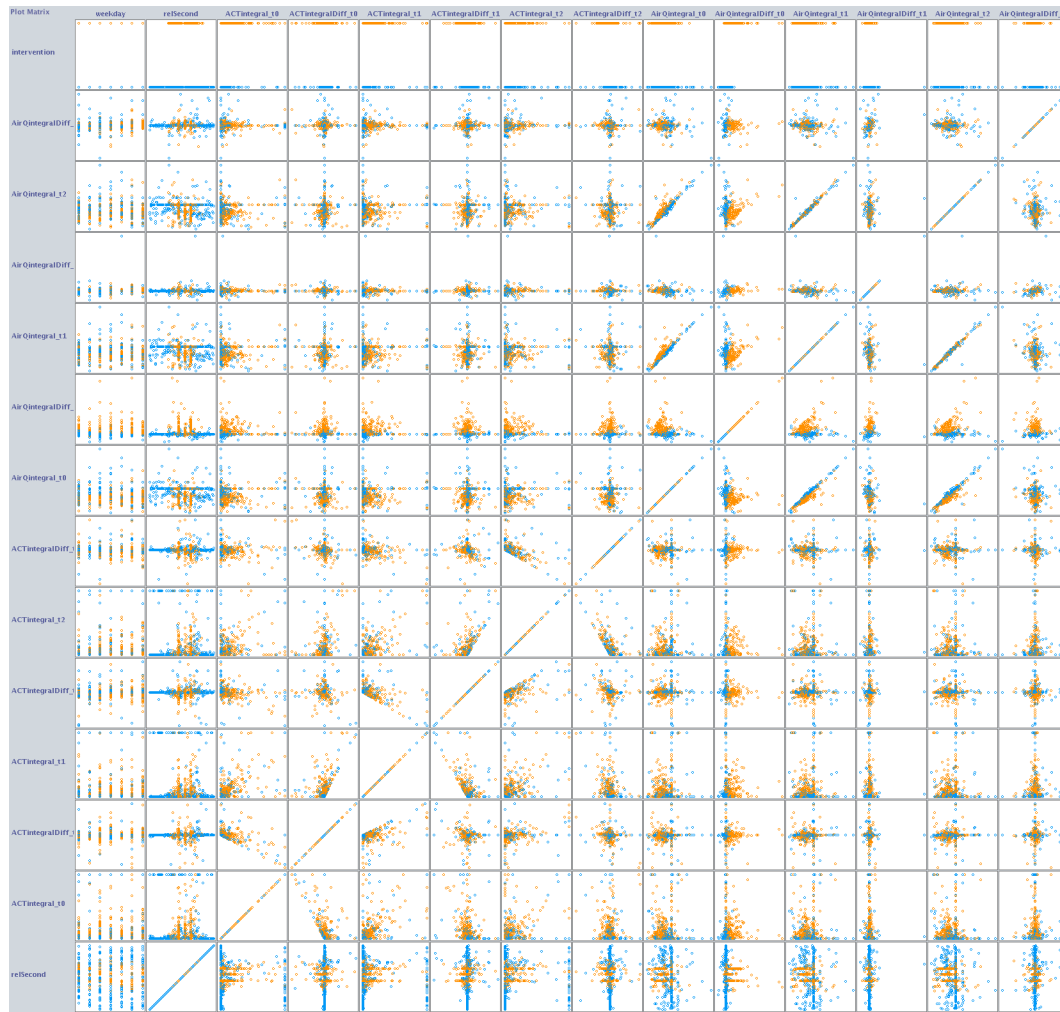


Figure 59: Relationship among two parameters and intervention state (blue...no intervention, orange...intervention)

Of course, there are many more possibilities to come up with additional inputs for the feature vector (e.g. different statistical measures within the timeslot). Further testing, experimentation and evaluation will be needed to be done in the timeframe until the release of GREAT as a product, and even afterwards, data from real life use will have to be analysed to further optimize the system.

12.4.2. Processing pipeline of the learning system

The GREAT learning system is implemented using a distributed approach. While all required components run on the same local controller, they run in separate processes. This enables us to use open source machine learning frameworks with a wide range of available classifiers, in a loosely coupled manner without introducing direct dependencies in the core system.

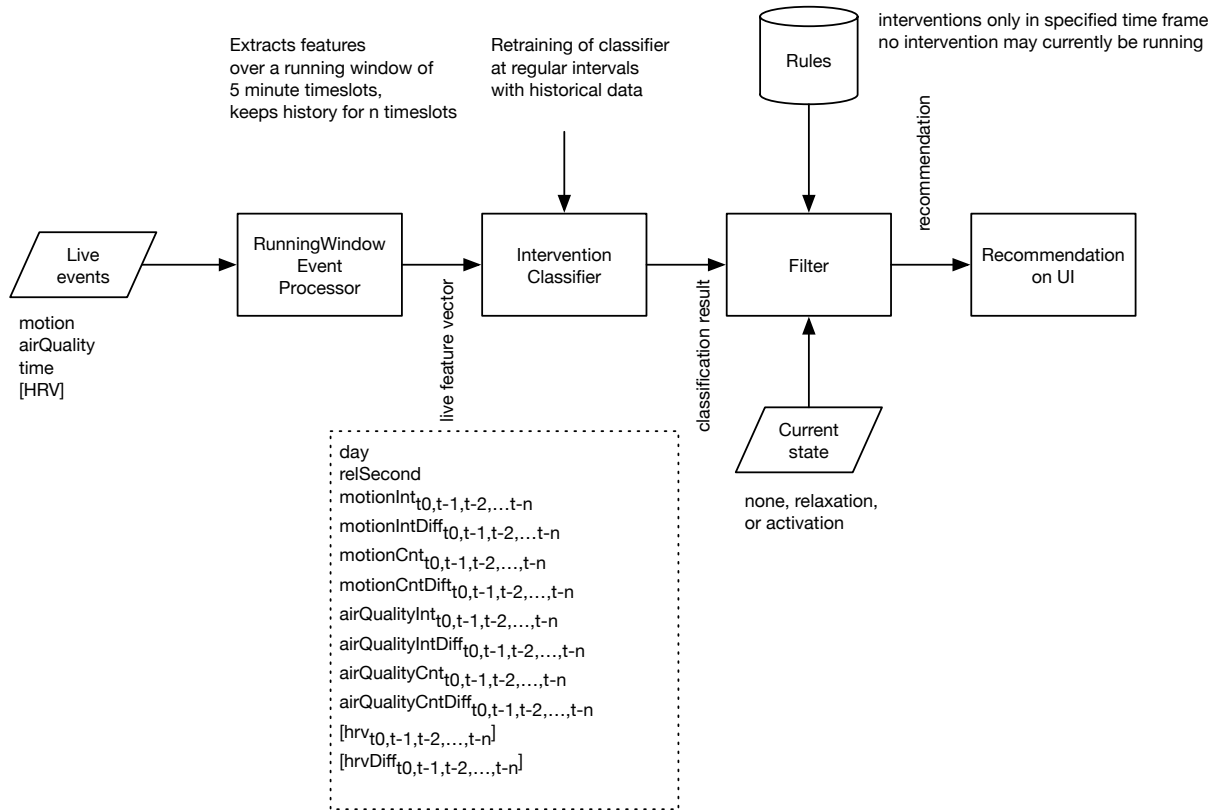


Figure 60: Learning system pipeline from live events to recommendations

Figure 60 shows the current version of the processing pipeline from live events to the final recommendation. Live events are processed inside the GREAT middleware using the RunningWindowEventProcessor extension. This extension generates a feature vector, that is passed to an external classification component over a TCP stream. The classification result is then again forwarded to the GREAT system, where a filter extension decides whether to pass on the recommendation to the UI client or to ignore it due to current state restrictions or defined rules. This filter stage allows for effective plausibility checks of classifications, while also providing a mean for enforcing customer defined restrictions on the system (e.g. only allow automatic recommendations within a specified timeframe).

The external InterventionClassifier is based on the open source WEKA framework¹ for machine learning, which allows us to flexibly train and adapt classifiers.

With a feature vector like described above with integral and cnt-parameters for motion and integral parameters for airQuality and a history depth of 3 time slots we obtained the best classification results with a RandomForest classifier. We used a k-fold cross validation with k=10, instead of a fixed percentage split of training and test data to get to most of our small data set.

Table 22: Performance metrics of classifiers for the given feature vector on the field test training set

CLASSIFIER	TPRATE	FPRATE	PRECISION	RECALL	F-MEASURE	MCC	ROCAREA	PRCAREA	CLASS
RANDOM FOREST	0.824	0.112	0.816	0.824	0.82	0.711	0.93	0.906	Activate
	0.71	0.105	0.714	0.71	0.712	0.606	0.913	0.811	Relax

¹ <https://www.cs.waikato.ac.nz/ml/weka/>

	0.789	0.113	0.795	0.789	0.792	0.678	0.93	0.891	No Intervention
MULTILAYER PERCEPTRON	0.706	0.135	0.752	0.706	0.728	0.579	0.838	0.768	Activate
	0.625	0.141	0.613	0.625	0.619	0.481	0.813	0.599	Relax
	0.787	0.152	0.753	0.787	0.769	0.629	0.881	0.795	No Intervention
LOGISTIC REGRESSION	0.681	0.12	0.772	0.681	0.724	0.577	0.825	0.781	Activate
	0.275	0.086	0.539	0.275	0.364	0.243	0.67	0.417	Relax
	0.85	0.357	0.569	0.85	0.682	0.473	0.778	0.563	No Intervention
SMO	0.674	0.167	0.707	0.674	0.69	0.512	0.763	0.635	Activate
	0.105	0.062	0.382	0.105	0.165	0.072	0.561	0.299	Relax
	0.857	0.409	0.538	0.857	0.661	0.433	0.733	0.52	No Intervention
LOGITBOOST	0.792	0.094	0.834	0.792	0.813	0.705	0.939	0.913	Activate
	0.71	0.099	0.724	0.71	0.717	0.615	0.912	0.818	Relax
	0.782	0.159	0.732	0.782	0.756	0.615	0.913	0.873	No Intervention

Table 22 shows the performance characteristics of different classifiers tried with the field test training set. It's interesting to note that the performance for activation interventions identification is better than for relaxation interventions. This might be an indication, that motion activity by itself (or the lack thereof) can be a predictor for activity interventions, but not so much for relaxing interventions.

In the context of GREAT the false positive rate is a much more critical factor, than the true positive rate, as accidentally triggered interventions are more irritating, than missed intervention suggestions. The reasons for the still high error rate might be due to the fact that we have to deal with an uneven distribution of None-Intervention/Intervention cases (e.g there are much more cases where the system shouldn't trigger an intervention, than there are for triggering an intervention) combined with a relatively small training set of only 3700 interventions in total, which are then split up for each zone at locations.

In a next step of development, we would therefore recommend looking into anomaly detection algorithms as a supplement to the classification tasks. For this a history enriched version of our processing pipeline might be suitable, like shown in Figure 61.

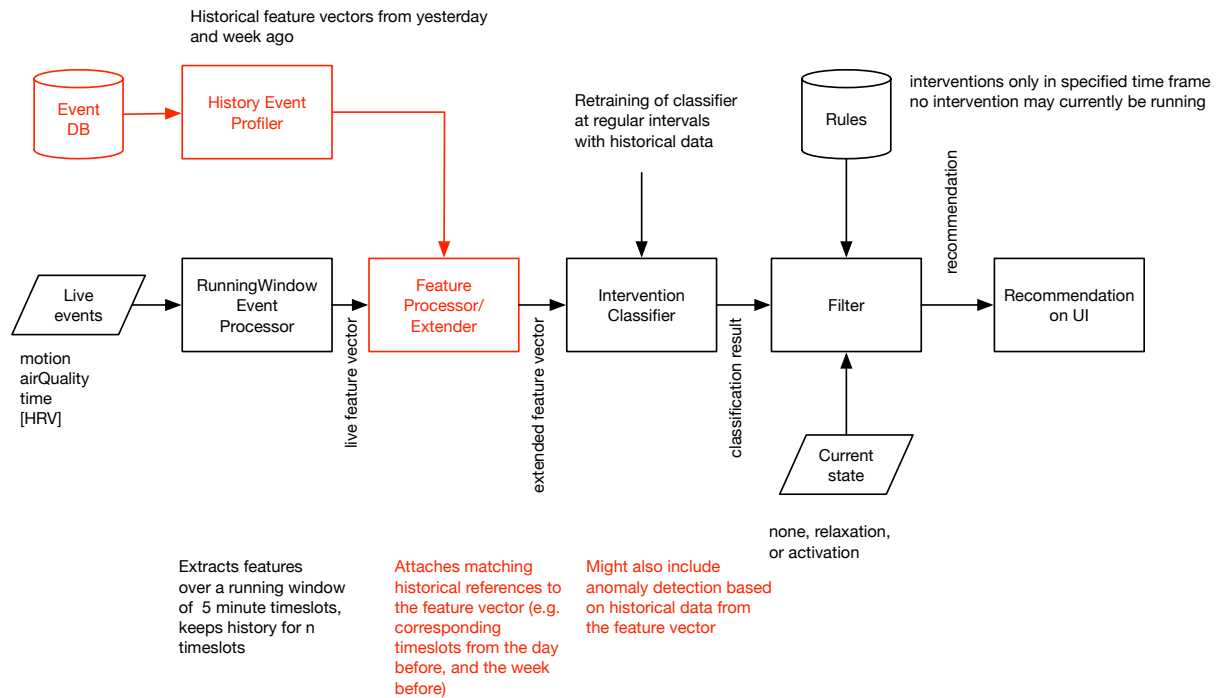


Figure 61: Extended learning system pipeline for considering historical data points too.

This architecture also allows for a flexible extension of the system with further input, in case new sensors are added to the system.

Currently the machine-learning based recommendation system is an experimental unit of the GREAT system. Further usage data needs to be gathered to provide a machine learning based recommendation system that's sufficiently reliable for use in the wild.

In chapter 12.5 and 12.6 we present alternative approaches for our automatic control system, based on experiences made during the field tests.

12.5. Schedule based operation

While working with our partners from the field test locations, we learned that especially in clinical/elderly home settings care givers are extremely occupied by their day-to-day tasks. Having a system, where they can manually trigger interventions to help them calm or activate persons may be useful to them, but they often forget about its existence besides all the other important things they have to keep in mind. In our field tests this led to a very low number of interventions from which we/our system could learn.

Since the daily life in a clinical setting is very structured, we decided to also implement a feature for schedule-based triggering of interventions, to reduce the mental burden for the care givers while still using our system. For this, care givers provided us with a schedule when they usually wanted people to be active or

relaxed. It turned out that this extension to the system was very well received among care giving personal.

This scheduling system can also be used to define time frames, where the automatic intervention recommendation system should be active. Details for this scheduling feature can be found in chapter 11.2.

12.6. Rule-Based Recommendations

Since it was not possible with the machine learning approach to let the GREAT system search for patterns in movement activity on its own, we decided to define patterns manually and let the GREAT system search for them during use. However, since a pattern cannot cover all cases of practical use, the definition of the pattern is continuously adapted by the GREAT system, based on the data collected so far. In this respect, it is a non-deterministic rule system. In some aspects it is also a recursive rule system, because the adjusted variables have an influence on further adjustments.

12.6.1. Processing Rule

The measure of motion activity is determined by the number of times the PIR sensors are triggered within five minutes. This gives us a noise signal filtering. In order to identify patterns in motion activity, we work exclusively with 24h profiles of this motion measure. Over all past days, an average 24h profile is first formed since the start of the measurement. A low-pass filter with finite impulse response (FIR low-pass to remove strong fluctuations in motion activity) is applied to this profile. This corresponds to a moving average with centering. By stretching and compressing the time series in the y-direction, the regular range for allowed deviations from the moving average of a currently registered motion measure is defined (see Figure 62).

During operation it is then registered how often the current degree of movement lies outside the normal range in direct succession (every 5 minutes). In the standard setting, it may not be above or below the normal range more than three times in succession (this corresponds to 15 minutes). If, in individual cases, the current motion measure falls below or exceeds the regular range three times in a sequence, a corresponding intervention is triggered. As long as an intervention is running, no new intervention can be triggered (see Figure 63). By default, the relaxation intervention lasts 40 minutes and the activation intervention 20 minutes.

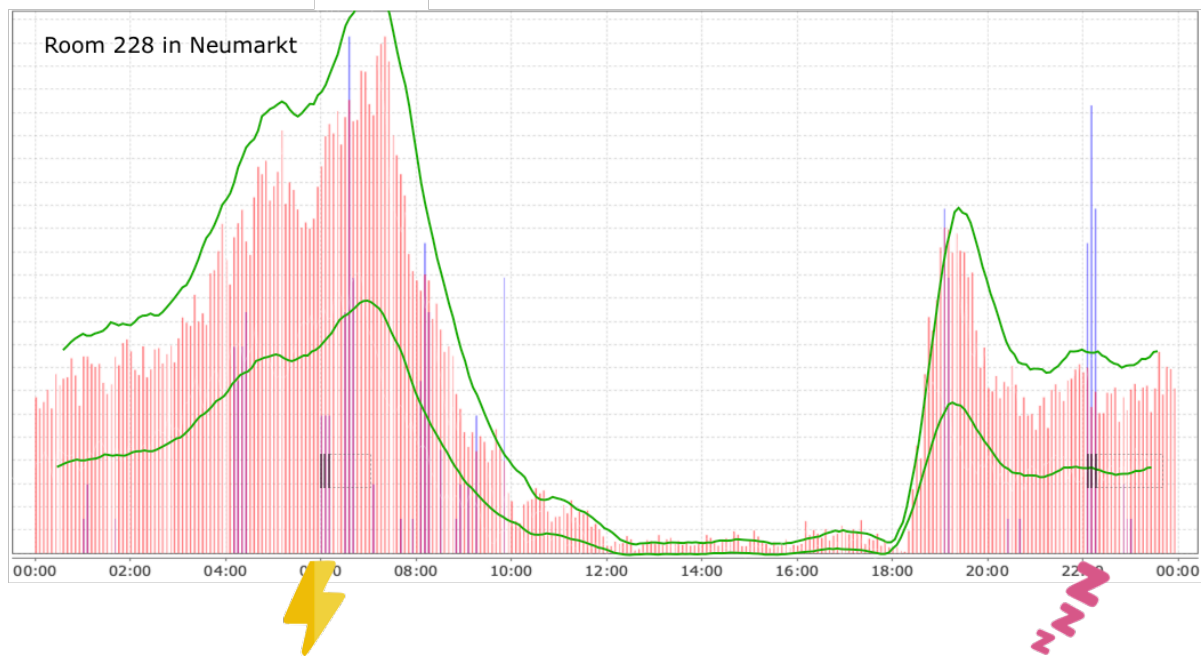


Figure 62: The red lines show the average 24h profile, which is calculated from all motion measures (of room 228 in Neumarkt) since the start of the measurement. The blue lines show the motion measure in room 228 in Neumarkt for one single day. The green lines indicate the normal range, generated from low-pass filter as well as compressing and stretching the time series. The flash symbol indicates where an activation intervention is started and the ZZZ symbol indicates where a relaxation intervention is started.

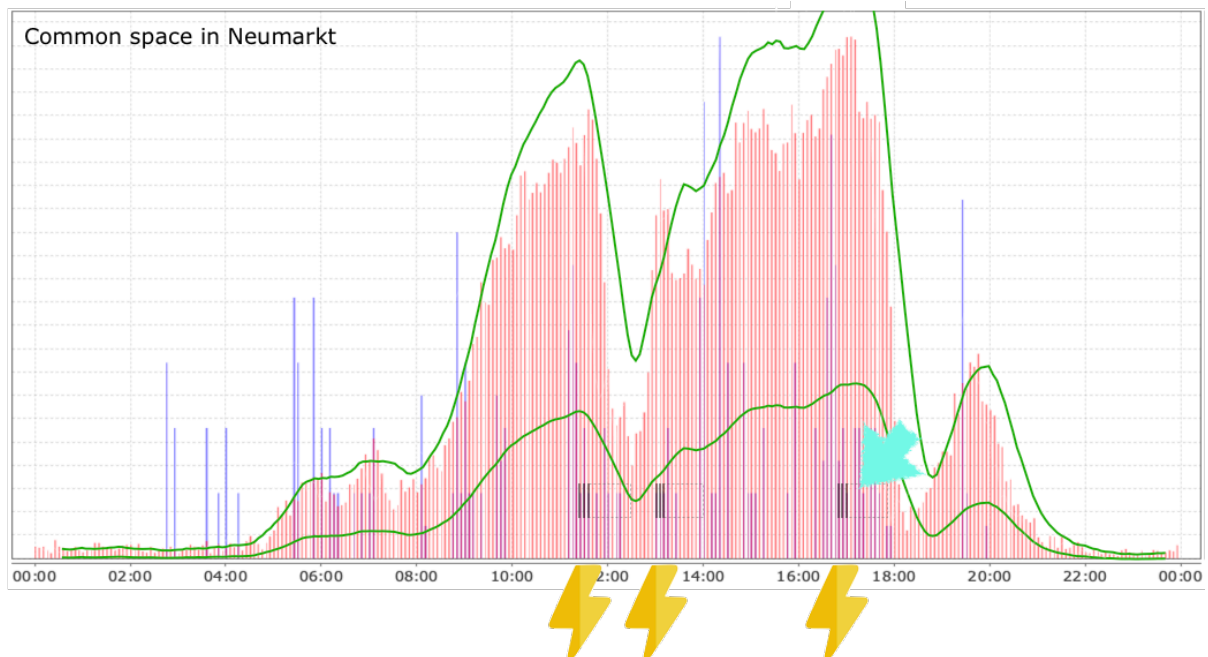


Figure 63: The arrow indicates a point at which an activation intervention would be triggered if an intervention was not already running.

12.6.2. Continuous Adjustment of Variables

If the same intervention is triggered twice in short succession, if the relaxation or activation intervention is not triggered at all or is triggered more than six times during the day, the GREAT system will adjust variables of the control system. All adjustments

are reset weekly as long as there is insufficient experience with the continuous adjustment of variables. The two onset values mentioned can also be adjusted at a later date.

The window size for the low-pass filter and the compressing and stretching factor is adjusted if an intervention is triggered more than six times in one day or not at all. In the first case the factors are increased by 20%, in the second case they are decreased by 20%. If the onset value is still reached the following day, then the value for the frequency of deviation from the regular range (which is necessary to trigger an intervention) is reduced or increased by one single case.

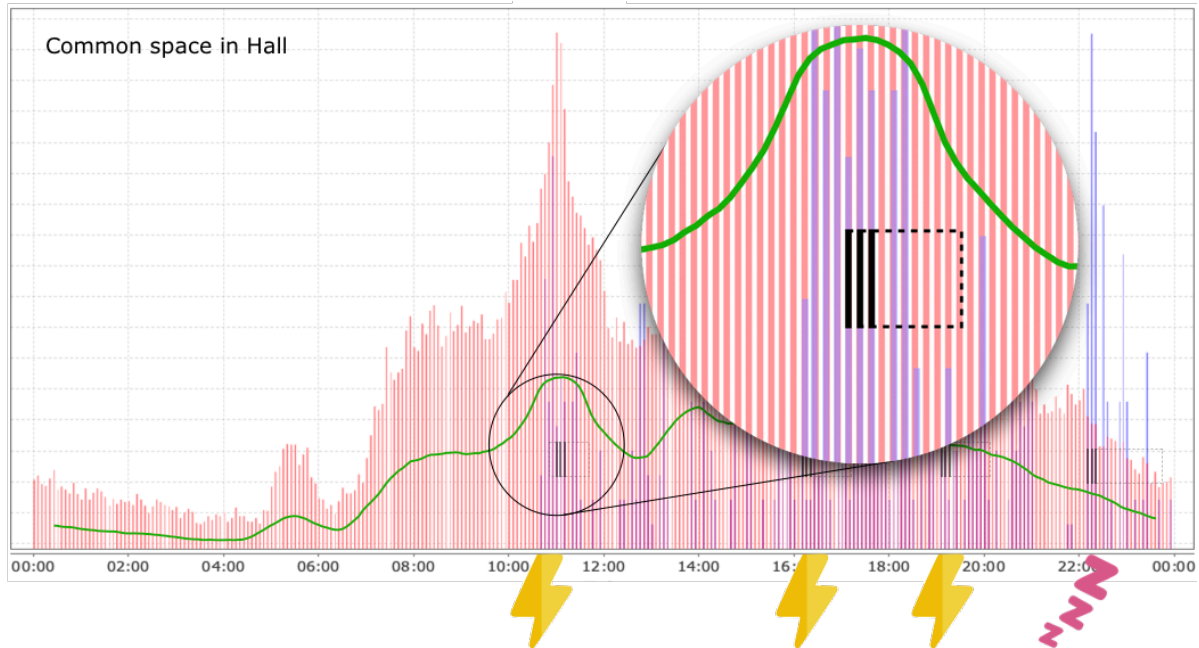


Figure 64: In this example, the same intervention is triggered twice directly after each other. If this had happened a second time on that day, the duration and intensity of the intervention would have been adjusted.

The duration and intensity of the intervention is adjusted, if the same intervention is triggered twice directly after each other (maximum 5 minutes after the end of an intervention) in two consecutive cases per day. Then the duration of the intervention is increased by five minutes. If this case occurs again, the intensity of the intervention is increased by 25% (more fragrance atomization, adjustment of the brightness and color of the light and the frequency of the sound) (see Figure 64).

12.6.3. Block diagram and parameters of the control system

Figure 65 shows the basic blocks of the adaptive rule-based system, with the described parameters.

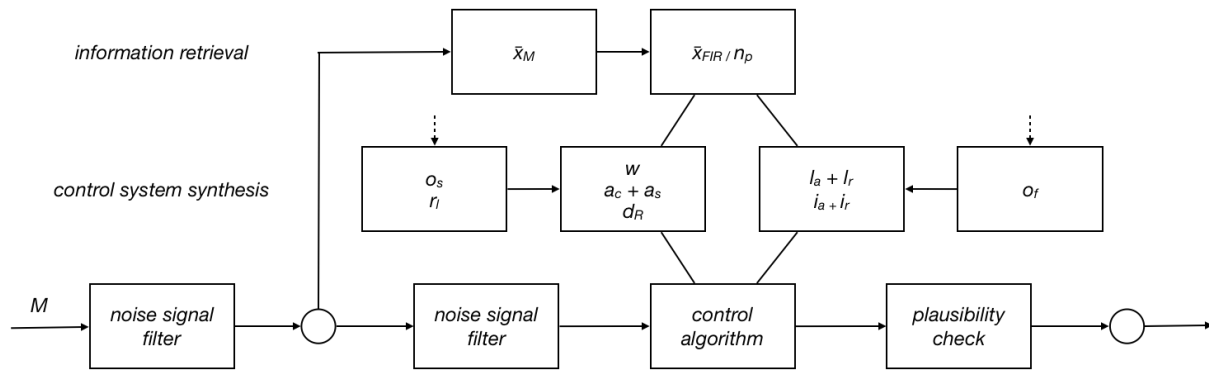


Figure 65: Block diagram of the control system.

M ... measure of physical activity

\bar{x}_M ...mean value of physical activity

n_p ...number of past days

\bar{x}_{FIR} ...moving average of the movement activity

w ...window size for low pass filter

a_c ...compressing factor

a_s ...stretching factor

d_R ...frequency of deviation from the regular range

I_a ...duration of the activation intervention

I_r ...duration of the sedation intervention

i_a ...intensity of the activation intervention

i_r ...intensity of the sedation intervention

O_s ...maximum value for successive interventions

O_f ...maximum value for interventions per day

r_l ...period for the reset

12.7. Summary and further research

Developing daily routines and activities that are meaningful is one of the most challenging aspects of providing care for PwD [16]. Our system aims at enhancing the quality of life of PwD and their caregivers by tackling this challenge. This is to be achieved by inducing the right mood for predefined activities (i.e. eating, sleeping, walking) in PwD and addressing behavioural problems such as agitation and apathy.

Over the last year and a half, we have been working to develop, implement and validate an intelligent, modular, persuasive ambient system to prepare PwD for new or changing activities during the day and thereby assist the care recipients as well as the caregivers. The system should be usable out of the box in dementia care facilities and households with PwD.

Based on our experience so far and the challenges we are facing we have come to question if automated recommendations triggered by frequently unreliable input data is the most appropriate approach for achieving the objectives of our project. Machine learning approaches require a host of different parameters and benefit from big data samples. If the training samples contain situations with certain patterns that are not related to the level of agitation or apathy of members of the target group, the trained algorithm will trigger false positives.

Given the reduced set of parameters and the relatively low number of test cases, different approaches may have to be considered, such as the use of anomaly detection algorithms.

In addition to this, we might use image-recognition based approaches for deducing activity levels. To alleviate privacy concerns, we could derive an activity index from different movement patterns without the need for external processing of the video feed (black box approach). [17] provide a comprehensive survey of recent approaches to activity recognition based on different video-based sensors. Some studies used Doppler radar sensors capable of penetrating rigid objects (occlusion prevention).

As pointed out, a major challenge for the machine learning system to predict agitation situations reliably is the noise to be expected in the data generation process. Since the data does not come from the actual PwD but from the carer, the chain of events that lead to a change in the stress level might be triggered by some other factor than the PwD. In effect, the indirect measurement might lead to low predictive quality of the system because changes in the stress level may be caused by factors not immediately related to PwD.

The two alternative approaches regarding automated control, scheduling and rule-based reasoning appear to be more promising for the GREAT system for the type of decisions to be made under the given sensor data. However, the rule-based system needs further testing and tuning in the field before the release of a final product.

13. Setup Procedure of the Final GREAT Product

The configuration steps described in previous sections of this document were well fitted for setting up the field test prototypes. For the final product, however, the setup process needs to be much simpler.

Our goal is to have a product, that can be easily configured by end users, without the need for using the Intefox configuration software.

From a user perspective, steps involve unpacking, connecting the devices to power outlets, and launching the GREAT app for completing the setup. The app will also be used if new modules need to be added to the configuration or if existing modules should be reconfigured to connect to a different network.

A typical setup procedure will work like described in the following steps:

The GREAT package contains the GREAT controller and one or more modules - either lamp, scent, or sound modules. The system is preconfigured, so only power and network connections need to be plugged in.

When the GREAT controller is powered on, it will launch in access point mode, offering an independent GREAT WLAN network. All new GREAT modules will automatically search for an available GREAT network and connect to it once it's available.

The customer then launches the app and waits for the system to become available. In the standard success scenario, the system will be ready to use then (see Figure 66).

Configuration by using the GREAT app

Upon start, the GREAT app will send a UDP broadcast command to scan all available controllers within the network. If the controller is connected via LAN, the system will immediately be visible in the app, saying that the system is ready to use (see Figure 66).

If the controller should be connected to a customer provided WIFI instead and no wired connection is available for the GREAT controller, the app will not be able to find the controller initially. The mobile device running the app then instructs the user to connect to the GREAT WLAN network provided by the controller (see Figure 67).

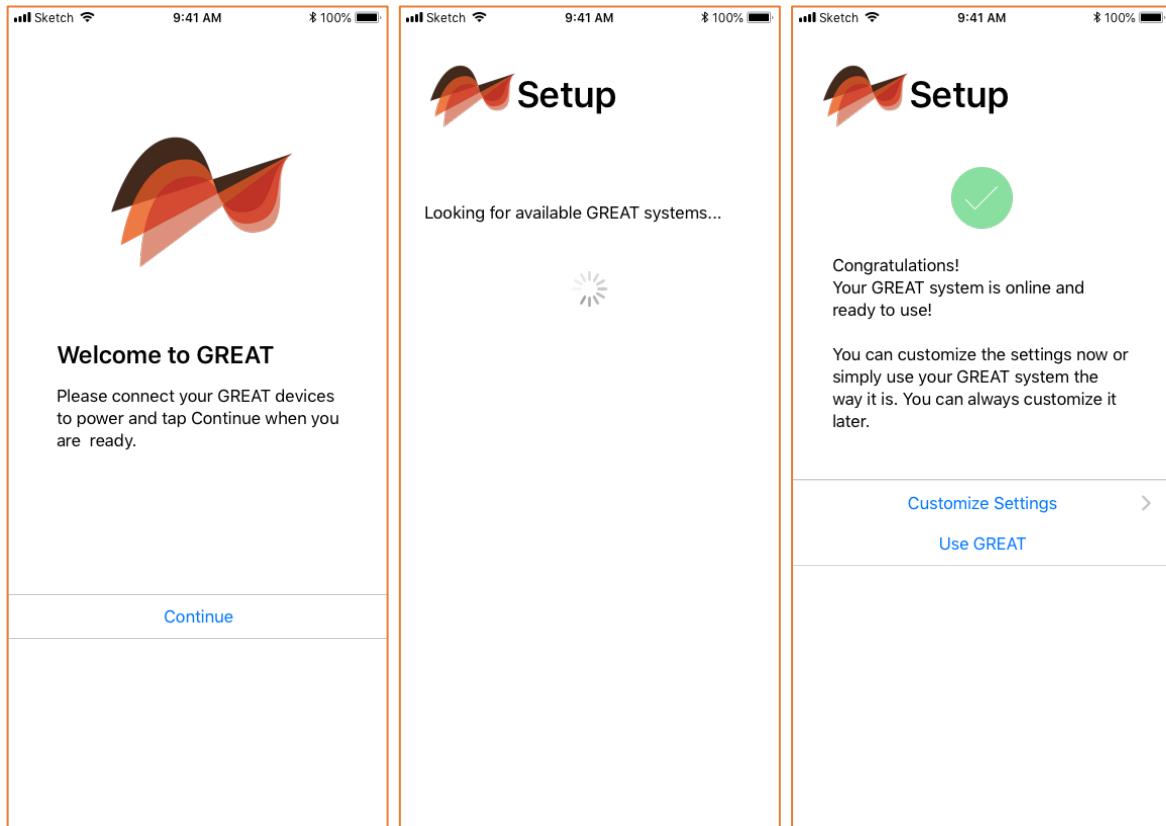


Figure 66: Initial setup of the GREAT system without additional configuration necessary

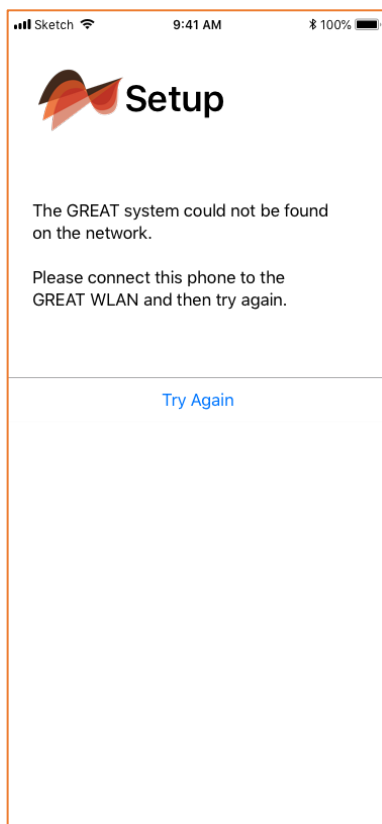


Figure 67: Failed search - giving instructions for changing the WLAN

Once the app is connected to the controller, the option is given to either use the system directly, or to customize its settings (see Figure 66, right), like adding further modules or changing the WLAN network connection (see Figure 68).

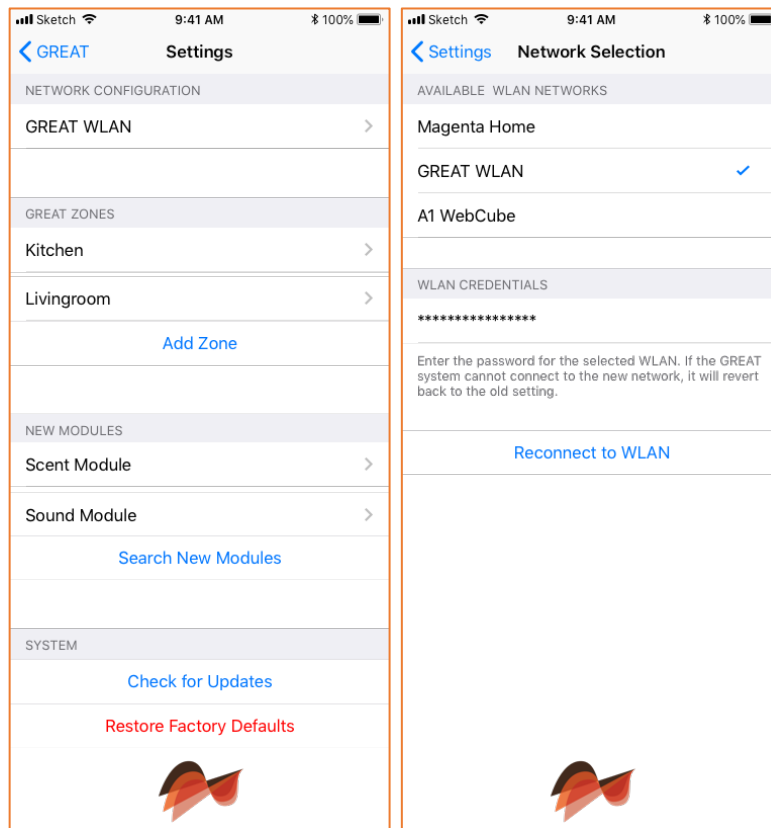


Figure 68: Main settings screen (left) and WLAN selection screen (right)

On the network selection screen, users can choose one of the available WIFI networks to connect to and then enter the network credentials.

After tapping “Reconnect to WLAN” the controller will then try to connect to the provided WIFI network, while at the same time, keeping its own GREAT WLAN network up, until configuration is finished.

All unconfigured devices will automatically try to connect to the controller over the GREAT WLAN network. Once they are connected to the controller, the controller will send its configured WLAN network parameters to the modules over an encrypted channel, so the modules can then also connect via the customer provided network.

If a module is already configured and it is not able to connect to the controller over the configured WLAN network, it will automatically fall back to trying to connect to the default GREAT WLAN network (e.g. the module has been moved to another location with different network settings).

The app will show all currently available new modules on the main settings screen (see Figure 68, left), where users can easily add them to the current configuration by tapping them, and selecting a zone from the available options on the followup screen (see Figure 69, left)

Users can also add more zones to the configuration. By tapping “Add Zone” on the main settings screen (see Figure 68, left), a new zone will be created, that can then be customized on its detail screen (see Figure 69, right).

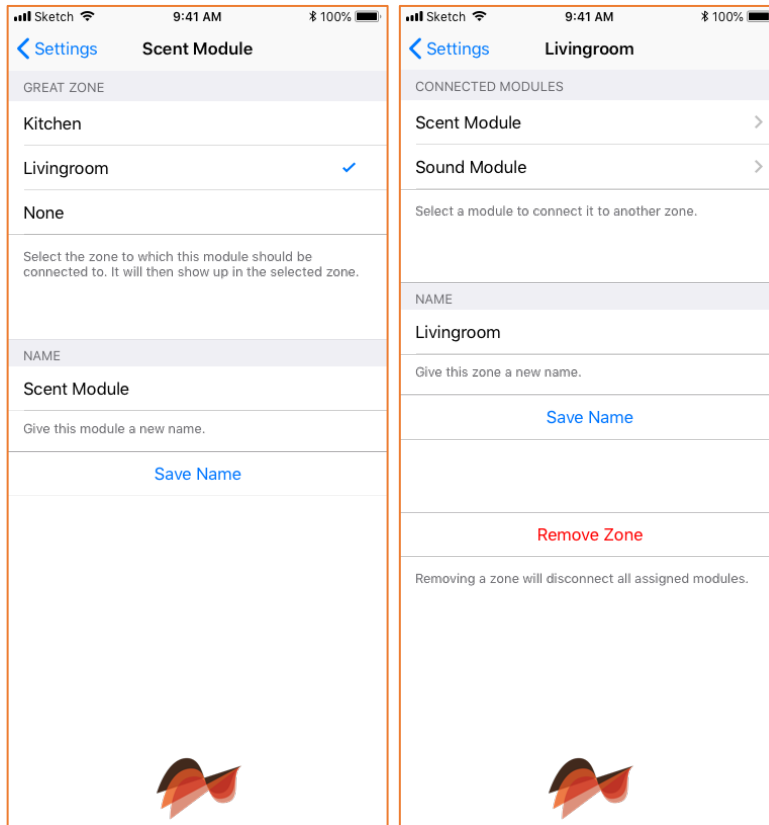


Figure 69: Module management (left) and zone management (right)

Zones can also be removed from the configuration by tapping “Remove Zone”. All modules assigned to the zone that should be deleted will then again show up as new modules on the main settings screen.

If an existing module should be reassigned to a different zone, the module can be accessed via the currently assigned zone. By tapping the module, a details screen will be shown, where the module can be assigned to a different zone by tapping on the intended zone name (see Figure 69, left).

The main settings screen also offers an update function, to keep controller and modules up to date. The app will compare the installed version numbers with available version numbers, and then either confirm that the controller and the modules are up-to-date (see Figure 70, left) or show all the updates available. Users can then decide to update all of them, or only selected ones (see Figure 70, right).

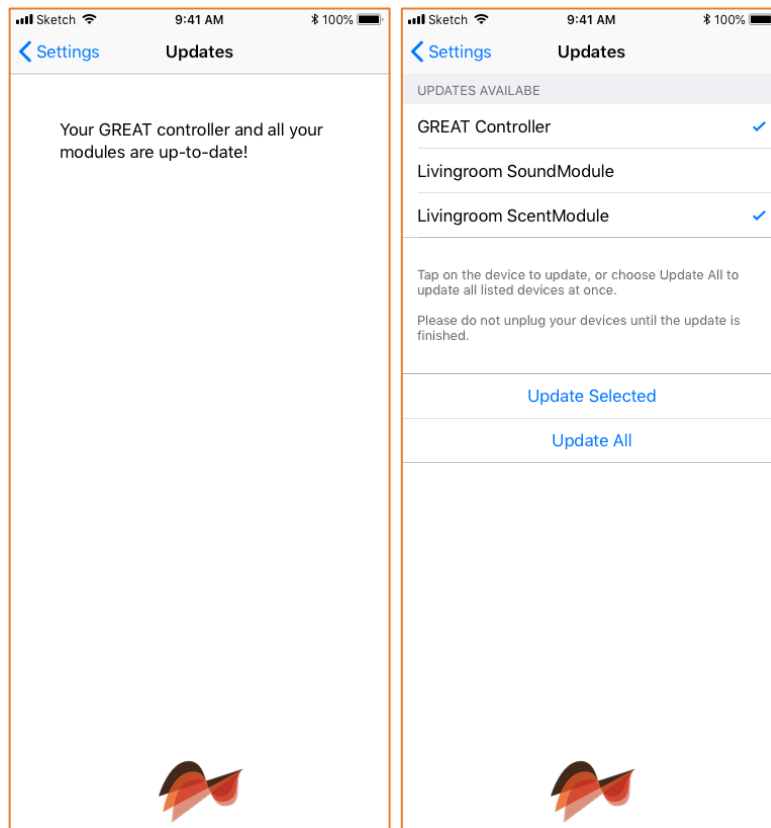


Figure 70: Update handling of the GREAT system components

Once the configuration process in the settings part is finished, the controller will automatically update its configuration in the background and reflect the newly created zones and their assigned modules on the main control screen, like described in chapter 10.

14. References

- 1 Barsade, S. G., & Gibson, D. E. (2012). Group affect: Its influence on individual and group outcomes. *Current Directions in Psychological Science*, 21(2), 119-123.
- 2 Hackman, J. R., & Katz, N. (2010). Group behavior and performance. In S. T. Fiske, D. T. Gilbert, & G. Lindzey (Eds.), *Handbook of social psychology* (5th ed., pp. 1208–1251). New York, NY: Wiley.
- 3 Kelly, J. R., & Barsade, S. G. (2001). Mood and emotions in small groups and work teams. *Organizational Behavior and Human Decision Processes*, 86, 99–130.
- 4 Parkinson, B., Fischer, A. H., & Manstead, A. S. R. (2005). *Emotions in social relations: Cultural, group, and interpersonal processes*. New York, NY: Psychology Press.
- 5 Parkinson, B. (2011). Interpersonal emotion transfer: Contagion and social appraisal. *Social and Personality Psychology Compass*, 5(7), 428-439.
- 6 Ekkekakis, P. (2013). *The measurement of affect, mood, and emotion: A guide for health-behavioral research*. Cambridge University Press.
- 7 American Psychiatric Association. (2013). *Diagnostic and statistical manual of mental disorders (DSM-5®)*. American Psychiatric Pub.
- 8 Kramer, A. D., Guillory, J. E., & Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24), 8788-8790.
- 9 Keltner, D., & Haidt, J. (1999). Social functions of emotions at four levels of analysis. *Cognition & Emotion*, 13(5), 505-521.
- 10 Williams, C. L., & Tappen, R. M. (2007). Effect of exercise on mood in nursing home residents with Alzheimer's disease. *American Journal of Alzheimer's Disease & Other Dementias®*, 22(5), 389-397.
- 11 Udelsman, R., Norton, J. A., Jelenich, S. E., Goldstein, D. S., Linehan, W. M., Loriaux, D. L., & Chrousos, G. P. (1987). Responses of the hypothalamic-pituitary-adrenal and renin-angiotensin axes and the sympathetic system during controlled surgical and anesthetic stress. *The Journal of Clinical Endocrinology & Metabolism*, 64(5), 986-994.
- 12 Schulz, P., Kirschbaum, C., Prüßner, J., & Hellhammer, D. (1998). Increased free cortisol secretion after awakening in chronically stressed individuals due to work overload. *Stress medicine*, 14(2), 91-97.
- 13 Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. "O'Reilly Media, Inc."
- 14 Maier, E., Reimer, U., Laurenzi, E., Ridinger, M., & Ulmer, T. (2014). A mobile solution for stress recognition and prevention. In *Proc. Int'l Conf. Health Informatics (HealthInf)* (pp. 428-433).
- 15 Reimer, U., Laurenzi, E., Maier, E., & Ulmer, T. (2017). Mobile Stress Recognition and Relaxation Support with SmartCoping: User-Adaptive Interpretation of Physiological Stress Parameters. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.
- 16 Rahman, S., 2014. *Living Well with Dementia: The Importance of the Person and the Environment for Wellbeing*, 1 edition. ed. CRC Press, London; New York.
- 17 Uddin, Z., Khaksar, W., Torresen, J. (2018). Ambient Sensors for Elderly Care and Independent Living: A Survey. In: *Sensors* 18 (7).

15. List of Figures

Figure 1: GREAT Components Overview, Source: GREAT consortium.....	7
Figure 2: GREAT Distributed System Overview, Source: GREAT consortium.....	8
Figure 3: Distributed GREAT installations connected over VPN.....	9
Figure 4: OSGi Architecture Diagram, Source: OSGi Alliance, https://www.osgi.org/developer/architecture	10
Figure 5: Screenshot of the Intefox configuration software showing connections between individual elements.....	11
Figure 6: Basic architecture of the foxcore server	16
Figure 7: fox.configurator, example of adding a new light object.....	17
Figure 8: Online bundle manager.....	18
Figure 9: Basic structure of the event logging architecture	18
Figure 10: Select the created event logger and edit the properties (URL and Context)	20
Figure 11: Properties of the event logger service.....	20
Figure 12: In this example, the 'Temperature' output of all 'Temperature sensors' will be logged, even if they are being created later.	22
Figure 13: Entity relationship diagram for the logging data backend	29
Figure 14: Example login request.....	31
Figure 15: Example description request	33
Figure 16: Example structure request.....	35
Figure 17: Example long poll request with no changes	36
Figure 18: Example long poll request with changes.....	36
Figure 19: Example cmd request to switch two lights on.....	37
Figure 20: Example biodynamic light definition	38
Figure 21: Example: Activation Light 'cue' definition.....	39
Figure 22: Example: Calming light 'cue' definition	39
Figure 23: Configuration of the Light curve object in conjunction with the light device	40
Figure 24: Luminaire schematic	46
Figure 25: Communication between the Intefox music module extension and the sound module.	48
Figure 26: The available connections for the music module extension for the Intefox system and its parameter settings.....	53
Figure 27: Dimensions of sound characteristics regarding receptibility, valence and direction.....	63
Figure 28: Integration of the scent module into the GREAT controller system based on Intefox.....	66
Figure 29: Input and output events (left) and parameter settings (right) of the scent module extension.	68
Figure 30: Forwarding of TCP310 EnOcean Telegrams via UDP	73
Figure 31: Heart Inter-Beat-Interval.....	76
Figure 32: Stress index values and 1-hour phases	77
Figure 33: Everion sensor setup	78

Figure 34: PC Tool VSM1 pairing with sensor.....	78
Figure 35: Data file structure	84
Figure 36: Integration of Biovotion HRV statistics into the GREAT system.	85
Figure 37: Screenshots of the main menu, the scent-, and sound-module offering control for starting an activation or relaxation session.	86
Figure 38: Screenshots of the light control page and the status view for the motion detector.....	87
Figure 39: GREAT user client main screen with all elements enabled.	88
Figure 40: GREAT user client: starting an intervention	89
Figure 41: GREAT User client with customization options before the intervention (left,) and status feedback and cancel option during an ongoing intervention (right)	90
Figure 42: GREAT User client: after an intervention (asking for feedback)	91
Figure 43: Assigning a user to the GREAT user client	92
Figure 44: Assigning a Light curve object to the GREAT user client	92
Figure 45: GREAT user client settings.....	93
Figure 46: GREAT user client room settings of enable/disable modules and buttons..	93
Figure 47: The GREAT Manager system structure consisting of a web-based front end for administration, the backend services for configuration/data persistence and the local GREAT devices communicating over a REST API with the backend.....	94
Figure 48: Data model of the GREAT Manager backend.....	95
Figure 49: Login and main menu of the GREAT Manager interface.....	95
Figure 50: The lightcurve editor screen allows for customizing light curves.....	96
Figure 51: Schedule editor for creating schedules when certain interventions should be triggered.....	97
Figure 52: Inputs/Outputs and properties of the GREAT-Scheduler extension.....	98
Figure 53: Playlist editor for customizing sound playlists for activation and relaxation sessions.	100
Figure 54: Data download area for downloading packages for offline analysis.	101
Figure 55: Machine learning approach with classifier variants.....	104
Figure 56: Stress Index and intervention events (scent and sound)	104
Figure 57: System trigger points for learning the trigger situation	105
Figure 58: Binary motion data characterization within a time frame.	109
Figure 59: Relationship among two parameters and intervention state (blue...no intervention, orange...intervention)	110
Figure 60: Learning system pipeline from live events to recommendations	111
Figure 61: Extended learning system pipeline for considering historical data points too.....	113
Figure 62: The red lines show the average 24h profile, which is calculated from all motion measures (of room 228 in Neumarkt) since the start of the measurement. The blue lines show the motion measure in room 228 in Neumarkt for one single day. The green lines indicate the normal range, generated from low-pass filter as well as compressing and stretching the time series. The flash symbol indicates where an activation intervention is started and the ZZZ symbol indicates where a relaxation intervention is started.	115
Figure 63: The arrow indicates a point at which an activation intervention would be triggered if an intervention was not already running.	115

Figure 64: In this example, the same intervention is triggered twice directly after each other. If this had happened a second time on that day, the duration and intensity of the intervention would have been adjusted.....	116
Figure 65: Block diagram of the control system.	117
Figure 66: Initial setup of the GREAT system without additional configuration necessary.....	120
Figure 67: Failed search - giving instructions for changing the WLAN.....	120
Figure 68: Main settings screen (left) and WLAN selection screen (right).....	121
Figure 69: Module management (left) and zone management (right)	122
Figure 70: Update handling of the GREAT system components.....	123

16. List of Tables

Table 1: Configuration tables	26
Table 2: RAW value tables	28
Table 3: Aggregated value tables.....	28
Table 4: Login Parameters.....	30
Table 5: Request object types (t)	32
Table 6: Request values (v)	32
Table 7: Command (cmd) parameters	37
Table 8: Description of input events of the music module extension.	53
Table 9: Description of output events of the music module extension.	54
Table 10: Description of parameters of the music module extension.	55
Table 11: Description of input events of the scent module extension.....	68
Table 12: Description of output events of the scent module extension.....	69
Table 13: Description of parameters of the scent module extension.....	69
Table 14: Description of parameters for the repeater script.	74
Table 15: Thermokon "EasySens" SR-MDS BAT specification	75
Table 16: Vitals of light skin algo mode	80
Table 17: Value specifications.....	80
Table 18: Quality value specification	82
Table 19: Inputs of the scheduler extension	98
Table 20: Outputs of the scheduler extension	98
Table 21: Parameters of the scheduler extension.....	99
Table 22: Performance metrics of classifiers for the given feature vector on the field test training set	111