Vorgelegt von Alexis Danilo Morgado dos Santos

Matrikelnummer 1631401

Studiengang Technische Informatik

Fachsemester 7

Betreuer Peter Barth

Projekt

|  |  |
| --- | --- |
|  | **Fakultät für Informationstechnik** |

**Praktikumsbericht**

Im Studiengang:

technische Informatik

im Bereich

Implementierung und Konzeption einer MQTT/Nearby Hybrid Middleware

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 01.04.2020, Alexis dos Santos |  | Peter Barth |

# Eidesstattliche Erklärung

Ich versichere, dass ich diesen Bericht zum praktischen Studiensemester selbstständig und nur unter Verwendung der angegebenen Quellen- und Hilfsmittel angefertigt habe. Die Stellen, an denen Inhalte aus den Quellen verwendet wurden, sind als solche eindeutig gekennzeichnet. Die Arbeit hat in gleicher oder ähnlicher Form bei keinem anderen Prüfungsverfahren vorgelegen.

|  |
| --- |
| Mannheim, den 01.04.2020 |
|  |
| Alexis Danilo Morgado dos Santos |

# Danksagung

# Zusammenfassung

# Abstract

The present work deals with the realization of a demonstrator for partial automation and thus modernization and optimization of a laboratory. For this purpose, a network will be created, which is managed by a Raspi router and should provide a visualization of the connected devices. Furthermore, a terminal will be implemented, which will provide an RFID registration system and speed up the ordering process by using a 2d scanner, which can be controlled via a user interface.­­

# Inhaltsverzeichnis

Eidesstattliche Erklärung 2

Danksagung 3

Kurzfassung 4

Abstract 5

Inhaltsverzeichnis 6

1 Einleitung 7

1.1 Motivation 7

1.2 Zielsetzung 7

1.3 Aufgabenstellung 7

2 Grundlagen 8

2.1 JAR vs . AAR 8

2.2 Nearby 8

2.3 MQTT 8

2.4 Parallele Programmierung 8

3 Anforderungsanalyse 9

3.1 Funktionale Anforderungen 9

3.2 Vergleichen von Nearby und MQTT 10

4 Software-Design 11

5 Implementierung des Systems 14

Quellenverzeichnis 17

# Einleitung

## Motivation

## Zielsetzung

* Möglichst einfach zu bedienen
* Gut dokumentiert
* Vorallem Nachvollziehbarere SW Entwurf z.B. durch Verwendung bekannten Entwurfmuster
* Anwendung in unterschiedlichen Anwendungen strukturieren
* Durch spätere Anforderungsanalyse werden Anforderungen später dazu spezifiziert

## Aufgabenstellung

* Middleware für Android, die MQTT und Nearby verwendet.
* Wie auch MQTT und Nearby soll das zu entwerfende System eine Publish/ Subscribe API anbieten.
* Prioritätenbasierte

# Anforderungsanalyse

(Zusammenfassung der folgenden Unterkapitel)

Im Folgenden sollen die grundlegenden Anforderungen an das System erfasst werden. Entsprechend der MoSCoW-Methode werden diese dabei entsprechend ihrer Wichtigkeit priorisiert.

* D.h.
  + M – Must have: Kernanforderungen, welche umgedingt umzusetzen sind
  + S – Should have: Sollten im best-case umgesetzt werden
  + C – could have: Können nach Erfüllung der höherpriorisierten Anforderungen noch genügend Zeit bestehen, werden diese Erfüllt
  + W – Won’t have: Können in Zukunft umgesetzt werden

In diesem Kapitel sollen die konkreten Anforderungen an das zu entwickelnde System (Android-Bibliothek) ermittelt werden. Um die Anforderungsanalyse zu erleichtern sollen hierfür zunächst die funktionalen Anforderungen dreier Szenarien (Beispielanwendungen) bestimmt werden. Die ermittelten Anforderungen ermöglichen eine Deduktion, aus welcher sich die konkreten Anforderungen an die Nearfly-Bibliothek ergeben.

## Szenario 1 – Messenger App

Das 1. Szenario ist die Entwicklung einer Messenger App, in welche konventionelle Chatrooms erstellt werden und durch mehrere Benutzer betreten werden können. Basierend auf klassische Messenger Apps entstehen hierfür folgende funktionale Anforderungen.

**Bild- und Text senden:** Das System muss dem Nutzer die Möglichkeit geben, Text- und Bildnachrichten zu versenden, welche von allen anderen Benutzern empfangen werden.

**Chatroom erstellen:** Ein Nutzer kann einen eigenen Chatroom erstellen.

**Chatroom beitreten:** Ein Nutzer kann einen erstellten Chatroom beitreten.

**Echte Parallelität**: Werden Bild- und Textnachrichten beinah parallel versendet, muss die kleinere beider Nachrichten schneller ankommen, auch wenn diese nach der größeren gesendet worden ist.

**Differenzierbarkeit:** Jedem Nutzer müssen ein Name und eine Farbe zugewiesen werden.

## Szenario 2 – Shared Touchpoint Canvas

Das 2. Szenario umfasst das Entwickeln einer minimalistischen Anwendung, welche eine Leinwand beinhaltet, die von allen Benutzern gemeinsam bemalt werden kann. Jedem Benutzer wird zu Beginn eine Farbe zugeteilt. Berührt ein Benutzer die Leinwand, entsteht ein kolorierter Abdruck (Touchpoint), welcher nach Ablauf einer gewissen Zeit verschwindet. Die Umsetzung der fundamentalen Funktionen inkludiert folgende Anforderungen:

**Touchpoints Broadcasting:** Falls die Leinwand berührt wird, muss das System die Berührung erfassen und diese den anderen Benutzern mitteilen.

**Shared Canvas**: Falls ein Berührungspunkt empfangen wird, muss das System diesen mit der dem Benutzer zugewiesenen Farben an die entsprechende Position zeichnen.

**Differenzierbarkeit:** Falls die Anwendung gestartet wird, muss das System dem Benutzer eine Farbe zuweisen.

**Auto-Clean:** Nach einer festgelegten Zeit, muss das System die kolorierten Touchpoints aus der Leinwand entfernen.

**Join-Benachrichtigung:** Falls ein Benutzer die Anwendung startet, muss das System alle aktiven Benutzer derselben Anwendung benachrichtigen.

## Szenario 3 – Bouncing Ball

Das 3. Szenario umfasst das Entwickeln einer App, welche eine Kugel beinhaltet, die durch schwenken des Android Gerätes balanciert wird. Jeder Benutzer balanciert dieselbe Kugel zeitgleich, sodass das Verhalten der Kugel, die Summe der Aktionen aller Spieler ist.

* A1: Das System muss die kontinuierlich die Neigungsdaten aller Spieler erfassen und an alle Spieler senden.
* A2: Nach dem Erhalt der Neigungsdaten aller Spieler, muss das System ein resultierendes Verhalten für die Kugel berechnen.
* A3: Das System muss jedem Spieler die Möglichkeit geben, das Spiel synchron (als Team) zu starten.
* A4: Das System muss jedem Spieler die Möglichkeit geben, das Spiel zum Anfang einer Runde beizutreten.

## Funktionale Anforderungen

Bei Betrachtung der ermittelten Anforderungen lassen sich korrelierende Anforderungen erkennen. So ist etwa das Betreten eines Raumes, d.h. die konkrete Trennung von Datenströmen innerhalb derselben Anwendung in allen drei Szenarien enthalten. Weiterhin konnten folgende funktionale Anforderungen identifiziert werden:

**Channelbased Broadcasting:** Zu sendende Nachrichten müssen einem Empfangskanal zugewiesen werden.

**Kontextualität:** Nachrichten sind nur im Kontext derselben Anwendung sichtbar

**channelbased listening:** Das System kann auf eingehende Nachrichten unterschiedlicher Kanäle verschieden reagieren.

**Traceability:** Der Sender einer Nachricht muss identifiziert werden können

**Priorisierte Textnachricht:** Werden 2 Nachrichten unterschiedlicher Größe versandt, kommt die kleinere beider Nachrichten erster an.

Zudem lassen sich aus den bestehenden Anforderungen folgende weitere Bibliotheksnähere Anforderungen extrapoliere:

**Flawless Switch:** Wechselt das System die unterliegende Technologie während des Betriebes, soll dies möglichst reibungslos funktionieren.

**Autoverbinndungsaufbau:** Scheitert die Verbindung muss das System in einen kontrollierten Zustand wechseln, in welchem die Verbindung wiederaufgebaut wird, dabei soll eingestellt werden können ob nicht versandte (pending) Nachrichten zwischengespeichert oder verworfen werden sollen.

**Debug-Tool:** Das System soll dem Entwickler die Möglichkeit geben, die höchste auftretenden Round-Trip Time während des Betriebes zu messen.

## Nichfunktionale Anforderungen

**Parallelität:** Das Empfangen, wie auch das Senden muss parallel geschehen.

**Zielumgebung**: Das System muss eine Kompatibilität zu allen Android Betriebssystemen der API 24 (Android 7.0) oder höher aufweisen.

**Leichte Benutzbarkeit:** Die API soll möglichst minimalistisch gehalten werden und damit möglichst Einstiegsfreundlich für die Entwickler sein.

**Konsistenz:** Unabhängig der verwendeten Technologie (Nearby/ MQTT), soll sich das System möglichst ähnlich verhalten.

## Akteuere

Aus der Anforderungsanalyse lassen sich ingesamt 2 Akteure identifizieren. Hau

# Grundlagen

## JAR vs. AAR

## Nearby

## MQTT

# Software-Design

Im folgenden wird Design veranschaulicht:

ExtMessage: Erweiterte Nachricht, welche hilft Nearby-Messsages an MQTT-Messages assimilieren. Beinhaltet bisher: Topic, Payload… zukünftig wahrscheinlich noch sequence number, Gesamtanzahl der Chunks pro Paket, Priorität, sendLatest + evtl. timestamp

**Nearby Unpublisher:** Wird gebraucht, da Nearby im gegensatz zum MQTT Protokoll Nachrichten „persistent“ publisht. Damit werden gepublishte Nachrichten, ähnlich MQTT publishes mit aktiver retain flag, bis zu ihrem unpublish von allen subscribern ein mal empfagen.

* Nach publish, wartet dieser durch sleep bis zum aublauf der timetolive der gepublishten Nearby Nachricht und unpublisht diese.
* Sonst legt dieser sich schlafen, ein erneutes publishen wacht diesen durch ein notify auf

Neafly Service: Service Klasse, welche sowohl den selbst implementierten Neafby Service, wie auch den MQTT Service beinhaltet.

* Integrierter Nearby Service beinhaltet: Kanalfilter,

Message Listener(Interface): Dient zur dependency inversion zwischen subManager und Nearfly Service durch z.B. onMessage/ onStatus-Methoden

SubPreprocessor:

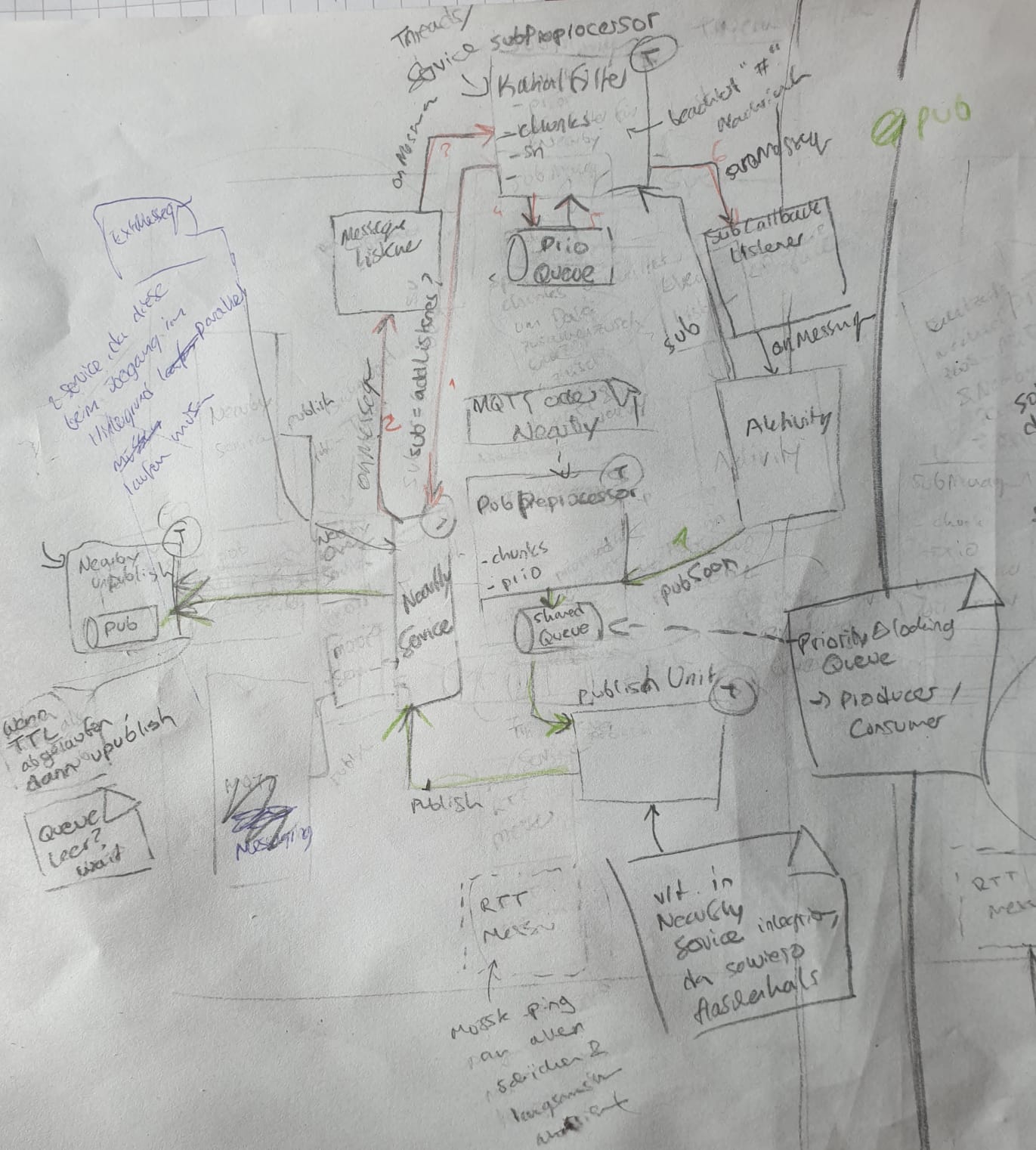
* Speichert erhaltene Datenpakete nach Eintreffen in PriorityBlockingQueue
* Hat einen Counter (Sequence Number), welcher
* Zuständig erhaltene Chunks zusammenzufügen
* Trifft eine not chunked oder die letzte einer chunked Nachricht ein, werden wenn nötig deren Fragmente aus einer PriorityQueue geholt und der SubCallBack Listener getriggert.

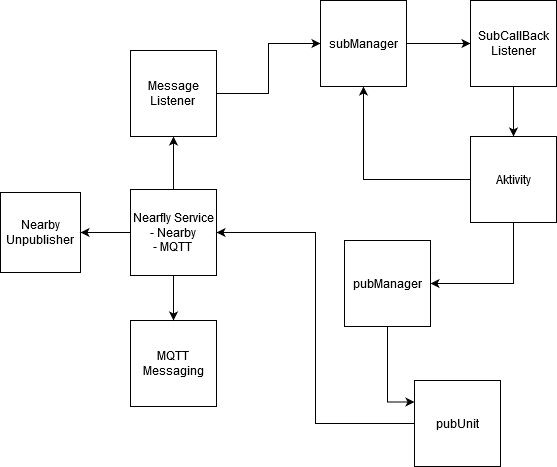
SubCallBack Listener: Benachrichtigt Aktivity bei eintreten neuer Nachrichten

pubPreprocessor:

* Zerlegt Nachrichten in Chunks und legt diese in unbounded Priorityqueue rein
* Kann evtl. Threadpool für das publishen von Nachrichten verwenden

pubUnit:

* Worker, welcher bei gefüllter PriorityQueue für das publishen der chunked Nachrichten zuständig ist.
* 

­



**Abbildung 2:** Software-Architektur

Nearfly API hat folgende Funktionen:

* Connect(MqttOptions, NearbyOptions)
* Subscribe(channel, QoS)
* Unsubscribe(channel)
* Publish(channel, payload, priority)

# Implementierung des Systems

ich schreibe Ihnen aufgrund der Nearby Connection API. Soweit habe ich eine minimalistische Beispiel-App entwickelt, welche praktisch nur zum connecten und publishen der Devices zuständig ist. Das Verbinden an sich klappt durch Discovery(D) und Advertising(A). Möglich ist hierbei, das 1 Device D und der andere A spielt oder beide A&D zeitgleich. Das Verbinden von 2 nahliegenden Devices geht Flott(wenige Sekunden), da bloß 1 Verbindung benötigt wird. Der Nachrichtenaustausch danach deutlich schneller. Versucht man jetzt ein 3. Device hinzuzufügen, fängt die Nearby-Technology an zu humpeln. Während die erste Verbindung vergleichbar dem 2 Devices Anwendungsfall recht zügig geht, kann es bei den folgenden Verbindungen je nach Glück von einer halben Minute bis zu mehreren wenigen Minuten dauern. Extrapoliert man das Verhalten, wird das z.B. bei 5 Devices (bei 10 benötigten Verbindungen) schon deutlich schwieriger. Das obere Verhalten gilt für ein Clustering-Netzwerk. Setzt man jedoch ein Device als localen Server(Stern-Netzwerk), braucht man nurnoch einen der Discovert und alle anderen müssen nurnoch publishen, geht das deutlich besser. Ich experementiere soweit noch mit Versionen, Einstellungen ... Mein anliegen ist nun folgender: Sie haben im Exposé erwähnt gehabt, das Sie keinen Smartphone als Server haben wollten, derzeitig scheint das jedoch tatsächlich dahin zu führen.

# Diskussion

# Zusammenfassung und Ausblick

Quellenverzeichnis

amplify (2018;): Die smarte Schönheit. o.O. Online verfügbar unter https://amplify.pepperl-fuchs.com/de/inhalte/178/die-smarte-schoenheit, zuletzt geprüft am 20.05.2019.

Ay, Cengiz (o.J.): Exploits: Ausnutzung technischer Sicherheitslücken in Systemen. Bad Colberg - Heldburg. Online verfügbar unter https://www.edv-lehrgang.de/exploits/, zuletzt aktualisiert am 13.01.2019, zuletzt geprüft am 07.06.2019.

BMBF (o.J.): Industrie 4.0. o.O. Online verfügbar unter https://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html, zuletzt geprüft am 15.08.2019.

Böttcher, Björn (2016): Breites Spektrum und offene Fragen: Welche Netzwerke braucht die IoT-Kommunikation? München. Online verfügbar unter https://www.computerwoche.de/a/welche-netzwerke-braucht-die-iot-kommunikation,3227026, zuletzt aktualisiert am 25.04.2016, zuletzt geprüft am 10.06.2019.

chemie.de (o.J.): Radionuklid. Hg. v. LUMITOS AG. Berlin. Online verfügbar unter https://www.chemie.de/lexikon/Radionuklid.html, zuletzt geprüft am 23.07.2019.

Elektronik Kompendium (2019): Router. Hg. v. Elektronik Kompendium. Online verfügbar unter https://www.elektronik-kompendium.de/sites/net/1404181.htm, zuletzt aktualisiert am 15.07.2019, zuletzt geprüft am 15.08.2019.

Erling, Johnny (2018): Digitale Diktatur in China: Zur Strafe kommt der Hund weg. Zur Strafe kommt der Hund weg. In: *Welt*, 20.11.2018. Online verfügbar unter https://www.welt.de/politik/ausland/article184203958/Digitale-Diktatur-in-China-Zur-Strafe-kommt-der-Hund-weg.html, zuletzt geprüft am 20.05.2019.

Fleisch, Elgar; Thiesse, Frédéric (2014): Internet der Dinge. Zürich, Schweiz. Online verfügbar unter http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/technologien-methoden/Rechnernetz/Internet/Internet-der-Dinge, zuletzt aktualisiert am 26.09.2014, zuletzt geprüft am 15.08.2019.

Forschungsunion (2013): Umsetzungsempfehlungen für das ZukunftsprojektIndustrie 4.0. Hg. v. catech – Deutsche Akademie der Technikwissenschaften e. V. Frankfurt am Main. Online verfügbar unter https://www.bmbf.de/files/Umsetzungsempfehlungen\_Industrie4\_0.pdf, zuletzt geprüft am 15.08.2019.

Fraunhofer (o.J.): Wissenschaftliche Exzellenz. München. Online verfügbar unter https://www.fraunhofer.de/de/ueber-fraunhofer/wissenschaftliche-exzellenz.html, zuletzt aktualisiert am 17.03.2019, zuletzt geprüft am 17.03.2019.

Fraunhofer (2014a): Automatisierung für die Medizin und Biotechnologie. München. Online verfügbar unter https://pamb.ipa.fraunhofer.de/de/ueber-uns.html, zuletzt aktualisiert am 18.03.2019, zuletzt geprüft am 18.03.2019.

Fraunhofer (2014b): Fraunhofer am Klinikum Mannheim. Online verfügbar unter https://pamb.ipa.fraunhofer.de/content/dam/pamb/de/images/Presse\_Medien/Faltblatt-Fraunhofer%20am%20Klinikum%20Mannheim.pdf, zuletzt geprüft am 17.03.2019;

Fraunhofer (2018): Über das Projekt | Lab-IoT. PAMB Mannheim. Mannheim. Online verfügbar unter https://websites.fraunhofer.de/Lab-IoT/?page\_id=14, zuletzt geprüft am 26.07.2019.

Fraunhofer (2019): Zahlen und Fakten. München. Online verfügbar unter https://www.fraunhofer.de/de/ueber-fraunhofer/profil-struktur/zahlen-und-fakten.html, zuletzt aktualisiert am 17.03.2019, zuletzt geprüft am 17.03.2019.

Geißler, Otto (2018): Was ist ein Frontend und ein Backend? Vogel Communications Group GmbH & Co. KG. Augsburg. Online verfügbar unter https://www.datacenter-insider.de/was-ist-ein-frontend-und-ein-backend-a-714429/, zuletzt geprüft am 19.07.2019.

GS Lexikon (o.J): Digitalisierung. o.O. Online verfügbar unter https://www.gruenderszene.de/lexikon/begriffe/digitalisierung?interstitial, zuletzt geprüft am 14.08.2019.

herold (2017): Energieautark: Definition und Begriffserklärung - HEROLD.at. Mödling, Österreich. Online verfügbar unter https://www.herold.at/blog/glossar/energieautarkie/, zuletzt aktualisiert am 07.06.2019, zuletzt geprüft am 08.06.2019.

idw (2011): Fraunhofer-Projektgruppe PAMB entsteht in Mannheim. Hg. v. idw - Informationsdienst Wissenschaft. o.O. Online verfügbar unter https://idw-online.de/de/news408715, zuletzt geprüft am 15.08.2019.

itwissen (2013): Overhead :: OH (overhead) :: ITWissen.info. o.O. Online verfügbar unter https://www.itwissen.info/Overhead-overhead-OH.html, zuletzt geprüft am 28.06.2019.

Juniper Research (2018): IoT Connections to Grow 140% to Hit 50 Billion By 2022, As Edge Computing Accelerates ROI. Online verfügbar unter https://www.juniperresearch.com/press/press-releases/iot-connections-to-grow-140-to-hit-50-billion, zuletzt geprüft am 04.04.2019.

LineFeed (2017): Was sind Docker-Container? Vogel Communications Group GmbH & Co. KG. o.O. Online verfügbar unter https://www.dev-insider.de/index.cfm?pid=1&pk=597762&p=1, zuletzt geprüft am 26.06.2019.

Lord, David (2018): Flask. o.O. Online verfügbar unter https://github.com/pallets/flask, zuletzt aktualisiert am 08.06.2019, zuletzt geprüft am 12.08.2019.

Luber, Stefan (2017): Was ist der Raspberry Pi? Vogel Communications Group GmbH & Co. KG. Augsburg. Online verfügbar unter https://www.dev-insider.de/was-ist-der-raspberry-pi-a-598750/, zuletzt geprüft am 26.06.2019.

Luber, Stefan (2018): Was ist ein Proxy. o.O. Online verfügbar unter https://www.ip-insider.de/was-ist-ein-proxy-server-a-665349/, zuletzt aktualisiert am 01.08.2018, zuletzt geprüft am 12.08.2019.

Maier, Florian (2016): Die mächtigsten IoT-Unternehmen. o.O. Online verfügbar unter https://www.cio.de/a/die-maechtigsten-iot-unternehmen,3255962, zuletzt aktualisiert am 11.04.2016, zuletzt geprüft am 13.08.2019.

Mattern, Friedemann; Flörkemeier, Christian (2010): Vom Internet der Computer zum Internet der Dinge. ETH Zürich. Zürich, Schweiz. Online verfügbar unter https://www.vs.inf.ethz.ch/publ/papers/Internet-der-Dinge.pdf, zuletzt geprüft am 08.06.2019.

Maurer, Jürgen (2018): IoT-Studie von COMPUTERWOCHE und CIO. Das Internet of Things nimmt langsam Fahrt auf. München. Online verfügbar unter https://www.computerwoche.de/a/das-internet-of-things-nimmt-langsam-fahrt-auf,3332250, zuletzt aktualisiert am 23.02.2018, zuletzt geprüft am 23.05.2019.

Obermaier, Dominik (2018): Evolution der IoT-Kommunikation: MQTT 5. Heise Medien. o.O. Online verfügbar unter https://www.heise.de/developer/artikel/Evolution-der-IoT-Kommunikation-MQTT-5-3941656.html, zuletzt aktualisiert am 22.07.2019, zuletzt geprüft am 22.07.2019.

Orebaugh, Angela (2015): Internet der Dinge: Was zu tun ist, um IoT-Security Realität werden zu lassen. München. Online verfügbar unter https://www.computerweekly.com/de/meinung/Internet-der-Dinge-Was-zu-tun-ist-um-IoT-Security-Realitaet-werden-zu-lassen, zuletzt geprüft am 07.06.2019.

Probst, Laurent; Pedersen, Bertrand; Dakkak-Arnoux, Lauriane; PwC (2017): Digital Transformation Monitor. Energy harvesting to power the rise of the Internet of Things. European Commission. o.O. Online verfügbar unter https://ec.europa.eu/growth/tools-databases/dem/monitor/sites/default/files/DTM\_Energy%20harvesting%20v1\_0.pdf, zuletzt geprüft am 08.06.2019.

Rentrop, Christian (2018a): Der Raspberry Pi im Internet of Things. Vogel Communications Group GmbH & Co. KG. o.O. Online verfügbar unter https://www.dev-insider.de/der-raspberry-pi-im-internet-of-things-a-714895/, zuletzt geprüft am 26.06.2019.

Rentrop, Christian (2018b): Was leisten Raspberry Pi und Raspbian? Vogel Communications Group GmbH & Co. KG. o.O. Online verfügbar unter https://www.dev-insider.de/was-leisten-raspberry-pi-und-raspbian-a-710998/, zuletzt geprüft am 26.06.2019.

Sawall, Achim (2015): Mobilfunkpreise in Deutschland sinken weiter. Online verfügbar unter https://www.golem.de/news/statistisches-bundesamt-mobilfunkpreise-in-deutschland-sinken-weiter-1506-114895.html, zuletzt aktualisiert am 26.06.2015, zuletzt geprüft am 15.08.2019.

Schnabel, Patrick (2019): LPWAN - Low Power Wide Area Network. Ludwigsburg. Online verfügbar unter https://www.elektronik-kompendium.de/sites/kom/2207181.htm, zuletzt aktualisiert am 10.05.2019, zuletzt geprüft am 10.06.2019.

selfhtml (2019): JavaScript/Tutorials/Umgang mit Callback-Funktionen. o.O. Online verfügbar unter https://wiki.selfhtml.org/wiki/JavaScript/Tutorials/Umgang\_mit\_Callback-Funktionen, zuletzt aktualisiert am 03.01.2019, zuletzt geprüft am 13.08.2019.

Siemons, Mark (2018): Chinas Sozialkreditsystem. Die totale Kontrolle. In: *FAZ*, 11.05.2018. Online verfügbar unter https://www.faz.net/aktuell/feuilleton/debatten/chinas-sozialkreditsystem-die-totale-kontrolle-15575861.html, zuletzt geprüft am 20.05.2019.

Spektrum (o.J.): Radiochemie. Spektrum Akademischer Verlag. Heidelberg. Online verfügbar unter https://www.spektrum.de/lexikon/chemie/radiochemie/7788, zuletzt aktualisiert am 04.12.2014, zuletzt geprüft am 23.07.2019.

Strietzel, Joseph (2018): Digitalisierung einfach erklärt. o.O. Online verfügbar unter https://www.bilendo.de/blog/digitalisierung-einfach-erklart, zuletzt aktualisiert am 03.04.2018, zuletzt geprüft am 14.08.2019.

The HiveMQ Team (2015): MQTT Essentials Part 2: Publish & Subscribe. Landshut. Online verfügbar unter https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/, zuletzt aktualisiert am 07.06.2019, zuletzt geprüft am 11.06.2019.

Tomás, Juan Pedro (2018): Nearly 250,000 US farmers already using IoT technology, study finds, zuletzt aktualisiert am 16.05.2018, zuletzt geprüft am 20.05.2019.

Trelle, Tobias (2015): MongoDB für Software-Entwickler. o.O. Online verfügbar unter https://www.informatik-aktuell.de/betrieb/datenbanken/mongodb-fuer-software-entwickler.html, zuletzt aktualisiert am 28.05.2015, zuletzt geprüft am 12.08.2019.

Wikipedia-Autoren (2019a): Fraunhofer-Gesellschaft. Hg. v. Die freie Enzyklopädie. Wikipedia. Online verfügbar unter https://de.wikipedia.org/w/index.php?title=Fraunhofer-Gesellschaft&oldid=190758363, zuletzt aktualisiert am 04.07.2019, zuletzt geprüft am 15.08.2019.

Wikipedia-Autoren (2019b): Fraunhofer-Institut für Produktionstechnik und Automatisierung. Hg. v. Die freie Enzyklopädie. Wikipedia. o.O. Online verfügbar unter https://de.wikipedia.org/w/index.php?title=Fraunhofer-Institut\_f%C3%BCr\_Produktionstechnik\_und\_Automatisierung&oldid=189059688, zuletzt aktualisiert am 29.05.2019, zuletzt geprüft am 15.082019.