



Studiengang: Informationstechnik

Entwicklung eines externen Sensornetzes mit WLAN Kopplung und Visualisierung

STUDIENARBEIT
Im Rahmen des 5. und 6. Theoriesemesters

Abgabedatum 11.5.2015

Verfasser

Matrikelnummer

Kurs

Betreuer

Maik Maier, Nicolai Staegge

4050846, 4615051

TINF12B3

Herr Prof. Hans-Jörg Haubner

Erklärung

Gemäß §5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.

Ich habe die vorliegende Studienarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Karlsruhe, Datum

Unterschrift

Unterschrift

Inhaltsverzeichnis

1 Einleitung und Intention	1
1.1 Ziel dieser Arbeit	2
2 Internet of Things	3
2.1 Geschichte	4
2.2 Ziele und Anwendungsbeispiele	4
2.3 Hardware für Privatanwender	5
2.3.1 Arduino-Plattform	5
2.3.2 Raspberry Pi	8
2.4 Sicherheitsaspekte	11
2.4.1 Sicherheitsmodell nach Gordon Cooper	12
2.4.2 Verschlüsselungsstandard AES	14
3 Theoretische Grundlagen	16
3.1 Wireless Sensor Networks	16
3.1.1 Ubiquitäres Rechnen	16
3.1.2 Motivation von Sensornetzen	17
3.1.3 Bestandteile	19
3.1.4 Topologien	19
3.1.5 Schwierigkeiten	21
3.2 Adhoc-Netzwerke	24
3.2.1 MANETs	24
3.2.2 Routingprotokolle	26
3.3 IEEE 802.15.4	30
3.3.1 Komponenten	31
3.3.2 Unterstützte Topologien	32
3.3.3 Schichten	33
3.3.4 Rahmenstruktur	33
3.3.5 Kommunikation	34
3.3.6 Medienzugriff	35
3.3.7 Sicherheitsaspekte	36
3.4 SunSPOT	36
3.4.1 Gerätearten	37
3.4.2 Verfügbare Boards	37

4 Praktische Arbeiten	39
4.1 Erste Schritte	39
4.1.1 Installation	39
4.1.2 Übung 1 - Flashlight	40
4.2 Implementierung einer Raumüberwachung	41
4.2.1 Idee	41
4.2.2 Umsetzung	42
4.3 Erweiterungsmöglichkeiten	48
4.3.1 Visualisierung Webseite	48
4.3.2 Erweiterung durch verschiedene Sensoren	49
4.3.3 Benachrichtigung des Benutzers	50
5 Vergleichbare Projekte	54
5.1 Smart Greenhouse	54
5.2 Bot-So	56
6 Zusammenfassung und Fazit	58
Literaturverzeichnis	xiv

Abbildungsverzeichnis

2.1	Das Arduino-Logo [Arda]	6
2.2	Die Front der dritten Arduino Uno Revision [Ardc]	7
2.3	Das Logo der Raspberry Pi Foundation [Ras]	9
2.4	Die Oberseite des Raspberry Generation 2 B-Modells [Mul15]	9
2.5	Das Camera Modul für den Raspberry Pi [Con]	10
2.6	Sicherheitsmodell nach Gordon Cooper [Coo15]	12
3.1	Mikroprozessoren-Transistoren im Laufe der Zeit [Wgs]	18
3.2	Peer-To-Peer Netzwerk [Kos09]	20
3.3	Stern Netzwerk [Kos09]	20
3.4	Baum Netzwerk [Kos09]	21
3.5	Vermischtes Netzwerk [Kos09]	21
3.6	Einordnung von MANETs [TS11]	25
3.7	Mögliche Verbindungen eines Beispiel MANETs [TS11]	25
3.8	optimiertes LSR durch Multi-Point-Relays im OLSR [TS11]	28
3.9	'Route Discovery' im AODV-Protokoll [TS11]	29
3.10	Clustering im CGSR-Protokoll [TS11]	30
3.11	Topologien in IEEE 802.15.4 [Ins11]	32
3.12	Definierte Schichten in IEEE 802.15.4 [Ins11]	33
3.13	Superraumen-Struktur eines LR-WPANs [Ins11]	34
3.14	Framestruktur eines Data-Frames im MAC-Layer [Ins11]	35
3.15	Anatomie eines Standard SunSPOT-Sensors [Uni]	37
4.1	Leuchtende LEDs des 'Flashlight'-Programms	41
4.2	Einbruchsstatistik in Deutschland [Tru15]	41
4.3	Logdaten der Basisstation bei Servererstellung	44
4.4	JFrame mit textueller Ausgabe	45
4.5	Konsolenausgabe mit Timestamp	45
4.6	Mockup einer Webseite	48
4.7	Visualisierung mithilfe eines Raspberries und Webserver	49
4.8	Aufbau eines Sensornetzes mithilfe eines Raspberry Pis	50
4.9	Mockup einer Push-Benachrichtigung	51
4.10	Aufbau eines Benachrichtigungssystems	52
5.1	Das Smart Greenhouse Entwicklerteam [epa14]	55
5.2	Der Bot-So Roboter [Par14]	57

Tabellenverzeichnis

2.1 Auflistung der Kombinationsmöglichkeiten bei verschiedenen Schlüssellängen	15
4.1 Auflistung der mobil abrufbaren Kommunikationskanäle	50

Listings

2.1	Simpler Arduino-Code, der eine LED blinken lässt	7
4.1	Ausschnitt aus der setup()-Methode	42
4.2	Öffnen des serverseitigen Ports	43
4.3	Auswerten des empfangenen Paketes	44
4.4	Aufbau der Verbindung auf Port 67	46
4.5	Schleife zur Detektion von Bewegung	46
4.6	Methode zur Bewegungserkennung	47
4.7	Beispielcode zum Senden einer Pushover-Nachricht	52
1	Code HostStudi.java	viii
2	Code MovementDetection.java	xi

Abkürzungsverzeichnis

AES	Avanced Encryption Standard
AODV	Ad-hoc On-Demand Distance Vector
CGSR	Clusterhead Gateway Switch Routing
CPU	Central Processing Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FFD	Full Function Device
G	G-Kraft - Gewichtskraft
I/O	Input/Output
I²C	InterIntegrated Circuit
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoE	Internet of Everything
IoT	Internet of Things
LED	Licht-emittierende Diode
LR-WPAN	Low-rate Wireless Personal Area Network
M2M	Machine-to-Machine
mA	Milli-Ampere
MAC	Media Access Control
MANET	Mobile Ad-hoc Netzwerk
MHz	Megahertz
OLSR	Optimized Linke State Routing
PAN	Personal Area Network
RFD	Reduced Function Device
RGB	Rot, Grün und Blau
SPOT	Small Programmable Object Technology
SRAM	Static Random Access Memory
USART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

Kapitel 1

Einleitung und Intention

Der Computer ist mittlerweile zum festen Bestandteil im alltäglichen Leben geworden. Mit ihm können viele Aufgaben wie Recherchen, komplexe Rechnungen und Kommunikation vereinfacht und schnellstmöglich erledigt werden. Während vor einigen Jahren noch der Desktop-PC die beliebteste Wahl darstellte, geht der Trend mittlerweile in Richtung der mobilen Endgeräte wie z.B. Smartphone, Laptop oder auch Tablet. Menschen wollen sich nicht an einen Ort binden, an dem sie ihren Computer benutzen können und sehnen sich nach dem Wunsch, dass alle Alltagsgegenstände per Smartphone oder Tablet kontrollierbar werden.

Diese Vernetzung aller elektronischen Geräte in einem Haushalt wird als “Internet of Things” (kurz IoT) bezeichnet. Die grundsätzliche Idee besteht darin, dass alle elektronischen Geräte wie z.B. Kühlschrank, Backofen u.a. miteinander kommunizieren können und der Nutzer über sein mobiles Endgerät alle Daten der vernetzten Geräte einsehen und diese auch auf seinen Wunsch hin steuern kann. Nähere Informationen zu IoT folgen im nächsten Kapitel.

Zur beispielhaften Demonstration des Aufbaus eines solchen Netzes elektronischer Geräte beschäftigt sich diese Studienarbeit mit Oracle SunSpot, einem Sensornetzwerk bestehend aus 2 Sensoren und einer Basisstation. Im Folgenden wird die Inbetriebnahme und Programmierung dieser Sensoren vorgenommen und die darin enthaltene Technik erklärt. Ziel der Studienarbeit ist es, mit Hilfe von SunSpot eine rudimentäre Raumüberwachung zu programmieren, indem bewegte Fenster oder Türen bei Abwesenheit des Besitzers der Wohnung erkannt werden, die Basisstation die Werte sammelt und sie an den Besitzer meldet.

1.1 Ziel dieser Arbeit

Im Rahmen dieser Studienarbeit sollen theoretische Inhalte zum Thema Sensornetze behandelt sowie praktische Arbeiten unter Verwendung von Oracle SunSPOT erstellt werden. Im ersten Teil der Arbeit wird eine Wissensbasis geschaffen, welche Grundvoraussetzung zum Durchführen der praktischen Arbeiten ist. Diese beinhaltet theoretische Kenntnisse über das Internet Of Things, Wireless Sensor Networks, den IEEE-Standard 802.15.4, die in den SunSPOTS verbaute Hardware sowie die dazu benötigte Software.

Die durchgeführten praktischen Arbeiten werden im zweiten Teil dieser Arbeit aufgelistet und beschrieben. Es soll eine rudimentäre Einbruchserkennung realisiert werden, welche Bewegungen durch einen an Türen oder Fenstern angebrachten SunSPOT Sensor erkennt und die gesammelten Informationen an eine Host-Applikation sendet, welche die empfangenen Daten aufbereitet. Im Falle eines unerwünschten Eindringens in einen Raum soll eine entsprechende Meldung an den Wohnungsbesitzer gesendet werden.

Des weiteren soll ein Ausblick gegeben werden, inwieweit das im Rahmen der Studienarbeit erstellte Überwachungssystem erweiterbar ist. Zusätzlich soll die mögliche Integration weiterer Sensoren oder die Verwendung eines Raspberry Pi erläutert werden.

Gegen Ende der Arbeit sollen 'Smart Greenhouse' und 'Bot-So', beide Gewinner der 'IoT-Developer-Challenge', vorgestellt werden. Die beiden Projekte verfolgen ein vergleichbares Ziel wie das in dieser Studienarbeit vorgestellte Konzept und zeigen, wie das IoT zukünftig im täglichen Leben Einzug halten wird.

Abschließend werden die Ergebnisse der Arbeit kurz zusammengefasst und einem Fazit unterzogen. Dieses beinhaltet die Reflexion über die erreichten Ziele und persönliche Erfahrungen in Hinsicht auf die Arbeit mit Sensoren, insbesondere mit Oracle SunSPOT.

Kapitel 2

Internet of Things

Als im Februar 1946 ENIAC, der erste elektronische sowie programmierbare Universalrechner vorgestellt wurde, wog dieser 27 Tonnen und füllte einen gesamten Raum. Für private Anwendungen waren diese Rechnersysteme nicht geeignet. Mit der voranschreitenden Entwicklung werden Computer immer kleiner und leistungsfähiger. Es erschließen sich immer neue Anwendungen von Computersystemen, die hauptsächlich den Menschen in seinem Alltagsleben unterstützen sollen.

Rund um 1990, als das Internet kommerzialisiert und somit für jeden zugänglich wurde, begann eine rasante Entwicklung neuer Technologien. Bis heute hat es unsere Arbeitsweise sowie unser Privatleben verändert und dieser Trend schreitet unabremst voran.

Mit dem Web 2.0 wurden Webseiten interaktiv sowie Videos und Bilder im Internet eine Selbstverständlichkeit. Soziale Netzwerke verbinden die Nutzer durch verschiedenste Arten der Kommunikation. Dieses sich immer weiter aufspannende Kommunikations- und Informationsnetz, erreicht nun auch unsere kleinsten elektronischen Geräte.

Das Haus wird durch ein komplexes Sicherheitssystem überwacht, die Tür benötigt nur den Fingerabdruck um sich automatisch zu öffnen, der Fernseher reagiert auf Spracheingaben und in der Zukunft erstellt der Kühlschrank autonom den Einkaufszettel.

All diese Informationen werden über das Internet zu einer zentralen Sammelstelle oder dem Menschlichen Akteur zugespielt, das Internet of Things (IoT) ist entstanden.

2.1 Geschichte

Etwa um 1982 ärgerten sich drei Studenten der School of Computer Science, an der Carnegie Mellon University über den Getränkeautomaten ihres Instituts. An manchen Tagen liefen sie zu dem Automaten und erhielten entweder keine Getränke, oder zu warme, da diese erst kürzlich nachgefüllt wurden. Um diese Problematik zu lösen entwickelten die Studenten John Zsarnay neue Hardware die mit Software von David Nichols und Ivor Durham die Füllstände, sowie die Temperatur der Getränke überwachte [Car98].

Über den damals an der Universität vorhandenen Vorgänger des Internets, das sogenannte Arpanet konnte direkt beim Automaten der aktuelle Status nachgefragt werden. Dieser antwortete zum Beispiel mit:

1	>	EMPTY	EMPTY	1h 3m
2	>	COLD	COLD	1h 4m

Hiermit informierte der Automat darüber, dass kalte Getränke in der mittleren sowie linken Schiene vorhanden seien, die Getränke der rechten Schiene jedoch noch warm seien. Die angegebene Zeit informierte darüber, wie lange die Getränke sich bereits im Automat befanden. Nach drei Stunden nahm der Automat an, dass die Getränke ausreichend gekühlt seien.

Bereits 1991 schrieb der Amerikanische Informatiker Mark Weiser eine Vision, wie technische Geräte der Zukunft untereinander vernetzt sein könnten [Wei91]. Den Namen IoT erhielt das ganze jedoch erst 1999.

2.2 Ziele und Anwendungsbeispiele

Aus Spielereien und purem Erfindergeist wurde in wenigen Jahren eine ganze Industrie, die sich heute nur mit Produkten des IoT beschäftigt. Es entstanden bereits viele Projekte, denen man im Alltag begegnet, ohne sie wahrzunehmen. Diese lassen sich in 3 Hauptkategorien unterteilen, die gleichzeitig die Ziele des IoTs darstellen:

- Automatisierung
- Informationsgewinnung über bessere Vernetzung
- Entertainment

In vielen dieser Kategorien gibt es bereits umgesetzte Projekte. Manche davon begegnen uns bereits heute unbemerkt im Alltag. In der folgenden Liste sind einige der Erfolgreichsten davon zusammengestellt.

- Umweltsensoren (Temperatur Feuchtigkeit Erschütterung Lautstärke Luftzusammensetzung)
- Lichtsteuerung
- Haushaltshilfen
- Bestandsaufnahme / Nachfuhrkontrolle
- Überwachungsfunktionen
- „Smart Signs“ - Autobahn
- Entertainment
- Haussteuerung
- Prozessüberwachung (Ventile, Flussraten usw.)
- Diagnose / Lebensüberwachung usw.

2.3 Hardware für Privatanwender

In den letzten Jahren sind viele Privatanwender aufgrund der sich bietenden Möglichkeiten dazu übergegangen, eigene Anwendungen mit kleinen IoT-Systemen selbst zu entwickeln. Hierzu werden meist kleine Einplatinencomputer verwendet, da diese sehr günstig zu erhalten sind. Zwei der erfolgreichsten Einplatinen Computer sind die Modelle der Raspberry Pi und Arduino Familie.

2.3.1 Arduino-Plattform

Arduino ist ein Unternehmen, das all seine Produkte in enger Zusammenarbeit mit zugehörigen Community entworfen hat. Alle Entwicklungen basieren auf den Ideen und Konzepten, die die Gemeinschaft um das Unternehmen entwickelt hat.



Abbildung 2.1: Das Arduino-Logo [Arda]

Das Unternehmen entwickelt Mikrokontrollerplatinen, die ihre Umwelt durch Sensoren wahrnehmen und auf äußere Einflüsse reagieren können. Alle Produkte sind vorgefertigt oder als Selbstbau-Kit erhältlich. Da die Baupläne aller Platinen durch Arduino quelloffen im Internet erhältlich sind, können die Produkte auch von Grund auf nachentworfen werden.

Das erste Arduino-Board, dass 2005 vorgestellt wurde, hat bereits viele Revisionen erhalten. In der aktuellsten Version ist es mit einem 16 Megahertz (MHz) 8-Bit Microcontroller ATmega328 der Firma Atmel bestückt. Es besitzt zusätzlich 32 Kilobyte Flash-Speicher von dem 0.5 Kilobyte durch den Bootloader belegt werden. Zur Ein- und Ausgabe, sowie zum Anschließen von Sensoren besitzt die Platine 14 digitale Input- oder Output-Pins, sowie 6 analoge input Pins und einen Universal Serial Bus (USB)-Anschluss. Betrieben wird die Platine mit einem 7-12V Netzteil [Ardb]. Die Platine wird von dem italienischen Unternehmen SmartProjects produziert, ist jedoch aufgrund des quelloffenen Platinenplans auch selbst zusammensetzbare.

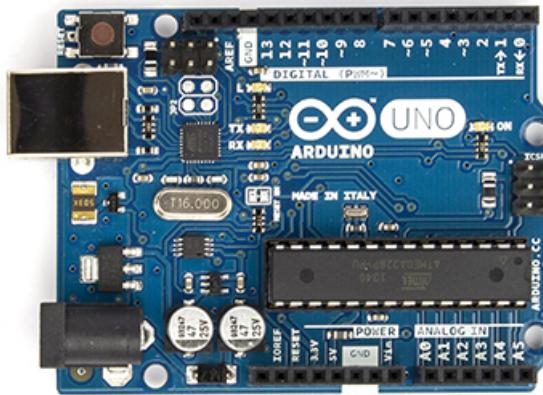


Abbildung 2.2: Die Front der dritten Arduino Uno Revision [Ardc]

Die Arduino-Plattform besitzt eine eigene Integrated Development Environment (IDE), in der die Software für den Microcontroller entwickelt werden kann. Der Mikrocontroller wird mit einer stark vereinfachten Form der Programmiersprachen C und C++ angesprochen. Insbesondere technische Informationen wie Header-Files werden vor dem Programmierer verborgen. Für Anfänger bietet die IDE mehrere Einstiegshilfen, die das Programmieren erleichtern.

Ein Beispiel für die Einstiegshilfen sind die strukturgebenden Funktionen `setup()`, sowie `loop()`. Während in der Setup-Methode alle Aktionen definiert werden müssen, die beim Programmstart durchgeführt werden sollen, wird der Code in der Loop-Methode unbegrenzt oft wiederholt. Ein Beispielcode, der eine an Pin 13 angeschlossene LED blinken lässt, lautet wie folgt:

Listing 2.1: Simpler Arduino-Code, der eine LED blinken lässt

```
1 int led = 13; //LED an Pin 13
2
3 void setup() {
4     pinMode(led, OUTPUT);
5 }
6
7 void loop() {
8     digitalWrite(led, HIGH);
9     delay(1000);
10    digitalWrite(led, LOW);
11    delay(1000);
12 }
```

Seit der Entwicklung der Arduino-Boards erfreuen sich diese großer Beliebtheit. So wurden bis 2013 insgesamt 700.000 zusammengesetzte Platinen verkauft [Cua13].

Viele der Projekte, die mit Arduino-Platinen durchgeführt werden, beinhalten Benutzerinteraktion. Auch für Kunstinstallationen werden die Einplatinencomputer gerne verwendet [Ele15].

2.3.2 Raspberry Pi

Die in Großbritannien angesiedelte Wohltätigkeitsorganisation "Raspberry Pi Foundation" hat es sich zur Aufgabe gemacht, Einplatinencomputer für Menschen zu entwickeln, die Hardware- oder Programmierkenntnisse erwerben wollen. Um auch für Jugendliche und Studenten ansprechend zu sein, wurde ein besonders geringer Verkaufspreis angestrebt. Die verschiedenen Modelle des Raspberry Pis kosten zwischen 20 und 35 US-\$ und sind somit sehr günstig.

Der Name entstand durch die Kombination zweier Gedanken. Viele Computer wurden nach Früchten benannt (z.B. Apple). Aus diesem Grund entschieden sich die Gründer für Raspberry, dem englischen Wort für Himbeere. Der Namenszusatz Pi entstand aus dem ursprünglichen Plan, die Raspberry Pi Einplatinencomputer mit einem Phyton Interpreter auszuliefern, daher die Abkürzung PI. Als Ergebnis eines öffentlich ausgeschriebenen Wettbewerbs wurde eine stilisierte Himbeere als Logo für die Wohltätigkeitsorganisation gewählt.

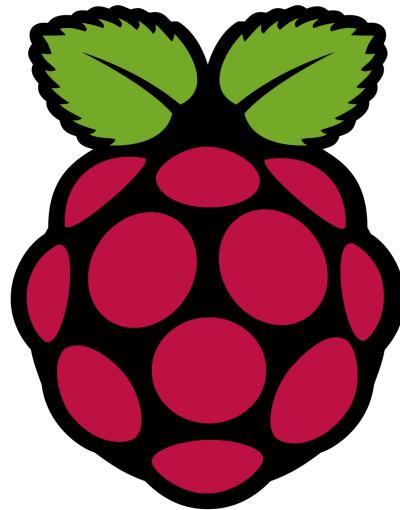


Abbildung 2.3: Das Logo der Raspberry Pi Foundation [Ras]

Insgesamt bietet die Raspberry Pi Foundation sechs Modelle des Computers an. Eine sehr grundlegende Ausführung, das Compute Module, das nur in Stückzahlen über 100 verkauft wird, sowie die günstigen Modelle A und A+, die mit 25 und 20 US-\$ am unteren Preisende liegen. Die Modelle B, B+, sowie das überarbeitete Modell Generation 2 B sind am oberen Preisende, mit jeweils 35 US-\$ angesiedelt.



Abbildung 2.4: Die Oberseite des Raspberry Generation 2 B-Modells [Mul15]

Mit dem Anfang 2015 vorgestellten Generation 2 B - Modell wurde ein Performancesprung realisiert. Mit dem System-on-a-Chip Broadcom BCM2836, das sowohl CPU als auch GPU des Einplatinencomputers darstellt, sind auf vier Kernen Berechnungen mit bis zu **900 Mhz!** (**Mhz!**) möglich. Dies wird mit 1 Gigabyte Arbeitsspeicher zu einem, für den geringen Preis, leistungsstarken System kombiniert. Die Hardwarekomponenten des Generation 2 B - Modells sind so leis-

tungsstark, dass das Ausführen von Windows 10 möglich ist.

Auch in Sachen I/O-Anschlüsse ist der Raspberry Pi gut ausgerüstet. Er stellt insgesamt 17 Pins bereit, die sowohl als Input, sowie Output genutzt werden können. Zusätzlich verfügt der Raspberry Pi über 4 USB-Ports, einen HDMI-Anschluss und einen MicroSD-Karten Slot. Dieser wird als Speicher für das Betriebssystem und alle selbstentwickelten Programme verwendet.

Um den Raspberry Pi mit Netzwerken zu verbinden besitzt der Raspberry ein 100Mbit Ethernet Port. Zusätzlich sind viele Treiber vorinstalliert, die die Verwendung von USB-WLAN-Adaptoren ermöglichen.

Über Micro-USB wird die Platine mit 5V versorgt, die sich mit der Stromstärke von bis zu 800mA auf 4 Watt aufsummieren.

Um Hardwareerweiterungen zu ermöglichen, bietet der Raspberry Pi einen seriellen Anschluss. Eine beliebte Erweiterung ist das Camera Module, das wie der Name bereits sagt, eine Kamera mit sich bringt. Diese besitzt eine Auflösung von fünf Megapixel und ist besonders in Projekten mit Überwachungsszenarien beliebt. Um auch nachts oder in schwach belichteten Umgebungen Aufnahmen tätigen zu können, ist das Camera Modul unter dem Namen NoIR ohne Infrarotfilter erhältlich. Dies ermöglicht in Kombination mit Infrarotlicht Aufnahmen in Dunkelheit.

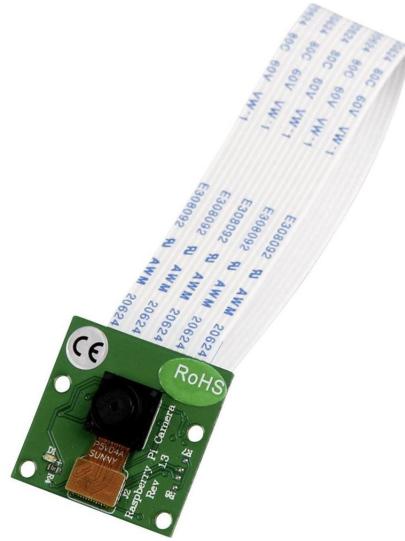


Abbildung 2.5: Das Camera Modul für den Raspberry Pi [Con]

Der Raspberry Pi ist mit seinen 85 x 56mm so groß wie eine Kreditkarte. Zusätzlich zum platzsparenden Format ist der Einplatinencomputer mit 45 Gramm ein Leichtgewicht.

2.4 Sicherheitsaspekte

Über das Internet werden immer mehr Informationen versendet. Ein kleiner Bestandteil hiervon sind auch die Informationen, die verschiedene Geräte des IoTs untereinander austauschen. Diese automatisch abgewickelte Machine-to-Machine (M2M)-Kommunikation stellt ein hohes Sicherheitsrisiko für Unternehmen und Privatpersonen dar. Durch einen gezielten Angriff könnten personenbezogene oder sicherheitsrelevante Informationen an die Öffentlichkeit gelangen. Aufgrund dessen müssen sich die Verantwortlichen die Frage stellen, wie man mit dieser Herausforderung in der Zukunft umgeht.

Sicherheitsexperten sind sich bereits heute einig, dass das IoT ein enormes Risiko darstellt [Wil15]. Für Privatpersonen äußert sich dies hauptsächlich in der Sicherheit ihrer elektronischen Geräte. So musste der Automobilhersteller BMW kürzlich ein Softwareupdate für seine Automobile mit dem ConnectedDrive-System ausliefern, da es Hackern ohne Spuren zu hinterlassen gelungen war, die Türen mittels Smartphones zu öffnen [ZEI15].

Zusätzlich zur Beschädigung oder Entwendung von Eigentum durch Hacker besteht ein Risiko, private Informationen zu verlieren. Gerade Informationen, wie E-Mail-Adressen oder Passwörter, stehen in großem Interesse der Hacker. Solche könnten aus einem privaten IoT entwendet werden und für weitere Cyber-Kriminelle Aktionen genutzt werden.

Auch Firmen müssen sich die Risiken bewusst machen, die sie durch die Nutzung von IoT-Geräten eingehen. Oftmals nutzen Unternehmen IoT-Systeme als Infrastrukturkomponenten, über die sie einen Service betreiben. Solche Netze stellen einen idealen Angriffspunkt für Wirtschaftsspionage oder Denial of Service - Attacken dar. Bricht ein Hacker in solche Systeme ein, wird es ihm schnell möglich sein, einen Millionenschaden anzurichten.

Da die Entwicklung des IoT derzeit noch an Fahrt aufnimmt, muss jedem Nutzer bewusst sein, dass auch die Sicherheit noch nicht vollständig entwickelt ist. Viele Aspekte werden sich in den nächsten Jahren weiterentwickeln, die Sicherheit wird jedoch zunehmend eine der wichtigsten Komponenten des IoTs.

Jede am IoT teilnehmende Komponente muss einen Zugang zur aufgebauten Netzwerkumgebung besitzen, um ihre Metadaten an andere Teilnehmer zu kommunizieren. Über diese zusammengesammelten Informationen trifft das System eine Entscheidung über folgende Handlungen. Hierbei entsteht ein Sicherheitsri-

siko.

Mit netzwerkfähigen Geräten ist es möglich, sich in das aufgebaute Netzwerk einzuklinken und sich ebenfalls als Komponente auszugeben. Dies ermöglicht einem Angreifer Daten abzufangen oder zu seinen Zwecken zu manipulieren. Dies kann das Verhalten des gesamten Systems verändern.

Ebenfalls ist es möglich, dass ein Angreifer ein Teilnehmer des IoTs übernimmt und somit Zugriff auf das Netzwerk erhält.

2.4.1 Sicherheitsmodell nach Gordon Cooper

Wie schützt man das IoT?

Um diese Frage beantworten zu können, müssen die Schnittstellen und Eckpunkte des IoTs definiert werden die Hackern Angriffsflächen bieten. Zu diesem Zweck hat Gordon Cooper ein Sicherheitsmodell aufgestellt, anhand dessen Sicherheitsmaßnahmen erklärt werden können. Dieses Sicherheitsmodell wird im Abbildung 2.6 dargestellt.

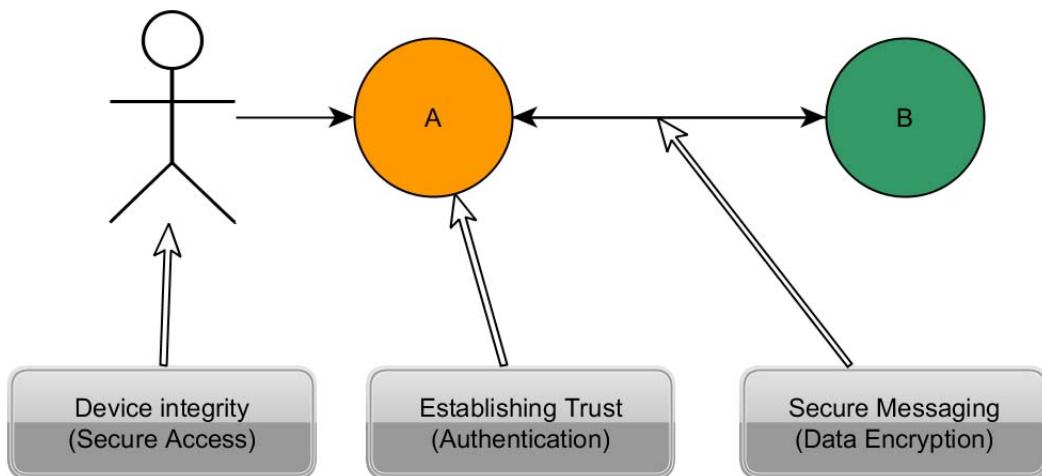


Abbildung 2.6: Sicherheitsmodell nach Gordon Cooper [Coo15]

Um die Integrität der IoT-Komponenten zu schützen, muss der Zugang und somit die Anmeldung an jedem Knoten geschützt werden. Dies wird im Sicherheitsmodell nach Gordon Cooper als "Secure Access" bezeichnet.

Wie im privaten wird auch das Funknetz, in dem sich die Teilnehmer des IoTs einzuklinken, durch einen Sicherheitsschlüssel geschützt. Dieser muss von den Geräten,

sowie allen Benutzern angegeben werden, um Zugang zum Netz zu erhalten. Eine wichtige Aufgabe besteht darin, diesen Schlüssel zu schützen. Hierbei muss unter anderem die Weitergabe, sowie die Kommunikation im Netz durchdacht und überwacht werden. Besonders bei der Schlüsselwahl ist Vorsicht geboten. Es sollten niemals Namen, Geburtsdaten oder andere Informationen, die einer Person zugeordnet werden können, verwendet werden. Zusätzlich sind gebräuchliche Wörter nicht empfehlenswert, da diese leicht durch Wörterbuchangriffe überlistet werden können. Damit die Sicherheit des Passworts gewährleistet ist, sollte es mindestens 8 Zeichen enthalten und groß/klein Schreibung, sowie Sonderzeichen und Zahlen enthalten. Eine allgemeingültige Faustregel ist: Ums so länger, umso besser.

Eine weitere Sicherheitsmaßnahme sind maßgeschneiderte Zugriffsrechte. Jeder Teilnehmer im IoT darf nur so viele Rechte besitzen, wie unbedingt nötig. Hierdurch wird sichergestellt, dass Angreifer beim Erlangen eines Zugangscodes nur eingeschränkte Handlungsmöglichkeiten besitzen und der mögliche Schaden gering gehalten wird.

Um die Anmeldung an IoT-Geräten möglichst sicher zu gestalten, sollte ein asymmetrisches Schlüsselpaar verwendet werden. Hierbei wird pro Kommunikationsteilnehmer ein Schlüsselpaar verwendet, das aus einem öffentlichen Schlüssel, dem "Public Key" und einem privaten Schlüssel, dem "Private Key" besteht. Wie der Name bereits sagt, ist der "Private Key" nur beim Schlüsselinhaber vorzufinden und darf nicht öffentlich werden. Der "Public Key" hingegen wird jedem Kommunikationspartner bereitgestellt und kann über unsichere Kommunikationswege geteilt werden.

Die Verschlüsselung der Kommunikation geschieht mit einem Algorithmus, der das Entschlüsseln nur mit dem nicht zum Verschlüsseln verwendeten Schlüssel ermöglicht. Dies bedeutet, dass von dem Inhaber des "Private Keys" versendete Nachrichten auch unbestreitbar von ihm stammen. Durch die hiermit stattfindende Authentifikation wird der zweite Sektor "Authentication" des Sicherheitsmodells nach Gordon Cooper erfüllt.

Da jede Kommunikation zwischen zwei IoT-Komponenten potentiell sensible Daten beinhalten kann, muss permanent ein verschlüsselter Kanal verwendet werden. Dies ist im Sicherheitsmodell mit dem Sektor "Secure Messaging - Data

Encryption” beschrieben. Um dies effizient zu gestalten, wird das beschriebene Public-Key-Verfahren genutzt, um einen symmetrischen Schlüssel auszuhandeln. Bei diesem handelt es sich um einen Schlüssel, der sowohl zum Ver- als auch Entschlüsseln verwendet wird. Dieser Session-Key genannte Sitzungsschlüssel wird bei der Kommunikationsaufnahme der Komponenten generiert und mithilfe des Public-Keys des Kommunikationspartners verschlüsselt. Daraufhin ist es dem Empfänger möglich, den symmetrischen Schlüssel für die weitere Kommunikation zu verwenden. Da der Session-Key bei jeder Sitzung neu ausgehandelt wird, ist er nur den Kommunikationspartnern bekannt und bietet ausreichende Sicherheit.

2.4.2 Verschlüsselungsstandard AES

Der aktuell gültige Kryptographiestandard ist das im Jahr 2000 veröffentlichte Advanced Encryption Standard (AES). Bei diesem Verfahren wird eine symmetrische Block-Cipher verwendet, die mit einem 128, 192 oder 256 Bits langen Schlüssel gleichlange Datenblöcke verschlüsselt [Haa08].

Solange ein Verschlüsselungsverfahren wie AES nicht durchbrochen wird, kann es nur durch sogenannte Brute-Force-Attacken angegriffen werden. Hierbei werden alle Möglichkeiten durchprobiert, bis man zufällig auf den richtigen Schlüssel trifft. Bei AES mit 128 Bit Schlüssellänge sind hierfür bis zu $2^{128} \equiv 3,4 * 10^{38}$ Versuche nötig. Bei 256 Bit sind es bereits $2^{256} \equiv 1.2 * 10^{77}$ Versuche. Selbst mithilfe der aktuellsten Supercomputer würde dies Millionen Jahre in Anspruch nehmen.

Schlüssellänge	Kombinationsmöglichkeiten
1-Bit	2
2-Bit	4
4-Bit	16
8-Bit	256
16-Bit	65536
32-Bit	$4.2 \cdot 10^9$
56-Bit (DES)	$7.2 \cdot 10^{16}$
64-Bit	$1.8 \cdot 10^{19}$
128-Bit(AES)	$3.4 \cdot 10^{38}$
192-Bit(AES)	$6.2 \cdot 10^{57}$
256-Bit (AES)	$1.2 \cdot 10^{77}$

Tabelle 2.1: Auflistung der Kombinationsmöglichkeiten bei verschiedenen Schlüssellängen

Jede in AES erlaubte Schlüssellänge bietet aufgrund des komplexen Verschlüsselungsverfahrens und der daraus resultierenden Rechenzeit pro Angriff ausreichende Sicherheit. Da 128 Bit Schlüssel jedoch 40% weniger Rechenleistung als 256 Bit Schlüssel benötigen, wird meist nur die minimale Schlüssellänge implementiert.

Kapitel 3

Theoretische Grundlagen

In diesem Kapitel erklären wir kurz, dass wir in den folgenden Unterkapiteln Grundlagen erklären. Worum es sich hierbei handelt können Sie den folgenden Kapiteln entnehmen.

Text muss noch abgeändert werden!

3.1 Wireless Sensor Networks

Hier wird alles zu Wireless Sensor Networks erklärt. Bisher steht das erst als Gerüst.

3.1.1 Ubiquitäres Rechnen

1988 verwendete Mark Weiser erstmals den Begriff 'ubiquitous computing' (dt. ubiquitäres Rechnen), um seine Vision nach einem stets verfügbaren Rechensystem, welches dem Nutzer unsichtbar erscheinen soll, zum Ausdruck zu bringen. Der Computer soll sich so in den Alltag integrieren, dass die Menschen ihn gar nicht mehr bemerken. Nach seiner Vorstellung verbessere das ubiquitäre Rechnen die Erfahrungen, die man mit Computern macht, da die Rechner dem Nutzer nahtlos verfügbar gemacht werden, ohne dabei effektiv sichtbar zu sein.

Weiser zufolge sind die besten Technologien diejenigen, die scheinbar verschwinden, tatsächlich jedoch nur in den Hintergrund geraten und unsichtbar werden. Der Mensch soll nicht in der Welt des Computers leben, sondern der Rechner soll sich in die Welt des Menschen integrieren. In Lichtschaltern, Thermostaten, Stereoanlagen und Backöfen werden bereits heute kleine Rechner verbaut, die helfen sollen, den Alltag zu erleichtern und die Idee des 'Internet of

Things' weiter zu verfolgen.

Da Ubiquitäres Rechnen zuverlässig und unsichtbar funktionieren soll, ist die Technologie der unsichtbaren Rechenmodule von großer Bedeutung. Voraussetzungen sind z.B. leistungsstarke Prozessoren, ausreichend Speicherplatz, drahtlose Kommunikation, Sensoren und Aktoren (die z.B. mit der Umwelt und dem Menschen interagieren). Der Mensch muss nicht für alle Anwendungsfälle von ubiquitärem Rechnen direkt eingebunden werden, da die Systeme auch autonom arbeiten können [WB12].

3.1.2 Motivation von Sensornetzen

Sensornetze sind sehr flexibel und können unter anderem dafür eingesetzt werden, um

- Umwelteinflüsse wahrzunehmen ('sensing')
- Umwelteinflüsse zu verarbeiten und zu analysieren ('computing')
- Daten zu übertragen ('transport')
- Netzwerke für verteilte Systeme aufzubauen ('networking')
- Die Umwelt zu beeinflussen und zu verändern ('actuation')

Für viele Anwendungsfälle und Szenarien, in denen mit der Umwelt interagiert wird, soll die Benutzung von drahtlosen Sensornetzen zukünftig ausgebaut und etabliert werden. Der Einsatz von Sensornetzen kann dabei verschiedene Motivationen und Anforderungen haben:

- Direkte Interaktion mit Menschen ist nicht möglich oder nicht erforderlich (z.B. bei Überwachung einer Maschine in der Industrie)
- Der Mensch soll nur im Notfall alarmiert werden (z.B. in Notfällen oder wenn die Sensoren bestimmte Schwellenwerte erreichen)
- Es handelt sich um ein autonomes System, welches nur Selten das Handeln eines Menschen erfordert

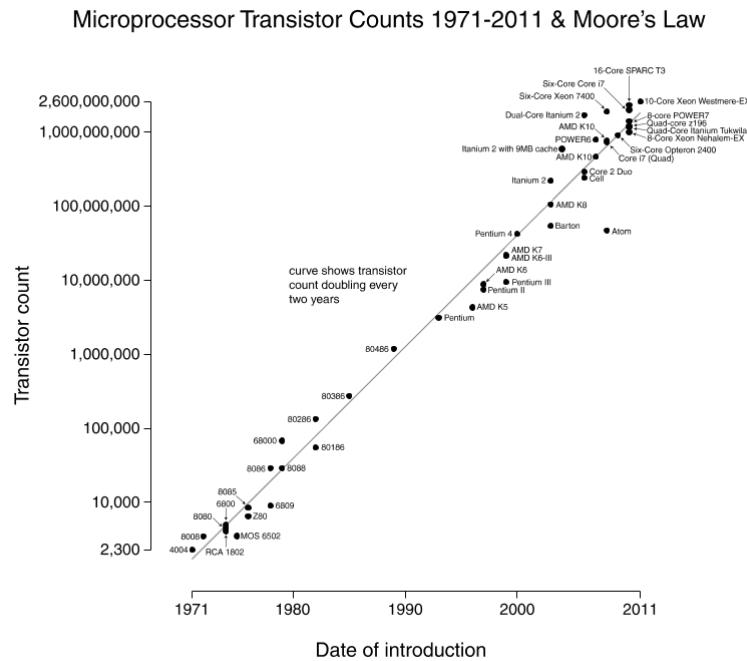


Abbildung 3.1: Mikroprozessoren-Transistoren im Laufe der Zeit [Wgs]

Eine weitere Motivation der Verwirklichung von drahtlosen Sensornetzen ist der Technologiefortschritt, der es möglich macht, immer kleinere Rechengeräte mit mehr Leistung herzustellen und miteinander zu vernetzen. Um diesen Fortschritt zu verdeutlichen, formulierte Gordon Moore 1965 ein Gesetz, welches besagt, dass sich die Anzahl der integrierten Schaltkreise auf einem Mikroprozessor alle 18-24 Monate verdoppelt. Im Gegensatz zu den Anzahl der Schaltkreise steigt die Rechenleistung der Prozessoren allerdings nicht linear an, da immer mehr Schaltkreise für den Cache des Prozessors verwendet werden, was nur geringfügig der Leistungssteigerung dient. Das Ende der Gültigkeit des Mooreschen Gesetzes wurde schon des öfteren wegen unüberwindbarer technischer Grenzen vorausgesagt, diese wurden jedoch bisher alle mit dem Einsatz neuer technischen Mittel und Materialien überwunden. Momentan schätzt der Halbleiterhersteller Intel, welcher 1968 von Moore mitbegründet wurde, dass das Mooresche Gesetz noch mindestens bis 2023 seine Gültigkeit behält. Mittlerweile existieren beim Konzern sogar explizite Pläne, die das Einhalten des Mooreschen Gesetzes sicherstellen sollen. [WB12] [Kah12] [Tuo02].

3.1.3 Bestandteile

Um die Kommunikation der einzelnen Knoten untereinander zu koordinieren, gibt es neben den normalen Sensoren weitere Bestandteile eines 'Wireless Sensor Network'.

Aktoren werden dazu benötigt, um das Sensornetz Einfluss auf die Umwelt nehmen zu lassen.

'Aggregating Nodes' werden dazu gebraucht, um die Daten von verschiedenen Sensoren zu sammeln, zu verarbeiten und zu kombinieren.

'Sink Nodes' sammeln die Daten von verschiedenen Sensoren und geben diese an die Basisstation bzw. das Backend-System weiter.

Backend-Systeme dienen zu weiteren Verarbeitung und Analyse der Daten. Diese sind von Vorteil, wenn z.B. komplexere Berechnungen durchgeführt werden sollen oder die Daten langfristig gespeichert werden sollen [WB12].

3.1.4 Topologien

Bei einem Aufbau eines Sensornetzes stellt sich grundsätzlich die Frage, wie die einzelnen Sensoren miteinander Verbindungen aufbauen und kommunizieren sollen. Ein solche Verbindungsstruktur nennt sich in der Informatik 'Topologie'. Da das Sensornetz insgesamt zuverlässig arbeiten soll, Kosten und Komplexität jedoch gering gehalten werden sollen, wurden speziell für die drahtlosen Sensornetze neue Ansätze im Bereich der Topologie erforscht. Im folgenden sollen 4 Topologie-Alternativen näher erläutert werden.

Peer-to-Peer Netzwerke erlauben es, dass jeder Knoten im Netz (in unserem Fall der Sensor) mit jedem anderen Knoten direkt Kontakt aufnehmen kann. Jedes 'Peer-Gerät' ist gleichzeitig Client und Server gegenüber anderen Knoten im Netzwerk.

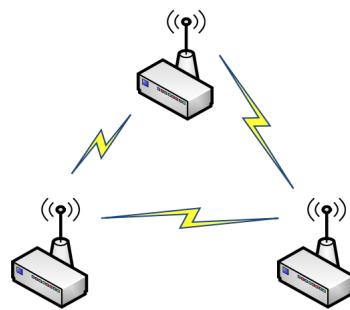


Abbildung 3.2: Peer-To-Peer Netzwerk [Kos09]

Bei der Stern-Topologie sind die Sensoren an ein zentrales Kommunikationsgerät angebunden. In diesem Fall kommunizieren die einzelnen Knoten nicht direkt miteinander. Jegliche Art von Kommunikation wird über das zentrale Gerät (auch Hub genannt) geroutet. Der Hub wird hier als Server betrachtet, wohingegen die Knoten (Sensoren) die Clients darstellen.

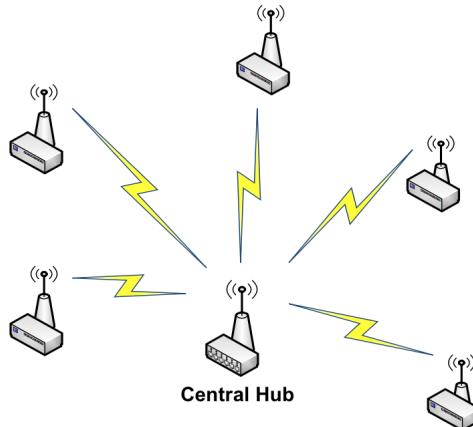


Abbildung 3.3: Stern Netzwerk [Kos09]

Die Baum-Topologie stellt eine Hybridvariante aus Peer-to-Peer und Stern dar. Sie nutzt einen sogenannten 'Root-Knoten' als zentraler Router. Eine Ebene darunter liegen die Hubs, an denen wie in der Stern-Topologie die Sensoren angebunden sind.

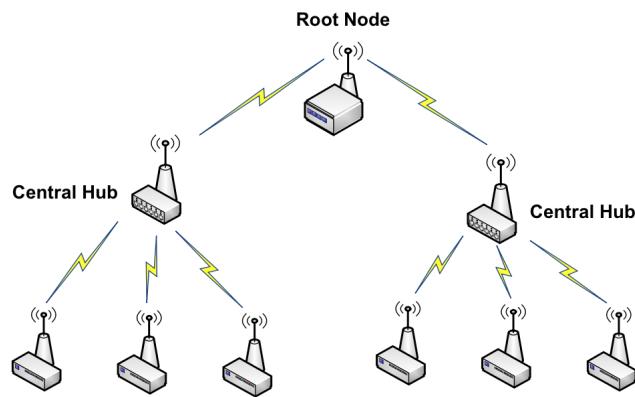


Abbildung 3.4: Baum Netzwerk [Kos09]

Eine weitere Mögliche Variante ist ein vermaschtes Netz. Die Knoten sind untereinander ohne zentralen Hub verbunden und die Daten werden einfach von Knoten zu Knoten weitergesendet, bis sie ihr gewünschtes Ziel erreicht haben [Kos09].

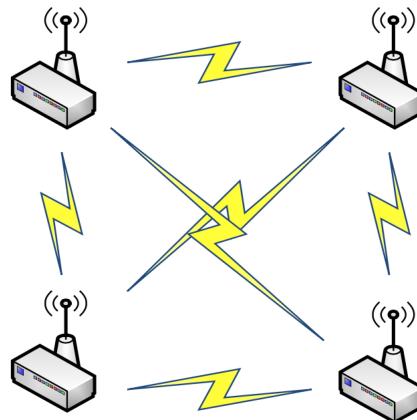


Abbildung 3.5: Vermaschtes Netzwerk [Kos09]

3.1.5 Schwierigkeiten

Beim Planen eines drahtlosen Sensornetzwerkes stellen sich vermehrt eine Schwierigkeiten heraus, die zu berücksichtigen sind. Viele dieser Schwierigkeiten sind auch untereinander abhängig und beeinflussen gleichzeitig die verwendete Elektronik, physikalische Aspekte, Rechenleistung oder auch Lebensdauer eines Sensors.

Die erste Schwierigkeit, die es bei einem Sensor zu betrachten gilt, ist die Versorgung des Sensors mit Energie. Dazu gibt es unterschiedliche Ansätze, wie z.B.

die Versorgung des Sensors mit Batterien. Dies hat allerdings den Nachteil, dass deren Lebenszyklus beschränkt ist und Batterien früher oder später ausgetauscht werden müssen. Ein Akkumulator eignet sich hier schon besser, allerdings bleibt die Frage, wie der Akkumulator mit neuem Strom versorgt werden soll.

In Zusammenhang mit dieser Fragestellung ist die Betrachtung der möglichen Gewinnung oder Rückgewinnung von Energie durch den Sensor von Vorteil. Wie kann ein Sensor Energie gewinnen, ohne dass er damit direkt versorgt werden muss? Möglichkeiten wären die Gewinnung von Solarenergie durch den Sensor, das Nutzen von Temperaturunterschieden in der Umwelt, die Rückgewinnung der Bewegungsenergie (z.B. durch Wind o.ä.) oder das Ausnutzen von Erschütterungen und Vibrationen. Diese Möglichkeiten sollten nur in Betracht gezogen werden, wenn der Sensor eine lange Lebensdauer mit sich bringen soll. Ist der Einsatz der Sensoren in absehbarer Zeit vorüber, lohnt es sich aus Produktions- und Kostengründen die begrenzte Lebenszeit des Sensors hinzunehmen, um ihn danach z.B. durch neue Sensoren auszutauschen.

Ein weiterer wichtiger Aspekt im Bereich Energie ist die Energieeffizienz. Dabei sollen alle Teile in einem Sensorknoten möglichst effizient nutzen, um die Energie sinnvoll und langsam aufzubrauchen. Nicht nur der Sensor selbst spielt dabei eine wichtige Rolle, sondern auch wie das Netzwerk um ihn herum aufgebaut und genutzt wird. Folgende Aspekte spielen bei der Energieeffizienz eine erhebliche Rolle:

- Wahrnehmen der Daten durch die Sensoren
- Verarbeitung der Daten
- Sicherung der Daten
- Übertragung der Daten
- Empfang von Daten

Damit der Energieverbrauch weiter eingeschränkt werden kann, sollten Sensoren nur aktiv sein, wenn sie wirklich benötigt werden. Ansonsten sollten sie sich in einen Sleep- bzw. Energiesparmodus begeben, um Energie zu sparen. Des Weiteren wäre zur ausschließlichen Wahrnehmung der Umgebung ein "Controller- bzw. Sensormodus und zum Senden und Empfangen von Daten ein "Radio bzw.

Übertragungsmodus von Vorteil.

Eine weitere Schwierigkeit, die sich stellt, ist der Einsatz bzw. die Verteilung der Sensoren in der Umwelt und ihre Selbstverwaltung. Die Knoten könnten entweder zufällig in der Umgebung platziert oder systematisch angeordnet werden. Hier entscheidet der jeweilige Anwendungszweck, wobei das systematische Platzieren der Sensoren meistens sinnvoller und effizienter ist. Des Weiteren sollte unterschieden werden, ob aktive oder passive Sensoren eingesetzt werden sollen. Auch hier muss je nach Anwendungsfall unterschieden werden. Passive Sensoren eignen sich besser, wenn Daten nur erfasst und übermittelt werden sollen. Aktive Sensoren sollten eingesetzt werden, wenn auf die Erfassung der Daten eine eventuelle Aktion bzw. Reaktion mit der Umwelt erforderlich ist.

Sensoren sollten bestimmte Informationen über sich selbst und ihre Nachbarn wissen bzw. ermitteln können. Dazu gehören unter anderem ihre eigene Position, die Ortung der Nachbarknoten und ihre Identifikation, ihre eigene Knotenkonfiguration und ihre kürzeste Route zu einer Basisstation. Denn sobald ein Sensornetz einmal in Betrieb genommen wurde, muss es in der Lage sein, sich autonom betreiben und verwalten zu können. Dazu zählen die Anpassung an veränderte Umweltbedingung und das Kompensieren von Fehlern. Beim Ausfall eines Sensors soll das Sensornetz weiterhin aktiv und funktionsfähig bleiben.

Auch in Hinsicht auf die Sicherheit gibt es einige Aspekte zu betrachten. Manche Sensornetzwerke übertragen empfindliche und kritische Informationen, was sie zu einem beliebten Angriffsziel macht. Sie können sowohl von innen, von außen als auch direkt an den Knoten angegriffen werden. Es stellt sich als schwierig heraus, solche Netzwerke vor Angriffen zu schützen, da sie entfernt und selbstständig arbeiten, drahtlos kommunizieren und meistens keine speziellen Sicherheitsfeatures besitzen. Dies ist aus Energie-, Kostengründen und Gründen der Form und Größe der Sensoren meist nicht realisierbar. Übliche Sicherheitstechniken sind meist nicht durchführbar, da den Knoten üblicherweise die Rechen-, Kommunikations- und Speicherressourcen fehlen. Man braucht neu entwickelte Sicherheitsmechanismen für Sensornetze, die spezielle Lösungen für die Erkennung von Eindringlingen, Verschlüsselung, Schlüsselverwaltung und Verteilung und Registrierung von neuen Knoten besitzen, sodass die Ressourcen der Sensoren ausreichen und das Sicherheitskonzept realisierbar ist [WB12].

3.2 Adhoc-Netzwerke

Adhoc-Netze sind in sich geschlossene Netzwerke, organisieren sich selbst und haben keine bestimmte Hierarchie. Sie bauen sich nur für die Dauer einer Datenübertragung auf, besitzen keine festgelegte Kommunikationsstruktur und verwalten und organisieren sich selbst.

Adhoc-Netze sind leistungsfähige und zählen als so genanntes 'Self Organized Network' (SON), welche gute Lastverteilung betreiben und ohne zentrales Management auskommen. Die Endgeräte übernehmen in diesem Fall das Routing und speichern die Routingtabellen selbst ab. Geräte, die sich dem Netzwerk anschließen, werden dynamisch in das Netz eingefügt. Bei Netzwerken nach IEEE 802.11 (WLANs) und IEEE 802.15 (WPANs - hier im Speziellen IEEE 802.15.1 - Bluetooth) werden alle Geräte selbstständig erkannt und werden dem Netz hinzugefügt. Sie sind fortan Bestandteil des Gesamtnetzes.

Bei Adhoc-Netzen mit vielen Geräten (das kann z.B. ein Sensornetz sein) wird zumeist eine Multihop-Verbindung bevorzugt. Das bedeutet, dass die Daten von einem Netzknoten, z.B. einem Sensor oder Rechner, zu dem nächsten Netzknoten weitergeleitet werden, bis es sein Ziel erreicht hat. Fällt ein Knoten aus, wird wenn möglich ein anderer Weg für die Übertragung genutzt, um Ausfälle zu vermeiden.

Ad-hoc-Netzwerke bestehen virtuell für einen begrenzten Zeitrahmen. Ad-hoc bedeutet etwa "für den Augenblick gemacht". Sie werden in WLANs, WPANs, in Sensornetzen (WPANs mit geringer Datenrate - siehe IEEE 802.15.4) und in Funknetzen von Rettungsdiensten, Polizei und Militär benutzt [LLK12].

3.2.1 MANETs

MANET bezeichnet die Abkürzung für den Begriff 'Mobile Ad-hoc Network'. Jochen Schiller beschreibt in seiner Literatur 'Mobile Communications' den Begriff des MANETs wie folgt:

„Ad-hoc-Netze kommen ohne jegliche Infrastruktur aus, insbesondere ohne eine ausgezeichnete Basisstation, welche den Medienzugriff zentral steuert. Diese Netzvariante erlaubt die spontane, nicht vorab geplante Kommunikation zwischen mobilen Endgeräten, wobei einige oder alle Endgeräte auch Daten von anderen Endgeräten weiterleiten können.“

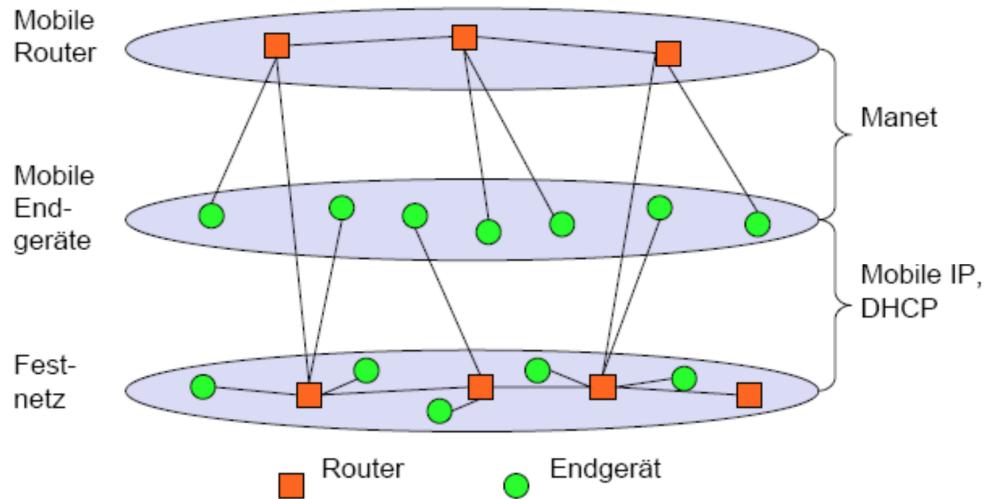


Abbildung 3.6: Einordnung von MANETs [TS11]

MANETs zeichnen sich vor allem dadurch aus, dass sie keine feste Infrastruktur besitzen. Es herrscht innerhalb des Netzes eine dynamische Topologie, was bedeutet, dass zu jedem Zeitpunkt bestimmte Knoten wegfallen und neue dazu kommen können. Dies führt dazu, dass bislang bekannte und zulässige Routen wegfallen, dafür aber auch neue Routen entstehen können. Unter den Geräten besteht eine spontane Vernetzung. Das bedeutet, dass jedes Gerät sowohl Endpunkt einer Übertragung sein kann, jedoch auch in der Lage sein muss, Daten weiterleiten zu können. Durch dieses Weiterleiten von Daten entsteht eine so genannte 'Multihop-Umgebung', da die Daten nicht vom Sender direkt zum Empfänger gelangen, sondern den Empfänger über mehrere Zwischenstationen erreichen.

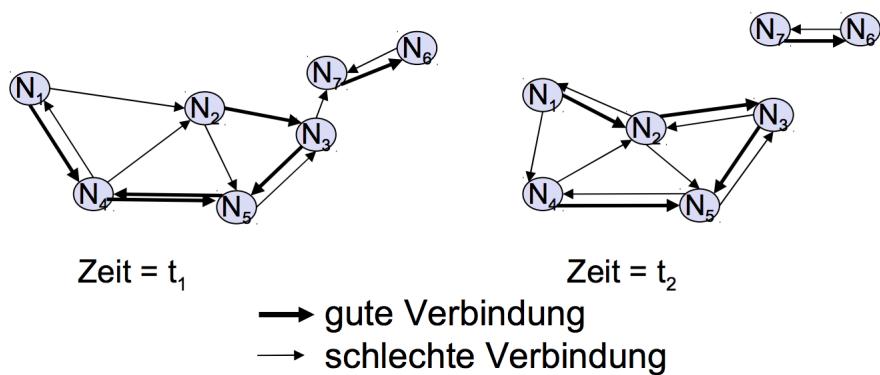


Abbildung 3.7: Mögliche Verbindungen eines Beispiel MANETs [TS11]

Mobile Ad-hoc Netzwerke besitzen eine stark begrenzte Bandbreite. Da sie meistens auf stromsparende Übertragungsprotokolle setzen, können dementsprechend auch nur geringe Mengen an Daten übertragen werden. Des Weiteren bestehen zumeist durch die unterschiedlichen Übertragungsleistungen der Geräte asymmetrische Verbindungen, welche sich für die Übertragung von Daten innerhalb des Netzes besser eignen.

Ein Problem bei MANETs stellt die beschränkte Möglichkeit der Energieversorgung dar, da die Endgeräte meistens abhängig von Batterien oder Akkus sind. Durch die hohe Menge an Geräten in einem Netz steigt dazu die Summe des potenziellen Energiebedarfes pro Gerät weiter an. Auch die Sicherung des Netzes vor physischen Angriffen ist stark beschränkt. So sind z.B. Denial-of-Service-Attacken (DoS), Überwachungsangriffe oder auch das Verfälschen von Nachrichten oft leicht zu realisieren [TS11].

3.2.2 Routingprotokolle

Klassische Routingprotokolle versagen bei dem Versuch, sie innerhalb von mobilen Ad-hoc Netzwerken einzusetzen. Gründe dafür sind folgende:

MANETs zeichnen sich durch eine hohe Dynamik aus, das Netz verändert sich spontan, schnell und ständig. Klassische Protokolle besitzen eine zu langsame Konvergenz, um dem Anspruch der sich ständig verändernden Ad-hoc Netze gerecht zu werden und sind somit nicht geeignet.

Ad-hoc Netze besitzen wie bereits erwähnt meist eine geringe Bandbreite. Hinzu kommt, dass den angeschlossenen Knoten und Geräten nur eine gewisse Rechenleistung zur Verfügung steht. Sie sind somit mit klassischen Routingprotokollen überfordert, da diese meist viel Rechenleistung und Übertragungsleistung erfordern, welche mit den mobilen Geräten nicht realisierbar sind. Der Overhead der klassischen Protokolle ist also für diese Art von Netzwerken zu groß.

Es gibt in mobilen Ad-hoc Netzen gewisse Metriken, die beim Betrieb berücksichtigt werden müssen. Die klassischen Protokolle interessieren sich nicht für diese Metriken und lassen sie außen vor. Bei der Auswahl von Routen müssen unter anderem folgende Aspekte betrachtet werden:

- Batterilaufzeit der Geräte

- Zeit der Verbindung zwischen zwei Geräten
- Energiebedarf
- Zuverlässigkeit der Verbindung

Zusammengefasst lässt sich sagen, dass Routingprotokolle für MANETs vor allem eine große Skalierbarkeit im Hinblick auf eine große Anzahl von Geräten, Flexibilität und Effizienz im Hinblick auf Komplexität, Energieverbrauch und Speicherverbrauch erfordern. Diese werden mittlerweile intensiv erforscht, da Adhoc Netze im Zusammenhang mit dem 'Internet of Things' (IoT) oder auch dem 'Internet Of Everything' (IoE) zunehmend an Bedeutung gewinnen [TS11].

Im Folgenden sollen die Protokolle OLSR, AODV und CGSR kurz aufgezeigt und erklärt werden.

OLSR

Die Abkürzung OLSR steht für 'Optimized Link State Routing' und bezeichnet ein flaches, proaktives Linkstate-Protokoll. Proaktiv bedeutet in diesem Fall, dass in gewissen Zeitabständen automatisch Kontrollnachrichten ausgetauscht werden. Bei einem Linkstate-Protokoll sendet z.B. jeder Router den anderen Knoten im Netz den Zustand der Verbindung zu seinen Nachbarn.

Das OLSR-Routingprotokoll ist als RFC 3626 spezifiziert und erweitert normale Link-State-Protokolle, in dem die Komplexität jener Protokolle vereinfacht wird. Bei OLSR hat jeder Knoten einen Überblick über alle andere Knoten im Netzwerk und den Routen dort hin, somit kann jeder Knoten seine Routen mit dem 'Shortest Path Algorithm' eigenständig errechnen. Allerdings haben nicht alle Knoten die gleichen Bedeutungen und Aufgaben.

Um Nachbarknoten zu finden, werden sie mit einer 'Hello-Message' gesucht. Die Kontrollpakete werden in entsprechenden Zeitabständen erstellt und enthalten die Knotenadresse, eine Sequenznummer und die Nachbarknoten mit Distanzinformationen. Folglich enthält jeder Knoten die Distanzinformationen von allen anderen Knoten, so kann die Topologie erstellt und die Routen mit o.g. 'Shortest Path Algorithm' berechnet werden.

Das OLSR unterscheidet sich von anderen Link-State-Algorithmen, da es ein optimiertest Link-State-Routing verwendet. Es werden so genannte 'Multi-Point-Relays' ausgewählt und verteilt, was für ein effizienteres Routing sorgt. Die Relays

sind im Stande, die Kontrollnachrichten an die angeschlossenen Knoten weiterzuleiten.

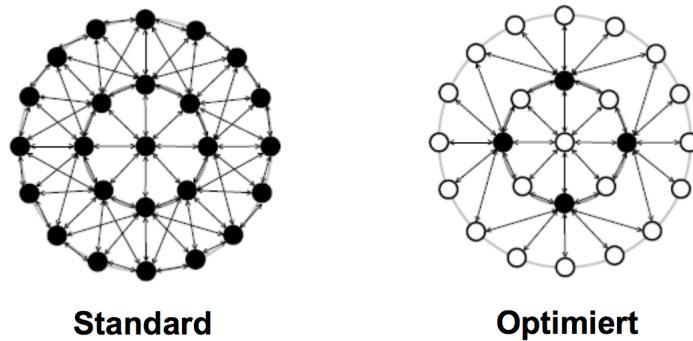


Abbildung 3.8: optimiertes LSR durch Multi-Point-Relays im OLSR [TS11]

AODV

Das Ad-hoc On-Demand Distance Vector Routingprotokoll ist ein flaches, reaktives Distanzvektorprotokoll. Reaktive Protokolle tauschen im Gegensatz zu proaktiven Protokollen nur Kontrollnachrichten aus, wenn neue Routen benutzt werden sollen. Sie benötigen weniger Bandbreite und sind dynamischer, allerdings besteht nur eine Teilkenntnis des Netzwerks, somit müssen die Routen anders berechnet werden.

AODV ist in RFC 3561 spezifiziert und ist für IPv4-Netze vorgesehen. Es erlaubt eine theoretisch unbegrenzte Anzahl von Geräten im Netzwerk. Bei AODV kennt jeder Knoten nur den 'Next Hop', also den nächsten Knoten und die Länge der Gesamtroute. Das Protokoll definiert insgesamt zwei Routingalgorithmen ‚Route Discovery‘ und ‚Route Maintenance‘.

Bei der 'Route Discovery' wird die Route aufgebaut, wozu zwei Nachrichtentypen benötigt werden. Es wird zunächst eine 'Route-Request-Nachricht' (RREQ) per Broadcast zum Ziel gesendet. Das Ziel antwortet wiederum mit einer 'Route-Reply-Nachricht' (RREP) per Unicast zurück an den Absender. Sollte eine bidirektionale Verbindung zwischen zwei Geräten herrschen, wird per Route Discovery auch eine bidirektionale Route aufgebaut.

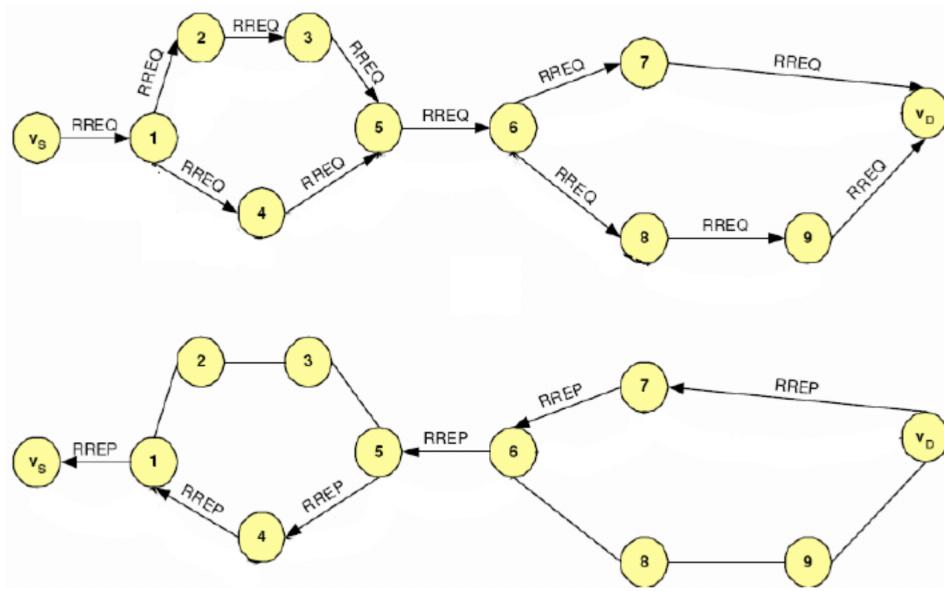


Abbildung 3.9: 'Route Discovery' im AODV-Protokoll [TS11]

Bei 'Route Maintenance' handelt es sich um die Verwaltung der Routen. Es wird eine 'Route-Error-Nachricht' (RRER) versendet, sobald eine defekte Route diagnostiziert wird. Diese Nachricht wird an alle Nachbarn versendet, so dass jeder Knoten über das Wegfallen der Route informiert wird.

CGSR

Beim Clusterhead-Gateway Switch Routing (CGSR) handelt es sich um ein hierarchisches Protokoll, was bedeutet, dass für verschiedene Knoten unterschiedliche Rollen vorgesehen sind.

Das Netzwerk wird bei CGSR in Cluster aufgeteilt, die sich allerdings teilweise überdecken müssen. Für jedes dieser Cluster wird ein 'Clusterhead' ernannt. Ein Knoten, welcher sich in zwei Clustern gleichzeitig befindet, wird als Gateway bezeichnet.

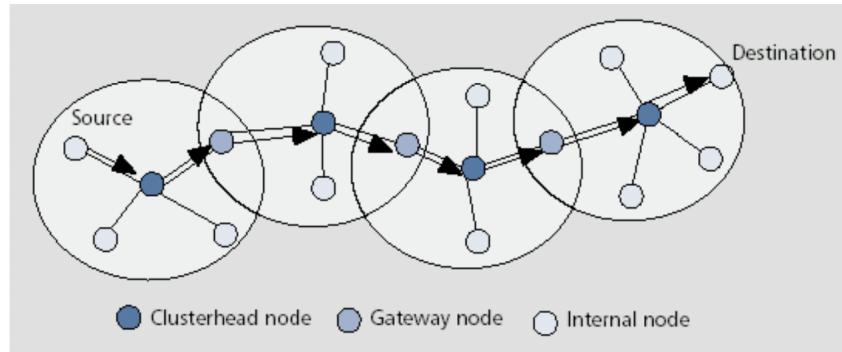


Abbildung 3.10: Clustering im CGSR-Protokoll [TS11]

Das Clusterhead-Gateway Switch Routing basiert auf einem Distanzvektor-Protokoll. Jeder Knoten im Netzwerk speichert sowohl eine Distanzvektor-Routingtabelle, als auch eine 'Cluster Member'-Tabelle. Die Distanzvektor-Tabelle enthält neben den normalen Routingeinträgen einen Routingeintrag zum Clusterhead jedes Clusters. In der 'Cluster Member'-Tabelle wird für jeden Knoten der Clusterhead gespeichert, damit klar ist, welcher Knoten zu welchem Cluster gehört.

Der große Vorteil von CGSR besteht in der signifikanten Reduzierung der Größe der Routingtabellen im Vergleich zu anderen Distanzvektorprotokollen, somit ist das CGSR für Ad-hoc Netzwerke dank seiner geringeren Komplexität gut geeignet.

3.3 IEEE 802.15.4

Das ‚Institute of Electrical and Electronics Engineers‘ (IEEE) definiert in seinem Standard 802.15.4 Protokolle für ein ‚Low Data Rate - Wireless Personal Area Network‘ (LR-WPAN). Die Art der Geräte, die für eine solche Art von Netz verwendet werden sollen, ist dabei nicht genauer spezifiziert. Sinnvoll ist der Einsatz vor allem bei Systemen wie Sensoren, Lichtquellen oder Schaltern, bei welchen eine niedrige Datenrate vollkommen ausreicht um miteinander zu kommunizieren. Grundsätzlich ist die Verwendung des Standards auch bei höherwertigen Geräten wie z.B. Handys oder anderen Multimediasystemen möglich, allerdings ist die Sinnhaftigkeit eines solchen Einsatzes fragwürdig, da in Hinsicht auf die geringe Datenübertragungsrate viele Übertragungs- und Kommunikationsfunktionen solcher Geräte wohl kaum realisierbar wären.

Zur Kommunikation mehrerer Teilnehmer in einem nach 802.15.4-standardisierten Netzes werden die o.g. Topologien ‚Stern‘ und ‚Peer-to-Peer‘ unterstützt. Sollte

die Spezifikation ‚ZigBee‘, welche den 802.15.4 Standard erweitert, verwendet werden, können darüber hinaus auch vermaschte Netze realisiert werden.

Der Standard sendet laut Definition festgelegtem Frequenzspektrum auf den lizenzenfreien Frequenzen 868-868,8 MHz (Europa), 902-928 MHz (Nordamerika, Australien) oder 2400 bis 2483,5 MHz (weltweit). Um die Frequenzen darüber hinaus zu spreizen, wird ein ‚Direct Sequence Spread Spectrum‘-Verfahren (DSSS) verwendet.

Um den Zugriff auf das Medium untereinander zu koordinieren, wird CSMA/CA („Carrier Sense Multiple Access/Collision Avoidance“) verwendet. Das bedeutet, dass, bevor ein Gerät mit einer Übertragung beginnt, überprüft wird, ob das Übertragungsmedium nicht bereits von einem anderen Endgerät genutzt wird. Ist das Medium nicht belegt, kann das Endgerät, welches das Medium überprüft hat, anfangen, seine Daten zu übertragen.

Der Standard erreicht Übertragungsraten zwischen 20 KB/s - 40 KB/s in den Frequenzbereichen von 868 MHz und 902-928 MHz, wohingegen im 2,4 GHz-Bereich Raten bis zu 250 KB/s realisiert werden können. Diese relativ geringen Datenraten zeigen bereits, dass der Standard nicht für eine Übertragung großer Daten konzipiert wurde. Es soll viel mehr eine energiesparende Übertragung geringer Datenmengen verwirklicht werden können [Hes05] [Ins11].

3.3.1 Komponenten

Man unterscheidet bei den Gerätetypen eines 802.15.4 WPAN Netzwerkes grundsätzlich zwischen so genannten ‚full-function devices‘ (FFD) und ‚reduced function devices‘ (RFD). FFDs sind Knoten, die den Standard in vollem Umfang unterstützen, während hingegen RFDs nur für Teile des Standards ausgelegt sind. Bei der Kommunikation ist zu beachten, dass FFDs sowohl mit anderen FFDs als auch mit RFDs Daten austauschen können, wohingegen RFDs nur mit FFDs kommunizieren können.

Jedes ‚full-function device‘ eines LR-WPANs agiert gleichzeitig als so genannter ‚PAN Coordinator‘, welcher Mechanismen zur Administration des Netzwerks bereitstellt. Dazu gehören zum Beispiel die Adressierung der einzelnen Knoten und die Verwaltung der Slots zur Datenübertragung bei Verwendung von ‚Slotted CSMA/CA‘. Außerdem ist der PAN Coordinator je nach Topologie für weitere Themen wie z.B. das Routing oder auch für Sicherheitsaspekte verantwortlich.

Ob ein Endgerät dem Netzwerk beitreten darf, entscheidet ebenfalls ein PAN Coordinator. Er kann selbiges bei Bedarf auch wieder aus dem Netzwerk ausgliedern. Die Verbindung zu übergeordneten Netzen wie z.B. dem Internet kann

ebenfalls über den PAN Coordinator hergestellt werden. Ob die Verbindung zu anderen Netzen mit Hilfe von 802.15.4 oder mit einem anderen Standard verwirklicht wird, obliegt der Entscheidung des Netzwerkadministrators [Hes05] [Ins11].

3.3.2 Unterstützte Topologien

Wie bereits erwähnt unterstützt IEEE 802.15.4 sowohl die Stern- als auch die Peer-to-Peer-Topologie. Bei der Stern-Topologie übernimmt der PAN Coordinator die Rolle des in 3.1.4. erwähnten zentralen Hubs, welcher die Kommunikation zwischen den restlichen Knoten regelt. Er baut Verbindungen zwischen Knoten auf, steuert dessen Kommunikation und beendet die Verbindung wieder.

Eine direkte Verbindung zwischen den Geräten existiert nur, falls mit der Peer-to-Peer-Topologie gearbeitet wird. Auch hier existiert ein PAN Coordinator, dieser ist jedoch nicht zwingend für die Kommunikation von zwei Knoten verantwortlich, da jeder Knoten frei mit seinen Nachbarn Daten austauschen darf. Auf Basis von Peer-to-Peer kann folglich auch ein vermaschtes Netz aufgebaut werden (Mesh-Topologie). Diese Art von Topologie wurde allerdings nicht direkt im Standard festgelegt, sondern in höhere Netzwerkschichten verlagert, so dass sie von genaueren Spezifikationen definiert werden muss (Bsp.: ZigBee).

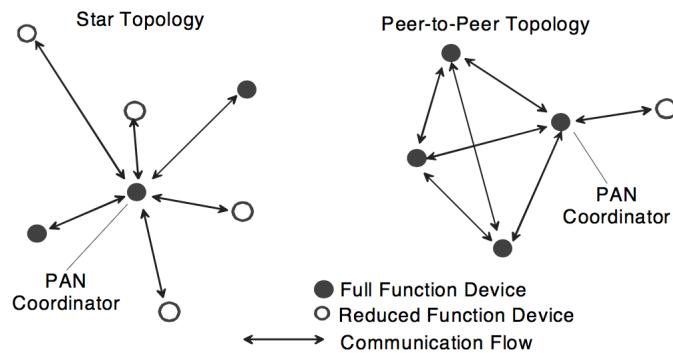


Abbildung 3.11: Topologien in IEEE 802.15.4 [Ins11]

Mit der Peer-to-Peer Topologie können außerdem mehrere Teilnetze zusammengefügt werden, so dass diese zu Clustern zusammengefasst werden. Ein Cluster besteht dabei aus einem PAN Coordinator sowie einigen zusätzlichen Geräten. Um clusterübergreifende Kommunikation zu ermöglichen, stellt der PAN Coordinator das Gateway in andere Cluster dar. Diese Funktion kann jedoch theoretisch jedes beliebige FFD übernehmen. Durch das Zusammenfügen mehrerer Cluster können große Netzwerke wie z.B. Sensornetze aufgebaut werden.

Der IEEE 802.15.4 Standard definiert selbst nur den Physical Layer und den

MAC Layer, somit müssen höhere Schichten wie z.B. Vermittlung, Sicherheit und Anwendung durch weitere Technologien konkreter realisiert werden (Bsp.: ZigBee) [Hes05] [Ins11].

3.3.3 Schichten

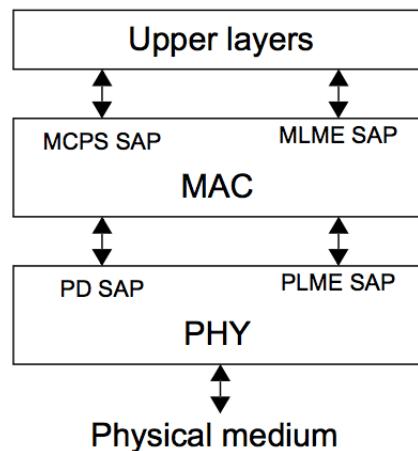


Abbildung 3.12: Definierte Schichten in IEEE 802.15.4 [Ins11]

IEEE 802.15.4 orientiert sich an den Schichten des ISO-OSI Referenzmodells und definiert dabei selbst nur den Physical Layer sowie den MAC Layer.

Die physikalische Schicht stellt die Kommunikation über so genannte ‚PHY Protocol Data Units‘ (PPDU) per Radio-Kanal bereit. Mit dem Physical Layer ist es weiterhin möglich, die Sende-Empfangseinheit zu aktivieren oder deaktivieren, Signale zu entdecken, die Qualität eines Kanals anzuzeigen und einen Kanal zu selektieren sowie zu belegen. Dies ist jedoch rein auf physikalischer Ebene, die logische Belegung und Selektion erfolgt im MAC Layer. Dieser Layer stellt das Beacon Management (s. 3.3.4) bereit, regelt den Kanalzugriff mit garantierten ‚Zeitschlitzten‘, übernimmt jegliche Fehlerbehandlung inklusive Bestätigung und realisiert gewisse sicherheitstechnische Aspekte. 802.15.4 lässt Verschlüsselung auf MAC Layer Ebene grundsätzlich zu, der dazugehörige Schlüsselaustausch findet jedoch auf darüberliegenden Schichten statt [Hes05] [Ins11].

3.3.4 Rahmenstruktur

Durch die in der Spezifikation festgelegte Superrahmenstruktur wird die Kommunikation zwischen den Endgeräten geregelt. Dabei werden vom PAN Coordinator in gewissen Zeitabständen so genannte Beacons versendet, welche eine eindeutig Identifikationskennung des absendenden Koordinators beinhalten und unter

anderem zur Netzsynchronisation und Eingliederung neuer Geräte in das Netz dienen. Der Beacon befindet sich im ersten der 16 Zeitschlitz des Superrahmens, wobei jener nicht zwingend vorhanden sein muss. Bei fehlendem Beacon wird bereits der erste Zeitschlitz zur normalen Datenübertragung genutzt.

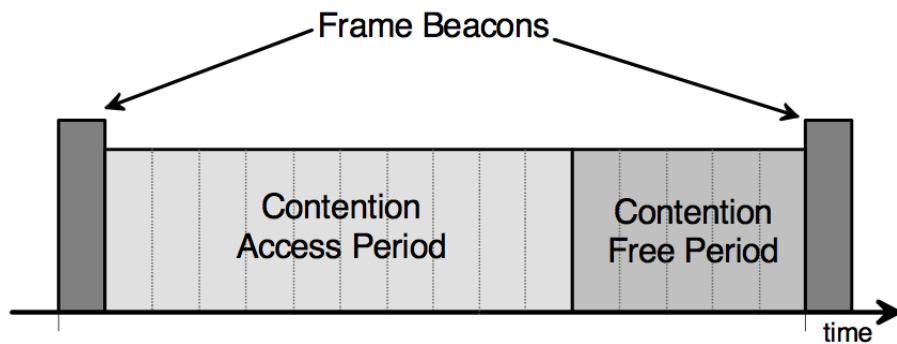


Abbildung 3.13: Superrahmen-Struktur eines LR-WPANs [Ins11]

Falls nötig, kann einem Endgerät ein garantierter Zeitschlitz („guaranteed time slot (GTS)“) vom PAN Coordinator zugesichert werden, in welchem es störungsfrei Daten übertragen kann. Falls diese Methode angewendet wird, findet die dazugehörige Kommunikation in der „Contention Free Period“ (CFP) statt. Es findet durch die fest zugeteilten Zeitschlitz ein eindeutiger Zugriff auf das Medium statt, weswegen es in dieser Periode zu keiner Konkurrenz beim Zugriff auf das Medium kommen kann. Neben der CFP gibt es ebenfalls eine „Contention Access Period“ (CAP), in der alle angeschlossenen Geräte unter Benutzung von CSMA/CA Daten übertragen können. Zusätzlich ist es neuen Geräten in dieser Zeit gestattet, sich in das Netzwerk zu integrieren. Falls vom PAN Coordinator keine garantierten Zeitschlitz vergeben werden, gehören alle Zeitschlitz zur CAP [Hes05] [Ins11].

3.3.5 Kommunikation

In der Spezifikation werden drei Arten von Kommunikationsmöglichkeiten erwähnt. Die ersten beiden Arten betreffen die Kommunikation zwischen PAN Coordinator mit einem Endgerät und umgekehrt, wobei der dritte Fall von einer direkten Kommunikation von Endgerät zu einem anderen Endgerät spricht. Der Informationsaustausch findet mit Hilfe so genannter Frames statt, von welchen wiederum vier unterschiedliche Arten existieren:

- Beacon Frames, welche wie oben erwähnt Informationen des PAN Coordinators enthalten
- Data Frames, welche Daten übertragen
- Acknowledgement Frames, welche den Empfang von anderen Frames bestätigen
- MAC-Command Frames

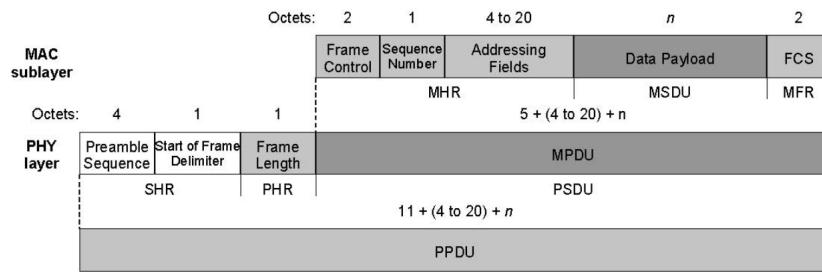


Abbildung 3.14: Framestruktur eines Data-Frames im MAC-Layer [Ins11]

Am Anfang eines Frames steht stets ein 6 Byte-Header. Danach folgen den Anforderungen an den Frametyp entsprechend eine Packet Data Unit (PDU), welche neben einer Identifikation des Frames ebenfalls Nutzdaten beinhalten kann. Die im Frame enthaltene Payload wird in Anlehnung an das ISO/OSI-Schichtenmodell entsprechend der einzelnen Schichten gekapselt. Somit wird garantiert, dass beispielsweise der Physical Layer ausschließlich seine Header einsehen kann. Die Adressierung an andere Applikationen höherliegender Schichten ist für den Physical Layer nicht ersichtlich. [Hes05] [Ins11].

3.3.6 Medienzugriff

Ein PAN Coordinator ist nicht dazu verpflichtet, Beacons auszusenden. Die Entscheidung über die Verwendung von Beacons im Netzwerk obliegt dem Netzwerkadministrator, folglich muss der Medienzugriff der Endgeräte bei Verwendung oder Nichtverwendung gesondert betrachtet werden.

Falls keine Beacons zum Einsatz kommen, wird als Medienzugriffsverfahren ‚Unslotted CSMA/CA‘ benutzt. Bei diesem Verfahren wartet ein Gerät eine zufällige Zeit und überprüft nach Ablauf der Zeit den Zustand des Mediums. Sollte das Medium frei sein, wartet das Endgerät eine weiteres Mal eine so genannte ‚Back-Off‘-Zeit ab und beginnt anschließend, bei freiem Medium seine Daten zu übertragen. Ist das Medium bereits belegt, wartet das Gerät eine weitere zufällige Zeitspanne ab, nach welcher das Medium erneut auf seinen Zustand geprüft wird.

Beim Einsatz von Beaconframes wird hingegen ‚Slotted CSMA/CA‘ verwendet. Das Verfahren ist dem ‚Unslotted‘-Verfahren grundsätzlich sehr ähnlich, allerdings wird die o.g. ‚Back-Off‘-Zeit so synchronisiert, dass das Gerät die Datenübertragung genau in seinem zugewiesenen Zeitschlitz beginnt und durchführt. Das Versenden von Acknowledgement Frames ist im Standard definiert, ihre Benutzung ist allerdings optional. Sie werden ohne Benutzung von CSMA/CA übertragen, da sie lediglich eine Größe von 11 Byte besitzen. Innerhalb der Datenpakete kann mit Hilfe einer CRC-Prüfziffer die Integrität nach der Übertragung überprüft werden, um z.B. fehlerhafte Übertragungen zu diagnostizieren [Hes05] [Ins11].

3.3.7 Sicherheitsaspekte

Im Dokument des IEEE 802.15.4 Standards werden ebenfalls einige Sicherheitsaspekte aufgelistet. Erwähnenswert ist hier die Möglichkeit, den Datentransfer auf MAC Layer-Ebene unter Verwendung von AES („Advanced Encryption Standard“) zu verschlüsseln. Wie jedoch bereits erwähnt muss der Austausch der Schlüssel auf höheren Schichten stattfinden, da dieser nicht im Standard selbst definiert ist [Hes05] [Ins11].

3.4 SunSPOT

Die Firma Oracle besitzt im Rahmen seiner Java-Technologie eine Vormachtstellung im Bereich der Smartphones. Auf der Welt sind schätzungsweise über eine Milliarde Smartphones mit der Java-Technologie lizenziert. [Hor08]

Ziel von Oracle ist es, auch in den zukunftsnahen Technologien mit ihrer Programmiersprache Java auszustatten und diese Produkte zu etablieren.

Ein erster Schritt in diese Richtung ist das von Oracle entwickelte ‚SunSPOT‘-Sensornetzwerk. SunSPOT bedeutet ‚Sun Small Programmable Object Technology‘ und ist eine Plattform für Java-basierte drahtlose Sensornetzwerke. Sie bestätigt den Trend, dass in immer kleiner werdenden Geräten zunehmend leistungsfähigere Technologien eingesetzt werden. Dabei ist wichtig, dass jene Geräte, am Besten drahtlos, miteinander kommunizieren können und jederzeit von überall auf der Welt steuerbar bleiben. Das SunSPOT Starter Paket besteht aus einer Basisstation und 2 Sensoren.

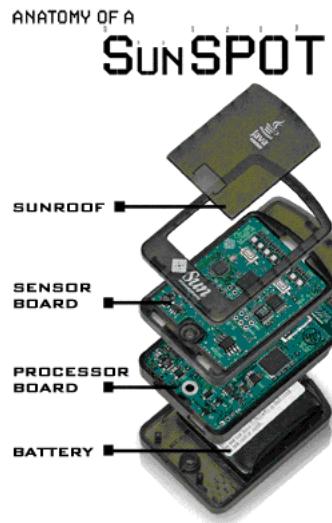


Abbildung 3.15: Anatomie eines Standard SunSPOT-Sensors [Uni]

Die Hardware der SunSPOT-Sensoren ist modular aufgebaut. Das bedeutet, dass man die verfügbaren Boards frei nach Belieben aufeinander stecken und somit verbinden kann. Dabei können maximal bis zu 3 Boards + Stromversorgung miteinander verknüpft werden. [Hor08]

3.4.1 Gerätearten

Man unterscheidet bei den SunSPOT-Geräten grundsätzlich zwischen dem Basisstation SPOT und den Free Range SPOTS. Der sogenannte 'Basestation SPOT' hat ein eSPOT Prozessor-Board ohne eigene Batterie. Er wird mit Hilfe des USB-Ports mit Strom versorgt. Die Station dient grundsätzlich als Schnittstelle zwischen Free Range SPOTS und dem PC (der Workstation) auf Basis von IEEE 802.15.4. Die Free Range SPOTS enthalten ebenfalls ein Prozessor-Board (auch Mainboard genannt), hinzu kommt ein wiederaufladbarer Li-Ion Akkumulator. Des Weiteren enthalten die SPOTS in der Standardauslieferung das eDemo Sensor Board. Alle Boards, welche man dank modularem Aufbau hinzustecken kann, sind separat erhältlich und werden im Folgenden näher erläutert.

3.4.2 Verfügbare Boards

Das sogenannte eSPOT Prozessor-Board besitzt in der aktuellsten Version eine 400 MHz 32-bit ARM CPU von Atmel, zusammen mit einem Flashspeicher von

8 Megabytes und einem Megabyte SRAM Hauptspeicher. Weiterhin ist es ausgestattet mit einem Radio Transceiver basierend auf IEEE 802.15.4 und einer USB 2.0 - Full Speed Schnittstelle. Der im SunSPOT integrierte Akkumulator hat eine Leistungsfähigkeit von 770mAh. Der maximale Energieverbrauch liegt zwischen 40-100 mA, abhängig von der Nutzung der integrierten LEDs, des Transceivers und anderer angeschlossener Geräte. [Hor08] [Ora10b]

Der Free Range SPOT wird dazu standardmäßig mit dem eDemo Sensor Board ausgeliefert. Es besitzt in der aktuellen Version einen 2G/4G/8G 3-Achsen-Beschleunigungssensor, einen Lichtsensor, 8 RGB 24bit LEDs, einen Infrarot-Sender & Empfänger, ein kleiner Lautsprecher, 2 Knopfschalter, 4 analoge Eingänge, 4 I/O Pins, diverse weitere I²C- und USART-Interfaces, einen EEPROM und 4 100mA Ausgangspins, mit denen es möglich ist, den SunSPOT-Sensor z.b. an weitere Lautsprecher oder andere Geräte anzuschließen. [Hor08] [Ora10a]

Weitere Boards, welche man nach Bedarf dazustecken kann, sind das eProto-Board, ein Board welches direkte Zugriffe auf das Prozessorboard ermöglicht und einen SD-Kartenslot besitzt, damit man die Daten dauerhaft speichern kann, das eSerial Board zum Verbinden via RS232 und das eFlash SD-Kartenleser Board. [Hor08]

Kapitel 4

Praktische Arbeiten

Nach dem Erarbeiten der theoretischen Grundlagen werden nun praktische Arbeiten mit Oracle SunSPOT realisiert . In den kommenden Abschnitten wird die Inbetriebnahme, die Installation des SunSPOT-SDKs sowie eine kurze Übung erläutert, danach findet die eigentliche Implementierung der geplanten Raumüberwachung statt.

4.1 Erste Schritte

Um den Umgang mit den SunSPOTS zu erlernen und ihre Software und Hardware kennen zu lernen, bietet Oracle auf der Website eine Anleitung inklusive Übungen an, welche sukzessive abgearbeitet werden kann. Die Anleitung umfasst die Installation der Software sowie das Ausführen von Testprogrammen, welche die Fähigkeiten des SunSPOTS demonstrieren.

4.1.1 Installation

Zur Installation der Verwaltungsapplikation 'SunSPOT Manager' und dem SunSPOT SDK, mit welchem man die Programm für die SunSPOTS schreibt, startet man das 'SunSPOT Manager Java Webstart' Programm. Das Webstart-Programm lädt darauf hin den Manager und das SDK herunter und installiert es auf der Workstation. Zur einwandfreien Benutzung des Managers müssen folgende Tools vorhanden sein:

- ein aktuelles Java Development Kit
- Apache ANT
- optional: NetBeans IDE

Das Installationsprogramm weist auf ein Fehlen oben genannter Tools hin und installiert diese bei Bedarf nach. Sollte dies nicht automatisch durch das Programm geschehen, können diese Programme auch manuell nachinstalliert werden. Während der Installation wird nach dem SunSPOT SDK gefragt, welches installiert werden soll. Je nach Version der SPOTS ist hier die entsprechende Version auszuwählen.

Normalerweise wird beim Anschließen per USB der entsprechende Treiber für die SPOTS automatisch installiert. Bei aktuellen 64-bit Windows-Versionen (7/8/8.1) kommt es allerdings zu Schwierigkeiten. Eine spezielle Treiber-Datei wurde am 7. Mai 2010 durch Bob Alkire im Oracle-Blog veröffentlicht und kann dort heruntergeladen werden. Mit ihr ist es möglich, die SPOTS auch mit aktuellen 64-bit Windows-Plattformen zu verwenden [Alk10].

Sobald der Manager und alle zugehörigen Treiber erfolgreich installiert wurden, können Programme mit Hilfe von NetBeans oder einer anderen IDE geschrieben und auf den SPOT übertragen werden.

4.1.2 Übung 1 - Flashlight

In der ersten Übung soll das erste Beispielprogramm 'Flashlight' auf den SPOT übertragen, dort ausgeführt und die zugehörigen Programmausgaben (Konsolen- & Debugausgaben) betrachtet werden.

Nach dem erfolgreichen Übertragen des Programms blinken die LEDs des SPOTS in kurzem Abstand, vorausgesetzt, es wird ein bestimmter Helligkeitswert, ausgelesen vom Lichtsensor, nicht überschritten. Mit Hilfe des rechten Knopfschalters auf dem Sensorboard kann außerdem die Farbe der LEDs geändert werden. Die Konsolenausgabe des Programms besteht aus dem momentanen Helligkeitswert, der vom Lichtsensor erfasst wird.



Abbildung 4.1: Leuchtende LEDs des 'Flashlight'-Programms

Für die geplante Raumüberwachung ist diese Übung insofern hilfreich, als das hier der Umgang mit den LEDs und deren Verhalten auf äußere Einflüsse gezeigt wurde. In der späteren Implementierungsphase der Raumüberwachung sollen LEDs anzeigen, ob eine Bewegung und somit ein illegales Eindringen in den Raum stattfand.

4.2 Implementierung einer Raumüberwachung

4.2.1 Idee

Nicht nur für Museen, Villen oder Betriebe ist ein Sicherheitssystem sinnvoll, denn besonders in privaten Wohnungen und Häusern wird gerne eingebrochen.

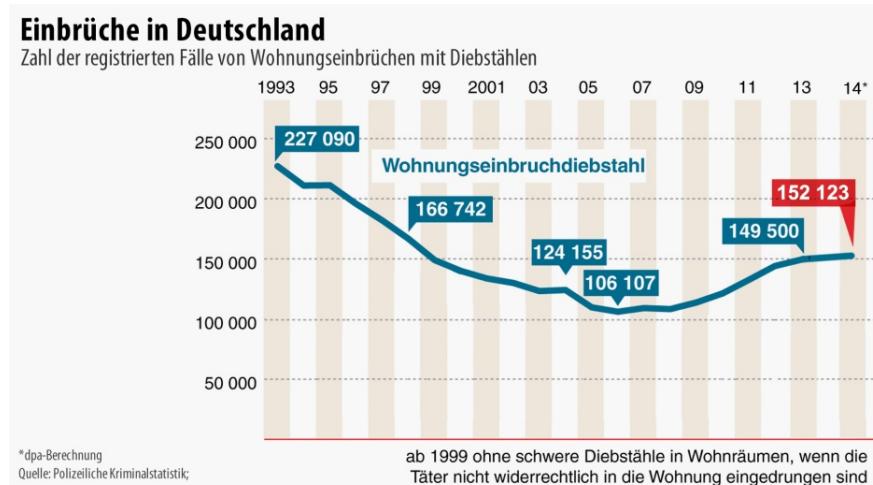


Abbildung 4.2: Einbruchsstatistik in Deutschland [Tru15]

Allein im Jahr 2014 wurde über 150.000 mal in Deutschland ein Einbruch mit Diebstahl gemeldet. Die Aufklärungsrate der Verbrechen hingegen ist jedoch sehr gering. Um sich vor solchen Straftaten zu schützen werden Überwachungsanlagen immer beliebter.

Im Rahmen dieser Studienarbeit soll eine Raumüberwachung entwickelt werden, die mithilfe von untereinander vernetzten Sensoren Einbrüche frühzeitig erkennt, und an den Besitzer meldet. Die Überwachung soll direkt an Fenster und Türen stattfinden, mithilfe von Vibrationssensoren, Mikrofonen oder Kameras sollen außergewöhnliche Vorkommnisse ausgewertet werden. Bei einem Einbruch kann dies dem Besitzer über das Internet mitgeteilt werden.

4.2.2 Umsetzung

Im Rahmen der Studienarbeit wurden mit der NetBeans IDE und der Programmiersprache Java zwei Applikationen erstellt, welche einen ersten Ansatz für eine Einbruchsicherung realisieren. Sie sind beliebig erweiterbar, so dass diese als Grundlage für eine komplexere Raumüberwachung dienen können. Die Gesamtimplementation wurde in zwei Teilapplikationen aufgeteilt. Die erste Applikation ist die ‚Desktop-Applikation‘, welche auf dem Rechner läuft und die Daten vom SunSPOT-Sensor empfängt. Auf dem SunSPOT läuft der zweite Teil der Implementation, welcher Bewegungen des Sensors feststellt und diese an eine SunSPOT-Basisstation weiterleitet. In den folgenden Abschnitten wird die genauere Funktionsweise der zwei Teilapplikationen behandelt.

Desktop-Applikation

Die Desktop-Applikation der praktischen Arbeit umfasst knapp 100 Zeilen und dient zum Aufbau einer Radiogram-Verbindung zwischen SunSPOT und Basisstation sowie zum Empfang der vom SPOT gesendeten Daten.

Listing 4.1: Ausschnitt aus der setup()-Methode

```

1 private void setup() {
2     fr = new JFrame("Einbruchsicherung");
3     status = new JTextArea();
4     JScrollPane sp = new JScrollPane(status);
5     fr.add(sp);
6     fr.setSize(360, 200);
7     fr.validate();
8     fr.setVisible(true);

```

9 }

Im ersten Schritt wird die Methode ‚setup()‘ definiert, welche zur besseren Visualisierung der Informationen einen JFrame erstellt, in welchen nachher eingetragen werden kann, dass der Sensor eine Bewegung erfahren hat. Sie wird zum Start des Programms aufgerufen, damit das Fenster auch entsprechend erstellt wird.

Des weiteren ist eine Methode ‚run()‘ vorhanden, welche direkt nach der ‚setup()‘-Prozedur aufgerufen wird. Diese Funktion kümmert sich sowohl um das Öffnen eines serverseitigen Ports als auch um den Empfang von Datenpaketen, welche an diesen Port versendet werden.

Um Daten vom Sensor empfangen zu können, muss in der Desktop-Applikation auf einem selbst festgelegten Port nach Verbindungen ‚gelauscht‘ werden.

Listing 4.2: Öffnen des serverseitigen Ports

```

1 try {
2     // Oeffnen einer Radiogram-Verbindung auf der
3     // Desktop-Seite
4     // um Daten von Sensoren empfangen zu koennen
5     rCon = (RadiogramConnection)
6         Connector.open("radiogram://:" + HOST_PORT);
7     dg = (Radiogram)
8     rCon.newDatagram(rCon.getMaximumLength());
9 }
10
11 catch (Exception e) {
12     System.err.println("setUp caught " +
13         e.getMessage());
14     throw e;
15 }
```

Der oben gezeigte Codeausschnitt eröffnet einen serverseitigen Port mit Hilfe der vorher festgelegten HOST_PORT Variable und lauscht nun fortan auf diesem für Pakete, die vom SunSPOT an die Basisstation versendet werden.

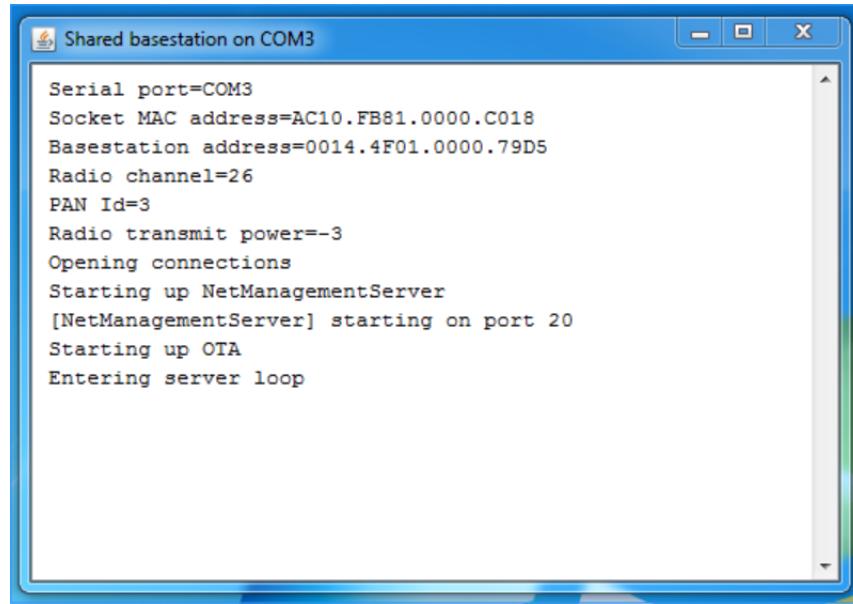


Abbildung 4.3: Logdaten der Basisstation bei Servererstellung

Der zweite Teil der „run()“-Methode besteht aus einer while-Schleife, welches auf ankommende Pakete wartet und diese entgegennimmt. Nur wenn der SPOT eine Bewegung erfährt, sendet er ein Paket an die Basisstation. So kann die Desktop-Applikation sicher sein, dass immer, wenn ein Paket gesendet wird, eine Bewegung stattfand.

Listing 4.3: Auswerten des empfangenen Paketes

```

1  while (true) {
2      try {
3          // Empfangen und Lesen des Pakets
4          rCon.receive(dg);
5          long time = dg.readLong();
6          System.out.println(time);
7          status.append("Bewegung festgestellt!\n");
8      }
9      catch (Exception e) {
10         System.err.println("Exception " + e + " beim
11             Lesen der Daten.");
12         throw e;
13     }
}

```

Innerhalb des übermittelten Paketes befindet sich die aktuelle Uhrzeit, um die Bewegung nachher genau zeitlich einordnen zu können. Sobald die Desktop-Applikation ein Paket empfängt, gibt sie auf dem Bildschirm innerhalb des erstellten Frames den Text 'Bewegung festgestellt!' aus. So wird dem Nutzer unmissverständlich klar gemacht, dass der Sensor in Bewegung war.

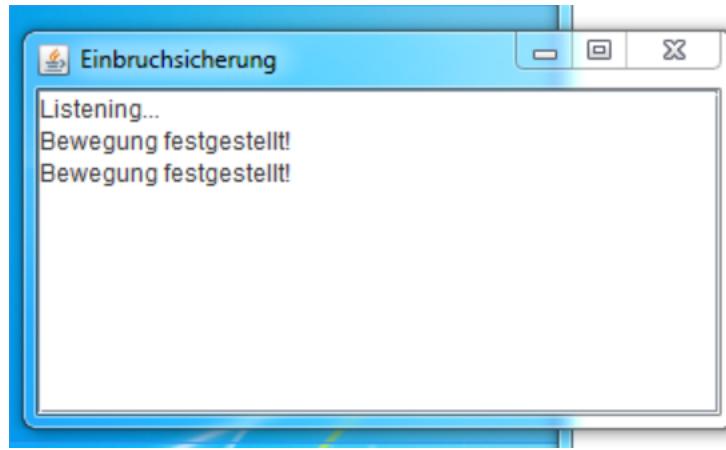


Abbildung 4.4: JFrame mit textueller Ausgabe

Innerhalb der Konsole werden neben Debugausgaben zur Basisstation und die Liste eröffneter Ports der Zeitstempel, welcher mit der Bewegung vom SunSPOT übermittelt wird, ausgegeben.

```

ant -f "C:\\\\Users\\\\Nicolai Staege\\\\Desktop\\\\SunSPOT\\\\SendDataDemo\\\\Studi_Sensor_Desktop" host-run
init:
Checking for shared basestation...
Looking for basestation on host: 172.16.251.129
Shared basestation result: false
Please wait while connected Sun SPOTS are examined...
Please wait while connected Sun SPOTS are examined...

Using Sun SPOT basestation on port COM3
Deleting: C:\\SunSPOT\\dk\\temp\\spotselector-1176218597
Starting new shared basestation...
host-compile:
[radiogram] Adding: Server on port 8
[radiogram] Adding: Server on port 8
[radiogram] Adding: Server on port 67
[radiogram] Adding: Server on port 67
1430843447638
1430843447638
1430843449991
1430843449991

```

Abbildung 4.5: Konsolenausgabe mit Timestamp

SPOT-Applikation

Der Desktop-Applikation gegenüber steht die Applikation, welche auf dem SunSPOT betrieben wird. Diese erkennt Bewegungen des SPOTS und sendet diese über eine Radiogram-Verbindung an die auf dem selben Port lauschende Basisstation.

Listing 4.4: Aufbau der Verbindung auf Port 67

```

1 try {
2     // Oeffne Verbindung auf Port 67 - Port ist
3     // generell beliebig
4     // Desktop Applikation muss allerdings den
5     // gleichen Port benutzen
6     rCon = (RadiogramConnection)
7         Connector.open("radiogram://broadcast:" +
8             HOST_PORT);
9     dg = rCon.newDatagram(50);    // Testsenden
10 }
11 catch (Exception e) {
12     System.err.println("Exception " + e + " beim
13         Verbindungsaufbau.");
14     notifyDestroyed();
15 }
```

Im oben aufgelisteten Codeausschnitt erstellt der SunSPOT eine Radiogram-Connection als Broadcast auf Port 67. Um die Verbindung zu testen, wird ein kleines Datagram Package erstellt und versendet. Falls die Verbindung fehlschlagen sollte, wird eine Fehlermeldung ausgegeben.

Listing 4.5: Schleife zur Detektion von Bewegung

```

1 while (true) {
2     try {
3         if (isMoving()) {
4             // Uebermittle die aktuelle Zeit
5             long now = System.currentTimeMillis();
6
7             // LED leuchtet, um zu zeigen, dass sich der
8             // Sensor bewegt hat
9             led.setRGB(255, 255, 255);
10            led.setOn();
11            Utils.sleep(50);
12            led.setOff();
13
14             // Zeit wird ins Package geschrieben, Package
15             // wird verschickt
```

```

14     dg.reset();
15     dg.writeLong(now);
16     rCon.send(dg);
17 }
18 }
19 catch (Exception e) {
20     System.err.println("Exception " + e + " während
21         des Senden eines Paketes.");
22 }

```

In einer Endlosschleife wird festgestellt, ob sich der Sensor bewegt hat. Dies wird mit Hilfe der Funktion 'isMoving()' realisiert. Falls eine Bewegung stattfindet, wird die aktuelle Systemzeit ermittelt und in ein Datagram-Paket geschrieben. Eine einzelne LED am SPOT leuchtet, um die erfahrene Bewegung anzudeuten. Das Paket wird daraufhin über die aufgebaute Verbindung versendet. Sollte dieses Senden fehlschlagen, wird ebenfalls eine Fehlermeldung ausgegeben.

Listing 4.6: Methode zur Bewegungserkennung

```

1 protected boolean isMoving() throws IOException {
2     IAccelerometer3D accel = (IAccelerometer3D)
3         Resources.lookup(IAccelerometer3D.class);
4     return accel.getAccelZ() < 0 ||
5         accel.getAccelZ() > 0.1;
6 }

```

Die Bewegung wird mittels des eingebauten Beschleunigungssensor ermittelt. Erreicht die erfahrene Beschleunigung einen gewissen Schwellwert, gibt die oben gezeigte Subroutine den Wert 'true' zurück und zeigt somit an, dass der Sensor entsprechend beschleunigt wurde.

Sobald sowohl Desktop-Applikation als auch SPOT-Applikation gestartet wurden, beginnt das Zusammenspiel der beiden Teilimplementierungen. Die Desktop-Applikation wartet darauf, Pakete, welche die SPOT-Applikation versendet, zu empfangen. So wird bei jeder Bewegung die Desktop-Applikation benachrichtigt. Eine einfache Einbruchserkennung ist hiermit realisiert.

4.3 Erweiterungsmöglichkeiten

Das entwickelte System kann durch mehrere Komponenten erweitert werden. Hierbei handelt es sich um drei Hauptkategorien: Die Visualisierung, weitere Sensoren und eine Benachrichtigung des Wohnungsbesitzers.

4.3.1 Visualisierung Webseite

Durch die Kombination der SunSpot-Sensoren mit einem kleinen Computer lässt sich das System mit einer interaktiven Visualisierung erweitern. Hierbei erhält der Computer von der SunSPOT-Basisstation Informationen über die Umgebung. Dies können beispielsweise die aufgezeichneten Vibrationen der Sensoren sein. Auf dem Computer wird ein Webserver eingerichtet, der mit den erhaltenen Informationen eine Webseite erstellt, auf die der Wohnungsbesitzer zugreifen kann. Hier erhält er einen Überblick über den Status seiner Wohnung und besitzt die Möglichkeit mit dem Sicherheitssystem zu interagieren.

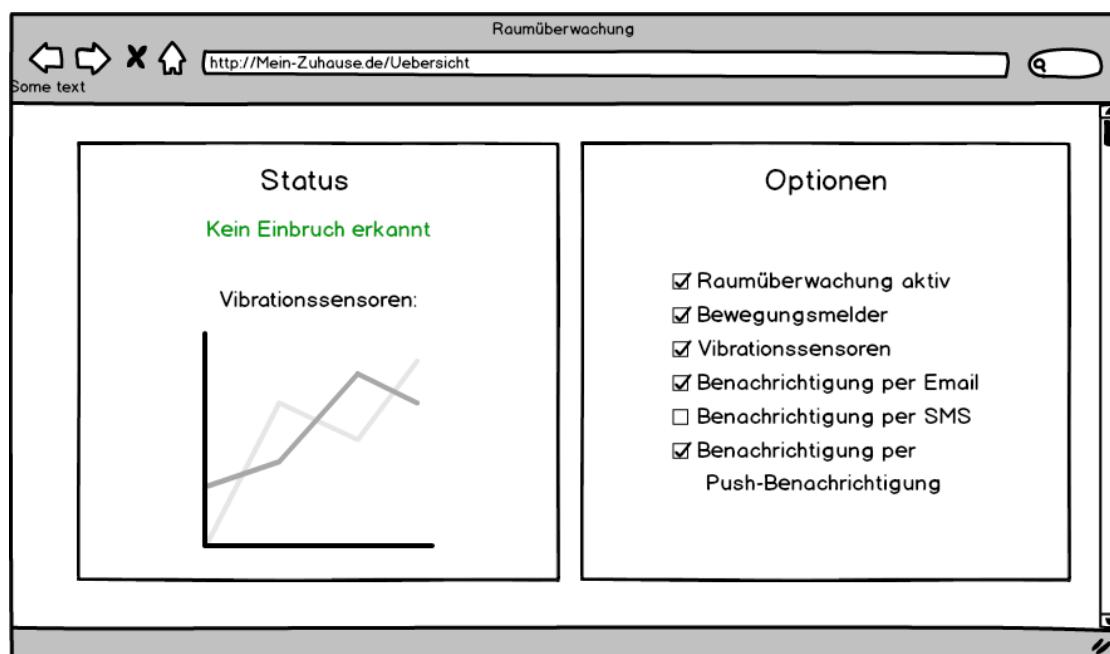


Abbildung 4.6: Mockup einer Webseite

Die Umsetzung dieser Erweiterung ist mithilfe eines Raspberry Pis möglich. Dieser besitzt einen USB-Anschluss, über den die Basisstation der SunSpots angeschlossen werden kann. Zusätzlich kann sich der Raspberry Pi über Ethernet oder WLAN mit dem Internet verbinden.

Mithilfe der Debian-Linux Distribution Raspbian kann ein Tomcat- oder Apache

Webserver aufgesetzt werden, der die Webseite des Überwachungssystems bereitstellt. Auf der Webseite kann dem Besitzer das Aktivieren, sowie Deaktivieren der Überwachung oder einzelner Sensoren ermöglicht werden.

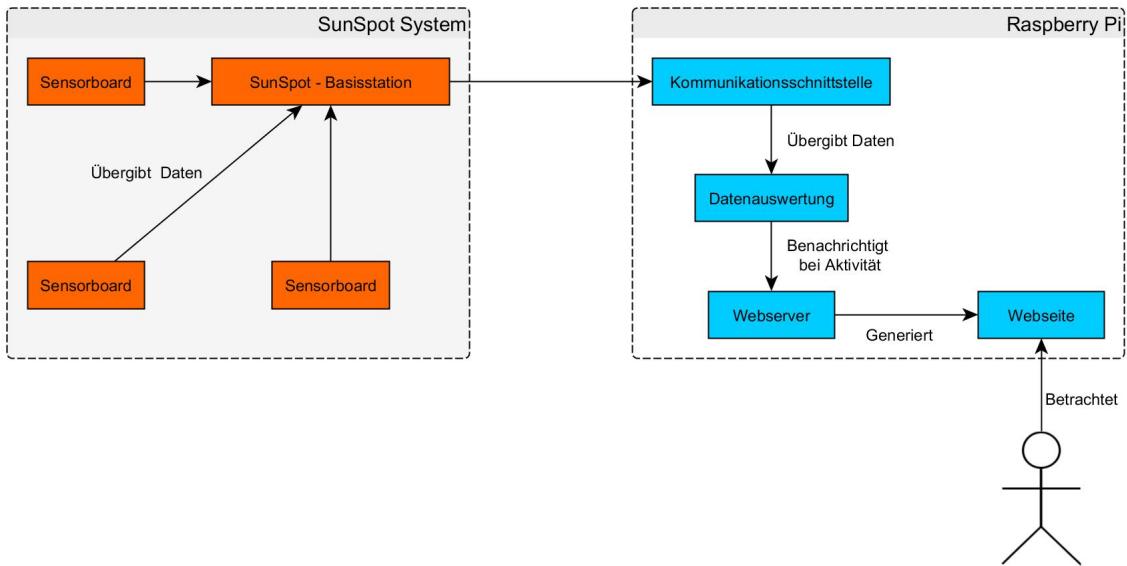


Abbildung 4.7: Visualisierung mithilfe eines Raspberrys und Webserver

4.3.2 Erweiterung durch verschiedene Sensoren

Eine Erweiterungsmöglichkeit ist die Ergänzung verschiedener Sensoren. Durch ein Mikrofon können ungewöhnliche Geräusche aufgezeichnet werden. Diese können von Einbruchsversuchen an einer Tür oder durch eine zerbrechende Scheibe entstehen. Eine großer Vorteil eines Mikrofons ist, das auch jegliche Kommunikation der Einbrecher aufgezeichnet werden kann. So kann ebenfalls entschieden werden, ob es sich um einen Einbrecher handelt, oder ob Verwandtschaft mit einem Ersatzschlüssel die Wohnung betreten hat.

Durch Bewegungsmelder werden häufig Beleuchtungen aktiviert, in diesem Szenario kann er jedoch zur Einbruchserkennung verwendet werden: Falls sich etwas bewegt, jedoch niemand zu Hause sein sollte, wird ein Alarm ausgelöst. Da die Überwachung durch Ultraschall oder Infrarot geschieht, kann es im Raum auch dunkel sein.

Eine weitere Möglichkeit ist das Ergänzen einer Kamera. Diese kann entweder als Bewegungsmelder eingesetzt werden, oder Bilder und Videos der Wohnung aufnehmen. Im Falle eines Einbruchs kann so der Täter direkt fotografiert werden. Dies ermöglicht eine gezielte strafrechtliche Verfolgung. Die Kamera kann jedoch

auch auf Wunsch des Wohnungsbesitzers Bild- oder Videomaterial aufzeichnen, das über das Internet einsehbar ist. Auf diese Weise kann man einen Blick in die Wohnung werfen, ohne selbst anwesend zu sein.

Die verschiedenen Sensoren können beispielsweise an einen Raspberry Pi angeschlossen werden, der die Eingangssignale überwacht und bei ungewöhnlichen Werten anschlägt.

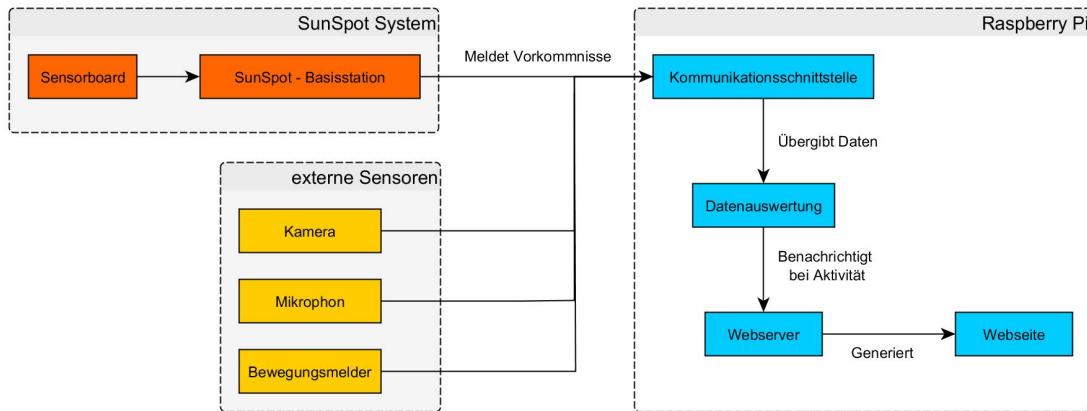


Abbildung 4.8: Aufbau eines Sensornetzes mithilfe eines Raspberry Pis

4.3.3 Benachrichtigung des Benutzers

Mit der Verbreitung von Smartphones und mobilem Internet geht permanente Erreichbarkeit einher. Doch solange man nicht zu Hause ist, möchte man gerne wissen, was in den eigenen vier Wänden passiert. Aus diesem Grund soll das Überwachungssystem im Falle eines Einbruchs den Besitzer alarmieren. Dies kann über eine Vielzahl von Möglichkeiten geschehen.

Informationskanal	Unverzögert	Übertragung von Medien
SMS	Ja	Nein
Email	Nein	Ja
Push-Notifications	Ja	Nein

Tabelle 4.1: Auflistung der mobil abrufbaren Kommunikationskanäle

Die klassische Email ist ein Kommunikationskanal, der auch unterwegs abgerufen werden kann. Der Nachteil hiervon ist jedoch, dass das Abrufen meist zeitverzögert stattfindet und die Reaktion auf eine Einbruchsmeldung möglicherweise zu spät stattfindet. Sie bietet jedoch einige Vorteile: Man kann sie beinahe überall abrufen, da die Email ein universelles Kommunikationsmittel darstellt. Zudem

können sowohl Text als auch Bilder oder Videos in einer Email verschickt werden.

Die SMS ist im Vergleich zur Email unverzögert. Direkt nach dem Eintreten des Unfalls wird der Benutzer informiert und kann reagieren. Technologie-bedingt können in einer SMS keine Bilder oder Videos übertragen werden, weshalb sie nicht das volle Potential eines Smartphones ausschöpft.



Abbildung 4.9: Mockup einer Push-Benachrichtigung

Die Push-Benachrichtigung ist ein zentraler applikationsübergreifender Kommunikationskanal mobiler Geräte. Dabei wird die Kommunikation nicht vom Client (Smartphone / Tablet) initiiert, sondern von einem Server des Herstellers. Auf diese Weise ist es möglich, dem Besitzer des Sicherheitssystems ohne sein Zutun eine Nachricht zu übermitteln. Diese kann einen Text oder Link enthalten. Durch die Verwendung von Push-Benachrichtigungen kann das Smartphone des Besitzers angewiesen werden, eine App des Sicherheitssystems zu öffnen und die aktuellen Meldungen anzuzeigen. Ebenfalls ist es denkbar, dass der Benutzer nach dem Empfang der Push-Benachrichtigung automatisch eine vom Sicherheitssystem erstellte Webseite öffnet, auf der er Bilder oder Videos des Einbruchs ansehen kann.

Da die Push-Benachrichtigung die modernste Form der Benachrichtigungen darstellt, ist die Verbreitung noch relativ gering. Mit modernen Mitteln ist sie jedoch leicht zu implementieren. Da die Push-Systeme der verschiedenen mobilen Betriebssysteme technologisch unterschiedlich aufgebaut sind, sollte eine Plattformübergreifende Lösung eingesetzt werden. Hierfür gibt es Dienste wie 'Pushover', die für mehrere Smartphone Betriebssysteme Anwendungen bereitstellen, die Push-Notifications empfangen können. Mithilfe einer serverseitigen HTTP API ermöglicht Pushover das Senden von Push-Benachrichtigungen an zuvor angemeldete Gruppen oder Einzelgeräte. Um Nachrichten empfangen zu können müssen die Endgeräte die Pushover Application installiert haben, die für Android-, iOS- und Desktopcomputer erhältlich ist.

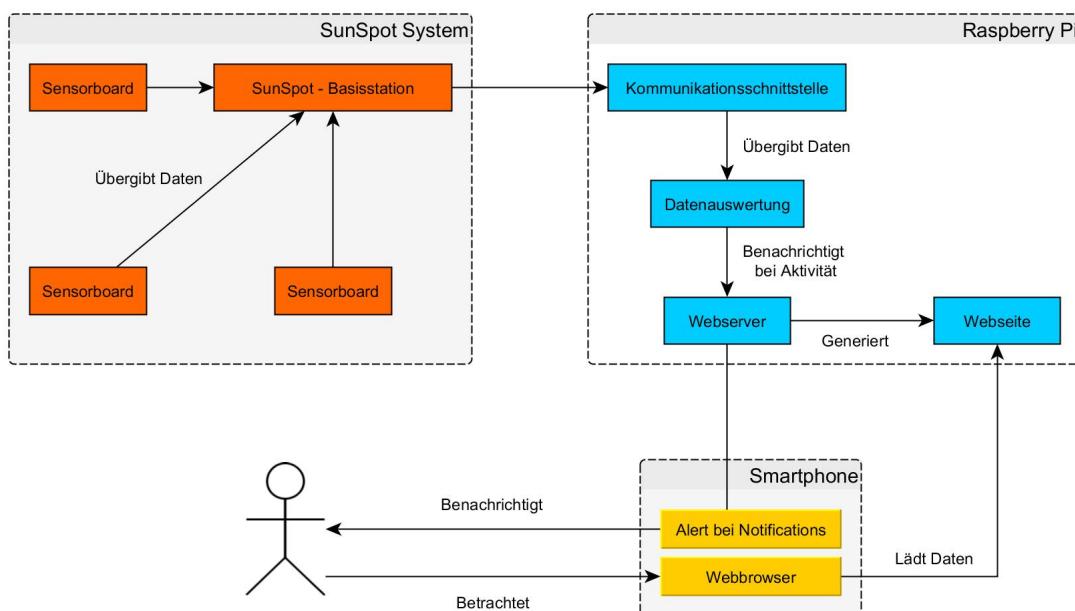


Abbildung 4.10: Aufbau eines Benachrichtigungssystems

Die zu versendenden Nachrichten müssen an die HTTP API übergeben werden. Hierfür stellt Pushover bereits Codeausschnitte für eine Vielzahl von Geräten bereit. Von einem Raspberry Pi, der im Beispielaufbau in Abbildung 4.10 enthalten ist, können Pushover-Nachrichten per Kommandozeile an die API übergeben werden.

Listing 4.7: Beispielcode zum Senden einer Pushover-Nachricht

```

1 curl -s \
2 --form-string "token=abc123" \
3 --form-string "user=user123" \

```

```
4 --form-string "message=hello world" \
5 https://api.pushover.net/1/messages.json
```

Kapitel 5

Vergleichbare Projekte

5.1 Smart Greenhouse

Das Projekt Smart Greenhouse ist wie das im folgenden Kapitel 5.2 *Bot-So* beschriebene Projekt ein Gewinner der Kategorie 'professional', der IoT Developer Challenge. Der Begriff Smart Greenhouse steht für eine IoT-Komponente und Applikation, mit der ein Gewächshaus überwacht und kontrolliert werden kann. Das Konzept hierfür entstand, als das Team, bestehend aus Dzmitry Yasevich, Pavel Vervenko und Vladimir Redzhepov, die JavaOne-Messe Russland im April 2013 besuchten. Die Gründer des Projekts sahen dort eine Präsentation eines intelligenten Hauses, das mit Hilfe verschiedener Roboter und weiteren, durch Java betriebene vernetzte Geräte, viele Komfortfunktionen für Bewohner bereitstellte. "Wir waren beeindruckt und hatten eine Idee, etwas ähnliches wie das zu entwickeln vgl. [Far15b]."



Abbildung 5.1: Das Smart Greenhouse Entwicklerteam [epa14]

Das Team begann damit, als Grundlage den Raspberry Pi festzulegen. Hiermit besaßen sie einen gut ausgestatteten Einplatinencomputer mit 700 MHz und 512 Mb Arbeitsspeicher, der mit 35 US\$ sehr preisgünstig ist. Doch bereits hiermit entstanden die ersten Bedenken. Um die Elektronik vor dem im Gewächshaus angebrachten Wässerungssystem zu schützen, musste alles gründlich isoliert und vor eindringendem Wasser geschützt werden.

Da die Pi4j Bibliothek nicht die nötigen Treiber bereitstellt, die zum Kommunizieren mit den im Gewächshaus angebrachten Feuchtigkeitssensoren benötigt sind, hat das Team eigene Treiber, sowie eine Linux-basierte Betriebssystemdistribution entwickelt, die speziell auf dieses Projekt ausgerichtet ist.

Um dies zu ermöglichen, holten sie sich die Hardwareexperten Vasili Slapik und Vladimir Redzhepov ins Team, die eine entscheidende Rolle in der Entwicklung des Projektes spielten.

Im fertiggestellten Projekt sind folgende Funktionen enthalten:

- Erstellung eines Belichtungsplans
- Erstellung eines Bewässerungsplans
- Temperaturkontrolle

- Luftfeuchtigkeitskontrolle
- Automatisches Management und ferngesteuerte Überwachung des aktuellen Gewächshauszustandes
- Automatische Erstellung eines Wachstumsverlaufs
- Schutz vor Überspannung / Stromausfall

Die Entwickler stellen dieses Projekt jedem Farmer kostenlos zur Verfügung. Da als Hardware nur ein Raspberry Pi, sowie weitere kostengünstige Komponenten benötigt werden, kann das Smart Greenhouse von Bastlern problemlos nachgebaut werden. Das Betriebssystem, sowie die nötige Software kann aus der Dropbox des Entwicklerteams kostenlos heruntergeladen werden [Far15b].

5.2 Bot-So

Das Bot-So Projekt ist ein IoT-Roboter, der sich aus einem Raspberry Pi, Oracle Java SE Embedded 8, einem Bewegungssensor, einer Videokamera und einem Anschluss an den Kurznachrichtendienst Twitter zusammensetzt. Ziel ist es, Benutzern zu ermöglichen, ihr Zuhause oder ihre Arbeitsstelle zu überwachen, während sie selbst nicht anwesend sind. Die Kamera ist an einem Schwenkarm befestigt, der durch Rotationen einen Blickwinkel von 120° ermöglicht. Um Sprachausgabe zu ermöglichen, besitzt der Roboter eine kleine Soundkarte, an die ein Lautsprecher angeschlossen werden kann. Dies ermöglicht das Wiedergeben von zuvor aufgenommenen Nachrichten [Vam14]. Über Ethernet oder WLAN kann sich der zugrundeliegende Raspberry Pi mit dem Internet verbinden. Die für das komplette System benötigte Energie wird über USB geliefert.

Der Roboter kann entweder direkt angewiesen werden, einen Raum zu scannen und Videomaterial an den Besitzer zu senden, oder im Überwachungsmodus operieren. In diesem Modus werden Bilder oder Videos aufgenommen, sobald der Bewegungssensor anschlägt [Far15a].

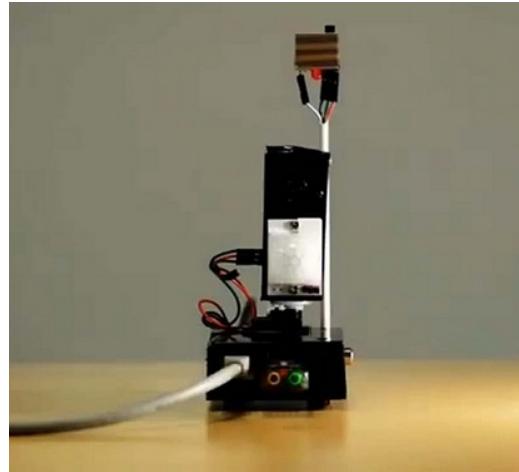


Abbildung 5.2: Der Bot-So Roboter [Par14]

Auf dem Raspberry Pi arbeitet die weit verbreitete Debian Linux Version Raspbian, die mithilfe von Jetty, Log4j, Google Drive APIs und Java Mail die Kommunikation zum User aufnimmt. Um mit Twitter zu interagieren, bedient das Bot-So Projekt der Twitter4j Bibliothek.

Die Kommunikation mit dem Roboter ist nahezu intuitiv. Er reagiert auf Twitter-Nachrichten wie 'Temp', woraufhin er die aktuelle Umgebungstemperatur mitteilt. Um einen Blick in die Umgebung des Roboters zu werfen, kann man sich entweder dem Tweet 'Take 3' bedienen, oder auf 'Sweep Room' ausweichen. Während bei 'Take 3' drei Bilder mit verschiedenen Ausrichtungen aufgezeichnet werden, fährt der Schwenkarm des Roboters bei 'Sweep Room' einmal von links nach rechts durch den Raum und zeichnet dabei ein Video auf. Die Dateiübertragung erfolgt über Google-Drive. Nach erfolgreichem Aufzeichnen von Video- oder Bildmaterial wird dies von dem Roboter an Google übertragen und ein Link an den Besitzer getwittert.

Das Robotersystem wurde von den Indern Debraj Dutta, Avinaba Brandhu Majumder, Tapas Bose und Sudhantha Krishan entwickelt. Zusammen haben sie die IoT Developer Challenge gewonnen und daraufhin die Java-One Messe besucht, um ihr Projekt vorzustellen.

Die Entwickler haben sich dazu entschieden, bei der Entwicklung des Projekts nur auf Open-Source Ressourcen zuzugreifen. Aufgrund dessen kann der Bot-So Roboter von jedem Bastler nachgebaut werden. Die Software hierzu kann auf Github heruntergeladen werden.

Kapitel 6

Zusammenfassung und Fazit

Zusammenfassung

Das IoT spielt in der sich aktuell rapide entwickelnden Welt der Informations-technik eine entscheidende Rolle. Es lässt sich bereits heute absehen, dass diese Rolle in der Zukunft zunehmend an Bedeutung gewinnt. Besonderes Augenmerk ist hierbei auf die Sicherheit privater Daten sowie den technischen Fortschritt immer kleiner werdender System-on-a-Chip-Lösungen zu richten. Auch in der Industrie können durch innovative Projekte und intelligent vernetzte Sensoren neue Überwachungsmechanismen entwickelt werden.

Der geplante Umfang dieser Studienarbeit konnte aufgrund mehrerer Faktoren nicht vollständig erreicht werden. Oracle SunSPOT ist ein Projekt der Firma Sun, das im Jahre 2007 vorgestellt wurde. Da mittlerweile 8 Jahre vergangen sind und das System stark veraltet ist, hat sich Oracle dazu entschieden jeglichen Support, sowie Onlinehilfe zum Anfang des Jahres 2015 einzustellen. Mit dieser Entscheidung, die in Mitten der Erstellung dieser Studienarbeit getroffen wurde, beeinflusst die praktische Umsetzung in großem Umfang.

Die praktische Umsetzung beschränkt sich auf die Implementierung eines Ein-bruchssensors durch Oracle SunSPOTS. Da der Support für moderne Betriebs-systeme nicht gegeben ist, war es nicht möglich eine Erweiterung durch Deskto-pwendungen zu ermöglichen.

Der theoretische Abschnitt der Arbeit ist in vollem Umfang umgesetzt ent-spricht somit den Anforderungen an diese Arbeit.

Fazit

Wir, die beiden Autoren, haben während der Umsetzung viele Kenntnisse erworben und unser bestehendes Wissen über Sensornetze und das IoT erweitert. Mit dem Einsatz modernerer Sensoren wäre allerdings eine umfangreichere praktische Arbeit möglich gewesen.

Insgesamt sind wir mit den erarbeiteten theoretischen und praktischen Ergebnissen zufrieden und werden auch in Zukunft mit großem Interesse das Thema IoT und dessen Weiterentwicklung im Hinblick auf Sensornetze, Raumüberwachung und Einbruchssicherung verfolgen.

Anhang

Code HostStudi.java

Listing 1: Code HostStudi.java

```
1  /*
2  * HostStudi.java
3  *
4  * by Nicolai Staegge & Maik Maier
5  * DHBW Karlsruhe
6  * Germany
7  *
8  */
9
10 package org.sunspotworld.demo;
11
12 import com.sun.spot.io.j2me.radiogram.*;
13 import com.sun.spot.peripheral.ota.OTACommandServer;
14 import com.sun.spot.util.IEEEAddress;
15 import com.sun.spot.util.Utils;
16 import java.awt.Color;
17 import javax.microedition.io.*;
18 import javax.swing.JFrame;
19 import javax.swing.JScrollPane;
20 import javax.swing.JTextArea;
21
22
23 /**
24 * Diese Klasse empfaengt die daten eines SPOTS
25 * um eine Bewegung anzuzeigen
26 *
27 * @author: Nicolai Staegge, Maik Maier
```

```
28 *
29 */
30 public class HostStudi extends JFrame {
31     private static final int HOST_PORT = 67;
32     private JTextArea status;
33     private JFrame fr;
34     private void setup() {
35         fr = new JFrame("Einbruchsicherung");
36         status = new JTextArea();
37         JScrollPane sp = new JScrollPane(status);
38         fr.add(sp);
39         fr.setSize(360, 200);
40         fr.validate();
41         fr.setVisible(true);
42     }
43
44     private void run() throws Exception {
45         RadiogramConnection rCon;
46         Radiogram dg;
47         try {
48             // Lffnen eines serverseitigen Ports
49             rCon = (RadiogramConnection)
50                 Connector.open("radiogram://:" +
51                             HOST_PORT);
52             dg = (Radiogram)
53             rCon.newDatagram(rCon.getMaximumLength());
54         } catch (Exception e) {
55             System.err.println("setUp caught " +
56                 e.getMessage());
57             throw e;
58         }
59         status.append("Listening...\n");
60         while (true) {
61             try {
62                 // Empfange Datenpaket vom Spot
```

```
62         rCon.receive(dg);
63         long time = dg.readLong();           // lese
64         System.out.println(time);
65         status.append("Bewegung festgestellt!\n");
66
67     } catch (Exception e) {
68         System.err.println("Fehler " + e + " beim
69             Lesen des Pakets.");
70         throw e;
71     }
72 }
73
74 /**
75 * Hauptprogramm starten
76 *
77 * @param args any command line arguments
78 */
79 public static void main(String[] args) throws
80     Exception {
81     OTACommandServer.start("HostStudi-GUI");
82     HostStudi app = new HostStudi();
83     app.setup();
84     app.run();
85 }
```

Code MovementDetection.java

Listing 2: Code MovementDetection.java

```
1  /*
2  * MovementDetection.java
3  *
4  * by Nicolai Staegge & Maik Maier
5  * DHBW Karlsruhe
6  * Germany
7  *
8  */
9
10 package org.sunspotworld.demo;
11
12 import com.sun.spot.io.j2me.radiogram.*;
13 import com.sun.spot.resources.Resources;
14 import com.sun.spot.resources.transducers.
15 IAccelerometer3D;
16 import
17     com.sun.spot.resources.transducers.ITriColorLED;
18 import
19     com.sun.spot.resources.transducers.ILightSensor;
20 import com.sun.spot.util.Utils;
21 import java.io.IOException;
22 import javax.microedition.io.*;
23 import javax.microedition.midlet.MIDlet;
24 import javax.microedition.midlet.
25 MIDletStateChangeException;
26 /**
27 * Diese Klasse entdeckt die Bewegung des SPOTs an
28 * der Tuer und sendet
29 * daraufhin ein Signal an die Basestation, um zu
30 * signalisieren, dass
31 * die Tuer bewegt wurde.
32 * Die Desktop Applikation zeigt "Bewegung entdeckt"
33 * an.
34 *
```

```
31 * @author: Nicolai Staegge, Maik Maier
32 *
33 */
34 public class MovementDetection extends MIDlet {
35
36 private static final int HOST_PORT = 67;
37
38 protected boolean isMoving() throws IOException {
39     IAccelerometer3D accel = (IAccelerometer3D)
40         Resources.lookup(IAccelerometer3D.class);
41     return accel.getAccelZ() < 0 ||
42         accel.getAccelZ() > 0.1;
43 }
44
45 protected void startApp() throws
46     MIDletStateChangeException {
47     RadiogramConnection rCon = null;
48     Datagram dg = null;
49
50     String ourAddress =
51         System.getProperty("IEEE_ADDRESS");
52
53     ILightSensor lightSensor = (ILightSensor)
54         Resources.lookup(ILightSensor.class);
55     ITriColorLED led = (ITriColorLED)
56         Resources.lookup(ITriColorLED.class, "LED7");
57
58     try {
59         // Oeffne Verbindung auf Port 67 - Port ist
60         // generell beliebig
61         // Desktop Applikation muss allerdings den
62         // gleichen Port benutzen
```

```
61      rCon = (RadiogramConnection)
62          Connector.open("radiogram://broadcast:" +
63              HOST_PORT);
64      dg = rCon.newDatagram(50); // Testsenden
65  }
66  catch (Exception e) {
67      System.err.println("Exception " + e + " beim
68          Verbindungsaufbau.");
69      notifyDestroyed();
70  }
71
72 while (true) {
73     try {
74         if (isMoving()) {
75             // Uebermittle die aktuelle Zeit
76             long now = System.currentTimeMillis();
77
78             // LED leuchtet, um zu zeigen, dass sich der
79             // Sensor bewegt hat
80             led.setRGB(255, 255, 255);
81             led.setOn();
82             Utils.sleep(50);
83             led.setOff();
84
85             // Zeit wird ins Package geschrieben, Package
86             // wird verschickt
87             dg.reset();
88             dg.writeLong(now);
89             rCon.send(dg);
90         }
91     }
92     catch (Exception e) {
93         System.err.println("Exception " + e + "
94             waehrend des Senden eines Paketes.");
95     }
96 }
```

```
92
93 protected void pauseApp() {
94 // This will never be called by the Squawk VM
95 }
96
97 protected void destroyApp(boolean arg0) throws
98 MIDletStateChangeException {
99 // Only called if startApp throws any exception
100 // other than MIDletStateChangeException
101 }
```

Literaturverzeichnis

- [Alk10] ALKIRE, Bob: *SPOTs on 64-bit Windows 7.* https://blogs.oracle.com/ralkire/entry/spots_on_64_bit_windows, May 2010
- [Arda] ARDUINO CC: *Arduino-Logo.* <http://http://www.arduino.cc/>,
- [Ardb] ARDUINO CC.: *Arduino Uno.* <http://www.arduino.cc/en/Main/ArduinoBoardUno>,
- [Ardc] ARDUINO CC: *Arduino Uno R3 Front.* http://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg,
- [Car98] CARNEGIE MELLON UNIVERSITY COMPUTER SCIENCE DEPARTMENT: *The Only Coke Machine on the Internet.* https://www.cs.cmu.edu/~coke/history_long.txt, Juni 1998
- [Con] CONRAD ELECTRONIC GMBH U. Co KG: *Raspberry Pi® Kamera-Gehäusemodul Raspberry Pi®.* <http://www.conrad.at/ce/de/product/622959/Raspberry-Pi-Kamera-Gehaeusemodul-Raspberry-Pi->,
- [Coo15] COOPER, Gordon: Sicherheit fest eingebaut. In: *Design und Elektronik* Bd. 3. www.elektroniknet.de, April 2015
- [Cua13] CUARTIELLES, David: *Arduino FAQ – With David Cuartielles.* <http://medea.mah.se/2013/04/arduino-faq/>, April 2013
- [Ele15] ELEKTOR: *Klangensitive LED-Kunst mit Arduino.* <http://www.elektor.de/news/klangensitive-led-kunst-arduino/>, März 2015
- [epa14] EPAM: *KRS 3209.* https://dl.dropboxusercontent.com/content_link/tuvLTsPbLsfH3jstipu8eJ5qf5IolE0Qm3maM4mhqwmgjZuDnTsKG7oWDaNMjiU5, Oktober 2014

- [Far15a] FARNHAM, Kevin: Bot-So - Monitor your home while you're away. In: *Java Magazine*. Oracle Corporation, Januar / Februar 2015
- [Far15b] FARNHAM, Kevin: Smart Greenhouse - Control the light, watering, temperature, and humidity of your greenhouse—automatically. In: *Java Magazine*. Oracle Corporation, März / April 2015
- [Haa08] HAAN, Kristian L.: *Advanced Encryption Standard (AES)*. <http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html>, Februar 2008
- [Hes05] HESSE, André: *IEEE 802.15.4 und ZigBee*. http://yukon.e-technik.tu-ilmenau.de/~webkn/Abschlussarbeiten/Hauptseminararbeiten/hs_hesse.pdf, 2005
- [Hor08] HORAN, Bernard: *Sun SPOTS*. <http://academic.fuseyism.com/ambassador/slides/sunspot.pdf>, 2008
- [Ins11] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: *IEEE 802.15.4 - Spezifikation*. <http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf>, 2011
- [Kah12] KAHLE, Christian: *Intel: Mooresches Gesetz gilt noch mind. 10 Jahre*. <http://winfuture.de/news,72001.html>, September 2012
- [Kos09] KOSMERCHOCK, Steven: *Wireless Sensor Network Topologies*. http://www.k5systems.com/TP0001_v1.pdf, 2009
- [LLK12] LIPINSKI, Klaus ; LACKNER, Hans ; KAFKA, Gerhard: *Ad-hoc-Netz*. <http://www.itwissen.info/definition/lexikon/Ad-hoc-Netzwerk-ad-hoc-network.html>, March 2012
- [Mul15] MULTICHERRY: *Top half of Raspberry Pi 2 Model B v1.1 viewed directly from above*. [http://commons.wikimedia.org/wiki/File:Raspberry_Pi_2_Model_B_v1.1_top_new_\(bg_cut_out\).jpg](http://commons.wikimedia.org/wiki/File:Raspberry_Pi_2_Model_B_v1.1_top_new_(bg_cut_out).jpg), Februar 2015
- [Ora10a] ORACLE CORP.: *SunTM SPOT eDEMO Technical Datasheet Rev 8.0*. <http://www.sunspotworld.com/docs/Yellow/edemo8ds.pdf>, Oktober 2010

- [Ora10b] ORACLE CORP.: *SunTM SPOT Main Board Technical Datasheet Rev 8.0.* <http://www.sunspotworld.com/docs/Yellow/eSPOT8ds.pdf>, Oktober 2010
- [Par14] PARTHASARATHY, Anand: *This robot responds to your tweets!* <http://www.deccanchronicle.com/140719/technology-latest/article/robot-responds-your-tweets>, Juli 2014
- [Ras] RASPBERRY PI FOUNDATION: *Logo der Raspberry Pi Foundation.* <https://www.raspberrypi.org/wp-content/uploads/2011/10/Raspi-PGB001.png>,
- [Tru15] TRUSCHEIT K., FRANKFURTER ALLGEMEINE ZEITUNG GMBH: *Einbrüche in Deutschland - Zahl der registrierten Fälle von Wohnungseinbrüchen mit Diebstählen.* <http://www.faz.net/aktuell/gesellschaft/kriminalitaet/zahl-der-einbrueche-in-deutschland-steigt-auch-2014-13535578.html>, April 2015
- [TS11] TIMM, Constantin ; SPINCYK, Olaf: *Software ubiquitärer Systeme - Ad-hoc-Netzwerke.* <https://ess.cs.tu-dortmund.de/Teaching/SS2011/SuS/Downloads/04.1-Adhoc-Netzwerke.pdf>, 2011
- [Tuo02] TUOMI, Ilkka: *The lives and deaths of moore's law.* <http://firstmonday.org/ojs/index.php/fm/article/view/1000/921>, November 2002
- [Uni] UNIVERSITY OF SOUTHERN CALIFORNIA: *Standardmäßiger Aufbau eines SunSPOT-Sensors.* http://anrg.usc.edu/ee579_2012/Group07/img/spotanatomy.jpg,
- [Vam14] VAMSI, Krishna: *Now, a robot that can keep eye on home via Twitter.* <http://indianexpress.com/article/technology/gadgets/now-a-robot-that-can-keep-eye-on-home-via-twitter/>, September 2014
- [WB12] WOLF, Lars ; BÜSCHING, Felix: *Wireless Sensor Networks - Introduction and Applications.* https://www.dropbox.com/sh/l2kch3izg7lwdpl/AABu1b-vt8FCuUD2iU905F0ca/IoT/RecentTopics_Chapter02_WSN-Introduction-and-Applications.pdf, 2012

- [Wei91] WEISER, Mark: *The Computer for the 21st Century.* <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>, September 1991
- [Wgs] WGSIMON: *Mooresches Gesetz.* http://commons.wikimedia.org/wiki/User:Wgsimon#mediaviewer/File:Transistor_Count_and_Moore%27s_Law_-_2011.svg,
- [Wil15] WILLEMS, Eddy: *IoT: The Internet of Things... ehm... / Ein Balance-Akt zwischen Benutzbarkeit und Sicherheit?!* <https://tcadistribution.wordpress.com/tag/sicherheitsexperten/>, März 2015
- [ZEI15] ZEIT ONLINE, dpa, AFP, RAV: *Hacker konnten BMW-Türen jahrelang per Handy öffnen.* <http://www.zeit.de/mobilitaet/2015-01/bmw-hacker-sicherheit>, Januar 2015

Notes