FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES
Angewandte Informatik

Bachelorarbeit
in der Angewandten Informatik
Nr. AI-2010-BA-30

Überwachung und Steuerung von EIB/KNX-Komponenten mittels Netzwerk-Management-Software

Sebastian Renker

Abgabedatum: 31.07.2010

Prof. Dr.-Ing. Gunar Schorcht
MSc. Oleksandr Artemenko

Kurzfassung

Diese Bachelorarbeit beschäftigt sich mit dem Monitoring und der rudimentären Steuerung von EIB/KNX-Komponenten mittels einer Netzwerk-Management-Software. Nach einer Begriffsabgrenzung und einer Einordnung des Netzwerkmanagements erfolgt eine Evaluierung geeigneter Open Source Software zur Integration von Überwachung und Steuerung. Danach werden die ausgewählte Software und die EIB/KNX-Technologie in ihren Grundlagen erläutert. Im Anschluss werden ein Plugin für das Monitoring der Buskomponenten und ein Programm zur Steuerung von Aktoren entwickelt. In einer Untersuchung werden die beiden Entwicklungen auf verschiedene Kriterien hin überprüft. Das Laborexperiment zeigt, dass EIB/KNX-Geräte in Verbindung mit den entwickelten Softwarekomponenten in eine Netzwerk-Management-Software integriert werden können.

Schlüsselwörter: EIB, KNX, Nagios, Monitoring, Gateway, Überwachung, Steuerung, Gebäudeautomation, ETS3, Open Source, Linux, Perl

Abstract

This bachelor thesis deals with the monitoring and rudimentary control of EIB/KNX-Components through a network management software. After a disambiguation and classification of network management systems, there will be an evaluation of open source software solutions that integrate monitoring and control functions. Subsequently, the most convenient software solutions, as well as the EIB/KNX technology, will be introduced. The development of a plug-in for the monitoring of bus components and a program for control of actuators will then be discussed. In an analysis, the development of these two elements will be reviewed on various criteria. A laboratory experiment shows, that EIB/KNX-Devices associated with the developed software components can integrate into a network management software.

Keywords: EIB, KNX, Nagios, monitoring, gateway, control, building automation system, ETS3, Open Source, Linux, Perl

Inhaltsverzeichnis

Abbildungsverzeichnis **V**

Tabellenverzeichnis **VII**

Listings **IX**

Abkürzungsverzeichnis **XI**

1 Einführung **1**

2 Grundlagen **3**
2.1 Netzwerk-Management . . . 3
2.1.1 Historischer Hintergrund . . . 3
2.1.2 Begriffsabgrenzung Netzwerk-Management . . . 3
2.1.3 Begriffsabgrenzung Monitoring . . . 4
2.2 Produktauswahl Monitoringsoftware . . . 5
2.2.1 Evaluierung . . . 6
2.2.2 Auswahl eines geeigneten Monitoringsystems . . . 8
2.3 Monitoringsystem Nagios . . . 9
2.3.1 Aufbau und Funktionsweise . . . 10
2.3.2 Plugins . . . 12
2.4 EIB/KNX-Technologie . . . 14
2.4.1 EIB/KNX-Geräte . . . 15
2.4.2 Aufbau des Bussystems . . . 16
2.4.3 Kommunikation . . . 18
2.5 PC-KNX-Kommunikation . . . 20

3 Realisierung der Softwarekomponenten **23**
3.1 Konzeption . . . 23
3.1.1 Bus-Zugriff . . . 23
3.1.2 Überwachungsplugins . . . 24
3.1.3 Steuerungsfunktion . . . 27
3.1.4 Testkriterien . . . 30
3.2 Umsetzung . . . 33
3.2.1 Testumgebung EIB/KNX-Szenario . . . 33
3.2.2 Installation und Konfiguration der Testumgebung . . . 35

3.2.3 Implementierung der Softwarekomponenten 37

4 Untersuchung **41**
4.1 Überwachungsplugins 41
4.1.1 Geräteunterstützung 41
4.1.2 Richtigkeit der Telegramme 42
4.1.3 Zuverlässigkeit 43
4.1.4 Entwicklungsrichtlinien 43
4.1.5 Abfragezeit 44
4.1.6 Busauslastung 45
4.2 Steuerungsfunktion 45
4.2.1 Geräteunterstützung 45
4.2.2 Richtigkeit der Telegramme 46
4.2.3 Zuverlässigkeit 47
4.2.4 Versandzeit 47
4.2.5 Gruppenadressarten 48
4.3 Ergebnisse der Untersuchung 48

5 Zusammenfassung **51**
5.1 Ergebnisse 51
5.2 Ausblick 52

Literaturverzeichnis **53**

A Anhang **57**
A.1 Datenpunkt-Typen 57
A.2 Konfiguration und Aufruf der Dienste in Nagios 58
A.3 Alle eingebundenen Plugins 59
A.4 Komponentenliste 60
A.5 Programmablaufplan 61
A.6 Testaufbau im EIB/KNX-Labor 62
A.7 Konfigurationsverzeichnisse Nagios 62
A.8 Checkliste Nagios Plugin Development Guide 63
A.9 Detailansicht des Telegramms Dimmen 64
A.10 Testprogramm testtime.sh 65
A.11 Dateistruktur der Nagioskonfiguration 65

Selbständigkeitserklärung **67**

Abbildungsverzeichnis

2.1 Zeitlicher Verlauf von Zuständen eines überwachten Dienstes . . 11
2.2 Check-Reihenfolge nach Serviceausfall 12
2.3 Modulare Plugin-Architektur von Nagios 13
2.4 Aufbau eines EIB/KNX-Gerätes 15
2.5 Topologie des EIB/KNX-Busses 17
2.6 Aufbau eines Telegramms 18
2.7 Aufbau der physikalischen Adresse 19
2.8 Aufbau der Gruppenadresse 19
2.9 Gruppenadressen nehmen Datenpunkt-Typen an 20
2.10 EIB/KNX-Schnittstellen 21

3.1 Konzept für die Realisierung des Buszugriffs 24
3.2 Diagnosefunktion der ETS3 25
3.3 Schwellenwerte der Parameter für die verschiedenen Zustände . 26
3.4 Konzept für die Realisierung der Steuerungsfunktion 29
3.5 Skizze der Testumgebung im EIB/KNX-Labor 34
3.6 Eingebundene EIB/KNX-Plugins in Nagios 39
3.7 Aufbau der Testumgebung im EIB/KNX-Labor 40
3.8 Integration der Steuerung 40

4.1 Parameter der analogen Sensorschnittstelle 42
4.2 Telegramme bei Abfrage der Busspannung 43
4.3 Busauslastung durch Nagios-Plugins 45
4.4 Telegramme der Steuerungsfunktion 46

5.1 Zusammenfassung von Checks mit check_multi 52

A.1 Konfiguration und Aufruf Nagios-Service 58
A.2 Plugins unter Nagios 59
A.3 Programmablaufplan eines Plugins 61
A.4 Aufbau der Testumgebung im EIB/KNX-Labor 62
A.5 Detailansicht Telegramm Dimmen 64
A.6 Detailansicht Telegramm Dimmen 64

Tabellenverzeichnis

2.1 Leistungsmerkmale der Monitoring-Lösungen 8
2.2 Rückgabewerte von Service-Checks 10
2.3 Rückgabewerte von Host-Checks 11
2.4 Vergleich der physikalischen Anbindung an EIB/KNX 21

3.1 Flags an Speicherstelle 0x010D (Ausführungsfehler) 26
3.2 Flags an Speicherstelle 0x0060 (Ausführungszustand) 26
3.3 Vergleich von serverseitigen Technologien 29
3.4 Testkriterien für das Überwachungsplugin 32
3.5 Testkriterien für Steuerungsfunktion 33

4.1 Zuverlässigkeit der Nagios-Plugins 43
4.2 Ausführungszeiten der Nagios-Plugins 44
4.3 Ausführungszeiten der Steuerung 47
4.4 Zusammenfassung Testergebnisse Überwachungsplugins 48
4.5 Zusammenfassung der Testergebnisse Steuerungsfunktion 49

A.1 Datenpunkt-Typen (Auszug) 57
A.2 Verwendete Datenpunkt-Typen 57
A.3 Komponentenliste 60
A.4 Konfigurationsverzeichnisse Nagios 62
A.5 Checkliste Nagios Plugin Development Guide 63

Listings

3.1 Hinzufügen zur Datei sources.list 35
3.2 Starten des eibd 36
3.3 Anpassungen für das Erlauben externer Kommandos 36
3.4 Abfrage der Busspannung 38
3.5 Abfrage des Ausführungszustands 38
3.6 Abfrage des Ausführungsfehlers 38
3.7 Abfrage des Programmiermodus 38
3.8 Abfrage der Maskenversion 39
3.9 Versand eines Steuerungsbefehls an eine Gruppenadresse 40

A.1 Bash-Script testtime.sh 65
A.2 Dateistruktur der Nagioskonfiguration 65

Abkürzungsverzeichnis

ADC	Analog-Digital-Converter
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Page
AST	Anwendungsschnittstelle
BAU	Bus Access Unit
BCU	Bus Coupling Unit
BSD	Berkeley Software Distribution
BSI	Bundesamt für Sicherheit in der Informationstechnologie
CGI	Common Gateway Interface
CPAN	Comprehensive Perl Archive Network
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DPT	Datenpunkt-Typ
EEPROM	Electrically Erasable Programmable Read-Only Memory
EHS	European Home Systems
EIBA	European Installation Bus Association
eibd	EIB Daemon
EIB	European Installation Bus
EIS	EIB Interworking Standard
ETS3	Engineering Tool Software 3
GNU	GNU is not Unix

GPL	GNU General Public License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IIS	Internet Information Server
ISO	International Organization for Standards
JVM	Java Virtual Machine
JSP	JavaServer Pages
LAN	Local Area Network
MRTG	Multi Router Traffic Grapher
NRPE	Nagios Remote Plugin Executor
NSCA	Nagios Service Check Acceptor
PHP	PHP: Hypertext Preprocessor
RAM	Random Access Memory
RFC	Request for Comments
SDK	Software Developement Kit
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TP	Twisted Pair
URL	Uniform Resource Locator
USB	Universal Serial Bus
VM	Virtuelle Maschine

1 Einführung

Nicht nur die Vernetzung von IT-Systemen schreitet im heutigen Informationszeitalter immer weiter voran. Auch in der Gebäudetechnik ist eine Verbreitung von Netzwerken und allgegenwärtigem (ubiquitous) Computing zu beobachten. Auf Basis des einheitlichen Automatisierungsstandards European Installation Bus (EIB) und dessen Weiterentwicklung KNX (sprich: „Konnex“) werden immer mehr Gebäude zusätzlich zur traditionellen Stromversorgung ausgestattet. Dadurch ist es möglich, die programmierbaren EIB/KNX-Komponenten über einen Bus anzusteuern. Diese Trennung von Steuerung und Stromversorgung erhöht die Flexibilität und bietet ganz neue Anwendungsfälle, die durch herkömmliche Verkabelung nicht oder nur sehr aufwendig möglich wären [Deu10].

Die allgegenwärtigen und programmierbaren Geräte sowie der Trend zur Vernetzung bergen aber auch Gefahrenpotential. Die Abhängigkeit des Menschen von funktionierender Technik und Infrastruktur wird bei einem unerwarteten Ausfall der IT eines Unternehmens deutlich. Neben der Störung von Betriebsabläufen droht dem Unternehmen schnell ein hoher finanzieller Verlust. Beispielsweise gilt ein dreitägiger Ausfall der Firmen-IT bei 25 Prozent aller Unternehmen als existenzbedrohend [Hin07].

Im Bereich des Netzwerkmanagements gibt es für die Überwachung der Funktionsfähigkeit von IT-Infrastuktur bereits Systeme, mit denen ein Monitoring von unterschiedlichen Geräten und Diensten in einem Netzwerk möglich ist. Durch ständige Überwachung (Monitoring) und zeitnahe Benachrichtigung durch ein Eskalationsmanagement können Fehlfunktionen schnell erkannt, lokalisiert und verantwortliches Personal informiert werden, was zur Verringerung oder gar Vermeidung der Ausfallzeit führt. Darüber hinaus können strukturelle Probleme des Netzwerks erkannt und Kapazitätsplanungen aufgrund der langfristigen Monitoring-Daten vereinfacht werden.

In der Gebäudeautomation fehlt bisher dieses Konzept der stetigen Überwachung. Derzeit ist es nur möglich, die Funktionalität einzelner Geräte manuell zu testen, beispielsweise um eine Fehlfunktion zu diagnostizieren. Im Rahmen dieser Arbeit wird untersucht, ob und wie sich die Überwachung von EIB/KNX-Komponenten in eine Netzwerk-Management-Software integrieren lässt. Im Laufe der Arbeit werden die Leitfragen beantwortet, welche Parameter für die Funktionsfähigkeit eines Gerätes verantwortlich sind und welche Geräte auf diese Weise unterstützt werden. Außerdem wird der Frage nachgegangen, ob eine Integration mit quelloffenen und kostenlos zu beziehenden

Programmen realisiert werden kann.

Durch eine erfolgreiche Integration könnten neben dem frühzeitigen Erkennen von defekten EIB/KNX-Komponenten auch die Grundlagen für die Überwachung von Temperatur- und Feuchtigkeitswerten gelegt werden. Als Ziel für eine allumfassende Geräteüberwachung soll neben der Monitoringfunktion auch die rudimentäre Steuerung von Aktoren (Lichter, Jalousie, Heizungsventile, etc.) möglich sein. Die Integration der Steuerung kann dem Bedienpersonal weiteren Komfort bei der Kontrolle bieten, um etwa die Lichter im Rechenzentrum gezielt ein- und auszuschalten und um die Gebäudesicherheit zu erhöhen. Die Überwachung könnte auch in einer Leitwarte zusammengeführt werden. Die zentrale Steuerung von Lichtern, Heizungsventilen oder Jalousie könnte neben Strom auch Heizenergie einsparen. Zentrale Leitfragen in Bezug auf die Steuerungsfunktion sind ebenfalls, welche Geräte gesteuert werden können und welche Einschränkungen es gibt. Die Frage nach der Integration in die Netzwerk-Management-Software ist ebenfalls von wichtiger Bedeutung.

Die Arbeit gliedert sich neben dieser Einführung in vier weitere Kapitel. Zunächst werden im zweiten Kapitel die notwendigen Grundlagen der beiden Themenschwerpunkte Netzwerk-Management und EIB/KNX gelegt. Nach der Einführung ins Netzwerk-Management erfolgt die Auswahl einer geeigneten Netzwerk-Management-Software zur Lösung der Problemstellung. Im dritten Kapitel wird die evaluierte Software zur Lösung des Problems und der Umsetzung des entwickelten Konzepts herangezogen. Eine Untersuchung der Entwicklungen und der Vergleich mit aufgestellten Soll-Kriterien erfolgt im vierten Kapitel. Zum Schluss werden im letzten Kapitel die Ergebnisse des Vergleichs bewertet und ein Ausblick für zukünftige Entwicklungen geliefert.

2 Grundlagen

Für das Verständnis dieser Arbeit werden zuerst einige wichtige Begriffe und Sachverhalte erläutert, die in den nachfolgenden Kapiteln als Grundlage vorausgesetzt werden. Diese umfassen das Themengebiet Netzwerk-Management mit Einführung in das Teilgebiet Netzwerk-Monitoring und einer Softwareevaluierung, auf deren Grundlage in den späteren Kapiteln die Umsetzung und Untersuchung erfolgt. Darüber hinaus erfolgt eine Einführung in die EIB/KNX-Technologie.

2.1 Netzwerk-Management

Der folgende Abschnitt definiert den Begriff des Netzwerkmanagements und ordnet das Netzwerk- und Systemmonitoring in dessen Kontext ein.

2.1.1 Historischer Hintergrund

In den frühen 1980er Jahren waren Netzwerke weniger als Infrastruktur, vielmehr aber als Forschungsobjekte zu betrachten. Zudem waren die Netzwerke oft klein und homogen aufgebaut, was den administrativen Aufwand gering hielt und die Einführung eines Netzwerk-Managements nicht erforderlich machte. Mit der Zeit erkannten die Unternehmen den Kosten- und Produktivitätsvorteil von Netzwerken und begannen ihre Infrastrukturen auszubauen. Die einst getrennten Netze wuchsen zu einer großen globalen Infrastruktur zusammen und wurden damit komplexer, heterogener und größer. Die unterschiedlichen Netzwerk-Technologien verschiedener Hersteller sowie die große Anzahl von Hard- und Softwarekomponenten mussten infolgedessen systematisch verwaltet werden [KR02].

2.1.2 Begriffsabgrenzung Netzwerk-Management

Allgemein lässt sich definieren, dass „Netzwerk-Management einen Service darstellt, der eine Vielzahl verschiedener Werkzeuge, Anwendungen und Geräte umfasst, die den Netzwerk-Manager bei der Überwachung und Verwaltung eines Netzwerks unterstützen“ [Cis01]. Das Bundesamt für Sicherheit in der Informationstechnologie (BSI) definiert im IT-Grundschutzkatalog darüber

hinaus den Begriff des Systemmanagements, welches sich vom Netzwerkmanagement durch die Einteilung in Netzkomponenten (Switches, Router etc.) und Systeme (Anwendungsserver, Speichersysteme etc.)[BSI09] unterscheidet.

Die International Organization for Standards (ISO) entwickelte dazu einen allgemeinen Ansatz zur Definition der funktionalen Aufgaben des Netzwerk-Managements:

Performance-Management: Das Ziel des Performance-Managements ist es, die Leistung (Netzwerk-Durchsatz, Benutzer-Antwortzeiten, Leitungsauslastung etc.) eines Knotens im Netzwerk zu messen und die gewonnenen Daten zur Optimierung des gesamten Leistungsverhaltens des Netzwerks zu nutzen [KR02] [Kau98].

Konfigurationsmanagement: Das Konfigurationsmanagement enthält Hilfsmittel und Funktionen zur Planung, Erweiterung und Änderung der Konfigurationen sämtlicher Geräte in einem Netzwerk. Dies umfasst auch die Pflege und Dokumentation von Konfigurationsinformationen. [Kau98]

Accounting-Management: Mit Hilfe des Accounting-Managements (Abrechnungsmanagement) können Zugriffe der Benutzer auf Netzwerkressourcen definiert, protokolliert und kontrolliert werden. Auf dieser Grundlage lassen sich Kosten und Gebühren berechnen, die den Nutzern in Rechnung gestellt werden können [KR02].

Sicherheitsmanagement: Durch das Sicherheitsmanagement soll eine Manipulation (versehentlich oder mit Absicht) im Netzwerk durch entsprechende Überwachung und Kontrolle verhindert werden [Cis01].

Fehlermanagement: „Das Fehlermanagement umfasst alle Funktionen, die zur Vermeidung, Erkennung und Behebung von Fehlern benutzt werden können“ [KR02] und gehört deswegen zu den am weitesten verbreiteten Elementen des ISO-Netzwerk-Management. Der Fokus im Fehlermanagement liegt vor allem auf dem Erkennen von Symptomen und dem daraus resultierenden Isolieren des Problems [Cis01].

Das Beobachten von Netzen und Systemen lässt sich demnach den Teilaufgaben Performance-Management, Konfigurationsmanagement und Fehlermanagement zuordnen. Damit ist Monitoring ein Teilgebiet des Netzwerk- und Systemmanagements, das im nächsten Kapitel abgegrenzt wird.

2.1.3 Begriffsabgrenzung Monitoring

Allgemein definiert [Hei02] den Begriff des Monitorings als „die Leistungsmessung und die Beobachtung des zeitlichen Ablaufgeschehens (Ablaufbeobachtung) in Computersystemen (Hardware-/Software-Konfiguration)“. Dabei gibt

die Messung der Leistung nicht nur darüber Auskunft, wie schnell ein System arbeitet, sondern auch über die Auslastung der Komponenten der Hardware und Software. Zusammenfassend ist das Ziel des Monitorings:

- Unterstützung beim Entwurf von Computersystemen
- Unterstützung bei der Konfigurationsplanung und der Fehlersuche
- Optimierung von Computersystemen

Als Teil des Netzwerk- und Systemmanagements kann das Monitoring helfen, Gefährdungen zu verhindern bzw. frühzeitig zu erkennen. Die wichtigsten Gefährdungen sind hierbei laut [BSI09]:

- Ausfall oder Störung von Netzkomponenten
- Informationsverlust bei erschöpftem Speichermedium
- Datenverlust bei erschöpftem Speichermedium
- Ausfall von Komponenten eines Netz- und Systemmanagementsystems
- Ausfall von IT-Systemen

Konkret empfiehlt das BSI geeignete „Vorkehrungen und Aktivitäten für die Sicherstellung des effektiven Einsatzes“ mit Hilfe eines Management-Systems [BSI09]. Als Maßnahmen hierfür sind unter anderen die „Überwachung der Netzkomponenten auf ihre korrekte Funktion [und] das Monitoring der Netzperformance“ [BSI09] vorgesehen. Bei Serversystemen ist das Beobachten der „Verfügbarkeit und Auslastung des Systems und der angebotenen Dienste“ empfohlen [BSI09].

Je nachdem, welche Teile einer IT-Infrastruktur beobachtet werden sollen, wird zwischen Netzwerkmonitoring und Systemmonitoring unterschieden, wobei die Grenzen teilweise fließend sind (vgl. Abschnitt 2.1.2). Beispielsweise lässt sich ein Router als Netzwerkkomponente klassifizieren, kann aber gleichzeitig als eigenständiges System angesehen werden [Hag08].

2.2 Produktauswahl Monitoringsoftware

Im Bereich des Netzwerk- und Systemmonitorings gibt es eine breite Auswahl an kommerziellen und unter GNU General Public License (GPL) stehender und damit freier und quelloffener Softwarelösungen. Frei bedeutet in diesem Zusammenhang, dass die Software kostenfrei bezogen (heruntergeladen) werden kann. Professioneller Support oder andere Dienstleistungen sind dagegen kostenpflichtig [BSI10]. Aus Gründen der Unterstützung freier Software und offener Standards wird in dieser Arbeit auf die Evaluierung von kommerzieller Monitoring-Software - für deren Anschaffung im Übrigen die finanziellen Mittel fehlen - verzichtet.

Die Auswahl der Open Source-Software ließ sich bei der Evaluierung auf Nagios, Zabbix, OpenNMS und Zenoss einschränken, da die Projekte Multi Router Traffic Grapher (MRTG), BigSister und Hobbit seit mindestens zwei Jahren nicht mehr weiterentwickelt wurden [www10a] [www10b] [www10c]. Die übrigen freien Monitoringsysteme Catci, Munin und MRTG wurden ebenfalls nicht berücksichtigt, da diese für Langzeitbeobachtungen und grafische Auswertungen von Messwerten und deren Verlauf spezialisiert sind (Performance-Management).

2.2.1 Evaluierung

Alle evaluierten Managementlösungen können die gängigen Betriebssysteme (Linux, Windows, Mac OS X) bzw. deren Dienste monitoren und sind mindestens unter Linux-Distributionen lauffähig. Alle Tools bringen darüber hinaus die Möglichkeit des verteilten bzw. redundanten Monitorings mit, womit eine gewisse Ausfallsicherheit gewährleistet werden kann.

Vor der Entwicklung der Erweiterungen kann bei allen Systemen festgestellt werden, dass ein Plugin zur Überwachung bzw. Steuerung von EIB/KNX-Komponenten nicht existiert. Abfrageintervalle von Tests können flexibel bestimmt werden, was eine Auslastung des Netzes verringert. Außerdem ist bei den vier Softwarepaketen agentenloses Beobachten möglich. Dies bedeutet, dass keine zusätzliche Software auf dem zu prüfenden System installiert werden muss, welches die Daten einsammelt und an die Management-Software versendet. Nagios, Zabbix, OpenNMS und Zenoss berücksichtigen die Netztopologie, um Ausfälle schnell einzugrenzen und unnötige Alarmmeldungen im Eskalationsmanagement zu vermeiden. Alle getesteten Systeme bieten eine Weboberfläche, die den Status der überwachten Systeme darstellt.

Die Dokumentation bzw. die Unterstützung durch die Entwicklergemeinschaft ist bei allen evaluierten Paketen gut bis sehr gut, was sich durch das Vorhandensein von Wikis, Foren, Blogs und ausführlichen Benutzerhandbüchern feststellen lässt. Bei sehr gut bewerteten Programmen ist darüber hinaus deutschsprachige Literatur vorhanden. Außerdem ist bei allen ein professioneller Support verfügbar, der im kommerziellen Umfeld genutzt werden kann.

In den folgenden Abschnitten werden Besonderheiten und Alleinstellungsmerkmale erläutert, die es ermöglichen, die betrachteten Lösungen voneinander zu unterscheiden.

Nagios

Die Software Nagios läuft ausschließlich auf Linux und wird über Textdateien konfiguriert. Dies hat zur Folge, dass der Aufwand bei der Installation und Einrichtung höher ausfällt als bei der Konkurrenz, bei denen dies komplett über eine Weboberfläche ausgeführt werden kann. Die Datenhaltung erfolgt

nicht zwingend über eine Datenbank, sondern kann auch durch Dateien (sogenannte Flatfiles) realisiert werden. Besonders hervorzuheben ist die Vielfalt und Anpassungsfähigkeit, die durch über 1900 Plugins gewährleistet wird. Die sehr große Entwicklergemeinde stellt diese kostenlos zur Verfügung. Außerdem erfährt Nagios unter den getesteten Monitoringlösungen die größte Verbreitung [Nag09].

Zabbix

Als Plattform unterstützt Zabbix, ähnlich wie Nagios, eine hohe Anzahl an Linux-Distributionen, jedoch nicht Windows und Mac OS X. Durch eine ausgefeilte PHP-basierte Weboberfläche fällt der Installations- und Konfigurationsaufwand der unter GPL stehenden Software relativ gering aus. Vorausgesetzt wird eine Datenbank für die Datenhaltung. Die Darstellung von Karten bzw. die Visualisierung der überwachten IT-Infrastruktur können sehr einfach erfolgen. Die Dokumentation des Systems ist durch eine aktive Community und ausführliche Handbücher gewährleistet und kann ggf. um kommerziellen Support erweitert werden [Zab10].

OpenNMS

Die javabasierte Netzwerkmanagementsoftware OpenNMS ist durch ihre plattformunabhängige Programmiersprache neben Linux auch auf Windows und Mac OS X einsetzbar. Allerdings unterliegt das System hierdurch gewissen Einschränkungen, da individuelle Anpassungen aufwendig sind. Der Installations- und Konfigurationsaufwand ist unter den getesteten somit am größten, da die Software ein allumfassendes Netzwerk-Management nach den Kriterien der OSI für sich beansprucht (siehe Kapitel 2.1.2). Auch lässt sich die Darstellung der Infrastruktur in der Weboberfläche nur aufwendig durch Einbindung von Zusatzsoftware bewerkstelligen [Ope10].

Zenoss

Das ebenfalls unter GPL stehende Zenoss bietet als Besonderheit ein modernes auf Asynchronous JavaScript and XML (AJAX) basierendes Webinterface. Es lässt sich vollständig über den Browser konfigurieren und verwalten. Die Kernfunktionen kann Zenoss über die Plugin-Architektur, sogenannte ZenPacks, erweitern und es werden sogar Nagios-Plugins unterstützt. Neben der kostenfreien Version von Zenoss wird eine kommerzielle Version vermarktet, die die Core-Version um einige Funktionalitäten erweitert und kommerziellen Support beinhaltet. Da das Projekt noch relativ jung ist, hat es unter den verglichenen Alternativen die bisher geringste Verbreitung [Zen10].

2.2.2 Auswahl eines geeigneten Monitoringsystems

Die genannten Leistungsmerkmale der evaluierten Monitoringsoftware werden in Tabelle 2.1 zusammenfassend dargestellt.

Leistungsmerkmal	**Nagios**	**Zabbix**	**OpenNMS**	**Zenoss**
Plattformunabhängiges Monitoring	ja	ja	ja	ja
Berücksichtigung der Netztopologie	ja	ja	ja	ja
Kostenloser Bezug	ja	ja	ja	ja
Professioneller Support	ja	ja	ja	ja
Eskalationsmanagement	ja	ja	ja	ja
Verteiltes und redundantes Monitoring	ja	ja	ja	ja
EIB/KNX-Überwachung	nein	nein	nein	nein
Agentenloses Monitoring	ja	ja	ja	ja
Installations-/Konfigurationsaufwand	⊙	⊕	⊖⊖	⊕
Darstellung	⊕	⊕⊕	⊙	⊕
Überwachungsflexibilität	⊕⊕	⊙	⊕	⊕⊕
Dokumentation	⊕⊕	⊕⊕	⊕	⊕
Programmiersprache für Erweiterung	beliebig	beliebig	Java	beliebig
Community Support	⊕⊕	⊕⊕	⊙	⊕
Verbreitung	⊕⊕	⊕	⊙	⊖
Installations-Plattform	Linux	Linux	Linux, Win, Mac	Linux, Mac

Tab. 2.1: Leistungssmerkmale der Monitoring-Lösungen

⊕⊕ entspricht Schulnote 1 (sehr gut)
⊕ entspricht Schulnote 2 (gut)
⊙ entspricht Schulnote 3 (befriedigend)
⊖ entspricht Schulnote 4 (ausreichend)
⊖⊖ entspricht Schulnote 5 (mangelhaft)

Zusammenfassend lässt sich sagen, dass alle getesteten Softwareprodukte grundsätzlich zur Lösung der Problemstellung geeignet sind. Allerdings kann Nagios gegenüber den Konkurrenten in den entscheidenden Auswahlkriterien Dokumentation und den verwendbaren Programmiersprachen für Erweiterungen punkten. Die vielzähligen bereits vorhandenen Überwachungsplugins und Erweiterungen unterstreichen diese Eigenschaften zusätzlich. Eine Plugin-Entwicklung erreicht darüber hinaus durch die hohe Verbreitung der Software

viele potentielle Anwender und Entwickler. Außerdem lässt sich das Nagios-Plugin auch in das Zenoss Monitoring Sytem einbinden und dort verwenden.

Nagios wird dementsprechend im weiteren Verlauf der Arbeit als Basis für die Entwicklung des Monitorings bzw. der Steuerung von EIB/KNX-Komponenten verwendet und im nächsten Abschnitt ausführlich behandelt.

2.3 Monitoringsystem Nagios

Nagios ist ein Monitoringsystem, das eine komplexe IT-Systemlandschaft überwachen kann. Es hält Ressourcen (CPU-Last, Festplattennutzung etc.) und Dienste (SMTP, POP3, HTTP, DNS etc.) auf unterschiedlichen Architekturen (Unix, Linux, Windows etc.) unter ständiger Beobachtung. „Ziel der Software ist es, Administratoren schnell über bedenkliche (Warning) oder kritische Zustände (Critical) zu benachrichtigen. Was als 'bedenklich' oder 'kritisch' gilt, legt der Admin in der Konfiguration fest“ [Bar09]. Bedenkliche Situationen können somit identifiziert werden, bevor es zu einer Beeinträchtigung der Geschäftsprozesse bzw. zum Ausfall von Diensten kommt. Durch eine ständige Beobachtung können darüber hinaus Trends in der Entwicklung der IT-Infrastruktur erkannt und notwendige Erweiterungen frühzeitig geplant werden. Auch das Einhalten von Service Level Agreements (SLAs) kann überprüft, ausgewertet und analysiert werden. Hierzu steht eine umfangreiche Reporting-Funktionalität zur Verfügung.

Nagios liegt aktuell in der Version 3.2.1 [Gal10] vor (Stand: 29. Juli 2010). Die freie Software steht unter GPL und wird von einer OpenSource-Community rund um den Erfinder Ethan Galstad ständig weiterentwickelt und verbessert. Nagios läuft unter zahlreichen Unix-Derivaten (z. B. Linux, BSD, UNIX), kann aber ungeachtet dessen auch andere Plattformen wie zum Beispiel Windows überwachen. Die Konfiguration des Systems erfolgt über Konfigurationsdateien im Textformat (*.cfg), die mit einem einfachen Texteditor erstellt und angepasst werden können. Die spätere Überwachung und rudimentäre Steuerung wird über ein Webinterface direkt im Browser bewerkstelligt. Der Status der zu überwachenden Objekte ist dabei nach dem Ampel-Schema (rot, orange, grün) dargestellt.

Die persistente Datenhaltung der Zustandsdaten erfolgt in Nagios wahlweise über Textdateien (sog. Flatfiles) oder über ein MySQL- oder PostgreSQL-Datenbanksystem. Letztere Möglichkeit soll ab Nagios Version 4 Standard werden und wird in Version 3 notwendig, wenn eine alternative Weboberfläche eingesetzt werden soll [Hel09] [Bar09].

2.3.1 Aufbau und Funktionsweise

Eine der Stärken von Nagios liegt in seinem modularen Aufbau (siehe Abb. 2.3). Der Kern von Nagios selbst enthält keinerlei Tests für die Überwachung, sondern stellt vielmehr ein Überwachungsframework dar, in das alle benötigten Überwachungsprozesse durch Plugins eingepflegt werden. Der Aufruf dieser Plugins wird durch Nagios gesteuert und bildet die Schnittstelle zu externen Programmen [Rie06].

Plugins & Checks

Über eingebundene Plugins werden die eigentlichen Überprüfungen ausgeführt, die sich in Host- und Serviceüberprüfungen (Checks) einteilen lassen:

Service-Check: Service-Checks stellen die Hauptfunktionalitäten von Nagios dar und überprüfen die Dienste eines Hosts. Über diese können qualitative Aussagen zur Funktion von Services getroffen werden. Der Zustand eines Services wird über Grenzwerte festgelegt. Service-Checks werden in regelmäßigen Zeitabständen ausgeführt.

Host-Check: Bei einem Host-Check wird ausschließlich die Erreichbarkeit des Rechners überprüft, ohne eine qualitative Aussage über den Netzstatus zu treffen. Es werden also weder verlorengegangene Pakete noch Antwortzeiten ausgewertet. Host-Checks werden nur bei Bedarf ausgeführt, etwa um zu überprüfen, ob alle Services eines Hosts ausfallen, weil dieser nicht erreichbar ist.

Die Ergebnisse der Tests werden innerhalb des Plugins ausgewertet und über definierte Rückgabewerte an Nagios zurückgeliefert. Für einen Service-Check gibt es vier verschiedene Rückgabewerte:

Zustand	Rückgabe	Bedeutung	Farbe
OK	0	Service ok	Grün
WARNING	1	Service im bedenklichen Bereich	Orange
CRITICAL	2	Service im kritischen Bereich	Rot
UNKNOWN	3	Aufruf des Plugins ist fehlgeschlagen	Grau

Tab. 2.2: Rückgabewerte von Service-Checks [Bar05]

Ein Host-Check kann demgegenüber die in Tabelle 2.3 dargestellten Rückgabewerte besitzen. Der Zustand „Unreachable“ bedeutet im Unterschied zum Zustand „Down“, dass auf dem Weg zum eigentlich Host bereits ein Problem aufgetreten ist. Dies ist der Fall, wenn zum Beispiel ein Router ausgefallen ist und alle über diesen Router zu erreichenden Hosts nicht angesprochen werden können.

Zustand	Rückgabe	Bedeutung	Farbe
UP	0	Host ist erreichbar	Grün
DOWN	1	Host ist ausgefallen	Rot
UNREACHABLE	2	Host ist nicht erreichbar	Grau

Tab. 2.3: Rückgabewerte von Host-Checks [Bar05]

Hard-State und Soft-State

Tritt ein Problem mit einem Service zum ersten Mal auf, so wird der Zustand zunächst als Soft-State erfasst. Danach wird der Test in kürzeren Abständen wiederholt (in Abbildung 2.1 ist der Abstand eine Minute), bis die maximal festgelegte Anzahl (hier: 5) an Überprüfungsversuchen erreicht ist. Hat sich an dem Zustand nichts geändert, gilt der Status des Services als Hard-State und es erfolgen Benachrichtigungen an den verantwortlichen Personenkreis. Die weitere Überprüfung erfolgt wieder in den normalen Check-Intervallen (in der Abbildung alle 5 Minuten).

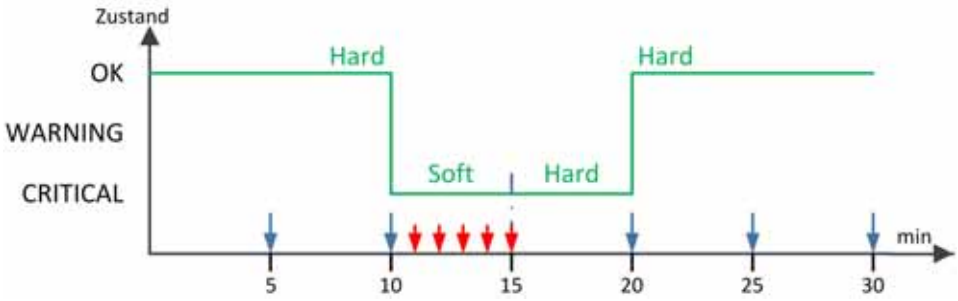

Abb. 2.1: Beispiel für den zeitlichen Verlauf von Zuständen eines überwachten Dienstes [Bar05]

Wechselt der Status aus einem Hard-State wieder zurück in den Zustand OK, der immer als Hard-State gilt, so handelt es sich um ein *Hard Recovery*. Ein Wechsel aus dem Soft-State in den OK-Zustand bezeichnet man als *Soft Recovery*. Im letztgenannten Fall werden keine Benachrichtigungen mit Informationen über den Statuswechsel generiert.

Benachrichtigungen

Um die Administratoren vor einer Flut von Statusmeldungen zu bewahren, enthält Nagios zahlreiche Mechanismen, die die Zahl auf ein nötiges Minimum von Benachrichtigungen reduzieren. Beispielsweise erhält der Administrator bei Ausfall eines Switches nicht für jeden Dienst und jeden Host, der hinter diesem betrieben wird, eine Ausfallmeldung. Er erhält lediglich eine Meldung,

dass der Switch ausgefallen ist (Host-Check). Diese Abhängigkeiten in der Topologie des Netzwerks werden über den Parent(Eltern)-Parameter konfiguriert, der den nächsten Netzknoten auf dem Weg zum Nagios-Server angibt (siehe Abbildung 2.2).

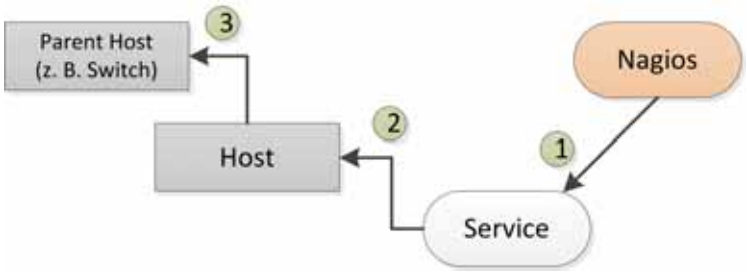

Abb. 2.2: Check-Reihenfolge nach einem Serviceausfall [Bar05]

Verantwortliche Personengruppen können zeitnah über verschiedene Kanäle wie E-Mail, SMS oder Sprachserver in Abhängigkeit der Zustände oder Ereignisse informiert werden. Bei Systemen, die zum Beispiel rund um die Uhr überwacht werden, lassen sich darüber hinaus Zeiträume für die Benachrichtigungen, Verantwortlichkeiten sowie Eskalationsstufen definieren. Sollte beispielsweise ein Problem nicht innerhalb einer festgesetzten Frist durch die verantwortlichen Techniker behoben worden sein, wird automatisch der Abteilungsleiter informiert [Bar09].

Darüber hinaus kann für jede Dienstüberprüfung konfiguriert werden, bei welchem Zustand (critical, warning, unknown) Benachrichtigungen verschickt werden sollen. Bei den Zuständen muss es sich um Hard-States handeln (vgl. Kapitel 2.3.1).

Weboberfläche

Das Webinterface von Nagios setzt einen funktionierenden Webserver mit Common Gateway Interface (CGI)-Unterstützung voraus. Die Oberfläche ermöglicht unterschiedliche Sichtweisen auf den Zustand der überwachten IT-Landschaft, zum Beispiel auf alle Dienste eines Servers. Darüber hinaus ist es möglich, Nagios zu steuern, um etwa Checks neu anzustoßen oder den Nagios-Dienst neu zu starten. Die Authentifizierung für das Web-Frontend erfolgt über eine dateibasierte *Basic-Authentication* und kann zusätzlich die Verbindung mit Hypertext Transfer Protocol Secure (HTTPS) verschlüsseln.

2.3.2 Plugins

Eine große Stärke von Nagios ist die Unterstützung verschiedener Überprüfungsmethoden, die in Abbildung 2.3 dargestellt sind und im vorliegenden Abschnitt kurz vorgestellt werden.

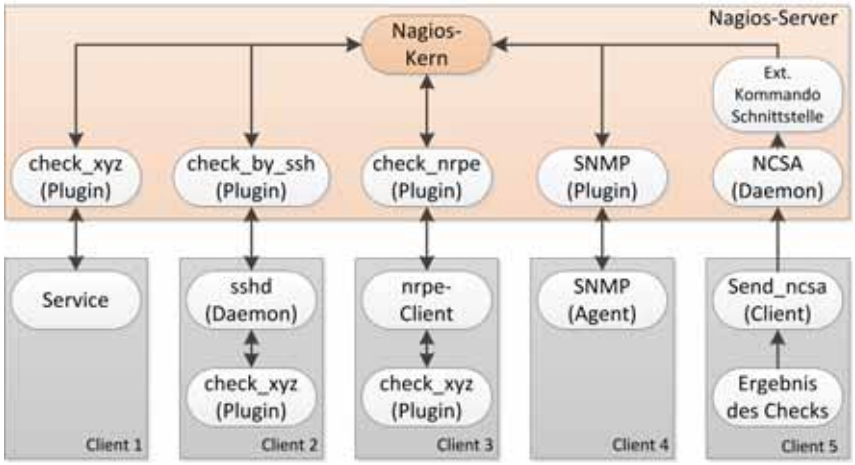

Abb. 2.3: Plugin-Architektur von Nagios [Bar09]

Im ersten Fall (von links beginnend) überprüft Nagios die Funktion des Netzwerkdienstes (Service) durch den direkten Aufruf des entsprechenden Check-Plugins. Der Dienst des Clients reagiert mit einer entsprechenden Antwort (z. B. einer ICMP-Antwort), die Nagios auswerten und darstellen kann.

Bei Client 2 startet Nagios ein Vermittlungsplugin, dass die Verbindung zum entfernten Check-Plugin herstellt. Auf dem Clientsystem ist es dann möglich, das Check-Programm oder auch beliebige Befehle auszuführen. Das vermittelnde Plugin stellt eine Verbindung per Secure Shell (SSH) her.

Beim dritten Client wird das Plugin ebenfalls auf dem Clientrechner ausgeführt, jedoch diesmal über den Nagios-Vermittlungsdienst Nagios Remote Plugin Executor (NRPE) statt per SSH. Wie bei allen vorhergehenden Beispielen stößt Nagios die Abfrage aktiv an.

In der vierten Variante wird die Abfrage über das weit verbreitete Netzwerk-Management-Protokoll Simple Network Management Protocol (SNMP) ausgeführt. Hierzu muss der Client über einen SNMP-Agenten verfügen, der von verschiedenen SNMP-Plugins angesprochen werden kann, um die lokalen Ressourcen des Clients zu überwachen. Auch hier erfolgt die Abfrage durch Nagios.

Bei der fünften Methode handelt es sich, im Gegensatz zu den vorausgegangenen Checks, um einen passiven Check, bei dem Nagios auf eintreffende Informationen wartet. Der Nagios Service Check Acceptor (NSCA)-Daemon nimmt die übermittelten Ergebnisse entgegen und leitet sie über die externe Kommando-Schnittstelle an Nagios weiter. Der Client leitet hierzu mittels des Programms send_nsca die Informationen an den Nagios-Server weiter. Ein Anwendungsszenario ist dabei die verteilte Überwachung mit mehreren Nagios-Servern, bei denen die verteilten Systeme ihre Testergebnisse an einen zentralen Nagios-Server übermitteln.

2.4 EIB/KNX-Technologie

In diesem Abschnitt wird näher auf die Technologie EIB/KNX eingegangen, die mit dem vorher erläuterten Netzwerkmanagement später zusammenarbeiten soll.

Der KNX-Standard bezeichnet einen Feldbus, der auf Grundlage des etablierten EIB weiterentwickelt wurde. Hierzu schlossen sich 1999 die führenden Verbände der Gebäudesystemtechnik BatiBUS, EIB und European Home Systems (EHS) zusammen und gründeten die „Konnex Association“, die die konkurrierenden Standards auf eine gemeinsame Systemplattform stellten. Die European Installation Bus Association (EIBA) sorgt hierbei mit der Festsetzung und Weiterentwicklung von Prüf- und Qualitätsstandards für die Kompatibilität der Produkte verschiedener Hersteller und vergibt bei erfolgreich geprüften Geräten das EIB/KNX-Warenzeichen. Darüber hinaus sorgt die Gesellschaft für die Verbreitung des Standards, Absatzförderung und Koordination der Werbung [Mey04] [Sch04].

Der KNX-Standard ist nach [Ass08] international anerkannt als

- Internationaler Standard (ISO/IEC14543-3)
- Europäischer Standard (CENELEC EN50090, CEN EN 13321-1/2)
- Chinesischer Standard (GB/Z 20965)
- US-Amerikanischer Standard (ANSI/ASHRAE 135)

Da alle EIB-Geräte vollständig kompatibel zur Erweiterung KNX sind, werden die Begriffe in der Literatur häufig synonym gebraucht, weswegen in dieser Arbeit die Schreibweise EIB/KNX verwendet wird.

Die Systemoffenheit umfasst bei allen zertifizierten EIB-Geräten die folgenden Stufen:

Austauschbarkeit: Die Möglichkeit, Geräte innerhalb eines Systems auszutauschen, ohne dass sich Funktionalitäten und Systemverhalten ändern.

Interoperabilität: Die Fähigkeit von kompatiblen Geräten, im Rahmen eines Gesamtsystems Funktionen und Anwendungen zu realisieren und zu gewährleisten.

Kompatibilität: Die Möglichkeit, Geräte miteinander auf physikalischer Ebene, zum Beispiel auf Basis eines Standards, kommunizieren zu lassen.

Allerdings umfasst die Systemoffenheit bzw. der offene Standard nicht den kostenlosen Zugang zu den Spezifikationen. Erst durch eine kostenverursachende Mitgliedschaft in der Konnex Association können die notwendigen technischen Informationen, die zur Weiterentwicklung benötigt werden, eingesehen werden.

2.4.1 EIB/KNX-Geräte

Die Geräte der EIB/KNX-Technologie lassen sich vier verschiedenen Kategorien zuordnen:

Systemgeräte: Spannungsversorgung, Linienkoppler, Bereichskoppler, Linienverstärker, Busankoppler, Schnittstellen (RS232, Ethernet, USB) etc.

Aktoren: Schaltaktor, Jalousieaktor, Binärausgang etc.

Sensoren: Tastsensor, Bewegungssensor, Glasbruchsensor etc.

Sonstige: Logikmodul, Controlpanel etc.

Diese Komponenten gibt es in verschiedenen Ausführungen, da die Einbauorte und -möglichkeiten variieren können [MHH09].

Alle EIB/KNX-Busteilnehmer bestehen grundsätzlich aus drei Bestandteilen:

- Busankoppler Bus Access Unit (BAU)
- Anwendungsmodul (Taster, Relais etc.)
- Anwendungsprogramm (Software)

Die Verbindung zwischen BAU und Anwendungsmodul wird über eine zehnpolige Stiftleiste, der Anwendungsschnittstelle (AST) (siehe Abbildung 2.4) realisiert. Bei einer kompakten Bauweise des Gerätes befinden sich Busankoppler und Anwendungsmodul im gleichen Gehäuse [MHH09].

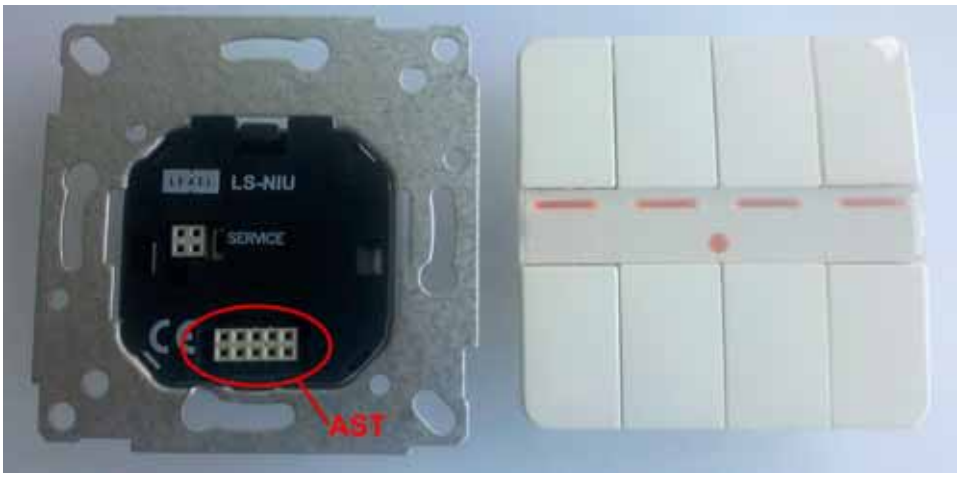

Abb. 2.4: Aufbau eines EIB/KNX-Schalters (4-fach) mit Busankoppler (BCU, links) und Anwendungsmodul (rechts)

2.4.2 Aufbau des Bussystems

Der Aufbau von EIB/KNX lässt sich unterteilen in das Bussystem und die Topologie, die in diesem Teilkapitel erläutert werden.

Bussystem

Beim EIB/KNX-Bus handelt es sich um ein dezentral aufgebautes Bussystem. Jeder Teilnehmer des Busses besitzt einen eigenen programmierbaren Microcontroller. Dies macht ein zentrales Steuergerät überflüssig und schützt gegen einen Totalausfall des Systems. Fällt ein Teilnehmer aus, ist nur dessen Funktion für die Zeit des Ausfalls nicht verfügbar. Das Bussystem wird im Regelfall über zwei Drähte (Twisted Pair (TP)) mit Spannung versorgt, die zwischen 21 und 30 Volt Gleichspannung liegen sollte. Über die gleichen Drähte erfolgt auch die Programmierung und Kommunikation der Busgeräte. Um Kollisionen in der Kommunikation beim gemeinsam genutzten Medium zu vermeiden, wird das Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)-Verfahren verwendet [Sch04]. Darüber hinaus stellt EIB/KNX über Twisted Pair die kostengünstigste Variante dar und ist die am häufigsten anzutreffende Übertragungsart insbesondere bei Neubauten [MHH09].

Alternativ kann die Signalübertragung auch über das Niederspannungsnetz per Powerline-EIB, Funk oder Lichtwellenleiter erfolgen, was aber im weiteren Verlauf der Arbeit unberücksichtigt bleiben soll. Diese Technologien werden meist in privaten Haushalten nachgerüstet und werden nur von wenigen Herstellern angeboten [RKR00] [MHH09].

Die Programmierung der Teilnehmer erfolgt über die Engineering Tool Software 3 (ETS3), die von der Dachorganisation Konnex Association bereitgestellt wird. Über sie kann der Electrically Erasable Programmable Read-Only Memory (EEPROM) jedes Teilnehmers mit einer Applikation programmiert werden. Es gibt mehrere Schnittstellen von PC zu KNX-Bus, die in Kapitel 2.5 näher beschrieben werden. Angesprochen werden alle Teilnehmer durch eine eindeutige physikalische Adresse, mit deren Hilfe die Kommunikation mittels Telegrammen auf dem Bus erfolgt. Wird zum Beispiel ein Taster (Sensor) betätigt, so sendet dieser ein Telegramm mit den entsprechenden Nutzinformationen über den Bus. Alle anderen Teilnehmer „lauschen“ diese Informationen mit. Geräte, die die gleiche Ziel-Gruppenadresse (logische Adresse oder Schaltfunktion, siehe Abschnitt 2.4.3)) besitzen, werten das Telegramm und dessen Daten aus. Die Übertragungsgeschwindigkeit des Busses beträgt 9600 bit/s. Daraus ergibt sich eine Übertragungszeit von 104μs pro Bit [Sch04].

Topologie

Die kleinste Einheit in der EIB/KNX-Topologie ist die Linie, die in der kleinsten Konfiguration eine Spannungsversorgung mit Drossel sowie einen Sensor und einen Aktor (siehe Kapitel 2.4.1) umfasst. In einer Linie können bis zu 64 Geräte angeschlossen werden. Sollte diese Anzahl nicht ausreichen, kann jede Linie zusätzlich mit vier Liniensegmenten durch einen Linienverstärker erweitert werden. Somit ist eine maximale Anzahl von 255 Teilnehmern möglich. Die Länge der Leitung einer Linie darf 1000 Meter nicht übersteigen. Über Linienkoppler können bis zu 15 Linien zu einem Bereich zusammengeschaltet werden. Möglich ist darüber hinaus, diese Bereiche wiederum über Bereichskoppler zu verbinden. Auch hier sind 15 Bereiche möglich. Bei maximaler Ausbaustufe aller Bereiche, Linien und Teilnehmer ergibt sich somit eine Gesamtzahl von $15 \cdot 15 \cdot 255 = 57.375$ Geräten in einem EIB-System (siehe Abbildung 2.5). Ein Koppler trennt die Linien untereinander galvanisch, sodass Kurzschlüsse in einer Leitung nur eine Linie außer Funktion setzen können [Sch04].

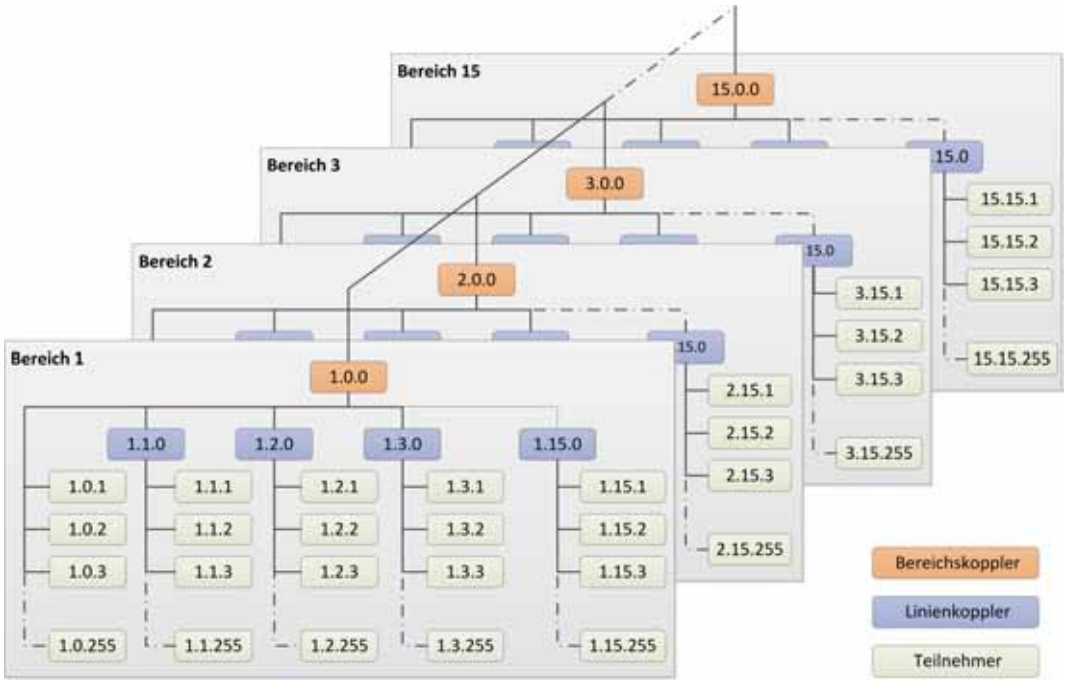

Abb. 2.5: Topologie des EIB/KNX-Busses in voller Ausbaustufe [Sch04]

2.4.3 Kommunikation

Die Kommunikation der EIB/KNX-Komponenten wird zur näheren Erläuterung in den folgenden vier Abschnitten dargestellt.

Telegramm

Der Aufbau eines EIB/KNX-Telegramms besteht im Wesentlichen aus sieben Bestandteilen (siehe Abbildung 2.6).

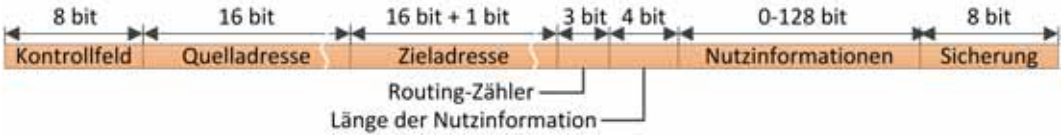

Abb. 2.6: Aufbau eines Telegramms [Mey04]

Im Kontrollfeld wird dem Bus eine Übertragung angezeigt, die Priorität des Telegramms festgelegt und signalisiert, ob es sich um eine Wiederholung des zu sendenden Telegramms handelt. Die Quelladresse beinhaltet die physikalische Adresse des Absenders, wohingegen die Zieladresse meist aus der logischen Adresse besteht. Die beiden Adressierungsarten werden im nächsten Abschnitt näher beschrieben. Der Routingzähler wird beim Durchlaufen einer Koppeleinrichtung dekrementiert und hat den Startwert 6. Sobald der Wert 0 erreicht ist, wird das Telegramm verworfen. Somit lassen sich kreisende Telegramme verhindern, die durch einen Verdrahtungsfehler entstehen können. Nach der Angabe der Länge der Nutzinformationen werden die eigentlich zu übertragenden Daten versendet. Dabei kann die Länge von 0 Bits bis maximal 128 Bits variieren. Die Sicherung sorgt dafür, dass Übertragungsfehler durch eine Paritätsprüfung erkannt werden. Nach einer fest vorgegebenen Wartezeit bestätigen alle angesprochenen Teilnehmer gleichzeitig den Erhalt des Telegramms. Wenn mindestens ein Teilnehmer den Inhalt nicht verarbeiten konnte, meldet er dies durch das Senden eines NAK (Not Acknowledge). Das Telegramm wird daraufhin mit gesetztem Wiederholungsbit neu gesendet [Mey04].

Adressierung

Die Adressierung der Geräte im EIB-Bussystem kann in physikalische Adresse und logische bzw. Gruppenadresse unterteilt werden. Die physikalische Adressierung (siehe Abbildung 2.7) zeigt die Position des Teilnehmers in der Busstruktur durch einen 16 Bit großen Wert an. Ein Gerät mit der Adresse 1.3.7 ist beispielsweise im ersten Bereich der dritten Linie der siebte Teilnehmer. Die Adresse eines Sensors oder Aktors (vgl. 2.4.1) kann durch die Programmierung

über die ETS3-Software geändert werden. Die Software sendet dabei so lange die neue Adresse über den gesamten Bus, bis beim zu programmierenden Gerät die Programmiertaste gedrückt wird. Nach erfolgreicher Programmierung wird dies durch eine aufleuchtende LED am Gerät angezeigt. Zu erkennen ist die physikalische Adresse an der durch Punkte getrennten Schreibweise. Sie wird hauptsächlich für die Programmierung und Fehlersuche verwendet.

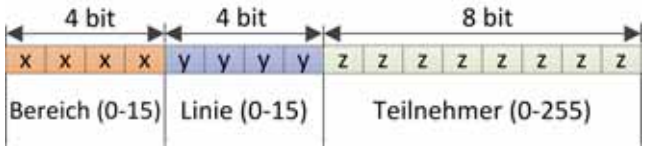

Abb. 2.7: Aufbau der physikalischen Adresse [Sch04]

Die Gruppenadresse sorgt für die logische Verbindung von Sensoren und Aktoren, ähnlich zu einem Schaltdraht. Aktuelle Geräte haben bis zu 254 solcher „Drähte“, können also 254 Gruppenadressen aufnehmen. Wird ein Telegramm über den Bus mit einer Gruppenadresse als Ziel geschickt, reagieren alle Geräte, denen die gleiche logische Adresse zugeordnet wurde. Somit ist es möglich, bei Betätigung eines Schalters mehrere Aktoren reagieren zu lassen, indem man ihnen die gleiche logische Adresse zuordnet. Die Notation der logischen Adresse erfolgt entweder zweistufig (Hauptgruppe/Untergruppe) oder dreistufig (Hauptgruppe/Mittelgruppe/Untergruppe), getrennt durch einen Schrägstrich, die je nach Wunsch des Projektierenden in der ETS3 eingestellt werden kann. Sie stellt lediglich eine Interpretation für eine flexiblere Zuordnung der logischen Verbindung von Sensor und Aktor dar (siehe Abbildung 2.8). Beispielsweise könnte mit einer dreistufigen Adressierung die erste Hauptgruppe für die Gebäudeetage, die Mittelgruppe für die Gewerkfunktion (z. B. Licht) und die Untergruppe für die örtliche Schaltfunktion (z. B. Licht dimmen) stellvertretend stehen [MHH09].

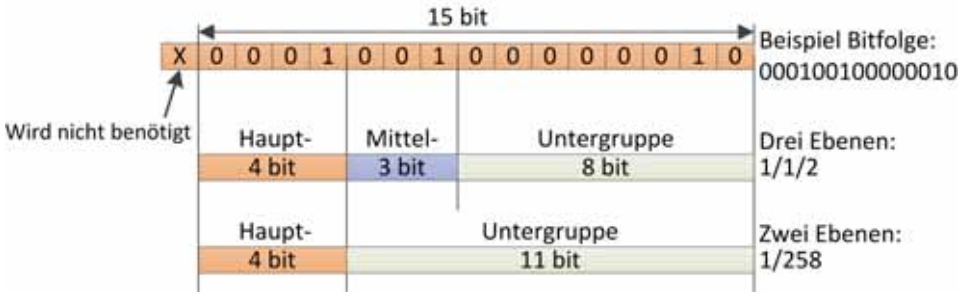

Abb. 2.8: Aufbau der Gruppenadresse [Sch04]

Interoperabilität

Um die Zusammenarbeit der Geräte von verschiedenen Herstellern zu realisieren, wurden Vereinbarungen getroffen, die die Datentypen (Format und Codierung) und die Dimensionen (Einheit und Bereich) festlegen. Diese als Datenpunkt-Typ (DPT) bezeichneten Standards (früher: EIB Interworking Standard (EIS)-Typen) sind durch einen Haupttypen und eine durch einen Punkt separierten Subtypen eindeutig benannt (z. B. 7.001). Die 16-bit Hauptnummer bezeichnet dabei den Datentypen, die 16-bit Subnummer die Dimension. Die Hauptnummern sind im Anhang A.1 zu finden, die vollständige Dokumentation ist in den KNX-Spezifikationen hinterlegt [Ass10].

Kommunikationsobjekte

Den Kommunikationsobjekten, die als logische Schnittstelle der Geräte fungieren, können mittels ETS3 Gruppenadressen zugeordnet werden und enthalten Informationen, mit welchen Datenpunkt-Typen sie verbunden bzw. angesprochen werden können. Beim Versuch, ein Objekt mit der Objektfunktion „Schalten“ (DPT 1) mit einem anderen DPT zu verbinden, würde die ETS3-Software einen Fehler melden [Mey04].

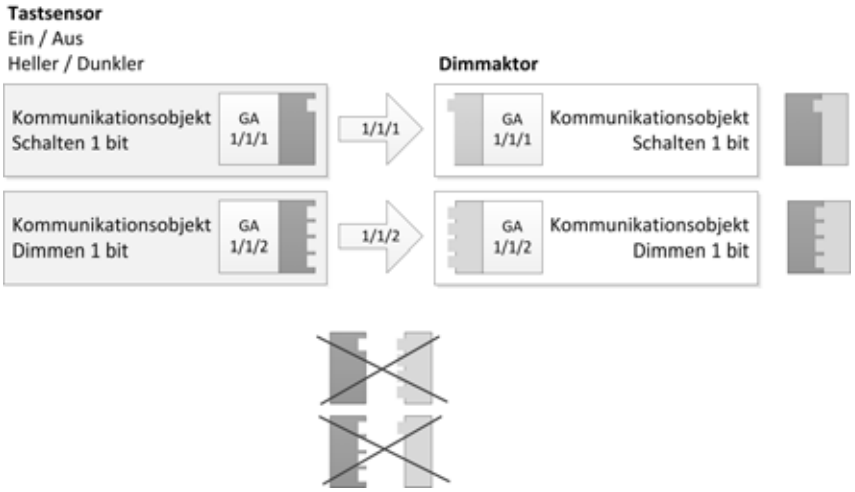

Abb. 2.9: Gruppenadressen nehmen Datenpunkt-Typen an [Mey04]

2.5 PC-KNX-Kommunikation

Für die Kommunikation des EIB/KNX-Busses mit dem PC bzw. der ETS3-Software stehen drei Möglichkeiten zur Verfügung:

Die älteste Methode ist der Anschluss mittels **RS-232**-Schnittstelle, die heute als EIA-232 bezeichnet wird. Sie stellt eine Punkt-zu-Punkt-Verbindung dar

und ist somit in ihrer Flexibilität begrenzt. Die Maximallänge des Anschlusskabels hängt direkt von der verwendeten Baud-Rate ab. Ein weiterer Nachteil ist, dass diese Schnittstelle an neuartigen PCs oder Notebooks nicht mehr vorhanden ist und per Adapter nachgerüstet werden muss.

Die **USB**-Schnittstelle gilt als Nachfolger von RS-232 bei der Verbindung mit dem EIB/KNX-Bussystem. Mit ihrer Hilfe werden ebenfalls Punkt-zu-Punkt-Verbindungen hergestellt. Allerdings ist diese durch eine geringe Maximallänge von 3-5 Metern begrenzt, die nur mit Hilfe von USB-Hubs oder Verstärkern auf circa 30 Meter erhöht werden kann. Ein weiterer Vorteil ist die hohe Verbreitung der Schnittstelle bei alter und neuer Hardware.

Die flexibelste Lösung, die außerdem kompatibel mit der vorhandenen Local Area Network (LAN)-Verkabelung von Gebäuden ist, ist die Anbindung per **Ethernet** durch einen IP-Router. Neben den sehr hohen Übertragungsraten ist die Kopplung mehrerer verteilter Standorte über diese Gateways möglich. Die Programmierung der Geräte kann flexibel von überall aus dem LAN erfolgen.

Abb. 2.10: RS-232-, USB- und Ethernet-Schnittstellen Siemens, Quelle: http://www.siemens.de

Technologie	Verbreitung	Leitungslänge	Flexibilität
EIA-232	gering	152 Meter (bei 9.600 Baud)	gering
USB	hoch	3-5 Meter	gering
Ethernet	hoch	100 Meter	hoch

Tab. 2.4: Vergleich der physikalischen Anbindung an EIB/KNX

Da die Kosten der benötigten Hardware-Bauteile ungefähr gleich hoch sind und die Ethernetanbindung die genannte Flexibilität mit sich bringt, sollte bei einer Planung für einen Neubau die Anbindung per Ethernet bevorzugt werden. Im weiteren Verlauf der Arbeit wird ebenfalls Bezug auf diese Anbindung genommen.

3 Realisierung der Softwarekomponenten

Dieses Kapitel beschreibt nach der Konzeption zur Lösung der Problemstellung und dem Aufstellen von Testkriterien die Umsetzung des Programmierteils. Dieser Teil ist des Weiteren untergliedert in die Entwicklung des Überwachungsplugins und der Steuerungsfunktion. Näher eingegangen wird daher auf zentrale Quelltextauszüge der Softwarekomponenten.

3.1 Konzeption

Die detaillierte Planung der beiden Kernkomponenten zur Überwachung und Steuerung der EIB/KNX-Knoten sind Inhalt dieses Abschnitts. Hierfür wird zuerst die Art und Weise des Bus-Zugriffs evaluiert, um daraus ein Konzept für die Überwachungs- und Steuerungsfunktion zu entwickeln.

3.1.1 Bus-Zugriff

Um den EIB/KNX-Bus mit dem Testsystem zu verbinden, ist eine Hardware-Schnittstelle zum Bus notwendig. Wie in Kapitel 2.5 beschrieben gibt es hierfür grundsätzlich drei Möglichkeiten (RS-232, USB, Ethernet). Aus Gründen des flexiblen Einsatzes im LAN wird ein EIB/KNX-Router der Firma Siemens verwendet, der als Gateway von EIB/KNX zu Ethernet fungiert. Der Zugriff durch eine Softwareschnittstelle auf den Bus kann auf verschiedene Arten erfolgen:

Plattformübergreifend kann der Zugriff per OSGi Java-Framework erfolgen.

Windows basierend kann mittels OPC, einem standardisierten Software-Interface, auf den Bus zugegriffen werden.

Linux bietet durch den EIB Daemon (eibd) eine Möglichkeit, den EIB/KNX-Bus unter anderem mit Kommandozeilen-Tools anzusprechen.

Da Nagios Linux als Betriebssytem voraussetzt, wird für die Entwicklung des Überwachungsplugins und der Steuerungsfunktion der Einsatz des eibd bevorzugt. So kann das Überwachungsplugin und die Steuerungsfunktion ebenfalls auf dem Nagios-Server selbst ausgeführt bzw. entwickelt werden, ohne dass

zusätzliche Lizenzkosten z. B. für eine Windows-Lizenz anfallen. Die Verwendung des eibd hat darüber hinaus den Vorteil, dass sich Scriptsprachen wie Perl oder Shell-Script für die Umsetzung eignen, die eine hohe Popularität bei der Pluginentwicklung für Nagios besitzen. Dies erleichtert eine Weiterentwicklung durch die Entwickler-Gemeinschaft. Außerdem stehen für die Entwicklung Perl-Module für Nagios bereit, die die Programmierung an vielen Stellen deutlich vereinfachen (siehe Abschnitt 3.2.3) und zu großen Teilen vereinheitlichen. Das Zugriffskonzept auf den EIB/KNX-Bus ist in Abbildung 3.1 schematisch dargestellt.

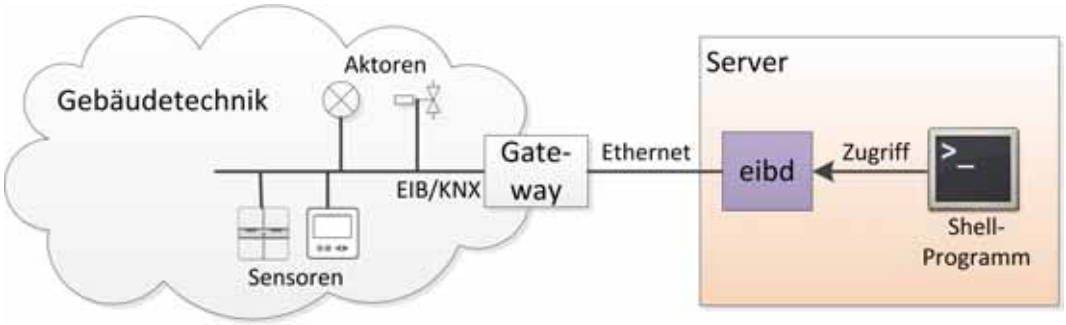

Abb. 3.1: Konzept für die Realisierung des Buszugriffs

Der eibd stellt die Verbindung zum IP-KNX-Gateway her (vgl. Kapitel 2.5) und sorgt für einen kontrollierten Zugriff auf den Bus. Generell kann die Verbindung auch per Universal Serial Bus (USB) oder EIA-232 hergestellt werden und bietet somit eine flexible Verwendung unabhängig von der Verbindungsoption zum EIB/KNX-Bus [Sch09b].

Der EIB Daemon (eibd), entwickelt von *Martin Kögler* an der Technischen Universität Wien, wird als komplettes Software Developement Kit (SDK) - das sogenannte BCU-SDK - zur Verfügung gestellt und bietet bereits funktionsfähige Kommandozeilenprogramme für die Linux Shell, um durch den eibd auf EIB/KNX-Bus zugreifen [Kög10].

Die verwendeten Programme für das Überwachungsplugin und die Steuerungsfunktion werden in den nächsten beiden Unterkapiteln vorgestellt.

3.1.2 Überwachungsplugins

Die Überwachungsplugins müssen die Daten des Gerätes abfragen und auswerten, die den korrekten Betrieb des EIB/KNX-Moduls kennzeichnen. Diese Werte können auch für jeden Knoten einzeln durch die Diagnosefunktion „Geräte-Info“ der ETS3 abgefragt werden, deren Ausgabe in Abbildung 3.2 als Screenshot beispielhaft dargestellt ist.

Laut [Mey04] sind für das einwandfreie Funktionieren der EIB/KNX-Komponenten vier Parameter wichtig:

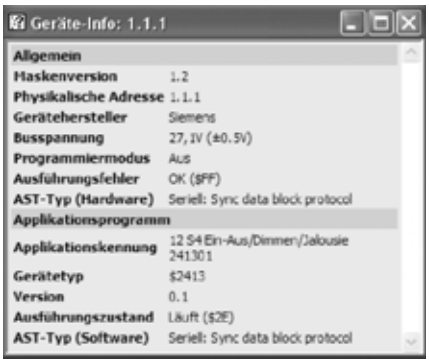

Abb. 3.2: Diagnosefunktion (Geräteinfo) der ETS3

Die *Spannung* muss im Bereich von 21-30 Volt liegen, da die Geräte sonst mit zu wenig Spannung versorgt werden, was zu Fehlfunktionen der Module führen würde. Bei einer zu hohen Spannung besteht die Gefahr, dass die Geräte beschädigt und im schlimmsten Fall zerstört werden. Der Spannungswert kann über den Analog-Digital-Converter (ADC) auf Kanal 1 ausgelesen werden [Sie05a].

Alle im *Programmiermodus* befindlichen Geräte können nicht normal betrieben werden. Falls sich ein Aktor oder Sensor unabsichtlich in diesem Modus befindet besteht außerdem die Gefahr, dass dieser versehentlich programmmiert wird. Deswegen sollte sich zur gleichen Zeit immer nur ein Gerät im Programmiermodus befinden. Ob sich ein Gerät im Programmiermodus befindet, wird in Bit 0 an der Speicheradresse 0x0060 angezeigt [Sie05a].

Sollten bei der Ausführung eines Applikationsprogramms des EIB/KNX-Gerätes *Ausführungsfehler* auftreten, werden diese in einem 1 Byte großen Flagregister gesetzt. Der Wert wird im EEPROM an Speicherstelle 0x010D abgelegt (siehe Tabelle 3.1). Grundsätzlich muss der Wert dieses Registers 0xFF aufweisen. Das Flag ist gesetzt, wenn das entsprechende Bit=0 ist (Bit=1 bedeutet kein Fehler) [Sie05b].

Der *Ausführungszustand* beschreibt den aktuellen Zustand des Applikationsprogramms und es sollten für die korrekte Funktion die unteren vier Bits den Wert 0xE aufweisen. Diese Werte bedeuten, dass die notwendigen Layer (Application, Transport und Link) aktiviert sind. Der Wert wird im unteren Random Access Memory (RAM)-Bereich des Geräts auf die Speicheradresse 0x0060 geschrieben und hat die in Tabelle 3.2 beschriebenen Flags [Sie05b].

Daraus ergeben sich für die Integration in Nagios folgende Schwellenwerte für die Zustände Critical, Warning und Ok der überprüften Parameter, dargestellt in Abbildung 3.3.

Sämtliche Parameter werden in Nagios als zu überwachender Dienst behandelt. Die Schwellenwerte (siehe Abbildung 3.3) werden dem Plugin als Parameter übergeben und unter `check_command` im Abschnitt der Service-Definition

Bit#	Bezeichnung	Beschreibung
7	-	Muss 1 sein
6	SYS3_ERR	Interner Systemfehler: Memory Control Block defekt
5	SYS2_ERR	Interner Systemfehler: Überhitzungsschutz
4	OBJ_ERR	RAM Flag Fehler
3	STK_OVL	Stacküberlauf festgestellt
2	EEPROM_ERR	EEPROM-Check hat Prüfsummenfehler festgestellt
1	SYS1_ERR	Interner Systemfehler: Paritätsbit defekt
0	SYS0_ERR	Interner Systemfehler: Nachrichtenspeicher-Offset defekt

Tab. 3.1: Flags an Speicherstelle 0x010D (Ausführungsfehler) [Sie05b]

Bit#	Bezeichnung	Beschreibung
7	PARITY	Gerade Parität für Bit 0-6
6	DM	BCU in Download Modus
5	UE	Benutzerprogramm läuft
4	SE	Serielles Interface aktiviert
3	ALE	Applikationslayer aktiviert
2	TLE	Transportlayer aktiviert
1	LLM	Linklayer aktiviert
0	PROG	Gerät ist im Programmiermodus

Tab. 3.2: Flags an Speicherstelle 0x0060 (Ausführungszustand) [Sie05b]

Abb. 3.3: Schwellenwerte der Parameter für die verschiedenen Zustände

angegeben. Die genaue Konfiguration am Beispiel der Abfrage des Programmiermodus ist im Anhang A.2 angegeben. Die vollständigen Konfigurationsdateien befinden sich auf beiliegender CD.

Als Hostcheck für jede EIB/KNX-Komponente, vergleichbar mit dem ICMP-

Echo-Request bei einem Server, kommt die Abfrage der Maskenversion zum Einsatz. Die Maskenversion identifiziert die Bus Coupling Unit (BCU)-Version und damit den Funktionsumfang der Application Programming Interface (API) des Geräts. Der zurückgelieferte Wert ist dabei nebensächlich. Ein Timeout der Anfrage bzw. das Ausbleiben einer Antwort deutet darauf hin, dass das Gerät nicht vorhanden ist, worauf die Komponenten als „Down“ zu markieren sind (vgl. Abschnitt 2.3.1).

Die in Unterkapitel 3.1.1 angesprochenen Kommandozeilenprogramme `mread`, `progmodestatus`, `madcread` und `mmaskver` erfüllen folgende Anforderungen, um die Abfrageparameter aus den Geräten auszulesen [Kög08]:

- Speicherstellen auslesen (0x0060 und 0x010D): `mread`
- Status des Programmiermodus auslesen: `progmodestatus`
- Spannung des ADC auslesen: `madcread`
- Maskenversion auslesen: `mmaskver`

Die Rückgabe der ausgelesenen Werte erfolgt auf der Standard-Ausgabe (stdout) und kann mit Hilfe von regulären Ausdrücken weiter verarbeitet werden. Als Programmiersprache für das eigentliche Plugin kommt Perl zum Einsatz, was verschiedene Vorteile mit sich bringt:

1. Die Scriptsprache Perl ist sehr weit verbreitet und auf über 100 Plattformen lauffähig [Per10].
2. Mit Perl können fertige Module für die Entwicklung von Plugins für Nagios genutzt werden [Bar09].
3. Die Performance von Perl innerhalb von Nagios kann mit dem integrierten Interpreter ePN beschleunigt werden [Bar09].
4. Perl ist eine Open Source-Sprache, lizensiert unter der GPL [Per10].

Unabhängig von der verwendeten Programmiersprache werden bei der Entwicklung der Überwachungsplugins die *Nagios plug-in development guidelines* berücksichtigt, um eine Weiterentwicklung durch Dritte nicht unnötig zu erschweren und bewährte Vorgehensweisen anzuwenden [Tea09]. Diese sind ebenfalls Teil der Untersuchung bzw. der Testkriterien der Plugins (siehe Kapitel 3.1.4).

3.1.3 Steuerungsfunktion

Um die Steuerungsfunktion über die Oberfläche von Nagios zu realisieren, muss eine Schnittstelle zu den im BCU-SDK enthaltenen Kommandozeilenprogrammen hergestellt werden, die wiederum über den eibd die Kommunikation zum

Bus übernehmen. Durch einen Mausklick auf eine Schaltfläche der Weboberfläche soll es dann möglich sein, die Aktoren des Testsystems zu steuern. Um die Aufgabe umzusetzen, stehen verschiedene serverseitige Technologien zur Verfügung. Am weitesten verbreitet sind:

- PHP: Hypertext Preprocessor (PHP)
- Active Server Page (ASP).net
- JavaServer Pages (JSP)
- Common Gateway Interface (CGI)

Das wichtigste Auswahlkriterium für eine geeignete Schnittstelle ist die Forderung nach einem *offenen Standard*, der unabhängig von einem Hersteller entwickelt wird. Damit soll gewährleistet werden, dass auch in Zukunft der Weiterentwicklung durch Dritte oder einer Community nichts im Wege steht. Ein weiteres Kriterium ist ein *geringer Konfigurationsaufwand*. Nach Möglichkeit sollte die Schnittstelle kein weiteres Apache-Modul benötigen bzw. das entsprechende Modul in der Grundkonfiguration mitbringen und auf Linux lauffähig sein. Dies hält, neben dem Einrichtungsaufwand, auch die Angriffsfläche in Bezug auf die Systemsicherheit gering. Die dritte Bedingung für eine Bevorzugung ist der einfache *Zugriff auf die Kommandozeilen-Programme* des BCU-SDK, das für die Verbindung zum Bus mithilfe des eibd genutzt werden soll. Zum Schluss sollte, wie beim Überwachungsplugin, nach Möglichkeit auch *Perl als Scriptsprache* eingesetzt werden, um den Entwicklungsaufwand gering zu halten und um die gesammelten Erfahrungen weiter anzuwenden und ausbauen zu können.

PHP als freie und offene Scriptsprache ist plattformunabhängig, sehr weit verbreitet und bietet umfangreiche Möglichkeiten für die Entwicklung dynamischer Webseiten. Für die Unterstützung durch den Apache Webserver müssen entsprechende PHP-Module konfiguriert werden. Die Einbindung erfolgt direkt im HTML-Code. Die Nutzung von Perl durch PHP wird nicht unterstützt.

Das von Microsoft entwickelte *ASP*.net lief ursprünglich nur unter Windows bzw. dessen Webserver Internet Information Server (IIS). Mittlerweile gibt es allerdings auch Portierungen für Linux bzw. nachrüstbare Module für einzelne Webserver, darunter auch der Apache. Allerdings handelt es sich nicht um eine quelloffene, sondern um eine proprietäre Software. ASP.net unterstützt neben Perl auch andere Scriptsprachen und kann seine Stärken in reinen Microsoft-Umgebungen durch Anbindungen an Microsoft-Produkte ausspielen.

Mit *JSP* von SUN Microsystems können Java-Programme direkt in den HTML-Quellcode eingebunden werden. Zur Ausführung der JSP ist die Java Virtual Machine (JVM) nötig, die in den eingesetzten Webserver integriert werden muss. JSP ist plattformunabhängig und bietet zahlreiche Klassenbibliotheken. Externe Programme können nicht mit Perl angesprochen werden.

Die *CGI*-Schnittstelle ist als offener Standard konzipiert, in RFC 3875 spezifiziert und darüber hinaus plattformunabhänig. Der Konfigurationsaufwand zur Einbindung in den Webserver ist minimal, da das CGI-Modul, standardmäßig integriert, ebenfalls von Nagios genutzt wird. Ein weiterer Vorteil von CGI ist, dass nahezu jede Programmiersprache verwendet werden kann, darunter auch Perl. Ein Nachteil der Lösung ist die relativ geringe Perfomance, da für jeden Aufruf eines Perl-Scripts ein eigener Interpreter-Prozess gestartet werden muss [TT09].

	CGI	**PHP**	**ASP.net**	**JSP**
OpenSource	ja	ja	nein	ja
Konfigurations-aufwand	gering	mittel	hoch	hoch
Linux-Unterstützung	ja	ja	ja	ja
Perl-Unterstützung	ja	nein	ja	nein
Webserver-Unterstützung	hoch	hoch	mittel	mittel
Weitere Vorteile	viele Sprachen verwendbar	viele Schnitt-stellen verfüg-bar	viele Sprachen verwendbar	viele Klassen-bibliotheken nutzbar
Weitere Nachteile	Geringe Performance	-	Nicht für alle Webserver ver-fügbar	-

Tab. 3.3: Vergleich von serverseitigen Technologien

Nach dem Vergleich der verschiedenen Möglichkeiten (siehe Tabelle 3.3) zur Interaktion mit dem EIB/KNX-Bus wird als Schnittstelle CGI ausgewählt. Der offene Standard bietet die größte Flexibilität in Bezug auf die Webserverwahl, Perl wird als Programmiersprache unterstützt und bringt den geringsten Konfigurationsaufwand mit sich. Die geringe Performance spielt für die rudimentären Steuerungsfunktionen keine Rolle, da sich in der Praxis die Belastung durch hohe Zugriffszahlen nicht ergeben sollte.

Durch die Wahl der CGI-Schnittstelle ist das Konzept in Abbildung 3.4 für die Steuerung der EIB/KNX-Komponenten entwickelt worden.

Abb. 3.4: Konzept für die Realisierung der Steuerungsfunktion.
Quelle: Eigene Darstellung in Anlehnung an [TT09].

Mit Hilfe der Weboberfläche der Steuerung wird im ersten Schritt ein HTTP-Request ausgeführt, der nach einem Klick auf die Steuerschaltfläche die Formulardaten überträgt und das CGI-Script aufruft. Der Webserver sorgt als nächstes für die Ausführung des Perl-Scripts, das durch die CGI-Schnittstelle aufgerufen wird. In diesem werden wiederum die nötigen Kommandozeilenprogramme `groupwrite` bzw. `groupswrite` aufgerufen und das von ihnen erzeugte Telegramm zum EIB/KNX-Bus weitergeleitet. Den Rückgabewert, also ob der Befehl erfolgreich abgesetzt werden konnte oder ob Fehler aufgetreten sind, erhält das Perl-Script als Antwort. Im sechsten Schritt leitet er diese abschließend an den Webserver weiter. Im HTTP-Response schickt der Webserver alle vom Perl-Script erzeugten Ausgaben an den Client weiter, wie zum Beispiel den Aufruf der Bestätigungsseite. Diese informiert den Benutzer über das Ergebnis der Steuerungsanweisung.

3.1.4 Testkriterien

Neben den Testkriterien wird ein Sollzustand festgelegt, den es zu erfüllen gilt. Im nachfolgenden Kapitel werden die Testmethoden erläutert und die Ergebnisse der Messungen präsentiert. Abschließend erfolgt in Kapitel 5 eine Bewertung.

Überwachungsplugin

Die Überwachung der EIB/KNX-Geräte durch die Nagios-Plugins wird auf sechs Kriterien hin untersucht, um die Funktionsfähigkeit zu ermitteln.

Geräteunterstützung: Im Idealfall sollen alle Gerätetypen unabhängig vom Hersteller unterstützt werden, um eine breite Einsatzmöglichkeit der Plugins zu gewährleisten. Bei vielen Installationen und vor allem Neuplanungen werden oft Geräte eines Herstellers verwendet. Allerdings sollte die spätere Erweiterung durch Geräte anderer (ggf. kostengünstigere) Hersteller nicht eingeschränkt sein, weswegen diese Testbedingung mit 25% gewichtet wird.

Richtigkeit der Telegramme Das Telegramm für die Abfrage der Werte sowie dessen Antwort müssen die korrekten und zu erwartenden Werte aufweisen. Für das reibungslose Funktionieren der Plugins ist dieses Kriterium essentiell und wird deswegen mit 25% bewertet.

Zuverlässigkeit: Um eine verlässliche Überwachung zu gewährleisten, müssen die Plugins zuverlässig funktionieren. So können Fehlalarme und falsche Testergebnisse durch die Plugins vermieden werden. Die Zuverlässigkeit

stellt zwar ein wichtiges Kriterium dar, kann aber durch den Hard-Soft-State-Mechanismus von Nagios ausgeglichen werden und wird daher mit 15% bewertet.

Entwicklungsrichtlinien: Die Richtlinien zum Entwickeln von Nagios-Plugins [Tea09] sollen eingehalten werden. Dies ist wichtig, da eine Weiterentwicklung durch Dritte nicht behindert werden und der Veröffentlichung der Plugins auf der Austauschplattform `http://exchange.nagios.org/` nichts im Wege stehen soll. Für die Funktionalität der Plugins stellt dieses Qualitätsmerkmal eine untergeordnete Rolle dar und geht daher mit 15% in die Gesamtwertung ein.

Abfragezeit: Die Zeit, die zur Abfrage eines Wertes benötigt wird, sollte 10 Sekunden nicht übersteigen. Bei Laufzeiten von länger als 10 Sekunden erfolgt unter Zugrundelegung der Standardwerte von Nagios ein Timeout und somit der Abbruch der Anfrage. Die Abfragezeit wird in der Gesamtbewertung mit 10% gewichtet, da längere Zeiten durch Setzen eines höheren Timeouts notfalls umgangen werden könnten.

Busauslastung: Die Nagios-Plugins erzeugen durch die regelmäßigen Abfragen zwangsläufig Last auf dem EIB/KNX-Bus. Bei einem Abfrageintervall von 5 Minuten soll in der Testumgebung die Last durchschnittlich nicht höher als 5% liegen. Die maximale Busauslastung darf nicht auf Werte über 90% steigen. Gewichtet ist dieses Kriterium mit ebenfalls 10%, da eine höhere Auslastung durch seltenere Abfrageintervalle kompensiert werden könnte.

Die Kriterien, Soll-Zustand und Gewichtung sind in Tabelle 3.4 zusammengefasst.

Steuerungsfunktion

Für die Steuerungsfunktion werden im Folgenden fünf Testkriterien festgelegt, die für den Einsatz in der Praxis als sinnvoll erachtet werden:

Geräteunterstützung: Alle vorhandenen Aktoren im Testaufbau (Licht, Jalousie, Heizventil, dimmbares Licht) sollen grundsätzlich beeinflussbar bzw. kontrollierbar sein. Dies umfasst das Ein-/Ausschalten der Lichtbänder und Dimmen des Lichts mit relativen und absoluten Werten. Auch das Stellen des Heizventils soll mit relativen und absoluten Prozentwerten steuerbar sein. Die Jalousie muss darüber hinaus schrittweise und komplett geöffnet und geschlossen werden können. Außerdem muss das Bedienen aller Lichtbänder gleichzeitig (Zentralfunktion, alle Lichtbänder an/aus) möglich sein. Dieses Kriterium stellt die Kernfunktion

Kriterium	Soll-Zustand	Wichtung
Geräteunterstützung	Alle verwendeten Aktoren und Sensoren sollen abgefragt werden können	25%
Richtigkeit der Telegramme	Anfrage und Rückantwort aller erzeugten Telegramme müssen korrekt sein	25%
Zuverlässigkeit	Die Fehlerquote bei der Abfrage darf die Grenze von 0,5% nicht übersteigen	15%
Entwicklungsrichtlinien	Die *Nagios plug-in development guidelines* [Tea09] sollen vollständig eingehalten werden	15%
Abfragezeit	Die Zeit für eine Abfrage darf 10 Sekunden Standardtimeout nicht überschreiten	10%
Busauslastung	Die Auslastung des Busses darf durchschnittlich 5%, maximal 90% betragen	10%

Tab. 3.4: Testkriterien für das Überwachungsplugin

der Steuerung dar und geht daher mit 40% in die Gesamtwertung der Steuerung ein.

Richtigkeit der Telegramme: Die versendeten Befehle, kodiert in Telegrammen für den EIB/KNX-Bus, sollen die zu erwartenden Werte aufweisen. Für den problemlosen Aufruf ist diese Bedingung elementar und aufgrund dessen mit 20% in der Gesamtbewertung gewichtet.

Zuverlässigkeit: Das Absetzen eines Steuerungswunsches muss verlässlich funktionieren und ist daher von hoher Bedeutung für die Bewertung der Gesamtfunktionalität. Aus diesem Grund ist dieses Kriterium mit 20% gewichtet. Die Zuverlässigkeit muss minimal 99,5% betragen.

Versandzeit: Das Versenden des Telegramms durch Ausführen des Steuerungsbefehls soll weniger als 500ms dauern, um ein Schaltverhalten mit geringer Verzögerung zu erreichen. Die Schaltzeit soll in etwa dem Betätigen eines reellen Schalters entsprechen, um den Benutzerkomfort nicht einzuschränken, und ist deshalb mit 15% bewertet.

Gruppenadressarten: Beide Adressierungsarten (vgl. Abbildung 2.8) sollen verwendet werden können, um Gruppenadressen zu spezifizieren. Hiermit kann der Projektant die eingesetzte Adressierungsart weiter verwenden, wodurch (insbesondere Umrechnungs-) Fehler vermieden werden. Da diese Bedingung nicht die grundsätzliche Funktion bzw. den potentiellen Nutzen in Frage stellt, wird sie nur mit 5% bewertet.

Die Kriterien, Gewichtung und der Soll-Zustand sind in Tabelle 3.5 zusammengefasst. Die Testmethoden werden in Kapitel 4 dargestellt.

Kriterium	Soll-Zustand	Wichtung
Geräteunterstützung	Licht, Jalousie, Heizventil, dimmbares Licht, Zentralfunktion sollen steuerbar sein	40%
Richtigkeit der Telegramme	Das Telegramm wurde korrekt erzeugt und enthält die zu erwarteten Daten	20%
Zuverlässigkeit	Die Zuverlässigkeit des Steuerungsbefehls soll 99,5% betragen	20%
Versandzeit	Versenden des Steuerungstelegramms soll weniger als 500ms dauern	15%
Gruppenadressarten	Zwei- und dreistufige Adressen können verwendet werden	5%

Tab. 3.5: Testkriterien für Steuerungsfunktion

3.2 Umsetzung

Dieser Abschnitt behandelt die Installation und Konfiguration des benötigten Betriebssystems, des Nagios-Dienstes und des BCU SDK. Im zweiten Unterkapitel werden Kernkomponenten des Überwachungsplugins und der Steuerungsfunktion näher erläutert.

3.2.1 Testumgebung EIB/KNX-Szenario

Im EIB/KNX-Labor der Fachhochschule Erfurt wurde aus den vorhandenen Komponenten das in Abbildung 3.5 skizzierte Test-Szenario aufgebaut. Die detaillierte Beschreibung der Komponenten mit Artikelnummer, Bezeichnung und Applikation befinden sich im Anhang A.3. Ein Foto des Testaufbaus ist als Anhang A.6 hinterlegt.

Die Programmierung der Komponenten sollte sich so weit wie möglich an einem praktischen Beispiel orientieren. Die generelle Funktionalität wurde in fünf Hauptgruppen unterteilt, die im weiteren Verlauf näher charakterisiert werden:

- Beleuchtungsfunktion
- Zentralfunktionen
- Jalousie
- Raumregelung
- Display

Die *Beleuchtungsfunktion* umfasst das einfache Schalten der Lichtbänder 1-3 bzw. die zusätzliche Dimmung des Lichtbands 3. Außerdem ist eine Ausschaltverzögerung, wie sie etwa in einem Treppenhaus zum Einsatz kommt, und ein

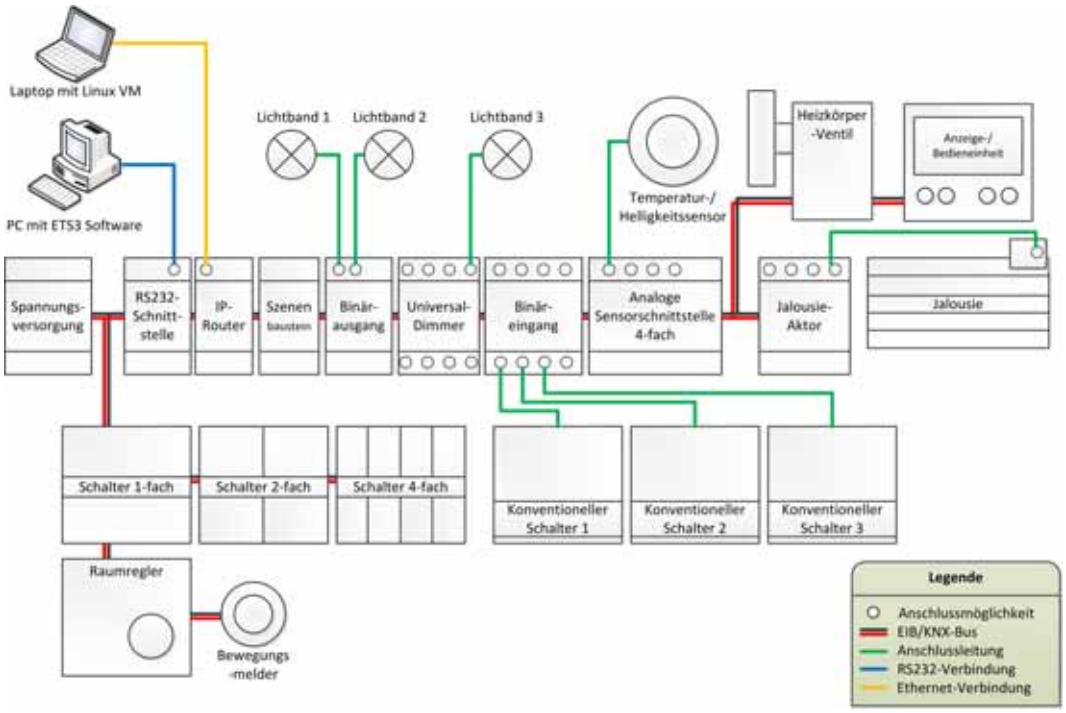

Abb. 3.5: Skizze der Testumgebung im EIB/KNX-Labor

Dauerlicht (etwa für die Reinigung, Umzug etc.) integriert. Als Aktor kommt der 4-fach-Taster zum Einsatz.

Zur *Zentralfunktion* zählt das zentrale Ausschalten der Beleuchtung, welche durch den 2-fach-Taster angesteuert wird. Dabei werden alle vorhandenen Leuchtbänder ausgeschaltet, um sicherzustellen, dass keine Lampen (zum Beispiel über Nacht) leuchten.

Die Steuerung der *Jalousie* beinhaltet neben dem kompletten Auf- und Abfahren der Lamellen auch den Schrittbetrieb und eine Sicherheitsfunktion. Bei zu starkem Wind könnten ohne diese Funktion die Lamellen beschädigt werden, weswegen bei einer gewissen Windgeschwindigkeit die Jalousie in eine Sicherheitsstellung gefahren wird. Der „Windalarm“ wird über einen konventionellen analogen Schalter simuliert, die Steuerung der Jalousie über einen 1-fach-EIB/KNX-Schalter.

Bei der *Raumregelung* ist zusätzlich die Simulation eines geöffneten Fensters sowie die Heizzustände Frostschutz (simuliert durch konventionellen Schalter) und Nachtabsenkung (einschaltbar durch 2-fach Schalter) realisiert worden. Die Regelung der Raumtemperatur wird über einen Heizventil-Aktor umgesetzt.

Für die *Anzeige* und die Steuerung der Geräte kommt ein monochromes 4-Zeilen-Display hinzu, das neben statischem Text die Temperatur, den Luxwert des Helligkeitssensors, die Temperatur des Raumreglers und die Ausgabe der

Ventilposition wiedergibt. Außerdem kann mit dessen Hilfe Lichtband 1 und 2 geschaltet und der Basissollwert für den Raumregler eingestellt werden.

3.2.2 Installation und Konfiguration der Testumgebung

Die Installation und Konfiguration der Testumgebung erfolgte in vier Schritten.

1. Installation Linux
2. Installation Nagios
3. Installation BCU SDK und eibd
4. Konfiguration Nagios

Zuerst wurde eine Testumgebung in eine Virtuelle Maschine (VM) installiert. Die eingerichtete VM ist mit einem 2,5 GHz Prozessor und 512 MB RAM ausgestattet. Da Nagios Linux als Betriebssystem voraussetzt, entschied sich der Autor für die Distribution Ubuntu in Version 10.4, was die aktuelle Stable-Version darstellt (Stand: 29. Juli 2010). Das auf Debian basierende Linux-Derivat bietet den Vorteil der einfachen Installation mittels des Paketverwaltungsprogramms Aptitude, unterstützt eine Vielzahl an Hardware und ermöglicht die einfache Nachinstallation der restlichen Softwarepakete.

Nach der Installation und dem Einspielen aller vorhandenen Updates des Systems wurde Nagios in Version 3.2 durch Aptitude installiert. Die Routine installiert und konfiguriert gleich den Apache Webserver mit, der für den Betrieb und die Bedienung von Nagios per Weboberfläche empfohlen wird. Der Webserver ist sehr weit verbreitet, bietet umfangreiche Konfigurationsmöglichkeiten und ist sehr gut dokumentiert. Grundsätzlich kann aber jeder Webserver mit aktivierter CGI-Unterstützung genutzt werden. Nagios benötigt für den Betrieb eine Gruppe und einen Benutzer, unter dessen Rechten der Nagios-Server und der Aufruf der Plugins läuft. Das Setup-Programm legt hierfür die Gruppe `nagios` an und fügt dieser den gleichnamigen Benutzer `nagios` hinzu.

CGI-Scripte führt Nagios unter der User-ID der Gruppe aus, mit dessen Rechten der Webserver Apache läuft (`www-data`). Damit die Scripte von Nagios ausgeführt werden können, muss der Benutzer nagios der Gruppe www-data hinzugefügt werden.

Um auch bei der Installation des BCU SDKs die Programmpakete durch Aptitude zu installieren, müssen die Paket-Quellen (siehe Listing 3.1) der Datei `\etc\apt\sources.list` hinzugefügt werden.

```
1 deb http://www.auto.tuwien.ac.at/~mkoegler/debian eib main
```

Listing 3.1: Hinzufügen zur Datei sources.list

Außerdem benötigt die Bibliothek `libstdc++5`, die direkt von der Debian-Webseite (`http://packages.debian.org/lenny/libstdc++5`) bezogen werden kann.

Nach der fertig gestellten Installation muss der eibd mit den in Listing 3.2 angegebenen Optionen gestartet werden, um sich mit dem IP-Router zu verbinden.

```
1 /usr/bin/eibd -d -p /var/run/eibd.pid -u -i ipt
     :192.168.1.2:3671
```

Listing 3.2: Starten des eibd

Das Programm eibd wird als Daemon (-d) gestartet und die erzeugte Prozess-ID (PID) in eine Datei geschrieben (/var/run/eibd.pid). Die Option -u sorgt dafür, dass der Daemon zur lokalen Interprozesskommunikation einen Unix Domain Socket benutzt und durch die Option -i für die eingehende Verbindung am Port 6720 lauscht. Der letzte Aufrufparameter wird als Uniform Resource Locator (URL) bezeichnet, die eine Verbindung zum IP-Router herstellt (IP-Router mit der IP-Adresse 192.168.1.2 auf Port 3671) und über das EIBnet/IP Tunneling-Protokoll kommuniziert. Neben dieser URL gibt es noch einige andere, z. B. wenn die Verbindung über die USB-Schnittstelle hergestellt werden soll [Kög08].

Um eibd bei jedem Neustart des Servers automatisch zu starten, empfiehlt es sich, ein Startscript für den Daemon zu erstellen.

Bei der Konfiguraton von Nagios wird größtenteils die Standardkonfiguration belassen. Als Anpassung erfolgt das Zulassen von externen Kommandos, damit Nagios über die Weboberfläche mittels CGI-Scripten gesteuert werden kann. In der Datei `/etc/nagios3/nagios.cfg` wird dazu der Parameter `check_external_commands=1` gesetzt. Danach wird der Besitzer des Ordners mit Command-File (`/var/lib/nagios3/rw/`), das die Steuerung ermöglicht, auf den Benutzer und die Gruppe `nagios` geändert. Außerdem bekommen der Benutzer und die Gruppe Lese-/Schreib- und Ausführungrechte des Ordners und das Sticky-Bit wird gesetzt, damit neue Dateien die gleiche Berechtigung bekommen (siehe Listing 3.3).

```
1 chown nagios.nagios /var/lib/nagios3/rw/
2 chmod u+rwx /var/lib/nagios3/rw/
3 chmod g+rwx /var/lib/nagios3/rw/
4 chmod g+s /var/lib/nagios3/rw/
```

Listing 3.3: Anpassungen für das Erlauben externer Kommandos

Die Konfigurationsdateien werden in Nagios durch die Datei- und Ordnerstruktur objektorientiert angelegt. Somit können Hosts und Services schnell gefunden und bearbeitet werden (siehe Listing A.11 im Anhang).

3.2.3 Implementierung der Softwarekomponenten

Wie bereits im vorhergehenden Kapitel beschrieben, teilt sich dieser Abschnitt in die Implementierung des Überwachungsplugins und die der Steuerungsfunktion der EIB/KNX-Komponenten. Hierbei werden im ersten Teil Aspekte beleuchtet, die alle fünf Plugins betreffen, um dann im Detail auf den Kern des jeweiligen Plugins einzugehen. Der vollständige und kommentierte Code für beide Teile befindet sich auf der beiliegenden CD.

Überwachungsplugins

Ausgehend vom Programmablaufplan (siehe Anhang A.3) erfolgt bei allen entwickelten Plugins in einem ersten Schritt die Instantisierung der Perl-Klasse `Nagios::Plugin`. Die Klasse übernimmt im Programmverlauf das Vergleichen der gesetzten Schwellenwerte mit den ermittelten Werten und das Hinzufügen von Performance-Werten, die später gesondert in einem Verlauf ausgewertet werden können. Nach der Initialisierung der Variablen werden die übergebenen Parameter, mit denen das Plugin gestartet wurde, in diese gespeichert. Für das Extrahieren der Parameter bietet Perl über das Modul `Getopt::Long` die Funktion GetOptions an, die die definierten String-Werte und eine mögliche Mehrfachangabe von Verbose- und Debug-Parametern an die definierten Variablen übergibt. Sollten Parameter nicht mitgeliefert worden sein, die aber zwingend benötigt werden (URL und physikalische EIB/KNX-Adresse) oder undefinierte Parameter angegeben worden sein, beendet sich das Programm mit einer entsprechenden Meldung und dem Exit-Code „Unknown“. Die eingebettete Perl Online Documentation (POD) wird durch den gesetzten Parameter -h bzw. –help aufgerufen. Die Version des Plugins liefert der Aufruf mit Parameter -V bzw. –version zurück.

Ist kein Hilfe- oder Versionsparameter und kein anderer ungültiger Parameter angegeben worden, erfolgt das Ausführen der Hauptteils des Plugins, was auf der rechten Seite des Programm-Ablauf-Plans im Anhang A.3 dargestellt ist. Bevor die Abfrage von Netzressourcen erfolgt, wird der Timeout aktiviert, der nach standardmäßigen 10 Sekunden Prozesslebenszeit selbigen Prozess mit der Meldung „Timeout reached“ beendet. Während der regulären Laufzeit erfolgt die Abfrage des EIB/KNX-Geräts mit Hilfe des BCU SDK-Kommandozeilenprogramms. Aufgerufen wird es mit der physikalischen Adresse und der URL zum eibd. In der Testumgebung ist der eibd auf der gleichen Maschine installiert, weswegen die URL 127.0.0.1 (Localhost) lautet. Solange dieses eine Ausgabe zurückliefert, wird mit regulären Ausdrücken die Gültigkeit der Ausgabe überprüft. Bei nicht gültigen Werten, etwa bei einer Fehlermeldung des Kommandozeilenprogramms, erfolgt das Beenden des Prozesses mit Critical als Rückgabewert. Bei der Ermittlung gültiger Daten werden diese den Performance-Daten der Nagios-Plugin-Instanz hinzugefügt und die Werte

mit den gesetzten Schwellenwerten für Critical und Warning verglichen, was beides die Nagios-Klasse übernimmt. Danach wird der Timeout deaktiviert und das Prüfergebnis (Ok, Warning, Critical) zusammen mit einer einzeiligen Statusmeldung zurückgeliefert. Der Pluginaufruf ist damit beendet.

Die Kernfunktion für die Busspannung ist der Programmaufruf `madcread`, der den Analog-Digital-Converter des Gerätes auf Kanal 1 einmal ausliest (siehe Listing 3.4).

```
1 open (OUT, "LANG=C /usr/bin/madcread $url $what 1 1 2>&1 |")
2   or $np->nagios_die("can't start /usr/bin/madcread")
```

Listing 3.4: Abfrage der Busspannung

Das Kommandozeilenprogramm liefert als Ausgabe „Value:“ und einen ganzzahligen Integer-Wert zurück. Für die Auswertung muss dieser ganzzahlige Wert aus der Ausgabe extrahiert werden, was mittels regulären Ausdrücken erfolgt. Durch den nun allein stehenden Wert lässt sich mit dem Faktor 0.15 die Spannung berechnen, die am abgefragten EIB/KNX-Gerät anliegt. Die Auflösung des ADC beträgt demnach 0.15 Volt.

Um den Ausführungszustand und den Ausführungsfehler zu ermitteln, müssen Speicherplätze des unteren RAM (Lower RAM) und des EEPROM ausgelesen werden. Für das Speicherleseprogramm `mread` ergibt sich jedoch nur ein Unterschied in der Adresse (siehe Listing 3.5 und 3.6).

```
1 open (OUT, "LANG=C /usr/bin/mread $url $what 60 1 2>&1 |")
2   or $np->nagios_die("can't start /usr/bin/mread");
```

Listing 3.5: Abfrage des Ausführungszustands

```
1 open (OUT, "LANG=C /usr/bin/mread $url $what 10D 1 2>&1 |")
2   or $np->nagios_die("can't start /usr/bin/mread");
```

Listing 3.6: Abfrage des Ausführungsfehlers

In beiden Fällen ist die Länge des Registers 1 Byte, woraus sich der zweite Aufrufparameter (nach der URL und der EIB/KNX-Adresse) ergibt. Als Rückgabewert wird direkt der hexadezimale Wert der Speicherstelle geliefert, der ebenfalls durch einen regulären Ausdruck überprüft wird.

Der Programmiermodus, abgefragt durch das Programm `progmodestatus`, liefert als Rückgabewert eine Zeichenkette, die bei eingeschaltetem Programmiermodus des Geräts „in programming mode“ lautet, bei ausgeschaltetem Modus mit einem vorangestellten „not“. Die beiden Zustände werden durch eine einfache if-Bedingung verglichen. Zur Abfrage werden lediglich URL und physikalische Adresse (gespeichert in der Variable $what) des Gerätes benötigt (siehe Listing 3.7).

```
1 open (OUT,"LANG=C /usr/bin/progmodestats $url $what 2>&1 |")
2   or $np->nagios_die("can't start /usr/bin/progmodestatus");
```

Listing 3.7: Abfrage des Programmiermodus

Der Rückgabewert der Maskenversion, abgefragt für den Host-Check eines Gerätes, wird nicht weiter ausgewertet, da der Aufruf des Plugins ohne Angabe von Schwellenwerten aufgerufen wird. Das Programm muss lediglich irgendeine Maskenversion liefern. Das Programm `mmaskver` liefert dazu den String „Mask:“ und einen vierstelligen Maskenwert zurück, der ebenfalls durch reguläre Ausdrücke extrahiert wird.

```
1 open (OUT, "LANG=C /usr/bin/mmaskver $url $what 2>&1 |")
2   or $np->nagios_die("can't start /usr/bin/mmaskver");
```

Listing 3.8: Abfrage der Maskenversion

Die fertig eingebundenen Plugins werden wie in Abbildung 3.6 zu sehen dargestellt:

Host	Service	Status	Status Information
1.1.10: Jalousieaktor	ProgrammingMode	OK	PROGRAMMINGMODE OK - not in programming mode (0)
	RunningError	OK	RUNNINGERROR OK - Errorflag: 255 (0xFF
	RunningState	OK	RUNNINGSTATE OK - Errorflag: 14 (0xE)
	Voltage	OK	VOLTAGE OK - Voltage: 26.85 Volt

Abb. 3.6: Eingebundene EIB/KNX-Plugins in Nagios

Steuerungsfunktion

Für die Steuerung der Komponenten mittels Weboberfläche (siehe Abbildung 3.7) muss der entsprechende Datenpunkt-Typ betrachtet werden, der das Format des gesendeten Wertes festlegt. Im einfachsten Fall handelt es sich beim Ein- bzw. Ausschalten von Aktoren um einen DPT 1, der einen Boolean-Wert (1 Bit) entgegennimmt. Die genaue Bedeutung des Wertes hängt vom verwendeten EIS-Untertyp ab. Im genannten Beispiel ist dies der Typ 1.001 (DPT_Switch). Ein gesetztes Bit bedeutet demnach „An“, eine 0 entsprechend „Aus“ [Ass10]. Die verwendeten Datenpunkt-Typen können aus Tabelle A.2 im Anhang entnommen und in ihrer Vollständigkeit in [Ass10] nachgeschlagen werden. Speziell für die Ansteuerung des Lichtbands 3 mit absoluten Dimmwerten (z. B. Helligkeit: 50%) musste das Szenario um den Datenpunkt-Typ 5.001 ergänzt werden, da dieser ursprünglich nur zur Annahme von relativen Dimmwerten (beispielsweise 12% heller) eingerichtet war.

Bei der Betrachtung der Datenpunkt-Typen liegt besonderes Augenmerk auf die Länge der Daten, die an die Gruppenadressen gesendet werden. Sind diese länger als 6 Bits, muss das Programm `groupwrite` verwendet werden, bei kürzeren Daten kommt `groupswrite` zum Einsatz.

Im Code-Listing 3.9 ist der Aufruf beispielhaft für `groupswrite` und dem Einschalten des Lichtbandes 1 dargestellt. Wenn der übergebene Post-Parameter „Lichtband1_aus“ ist, wird der Systembefehl `groupswrite` mit dem Parameter zum eibd (127.0.0.1), der Gruppenadresse (1/1/1) und dem zu schreibenden Wert (0) ausgeführt.

```
1 if ($_ eq "Lichtband1_aus")
2 {
3   system("groupswrite","ip:127.0.0.1","1/1/1","0") == 0 or
      die "system @args failed: $?"; }
```

Listing 3.9: Versand eines Steuerungsbefehls an eine Gruppenadresse

Abb. 3.7: Aufbau der Testumgebung im EIB/KNX-Labor

Die in Abbildung 3.7 dargestellte Steuerung der EIB/KNX-Komponenten kann in Nagios keinem Aktor direkt zugeordnet werden, da ein Aktor auf mehreren Wegen gesteuert werden kann. Lichtband 3 kann beispielsweise sowohl an- und ausgeschaltet, als auch mit relativen und absoluten Werten gedimmt werden. So wurde die Integration über eine separate Weboberfläche realisiert, die jeweils den beteiligten Aktoren oder auch Sensoren hinterlegt werden kann (siehe Abbildung 3.8). Das Einbinden der Website erfolgt über die Host-Konfiguration mit dem Parameter `action_url` und dem relativen Pfad zum Webserver (`/docs/eibknx_controls.htm`).

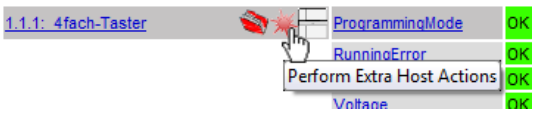

Abb. 3.8: Integration der Steuerung in die Nagios Weboberfläche

4 Untersuchung

In diesem Kapitel erfolgt die Untersuchung der entwickelten Nagios-Plugins und der Steuerungsfunktion. Die detaillierten Messergebnisse beider Entwicklungen finden sich vollständig auf der beiliegenden CD.

4.1 Überwachungsplugins

Dieses Kapitel beschreibt die ausgewählten Testmethoden zur Überprüfung der festgelegten Testkriterien für die fünf entwickelten Plugins. In einem Soll-Ist-Vergleich werden für jedes Testkriterium die ermittelten Werte den Soll-Werten aus Teilkapitel 3.1.4 gegenübergestellt.

4.1.1 Geräteunterstützung

Testmethode

Alle zur Verfügung stehenden Geräte werden in Nagios mit den entwickelten Plugins überwacht. Das Plugin zur Überprüfung der Maskenversion wird dabei als Host-Check konfiguriert, alle anderen als Service-Checks. Zur Überprüfung der korrekten Anzeige des abgefragten Status werden die Daten mit der Diagnose-Funktion des ETS3 verglichen. Außerdem sollen alle Zustandsanzeigen aller Geräte nach der Programmierung den Normalzustand „OK“ anzeigen.

Soll-Ist-Vergleich

Die Unterstützung der 13 getesteten Geräte lag bei elf Geräten, die fehlerfrei ausgelesen werden konnten. Lediglich die Anzeige-/Bedieneinheit (Infodisplay) der Firma Siemens und die analoge Sensorschnittstelle des Herstellers Gira zeigten nach fertiggestellter Programmierung Ausführungsfehler und einen fehlerhaften Ausführungszustand an (siehe Anhang A.3). Das Auslesen der Werte über die ETS3-Software lieferte die gleichen Werte. Die Fehlerregister des Infodisplays konnten manuell zurückgesetzt werden. Danach funktionierte das Auslesen des Wertes wie erwartet. Dieser Vorgang misslang jedoch bei der Sensorschnittstelle.

Zusätzlich konnte die Busspannung an der Sensorschnittstelle nicht ermittelt werden. Die ausgelesenen Werte der Plugins stimmen mit den Werten

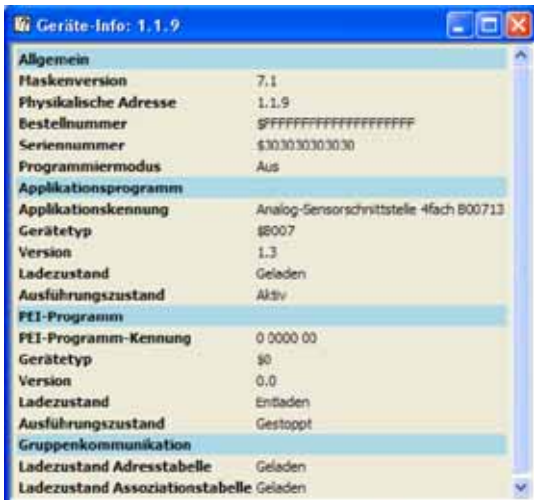

Abb. 4.1: Parameter der analogen Sensorschnittstelle

der ETS3-Diagnosefunktion überein. Da die analoge Sensorschnittstelle trotz angezeigter Fehler funktionierte und die Diagnosefunktion noch andere ungewöhnliche Werte (unter anderen Seriennummer und Bestellnummer) des Gerätes lieferte, kann von einem defekten bzw. überschriebenen Speicherbereich ausgegangen werden (siehe Abbildung 4.1). Der Fehler ist demnach nicht dem Plugin zuzuschreiben, sondern lässt vielmehr ein defektes Gerät vermuten.

Abschließend konnte festgestellt werden, dass alle zur Verfügung stehenden Geräte unterstützt werden und alle Plugins die Geräte korrekt auslesen.

4.1.2 Richtigkeit der Telegramme

Testmethode

Zur Überprüfung der Richtigkeit der Telegramme werden diese mit dem Busmonitor der ETS3 mitgelesen und analysiert. Der Aufruf jedes Plugins erfolgt manuell, wobei die erzeugten Telegramme aufgezeichnet werden.

Soll-Ist-Vergleich

Für alle fünf Plugins konnten nach deren Ausführung die generierten Pakete aufgezeichnet und deren Richtigkeit festgestellt werden. Der Busmonitor decodiert die Telegramme und liefert die übertragenen Werte als Detailinfo zu jedem Paket. Für das Plugin zur Überprüfung der Busspannung stellt sich dies wie in Abbildung 4.2 beispielhaft dar.

Abb. 4.2: Telegramme bei Abfrage der Busspannung

4.1.3 Zuverlässigkeit

Testmethode

Für jedes Plugin werden 1000 Abfragen an ein Gerät versendet und deren Rückgabewerte aufgezeichnet. Die Erfolgsquote ermittelt sich aus der Anzahl der Abfragen und den erfolgreichen Rückgabewerten.

Soll-Ist-Vergleich

Bei den Messungen kam es zu den in Tabelle 4.1 dargestellten Ergebnissen.

Plugin	**Anzahl der Pluginaufrufe**	**davon erfolgreich**	**Erfolgsquote**
check_voltage.pl	1000	999	99.9%
check_programmingmode.pl	1000	1000	100%
check_runningerror.pl	1000	1000	100%
check_runningstate.pl	1000	1000	100%
check_maskversion.pl	1000	1000	100%

Tab. 4.1: Zuverlässigkeit der Nagios-Plugins

Beim Plugin zur Überprüfung der Busspannung kam es bei einer Abfrage zu einem Abbruch. Der Aufruf wurde mit der Meldung „VOLTAGE CRITICAL - unreadable voltage“ beendet. Alle anderen Plugins erfüllten die Abfragen mit einer Erfolgsquote von 100% und halten damit die Soll-Kriterien ein.

4.1.4 Entwicklungsrichtlinien

Testmethode

Aus den Nagios plug-in development guidelines [Tea09] wird eine Checkliste mit allen anwendbaren Kriterien erzeugt und auf jedes Plugin angewendet. Es erfolgt eine Überpfrüfung, ob die einzelnen Punkte eingehalten werden.

Soll-Ist-Vergleich

Bei der Überprüfung der Entwicklungsrichtlinien für Nagios-Plugins konnten alle Vorgaben eingehalten werden. Auf den Einsatz des laut Richtlinie zu verwendenden Perl-Moduls `utils.pm` für die Validierung von Eingaben, Ausgaben von Rückgabewerten und Timeoutroutinen wurde jedoch verzichtet. Im Perl-Modul selbst findet sich die Anmerkung, dass es veraltet ist und stattdessen das `Nagios::Plugin`-Modul von Comprehensive Perl Archive Network (CPAN), einem Online-Repository für Perl-Module, verwendet werden soll. Dieser Empfehlung wurde Folge geleistet. Die Richtlinien konnten daher vollständig eingehalten werden (siehe Anhang A.8).

4.1.5 Abfragezeit

Testmethode

In Verbindung mit der Zuverlässigkeit werden für die 1000 Abfragen die Ausführungszeiten durch das Linux-Programm `time` festgestellt und protokolliert. Das Programm liefert als Rückgabewert die tatsächliche Ausführungszeit bis zur Beendigung des Programms.

Soll-Ist-Vergleich

Das verwendete Shell-Script zum Testen der Abfragezeit (siehe Anhang A.10) ermittelte nach 1000 Aufrufen je Plugin die in Tabelle 4.2 gelisteten Zeiten.

Plugin	**Minimale Abfragezeit**	**Maximale Abfragezeit**	**Mittlere Abfragezeit**
check_voltage.pl	250ms	840ms	370ms
check_programmingmode.pl	240ms	430ms	280ms
check_runningerror.pl	330ms	460ms	370ms
check_runningstate.pl	350ms	490ms	370ms
check_maskversion.pl	320ms	510ms	370ms

Tab. 4.2: Ausführungszeiten der Nagios-Plugins

Da die Plugins die gleiche Struktur besitzen, sind ähnliche Laufzeiten erwartungsgemäß im Mittel eingetreten. Die Laufzeiten liegen bei den ermittelten Maximalwerten und dem Mittelwert der Testergebnisse weit unterhalb der 10 Sekunden des Standard-Timeouts.

4.1.6 Busauslastung

Testmethode

Nach der Einbindung aller Plugins für alle Geräte des Testaufbaus werden zehn Minuten lang sämtliche Telegramme aufgezeichnet. Die Analysefunktion des Busmonitors der ETS3 ermittelt daraus die mittlere und die maximale Auslastung des Busses während der Aufzeichnung. Innerhalb der Testzeit und einem Überprüfungsintervall von fünf Minuten fragt jedes Plugin mindestens zweimal die zu ermittelden Kennwerte ab.

Soll-Ist-Vergleich

Die vier Nagios-Plugins (nur Service-Checks ohne den Host-Check) der 13 eingebundenen Geräte verursachten innerhalb von zehn Minuten 1164 Telegramme. Die maximale Auslastung des Busses lag während dieser Zeit bei 69,77% und der Mittelwert betrug 4,50% (siehe Abbildung 4.3). Die gemessene Buslast liegt damit innerhalb des festgelegten Sollwerts.

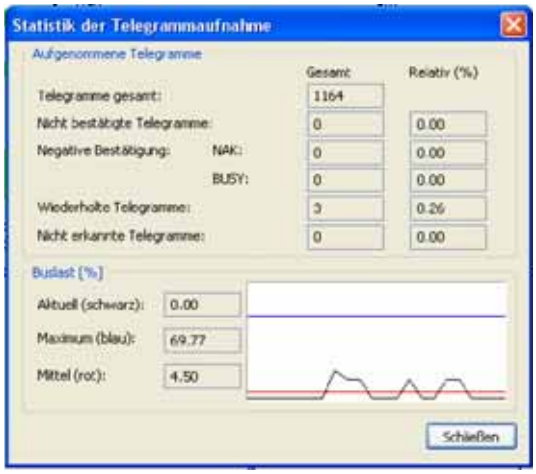

Abb. 4.3: Busauslastung durch Nagios-Plugins

4.2 Steuerungsfunktion

4.2.1 Geräteunterstützung

Testmethode

Nachdem ein Steuerbefehl über die Weboberfläche abgesetzt wurde, wird die Reaktion der Aktoren beobachtet und mit der zu erwartenden Reaktion verglichen (Blackbox-Test).

Soll-Ist-Vergleich

Alle schaltbaren Akoren (Lichtbänder, Jalousie, Heizventil) konnten durch die Weboberfläche der Steuerung angesprochen und gesteuert werden. Dabei wird auch die Zentralfunktion „Alle Lichtbänder an/aus“ unterstützt.

4.2.2 Richtigkeit der Telegramme

Testmethode

Die abgesetzten Telegramme zur Steuerung werden über die ETS3-Software bzw. mit dessen Busmonitor aufgezeichnet und mit den zu erwareten Telegrammdaten verglichen. Die Telegramme müssen hierbei identisch sein.

Soll-Ist-Vergleich

Nachdem alle Funktionen der Reihe nach auf der Weboberfäche (siehe Abbildung 3.7) betätigt wurden, zeichnete der Busmonitor die generierten Telegramme ebenfalls der Reihe nach auf (siehe Abbildung 4.4). Mithilfe der automatischen Decodierung der Telegramme durch den Busmonitor konnte die Richtigkeit aller Telegramme direkt abgelesen und bestätigt werden.

#	Zeit	Quelladr.	Quelle	Zieladr.	Ziel	R	Typ	Daten
1	17:19:19.953	1.1.55	IP-Router	1/1/0	Leuchtband1 E/A	7	Write	$01
3	17:19:21.421	1.1.55	IP-Router	1/1/0	Leuchtband1 E/A	7	Write	$00
4	17:19:23.265	1.1.55	IP-Router	1/1/1	Leuchtband2 E/A	7	Write	$01
5	17:19:24.734	1.1.55	IP-Router	1/1/1	Leuchtband2 E/A	7	Write	$00
7	17:19:26.437	1.1.55	IP-Router	1/1/3	Leuchtband3 E/A	7	Write	$01
9	17:19:28.703	1.1.55	IP-Router	1/1/5	Leuchtband3 Dimmwert	7	Write	$00
12	17:19:35.500	1.1.55	IP-Router	1/1/5	Leuchtband3 Dimmwert	7	Write	$7F
13	17:19:37.875	1.1.55	IP-Router	1/1/4	Leuchtband3 Dimmen	7	Write	
15	17:19:39.531	1.1.55	IP-Router	1/1/4	Leuchtband3 Dimmen	7	Write	
16	17:19:41.437	1.1.55	IP-Router	3/0/1	Jalousie Auf/Ab	7	Write	$00
18	17:19:42.968	1.1.55	IP-Router	3/0/1	Jalousie Auf/Ab	7	Write	$01
19	17:19:44.937	1.1.55	IP-Router	3/0/2	Jalousie Lamelle	7	Write	$00
21	17:19:46.671	1.1.55	IP-Router	3/0/2	Jalousie Lamelle	7	Write	$01
22	17:19:48.578	1.1.55	IP-Router	4/0/0	Stellantrieb	7	Write	$FF
24	17:19:50.515	1.1.55	IP-Router	4/0/0	Stellantrieb	7	Write	$00
26	17:19:54.453	1.1.55	IP-Router	4/0/0	Stellantrieb	7	Write	$7F
28	17:19:56.625	1.1.55	IP-Router	2/0/0	Beleuchtung Aus	7	Write	$01
29	17:19:58.421	1.1.55	IP-Router	2/0/0	Beleuchtung Aus	7	Write	$00

Abb. 4.4: Telegramme der Steuerungsfunktion

Die Ursache für die fehlenden Werte in der Darstellung für Telegramm 13 und 15 (relatives Dimmen) konnte nicht näher bestimmt werden. In der Detailansicht des entsprechenden Telegramms sind die Werte jedoch vorhanden und korrekt (siehe Anhang A.9).

4.2.3 Zuverlässigkeit

Testmethode

Um die Zuverlässigkeit zu testen, werden 200 Steuerbefehle abgesetzt und durch den Busmonitor der ETS3 aufgezeichnet. Aus der Anzahl der abgesetzten Befehle und der erfolgreich aufgezeichneten Befehle wird eine Erfolgsquote ermittelt.

Soll-Ist-Vergleich

Bis auf ein Übertragungstelegramm konnten alle anderen durch den Busmonitor aufgezeichnet werden. Die Zuverlässigkeitsquote liegt daher bei 99,5%.

4.2.4 Versandzeit

Testmethode

Die Zeit, die zum Versenden eines Steuerbefehls benötigt wird, wird durch Firebug, eine Erweiterung des Browsers Firefox, bestimmt und aufgezeichnet. Firebug ermittelt die Ladezeit der Bestätigungsseite, die nach der Ausführung des Steuerungsbefehls generiert wird. Zur Messung werden 200 Steuerbefehle nacheinander automatisiert abgegeben.

Soll-Ist-Vergleich

Die Ausführungszeit der Steuerung ist im Vergleich zur Überwachungsfunktion geringer, da nicht auf den Empfang oder die Empfangsbestätigung anderer Telegramme gewartet werden muss. Der fehlende Quittungsmechanismus sorgt für die in Tabelle 4.3 dargestellten durchschnittlichen Werte, die weit unterhalb des gesetzten Sollwerts liegen.

Kennzahl	**Ausführungszeit**
Maximalwert	79ms
Minimalwert	39ms
Mittelwert	53,64ms

Tab. 4.3: Ausführungszeiten der Steuerung

4.2.5 Gruppenadressarten

Testmethode

Der Aufruf der Steuerung erfolgt zuerst als 3-stufige Adressparameterübergabe und danach als gleichwertige 2-stufige Parameterübergabe (siehe Abbildung 2.8). Die Reaktion bzw. das erzeugte Telegramm müssen identisch sein.

Soll-Ist-Vergleich

Das Ändern der verwendeten Gruppenadresse von 3-stufig in das 2-stufige Pendant zeigte im Schaltverhalten der Aktoren keinen Unterschied. Die Programme des BCU-SDKs `groupwrite` und `groupswrite` erzeugen exakt die gleichen Telegramme. Es werden also beide Gruppenadressierungsarten unterstützt.

4.3 Ergebnisse der Untersuchung

Die Testergebnisse für die Nagios-Plugins sind in Tabelle 4.4 abgebildet. Alle Testkriterien konnten in einem Soll-Ist-Vergleich eingehalten werden.

	BS*	**PM***	**AF***	**AZ***	**MV***	**Sollwert**	**Wichtung**
Geräte-Unterstützung	13/13	13/13	13/13	13/13	13/13	eingehalten	25%
Richtigkeit der Telegramme	korrekt	korrekt	korrekt	korrekt	korrekt	eingehalten	25%
Zuverlässigkeit	99,9%	100%	100%	100%	100%	eingehalten	15%
Entwicklungs-richtlinien	17/17	17/17	17/17	17/17	17/17	eingehalten	10%
Maximale Abfragezeit	840ms	430ms	460ms	490ms	510ms	eingehalten	10%
Busauslastung (Max/Mittel)	69,77% / 4,50%					eingehalten	10%
Gesamt						**6/6**	**100%**

Tab. 4.4: Zusammenfassung Testergebnisse Überwachungsplugins

*Legende:
BS = Busspannung, PM = Progammiermodus, AF = Ausführungsfehler
AZ = Ausführungszustand, MV = Maskenversion

Die Ergebnisse der Steuerung der EIB/KNX-Komponenten zeigt Tabelle 4.5. Die aufgestellten Sollwerte sind nach Anwendung der Testmethoden eingehalten worden.

	Steuerungsfunktion	**Sollwert**	**Wichtung**
Geräteunterstützung	Alle Aktoren	eingehalten	40%
Richtigkeit der Telegramme	korrekt	eingehalten	20%
Zuverlässigkeit	99,5%	eingehalten	20%
Versandzeit (maximal)	79ms	eingehalten	15%
Gruppenadressen	2-/3-stufig	eingehalten	5%
Gesamt		**5/5**	**100**%

Tab. 4.5: Zusammenfassung der Testergebnisse Steuerungsfunktion

5 Zusammenfassung

In diesem letzten Abschnitt werden die Ergebnisse der Ausarbeitung vorgestellt. Außerdem wird ein Ausblick auf Weiterentwicklungen der Überwachungsplugins und der Steuerungsfunktion gegeben.

5.1 Ergebnisse

Die Implementierung der Nagios-Plugins ermöglicht es erstmals, Gebäudekomponenten auf Basis von EIB/KNX stetig zu überwachen. Durch die Integration in die Netzwerk-Management-Software Nagios kann zudem ein ausgefeiltes und weit verbreitetes Überwachungsframework genutzt werden. Somit ist es möglich, durch das Eskalationsmanagement von Nagios Benachrichtigungen zu verschicken und dadurch Ausfallzeiten zu verringern oder gar zu vermeiden. Darüber hinaus kann das Monitoring mit der Überwachung von Servern und Netzwerkkomponenten vereint in einer einzigen Oberfläche erfolgen. Als Voraussetzung für die Entwicklung konnte die Frage beantwortet werden, welche Kennwerte die Funktionsfähigkeit eines EIB/KNX-Gerätes repräsentieren.

Die vorgestellten Überwachungsplugins eignen sich aufgrund der Messergebnisse für die Anwendung in der Praxis, da alle Plugins zuverlässig, schnell und korrekt arbeiten. Die Entwicklung konnte ebenfalls unter Berücksichtigung der vorgegebenen Entwicklungsrichtlinien erfolgreich abgeschlossen werden, was den Einsatz und die Verbreitung zusätzlich fördert. Als Schwachstelle kann sich in Umgebungen mit vielen EIB/KNX-Geräten jedoch die Buslast erweisen. Hat sie in der Laborumgebung mit 13 überwachten Geräten noch eine mittlere Buslast von knapp 5% erzeugt, so kann dieser Wert bei Gebäuden mit mehreren hundert Geräten den Bus komplett auslasten bzw. an seine Leistungsgrenzen führen. Abhilfe könnte hier jedoch die Herabsetzung der Prüfintervalle von Nagios schaffen.

Die Integration der Steuerung ermöglicht es dem Benutzer, aus der Nagios-Weboberfläche die Webseite zur rudimentären Bedienung von Aktoren aufzurufen. Eine Steuerung kann auch ohne eine bestehende Nagios-Installation erfolgen. Die Untersuchung der Entwicklung hat gezeigt, dass die EIB/KNX-Aktoren zuverlässig und korrekt angesteuert werden können.

In der Arbeit konnte gezeigt werden, dass eine Überwachung und Steuerung herstellerunabhängig möglich ist. Außerdem konnte bei der Entwicklung auf kommerzielle Softwareprodukte und proprietäre Programmiersprachen ver-

zichtet und auf quelloffene Software zurückgegriffen werden. Die Überwachung und Steuerung von EIB/KNX-Komponenten mittels Netzwerk-Management-Software konnte abschließend erfolgreich realisiert werden.

5.2 Ausblick

Für einen Einsatz der Nagios-Plugins in der Praxis, beispielsweise in einem Rechenzentrum oder in einer großen Niederlassung, sollte die Übersichtlichkeit der Darstellung erhöht werden. Eine große Anzahl an zu überwachenden EIB/KNX-Komponenten mit jeweils vier überwachten Werten lässt die Liste mit Zuständen in der Nagios-Oberfläche schnell umfangreich und unübersichtlich werden. Besonders wenn neben dem Monitoring von EIB/KNX-Komponenten noch andere Server, Switches und Router überwacht werden müssen. Abhilfe schafft hier die Integration des Nagios-Plugins `multi_check`, das die Ergebnisse der vier Überprüfungsroutinen als einen Rückgabewert zusammenfasst (siehe Abbildung 5.1).

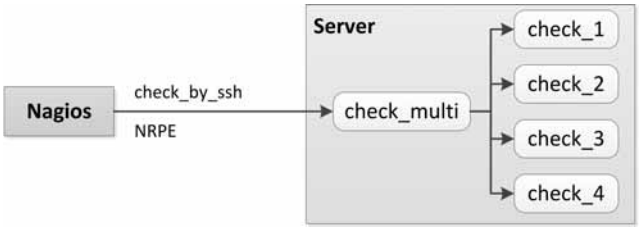

Abb. 5.1: Zusammenfassung von Checks mit check_multi [Bar09]

Ebenfalls erhöht die Integration eines Gebäudeplans die Zuordnung von Aktoren und Sensoren zu den realen Geräten. Hierfür empfiehlt es sich, die Erweiterung NagVis zu installieren, die die Hinterlegung eines Planes (als sog. Map) ermöglicht [Sch09a]. Bei Problemen mit EIB/KNX-Geräten können die fehlerhaften Geräte schnell lokalisiert und ausgetauscht werden.

Als nächsten Entwicklungsschritt für den Praxiseinsatz ist die Abfrage von Temperatur- und Luftfeuchtigkeitswerten empfehlenswert. Auch hier sind Programme aus dem BCU SDK einsetzbar. Beispielsweise können die regelmäßig verschickten Telegramme eines Multisensors durch den Busmonitor aufgezeichnet, durch ein Perl-Script ausgewertet und an den Nagios-Server versendet werden.

Auch die Steuerungsfunktion kann in Sachen Benutzerkomfort weiterentwickelt werden, z. B. durch die Hinterlegung eines Raumplanes und der Einzeichnung der Aktoren und Sensoren. So könnte der Benutzer Piktogramme von Schalter oder dem zu steuernden Licht anklicken und die Bedienung intuitiver und übersichtlicher gestalten.

Literaturverzeichnis

[Ass08] Association, Konnex: *Standardisation*. http://www.knx.org/knx-standard/standardisation/. Version: August 2008. – Seitenaufruf: 29. Juli 2010

[Ass10] Association, Konnex: *System Specifications-Interworking-Datapoint Types*. http://www.knx.org/fileadmin/downloads/03%20-%20KNX%20Standard/KNX%20Standard%20Public%20Documents/03_07_02%20Datapoint%20Types%20v1.5.00%20AS.zip. Version: April 2010

[Bar05] Barth, Wolfgang: *Nagios. System- und Netzwerkmonitoring*. 1. Ausgabe. Open Source Press, 2005. – ISBN 9783937514918

[Bar09] Barth, Wolfgang: *Nagios. System- und Netzwerkmonitoring*. 2. Jubiläumsausgabe. Open Source Press, 2009. – ISBN 9783937514918

[BSI09] BSI: *IT-Grundschutz-Kataloge*. 11. Ausgabe. Bundesamt für Sicherheit in der Informationstechnologie, 2009 http://www.bsi.bund.de/grundschutz

[BSI10] BSI: *FAQ zu Open Source Software*. https://www.bsi-fuer-buerger.de/cln_174/BSIFB/DE/Themen/OpenSourceSoftware/FragenUndAntworten/fragenundantworten_node.html. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Cis01] Cisco: *Handbuch Netzwerk-Technologien. Komplettes Grundwissen zu Networking und Internetworking*. 1. Ausgabe. Markt+Technik, 2001. – ISBN 9783827260802

[Deu10] Deutschland, KNX: *KNX - eine wachsende Erfolgsstory*. http://www.knx.de/presse/index.php?ID=51. Version: April 2010. – Seitenaufruf: 29. Juli 2010

[Gal10] Galstad, Ethan: *Nagios Overview*. http://www.nagios.org/about/overview. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Hag08] Hagen, Constantin: *Netzwerk-, System- und Applikationsmonitoring für die EDV der DR. JOHANNES HEIDENHAIN GmbH*, Fachhochschule Salzburg GmbH, Diplomarbeit, Juni 2008

[Hei02] Heinrich, Lutz J.: *Informationsmanagement - Planung, Überwachung und Steuerung der Informationsinfrastruktur*. 7. Ausgabe. Oldenbourg, 2002. – ISBN 9783486258424

[Hel09] Heller, Martin: *Installation der NDO Utils*. `http://www.nagios-wiki.de/nagios/ndo/installation`. Version: Juni 2009. – Seitenaufruf: 29. Juli 2010

[Hin07] Hintemann, Dr. R.: *Hochverfügbarkeit - IT-Ausfälle vermeiden*. `http://www.securitymanager.de/magazin/artikel_1281_it_ausfaelle_vermeiden.html`. Version: Januar 2007. – Seitenaufruf: 29. Juli 2010

[Kau98] Kauffels, Franz-Joachim: *Netzwerk- und System-Management. Probleme, Standards, Strategien*. Datacom Vlg., Bergheim, 1998. – ISBN 9783892381259

[Kög08] Kögler, Martin: *Free Development Environment for Bus Coupling Units of the European Installation Bus*. `http://www.auto.tuwien.ac.at/~mkoegler/eib/sdkdoc-0.0.4.pdf`. Version: Dezember 2008. – Seitenaufruf: 29. Juli 2010

[Kög10] Kögler, Martin: *EIBD*. `http://www.auto.tuwien.ac.at/~mkoegler/index.php/eibd`. Version: 2010. – Seitenaufruf: 29. Juli 2010

[KR02] Kurose, James ; Ross, Keith: *Computernetze. Ein Top-Down-Ansatz mit Schwerpunkt Internet*. 1. Ausgabe. Pearson Studium, 2002. – ISBN 9783827370174

[Mey04] Meyer, Willi: *EIB Tool Software. Das Praxisbuch für ETS 3 Professional, ETS 3, Starter und ETS 2*. 1. Ausgabe. Hüthig & Pflaum, 2004. – ISBN 9783810102126

[MHH09] Merz, Herrmann ; Hansemann, Thomas ; Hübner, Christof: *Gebäudeautomation. Kommunikationssysteme mit EIB/KNX, LON und BACnet*. 2. Ausgabe. Hanser Fachbuch, 2009. – ISBN 9783446421523

[Nag09] Nagios: *Nagios Core Version 3.x Documentation*. `http://nagios.sourceforge.net/docs/3_0/`. Version: 2009. – Seitenaufruf: 29. Juli 2010

[Ope10] OpenNMS: *Official Documentation*. http://www.opennms.org/wiki/Docu-overview. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Per10] Perl.org: *White papers: Technical Showcases*. http://www.perl.org/about.html. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Rie06] Rieger, Götz: Netzwerk unter Kontrolle. In: *c't - Magazin für Computertechnik* 3 (2006), Februar, Nr. 3, S. 206

[RKR00] Rose, Michael ; Kriesel, Werner ; Rennefahrt, Jens: *EIB für die Gebäudesystemtechnik in Wohn- und Zweckbau*. 3. Ausgabe. Hüthig, 2000. – ISBN 9783778526439

[Sch04] Scherg, Rainer: *EIB planen, installieren und visualisieren (ETS 3)*. 1. Ausgabe. Vogel Verlag Und Druck, 2004. – ISBN 9783802319600

[Sch09a] Scherbaum, Tobias: *Praxisbuch Nagios*. 1. Ausgabe. O'Reilly, 2009. – ISBN 9783897218802

[Sch09b] Scherff, Alfred: *Eibd*. http://knx-user-forum.de/lexikon/eibd-84/knx-eib-1.html. Version: 2009. – Seitenaufruf: 29. Juli 2010

[Sie05a] Siemens AG (Hrsg.): *BCU 1 Help*. Siemens AG, September 2005. http://www.auto.tuwien.ac.at/~mkoegler/eib/doc/bcu1help.pdf. – Seitenaufruf: 29. Juli 2010

[Sie05b] Siemens AG (Hrsg.): *BCU 2 Help*. Siemens AG, September 2005. http://www.auto.tuwien.ac.at/~mkoegler/eib/doc/Bcu2Help_v12.chm. – Seitenaufruf: 29. Juli 2010

[Tea09] Team, Nagios Plugins D.: *Nagios plug-in development guidelines*. http://nagiosplug.sourceforge.net/developer-guidelines.html. Version: 2009. – Seitenaufruf: 29. Juli 2010

[TT09] TEIA-Team: *WebSite-Administration & Grundlagen Apache*. http://bit.ly/a8Wv2J. Version: 2009. – Seitenaufruf: 29. Juli 2010

[www10a] www: *Big Sister System and Network Monitor*. http://www.bigsister.ch. Version: 2010. – Seitenaufruf: 29. Juli 2010

[www10b] www: *The Hobbit Monitor*. http://hobbitmon.sourceforge.net/. Version: 2010. – Seitenaufruf: 29. Juli 2010

[www10c] WWW: *Tobi Oetiker's MRTG - The Multi Router Traffic Grapher*. `http://oss.oetiker.ch/mrtg/`. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Zab10] ZABBIX: *Zabbix 1.8 Manual*. `http://www.zabbix.com/documentation/1.8/complete`. Version: 2010. – Seitenaufruf: 29. Juli 2010

[Zen10] ZENOSS: *Zenoss Features*. `http://www.zenoss.com/product/network-management`. Version: 2010. – Seitenaufruf: 29. Juli 2010

A Anhang

A.1 Datenpunkt-Typen

Haupttyp	Name	Bedeutung	Länge
1.xxx	DPT B_1	1 Bit Wert	1 Bit
2.xxx	DPT B_2	1 Bit Control, 1 Bit Wert	2 Bit
3.xxx	DPT B_1U_3	1 Bit Control, 3 Bit vorzeichenloser Wert	4 Bit
4.xxx	DPT A_8	8 Bit Buchstabencode (z. B. ASCII)	8 Bit
5.xxx	DPT U_8	8 Bit vorzeichenloser Wert	8 Bit
6.xxx	DPT V_8	8 Bit Wert (mit Vorzeichen) als 2er Komplement	8 Bit
7.xxx	DPT U_{16}	16 Bit vorzeichenloser Wert	16 Bit
8.xxx	DPT V_{16}	16 Bit Wert (mit Vorzeichen) als 2er Komplement	16 Bit

Tab. A.1: Datenpunkt-Typen (Auszug) [Ass10]

DPT	Name	Länge	Codierung	Verwendung
1.001	DPT_Switch	1 Bit	0=Off, 1=On	Lichtband an/aus
5.001	DPT_Scaling	8 Bit	0x7F=50%	Lichtband 3 dimmen (absolut auf 50%)
3.007	DPT_Control_ Dimming	4 Bit	1100=12% heller 0100=12% dunkler	Lichtband 3 dimmen (relativ um 12 Prozent)
1.009	DPT_OpenClose	1 Bit	1=Open, 0=Close	Jalousie öffnen/schließen
1.008	DPT_UpDown	1 Bit	0=Up, 1=Down	Jalousie stufenweise hoch/runter
5.001	DPT_Scaling	8 Bit	0x7F=50%	Heizungsventil anfahren (absolut auf 50%)
1.001	DPT_Switch	1 Bit	0=Off, 1=On	Alle Lichtbänder an/aus

Tab. A.2: Verwendete Datenpunkt-Typen

A.2 Konfiguration und Aufruf der Dienste in Nagios

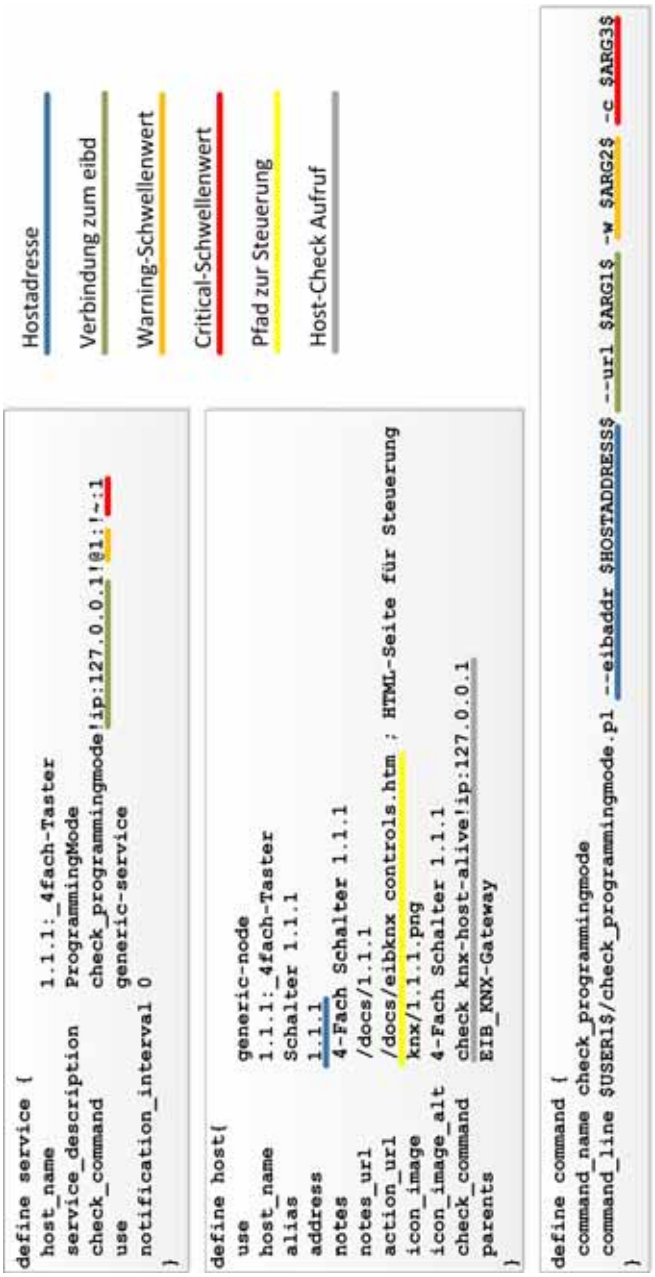

Abb. A.1: Nagios-Konfiguration und Aufruf der Services am Beispiel der Abfrage des Programmiermodus

A.3 Alle eingebundenen Plugins

Abb. A.2: Plugins unter Nagios

A.4 Komponentenliste

Kompon.	Bezeichnung	Applikation	Phys. Adr.	Hersteller
Sensor	Taster 4-fach 5WG1 245-2AB_1	12 S4 Ein-Aus /Dimmen/Jalousie 241301	1.1.1	Siemens
Sensor	Taster 2-fach 5WG1 243-2AB_1	12 S2 Ein-Aus-Um /Dim/Jalo/Anzeige 221301	1.1.2	Siemens
Sensor	Taster 1-fach 5WG1 241-2AB_1	12 S1 Ein-Aus-Um /Dim/Jalo/Anzeige 211301	1.1.3	Siemens
Sensor	Anzeige-/Bedieneinheit 5WG1 585-2AB_1	01 07 Anzeige-/Bedieneinheit 801502	1.1.4	Siemens
Aktor	Binärausgang 5WG1 562-1AB01	11 A2 Binär 520401	1.1.5	Siemens
Aktor	Universaldimmer 5WG1 527-1AB02	21 A1 Universaldimmer 906703	1.1.6	Siemens
Sensor	Szenenbaustein 5WG1 300-1AB01	12 CO Szene 740601	1.1.7	Siemens
Sensor	Binäreingang 5WG1 263-1AB01	11 S6 Ein-Aus zyklisch 261902	1.1.8	Siemens
Aktor	Analog-Sensorschnittstelle 1021 00	Analog-Sensorschnittstelle 4fach B00713	1.1.9	Gira
Aktor	Jalousieaktor 6481 90	Jalousie.Rollo.Sicherheit. Position. Hand 5361/2.0	1.1.10	Merten
Sensor	Präsenz/Lichtregelung 6305 92	Regelung / Bewegung oder Alarm 1340/2	1.1.11	Merten
Sensor	Raumtemperaturregler 5WG1 252-2AB_3	12 S1 Temperaturregelung 210B04	1.1.12	Siemens
Aktor	Elektro. Stellantrieb 731 9 200	Cheops drive V1.1	1.1.13	Theben
Komm.	IP-Router 5WG1 146-1AB01	IP-Router 001001	1.1.55	Siemens
Komm.	RS-232-Schnittstelle 5WG1 148-1AB04	10 CO Dummy 700002	1.1.64	Siemens

Tab. A.3: Komponentenliste

A.5 Programmablaufplan

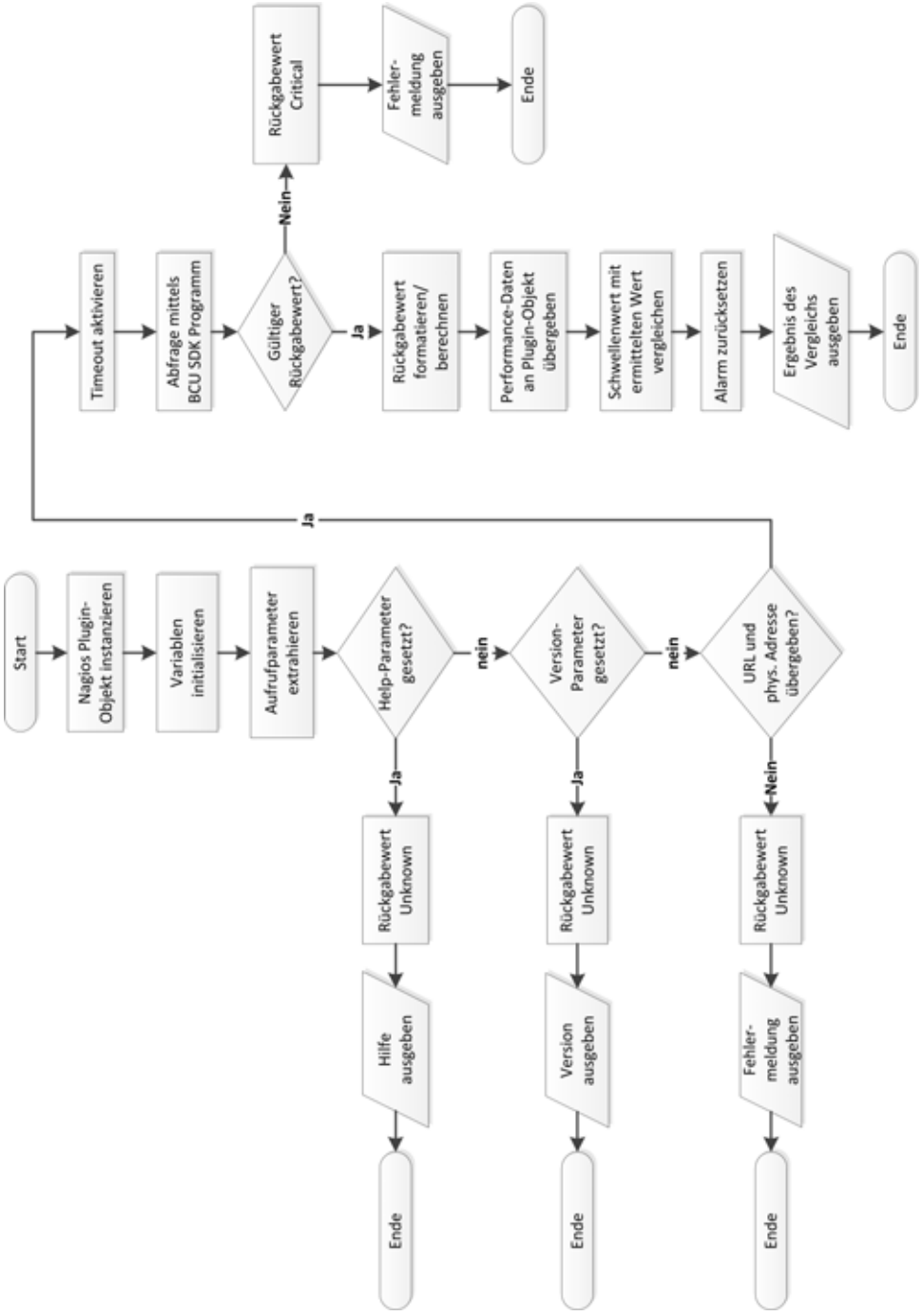

Abb. A.3: Programmablaufplan eines Plugins

A.6 Testaufbau im EIB/KNX-Labor

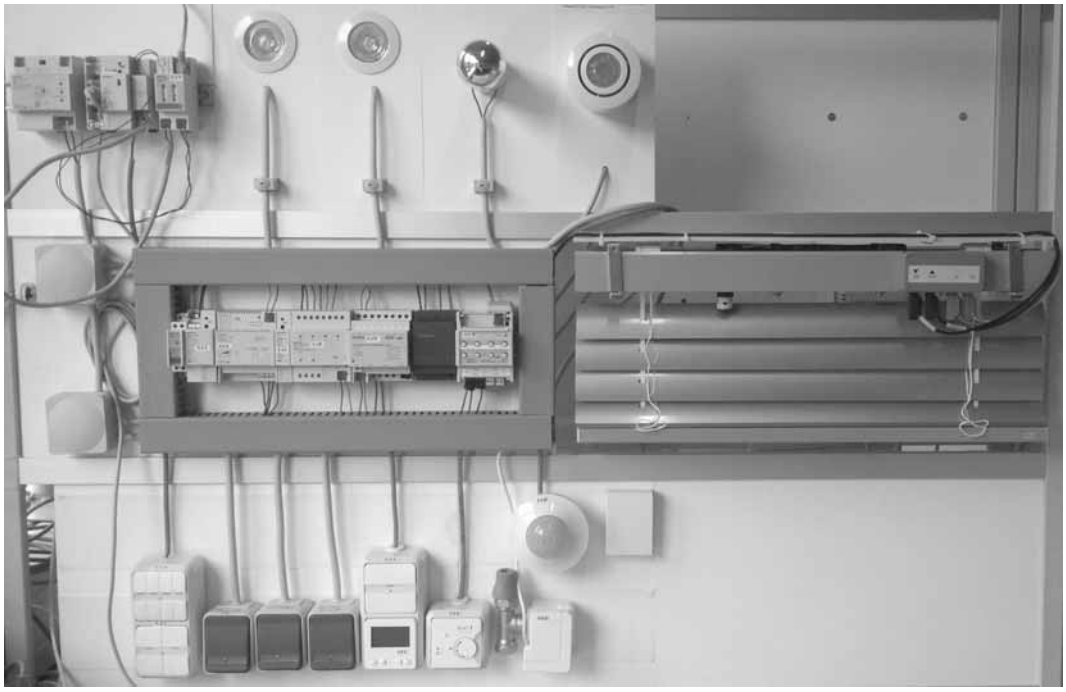

Abb. A.4: Aufbau der Testumgebung im EIB/KNX-Labor

A.7 Konfigurationsverzeichnisse Nagios

Verzeichnis	Beschreibung
/usr/sbin/nagios3	Nagios Binary
/etc/nagios3	Konfiguration Nagios
/etc/nagios-plugins/config	Konfiguration Nagios-Plugins
/var/log/nagios3/nagios.log	Logdatei Nagios
/usr/lib/nagios/plugins	Pluginverzeichnis
/var/www/	Root-Verzeichnis Apache Webserver
/usr/lib/cgi-bin/nagios3	Interne Nagios CGI-Scripte
/usr/share/nagios3/htdocs/images/logos/base	Icon-Verzeichnis

Tab. A.4: Konfigurationsverzeichnisse Nagios

A.8 Checkliste Nagios Plugin Development Guide

Eigenschaft	Busspannung	Ausfiihrungszustand	Ausfiihrungsfehler	Programmiermodus	Maskenversion
Einzeilige Ausgabe: CHECK_ART STATUS - Information text	✓	✓	✓	✓	✓
Ein- bis mehrfache Verbose- und Debug-Ausgabe: Bis zu drei Level (-vvv bzw. -ddd)	✓	✓	✓	✓	✓
Hilfe-Funktion aufrufbar: Mit Parameter -h|–help	✓	✓	✓	✓	✓
Lesbarkeit des Codes und der Ausgabe: 80x25 Zeichen	✓	✓	✓	✓	✓
Rückgabewete: 0,1,2,3 für OK,Warning,Critical,Unknown	✓	✓	✓	✓	✓
Angabe von Schwellenwerte und -bereich: Alle Formate unterstützt	✓	✓	✓	✓	✓
Performace Daten: Format/Einheiten einhalten und angeben	✓	✓	✓	✓	✓
Übersetzung: Englisch in Variablen, Kommentaren, Hilfe und Ausgabe	✓	✓	✓	✓	✓
Systemkommandos: Programme mit vollem Pfad aufrufen	✓	✓	✓	✓	✓
Temporäre Dateien und symbolische Links: Verwendung vermeiden	✓	✓	✓	✓	✓
BEGIN-END-Blöcke: Nicht verwenden	✓	✓	✓	✓	✓
Variablen: Alle Variablen deklarieren	✓	✓	✓	✓	✓
DATA-Handles und Globale Variable: Nicht verwenden	✓	✓	✓	✓	✓
Timeouts: Bei Netzzugriffen verwenden	✓	✓	✓	✓	✓
Perl-Modul: Verwenden von Getopt::Long	✓	✓	✓	✓	✓
Reservierte Optionsparameter: Reservierte Optionen einhalten	✓	✓	✓	✓	✓
Testfälle Testfälle mittels Perl-Modul Test::More	✓	✓	✓	✓	✓

Tab. A.5: Checkliste Nagios Plugin Development Guide

A.9 Detailansicht des Telegramms Dimmen

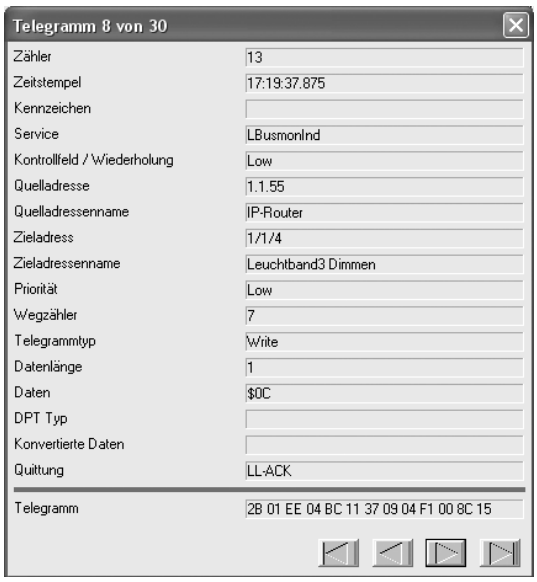

Abb. A.5: Detailansicht Telegramm Dimmen (12% heller)

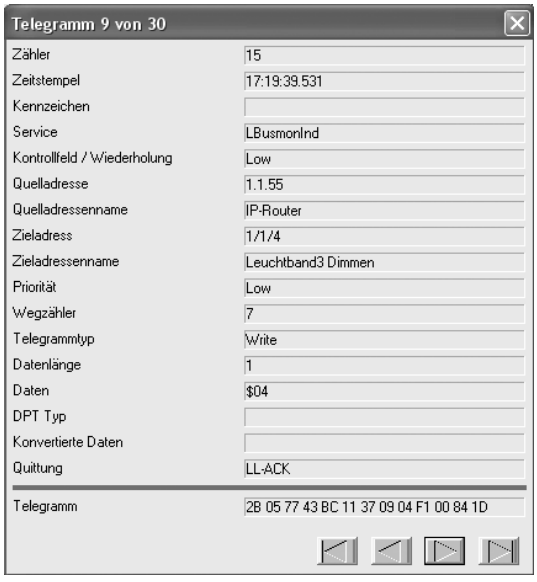

Abb. A.6: Detailansicht Telegramm Dimmen (12% dunkler)

A.10 Testprogramm testtime.sh

```
#!/bin/sh
#
# Beispielaufruf
# testtime.sh 1000 check_voltage.pl
#
if [ $# -eq 0 ]
then
echo "Error - Anzahl der Testmessungen fehlt"
echo "Syntax : $0 number"
exit 1
fi

# Der erste Paramter ist die Anzahl der Tests
# Der zweite Parameter ist Pluginname
progname=$2

for i in $(seq $1); do
  time -f  "%C %U %S %e %P" -a -o /tmp/testtime_$progname.
     csv ./$progname -U ip:127.0.0.1 -E 1.1.1 >> /tmp/
     output_$progname.txt
done
```

Listing A.1: Bash-Script testtime.sh

A.11 Dateistruktur der Nagioskonfiguration

```
|-- knx
|   |-- services
|       |-- voltage.cfg         Spannung
|       |-- runtimererror.cfg   Ausführungsfehler
|       |-- runtimestate.cfg    Ausführungszustand
|       `-- progmode.cfg        Programmiermodus
|   `-- nodes
|       |-- 1                   Bereich 1 => 1.x.x
|           |-- 1               Linie 1   => 1.1.x
|           |-- 2               Linie 2   => 1.2.x
|               |-- 1.cfg       Gerät 1   => 1.2.1
|               `-- 2.cfg       Gerät 2   => 1.2.2
|       `-- 2                   Bereich 2 => 2.x.x
```

Listing A.2: Dateistruktur der Nagioskonfiguration

Selbständigkeitserklärung

Erklärung

Ich erkläre, dass ich die vorliegende Bachelorthesis selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

_____ Erfurt, den _____