FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Microlocation based configuration for KNX installations

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Daniel Ramsauer
Matrikelnummer 1226696

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Wolfgang Kastner
Mitwirkung: Univ. Ass. Dipl.-Ing. Stefan Seifried, BSc.

Wien, 27. November 2016

Daniel Ramsauer	Wolfgang Kastner

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Microlocation based configuration for KNX installations

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business Informatics

by

Daniel Ramsauer
Registration Number 1226696

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Wolfgang Kastner
Assistance: Univ. Ass. Dipl.-Ing. Stefan Seifried, BSc.

Vienna, 27th November, 2016

Daniel Ramsauer

Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Daniel Ramsauer
Gartenheimstraße 23/4

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. November 2016

Daniel Ramsauer

Abstract

Building automation is getting more and more attention with the rise of the Internet of Things (IoT). Due to the release of the Internet Protocol Version 6 (IPv6), it is possible to address a huge amount of devices. This enables the IoT with its various areas of application. Apart from the concept of smart cities, smart buildings will have an important task in aiding the tenant of such a building and increase the comfort and efficiency. There are challenges left to solve, to fully integrate the IoT in our daily life.
The installation and configuration of a building automation system is often done by a single vendor. In most cases this means, that users of this system are not able to change device configurations of the system with ease. Furthermore, if such a system is installed in a big building, such as an office building, there is a high amount of smart devices which can be configured by specialized software and experts. Therefore, microlocation may help, in order to filter only important or near devices for the user, so that the usability of the system and its configuration increases. The different approaches to locate the user inside the building depend heavily on the used technology, such as Bluetooth or Ultra-wideband (UWB)-based technologies. These technologies as well as their drawbacks and solutions are described in detail.
Regarding the implementation of an application which enables users to easily group existing KNX devices, there are a few prerequisites that have to be fulfilled. Next to the existing KNX system there has to be some kind of middleware to enable the users to connect to the KNX devices with a mobile application. Additionally to that, a lightweight protocol, such as the Constrained Application Protocol (CoAP) has to be used in order to reduce the bandwidth usage of a mobile device, due to limited transfer rates.
Within this thesis, a proof of concept solution has been developed to verify that an easy way of setting up new memberships of KNX devices is possible within the current specifications.
In conclusion, the proposed solution for an easy way to create KNX device groups has been proven feasible and can be extended to further increase the range of applications.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Due to the rise of the IoT in the recent years, automation is getting more and more attention not just in the industrial domain (e.g. cyber-physical production systems) but also in the context of smart buildings. Typically, the configuration of building automation systems is done at installation time. This means that only a qualified person is able to change the functionality of given devices. This is a problem in fast changing environments. Therefore, an approach has to be taken, where users can easily and fast re-arrange a building automation system during runtime.

1.2 Problem Statement

The focus of this work is on the discussion of current possibilities to locate users inside a building with microlocation, which is a subcategory of location-based services, alongside the necessary technologies that enable this localization. Furthermore, this work states current challenges for the configuration and commissioning of building automation installations. One of the objectives of this work is to present a natural way of re-arranging of complex building automation systems based on the KNX automation protocol. In this context, the term *natural* means an easy and straightforward way for the user to define new device groups. Therefore, the essential technologies and their prerequisite for enabling such a *natural* configuration, are discussed. Another important requirement is, that the proposed solution shall be built upon existing technologies.

1.3 Aim of the Work

The major aim of this work is the discussion of the microlocation concept and its necessary prerequisites. This further includes the IoT and its basic concepts. Based on

that, approaches to configure current KNX installations are stated in addition to the most common building automation protocols. Additionally, the requirements to enable a fast and easy way for the users to define new memberships in such KNX installations are discussed. A detailed look at the technologies that enable microlocation is given, alongside the technologies to enable a lightweight communication between a client device and gateway, which maintains the states of the KNX system.

1.4 Structure

At the beginning, an overview about the motivation, the problem statement and the aim of this work are stated. Chapter 2 introduces state of the art concepts, which are necessary to enable building automation systems. The IoT gets discussed alongside its services and challenges. Furthermore, the configuration and commissioning of building automation devices are stated and the chapter closes with a detailed look at the concept of microlocation. Chapter 3 provides a detailed look at the requirements and technologies that enable a natural way to define groups within KNX-based devices. In Chapter 4, the proof of concept solution is discussed and the work closes with a summary and an outlook.

CHAPTER 2

State of the Art

2.1 Internet of Things

2.1.1 Introduction

The main concept of the IoT is the interconnection of various *Things*. These *Things* can be sensors, actuators or any object with computational capabilities. Upon connection, they can cooperate with each other and exchange data. For this to work, IoT depends on the *Big Data* approach (see Section 2.1.2), which is basically a concept to process, store and analyze huge amounts of data that IoT *Things* generate. In order to exchange data, there are multiple wired and wireless standards available. The most common protocols are Bluetooth, Wireless Local Area Network (WLAN) and Ethernet. With regard to the huge amount of possible *Things* in the IoT, the release of IPv6, which its large address space, was a precondition to enable direct interaction of devices. Without the new Internet protocol version, it would not be possible to support the wide range of applications and concepts of the IoT.
The following sections provide an overview over common concepts in the IoT.

2.1.2 Concepts

Things in IoT

Things are all devices that have a unique identifier, computing capabilities and the ability to transfer data over a network. Most of them only have limited memory and computing capabilities. There are two different categories of *Things*, active and passive ones. Active *Things*, such as smart phones, have their own calculation power and can be used for data processing. Passive *Things*, such as daylight sensors, only provide data to other *Things*. These passive *Things* "wake up" for a short amount of time, process the data and then go back to sleep mode again. This cycle is repeated in order to minimize the energy

consumption. Due to the fact that there are many *Things* inside a building or even a whole city (see Section Smart Cities 2.1.2), easy management of the huge amount of devices is crucial. The sheer amount of *Things* has also a great impact on the overall energy consumption, thus there is a necessity to optimize energy management as well.

Big Data

In order to analyze and process all the information that is collected by *Things* in the IoT, *Big Data* is necessary. *Big Data* is defined by [1] with five different Vs: Volume, Variety, Velocity, Value and Veracity. Volume stands for the huge amount of data, that is processed (Tera- to Zettabyte). The Variety aspect defines the storage of different kinds of data, such as text, graphics, audio and video. Velocity is another important characteristic, which means to process and analyze data in a fast manner. Value means that *Big Data* increases the company value, and Veracity stands for rating of the data, due to the fact, that the data quality can vary.

Without *Big Data*, the IoT would not be feasible. The huge amount of data created by billions of distributed sensors could not get analyzed efficiently without systems that utilize the *Big Data* concept. The fast and real-time processing of data is very important to provide applications and services to users or even other machines or systems within the IoT. This is due to the fact, that the data (e.g. sensor data) that is important to users or other *Things*, is more valuable, if it gets processed by *Big Data* the instant it arrives. [2] provides a more in depth look into *Big Data* and its association with the IoT.

Smart Buildings

The concept of smart buildings is inter-related with the IoT. The Institute of Building Efficiency [3] defines them as "buildings [that] deliver useful building services that make occupants productive (e.g. illumination, thermal comfort, air quality, physical security, sanitation, and many more) at the lowest cost and environmental impact over the building lifecycle." In order to enable these features, the integration of the IoT has to be implemented, starting from the design phase up to the fully functional building. Additionally, a smart building usually has some level of energy independency such as its own power generation through renewable resources and incorporation of energy efficient technologies.

It is of importance, that every subsystem of the building is connected and is able to share information with the others. Hence, there is a necessity for system interoperability, which further optimizes the total building performance. Moreover, connection to the smart grid (or smart power grid [4]) enables smart buildings to provide excessive energy to the grid or other buildings, as well as request it based on agreements.

A smart building has to be able to store, process and analyze all information gathered by interacting entities, such as tenants, actuators or sensors, to deliver its services.

The following points summarize the characteristics of a smart building:

- Interconnectivity of different subsystems
- Providing services to tenants
- Connection to other buildings
- A possible connection to the smart grid, to further enhance the services

The integration of microlocation (see Section 2.3) enhances the performance of smart buildings, due to the possibility to locate a tenant inside a building with maximum accuracy.

Smart Cities

By 2050, the UN forecasts the number of world population that lives in cities up to 70 percent [5]. This means that the urban infrastructure, water and energy supply, as well as transportation, healthcare, education and safety have to get more efficient. In order to reduce the costs of these services, and still deliver the quality that the citizens expect, cities have to get smarter [5]. There are different definitions of *Smart Cities*, which are summarized in [6]. This work is based on the following definition by [7]: "The use of Smart Computing technologies to make the critical infrastructure components and services of a city - which include city administration, education, healthcare, public safety, real estate, transportation, and utilities - more intelligent, interconnected, and efficient." This means that a *Smart City* heavily depends on the *Things* from the IoT as well as *Big Data* to process the gathered information and to analyze every aspect in the city to provide its services. Until a city can transform into a *Smart City*, there are a few challenges left to solve. The details of these challenges can be found in [5].

2.1.3 Application and Services

Within the context of microlocation, there are a few selected applications, that are enabled by the IoT.

Intelligent Transport Systems

The main idea behind Intelligent Transport Systems (ITS) is to apply information and communication technologies, such as sensors and other things, to every transport mode (road, rail, air water). Then, this information is used to provide services to its passengers and to freight transportation. In order to fully utilize ITS, *Big Data* is of utter importance. The processing of the best routes or the most efficient freight loading gets calculated with optimized algorithms with the help of *Big Data*. ITS includes several different systems, such as automatic traffic management systems, fleet management and location

systems, public transport information systems as well as cooperative vehicular systems and intelligent car parking systems. Further information on ITS can be found in [8].

Product Tracking

One of the first commercial application forerunner of the IoT was product tracking. It enabled companies to further improve their logistic services [9]. Due to different technologies, the movement of goods can be tracked in real-time, shelf space can be managed more efficiently, inventory control is improved and furthermore, the amount of human involvement is reduced.

Homes and Offices

A Smart Building equipped with the *Things* of the IoT enhances the life of inhabitants of offices and homes. Room heating can be adapted to the tenant's preferences and the weather and lighting can change according to the time of day. The energy consumption can be reduced by switching off electrical equipment when not needed. Smart Building equipment with different ways of producing thermal heat, such as pellets or gas, could switch between them, according to the prices for the resources on the market [10].

Gaming

The possibility to track a user inside a Smart Building with *Microlocation* opens new ways of gaming for players. A player could be equipped with a variety of devices for location tracking, sense acceleration, noise, voice, heart rate, blood pressure and visual information. All information together is then used to simulate games within the building. One example would be to get from point A to point B without touching the floor. With the usage of the Radio-frequency identification (RFID) technology, various objects are placed randomly inside the building and have to be collected in a certain order. The sensors measure different ways of jumping (long jump, high jump, etc.) and reward the players with different amounts of points. As soon as a player touches the floor, the game makes a noise to alarm the player that this was wrong. In order to keep the excitement high, awards and difficulty get adapted to the achieved scores [10]. Furthermore, virtual reality technologies can be used to enhance such an experience.

2.1.4 Challenges

The IoT faces many different challenges. From an operational point of view, after-sale service processes have to be adapted to satisfy the requirements of connected products. These enable new forms of marketing tools, which lead to a more direct or extended communication with customers. At a technological level, the implementation of an IoT application needs to integrate different information and communication technologies. Companies that are producing these connected products have to build an active ecosystem around these IoT platforms to provide support for customers as well as for industrial partners [11].

Standardization is another challenge that arises due to the broad amount of application and services of the IoT. Since there are multiple contributors and vendors, there are different technologies used and interfaces have to be deployed to ensure connectivity among them. Many standardization organizations, such as the International Organization for Standardization (ISO), the European Telecommunications Standards Institute (ETSI) and different working groups are collaborating together on achieving the IoT [10]. Besides new technologies, existing standards have to be revised and adjusted to fit the IoT. Without standardization, the interworking of things would be difficult, due to different interfaces, runtime behavior and configuration of the devices.

Addressing and networking is another issue that has to be solved to enable the IoT. As stated before, IPv6 enables the IoT with its huge amount of addresses, but some technologies (such as RFID) do not provide same sized identifiers/addresses. Due to the fact that most of the future things in the IoT have a unique IPv6 address, different approaches have been investigated and proposed to integrate the technologie standards into IPv6 addresses, so that these technologies can be globally identified as well. One approach would be to map the identifiers of those technologies to IPv6 addresses, but then a gateway would have to store these mappings and ensure, that each mapped IPv6 address is unique as well. Furthermore, these solution don't offer mobility between different systems. More details to the challenges that arise with these technologies and IPv6 can be found in [10].

In case of networking, the existing transport layer has to be revised. For example, the Transmission Control Protocol (TCP) is connection oriented. Therefore, it begins a new session with a Three-Ways-Handshake. This setup procedure is unnecessary, due to the fact that the communication within the IoT involves the exchange of small amount of data. More details to this challenge are described as well in [10].

Another challenge is the configuration of the *Things* of the IoT. Due to its large numbers, there has to be an efficient and easy way to configure many devices at once. In order to further increase the usage of these devices, and to react to changes inside the system, the devices have to be able to adapt themselves to the changing environment without the involvement of humans.

Other challenges are device level energy supply, security and personal privacy, as well as the support for the latest and evolving standards ([9], [10]).

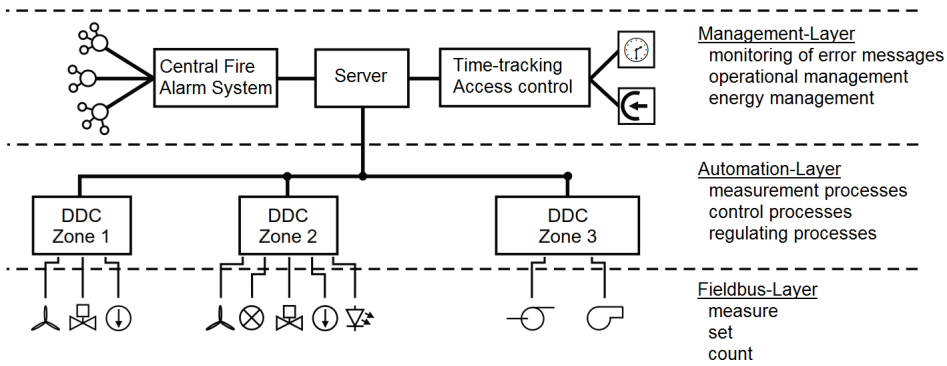
2.2 Building Automation

The rate of automation in buildings is continuously rising throughout the past years. On one hand, this is due to the growing demand of comfort. On the other hand the automation is necessary for energy-saving and energy management (see Section 2.2.1). Furthermore, this automation should provide more security for private homes and more

flexibility in cases of functional building.

The hierarchical structure of building automation consists basically of three layers: The Fieldbus-Layer, the Automation-Layer and the Management-Layer (see Figure 2.1). The Management-Layer provides complex processing and analyzing of the given information. The Automation-Layer manages the simple, hardware-based control functions, whereas the Fieldbus-Layer is providing information from its actuators and sensors to its layer above.

Figure 2.1: Building automation protocol layers



The communication between these layers is based on the used protocol (see Section 2.2.2). In most cases it is wired, due to the fact that it provides a higher reliability. However, this is reflected in higher costs, such as a longer installation process, the need to install extra cables as well as to merge these cables in one or more cabinets. The need for extra cabling is based on the separation between the power line and the control or fieldbus line of the device.

2.2.1 Energy Management

One of the main tasks covered by building automation is to reduce the energy consumption of all interconnected devices. There are many different management functions that are most commonly used [12], whereas only a few are presented in this work.
One energy management function is the controlling of the Heating, Ventilation and Air Conditioning (HVAC). Connected with a sensor that measures the temperature, the HVAC system is controlled accordingly. This means for instance, that if the outside temperature is low, the temperature gets raised. This reduces the energy costs as well as it avoids health impairments due to the lower temperature difference.
Another energy management function is event-based switching. One example for this energy reducing measure is the controlling of the energy consumers inside a smart building. A hotel would be able to switch off all consumers inside a room and lower the temperature. Another, more general example, is the ability to switch on lights in a staircase, a hallway

or a room, in case a motion detector senses movement.
More energy management functions, such as time-based switching, can be found in [12].

2.2.2 Building Automation Protocols

This work takes a closer look at three different building automation protocols that are commonly used: KNX, Local Operating Network (LON) and Building Automation and Control Network (BACnet). KNX and LON are mostly used in the Fieldbus-Layer, whereas BACnet is used in the Management -Layer and the Automation-Layer alongside other protocols [12].

KNX

This work is based on the current status of the KNX specification as of year 2016 (see [13]). KNX is a standardized bus system, which enables data exchange between devices and systems even if they are of different vendors. It is based on the Fieldbus-Layer and each KNX device is connected to this physical layer. KNX has several physical layers defined, such as twisted pair (KNX.TP), power line (KNX.PL), radio frequency (KNX.RF) and Ethernet (KNXnet/IP) [13].
Data is packed into telegrams, which are distributed to all members of a KNX group. Hence no central communication broker is needed. Every KNX device has its own configuration that can be changed and configured any at time. This makes KNX very flexible, due to the fact that the behavior of any actuator can change.

Basically, the deployment procedure for KNX devices is as follows. First the hardware has to be installed and connected to the fieldbus. Thereafter, each device must be assigned a unique individual address. This step sets the topology of the bus system, which means that KNX devices get assigned to areas and lines. After this, the device has to be configured and parameterized. This is done through the Engineering Tool Software (ETS). Communication relationships between devices are then established by the means of group addresses. In order to exchange data, different physical layers may be used. The most common would be twisted pair (KNX.TP) and power line (KNX.PL).

The assignment of the individual address should be as meaningful as possible to make identification easy. An example would be to address it based on the order of installation in a room. Each address has the same layout: Area.Line.Address. Inside a KNX telegram. the area and the line get 4 Bit each and the address has 1 Byte (see Figure 2.2). Therefore, there can be 16 areas, 16 lines and 256 devices per line, which means in total there can be 65 536 devices in a system. For more in-depth informations [12] provides different types of addressing schemata and examples.

A basic communication cycle is as follows. An event gets triggered, such as the push of a light switch. This switch then sends a telegram with a certain group address and all devices within that group are confirming that telegram with an acknowledgement

Figure 2.2: KNX address

Area	Line	Device Address
AAAA	**LLLL**	**DDDDDDDD**
4 bit	4 bit	1 byte

telegram. Further information can be found in [12] and in the KNX standard [13].
KNX provides a worldwide standard for home and building automation. Furthermore, its configuration software (ETS) is manufacturer independent and supports several communication media.
While writing this work, a standard for KNX is being prepared in which KNX will support webservices (called KNX WS).

The drawback of KNX is that products are relatively expensive compared to conventional installation technologies and KNX is only economic if the installation needs to be flexible. Additionally, the standard does not provide any kind of discovery mechanism, which means that it is hard to find a specific device in a complex installation.

LON

The Local Operating Network, developed by the Echelon Corporation and located on the Fieldbus-Layer and above (see Figure 2.1), is used to solve different automation tasks. It features devices with its own intelligence and groups them together to local networks. This enables decentralized comfort solutions for single rooms. Similar to KNX, the sensors and actuators within LON exchange their information over a bus line. The sensors are separated from the power line and get connected to the local controller (e.g. a switch or a dimming actuator). The concept of LON is to process the information locally. For example, a device equipped with a switch and a temperature controller would be able to control the light as well as the temperature inside a room. As soon as a user presses the switch, the light is controlled according to a light sensor and the room temperature is adjusted based on the set value. The intelligence for this is built-in the device. Additionally, the current values can be sent to a centralized server and controlled from there. Just as with KNX, the configuration of LON devices can be changed anytime and makes LON very flexible.

Each device has one or more built-in Neuron-Chips, which also provide processing functionality. Every Neuron-Chip is a so-called LON-Node. After physically connecting these devices, they communicate with each other with the LonTalk-Protocol. In case that the amount of nodes exceeds 64, a LON-Repeater has to separate the nodes into two networks. Hence, each network has a maximum of 64 nodes. With a repeater it is possible to address up to 128 nodes, grouped together into a subnet. In order to address more than 128 nodes, a new subnet has to be created and in between these subnets there

has to be a router. This structure is called a Domain. It is possible to address up to 255 subnets, which means that the maximum size of a Domain is 32 385. In the event that this limit is exceeded it is possible to connect two domains with a powerful router.

In order to configure the LON-Nodes, the vendors provide specific programs with templates that can be used to adapt the nodes to the user's needs. The most common tool to configure a network is LonMaker. A USB-stick with a built-in LON-Node provides the connectivity to the LON-Network. For a more in-depth look at LON and its configuration, see [12].

BACnet

The Building Automation and Control Network (BACnet) is a communication protocol that has been developed by the American Society of Heating, Refrigeration and Air-Conditioning Engineers (ASHRAE). It provides a standardized protocol for building automation, so that devices in a system can exchange information with each other. Its main goal is to be vendor independent and to provide interoperability. It is most commonly used at the Management-Layer, due to its strengths in management functionality and often uses KNX or LON as protocols in the Fieldbus-Layer. In order to communicate, BACnet is able to use existing local area networks, which makes it very cheap to use. Furthermore it is capable to utilize other networks such as LonTalk.

BACnet uses an object-oriented approach to represent all information within a device. Each device can have one or more objects, depending on the functionality that this device offers. A number of objects is standardized. Each object has its own unique identifier (32-bit) which is split into a 10-bit long *objecttype* and a 22-bit long unique *objectnumber*. For easy identification each object has a unique object name as well. Furthermore, each device has an object assigned to itself. The 22-bit long identifier enables BACnet to address $2^{22} = 4\ 194\ 305$ devices. An object can have many different properties, depending on the vendor and the functionality of the device.

In order to enable communication with other protocols, BACnet uses gateways. A Gateway converts a message from one protocol into another protocol. Further information about BACnet can be found in [12].

2.2.3 Commissioning of IoT Devices/Building Automation

Middleware

A middleware is a software which is located between the operating system and an application. It provides an abstraction layer to assist programmers and applications to communicate with services or distributed systems in the network and/or to connect different software together. The drawbacks are, that a middleware adds additional overhead, which results in a higher memory usage as well as increased usage of computational

resources. Due to the fact, that there is more data to process and send, more processing power is needed, as well as a higher network bandwidth to transport above mentioned data. In building automation, the middleware concept is very important. There are several different protocols and technologies, that need to work together to enable outright automation. Thus, the middleware translate between them and provides a common interface that a user or a programmer can interact with. See [14] for in-depth information.

Existing Approaches of KNX Configuration

Within the scope of this work, there are two different companies that provide KNX-based installations. Loxone [15], an Austrian company, focuses on Smart Homes, and Comfortclick [16], a Slovenia company, which focuses on Smart Buildings.
Both companies provide solutions for the front-end of KNX installations (servers, KNX devices, etc.). In terms of configuration, the KNX configuration tool ETS (see [17]) is used. In terms of physical installation, each KNX device has to be connected to the fieldbus system. This means in most cases extra wiring and planning. Important to note is, that they do not provide software to reconfigure existing KNX installations.

In both cases the configuration of their software is similar. First, the controller or gateway has to be connected to the corresponding server. Next, every connected device has to be registered into the system. This basically means, that it has to get an address and an identifier. In order to add logic (e. g. flipping a switch will turn on a light), each device has to be selected and functionality has to be added. This means that a user has to know exactly which actuator, sensor or light he wants to group together. In case the name to identify the device alone isn't enough, a list with device reference or something similar to identify the object is necessary. For example, an office room could have a high amount of sensors and actuators. Both companies provide similar mobile applications, where a user can control the listed devices. Basically, these front-ends list all devices which are registered in the system. In case of a huge office building, or even a big house with lots of KNX devices, this approach is not suitable due to the sheer amount of connected hardware. A more in depth configuration guide can be found at [18] for Loxone and [19] for Comfortclick.

A possible solution for this problem would be the use of QR-codes (see Chapter 2.3.5) to locate KNX devices and to configure them with a mobile device such as a smartphone. An application can scan these codes and interact with a middleware that is connected to the KNX server. The middleware handles the incoming requests to configure the devices accordingly, which enables the user to change the configuration of the system without tracking down the correct devices in the user interface.

The proof of concept (see Chapter 4) requires the initial deployment of the hardware (sensors, actuators, etc.) along with the corresponding localization technology (QR-codes, see Section 2.3.5). Furthermore, the infrastructure for the server (IoTSyS , see [20]) and the client (MCBAS, see Section 4) has to be installed and configured. This provides a

solution for large KNX installations, such as office buildings, and enables the configuration for end-users.

2.3 Microlocation

2.3.1 Introduction

Microlocation is a subcategory of location-based services. The location of a user inside a building is of primary importance to provide him with effective and efficient solutions. For outdoor applications, such as car navigation systems, the Global Position System (GPS) with its approximately 10m accuracy is often precise enough for a user to identify its position on a map. Due to the fact that GPS performs poorly indoors and its high energy demand, it is not suitable for permanent location tracking inside buildings. The IoT depends heavily upon the locating of the user. This leads to an increase of microlocation-enabled hardware and protocols that can be used to develop microlocation applications. As stated in the Section 2.1, smart buildings depend on the user's location. Hence, there is a necessity for microlocation enabling technologies and techniques to provide location based services [21].

In order to further enhance the localization of a user, the concept of geofencing is used. This concept relates a geographic area to certain object. This fence gets associated with predefined action conditions and actions (see [22]). An action condition, when satisfied, causes certain actions to be triggered. For example, when a user enters or leaves a geographic area. Different actions can be triggered based on the time of day or the season, as well as based on the movement inside the fence. There is no limiting factor in the size of the geographic area and it is possible to define areas within areas to create complex action conditions. In conjunction with context aware systems, this concept can provide targeted services for tenants as soon as they trigger a condition.

The next section focuses on different microlocation-enabling services. In the succeeding section, a look at the challenges that arises in microlocation is provided. Followed by an overview of position measurement systems, which shows the different forms of localization. Finally, technologies which enable microlocation, are discussed in the last section.

2.3.2 Microlocation Enabled Services

Microlocation in smart buildings enables many services for various residential and enterprise applications. This section provides an overview over four different scenarios.

Targeted e-marketing

Microlocation can be used to inform nearby costumers of special offers which are inside a geofence. Since a smart building is able to be context aware, this feature helps advertisement and can attract customers to other shops as well as it enables many other

marketing solutions. For example, a costumer that just bought new shoes gets a special discount for a matching T-shirt in another shop inside the geofence.

Tenant assistance

With the usage of microlocation and geofencing, tenant assistance has a wide-spread field of applications. It is primarily designed to assist and provide information to the users within the range of the system. In combination with context awareness, this feature gets further expanded. One example would be a visit to a microlocation-enabled museum. Users that already have visited the exhibition, could have left messages or comments near a certain piece of art. Every user that has subscribed to this comment-service is informed about this comment, when near this piece of art. Furthermore, the mobile device of the user is aware of preferences, and the microlocation service is able to lead the user to the exact part of the museum. Another example are intelligent elevators inside an office building. Elevators are called if a user enters the lobby and they are idle. Or after a meeting an elevator my wait for the tenants to bring them to different floors based on their calendar or learned behavior pattern. With this in mind, the room where a meeting is held will get cooled based on the thermal load caused by the amount of participants. This increases the comfort of the users as well as the energy efficiency of the building.

Energy efficiency

Energy efficiency is a key aspect of smart buildings and microlocation. However, buildings need to meet some prerequisites till the energy efficiency can be increased by microlocation. For example, the lights have to be connected to a controller, such as demand side management or an electricity consumption controller, to automatically switch them on and off. There has to be an electricity consumption controller to provide efficient solutions as well as energy storage capabilities. According to the users' location, the system uses the minimal needed energy to comfort the user. If a tenant leaves the building for a longer period of time, the system reduces the HVAC or other devices to save energy. There are many other possibilities with microlocation and geofencing to increase energy efficiency.

Disaster management

The effects of disasters (e.g. a fire, flood or terrorists attack) can be reduced by the help of disaster management. In contrast to traditional buildings, smart buildings can aid this management due to microlocation and the IoT. In case of a fire, the HVAC can be deactivated inside a room to prevent the fire from spreading, as well as every tenant gets an alert using microlocation-enabled services. This can further be used to calculate a safe escape route, if the normal routes are blocked by fire and smoke and the smart building is aware of it. Disaster managers and the fire department will get accurate information and are provided with information about tenants inside the burning building and their location. This increases the survival rate in case of a disaster.

2.3.3 Challenges and Solutions

In order to use microlocation services to full extent, some challenges have to be solved.

Accuracy

Microlocation only works, if the position of the user is accurately measured or calculated. Furthermore, the accuracy has to be as high as possible to fully use microlocation enabled services. Many technologies, such as GPS or WLAN, are not suitable for microlocation due to low accuracy. Other technologies are not providing enough accuracy to fulfil the need of microlocation enabled services. Thus, there is a necessity to enhance the known technologies by applying filters, such as the Kalman filter, particle filter, simultaneous localization and mapping or other filter technologies.

Interoperability

As with many technologies, there are no clear standards for microlocation enabling technologies, which results in bad interoperability. There are three scopes: Technology, software development kits and protocols. In case of technology, there are various different microlocation enabling technologies that utilize different concepts. Even within a technology (e.g. iBeacon) there are several vendors that offer no compatibility to other products. For the user, it is not important what technology is used. However, it is important that the location is as accurate as possible. In the end, different microlocation enabling technologies need to interoperate with each other.
The same problems apply to software development kits (SDKs), due to the fact that every vendor provides his own SDK for application development. This results in a lock out of other technologies, because of the bound to the vendor-specific SDK. Special integration layers have to be developed to enable the interaction with different vendor-specific technologies. The drawback is that if one of the components is upgraded in the future, parts of these layers have to be updated. There are also no standardized protocols for microlocation enabling services, which results in a lack of interoperability. In order to increase efficiency as well as make microlocation enabled services work together, all these systems have to interoperate.

Privacy

The user's location has to be made accessible for microlocation enabled services to work. This leads to privacy issues. The agreement of the tenant is needed to reveal his position and to actually use the services. Some users will hesitate to constantly reveal their position and won't give their approval to use microlocation enabled services. Furthermore, the location data may be stored in order to provide the tenants with context based services. It is very important that this data is securely stored and used, so that the users of the system don't have to worry that their privacy is compromised. Winning the trust of the tenants is an important factor in the growth of this technology.

Energy Consumption

The energy consumption is of significant importance to microlocation enabled technologies. In case of microlocation enabling devices, the energy consumption increases with the number of devices (e.g. a large office building). Thus they have to be as efficient as possible and have to scale their individual transmission power and time period to further reduce energy costs. In case of user devices, the main problem lies in the limited battery capacity of the device.

Security

The devices involved in microlocation have to be cost efficient and minimize power usage. Most of these devices are unattended, which increases the challenge for security. However, small and simple devices could lack the required computation power and memory storage to utilize security protocols and authentication mechanisms. The used security mechanism has to be adaptable, due to the large number of possible endpoints. New security protocols and methods used within microlocation enabled technologies have to pay special attention to energy-constrained devices as well as to be reliable and quick.

Artificial Intelligence-Enabled Devices

One of the envisioned microlocation enabled services are smart devices. A smart device will use microlocation data to efficiently fulfil its assigned task. Furthermore, it will use available contextual information to increase the reliability of the provided services. In order to implement these features, artificial intelligence algorithms can be utilized. Such algorithms have to be lightweight due to the lack of computational power of some devices, as well as cost efficient.

2.3.4 Position Measurement Systems

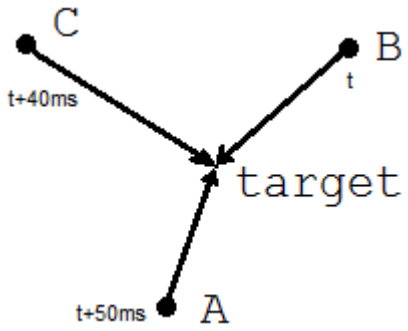
This section focuses on the different forms of representations of a location. A location can be defined by its relative position to other known objects, by one- up to three-dimensional coordinates, by a reference grid for all located objects, or by a symbolic location in a smart building, e.g. first floor. There are two techniques to estimate the position of a target.

Triangulation

The triangulation technique uses three different dimensions to estimate the position of a target. There are two different types of triangulation: Lateration and angulation. Lateration, also known under the term range measurement, estimates the position by measuring the distance to known reference points. Angulation estimates the targets location by computing its relative angles to known reference points.

There are five different triangulation techniques to calculate the targets distance:

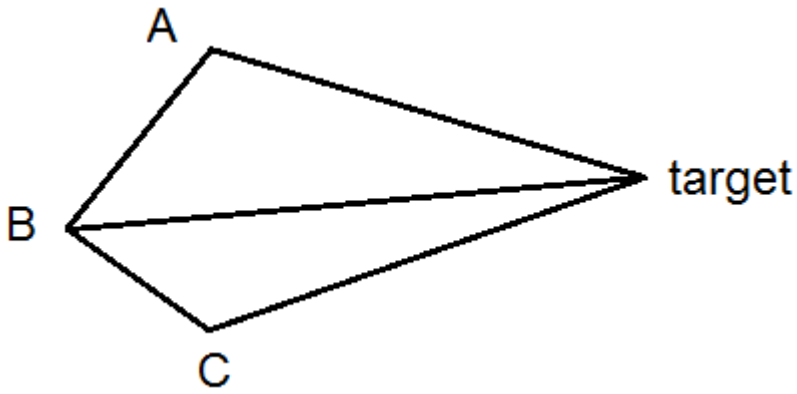
Figure 2.3: TOA example



Time of arrival Time of Arrival (TOA) calculates the distance between the target and the reference points using the propagation time (see Figure 2.3). With this method, there have to be at least three reference points to locate a target in a two-dimensional environment. As seen in Figure 2.3, the different propagation times from the reference points to the target are used to get the corresponding distance. The timestamp of a reference point (e.g. A) gets sent along with the transmitted signal, so that the target knows about the sender's local time at the time of sending. [23] provides a more in depth look about the different approaches to calculate the target's position utilizing this information.

The drawback of this method is that each point has to be precisely time synchronized with each other to get correct distance and a timestamp has to be sent as well. There are a few methods to calculate the position of the target. A simple approach would be a geometric approach to calculate the intersection points of the TOA circles. Another would be to use the least-squares algorithm.

Figure 2.4: TDOA example



Time difference of arrival Time difference of Arrival (TDOA) is a method that uses the time difference at which a specific signal arrived, measured at various measurement points. For a single measurement, the target has to be on the hyperboloid with a constant range difference to two measuring points. [24]

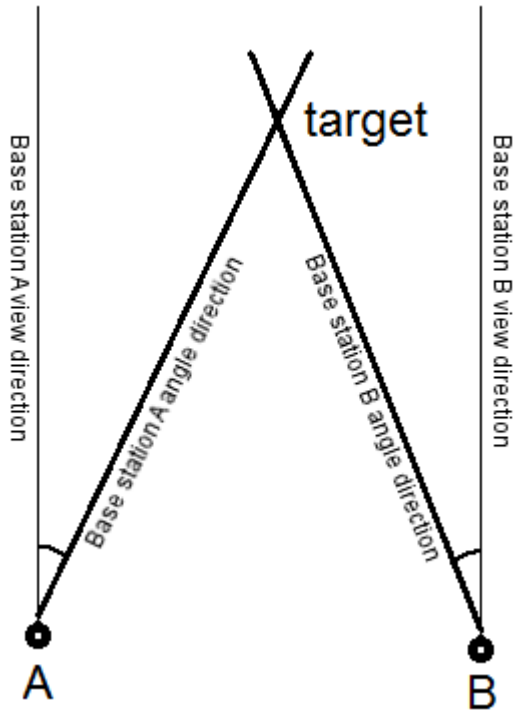
Figure 2.4 shows how the location of the target is estimated in a two dimensional environment through the intersection of two or more TDOA measurements. The TDOA measurements at the three points (A, B and C) form two different hyperboloids that can be used to locate the target. The resulting hyperboloid equation can be solved with Taylor-series expansion. A more in depth look on the hyperboloids calculation is described in [23].

Received signal strength (Rss-based) Received Signal Strength (RSS) is a method that tries to calculate the distance to the target using the emitted signal strength. This is an alternative to the TOA and the TDOA approach, because they are both affected by the multipath effect.
Multipath is a propagation phenomenon in which a radio signal reaches the receiver by two or more paths. Indoors this can be caused by reflection on metal bodies or other radio wave reflecting surfaces. Furthermore, there is shadowing, which is caused by objects that are obstructing the propagation path between transmitter and receiver, which leads to a fluctuation of the signal power.
These two effects result in a less accurate estimated position of the target, especially in indoor environments. The RSS-based method aims at calculating the signal strength loss due to the propagation path. However, this method is affected by extreme shadowing and multipath fading inside buildings. One method to enhance the accuracy of the results is to use premeasured values of RSS outlines that are centered on the transceiver. Another approach is to use multiple measurements at different stations as well as fuzzy logic algorithms.

Round trip of flight Round trip of flight (RTOF) uses a similar approach as TOA. It calculates the distance to a target with multiple known reference points. This is done by a measuring unit, sending a signal to the target which then reflects this signal back to the sender along with a timestamp. This timestamp then is used to calculate the targets distance. Therefore, time synchronization is not as critical as it is with TOA. The drawback is that this approach is affected by the processing time or delay of the target to reflect the signal. The same positioning algorithms for TOA can be used for RTOF.

Angulation The second type of triangulation is called angulation. In order to calculate the location by angulation, a technique called Angle of Arrival (AOA) can be used. The position is obtained by a pair of angle directions lines which will intersect at some point, as illustrated in Figure 2.5. The angle gets measured on a base station. In a two dimensional environment, there have to be at least two such stations to estimate the position of the target. In a three dimensional environment, this technique needs a

Figure 2.5: AOA example



minimum of three measuring units. Another advantage is, that AOA does not require time synchronization between the stations but it requires large and complex hardware, because AOA needs multiple antenna arrays or mechanically-agile directional antennas and if the target moves away from the station, the accuracy drops. This happens due to the fact that an increase in distance to the transmitter raises the probability for reflections from surrounding objects.

Proximity

The second technique is the proximity-based algorithm. It maintains a dense grid of antennas, and by associating every target to the closest antenna, whose position is known, it can provide information about the relative position. If more than one antenna detects a target, the closest antenna is denoted by the strongest signal strength. This algorithm is relatively easy to implement and is often used in conjunction with RFID or Infrared (IR) systems.

2.3.5 Microlocation Enabling Technologies

In order to use microlocation effectively, there are a two important criteria, which have to be met. The technology has to be accurate enough to locate users inside a building

and the overall energy consumption should be as low as possible.

Bluetooth

Bluetooth Version 4, also known as Bluetooth Low Energy (BLE) is the newest iteration of the Bluetooth standard. It defines optimizations to reduce energy consumption. On one hand, by reducing the time to initialize a connection between devices. On the other hand by implementing sleep cycles between synchronous send cycles. Such a BLE device can either be a master or a slave. Whereas a master can be connected to various slaves, a slave can only be connected to one master. Due to its efficient energy usage, BLE is very attractive for small devices, which are known as iBeacons. Introduced by Apple, the iBeacon protocol enables BLE-enabled devices to detect their proximity to iBeacons. An iBeacon device sends a periodically beacon, which can be picked up by a BLE-enabled device to approximate its position within an area. This iBeacon can be powered by any coin cell battery with possible yearlong runtime, which makes them very useful for microlocation.

To further increase the usage of iBeacons, Apple has standardized the iBeacon advertisement format. For a clear identification between different iBeacons, there is a Universally Unique Identifier (UUID) which is 16 bytes long, has an optional major value with the size of 2 bytes, and a second optional 2 byte minor value. Together, they form the ID of an iBeacon, whereas the UUID stands for a company of a brand, the major value is used to differentiate between groups of iBeacons and the minor value is used to distinguish or identify a beacon within a group of iBeacons. Furthermore, iBeacons perform two tasks: Distance measurement and ranging. Distance measurement is used to measure the distance to a particular beacon. For this, the BLE-enabled device uses the Received Signal Strength Indicator (RSSI). This value provides the device with the proximity of the iBeacon as well as the accuracy of the measurement. The ranging task supplies the device with an indicator for the range between its own position and the iBeacon. There are four ranges: Immediate, Near, Far and Unknown. Immediate means very close, near is between 1 and 3 meters and far stands for a low accuracy of the estimated proximity.

In order to use Bluetooth in a location-based service aspect, the BLE-enabled receiver picks up the beacon and sends its UUID to the server or the cloud. Then the system associates the UUID to the relevant area or zone and sends back the particular information associated with the UUID. This can be used to enhance the consumer experience inside a mall or to provide other microlocation-based services in smart buildings. Furthermore, its accuracy is about 10cm and has a low energy consumption, which makes iBeacon suitable for microlocation purposes. The drawback on this technology is that the iBeacons have to be distributed across a room to reliably locate a user.

UWB-based

UWB is a radio technology that uses a large bandwidth (greater than 500 MHz). This means, that it provides a higher reliability due to the fact that there is a wide range of signals, each carried by different frequencies. Therefore, the probability that signals can go around obstacles increases and the power spectral density decreases because of the large number of frequencies. This decreases the interferences as well as the probability of interception. It has two different phases: Ranging and localization.

Ranging tries to estimate the distance or angles between two nodes. For this, it uses mainly the TOA technique (see Section 2.3.4), because of the good time domain resolution of UWB, which provides subcentimeter resolution due to the wide bandwidth. [25] gives a more in depth look into ranging.

The goal of localization is to estimate the position of the local device. Therefore, various methods can be used like the Nonlinear Least Square (NLS) estimator. The TOA-based localization estimation method that utilizes the NLS estimator is described in [25].

This technology is especially interesting for microlocation because it can provide accuracy as high as 10cm and has a low energy consumption. The drawback of this technology are the limited range of 10-20m as well as the high cost.

Wireless positioning systems

Primarily used for geofencing, Wireless Positioning Systems (WPS) use transformed radio frequency signal measurements to get estimated distances and angles. These are then used to calculate the location of the signal source by using for example TDOA and AOA (see Section 2.3.4). For indoor settings, Wireless (WiFi) Access Points (APs) are used. When deploying this technology, it's important to pay attention to privacy. A third party could violate a user's privacy by eavesdropping an AP or attacks by a man-in-the-middle attack. Due to the fact that the accuracy is around three meters and it has a high energy consumption, it is not suitable for microlocation.

Magnetic Field mapping

Magnetic field mapping uses magnetic fields inside a building to locate the user or an object [26]. The most modern mobile phones are equipped with sensors that can sense this field and provide accurate positioning data. Unlike other technologies, there are far fewer privacy concerns as well as low costs and low power consumption. This makes this technology suitable for microlocation, although its accuracy is ranging from 20cm to 2 meters.

Radio Frequency Identification

RFID uses electromagnetic fields to exchange data between RFID-enabled devices. The RFID system consists of three basic components: RFID readers, RFID tags and the communication between those two. The communication uses specialized protocols and frequencies for data reception and data transmission. RFID readers will read data that is sent by RFID tags. There are two different types of tags, active tags, which are equipped with a battery as well as radio transceivers to boost the range, and passive tags. Passive tags reflect the received RF signal emitted from readers by adding information through modulation. Thus, they need no battery to operate which makes them cheaper and lighter than active tags. Although the range is limited to a few meters, RFID can be used together with other technologies such as WiFi to further increase the range of applications. Its accuracy is as high as 20-30cm which makes it a very suitable technology for microlocation. Although, the usage of RFID is limited by its maximum range if it is not combined with WiFi. If combined with WiFi, the energy costs of RFID are high.

GPS

GPS uses 24 orbital satellites to locate the receiver (e.g. a smart phone) [27]. Each satellite orbits the earth two times a day, so that every spot on earth can reach at least four satellites at any given time. Each satellite is equipped with a sender, a receiver, a few antennas, oscillators, microprocessors, solar panels and a driving system. Primary, a satellite sends signals which can be received by a GPS-receiver on earth. The main problem with GPS for microlocation is that it only works outdoors. Even then, its accuracy is only around 10 meters, which is not suitable for microlocation enabled services. Furthermore, GPS has a high energy consumption.

QR-Code

QR-codes are two dimensional codes that are based on a square matrix [28]. They basically consist of white and black squares, which represent binary-coded data. Special marks on three out of the four corners are used for orientation. These marks are getting scanned or photographed by a camera and then are decoded to get the encoded data. The built-in error correction mechanism enables QR-codes to get decoded correctly, even if 30-percent of the code is unreadable. It is even possible to distort the image of the code to a certain degree and a QR-code reader will still be able to read the encoded message successfully. Even though there are many different applications, the encoding of text works approximately the same. First the application has to evaluate the size of the code by the length of the text. Then the special marks for orientation are getting drawn on the empty square. After this, the text is encoded into a sequence of bits and then another sequence for error-correction is generated. Then the encoded string gets drawn onto the square from right to left in a zigzag pattern. A mask is drawn above the code to make an easy guessing of the code difficult. The number of the mask gets drawn into the symbols that are used for orientation.

For microlocation enabled services, this technology is suitable and conditionally useful. In order to locate a tenant anywhere, he has to scan the QR-code and send the data to the system. The system then identifies the encoded location in the QR-code, which results in an absolute position of the user. Apart from the power usage of the camera, there are no further energy costs involved.

Risks arise with QR-codes due to the fact that it can carry executable data in form of Uniform Resource Locators (URLs). As soon as a scanner reads the QR-code many phone applications browse to the encoded URL and if the URL hosts malicious code, the user's device can get compromised.

CHAPTER 3

Design

In this chapter, the requirements and technologies for a viable solution to the location-based commissioning of KNX installations are elaborated.
A user-friendly application should abstract the complex and interconnected systems that are necessary to enable configuration of KNX-based installations. In order to accomplish a natural way for a user to re-assemble KNX functionality, there are a few essential requirements. First of all, there has to be an existing KNX system that is connected to a central controller. Within the scope of this work, the controller was implemented by a middleware, which provides connectivity to the KNX system. The necessary requirements for this middleware are discussed in Section 3.1.

Another important enablement factor for a natural way to setup new relationships between KNX devices is the ability to determine the user's location.
Users inside a big smart building, such as an office building, with many different KNX-devices have to search for devices. In many cases, this will take a while due to the high amount of devices. Aggrevating this situation, users may forget or overlook devices. An example is a KNX-enabled conference room. Users could configure a switch to dim the lights accordingly to the preferences of the upcoming presentation. Additionally, selected lights can be used to support the presentation in form of highlighting products on the table. Without microlocation, this would be a tedious work and while doing so, users may miss one or more of the lights in the room and would have to do the configuration again.
In order to solve this problem, the search for the devices can be handled by using microlocation (see Section 2.3). The application to group the devices (see Section 4.2) will get the position of the user and only let the user configure devices which are near to the user's current location inside the building.
In connection to the above example, users would be able to reliably get all the lights inside the conference room and then configure them, without the need to explicitly search

for them in the first place.
This increases the usability of the KNX system as well as it enables users to setup new groups of devices as fast as possible. Within the proof of concept (see Chapter 4), microlocation was implemented by using QR-codes (see Section 2.3.5). This allows the application to be used on every device with the ability to scan a QR-code.
A key requirement for the proof of concept is lightweight communication between the application and the middleware. Due to the restriction by the middleware, three different protocols can be used for the proof of concept. The details to them and the final choice are elaborated in Section 3.3.

KNX was chosen due to the high interoperability of devices from different vendors, free topology choice and the ability to address devices in groups (see Section 2.2.2).
The next section discusses further reasoning behind the choice of the middleware.

3.1 Requirements for the Middleware

The middleware should be able to address KNX devices during runtime and provide access to to users. Furthermore, it should provide connectors to state of the art communication protocols.
In [29], the author describes the integration of building automation systems by using RESTful BACnet Web services.
BACnet provides a solution for building automation and control systems. It consists out of four layers and is not bound to any specific network medium. Therefore, any physical/data link layer combination can be used.
By the use of the BACnet object model, users can access devices by using the Representational State Transfer (REST) interface of the BACnet/WS server. For the proof of concept, this approach is not suitable, due to the fact that it currently does not support the setup of memberships of groups of devices during runtime.

EnTiMid is a middleware, which aims to offer an abstraction for high level services to access physical devices. It was developed to provide a universal middleware for building automation protocols. It is described in [30]. Due to the fact, that this middleware is not open source and publicly available, it was not considered for the proof of concept.

IoTSyS is an open source middleware that provides interfaces for users to configure devices. Its feature set includes several technology adapters, such as KNX or BACnet. In order to communicate with this middleware, several communication protocols are supported. Furthermore, IoTSyS offers important non-functional features, such as scalability and security. Due to the fact, that no other open source software provides means of defining groups during runtime, IoTSyS got used for the implementation of the proof of concept. The next section describes IoTSyS in more detail.

IoTSyS

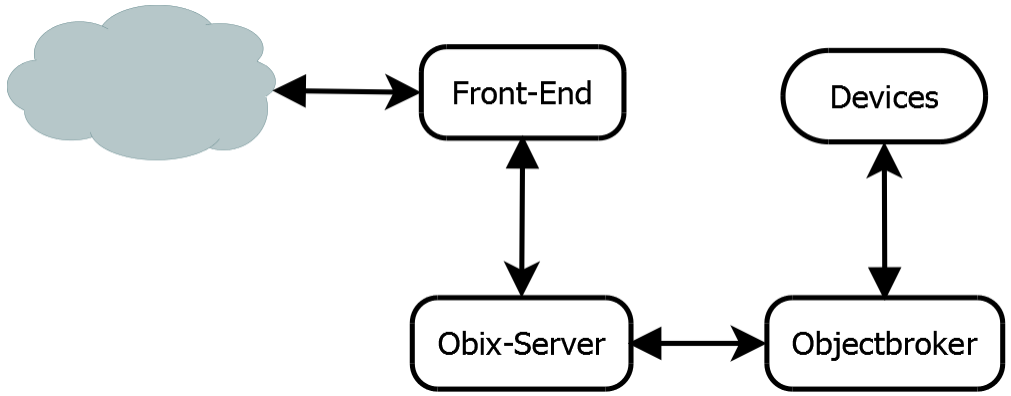
The middleware IoTSyS (see [20] and [31]) provides a communication stack for embedded devices based on IPv6, oBIX (see Section 3.2) and web services in order to provide interoperable interfaces for smart objects. The project was started within the frame of the FP7 IoT6 [32] European research project and is maintained by the Automation Systems Group at the Vienna University of Technology.

As already stated, IoTSyS provides different interfaces in order to enable clients to configure smart devices (based on e.g. KNX, BACnet or EnOcean) during runtime. From a users' perspective, those devices are presented in a uniform way.

The proof of concept uses Open Building Information Xchange (oBIX) contracts (see Section 3.2) to represent standard devices types. Furthermore, IoTSyS supports various protocol bindings for oBIX, such as CoAP (see Section 3.3) and offers eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) encodings. Each device has its own IPv6 address communication.

Figure 3.1 gives a rough overview of the IoTSyS architecture. The front-end is

Figure 3.1: IoTSyS architecture



connected to the local network or the Internet and handles incoming requests. It decodes the request and determines the requested resources path. This request is then passed to the oBIX-server, which converts the given payload into an oBIX object (done by the ObixDecoder). An object broker then finds and writes to the corresponding object. The object transforms this request into a write request to the device. After processing the changes, the object returns its new state, which is passed back along the chain to the front-end.

The proof of concept uses a mobile application for Android and a CoAP interface to communicate with the IoTSyS middleware. The IoTSyS project is hosted on [20].

3.2 Information Model

An information model is necessary to support interoperability between different systems. It is used to standardize the representation of entities in a certain domain, like KNX devices. An important aspect is the specification of common data types, that specify the format and encodings of datapoints. This enables the interworking between different devices and further ensures interoperability.
In order to decrease the complexity and performance requirements, the device interaction has to be stateless. Therefore, a datapoint-centric information representation approach has to be taken. For this approach, the interaction between devices is based on simple input and output datapoints offered by devices. As soon as a datapoint changes, the corresponding state of a device gets modified accordingly. Additionally, this approach supports interoperability between different devices. There are a few protocols and models, which must be considered for an information model.

The BACnet Web Service (BACnet/WS) was created to enrich the BACnet protocol with a web service interface. The standard is maintained by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). Its generic design allows it to be used for any kind of technology. It is based around a server/client architecture. One of its base elements is a hierarchy of *nodes*, whereas a *node* is described by attributes. Furthermore, BACnet/WS specifies a fixed set of standard attributes. Additionally, BACnet/WS offers a set of services, to provide read and write access to the node hierarchy. Therefore, the BACnet specification follows a RESTful web service approach. Further information can be found at [33].

The OPC Unified Architecture (OPC UA) is based around a server/client architecture. It exchanges message with a platform neutral binary message format and a Simple Object Access Protocol (SOAP)-based Web service protocol binding. It provides its functionality through services, called service sets. Each set allows access to a specific aspect of an OPC UA server. The data model allows modeling of complex data, as well as modeling of meta data. The stored information is represented hierarchically and is called address space. The information is stored in *nodes* and the references between them. In order to describe such *nodes*, a node has several attributes. The server has to provide the *Attribute Service Set*, which is a collection of services, to read or write data to such an attribute. OPC UA is a heavy-weight SOAP Web service, which is incompatible to the RESTful design paradigm, therefore unsuitable for a lightweight solution. [34] provides a more detailed look into OPC UA.

Another alternative is oBIX. It is an open web service interface for building automation systems. Due to the fact that it is platform independent and uses an object oriented approach, it is very flexible. The feature set includes contracts, which can be compared to templates for objects, and an abstraction of the datapoint concept within the automation system. It is accessed by using web services and follows the RESTful design paradigm.

Furthermore, oBIX has higher extensibility compared to other web service interfaces. The features of oBIX are suitable for the integration of state of the art communication technologies. Therefore, oBIX was chosen for the proof of concept implementation. The following section describes oBIX in more detail.

oBIX

The Open Building Information Exchange is an open XML- and Web services-based standard, especially designed for building control systems. It was standardized by the Organization for the Advancement of Structured Information Standards (OASIS) [35]. Its main purpose is to define a standard for communication between mechanical and electrical systems, and enterprise applications within a building. In order to achieve this, oBIX defines features for the following components: XML, networking, normalization and foundation. Due to their significance XML and Networking are further examined. Normalization aims to standardize a common representation of Machine-to-Machine (M2M) features such as datapoints (e.g. sensors or actuators) and alarms. Foundation means that oBIX's goal, is to lay down the groundwork for a common object model and XML syntax, which serves as a foundation for new specifications.

In order to represent information from M2M systems, oBIX implements a common XML-based syntax. It's based on a small, but extensible data model and maps to a simple and fixed XML syntax. In order to ensure extensibility, it introduces a concept called "contracts".
Contracts are basically templates, that are used to store common characteristics. These characteristics can be a name of a resource and a Uniform Resource Identifier (URI), where the system finds this resource. These contracts are standardized, so that they can easily be used in future functionality and other oBIX objects. In Java terminology, contracts would be the equivalent to interfaces. An example contract is shown in Listing 3.1. It defines a position, which is based on a QR-code.

Listing 3.1: oBIX contract example

```
<obj href="/con/Position">
        <real name="qrcodeEncoded" val="noqrcode"/>
</obj>
```

The basic oBIX object model uses a small amount of object types. An example for such a basic object is shown in Listing 3.2.

Listing 3.2: oBIX basic object example

```
<obj is="Position" href="/Switch1/position">
        <real name="qrcodeEncoded" val="switch1position"/>
</obj>
```

The object example shows how a contract is implemented in an oBIX object. The attribute "Position" is a reference to a contract, which is shown in Listing 3.1. The set

value "switch1position" is the encoded QR-code, which is used to identify the postion of the "Switch1". If the oBIX object does not define a value for the QR-code, it will get the predefined one from the referenced contract, which is in this case "noqrcode".

oBIX utilizes a client/server architecture to transfer encoded data. oBIX provides different encodings for the data, such as XML or JSON. The server hosts the data and the services and response to incoming requests. The client sends these requests, to gain access to the services and the stored objects. This architecture relies on three request and response types: Read, Write and Invoke. In order to request an oBIX object as document over the network, a client has to use the read request which contains its URI. A successful read request will result in a response, which contains the object to their full extent, whereas this information includes a tree of children down to references. If the read request was unsuccessful, the response is an "err" object (e.g. obix:BadUriErr). The write request overwrites the current state of existing objects. As with the read request, the write request has to contain the URI of an object in addition to its new state. If the write is successful, the response contains the object to their full extent as well. It is important, that the client checks the response, because the server is free to ignore the write request. An unsuccessful write will result in a response including an "err" object.

The invoke request needs a URI of an "op" object and an input argument object to trigger an operation. A successful invocation request results in a response which, contains the output object. If the request is unsuccessful, then just like the other two request types, the response encloses an "err" object. For the discovery of all URIs on an oBIX server, the server has to provide an object which implements obix:lobby. This object serves as an entry point for clients. All vendors should place their specific objects used for data and service discovery in the lobby. This is useful, because the client only needs to know the URI of the lobby and has access to the other objects.

Another important aspect of oBIX is the access to real-time information. In many cases there are resources (e.g. sensors) which change their data very fast. In order to distribute these changes to the clients, oBIX introduces a concept called watches. Watches are essentially objects, which are created by the client on the server's WatchService URI. The client then registers objects on this new watch and polls its URI, to get the events that have occurred since the last poll. oBIX supports another two important features, which are common in automation systems. The first feature enables oBIX to create a historical archive of datapoints over time, called history. This uses the obix:History contract to expose any object as standard history. The second feature is called alarming. It specifies a model to query, watch and acknowledge alarms.
All this features together make oBIX a very powerful tool to implement an automation system and to easily extend it as well.

3.3 Communication Protocols

In terms of protocol bindings, IoTSyS offers support for CoAP, the Hypertext Transfer Protocol (HTTP) and SOAP.

HTTP is a stateless application-level protocol, which provides a synchronous request/response model, specified by [36] and built upon the Transmission Control Protocol and the Internet Protocol (TCP/IP). The HTTP-request message refers always to a certain resource, which is identified by its URL. Furthermore, it contains message type information (e.g. GET) and can contain application data, which is called *body*. Additionally, the application metadata describes the preferred data type for the response (e.g. text/html). The HTTP-response message contains a status code (e.g. *200 OK*). Compared to other protocols, HTTP is reliable due to its TCP/IP basis, but also has a large overhead.

SOAP is a stateless, lightweight protocol, which uses XML to exchange information in a decentralized, distributed environment. In June 2003, SOAP became a W3C recommendation (see [37]). Two major design goals are simplicity and extensibility. SOAP uses HTTP as transport protocol, although it is capable of using other protocols (e.g. the Simple Mail Transfer Protocol (SMTP)). It is also designed to be platform and language independent. Due to the fact that it uses XML for data representation, the bandwidth usage is higher compared to other encodings. Furthermore, SOAP has a large overhead as well.

CoAP is primarily a one-to-one protocol for transferring state information between a client and a server. Due to the fact that CoAP uses User Datagram Protocol (UDP), the computational requirements of a stack implementation are reduced, compared to TCP/IP. Furthermore, if used in a constrained environment, UDP-based communication is the preferred choice. Additionally, CoAP provides service discovery as well as asynchronous push-based communication.

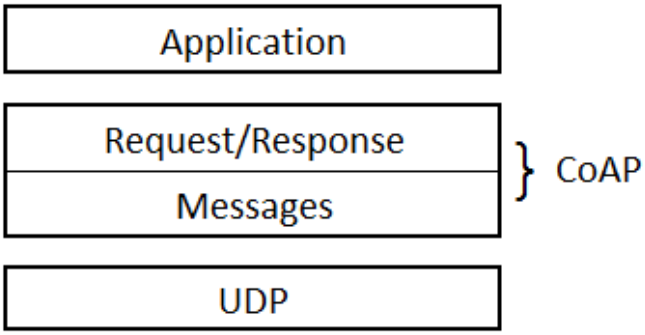
The stated reasons and the fact, that the mobile application should have low energy usage, CoAP was chosen as communication protocol between the middleware and the mobile application. Additionally, CoAP has a smaller message overhead compared to HTTP and SOAP. Furthermore, CoAP provides JSON encoding, which enables an efficient decoding on the mobile application. The following section gives a detailed look into this UDP-based protocol.

The Constrained Application Protocol (CoAP)

The Constrained Application Protocol is a lightweight UDP-based protocol used primarily for M2M communication and was standardized by the Internet Engineering Task Force (IETF) [38]. It is suitable for constrained nodes (microcontrollers with limited processing power and memory) and constrained networks, which often have a high packet error rate and low throughput. In order to reduce the risk of packet loss in these networks, one

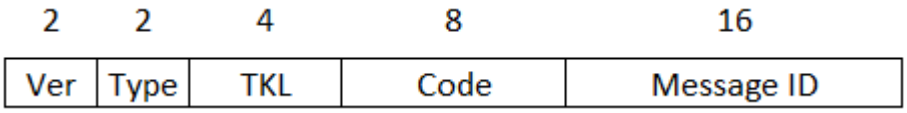
design goal, is to keep message overhead small, which limits the need for fragmentation. Besides the design goal to keep CoAP lightweight, it is also layed out to be as generic as possible. So it can be used in a wide range of use cases, especially for constrained environments. CoAP uses a RESTful approach for M2M applications with similar vocabulary as HTTP. Additionally, CoAP offers multicast support and asynchronous message exchange, as well as features for M2M, such as built-in discovery. In order to provide this, the Request/Response and asynchronous messaging are features of the CoAP Header, shown in the figure below.

Figure 3.2: CoAP layer



In contrast to HTTP, CoAP uses asynchronous UDP messages to exchange information. It is based on a client-server model and defines four message types: Confirmable, Non-confirmable, Acknowledgment and Reset. These four types of messages are defined in the type field of the header, shown in Figure 3.3.

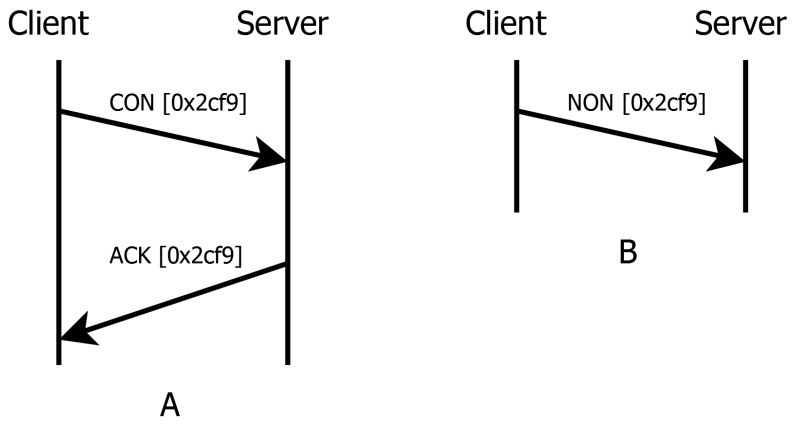
Figure 3.3: CoAP message format header



The version (VER) field indicates the CoAP version number. For Version 1.0 this 2-bit integer values gets set to 10. The type field indicates the message type (Confirmable, Non-Confirmable, Acknowledgment or Reset). The token length (TKL), which follows after the type field, stands for the variable-length token field. It is a 4-bit long integer. The token value is used to correlate request and response and follows optionally after the

header. The Code field indicates the code of the message. In case of a request, the code field indicates the Request method (e.g. GET); in case of a response, a Response code (e.g. 2.01 for "created"). It is split into two parts, the first 3-bit class range from 0 to 7 and the 5-bit class from 00 to 31. A code could be a request code or a response code (e.g. 2.05 for a success and 4.01 for an error response). The message ID field is used to detect message duplication and to match corresponding requests and responses together.

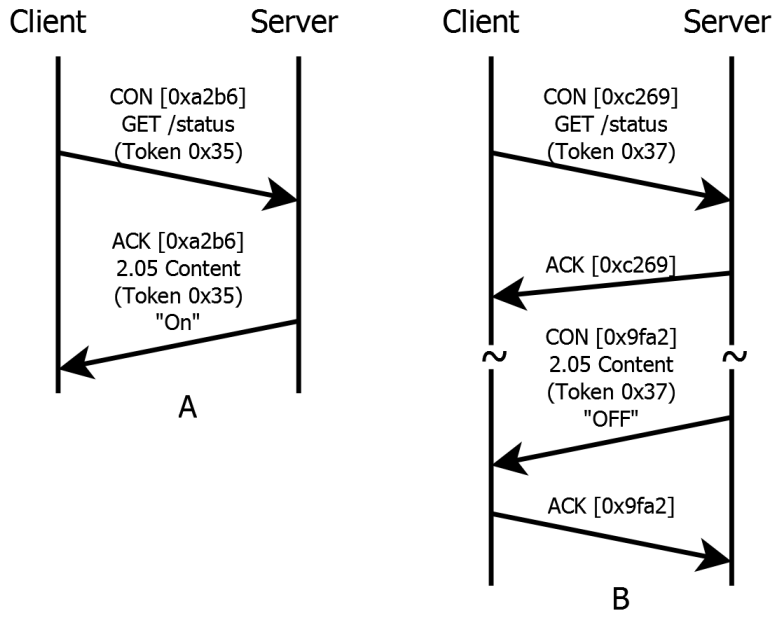
Figure 3.4: CoAP message types



The message exchange happens between a client and a server, which are both endpoints. In Figure 3.4 example A, the client sends a Confirmable message and the server responses with an acknowledgment message. Every message has to contain a Message ID (in this case 0x2cf9), which is used to identify the corresponding response. In order to initiate a reliable transmission, the message has to be marked as Confirmable. This message requires a response from the server, either an acknowledgment or a rejection. If the sender does not get a response, it will retransmit the request at increasing intervals until it receives a response or runs out of attempts. The amount of attempts (MAX_RETRANSMIT) can be freely adjusted, although it is recommended, that it is not too small. In case of a Non-Confirmable message, example B, the client sets according type in the type field of the CoAP message header, and appends a message ID for duplicate detection. In case that a recipient is not able to process the message, it can reply with a reset message to inform the client.

In order to request data by using CoAP messages, the client has to set the code field in the CoAP header. As seen in the Figure 3.5 example A, where the method code is set to GET. The Confirmable message includes the necessary request information, to inform the server about the type of the response. In the mentioned example, the client wants to know the status of a device. The token is used to identify the response separately to the message ID. As depicted by Figure 3.5 example B, where the client can identify the second (asynchronous) response from the server as the reply to the request.

Figure 3.5: CoAP message type with code field and asynchronous message type



The other three methods are similar to the methods in HTTP. POST is used to create a new resource or update an existing one. If a new resource has been created successfully, the response contains the URI to the new resource, as well as the 2.01 ("created") response code. If the resource is changed, the response is 2.04 ("changed"). If the request leads to deleting a resource, the server response with a 2.02 ("deleted") response. This is the same behavior as when the client calls the DELETE method. PUT updates or creates a new resource. If the resource already existed at the requested URI, the response results in a 2.04 ("changed") response code. As defined in the standard, a request messages can either be Confirmable or Non-Confirmable. Response messages can be Confirmable, Non-Confirmable or Acknowledged. An empty Request/Response header is called a reset message. A reset message is sent, if information to process the request is missing. This is usually seen after a reboot of an endpoint, because this endpoint would then lose its state.

Due to the fact that CoAP uses UDP, messages are transported in an unreliable way. Thus, the standard defines a lightweight feature set to reduce the retransmission risk for messages: A simple stop-and-wait function, as well as duplication detection. The stop-and-wait function uses the Confirmable flag for retransmission reliability. In order to handle message duplication, a server has to acknowledge each received duplicated message but only delivers one. This ensures that the client won't send any more repeated messages to the server.

A useful feature of CoAP is the caching of responses. Caching is designed to minimize network bandwidth usage and response time. It uses previously sent responses in conjunction with two mechanisms: Freshness and validation. Freshness determines an expiration time to ensure, that a cached response is still valid, which means that it can be reused. Validation is executed after the server gets a GET request with the ETag option set. The server then will choose a stored response, if it has stored more than one for this GET request, and update the freshness of it.

Due to the fact, that CoAP uses specialized HTTP methods, CoAP supports easy mapping to HTTP and vice versa. In order to identify servers, endpoints and resources in a network, a client can use a discovery mechanism. A server has to listen on incoming requests on the standard port (5683). In order to simplify the discovery, the client, as well as the server, can use CoAP's UDP-Multicast. A multicast request has to be non-confirmable and is identifiable by its multicast address.

CHAPTER 4

Proof of Concept

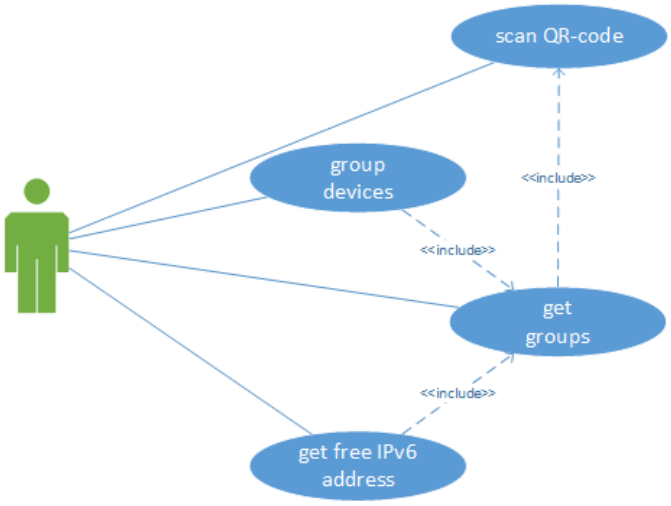
For the purpose of demonstrating the concepts mentioned in the previous chapters, a proof of concept was conducted by utilizing the open source middleware IoTSyS. The idea was to demonstrate that the user can group devices of a KNX system easy and fast. Furthermore, the user should only be able to access devices that are near him based on microlocation. As already stated in the Chapter 3, microlocating was implemented by using QR-codes representing a device. The developed mobile application (see Section 4.2) enables the user to connect to the IoTSyS-middleware, which manages the KNX devices and create new device groups based on the scanned QR-codes.

An introductionary use case is as follows: The user starts the application, scans all the QR-codes on the devices which he wants to group together and send those devices to the middleware. The middleware then adds and stores these devices to a new group. As soon as a device is triggered within the group, the middleware is aware that this device is within a group and sends the state change to all other members of the group. Figure 4.1 shows the use case diagram for the Mobile Configurator for Building Automation Systems (MCBAS).

4.1 Implementation

Support for localization was added to IoTSyS, as its functionality didn't support the localization of devices. Simulated devices were added by creating an entry in an XML configuration file, called "Devices.xml". In this file, all the devices with their properties are listed and defined.

Figure 4.1: Use case diagram



Listing 4.1: IoTSyS basic virtual device example

```
<device>
  <type>at.ac.tuwien.auto.iotsys.commons.obix.objects.iot.
        actuators.impl.LightSwitchActuatorImpl
  </type>
  <href>virtualLight</href>
  <historyEnabled>false</historyEnabled>
  <historyCount>10</historyCount>
  <groupCommEnabled>true</groupCommEnabled>
</device>
```

In order to enable the functionality of devices, the oBIX object broker within the middleware has to implement a basic contract about the devices listed in "Device.xml" (see Listing 4.1). In Listing 4.2, the matching contract for the device described in Listing 4.1 is shown.

Listing 4.2: IoTSyS contract example

```
package at.ac.tuwien.auto.iotsys.commons.obix.objects.iot.actuators;

import obix.Bool;

public interface LightSwitchActuator extends Actuator {
        public static final String CONTRACT="iot:LightSwitchActuator";

        public static final String VALUE_CONTRACT_NAME = "value";
        public static final String VALUE_CONTRACT_HREF = "value";

        public static final String switchContract = "<bool name='" +
            VALUE_CONTRACT_NAME + "' href='" + VALUE_CONTRACT_HREF + "'
            val='false'/>";
        public Bool value();
}
```

The middleware implements this contract and is able to manage the device. In case of a real device, e.g. a KNX device, the XML-file contains the specific address for this device.

In order to add localization to the devices, the location property was added to the XML-file without changing the contract. Only the "Devices.xml"-file has to be adapted, in order to support microlocationing. An example is shown in Listing 4.3.

Listing 4.3: IoTSyS basic virtual device with location example

```
<device>
  <type>at.ac.tuwien.auto.iotsys.commons.obix.objects.iot.
        actuators.impl.LightSwitchActuatorImpl
  </type>
  <href>virtualLight</href>
  <historyEnabled>false</historyEnabled>
  <historyCount>10</historyCount>
  <groupCommEnabled>true</groupCommEnabled>
  <location>virtualLightQR</location>
</device>
```

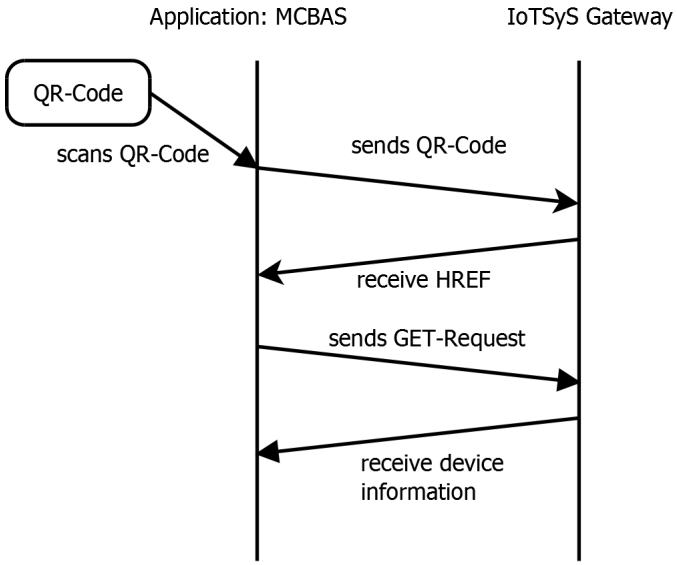
The location tag encloses the QR-code, which is added to the object inside the object broker of IoTSyS. This enables the gateway, which is located on the front-end of IoTSyS, to provide information about the devices location (QR-code).

The communication between the mobile Android application MCBAS and IoTSyS is done using CoAP. After scanning the QR-code of a device, the application sends this code in form of a POST-request (as payload) to a service endpoint on the server (e.g. coap://localhost/qrcode). The functionality to support this service got added to IoTSyS. The server iterates over all devices and returns the link (e.g. VirtualDevices/virtualLight) to the device with the corresponding QR-code. Afterwards the application sends a GET-request to the server (e.g. coap://localhost/VirtualDevices/virtualLight) and gets as response the device informations encoded in JSON. The application then converts this JSON into a useable format (a "JSONObject") and is able to present it to the user. The message flow is shown in Figure 4.2 and in Figure 4.3 as sequence diagram.

After the user scanned and confirmed all devices, the application sends the link of each device with a grouping command to the server
(e.g. coap://Server-Address/DeviceGroup/device/DataPoint/groupComm/joinGroup). Additionally, a new IPv6 address must be added as payload to group the devices together. It is important to note, that each device can have multiple functions. For example, a sensor that provides multiple datapoints, such as temperature and light level. In order to distinguish these measurements, IoTSyS assigns each of these datapoints another groupComm value. This means, that one sensor with different functionality can be in different groups for each datapoint. For the application this means that it has to send one POST-request for each datapoint of a device. The structure of such a groupComm address is as follows: coap://Server-Address/DeviceGroup/device/DataPoint/groupComm/.

There are two possibilities to manage the group from the datapoint. With "joinGroup",

Figure 4.2: MCBAS basic message exchange



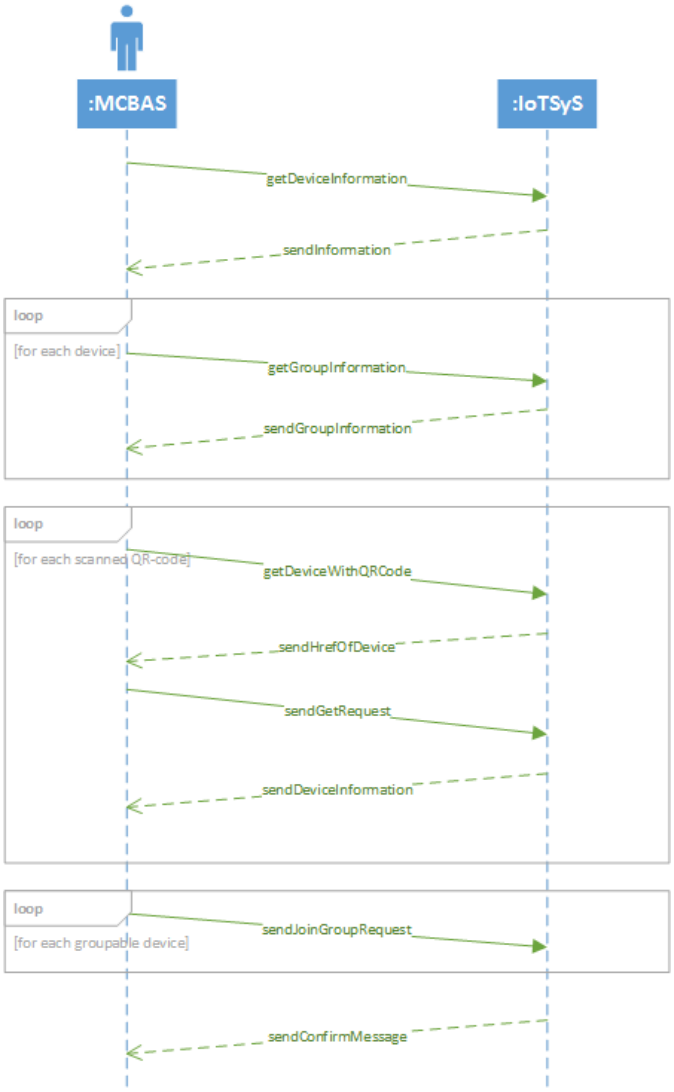
the device joins the group with the given IPv6 address in the payload of the request. With "leaveGroup", the device leaves the group denoted by the IPv6 address in the payload. In connection with the above examples, a joinGroup address could look like this: coap://localhost/VirtualDevices/virtualLight/value/groupComm/joinGroup.
Finally, all devices have been sent to the server and the application displays a confirm message.

4.2 MCBAS and IoTSyS groups

MCBAS is an Android application that was developed for the proof of concept application. This section provides interesting code snippets as well as in depth information about the group function ability of IoTSyS.

The existing IoTSyS implementation did not support the ability to list all groups used by the middleware. This means, that it is not possible to send a request and get all IPv6 addresses that are already allocated. The implementation of this feature would be beyond the scope of this work.
In order to work around this problem and to still get all used groups, a different approach has been taken.

Figure 4.3: Sequence diagram



Listing 4.4: IoTSyS VirtualLight GET XML-Response

```
<obj href="virtualLight/" is="iot:LightSwitchActuator"
   location="virtualLightQR">
  <bool name="value" href="virtualLight/value" val="false"
     displayName="On/Off" writable="true"/>
  <ref name="value groupComm"
     href="virtualLight/value/groupComm" is="iot:GroupComm"/>
</obj>
```

Each device provides information about its functionality, as well as it lists all data-

points that can be grouped. As seen in Listing 4.4, the attribute "value groupComm" is an identifier for the ability to group a value. In order to get information whereas this datapoint is already in a group, and if so in which one, the application has to send a new GET-Request to the server. Such a request can be formed like this: coap://localhost/VirtualDevices/virtualLight/value/groupComm.

The server response with XML or JSON is seen in Listing 4.5.

Listing 4.5: IoTSyS XML groupComm example

```
<obj name="groupComm" href="groupComm/" is="iot:GroupComm">
  <list name="groups" href="groupComm/groups" of="obix:str">
    <str val="FF15::15"/>
  </list>
  <op name="joinGroup" href="groupComm/joinGroup"
     in="obix:str" out="obix:list"/>
  <op name="leaveGroup" href="groupComm/leaveGroup"
     in="obix:str" out="obix:list"/>
</obj>
```

Within Listing 4.5, one address is already used. This address is seen as child element of the "groups" element: It is "<str val="FF15::15"/>".
In order to get all used addresses, the application has to send such a request for each datapoint of each device to the server and collect the results. Afterwards, it has to compute a free IPv6 address to ensure that no new group will be added to an existing one. For this, the application will convert the IPv6 addresses to a byte array and increase its value if necessary until a free IPv6 address is found.
Figure 4.4 shows the interaction overview diagram of the application.

The chapter closes with a few screenshots of MCBAS. The first set of screenshots (see Figure 4.5) illustrates the starting screen on the left, and the same screen after requesting all the available groups from IoTSyS. The second set of screenshots (see Figure 4.6), shows on the left all scanned devices with their repective HREF-identifier, and on the right the available datapoints, that can be grouped together.

Figure 4.4: Interaction overview diagram

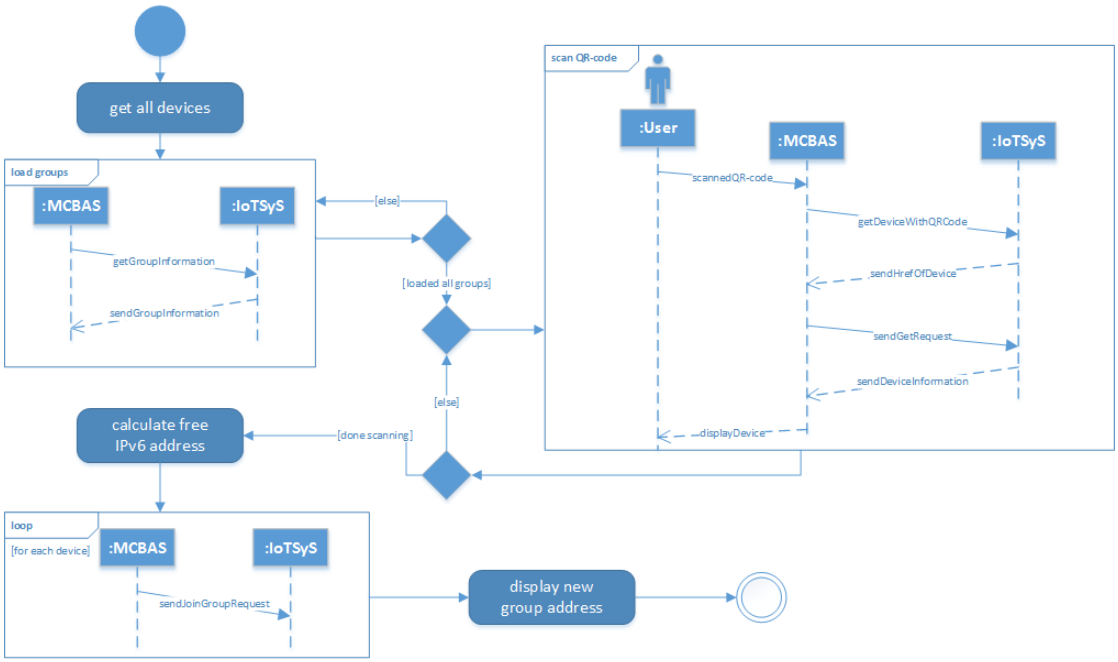


Figure 4.5: MCBAS screenshots first screen

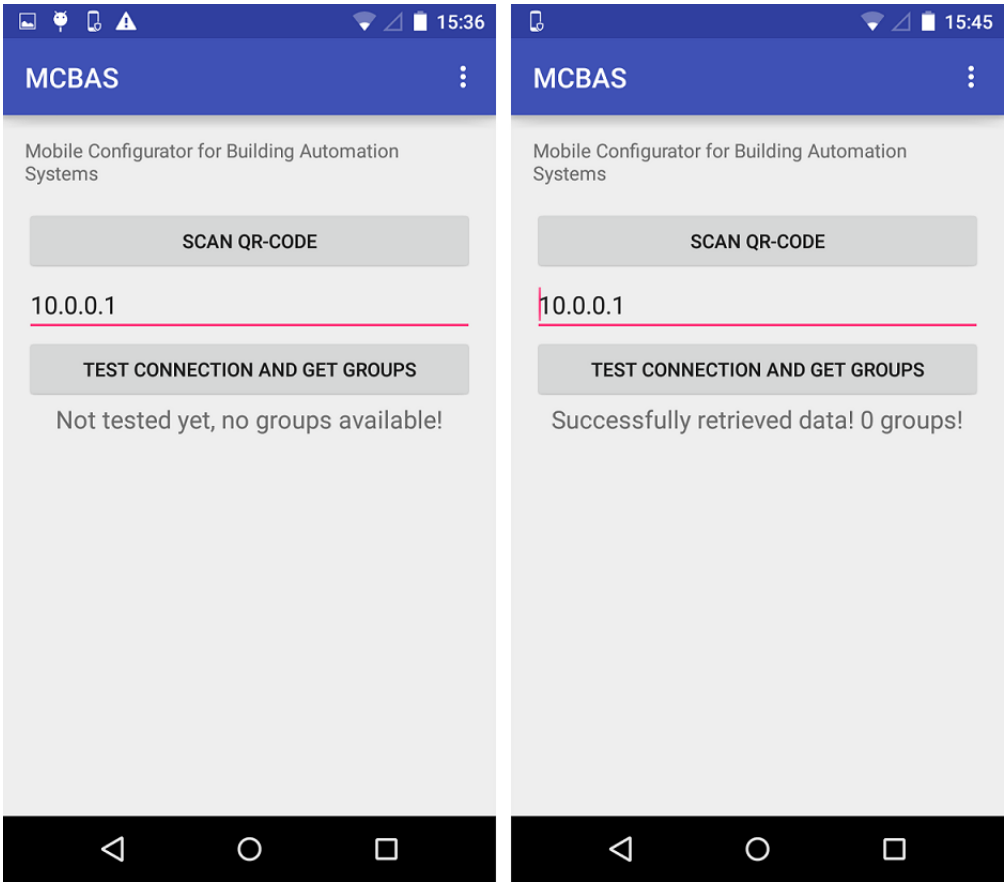
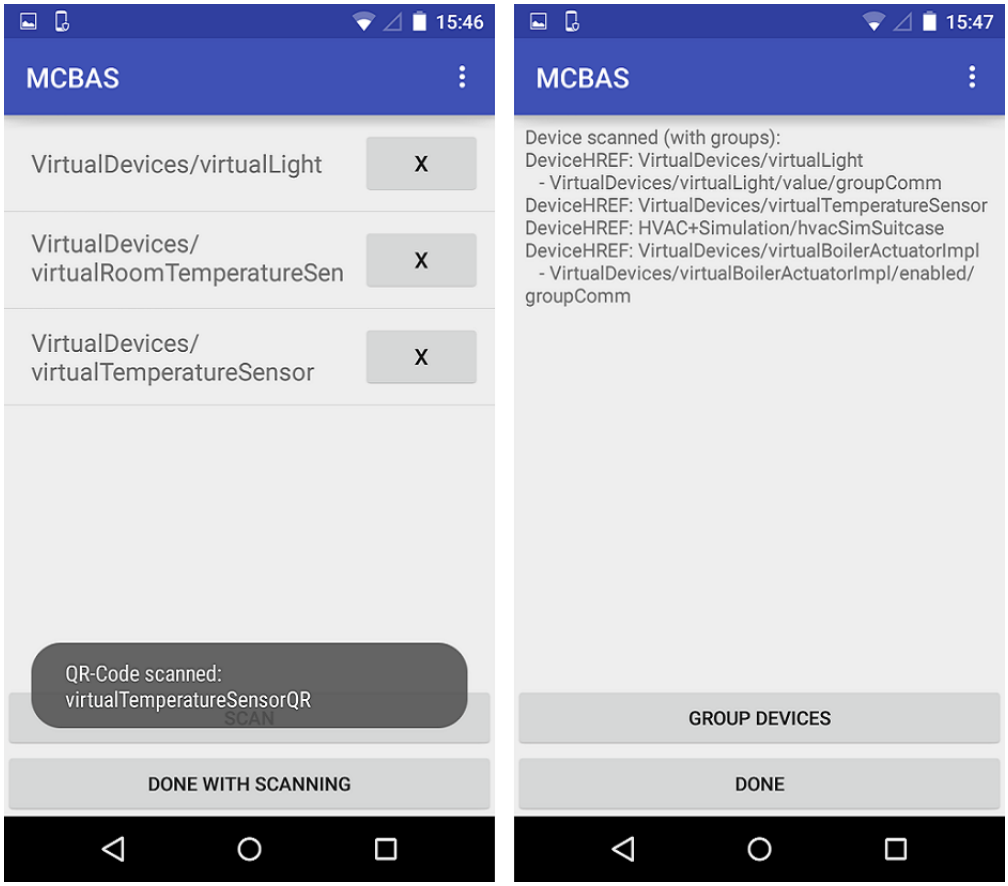


Figure 4.6: MCBAS screenshots second & third screen



CHAPTER 5

Summary and Future Work

This work proposed a solution to enable users to naturally group KNX devices that are detected by microlocation. Due to the release of IPv6, the IoT got enabled and increased the range of applications, as stated in Chapter 2.
Building automation protocols, such as KNX, which enabled efficient energy management were introduced and their usage in current building automation installations was described. Furthermore, the different technologies to enable microlocation were discussed and their drawbacks have been lined out. As stated in Section 2.3, GPS and WPS are not suitable for microlocation. Instead, other technologies, such as RFID-, UWB- or Bluetooth-based approaches are more suitable to locate tenants inside a building. There are a few position measurement systems that utilize different techniques to successfully locate users, as stated in Section 2.3.4. Some use cases of microlocation enabled services have been discussed, alongside challenges that arise with microlocation and possible solutions.
In order to utilize the proof of concept, the requirements and the required technologies have been stated in Chapter 3. Besides the need of existing infrastructure, a suitable middleware had to be used in order to communicate between the mobile application and the KNX system. The middleware maintains an abstraction of the KNX devices and enables the configuration at runtime. Different interfaces are provided and within the scope of the proof of concept, CoAP (see Section 3.3) was used to ensure lightweight communication. The management of the vast amount of devices at the middleware was implemented using oBIX (see Section 3.2).

In future work, the application can be extended to not only support the creation of new device groups, but also the ability to configure existing groups, as well as specific devices. However, new management procedures have to be introduced at the middleware in order to list all existing groups. Furthermore, there is potential to decrease bandwidth usage by integrating all existing groups into one response of the GET-request as shown in Listing 4.4.

The building automation is an evolving domain, which got recent attention through the emerge of the IoT. New standards are getting introduced, such as KNX WS, allowing for a broader range of applications and services. With the usage of microlocation, building automation provides efficient and comfort enhancing solutions for the users as well as for the industry.

List of Figures

Acronyms

AOA Angle of Arrival. 18, 19, 21

AP Access Point. 21

ASHRAE American Society of Heating, Refrigerating and Air-Conditioning Engineers. 28

BACnet Building Automation and Control Network. 9, 11, 26–28

BACnet/WS BACnet Web Service. 28

BLE Bluetooth Low Energy. 20

CoAP Constrained Application Protocol. vii, 27, 31–35, 39, 47

ETS Engineering Tool Software. 9, 10, 12

ETSI European Telecommunications Standards Institute. 7

GPS Global Position System. 13, 15, 22, 47

HTTP Hypertext Transfer Protocol. 31, 32, 34, 35

HVAC Heating, Ventilation and Air Conditioning. 8, 14

IETF Internet Engineering Task Force. 31

IoT Internet of Things. vii, 1–7, 13, 14, 47, 48

IPv6 Internet Protocol Version 6. vii, 3, 7, 27, 39, 40, 42, 47

IR Infrared. 19

ISO International Organization for Standardization. 7

ITS Intelligent Transport Systems. 5, 6

JSON JavaScript Object Notation. 27, 30, 31, 39, 42

LON Local Operating Network. 9–11

M2M Machine-to-Machine. 29, 31, 32

MCBAS Mobile Configurator for Building Automation Systems. 12, 37, 39, 40

NLS Nonlinear Least Square. 21

oBIX Open Building Information Xchange. 27–30, 38, 47

OPC UA OPC Unified Architecture. 28

REST Representational State Transfer. 26

RFID Radio-frequency identification. 6, 7, 19, 22, 47

RSS Received Signal Strength. 18

RSSI Received Signal Strength Indicator. 20

RTOF Round trip of flight. 18

SDK software development kit. 15

SLAM Simultaneous Localization And Mapping. 15

SMTP Simple Mail Transfer Protocol. 31

SOAP Simple Object Access Protocol. 28, 31

TCP Transmission Control Protocol. 7

TCP/IP Transmission Control Protocol and the Internet Protocol. 31

TDOA Time difference of Arrival. 18, 21

TOA Time of Arrival. 17, 18, 21

UDP User Datagram Protocol. 31, 32, 34, 35

URI Uniform Resource Identifier. 29, 30, 34

URL Uniform Resource Locator. 23, 31

UUID Universally Unique Identifier. 20

UWB Ultra-wideband. vii, 21, 47

WiFi Wireless. 21, 22

WLAN Wireless Local Area Network. 3, 15

WPS Wireless Positioning Systems. 21, 47

XML eXtensible Markup Language. 27, 29–31, 37, 39, 42

Bibliography

[1] D. Fasel and A. Meier, "Big data - grundlagen, systeme und nutzungspotenziale," *Wiesbaden. Springer Fachmedien Wiesbaden*, pp. 1–380, 2016.

[2] R. Stackowiak, V. Mantha, A. Licht, and L. Nagode, "Big data and the internet of things," *Apress*, pp. 1–220, 2015.

[3] "Institute for building efficiency." `http://www.buildingefficiencyinitiative.org/articles/what-smart-building`. Accessed: 31.08.2016.

[4] J. Ekanayake, K. Liyanage, J. Wu, A. Yokoyama, and N. Jenkins, "Smart grid - technology and applications," *Chichester - Wiley*, pp. 1–290, 2012.

[5] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges," *Computer vol. 44*, pp. 32–39, 2011.

[6] T. Nam and T. A. Pardo, "Smart city as urban innovation - focusing on management, policy, and context," *Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance, ACM, New York*, pp. 185–194, 2011.

[7] D. Washburn, U. Sindhu, S. Balaouras, R. A. Dines, N. M. Hayes, and L. E. Nelson, "Helping cios understand smart city initiatives - defining the smart city, its drivers, and the role of the cio," *Forrester Research*, pp. 1–17, 2010.

[8] A. Perallos, U. Hernandez-Jayo, E. Onieva, and J. I. Garcia-Zuazola, "Intelligent transport systems," *Wiley*, pp. 1–341, 2015.

[9] H. Kopetz, "Real-time systems: Design principles for distributed embedded applications," *Real-Time Systems Series*, pp. 307–323, 2011.

[10] L. Atzory, A. Iera, and G. Morabito, "Internet of things a survey," *Computer Networks vol. 54*, pp. 2787–2805, 2010.

[11] F. Wortmann and K. Fluechter, "Internet of things - technology and vaule added," *Business and Information Systems Engineering vol. 57*, pp. 221–224, 2015.

[12] Merz, Hermann and Hansemann, Thomas and Huebner, Christof, "Gebaeudeautomation - Kommunikationssysteme mit EIB/KNX, LON und BACnet," *Muenchen. Carl Hanser Verlag GmbH and Co. KG*, 2016.

[13] KNX Association, "EN 50090-1:2011: Home and Building Electronic Systems (HBES)," tech. rep., Cenelec, 2011.

[14] B. Voglmayr, "Middleware-konzepte fuer verteilte automatisierungssysteme basierend auf iec 61499," *Fakultaet fuer Elektrotechnik und Informationstechnikder Technischen Universitaet Wien*, 2007.

[15] "Loxone Website." `http://www.loxone.com`. Accessed: 17.09.2016.

[16] "Comfortclick Website." `https://www.comfortclick.com`. Accessed: 17.09.2016.

[17] "ETS Website." `https://www.knx.org/in/software/overview/index.php`. Accessed: 17.09.2016.

[18] "Loxone online documentation." `http://www.loxone.com/dede/service/dokumentation/loxone-config/eib-knx/grundlagen.html`. Accessed: 17.09.2016.

[19] "Comfortclick online documentation." `https://www.comfortclick.com/Software/Manuals/BOSConfig?print=false`. Accessed: 17.09.2016.

[20] "IoTSyS Github Page." `https://github.com/mjung85/iotsys`. Accessed: 18.09.2016.

[21] Faheem Zafari, Graduate Student Member, IEEE, Ioannis Papapanagiotou, Member, IEEE, and Konstantinos Christidis, Member, IEEE, "Microlocation for Internet-of-Things-Equipped Smart Buildings," *IEEE Internet of Things Journal, VOL. 3, No. 1*, 2016.

[22] U. Bareth, A. Kupper, and P. Ruppel, "geoxmart - a marketplace for geofence-based mobile services," *IEEE 34th Annual Computer Software and Applications Conference*, pp. 101–106, 2010.

[23] H. Hui Liu, P. Darabi, P. Banerjee, and P. Jing Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol.37(6), [Peer Reviewed Journal]*, pp. 1067–1008, 2007.

[24] B. T. Fang, "Simple solutions for hyperbolic and related position fixes," *IEEE Transactions on Aerospace and Electronic Systems Vol. 26, No. 5*, pp. 748–753, 1990.

[25] C.-C. Chong, I. Guvenc, F. Watanabe, and H. Inamura, "Ranging and localization by uwb radio for indoor lbs," *NTT DOCOMO Technical Journal Vol. 11 No. 1*, pp. 41–48, 2009.

[26] M. Frassl, M. Angermann, M. Lichtenstern, P. Robertson, B. J. Julian, and M. Doniec, "Magnetic maps of indoor environments for precise localization of legged and non-legged locomotion," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2013.

[27] Bernhard Haselgruebler, "Ueber die Qualitaet der Standortbestimmung fuer Location Based Services im Kontext der Entwicklung von Mobilfunk- und globalen Satellitennavigationssystemen," *Department fuer Geodaesie und Geoinformation, Forschungsgruppe Geoinformation, Technische Universitaet Wien*, 2015.

[28] ISO/IEC JTC 1/SC 31, "Information technology Automatic identification and data capture tchniques Bar code symbology QR Code," ISO 18004, ISO.org, June 2015. Accessed: 26.08.2016.

[29] Schachinger, Daniel and Stampfel, Christoph and Kastner, Wolfgang, "Interoperable integration of building automation systems using RESTful BACnet Web services," *Industrial Electronics Society, IECON 2015 - 41st Annual Conference of the IEEE*, 2015.

[30] Nain, Gregory and Daubert, Erwan and Barais, Olivier and Jezequel, Jean-Marc, "Using MDE to build a schizophrenic middleware for home/building automation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),Vol.5377, pp.49-61 [Peer Reviewed Journal]*, 2008.

[31] Markus Jung, "An integration middleware for the Internet of Things," *Fakultaet fuer Informatik der Technischen Universitaet Wien*, 2014.

[32] "FP7 European research project Homepage." `http://www.iot6.eu/`. Accessed: 08.10.2016.

[33] "American Society of Heating, Refrigerating and Air-Conditioning Engineers Homepage." `https://www.ashrae.org/home`. Accessed: 01.11.2016.

[34] "OPC Foundation Homepage." `https://opcfoundation.org/developer-tools/specifications-unified-architecture`. Accessed: 01.11.2016.

[35] "OASIS Open Building Information Exchange." `http://www.obix.org/obix.htm`. Accessed: 12.07.2016.

[36] "Hypertext Transfer Protocol (HTTP)," RFC 2616, RFC Editor, June 1999. Accessed: 01.11.2016.

[37] W. Recommendation, "SOAP Version 1.2 Part 0: Primer (Second Edition)," tech. rep., W3C, 2016. Accessed: 30.10.2016.

[38] Zach Shelby, Klaus Hartke and Carsten Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, RFC Editor, June 2014. Accessed: 12.07.2016.