Hochschule für Technik
Stuttgart

Schellingstrasse 24
D-70174 Stuttgart

T +49 (0)711 8926 0
F +49 (0)711 8926 2666

www.hft-stuttgart.de
info@hft-stuttgart.de

Hochschule
für Technik
Stuttgart

HFT Stuttgart
iCity

Smart Public Building 2018

Proceedings

1st Conference on Smart Public Buildings &
openHAB Foundation Academic Initiative Workshops

October 19-20, 2018
Stuttgart, Germany

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Hochschule für Technik
Stuttgart

Smart Public Building 2018
Conference Proceedings

Volume 156

Stuttgart, December 2018

Publisher:
Prof. Dr.-Ing. Dieter Uckelmann
University of Applied Sciences Stuttgart
Faculty Geomatics, Computer Science and Mathematics
Schellingstraße 24, D-70174 Stuttgart
Phone: +49 (0)711 8926 2632
Fax: +49 (0)711 8926 2556
E-Mail: dieter.uckelmann@hft-stuttgart.de
www.hft-stuttgart.de

ISBN: 978-3-940670-66-3

Preface

In the last few years, smart home technologies have gained a broader user acceptance. Besides being distributed in private homes, these new technologies have the potential to complement classical building automation for public and commercial buildings – especially regarding the retrofitting of existing buildings. The research project Smart Public Building at the University of Applied Sciences Stuttgart addresses the requirements of public institutions for utilization of smart home technologies. It is intended to develop new approaches and concepts based on openHAB, an established open source platform for smart home automation. The decision to build on open software is not only due to vendor independency, which is a point regarding the heterogeneous, mostly closed smart home solutions on the market. But moreover open and free software provides the opportunity to develop extensions for the project's specific needs and then share the results with others who want to apply or develop them further.

These proceedings contain papers related to research and best practices in the area of smart (public) buildings and openHAB, with contributors from academia as well as from industry. All contributions, and even some more, were presented at the first conference "Smart Public Building" that was held on October 19, 2018, at the University of Applied Sciences in Stuttgart, right before Smart Home Day and EclipseCon Europe in Ludwigsburg. The spectrum covered by the presented topics and fields – from automation, standards and functions of smart buildings, e.g. regarding comfort, energy management and privacy, to multi-user and distributed systems, indoor positioning, self-learning systems as well as visualization – indicates a huge necessity and thus relevance of research in the area.

With this publication we are pursuing the overall goal to make scientific works and results related to smart buildings open accessible and therefore publicly available. It is furthermore part of our effort to promote the work of junior and senior scientists, giving them the opportunity to present and discuss their theses. Last but not least we want to facilitate a community-building process for universities, institutions and private stakeholders to leverage different experiences and approaches of smart buildings. A first step has been taken by organizing and hosting the first Smart Public Building conference as well as the openHAB workshops for users and developers, which took place the day after. Further steps to build an academic community related to research on smart building infrastructures will follow.

Finally, we would like to thank the German Federal Ministry of Education and Research (BMBF), which is funding research on smart public buildings through the Smart Public Building project (FKZ 13FH9E03IA), as part of which this conference was organized and held.

Dieter Uckelmann, Myriam Guedey

November, 2018

Contents

1 SMART PUBLIC BUILDING – CHALLENGES AND FIRST FINDINGS

D. Uckelmann [a, *], B. Wohlfarth [a], M. Guedey [a]

[a] Department of Geomatics, Computer Science and Mathematics, University of Applied Sciences Stuttgart
Schellingstraße 24, D-70174 Stuttgart (Germany),
(dieter.uckelmann, 82wobe1mul, myriam.guedey)@hft-stuttgart.de

KEY WORDS: Smart Public Building, Automation, Sensor, openHAB, Smart Home

ABSTRACT:
Recent years have seen a rapid increase in the use of innovative smart home technologies, gaining more and more user acceptance. Besides being used in private homes, these smart technologies may also complement classical building automation in public and commercial buildings. The project Smart Public Building (SPB) at the University of Applied Sciences Stuttgart addresses the needs of public institutions for the utilization of smart home technologies. It will develop and prototypically implement new concepts and applications based on openHAB, an established open source platform for smart home automation. The following paper outlines the overarching goal of the project, addresses its scientific and technical objectives and further presents first findings. Special attention will be paid to concepts that have already been created in subprojects affiliated to SPB. Finally, a conclusion of the course of the project is drawn and an outlook given.

1.1 Introduction

1.1.1 Current State

According to Statista, sales in the smart home market in Germany amount to approximately EUR 697.3 million in 2020 and will reach a market volume of EUR 2,457.1 million and a penetration rate of 6.23% in 2020 (Statista, 2016). However, the illustrated smart home market only includes sale of networked home automation devices and related services (e.g. software usage, monitoring services) to private end users. Public buildings and an assessment of the potentials of smart home technologies for corresponding smart public buildings are not subject of this study.

The smart home incorporates control, monitoring and control devices that are connected to the Internet via a central control unit or directly to control the Internet of Things. Investments in technology have been made in the private sector by enthusiasts so far. Smart home components that enhance home comfort are currently often luxury goods for a niche market (Viani et al., 2013). Economic considerations step into the background.

So far the German smart home market is very fragmented. Whether individual technologies and providers will prevail and which these are remains to be seen. A current commitment to one or the other provider could therefore turn out to be a bad investment in the short term. Furthermore, the different product lifecycles of buildings and ubiquitous technologies pose a challenge. Current technologies can become obsolete in five years, while buildings – especially in the public sector – often have life cycles of 50 years or more. One requirement for more planning security in dealing with obsolescence is manufacturer independence with regard to technologies and services in the smart home (Stengel, 2015). Open source-based approaches offer a possible solution (Hsien-Tang, 2013).

1.1.2 Motivation

With sensor and control technology in smart homes, energy consumption can be reduced, the comfort of use increased and the building's value increased. In recent years, developments in the field of smart homes have

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 1–13,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

been detached from classical building automation. Instead of a structured but complex, mostly cabled automation technology in the private sector, wireless “single-purpose” solutions have prevailed, achieving high growth rates through simple installation, app control and use of the Internet of Things.

While private homes benefit from the dynamic development of new sensors, actuators, interfaces and applications, their use in commercial and public buildings has fallen short of expectations. Contributing to this are complex installations, a difficult-to-evaluate cost-effectiveness, heterogeneity of solutions and top-down planning without involvement of users.

In order to reduce energy consumption and CO2 emissions of commercial and public buildings while increasing their utilization and comfort, it is necessary to develop new approaches for the use of smart technologies. In doing so, specific challenges of larger institutions such as availability, interoperability, administration and maintainability must be mastered.

1.1.3 Project Goals

As part of the Smart Public Building project, corresponding requirements are being identified and examined. Furthermore, new applications for smart home components in public buildings are going to be developed where the University of Applied Sciences serves as example in a bottom-up approach – involving relevant user groups as well as developer communities in the field of smart buildings. The prototypical installation within the college also creates a showcase for other colleges and public institutions to apply and develop the tested scenarios further. For the server and client infrastructure an open source approach based on openHAB is chosen. openHAB is a Java and Eclipse-based application for integrating and controlling heterogeneous smart home components. This ensures that the project benefits from the work of an already existing global developer network and on the other hand contributes to this community with its own developments, as well.

This report presents first results and concepts for the use of smart home components in public buildings.

1.1.4 Scientific and Technical Objectives

The scientific questions to be answered in the course of the project are as follows:

1. What are the special **requirements** of smart public buildings for the use of smart technologies? First, a classification (e.g. museums, schools, offices) of public buildings is made, and an application matrix is created, which will be extended by user participation throughout the project.
2. Which smart home **technologies** are suitable for smart public buildings? The degree of eligibility is determined using a triple bottom line (environmental, social and economic aspects).
3. What special **opportunities** do public buildings offer in the use of smart technologies? On the one hand, this concerns integration and networking of other (partially) public infrastructures, such as e-charging stations, car and bike sharing and parking lots. Secondly, the role of public buildings as part of the smart technology participation and dissemination process will be explored.
4. What **impact** do smart technologies in public buildings have on privacy and how can proactive protection of privacy be ensured?

The project’s technical work objectives are implemented prototypically at the University of Applied Sciences. Developed components should be made accessible to other public institutions online. In detail, the following goals should be achieved:

1. Provision of a basic software and development environment based on openHAB and Eclipse SmartHome.
2. Assembly of a room sensor based on established processor boards (for example Arduino, Raspberry Pi).
3. Development of specific example applications for smart public buildings.

1.1.5 State of Research

Home and building automation are similar in function, but increasingly different in the technologies used. According to VDI 3813 Part 2, typical sensor functions of building automation include monitoring of presence, window status, dew point, air temperature, brightness, air quality, wind speed and precipitation. Controlled by actuators are light, sunblind and drives (e.g. for fans). However, the technologies and protocols used differ

greatly between home and building automation. So far, approaches to harmonize the communication of heterogeneous components by standardized network protocols (e.g. BACnet – building automation and control networks, see DIN EN ISO 16484) turn out rather as entry barrier for innovative young enterprises in the field because of their complexity. As a consequence, only few smart home components available on the market support BACnet, LonWorks, EIB/KNX or OPC UA. Though these systems can provide Internet connection, they work largely independently from the Internet (Stengel, 2015). The smart home, on the other hand, builds upon wireless sensor networks (e.g. ZigBee, Z-Wave, Bluetooth LE), Internet-based standards (e.g. IPv6, 6LoWPAN), apps and lightweight interfaces (e.g. RESTful). In addition, classic building automation requires precise, long-term planning as well as cabling that is usually expensive and therefore reasonable for new buildings, whereas the development of the smart home is modular and based on wireless technologies, showing promise also for the equipment of existing facilities.

Energy management in buildings can make a substantial contribution to reducing energy consumption and CO2 emissions. According to Directive 2010/31/EU, buildings account for around 40% of energy consumption in the EU. The EU project Bricker is dedicated to energy reduction in public buildings through new heating and cooling technologies. However, there are also potential savings when using existing systems. For example, heating can be regulated down and light can be switched off in unused rooms – to predict the use of a building an adaptive control based on intelligent methods is necessary. But requirements for energy management in buildings go beyond heating regulation and lighting control. In cases where selective measurements aren't sufficient, smart metering enables a time- and cost-oriented control of electric consumers to determine potential savings. Seattle's Smart Building Center offers public and commercial building operators the option of borrowing a range of sensors and gauges. Furthermore, the integration of photovoltaic systems as well as of energy storage devices and consumers has become more important in recent years. Electric vehicles with their batteries are ideal for integration into the smart energy management of buildings (Paetz et al., 2010). In this context, a cooperation with the parallel i_City project on e-Bike sharing and the Hofdienergarage in Stuttgart, a car park with charging stations, is planned.

In addition, the needs of individual stakeholders (facility managers, space users, external partners) must be taken into account and the best compromise between comfort, economy and environmental compatibility must be achieved (Kim 2014, Carreira et al. 2014). For this purpose, it is necessary to gain information about usage behavior (e.g. room occupancy) and to involve users participatively (Yu et al., 2013). Comfort can be determined via user feedback by means of an app. The assessment of the perceived temperature in lecture rooms via students as social sensors, for example, can then be incorporated into the controller. By evaluating the actual use of space, spatial planning can be improved.

One particular interest in public buildings is data protection. Public entities are not only bound by relevant laws, but also play a pioneering role in public perception. A distinction should be made between conscious and unconscious (ab)use of data. Functional benefits and data protection issues therefore have to be considered equally from the outset, including systems for backup and control. On the other hand, recommendations of public authorities often appear to be outdated. For example, the State Criminal Police Office NRW (LKA North Rhine-Westphalia, 2014) advises to only connect devices to the Internet if necessary, e.g. for updates. However, in the Internet of Things era other measures are required.

1.2 Current Research Status

1.2.1 Classification of Public Buildings and Questionnaires

People stay up to 90% of their time in private or public spaces and buildings (Böwing, 2015), and visitors of public facilities tend to request comfort in such buildings just as in their private homes. The integration of smart home applications into public buildings may facilitate not only comfort, but also aspects of sustainability, such as the conservation of resources. These applications make it possible to detect and rectify faults faster, to respond to external influences immediately and to control the entire building technology automatically and on demand. Users and operators might therefore benefit from an increased comfort, assistance in daily routines and cost advantages. A study by *trend: research* entitled "Smart Building, Intelligent Commercial and Industrial Building Automation in Germany by 2025" amounts the demand for intelligent building systems in the industrial and public building sector to 32% (Smart Home Magazin, 2016). This number indicates a huge interest as well as a need for action to integrate and explore the possibilities of smart technologies in this area.

The aim of the undertaken classification and survey was to clarify which existing applications in the smart home sector could be used in public buildings, whether and to what extent smart applications in public buildings are

already being used and what new requirements must be met during installation. Technical managers of 50 public buildings of the city of Stuttgart were interviewed. The result shows that smart applications in public buildings are not yet widely used.

Beforehand, the public buildings were classified in order to differentiate between building types reasonably.

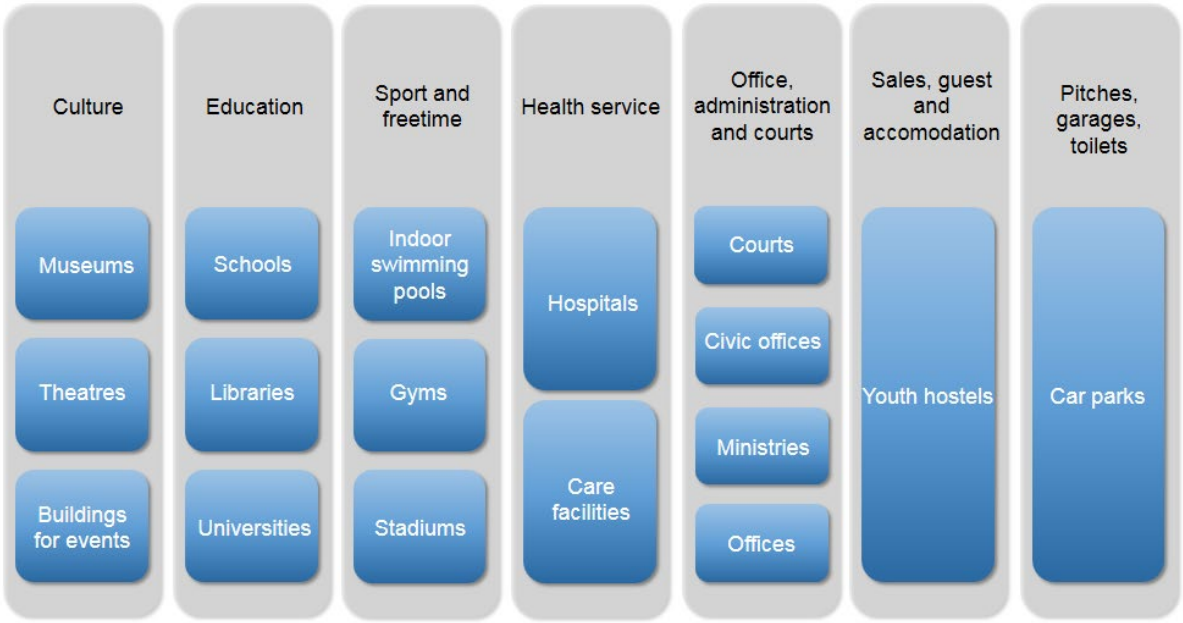


Figure 1. Building classes of public buildings

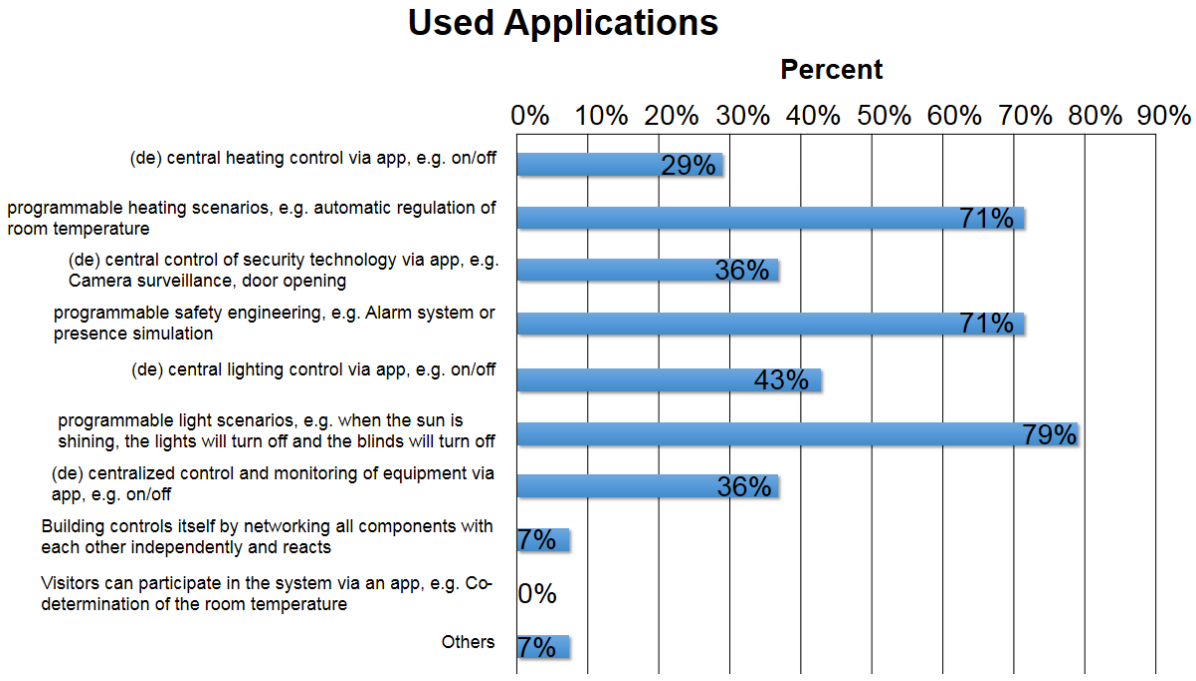


Figure 2. Applications used in public buildings (n = 14)

Current use of smart technologies in public buildings is limited to applications of classical building automation. Data transfer is largely wired and remote control via Internet is not quite an option. Many smart homes are one step ahead. Besides that, the use of smart applications is regarded critically by most respondents.

1.2.2 Implementation of Location-based Services with Bluetooth Beacons

Location-based services (LBS) provide information to users of mobile devices depending on their current location. The most popular application of LBS is navigation via the Global Positioning System. Location-based services are subdivided into reactive and proactive services (Markgraf, 2018). Reactive services must be requested by the user, e.g. search queries in the immediate vicinity. Proactive services, such as retail coupons, will be automatically made available when the user enters a certain zone. To determine the user's location, various technologies can be used. Within buildings an alternative to GPS has to be considered, as building materials such as concrete interfere with the GPS signal.

One of the technologies that can be applied both inside and outside the building is localization via Bluetooth beacons. Beacons are small hardware transmitters that communicate with mobile devices via Bluetooth Low Energy (BLE). For this communication to take place, it is necessary to have a corresponding application installed on the mobile device that can interpret the beacon's signal and perform appropriate actions.

Within SPB, the aim is to simplify orientation in public buildings with the help of Bluetooth beacons. Visitors should therefore find their way around faster and get relevant information displayed directly on their mobile devices. At the HFT campus, Bluetooth beacons were temporarily installed in lecture halls and hallways as a proof-of-concept. Via indoor navigation, visitors should be able to access maps with a marked route and a search function via their smartphone.

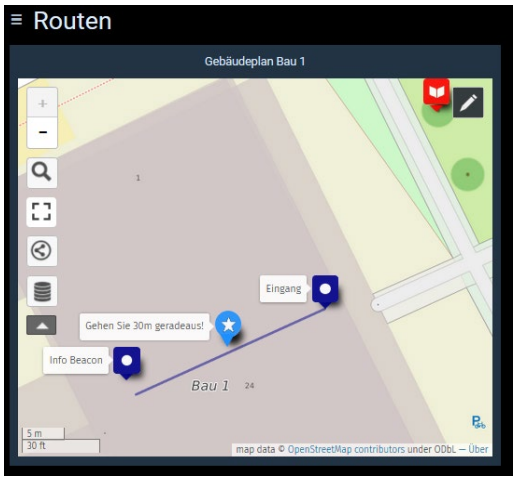


Figure 3. Indoor navigation via BLE

In addition, visitors of a room should receive selective access to data of that room, e.g. occupancy, sensor data (temperature, humidity, CO2) as well as information about the respective event. Bluetooth beacons can therefore be installed in front of lecture halls, and visitors then get the relevant information on their smartphones using openHAB (see Figure 4).

Furthermore, a help function should be offered to present the building's layout and provide information on behavior in emergency situations.

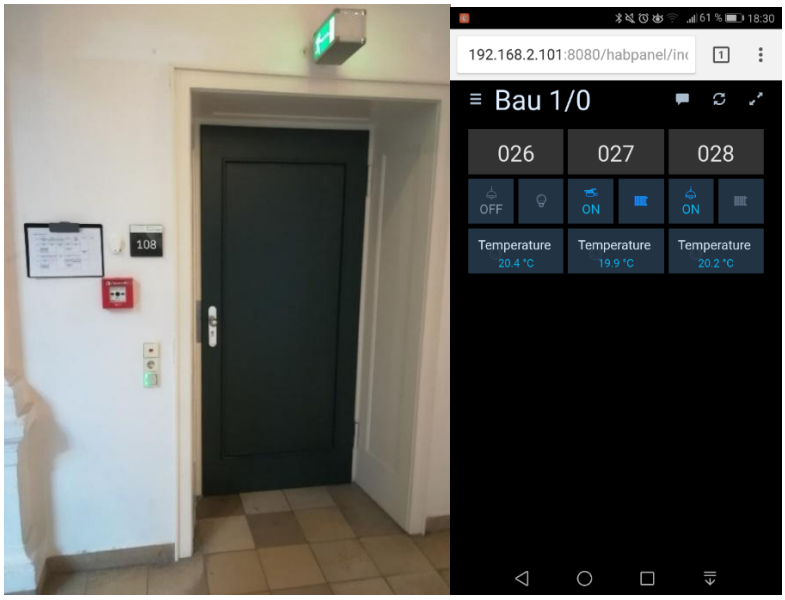


Figure 4. Display of room information in openHAB

1.2.3 Use of Social Sensors

Usually, the control of conditions in public buildings is determined by different sensors. The regulation of the room temperature, for example, is coupled with measurements of temperature sensors. Here, attention is often paid only to the prevailing, absolute room temperature. According to this temperature, the heating adapts itself. In other cases, the room temperature is controlled by a central unit of building automation. Adapting temperature according to measurements by sensors goes by absolute values – individual "perceived" values are usually ignored, why there can be a difference between measured and perceived temperature (Aschendorf, 2014).

Especially in cold months, heaters are often set to a higher temperature without regard to current outside conditions. The public utility does not need to measure and reconcile values, but automatically turns on the heating over the winter months. Here, it is ignored whether it is a warmer or colder day and whether activation of the heating is required at all. In addition, during warmer days, employees of public institutions may open the windows because the heating is set too high. The relatively large area of public facilities thus consumes a great deal of energy.

In this proposal, room users are included as "social sensors". A voting application using a smartphone should allow room users to vote on "perceived" values. As a consequence, there should be changes of state in space, e.g. automatic deactivation or activation of the heating. For this purpose, a threshold value is set, which triggers a state change when it is reached.

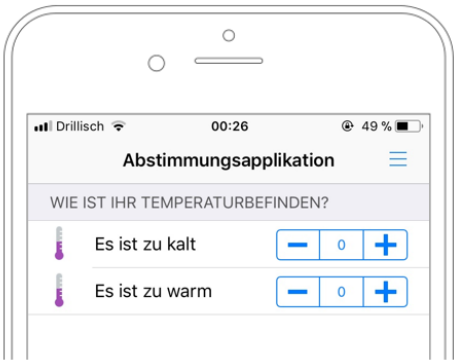


Figure 5. Voting application in openHAB on a smartphone

The professor of each lecture acts as an authoritarian entity involved in the process either through manual intervention, or passive monitoring of the voting results.

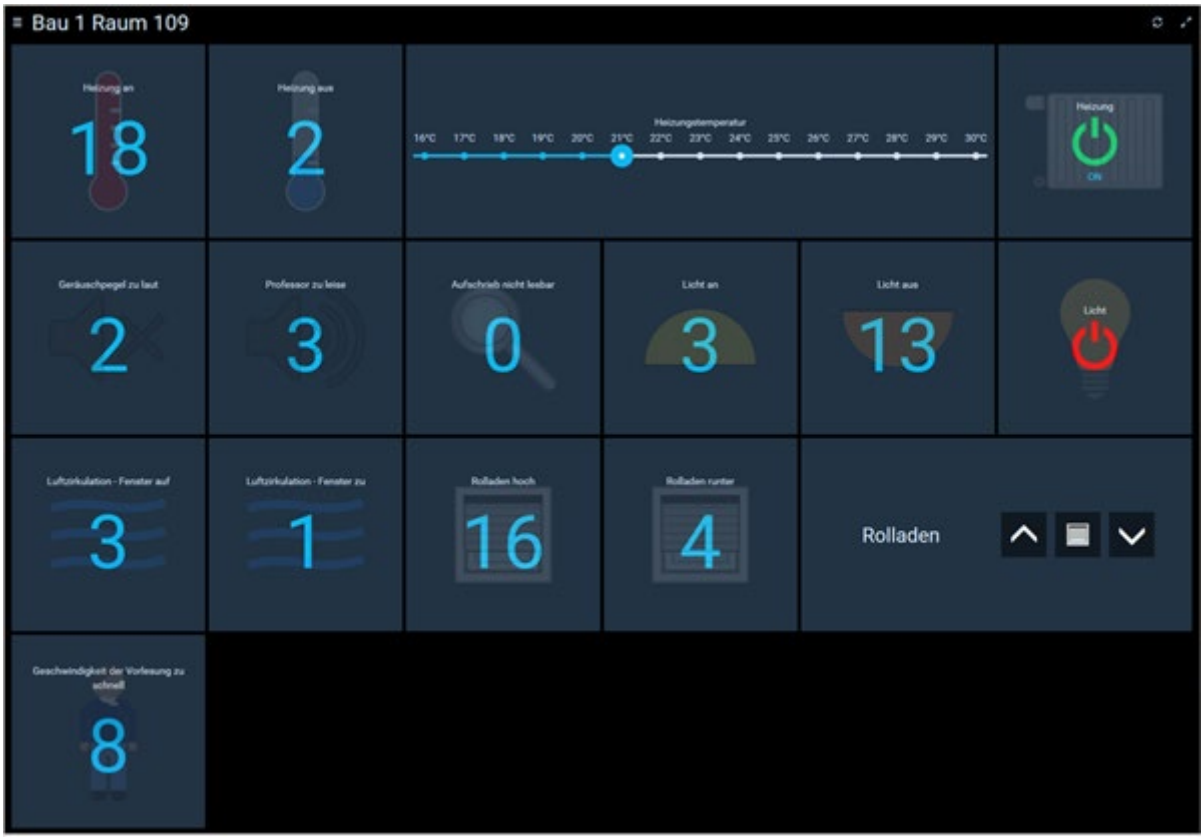


Figure 6. A HABpanel view of a lecture room, including real sensor data, perceived noise levels, ratings of lecture speed and other data influencing students

An application that presents a vote on the conditions in a lecture room has been created in openHAB, using the creation of items and their graphical representation. Generated automations pick up the values of a vote, summarize them and trigger events if certain conditions occur. The resulting prototype should be further developed to ensure usability in practice. A user administration and the integration of a tool for a more comprehensive visualization of the voting results are necessary extensions.

1.2.4 Damage Detection in Public Buildings

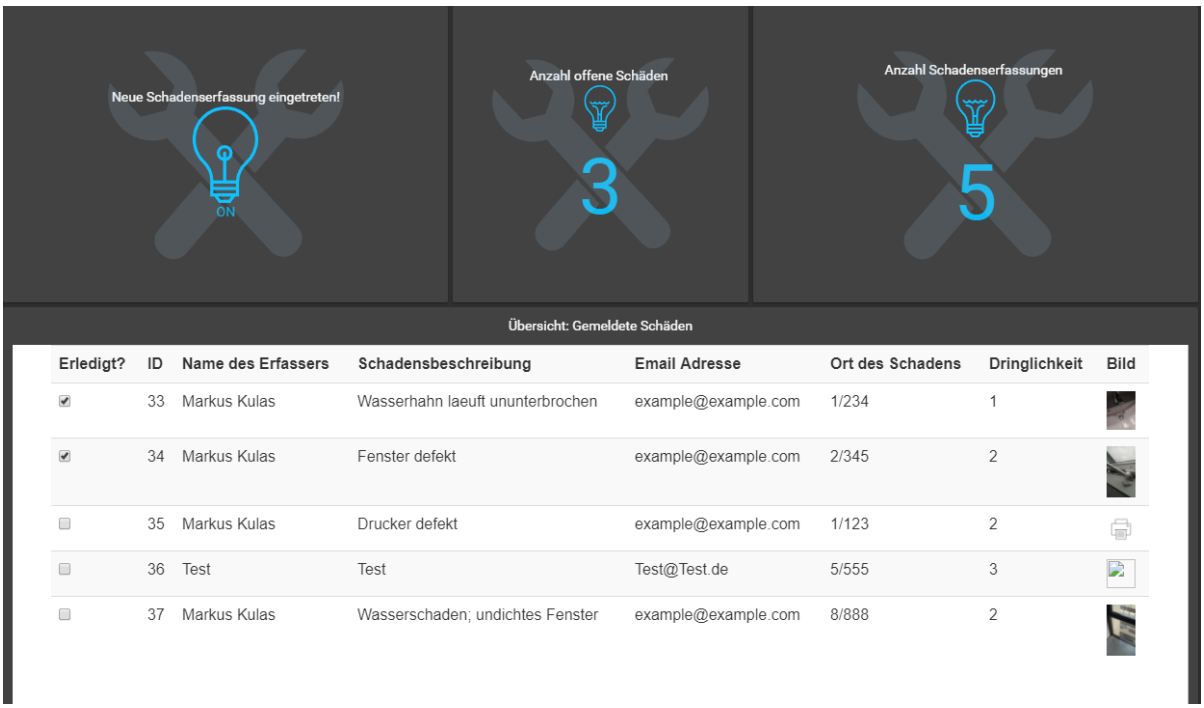
Inspection and maintenance of public buildings, often due to cost-saving measures, are challenging tasks for municipalities and those responsible for the facilities. Buildings and their equipment age, inherently they wear and loose function during this process (Offensive Gutes Bauen, 2018). It is therefore essential to deal with the associated problems.

Defective radiators, nonfunctional shutters or other missing (wear) equipment at colleges, universities or libraries are just a few examples of unavoidable damage.

Damage detection in public buildings can be achieved in different ways. Most often damage is not recognized by the responsible staff first, but rather by persons who use the affected premises on a daily basis. However, reporting of damages to the responsible personnel of a building is usually omitted. As a result, responsible staff is either late or even not aware of occurred damages and a removal may therefore be delayed accordingly.

So far, there is no standardized way to report damage or other deficiencies to the facility management. To counteract this problem, a concept has been developed, that allows damage detection in public buildings using a smartphone. The data can be made available to the responsible personnel via openHAB. Detected damage should be recorded as simple and general as possible. By reducing the effort for the user, a higher motivation to participate in the removal of damage shall be achieved, resulting in the long-term preservation of a public building.

The building management can retrieve the damage reports via smartphone as well. By means of a classification form, it is possible for the person in charge to complete the tasks prioritized. A damage can be marked as completed after rectification and triggers a notification to the damage detector.



Erledigt?	ID	Name des Erfassers	Schadensbeschreibung	Email Adresse	Ort des Schadens	Dringlichkeit	Bild
☑	33	Markus Kulas	Wasserhahn laeuft ununterbrochen	example@example.com	1/234	1	
☑	34	Markus Kulas	Fenster defekt	example@example.com	2/345	2	
☐	35	Markus Kulas	Drucker defekt	example@example.com	1/123	2	
☐	36	Test	Test	Test@Test.de	5/555	3	
☐	37	Markus Kulas	Wasserschaden; undichtes Fenster	example@example.com	8/888	2	

Figure 7. Presentation of damage reports in openHAB

A timely detection of damage by involving a building's users can lead to quicker damage repair and thus supports the error-free operation of a public building. Furthermore, standardization and simplicity of reporting increases data quality and facilitates usability and comfort on both sides – the detectors as well as the building management. Sources of error as well as transmission times are minimized, which leads to better documentation.

1.2.5 Collection of Safety-related Facilities

The primary goal of smart home solutions is to provide residents comfort and convenience while increasing the home's energy efficiency as well as its safety. However, requirements of public buildings are different from those of a private home. The focus of this subproject is on intelligent energy management and safety aspects, such as devices intended to guarantee or assist security (Hoffmann, 2017).

Security-specific devices may be a fire extinguisher, defibrillators, first aid kits or fire and smoke detectors. These facilities must be serviced and checked at regular intervals. The goal was to develop an application via openHAB that should facilitate the management of collections of safety-relevant facilities in buildings. Therefore, the facilities have been equipped with contact sensors in order to detect the removal of a device, e.g. a fire extinguisher or defibrillator.

Figure 8. Contact sensor attached to a fire extinguisher

A PHP form, which is presented via the openHAB interface, represents the data sheet of a device. It includes all relevant information, such as the next due date for maintenance.

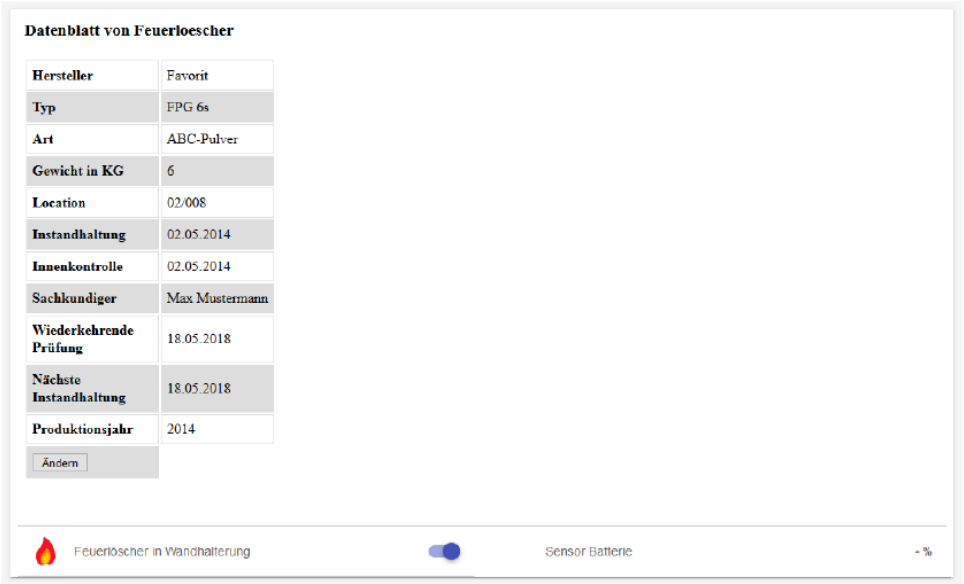


Figure 9. PHP form of a fire extinguisher

Push notifications will inform responsible persons.

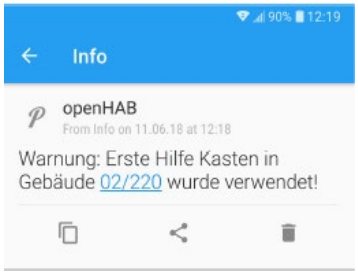


Figure 10. Push notification via openHAB

For navigation aid, a Near Field Communication (NFC) transponder can be attached to the device. Using a NFC enabled smartphone, it is possible to navigate to the device's datasheet in the app directly. This simplifies identification of security-relevant devices in everyday use. Writing the information to the transponder is done via the openHAB app. Once the transponder has been written, it is possible to navigate to the selected sitemap by touching it with the smartphone.

1.2.6 Implementation of an Intelligent Charging Station for e-Bikes

With the rise of e-mobility, electric vehicles, scooters and bicycles have become part and parcel of today's traffic. Electric bikes, for example, can easily be an alternative to cars for distances up to 30 kilometers. They are more environmentally friendly and have great potential to close the gap between the regular bike and the car. Test drivers of electric bikes were more motivated to replace the car with an electric bike instead of a regular bike. And there are even more reasons for buying an e-bike, such as "cycling without sweating", "being mobile without harming the environment" and "driving less" (Neupert et al., 2013).

This scenario deals with the design and implementation of an intelligent electric charging station for e-bikes at the University of Applied Sciences Stuttgart. With the help of openHAB, a communication between the charging station and the user is to be created. The integration in openHAB gives the user the opportunity to track the state

of charge of his electric bicycle. The charging station should be able to display via openHAB whether it is available for charging or whether it is occupied, and via push notifications the user could be informed about the station's status. The scenario also includes a billing system with incentives, e.g. discounts for specific user groups such as university employees.

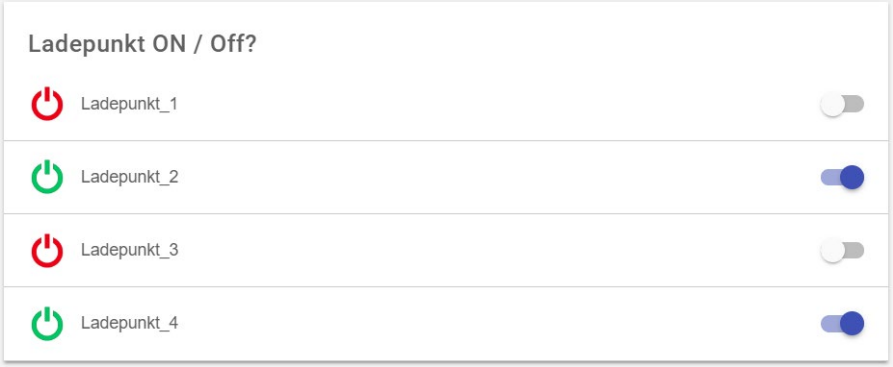


Figure 11. Status of charging points displayed in openHAB

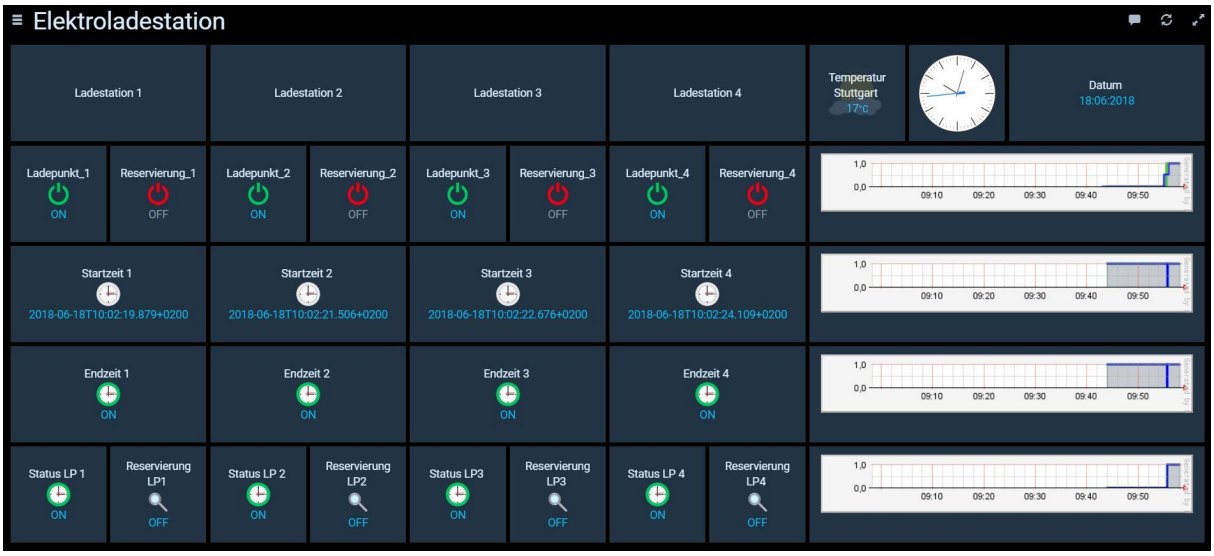


Figure 12. Illustration of electric charging stations via HABpanel

The billing system should integrate various payment methods and differentiate between user groups, e.g. external and internal users (from university's perspective), to allow specific benefits. An associated electricity tariff and billing to the minute should make the charging station available as soon as possible for subsequent users. With the help of an interface, the data can be exchanged between openHAB and the billing system and an invoice for charging can be generated. At present, there exist several free and open source projects for billing scenarios of all kinds. It has to be examined, which meets the described requirements best.

1.2.7 Smart Lecture Rooms

Only through data, which are recorded in buildings, a smart city can emerge (Mertens, 2017). Sooner or later public institutions are thus obliged to record various data via smart technology. Thereby they can contribute significantly to the development of a smart city. The newly built city library in Stuttgart, for example, has been equipped with 200 high-frequency presence detectors for energy-efficient lighting control in order to use artificial light only when necessary (Gebäudedigital, 2013).

A similar, even more extensive "Smart Lecture Room" system is currently tested at the HFT Stuttgart. A hardware platform for the monitoring of functional rooms (lectures, sessions, etc.) has been assembled and

linked to openHAB. The installed sensors detect window contact, measure power consumption as well as temperature (for controlling the heater), light, presence and humidity (latter gathered by multi-sensors) and report them to openHAB. The aim is to increase energy efficiency of the monitored rooms as well as to improve comfort of use and room utilization. Three lecture rooms have been equipped with sensors so far. In a next step, long-term measurements have to be carried out and analyzed in order to gain information about usage patterns.

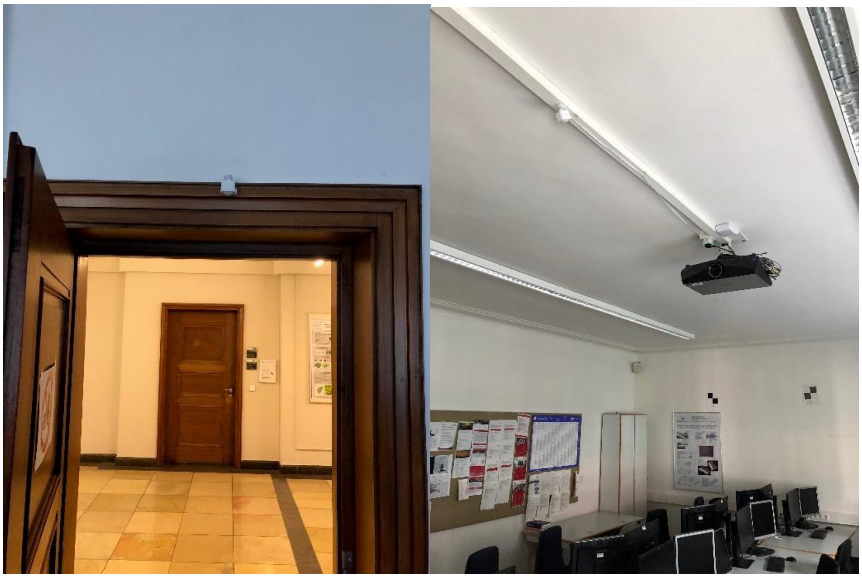


Figure 13. Multi-sensor attached above a door

The Z-Wave network accesses the various sensors from a central server. Through the use of Z-Wave extenders the signal is amplified and repeated over the different rooms.

Using a MySQL database, the sensor data is collected and stored persistently. The data is displayed via the external visualization tool Grafana and presented via the openHAB user interface.

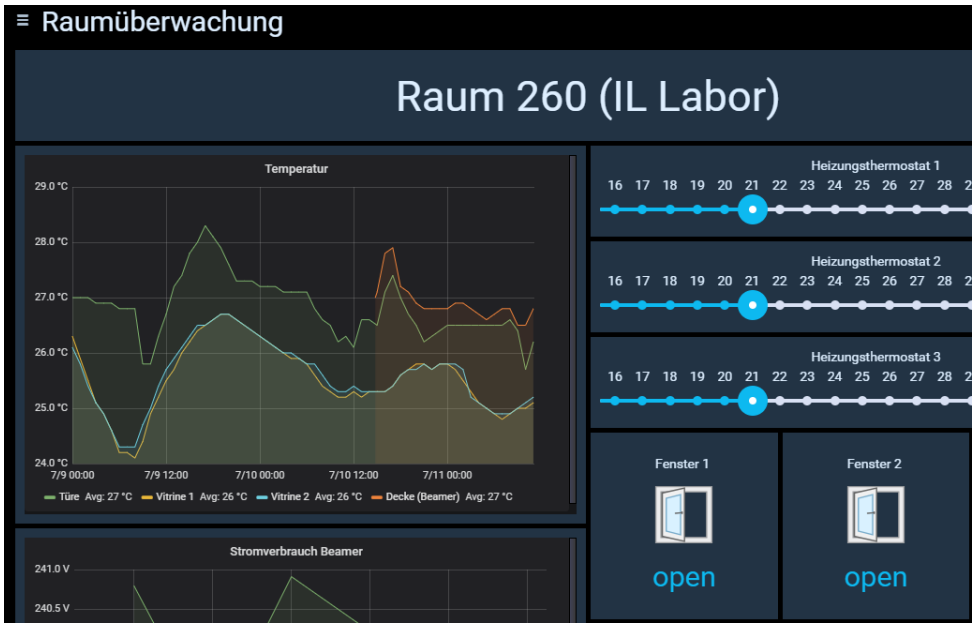


Figure 14. Representation of sensor data via HABpanel

In the future, by determining rules, the lecture rooms can be managed based on the usage patterns. Various scenarios can be applied here, such as turning off the heating at night, or the general regulation of a room in case of presence detection.

1.3 Conclusion and Outlook

Smart Public Building focusses on the applicability of smart home technologies for public buildings. The prototypical implementation of such technologies at the HFT campus will ideally have a positive effect in terms of adoption and further development at other universities and public facilities. Extending the open source system openHAB for home automation to public buildings offers above-average prospects of technical success and dissemination of the developed extensions. In addition, with the openHAB community there is a backup of continuation beyond the project.

First concepts and basic installations within subprojects have been taken place. However, further investigations on the central use of the openHAB infrastructure in public buildings are required, taking into account specific (also technical) demands of public facilities, such as network restrictions.

Affiliated to the Smart Public Building project further bachelor's and master's theses will be created. One of the next steps includes development and prototypical implementation of a concept for automatic presence detection in public buildings. Furthermore, the power and water meter of the University of Applied Sciences (building 2) should be equipped with sensors to measure consumption continually.

The already existing concepts will be further developed and put into operation prototypically in the course of the project. For example, lecture rooms will be equipped with CO2 sensors as well as with intelligent light switches. Furthermore, access to the installed openHAB prototypes via smartphones and tablets will be made possible.

1.4 References

Aschendorf, B., 2014. *Energiemanagement durch Gebäudeautomation: Grundlagen - Technologien - Anwendungen.* Springer Fachmedien, Wiesbaden.

Böwing, R., 2015. *SmartHome und SmartBuilding - Zukünftiges Wohnen und Bauen*. https://www.makeyourhome.de/story-artikel/smarthome_und_smartbuilding_-_zukuenftiges_wohnen_und_bauen-56 (accessed December 1, 2016)

Carreira, P., Resendes, S., & Santos, A., 2014. Towards automatic conflict detection in home and building automation systems. In: *Pervasive and Mobile Computing*, Vol. 12, pp. 37-57.

Gebäudeautomatation, M. -M., 2012. *Marktstudie Gebäudeautomation Schweiz 2012.* http://www.mega-planer.ch/fadaladdondlz/files/.addonpublikationeintragfile/publikationen/63.pdf/Marktstudie%20Geb%C3%A4udeautomation%20Schweiz%202012.pdf (accessed April 4, 2016)

Gebäudedigital, 2013. *Gebäudedigital*. https://www.gebaeudedigital.de/allgemein/stadtbibliothek-stuttgart/ (accessed January 18, 2018)

Hoffmann, D. A., 2017. *Smart Home und Smart Building.* https://www.bitkom.org/Presse/Anhaenge-an-PIs/2017/02-Februar/Bitkom-Pressekonferenz-Smartphone-Markt-Konjunktur-und-Trends-22-02-2017-Praesentation.pdf (accessed May 10, 2018)

Hsien-Tang, L., 2013. Implementing Smart Homes with Open Source Solutions. In: *International Journal of Smart Home*, Vol. 7(4), pp. 289-296.

Kim, B., 2015. Consensus-Based Coordination and Control for Building Automation Systems. In: *IEEE Transactions on Control Systems Technology,* Vol. 23(1), pp. 364-371.

LKA North Rhine-Westphalia, 2014. *Smart Home und Connected Home: Empfehlungen zur Sicherung digitaler Haustechnik.* https://www.polizei.nrw.de/media/Dokumente/Behoerden/LKA/140811_LKA_SmartHome_Empfehlungen.pdf (accessed November 19, 2018)

Markgraf, D., 2018. *Definition »Location-based-Services«. Gabler Wirtschaftslexikon.* Springer Fachmedien Wiesbaden.

Mertens, O., 2017. *Haufe*. https://www.haufe.de/immobilien/wirtschaft-politik/smart-building-als-baustein-zur-smart-city/smart-building-als-baustein-zur-smart-city_84342_410364.html (accessed January 18, 2018)

Neupert, H., Schröder, J., & Schulz, M., 2013. *The eBike Book. Future. Lifestyle. Mobility.* teNeues, Kempen, pp. 15-59.

Offensive Gutes Bauen, 2018. *Offensive Gutes Bauen*. https://www.offensive-gutes-bauen.de/praxishilfen-und-unterstuetzung/das-instrument-fuer-gute-kommunikation-und-kooperation-aller-am-bau-beteiligten/praxishilfen-der-inqa-bauen-partner/ (accessed January 15, 2018)

Paetz, A., Jochem, P., & Fichtner, W., 2010. Smart Home & E-Mobility - Effekte von Anreizsystemen. In: *VDE-Kongress 2010 - E-Mobility: Technologien - Infrastruktur - Märkte.* VDE, Leipzig.

Smart Home Magazin, 2016. *Intelligente Gebäudetechnik heute und morgen*. http://smarthomemagazin.eu/studie-zeigt-wachstumspotenzial-fuer-intelligente-automation-in-gewerbegebaeuden/#more-699 (accessed November 30, 2016)

Statista, 2016. *Smart Home*. https://de.statista.com/outlook/279/137/smart-home/deutschland# (accessed April 4, 2016)

Stengel, B., 2015. Ethische Überlegungen zu Smart Home. In: *International Review of Information Ethics,* Vol. 22, pp. 92-100.

Viani, F., Robol, F., Polo, A., Rocca, P., Oliveri, G., & Massa, A., 2013. Wireless architectures for heterogeneous sensing in smart home applications: Concepts and real implementation. In: *Proceedings of the IEEE,* Vol. 101(11), pp. 2381-2396.

Yu, D.-Y., Ferranti, E., & Hadeli, H., 2013. An intelligent building that listens to your needs. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing,* ACM, New York, pp. 58-63.

2 CAMERA-BASED FALL DETECTION SYSTEM WITH THE SERVICE ROBOT SANBOT ELF

J. Bauer [a, *], L. Gründel [a], J. Seßner [a], M. Meiners [a], M. Lieret [a], T. Lechler [a], C. Konrad [a], J. Franke [a]

[a] Institute for Factory Automation and Production Systems, Technical Faculty, Friedrich-Alexander University Erlangen-Nuremberg, Egerlandstr. 7-9, D-91058 Erlangen (Germany), jochen.bauer@faps.fau.de

KEY WORDS: Convolution Neural Network, Fall Detection, Keras, Machine Learning, Sanbot, TensorFlow

ABSTRACT:
Due to the demographic change and an aging population the care of the elderly is getting more and more attention, especially with the simultaneously increasing shortage of skilled workers in the health care sector. Improvements in the field of robotics are intended to help with these challenges. The use of service robots promises the possibility to reduce the overall costs of the health care system and enables older people to stay longer in their own homes. Falls among elderly lead to severe consequences for these people. In the course of this work a Convolutional Neural Network is deployed on the service robot Sanbot Elf to detect fallen people.

2.1 Introduction

2.1.1 Service Robotics

The International Federation of Robotics defines a service robot as a robot that performs useful tasks for people or things that are not considered industrial automation (International Federation of Robotics, 2016). For this reason, they are also called “non-industrial robotics” (Decker et al., 2011). The execution of service robots is either semi-autonomous with a robot human interaction or fully autonomous without operational intervention by the user (International Federation of Robotics, 2016). Humanoid service robots offer a further dimension of robot human interaction and are therefore expanding in many fields of application. Due to their human like body, they are better adapted to the human environment, thus leading to a higher acceptance among users (Haun, 2013). A robot is only able to develop human like traits or ultimately thought structures in this way, since the human being himself experiences and learns through the connection of his body to his environment (Haun, 2013). However, the acceptance does not increase with increasing human traits unceasingly. If a robot is adapted to the exterior of a human being, the acceptance of the user decreases again after a certain degree of similarity, which Masahiro Mori calls the *Uncanny Valley* (Mori et al., 2012).

The robot Sanbot Elf of the Chinese company QIHAN and the robot Pepper of the Japanese company SoftBank each represent a prototype of a humanoid service robot. The application of Pepper extends, among other things due to the high acquisition costs of approx. 20,000 euros (Generation Robots, 2018), primarily to the use in shops to welcome, inform and entertain customers (SoftBank Robotics, 2018). Sanbot Elf on the other hand enables further fields of application in the private sector due to its lower purchase price of about 10,000 euros (Cyber Robotics Technology Limited, 2018).

2.1.2 Challenges in Service Robotics

New technologies affect different social and societal levels. On a small scale, they influence the relationships between people or on a large scale social perspectives and traditions. In addition, the economic labor market situation also plays a role, as does the possibility of intervening in the development and introduction of these technologies in order to represent one’s own interests. Becker et al. name here the impact on jobs, general technological development, acceptability and interpersonal communication as factors of social and societal challenges (Becker et al., 2013).

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 15–28,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

2.1.2.1 Impact on Jobs and Acceptability

In order to advance the dissemination of service robots, the greatest possible acceptance within the user groups is necessary. In this context, the professional user group in particular comes to the fore, whose acceptance is of decisive importance for the further establishment of service robots. In the health care sector, the care itself plays a significant role in this respect. Robots are accepted if there is a benefit for one's own work that makes it more efficient and pleasant. At the same time, however, a progressive loss of jobs due to the increasing use of robots is feared. Social upheavals are often driven by comprehensive technological changes. They influence human communication and behavior. The human being adapts to the new circumstances and is therefore rather a passive participant in this process (Becker et al., 2013).

2.1.2.2 Human Communication

Due to technical achievements in recent decades, direct human communication is increasingly being supplemented and replaced by electronic transmission technologies. The widespread use of smartphones and social networks makes the exchange and connection between people uncomplicated and thus shifts communication to another level. In cities, there is a trend towards single and two-person apartments. As a result, older people are no longer necessarily supported by their direct relatives in the case of long-term care. Thus, direct contact with caregivers or assistants, in addition to their actual activities, is of great social importance (Melson, 2010).

In this context, an increasing use of service robots may contribute to two opposing effects. On the one hand, by relieving and assisting with monotonous and heavy work, caregivers may be able to engage more in personal interaction with those in need and thus play a significant social role. On the other hand, the possibility in the increased use of service robots may reduce direct interpersonal contact, as caregivers are less likely to be on site.

2.1.2.3 Ethical and Legal Issues

When considering ethical aspects in the challenges of service robotics, two different fields have to be considered. The circumstances and conditions in the respective application case and the robotics itself. In the first case, the question arises as to whether the use of robots in the application area violates fundamental ethical values. If this is the case, the progressive spread and establishment of these robot systems will at least be made more difficult. The example of the health care system raises questions such as "dehumanization" in the care of elderly people by robots, as this is also referred to as inhuman. In addition, the impression may possibly arise that affected people are excluded from society or that they are regarded as test subjects (Becker et al., 2013). Such challenges need to be considered.

Ethical aspects concerning the robot itself raise the question of whether and to what extent ethical actions should be brought closer to them (Becker et al., 2013). Universally applicable principles do not exist, since they would be neither practicable nor feasible (Guo & Zhang, 2009). Robots are usually designed for a specific purpose, which is why it is probably sufficient to give them a code of ethics adapted to this area (Becker et al., 2013).

Ethical aspects frequently give rise to legal questions and should therefore always be seen in context with one another (Becker et al., 2013). In 1950 the science fiction author Isaac Asimov formulated the Four Laws of Robotics. However, conceived as universal principles, these laws reach their limits in some situations. Therefore, it is necessary to invest more effort and resources into changes of systems or the whole society in order to prevent the occurrence of ethical dilemmas (Helbing 2013, Helbing & Pournaras 2015).

2.1.3 Motivation

According to Carone & Costello (2006), the old-age dependency ratio in the European Union is expected to rise to 54% by 2050. This quotient describes the ratio between the number of people in the population group of 65-year-olds and older relative to the number of 15 to 64-year-olds. Thus, an ever larger group of elderly people is faced with an ever smaller group of young people. Against this background, the focus is increasingly on how to take care of these people. In this context, possibilities for longer care in one's own home also play a significant role. An important aspect here is the recognition of a fallen person. According to Halter et al. (2009), 87% of all fractures of older people are due to falls. According to statistics from the National Council on Aging in the USA, falls resulting in death also increase exponentially with age. This connection is shown in Figure 1. In the 65 to 69 age group, the fall rate is 5.4 (women) or 10.6 (men) per 100,000 inhabitants. In the following two groups, this rate rises to 19 and 34 respectively. However, people aged 85 and older are most severely affected. In this age group the fall rates show the highest values with 106.4 and 153.2 respectively. The difference in the values between women and men can be attributed to the fact that men are more likely to have comorbid conditions than

women of the same age (Stevens, 2007). This presence of several diseases makes them more susceptible to the consequences of a fall (Stevens, 2007).

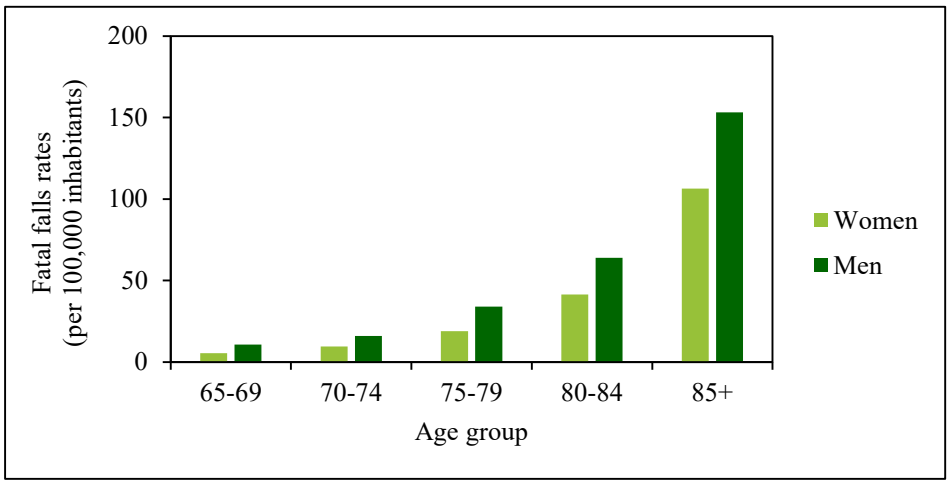


Figure 1. Fatal falls rate per 100,000 inhabitants in the USA in 2001, broken down into age groups and genders (Stevens, 2005)

It may be possible to mitigate the consequences of a fall and reduce the mortality rate by using a system that identifies a fallen person. The amount of time people lie on the ground after a fall is a significant factor influencing the severity of the effects of a fall (Igual et al., 2013). Many elderly people are no longer able to stand up on their own after a fall (Igual et al., 2013). Longer lying leads to symptoms such as painful pressure points, hypothermia or dehydration (Igual et al., 2013). A fall detection system tries to minimize the time spent lying on the ground by automatically initiating appropriate help measures after a fall has been detected.

Another positive effect of using such a system is the general safety it provides to people. After a fall, it is possible for people to develop a fear of falling again (Friedman et al., 2002). This fear of falling increases the risk of falling again (Friedman et al., 2002). It has possible negative effects such as depression, less activities and a generally lower quality of life (Scheffer et al., 2008). Therefore, a fall detection system offers possibilities to mitigate these effects and even prevent falls. In particular, the immediate environment of the elderly, such as either their own home or a care/retirement facility, plays an important role and is therefore defined in this paper as an area of application.

2.1.4 Robot Basics and Platforms

An industrial robot consists of a large number of components that are necessary for its operation. The manipulator represents the basic structure to which all other components are connected (Jazar, 2010). On the hardware side, actuators enable the movements of the robot, which change the configurations of the partial elements against internal and external forces in response to signals from the controller. The sensors of a robot are used to acquire information about the internal and external status.

The core processing unit serves as the core for monitoring and controlling the robot, since all information collected by the sensors converges at this point. Using the data, the processor plans the robot's movements based on the kinematic limitations of the robot and determines how much and how fast the joints and limbs must move in order to achieve a desired position and speed. In addition, the processor monitors the activities carried out together by the controller and sensors (Niku, 2011).

Table 1 gives an overview of currently commercially available humanoid service robots as well as those that are still in the development stage or have been designed entirely for research purposes. Due to the rapid development in this field, it is not possible to present all systems comprehensively, which is why the table does not claim to be complete. Only exemplary current service robots for different areas are listed.

	Name	Manufacturer	Main application area
Commercially available	BUDDY	Blue Frog Robotics	Companion
	Cobalt robot	Cobalt Robotics	Security
	OSHbot	Lowe's Innovation Lab	Retail
	Nao	SoftBank Robotics	Education, Research
	Pepper	SoftBank Robotics	Public
	wGO	Follow Inspiration S.A.	Retail, Public
	REEM	PAL Robotics	Public, Security
	Ramsee	Gama2Robotics	Security
	Sanbot	QIHAN	Public, Retail
Research robots	Care-O-bot	Fraunhofer IPA	Healthcare, Household
	Romeo	SoftBank Robotics	Healthcare, Household
	ASIMO	Honda	Research
	AMIGO	Eindhoven University of Technology	Healthcare, Household
	MOnarCH	Carlos III University of Madrid	Healthcare
	SPENCER	Albert-Ludwigs-Universität Freiburg	Airport
	Atlas, PETMAN	Boston Dynamics	Security
	Rollin' Justin	DLR	Household, Outer space
	Roboy	TU München	Public, Research

Table 1. Overview of commercially available robots and research robots

2.1.5 Machine Learning Basics

With the continuously improved and more cost-effective hardware, more and more application fields and tasks are becoming possible. In order to enable (service) robots to perform complex tasks with a certain degree of autonomy, a form of independent learning is necessary. The field of machine learning offers suitable methods for this.

The development of an artificial neural network (ANN) resulted from the motivation to copy the information processing capabilities of a mammal's brain (Rojas, 1996). Based on the basic structure of neurons, ANNs consist of a multitude of artificial neurons or nodes (Haykin, 1999).

Mathematically, such a mesh is defined as a directed graph having the following characteristics (Bishop 2006, Müller et al. 1995):

- Input signals x_i that are either the output signals of a node i or initial data.
- A weight w_{ij} for each connection between two nodes i and j.
- One bias b_j for each node j.
- An activation function h.

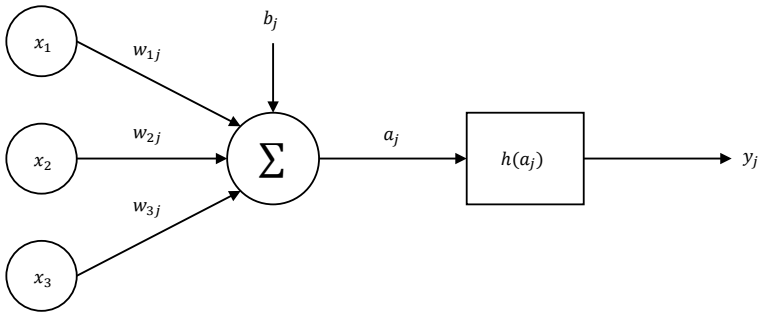


Figure 2. Schematic structure of a nonlinear artificial neuron (Haykin, 1999)

Figure 2 shows an example of a neuron j where three input signals x_1, x_2 and x_3 are multiplied by a weight w_{1j}, w_{2j} and w_{3j} respectively. In addition, for each node a bias b_j is added to the sum of these values. The

subsequent intermediate result a_j is converted into the output signal y_j with an activation function $h(a_j)$. This activation function allows the creation of a nonlinear model.

Figure 3 shows the structure of a simple deep feedforward network that has a single hidden layer and is called *fully connected* because all nodes of a layer are connected to all nodes of the next layer. *Partially connected*, conversely, means that some of these connections are not present (Haykin, 1999).

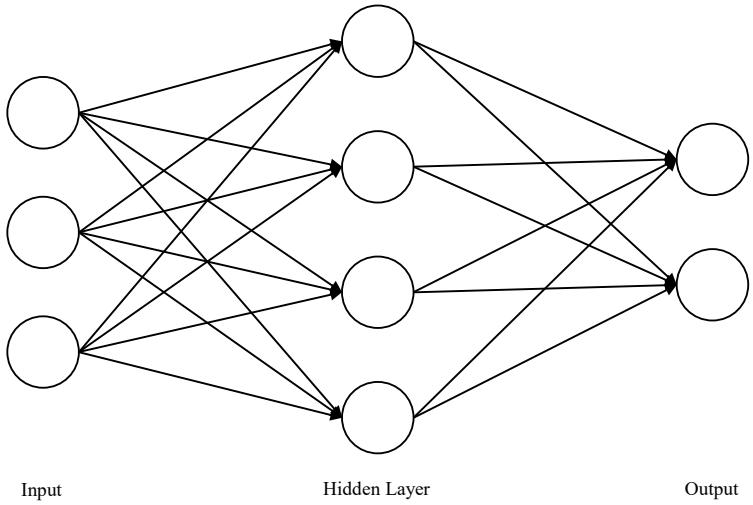


Figure 3. Construction of a simple 3-layer network with one hidden layer (Bishop, 2006)

The term *feedforward* has its origin in a peculiarity of the network. The information always flows in one direction from the input signals through the activation function to the result. Thus, there are no connections from a neuron which lead back to the neuron (Goodfellow et al., 2016).

Convolutional Neural Networks (CNN) originate from the work of Le Cun et al. (1989, 1998) and are mainly used to process data with a grid-like structure. Such a net differs from normal ANN by using a mathematical operation called convolution instead of a standard matrix multiplication in at least one layer (Goodfellow et al., 2016). Figure 4 shows an example of such a neural network.

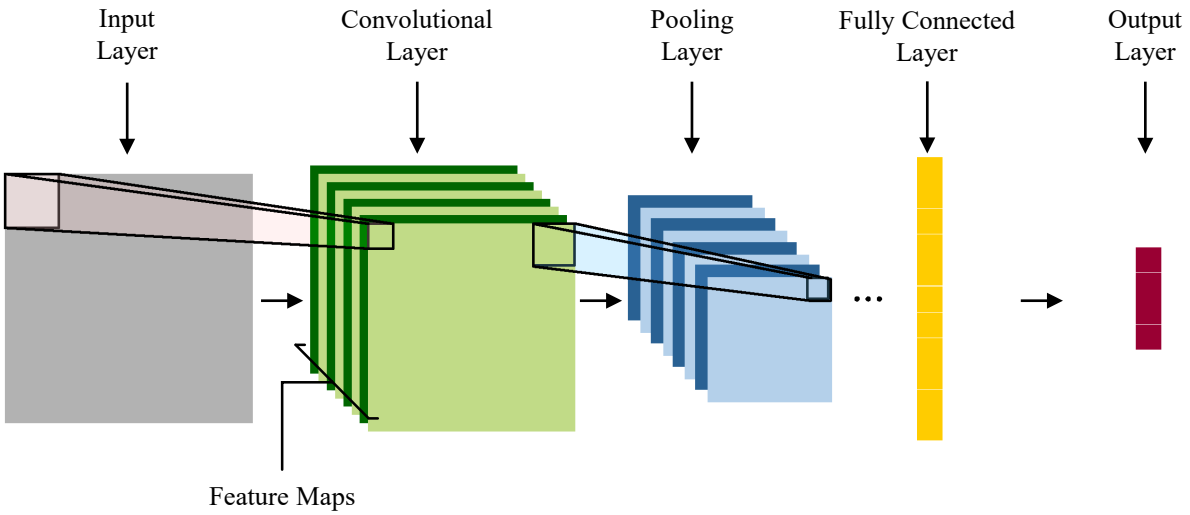


Figure 4. Typical structure of a CNN. An input layer is followed by convolutional and pooling layers, which end in a fully connected and output layer (Peng et al., 2016)

The goal is to create a model for the approximation of functions by statistical generalizations (Goodfellow et al., 2016). This model receives a training data set S with an unknown distribution D, which is available at the

beginning, in order to output a prediction y (Shalev-Shwartz & Ben-David, 2016). The learning algorithm used in the model aims to minimize the error between its prediction and the actual data (Shalev-Shwartz & Ben-David, 2016). Training is done to improve the predictive accuracy of a model by creating an approximation function that is adjusted by minimizing a cost function.

2.2 Related Challenges

Due to the demographic change and the accompanying aging population, some areas, such as health care, are facing new challenges (Rothgang et al., 2016). The progressive robotization is intended to remedy this situation. Certain activities, previously typically performed by humans, are now delegated to service robots.

Developments in service robotics are currently at an intermediate stage. An ever greater acceptance of robots in the population contributes to higher volumes in the production of service robots, as a result of which prices also fall (Haun, 2013). This in turn leads to ever new potential areas of application in which in the past, due to the still high prices, the economic use of service robots was only possible to a limited extent. Nevertheless, only a few service robots are commercially available and affordable for the end user.

The majority of systems that are able to identify fallen persons are basically divided into two groups: Context-sensitive systems and portable devices. Context-sensitive systems use data provided by sensors, which are usually installed in a stationary manner in the desired environment. These include cameras, microphones, infrared and pressure sensors, and sensors in the ground. Portable devices are worn by the user directly or in close proximity to the body and detect a fall by means of the implemented sensors. Accelerometers are most frequently used for this purpose, which may also be supported by gyroscope sensors to determine the exact position of the fallen person (Igual et al., 2013).

Context-sensitive systems have the disadvantage that the sensors are stationary in the monitored environment. This means that unrestricted and flexible monitoring of the desired areas is not possible (Rougier et al., 2011). Portable devices, on the other hand, offer flexible and mobile monitoring because the sensors are mounted on or in the immediate vicinity of the body of the person being observed. However, the disadvantage is the obligation to carry the devices uninterruptedly in order to ensure continuous security through the system (Solbach & Tsotsos, 2017).

The Sanbot Elf combines the advantages of both approaches and avoids the respective disadvantages. Since the sensors are installed in the Sanbot, it is no longer necessary for them to be on or near the user's body. Furthermore, the Sanbot is able to move in the desired operating area, which is why the disadvantage of stationary sensors is not given. For these reasons, the Sanbot Elf is ideally suited to implement a system for detecting fallen persons using its cameras.

The use of ML methods results in a new approach to fall detection and better results than other classification tools (Yu et al., 2017). Especially the already mentioned CNNs are suitable for the processing and classification of image material. Using this type of ANNs on the Sanbot Elf promises a fall detection that does not have the disadvantages of context-sensitive and portable systems mentioned above.

2.3 Solution Concept

2.3.1 Robot Platform Sanbot Elf

The Sanbot Elf consists of 51 sensors that enable it to interact with its environment. A selection of these sensors is shown in Figure 5. The head unites most of the sensors that make these interactions possible. An HD color camera records videos and images and thus enables the perception of the surroundings. In addition to the color camera, a 3D camera is also embedded in the head of the Sanbot, which is able to capture depth information of the environment. The face is visualized by a low-resolution LED display on which the eye areas are simplified. LEDs are also mounted on the head, arms and underside, which glow in different colors. A total of seven microphones are mounted around the robot for participation in conversations, enabling a 360° localization of voices or noises. Another feature of the head is the rear mounted projector and an additional lamp on the front of the head that provides the Sanbot Elf with the ability to operate in dark environments. Furthermore, a total of seven tactile sensors are implemented on the front and top of the head. These sensors enable interaction with the service robot not only through voice commands, but also through specific touches. The entire head can be moved

horizontally by 180° and vertically by 30° and is thus able to turn to localized voices (QIHAN Technology, 2018a, b).

The chest of the 19 kg Sanbot Elf is covered by a 10.1 inch Full-HD touch display, which serves as the main input device. There is another color camera below the screen, but with a lower resolution than the primary camera. At the bottom of the torso there are several touch sensors that are supplemented by infrared sensors. Thus, the Sanbot Elf is able to measure distances between itself and other objects and to use them for its movements. On the front are three loudspeakers, which enable the Sanbot to react to external influences and to communicate with people. Furthermore, inside the Sanbot Elf there is a gyroscope sensor as well as an electronic compass, which allow the determination of the Sanbot's orientation (QIHAN Technology, 2018a, b).

Additionally, there is a PIR sensor on the central part of the torso. Such a passive infrared sensor measures temperature fluctuations and is therefore capable of detecting movements (Song et al., 2008). For this reason, this sensor is used to detect and track people in front of the robot.

The service robot has an omnidirectional drive that enables it to move from any position in any direction. Moreover, this drive enables a 360° rotation on the spot and a maximum speed of 0.8 m/s. For the safe recognition and avoidance of obstacles, being persons or objects, infrared sensors installed at the bottom of the torso serve as bumpers. This ensures reliable movements in its surroundings, which do not result in any damage (QIHAN Technology, 2018a, b).

The operating system of the Sanbot Elf is Android 6 which is accessible through the integrated touch screen mentioned before. Due to this fact, the detection system for fallen persons is deployed as an Android app on the Sanbot Elf.

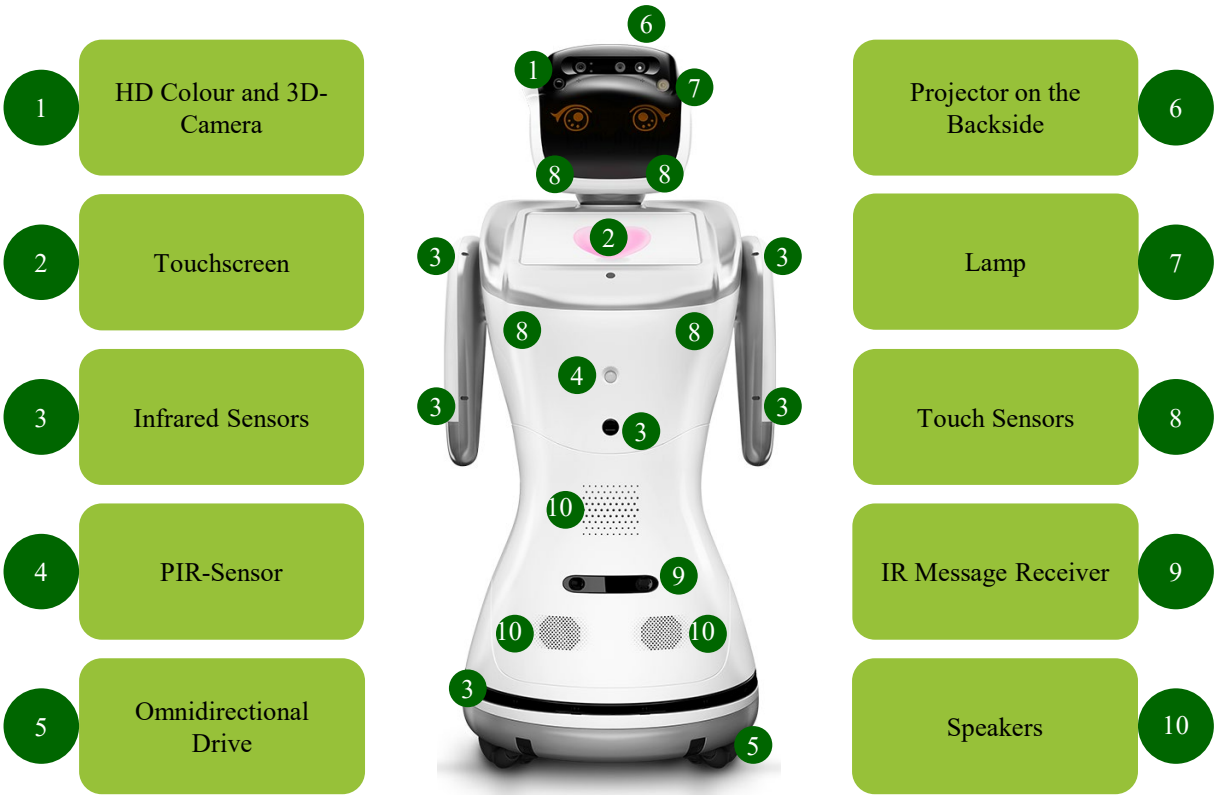


Figure 5. Hardware built into the Sanbot Elf. Picture of the Sanbot Elf: QIHAN Technology (2018a)

2.3.2 Process of the Fall Detection System

Figure 6 shows the basic concept of a system for the detection of fallen persons, which is developed in the course of this paper. Using the color camera integrated in the Sanbot Elf, the camera stream of the area to be monitored is recorded. The individual image data of this video stream is continuously processed and forwarded to a previously trained CNN. This CNN analyses the individual images and divides them into two classes:

- Class 0: *fallen*
- Class 1: *not fallen*

This classification takes place without interruption as long as the video stream of the camera is active. A fallen person, who is no longer able to stand up on his own, will remain on the ground for a long time. For this reason, a person will be considered to be in need of help if they remain on the ground for a predefined period of time. If the CNN classifies a person, who has fallen during this period on the basis of the image data, follow-up measures are initiated. The time span here is set to 60 seconds, as it should normally be possible for a fallen person to at least sit up during this time to no longer be classified as having fallen.

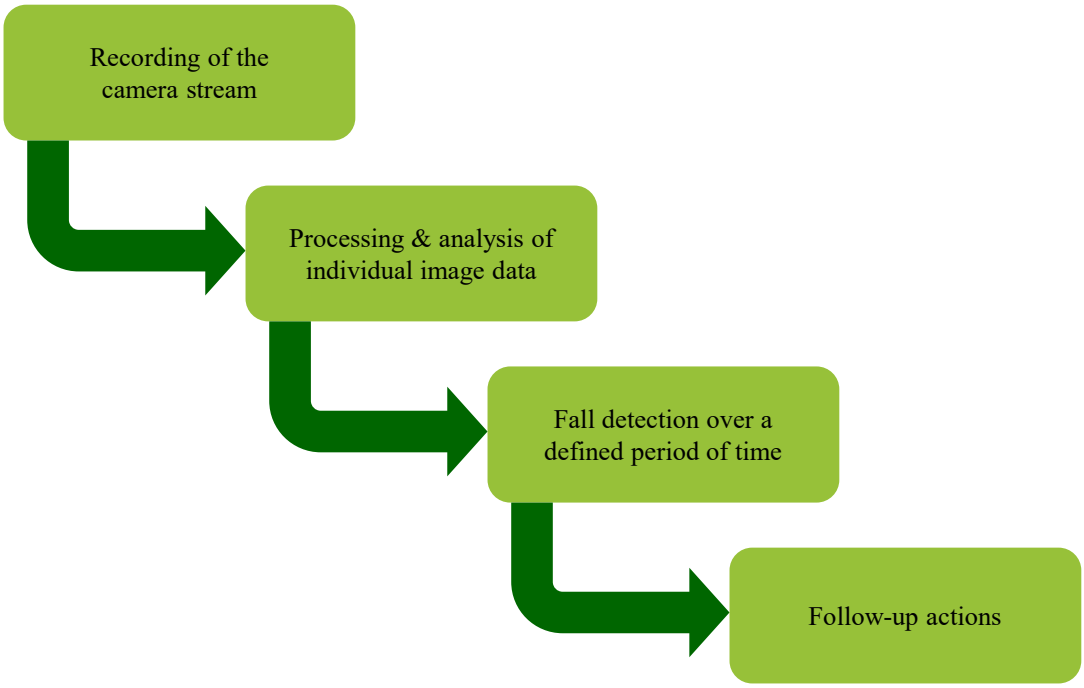


Figure 6. Approach for a solution system to detect fallen persons based on the analysis of image data obtained from a camera

2.4 Implementation

In addition to the IDE Android Studio 3.0.1 and the *QIHAN SDK 1.1.8* for app development, software for the development of neural networks is used. Therefore, PyCharm with the programming language Python is selected as the development environment. The creation and training of the neural networks is realized by means of the open source library Keras, which contains different ML libraries including the TensorFlow (TF) library. The training of the ANN then takes place on a mobile GPU. The implementation process is structured as follows:

1. Acquisition and preparation of training data
2. CNN model constellation
3. Training and optimization of the CNN
4. Porting the CNN to Android
5. Integrating the CNN into the app

As the name implies, in TF the ML algorithms are constructed as data flow graphs, where the nodes represent the operations performed and their connections (or edges) represent the data flow. Tensors (n-dimensional fields) are used between the nodes as a general exchange format containing primitive data types such as *int32*, *float32* and *string*. TF allows the user to realize applications on different platforms like distributed clusters, local workstations or mobile devices. The scripting language Python acts as a wrapper for the creation of graphs and allows the user to customize and extend the graphs and models without having to change the basic system (Abadi et al., 2016).

2.4.1 Acquisition and Preparation of Training Data

In principle, the information that is required at the beginning is the information that is predicted by the ANN. In the present case of the recognition of fallen persons, this is image data that depicts the state of a fallen person. As already described in section 2.3.2, CNN image data is classified into two classes: On the one hand, persons who

have belong to class 0 (fall) and on the other hand persons who have not fallen (class 1). For this reason, image data for binary classification is necessary for both the positive and the negative class. No freely accessible data set with pictures of fallen persons that is suitable for this work is publically available, which is why a separate data set is created in the course of this work. This data set consists of images that are obtained via both a Google image search and by own images of people lying on the ground.

2.4.2 Network Structure

There are two different procedures for choosing a suitable CNN. One option is to either create your own model or use an existing model of a CNN and train it completely on the basis of the existing data set. This gives the user the greatest possible freedom in network structure. Complex models, however, sometimes reach millions of parameters, for whose training correspondingly large amounts of image data are required (Google LLC, 2018). This in turn also requires adequate computing power to carry out the training in an acceptable amount of time (Google LLC, 2018).

If one or both of these requirements are not met, another option is to use pretrained CNNs. These complex CNNs are already trained on large data sets with hundreds or thousands of different classes and can be retrained to new classes without retraining the weights of the entire model.

The CNN used in this paper to classify images is shown in Figure 7 and is based on the CNN image classification architectures proposed by Yann LeCun (LeCun & Bengio, 1995).

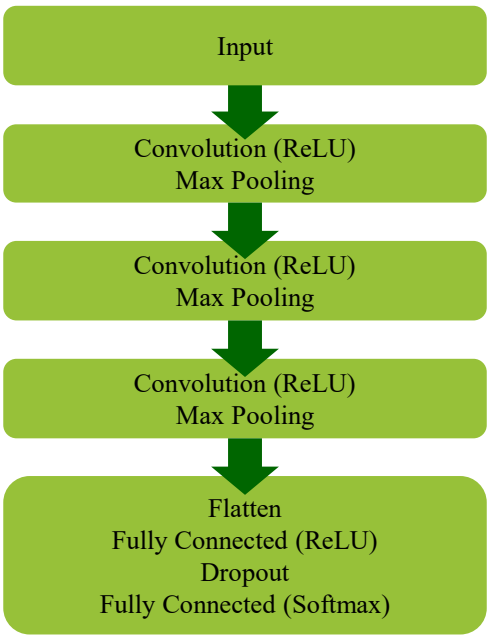


Figure 7. Construction of a CNN consisting of repeating blocks with convolutional and max pooling layers. An output block consisting of a flatten layer and a fully connected layer follows while dropout is used as regulation

A data set widely used in visual object recognition is provided by the ImageNet project. It contains 21,841 different classes consisting of a total of 14,197,122 images with corresponding labels (as of 30.04.2010) (Stanford Vision Lab, 2016). A CNN trained on the basis of this data set has thus learned helpful features for a wide range of classification problems. Therefore, it can be used for the classification of unknown objects as well. This circumstance is exploited by using such a CNN and retraining it to own classes by using a much smaller own data set. For this purpose, the last block for outputting the results is separated from the rest of the model and replaced by the same output block in Figure 7. This block contains the necessary configurations for the classification of the fallen persons.

With a total number of 1,064 images for class 0 only a limited data set is available. This is due to the effort involved in generating such image data. Consequently, the classification model in this paper is based on the use of an already trained CNN. By retraining this CNN, useful results can be achieved with the present limited data set.

The best-known CNNs, some of which are listed on the official TF website, that have achieved good results in past classification competitions are the *Inception, Inception-ResNet, NASNet, VGG16* and *ResNet* models. There are also *MobileNets* that have been developed especially for the use on devices that do not have large computing capacities (Howard et al., 2017). Furthermore, the use of the given energy on mobile device is a crucial factor because the higher the number of nodes in a model the higher the required energy for the classification. The focus here is on weighing accuracy against resource-saving performance. In this work, the *Inception v3* model (Figure 8) is retrained and used for the classification problem. This model is based on the work of Szegedy et al. (2016) and consists of blocks of layers arranged one after the other.

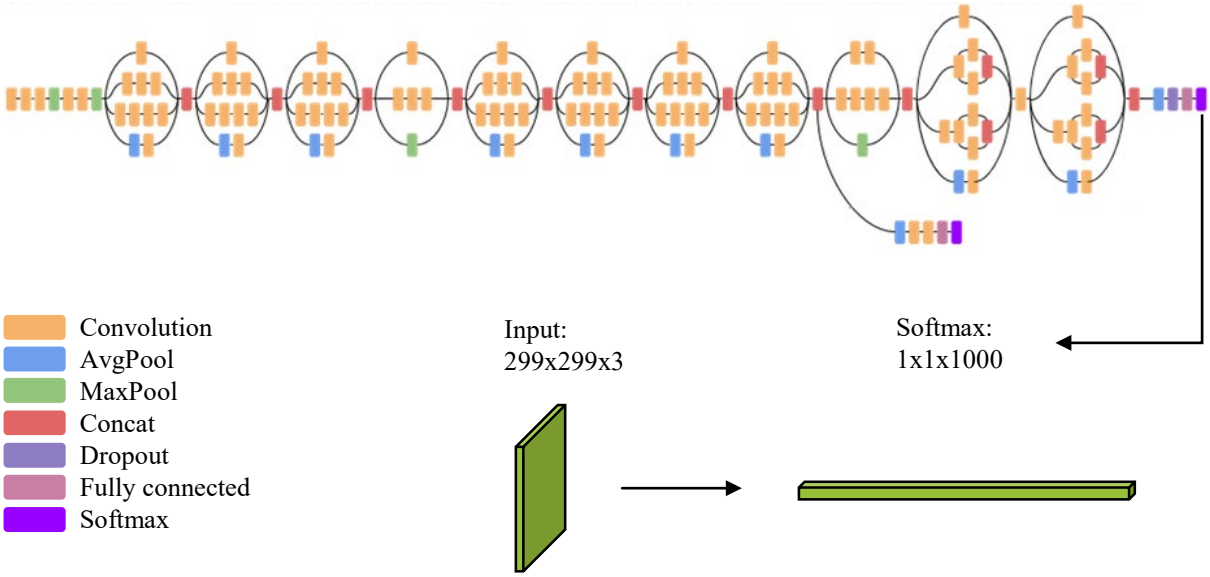


Figure 8. Structure of the Inception v3 model developed by Google. A layer of size 299x299x3 serves as input, followed by blocks of different layers to perform certain mathematical operations. The Softmax output layer of size 1x1x1000 is used to classify 1,000 different classes (Google Developers, 2018).

2.4.3 Training

At the beginning of each classification problem, the existing data set must be split into three separate subsets. With 688 images per class (66 percent), the training data set contains the majority of this data set. The remaining set is divided between the validation set with 320 images per class. The test data set contains 56 images for class 0 and 41 images for class 1. This is followed by the selection of a basic model whose weights have already been trained, for example using the *ImageNet* data set. Furthermore, the hyperparameters epochs and batch size as well as the input size of the images are specified. The existing image data is then preprocessed and multiplied by a data generator. The images are, among other things, rotated, scaled, distorted, enlarged or reduced. These random transformations of the images increase their total number and ultimately lead to better results.

The process of creating an overall model is shown in Figure 9. In order to be able to introduce own classes into the model, the top *Fully Connected Layers*, which are responsible for the actual classification, have to be separated from the rest of the model. For the classification, a superordinate model adapted to the problem is now created, trained and reunited with the base model. The previously generated image data is once routed through the basic model and processed. The resulting output is saved and then serves as an input layer for the higher-level model. Using these features, also known as bottleneck features, the actual training of the higher-level model is carried out. This leads to a more efficient training, since a run through the base model is bypassed in each iteration. Finally, the trained superordinate model is reunited with the base model. As a final step, the prediction accuracy is checked against the overall model using the test data set.

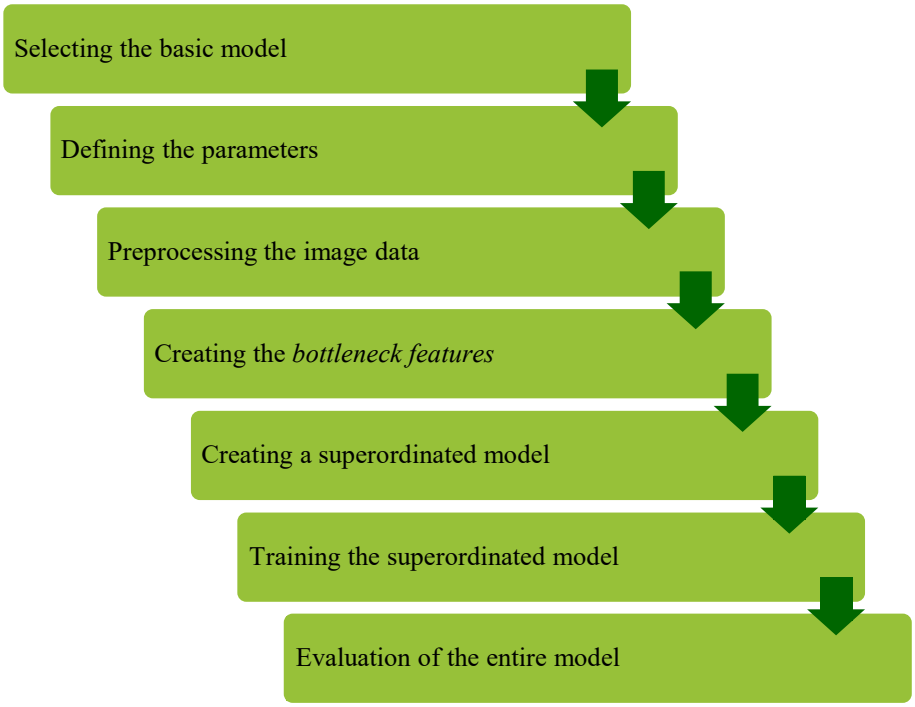


Figure 9. Process of creating an overall model for classifying your own image data using a previously trained basic model

2.4.4 Porting and Integrating the CNN into the App

Keras allows a simpler generation of neural network models. However, the resulting models are not suitable for immediate use with TF. In order to use this format again, it is necessary to convert the models from Keras' own format back to the TF format.

The application of a TF model to classify images on an Android or iOS operating system is possible with two different solutions: TF Mobile and TF Lite. In the course of this work TF Mobile is chosen due to the fact that the training is done on a Windows operating system and this solution offers more stability.

For the use of TF on Android devices there is a Google demo application (Google LLC, 2016) available, which includes four functions. One of these functions is TF Classify that makes simple classification of objects possible. The previously created files for the own CNNs and the corresponding labels are added to the project and now allow a classification based on the own defined classes. This function is then added to the app being deployed on the Sanbot Elf to detect fallen persons.

2.5 Results and Validation

2.5.1 Validation of the ANN

Due to existing difficulties in the app, the display of the camera stream and the classification are evaluated separately. For the classification a locally stored image of class 0 and 1 is used and analyzed by the algorithm and the CNN. This procedure shows the feasibility of the present fall detection and its use in the application, since both images are correctly classified.

This paper focuses on the application of an already trained model and its retraining to the desired classes. The CNN models *Inception v3* and *MobileNet v1* are being used. They are among the best-known and most frequently used models in past competitions. As a comparison, an additional model is used whose total weights are trained solely by the existing data set and is subsequently referred to as CNN1. The adjustment of these parameters to the respective conditions and their analysis is also referred to as *hyperparameter tuning*. A comprehensive tuning of the hyperparameters, however, is outside the focus of this paper, which is why the most common settings are used, which have proven to be promising in the past (Table 2).

Hyperparameter	Value
Cost function	Categorical cross entropy
Epochs	50
Regulation	Dropout
Optimization	SGD (momentum = 0.9)
Learning rate	0.0001
Activation function	ReLU, Softmax
Batch size	16
Size of the Input Layer	224x224x3

Table 2. Selection of the most important parameters for the CNN training

For the final evaluation of the models, the test data set is used. The results are shown in the confusion matrix in Table 3. The *Matthews Correlation Coefficient* (MCC) can be used to evaluate an ANN using the entire confusion matrix. A value of $MCC = 1$ describes a perfect prediction while $MCC = -1$ indicates a total contradiction between prediction and reality. A value of zero means that the result of the model is no better than random predictions (Matthews, 1975).

Name	Accuracy	Costs	MCC
CNN1	0.92	0.24	0.80
Inception v3	0.98	0.06	0.96
MobileNet v1	0.99	0.06	0.94

Table 3. Results of the evaluation of the CNN1, *Inception v3* and *MobileNet v1*

The CNN1 is inferior to the *Inception v3* and *MobileNet v1* in all metrics, i.e. cost, accuracy and MCC. In comparison with the other models, the cost value is relatively high at 0.24 and the result of the test data set by the MCC comparatively low at 0.80. It should be noted that the model has only approximately 2.8 million weights, which are completely trained on the basis of a small data set. The *Inception v3* has 34.8 million weights whereas the *MobileNet v1* consists of 16 million weights.

The *Inception v3* and *MobileNet v1* both have a cost value of 0.06. This value can be attributed to the fact that in both models the base model has already been trained using 1,000 different classes of the ImageNet data set. As a result, the models have already learned helpful features for distinguishing objects that have a positive effect on the classification problem at hand. The final test for the *Inception v3* with a value of 0.96 for the MCC is slightly better than for the *MobileNet v1* with a value of 0.94. This can be explained, among other things, by the almost twice as high number of weights in the network. The performance of the newer *MobileNet v1* shows that the development of more powerful and more efficient ANN is increasing.

Based on the results obtained, the retrained *Inception v3* is used for implementation in the app, as it shows the best performance of all models. In the light of the limited battery capacity of the Sanbot Elf, it is also possible to use the *MobileNet v1*, which, with a minimally lower performance, enables a resource-saving deployment due to its reduced size. In a real application, however, it is advisable to use the *Inception v3*.

2.5.2 Validation on External Influences

In addition to persons who have fallen, the data set at hand represents a high variability of images deviating from this. In a domestic environment, however, it is likely that people will also adopt postures that may resemble those of a fallen person. For example, the difference between a sleeping person and a fallen person is not recognizable by the ANN. In addition, there are situations in which people sit on the floor or squat. In isolated cases items of clothing spread out on the ground, such as a bathrobe or a motorcycle suit, are classified as a fallen person by the CNN. This circumstance may be remedied by a data set that is increasingly oriented towards

human poses. This also gives human body parts such as head, hands and feet a higher weight in the classification.

2.6 Discussion

The applications can be divided fundamentally into four areas, which are the subject of this chapter: retail, public, education as well as the health care sector, ambient assisted living (AAL) and smart home environments.

In shops, the Sanbot Elf plays the role of an information medium. Through its face recognition it is possible to locate customers entering a store and greet them directly. The customer is made aware of current products or offers, which are shown directly on the display through the integrated product preview.

The public area includes, apart from the retail mentioned above, freely accessible buildings such as public administration buildings, railway stations or airports. At airports and railway stations an application is conceivable to provide passengers with information about their travel times, ticket prices or current offers from the shops. Other areas such as police stations, courts or libraries are conceivable as general sources of information and aids (QIHAN Technology, 2018a).

The increasing digitalization does not stop at the education system and changes the classical teaching and its learning methods (Dräger & Müller-Eiselt, 2015). Due to its features, the Sanbot Elf is able to carry out a wide variety of activities. The built-in projector makes additional external hardware for presentations redundant, while the touch display of the Sanbot Elf makes it possible to interactively bring lesson content closer to the students.

The application of the Sanbot Elf in the health sector is one of the largest areas of application. Since health care extends to the private environment in addition to large facilities, the overlapping AAL and smart home environments are also included in this context. The Sanbot Elf offers the possibility of continuous monitoring and care of patients. In general, the service robot serves as a supporting assistance system for the workers as well as the people to be cared for. In senior citizens' and nursing homes, the Sanbot Elf serves not only as an assistant, such as supplying the residents with drinks, but is also used for entertainment by playing music or watching films via the built-in projector. In addition to facilities such as hospitals or retirement homes, the Sanbot Elf is also deployed in private homes. Thus, continuous care of people is also possible, without one person necessarily being on site all the time. This is especially helpful when affected persons are no longer able to provide for themselves independently. In addition, there is a reminder function that draws attention to take certain medications, the performance of required exercises or the adherence to doctor's appointments (QIHAN Technology, 2018a). The interaction possibilities of the Sanbot also make remote monitoring by doctors possible, who can then identify initial problems more quickly with the help of video chats.

2.7 References

Abadi, M., Barham, P., Chen, J., et al. (Eds.), 2016. TensorFlow: A System for Large-Scale Machine Learning (th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)).

Becker, H., Scheermesser, M., Früh, M., Treusch, Y., Auerbach, H., Hüppi, R. A., & Meier, F., 2013. Robotik in Betreuung und Gesundheitsversorgung. Vdf, Hochschulverlag AG an der ETH Zürich (TA-SWISS, 58), Zurich.

Carone, G., & Costello, D., 2006. Can Europe Afford to Grow Old? In: *Finance and Development* 43.

Cyber Robotics Technology Limited (Ed.), 2018. *Sanbot Elf.* https://www.robotics.com.hk/product/sanbot-elf/ (accessed May 26, 2018)

Decker, M., Dillmann, R., Dreier, T., Fischer, M., Gutmann, M., Ott, I., & Spiecker genannt Döhmann, I., 2011. Service robotics: do you know your new companion? Framing an interdisciplinary technology assessment. In: *Poiesis & Praxis* 8 (1), pp. 25–44. DOI: 10.1007/s10202-011-0098-6

Dräger, J., & Müller-Eiselt, R., 2015. Die digitale Bildungsrevolution: Der radikale Wandel des Lernens und wie wir ihn gestalten können. Deutsche Verlags-Anstalt (DVA), München.

Generation Robots (Ed.), 2018. *Pepper for Business Edition.* https://www.generationrobots.com/en/402422-pepper-for-business-edition-humanoid-robot-2-years-warranty.html (accessed May 26, 2018)

Google Developers, 2018. *Cloud TPU. Advanced Guide to Inception v3 on Cloud TPU.* https://cloud.google.com/tpu/docs/inception-v3-advanced (accessed July 11, 2018)

Google LLC, 2016. *TensorFlow Android Camera Demo. Edited by TensorFlow.* https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android (accessed July 18, 2018)

Google LLC, 2018. *TensorFlow. Image Retraining.* https://www.tensorflow.org/hub/tutorials/image_retraining (accessed July 11, 2018)

Guo, S., & Zhang, G., 2009. Robot rights. In: *Science (New York, N.Y.)* 323 (5916), p. 876. DOI: 10.1126/science.323.5916.876a

Halter, J. B., Ouslander, J. G., Studenski, S., High, K. P., Asthana, S., Supiano, M. A., & Ritchie, C. S., 2009. Hazzard's geriatric medicine and gerontology. 6th ed. McGraw-Hill Education Medical, New York.

Haun, M., 2013. Handbuch Robotik. Programmieren und Einsatz intelligenter Roboter. 2., Aufl. 2013. Springer Berlin (VDI-Buch), Berlin.

Helbing, D., 2013. Globally networked risks and how to respond. In: *Nature* 497 (7447), p. 51.

Helbing, D., & Pournaras, E., 2015. Society: Build digital democracy. In: *Nature News* 527 (7576), p. 33.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al., 2017. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In: *arXiv preprint arXiv:1704.04861*

International Federation of Robotics, 2016. *World Robotics. Service Robots. Chapter 1.2.* https://ifr.org/img/office/Service_Robots_2016_Chapter_1_2.pdf (accessed May 22, 2018)

LeCun, Y., & Bengio, Y., 1995. Convolutional Networks for Images, Speech, and Time-series. In: *The handbook of brain theory and neural networks* 3361 (10), p. 1995.

Matthews, B. W., 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. In: *Biochimica* et *Biophysica Acta (BBA) - Protein Structure* 405 (2), pp. 442–451. DOI: 10.1016/0005-2795(75)90109-9

Melson, G., 2010. Child development robots: Social forces, children's perspectives 11, pp. 227–232.

Mori, M., MacDorman, K. F., & Kageki, N., 2012. The Uncanny Valley: The Original Essay by Masahiro Mori. In: IEEE *Robotics & Automation Magazine*, pp. 98–100.

QIHAN Technology, 2018a. http://en.sanbot.com/ (accessed June 6, 2018)

QIHAN Technology, 2018b. *Robot S1-B2 User Manual. V1.3.* http://en.sanbot.com/support/ (accessed June 14, 2018)

SoftBank Robotics (Ed.), 2018. *Who is Pepper?* https://www.softbankrobotics.com/emea/en/robots/pepper (accessed May 26, 2018)

Song, B., Choi, H., & Lee, H. S., 2008. Surveillance Tracking System Using Passive Infrared Motion Sensors in Wireless Sensor Network. In: *International Conference on Information Networking, 2008*. ICOIN 2008; 23–25 Jan. 2008, Busan, Korea (South). 2008 International Conference on Information Networking. Busan, South Korea, 1/23/2008–1/25/2008. International Conference on Information Networking; ICOIN. Piscataway, NJ: IEEE Service Center, pp. 1–5.

Stanford Vision Lab, 2016. *ImageNet database.* http://image-net.org/ (accessed July 11, 2018)

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (Eds.), 2016. Rethinking the Inception Architecture for Computer Vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27–30 June 2016.

3 SCALING HOME AUTOMATION TO PUBLIC BUILDINGS: A DISTRIBUTED MULTIUSER SETUP FOR OPENHAB 2

F. Heimgaertner [a, *], S. Hettich [a], O. Kohlbacher [a], M. Menth [a]

[a] University of Tuebingen, Department of Computer Science, Tuebingen, Germany,
(florian.heimgaertner, menth, oliver.kohlbacher)@uni-tuebingen.de, stefanhettich@gmail.com

KEY WORDS: Authorization, Building Management Systems, Decentralised Control, Distributed Control, Home Automation, Distributed OpenHAB 2 Setup

ABSTRACT:
Home automation systems can help to reduce energy costs and increase comfort of living by adjusting room temperatures according to schedules, rules, and sensor input. OpenHAB 2 is an open-source home automation framework supporting various home automation technologies and devices. While OpenHAB is well suited for single occupancy homes, large public buildings pose additional challenges. The limited range of wireless home automation technologies requires transceivers distributed across the building. Additionally, control permissions need to be restricted to authorized persons. This work presents OpenHAB-DM, a distributed OpenHAB 2 setup with extensions introducing user authentication, access control, and management tools for decentralized OpenHAB node deployment.

3.1 Referenced Paper

F. Heimgaertner, S. Hettich, O. Kohlbacher, & M. Menth, 2017. Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2. *2017 Global Internet of Things Summit (GIoTS)*, Geneva, 2017, pp. 1-6. DOI: 10.1109/GIOTS.2017.8016235

IEEE (accessed November 21, 2018):
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8016235

Preprint version (accessed November 21, 2018):
https://atlas.informatik.uni-tuebingen.de/~menth/papers/Menth17i.pdf

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, p. 29,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

4 SIMPLIFY OPENHAB UTILIZATION IN PUBLIC INSTITUTIONSUSING CLOUD TECHNOLOGIES

A. Dobler [a,*], D. Uckelmann [a], G. Lückemeyer [a]

[a] Department of Geomatics, Computer Science and Mathematics, University of Applied Sciences Stuttgart
Schellingstraße 24, D-70174 Stuttgart (Germany)
(alexander.dobler, dieter.uckelmann, gero.lueckemeyer)@hft-stuttgart.de

KEYWORDS: openHAB, Public Buildings, Docker, Swarm, Cloud

ABSTRACT:
Smart Home solutions like openHAB help to automate the control of connected devices and grant an overview of monitored appliances, as well as reduce energy consumption. Public institutions like universities or schools could also benefit from such solutions, especially considering they are often located in old buildings. But while openHAB is a feasible solution for home users, public institutions usually have other requirements towards their software, e.g. regarding security, maintainability and fault tolerance. Beside this, public institutions usually span a bigger area, often with multiple buildings which have to be connected with each other. This work analyses how the infrastructure of a common openHAB stack looks like, by examining which software components are usually used and how they are interrelated. It tries to point out possible shortcomings concerning the needs of a public building. Furthermore it showcases how these shortcomings can be tackled by using cloud technologies locally. To achieve this, an example setup was created using Docker and it's orchestrator Docker Swarm. It is explained how the single applications of an openHAB stack are integrated into the swarm, how to manage multiple possible instances and what to consider to simplify it's usage. The outcome of this work shows that such a setup helps to improve the security, by running in a separate network with strictly explicit entry points. It also demonstrates that maintenance can be kept low by allowing for an easier and central configuration as well as simpler backups of user data. This in return helps to react quicker in case of failures or changes to the system and therefore to achieve better availability.

4.1 Introduction

Big buildings or complexes need a way of monitoring, managing and automating their infrastructure to ease the work of facility managers and help them to keep an overview, as well as to control building appliances. An intelligent control of climate, heating and light systems in a building can also help to reduce energy consumption and save money (DIN EN 15232-1:2017-12, 2017). For these use cases many commercial options are available (Cisco Systems, 2018; Ultimo Software Solutions, 2018; SoftGuide GmbH, 2018), but they tend to be expensive, closed and tied to proprietary hardware.

In recent years the interest and demand is on the rise to control own private homes too, pushed by smart speakers like Amazon's Alexa and Google Home (Martin, 2018). But beside these commercial offerings, the smart home sector also offers a small range of open and extensible platforms like openHAB, FHEM or Home Assistant. They act as a hub and allow to interconnect many different platforms with each other. Out of these openHAB offers a matured and solid solution with a core that is used by commercial consumer grade solutions like QIVICON by Deutsche Telekom (Deutsche Telekom AG, 2016).

The usage of such an open system in public buildings, especially in educational facilities would be an interesting choice. Beside the already mentioned advantages for monitoring and energy savings, it would also be a platform which is extendible and open for improvements. These extensions may even be developed in projects by students, providing them with interesting educational tasks, as well as improving the infrastructure of their facility. OpenHAB would also be an economical choice as it's bare usage is free (except for maintenance and operation) and not tied to (expensive) proprietary hardware. To make this possible, openHAB as a software needs first be checked for the demands of public facilities or larger environments in general and adapted if necessary.

*Corresponding author

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 31–41,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

This work is derived as a part from an ongoing master thesis with the goal to adapt openHAB to be used in public institutions. This paper focuses on making it easier to deploy and operate openHAB as a solution for multiple buildings. To achieve this openHAB is considered as part of a bigger stack combined with other components. The resulting solution uses *Docker* and *Docker Swarm* to enable a simpler setup and operation.

4.2 Analysis and Concept

4.2.1 Requirements in Public Buildings

The requirements for smart home software and software which is meant to be used in bigger buildings usually differ a lot. First there is the general purpose of smart home and smart public building software. A big focus for smart home users is entertainment and comfort, for example being able to control lights without getting up, while also controlling the music playing in the living room (techuk, 2017; GfK, 2016). The market for smart buildings on the other hand is mainly driven by possible energy cost savings, they also should allow for visualization and monitoring (trend:research, 2015).

The scope in which public and home systems operate differs too. This starts with possibly different devices they need to control (e.g. entertainment systems or not), but mainly concerns with the size of the area to manage. While a smart home solution is in control of a single flat or house, a public institution can easily span a much bigger area. The HFT Stuttgart for example has eight buildings spread across an area of roughly 35,000 square meters. For a smart public building this means, it not only has to cope with much more devices (like sensors or lights), but it also needs to ensure to cover a bigger area with possibly independent sections (like buildings). This may require to represent these sections in another UI layer, not needed in a single home. Also important to consider is that public institutions themselves can vary a lot. Not only by size, but also considering equipment and infrastructure. While a university may sport a well grown server landscape, this may be different for small schools. A possible solution would ideally be able to operate within both.

Public and home systems also differ concerning the installation and operation of theses systems. The biggest challenge towards the adoption of a smart building system is actually the same as for smart home products, the cost (trend:research, 2015; techuk, 2017). In addition, public buildings like schools often do not even receive money for urgently needed refurbishments (van Laak and Schröder, 2017). This means in return, the initial setup cost and necessary resources for a smart building solution have to be kept low to enable a successful adoption. Beside the initial setup, another cost driver of software in general is the ongoing maintenance and operation, like ensuring availability, managing failures of soft- and hardware, as well as running updates and supporting users (David et al., 2002).

Two key aspects while operating any distributed software system, which are helpful in this regard, are replication and fault tolerance. Replication allows to gain a higher reliability by keeping data available even in case of system errors (Tanenbaum and van Steen, 2008, p. 303ff). As data may be changed in multiple locations, it still needs to be consistent through out the system to ensure correctness. Working with many distributed components means, that also partial failures are possible. These shall have a minimal influence on other parts of the system, which again leads to higher availability and less effort during recovery (Tanenbaum and van Steen, 2008, p. 354ff).

Another important aspect is security. While security can be provided by several mechanisms such as encryption, this also means that it is a broad field and difficult to do pr operly. Therefore it is important to define a set of security policies first, which are used to find fitting mechanisms to reach compliance (Coulouris et al., 2002, p. 298)(Tanenbaum and van Steen, 2008, p. 413ff).

4.2.2 A Common openHAB Stack

Although openHAB by itself provides a lot of functionality and its architecture is very modular, most openHAB setups come with at least one or two additional software components. These components usually provide functionality outside of the core focus of openHAB, like persisting data or managing telemetry data transport. Even the developers of openHAB seem aware of this. In their official openHAB distribution for the Raspberry Pi single board computer, *openHABian*, they directly included an installer menu for some of the most used external components like *Node-RED* and *Mosquitto* (openHAB Community, n.d.b). Further on, a set of commonly used components is described and how they also benefit in bigger setups.

Data Persistence

A common example for an additional component is item state persistence. While openHAB is able to store data in a local database, like it does for item configurations and settings, it does not rely on it to keep track of item state changes over time. To achieve this openHAB uses extensions called persistence services that allow it to store state changes in a range of external databases like MySQL or InfluxDB and can be installed like bindings directly through its web interface PaperUI. It is an important addition as it allows us to work with and analyse historical item data.

Rule Engine (Node-RED)

While openHAB provides its own textual rule engine, many users additionally rely on the external application Node-RED. As mentioned before it is even included as an install option in the official openHAB Raspberry Pi distribution (openHAB Community, n.d.b). Instead of using text files, Node-RED allows to define rules using a visual representation, displaying rules as a flow of connected nodes. It is also able to work with multiple instances of openHAB and therefore can be an advantage in a multi building scenario.

MQTT

Another often used external service or in this case rather protocol is MQTT (even in advanced setups (Pham et al., 2015; Fu et al., 2017; Wlotzka, 2016)). MQTT is a protocol tailored towards machine to machine (M2M) use cases (mqtt.org, n.d.). It allows to easily connect additional sensors and actuators not natively supported by openHAB, as well as to connect with other smart home systems. It can also be used to share events between multiple openHAB instances and thanks to the gained flexibility it can be an helpful addition.

Web server / Reverse proxy

A web server is something not strictly needed for an openHAB setup, but it provides a big addition, namely authentication. By default an openHAB instance is quite exposed to the local network. To reduce risks arising from this, the openHAB documentation recommends the usage of a web server as a reverse proxy(openHAB Community, n.d.a) and enable the basic authentication method of the HTTP standard. The web server is essentially receiving requests from a client and forwards them to the fitting application and handles additional tasks such as encryption, data caching and authentication.

Sensors and actuators

The last external part we consider here is not strictly an external component. It describes how devices controlled by openHAB are integrated. Sensor and actuator devices, like switches and lights, can be connected to openHAB with several different methods. One option is to connect devices directly to openHAB, in these cases openHAB

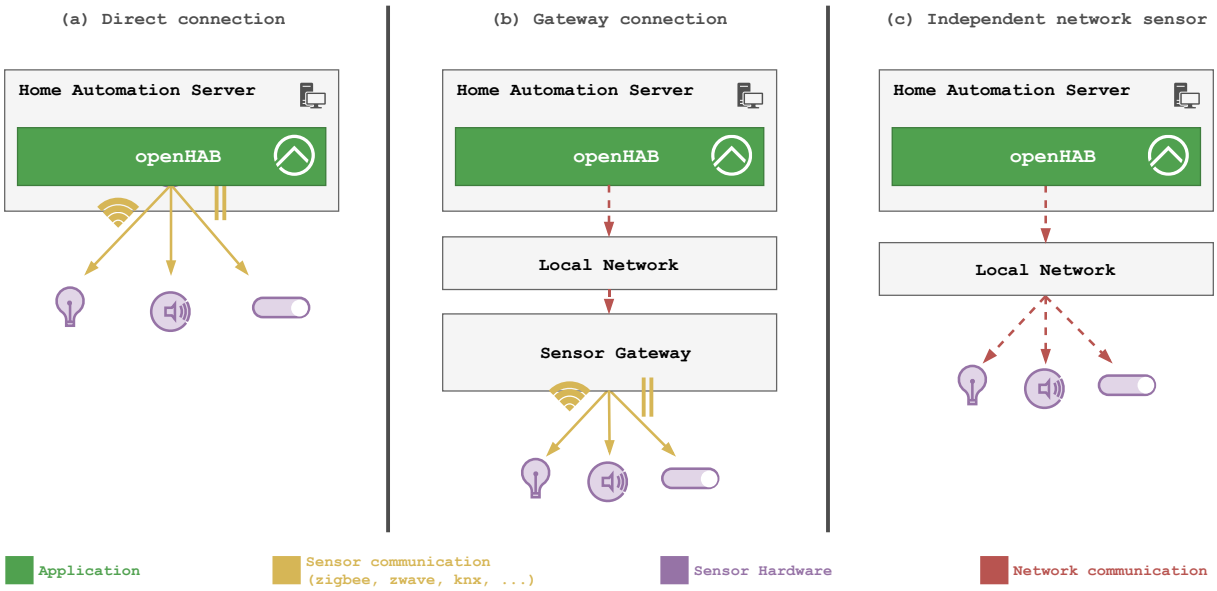


Figure 1. Connection Types of openHAB Sensors

uses available interfaces (either wired or with the help of radio hardware) and manages connected devices on its own (see Figure 1 (a)). An example here is the *Z-Wave* binding of openHAB, it uses a connected Z-Wave serial adapter to directly communicate with Z-Wave devices.

In most cases however, the devices are not directly attached to openHAB, but rely on a third-party hub or gateway

to communicate with it (see Figure 1 (b)). These vary in appearance, being either other servers or more often small embedded devices. These gateways are connected to the real physical actuators and sensors, either by wires or by using a wireless connection based on a radio standard. For both wireless and wired connections a range of different protocols are available which the gateways may use to interact with the sensors and actuators, e.g. Z-Wave, ZigBee or KNX. OpenHAB then connects to these gateways, usually accessing them trough an IP connection, e.g. by using an exposed REST API, to get access to all connected devices of the gateway.

The last option are single devices which directly reside in the same local network and can be accessed through an IP connection by openHAB (see Figure 1 (c)). Towards openHAB they behave similar to sensors using a gateway but operate independently.

4.2.3 Current State

The previous section outlined, a complete openHAB setup usually consists of way more parts then openHAB itself and that public buildings can also take advantage of them. It is therefore necessary to consider openHAB together with these (and possibly further) components as a complete stack when looking for shortcomings and possible solutions, as well as to keep in mind that we additionally have to deal with multiple possible instances within separate buildings.

4.2.3.1 Security

To begin with, we will look at the state of security of our components. The previously described basic stack, when run at home, could look similar to what is pictured in figure 2. It shows the single components running as services on a single machine. It also shows how a network setup could look like using the defaults that adhere when setting up the single components. In particular, the many open accessible ports may attract attention. When applications are exposed to the network like this, they may be misused. Especially considering networks of public institutions with a mixture of staff, students and guests having access to the network.

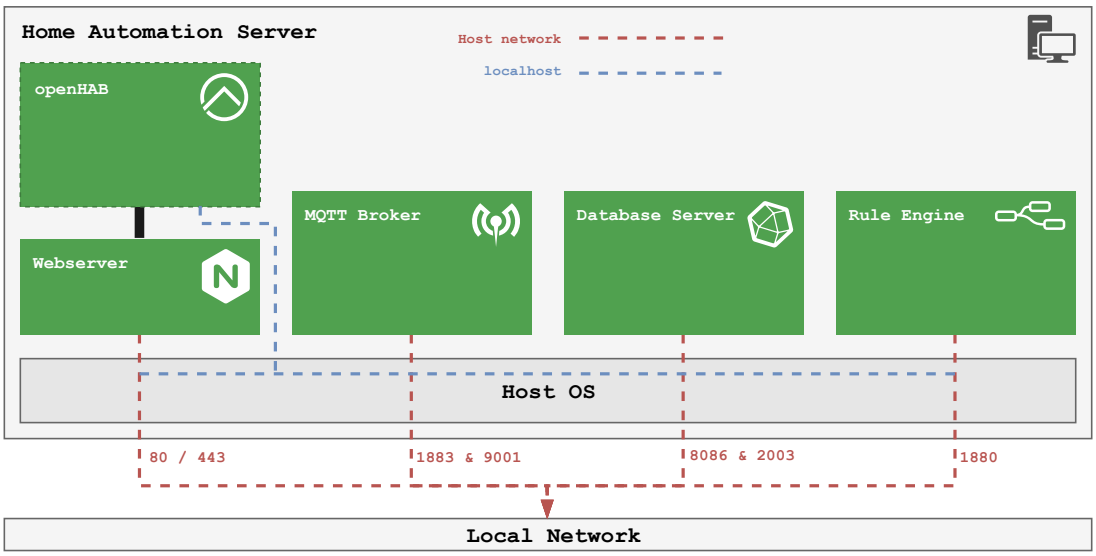


Figure 2. A typical full openHAB setup

The main problem illustrated here is not the security of the applications themselves. Some of the pictured examples (Mosquitto, InfluxDB, Node-RED) are capable of encrypting their communication and have some means for access control (openHAB only with the help of a web server). The issue here is, that all applications are easily accessible through the network, although not always necessary. This is caused by the defaults used during their setup. Parts of these may be feasible in a private network, but they do not fit the needs of public environments. A reason for this is, that defaults usually don't have to follow a security policy which would define possible security mechanisms and rules, something that would be necessary in an public environment.

4.2.3.2 Operate and Maintain

Handling multiple different software parts can not only create security risks, it also increases the complexity during operation. Installations and managing configuration data can easily grow to be a maintenance intensive task in a distributed environment. Our openHAB solution does not only consist of many software components, it also has to be distributed across multiple buildings. In a classical on premise environment, it would be necessary to setup all needed applications on every machine, each with the correct configuration for both the OS and the applications. Every change of a configuration, executed on one of the machines, needs to be mirrored at a central location to keep track of the current settings. When one of the machines has an system error and won't start any more, these steps need to be repeated for all applications which ran on that machine. Most applications we are working with like openHAB and Node-RED are stateful applications, they keep track of state data using files on disk (e.g. in case of openHAB, all changes made from PaperUI). For this user data, backups are necessary to ensure a successful recovery when a system fails or to revert erroneous changes. All needed work grows easily with the amount of machines and applications that are used. Especially when executing setup and configuration steps manually, it can not only get tedious, but is also error prone as these steps would not be easily reproducible.

Installing multiple components on a single machine can lead to influences between the applications and cause unexpected problems. Possible is for example that a faulty application, which leads to erroneous system behaviour, prevents the other components from operating correctly. This can get especially problematic when adding new components to the system. In return this can lead to lower availability of our applications.

Summarized this means, to enable proper installations, configuration changes and backups, it is necessary to keep track of different sets of files, in different directories and in addition for many different buildings. This can increase the demand for additional educated personnel (which is another big challenge for smart buildings (trend:research, 2015)) and therefore the cost needed to run such a system.

A solution to this needs to simplify the handling of configuration files and application data as well as the management of backups. The goal here is to reduce the needed effort for these tasks and in return keep possible additional maintenance cost low.

4.2.4 Creating a Better Solution

A part of the mentioned shortcomings can be solved with solutions like openHABian. This ready to use image comes with most parts needed by a home user and allows to easily install additional components from an additional UI. For a public building a similar image could be build, but this would fast render as ineffective as we have to deal with different possible target environments (e.g. bare metal, VMs) and different needs towards the components. Another solution could be the use of a managed and automated infrastructure with the help of tools like *puppet* or *chef*, but they come with a high complexity and initial effort.

A different approach to tackle these issues, is the usage of containers. A technology with a growing popularity in recent years(Forrester Consulting, 2017). Container engines like *rkt*, *LXC* and Docker allow developers to package and execute applications in a sandboxed environment with all necessary dependencies. They can be combined with orchestrating tools like *kubernetes* and Docker Swarm to gain a solution with an easier and central management of applications and their configuration.

The following pages showcase how containers, together with an orchestration engine, can help to create a better environment for running an openHAB stack in a public building. As an example the described solution is based on Docker with the accompanying orchestrator Docker Swarm.

4.2.4.1 Moving to Docker

As the title of this paper already suggested, we will use cloud technologies to tackle some of the mentioned shortcomings. The term cloud can have several meanings and can be used on different levels (like IaaS, PaaS, SaaS) (Hentschel and Leyh, 2016) and therefore needs some explanation in this context. We do not rely on specific cloud providers like Amazon Web Services (AWS) or Google Cloud Platform, but we will use tools that allow us to execute applications in sandboxed environments and help us manage them independently from the used local infrastructure. In our example we use Docker and its orchestrator Docker Swarm as the main tool to show case the advantages of a cloud like environment. An advantage of Docker is, that it is easy to get started with. It is quick and simple to install on provisioned VMs (when available, e.g. in universities with their own PaaS infrastructure),

as well as on bare metal computers (e.g. in a school without proper server infrastructure). A special infrastructure is therefore not necessary and there are no licence fees for its usage.

4.2.4.2 Deploying with Docker

Deploying and running our applications in containers helps us to gain multiple advantages. The sandboxed environments make installations easier, the applications don't have to rely on packages installed on the host OS, but manage them in their own separated environment. Applications are also less likely to influence other components on the system. When they fail it only concerns the container they are running inside. What really simplifies the deployment itself though, is the easy definition of whole sets of applications working together. In case of Docker this is done with *Docker Compose*. A `docker-compose.yml` is a configuration file using the *YAML* format. It allows us to define a set of services with the application containers we need (see Listing 1 for a simplified example). When running a compose file, images containing the defined applications are automatically downloaded and started, manual installation routines are not required any more.

```
services:
  openhab:
    image: "openhab/openhab"
  nodered:
    image: "nodered/node-red-data"
  mqtt:
    image: "eclipse-mosquitto"
```

Listing 1. Simplified example with three services

This approach is extended by the container orchestrator Docker Swarm, considering the use case of multiple buildings we intend to use. Docker Swarm allows to connect machines (nodes) running the Docker daemon with a single command. The connected machines then operate as a cluster. When the stack defined in the compose file is executed, the connected nodes will now download and run the needed containers. As this happens by default, depending on the available resources, it is possible to assign labels to the nodes and depending on them distribute containers. Listing 2 pictures an example where two openHAB services are placed on different nodes depending on the assigned `building` label.

```
services:
  openhab-building1:
    image: "openhab/openhab"
    deploy:
      placement:
        constraints:
          - node.labels.building == b1
  openhab-building2:
    image: "openhab/openhab"
    deploy:
      placement:
        constraints:
          - node.labels.building == b2
```

Listing 2. Placement of two services in a swarm

Using containers and an orchestrator simplifies deployment a lot and allows us to get an easy overview of our applications. In addition it also handles most networking tasks by connecting all containers within their own network and makes them reachable by their service name across all nodes.

4.2.4.3 A More Secure Network

As indicated previously, a security policy is necessary to be able to achieve a defined level of security. Derived from the shortcoming described earlier considering the defaults of our stack and the surroundings of public buildings, we can define a sample set of policies that we will try to comply with:

1. By default access to none of the applications (or their ports) shall be allowed.
2. When access is needed, the applications (or their ports) are explicitly exposed to be usable. The goal is to offer as little open ports as possible.
3. Only users which are able to authenticate themselves, shall be able to use any of the applications.

These rules will help to minimize the exposed area of our system and restrict access to selected users. Especially when dealing with networks with possible uncertain participants these simple rules can help to gain a basic security level by lowering the target area.

The first step towards this was already done by moving our applications into Docker containers in the previous step. Docker takes charge of handling the networking between the containers. In case of a swarm stack, Docker uses a so called overlay network (Church, n.d.). This uses Virtual Extensible LANs (VXLAN) to create virtual networks between the connected Docker hosts and their containers. These virtual networks run separated on top of the underlying host. Therefore by default the containers in the overlay network are not reachable from the host nor the network the host is connected to. The applications inside the containers can still communicate easily with each other, even in between the hosts (see Figure 3).

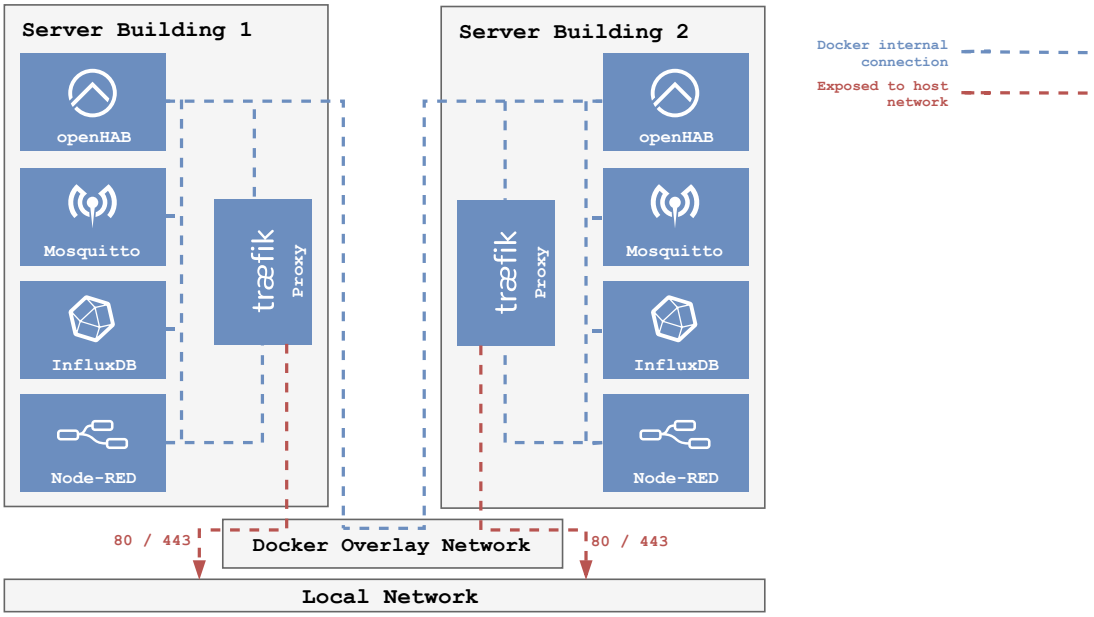


Figure 3. A stack using Docker networking and traefik

To be able to reach an application when it is necessary (e.g. for WebUIs) Docker allows to define accessible ports. For these, Docker creates bridges from the host port to the port of the container as if the application would run directly on the host (e.g. access from the local host network is also possible). Instead of simply exposing all WebUIs to the network, additional security (as well as simplicity) can be achieved by adding a web server or reverse proxy to the stack. It adds the mentioned ability of authenticating users and can reduce the number of our exposed ports to just one (either 80 or 443 with SSL). The reverse proxy then passes all requests to the correct applications and allows the usage of easy to remember subdomains instead of ports. In the pictured example we use *traefik* for this task, a reverse proxy that is tailored for cloud environments and can be configured using the deployment labels directly in our compose file. Other solutions like *nginx* could also be used, but don't offer the same level of integration by default.

With Docker networking and traefik we can cover a basic security policy. By default our applications are not reachable from the local network they are running on, in return only a single port is accessible which also handles user authentication.

4.2.4.4 Manage Configurations

Manually managing the configuration data of openHAB and it's accompanying components on multiple machines, turned out to be insufficient regarding maintainability and availability. While Docker Swarm helped us to define

an easier installation and deployment of our applications on a set of different machines, it is also important to run them configured correctly. Cloud native applications like traefik are mostly configurable directly from the compose file using either labels or environment variables. However, this is not necessarily possible in case of classical server applications like openHAB, Node-RED or Mosquitto. For these cases Docker Swarm has the possibility to define so called *Docker Config* entries. These are configuration files typically placed along the compose file and named in the `configs:` section. When the swarm stack is started (or updated) these files will be loaded into the *raft log*. This is an encrypted storage that Docker uses to ensure consensus between all nodes and therefore is available on each. From this raft log the file is then mounted into the containers by the local Docker daemon. Because the raft log is always distributed to connected machines, it is ensured that even new machines (e.g. after replacing a broken one) have the needed configuration files available and are able to re-spawn missing services seamlessly. Another advantage of this solution is, that all configuration files are collected at a single location together with the compose file. This enables us to use additional services to properly manage configuration changes (see Figure 4 *(a)*). A good solution for this is provided by version control systems (VCS) like Git. This allows us to keep track of all configuration changes and additionally keeps them at a different location. A VCS also allows to revert faulty changes and in case of a recovery it is easy to simply restore all settings from the repository.

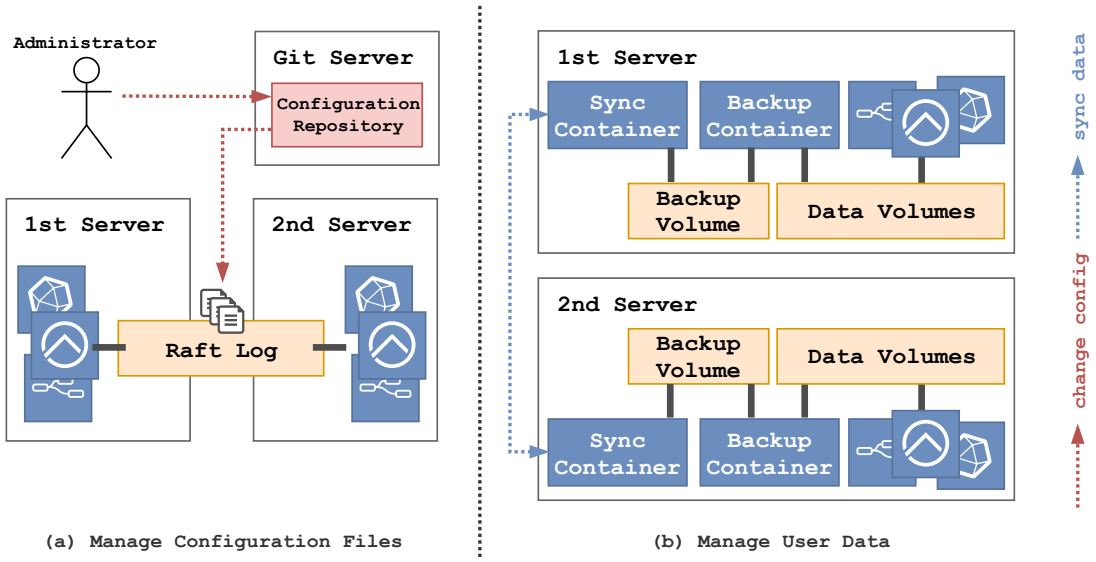


Figure 4. Manage user data and configurations within an orchestrator

4.2.4.5 Manage User Data

While Docker Config is a perfect solution for static configuration data and settings, it does not work for user or application data. This data sports regular changes also applied from within the corresponding application itself and requires write access, which is not possible with Docker Config.

User data persists the ongoing state of an application (e.g. channel links in openHAB) but can also represent temporary files or cached d ata. While the latter is usually not needed to be able to restore an application, state data is important. Docker makes it easier to separate these two. Persistent data is added to Docker using *Docker Volumes*. They represent a dedicated area on the local disk (e.g. in `/var/lib/docker/volumes`) and are managed by Docker. They enable us to dynamically assign it to application containers with the `-v` flag or directly in the compose file with the `volume:` entry.

Volumes sport a good solution on a single machine locally, but they are not designed for a distributed use case. While Docker Swarm synchronizes configuration data through all nodes, volumes are created locally on each node. This means if a machine with an `openhab_data` volume fails, it can be restarted on another node, but all user data stored previously in the volume is lost.

One solution to overcome this issue is to use a network storage solution like NFS. This central network storage can then be used by Docker with the help of a volume plugin as the backend for it's Docker Volumes. This allows to store the data in a separate location in the network. In case a machine stops working, a replacement can simply access the same data from the network storage. While this provides us with a nice solution in case of a system

failure, it relies heavily on a proper network connection with high transfer rates. All data needs constantly be moved across the network for each machine, generating a high load on the network. When the network (or the connection to the storage server in general) fails, the applications will not be able to access their data.

An alternative to network storage for this use case are software defined storage (SDS) solutions like *Ceph* or *GlusterFS*. Similar to network storage they are attached to Docker using a volume plugin. SDS solutions allow to define clusters of connected machines as a single storage target, but the data is replicated between all machines. This allows to have a replica of all data locally. In case of a failure, a new machine would also be added to the cluster and receives a local copy of all needed data to mount the volumes again. While this solution is better in terms of network requirements (changes are written locally and then mirrored), it still depends on a solid network. Own tests have shown that e.g. GlusterFS relies on a consent of replicas, this means when a single machine in a 3-machine cluster is isolated (by loosing connection to the other two) it will stop working even locally as it cannot replicate it's data any more.

Another way to deal with possible failures is to accept and define a certain tolerance of errors, e.g. data has to be available from at least the day before (Coulouris et al., 2002, p. 643). In that case the usage of classical backups can be a feasible solution. As Docker keeps its data in volumes, it is easy to locate all important files. *Volumerize* is a backup utility made for Docker Volumes and is deployed as a Docker container itself. It uses the robust tool *duplicity* as its backend and allows to define backups for Docker Volumes which are stored either locally or in a cloud storage like Amazons S3. It also is able to directly restore the data to a volume again. To ensure this data is available when a machine gets replaced, this can be combined with a synchronization solution like *unison* which keeps the backup data in sync between all machines (see Figure 4 *(b)*). This is only necessary when using a completely local backup though, when using cloud storage this would not be necessary. The advantage compared to the other solutions above is, that network access is only needed when synchronizing backups, not during normal operation, having a minimal impact and little needs towards the network. Another advantage of a backup solution is, that it also allows to move back to older revisions of the data, e.g. in case a change caused an error and is not reversible. This solution (volumerize and unison) is also completely defined in the Docker environment and as such does not rely on any external services.

To sum it up user data is essential to be able to continue operation after a system failure. A local backup and sync setup can be a feasible solution, especially in case of smaller setups and less reliable infrastructure. Network Attached Storage, SDS or a combination of both are a great solution when working in an environment where such solutions are already available (e.g. universities with a proper virtual infrastructure). They add the advantage of an easier and more straightforward failure recovery and should be used when possible.

4.3 Conclusion

This work pointed out that public buildings sport additional needs, not covered by a usual home system. Security and maintainability are key drivers towards a simple and cost effective solution. A complete openHAB stack consists of more parts then openHAB itself. This makes the setup and maintenance extensive, especially when considering multiple buildings. Security needs also special care as additional network communication is needed.

Using cloud technologies like Docker (swarm) can help to meet these needs. Container networking can help to create a more secure separated networking environment, while swarm definitions can reduce maintainability by collecting needed information at a central position. Using Docker Config together with a backup solution like volumerize (synced with unison) helps to maintain a higher availability and easier recovery.

While a local backup is a feasible solution in a simple environment, if better infrastructure is available it should be used. Network storage and cluster file systems can help to gain an even higher availability by removing the need of reinstalls and recovery in many cases.

The described solution of this work can be accessed on GitHub (Dobler, 2018) with further details on how to use it. While offering a first basic starting point, it needs to be extended. Additional tooling to ease the usage, as well as further components like a central log output and failure notifications are missing and should be added.

Further development of the used applications can also be helpful and enable a proper distributed setup. The available UIs for now are only aware of their own openHAB instance. Adding possibilities to switch between them or combine them in a single UI could be added. The backend could also be made aware of other instances, e.g. synchronizing configuration data between instances can make failover recovery even easier. Adding compatibility with cloud native configuration stores could also be an option.

4.4 References

Church, M., n.d. Docker Reference Architecture: Designing Scalable, Portable Docker Container Networks. URL https://success.docker.com/article/networking. [Accessed 08th August 2018].

Cisco Systems, 2018. Cisco Energy Management Suite - Product page. URL https://www.cisco.com/c/de_de/products/switches/energy-management-technology/index.html. [Accessed 20th February 2018].

Coulouris, G., Dollimore, J., and Kindberg, T., 2002. *Verteilte Systeme - Konzepte und Design*. Pearson Studium, 3., überarbeitete Auflage edition.

David, J. S., Schuff, D., and St. Louis, R., Jan. 2002. In: . *Commun. ACM*, 45(1):101–106. ISSN 0001-0782. doi: 10.1145/502269.502273.

Deutsche Telekom AG, 2016. QIVICON Medieninformation - Gemeinsam wachsen: Kommunikationskonzern baut mit Partnern Europas führende Smart Home-Plattform aus. URL https://www.qivicon.com/de/meta/presse/gemeinsam-wachsen-kommunikationskonzern-baut-mit-partnern-europas-fuehrende-smart-home-plattform-aus/. [Accessed 7th May 2018].

DIN EN 15232-1:2017-12, 2017. *Energieeffizienz von Gebäuden - Teil 1: Einfluss von Gebäudeautomation und Gebäudemanagement - Module M10-4, 5, 6, 7, 8, 9, 10*. Beuth Verlag.

Dobler, A., 2018. openHAB Public Building Stack. URL https://github.com/Dobli/openhab-pb-stack. [Accessed 30th September 2018].

Forrester Consulting, 2017. Containers: Real Adoption And Use Cases In 2017. URL https://i.dell.com/sites/doccontent/business/solutions/whitepapers/en/Documents/Containers_Real_Adoption_2017_Dell_EMC_Forrester_Paper.pdf.

Fu, S., Shih, C.-Y., Jiang, Y., Ceriotti, M., Huan, X., and Marrón, P. J., 2017. In: . *CoRR*.

GfK, 2016. GfK Future of Smart Home Study. URL https://www.gfk.com/fileadmin/user_upload/dyna_content/GB/documents/Innovation_event/GfK_Future_of_Smart_Home__Global_.pdf.

Hentschel, R. and Leyh, C., 2016. In: . *HMD Praxis der Wirtschaftsinformatik*, 53(5):563–579.

Martin, C., February 2018. Speakers Pushing Smart Home Market To 800 Million Devices. URL https://www.mediapost.com/publications/article/313842/speakers-pushing-smart-home-market-to-800-million.html. [Accessed 20th February 2018].

mqtt.org, n.d. Frequently Asked Questions - MQTT Homepage. URL http://mqtt.org/faq. [Accessed 1st July 2018].

openHAB Community, n.d.a. Securing access to openHAB - openHAB User Manual. URL https://docs.openhab.org/installation/security.html. [Accessed 21st February 2018].

openHAB Community, n.d.b. openHABian - openHAB User Manual. URL https://docs.openhab.org/installation/openhabian.html. [Accessed 17th February 2018].

Pham, L. M., Rheddane, A. E., Donsez, D., and Palma, N. D., 2015. In: . *Annals of Telecommunications - annales des télécommunications*.

SoftGuide GmbH, 2018. Aktuelle Marktübersicht - CAFM (computer aided facility management) und Gebäudemanagement. URL https://www.softguide.de/software/facility-management-cafm. [Accessed 20th February 2018].

Tanenbaum, A. S. and van Steen, M., 2008. *Verteilte Systeme - Prinzipien und Paradigmen*. Pearson Studium, 2., aktualisierte Auflage edition.

techuk, 2017. State of the Connected Home 2017. URL https://www.techuk.org/component/techuksecurity/security/download/11743?file=The_Connected_Home_-_FINAL_PUBLISHED.pdf.

trend:research, 2015. Smart Building – Intelligente Gewerbe- und Industriegebäudeautomation in Deutschland bis 2025.

Ultimo Software Solutions, 2018. Facility Management Software - Product page. URL https://www.ultimo.com/de/software/facility-management-software. [Accessed 20th February 2018].

van Laak, C. and Schröder, A., 2017. Sanierungsstau an Schulen - Wenn der Putz von den Wänden bröckelt. URL http://www.deutschlandfunk.de/sanierungsstau-an-schulen-wenn-der-putz-von-den-waenden.724.de.html?dram:article_id=406399. [Accessed 17th June 2018].

Wlotzka, V., 2016. Implementierung von MQTT Services in openHAB. Hochschule Düsseldorf, Fachbereich Elektro- & Informationstechnik.

5 OPENLICHT – A SELF-LEARNING LIGHTING SYSTEM BASED ON OPENHAB

K. Bierzynski [a, *], F. Kalleder [b], P. Lutskov [a], F. Rohde [a],
D. M. Rodríguez [a], J. Mena-Carrillo [a], R. Schöne [c], U. Aßmann [c]

[a] Infineon Technologies AG, Am Campeon 1-15, D-85579 Neubiberg (Germany),
(Kay.Bierzynski, Pavel.Lutskov, David.MoralesRodriguez, Juan.MenaCarrillo, Frank.Rohde)@infineon.com
[b] Bernitz Electronics GmbH, Ferdinand-Lassalle-Str. 9, D-82008 Unterhaching (Germany), f.kalleder@dled.eu
[c] Software Technology Group, Technische Universität Dresden, Nöthnitzer Str. 46, D-01069, Dresden
(Germany), (rene.schoene, uwe.assmann)@tu-dresden.de

KEY WORDS: openHAB, Lighting, Machine Learning, Microservice, MAPE-K Loop, Reference Attribute Grammar, Smart Home

ABSTRACT:
Today's smart devices often can only be remotely controlled and generate and deliver data. In the lighting domain the research project *OpenLicht* wants to improve this situation by developing an intelligent lighting system that integrates smart light bulbs and sensors to adapt lighting to user activities and preferences. As an additional goal the *OpenLicht* system should run at the network edge, because developments in the last years have shown that cloud-based infrastructures are not sufficient to fulfil all demands of the growing Internet of Things. As solutions to issues of cloud-based systems like reliability and privacy, edge and fog computing were suggested by the research community. These approaches are investigated and used in *OpenLicht* to realize a self-learning lighting system. As the base of this system the open Home Automation Bus (openHAB) was chosen, because of its state of development, size of community and range of devices that can be connected to it. In this paper the extensions and adaptations planned for openHAB to realize the self-learning lighting control are presented. Furthermore, implementation details are described for those parts that are already completed. Besides the software, a general overview of *OpenLicht* and its hardware details are discussed in this paper.

5.1 Introduction

Today's smart home technology market provides consumers a rich choice of devices that are connectable with each other and that are able to take advantage of cloud-based services. The product range includes security systems, connected thermostats and voice command devices. In Germany, most revenue is achieved by selling smart speakers, gateways and appliances (IoT Analytics, 2017). A survey conducted by *AudioAnalytic* concluded that most participants would be interested in intelligent smart home solutions.

Smart bulbs like Philips Hue and Osram Lightify provide convenient functionalities for manual light operation, but lack functionality that would enable intelligent behavior of the smart homes lighting installation. The basic relationship between user and lighting system is not changed. The user either operates the system manually, or has to program it, e.g., using rules and scripts. This lack of profound change severely limits the adoption of Smart Home technology, as technology experts have pointed out (Higginbotham, 2018).

The comprehensive smart home product range causes interoperability problems: smart home devices are not connectable, as they provide incompatible interfaces, e.g., ZigBee, Z-Wave or Bluetooth.

Due to its reliance on cloud-based services, smart home technology might suffer from latency-issues, as the data has to be sent to a remote cloud-provider. Moreover, the user's privacy is at risk, as cloud-providers can act independently from the user and might share and use the received data in an improper way. User surveys have shown that privacy issues are an important purchase barrier for potential smart home users (PwC, 2017).

The research project *OpenLicht* (OpenLicht Consortium, 2018) contributes to the solution of the aforementioned problems by exploring a lighting control approach that (1) is self-learning in that the user is the subject that technology is learning from, rather than the manual operator or the programmer of automatic light behavior and

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 43–52,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

that (2) complements cloud-based services with locally provided services. The interoperability problem is addressed by incorporating openHAB as a middleware platform into the system. We Shave chosen openHAB, as it provides a rich set of device- and interface-bindings and is popular with a broad user community.

The goal of this paper is to provide an understanding for the *OpenLicht* hardware- and software-system and the planned openHAB extensions that will allow its users to apply a self-learning approach to achieve intelligent lighting control. In the next section an overview about the project is given. Goals and solution approaches are presented. Section 5.3 presents a hardware setup for the *OpenLicht* system and discusses potential alternatives. Section 5.4 describes the envisaged software architecture as well as individual architectural components. Section 5.5 presents the planned demonstrators, while section 5.6 concludes the paper.

5.2 Project Overview

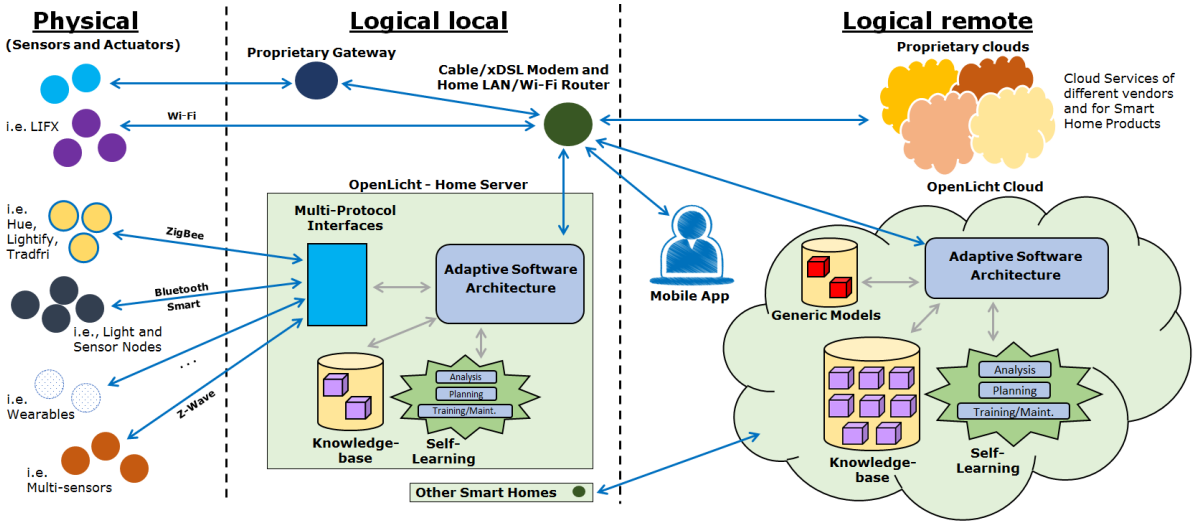


Figure 10. System architecture overview

Over the last years, many systems for smart lighting like Phillips Hue (Philips, 2018) or Ikea Tradfri (Ikea, 2018) appeared on the market. These systems provide a wide range of features such as daylight and occupancy simulation. However, apps or control devices are required for their use. This makes smart lights cumbersome and unintuitive to use. Furthermore, many of those systems need a connection to the cloud to work properly, which creates a privacy risk and threatens reliability, as already mentioned in the previous section.

OpenLicht, which is part of the Open Photonik program (VDI, 2018), explores how to alleviate these issues. State-of-the-art systems already employ automated light control; however, they are mostly rule-based. That means that users need to program those systems by themselves. Hence, the focus was set to a self-learning light system. Such a system is able to learn behavioral patterns and to correlate them with the preferences of the user. After a learn phase, it is ready to recognize what kind of activities users are doing and to adapt the light setting according to their preferences, current activities and the natural light. This idea is the central solution approach that is investigated in the scope of *OpenLicht*.

To get a better understanding on how a future smart home should look like, a survey was done. One question central for the learning part was where data about user activities and preferences should be stored and processed. The majority of the survey participants answered that they would prefer that the data remains in the home network for privacy reasons. To deal with this requirement, the self-learning system was combined with edge computing (Shi et al., 2016). Edge computing is an approach that describes the trend of moving data processing towards the network edge, leading to better latency and less traffic, both needed for the big amount of data created by the growing Internet of Things. Supporting this trend in the context of lighting applications, the automated machine learning framework *Ledge* was developed within the scope of *OpenLicht*. *Ledge* takes cleaned and labeled data of sensors or other data sources as well as configuration files that contain information about the execution environment as input and delivers C code of a trained machine learning model. The code can then be deployed on microcontrollers. It was designed to support makers and other developers in the development of intelligent lighting applications at the network edge. Besides *Ledge*, many small applications like openHAB bindings were written to process data and to connect sensors with XMC microcontrollers

(Infineon, 2018) and openHAB. The goal of these applications is to simplify prototyping, the usage of artificial intelligence and smart home hardware. At the end of the project, these small applications and the code of the complete system will be made available as open source software.

The overall envisioned architecture is depicted in Figure 10. It is divided into three parts, one containing all sensors and actuators, both of which are directly connected to gateways in the second part, the logical local processing part. In the logical parts, interfaces for remote accesses for example by apps are provided. Furthermore, it is the part where all processing takes place, either locally or remotely on cloud resources. In addition, the cloud resources are used to generate generic models from the data of multiple smart homes. These can be applied in a new smart home system to reduce the learning time. For evaluating the implementation of the architecture parts, a demo room was set up. Furthermore, a set of activities was defined that serve as test cases for the system in this room. Details of the implementation are described later in section 5.4.

Concerning hardware, central topics are design of a gateway for a self-learning system at the network edge and simplifications for development of light and sensor nodes. Two kinds of gateways are planned: The first type is a Raspberry Pi (Raspberry Pi Foundation, 2018) with an extension board developed in the project and assembled from open source components. This gateway serves as the central control unit of the *OpenLicht* system. The second type is a miniaturized gateway, which is used to scale the system. Furthermore, it is designed as a modular component that can serve as a control and processing unit for smaller applications. Besides these gateways, other hardware components are developed to simplify the prototyping and development of light solutions. To give makers and developers an overview and to support them in handling these components, a web platform is provided including tools that simplify searching for parts as well as for designing and simulating light solutions.

5.3 OpenLicht Hardware Concepts and Solutions

Core component of the *OpenLicht* hardware architecture is a central gateway, depicted in Figure 11, which is responsible for data processing and all control tasks. The gateway is connected to the internet via a Wi-Fi router which is considered to be available in almost all private households. For data transmission to the used sensors and actuators (exclusively lamps in the presented case) wireless communication interfaces are used in *OpenLicht*. The used radio interfaces are meant to be integrated directly into the gateway by usage of radio modules. In order to use proprietary radio interfaces or to include far-away areas, suitable proprietary gateways or bridges may also be integrated into the *OpenLicht* system.

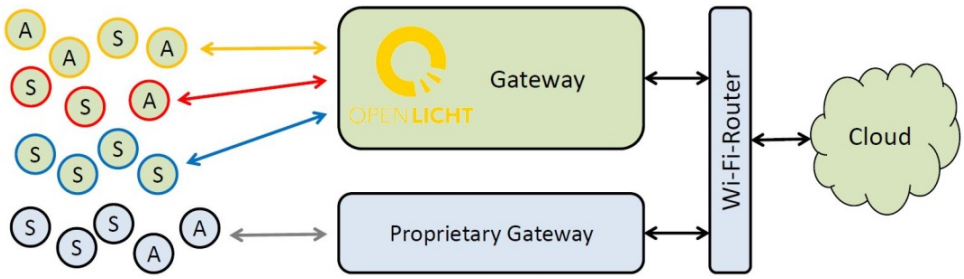


Figure 11. *OpenLicht* hardware architecture

To exchange data with sensors and actuators a wide range of wireless interfaces can be used in the system. Due to the high acceptance in the smart home sector and the great number of already existing products the following wireless interfaces are given priority in the *OpenLicht* project: ZigBee, Z-Wave, Bluetooth and Bluetooth Smart. Also, Wi-Fi is used because it can be assumed to be a standard component in most home networks and therefore no additional hardware is required.

In the smart home sector, a distinction has to be made between two different types of gateways. Some only work together with higher level control units or with the cloud, the others are largely independent and constitute such a higher level control unit by themselves. As part of the project, two different versions of the *OpenLicht* gateway are being developed.

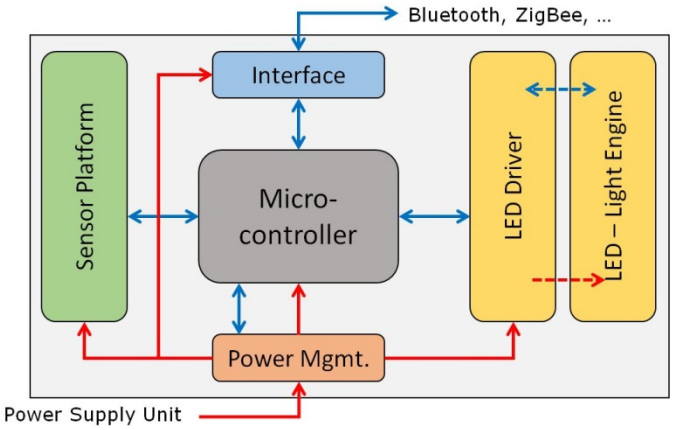


Figure 12. Architecture of the Light and Sensor Node

The open source version is based on a Raspberry Pi single-board computer with an additional expansion board. Due to additional components such as embedded interfaces, a security module as well as control elements for user interaction, the underlying hardware is extended. With its hardware capabilities, the Open Source Gateway is able to control the entire *OpenLicht* system. This includes not only the application of machine-learning algorithms, data processing and communication with the individual system components, but also the provisioning of suitable user interfaces like a web interface or app interfaces.

The miniaturized version of the gateway is based on a microcontroller architecture and is meant to be used for up- and down-scaling the *OpenLicht* system or for connecting smaller remote areas to it. To exchange data this gateway is equipped with pre-selected wired and wireless interfaces, e.g. Ethernet and Bluetooth, which are integrated directly into hardware. In stand-alone operation, the miniaturized version is able to control actuators, to capture and preprocess sensor data and to perform simple control algorithms. In the proposed *OpenLicht* system, this gateway requires a higher level control unit, such as the Open Source Gateway, which is able to control the entire system.

As part of the *OpenLicht* project, so called Light and Sensor Nodes will be developed. These may be sensors or actuators as well as a combination of both. A modular concept is applied to simplify the development of additional nodes. As shown in Figure 10Figure 12, functional modules are developed which can be combined according to the desired use case.

5.4 OpenLicht Software Concepts and Solutions

Based on the survey mentioned in section 5.2, requirements for the *OpenLicht* system were gathered. Among others, the following software requirements have been included: Integration of new components such as wearables, new sensors, as well as new software components during runtime; change between local and remote execution of computation; device-agnostic representation of sensors and actuators; learning of user activities and their preferences; maintaining service without internet connection.

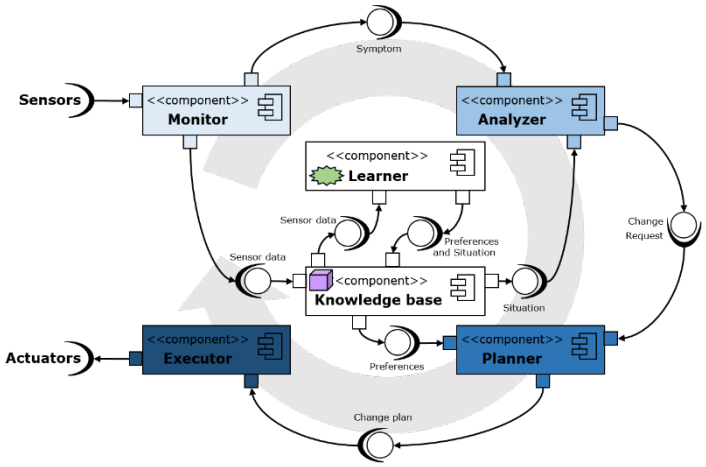


Figure 13. Extended MAPE-K feedback loop

Some of those requirements were met by employing openHAB as a middleware to connect to various sensors and actuators and thereby getting an abstract representation of them. Others were tackled in our project by developing extensions for openHAB described later in more detail.

The software architecture is organized as an enhanced MAPE-K (IBM, 2016) feedback loop, which was proposed by IBM and can since be found in many concepts and implementations of adaptive and self-adaptive systems. The name is an acronym of its four phases *monitor*, *analyze*, *plan* and *execute*, as well as their source of information, the *knowledge base*. In *OpenLicht*, this concept was extended with another component, the *learner*, which is responsible for learning activities and preferences of the user.

In the following, we will briefly describe the basic concept of the MAPE-K architectural approach and our extension. The *monitor* component continuously receives sensor data and writes this data into the *knowledge base*, possibly aggregating it beforehand. Once an update occurs, the *analyze* component checks, whether it was relevant, i.e., if a previously learnt activity can be recognized, or if other changes occurred that would require the system to adapt itself. In both of these cases, the *plan* component is invoked, otherwise the *monitor* component takes over. The *planner* is the central component, as it computes the reaction of the system to changes. Therefore, sensor data, learnt situations, user preferences are combined to derive the needed actions. Those will then be forwarded to the *execute* component which generates suitable commands to be sent to the actuators. The *knowledge base* is responsible for storing all information known to the system, e.g., sensors, their position and current state, learnt activities, preferences related to those activities, other relevant context information as well as historical decisions of the system.

Our extension introduces the *learner* component having three tasks: First, it takes all the data from sensors, lamps and users that are required for learning activities and user preferences. Secondly, it estimates the current user activity when triggered by *analyze.* Thirdly, it determines the user preference in context of the current activity and generates a fitting lamp configuration plan after the *analyze* phase and on request from the *plan* component. Besides these main tasks, the *learner* monitors the input and output of all trained machine learning models and retrains them when a change in the input data or model results is detected.

To relieve the home system from the possibly heavy computation needed to, e.g., train the model produced by the *learner*, a subset of the components can be moved to remote execution in a known cloud-based server or to other nodes in the home network. To achieve this, all components have incoming and outgoing ports which can be plugged together as needed as shown in the following figure.

The last part of the software architecture that needs to be discussed is the machine learning pipeline depicted in Figure 14. Activity recognition and preference planning are the central components that realize the learning features of the *OpenLicht* system.

The first is part of the *analyze* step of the MAPE-K loop and takes sensor data as input. However, data of more complex sensors like radar sensors cannot be directly used in many cases. Their data must be preprocessed to obtain suitable information for the recognition. After the pre-processing, the current activity is predicted with a voting classifier (Zhang et al., 2014) that uses multiple feedforward neural networks and decision trees. The exact number of the used models depends on the size of the smart home or apartment in which the system is deployed.

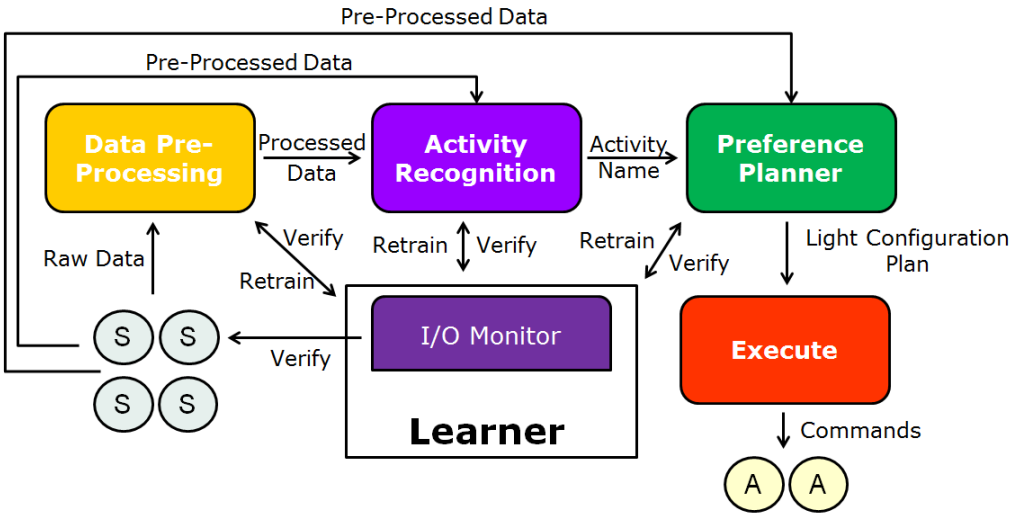


Figure 14. Learning process overview

The results of the prediction and data of some sensors are used as input for the preference planner component. This component is the central part of the *planner* and uses different sensor data as input depending on the trade-offs the system should make. The light intensity sensor is of fundamental importance, as it is necessary to configure the light in an area correctly. However, when the system needs to save energy, information from power meters will be utilized in the planning as well. Independent of used sensors, input data is processed by *DQN-Learning* (Mnih et al., 2015). This processing approach was chosen for the preference planner, as it is able to handle many different trade-offs due to its generality. After processing, the planner provides a list that contains the configuration of every lamp that is close to the current location of the acting user. This list is used to generate commands in the *execute* step.

The last component of the pipeline is the *I/O monitor*, which is part of the *learner*. This *monitor* is necessary, since the smart home is a dynamic environment. Weather, burglars, furniture rearrangement and new inhabitants are only some of the circumstances that can change the behavior of the sensor in and around the house and as a consequence the output of trained machine learning models. Hence, the *I/O monitor* records the input data after the data pre-processing and the results of the activity recognition and preference planner. They are then verified and tested for concept drift (Gama et al., 2014) by comparing them to historical data. In this context, *concept drift* means that the accuracy of the activity recognition or preference planner drops over time due to changing sensor behavior. When the *I/O monitor* detects a concept drift, it tries to find the cause in the input data and triggers a retraining of those models, whose results started to drift.

In the following subsections, we explain how we realized or plan to implement the MAPE-K loop parts with openHAB and extensions. However, the implementation of the cloud part of the system is not discussed as work on this part was not started yet. The *learner* is not covered within an additional subsection, since it is strongly interweaved with *analyze* and *plan*.

5.4.1 Monitor

The basic components of the system are the sensors and other data sources. The latter may include weather web sites, the state of the lamps, which are important for learning the preference of the user. To integrate them in the self-learning lighting system, the standard binding infrastructure of openHAB is used.

The next step in the system process is data pre-processing, whose kind and implementation depends on the sensors and their position within the data processing chain. On the lower levels, the data of one sensor is processed in most cases by filtering or normalizing the data. In some cases, such processing is provided by the sensor vendor and done directly on the sensor node. In case of no available pre-processing for sensors used in the *OpenLicht* demonstrators, we developed the necessary software and integrated it with the corresponding openHAB binding. We choose bindings as implementation points, because they are typically focused on a product or product family and may therefore preferably be extended with specialized data preparation and transformation services. An alternative implementation approach would be to let the bindings communicate with external sources that provide these services and delegate the processing to them. Delegation to external sources is one possible approach to scale the data pre-processing. Another strategy, we plan to utilize for the *OpenLicht* system, is to move the program with the services to nodes that are located logically between the sensor and the central gateway. This means that the binding would no longer communicate directly with the sensor, but with the program instead. Furthermore, it would be also possible to move the binding code at runtime from the central gateway to other nodes. In any case, the data pre-processing services would be monitored by the *I/O monitor* and be maintained by the *learner*, since they have a big impact on the data quality and often artificial intelligence approaches are their central components. After the data was transformed to some degree and checked by the *I/O monitor*, the higher level data pre-processing services aggregate the transformed data from multiple sensors into condensed information. These services can also be implemented in bindings to process data from product families. However, we tried to integrate them into the item infrastructure of openHAB to reuse as much of existing services as possible. In the approach we experimented with, the results from low level services are stored in items. These items are then used as input for the aggregation services. These services can be realized as custom rule triggers, when the result is not needed in other processes or as we did by using group items. They have a set of aggregation functions, but these are not sufficient for more complex processing. We tried to realize more functions and possibilities to use functions in conjunction with each other to solve this problem, which significantly enlarged the item definitions and made them more confusing. Furthermore, programming the services in this way was very restrictive. For these reasons, we decided to replace the representation of devices and the home by items with an approach that utilizes the *knowledge base* presented in section 5.4.5. Currently, the *knowledge base* runs as an extra system and is connected to openHAB over an MQTT broker. We are considering integrating the ideas behind the *knowledge base* into openHAB items.

The last component to be described in the *monitor* part is the *I/O monitor*. In the first version of the *OpenLicht* system, the *I/O monitor* is implemented as a set of openHAB rules with custom actions. For handling historical data, these rules access a SQL database that is connected over the *MariaDB JDBC* persistence service. However, the results of testing or verifying the data processing are archived by directly writing them into another SQL database that runs concurrently with the system and is separated from the database, which stores all item states. The system uses two databases to simplify testing and changing of the *I/O monitor*. In future iterations of the system, we plan to develop monitor functions that can be added to items or to *knowledge base* objects in a similar way as the aggregation functions are added to group items.

5.4.2 Analyze

In the *OpenLicht* system, the *analyze* phase of the MAPE-K loop consists mainly of the activity recognition. However, the transition from the data pre-processing to it is blurred, since the recognition is divided into many small parts and some of them, like determining the body posture of a person, could be categorized as pre-processing service or as an application of the *analyze* phase depending on the context and definition. Hence, the activity recognition and high level data pre-processing services are handled by the *I/O monitor*, *learner* and other parts of the system in the same way to provide flexibility for further developments.

The parts of the activity recognition are implemented in the first iteration of the *OpenLicht* system as openHAB custom actions. These actions use the optimized neural network and decision tree code of the *Weka* library (Hall, et al. 2009). Furthermore, they are integrated into openHAB rules, which communicate with each other. These rules, together with the custom action instances, form the mentioned voting classifier.

In the current system, all parts of the classifier run on a single Raspberry Pi together with the rest of the system. This will be changed in the future to enable scaling. It is planned to integrate a voter that is represented by one rule and an action instance into a *Docker* container (Rad et al., 2017). This allows the voter to be easily moved to another node of the home network. A special binding based on MQTT is intended for the communication with those other nodes. The container approach will not only be used to distribute the voting classifier, but for data processing of the system in general. Besides the implementation of the distribution, we are working at integrating the voters in the *knowledge base* for faster processing.

The voter actions do not include methods for learning, because separating the learning from the execution increases the system stability due to the possibility of testing and verifying the model before it is deployed to the running system or adapted at runtime. Hence, the *learner* component takes care of the training and is also implemented as a set of openHAB rules and custom actions. To select and train the algorithms, the *learner* uses *Weka* for activity recognition and the *Encog* library (Heaton, 2015) for the preference planner. It is planned for future system versions to unify both learning applications, to add more machine libraries to increase choice for developers and to only use the algorithm code from libraries. For this purpose, the *learner* will be reworked restructured to employ automated machine learning (Feurer et al., 2015). This means that some training steps are automated to reduce the amount of necessary support services and user input. The latter is especially important to realize training of the system at home. At the moment the system is not able to do this as it just reads the prepared data sets and learns from them. Afterwards, the learned parameters are sent to the corresponding custom actions, which save them in a file format of their library and load them when necessary. However, the actions only save the newest state of the algorithm parameters. Older states of the parameters are archived by the *learner* by writing them directly into the SQL database used by the *I/O monitor* for storing testing results. The adaptation of the models in response to changes is done in the *learner* as well. It has a copy of every model used in an action to do this and waits for the *I/O monitor* to trigger the relearning of them.

5.4.3 Plan

The preference planner makes up the *plan* step of the *OpenLicht* system. The planner is implemented like the activity recognition in the current system as a set of openHAB rules and custom actions. The actions use the algorithm code of the *Encog* library. Furthermore, the planner has two *DQN-Learner* models per room. One model is used for predicting the color of the lamps and the other for determining the light intensity and temperature. The advantage of splitting the prediction like this is that for lamps without color only one model needs to be used and maintenance becomes simpler. After both models have finished processing, their results are combined into one list and sent to the *execute* part. For future iterations of the system, we plan to implement the planner like the activity recognition using the *knowledge base*.

The *DQN-Learner* is a reinforcement learning algorithm and as such relies on user input to learn the correct configurations. In the *OpenLicht* system, the preference learning is designed as follows: First, the system only

gathers data about activities and light states. After the system has learned activities and corresponding light preferences, it starts to control the lights itself. The user can set the system into a learn state to adapt preferences any time after the initial training. Only the last part of this process is implemented in the current system. It was realized with a switch item that triggers some monitor rules, which are part of the *I/O monitor*. These rules gather information about the state changes of the lamps and send them to the *learner*. Then, the *learner* adapts the corresponding *DQN* models.

5.4.4 Execute

The implementation of the *execute* phase of the *OpenLicht* system is mainly provided by the openHAB binding infrastructure. Only an additional openHAB rule is needed that iterates through the list provided by the preference planner and triggers the bindings of the lamps whose state should be reconfigured.

5.4.5 Knowledge Base

To store all relevant data in a smart home, an extensible mechanism is needed to describe the structure and possible computation for this data. We use the rather unusual framework of Reference Attribute Grammars (RAGs) (Hedin, 2000) to cope with these challenges. RAGs are most commonly used in the compiler construction domain to describe the structure of parsed source code and to transform this representation into code of another language or into machine code. We use RAGs to describe the structure of the *knowledge base*, i.e., the structure of possible information to store. This allows us to additionally specify computation to infer new or to change existing pieces of information.

Specifically, we can mirror the information stored in openHAB and extend it with the learnt models like a decision tree or a neural network. Those models are produced by the *learner* component and pushed into the *knowledge base* accordingly. Using RAGs, we can specify an interpretation function for those models and, thus, evaluate them for the current situation. Hence, we can evaluate the decision tree and get the correct preference.

To implement our *knowledge base*, we employ JastAdd (Ekman & Hedin, 2007), an extensible RAG system extended by bidirectional relations (Mey et al., 2018). The structure of the data is specified using an EBNF-like syntax for production rules, i.e., to describe possible types in the *knowledge base*. With this, a tree-like structure results, which can be extended to a graph using reference edges. To specify computation, attributes can be defined for certain types. As JastAdd produces plain Java code, attributes are just Java methods with a special signature syntax.

An intriguing feature of JastAdd is its ability to incrementally evaluate the specified attributes. This means, if an attribute was previously computed and the *knowledge base* has not changed in the meantime, the value is cached, and will be returned immediately. If there were some changes, only those attributes will be recomputed that depend on changed data.

Mirroring the current state of all openHAB items in the *knowledge base* has also another advantage. We are able to write and process complex ECA-rules, which can use previous states of sensors, recognized activities and other context information not available in openHAB. Furthermore, the actions to execute upon recognizing the event of a rule can be more complex, such as updating the learnt models, or any other computation expressible with Java.

5.5 Demonstrator

As part of the research project, a demo room equipped with the *OpenLicht* system is set up. This room is used to evaluate the developed hardware and software, as well as to collect sensor data needed for machine learning and modelling. Later, it will also be the final demonstrator of the overall system. As central control component the open source version of the *OpenLicht* gateway is used. In addition to the developed Light and Sensor Nodes, lamps and sensors from various vendors are also integrated. A floor plan of the currently used demo room is shown in Figure 15. There are two areas, a work and a relax area, where the user can do various activities such as working on a computer, reading newspapers or sleeping. Based on the activity and on the user preferences the lighting in the room is adjusted.

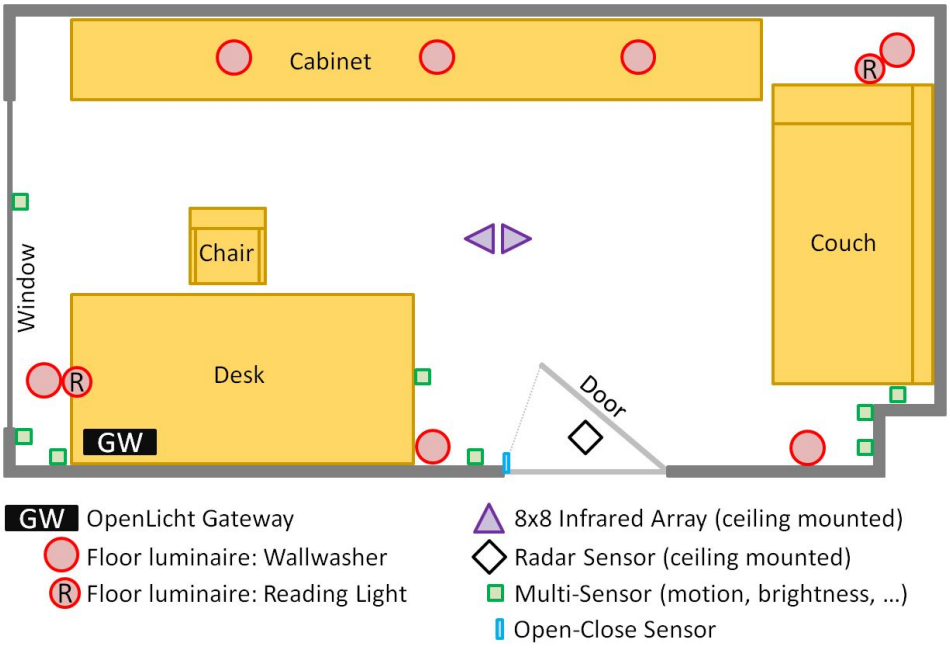


Figure 15. *OpenLicht* demo room

5.6 Conclusion

In this paper, we gave an overview about the research project *OpenLicht* and presented its main part an intelligent, self-learning lighting system.

The central requirements of the system are reliability, scalability, usability, privacy, support of various devices and communication protocols, as well as an intuitive user interface. Those goals are partly achieved by using the framework openHAB as system base and by employing the edge computing paradigm, which ensures that computation is done within the local network, to increase reliability and decrease latency. Moreover, machine learning algorithms are used to improve the usability by using them to automate the light control. For this automation, the applied algorithms learn user activities and preferences. Afterwards, the system can recognize the learned activities of the user and apply the fitting user preference pattern. The patterns are used to generate a light configuration plan. The infrastructure for the artificial intelligence of the OpenLicht system was integrated into openHAB as an extension module. Another component added, is the extensible *knowledge base*, based on Reference Attribute Grammars serving as a performant base for the whole system.

To support various communication protocols such as ZigBee, Z-Wave and Bluetooth Smart, radio modules, a Raspberry Pi single board computer and a specially developed extension board were combined to create a gateway.

The self-learning lighting system developed within OpenLicht will be evaluated in a demo room. Furthermore, small kits for users will be put together to evaluate a limited version of our OpenLicht system in their homes. With the results gathered during the evaluation, the software components will be finalized and possible limitations of the system will be determined. The hardware components and the *OpenLicht* platform will be assessed in the scope of fairs, competitions, and workshops in schools.

5.7 References

Ekman, T., & Görel H., 2007. The JastAdd System - Modular Extensible Compiler Construction. In: *Science of Computer Programming* 69, no. 1.

Feurer, M., Klein A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F., 2015. Efficient and Robust Automated Machine Learning. In: *Advances in Neural Information Processing Systems,* pp. 2962-2970.

Gama, J., Zliobaité, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A., 2014. A Survey on Concept Drift Adaptation. In: *ACM Computing Surveys (CSUR)* 46, no. 4.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H., 2009. The WEKA Data Mining Software: An Update. In: *SIGKDD Explorations Newsletter* 11, no. 1, pp. 10-18.

Heaton, J., 2015. Encog: Library of Interchangeable Machine Learning Models for Java and C#. In: *Journal of Machine Learning Research* 16, pp. 1243-1247.

Hedin, G., 2000. Reference Attributed Grammars. In: *Informatica (Slovenia)* 24, no. 3, pp. 301-317.

Higginbotham, S., 2018. *Rethinking the smart home in 2018.* https://staceyoniot.com/rethinking-the-smart-home-in-2018/ (accessed September 12, 2018)

IBM, 2016. An Architectural Blueprint for Autonomic Computing. *IBM White Paper*, 2016.

Ikea, 2018. *Ikea Tradfri Website.* https://www.ikea.com/gb/en/products/lighting/smart-lighting/ (accessed October 10, 2018)

Infineon, 2018. *XMC Microcontroller Website.* https://www.infineon.com/cms/en/product/microcontroller/32-bit-industrial-microcontroller-based-on-arm-cortex-m/32-bit-xmc4000-industrial-microcontroller-arm-cortex-m4/?redirId=41425 (accessed October 10, 2018)

IoT Analytics, 2017. *State of the Smart Home Market 2017.* https://iot-analytics.com/wp/wp-content/uploads/2017/12/StateofSmartHomeMarket2017-vf.pdf (accessed September 12, 2018)

Mey, J., et al., 2018. Continuous Model Validation Using Reference Attribute Grammars. In: *International Conference on Software Language Engineering.* ACM, 2018, Boston, USA.

Mnih, V., et al., 2015. Human-level control through deep reinforcement learning. In: *Nature* 518, p. 529.

OpenLicht Consortium, 2018. *OpenLicht Website.* http://openlicht.de/ (accessed October 15, 2018)

Philips, 2018. *Philips Hue Website.* https://www2.meethue.com/ (accessed October 10, 2018)

PwC, 2017. *Smart home, seamless life: Unlocking a culture of convenience.* https://www.pwc.com/CISconnectedhome (accessed September 14, 2018).

Rad, B. B., Bhatti, H. J., & Ahmadi, M., 2017. An Introduction to Docker and Analysis of its Performance. In: *International Journal of Computer Science and Network Security* 17, no. 3.

Raspberry Pi Foundation, 2018. *Raspberry Pi Website.* https://www.raspberrypi.org/ (accessed October 10, 2018)

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L., 2016. Edge computing: Vision and challenges. In: *IEEE Internet of Things Journal* 3, no. 5.

VDI, 2018. *Open Photonik Website.* https://www.photonikforschung.de/projekte/open-innovation/foerdermassnahme/open-photonik-innovationsprozess.html (accessed October 10, 2018)

Zhang, Y., Zhang, H., Cai, J., & Yang, B., 2014. A Weighted Voting Classifier Based on Differential Evolution. In: *Abstract and Applied Analysis, vol. 2014*.

5.8 Acknowledgements

This work is part of the research project *OpenLicht* (project number 13N14047) and is funded by the German Federal Ministry of Education and Research (BMBF). We thank our partner Deggendorf Institute of Technology for their cooperation in *OpenLicht*.

6 EMPLOYING BUILDING SPATIAL DATA, MAPS AND 3D MODELS IN A WEB-BASED INDOOR POSITIONING VISUALIZATION

R. Sihombing [a, *], V. Coors [a]

[a] Department of Geomatics, Computer Science and Mathematics, University of Applied Sciences Stuttgart
Schellingstraße 24, D-70174 Stuttgart (Germany), (rosanny.sihombing, volker.coors)@hft-stuttgart.de

KEY WORDS: Indoor Visualization, Web-based Visualization, 3D Model

ABSTRACT:
In order to raise energy consumption awareness, it is essential to inform all building occupants of energy consumption in particular areas which can be done by providing easy access to energy-related data and indoor positioning of related sensors. This paper proposes an alternative indoor positioning approach that can be employed by devices which not constantly moving from time to time and do not have spatial attribute nor equipped with a wireless indoor positioning system or similar system. By exploiting building georeferenced floor plans and associating sensors and the rooms in which they are mounted, sensors' location inside the building can be obtained for visualization in a web page. The proposed approach results in an indoor sensor data monitoring tool in 2D and 3D platforms which can be accessed easily in a web browser by all building occupants.

6.1 Introduction

To deal with a sustainable future, nowadays numerous higher institutions around the world aim to become climate neutral universities, which means they want to have a net zero carbon dioxide emissions. A recent study (Udas et al., 2018) mentions 600 universities in the USA have been committed to becoming carbon-neutral universities since 2009, and most of them have set a specific emission reduction target to be achieved by 2020. The study also mentions that in Germany, the Environment Campus of Birkenfeld, the Leuphana University of Lueneburg and the University of Applied Sciences for Sustainable Development Eberswalde are already carbon-neutral. Becoming a carbon-neutral university can be achieved by running a transdisciplinary research project addressing various sustainable concepts which includes relevant internal and external stakeholders' factors such as redevelopment strategy to ensure energy efficient campus building, integration of renewable energies, new financing model, and also human behavior inside the buildings.

To influence human behavior, it is important to provide building occupants with useful information so that they can adjust their behavior. In some cases, energy-related information is accessible to limited groups of people, mainly those who work directly with the data, but not to the largest group of the building occupants, for example, the students if it is in a university. Therefore, as part of the effort to raise awareness of energy consumption and to bring positive influence, web-based data visualization was created as a prototype platform (Coors et al., 2016) to make energy-related sensor information available to all groups of building occupants. The platform provides the building occupants access to the historical sensor data which they can use for further purpose in energy-related projects.

A simple indoor mapping study uses Scalable Vector Graphics (SVG) to draw a floor plan on top of the map (Ohrt & Turau, 2013) and gives a convincing result in terms of user-friendly access. However, it focusses on indoor navigation system instead of complementing the indoor localization with useful information regarding the room/building. Coors (Coors et al., 2016) has presented the first prototype of Living Lab platform, which provides information about environmental aspects of a building that was collected by building occupants using an Android mobile application. This information then can be displayed on a web-based map later on. The aim of this paper is to summarize how indoor positioning visualization is implemented to locate energy-related sensors which are not equipped with a wireless positioning system using the Living Lab web platform as the foundation. We describe how the energy-related data, spatial data, 2D map and 3D models have been integrated together for positioning visualization by linking the details of an energy-related sensor and the spatial data of a room where

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 53–60,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

the sensor is mounted. The focus is on indoor positioning visualization which is complemented by environmental aspects of the building rather than providing a navigation system.

This paper is structured as follows. Section 2 explains the proposed approach to achieve the objective of this paper, while section 3 presents the implementation of the proposed approach. The result is discussed in section 4 and then this paper is concluded in section 5.

6.2 Proposed Approach

The proposed approach is designed to be a loosely coupled design that separates the data source, application logic and positioning visualization in order to have more flexibility to improve each aspect separately (Figure 1).

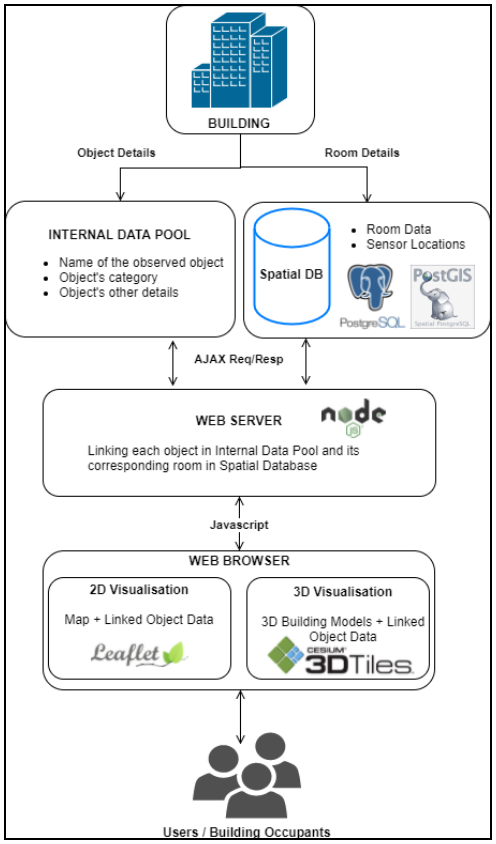


Figure 16. System design

6.2.1 Data Source

This aspect contains all the necessary data to obtain the position of a static object, which means it does not constantly move from time to time. The Internal Data Pool stores the details of the observed object inside the building, such as name and category, and the Building Spatial Database stores spatial data of the building, such as the MultiPolygon coordinates of the rooms on each floor, in which room an object is located and the floor/room number. A data pool does not necessarily mean another database; it could also be any other sources of data which contain valuable data.

6.2.2 Application Logic

This aspect links the data source and the positioning visualization in the web browser by exploiting the relation between the details of an object in the data pool and the details of the room where the object is located in the building database. For instance, if a temperature sensor is mounted in Room 001, then its name and measurement data stored in object data pool can be linked to the details of Room 001 stored in the building database. By doing so, the temperature sensor is now attributed with a spatial value. Furthermore, this aspect also handles the

calculation of the center of a MultiPolygon so that the calculation result can be used to put an object on top of a map and inside a 3D building model.

6.2.3 Positioning Visualization

For visualization purpose, each object is assumed to be located at the center of its respective room which can be obtained by calculating the center of the MultiPolygon of the room.

6.2.3.1 2D Visualization

As illustrated in Figure 2, in 2D visualization each room in each floor is visualized with a floor plan layer on top of a base map, followed by another layer where each object is placed in the 2D coordinate of its associated room's MultiPolygon center.

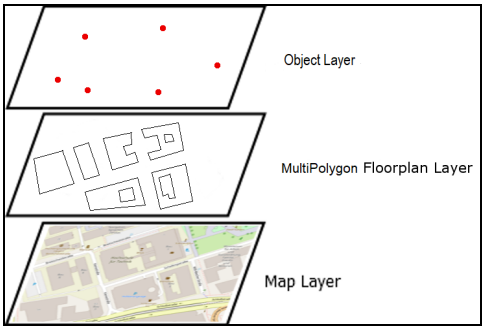


Figure 17. An illustration of a 2D visualization

6.2.3.2 3D Visualization

In 3D visualization, the building model is scaled, rotated and translated manually to get the desired transformation value on top of a 3D map (Figure 3). Since the coordinates for placing an object is defined in 2D, they need to be adjusted to fit into the 3D model by adding an altitude value into the coordinates. An object's altitude value can be defined manually according to a particular floor where the object is located.

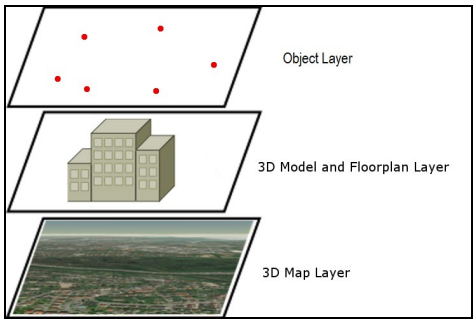


Figure 18. An illustration of a 3D visualization

6.3 Implementation

The proposed approach was implemented in one test building as a proof of concept web application. The building consists of five floors where on each floor, except in the basement, several rooms have a room temperature sensor mounted inside. The ground floor has around 40 rooms where 7 rooms have a temperature sensor inside. The first floor has around 45 rooms and also 4 temperature sensors, the second floor has around 57 rooms and 7 temperature sensors, and the third floor has around 55 rooms and 6 temperature sensors. Similar to the ground floor, all of room temperature sensors in the first, second and third floor are mounted in different rooms. Using the proposed approach, the positioning of a room temperature sensor was calculated and then visualized in a web browser. The implementation result is a web-based application which can be accessed by all building occupants.

6.3.1 Spatial and Room Temperature Data

The spatial database of the building was implemented using PostgreSQL (PostgreSQL, 2018) and extended with PostGIS (PostGIS, 2018a) to add spatial support. To complement the spatial database with some valuable non-spatial data, an existing internal data pool storing the details of a room temperature sensor is used. The details include the name of a sensor, the unit of measurement and the measurement values from time to time. Javascript and Node.js were employed for web server implementation which handles the application logic. The positioning visualization was implemented using LeafletJS (Agafonkin, 2018) for the 2D platform and CesiumJS (Cesium, 2018a) for the 3D platform.

6.3.2 The Map and 3D Model

As mentioned in section 1, the positioning visualization in this paper is based on the Living Lab web platform prototype, in which collected environmental aspects are displayed on a web-based map with georeferenced floor plans on top of it by utilizing LeafletJS. These map and floor plans are sufficient to be used further for data visualization, rather than experimenting with another 2D indoor map visualization platform.

The building 3D models consist of the ground floor, first floor, second floor and third floor. Each floor is a single 3D model that has its own interior structures including stairs (Figure 4). These 3D models were prepared in X3D format, which is not supported by CesiumJS. However, Cesium provides a tool to convert COLLADA (.dae) models or OBJ models to glTF for use with Cesium (Cesium, 2018b). Therefore, Blender (Blender Foundation, 2018) was used to convert the X3D test building model to OBJ format prior to glTF conversion.

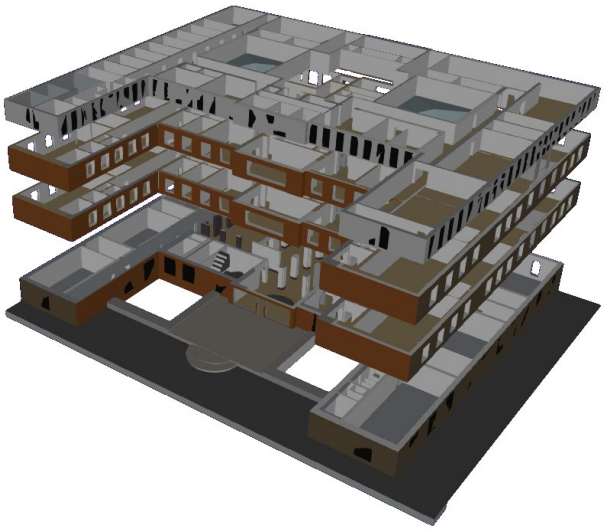


Figure 19. Overview of the 4-story test building. Each floor is a single 3D model which can be managed separately. Bottom to top: ground floor, first floor, second floor, third floor

6.3.3 Indoor Positioning Visualization

Each sensor is associated with a room's MultiPolygon center coordinate so that it can be located on a map. Obtaining the coordinate was achieved using *ST_Centroid* PostGIS function, which computes and returns the geometric center of geometry (PostGIS, 2018b). This calculation was done on the web server, and the result is pushed to the client side in GeoJSON format. The result was latitude and longitude values which are used to visualize the position of room temperature sensors. As pictured in Figure 5, small circles were drawn to mark the sensors' locations on the map. Figure 6 shows the screenshot of the web application and how the non-spatial data from the data pool is used to complement the positioning visualization. The script below shows how the latitude and longitude values are used for the positioning visualization on LeafletJS:

```
var sensorLayer = L.geoJson(null,{
        onEachFeature: onEachSensorFeature,
        pointToLayer: function (feature, latlng) {
                return L.circleMarker(latlng, sensorMarker);
```

```
        }
}).addTo(map);
```

The 3D model we used for the 3D positioning visualization consists of only four floors because the model has no basement floor. On the 3D platform, we transformed our glTF 3D model manually through several tests until the desired scale, rotation and translation values were achieved. Altitude value was set to 0 since the Cesium globe was used without a 3D terrain model in this implementation. Each floor of the building was added to the scene separately so that it could be handled independently. Figure 7 shows the whole test building with the first floor not visible to users. The script below shows how the 3D model was added and adjusted on CesiumJS:

```
entities.add({
        id: 'bau2_og2',
        position : (CENTRE_LON, CENTRE_LAT, 0),
        orientation: ROTATION_VALUE,
        model : {
                uri : 'bau_2_2.gltf',
                scale: SCALE_VALUE
        }
});
```

To visualize the room temperature sensors, sphere objects were used to mark the sensors inside the building (Figure 5 and Figure 7). These sphere objects were added separately using the same approach as was used with the 3D model. A specific rotation value is not needed in this case since it does not impact the appearance of the sphere. The values of longitude *(CENTRE_LON)* and latitude *(CENTRE_LAT)* were obtained using the same method as in 2D visualization. The altitude value for each sensor was defined manually by considering its respective floor location.

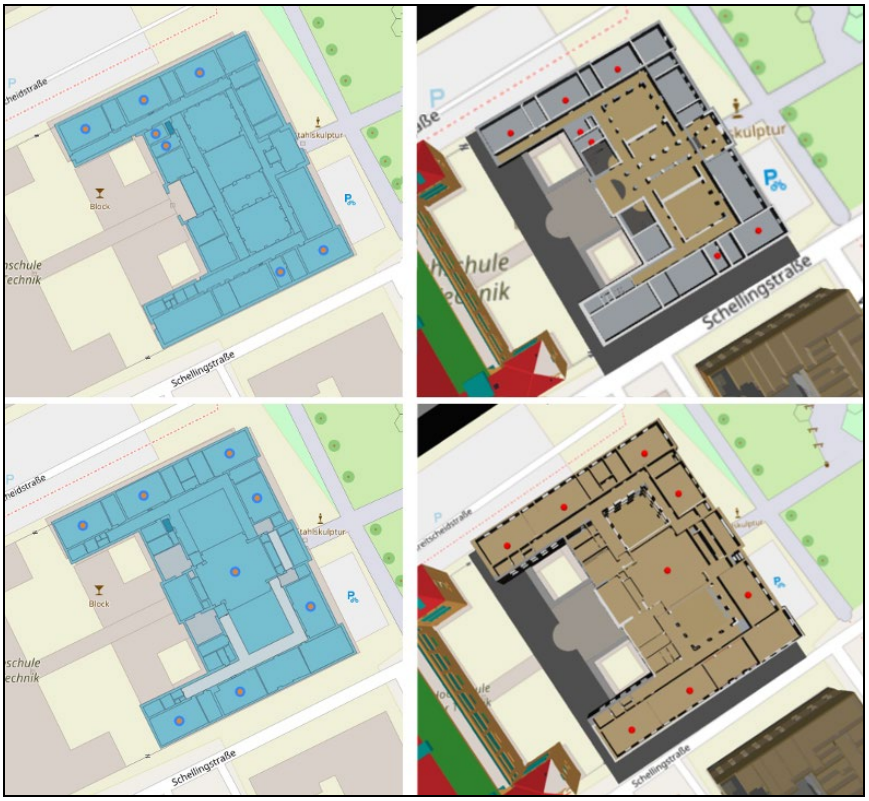


Figure 20. The sensors' positioning visualization on the ground floor (top) and second floor (bottom) in 2D (left) and 3D (right) platforms. Each circle on the map, or each sphere in the 3D model, represents a sensor's location. Base Map © OpenStreetMap contributors

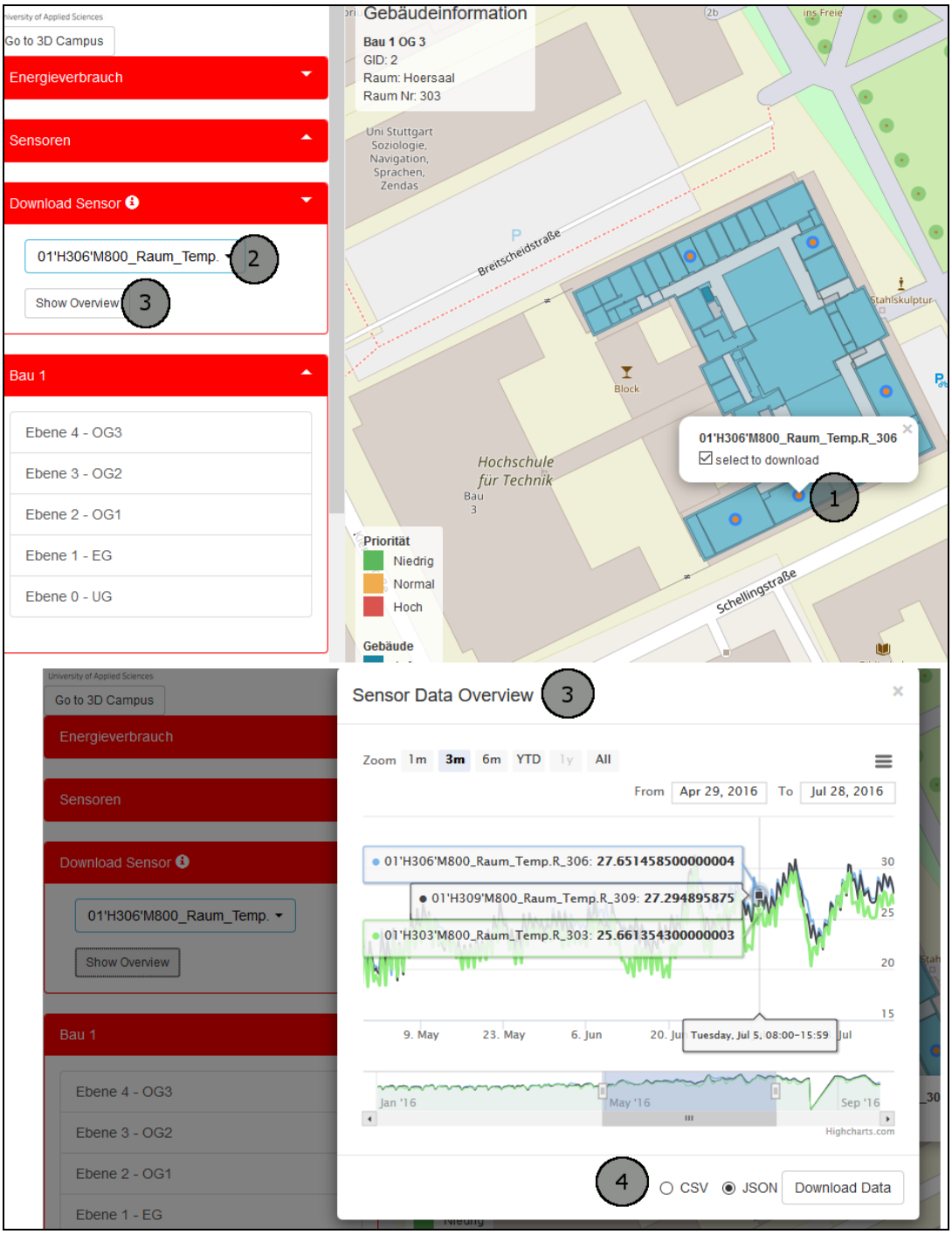


Figure 21. 2D visualization web user interface. (Top) Data download procedure. 1) Users can select sensors to download by ticking a pop-up checkbox. 2) A list of selected sensors. 3) A button to show an overview of selected sensors' data. (Bottom) Sensor data overview of selected sensors in a graph. 3) Room temperature measurements of three different rooms from 29 April 2016 until 28 July 2016. Users can see the measurement from a specific time interval by hovering the cursor on the graph. 4) Data format selection and a download button. Base Map © OpenStreetMap contributors

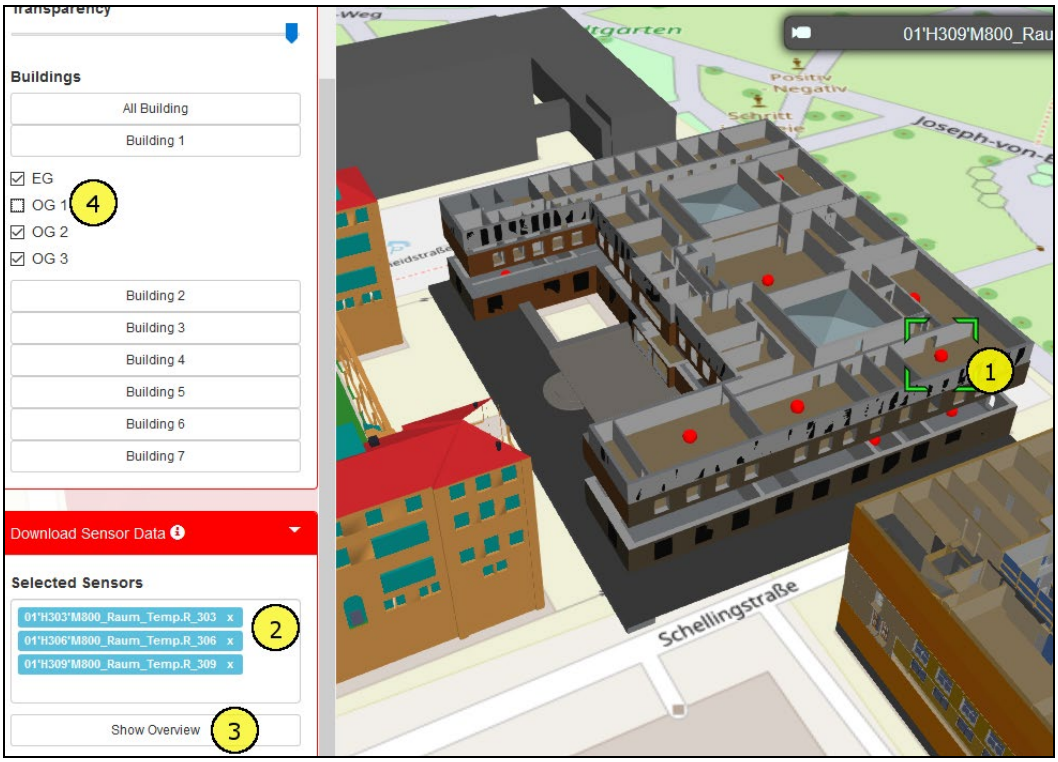


Figure 22. Web user interface and data download procedure in the 3D platform. 1) Selecting a sensor to download can be done by clicking the sensor. 2) A list of selected sensors. 3) A button to show sensor data overview before downloading the data. The data overview is the same as the one in the 2D platform. 4) Each floor of the building, including the sensors within, can be shown or hidden independently. Base Map © OpenStreetMap contributors

6.4 Results and Discussion

As expected, on both 2D and 3D platforms, the implementation showed that objects without built-in spatial attribute or not equipped with wireless technology could be correctly located in which they are placed. One downside regarding the precision of our approach is that it cannot yet obtain the exact position of an object inside a room, especially when the object is mounted to the wall or the ceiling. However, we feel strongly that it is also possible to apply the same approach to any non-moving devices in a building in order to monitor the rooms and its inventories or for another purpose which does not need highly precise positioning system, such as indoor navigation system for an emergency situation.

Since the floor plans stored in the spatial database is already georeferenced, integrating them into a map was straightforward. LeafletJS has the functionality to draw points as well as polygons based on given latitude and longitude values on top of a base map using a layer concept. By exploiting this functionality, placing a floor plan and an object, which are represented by a MultiPolygon and a point, on top of a base map can be done efficiently. Unlike the floor plans which are already georeferenced, integrating the 3D model in the visualization part was quite cumbersome since the 3D model used in the implementation was not georeferenced in advance. As the impact, obtaining the correct geographic coordinate and the correct transformation value of the 3D model must be done manually during the integration process. Georeferencing the 3D model in advanced is essential to ensure the accuracy of the building spatial data visualization which can impact the positioning visualization of the observed object. Furthermore, obtaining the correct altitude value of the observed object in order to have 3D coordinates was also a trivial process. This process can be simplified by transforming the georeference of the floor plans from 2D to 3D so that in 3D positioning calculation, the altitude value of the floor can be taken into account as well.

The system design of this paper is a loosely coupled design which separates data, positioning calculation and visualization. Hence, this system design provides flexibility in interpreting the coordinate of an object into visual information. This means each part can be developed further without affecting other existing parts. For instance, the positioning calculation method can be modified without changing the data or the visualization part, and the positioning visualization can be implemented in a mobile platform as well. Undeniably, the information

presented in the visualization and the accuracy of the positioning calculation rely heavily on the data quality in the data source and the calculation method in the application logic. To make sure the position of an object is always up to date, there should be a mechanism to update the data source every time an object is moved from one room to another. Therefore, it is crucial not only to improve the calculation method for better accuracy but also to enrich the data source and keep it up to date for a more valuable visualization.

6.5 Conclusion and Future Work

This paper presents a web-based indoor positioning visualization of static objects which are not equipped with wireless technologies, such as Bluetooth and wireless positioning system. The objects are linked to the spatial data of the building rooms where they are placed and then the coordinate of the room center is used to visualize the position of an object. To complement the positioning visualization, more details about the observed object that are related to the environmental aspects of the room in which it is placed can be visualized as well. The presented proof of concept has successfully made the location of energy-related sensors and their details available to building occupants. Thus, the platform presented in this paper has the potential to be used as an indoor energy-related issue monitoring tool. To further our research in the 3D domain, we plan to extend our approach by using CityGML Level of Detail (LoD) 4 as the 3D model so that a broader area and multiple buildings can be taken into consideration while ensuring the georeference of the 3D models at the same time.

6.6 References

Agafonkin, V., 2018. *Leaflet – a JavaScript library for interactive maps*. https://leafletjs.com/ (accessed July 15, 2018)

Blender Foundation, 2018. *Home of the Blender project – Free and Open 3D Creation Software*. https://www.blender.org (accessed July 15, 2018)

Cesium, 2018a. *Geospatial 3D Mapping and Virtual Globe Platform*. https://cesiumjs.org (accessed July 15, 2018)

Cesium, 2018b. *glTF Model Converter*. https://cesiumjs.org/convertmodel.html (accessed July 15, 2018)

Coors, V., Bartke, N., Fridrihsone, A., & Gerges, B., 2016. Using 3D building models in a research living lab for a climateneutral city campus. In: *Proceedings of the 11th Conference on Sustainable Development of Energy, Water and Environment Systems (SDEWES)*.

Jensen, P., & Gitahi, J., 2017. *Visualization of Time Series in Cesium*. http://gisstudio.hft-stuttgart.de/cesium.html (accessed July 15, 2018)

NodeJS, 2018. *Node.js*. https://nodejs.org/en/ (accessed July 15, 2018)

Ohrt, J., & Turau, V., 2013. Simple indoor routing on SVG maps. *International Conference on Indoor Positioning and Indoor Navigation* (c), pp. 1–6.

PostGIS, 2018a. *Spatial and Geographic Objects for PostgreSQL*. www.postgis.net (accessed July 15, 2018)

PostGIS, 2018b. *Spatial Relationships and Measurements*. www.postgis.net/docs/ST Centroid.html (accessed July 15, 2018)

PostgreSQL, 2018. *PostgreSQL: The world's most advanced open source database*. www.postgresql.org (accessed July 15, 2018)

Udas, E., Wölk, M., & Wilmking, M., 2018. The "Carbon-neutral University" a study from Germany. *International Journal of Sustainability in Higher Education 19(1)*, pp. 130–145.

Wijewardena, G. G., Vasardani, M., & Winter, S., 2016. Indoor Localization and Navigation Independent of Sensor Based Technologies. *Proceedings of the Eighth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pp. 19–26.

6.7 Acknowledgements

This work is funded by the Baden-Wuerttemberg Ministry of Science, Research and the Arts of Germany.

7 SMARTLOCATE – INDOOR LOCALIZATION WITH BLUETOOTH BEACONS AND SMARTPHONE SENSORS

J. Bauer [a, *], J. Bakakeu [a], M. Hopfengärtner [a], J. Bürner [a], T. Braun [a], F. Schäfer [a], M. Wittmann [a], A. Fehrle [a], B. Maußner [a], T. Lechler [a], M. Meiners [a], C. Konrad [a], J. Franke [a]

[a] Institute for Factory Automation and Production Systems, Technical Faculty, Friedrich-Alexander University Erlangen-Nuremberg, Egerlandstr. 7-9, D-91058 Erlangen (Germany), jochen.bauer@faps.fau.de

KEY WORDS: Ambient Assisted Living, Android, Beacon, Localization, IoT, Smart Home

ABSTRACT:
There is a lack of cost-effective and easily exchangeable system solutions for position determination in indoor areas. The article describes a solution system for indoor position determination. The approach is based on beacons and inertial sensors of smartphones. An accuracy of 50 cm is achieved by a clever combination of different methods. The combined methods use the methods of angulation, map matching, fingerprinting, cell-ID approach and a particle filter. The investment costs for a 120 m^2 apartment are around 150 euros, excluding the necessary Android smartphone.

7.1 Introduction

Numerous applications in the smart home sector require localization of objects or persons. Often, once a person has been localized, you want to identify the person as well or at least make the person distinguishable from other people in the environment. Through localization and connected identification, existing systems often only become applicable for multi-person households or benefit significantly from this functionality, e.g. in cross-room sound or surface heating systems (Hein et al. 2017, T. Braun et al. 2016). In addition, localization in the field of Ambient Assisted Living (AAL) plays an important role: fall detection software or applications that generate health-related information (Bauer et al. 2016, Lutze & Waldhör 2015) and want to display this information exactly at the place where the user is currently located. Researching groups are focusing on such applications in the context of the Internet of Things (IoT) due to the increased number of connected devices (Bauer et al., 2014).

Current fall detection systems face the challenge of reliably avoiding false alarms. Causes for such false alarms are, for example, when the mobile phone is exposed to excessive movement within a pocket or simply falls to the ground (GFDK Gesellschaft für digitale Kaufberatung mbH, 2017). If it is possible to detect that a fall has occurred in the sofa area, this information can be included in the result evaluation. This shows that the challenge so far has been to differentiate between normal movements such as bending, lying down or kneeling and actual falls (Parikh, 2010). The challenge of fall detection must therefore not be reduced to the localization topic, although this is of course helpful in this context. Another imaginable application for person recognition is the intelligent control of surface heating segments in the room. On the one hand, these provide a pleasant feeling of warmth, as they heat around the people and not the air in the room. On the other hand, energy can be saved, as less energy is lost through windows during ventilation. Energy can also be saved by regulating the individual segments accordingly, as heating is only provided in the immediate vicinity of people.

The Global Positioning System (GPS) is established for outdoor positioning – numerous mobile phones offer GPS-based navigation systems, such as the Google Maps app. Under good conditions, the achievable accuracy without correction is about 5 to 20 meters (GPS – Genauigkeit & Einschränkungen, 2017). The fact that most common GPS receivers also receive Wide Area Augmentation (WAAS) or European Geostationary Navigation Overlay Service (EGNOS) correction signals improves the actual achievable accuracy to 1 to 3 meters. However, optimum reception conditions are often not achieved: the GPS radio waves propagating quasi-optically due to the very high frequency should ideally have visual contact with as many satellites as possible. Errors in the position calculation are caused by excessive attenuation, reflection of individual or all GPS signals and by the reception of too few satellites.

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 61–70,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

However, GPS is not available for indoor use and there is a lack of sufficiently accurate, cost-effective, easily upgradable localization options. Here, there are electromagnetic, acoustic and optical methods, which are used to determine the position. These methods vary in the respective acquisition and installation costs. In 2015, 65% of Germans used a smartphone and 40% a tablet (Lutter et al., 2015). Thus it is helpful to consider existing possibilities of such devices when designing an indoor localization system. Android and iOS are common operating systems for mobile devices, i.e. smartphones, smartwatches and tablets. According to current statistics, Android's market share is 85% among smartphone operating systems, while iOS has a share of 14.7% (Statista GmbH, 2017). Both related vendors, Google and Apple, offer beacons in their product portfolios, too. In the context of this paper, four identical beacons of the second generation from Beaconinside (Beaconinside GmbH, 2017) are used (see Figure 23).

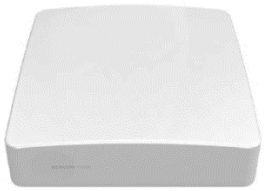


Figure 23. Beacon of the second generation of the company Beaconinside

These beacons send a Bluetooth signal and complement the standard inertial sensors in the devices. As an alternative to the battery-intensive Wi-Fi approach, localization in (Chandel et al., 2016) was performed by combining access to the inertial sensors of a smartphone with the use of beacons. This fusion, which was realized within a positioning system InLoc by the authors, turned out to be very promising. An analysis of the accuracy and performance of a comparable methodology is investigated in (Dabove et al., 2015). Indoor positioning is becoming more and more attractive due to beacons and more accurate inertial sensors in smartphones from the cost-benefit aspect (Borrmann et al., 2015). Beacons periodically transmit a Bluetooth signal with adjustable period length and transmission power. The minimum period duration is set to 100 milliseconds (ms), while the transmission power is set to –3 decibel-milliwatts (dBm).

Since the introduction (Android versions comparison, 2017) of Android 4.3 with the name Jelly Bean on 24.07.2013 Android devices support the Bluetooth Low Energy Technology (BLE) for low-energy communication: Device A sends a BLE signal and an identification number (ID). Device B receives this signal in a certain signal strength, the received signal strength intensity (RSSI), and assigns it to the beacon belonging to the ID. On the basis of this signal strength and linked calculations, the distance and thus the position of the communication participants can be estimated.

Now that the topic of localization has been sufficiently addressed, general and promising strategies for position determination are examined in detail. Subsequently, a decision is made as to which of these strategies will be incorporated into a solution concept for a cost-effective and easily retrofittable system for determining the position of people indoors. The paper concludes with an analysis of the performance of the described approach.

7.2 Approaches for Position Detection

An existing floor plan is helpful as a basis for determining the position. This floor plan can be provided by third parties, e.g. from existing architectural drawings, or it can be digitally generated and derived by mobile devices, e.g. a vacuum cleaner robot. In the following, a floor plan of an apartment is presented (see Figure 24), which played an important role in the implementation of the concept.

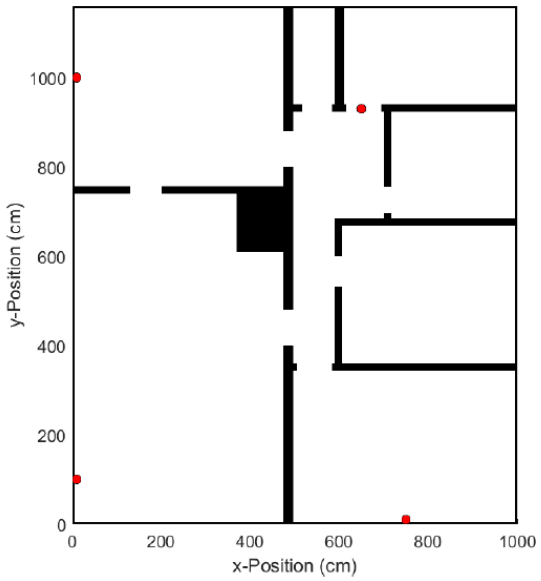


Figure 24. Floor plan of the apartment in which the proof of concept was carried out

Four beacons are placed in the 116 m^2 apartment. This is followed by the already described exchange of information between the transmitter and the receiver of the Bluetooth signal. In the specific application, a mobile phone with the Android operating system 6.0 Marshmallow (market launch 05.10.2015) (Brodersen, 2017) serves as a receiver for the Bluetooth signals of the beacons. It thus determines the received signal strength, the RSSI value of each beacon and, based on these RSSI values, the position of the mobile phone in the room or apartment.

The mobile phone contains numerous sensors that can be used for a possible indoor localization. As a basis for the relevant calculations, different coordinate systems can be used: the space coordinate system (R-KOS), the earth coordinate system (E-KOS), the body coordinate system (K-KOS) and the device coordinate system (G-KOS) (see Figure 25).

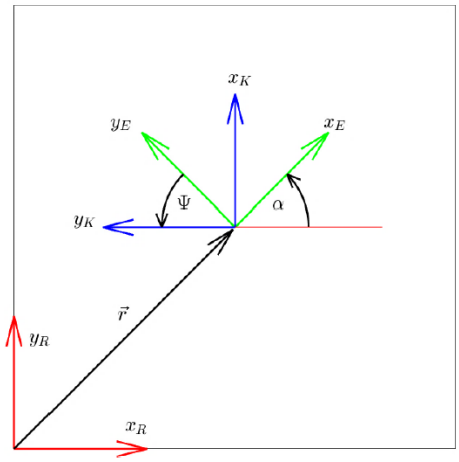


Figure 25. Coordinate systems relevant for a smartphone and their interaction, here the R-KOS, E-KOS, K-KOS

With the R-KOS, the x and y axes are on the plane of the building floor plan and the z axis is perpendicular to it. In contrast to the R-KOS, the E-KOS is based on the earth's surface. Relative to the R-KOS, there is a translational displacement around the position vector $\vec{r}$ and a rotation of the z-axis around the angle of rotation α. With the K-KOS the origin is at the current position of the person, x- and y-plane are tangential to the earth's surface, with the y-axis pointing in the current direction of motion of the person. Relative to the E-KOS, the z-axis is rotated by the angle Ψ, the time-dependent yaw angle, which represents a rotation of the direction of movement to the geomagnetic north direction. The G-KOS is permanently connected to the Android device and

therefore follows the movement of the smartphone (see Figure 26). The coordinates can be represented as vectors and can be transformed via a matrix.

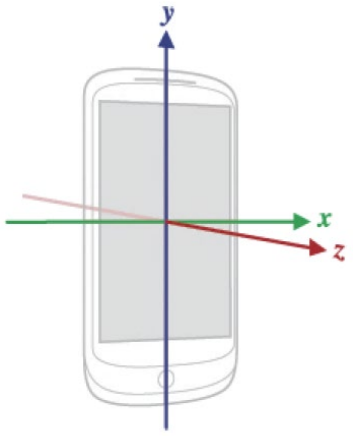


Figure 26. G-KOS is a device-associated coordinate system with a fixed connection to the terminal

After explaining the background to the sensors and physical conditions, the focus is now on the helpful calculation methods for position determination. The basic localization methods are methods that calculate the position of the unknown point on the basis of known points, such as angulation or lateration. Triangulation is widely used in cartography. If the length of one side and the angles of a triangle are known, the remaining side lengths can be determined by trigonometric formulas. In trilateration (see Figure 27), it is not the angles that are known, but the actual distances. Both approaches therefore allow the position of the unknown object to be determined. Angles can be measured much more accurately than distances. However, the calculation procedure for determining the position of the searched point is more complex. Consequently, angle-based methods are usually used for long distances and distance-based methods if the position often has to be determined anew. Multilateration therefore offers a technically simpler implementation option for position determination. For a determination in two-dimensional space using the multilateration method, three known points are necessary to determine the position of the fourth point. The solution of the nonlinear equation system leads to the position calculation of the searched point.

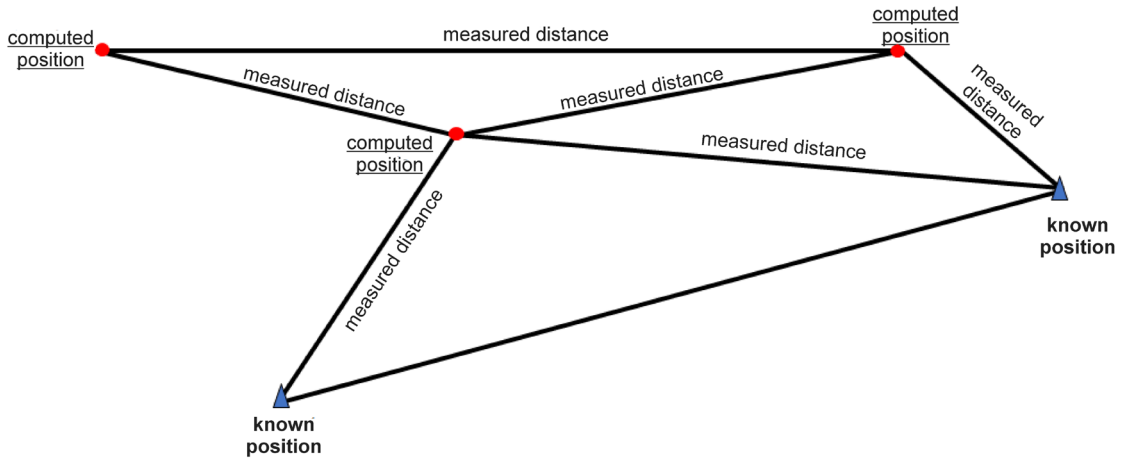


Figure 27. Trilateration principle for position determination based on less known starting points (Geographic Information Systems)

The distance between the transmitter and receiver object can be calculated from an RSSI value received by the transmitter i. The distance between the object and the transmitter resulting from the spatial components is calculated:

$$\tilde{d}_i(x(t),y(t)) = \sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} \qquad (1)$$

In order to calculate the position of the object, a cost function can be set up. This takes into account the calculated distances of all n transmitters. The cost function is then minimized to determine the position based on the RSSI values:

$$C(x(t), y(t)) = \sum_{i=1}^{n} (\tilde{d}_i(x(t), y(t)) - d_i(RSSI_i(t)))^2 = \sum_{i=1}^{n} (f_i(x(t), y(t), RSSI_i(t))^2 \qquad (2)$$

with

$$f_i(x(t), y(t), RSSI_i(t)) = \sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} - d_i(RSSI_i(t)) \qquad (3)$$

In the above equation, C (x(t), y(t)) denotes the position-dependent cost function at time t, which has a minimum at the current position of the object.

Within the framework of lateration, distances between two points have to be determined. There are various methods for measuring distances. The “time of arrival” method measures the transit time of a transmitted signal until it reaches its destination. Alternatively, two signals can be sent simultaneously and the time difference on arrival is determined. This method is called “time difference of arrival”. In addition to Bluetooth, there are other signals that can be used for distance measurement, such as Wi-Fi signal strength, infrared or ultrasonic signals. In addition, there are optical and acoustic methods. Each method has advantages and disadvantages, especially with regard to retrofitting and installation costs. In addition, optical localization systems are viewed critically by residents. Acceptance problems (Wolfangel, 2017) exist, for example, with cameras in the home. Against the background of the increasing networking of the residential environment, such systems can result in security gaps which are deliberately abused by crackers. Possible consequences of an external attack can be that residents are spied on or disturbances are caused. The protection of privacy in the personal living environment represents a basic need (Klebsch, 2017) for the user and, together with the topics of information security and user-friendliness, is essential for a sustainable solution concept to emerge.

In indoor localization, one is confronted with the fact that the position of the tenant changes dynamically. For this purpose, there are different procedures to take this general condition into account. The Cell ID method, also known as the proximity method, uses a specific RSSI threshold to assign a circular field to each beacon. The system therefore recognizes whether the person is in one of the circles or not (see Figure 28). The Cell ID method, with a target accuracy of 50 cm, usually ignores many areas of the residential environment.

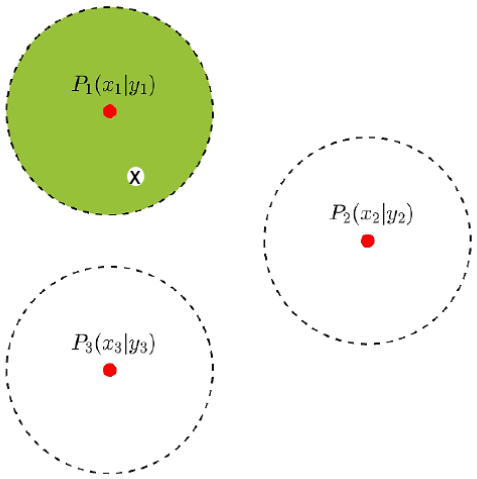


Figure 28. Proximity/Cell ID method for determining position on the basis of a threshold value based on the received signal strength. Location X is assigned to P1

The fingerprinting method can be used in addition to the proximity method. This method is divided into an off- and an online phase. In the offline phase, the resident generates various measuring points and stores the profile of the received signal strengths at the respective measuring point. The result is a map of measuring points and an RSSI value range for each measuring point. In the online phase, the received signal strengths can then be assigned to a measuring point and thus to a position. The fingerprinting procedure normally delivers very good results. However, it must be taken into account that the offline phase must be started again if a change is made.

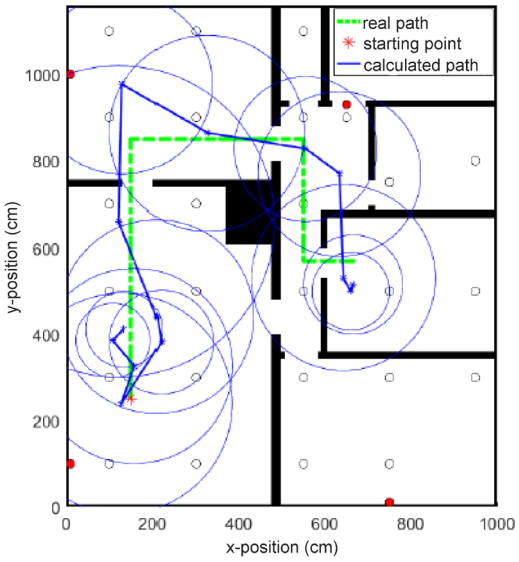


Figure 29. Fingerprinting method with predetermined measurement points for better allocation based on received signal strength

Also relevant is the dead reckoning method. Here only the inertial sensor technology of the mobile phone is used – more precisely – the mobile phone can determine in which direction and with which acceleration and thus with which speed it is moved. This data can then be merged to follow the path of the occupant on the existing floor plan. The advantage here is that no additional investment is necessary. However, the procedure also has two serious disadvantages: firstly, a fixed starting point is required and secondly, an error that occurs once propagates. For this reason, the procedure is a useful addition to other procedures, for example, so that these procedures in turn enable dead reckoning to detect a starting point or to provide for the detection of accumulated errors.

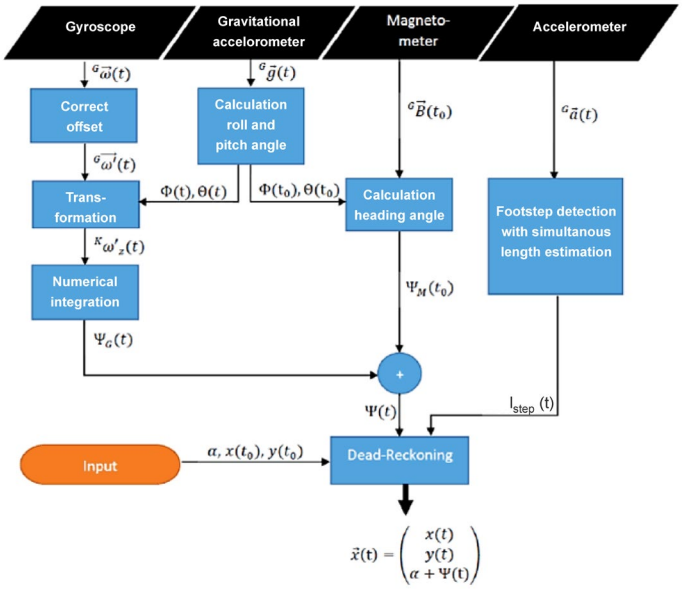


Figure 30. Flowchart of the dead-reckoning method

In order to be able to combine relevant procedures in a meaningful way, the object position supplied by the respective procedure must be compared and evaluated. The particle filter, also known as the sequential Monte Carlo method (SMC), is suitable for this purpose. The particle filter scatters particles in the state space and determines the most probable system state. The SMC method is particularly suitable here because it can be applied very flexibly.

7.3 SmartLocate Approach

Within the scope of the objective, a cost-effective, easy to retrofit and robust indoor navigation system was designed: SmartLocate, an Android based app. The developed algorithm for position determination is based on a combination of beacons, a given floor plan in combination with an Android smartphone, fingerprinting, inertial sensors and map matching. Based on the descriptions above, it was decided to combine fingerprinting, map matching, proximity and dead reckoning with a particle filter. With the particle filter, it is helpful to carry out the propagation step permanently, i.e. whenever there is a change of orientation or a step. The estimation step and the resampling of the particles are triggered when

- a step and thus possibly a wall is crossed,
- a new fingerprinting position value (approximately every 3 sec) is available,
- a new proximity position value (about every 4 sec) is present.

7.4 Results

SmartLocate achieves the targeted accuracy of around 50 cm by skillfully combining the methods (see Figure 31). In addition to the accuracy, both the time required to determine the position and the robustness of SmartLocate appear to be sufficient. After just a few meters or seconds, the person can be located in the home, as can be seen from the point clouds, which are becoming closer and closer (see Figure 32).

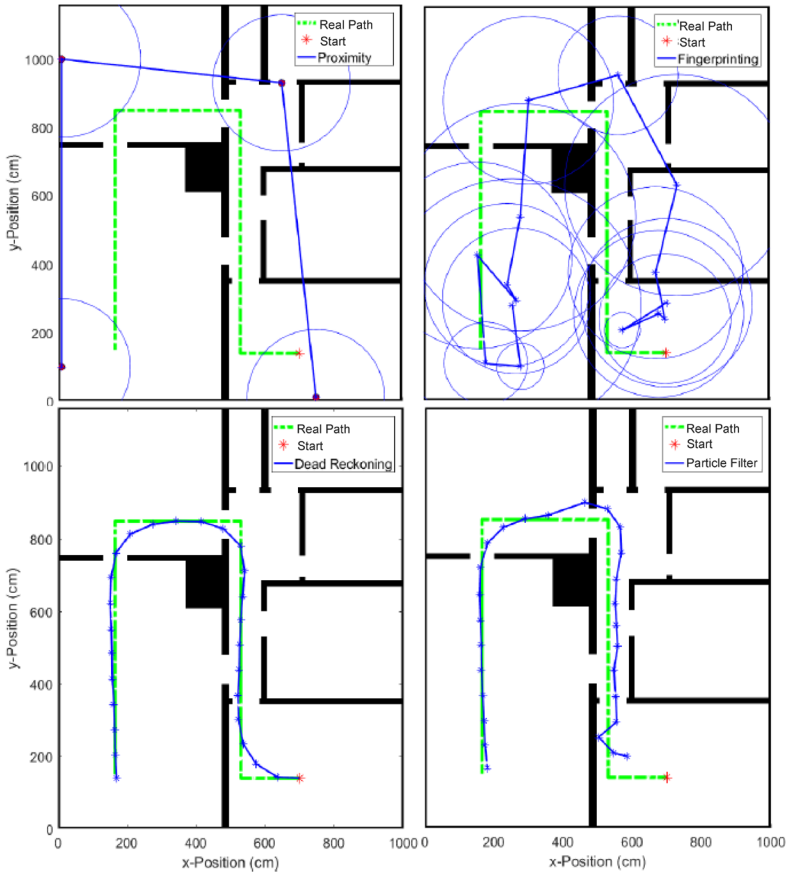


Figure 31. Isolated view of the methods used in SmartLocate – proximity, fingerprinting, dead reckoning and particle filter

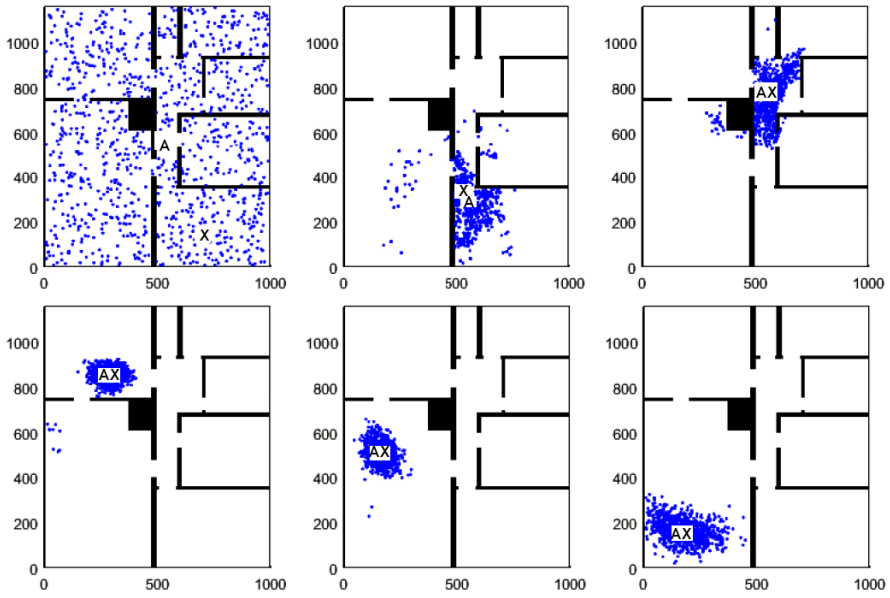


Figure 32. Mode of operation of the particle filter, starting from a holistically scattered particle cloud including position estimation (A) up to the narrowing close to the position of the target object (X). After a few steps, there is a congruence between A and X, known as AX

For the test scenario shown (see Figure 33), an accuracy of about 50 cm could be achieved after 10 steps. This accuracy is based on the evaluation of six test scenarios (see Figure 34). Such accuracy is probably sufficient for personal localization, especially when considering that beacons are often placed in less visible areas of the living environment, such as the ceiling, for aesthetic reasons.

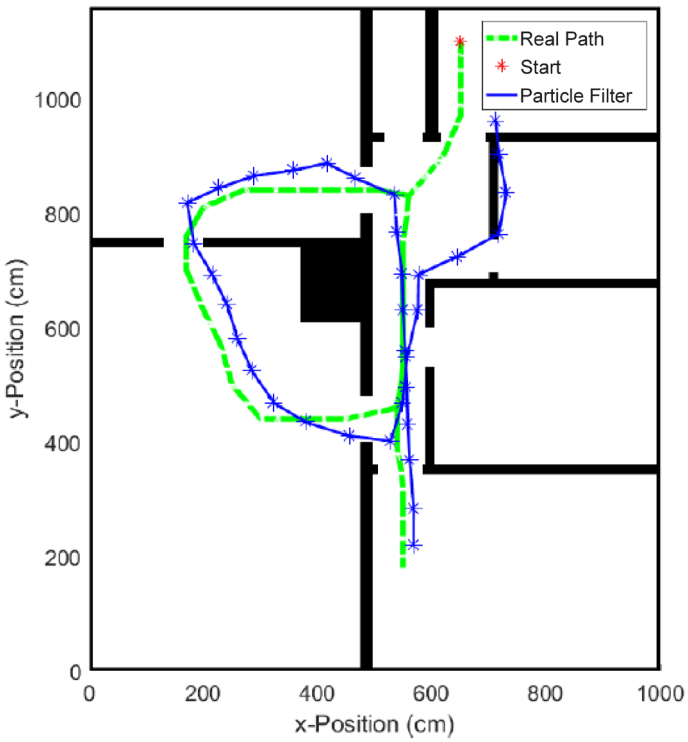


Figure 33. The test scenario for determining the average position error

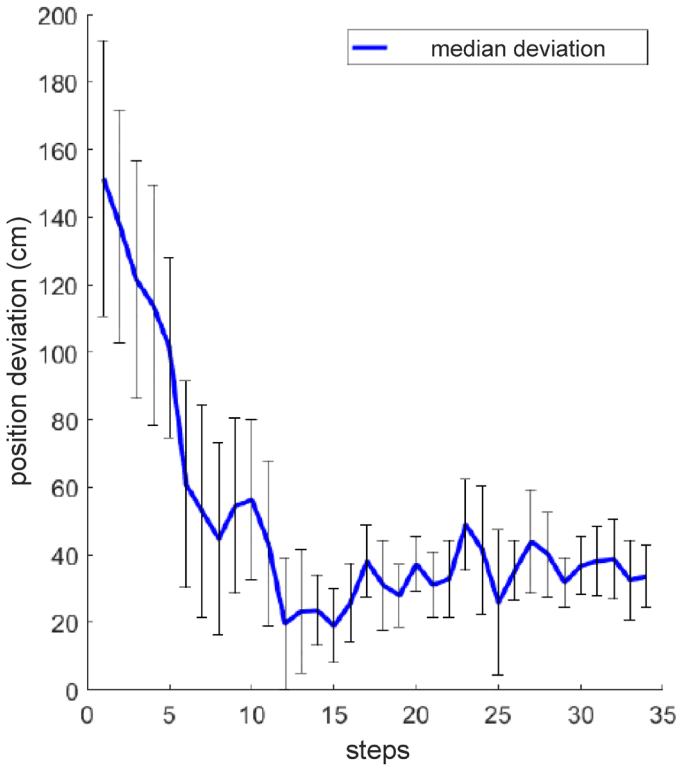


Figure 34. Mean position error for six selected test scenarios

SmartLocate will be further optimized in the future. In the described use case a mobile phone was used and served as receiver of the BLE signal. However, the mobile phone is often stored and charged at home. The next step is to test the use of a smartwatch as a BLE receiver.

It also seems helpful to pass on the determined position of the person or another IoT object to any existing smart home system and its visualization. In order to take this approach into account, a connection to the proven middleware "Eclipse SmartHome" or "openHAB" is currently being designed. A derivation of the floor plan data from a vacuum cleaner robot is also in progress, so that the prerequisite for the application of the solution system can also be generated automatically.

7.5 References

Android versions comparison, 2017. http://socialcompare.com/en/comparison/android-versions-comparison (accessed June 17, 2017)

Bauer, J., Kettschau, A.-K., Michl, M., Bürner, J., & Franke, J., 2014. Die intelligente Wohnung als Baustein im Internet der Dinge: Potenzialanalyse und Konzept einer domänenübergreifenden Lösung. In: Robert Weidner (Ed.): *Technische Unterstützungssysteme, die die Menschen wirklich wollen - erste transdisziplinäre Konferenz zum Thema*. Univ. der Bundeswehr Hamburg, Laboratorium Fertigungstechnik, pp. 298–307.

Bauer, J., Michl, M., Kettschau, A.-K., Wiebe, S., Vierow, V., Ge, M., & Franke, J., 2016. HealthRec - Empfehlungssystem zur Behandlung des metabolischen Syndroms im Smart Home. In: *Zukunft Lebensräume. Gesundheit, Selbstständigkeit und Komfort im demografischen Wandel.* 20. und 21.4.2016, Frankfurt am Main. VDE Verlag, Berlin, Offenbach, pp. 1–6.

Beaconinside GmbH, 2017. *Beaconinside Beacon (2nd Gen).* http://developers.beaconinside.com/docs/beaconinside-beacon (accessed June 17, 2017)

Borrmann, A., König, M., Koch, C., & Beetz, J., 2015. Building Information Modeling - Technologische Grundlagen und industrielle Praxis. Springer-Verlag, Berlin Heidelberg New York.

Braun, T., Bürner, J., Michl, M., Schaller, L., Böhm, R., & Franke, J., 2016. Innovative flexible heating system by the use of additive plasma coating technology: Heating where heat is needed by additive metallization of furniture and walls. In: *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 278–283.

Brodersen, B., 2017. *Die Updates auf Android 6.0 Marshmallow im Überblick. COMPUTEC MEDIA GmbH.* http://www.areamobile.de/specials/34609-die-updates-auf-android-6-0-marshmallow-im-ueberblick (accessed June 17, 2017)

Chandel, V., Ahmed, N., Arora, S., & Ghose, A., 2016. InLoc: An end-to-end robust indoor localization and routing solution using mobile phones and BLE beacons. In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8.

Dabove, P., Ghinamo, G., & Lingua, A. M., 2015. Inertial sensors for smartphones navigation. In: *SpringerPlus* 4, p. 834. DOI: 10.1186/s40064-015-1572-8.

GFDK Gesellschaft für digitale Kaufberatung mbH, 2017. *Seniorenhandy mit automatischer Sturzerkennung - so hilft es im Ernstfall.* http://www.seniorenhandy.com/automatische-sturzerkennung/ (accessed June 16, 2017)

GPS – Genauigkeit & Einschränkungen, 2017. Ingenieurbüro Martin Nathansen. http://gpso.de/technik/gpsgenau.html (accessed June 17, 2017)

Hein, M., Stöber, R., Meiler, M., Schaller, D., Zehle, R., Fischerauer, G., et al., 2017. Sensor solutions for an energy-efficient and user-centered heating system. In: *J. Sens. Sens. Syst.* 6 (1), pp. 27–35. DOI: 10.5194/jsss-6-27-2017.

Klebsch, W., 2017. VDE-Positionspapier "Smart Living". VDE Verband der Elektrotechnik Elektronik Informationstechnik e.V. Frankfurt am Main.

Lutter, T., Pentsi, A., Poguntke, M., Böhm, K., & Esser, R., 2015. *Zukunft der Consumer Electronics – 2015. Marktentwicklung, Schlüsseltrends, Konsumentenverhalten, Mediennutzung, Neue Technologien. With assistance of Christopher-Marcel Meinecke.* Bitkom e.V. Berlin. https://www.bitkom.org/Bitkom/Publikationen/Studie-Zukunft-der-Consumer-Electronics-2015.html (accessed November 19, 2018)

Lutze, R., & Waldhör, K., 2015. A Smartwatch Software Architecture for Health Hazard Handling for Elderly People. In: *2015 International Conference on Healthcare Informatics*, pp. 356–361.

Parikh, T., 2010. Sturzerkennung durch Rotationsmessung. PVA-MV AG.

Statista GmbH, 2017. Prognose zu den Marktanteilen der Betriebssysteme am Absatz von Smartphones weltweit in den Jahren 2017 und 2021. https://de.statista.com/statistik/daten/studie/182363/umfrage/prognostizierte-marktanteile-bei-smartphone-betriebssystemen/ (accessed June 16, 2017)

Wolfangel, E., 2017. *Vernetztes Leben. Wenn das Haus für uns denkt.* Spektrum der Wissenschaft Verlagsgesellschaft mbH. http://www.spektrum.de/news/smart-home-wenn-haeuser-fuer- uns-denken/1256592 (accessed June 17, 2017)

8 EVALUATION AND IMPLEMENTATION OF A WEB-BASED 2D/3D VISUALIZATION FOR SMART BUILDING CONTROL – STATE OF THE ART AND CHALLENGES

M. P. Jensen [a, *], D. Uckelmann [a], V. Coors [a]

[a] Department of Geomatics, Computer Science and Mathematics, University of Applied Sciences Stuttgart
Schellingstraße 24, D-70174 Stuttgart (Germany),
(62jema1mpg, dieter.uckelmann, volker.coors)@hft-stuttgart.de

KEY WORDS: Visualization, 3D, 2D, GIS, Web Mapping, Web Application

ABSTRACT:
Smart devices are becoming more prominent and are being integrated deeply in daily life. Smart buildings are benefiting from an increased number of specialized smart devices, which may be connected through corresponding network infrastructures and frameworks. With increasing numbers of sensors, more and more data is generated. To assist consumers in understanding and managing increasing amounts of data new means of visualization must be considered. 2D and 3D visualizations of sensors and sensor data being matched to corresponding locations would enable consumers to filter data according to their known perception of their environment. Geospatial Information Systems (GIS), for example, have been used for years to gather spatial data and visualize different sets of data to enable the consumer to assess the data faster and provide information with a higher precision. To monitor the status of buildings, sensors must be attached to physical objects, which puts them in a direct reference to locations and thus can be handled like spatial data. Different spatial visualization and control approaches for smart buildings are evaluated in this paper. A good visualization should enable consumers to control smart buildings easier and assess security, management or environmental effects quicker. A new web-based approach combining 2D and 3D visualization to display and control smart devices inside buildings is introduced. The proposed application should offer the consumer an intuitive way to navigate quickly through buildings while monitoring and controlling the smart devices inside. The results of this paper provide an outline for the implementation of a web-based application considering aspects of usability, performance and effectiveness.

8.1 Introduction

Today, millions of devices are equipped with a variety of sensors to gather data about position, orientation, temperature, humidity and others to use the information to provide comfort to the user. A report estimated the number of connected devices to roughly 20 billion by 2020. The report also states that the consumer segment will still be the biggest percentage of connected devices, but businesses continue to invest willingly more in IoT applications (Gartner Inc, 2018).

smart homes are becoming more prominent and the need to access information everywhere increases the importance of accessibility and mobility of software products. A recent study shows that consumers of the age 16 to 35 are extremely interested in the current development and the future of smart devices and smart homes. With this emerging market the need for efficient ways of data analysis is increasing importance of artificial intelligence and machine learning (GfK, 2018).

For smart cities, 3D city models to transfer information gathered by devices and sensors to allow a quicker insight into the data are already well established. GIS combine the ability to process spatial information and create a link between data that can by tied to the real world. To transfer the information, GIS are using maps for this process, which allows an intuitive way for navigation and orientation paired with the evaluation of data display (Semmo et al., 2012). These methods and systems can be connected to other streams of information. For example, shadow casting of a building for urban planning or energy efficiency.

Increasing numbers of devices and a deeper integration into the life of the consumer also required a better communication between each of the devices. The key is to know the exact needs of the consumer to be able to

* Corresponding author.

D. Uckelmann (Ed.): Proceedings of the 1st Conference on Smart Public Buildings, pp. 71–80,
19-20 October 2018, University of Applied Sciences Stuttgart, Germany.
ISBN: 978-3-940670-66-3

react and adapt to these needs. This development is centred around the perspective of a single main user in smart homes. This paper focusses on the perspective of a smart building, where the needs of multiple different users or user groups need to be considered. In this scenario users should be able to manage a smart building in a natural way. To get a deeper understanding of what is happening, buildings must be equipped with sensors and connected with smart devices that provide information about current conditions and help to manage building more efficiently. A by-product of the high number of devices and data allows to perceive views of different aspects that do not directly affect the building, but the surroundings and it enables to enrich information with data from external sources the create opportunities for new ways of visualization and information consumption (Meehan, 2018). Examples of this synergy could be to locate free parking lots close to a building of interest or emergency scenarios and disaster management. This is only possible with software platforms that allow devices to communicate and connect to a controlled central service to enable interaction with devices locally or remotely. openHAB (openHAB, 2018) for example is an open source application based on the Eclipse SmartHome (Eclipse Foundation, 2018) IoT framework. It is designed to provide a platform for Smart Home users that is technology agnostic and integrates different devices and systems into a central solution. Public buildings in most cases are bigger than private homes, form a more complex structure and service more occupants. To serve the needs of the occupants in the building the management tools for the systems should be fast and natural. As described before this papers focus is targeting public buildings and evaluates ways to provide an authorized user with a tool that combines 2D and 3D visualization to access data from smart devices and to control a public building. In this paper openHAB is used as a solid and open foundation to create a base infrastructure for the sensors and smart devices and is extended by GIS technologies to create and manage spatial data and enrich the information. Latest web technologies can be used to create a natural and mobile visualization toolkit to control a Smart Public Building.

8.2 Related Work

The openHAB community provides different graphical user interfaces (GUI) to access, manage and control sensors and smart devices in smart homes. From a smart home perspective, the available user interfaces allow an easy administration of the openHAB system installation. To operate smart devices the user interfaces are used either mobile or on wall mounted tablets and provide a dashboard overview to create and manage the devices. Figure 1 shows typical GUI views which can optionally be used in openHAB.

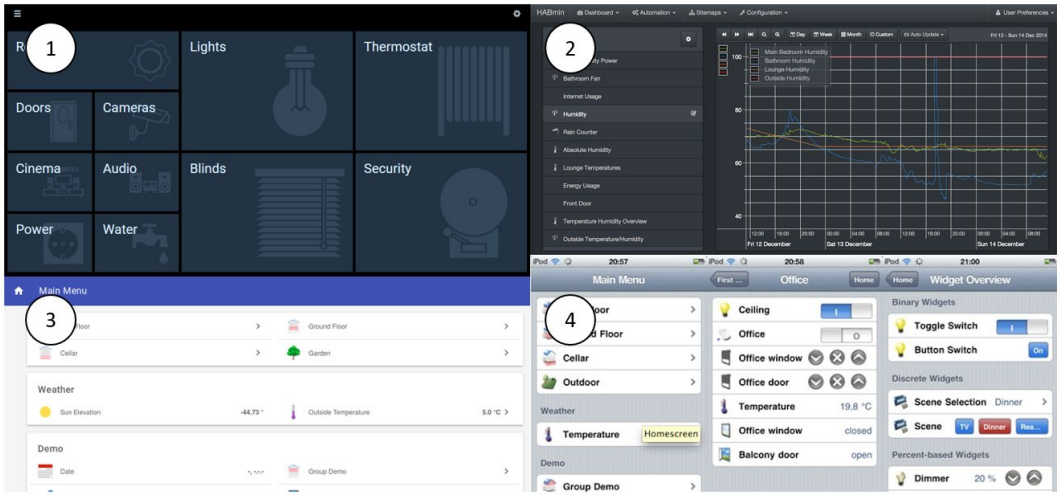


Figure 35. openHAB available GUIs. 1) HABPanel. 2) HABmin. 3) BasicUI. 4) ClassicUI. Based on openHAB documentation

A Proof of Concept (Schaus, 2018) has shown that a 3D visualization and a further integration into the HABPanel is possible. This concept used an interior design software to produce the 3D models of a building floor including 3D objects that represent smart devices. The device objects can be linked to openHAB and the user can control the properties of an item. The modelling of a private home is much easier because the building is known from the inside. For a public building, the modelling process is much more complex. Creating a 3D

model for the whole building or even several buildings and the interior is a time-consuming task, as prior research has shown.

To support the maintenance of fire safety installations of a factory building for example, floor plans have been used to visualize the locations of fire safety doors and vents. This kind of 2D visualization helps to assess the status of an installation faster and to react to high priority measures with a quicker response. A prototype was developed as a web application that enabled the user to avoid long lists of printed of information about parts that must be maintained and quickly assess the status and position of parts inside the factory buildings, to navigate and resolve the issues (Jensen, 2015). Figure 2 shows the resulting web application for a facility management use case.

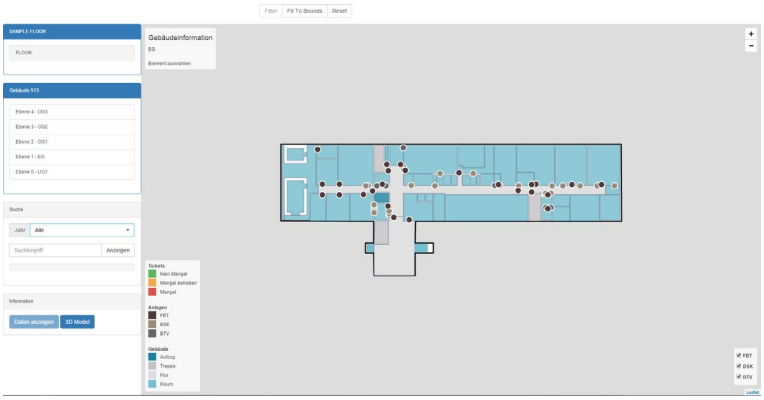


Figure 36. Prototype for facility management with spatial data and web mapping (Jensen, 2015)

Continued by the University of Applied Science Stuttgart (Hochschule für Technik Stuttgart, 2018) the application was extended to visualize environmental and safety related issues occurring in the building, as well as the visualization of location and status of sensors for real time measurement. The sensor data was provided by the Sensor Observation Service, implemented by 52° North (52° North, 2018) compliant to specification of the Open Geospatial Consortium (OGC, 2018). To visualize the data and floor plans, a web mapping library was used. Leaflet.js (Agafonkin, 2018) is a JavaScript based, mobile-friendly and light-weight web mapping library and is used to visualize the data. The spatial information is delivered by a web service, that returns the data as GeoJSON (GeoJSON, 2018). GeoJSON provides a structure to transfer data to represent spatial information and specifies the geometry types and coordinates, the used coordinate system and properties that can describe additional information about the geometry. The data contains the necessary information to display floor plans and sensor or device locations on a web map. Figure 3 shows the public web application of the virtual HFT campus.

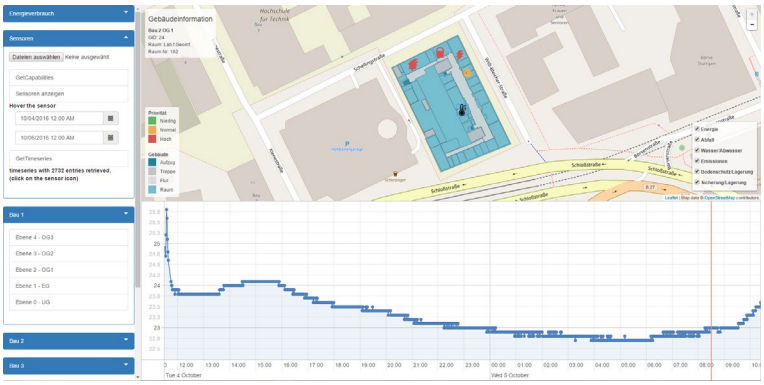


Figure 37. Campus HFT prototype with temperature sensor integration.
Base Map © OpenStreetMap contributors (OpenStreetMap Foundation, 2018)

A project group in the Masters' course Photogrammetry and Geoinformatics at the University of Applied Science Stuttgart worked on a web application to visualize time series data gathered by temperature sensors

installed in the buildings of the department. The project was mainly focused on the process and requirements for 3D visualization and the linking of sensor data and spatial locations. The application did not allow the interaction with the sensor in the real world, the purpose is to monitor time series of temperature data. A virtual globe was used to display 3D models of the HFT buildings (Gitahi & Jensen, 2018). Figure 4 shows the resulting web application displaying a 3D model of a single HFT building.

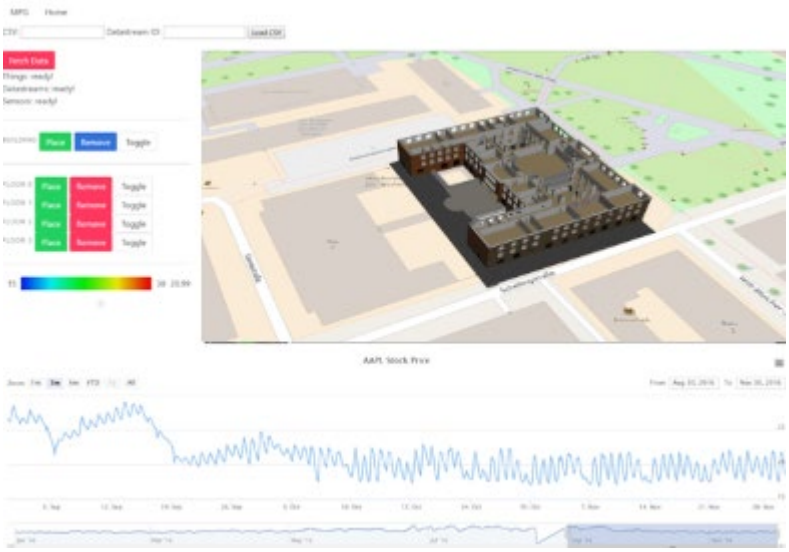


Figure 38. GIS Studio Project 3D sensor visualization of temperature time series.
Base Map © OpenStreetMap contributors (OpenStreetMap Foundation, 2018)

These prototypes show that spatial visualization can help to analyse and manage data more efficiently by enhancing existing systems and data with spatial information. For a public building with a higher installation of sensors and devices, a natural and user-friendly spatial visualization can improve the management and use of a Smart Public Building for the occupants. Commercial and proprietary software exists on the market that uses different visualization components for building management. Some provide editors to create the visualization objects and link sensor information to the entities (progea, 2018). These solutions provide a good set of tools to implement the sensor-visualization-model. To provide an alternative, the approach of using GIS to visualize the objects in 2D and 3D could be considered. The research goal is to evaluate the components and required workflow to implement a similar solution with the use of open source and free software packages and to provide a web-based visualization web client for IoT frameworks like openHAB.

The following parts describe an approach to implement a combination of 2D and 3D spatial visualization for devices that are controlled with openHAB.

8.3 Proposed Approach

To enable the user to control a bigger set of different sensor types and smart devices in a public building, a combination of 2D and 3D visualization is proposed. A web application allowing to view floor plans and 3D models of buildings that are enriched with spatial information of the sensors and the device location can be combined to control the things registered in openHAB. This approach will help to quickly assess the room situation and to access sensor information and controlling the smart devices. Figure 5 illustrates how information that can be displayed either in a 2D floor plan or a 3D building model to provide better orientation and to locate components quicker, especially if the system is used remotely.

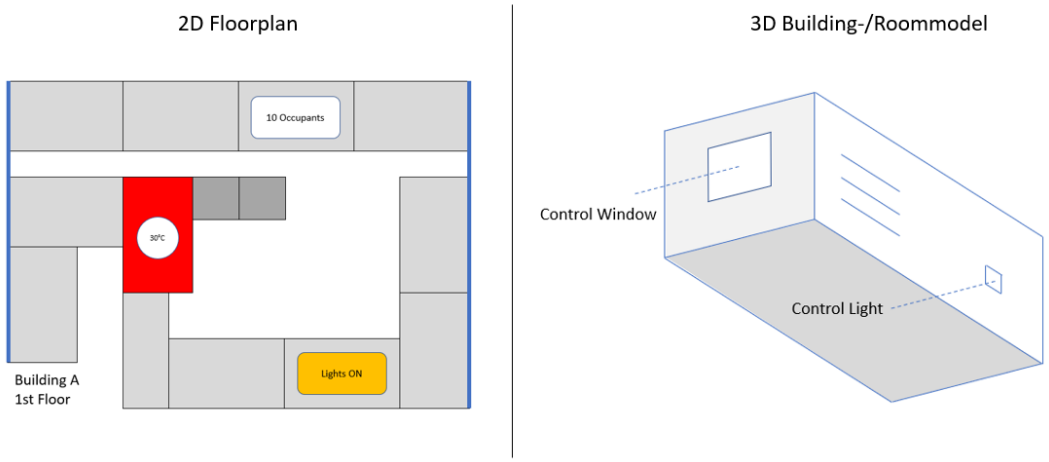


Figure 39. Illustration of data for 2D visualization and 3D visualization

The components should be separated in groups that are more suitable for 2D visualization like temperature, air quality and humidity sensor data. Providing a floor plan of the building as the base for the sensor visualization allows the user to navigate the spatial information in a natural way and assess the status of a room more efficiently than an approach that provides a list of sensors and room description. The information of the location of the sensor and the data gathered by the sensors can be displayed on top of the floor plan. Because all devices do have a link to a spatial location, the elements of a passive group can also be visualized in 3D space for special comparison of specific sensor information, like differences between sensor heights. For the most common use-cases the 2D visualization of items in the passive group is sufficient to provide meaningful information.

The second set of components can be grouped into a category of active devices that are located at a fixed position and can change a condition inside a room. Sound systems, shutters or a projector can be components that are fitting the properties of an active group component. For these devices, a 3D visualization can be integrated to combine the devices registered in openHAB with a spatial location inside a 3D model of a building or a room.

Combining these two ways of visualization allows the user to control the system in a natural way and the connection between devices and the spatial location is established quicker. Focusing public buildings, the visualization solution should be implemented for the use on a mobile device.

The next part will describe the design of a system that implements the proposed solution as a mobile web application and provides an overview of the requirements.

8.4 Requirements for Implementation

openHAB is focused on creating and controlling devices and does not provide a spatial visualization of the information. The open source philosophy of openHAB should be continued for the additional components that must be developed to achieve the proposed visualization solution (openHAB, 2018).

The following figure illustrates the suggested system architecture. A web service handles the communication between the openHAB installation and the database containing the spatial information. The user can access the information over a web client to visualize the spatial data and 3D models. Additionally, the web client can control the devices of the openHAB installation.

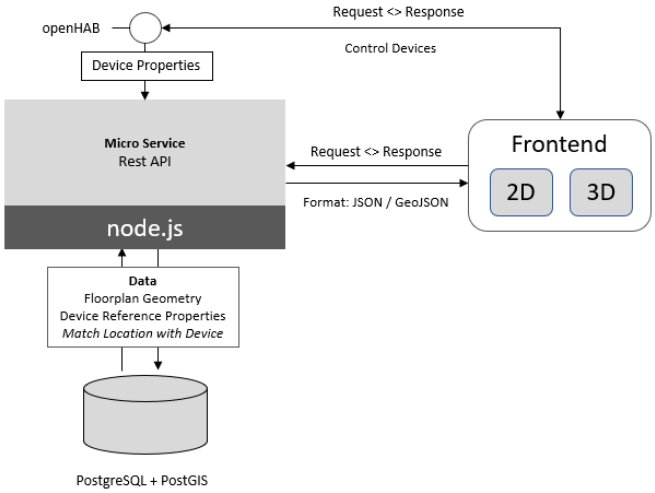


Figure 40. Suggested system architecture

8.4.1 Combination of Thing and Spatial Location

To display a device in the proposed 2D or 3D visualization a decision must be made to match selected devices to a specific location or an area. Depending on the procedure or measurement the device provides, groups of active and passive conditions can be modelled. An active group will be defined by devices that can be directly controlled by the consumer, where the device or a switch is placed at a specific location. The active devices will be placed in the 3D model of the building or a single room. A point geometry and the coordinates of the device location are stored together with a reference that points to the selected device. Passive devices to monitor conditions about temperature or air quality provide information that cover an area and cannot influence the environment by controlling the sensor directly. The passive data is visualized on a 2D floor plan, specifically linked to a room. A room is stored as a polygon geometry and the device is not referencing a specific location but an area. Figure 7 shows the proposed separation of devices and sensors into the suggested groups and the linked geometry to represent the item in the spatial environment.

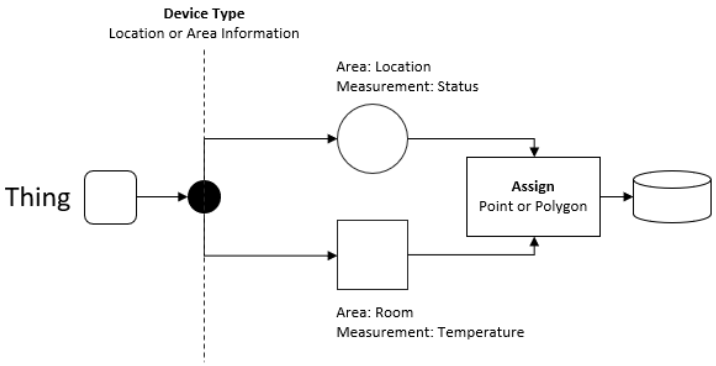


Figure 41. Differentiation of devices and groups of active and passive information

8.4.2 Spatial Data and 3D Model

Two types of data formats are required for the proposed solution. The data for the floor plans will be transferred as GeoJSON to be displayed on the web map. To produce the GeoJSON, the floor plans should be provided as Shapefiles (ESRI, 2018). A Shapefile is a format to work with geodata and can store different types of geometry as well as attribute information to describe the geometry. Shapefiles can be imported into a PostGIS enabled database and a SQL query will return the data as a GeoJSON structured response. The building model should be displayed on a virtual globe, this requires 3D models in a specific format.

CesiumJS (Cesium Consortium, 2018) provides the virtual globe and uses the 3D format glTF (The Khronos® Group, 2018) to display the models on the surface. In the example of the 3D sensor visualization the building models of the University of Applied Science Stuttgart had to be converted from the X3D format to glTF. The

conversion process is fast; most 3D modelling software allows to export geometry in a commonly used format to produce the required glTF files. An example would be Blender (Blender, 2018), an open source 3D modelling software.

8.4.3 Database and Spatial Data

PostgreSQL (The PostgreSQL Global Development Group, 2018) is an open source database that can be used as data store. The capabilities to process and manage spatial information can be provided by the PostGIS (PostGIS Project Steering Committee, 2018) extension. As described the database allows to store the necessary information about the coordinates of the geometry, the coordinate system that is used by the web mapping library and the virtual globe. Additionally, information that describes properties about the building, a floor or a room can also be stored and provide more details for the visualization. The database schema will consist of two tables, the GeoThing that links the spatial data to an openHAB device and the geometry table that contains the properties of the spatial data. Figure 8 illustrates the abstract database model and the required attributes that are stored.

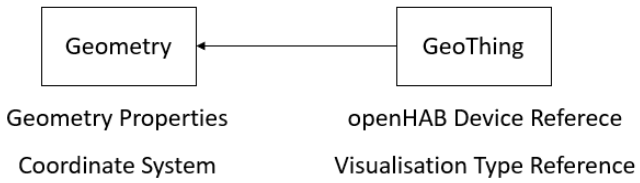


Figure 42. Abstract data base structure and store information

8.4.4 Restful API for Data Exchange

Node.js (Node.js Foundation, 2018) is a JavaScript runtime that allows rapid development of network applications and will be used for the implementation of a Restful API as a microservice. This is a common way to exchange data between web applications and services. Using a data format that can be processed by the application but also stays human-readable, like JSON (JSON, 2018) should be chosen, as this is widely used and supported by many programming languages and the go to format for data exchange (Guinard et al., 2011). This service must allow the web application to request the information about the spatial location of the devices. Depending on the type of data that is requested a public version can be available that shows status information relevant for every occupant of the building. A basic authentication procedure should decouple the public from the protected data that allows the management of devices or displays information that is not necessarily intended for every occupant. The API endpoint allows request data of the floor plans, device location and attribute data that describe the properties of the geometry that is required to be processed in the web application.

8.4.5 Visualization Components

Different JavaScript based web mapping libraries are available, the GitHub repositories of the following libraries show that they are actively maintained and commonly used in web projects. Both libraries can be used for the 2D visualization of floor plans and data linked to it. Leaflet.js, as described above, is focusing on a lightweight and mobile-friendly web mapping approach. An alternative to the already introduced leaflet.js is OpenLayers (OpenLayers, 2018) which aims to provide a more feature-rich web mapping library. For the case of the proposed solution only the web map and vector data capabilities of the libraries are required. The floor plans are visualized as vector data provided by the GeoJSON response from the Rest API. For the 3D visualization the virtual globe CesiumJS should be used, since the library was already successfully used to visualize temperature sensors and the corresponding time series of sensor data, described in chapter 2 of this paper.

8.4.6 User Interface and Basic Interaction

A simplistic user interface should be implemented with JavaScript to allow the user an intuitive navigation through the building elements. The data will be visualized by the web map elements that represent the smart device or a link to information gathered by a smart device. Furthermore, the focus of the visualization component is on the management of a public building where most of the interaction with the system is done remotely. In addition, a responsive and mobile-friendly user interface can extend the use case of the solution.

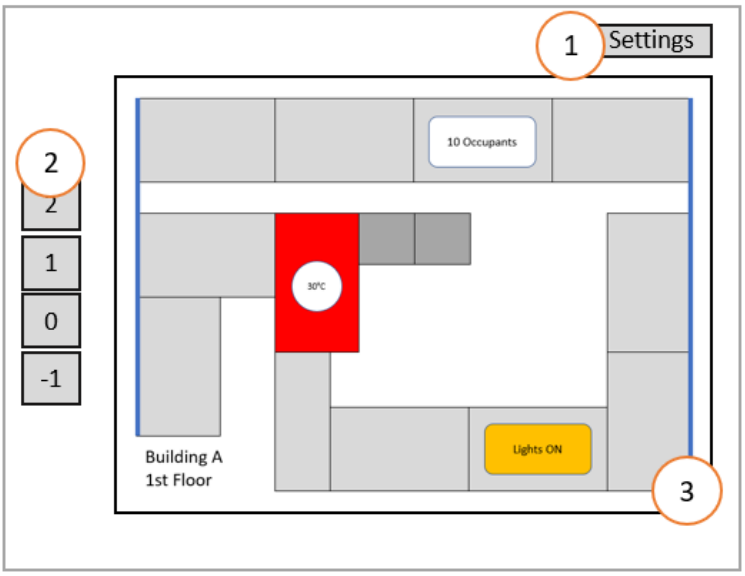


Figure 43. Illustration of the user interface. 1) Main application. 2) Floor controls. 3) Web map

8.5 Conclusion and Future Work

The integration of IoT systems to create Smart Public Buildings and the utilization of the gathered data provides useful information to the user. Using GIS and IoT frameworks together for data visualization might provide the necessary synergies to create an open source alternative to commercial products. GIS, especially with a high availability of open source software, provide a solid set of tools to cover the task of input data generation like floor plans and processing of spatial data. A bigger obstacle for the future development of IoT system in general is the importance of security and privacy of sensitive data, especially in Germany where new General Data Protection Regulation directive has been passed to protect the users privacy. These decisisions make it difficult to create systems that collect data that could affect a user and the improper use of data. For a solution that is operating in a public building a lot of information can be gathered and it has to be observed if this will impose a problem to data protection and privacy.

As described the proposed solution is using floor plans and building models to visualize data and to manage smart devices. Missing data must be made available e.g. by installing relevant sensors and recording their location. In case floor plans are not available or are provided in an unsuitable format, the plans must be converted or produced, which is a time-consuming task. The same condition does apply to 3D models.

A prototype that uses the different 2D and 3D visualization can negatively affect the performance of a device, especially a mobile device. It should be evaluated if an implementation with the proposed components provides enough performance for both the desktop and mobile use case. For further development the solution can be extended with components that allow to quickly place new sensors or devices in the virtual environment. This could be achieved by a mobile application that allows the user the place items on-site.

8.6 References

52° North, 2018. Home - *52° North Initiative for Geospatial Open Source Software GmbH*. Home - 52° North Initiative for Geospatial Open Source Software GmbH. https://52north.org/ (accessed August 27, 2018)

Agafonkin, V., 2018. *Leaflet - a JavaScript library for interactive maps.* https://leafletjs.com/ (accessed August 27, 2018)

Blender, 2018. blender.*org - Home of the Blender project - Free and Open 3D Creation Software.* https://www.blender.org/ (accessed September 1, 2018)

Cesium Consortium, 2018. *CesiumJS - Geospatial 3D Mapping and Virtual Globe Platform.* https://cesiumjs.org/index.html (accessed September 1, 2018)

Eclipse Foundation, 2018. *Eclipse SmartHome - A Flexible Framework for the Smart Home.* https://www.eclipse.org/smarthome/ (accessed August 27, 2018)

ESRI, 2018. *ESRI Shapefile Technical Description.* http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (accessed September 1, 2018)

Gartner Inc, 2018. *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015.* https://www.gartner.com/newsroom/id/3165317 (accessed October 5, 2018)

GeoJSON, 2018. *GeoJSON.* http://geojson.org/ (accessed August 27, 2018)

GfK, 2018. *GfK Future of Smart Study.* https://www.gfk.com/fileadmin/user_upload/dyna_content/GB/documents/Innovation_event/GfK_Future_of_Smart_Home__UK_.pdf (accessed August 23, 2018)

Gitahi, J., & Jensen, M., 2018. *Visualization of Time Series in Cesium.* http://gisstudio.hft-stuttgart.de/cesium.html (accessed August 27, 2018)

Guinard, D., Trifa, V., Mattern, F., & Wilde, E., 2011. From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices. In: D. Uckelmann, M. Harrison, & F. Michahelles, *Architecting the Internet of Things.* Springer Berlin Heidelberg.

Hochschule für Technik Stuttgart, 2018. *EMAS Webservice.* http://campus.hft-stuttgart.de/ (accessed August 27, 2018)

Jensen, M., 2015. *Konzeption und Implementierung einer webbasierten Kartenanwendung zur Visualisierung der Meldungsprotokolle aus dem Facility Management System der Firma Bosch.* Hochschule für Technik Stuttgart.

JSON, 2018. *JSON.* https://json.org/ (accessed September 5, 2018)

Meehan, B., 2018. *IoT and GIS: Transforming the Utility Industry and Improving Lives.* https://www.geospatialworld.net/article/iot-gis-transforming-utility-industry/ (accessed October 14, 2018)

Node.js Foundation, 2018. *Node.js.* https://nodejs.org/en/ (accessed August 31, 2018)

OGC, 2018. *Welcome to The Open Geospatial Consortium | OGC.* http://www.opengeospatial.org/ (accessed August 27, 2018)

openHAB, 2018. *Our Vision and Philosophy.* https://www.openhab.org/about/who-we-are.html (accessed August 27, 2018)

openHAB, 2018. *openHAB*. https://www.openhab.org/ (accessed August 27, 2018)

OpenLayers, 2018. *OpenLayers - Welcome.* http://openlayers.org/ (accessed September 3, 2018)

OpenStreetMap Foundation, 2018. *OpenStreetMap.* https://www.openstreetmap.org/copyright (accessed October 13, 2018)

PostGIS Project Steering Committee, 2018. *PostGIS - Spatial and Geographic Objects for PostgreSQL.* https://postgis.net/ (accessed August 31, 2018)

progea, 2018. *Integrierte Leittechnik für intelligente Gebäude.* https://www.progea.com/wp-content/uploads/2017/10/NEXT_BA_DE.pdf (accessed October 15, 2018)

Schaus, Y., 2018. *3D WebGL interactive smarthome control with openHAB, HABPanel & Sweet Home 3D.* https://github.com/ghys/habpanel-3dview (accessed September 1, 2018)

Semmo, A., Trapp, M., Kyprianidis, J., & Döllner, J., 2012. Interactive Visualization of Generalized Virtual 3D City. *Eurographics Conference on Visualization (EuroVis) 2012.* Blackwell Publishing. Interactive Visualization of Generalized Virtual 3D City: Interactive Visualization of Generalized Virtual 3D City.

The Khronos® Group, 2018. *glTF Overview - The Khronos Group Inc.* https://www.khronos.org/gltf/ (accessed September 1, 2018)

The PostgreSQL Global Development Group, 2018. *PostgreSQL: The world's most advanced open source database.* https://www.postgresql.org/ (accessed August 30, 2018)