Steuerung und Überwachung intelligenter Gebäudetechnik mit mobilen Endgeräten

Bachelorarbeit

zur Erlangung des akademischen Grades Bachelor of Science
an der Hochschule für Technik und Wirtschaft Berlin,
Fachbereich Wirtschaftswissenschaften II,
Studiengang Angewandte Informatik

vorgelegt von Björn Bittins
Matrikelnummer: 519448

1. Betreuer: Prof. Dr. Jürgen Sieck
2. Betreuer: Dr.-Ing. Michael Herzog

Berlin, den 4. Juni 2010

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Entwicklungen in der Computer- und Mobilfunktechnologie der letzten Jahre haben es ermöglicht, dass man heutzutage von fast jedem Ort aus erreichbar ist und fast jeden erreichen kann. Technische Geräte zur Kommunikation, Unterhaltung oder Steuerung sind mittlerweile so kompakt, dass sie ein elementarer Bestandteil unseres Lebens geworden, also allgegenwärtig sind. Laut Iftode, Borcea und anderen [IBR+04] wird noch nicht das volle Potenzial der mobil zur Verfügung stehenden Rechenleistung genutzt, denn "our pockets and bags are still jammed with a bunch of keys [...], access cards [and] credit cards [...]". Sie sind der Meinung, dass „Smartphones“die Geräteklasse sind, die die größte Chance haben eine „Universalfernbedienung“für die uns umgebenden Geräte zu werden und unsere Taschen wieder zu leeren, denn das Mobiltelefon ist "for some mysterious reason [...] the least likely to be left at home".

Zudem läuft „die Zeit der 'dummen Häuser'“ab [dpa09] und immer mehr Unterhaltungselektronik und Steuerungs- bzw. Überwachungtechnik zieht in unsere Häuser und Wohnungen ein. Licht, Heizung, Musikanlage und viele andere Geräte können heute über Rechner gesteuert werden. Dazu stehen verschiedene Bus-Technologien zur Verfügung, die es ermöglichen, Aktoren, und damit einzelne Geräte oder Gerätegruppen, im Haushalt anzusprechen.

Da die Technologien bereits verfügbar sind und der Mensch längst im Zeitalter der allgegenwärtigen Informationsverarbeitung, dem sogenannten „Ubiquitous Computing“, angekommen ist, dann erscheint es nur als logischer nächster Schritt, diese beiden Welten zu verbinden und moderne Haustechnik über „Smartphones“von fast jedem Ort aus zu steuern.

1.1 Motivation und Zielsetzung

Die Verbindung zwischen mobiler Welt und der Technik innerhalb der eigenen vier Wände ist keine neue Idee. Es existieren bereits Lösungen zur Steuerung von Gebäudetechnik mit mobilen Endgeräten. Die Integration dieser mobilen Endgeräte ist aber meist bei der Planung des Gesamtsystems vom Elektroinstallateur bzw. der Fachkraft zu berücksichtigen. Eine nachträgliche Installation durch den Endanwender gestaltet sich schwierig. Auch die Konfiguration der Anwendung auf dem mobilen Endgerät kann meist nur durch eine Fachkraft erfolgen. Meine Anwendung soll daher Wege aufzeigen, die eine Integration mobiler Endgeräte in ein Gebäudeautomationssystem auch nachträglich mit möglichst geringem Aufwand für Fachkräfte möglich machen.

Ziel dieser Arbeit ist eine Evaluierung der Möglichkeiten zur Integration eines mobilen Endgerätes in ein System zur Steuerung von intelligenter Gebäudetechnik. Es soll beispielhaft eine Anwendung zur Steuerung dieser Gebäudetechnik auf einem mobilen Endgerät entwickelt werden. Dazu werden Standards und Protokolle zur Vernetzung, Steuerung und Überwachung von Gebäude- und Haushaltstechnik untersucht. Der Schwerpunkt der Arbeit liegt in der Anbindung eines mobilen Endgerätes an ein Steuerungssystem für intelligente Gebäudetechnik. Der Konfigurationsaufwand der Anwendung soll für den Anwender möglichst gering und die Bedienung einfach und intuitiv gehalten sein. Die Anwendung soll beispielhaft die Steuerung und Überwachung einer Lichtquelle sowie das Auslesen eines Temperatursensors ermöglichen.

1.2 Aufbau der Arbeit

Im folgenden Abschnitt werden die grundlegenden Technologien, die potenziell zur Erstellung des Systems in Frage kommen, beschrieben. Anhand dieser Grundlagen wird ermittelt, welche Technologien und Geräte im Rahmen der Erstellung des Prototypen verwendet werden. Zudem werden bestehende Systeme für den mobilen Zugriff auf Gebäudesteuerungssysteme vorgestellt und Vor- und Nachteile dieser Anwendungen diskutiert. Dieser Überblick soll die Grundlage für die anschließende Anforderungsanalyse schaffen.

In Kapitel 3 wird die Anwendungsumgebung analysiert und darauf basierend Anforderungen an das System definiert. Es wird dabei auf die besonderen Gegebenheiten, die durch die Anbindung des Prototypen an ein Gebäude-

steuerungssystem zu beachten sind, eingegangen. Anwendungsfälle, die das System abbilden soll, werden identifiziert und definiert.

Basierend auf den zuvor spezifizierten Anforderungen wird in Kapitel 4 ein konzeptioneller Systementwurf erstellt. Hier wird insbesondere auf die Komponenten des Systems eingegangen sowie die Kommunikation zwischen mobilem Endgerät und dem Gebäudesteuerungssystem in Form von Schnittstellen und das Datenmodell spezifiziert. Zudem ist der Entwurf einer Benutzeroberfläche für die Anwendung des mobilen Endgerätes Teil dieses Kapitels.

Die technische Umsetzung des erarbeiteten Systementwurfs wird im Kapitel 5 beschrieben. Das Hauptaugenmerk liegt hierbei zum Einen auf der Art der Integration des mobilen Endgerätes in das Gebäudesteuerungssystem und zum Anderen auf der Implementierung der Smartphone-Anwendung.

Abschließend wird das entwickelte System in Hinblick auf die Erfüllung der Anforderungen hin untersucht und bewertet. Der Prototyp wird zudem in diesem Abschnitt demonstriert.

Kapitel 2

Grundlagen

In diesem Kapitel wird auf grundlegende Begriffe und Technologien eingegangen, welche elementare Bestandteile des Systems darstellen. Die hier besprochenen Grundlagen sind Voraussetzung für die Entwicklung eines Systems zur Steuerung und Überwachung intelligenter Gebäudetechnik mit mobilen Endgeräten.

Hierzu wird zunächst der Begriff 'Smartphone' definiert um diese Geräteklasse innerhalb der Klasse der mobilen Endgeräte genau abzugrenzen. Zudem werden die in diesem Bereich aktuell gängigen Plattformen und die für diese erhältlichen Endgeräte vorgestellt. Für die Auswahl des Endgerätes, das zur Umsetzung des Systems verwendet werden soll, spielen vor allem eine intuitive Bedienung sowie die Möglichkeit zur Kommunikation über Drahtlostechnologien eine übergeordnete Rolle.

Anschließend werden gängige drahtlose Kommunikationstechnologien beschrieben und auf die Möglichkeit zur Verwendung im System, zur Kommunikation zwischen mobilem Endgerät und dem Gateway des Systems zur Gebäudeautomation hin untersucht.

Zuletzt werden etablierte Technologien zur Gebäudeautomatisierung beschrieben und auf eine Eignung zur Verwendung mit dem zu entwickelnden System hin untersucht. Hier spielen vor allem die unkomplizierte Integration von mobilen Endgeräten in das Gebäudeautomationssystem und eine einfache Ansteuerung der Geräte in diesem System eine wichtige Rolle.

2.1 Smartphones

Smartphones (siehe Abbildung 2.1) sind mobile Endgeräte und können laut Iftode [IBR+04] oder auch Lin [LY09] als Kombination aus Mobiltelefon und PDA[1] gesehen werden. Nach Zheng und Ni [ZN06] weisen Smartphone charakteristische Eigenschaften, wie z.B. ein Farbdisplay, erweiterte drahtlose Kommunikationstechnologien, persistenten Speicher (externe Speicherkarten oder eingebaute Festplatten), ein fortschrittliches Betriebssystem (OS[2]) sowie andere Erweiterungen auf. Heutzutage gehören ein GPS-Empfänger und ein Beschleunigungssensor oder ein Gyroskop auch schon zur Standardausstattung der Smartphones der neuesten Generation.

Laut einer Studie von Gartner [Gar09] ist das Smartphone das mobile Endgerät mit den höchsten Wachstumsraten. Während der Absatz von Mobilfunktelefonen im zweiten Quartal 2009 weltweit um sechs Prozent zurück ging, konnte im Smartphone-Segment eine Absatzsteigerung von 27 Prozent verzeichnet werden. Gartner erklärt dies unter anderem mit dem Umstand, dass sich die Kunden, die sich überlicherweise im Segment der Mittelklasse der Mobilfunkgeräte umgesehen haben, nun auch verstärkt auf das höherpreisige Smartphone-Segment zurückgreifen um mehr Funktionalitäten für ihr Geld zu erhalten.

Weitere treibende Faktoren sind laut Gartner [Gar10b] [Gar10a] berührungsempfindliche Bildschirme (Touchscreen) und die Möglichkeit der Erweiterung der Funktionalitäten des Betriebssystems durch den Erwerb von Anwendungen (Applications, Apps) in sogenannten AppStores.

Die vielfältigen Kommunikationstechnologien wie z.B. UMTS , WLAN und Bluetooth , die eine nahezu lückenlose Konnektivität zu Online-Diensten oder den Datenaustausch zwischen Nutzern ermöglichen, machen diese Geräteklasse für eine immer größere Nutzergruppe attraktiv [Gün10].

Mobiltelefone begleiten uns in unserem täglichen Leben und insbesondere Smartphones zeichnen sich durch ihre Leistungsfähigkeit, einfache Bedienung durch große Bildschirme, meist sogar Touchscreens, und Erweiterbarkeit durch Anwendungen von Drittanbietern aus. Das macht Smartphones zu einer geeigneten Plattform zur Verwendung in dem zu entwickelnden System.

[1] Personal Digital Assistant.
[2] Operating System.

Abbildung 2.1: Beispiele aktueller Smartphone-Plattformen [HTC], [Nok], [Appe], [Mot]

Im Folgenden werden nun zwei Smartphone-Plattformen genauer betrachtet. Es handelt sich dabei um Google's Android und Apple's iPhoneOS, zwei relativ jungen Plattformen, denen viel Potenzial für die kommenden Jahre zugeschrieben wird. Zudem wird ein Überblick über weitere am Markt vertretene Smartphone-Plattformen gegeben.

2.1.1 Google Android

Am 21. Oktober 2008 wurde das Android Betriebssystem von Google offiziell vorgestellt. Es wird von der Open Handset Alliance, in der Google als Mitglied vertreten ist, weiterentwickelt. Bei Android handelt es sich laut Google um einen "[...] software stack for mobile devices that includes an operating system, middleware and key applications" [Goo10b]. Die Android-Plattform ist frei verfügbar und quelloffen. Für die Entwicklung von Anwendungen wird ein Eclipse[3]-Plugin und das auf Java SE[4] basierende Android SDK[5] zur Verfügung gestellt. Durch die Nutzung der Entwicklungsumgebung (IDE) Eclipse ist eine plattformübergreifende Entwicklung von Anwendungen für mobile Endgeräte, die die Android-Plattform unterstützen, möglich.

Das Android Betriebssystem besteht aus fünf Schichten (siehe Abbildung 2.2). Im Kern realisiert ein Linux Kernel, aktuell der Version 2.6, eine Abstraktion der zugrunde liegenden Hardware. Darauf aufbauend bieten C bzw. C++ Bibliotheken einen schnellen Zugriff auf Systemfunktionalitäten

[3]Eclipse ist eine Gemeinschaft die sich mit Open-Source-Projekten, unter anderem zur Entwicklung einer erweiterbaren Entwicklungsumgebung (IDE), befasst. Die zentrale Komponente ist dabei die Eclipse IDE. [Ecl10]

[4]Java Standard Edition

[5]Software Development Kit

wie z.B. Multimedia- und Grafikfunktionalitäten oder Datenbankdienste. Des weiteren ist auch die Android Laufzeitumgebung (Runtime) Bestandteil der Bibliotheksschicht. Die Android-Laufzeitumgebung umfasst dabei eine Reihe von Kern-Bibliotheken, die einen Großteil der Kern-Bibliotheken der Programmiersprache Java abbilden. Ein weiterer Bestandteil der Laufzeitumgebung ist die Dalvik VM[6], welche die Ausführung und Verwaltung von Android Anwendungen steuert. Dem Entwickler werden in der Schicht „Application Framework“Java-API's[7] zum einfachen Zugriff auf vordefinierte Dienste zur Verfügung gestellt. In der obersten Schicht bietet Android Kernanwendungen wie z.B. einen Mail-Client, einen Browser und Anwendungen für die Telefonie. [Goo10b]

Abbildung 2.2: Android System Stack [Goo]

2.1.2 Apple iPhoneOS

Apple's Betriebssystem für mobile Endgeräte ist iPhoneOS. Es ist eine vom Umfang angepasste und auf die ARM-Architektur portierte Version von Apple's Betriebssystem OS X. OS X und iPhoneOS bauen dabei auf gleiche Grundtechnologien wie z.B. dem OS X-Kernel und den Objective-C bzw. C/C++-Compilern, auf. IPhoneOS wurde zudem auf Effizienz und Kompaktheit optimiert und bietet dem Entwickler mit „Cocoa Touch“ein Programmier-Framework, das auf die Entwicklung von Anwendungen für Apple's mobile

[6]Virtual Machine
[7]Application Programming Interface

Endgeräte wie dem iPhone, iPod Touch und iPad, zugeschnitten ist. Auch „Cocoa Touch“ist von seinem Desktop-Pendant „Cocoa“abgeleitet und wurde zusammen mit der Benutzeroberfläche (GUI[8]) neu entworfen. Dadurch wird dem kleineren Formfaktor der mobilen Geräte Rechnung getragen und eine Gestensteuerung, auch mit mehreren Fingern gleichzeitig (Multitouch), ermöglicht. [App09a]

Wie schon Google für Android bietet auch Apple ein entsprechendes SDK zur Entwicklung von Anwendungen für die iPhone OS-Plattform an. Zur Distribution von Anwendungen ist eine kostenpflichtige Registration bei Apple's „iPhone Developer Program“nötig [Appa]. Für die iPhone OS-Plattform wird hauptsächlich in der Programmiersprache Objective-C entwickelt. Es handelt sich dabei um eine Erweiterung der Programmiersprache ANSI-C zur Unterstützung des Konzeptes der Objektorientierung. Diese Erweiterungen basieren zum Großteil auf der Programmiersprache Smalltalk. Eine Mischung von Objective-C, ANSI-C und C++ ist möglich [Appb]. Als Entwicklungsumgebung wird Apple's XCode verwendet, welche derzeit, genauso wie das iPhoneOS-SDK, nur für OS X verfügbar ist.

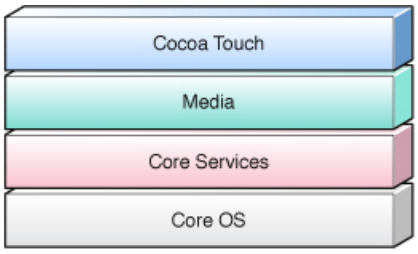


Abbildung 2.3: Schichten des iPhoneOS [Appc]

Das iPhoneOS-SDK von Apple ist, ähnlich zu Google's Android-SDK, in Schichten untergliedert. Die beiden unteren Schichten, "Core OS" und "Core Services" (vgl. Abbildung 2.3), bieten fundamentale Dienste, auf denen die Anwendungen aufbauen. Die beiden höheren Schichten, "Media" und "Cocoa Touch", bieten eine fortschrittliche, objektorientierte Abstraktion der Konstrukte der unteren Schichten. Anwendungsentwickler sollten daher, so weit möglich, die oberen Schichten bevorzugen, da komplexe Funktionen hier schon gekapselt sind und somit weniger Codezeilen zu schreiben sind [App09c].

[8]Graphical User Interface

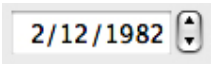


Abbildung 2.4: DatePicker unter OS X optimiert für Mauseingabe [Appb]

Abbildung 2.5: DatePicker unter iPhoneOS optimiert für Touch-Eingabe [Appa]

Die Entwicklung von Anwendungen für die iPhoneOS-Plattform ist zudem an Vorgaben von Apple gebunden. Diese Vorgaben sollen den Entwickler bei der Konzeption und der Entwicklung einer Anwendung unterstützen. Dazu zählen unter anderem das in der iPhoneOS-Architektur stark verwurzelte MVC-Modell[9] und die "Apple Guidelines", allem voran die "iPhone Human Interface Guidelines". Diese sollen im Zusammenspiel mit den im SDK angebotenen grafischen Benutzerelementen dem Entwickler die Erstellung von Benutzeroberflächen erleichtern. Durch den Wiedererkennungswert der Elemente sind die Oberflächen zum Einen intuitiv, und zum Anderen, durch die Definition von minimalen Abmessungen, auch physisch einfach zu bedienen (vgl. Abbildung 2.4 und Abbildung 2.5).

2.1.3 Weitere Mobilplattformen

Neben den zuvor genauer betrachteten mobilen Plattformen Android und iPhoneOS existieren weitere Plattformen und Betriebssysteme für Smartphones. Dazu gehören unter anderem webOS, welches auf den aktuellen Smartphones des Unternehmens Palm eingesetzt wird, Microsoft's Windows Mobile oder Nokia's Symbian OS. Auch diese bieten Entwicklungsumgebungen und SDK's zur Erweiterung des Anwendungsspektrums der Systeme. Von der

[9] Model-View-Controller.

hard- oder softwareseitigen Ausstattung unterscheiden sich die Geräte dieser Klasse kaum, so dass die Anforderungen auch von Geräten dieser mobilen Plattformen erfüllt werden. Eine eingehende Betrachtung dieser Plattformen würde aber den Rahmen dieser Arbeit übersteigen.

2.1.4 Zusammenfassung

Die beiden zuvor betrachteten mobilen Plattformen eignen sich zur Entwicklung des Prototypen, da sie eine Vielzahl von Drahtlostechnologien mitbringen und durch entsprechende SDK's um zusätzliche Anwendungen erweiterbar sind. Es handelt sich dabei um zwei sehr junge mobile Plattformen denen für die nahe Zukunft ein großes Potential zugesprochen wird. Die Android-Plattform bietet eine plattformunabhängige Entwicklung und eine Vielzahl von Herstellern, die Android auf ihren Endgeräten einsetzen. Für die iPhoneOS-Plattform sind nur wenige verschiedene Modelle verfügbar. Dazu gehören unterschiedliche Versionen des iPod Touch und des iPhone. Die eingeschränkte Auswahl an Endgeräten bei Apple muss dabei kein Nachteil sein. Seit der Einführung der ersten iPhone-Version in 2007 haben sich die grundlegenden Prinzipien, wie z.B. die Eingabemethoden, und die Fähigkeiten des Displays nicht geändert. Im Gegensatz dazu ist das Bedienkonzept der Android-Plattform zwar auf Touch-Eingabe ausgelegt, diese aber nicht zwingend vorgeschrieben. Die Beschränkungen der iPhoneOS-Plattform erleichtern dem Programmierer somit die Entwicklung der Anwendung, da keine unterschiedlichen Eingabemethoden oder Auflösungen des Displays berücksichtigt werden müssen.

Ein weiterer Aspekt, der für die Verwendung der iPhoneOS-Plattform spricht, ist der derzeitige Marktanteil der beiden Plattformen. Laut Gartner [Gar10c] führt im Segment der mobilen Betriebssysteme iPhoneOS mit 14,4% klar vor Android mit 3,9%. Ab April diesen Jahres kommt zudem in Deutschland mit dem iPad ein Gerät der Klasse der Tablet-PC auf den Markt, welches sich hervorragend als Info-Panel im Haushalt eignet, zudem aber auch Mobilfunktechnologien mitbringt, die einen mobilen Einsatz auch außerhalb der eigenen vier Wände ermöglicht. Obwohl die Android-Plattform für die Entwicklung des Prototypen genauso geeignet wäre, werde ich auf Grund der zuvor genannten Argumente auf der iPhoneOS-Plattform entwickeln.

2.2 Drahtlose Kommunikationstechnologien

Der Erfolg der mobilen Endgeräte basiert zu einem sehr großen Teil auf der Unterstützung von Technologien zur drahtlosen Kommunikation. So ist es heutzutage vielen mobilen Endgeräten, insbesondere den schon betrachteten Smartphones, möglich, Netzwerken ohne Hilfe von Kabeln beizutreten und über diese zu kommunizieren. Wie auch bei den drahtgebundenen Netzwerken gibt es bei den drahtlosen Netzwerken verschiedene Technologien und Standards. Diese sind auf einen speziellen Zweck bzw. ein abgegrenztes Einsatzgebiet hin entwickelt und optimiert worden, so dass in diesem Feld eine Vielzahl verschiedener Technologien zur Verfügung stehen. Diese Heterogenität und der Umstand, dass die verschiedenen Technologien nicht immer zueinander kompatibel und somit meist auch keine nahtlosen Wechsel zwischen den Netzwerken verschiedener Technologien möglich sind [Hub02][Bhe08], macht es nötig, die vorhandenen Drahtlostechnologien genauer zu betrachten. Dazu werden diese Technologien in den folgenden Abschnitten vorgestellt und darauf hin untersucht, ob ein Einsatz der jeweiligen Technologie im zu entwickelnden System möglich und zweckmäßig ist. Insbesondere muss hierbei die Ausdehnung der jeweiligen Netze und die Möglichkeit der Anbindung an ein Gebäudesteuerungssystem betrachtet werden, da das mobile Steuern und Überwachen sowohl lokal, innerhalb des Hauses, als auch von beliebigen anderen Standorten aus vorgenommen werden kann. Aus diesem Grund werden zunächst Drahtlostechnologien für Mobilfunknetze betrachtet, um eine geeignete Technologie für den Zugriff auf das System von unterwegs aus zu finden. Technologien für lokale Drahtlosnetzwerke (WLAN[10]) werden danach betrachtet. Im letzten Teil wird dann anhand einiger Technologien für Nahbereichsnetzwerke (WPAN[11]) erörtert, ob diese für einen Einsatz im System geeignet sind.

[10] Wireless Local Area Network
[11] Wireless Personal Area Network

2.2.1 Drahtlostechnologien für Mobilfunknetze

Mobilfunknetze sind Kommunikationsnetze, die große Bereiche, z.B. Länder, flächendeckend abdecken. Mit geeigneten mobilen Endgeräten wie Mobilfunktelefonen kann auf diese Netze zugegriffen und die Kommunikationsdienste genutzt werden. Solche Netze sind meist zellular aufgebaut und erlauben einen nahtlosen Wechsel zwischen den Zellen ohne, dass die Kommunikation unterbrochen wird. Zur Zeit werden in Deutschland Mobilfunknetze der zweiten (2G) und dritten (3G) Generation betrieben.

GSM

GSM[12] ist ein Mobilfunknetz der zweiten Generation. Die Ziele bei der Einführung dieses Netzes waren u.a. ein „breites Sprach- und Datendienstangebot“, „Kompatibilität zu leitungsgebundenen Diensten“und ein „automatisches europaweites Roaming und Handover“[Wal01, S. 135f.]. Der Testbetrieb dieses digitalen Netzes wurde 1991 aufgenommen und ist heute einer der am weitesten verbreiteten Mobilfunkstandards [GSM].

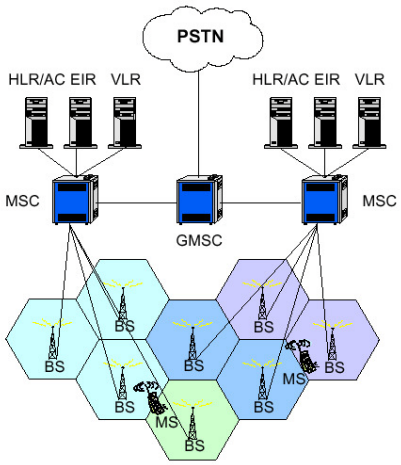


Abbildung 2.6: GSM-Funkzellen [Bun]

Wie Abbildung 2.6 zeigt, ist das GSM-Netz in Funkzellen unterteilt, die einen maximalen Radius von bis zu 35km haben können. Diese Architektur erlaubt eine Wiederverwendung der verfügbaren Frequenzen in nicht benachbarten Zellen sowie die Nutzung relativ niedriger Sendeleistungen. Die in Eu-

[12]Global System for Mobile Communications bzw. früher Groupe Speciale Mobile

ropa genutzten Funkfrequenzbänder sind GSM900 und GSM1800. Durch die Unterteilung dieser Frequenzbänder entstehen jeweils 124 FDM-Kanäle[13] für Up- und Downlink. Zusätzlich zu diesem Frequenzmultiplexverfahren wird, zur Steigerung der möglichen gleichzeitigen Verbindungen, ein Zeitmultiplexverfahren (TDM[14]) genutzt. Hierfür werden die Kanäle zusätzlich in acht Zeitschlitze mit jeweils 0,577ms Länge aufgeteilt [Wal01, S. 155ff.]. Somit ist ein physikalischer Kanal bei GSM900 „durch seine Trägerfrequenz und seinen [...] wiederkehrenden Zeitschlitz charakterisiert“[Wal01, S. 156].

Mit GPRS[15] steht zudem ein paketvermittelter Datendienst mit Datenübertragungsraten zwischen 56-114kbit/s in GSM zur Verfügung. Mit der Einführung der Erweiterung EDGE[16] für GPRS konnte über ein verbessertes Modulationsverfahren die Datenübertragungsrate auf ca. 200 kBit/s erhöht werden. GSM-Mobilfunknetze, die mit der Erweiterung GPRS ausgestattet sind, werden oft auch als 2.5G-Netze bezeichnet.

UMTS

UMTS[17] ist ein Mobilfunkstandard der dritten Generation (3G) und somit eine „evolutionäre Weiterentwicklung von 2G-Mobilfunksystemen, die auf GSM beruhen“[Leh03, S. 65]. Es wird vom 3GPP[18] gepflegt und ist Teil des internationalen ITU IMT-2000 Standards. Wie GSM setzt UMTS auf einen zellularen Netzaufbau und digitale Übertragung. Während GSM auf „schmalbandigen Sprach- und Datenverkehr ausgelegt ist, sollen bei UMTS zusätzliche Dienste mit höherer Qualität [...] eingeführt werden“[Leh03, S. 65]. Im Gegensatz zu GSM wird bei UMTS der Frequenzbereich um 2,1GHz genutzt. Die Aufteilung in gesonderte Frequenzbänder für Up- und Downlink bleibt. Auch das Funkzugriffsverfahren unterscheidet sich, da hier eine Kombination aus FDMA und W-CDMA[19] genutzt wird.

Durch diese neuen Technologien sind Datenübertragungsraten bis zu 384kbit/s möglich. Wird die Erweiterung HSDPA[20] verwendet sind Datenübertragungsraten bis zu 14 MBit/s möglich.

[13] Frequency Division Multiplexing
[14] Time Division Multiplexing
[15] General Packet Radio Service
[16] Enhanced Data Rates for GSM Evolution
[17] Universal Mobile Telecommunications System
[18] 3rd Generation Partnership Project
[19] Wideband Code Division Multiple Access
[20] High Speed Downlink Packet Access

Long Term Evolution (LTE)

Long Term Evolution (LTE) ist ein Standard des 3GPP und soll ein „mobile-communication system that can take the telecom industry into the 2020s“sein [DPSB08, S. 22]. Es ist als Nachfolger des derzeitigen Mobilfunkstandards UMTS gedacht. Ziel ist die Spezifizierung eines Mobilfunksystems durch das 3GPP, das aber nicht durch Rückwärtskompatibilität zu früheren 3GPP-Standards wie WCDMA und HSPA[21] beschränkt ist. Da der Anteil der paketvermittelten Dienste im weltweiten mobilen Datenverkehr immer wichtiger wird und mittlerweile größer ist als der Anteil der Sprachdienste [Net09, S. 5], ist LTE von Grund auf für IP-basierte Übertragung optimiert und ist darauf ausgelegt flexibel mit Bandbreiten zwischen 1-20 MHz umgehen zu können [DPSB08, S. 22-23]. LTE ist auf Datenraten von bis zu 300 MBit/s im Downlink bzw. 75 MBit/s im Uplink ausgelegt. Durch die Flexibilität bei der Nutzung des dem Netzbetreiber zur Verfügung stehenden Spektrums und durch die Nutzung von effizienten Multiplexing- (OFDM[22]) und Antennentechnologien (MIMO[23]) soll ein kostengünstiges Mobilfunknetz mit hohen Datenraten entstehen.

2.2.2 Drahtlostechnologien für lokale Netzwerke (WLAN)

Lokale Drahtlosnetzwerke (WLAN) haben eine weitaus kleinere Ausdehnung als Mobilfunknetze. Meist ist die Reichweite auf wenige Meter bis einige 100 Meter begrenzt. Lokale Drahtlosnetzwerke erweitern meist ein kabelgebundenes lokales Netzwerk (LAN) zur Integration mobiler Endgeräte in privaten Haushalten oder Firmenstandorten. Sie zeichnen sich im Gegensatz zu Mobilfunknetzen durch hohe Datenübertragungsraten aus. Dadurch ist eine einfache, drahtlose Anbindung von Multimediageräten oder anderen durchsatzintensiven Geräten möglich.

WiFi / Wireless LAN

Die Begriffe WiFi und WLAN werden in Deutschland dazu verwendet um Geräte, die nach dem IEEE-802.11-Standard zertifiziert sind, zu bezeichnen. Dieser Standard wurde 1997 von der IEEE spezifiziert und wurde mittlerweile

[21] High Speed Packet Access
[22] Orthogonal-Frequency-Division-Multiplexing
[23] Multiple-Input-Multiple-Output

durch mehrere Erweiterungen verbessert. Während im ursprünglichen Standard Übertragungsraten über Infrarotlicht und Radiowellen bis zu 2 MBit/s vorgesehen waren, bietet der Standard 802.11n, der im September 2009 vorgestellt wurde, Übertragungsraten von bis zu 600 MBit/s [Ins09]. Zur Übertragung werden nur lizenzfreie Frequenzbänder in den sog. ISM-Bändern[24] genutzt. Tabelle 2.1 zeigt die verschiedenen Varianten des IEEE-802.11 Standards mit deren maximalen Bruttodatenraten und den unterstützten Frequenzbereichen.

Variante	theor. Bruttodatenraten	Frequenzband
802.11	2 MBit/s	2,4GHz
802.11a	54 MBit/s	5GHz
802.11b	11 MBit/s	2,4GHz
802.11g	54 MBit/s	2,4GHz und 5GHz
802.11n	600 MBit/s	2,4GHz und 5GHz

Tabelle 2.1: Varianten des IEEE-802.11 Standards [Fre07, S. 562f]

Da es bei drahtlosen Netzwerken schwierig ist den physikalischen Zugang zu kontrollieren und zu unterbinden, sind Zugangskontroll- und Verschlüsselungsmechanismen Teil des Standards. Neben dem mittlerweile als unsicher geltenden WEP-Verfahren wurden als Übergangslösung die Verschlüsselungsmethode WPA und mit Verabschiedung des Standards IEEE-802.11i dessen Nachfolger WPA2 eingeführt.

Grundsätzlich stehen zur Kommunikation zwischen Endgeräten zwei Topologien zur Verfügung. Im Ad-hoc-Modus kommunizieren alle Endgeräte, die sich jeweils in Reichweite befinden, direkt miteinander. Im Infrastruktur-Modus hingegen läuft die gesamte Kommunikation über eine Vermittlungsinstanz, dem sog. Access-Point (AP)[Leh03, S. 122].

2.2.3 Drahtlostechnologien für Nahbereichsnetzwerke (WPAN)

Wireless Personal Area Networks (WPAN) sind drahtlose Nahbereichsnetzwerke für eine Kommunikation von bis zu zehn Metern. Diese Drahtlostechniken dienen dazu, Geräte des persönlichen Arbeits- bzw. Lebensbereiches zu

[24] Industrial, scientific, medical

vernetzen und somit kurzzeitige bzw. fliegend verlegte Kabelverbindungen überflüssig zu machen. Zu diesen Geräten gehören unter anderem Mobiltelefone, PDA's, Drucker und Multimediageräte. Die Datenraten aber auch die Sendeleistungen und somit der Stromverbrauch liegen typischerweise unter denen von WLAN-Technologien. Das Hauptaugenmerk liegt bei den WPAN-Technologien oft auf einem sehr einfachen und schnellen Verbindungsaufbau, da die Verbindungen zwischen den Geräten oft spontan initiiert werden und nur von kurzer Dauer sind.

Beispiele für WPAN-Technologien

In diesem Bereich gibt es viele verschiedene Technologien. Bluetooth, vom IEEE-Konsortium als IEEE 802.15.1 standardisiert, ist in mobilen Endgeräten wie Mobilfunktelefonen oder Notebooks und Peripheriegeräten wie Druckern oder Telefon-Headsets weit verbreitet. Ziel dieser Technik ist hauptsächlich der Ersatz von Kabelverbindungen zwischen diesen Geräten. Es gibt dabei drei Klassen von Bluetoothgeräten. Die Sendeleistung der einzelnen Klassen liegt dabei zwischen 1mW und 100mW. Damit können Reichweiten zwischen einem und 100 Meter erreicht werden. Die Datenübertragungsraten liegen zwischen ca. 730kBit/s bis 2MBit/s [Blu10].

Eine weitere von der IEEE unter IEEE 802.15.4 geführte Technik ist ZigBee. Diese soll hauptsächlich zur Verbindung von z.B. Haushaltsgeräten und Sensoren oder ähnlichen Geräten eingesetzt werden. Merkmale der ZigBee-Technologie sind Datenraten von bis zu 250kBit/s über typischerweise 50 Meter. Zudem kann ein ZigBee-Netz aus bis zu 65500 Geräten bestehen [Zig10].

Near Field Communication (NFC), von der ISO als ISO 18092 standardisiert, ist eine Technologie zum Austausch von Daten über kurze Strecken. Als typische Einsatzfelder werde die Zugriffskontrolle, mobile Bezahlfunktionen oder Ticketing-Funktionen genannt. Dazu setzt NFC auf eine Kombination aus SmartCard- und Drahtlostechnologien. Es wird der Frequenzbereich von 13,56MHz genutzt. Die Technologie bietet dabei Übertragungsraten von bis zu 424kBit/s bei einer Reichweite von nur zehn Zentimetern [Sto06]. Bisher hat NFC keine große Bedeutung erlangt und konnte sich gerade im Bereich der Zugangssteuerung nicht gegen Bluetooth durchsetzen.

Weniger verbreitet und bekannt ist KNX-RF. Hierbei handelt es sich um eine Drahtlostechnik, die speziell im Bereich der Gebäudeautomation zum Einsatz kommt. Sie wird dazu verwendet KNX[25]-fähige Geräte wie zum Beispiel Aktoren zur Steuerung von Lichtquellen oder Haushaltsgeräten ohne Sanierungsaufwand am Gebäude in ein KNX-System einzubinden. Die Technik arbeitet im 868MHz-Band, welches in Europa für sog. Short-Range-Devices (SRD) reserviert und somit weniger störanfällig ist. Bei Sendeleistungen von ein bis 25mW werden Reichweiten von 30 Metern (in Gebäuden) bis 100 Metern (Freifeld) bei Übertragungsraten von 16 kBit/s erreicht. Bei KNX-RF werden dabei zwei Arten von Geräten unterschieden. Geräte, die nicht permanent empfangsbereit sein müssen wie z.B. Sensoren unterstützen eine unidirektionale Kommunikation, d.h. sie haben keinen Empfänger und senden nur bei Bedarf. Die zweite Gerätekonfiguration unterstützt bidirektionale Kommunikation und ist für alle Geräte geeignet, die jederzeit fernsteuerbar und damit empfangsbereit sein müssen. [GK09, S. 234][KSH09, S. 93-96]

2.2.4 Zusammenfassung

Im Bereich der drahtlosen Kommunikationstechniken gibt es eine Vielzahl verschiedener Technologien und Standards für sehr unterschiedliche Einsatzbereiche. In den vorangegangenen Abschnitten wurde ein Überblick über gängige Technologien, unterteilt nach dem Merkmal der Netzausdehnung, gegeben. Zusammenfassend lässt sich herausstellen, dass die Technologien der drahtlosen Nahbereichsnetzwerke nicht für den zu entwickelnden Prototypen geeignet sind. Das liegt teilweise an den meist geringen Reichweiten dieser Technologien (Bluetooth, NFC) oder der mangelnden Unterstützung durch mobile Endgeräte (ZigBee, NFC, KNX-RF). Diese Techniken eignen sich besser zur Vernetzung der Geräte der Gebäudetechnik untereinander oder für Zugriffssteuerungssyteme. Die Drahtlostechnologie WiFi bietet hingegen eine größere Netzausdehnung und ist zudem in einer Vielzahl mobiler Endgeräte und in Form eines Access Points (AP) in vielen Haushalten schon verfügbar. Befindet man sich außerhalb des eigenen WiFi-Netzwerkes bieten sich die aktuellen Mobilfunktechnologien GSM und UMTS für den Zugriff auf das System über das Internet an. Die meisten der derzeit verfügbaren Smartphones sind mit diesen Technologien ausgestattet. Aus diesen Gründen werde ich mich beim Prototypen auf die Unterstützung der Techniken WiFi sowie GSM und UMTS beschränken.

[25]KNX ist ein Feldbus-System für die Gebäudeautomation. Es wird im Unterabschnitt 2.3.1 näher erläutert.

2.3 Gebäudeautomation

Wie Frank [Fra09], Merz [MHH10] und Kriesel [KSH09] übereinstimmend berichten, ist die Gebäudeautomation bei Wohn- und Zweckbauten auf dem Vormarsch. Dabei spielen ein gestiegenes Umweltbewusstsein und vor allem eine höhere Wirtschaftlichkeit durch Energieeinsparungen, aber auch ein gesteigertes Komfort- und Sicherheitsbedürfnis der Kunden eine übergeordnete Rolle. Mit der konventionellen Elektroinstallation sind diese Aufgaben aber kaum noch zu lösen. Um komplexe Automatisierungsfunktionen, wenn z.B. beim Verlassen des Hauses zeitgleich viele Verbraucher ab- und die Alarmanlage scharf geschaltet werden sollen, mit der konventionellen Elektroinstallation umzusetzen würde es einer kaum überschaubaren Verkabelung bedürfen [MHH10, S. 17-18]. Zudem würde man wegen der geringen Flexibilität, vor allem in Bezug auf Veränderungen in der Gebäudenutzung, schnell an die Grenzen der konventionellen Elekroinstallation stoßen (vgl. Abbildung 2.7) [KSH09, S. 2-3].

Das Bedürfnis nach einem einfach zu installierenden und flexiblen System für die Gebäudesteuerung und -automation führte zu Beginn der 1990er Jahre dazu, dass sich offene Bussysteme am Markt etablierten [MHH10, S. 34-35]. Durch diese ist es möglich komplexe Systeme zentral zu steuern und auf Veränderungen flexibel zu reagieren. Das Gebiet der Gebäudeautomation und der sogenannten „Smart Homes“ist zudem ein stetig wachsender Forschungsbereich. Viele Organisationen untersuchen dabei vorhandene Technologien auf ihre Praxistauglichkeit und Grenzen und versuchen zukünftige Szenarien des Wohnens aufzuzeigen. Beispiele dafür sind das „inHaus1“und „inHaus2“der Fraunhofer-Gesellschaft [Fra10] oder das „Haus der Zukunft“des österreichischen Bundesministerium für Verkehr, Innovation und Technologie [Bun10]. Einige dieser Systeme sollen in den folgenden Abschnitten vorgestellt werden.

2.3.1 KNX/EIB

Das Feldbussystem Konnex (KNX) ist ein in Europa weit verbreiteter Standard für die Gebäudeautomation. Er entstand durch die Vereinigung der verschiedenen europäischen Installationsbussysteme „European Installation Bus“(EIB), „BatiBUS“und „European Home Systems“(EHS) im Jahr 1999. Hauptsächlich basiert der daraus entstandene KNX-Standard auf dem aus Deutschland stammenden EIB [Fra09, S. 113], der „[...]um Konfigurations-

mechanismen und Datenübertragungsmedien der anderen beiden Systeme [...] erweitert“wurde [KSH09, S. 15]. Mittlerweile ist KNX auch von weiteren internationalen Normungsgremien, darunter auch ISO und ANSI, anerkannt worden und somit „[...] der weltweit erste offene Kommunikationsstandard für Steuerungssysteme in Heim und Gebäude“[KSH09, S. 17].

Um die Notwendigkeit eines standardisierten Bussystems für die Kommunikation in Gebäudetechniktechniksystemen zu verstehen, muss man die Vorgehensweise mit der konventionellen Elekroinstallation, wie sie heute noch sehr weit verbreitet ist, betrachten.

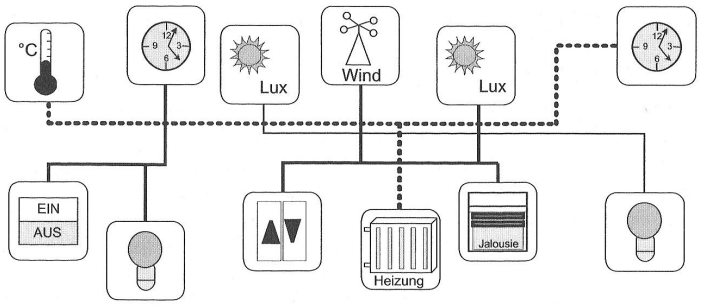


Abbildung 2.7: Kommunikation bei konventioneller Elektroinstallation [WFPb]

„Seit den Anfängen der Nutzung elektrischer Energie werden elektrische Verbraucher [...] direkt über einen Schalter mit elektrischer Energie versorgt“[KSH09, S. 1]. Das heißt also, dass in der konventionellen Elektroinstallation die Stromleitung auch zur Informationsübertragung (z.B. Ein/Aus) genutzt wird [Fra09, S. 15]. Komplexe Schaltungen von Verbrauchern können also nur durch eine ebenso komplexe Verdrahtung der einzelnen Verbraucher und Schalter realisiert werden (siehe Abbildung 2.7). Es entstehen laut Kriesel [KSH09, S. 1-2] bestenfalls Insellösungen, z.B. eines für die Beleuchtung, ein anderes für die Heizungssteuerung, die nicht untereinander verbunden sind. Mit steigender Anzahl der integrierten Geräte steigt zudem der Bedarf an Leitungswegen und die Komplexität der Verdrahtung überproportional an.

Bei dem Bussystem KONNEX (KNX) sind die Stromleitung und die Leitung zur Informationsübertragung voneinander getrennt (siehe Abbildung 2.8). Zur physikalischen Informationsübertragung können bei KNX/EIB verschiedene Medien eingesetzt werden. Am weitesten verbreitet ist dafür ei-

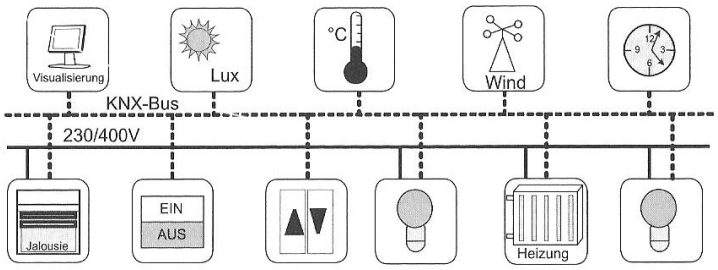


Abbildung 2.8: Informationsübertragung über das Bussystem KNX [WFPa]

ne verdrillte Zweidrahtleitung (KNX.TP[26]), es kann aber auch Power Line (KNX.PL), Funk (KNX.RF) und eingeschränkt auch Ethernet (KNXnet/IP) eingesetzt werden. Neben dieser Trennung von Strom- und Informationsleitung wird bei KNX zudem ein Schaltvorgang von zwei getrennten Geräteklassen vorgenommen. Sensoren wandeln dabei physikalische Größen, wie z.B. einen Tastendruck oder die Temperatur, in Informationen um und können diese weitergeben. Aktoren hingegen können diese Informationen verarbeiten und wieder in eine physikalische Größe umwandeln, z.B. einen Schaltkontakt schalten oder einen Jalousiemotor steuern. Durch diese, auch räumlich mögliche, Trennung von Schalt-Auslöser (Sensor) und Schalter (Aktor) und die Trennung von Strom- und Informationsleitungen kann das System durch die Möglichkeit der Umprogrammierung der Aktoren und Sensoren sehr flexibel gehalten werden.

Die Kommunikation der Busteilnehmer (Sensoren und Aktoren) erfolgt über die Busleitung. Alle Teilnehmer sind über einen Busankoppler an diesen angeschlossen und können Informationen der anderen Teilnehmer über Telegramme empfangen. Dadurch stehen sämtliche Informationen (Zustände/Werte) je nach Bedarf an jedem Punkt im System zur Verfügung und es ist keine zentrale Verwaltung nötig [KSH09, S. 4].

Das Bussystem ist dabei sehr übersichtlich strukturiert und unterteilt sich immer in Bereiche und Linien, egal wie komplex das zu erstellende System ist. Ein Bereich ist die übergeordnete Hierarchie. Insgesamt lassen sich 16 Bereiche mit jeweils 16 Linien, in denen wiederum theoretisch bis zu 256 Geräte angesprochen werden können, adressieren (siehe Abbildung 2.9). Die Unterteilung in Bereiche und Linien hat im Wesentlichen die Aufgabe Bereiche galvanisch und in Bezug auf die Telegrammübermittlung abzutrennen. Ähnlich wie bei der Segmentierung von IP-Netzen erreicht man dadurch eine

[26]Twisted Pair

geringe Belastung des Busses durch Daten- bzw. Telegrammverkehr, da nur Telegramme für linien- bzw. bereichsfremde Teilnehmer über Linien bzw. Bereiche hinweg geroutet werden. Zudem bleiben andere Linien und Bereiche bei Störungen einzelner Linien oder Bereiche funktionsfähig.

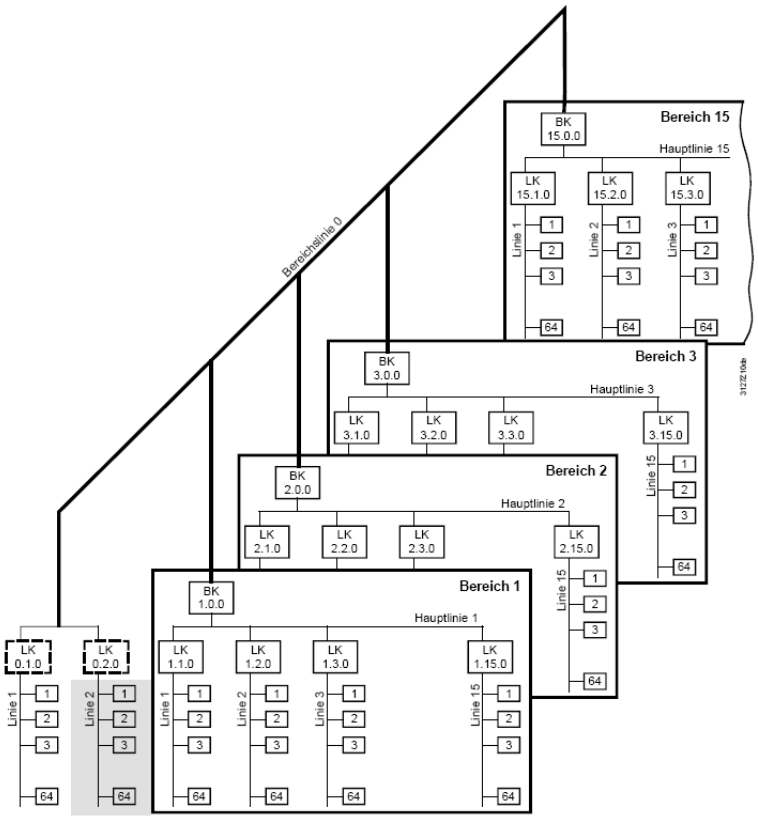


Abbildung 2.9: Topologie des KNX-Systems [Sie]

Wie schon angesprochen findet die Kommunikation zwischen den Teilnehmern über Telegramme statt. Eine Adressierung der Teilnehmer wird über physikalische Adressen, die jeden Teilnehmer eindeutig identifizieren und logische Adressen, sogenannte Gruppenadressen, realisiert. Durch letztere können mehrere Geräte zu einer Gruppe zusammengefasst werden. Meist werden die Gruppen, die in zwei oder drei Hierarchie-Stufen unterteilt sind, je nach Funktionsbereich der Geräte gebildet [FM09a, S. 110-111]. Jedes Telegramm, das an eine solche Gruppenadresse geschickt wird, wird von allen Teilnehmern dieser Gruppe empfangen und verarbeitet. Die physikalischen Adressen der einzelnen Teilnehmer ergeben sich aus der Topologie des Systems. Ist ein Gerät zum Beispiel im Bereich 2 an der Linie 15 angeschlossen, so wären die physikalische Adressen von 2.15.0 bis 2.15.255 gültig, wobei die Adresse

2.15.0 für den Linienkoppler, der die verschiedenen Linien mit der Hauptlinie verbindet, reserviert ist. Ein Telegramm enthält dabei neben Quell- und Zieladresse noch weitere Informationen wie z.B. Steuerungsinformationen und Nutzdaten. Die Übertragungsrate innerhalb des Bussystems liegt bei 9,6kBit/s [FM09b, S. 179].

Da das System äußerst flexibel ist und auch die einzelnen Busteilnehmer unter Umständen für unterschiedliche Zwecke eingesetzt werden können, ist es nötig, diese vor ihrer Verwendung zu programmieren. Für diese Programmierung benötigt man für die meisten Geräte einen PC, der mit dem Bussystem über ein entsprechendes Gateway (z.B. USB oder Ethernet) verbunden ist. Über die ETS[27] wird eine physikalische Adresse und ein parametriertes Anwendungsprogramm konfiguriert und auf das entsprechende Gerät übertragen. So ist es durch einfache Umprogrammierung zum Beispiel möglich, dass ein und derselbe Wipp-Schalter in einer Konfiguration über eine bestimmte Gruppenadresse mehrere Lichtquellen dimmt und in einer anderen Konfiguration unter einer anderen Gruppenadresse bei Betätigung einen bestimmten Wert sendet, z.B. für eine Klingelschaltung.

Die Flexibilität des Systems wird zudem durch zahlreiche verfügbare Schnittstellen bzw. Gateways zu anderen Systemen erhöht. So ist es unter anderem möglich DALI[28]-Lichtsteurungssysteme oder DMX-Veranstaltungssysteme in ein KNX-System einzubinden oder über ein GSM-Gateway eine Steuerung des Systems über den SMS-Dienst zu realisieren.

2.3.2 Weitere Bus-Technologien zur Gebäudeautomation

Neben dem KNX-Bussystem gibt es im Bereich der Gebäudeautomatisierung weitere Technologien, die eine intelligente Steuerung und Überwachung von Gebäudetechnik ermöglichen. Mit LON[29] wird nachfolgend ein weiteres international standardisiertes System für die Gebäudeautomation vorgestellt.

[27] Engeneering Tool Software.
[28] Digital Addressable Lightning Interface .
[29] Local Operating Network

Die Technologie LonWorks, die den Feldbus LON einsetzt, wurde ca. 1990 von der Firma "Echelon Corporation" entwickelt und ist seit Dezember 2008 von der ISO als internationale Norm anerkannt. Die Technik basiert auf einem ähnlichen Ansatz wie KNX und steht somit in Konkurrenz zu diesem System. Über entsprechende Gateways können beide Welten, die sich teilweise ergänzen, aber auch verbunden werden. [MHH10, S. 156-157]

Kerngedanke der Technologie war ursprünglich „[...] die Entwicklung einer universellen Technologie für dezentrale Netzwerke“[MHH10, S. 161]. Die Grundelemente (Knoten) des Systems sind bei LON Sensoren, Aktoren und Controller. Die Informationen werden dabei lokal in den Knoten verarbeitet wodurch eine zentrale Steuereinheit überflüssig wird. Ähnlich wie bei KNX sind Strom- und Informationsleitung getrennt (siehe Abbildung 2.10). Die Anbindung der Knoten an ein Übertragungsmedium wird über einen Transceiver vorgenommen und kann, je nach Medium, eine Bitrate von bis zu 1,25MBit/s erreichen. Als Übertragungsmedien können unter anderem verdrillte Zweidrahtleitungen (Twisted Pair), Power Line, Funk oder Lichtwellenleiter verwendet werden.

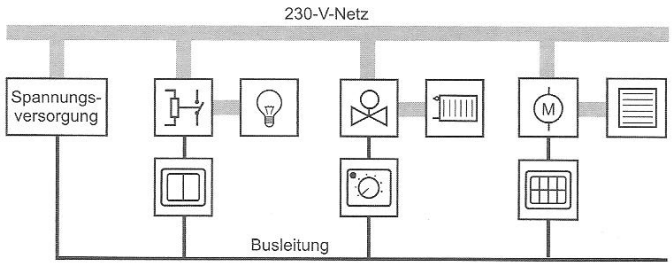


Abbildung 2.10: Informationsübertragung über das Bussystem LON [MHH]

Ähnlich wie bei KNX läuft die Kommunikation zwischen den einzelnen Knoten über Telegramme. Übertragen werden diese über das LonTalk-Protokoll. Alle Teilnehmer hören dabei das Übertragungsmedium ab und können selbst auf die Busleitung über ein CSMA[30] zugreifen. Pro Subnetz können bei LON 128 Geräte betrieben werden. Über Router können zudem bis zu 255 Subnetze verbunden werden. Diese Router bieten auch eine Filterfunktion, damit Telegramme des eigenen Subnetzes nicht weitergeleitet werden. Dadurch kann die Gesamtlast des Netzes gering gehalten werden.

[30]Carrier Sense Multiple Access

2.4 Beispiele für Anwendungen zum mobilen Zugriff auf Gebäudeautomationssysteme

Die Steuerung und Überwachung intelligenter Gebäudetechnik ist prinzipiell über mehrere Wege möglich. Weit verbreitet sind sogenannte „Home-Server“, ein zentraler Rechner, bei dem alle Informationen des Gebäudeautomationssystems zusammenlaufen und dort aufbereitet werden. Hauptfunktion dieser Server ist die Bereitstellung einer Visualisierung des Systems. Auf diese Visualisierung können Info-Panels, die im Haus verteilt sein können, aber auch PCs und in einigen Fällen auch mobile Endgeräte zugreifen. Bietet das eingesetzte Home-Server-System einen Webserver ist auch eine Steuerung und Überwachung über einen Browser von beliebigen Standorten möglich. Die verschiedenen Home-Server-Systeme unterscheiden sich in ihrem Funktionsumfang erheblich und bieten meist, bis auf die über einen eventuell vorhandenen Webserver gebotenen Funktionalitäten, keine einheitlichen Zugriffsverfahren. Zudem muss die Konfiguration meist von geschultem Fachpersonal vorgenommen werden. Vorteile dieser Herangehensweise sind die sehr gute Anpassbarkeit an das zugrunde liegende Gebäudeautomationssystem und eine plattformunabhängige Bereitstellung von Funktionalitäten über einen Webserver. Ein Beispiel eines solchen Systems ist der Gira HomeServer bzw. Gira FacilityServer (siehe Abbildung 2.11) [Gir10]. Über diese Server-Systeme können komplexe KNX/EIB-Systeme gesteuert und mit Visualisierungen dargestellt werden. Über einen Webserver ist der Zugriff auf das System mit einer Vielzahl von Geräten über einen Web-Browser möglich.

Abbildung 2.11: Beispiel eines Info-Panels mit Visualisierung (Gira) [Gir]

Eine weitere Möglichkeit zum mobilen Zugriff auf Gebäudeautomationssysteme sind native Anwendungen auf dem mobilen Endgerät. Dabei ist zu unterscheiden, ob durch die mobile Anwendung nur auf einen, von einem Home-Server-System bereitgestellten, Webserver zugegriffen wird oder ein KNXnet/IP-Gateway bzw. -Router direkt angesprochen wird. Die Anwendung „m..Remote“[ibs10] stellt dabei einen Vertreter der ersten Kategorie dar und bietet somit nur eine auf das mobile Endgerät angepasste Version der Funktionalitäten, die der zugrunde liegende Webserver zur Verfügung stellt. Ist ein direkter Zugriff auf die Schnittstelle zum Gebäudeautomationssystem möglich entfällt die ansonsten zwischengeschaltete zentrale Steuereinheit, z.B. der Home-Server. Die Kommunikation mit der Schnittstelle zum Gebäudeautomationssystem kann über standardisierte Protokolle und Nachrichten erfolgen. Ein Nachteil ist, das für die unterschiedlichen mobilen Plattformen eigene Anwendungen entwickelt werden müssen. Zu dieser Kategorie gehört die Anwendung „ayControl“[eas10]. Mit dieser Anwendung ist es möglich direkt mit dem KNX-System über eine entsprechende Schnittstelle zu kommunizieren. Wegen des umfassenden Konfigurationsaufwands wird „ayControl“nur an Systemintegratoren vertrieben. Diese passen die Anwendung auf das entsprechende System beim Kunden an, erstellen z.B. die Benutzeroberfläche in einer Drittanbieteranwendung und schalten die Konfiguration über ein Webportal von „easyMobiz“frei (siehe Abbildung 2.12).

2.5 Zusammenfassung

Technologien für die Gebäudeautomation erhalten Einzug in immer mehr Wohnhäuser und gewerbliche Gebäude. Da der Aufwand zur Erstellung komplexer Steuerungs- und Überwachungssysteme mit der konventionellen Elektroinstallation überproportional steigt wird immer häufiger auf standardisierte Technologien zur Gebäudeautomation zurückgegriffen. Diese bieten ab einer bestimmten Komplexität des Systems eine einfachere und kostengünstigere Installation und eine hohe Flexibilität für Erweiterungen und Veränderungen des Umfeldes. Mit KNX/EIB und LON wurden zwei stark verbreitete Systeme für die Gebäudeautomation vorgestellt. Auch wenn diese technisch und elektrisch nicht zueinander kompatibel sind, so basieren sie doch auf ähnlichen Grundgedanken und unterscheiden sich nur in wenigen Merkmalen. LON bietet dabei die schnellere Datenübertragungsrate und eine größere Auswahl bei den Instandsetzungsanwendungen. KNX/EIB ist hingegen bereits länger ein internationaler Standard mit einer in Europa höheren Marktdurchdringung. Beide Technologien wären zur Umsetzung der proto-

Abbildung 2.12: Beispiel nativer mobiler KNX-Anwendungen (m..Remote und ayControl) [ibs][eas]

typischen Anwendung geeignet. Auch hier werde ich mich aus Gründen des Umfangs auf eine Technologie beschränken. Das KNX/EIB-System scheint mir aufgrund der höheren Akzeptanz in Europa und der größeren Auswahl an Geräten, die diese Technik unterstützen, geeigneter zur Umsetzung des Prototypen.

Zur Steuerung und Überwachung der eben genannten Gebäudeautomationssysteme sind schon einige Anwendungen am Markt verfügbar. Diese setzen aber meist auf zwischengeschaltete Systeme wie z.B. einen zentralen Rechner im lokalen Netzwerk oder müssen kompliziert über Anwendungen Dritter oder Webportale der entsprechenden Anbieter konfiguriert werden. Aus diesen Gründen soll ein anderer Ansatz zur Erfüllung dieser Aufgaben gefunden werden.

Kapitel 3

Anforderungsanalyse

Im vorherigen Kapitel wurden potenziell nutzbare Technologien besprochen und Beispielsysteme zur Steuerung von Gebäudetechnik mit mobilen Endgeräten betrachtet. Auf dieser Grundlage sollen nun die Anforderungen an den zu entwickelnden Prototypen formuliert werden. Um hier einen detaillierten Anforderungskatalog erstellen zu können, müssen auch das anvisierte Einsatzgebiet und die Rahmenbedingungen erörtert werden. Dadurch sollen potentielle Herausforderungen und Probleme schon vor der Konzeption des Systems ins Auge gefasst werden.

3.1 Anwendungsumgebung

In diesem Abschnitt soll die Umgebung, in der die prototypisch zu entwickelnde Anwendung einmal eingesetzt werden soll, genauer betrachtet werden. Dazu muss zum einen das spätere Einsatzgebiet herausgestellt werden. Es muss erarbeitet werden, welche Personengruppen in welcher Umgebung welchen Umfang an Funktionalitäten nutzen sollen. Zum anderen sollen auch die Rahmenbedingungen betrachtet werden. Hier sollen die vorausgesetzten Systeme und deren Konfiguration und die benötigten Datenquellen, die von der Anwendung verarbeitet werden, beschrieben werden.

3.1.1 Einsatzgebiet

Die Anwendung soll im Umfeld der Gebäudeautomation im Wohn- und Zweckbau eingesetzt werden. Durch den Einsatz der Anwendung soll es Technikern bzw. Systemintegratoren ermöglicht werden ein Gebäudeautomationssystem beim Kunden ohne umfangreichen Konfigurationsaufwand grundlegend steuern und überwachen zu können um so eventuell erste Hinweise auf eine Fehlkonfiguration bzw. -funktion zu bekommen. Zudem soll über die Anwendung eine einfache Integration mobiler Endgeräte in neue und bestehende Gebäudeautomationssysteme möglich sein.

Grundfunktionalitäten

Über die mobile Anwendung sollen Gebäudeautomationssysteme, die dem KNX-Standard entsprechen, gesteuert und überwacht werden können. Insbesondere sind die Grundfunktionalitäten wie z.B. das Schalten von Lichtquellen und das Auslesen von Temperatursensoren ohne zwischengeschaltete Systeme möglich. Die Anwendung kommuniziert über das mobile Endgerät direkt mit dem Gebäudeautomationssystem. Dadurch soll es Fachpersonal möglich sein, mit wenig Konfigurationsaufwand auf die Grundfunktionalitäten eines Gebäudeautomationssystems zuzugreifen. Auf der anderen Seite sollen Endanwender ohne weitreichende Kenntnisse der Systeme in der Lage sein, ein mobiles Endgerät in ein bestehendes Gebäudeautomationssystem zu integrieren.

Erweiterte Funktionalitäten

Durch die Erweiterung des Funktionsumfangs über die im letzten Abschnitt angesprochenen Grundfunktionalitäten hinaus, ist es dem Anwender möglich Ereignisse aus dem Gebäudeautomationssystem heraus zu empfangen ohne, dass sich das iPhone gezwungenermaßen im gleichen lokalen Netzwerk befindet und die mobile Anwendung aktiv ist. Dazu ist aber der Einsatz eines lokalen Servers nötig, der alle Nachrichten des Gebäudeautomationssystems verarbeitet und sie an das mobile Endgerät über geeignete Techniken weiterleitet.

Ist das mobile Endgerät mit dem lokalen Netzwerk verbunden und somit in der Lage die KNX-Telegramme zu empfangen und zu verarbeiten, so ist es auch denkbar, dass zuvor definierte Telegramme bestimmte Aktionen in

der mobilen Anwendung auslösen. Das Anzeigen eines Live-Streams einer Webcam beim Auslösen einer im KNX-System integrierten Klingelschaltung soll als Beispiel für eine solche Funktionalität dienen.

3.1.2 Rahmenbedingungen

Zur Umsetzung des Prototyps werden bestimmte Rahmenbedingungen vorausgesetzt. Wie aus Kapitel 2 hervorgegangen ist, wird die Anwendung für die iPhoneOS-Plattform und insbesondere das iPhone 3GS entwickelt. Derzeitig ist diese Plattform in der Version 3.1.3 aktuell. Zudem wird eine Elektroinstallation nach KNX-Standard ISO/IEC 14543-3 mit mindestens einer Lichtquelle, einem Temperatursensor sowie den entsprechenden Aktoren und Schaltern vorausgesetzt. Ein KNXnet/IP-Router soll den Übergang zum Übertragungsmedium Ethernet gewährleisten. Die Anbindung des iPhones erfolgt über einen WLAN-Router (siehe Abbildung 3.1). Dieser muss die Weiterleitung von Multicast-IP-Paketen beherrschen und als VPN-Gateway agieren können. Da ein Gebäudeautomationssystem nach KNX-Standard intern nur eine Datenübertragungsrate von 9,6kBit/s unterstützt, unterliegt die Bandbreite zum mobilen Endgerät ihm Rahmen der Grundfunktionalitäten keinerlei besonderer Beschränkungen. Gerade im Hinblick auf die zuvor angesprochenen erweiterten Funktionalitäten wie z.B. der Anzeige von Live-Streams einer Webcam, ist eine Anbindung des mobilen Endgerätes über eine breitbandige mobile Drahtlostechnologie wie UMTS von Vorteil.

Neben diesen hardwaretechnischen Voraussetzungen ergeben sich noch weitere Voraussetzungen an die Konfiguration der verschiedenen Systeme und an die Datenquellen. Hier sind zwei Konfigurationsdateien aus der in Unterabschnitt 2.3.1 beschriebenen Software ETS zu nennen. In der Geräte-Konfigurationsdatei werden Geräte, die im KNX-System verfügbar sind, aufgelistet und mit Merkmalen beschrieben (vgl. Listing 3.1). Zu diesen Merkmalen gehören unter anderem die physikalische Adresse im KNX-System, der Hersteller und der Gerätename sowie der verwendete Datenpunkttyp[1] des Gerätes. Eine Pflege des Datenpunkttyps im ETS-Projekt ist hierbei besonders wichtig um aus diesem Merkmal geeignete Interaktionselemente für die KNX-Geräte auf der Benutzeroberfläche der Anwendung anzuzeigen. In der zweiten Konfigurationsdatei sind die Gruppenadressen des KNX-Systems angegeben.

[1]Der Datenpunkttyp ist Teil des KNX-Standards und beschreibt wie die Nutzdaten eines KNX-Telegramms zu interpretieren sind. So wird beispielsweise über den Datenpunkttyp "14.068", der einem KNX-Gerät zugeordnet wird, beschrieben, dass die Nutzdaten als "Common Temperature" zu interpretieren sind [KNX09, S. 13].

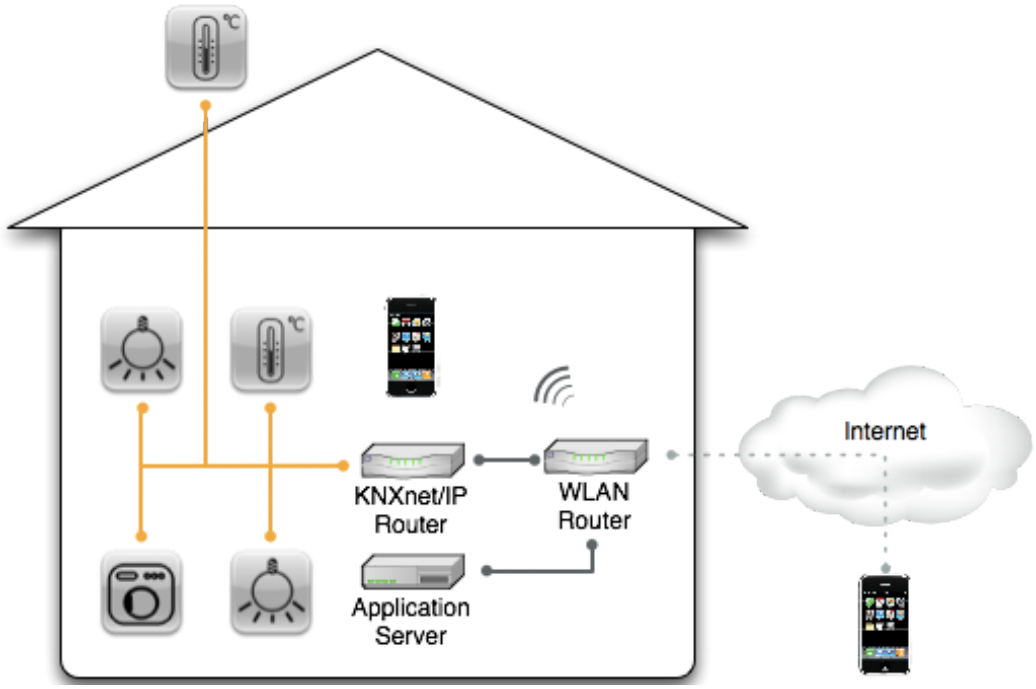


Abbildung 3.1: Darstellung der Anwendungsumgebung

Gruppen stellen dabei eine Zusammenfassung von Geräten nach Funktionen dar. So kann z.B. ein einzelner Temperatursensor über eine Gruppe "Außentemperatur" oder aber auch eine Zusammenfassung mehrerer Geräte, üblicherweise ein bzw. mehrere Schalter und ein bzw. mehrere Schaltaktoren, abgebildet werden. Üblicherweise werden in KNX-Systemen nicht einzelne Geräte über deren physikalische Adresse sondern Funktionen bzw. Gruppen über deren Gruppenadresse angesprochen und stellen somit die typische Form der Adressierung dar. Aus diesem Grund kann die Bezeichnung einer Gruppe dazu verwendet werden über eine definierte Syntax der mobilen Anwendung Parameter zum Ausführen bestimmter Funktionen, z.B. der URL einer Webcam, zu übergeben. Diese Syntax kann beim Einlesen der Gruppenkonfigurationsdatei verarbeitet werden und bei Erhalt von Nachrichten von dieser Gruppe definierte Aktionen auslösen. Eine weitere Datenquelle stellen die KNX-Telegramme der KNX-Geräte dar. Diese enthalten neben den eigentlichen Nutzdaten noch weitere Informationen über den Sender, den Empfänger und den Telegrammtyp. Der Telegrammtyp untergliedert sich dabei in Lese- (Read), Schreib- (Write) und Antworttelegramme (Response). Von der mobilen Anwendung werden ausschließlich Lese- und Schreibtelegramme über den KNXnet/IP-Router an die entsprechenden Gruppenadressen gesendet wer-

den. Diese beantworten entweder die Lesetelegramme mit einem Antworttelegramm oder setzen die entsprechenden Werte der Schreibtelegramme an den Geräten der Gruppe um.

```
<row>
  <colValue nr="1">1.11.40 Enthalpie-Sensor HLK Aussen</colValue>
  <colValue nr="2">0</colValue>
  <colValue nr="3">Ti Messwert</colValue>
  ...
  <colValue nr="6">0/0/2</colValue>
  <colValue nr="7">4 Byte</colValue>
  ...
  <colValue nr="13">4 byte float value DPT_Value_Common_Temperature</colValue>
  <colValue nr="14">Niedrig</colValue>
</row>
```

Listing 3.1: Auszug aus der Geräte-Konfigurationsdatei der ETS

3.2 Systemanforderungen

Aus der Aufgabenstellung, intelligente Gebäudetechnik über mobile Endgeräte steuern und überwachen zu können, und der soeben analysierten Anwendungsumgebung lassen sich Anforderungen an das zu entwickelnde System ableiten.

Die Geräte eines KNX-Systems sollen sich direkt vom mobilen Endgerät aus steuern und überwachen lassen. Für diesen Funktionsumfang sollen keine zusätzlichen Systeme wie z.B. Server-Anwendungen nötig sein. Die Konfiguration der mobilen Anwendung soll einzig und allein über den Import von zwei definierten Konfigurationsdateien, die durch die Software ETS erzeugt werden, erfolgen. Die Benutzeroberfläche soll aus den Informationen der Konfigurationsdateien erzeugt werden und dem Anwender geeignete Präsentations- und Interaktionselemente zur Steuerung und Überwachung der KNX-Geräte zur Verfügung stellen.

Das Auslesen und Ändern des Status bzw. der Sensorenwerte von KNX-Geräten soll ausschließlich auf Anfrage der mobilen Anwendung erfolgen. Ist die Anwendung nicht aktiv oder nicht mit dem lokalen Netzwerk verbunden, können Telegramme des KNX-Systems ohne ein geeignetes zwischengeschaltetes System nicht verarbeitet werden.

Eine optional zu entwickelnde Server-Anwendung kann alle Telegramme des KNX-Systems empfangen und verarbeiten. Sollte sich ein für diesen Dienst registriertes mobiles Endgerät nicht im lokalen Netzwerk befinden soll es ausfindig gemacht und Meldungen an das Gerät gesendet werden.

3.3 Analyse der Anwendungsfälle

Aus den Erkenntnissen, die aus der zuvor besprochenen Anwendungsumgebung und den Rahmenbedingungen gezogen werden können, sind im Gesamtsystem grundlegend zwei Szenarien zu identifiziert. Zum Einen wird die iPhone-Anwendung im lokalen Netzwerk betrieben oder ist durch geeignete Technologien wie VPN mit diesem verbunden. Innerhalb dieses Szenarios sind alle Funktionalitäten, wie z.B. das Auslesen von Sensorenwerten, das Schalten von Aktoren oder der Empfang von Telegrammen aus dem KNX-System heraus, verfügbar. Das zweite Szenario beschreibt die Situation, wenn die iPhone-Anwendung nicht mit dem lokalen Netzwerk verbunden oder nicht aktiv ist. Da in diesem Fall ein Empfang der Telegramme des KNX-Systems nicht möglich ist, sind Zwischensysteme nötig, die die Anwendung über den Eintritt bestimmter Ereignisse innerhalb des KNX-Systems informieren.

3.3.1 Szenario 1: Anwendung läuft im lokalen Netzwerk

Die folgenden Anwendungsfälle beschreiben das Szenario, wenn die iPhone-Anwendung im lokalen Netzwerk betrieben wird. Für die prototypische Umsetzung des Systems wird hier nur auf einen eingeschränkten Teil der möglichen Aktionen eingegangen (siehe Abbildung 3.2).

Die Anwendungsfälle "Gerätestatus erfragen" und "Gerätestatus ändern" umfassen generell die Möglichkeit eine Gerätegruppe innerhalb des KNX-Systems über eine Gruppenadresse anzusprechen und dessen Status abzufragen bzw. zu ändern. Mit dem Prototypen soll mindestens das Auslesen der Daten von Temperatursensoren und Lichtquellen sowie das Schalten von Lichtquellen möglich sein. Die beiden genannten Anwendungfälle beschreiben die Situation, wenn eine Anfrage oder das Ändern des Status von der iPhone-Anwendung aus initiiert wurde. Dazu werden von der iPhone-Anwendung Lese- bzw. Schreibtelegramme erzeugt und versendet. Die die im Telegramm

adressierte Gruppe beantwortet Lesetelegramme daraufhin mit einem entsprechenden Antworttelegramm oder setzt den Wert, der im Schreibtelegramm übermittelt wurde, an den Aktoren um. Zusätzlich ist es auch denkbar, dass Telegramme, die aus dem KNX-System heraus erzeugt werden, empfangen und verarbeitet werden können. Dabei könnten bestimmte Telegramme vordefinierte Aktionen auslösen wie z.B. das Anzeigen einer Meldung oder einer Webpage. Zur Umsetzung dieser Funktionalität ist aber ein Mechanismus nötig, über den die iPhone-Anwendung Informationen über die Art der auszuführenden Aktionen beim Empfang bestimmter Telegramme erhält. Dies wäre über eine definierte Syntax des Gruppenbezeichners möglich. Als letzten Anwendungsfall dieses Teilsystems betrachten wir die Aktion "ETS-Konfigurationsdateien einlesen". Der Anwender übergibt der iPhone-Anwendung zwei Konfigurationsdateien, die durch die ETS-Software für ein KNX-System erstellt werden. Diese Konfigurationsdateien werden verarbeitet und eine Benutzeroberfläche mit geeigneten grafischen Benutzerelementen, über die die Geräte steuer- und überwachbar sind, erstellt.

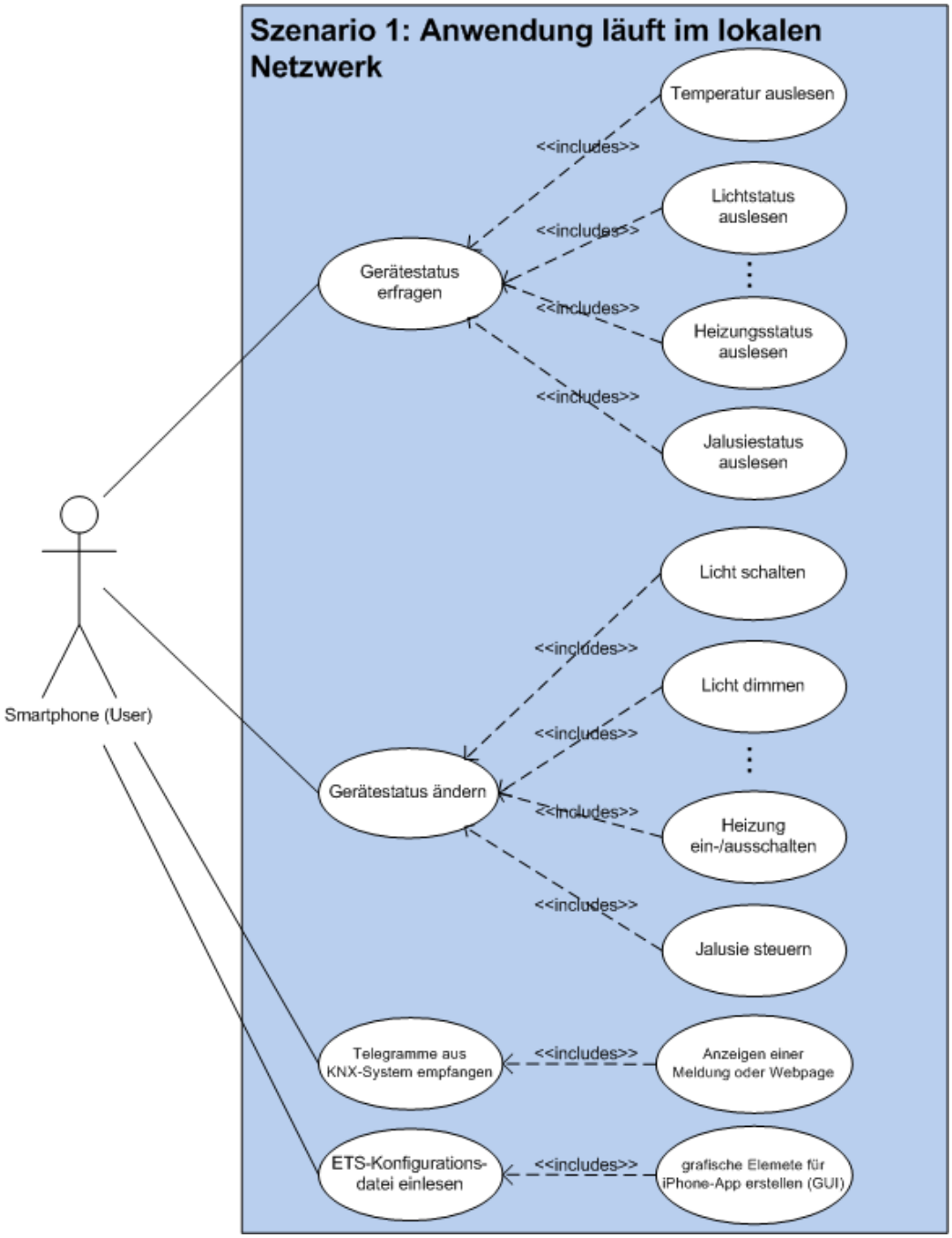


Abbildung 3.2: Anwendungsfälle des Szenario 1

3.3.2 Szenario 2: Anwendung läuft nicht im lokalen Netzwerk oder ist inaktiv

Im zweiten Szenario ist das Senden bzw. Empfangen von KNX-Telegrammen nicht möglich, da die Verbindung zum lokalen Netzwerk und somit zum KNX-System nicht gegeben ist. Denkbar wäre aber die Situation, dass auch dann zuvor definierte Telegrammarten an die mobile Anwendung ausgeliefert werden sollen. Als Beispiel kann hier das Auslösen einer KNX-fähigen Alarmanlage dienen. Über eine Server-Anwendung, die im lokalen Netzwerk betrieben wird, könnten solche Telegrammarten definiert und die entsprechenden Telegramme empfangen und verarbeitet werden. Über geeignete Techniken könnte diese Server-Anwendung zuvor registrierte iPhones ausfindig machen und eine entsprechende Meldung senden (siehe Abbildung 3.2).

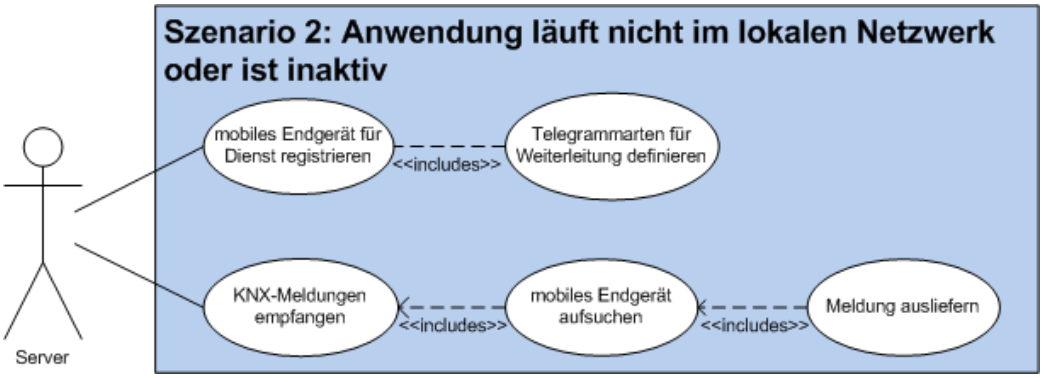


Abbildung 3.3: Anwendungsfälle des Szenario 2

Kapitel 4

Systementwurf

Im vorherigen Kapitel 3 wurden das Umfeld, in der die Anwendung betrieben wird, sowie die Rahmenbedingungen und die Anforderungen, die an das System gestellt werden, spezifiziert. Anhand der zwei entworfenen Szenarien können drei Teilsysteme ermittelt werden. Im Szenario 1 (vgl. Unterabschnitt 3.3.1) sind das die iPhone-Anwendung und das KNX-Gebäudeautomationssystem. Im Szenario 2 (vgl. Unterabschnitt 3.3.2) wird zusätzlich eine Server-Anwendung betrieben. In diesem Kapitel soll nun eine geeignete System-Architektur gewählt und Schnittstellen zwischen den einzelnen Teilsystemen definiert werden. Durch das Festlegen dieser Schnittstellen soll eine reibungslose Kommunikation zwischen den Teilsystemen sichergestellt werden.

4.1 Architektur

Im Gesamtsystem existiert mit dem KNX-Gebäudeautomationssystem ein Teilsystem, dass weitestgehend fest vorgegeben ist und im Rahmen der Systemimplementierung auch nicht verändert wird. Der KNX-Standard definiert mit den in Abschnitt 2.3 beschriebenen Telegrammen eine Schnittstelle zur Kommunikation mit dem System und bietet mit unterschiedlichen Geräten, in unserem Fall dem KNXnet/IP-Router, einen Zugriffspunkt zu diesem. Die Beschreibung des KNX-Systems erfolgt über Konfigurationsdateien in definierten Austauschformaten wie CSV[1] und XML[2].

[1] Comma-Seperated Values.
[2] Extensible Markup Language.

Durch die Eigenständigkeit dieses Teilsystems entsteht zwangsläufig eine verteilte Architektur des Gesamtsystems. Die Daten werden hierbei über eine drahtlose Schnittstelle ausgetauscht. Die mobile Anwendung dient dabei als visuelle Repräsentation des KNX-Systems. Zudem ist sie für das Erzeugen und Senden von Telegrammen zum Auslesen bzw. Ändern des Status von Geräten innerhalb des KNX-Systems sowie den Empfang von Telegrammen aus dem System zuständig. Im zweiten Szenario kommt eine Server-Anwendung hinzu, welche alle Telegramme des KNX-Systems empfängt, verarbeitet und ggf. an mobile Endgeräte weiterleitet (vgl. Abbildung 4.1).

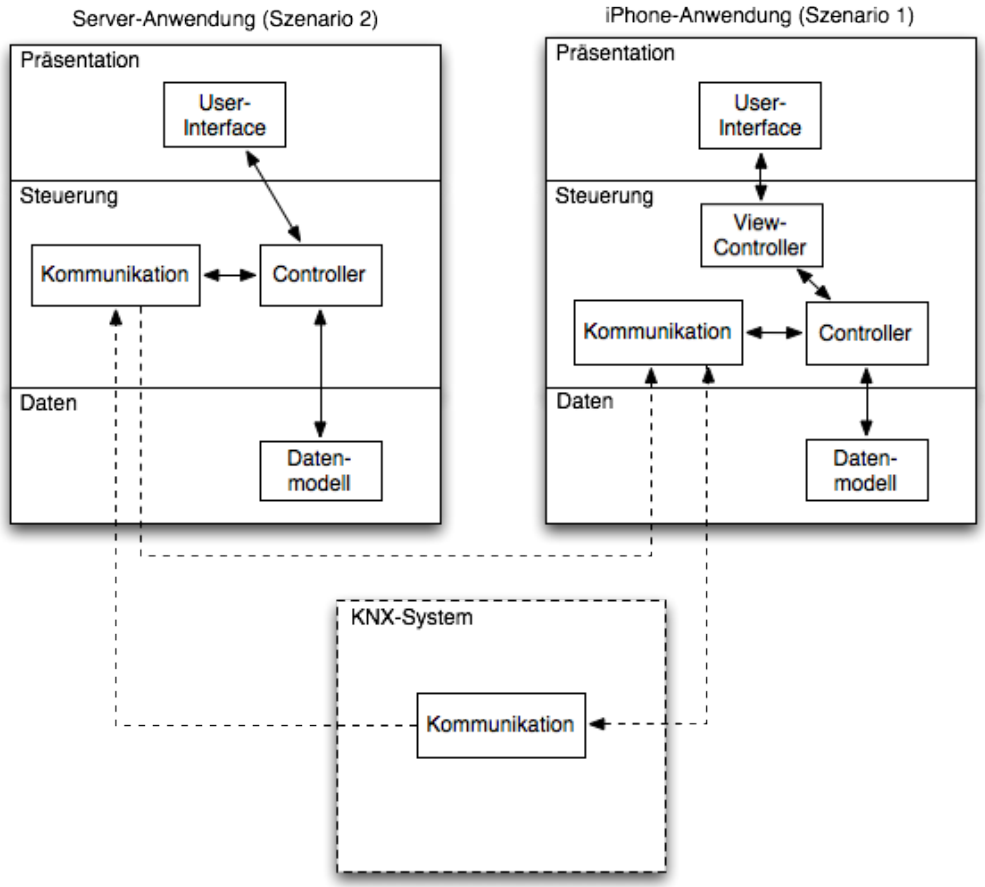


Abbildung 4.1: Architektur des Systems

Wie soeben beschrieben, sind die Komponenten und Schnittstellen des KNX-Systems nicht veränderbar und müssen als gegeben hingenommen werden. Die beiden anderen Teilsysteme können hingegen frei entworfen werden. Ein

geeignetes Architekturmuster für diese Teilsysteme stellt das *Model-View-Controller* (MVC) Architekturmuster dar. Bei diesem wird jedes System in drei Schichten unterteilt (siehe Abbildung 4.2). Jede dieser Schichten ist in sich abgeschlossen und stellt Funktionalitäten ausschließlich über definierte Schnittstellen zur Verfügung. Dadurch entsteht eine klare Strukturierung und die Wiederverwendbarkeit bzw. Austauschbarkeit einzelner Komponenten wird erleichtert, wodurch die Flexibilität des Systems insgesamt steigt. Im MVC-Architekturmuster werden dabei alle Objekte einer der drei Rollen *Daten (Model)*, *Präsentation (View)* und *Steuerung (Controller)* zugeordnet [App09b].

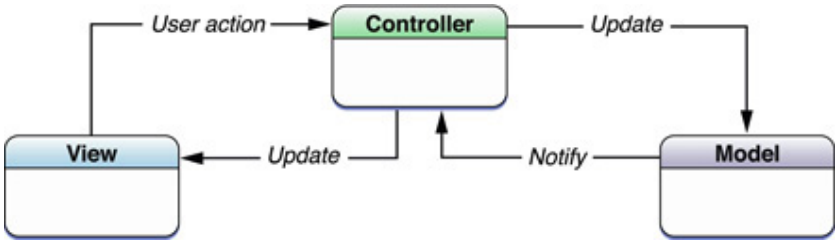


Abbildung 4.2: MVC-Architekturmuster nach Apple [Appd]

Das MVC-Architekturmuster, das bei der iPhoneOS-Plattform Anwendung findet [App09b], unterscheidet sich dabei etwas vom klassischen MVC-Modell. Während im klassischen Modell die *View*-Komponente direkt mit der *Model*-Komponente kommunizieren kann, ist die Kommunikation im modifizierten Modell ausschließlich über die *Controller*-Komponente möglich. Dadurch kann die Wiederverwendbarkeit und Erweiterbarkeit der einzelnen Komponenten verbessert werden. Die Anwendung des MVC-Architekturmusters ist für Anwendungen, die auf der iPhoneOS-Plattform basieren, zudem obligatorisch [App09b]. Für die Server-Anwendung, die im zweiten Szenario benötigt wird, kann das MVC-Modell ebenfalls Anwendung finden, da es zu einer übersichtlichen Struktur und besser definierten Schnittstellen zwischen den einzelnen Komponenten beiträgt. Da eine Implementierung der Server-Anwendung den Umfang dieser Arbeit übersteigen würde, wird in den folgenden Abschnitten nur grundlegend auf die Architektur dieses Teilsystems eingegangen.

4.1.1 Datenschicht

Die Objekte der *Datenschicht* kapseln die Daten der Anwendung und stellen Funktionalitäten zur Verarbeitung der Daten zur Verfügung. Zudem wird in dieser Schicht die persistente Speicherung von Daten, die über die Laufzeit der Anwendung hinaus verfügbar sein sollen, sichergestellt. Im Teilsystem der iPhone-Anwendung zählen dazu die Informationen aus der Gruppen- und Gerätekonfigurationsdatei sowie grundlegende Informationen zum KNX-System, die der Anwender angeben muss. Innerhalb der Server-Anwendung werden nur wenige Informationen zum mobilen Endgerät sowie den weiterzuleitenden Telegrammarten dauerhaft benötigt. Zwischen den einzelnen Teilsystemen werden KNX-Telegramme bzw. Repräsentationen dieser ausgetauscht, die keiner persistenten Speicherung bedürfen.

Datenmodell der iPhone-Anwendung

Wie schon zuvor angesprochen wurde, unterteilt sich die *Datenschicht* der iPhone-Anwendung in eine persistierte Abbildung der Informationen aus der Gruppen- und Gerätekonfigurationsdatei und den nicht-persistierten KNX-Telegrammen.

Die Telegramme sind im KNX-Standard beschrieben und besitzen einen festen Aufbau (siehe Abbildung 4.3). Für das Überwachen und Steuern von KNX-Geräten sind die drei Telegrammtypen *Read*, *Write* und *Response* wichtig. Während *Read*- und *Write*-Telegramme von der iPhone-Anwendung erzeugt und an das KNX-System gesendet werden, werden *Response*-Telegramme aus dem KNX-System empfangen und von der Anwendung verarbeitet. Die Nutzdaten werden anhand des Datenpunkttyps der Empfängergruppe des Telegramms interpretiert und verarbeitet (vgl. Unterabschnitt 3.1.2). Weitere wichtige Informationen in einem KNX-Telegramm sind neben dem Telegrammtyp und den Nutzdaten die Quell- und Zieladresse. Die Telegramme werden von einer Komponente der *Datenschicht* erzeugt oder verarbeitet und danach verworfen.

Das Datenmodell der persistent zu speichernden Daten der beiden Konfigurationsdateien orientiert sich stark an der Topologie eines KNX-Systems (vgl. Abbildung 2.9). Wie im Unterabschnitt 3.1.2 herausgestellt wurde, werden über die Gruppen Funktionen des KNX-Systems abgebildet und über die Gruppenadressen angesprochen. Diese untergliedern sich in zwei oder üblicherweise drei Hierarchiestufen, den Haupt-, Mittel- und Untergruppen.

Kontrollfeld (Steuerfeld)	Quelladresse (B.L.T.)	Zieladresse (B.L.T. oder HIU)	Adressum-schaltung	Routing-zähler	Länge Nutzinfo	Nutzinfo	Sicherung
			1 Bit	3 Bit	4 Bit		
8 Bit (1 Byte)	16 Bit (2 Byte)	16 Bit (2 Byte)	8 Bit (1 Byte)			2 bis16 Byte	8 Bit (1 Byte)
A	B	C	D			E	F

Abbildung 4.3: Aufbau eines KNX-Telegramms [FM]

Diese Hierarchie wird meist nach räumlichen und funktionalen Kriterien aufgebaut. So kann durch die Hauptgruppe die räumliche Trennung, z.B. „Obergeschoss“bzw. „Untergeschoss“usw., abgebildet werden und die Mittelgruppe nach Funktionsbereichen wie „Licht“oder „Jalousiesteuerung“gegliedert sein. Die Untergruppen bilden dann die einzelnen, steuerbaren Funktionen wie „Wohnzimmerlicht dimmen“ab. Diese Gliederung findet sich auch im Datenmodell der iPhone-Anwendung wieder. Ein KNX-System wird in der Anwendung als Projekt angelegt. Der Anwender muss dazu einen Projektnamen und wenige Informationen wie die physikalische KNX-Adresse des KNXnet/IP-Routers angeben. Alle weiteren Informationen werden aus der Gruppen- bzw. Gerätekonfigurationsdatei gelesen.

Listing 4.1 zeigt den typischen Aufbau einer Gruppenkonfigurationsdatei im CSV-Format. Der Aufbau einer Gerätekonfigurationsdatei wurde im Unterabschnitt 3.1.2 beschrieben (vgl. Listing 3.1). Aus diesen Informationen wird eine Datenstruktur erstellt, die das KNX-System abbildet. Ausgangspunkt ist dabei das Projekt. Analog zu den Hierarchiestufen der Gruppen sind einem Projekt eine oder mehrere Hauptgruppen zugeordnet. Diese enthalten eine oder mehrere Mittelgruppen, die ihrerseits wiederum aus ein oder mehreren Untergruppen bestehen. Die jeweiligen Gruppen sind durch einen Namen und eine Gruppennummer aus der Gruppen-Konfigurationsdatei gekennzeichnet. Jede Untergruppe hält weitere Informationen wie den Datenpunkttyp der Gruppe und weitere Geräteinformationen vor. Diese Datenstruktur wird zur persistenten Speicherung beim Beenden der Anwendung in ein XML-Format geschrieben und im Dateisystem des iPhones abgelegt. Beim erneuten Ausführen der Anwendung werden diese Daten geladen, so dass ein erneutes einlesen der Konfigurationsdateien überflüssig ist.

```
1  "Main";"Middle";"Sub";"Address"
2  "Haus 1";  ;  ;"0/-/-"
3   ;"Steuerung";  ;"0/0/-"
4   ;  ;"Licht schalten";"0/0/1"
5   ;  ;"Licht dimmen";"0/0/2"
6   ;"Sensoren";  ;"0/1/-"
7   ;  ;"Temperatur innen";"0/1/1"
```

Listing 4.1: Auszug aus der Gruppen-Konfigurationsdatei der ETS

Datenkonzept der Server-Anwendung

Die Server-Anwendung, die im zweiten Szenario benötigt wird, muss nur wenige Daten persistent speichern. Dazu gehören Informationen zum mobilen Endgerät wie z.B. der festen IP-Adresse im lokalen WLAN und die Mobilfunknummer sowie Informationen zu den Gruppen, deren Telegramme weitergeleitet werden sollen. Diese können in einer geeigneten Struktur abgebildet und in einem Austauschformat wie XML oder CSV gespeichert werden.

Zudem müssen *Write*- bzw. *Response*-Telegramme aus dem KNX-System empfangen und verarbeitet sowie in einer entsprechenden Repräsentation an das mobile Endgerät weitergeleitet werden können. Diese Telegramme können anschließend verworfen werden.

4.1.2 Steuerungsschicht

Die Objekte der *Steuerungsschicht* agieren als Vermittler zwischen den Objekten der *Daten*- und der *Präsentationsschicht* und koordinieren die Abläufe innerhalb der Anwendung während der Laufzeit. Folglich müssen diese Objekte die Daten der *Datenschicht* verarbeiten und den Objekten der *Präsentationsschicht* bei Bedarf zur Verfügung stellen. Zudem muss hier auf Ereignisse wie Nutzerinteraktionen, die in der Präsentationsschicht ausgelöst werden, reagiert werden.

Weiterhin sind die Objekte dieser Schicht für die Initialisierung aller benötigten Objekte beim Start der Anwendung zuständig. Dazu gehören unter anderem der Start des UDP[3]-Servers der Kommunikationskomponente und

[3]User Datagram Protocol

das Einlesen einer Projektliste, in der alle zuvor konfigurierten Projekte aufgeführt sind. Beim Beenden der Anwendung sind weitere Aktionen nötig, die von den Steuerungsobjekten koordiniert werden. Dazu gehören z.B. das Beenden des UDP-Servers und das persistente Speichern der Projektliste.

Konfiguration eines Projektes

Wie zuvor in Unterabschnitt 4.1.1 beschrieben wurde, ist das Projekt eine der zentralen Datenstrukturen der Anwendung und bildet ein KNX-Gebäudeautomationssystem ab. Zur Konfiguration eines neuen Projektes werden Eingaben des Anwenders von der Präsentationsschicht verarbeitet. Dazu gehören z.B. der Projektname und die URLs, unter denen die beiden Konfigurationsdateien verfügbar sind. In einem ersten Schritt wird die Gruppenkonfigurationsdatei verarbeitet und eine grundlegende Datenstruktur, die alle Informationen zu den verfügbaren Haupt-, Mittel- und Untergruppen beinhaltet, aufgebaut. Anschließend wird diese Datenstruktur durch Informationen aus der Gerätekonfigurationsdatei ergänzt. Anschließend wird die komplette Datenstruktur unter dem angegebenen Projektnamen in ein XML-Format geschrieben und persistent im Dateisystem des iPhones abgelegt (siehe Abbildung A.2). Abschließend wird der Projektname zur Projektliste, welche die Namen aller konfigurierten Projekte enthält, hinzugefügt.

UDP-Kommunikation

Eine weitere wichtige Aufgabe der Steuerungsschicht ist das Zurverfügungstellen von Funktionalitäten zum Senden und Empfangen von UDP-Multicast-Paketen über welche mit dem KNXnet/IP-Router kommuniziert wird. Dazu wird eine Serverkomponente bereitgestellt, die es ermöglicht, einer Mulitcast-Gruppe beizutreten und entsprechende Nachrichten dieser zu empfangen bzw. an diese zu senden. Wie aus Unterabschnitt 4.1.1 bekannt ist, ist der Aufbau eines KNX-Telegramms über einen Standard fest definiert. Durch den KNXnet/IP-Router werden diese Telegramme in UDP-Pakete verpackt und über die, dem KNX-Standard fest zugeteilte, Multicast-Adresse *224.0.23.12* verteilt. Über die Serverkomponente werden diese Daten empfangen und ausgewertet.

Empfangsseitig werden von der iPhone-Anwendung nur *Write-* bzw. *Response-*Telegramme verarbeitet, alle anderen Pakete werden verworfen. Anschließend wird ermittelt, ob der Absender des KNX-Telegramms in der Datenstruktur

des aktuellen Projekts vorhanden ist. Ist das der Fall, werden die Nutzdaten des Telegramms in der Datenstruktur zur entsprechenden Gruppe hinzugefügt. Wird diese Gruppe zudem gerade in der Präsentationsschicht über die Gruppendetailansicht angezeigt, wird eine entsprechende Nachricht an das Präsentationsobjekt zur Aktualisierung der Ansicht geschickt (siehe Abbildung A.4).

Das Senden von KNX-Telegrammen wird durch Ereignisse, die auf der Präsentationsschicht ausgelöst werden, angestoßen. Zum Beispiel werden bei der Auswahl eines Projektes *Read*-Telegramme für jede Untergruppe innerhalb dieses Projektes erzeugt und versendet. Zudem kann der Anwender das Senden von *Read*- bzw. *Write*-Telegrammen direkt durch die Aktionen *Reload* bzw. *Change* in der Präsentationsschicht auslösen. Dadurch kann der Status einer Gruppe abgefragt bzw. geändert werden. Abhängig vom Telegrammtyp wird ein Telegramm mit Nutzdaten (*Write*) oder ohne (*Read*) erzeugt, in ein UDP-Paket verpackt und an die KNX-Multicast-Adresse gesendet (siehe Abbildung A.3).

Konzept zum Auffinden mobiler Endgeräte durch eine lokale Server-Anwendung

Wie in Unterabschnitt 3.3.2 herausgestellt wurde, ist das direkte Ausliefern von Telegrammen an ein mobiles Endgerät nicht möglich, wenn dieses nicht mit dem lokalen Netzwerk verbunden und die mobile Anwendung nicht aktiv ist. Hier wird mit einem lokalen Server ein zwischengeschaltetes System nötig, welches alle Telegramme empfangen und ggf. weiterleiten kann. Dazu muss ein Mechanismus konzipiert werden, der es ermöglicht zu ermitteln, ob sich ein mobiles Endgerät im lokalen Netzwerk befindet auf dem die mobile Anwendung aktiv ist. Ist dies der Fall, so soll durch die Server-Anwendung keine Mitteilung an das Endgerät weitergeleitet werden, da dieses selbst die Möglichkeit hat die Telegramme des KNX-Systems zu empfangen. Eine Auslieferung von Meldungen hätte in diesem Fall die Folge, dass die durch die Telegramme ausgelösten Ereignisse mehrfach in der mobilen Anwendung ausgelöst werden. Im anderen Fall, wenn die Anwendung nicht aktiv oder das mobile Endgerät nicht mit dem lokalen Netzwerk verbunden ist, soll die Server-Anwendung für zuvor definierte Telegramme eine Nachricht an registrierte mobile Endgeräte senden. Für das iPhone steht mit dem Apple Push Notification Service (APNS) eine geeignete Technologie zur Verfügung, die es ermöglicht Nachrichten an eine nicht aktive Anwendung auf dem mobilen Endgerät zu senden (siehe Abbildung 4.4).

Abbildung 4.4: Prozesskette beim Ausliefern von Nachrichten über die Server-Anwendung

4.1.3 Präsentationsschicht

Die Objekte der *Präsentationsschicht* bilden die Schnittstelle zwischen dem Anwender und dem System. Über diese Benutzerschnittstelle (User Interface, UI) können die Daten der *Datenschicht* präsentiert und Interaktionen des Benutzers entgegengenommen werden. Die Weiterverarbeitung dieser Daten ist dabei nicht Teil dieser Anwendungsschicht.

Die Benutzerschnittstellen in dem zu entwickelnden System stellen die Daten ausschließlich grafisch über den Bildschirm der entsprechenden Geräte dar. Neben einer Benutzerschnittstelle für die iPhone-Anwendung zur Visualisierung des Systems und der Interaktion mit diesem ist zudem eine entsprechende Schnittstelle für die Server-Anwendung aus dem zweiten Szenario nötig.

Benutzerschnittstelle der iPhone-Anwendung

Beim Entwurf der Benutzerschnittstelle ist darauf zu achten, dass der Anwender nicht von Informationen überflutet wird und diese Informationen eindeutig präsentiert werden. Der Ansatz, durch einen minimalen Konfigurationsaufwand die Benutzung der mobilen Anwendung sehr einfach zu halten, würde durch eine undurchsichtige und komplizierte Benutzerschnittstelle konterkariert. Zudem sollten alle Interaktionselemente intuitiv bedienbar sein.

Mit den *iPhone Human Interface Guidelines* [App10b] definiert Apple Richtlinien zum Entwurf von Benutzerschnittstellen für Geräte der iPhoneOS-Plattform. Diese Richtlinien sollen den Entwickler einer Anwendung für diese Plattform sensibilisieren, die Benutzerschnittstelle nach ergonomischen Gesichtspunkten zu entwerfen und dabei auf die Besonderheiten der Geräte wie z.B. der Bildschirmgröße und -auflösung oder der Eingabemethode über Berührungen und Gesten zu achten. So werden unter anderem minimale Abmessungen für Interaktionselemente und Schriftgrößen definiert um eine Bedienung mit Fingern und eine optimale Lesbarkeit jederzeit zu gewährleisten. Zudem werden in diesen Richtlinien grafische Benutzerelemente wie z.B. Schalter und Tabellen definiert, so dass über den Wiedererkennungswert dieser Elemente die intuitive Bedienbarkeit der Anwendung erhöht wird.

Die Benutzerschnittstelle der iPhone-Anwendung untergliedert sich in mehrere Ansichten (siehe Abbildung 4.5) um zum einen die verschiedenen Funktionalitäten wie die Konfiguration oder die Steuerung und Überwachung der Geräte voneinander abzugrenzen. Zum Anderen können die benötigten Informationen auf mehrere Ansichten verteilt werden, was die Übersichtlichkeit der Darstellung erhöht.

In der iPhone-Anwendung sollen mehrere Projekte verwaltet werden können. Da ein Projekt den Ausgangspunkt zum Steuern und Überwachen der einzelnen Geräte bzw. Gerätegruppen darstellt, wird eine Liste mit allen

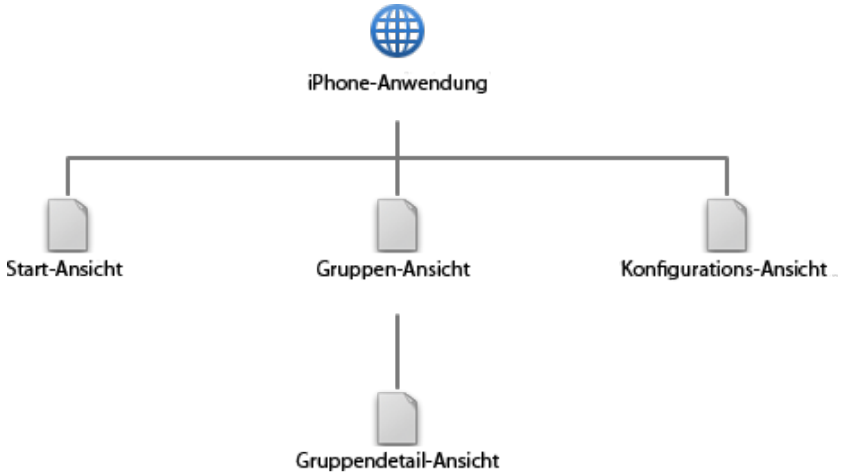


Abbildung 4.5: Darstellung der Abhängigkeiten der Ansichten (SiteMap)

konfigurierten Projekten auf der *Start-Ansicht* präsentiert. Diese Projekte sollen darüber hinaus unabhängig voneinander gelöscht werden können. Zudem können Texte zur Einleitung bzw. Beschreibung der Anwendung auf dieser Ansicht angezeigt werden (siehe Abbildung A.7). Auf einer weiteren Ansicht werden die Gruppen des aktuell gewählten Projektes aufgelistet (siehe Abbildung A.8). Wählt der Anwender auf dieser Ansicht eine der Gruppen aus, wird eine *Detailansicht* dieser Gruppe angezeigt (siehe Abbildung A.9). Die *Detailansicht* präsentiert übersichtlich die Informationen zur gewählten Gruppe. Hier wird dem Anwender der Status der Gruppe angezeigt und entsprechende Interaktionselemente zur Steuerung der Gruppe zur Verfügung gestellt. Diese sollten je nach Funktion, die durch die Gruppe ausgeführt werden soll, so ausgeprägt sein, dass eine intuitive Bedienung möglich ist (siehe Abbildung 4.6). Auf einer letzten Ansicht können, durch die Angabe weniger Informationen durch den Anwender, Projekte konfiguriert werden (siehe Abbildung A.10). Da sich die iPhone-Anwendung auf wenige Ansichten beschränkt und diese sehr flache Hierarchien (siehe Abbildung 4.5) aufweisen, ist eine einfache und intuitive Navigation über eine sog. „TabBar“, einer Navigationsleiste am unteren Bildschirmrand, möglich.

Benutzerschnittstelle der Server-Anwendung

Die Server-Anwendung ist zur Konfiguration des Dienstes zur Weiterleitung von KNX-Telegrammen auch auf eine Benutzerschnittstelle angewiesen. Da es sich um eine Web-Anwendung handelt, kann diese Benutzerschnittstelle

Abbildung 4.6: Auswahl von Interaktionselementen der Gruppendetailansicht

über einen Browser zur Verfügung gestellt werden. Die Schnittstelle muss Funktionalitäten zur Registrierung von mobilen Endgeräten und Telegrammarten anbieten. Dies kann jeweils über ein einfach gehaltenes Formular mit Feldern zur Texteingabe realisiert werden. Die Zuweisung der weiterzuleitenden Telegrammarten zum jeweiligen registrierten iPhone kann über zwei Listen erfolgen. In einer der Listen können die verfügbaren, in der zweiten Liste die dem entsprechende iPhone zugewiesenen Telegrammarten dargestellt werden. Die Zuweisung wird über geeignete Interaktionselemente wie Buttons ausgeführt oder zurückgenommen.

Kapitel 5

Implementierung

Nachdem im vorangegangenen Kapitel das System entworfen wurde, soll nun die Umsetzung dieses Entwurfs in einen Prototypen beschrieben werden. Dazu wird exemplarisch auf ausgewählte Komponenten der verschiedenen Schichten und besondere Problemstellungen während der Implementierung eingegangen.

5.1 iPhone-Anwendung zur Steuerung und Überwachung von KNX-Systemen

Ziel der Implementierung ist die Entwicklung eines lauffähigen Prototypen, der die im Unterabschnitt 3.3.1 beschriebenen Anwendungsfälle des ersten Szenarios ermöglicht. Dazu wird eine native Anwendung für die iPhoneOS-Plattform nach den im vorangegangenen Kapitel erarbeiteten Entwürfen umgesetzt. Für den Prototypen ist dabei nur ein Teil der in Abschnitt 3.3 vorgestellten Anwendungsfälle obligatorisch. Dazu gehören das Auslesen des Status von Lichtquellen und Temperatursensoren. Zudem sollen Lichtquellen geschaltet und gedimmt werden können. In Hinblick auf die Konfiguration der Anwendung ist das Einlesen definierter Konfigurationsdateien der Software ETS umzusetzen. Die Generierung grafischer Benutzerelemente zur Steuerung der KNX-Geräte aus den Informationen dieser Konfigurationsdateien ist ebenfalls Teil der Implementierung. Der Arbeitstitel dieser Anwendung ist „uniKNX“. In den folgenden Abschnitten wird zunächst die Entwicklungsumgebung, mit der die Anwendung erstellt wurde, vorgestellt.

Weiterhin wird die Struktur des Software-Projektes beschrieben und abschließend auf Problemstellungen, die während der Implementierung auftraten, eingegangen.

5.1.1 Entwicklungsumgebung

Als Entwicklungsumgebung (IDE[1]) wurde Xcode in der Version 3.2.2 von Apple genutzt. Um native Anwendungen für die iPhoneOS-Plattform erstellen zu können wird zudem das iPhone SDK[2] benötigt. Dieses beinhaltet alle Werkzeuge, die zur Erstellung von Anwendungen für die iPhoneOS-Plattform benötigt werden. Dazu gehören unter anderem eine Vielzahl von APIs[3], welche dem Entwickler viele Funktionalitäten über Schnittstellen zur Verfügung stellen. Zudem ist der „iPhone Simulator“ Bestandteil des SDK. Auf diesem können Anwendungen getestet werden, auch ohne das die entsprechenden mobilen Endgeräte wie iPhone oder iPad verfügbar sind. Ein weiterer wichtiger Bestandteil des SDK ist der „Interface Builder“, welcher die grafische Erstellung von Benutzeroberflächen für Anwendungen der iPhoneOS-Plattform ermöglicht. Für die Umsetzung des Prototypen wurde das SDK in der Version 3.1.3 benutzt. Programmiert wird in der Programmiersprache *Objectiv-C*. Dabei handelt es sich um eine Erweiterung der Programmiersprache *C* um Elemente zur Objektorientierung. Aufgrund dieser Abhängigkeit können innerhalb von Anwendungen für die iPhoneOS-Plattform neben *Objective-C* auch die Programmiersprachen *C* und *C++* verwendet werden.

5.1.2 Projektstruktur

Ausgangspunkt jeder iPhoneOS-Anwendung ist ein sogenanntes *AppDelegate*-Objekt. *Delegation* ist eine der wichtigen Softwaredesignmuster dieser Plattform. Dabei sendet ein Objekt, üblicherweise ein Framework-Objekt, das eine bestimmte Aufgabe auf generische Art und Weise löst, periodisch Nachrichten zu einer Art *Stellvertreter*-Objekt, welches die spezifische Weiterverarbeitung der Daten übernimmt. Im Falle des AppDelegate gehören dazu Aufgaben des Lebenszyklusmanagements der Anwendung. Es bietet dem zugrunde liegenden Betriebssystem eine definierte Schnittstelle um Systemmeldungen an eine Anwendung zu übermitteln. So ist das AppDelegate-

[1] Integrated Development Environment
[2] Software Development Kit
[3] Application Programming Interface

Objekt zuständig für anwendungsspezifische Aufgaben, die bei definierten Systemmeldungen ausgeführt werden sollen. Dazu gehören unter anderem die Nachrichten über das Starten oder Beenden der Anwendung oder bei Hauptspeicherknappheit.

Zur Strukturierung der einzelnen anwendungsspezifischen Klassen und zur Abgrenzung der Komponenten wurde das Projekt innerhalb des *Xcode*-Projektes in verschiedene *Gruppen* untergliedert.

- DataStructure
- DataStructure\Helper
- Controller
- Controller\ViewController
- Networking

Diese *Gruppen* untergliedern die Klassen der Anwendung nach funktionalen Gesichtspunkten und sind mit *Packages* vergleichbar. So beinhaltet die *Gruppe* „DataStructure“ alle Klassen zur Erzeugung von Objekten der *Datenschicht* und die *Gruppe* „Networking“ alle Klassen, die zum Empfangen und Senden von Daten über die Drahtlosschnittstelle benötigt werden. Eine Übersicht über die Komponenten der Anwendung ist im Klassendiagramm im Abschnitt A.1.3 zu sehen.

5.1.3 Datenschicht

Wie schon unter Unterabschnitt 4.1.1 beschrieben wurde, werden von den Objekten der Datenschicht zwei Datenstrukturen aufgebaut. Bei der Datenstruktur *Project* handelt es sich zum einen um eine Abbildung des KNX-Systems, das alle notwendigen Informationen zu diesem bereit hält. Zum anderen werden zur Kommunikation mit dem KNX-System Telegramme verwendet, welche in UDP-Pakete verpackt und an den KNXnet/IP-Router gesendet werden. Um diese Telegramme innerhalb der Anwendung abbilden zu können wurde die Struktur *KnxTelegram* definiert.

Aufbau der Datenstruktur *Project*

Die Datenstruktur *Project* bildet ein KNX-System innerhalb der iPhone-Anwendung ab und wird in einem zweistufigen Prozess aufgebaut. Grundlage bildet die Gruppenkonfigurationsdatei der KNX-Software ETS. Diese bietet die Möglichkeit die Gruppen eines KNX-Projektes inklusive aller Bezeichnungen und Gruppenadressen in ein CSV-Austauschformat zu exportieren (siehe Listing 4.1). Die statische Klasse *CSVParser* ist für die Verarbeitung von Dateien im CSV-Format implementiert worden. Zur Verarbeitung von Gruppenkonfigurationsdateien steht die Klassenmethode `+(Project) parseGroupAddressFile: (NSString*) filePath` zur Verfügung. Die Gruppenkonfigurationsdatei wird dabei zeilenweise verarbeitet. Für jeden gültigen Eintrag in dieser Datei wird einem Objekt der Klasse *Project* eine entsprechende Haupt-, Mittel- oder Untergruppe hinzugefügt. Dem aufrufenden Objekt wird dieses *Project*-Objekt zurückgegeben.

Durch den Import der Gerätekonfigurationsdatei wird die Datenstruktur *Project* um wichtige Informationen ergänzt. Die Gerätekonfigurationsdatei (siehe Listing 3.1) wird durch die Software ETS in einem XML-Format bereitgestellt. Das iPhone SDK bietet mit der Klasste *NSXMLParser* ein Mittel zur Verarbeitung von XML-Daten. Es bietet diese Funktionalitäten über Delegation an. Das bedeutet, dass man einem Objekt der Klasse *NSXMLParser* XML-Daten übergibt und zudem ein Objekt einer Stellvertreter-Klasse zur Verfügung stellen muss, die die *Delegate*-Methoden der *NSXMLParser*-Klasse implementiert. Diese Stellvertreterklasse ist in der iPhone-Anwendung die Klasse *DeviceConfigParser*. Das *NSXMLParser*-Objekt stellt die verarbeiteten Daten über diese Methoden dem Stellvertreterobjekt zur Verfügung, welches diese Daten dann weiterverarbeitet. Um sinnvoll mit dem *NSXMLParser* arbeiten zu können empfiehlt es sich, mindestens die *Delegate*-Methoden aus Listing 5.1 zu implementieren.

```
1  - (void)parserDidStartDocument:(NSXMLParser *)parser;
2  - (void)parserDidEndDocument:(NSXMLParser *)parser;
3  - (void)parser:(NSXMLParser *)parser parseErrorOccurred:
4          (NSError *)parseError;
5  - (void)parser:(NSXMLParser *)parser foundCharacters:
6          (NSString *)string;
7  - (void)parser:(NSXMLParser *)parser didStartElement:
8          (NSString *)elementName namespaceURI:(NSString *)
9          namespaceURI qualifiedName:(NSString *)qualifiedName
10         attributes:(NSDictionary *)attributeDict;
11 - (void)parser:(NSXMLParser *)parser didEndElement:
12         (NSString *)elementName namespaceURI:
13         (NSString *)namespaceURI qualifiedName:(NSString *)qName;
```

Listing 5.1: *Delegate*-Methodes der Klasse *NSXMLParser* (Auszug)

Mit den auf diese Art und Weise ermittelten Daten der Gerätekonfigurationsdatei wird das *Project*-Objekt ergänzt. Als problematisch stellte sich dabei das Verhalten des *NSXMLParser* bei Zeichenketten, die Umlaute enthalten, dar. Dieser kann die *Delegation*-Methode `- (void)parser:(NSXMLParser *) parser foundCharacters: (NSString *)string;` in diesem Fall mehrfach aufrufen anstatt die komplette Zeichenkette beim erstmaligen Aufrufen dieser Methode zu übergeben. Gelöst wurde das Problem indem die Zeichenkette zwischengespeichert und bei mehrmaligem Aufrufen der *Delegation*-Methode jeweils um den nächsten Teil der Zeichenkette ergänzt wurde (siehe Listing 5.2).

```
1  - (void)parser:(NSXMLParser *)parser foundCharacters:
2          (NSString *)string {
3          if (![string isEqualToString:@"\n"]) {
4                  if (foundCharacters != nil) {
5                          foundCharacters = [foundCharacters
6                                  stringByAppendingString:string];
7                  } else {
8                          foundCharacters = string;
9                  }
10         }
11 }
```

Listing 5.2: Das Sammeln und Ergänzen von Zeichenketten im *Delegate-Objekt* des *NSXMLParser* (Auszug)

Erstellen und Verarbeiten von KNX-Telegrammen

Die KNX-Telegramme stellen die zweite Datenstruktur, die in der iPhone-Anwendung eine wichtige Rolle spielen, dar. Diese werden von der statischen Klasse *PackageProcessor* erzeugt und verarbeitet. Mit *KnxTegram* wurde eine Struktur definiert um einfacher auf bestimmte Bereiche innerhalb der Daten des Telegramms zugreifen zu können oder diese mit Daten zu füllen. So zeigt z.B. das erste Bit der Struktur die Länge des Headers oder das siebente Bit den Nachrichten-Code an (vgl. Abbildung 4.3). Da einige Informationen auch bit-genau in der Struktur platziert werden bzw. von festen Bitpositionen gelesen werden müssen, wurden in den Methoden der Klasse *PackageProcessor* sogenannte *Bitshift*-Operatoren genutzt um Datenbereiche, die weniger als einen Byte umfassen, zu schreiben bzw. auszulesen. In Listing 5.3 ist ein Beispiel für die Anwendung von Bitshift-Operatoren zu sehen.

```
1  KnxTelegram t;
2  ...
3  /* setting the target adress (2 byte) */
4  // checking target adress type
5  if ([[target objectForKey:@"type"] isEqualToString:@"group"]) {
6    // target address type is group address
7    // -> bit 15 = 0, main 4 bits, mid 3 bits, sub 8 bits
8    t.target_addr |= 0 << 15;
9    t.target_addr |= [[target objectForKey:@"main"] intValue] << 11;
10   t.target_addr |= [[target objectForKey:@"mid"] intValue] << 8;
11   t.target_addr |= [[target objectForKey:@"sub"] intValue] << 0;
12 } else {
13   // target address type is physical address
14   // -> area 4 bits, line 4 bits, member 8 bits
15   t.target_addr |= [[target objectForKey:@"main"] intValue] << 12;
16   t.target_addr |= [[target objectForKey:@"mid"] intValue] << 8;
17   t.target_addr |= [[target objectForKey:@"sub"] intValue] << 0;
18 }
19 // swapping byte order for sending over network
20 t.target_addr = CFSwapInt16BigToHost(p.target_addr);
```

Listing 5.3: Schreiben der Zieladresse in die Struktur *KnxTelegram* mit Bitshift-Operatoren

Die Menge der Nutzdaten, die in einem KNX-Telegramm übertragen werden können, ist variabel und kann bis zu 16 Bytes betragen. Im KNX-Telegramm selber ist kein Hinweis auf die Art und Weise der Interpretation der Daten gegeben. Aus diesem Grund ist das Feld *payload* der Struktur *KnxTelegram* als Array ausgelegt. Als Datentyp der einzelnen Felder wird der C-Datentyp

u_int8_t, also ein nicht vorzeichenbehafteter Ganzzahlwert der in 8 Bits gespeichert wird, definiert. Dadurch soll eine Verfälschung der Daten durch Konvertierungen vermieden werden. Aus dem KNX-Telegramm sollen alle nötigen Informationen wie Quell- und Zieladresse, Kommandotyp und die Nutzdaten in einer Datenstruktur vom Typ *NSDictionary* abgelegt werden. Diese Struktur bietet die Funktionalität Daten unterschiedlichster Datentypen zu Schlüsselworten abzulegen. Dadurch entsteht eine Sammlung von Key/Value-Paaren, auf die im späteren Verlauf sehr komfortabel zugegriffen werden kann. Nachteil dieser und anderer Objective-C-Strukturen ist, dass nur Objekt-Datentypen darin abgelegt werden können, nicht aber C-Strukturen wie das *KnxTelegram* oder C-Arrays. So ist es unmöglich die Nutzdaten, die in einem C-Array vorliegen, direkt in einem Objekt vom Typ *NSDictionary* abzulegen. Das direkte Kopieren der Nutzdaten von einem C-Array in ein Objekt vom Typ *NSData* hatte während der Implementierungsphase immer eine Konvertierung bzw. Veränderung der Daten beim Auslesen aus diesem Objekt zur Folge. Dadurch war der ursprünglich übertragene Wert auch durch Funktionen zur Änderung der „byte order“ nicht mehr nachvollziehbar.

Zwei Alternativen konnten als Lösungsansatz ausgemacht werden. Das C-Array, welches die Nutzdaten beinhaltet, kann zum Beispiel zusätzlich zu der *NSDictionary*-Struktur an die Komponenten, die diese Informationen brauchen, weitergegeben werden. Dazu müsste aber entweder die Klasse *PackageProcessor* diese Komponenten „kennen“ oder als *Delegate* übergeben bekommen. Das würde die Wiederverwendbarkeit der Klasse *PackageProcessor* beeinträchtigen. Implementiert wurde der zweite Lösungsansatz. Hier werden die einzelnen Felder des C-Arrays *payload*, die jeweils vom Typ *u_int8_t* sind, in eine einzelne Variable eines ausreichend großen C-Datentyps geschrieben. Das kann je nach Länge der Nutzdaten der Typ *u_int16_t* bis *u_int64_t* sein. Die Daten dieser Variable lassen sich dann in ein Objekt des Datentyps *NSData* kopieren, ohne, dass Konvertierungen bzw. Veränderungen an den Daten festzustellen sind. Damit können die Nutzdaten der Struktur des Typs *NSDictionary* hinzugefügt werden.

5.1.4 Steuerungsschicht

Wie schon in Unterabschnitt 4.1.2 herausgestellt wurde, übernehmen die Klassen der Steuerungsschicht wichtige Aufgaben zur Steuerung des Lebenszyklus und sind für die Koordination der Abläufe innerhalb der iPhone-Anwendung verantwortlich.

Das *ApplicationDelegate*-Objekt

Das *AppDelegate*-Objekt ist Start- und Endpunkt jeder Anwendung für die iPhoneOS-Plattform. Es stellt eine Art Stellvertreter der gesamten Anwendung für das zugrunde liegende Betriebssystem dar. An dieses Objekt werden systemrelevante Nachrichten gesendet, z.B. das die Anwendung komplett im Speicher verfügbar ist oder aber das Programm beendet werden muss, weil entweder der Anwender das Beenden ausgelöst hat oder aber ein eingehender Anruf die Anwendung zum Beenden zwingt. Die *Delegate*-Methode `- (void)applicationDidFinishLaunching:(UIApplication *)application` wird vom Betriebssystem aufgerufen, sobald die Anwendung in den Speicher geladen und bereit zur Ausführung ist. Es werden hier weitere essentielle Objekte der Steuerungsschicht initialisiert. Im Fall der iPhone-Anwendung sind das das *MainController*-Objekt, welches alle allgemeinen Abläufe koordiniert und somit die Steuerzentrale der Anwendung darstelllt, und die verschiedenen *ViewController*-Objekte zur Steuerung der Benutzeroberflächen. Wird die *Delegate*-Methode `- (void)applicationWillTerminate: (UIApplication *)application` aufgerufen soll die Anwendung geschlossen werden. Innerhalb weniger Sekunden müssen hier letzte Aufgaben wie das Speichern von Konfigurationsdateien abgeschlossen werden, da die Anwendung andererseits systemseitig beendet wird.

Kommunikation

Aufgrund von Vorgaben aus dem KNX-Standard läuft die Kommunikation mit dem KNX-System über UDP-Pakete. In diese Pakete werden die KNX-Telegramme verpackt und an den KNXnet/IP-Router gesendet bzw. von diesem empfangen. Die Klassen, die für die Kommunikation mit dem KNXnet/IP-Router und zum Laden der Konfigurationsdateien benötigt werden, wurden als abgegrenzte Komponente innerhalb der Steuerungsschicht realisiert und befinden sich in der Gruppe *Networking*. Die Komponente erfüllt im wesentlichen zwei Aufgaben. Zum einen sollen die Konfigurationsdateien von einem Server heruntergeladen werden. Diese Aufgabe wurde in der Klasse *Downloader* sehr komfortabel durch die Funktionalitäten der Frameworks des iPhone SDK gelöst. Listing 5.4 zeigt alle nötigen Schritte um Daten über eine URL abzufragen.

```
1 +(NSData*) downloadDataFromUrl: (NSURL*) url returningResponse:
2         (NSURLResponse*) response error: (NSError**) error {
3   NSURLRequest *request = [ [NSURLRequest alloc] initWithURL: url];
```

```
4     NSData *data = [NSURLConnection sendSynchronousRequest:request
5       returningResponse: response error: error];
6           if (error) {
7                   // error handling
8           }
9           return data;
10  }
```

Listing 5.4: Implementierung der Klasse *Downloader*

Zum anderen besteht die zweite Aufgabe darin eine Kommunikation über das UDP-Protokoll zu ermöglichen. Über das von GoogleCode zur Verfügung gestellte Projekt „CocoaAsyncSocket“ [Goo10a] wird ein Framework bereitgestellt, dass die zur Kommunikation benötigten Komponenten des iPhone SDK zusammenfasst und dem Entwickler in vereinfachter Form anbietet. So wird dem Entwickler das Einrichten einer Warteschlange für das Senden und Empfangen von UDP-Paketen abgenommen. Durch die Unterstützung des *Delegation*-Architekturmusters und die Abarbeitung der nötigen Aufgaben in einem separaten *RunLoop*, einer Art Schleife die permanent Ereignisse der verschiedensten Eingabequellen abarbeitet, läuft die Kommunikation ohne die Hauptanwendung zu blockieren. Der UDP-Server, der für das Empfangen von UPD-Nachrichten zuständig ist, wird bei der Initialisierung eines Objektes der Klasse *UDPSockets* gestartet. Diese Klasse ist zudem das Delegate-Objekt für den UDP-Server. Beim Start des UDP-Servers wird zunächst der Multicast-Gruppe *224.0.23.12* beigetreten und der Port *3671* abgehört. Diese Multicast-Gruppe ist weltweit für KNX-Anwendungen reserviert.

Das Empfangen von Daten durch den UDP-Server löst das Aufrufen der entsprechenden *Delegate*-Methode am *UDPSockets*-Objekt aus. Dieses leitet die Daten über die Schnittstelle *UDPHandlerInterface* an das *MainController*-Objekt weiter. Hier wird es durch die *Datenschicht*-Klasse *PackageProcessor* verarbeitet und die Informationen an den *MainController* zurückgegeben. Durch diesen werden die Informationen zum einen an das *GroupDetailViewController*-Objekt weitergeben, welcher nun ermitteln muss, ob die Empfängergruppe gerade vom Anwender gewählt ist und in diesem Fall den empfangenen Status der Gruppe präsentieren. Ist das nicht der Fall so kann die Nachricht verworfen werden. Des weiteren wird die Statusinformation durch den *MainController* in der Datenstruktur *Project* hinterlegt um diese dem Anwender bei Auswahl der Gruppe schnellstmöglich präsentieren zu können.

Wie schon in Abschnitt 4.1.2 beschrieben wurde, wird das Senden von Paketen vom Anwender direkt oder indirekt über die Präsentationsschicht ausgelöst. Tritt ein bestimmtes Ereignis wie z.B. die Auswahl eines Projektes oder

das Drücken des "Reload"-Buttons ein, wird das *MainController*-Objekt vom entsprechenden *ViewController*-Objekt darüber informiert. Diese *ViewController*-Objekte greifen über die Schnittstelle *TelegramSenderInterface* (siehe Listing 5.5) auf das *MainController*-Objekt zu und übergeben diesem alle nötigen Informationen, z.B. Telegrammart und Gruppenadresse des Empfängers, zur Erstellung eines Telegramms. Dieses lässt das *MainController*-Objekt dann von der *PackageProcessor*-Klasse der *Datenschicht* erstellen. Dieses Telegramm wird dann schließlich an das *UDPSockets*-Objekt übergeben, welches dann das Telegramm als Nutzdaten in ein UDP-Paket verpackt und an die entsprechende Multicast-Gruppe sendet.

```
1 @protocol TelegramSenderInterface
2
3 -(void) sendReadTelegramToGroup: (NSDictionary*)groupDict;
4 -(void) sendWriteTelegramToGroup: (NSDictionary*)groupDict
5   withValue: (NSDictionary*)valueDict;
6
7 @end
```

Listing 5.5: Schnittstelle *TelegramSenderInterface* des *MainController* zum Senden von Telegrammen

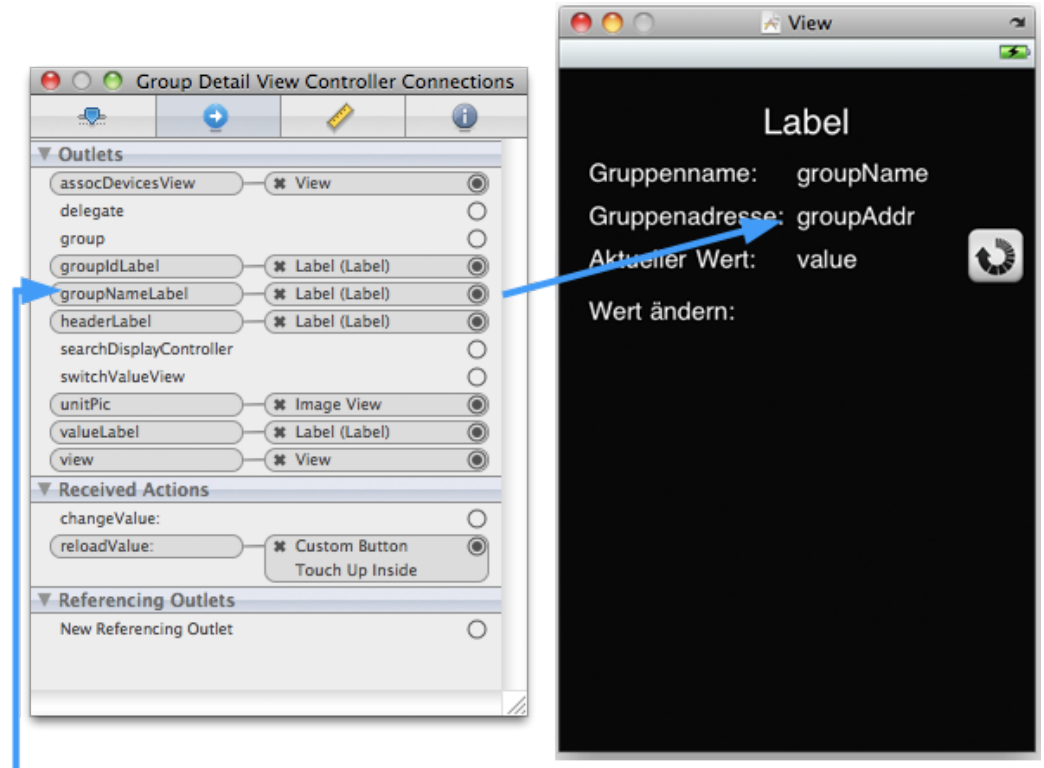
5.1.5 Präsentationsschicht

Die Klassen der *Präsentationsschicht* für Anwendungen der iPhoneOS-Plattform können sowohl im Quellcode als auch über die Anwendung „Interface Builder“ (IB) grafisch definiert werden. „Interface Builder“ ist dabei eine Anwendung zur grafischen Erstellung von Benutzeroberflächen und arbeitet mit der Entwicklungsumgebung Xcode zusammen. So können unter Xcode im Quellcode erstellte und von *UIViewController* abgeleitete Klassen mit speziellen Container-Dateien, den sogenannten NIB-Files[4], verknüpft werden. Diese NIB-Files enthalten dabei eine im IB grafisch erstellte Benutzeroberfläche. IB stellt dazu eine Vielzahl von grafischen Interaktionselementen wie Buttons, Textfelder oder Schalter und Präsentationselemente wie Labels, scrollbare Text-Ansichten (TextView) oder Bildplatzhalter (ImageView) über eine Objekt-Bibliothek (Library) zur Verfügung. Diese können innerhalb des IB platziert und konfiguriert werden. Neben den Attributen wie Koordinaten, Größe, Aussehen und Inhalt können auch Verbindungen zu Klassenvariablen und Methoden der mit dem NIB-File verbundenen *ViewController*-Klasse definiert werden. Dazu müssen Klassenvariablen mit dem Schlüsselwort *IBOutlet* und Methoden mit *IBAction* markiert werden (vgl. *Steuerungsschicht* in Abbildung 5.1).

Auf diesem Wege markierte Klassenvariablen sind dann im IB unter dem Objekt „File's Owner“[5] verfügbar und können mit den Elementen der Benutzeroberfläche verknüpft werden (siehe Abbildung 5.1). So können die Attribute eines im IB platzierten Buttons der Klasse *UIButton* zur Laufzeit durch den mit diesem NIB-File verknüpften *ViewController* über die entsprechen-

[4]Historisch nach „NextSTEP Interface Builder“ benannt.

[5]Hinter dem Objekt „File's Owner“ verbirgt sich ein Verweis auf den entsprechenden von *UIViewController* abgeleiteten *ViewController*.




```
@interface GroupDetailViewController : UIViewController {
    id              delegate;
    id              group;

    UILabel         *headerLabel;
    UILabel         *groupNameLabel;
    UILabel         *groupIdLabel;
    UILabel         *valueLabel;
    UIImageView     *unitPic;
    UIView          *switchValueView;
    UIView          *assocDevicesView;
    NSMutableArray  *switchControls;
}

@property (retain) id                   delegate;
@property (retain) id                   group;
@property (retain) IBOutlet UILabel     *headerLabel;
@property (retain) IBOutlet UILabel     *groupNameLabel;
@property (retain) IBOutlet UILabel     *groupIdLabel;
@property (retain) IBOutlet UILabel     *valueLabel;
@property (retain) IBOutlet UIImageView *unitPic;
@property (retain) IBOutlet UIView      *switchValueView;
@property (retain) IBOutlet UIView      *assocDevicesView;
```

Abbildung 5.1: Verknüpfung von Präsentationsobjekten zwischen *Steuerungsschicht* in *Xcode* und Präsentationsschicht in *IB*

de Klassenvariable angesprochen und verändert werden. Des weiteren kann dieser Button bei festgelegten Interaktionen durch den Anwender definierte Ereignisse auslösen. Zum Beispiel wird beim Berühren und darauffolgenden Loslassen des Buttons, was einer Art „Klick“ des Buttons entspricht, das Ereignis *Touch Up Inside* ausgelöst. Dieses Ereignis kann im IB mit einer mit dem Schlüsselwort *IBAction* markierten Methode verknüpft werden. Diese Methode wird beim Eintreten des Ereignisses am entsprechenden *ViewController* aufgerufen. Im Fall des Prototypen wurden die Benutzerschnittstellen größtenteils über IB grafisch erstellt und konfiguriert.

Erstellung der Interaktionselemente zur Steuerung von KNX-Geräten

Eine der wenigen Ausnahmen, in denen die Elemente der Benutzeroberfläche nicht im IB definiert und konfiguriert werden, sind die Interaktionselemente über die die KNX-Geräte gesteuert werden sollen. Diese Interaktionselemente werden erst zur Laufzeit, beim Aufruf des *GroupDetailView*, also beim Anzeigen der Detailübersicht einer KNX-Gruppe, erzeugt und auf der Benutzeroberfläche platziert. Die Entscheidung, welches grafische Interaktionselement zum Steuern einer KNX-Gruppe (mögliche Ausprägungen sind in Abbildung 4.6 aufgeführt) angezeigt werden soll, hängt vom Datenpunkttyp (siehe Unterabschnitt 3.1.2 Fußnote 1) der jeweiligen KNX-Gruppe ab. So impliziert zum Beispiel der Datenpunkttyp *1001*, dass sich der Status der KNX-Gruppe, der dieser Datenpunkttyp zugeordnet ist, über einen Binärwert darstellen lässt und als „An“ bzw. „Aus“ zu interpretieren ist (vgl. [KNX09, S. 17]). In diesem Fall soll ein einfacher binärer Schalter, über den die KNX-Gruppe steuerbar ist, dem Anwender präsentiert werden. Dazu wird im Quellcode des *GroupDetailViewController* ein Objekt der Klasse *UISwitch* erstellt, konfiguriert und der Benutzeroberfläche hinzugefügt. Listing 5.6 zeigt den Quellcode für dieses Beispiel.

```
1  // get dpt and create UISwitch object when dpt is "1001"
2  if ([[sg dptString] isEqualToString:@"1001"]) {
3    UISwitch *sw = [[UISwitch alloc] initWithFrame:CGRectMake
4      (158, 150, 0, 0)];
5    // set tag for later identification
6    [sw setTag:100100];
7    if ([sg currentValueDict]) {
8      // get current status
9      if ([[[sg currentValueDict] objectForKey:@"result"]
10       isEqualToString:@"Aus"]) {
11       [sw setOn:FALSE];
12     } else if ([[[sg currentValueDict] objectForKey:@"result"]
13       isEqualToString:@"Aus"]) {
14       [sw setOn:TRUE];
15     }
16   }
17   [switchControls addObject:sw];
18   // add switch to view
19   [[self view] addSubview:sw];
20   // set action method for "Touch␣Up␣Inside" event
21   [sw addTarget:self action:@selector(changeValue:)
22     forControlEvents:UIControlEventTouchUpInside];
23 }
```

Listing 5.6: Erstellen eines Interaktionselementes (*UISwitch*) zur Steuerung von KNX-Gruppen

Kapitel 6

Evaluierung und Demonstration des Prototypen

In diesem Kapitel wird im ersten Abschnitt eingeschätzt, inwieweit die in Kapitel 4 identifizierten Anforderungen vom Prototypen der iPhone-Anwendung zur Steuerung und Überwachung intelligenter Gebäudetechnik erfüllt werden. Im darauf folgenden Abschnitt wird das System kurz vorgestellt.

6.1 Evaluierung des Systems

Um eine genaue Bewertung der Ergebnisse dieser Arbeit vornehmen zu können, wird in diesem Abschnitt der Prototyp auf die Erfüllung von grundlegende Kriterien der Softwarearchitektur wie Modularität und Wiederverwendbarkeit der Komponenten hin analysiert. Zudem wird untersucht, ob die in Kapitel 3 gestellten Anforderungen umgesetzt wurden und die Benutzbarkeit der Anwendung gegeben ist.

6.1.1 Modularität und Erweiterbarkeit

Eine modular gestaltete, komponentenbasierte Anwendung erlaubt einen einfachen Austausch einzelner Komponenten. Dadurch kann z.B. schnell auf veränderte Rahmenbedingungen reagiert oder die Anwendung mit wenig Aufwand für ein anderes Einsatzgebiet angepasst werden.

Da der Zugriff auf die Funktionalitäten der Komponenten, wie in Kapitel 4 entworfen, über definierte Schnittstellen abläuft, ist das Austauschen dieser Komponenten leicht umsetzbar. So kann z.B. die Kommunikationskomponente durch das Austauschen der entsprechenden Klassen sehr einfach auf andere Kommunikationsprotokolle umgestellt werden, insofern diese die Methoden der entsprechenden Schnittstellen implementieren. Zudem lässt sich die Kommunikationskomponente in andere Projekte, die eine UDP-Kommunikation benötigen, einbinden und kann über die Schnittstellen betrieben werden.

Da sich der Prototyp mit dem Schalten bzw. Dimmen von Lichtquellen und dem Auslesen von Temperatursensoren auf einen kleinen Teil der im KNX-Standard definierten Steuerungs- und Überwachungsmöglichkeiten beschränkt, ist eine einfache Erweiterbarkeit der Anwendung essentiell. Auch diese Aufgabe lässt sich aufgrund der komponentenbasierten Architektur mit wenig Aufwand lösen. Dazu ist lediglich die Definition des zusätzlich zu unterstützenden Datenpunkttyps notwendig sowie die Zuweisung dieses Datenpunkttyps zu geeigneten Interaktions- bzw. Präsentationselementen im entsprechenden *ViewController*. Sollten keine geeigneten Elemente für die Benutzeroberfläche verfügbar sein, müssten diese zusätzlich implementiert werden.

6.1.2 Funktionalität und Benutzbarkeit

Wie aus der Beschreibung des Einsatzgebietes in Unterabschnitt 3.1.1 hervorgeht, liegt das Hauptaugenmerk auf dem Betreiben des Systems ohne zwischengeschaltete Systeme sowie einer einfachen Konfigurationen und Benutzbarkeit. Die Anwendung soll geschultem Fachpersonal einen schnellen Wechsel zwischen unterschiedlichen Gebäudeautomationssystemen, z.B. bei verschiedenen Kunden, ermöglichen und auch ungeschulten Endanwendern das Steuern und Überwachen von KNX-Technik ermöglichen.

Von den in Abschnitt 3.3 vorgestellten Anwendungsfällen wurden im Rahmen dieser Arbeit ein Teil der Anwendungfälle des ersten Szenarios umgesetzt. So wird durch den Prototypen explizit das Schalten und Dimmen von Lichtquellen und das Auslesen von Temperatursensoren unterstützt. Implizit ist das Steuern und Überwachen aller KNX-Geräte, welche die Datenpunkttypen, die für explizit unterstützte Anwendungsfälle implementiert wurden, möglich. Dazu gehört unter anderem das An- und Ausschalten sowie das schrittweise bzw. prozentuale Ändern des Status von KNX-Geräten. Zudem wurde eine einfache Konfiguration der Anwendung durch die Verarbeitung von Konfigu-

rationsdateien und die Generierung grafischer Benutzerelemente aus diesen Informationen umgesetzt. Das Senden und Empfangen KNX-konformer Telegramme ist ebenfalls möglich (vgl. Abbildung 6.1).

Im Prototypen ist durch die integrierte Verwaltung mehrerer Projekte, welche ein KNX-Gebäudeautomationssystem repräsentieren, sogar die dauerhafte Speicherung der KNX-Konfigurationen verschiedener KNX-Systeme möglich und somit ein schneller Wechsel zwischen diesen Systemen gewährleistet. Durch die sehr einfach gehaltene Konfiguration und die Beschränkung auf wenige, intuitiv bedienbare Benutzeroberflächen ist es auch dem ungeschulten Endanwender möglich ein mobiles Endgerät mit dieser Anwendung in ein entsprechendes Gebäudeautomationssytem einzubinden.

Im Zuge der Anforderungsanalyse wird zudem meist vom mobilen Zugriff auf ein Gebäudeautomationssystem gesprochen und VPN als geeignete Technologie zur sicheren Verbindung mit lokalen Netzwerken über öffentliche Netzwerke wie z.B. dem Internet genannt. Aufgrund von Beschränkungen der mir zum Testen zur Verfügung stehenden Gebäudeautomationsinstallation, konnte die Funktionalität der Anwendung außerhalb des lokalen Netzwerkes nicht getestet werden. Der Aufbau einer Verbindung zum lokalen Netzwerk über Technologien wie z.B. VPN ist nicht Bestandteil der prototypischen Anwendung sondern wird außerhalb dieser durch das iPhone gehandhabt. Auch die Weiterleitung von Multicast-Paketen über eine virtuelle Verbindung ist größtenteils von den Fähigkeiten des Routers im lokalen Netzwerk abhängig.

Da die Konfigurationsdateien, die durch die Software ETS erzeugt werden, nicht alle benötigten Informationen zum KNX-Gebäudeautomationssystem enthalten, kann nicht garantiert werden, dass sich alle Funktionen dieses Systems innerhalb der Anwendung in der gleichen Art und Weise abbilden lassen. So kann z.B. ein Schalter mit verschiedenen Parametern programmiert werden, ohne, dass diese unterschiedlichen Parameter in einer der Konfigurationsdateien ersichtlich wären. Ein solcher Schalter lässt sich zum einen so programmieren, dass ein entsprechendes Telegramm zum Ändern des jeweiligen Gruppenstatus bei jedem Tastendruck gesendet wird. Dies entspricht einer typischen Schaltlogik zum An- und Ausschalten von Geräten. Andererseits kann dieser Schalter auch jeweils beim Drücken und beim Loslassen des Schalters ein entsprechendes Telegramm senden, was z.B. einer Klingelschaltung entsprechen würde. Viele Details der Parametrierung der KNX-Geräte bleiben verborgen, da die Software ETS diese Informationen nicht exportieren kann. Für eine genaue Abbildung des Gebäudeautomationssystems innerhalb der Anwendung wären diese aber essentiell.

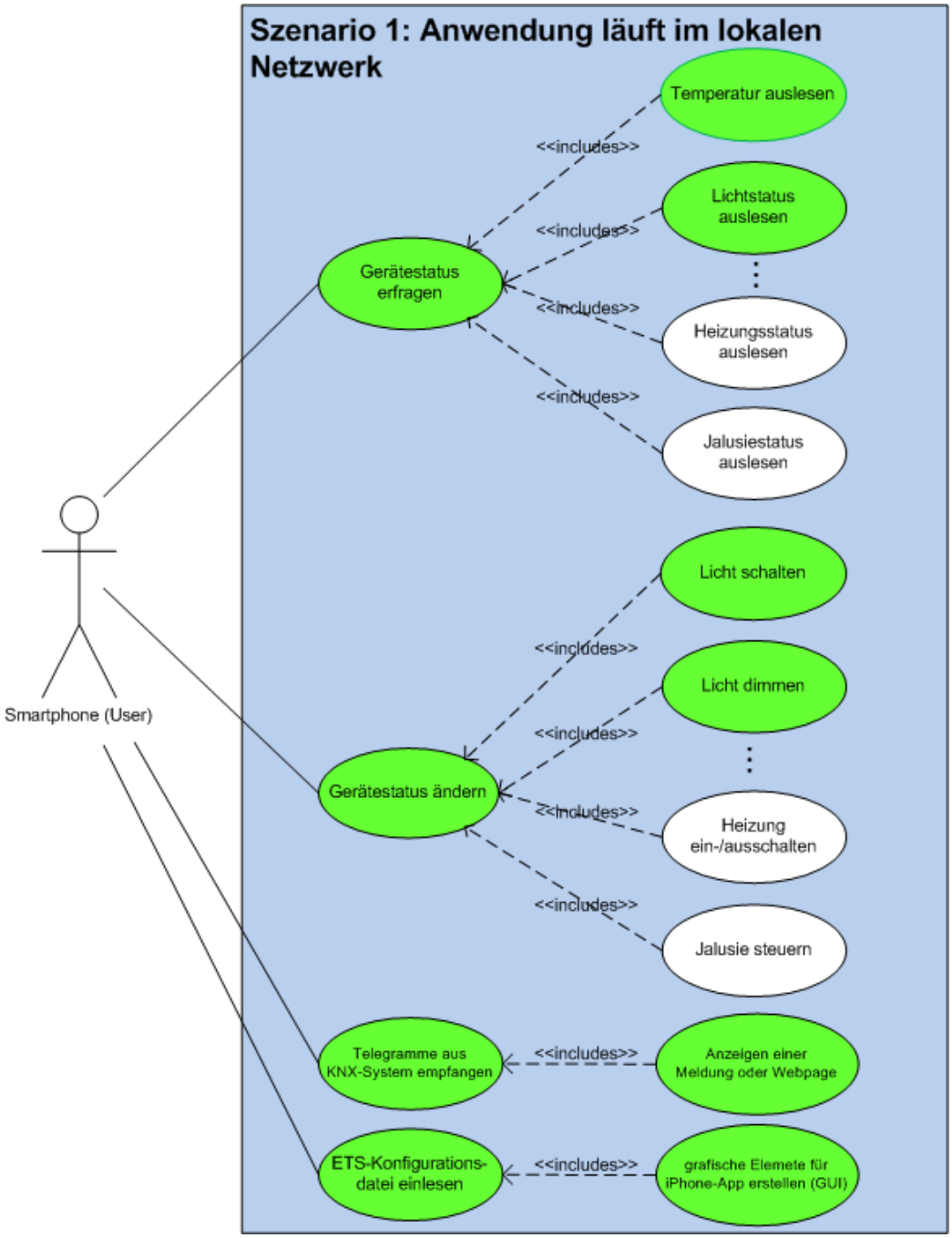


Abbildung 6.1: Im Prototypen umgesetzte Anwendungsfälle

6.2 Demonstration des Prototypen

In diesem Abschnitt soll das entstandene Gesamtsystem vorgestellt werden. Dazu zählt die Demonstration des Verlaufs der typischen Nutzung der mobilen Anwendung vom Start über die Konfiguration und die Auswahl eines Projektes oder einer KNX-Gruppe bis hin zur Steuerung einzelner KNX-Gruppen. Dieser Nutzungsverlauf wird mit Screenshots und Fotoserien zur Steuerung der Test-Gebäudeautomationsinstallation anschaulich präsentiert.

6.2.1 Anwendungsstart und Konfiguration eines Projektes

Die Anwendung wird durch die Auswahl des entsprechenden Icons auf dem *Home Screen* des iPhones gestartet. Nachdem die Anwendung geladen wurde, wird dem Anwender eine Liste mit allen bereits konfigurierten Projekten auf dem *HomeView* der Anwendung angezeigt (siehe Abbildung 6.2). Beim ersten Anwendungsstart sind hier noch keine Projekte auswählbar. Diese müssen erst noch konfiguriert werden.

Um über die Anwendung ein Gebäudeautomationssystem überwachen und steuern zu können ist die Konfiguration eines Projektes notwendig. Dieses repräsentiert das jeweilige Gebäudeautomationssystem innerhalb der Anwendung und stellt alle notwendigen Informationen zu diesem bereit. Ein Projekt wird in der *Konfigurationsansicht* angelegt (siehe 1. in Abbildung 6.3). Dafür werden nur wenige Informationen benötigt. Dazu gehört zum einen der Projektname. Da mit der Anwendung mehrere Konfigurationen von Gebäudeautomationssystemen verwaltet werden können, dient der Projektname als eindeutiges Unterscheidungsmerkmal. Des weiteren sind noch die interne KNX-Adresse, also die sogenannte physikalische Adresse, des KNXnet/IP-Routers und die URL der Gruppenkonfigurationsdatei für den Import der Gruppenadressen nötig (siehe 2. und 3. in Abbildung 6.3). Die physikalische Adresse des KNXnet/IP-Routers wird für die korrekte Erzeugung von standardgemäßen KNX-Telegrammen benötigt, da innerhalb dieser Telegramme eine gültige Absenderadresse angegeben werden sollte. Das KNX-System ist scheinbar tolerant gegenüber nicht verfügbaren Adressen solange diese der standardisierten Syntax einer Gruppen- oder physikalischen Adresse entsprechen. Um eventuell auftretenden Problemen entgegen zu wirken, sollte diese Adresse angegeben werden. Für den Import der Gerätekonfigurationsdatei ist auf einer zweiten *Konfigurationsansicht* die URL zur Gerätekonfigurati-

Abbildung 6.2: Demonstration: Start der Anwendung und *HomeView*

onsdatei anzugeben und diese zu importieren (siehe 4. und 5. in Abbildung 6.3). Der Import der beiden Konfigurationsdateien, die für den Prototypen noch über einen Webserver zur Verfügung gestellt werden müssen, wird in Hinblick auf mögliche andere Importmethoden, wie z.B. dem Text-Import aus der Zwischenablage, getrennt behandelt.

Abbildung 6.3: Konfiguration eines Projektes (Gruppen- bzw. Gerätekonfiguration)

6.2.2 Auswahl von Projekten und KNX-Gruppen

Sobald ein Projekt konfiguriert ist, erscheint es in der Projektliste auf der *Home*-Ansicht. Jedes Projekt kann über eine Schaltfläche auf dem jeweiligen Listeneintrag gelöscht werden (siehe *Home*-Ansicht in Abbildung 6.4). Die Auswahl eines Listeneintrags führt den Anwender zur *Gruppen*-Ansicht, auf welcher die im Projekt vorhandenen Gruppen aufgelistet sind. Auf dieser Ansicht werden die Bezeichner der verfügbaren Gruppen ebenfalls in einer Liste, gruppiert nach Haupt-, Mittel- und Untergruppe, angezeigt (siehe *Gruppen*-Ansicht in Abbildung 6.4). Auch hier wählt der Anwender den gewünschten Listeneintrag aus und bekommt daraufhin eine *Gruppendetail*-Ansicht zur gewählten Gruppe präsentiert. Über diese Detailansicht werden Statusinformationen angezeigt und die Steuerungselemente für diese Gruppe zur Verfügung gestellt. Sollten einzelne Wahlmöglichkeiten nicht verfügbar sein, also wenn z.B. ein Projekt noch nicht vollständig konfiguriert ist, wird der Anwender durch entsprechende Meldungen darauf hingewiesen. Durch diese einfache Navigation und die Verwendung einheitlicher Interaktionselemente für gleichartige Funktionen wird dem Anwender die Bedienung und das Zurechtfinden in der Anwendung erleichtert.

Abbildung 6.4: Auswahl von Projekten (*Home*-Ansicht) bzw. KNX-Gruppen (*Gruppen*-Ansicht)

6.2.3 Steuern und Überwachen von KNX-Geräten

Die Statusinformationen und Interaktionselemente zum Steuern einer Gruppe werden dem Anwender auf der Gruppendetailansicht präsentiert (siehe Abbildung 6.5). Neben diesen werden nur die Gruppenbezeichnung und die Gruppenadresse angezeigt. Auf weiterführende Informationen wird in dieser Ansicht verzichtet um den Anwender nicht mit unnötigen Informationen zu überfluten. Abbildung 6.6 bzw. Abbildung 6.7 zeigen beispielhaft das Schalten bzw. Dimmen einer Lichtquelle. Über die *Reload*-Schaltfläche kann der Status der Gruppe jederzeit aktualisiert werden (siehe *Reload* in Abbildung 6.5).

Beim Design der Interaktionselemente zur Steuerung und Überwachung der KNX-Geräte (siehe *Interaktionselemente* in Abbildung 6.5) wurde auf zwei Kriterien besonderen Wert gelegt. Zum einen wurde darauf geachtet, dass die Interaktionselemente intuitiv zu bedienen sind. Dies hat für den Anwender den Vorteil, dass er keine Einarbeitungszeit für die Nutzung des Systems benötigt, wie dies im Falle des Lesens eines detaillierten Benutzerhandbuches nötig wäre. Zum anderen wurde darauf Wert gelegt, dass die Interaktionselemente einen hohen Wiedererkennungswert haben. Dadurch können Anwender, die bereits mit anderen iPhone-Anwendungen Bedienungserfahrungen gesammelt haben, diese auf die prototypische Anwendung übertragen (siehe Abbildung 6.5).

Abbildung 6.5: Benutzerschnittstellen zur Steuerung und Überwachung von KNX-Geräten (Schalten bzw. Dimmen)

(a) Ausgangssituation: Lichtquelle ist aus

(b) Lichtquelle über iPhone-Anwendung einschalten

Abbildung 6.6: Demonstration: Schalten einer Lichtquelle

(a) Ausgangssituation: Lichtquelle ist aus

(b) Lichtquelle über iPhone-Anwendung auf 50% Lichtstärke einschalten

Abbildung 6.7: Demonstration: Dimmen einer Lichtquelle

(c) Lichtquelle über iPhone-Anwendung auf 100% Lichtstärke einschalten

Abbildung 6.7: Fortsetzung Demonstration: Dimmen einer Lichtquelle

Kapitel 7

Zusammenfassung und Ausblick

Abschließend sollen in diesem Kapitel die Ergebnisse dieser Arbeit zusammengefasst und beurteilt werden. Zudem wird ein Ausblick darauf gegeben, in welchen Bereichen das prototypisch entwickelte System ausbaufähig ist und wohin eine weitere Entwicklung zeigen könnte.

7.1 Zusammenfassung

Das Ziel dieser Arbeit war die Evaluierung von Möglichkeiten zur Integration mobiler Endgeräte in ein Gebäudeautomationssystem sowie die prototypische Entwicklung eines Systems zur Steuerung und Überwachung eines Gebäudeautomationssystems mit mobilen Endgeräten.

Zur Umsetzung dieses Ziels war zunächst eine Analyse des aktuellen Stands der Technik in den relevanten Bereichen nötig. Dazu wurden die Bereiche Smartphones, Drahtlostechnologien und Gebäudeautomationssysteme näher betrachtet und geeignete Technologien zur Umsetzung des prototypischen Systems ermittelt. Mit dem iPhoneOS wurde eine relativ junge Plattform für mobile Endgeräte gewählt, welche sich durch die Integration vieler Technologien, gerade im Bereich der Drahtlostechnologien, aber auch durch strikte Vorgaben, vor allem bei den Benutzerschnittstellen, zu einem vielseitigen und dabei trotzdem intuitiv bedienbaren System entwickelt hat. Für den Datenaustausch zwischen dem mobilen Endgerät und dem Gebäudeautomationssystem können eine Vielzahl von Drahtlostechnologien zum Einsatz kommen. Die WLAN-Technologie bietet sich hier für die Anbindung des mobilen

Engerätes an das Gebäudeautomationssystem über ein lokales Drahtlosnetzwerk an. Für die Umsetzung des prototypischen Systems wurde zudem der KNX-Standard für Gebäudeautomationssysteme, vor allem wegen der guten europaweiten Verbreitung und weltweiten Standardisierung, gewählt.

Nachdem im weiteren Verlauf der Arbeit die Anforderungen an das System identifiziert wurden, konnte ein Systementwurf erstellt werden. Hier wurde insbesondere auf eine flexible Architektur wert gelegt, welche eine einfache Weiterentwicklung und Erweiterung des Prototypen ermöglicht. Zudem wurden Konzepte für die einzelnen Anwendungskomponenten ausgearbeitet. In diesem Teil der Arbeit wurden auch Anforderungen und Konzepte vorgestellt, die über den Umfang, der zur Umsetzung des Ziels nötig ist, hinausgehen. Diese erweiterten Anforderungen und Konzepte sollen die Möglichkeiten für Weiterentwicklungen beispielhaft aufzeigen.

Anschließend wurden die obligatorischen Anforderungen anhand der Entwürfe umgesetzt. Ein Test der Funktionalitäten war auf Grund der starken Abhängigkeit vom Gebäudeautomationssystem fester Bestandteil dieser Phase. Aus diesem Grund wurde in diesem Teil der Arbeit schon auf Problemstellungen, die sich während der Implementierung ergaben, eingegangen und Lösungen dieser erarbeitet.

Abschließend wurde das System evaluiert und demonstriert. Dabei konnte herausgestellt werden, dass das entwickelte System die obligatorischen Anforderungen vollständig umsetzt. Der Prototyp ermöglicht das Steuern und Überwachen von KNX-Geräten über das iPhone. Darüber hinaus benötigt es keine weiteren, zwischengeschalteten Systeme. Die mobile Anwendung bedarf einer einfachen Konfiguration und ist intuitiv bedienbar. Durch die gewählte Architektur lassen sich Erweiterungen zudem mit wenig Aufwand realisieren.

7.2 Ausblick

In der Anforderungsanalyse und dem Systementwurf wurde mit dem dort beschriebenen zweiten Szenario (siehe Unterabschnitt 3.3.2) schon beispielhaft eine Möglichkeit zur Weiterentwicklung des Systems aufgezeigt. Dabei handelt es sich um einen Dienst, welcher definierte KNX-Telegramme an iPhones ausliefert, die nicht mit dem lokalen Netzwerk verbunden sind oder auf denen die mobile Anwendung nicht aktiv ist. Durch eine solche Erweiterung kann die mobile Anwendung über Ereignisse, die aus dem Gebäudeautomationssystem heraus ausgelöst werden, informiert werden, obwohl ein Empfang der

KNX-Telegramme nicht möglich ist. Zur Umsetzung dieses Dienstes könnte zum Beispiel die *Apple Push Notification Service*-Technologie von Apple eingesetzt werden.

Neben diesem Szenario ergeben sich weitere Möglichkeiten den Funktionsumfang des Systems zu erweitern oder auf neue Einsatzgebiete anzupassen. Hier sollte vor allem eine vielseitigere Unterstützung von Datenpunkttypen und geeigneten Interaktionselementen zur Steuerung der KNX-Geräte forciert werden. Im Hinblick auf Benutzerfreundlichkeit und eine intuitive Bedienung könnte auch die Darstellung der Informationen optimiert werden. Es wäre zum Beispiel denkbar den Status der KNX-Gruppen schon auf der *Gruppenansicht* darzustellen um so schnell einen Überblick über das Gebäudeautomationssystem zu bekommen. Praktisch wäre zudem die Möglichkeit definierte Funktionen beim Empfang von bestimmten KNX-Telegrammen auszuführen. So könnte zum Beispiel das Betätigen einer ins Gebäudeautomationssystem eingebundenen Klingel die Anzeige eines Videostreams einer Kamera auslösen.

Insofern eine gemeinsame Abbildung der Topologien unterschiedlicher Standards zur Gebäudeautomation gefunden werden kann, ist auch eine Erweiterung hinsichtlich der Unterstützung anderer Gebäudeautomationsstandards, wie z.B. LON, denkbar. Interessant für das Hotel- bzw. Gastgewerbe könnten erweiterte Zugriffs- und Sicherungsfunktionen für den Zugriff auf ein Gebäudeautomationssystem sein. So könnte durch die Anwendung das Steuern und Überwachen eines begrenzten Umfangs des Gebäudeautomationssystems temporär ermöglicht werden, also z.B. die Geräte eines Hotelzimmers über den Aufenthaltszeitraum eines Gastes durch diesen steuerbar sein. Die Gäste könnten die Anwendung so bequem auf ihrem eigenen mobilen Endgerät ausführen.

Wie schon in Unterabschnitt 2.1.4 deutlich gemacht wurde, wäre die Umsetzung der mobilen Anwendung auch für andere Smartphone-Plattformen wie Android oder Windows Mobile denkbar.

Glossar

DALI

Digital Addressable Lightning Interface (DALI) ist ein Protokoll zur Steuerung von lichttechnischen Betriebsgeräten.

DMX

Die *Open Handset Alliance* ist ein Verbund von 65 Technologie- und Mobilfunkunternehmen mit dem Ziel Innovationen im Bereich Mobilfunk voranzutreiben. Dazu wurde Android als erste freie und offene Mobil-Plattform entwickelt [Ope].

DMX

DMX ist ein Protokoll zur Steuerung von Veranstaltungstechnik.

ETS

Die *Engeneering Tool Software* (ETS) ist eine für Windows erhältliche Anwendung zur Programmierung von KNX-Geräten.

Gateway

Ein *Gateway* ermöglicht die Kommunikation zwischen Netzwerken, die unterschiedliche Protokolle benutzen. Es kann als Übergangspunkt zwischen unterschiedlichen Netzwerken bezeichnet werden.

Gyroskop

Ein *Gyroskop* ist ein Kreiselinstrument über das die Lage und Bewegungsrichtung eines Gegenstandes im Raum ermittelt werden kann

MVC

Das *Model-View-Controller*-Architekturmuster (MVC) dient der Strukturierung von Software-Entwicklungen. Dabei wird die Anwendungen

in drei Komponenten unterteilt: eine Präsentationskomponente, ein Datenmodell und eine Steuerungskomponente. Ziel dieses Musters ist eine verbesserte Wiederverwendbarkeit der einzelnen Komponenten.

SDK

Software Development Kit - Enthält Werkzeuge und Anwendungen zur Softwareherstellung.

Tablet-PC

Tablet-PC gehören zur Geräteklasse der Notebooks und können mit Stift oder Finger bedient werden.

Transceiver

Ein *Transceiver* bezeichnet ein Bauteil, welches Sender (Transmitter) und Empfänger (Receiver) kombiniert.

Literaturverzeichnis

[Bhe08] Bhebhe, Leo: *Multi-access Mobility in Heterogeneous Wireless Networks: Today and Tomorrow.* 2008

[DPSB08] Dahlman, Erik ; Parkvall, Stefan ; Sköld, Johan ; Beming, Per: *3G Evolution - HSPA and LTE for Mobile Broadband 2nd Edition.* Academic Press, 2008

[FM09a] Friedel, Prof. Dr. Ing Rainer ; Mahler, Dipl.-Ing. Hans-Detlev: Europäischer Installationsbus KNX/EIB (Teil 2). (2009)

[FM09b] Friedel, Prof. Dr. Ing Rainer ; Mahler, Dipl.-Ing. Hans-Detlev: Europäischer Installationsbus KNX/EIB (Teil 3). (2009)

[Fra09] Frank, Karlheinz: *KNX/EIB Grundlagen Gebäudesystemtechnik.* Huss-Medien, 2009

[Fre07] Freeman, Roger L.: *Radio System Design for Telecommunications.* Wiley, 2007

[GK09] Gessler, Ralf ; Krause, Thomas: *Wireless-Netzwerke für den Nahbereich: Grundlagen, Verfahren, Vergleich, Entwicklung.* Vieweg+Teubner, 2009

[Gün10] Günther, Andreas: *Konzeption und Entwicklung einer Smartphone-gestützten Schiffskontrolle zur Kollisionsvermeidung bei Schiffsmodellen.* 2010

[Hub02] Huber, Josef F.: Toward the Mobile Internet. (2002)

[IBR+04] Iftode, Liviu ; Borcea, Cristian ; Ravi, Nishkam ; Kang, Porlin ; Zhou, Peng: *Smart Phone: An Embedded System for Universal Interactions.* 2004

[KNX09] KNX Association: *KNX System Specifications - Interworking - Datapoint Types*. 04 2009

[KSH09] Kriesel, Werner ; Sokollik, Frank ; Helm, Peter: *KNX/EIB für die Gebäudesystemtechnik in Wohn- und Zweckbau*. Hüthig Verlag, 2009

[Leh03] Lehner, Franz: *Mobile und drahtlose Informationssysteme: Technologien, Anwendungen, Märkte*. Springer, 2003

[LY09] Lin, Feida ; Yen, Weiguo: *Operating System Battle in the Ecosystem of Smartphone Industry*. 2009

[MHH10] Merz, Hermann ; Hansemann, Thomas ; Hübner, Christof: *Gebäudeautomation*. Carl Hanser Verlag, 2010

[Wal01] Walke, Bernhard: *Mobilfunknetze und ihre Protokolle 1*. Viweg+Teubner, 2001

[Wal03] Walke, Bernhard: *UMTS The Fundamentals*. Wiley & Sons, 2003

[ZN06] Zheng, Pei ; Ni, Lionel M.: Spotlight: The Rise of the Smart Phone. (2006), 03

Internetquellen

[Appa] Apple Inc.: *Apple Developer*. http://developer.apple.com/, . – [Online; Stand 26. März 2010]

[Appb] Apple Inc.: *Mac Dev Center: Introduction to The Objective-C Programming Language*. http://developer.apple.com/Mac/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html, . – [Online; Stand 26. März 2010]

[App09a] Apple Inc.: *Apple iPhoneOS Technology Overview*. http://developer.apple.com/technologies/iphone/, 10 2009. – [Online; Stand 26. März 2010]

[App09b] Apple Inc.: *Cocoa Core Competencies: Model-View-Controller*. http://developer.apple.com/iphone/library/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html, 11 2009. – [Online; Stand 18. Mai 2010]

[App09c] Apple Inc.: *iPhone Dev Center: iPhoneOS Technology Overview: iPhone OS Technologies*. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1, 10 2009. – [Online; Stand 29. März 2010]

[App09d] Apple Inc.: *Mac Dev Center: Cocoa Fundamentals Guide: Cocoa Design Patterns*. http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html, 10 2009. – [Online; Stand 29. März 2010]

[App10a] Apple Inc.: *iPhone Dev Center: iPhone Human Interface Guidelines*. http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html, 02 2010. – [Online; Stand 29. März 2010]

[App10b] Apple Inc.: *iPhone OS Reference Library: iPhone Human Interface Guidelines*. http://developer.apple.com/iphone/library/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, 03 2010. – [Online; Stand 21. Mai 2010]

[Blu10] Bluetooth SIG: *Bluetooth: Get Technical*. http://www.bluetooth.com/German/Technology/Pages/default.aspx, 2010. – [Online; Stand 11. April 2010]

[Bun10] Bundesministerium für Verkehr, Innovation und Technologie Österreich: *Haus der Zukunft*. http://www.hausderzukunft.at/about/index.htm, 2010. – [Online; Stand 18. April 2009]

[dpa09] dpa: *Fenster auf, Heizung aus: Schlaue Häuser im Kommen*. http://www.n-tv.de/ratgeber/immobilienkredite/Schlaue-Haeuser-im-Kommen-article325538.html, 06 2009. – [Online; Stand 11. März 2010]

[eas10] easyMobiz mobile IT solutions GmbH: *ayControl - iPhone KNX Visualisierung*. http://easymobiz.at/de/produkte/aycontrol, 2010. – [Online; Stand 18. April 2010]

[Ecl10] Eclipse Foundation: *Eclipse Newcomers FAQ*. http://www.eclipse.org/home/newcomers.php, 2010. – [Online; Stand 23. März 2010]

[Fra10] Fraunhofer ISST: *inHaus - Innovationszentrum der Fraunhofer-Gesellschaft*. http://www.inhaus-zentrum.de/site_de/, 04 2010. – [Online; Stand 18. April 2010]

[Gar09] Gartner Inc.: *Gartner Says Worldwide Mobile Phone Sales Declined 6 Per Cent and Smartphones Grew 27 Per Cent in Second Quarter of 2009*. http://www.gartner.com/it/page.jsp?id=1126812, 08 2009. – [Online; Stand 12. März 2010]

[Gar10a] Gartner Inc.: *Gartner Says Consumers Will Spend $6.2 Billion in Mobile Application Stores in 2010*. http://www.gartner.com/it/page.jsp?id=1282413, 01 2010. – [Online; Stand 12. März 2010]

[Gar10b] Gartner Inc.: *Gartner Says Touchscreen Mobile Device Sales Will Grow 97 Percent in 2010*. http://www.gartner.com/it/page.jsp?id=1313415, 03 2010. – [Online; Stand 12. März 2010]

[Gar10c] Gartner Inc.: *Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009*. http://www.gartner.com/it/page.jsp?id=1306513, 02 2010. – [Online; Stand 29. März 2010]

[Gir10] Gira GmbH: *Gira Server für das Instabus KNX/EIB System*. http://www.gira.de/produkte/instabus-server.html, 2010. – [Online; Stand 18. April 2010]

[Goo10a] Google Inc.: *Google Code: CocoaAsyncSocket*. http://code.google.com/p/cocoaasyncsocket/, 2010. – [Online; Stand 25. Mai 2010]

[Goo10b] Google Inc.: *What is Android?* http://developer.android.com/guide/basics/what-is-android.html, 03 2010. – [Online; Stand 10. März 2010]

[GSM] GSM World: *Market Data Summary*. http://www.gsmworld.com/newsroom/market-data/market_data_summary.htm, . – [Online; Stand 5. April 2010]

[ibs10] ibs intelligent building services GmbH: *m..Remote for Mac*. http://www.mremote.de/Produkte_mRemote.html, 2010. – [Online; Stand 18. April 2010]

[Ins09] Institute of Electrical and Electronics Engineers, Inc.: *IEEE Ratifies 802.11n, Wireless LAN Specification to Provide Significantly Improved Data Throughput and Range*. http://standards.ieee.org/announcements/ieee802.11n_2009amendment_ratified.html, 09 2009. – [Online; Stand 05. April 2010]

[Net09] NetQB: *WCDMA and HSPA world coverage*. http://www.3g4g.co.uk/Hspa/HSPA_Pres_0803_Ericsson.pdf, 2009. – [Online; Stand 30. April 2010]

[Ope] Open Handset Alliance: *Open Handset Alliance*. http://www.openhandsetalliance.com/, . – [Online; Stand 23. März 2010]

[Sto06] StoLPaN: *NFC Technology*. http://www.stolpan.com/index.php?do=topic&topic=11, 2006. – [Online; Stand 30. April 2010]

[Zig10] ZigBee Alliance: *Bluetooth: Get Technical*. http://zigbee.org/, 2010. – [Online; Stand 11. April 2010]

Bildquellen

[Appa] Apple Inc.: *datepicker_a*. http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/Art/datepicker_a.jpg, . – [Online; Stand 29. März 2010]

[Appb] Apple Inc.: *datepicker_osx*. http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/CocoaFundamentals/Art/datepicker_osx.jpg, . – [Online; Stand 29. März 2010]

[Appc] Apple Inc.: *Layers of iPhoneOS*. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Art/SystemLayers.jpg, . – [Online; Stand 29. März 2010]

[Appd] Apple Inc.: *Model-View-Controller*. http://developer.apple.com/iphone/library/documentation/General/Conceptual/DevPedia-CocoaCore/Art/model_view_controller.jpg, . – [Online; Stand 18. Mai 2010]

[Appe] Apple Inc.: *photos-hardware-05-20090608*. http://images.apple.com/iphone/gallery/images/photos-hardware-05-20090608.jpg, . – [Online; Stand 22. März 2010]

[Bun] Bundesministerium für Sicherheit in der Informationstechnologie: *kommunikationgsm*. https://www.bsi.bund.de/cln_156/ContentBSI/Publikationen/Studien/anonym/kommunikationgsm.html, . – [Online; Stand 04. April 2010]

[eas] easyMobiz mobile IT solutions GmbH: *knx_iphone*. http://easymobiz.at/images/ayControl/knx_iphone.png, . – [Online; Stand 18. April 2010]

[FM] Friedel, Prof. Dr. Ing Rainer ; Mahler, Dipl.-Ing. Hans-Detlev: *Aufbau des KNX/EIB-Datentelegramms*. [FM09b, S. 180],

[Gir] Gira GmbH: *visual_instabus_1329_1206943960*. http://www.gira.de/abbildungen/visual_instabus_1329_1206943960.jpg, . – [Online; Stand 18. April 2010]

[Goo] Google: *Android Architecture*. http://developer.android.com/images/system-architecture.jpg, . – [Online; Stand 26. März 2010]

[HTC] HTC: *Download_03_HTC_HD2*. http://www.htc.com/uploadedImages/WWW/Press_Room/Product_Photo_Gallery/HTC_HD_2/Download_03_HTC_HD2.jpg, . – [Online; Stand 22. März 2010]

[ibs] ibs intelligent building services: *m..Remote*. http://www.mremote.de/Download_files/m..Remote_v1.3.pdf, . – [Online; Stand 18. April 2010]

[MHH] Merz, Hermann ; Hansemann, Thomas ; Hübner, Christof: *Funktionsprinzip der LON-Technik in der Raumautomation*. [MHH10, S. 158],

[Mot] Motorola: *Milestone horizontal clock*. http://mediacenter.motorola.com/imagelibrary/displaymedia.ashx?MediaDetailsID=782&SizeId=3, . – [Online; Stand 22. März 2010]

[Nok] Nokia: *nokia-5320*. http://www.testbericht.de/magazin/wp-content/uploads/2009/08/nokia-5320.jpg, . – [Online; Stand 22. März 2010]

[Sie] Siemens Buildung Technologies: *KNX Netzwerk*. http://www.eci.siemens.com/marketplaces/servlet/EciDownloadDocument?sdc_p=c175fil0m34o1000000000003psuz&sdc_sid=34370165457&sdc_rh=&documentName=%%2Fproducts%2Fdocumentation%2Fde_CH%2F20284.pdf&categoryId=520966&sdc_bcpath=1000000000003.s_0%2C&sdc_m4r=, . – [Online; Stand 16. April 2010]

[WFPa] Werner Kriesel ; Frank Sokollik ; Peter Helm: *Informationsübertragung über das Bussystem KONNEX (KNX).* [KSH09, S. 3],

[WFPb] Werner Kriesel ; Frank Sokollik ; Peter Helm: *Insellösungein mit getrennten Funktionsnetzen.* [KSH09, S. 2],

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Anhang A

A.1 Diagramme

A.1.1 Flussdiagramme

A.1.2 Flussdiagramm Gruppe steuern und überwachen

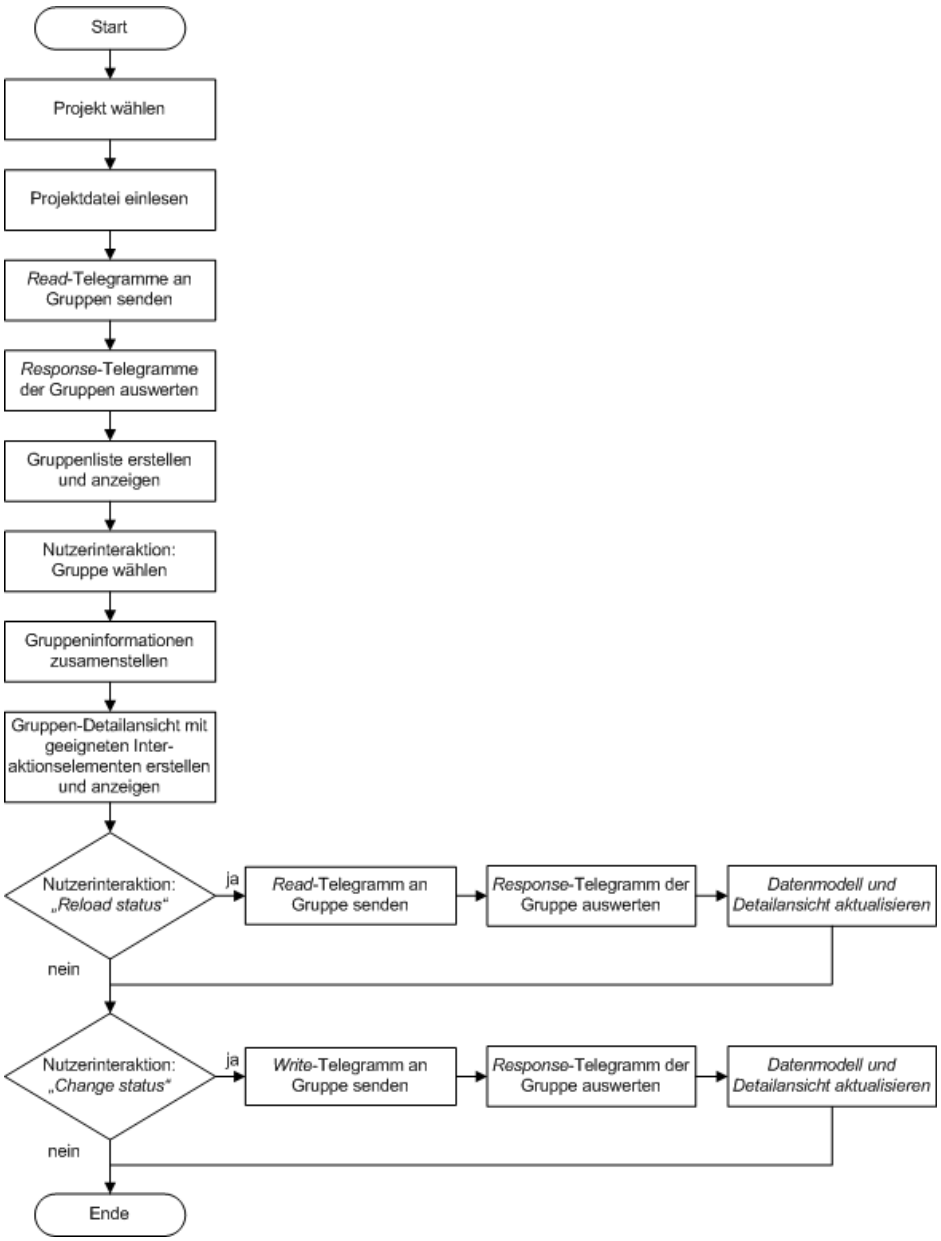


Abbildung A.1: Prozesskette beim Steuern und Überwachen von KNX-Gruppen

Flussdiagramm Projektkonfiguration

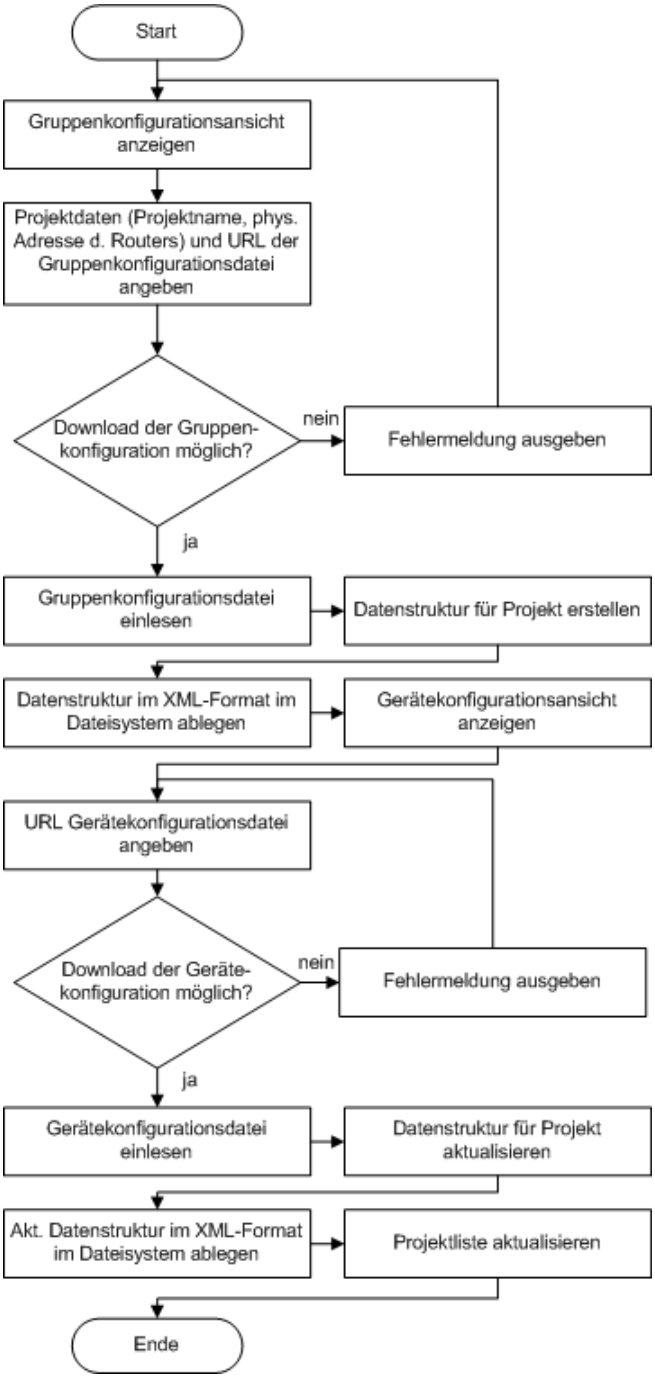


Abbildung A.2: Prozesskette beim Anlegen eines neuen Projektes

Flussdiagramm Telegramm senden

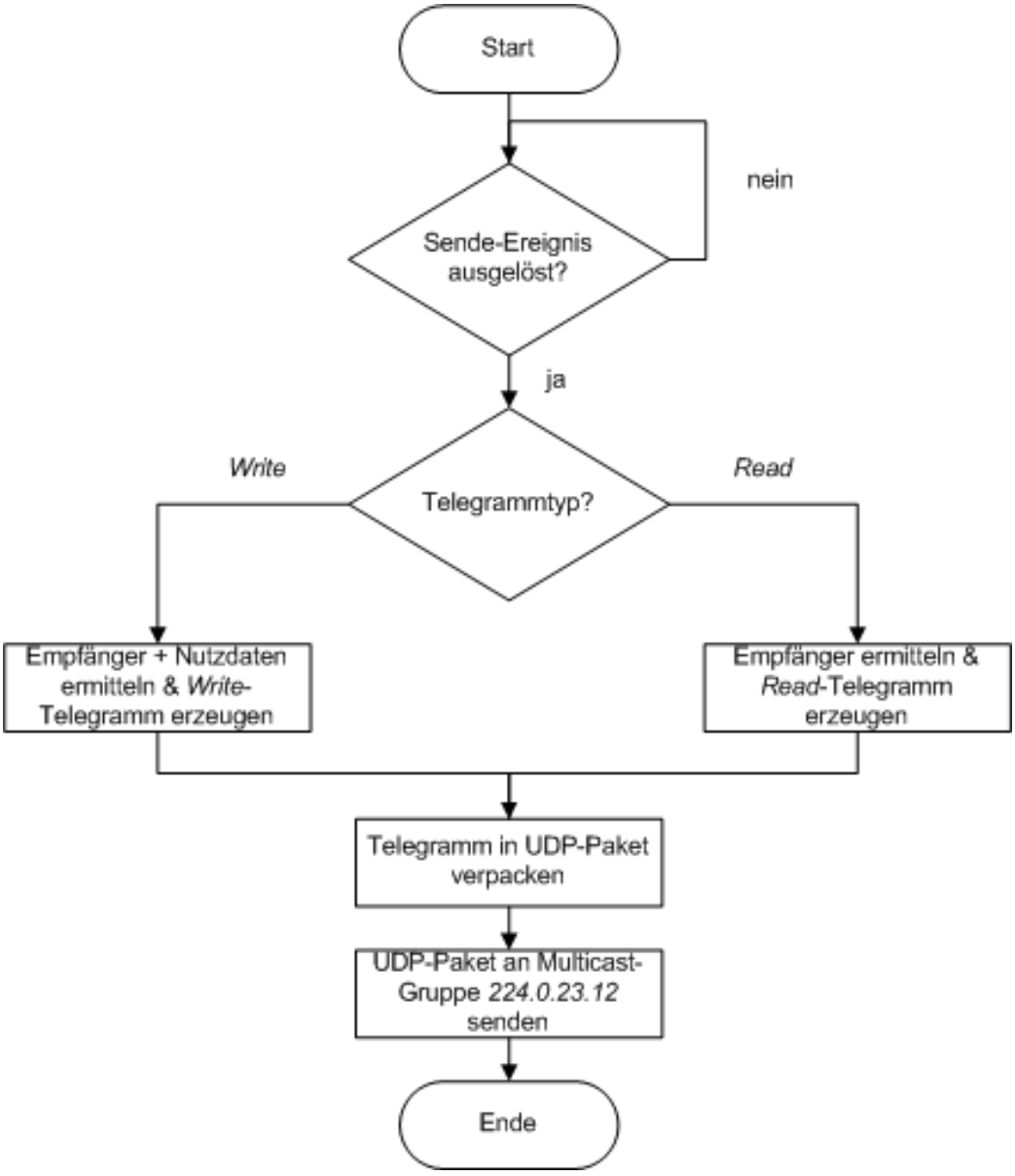


Abbildung A.3: Prozesskette beim Senden eines Telegramms

Flussdiagramm Telegramm empfangen

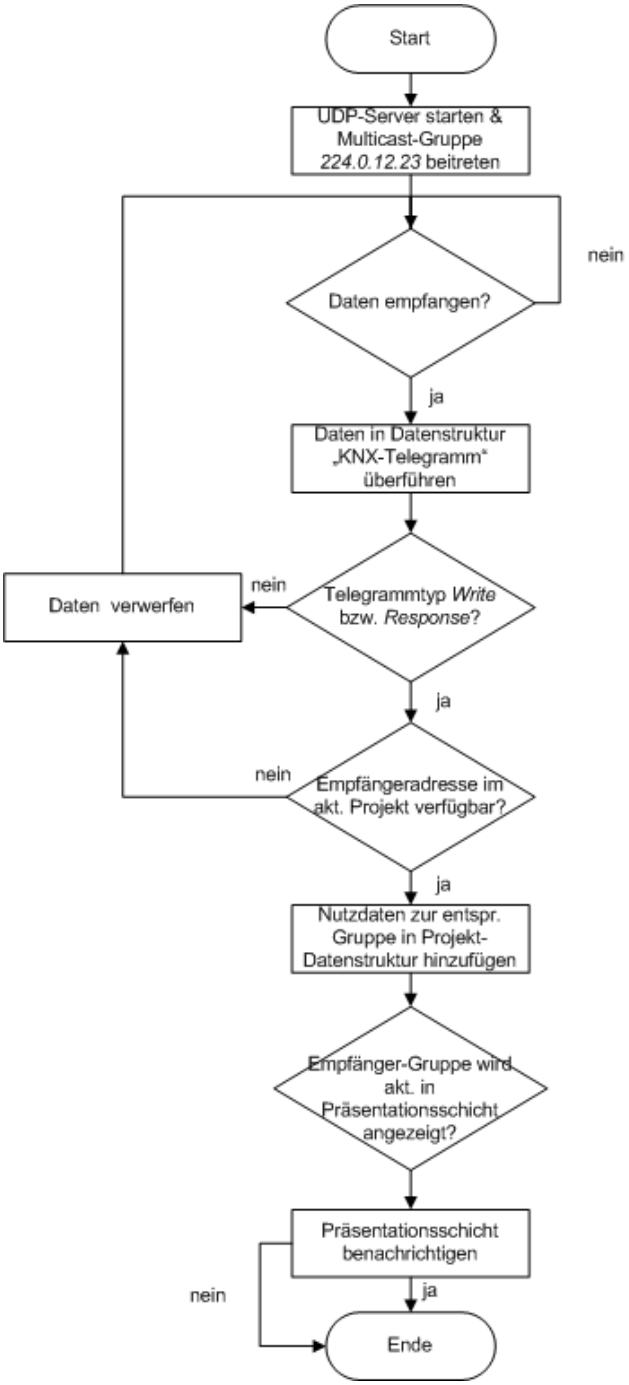


Abbildung A.4: Prozesskette beim eines Telegramms

Flussdiagramm Nachricht über Server-Anwendung senden

Abbildung A.5: Prozesskette beim Ausliefern von Nachrichten über die Server-Anwendung

A.1.3 Klassendiagramme

Komponentenübersicht

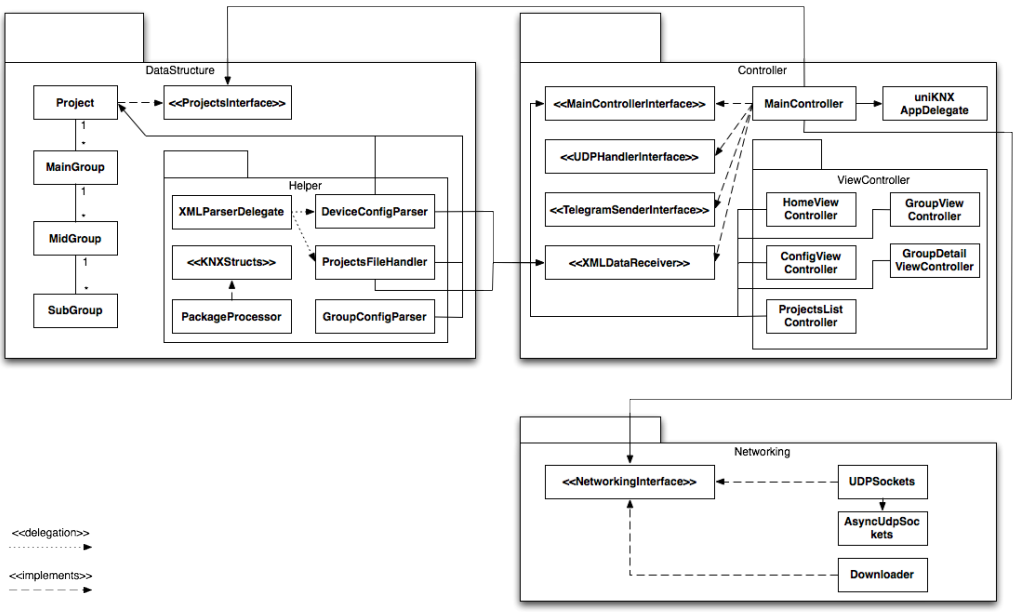


Abbildung A.6: Paketdiagramm *uniKNX* - Komponentenübersicht

A.2 Benutzerschnittstellen

A.2.1 Entwürfe der iPhone-Anwendung

Entwurf Start-Ansicht

Abbildung A.7: Entwurf der Start-Ansicht der iPhone-Anwendung

Entwurf Gruppenansicht

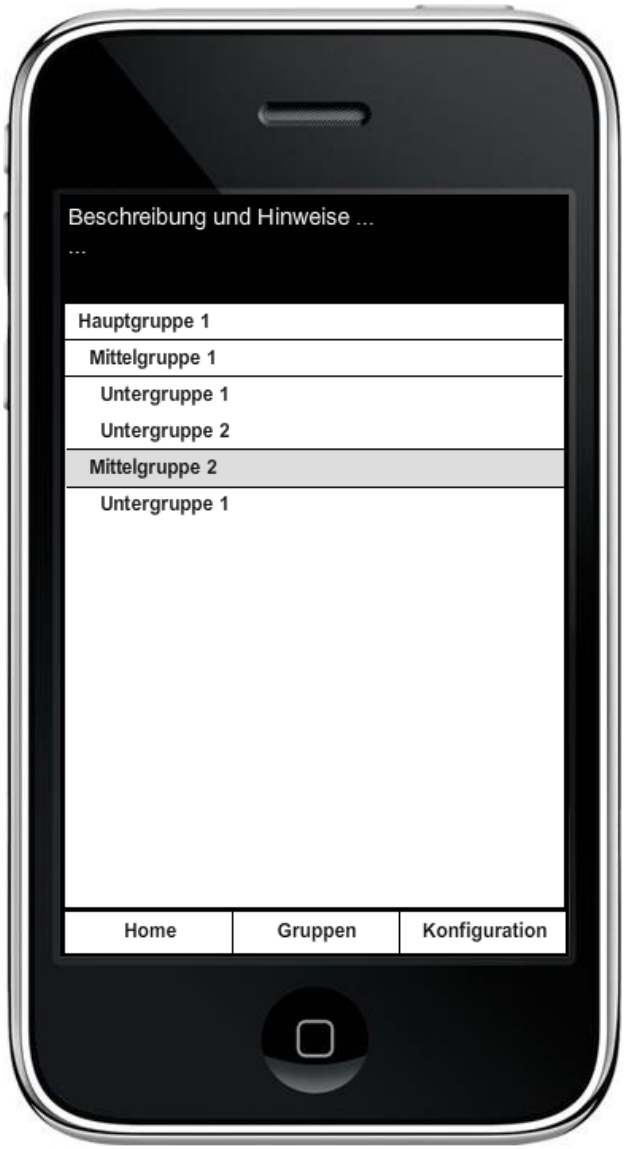


Abbildung A.8: Entwurf der Gruppenlistenansicht der iPhone-Anwendung

Entwurf Gruppendetailansicht

Abbildung A.9: Entwurf der Gruppendetailansicht der iPhone-Anwendung

Entwurf Konfigurationsansicht

Abbildung A.10: Entwurf der Konfigurationsansicht der iPhone-Anwendung

A.3 Fotos

A.3.1 Aufbau der KNX-Gebäudeinstallation

Abbildung A.11: KNX-Gebäudeautomationsinstallation