Frankfurt University of Applied Sciences

Fachbereich 2 – Studiengang Informatik

Bachelorthesis

Realisierungsmöglichkeiten von Edge- und Fog-Computing auf Basis des Einplatinencomputers Raspberry Pi

---

Eingereicht zum Erlangen des akademischen Grads

Bachelor of Science

Autor: James Toribio

Matrikel-Nummer: 1015528

Referent: Prof. Dr. Christian Baun

Korreferent: Prof. Dr. Thomas Gabel

Eidesstattliche Erklärung

Hiermit erkläre ich, James Toribio, geboren am 01.08.1991 in Frankfurt am Main, dass ich die vorliegende Bachelorarbeit mit dem Titel „Realisierungsmöglichkeiten von Edge- und Fog-Computing auf Basis des Einplatinencomputers Raspberry Pi“ selbstständig verfasst und auch keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen der Arbeit, in dem der Wortlaut oder die Sinnmäßigkeit von anderen Werken entnommen wurden, wurden kenntlich gemacht.
Die Bachelorarbeit hat in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

Danksagung

Meinen besonderen Dank vergebe ich an meine Familie und Freunde, die mich in jeder Lebenssituation unterstützt und mir so dieses Studium ermöglicht haben. Ich möchte auch den Professoren, Tutoren und Kommilitonen der Frankfurt University of Applied Sciences danken, die ich im Laufe meines Studiums kennenlernen durfte und mir geholfen haben. Vielen Dank auch an meinen Betreuer dieser Bachelorarbeit, Herrn Prof. Dr. Baun und Herrn Prof. Dr. Gabel, der sich als Korreferent für diese Arbeit zur Verfügung gestellt hat.

Zusammenfassung

Immer mehr Geräte werden mit dem Internet vernetzt. Statista sagt voraus, dass bis 2025 über 75 Billionen IoT-Geräte weltweit mit dem Internet vernetzt werden [1]. Damit steigen die entstehenden Datenmengen, die über das Internet in die Cloud wandern. Beispielsweise soll ein autonom fahrendes Auto bereits vier TB Daten pro Tag erzeugen, welche dann in die Cloud wandern müssen, um analysiert zu werden. Ein Versuch dieses Datenproblem zu lösen ist es, Informationen am Rand (engl. „Edge“) einer IT-Infrastruktur oder am Endgerät selbst zu verarbeiten [2].

Die vorliegende Bachelorarbeit dokumentiert die aktuellen (Stand: November 2018) Trends im Bereich des Fog- und Edge-Computings. Dabei werden die Unterschiede zwischen den Cloud-Computing-Diensten „IaaS“, „PaaS“ und „SaaS“, sowie weiteren Cloud-Computing-Modellen erläutert. Die Begriffe Fog- und Edge-Computing werden gegenübergestellt und die Unterschiede erläutert. Software-Lösungen im Gebiet des Edge- bzw. Fog-Computings für den Einplatinencomputer Raspberry Pi werden analysiert und anschließend auf dem Raspberry Pi realisiert. Der Fokus dieser Arbeit liegt jedoch im Bereich des Edge Computings. Für die Lösungen werden detaillierte Installationsanleitungen dargelegt und die Bedienung erläutert.

Abstract

The number of devices connected to the internet will increase in the future. Statista predicts, that until the year of 2025 there will be over 75 billion IoT-Devices connected to the internet [1]. This increases the amount of generated data which will be transferred to the cloud via the internet. For instance, an autonomously driving car already generates about four TB data each day, which will have to move to the cloud to be analyzed. One attempt to solve this data problem is, to process the information on the edge of an IT-infrastructure or on the device itself [2].

This bachelor thesis documents the current (Effective November 2018) trends in the field of Fog Computing and Edge Computing. In the process, the differences between the Cloud Computing services „IaaS", "PaaS" and "SaaS" as well as other Cloud Computing models will be explained. The terms Fog Computing and Edge Computing will be compared to each other and the differences depicted. Available software solutions in the field of Edge Computing and Fog Computing for the single-board computer Raspberry Pi will be analyzed and prototypes of the solutions will be installed on the Raspberry Pi. The focus of this thesis lies in the field of Edge Computing. For the solutions, detailed installation manuals will be provided and the operations will be explained.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACR = Azure Container Registry

AMQP = Advanced Message Queuing Protocol

API = Application Programming Interface

ARM = Advanced RISC Machines

AWS = Amazon Web Services

BCM = Broadcom

BLE = Bluetooth Low Energy

C2D = Cloud-to-Device

CLI = Command Line Interface

CPU = Central Processing Unit

CSI = Camera Serial Interface

D2C = Device-to-Cloud

GB = Gigabyte

GHz = Gigahertz

GPIO = General Purpose Input Output

GPU = Graphics Processing Unit

GUI = Graphical User Interface

HDMI = High Definition Multimedia Interface

IoT = Internet of Things

IT = Information Technology

KI = Künstliche Intelligenz

LPDDR2-SDRAM = Low Power Double Data Rate 2 Synchronous Dynamic Random Access Memory

M2M = Machine to Machine

MB = Megabyte

MHz = Megahertz

MQTT = Message Queuing Telemetry Transport

MSP = Managed Service Provider

PC = Personal Computer

PoE = Power-over-Ethernet

RAM = Random-Access Memory

RCA = Radio Corporation of America

SD = Secure Digital

SDK = Software Development Kit

SoC = System-on-a-Chip / System-on-Chip

SSL = Secure Sockets Layer

TB = Terabyte

TLS = Transport Layer Security

USB = Universal Serial Bus

VS = Visual Studio

WLAN = Wireless Local Area Network

ZB = Zettabyte

Abbildungsverzeichnis

Tabellenverzeichnis

1 Stand der Technik

Im Kapitel Stand der Technik wird eine Übersicht der in dieser Arbeit eingesetzten Technik gegeben.

1.1 Raspberry Pi

Der Raspberry Pi (auch Raspi oder RPi genannt) ist ein Einplatinencomputer der Raspberry Pi Foundation. Er wurde so konzipiert, dass die Kosten möglichst gering sind. Die Raspberry Pi Foundation möchte damit einen kostengünstigen, leistungsfähigen Einplatinencomputer anbieten [3]. Das offizielle Betriebssystem für alle Raspberry Pi Modelle ist Raspbian. Auf der offiziellen Seite von Raspberry Pi werden noch neun weitere Betriebssystemabbilder von anderen Anbietern angeboten: Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IoT Core, OSMC, LIBREELEC, PINET, RISC OS, WEATHER STATION und ICHIGOJAM RPI [4]. Für die Installation des Betriebssystems wird eine MicroSD-Karte benötigt. Auf der MicroSD-Karte wird das Betriebssystem aufgespielt. Neben der MicroSD-Karte werden auch noch eine USB-Tastatur, eine USB-Maus und ein Monitor benötigt, um die ersten Konfigurationseinstellungen vorzunehmen. Der Monitor wird über den HDMI-Anschluss angebunden.

1.1.1 Raspberry Pi Modelle und Raspberry Sense HAT

Derzeit gibt es 14 Raspberry Pi Modelle (Stand: Oktober 2018) [5], die im Anhang auf S. 63-65 in den Tabellen 5, 6 und 7, aufgelistet sind. Auf die Modelle Raspberry Pi 3 B und Raspberry Pi 3 B+ wird genauer eingegangen, da diese zwei Modelle für die vorliegende Arbeit relevant sind. Außerdem wird in dieser Arbeit das Raspberry Sense HAT zum generieren von Sensordaten verwendet.

1.1.2 Raspberry Pi 3 Modell B

Das im Februar 2016 erschienene Raspberry Pi 3 Modell B ersetzt seinen Vorgänger, den Raspberry Pi 2 Modell B und bietet eine höhere Leistung [6]. Der Raspi 3 B ist nun mit einem 1,2 GHz 64-Bit Vier-Kern-Prozessor ausgerüstet. Sein Vorgänger, der Raspi 2 Modell B, hat ein 900 MHz 32-Bit Vier-Kern-Prozessor. Die Arbeitsleistung wird dadurch im Vergleich zum Vorgängermodell um ca. 50-60% verbessert. Zusätzlich hat

die Raspberry Pi Foundation einen integrierten 802.11n WLAN und BLE 4.1 Chip in das Raspberry Pi 3 Modell B eingebaut, welche bei seinen Vorgängern nicht vorhanden war [7]. Die Stromzufuhr wurde von 1,8A auf 2,5A erhöht, da die extern verbundenen USB-Geräte einen höheren Stromverbrauch haben könnten. Es wird empfohlen dazu einen 2,5A Adapter zu benutzen [6]. Preislich ist der Raspberry Pi 3 Modell B bei ca. 32€. (Stand: November 2018, amazon.de)

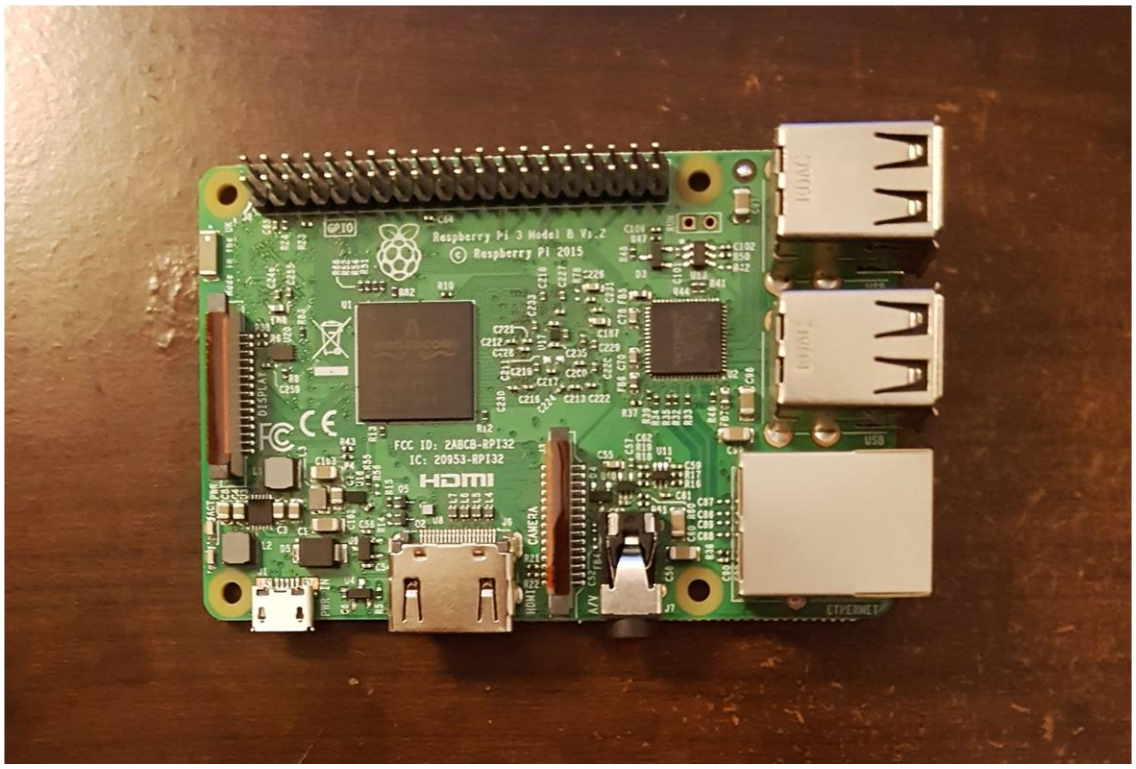


Abbildung 1: Raspberry Pi 3 Modell B

Technische Details:

Taktfrequenz: 1,2 GHz	Schnittstellenerweiterung: GPIO 40 Pin	Peripheriegeräte: 17 × GPIO
Chip (SoC): Broadcom BCM2837	Video-Ausgänge: HDMI und RCA, plus 1 CSI-Kameraanschluss	Stromversorgung: 5v 2.5A über Micro-USB
Prozessor: ARM Cortex-A53 64 Bit Quadri-Core	Audio-Ausgänge: 3,5 mm Stereobuchse oder HDMI	Abmessungen: 85,60 mm × 53,98 mm × 17 mm
Speicherkapazität (SDRAM): 1	Datenspeicher: MicroSD-Karte	Gewicht: 45 g

GB LPDDR2		
Anzahl der USB 2.0-Schnittstellen: 4	Netzwerkanschluss: 10/100 Ethernet, WiFi 802.11n und Bluetooth 4.1 (BLE - Low Energy)	Grafikprozessor: Broadcom VideoCore IV Double Core (OpenGL ES 2.0, H.264 Full HD mit 30 ips)

Tabelle 1: Technische Details Raspberry Pi 3 Modell B [1]

1.1.3 Raspberry Pi 3 Modell B+

Der am 14. März 2018 veröffentlichte Raspberry Pi 3 Modell B+ ist die überarbeitete Version vom Raspberry Pi 3 Modell B. Äußerlich hat sich das verbesserte Modell kaum verändert, weswegen an dieser Stelle keine Abbildung des neuen Modells nötig ist. Preislich liegt der Raspi 3 B+ bei ca. 38€ (Stand: Dezember 2018, amazon.de).

Tabelle 2 zeigt die Verbesserungen auf:

Taktfrequenz: Von 1,2 GHz auf 1,4 GHz	Netzwerkanschluss: Von 10/100 MBit auf 10/100/1000 MBit	WLAN: Von 2,4 GHz auf 2,4 GHz / 5 GHz	Von Bluetooth 4.1 auf 4.2	PoE Unterstützung hinzugefügt (vorbereitet)

Tabelle 2: Technische Verbesserungen Raspberry Pi 3 Modell B+ [2]

1.1.4 Raspberry Sense HAT

Das Raspberry Sense HAT ist eine Erweiterungsplatine u. a. zur Messung von Luftfeuchtigkeit, Temperatur und Luftdruck. Außerdem besitzt die Platine eine Beschleunigungserkennung, einen Magnetfeld-Sensor, ein LED-Feld und einen Mini-Joystick. Für die vorliegende Arbeit werden zum Testen lediglich die Sensoren für die Messung von Luftfeuchtigkeit, Temperatur und Luftdruck verwendet. Abbildung 2 zeigt das Sense HAT, installiert auf einem Raspberry Pi 3 Modell B. Das Sense HAT wird über die 40 GPIO-Pins auf dem Raspi verbunden [8].

[1] www.generationrobots.com, *Raspberry Pi 3 Modell B*, URL: https://www.generationrobots.com/de/402366-raspberry-pi-3-modell-b.html [Aufgerufen am 29.12.2018]
[2] Manuel, *Raspberry Pi 3B+ und 3B im Vergleich*, URL: https://www.datenreise.de/raspberry-pi-3b-vs-3b-unterschied/ [Aufgerufen am 29.12.2018]

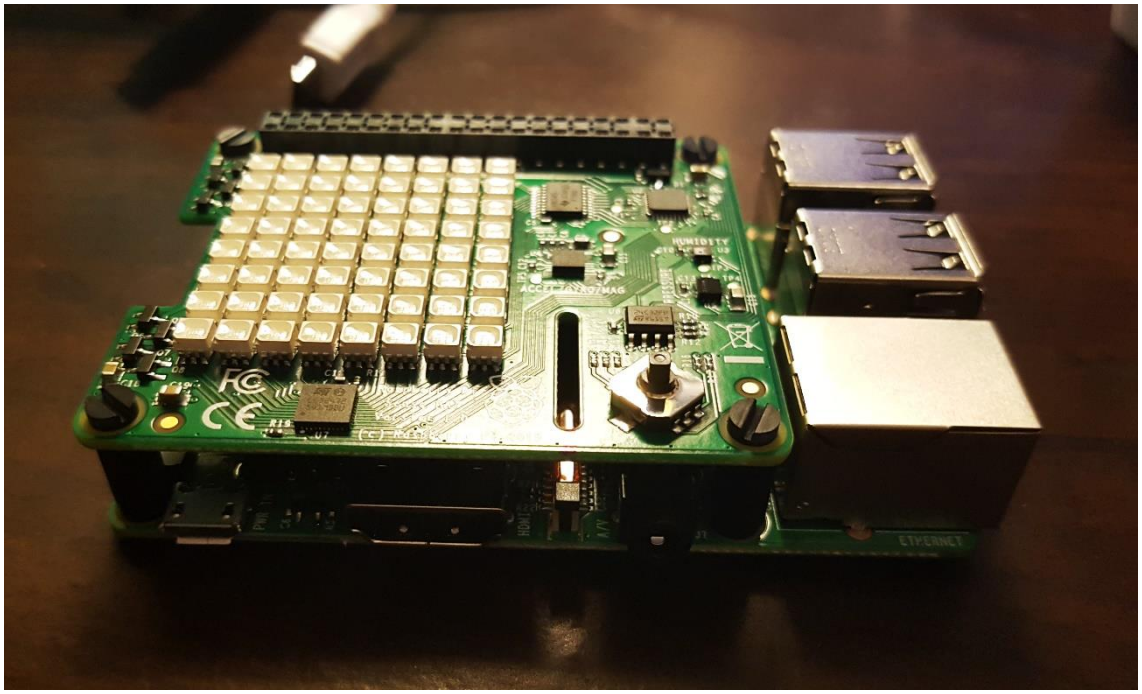


Abbildung 2: Sense HAT

1.2 Cloud Computing

Im Kapitel Cloud Computing wird das Cloud Computing im Allgemeinen, die Charakteristik von den Cloud-Arten sowie die Charakteristik von den verschiedenen Cloud-Diensten beschrieben.

1.2.1 Cloud Computing im Allgemeinen

Cloud Computing bezeichnet den Einsatz oder Bereitstellung von Anwendungen über das Internet (die „Cloud“). Bei Bedarf kann der Benutzer beispielsweise über einen Webbrowser auf die für ihn bereitgestellten Anwendungen zugreifen. Das hat den Vorteil, dass Anwendungen nicht mehr lokal auf den Rechner installiert werden müssen. Ein weiterer Vorteil ist die Skalierbarkeit für Unternehmen. Bei erhöhtem Ressourcenbedarf kann das Unternehmen zusätzliche Ressourcen dazu buchen. Die meisten Cloud Computing-Dienste sind nach dem „pay per use“-Modell ausgelegt, d. h. der Nutzer zahlt lediglich für den Gebrauch der Dienstleistung [9]. Dafür können Kosten für die Infrastruktur und Wartung von lokalen Datencentern eingespart werden [10].

1.2.2 Arten von Cloud Computing

Auf dem Markt existieren vier verschiedene Cloud-Computing-Arten, um individuelle Lösungen anzubieten und unterschiedliche Nutzer anzusprechen.

Public Cloud (öffentliche Cloud)

Bei einer Public Cloud stellt der Anbieter seine Leistungen für jeden berechtigten Nutzer im Internet zur Verfügung. Auch hier kann der Nutzer seine benötigten Ressourcen beliebig skalieren. Abgerechnet wird nach dem „pay per use“-Prinzip. Die Daten des Nutzers sind hierbei vor anderen Nutzern der öffentlichen Cloud gesichert, sodass diese keinen Zugriff auf vertrauliche Daten haben [11].

Private Cloud (private Cloud)

Die Private Cloud bezeichnet Dienste, die nur von bestimmten Benutzern im Internet oder Intranet in Anspruch genommen werden können. Charakteristisch ist die Private Cloud ähnlich wie die Public Cloud. Der Unterschied hier liegt darin, dass die Private Cloud oftmals im eigenen Rechenzentrum positioniert ist. Der Vorteil liegt in der Sicherheit und dem Datenschutz, da die Server im eigenen Haus hinter der eigenen Firewall liegen. Dies hat jedoch den Nachteil der Kosten für die Infrastruktur und Wartung [12].

Hybrid Cloud (Hybrid Cloud)

Die Hybrid Cloud ist die Fusion aus der Public Cloud und der Private Cloud, um mehr Flexibilität zu erreichen und die Vorteile der beiden Cloud Computing-Arten in sich zu kombinieren. Während datenschutzkritische Anwendungen über die Private Cloud verfügbar sind, können datenschutzunkritische Anwendungen durch die Public Cloud genutzt werden. So sind Einsparungen für zusätzliche Hardware möglich. Die Flexibilität wird erreicht, indem der Nutzer bzw. das Unternehmen Ressourcen nach Bedarf von der Public Cloud in die Private Cloud (oder umgekehrt) verschiebt [13].

Community Cloud

In einer Community Cloud können sich Unternehmen oder Organisationen, die die gleichen oder ähnliche Unternehmensziele verfolgen, zusammentun und ihre privaten Clouds zu einem Pool verbinden. Die Ressourcen und Anwendungen sind nur von den

Community-Mitgliedern zugänglich. Der Vorteil für ein Unternehmen sind die Vorzüge des Cloud-Computings wie bei einer öffentlichen Cloud zu genießen, ohne jedoch auf die Sicherheit einer privaten Cloud zu verzichten. Geographisch liegt die Community Cloud im eigenen Rechenzentrum oder außerhalb, bei einem externen Dienstleister (MSP) [14][15].

1.3 Arten von Cloud Computing-Diensten

Für das Cloud Computing werden verschiedene Dienste angeboten, um individuelle Lösungen zu finden. Im Unterkapitel Arten von Cloud Computing-Diensten werden diese verschiedenen Dienste erläutert.

IaaS (Infrastructure-as-a-Service)

Bei Infrastructure-as-a-Service werden Dienste wie Rechenleistung, Datenspeicher und Kommunikationsverbindungen von einem Anbieter bereitgestellt. Dadurch braucht der Benutzer keine Investitionen für seine eigene Hardware vornehmen und die Infrastruktur kann sich je nach Rechenauslastung dynamisch anpassen. [16] Da dieser Dienst dazu tendiert, Hardware-lastig zu sein, ist dieser Dienst auch unter der Bezeichnung Hardware as a Service (HaaS) bekannt[3].

Beispiele für Anbieter von IaaS:
AWS (Amazon Web Services)
Microsoft Azure
T-Systems Cloud Integration Center [17]

PaaS (Platform-as-a-Service)

Bei PaaS wird dem Benutzer eine komplette Cloud-basierte Infrastruktur zur Verfügung gestellt. Dazu zählen die Hardware, Software und das Hosting für die vom Benutzer selbstentwickelte Anwendung. Die Dienste PaaS und IaaS werden vom Anbieter weitgehend kombiniert angeboten [18].

[3] Vgl. Bedner, Mark: Cloud Computing: Technik, Sicherheit und rechtliche Gestaltung, Institut für Wirtschaftsrecht an der Universität Kassel, Band 14, Kassel: kassel university press GmbH 2013, S. 29

Beispiele für PaaS:
Amazon Elastic Beanstalk [18]
Microsoft Azure [19]
T-Systems Cloud Integration Center

SaaS (Software-as-a-Service)

Software-as-a-Service ist eine Dienstleistung von Anbietern, die ihre Anwendungen in der Cloud zur Verfügung stellen. Der Benutzer muss damit keine Software lokal auf seinen Computer installieren, sondern verbindet sich online über einen Webbrowser zur Anwendung. Der Benutzer muss sich lediglich registrieren, um die Leistungen des Anbieters wahrzunehmen und um seine Kontoinformationen zu speichern. Auch die Wartung und Bereitstellung der Dienste liegt in der Verantwortung des Anbieters [20][21].

Beispiele für Anbieter von SaaS:
Dropbox
Microsoft Office 365 [22]
SAP [23]

1.4 Cloud-Computing-Architekturen

Im Kapitel Cloud-Computing-Architekturen werden die Modelle Edge Computing und Fog Computing beschrieben.

1.4.1 Edge Computing

Edge Computing bezeichnet eine dezentralisierte Cloud-Computing-Architektur. Die generierten Daten (z. B. von lokalen Rechnern oder IoT-Geräten), werden unmittelbar am Gerät selbst oder in der Nähe, am Rand (engl. „Edge“) einer IT-Infrastruktur, verarbeitet. Dies entlastet die Netzbandbreite zu einem Rechenzentrum (die Cloud) [24]. Abbildung 3 stellt die Edge-Computing-Architektur grafisch dar.

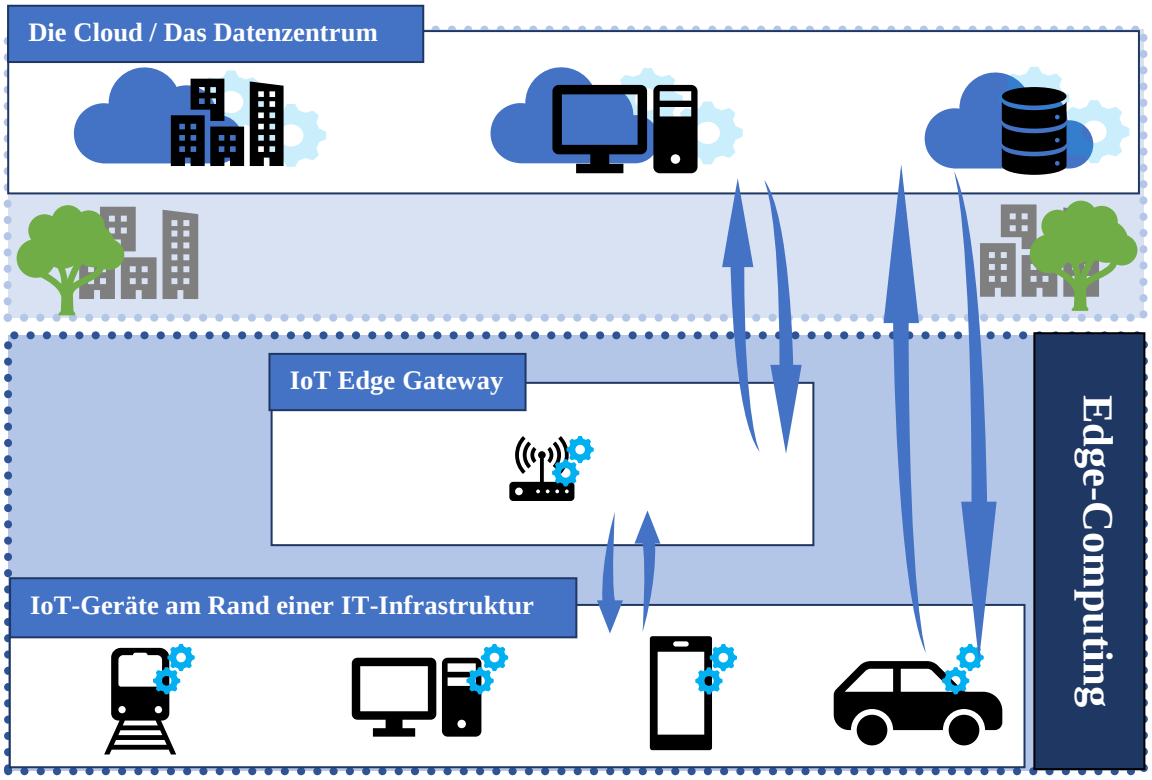


Abbildung 3: Edge-Computing-Architektur

IoT-Geräte am Rand einer IT-Infrastruktur:

Sensoren von IoT-Geräten generieren Unmengen an Daten pro Sekunde [2]. Abbildung 2 stellt den Anstieg und die Prognose der Erzeugung von zukünftigen Datenmengen für Echtzeitdaten in der globalen Datensphäre grafisch dar. So sollen Echtzeitdaten im Zeitraum von 2018 bis 2025 um ca. 23% steigen. Das ist ein Anstieg um ca. 42 ZB (aufgerundet 42.000.000.000.000 GB).

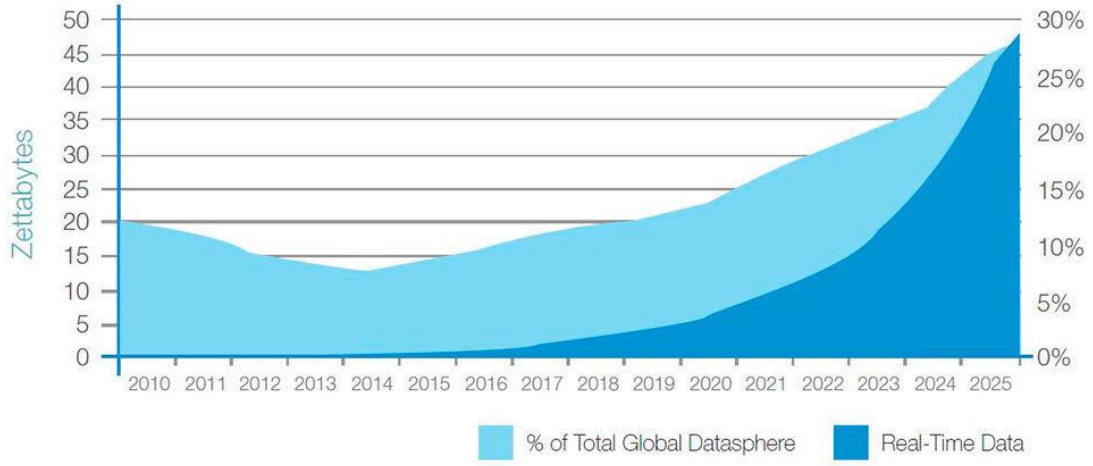


Abbildung 4: Die Gesamtmenge an Echtzeitdaten explodiert[4]. (Quelle: www.industry-of-things.de)

Zeitkritische IoT-Geräte benötigen die erzeugten Daten zeitnah für eine zügige Handlung [2]. Die vom IoT-Gerät generierten Daten werden am Gerät selbst oder am IoT Edge Gateway verarbeitet [25][26]. Die Informationen werden vorerst nach Dringlichkeit priorisiert und entsprechend weitergeleitet. Da nicht alle Daten im Datenzentrum bearbeitet werden, wird die Bandbreite entlastet und Latenzzeiten verringert. Somit können Entscheidungen annäherungsweise in Echtzeit getroffen werden. Die zeitkritischen Daten sind transient und werden gelöscht. Nur noch die für Analysen relevanten Informationen werden in die Cloud / das Datenzentrum gesendet [2].

IoT-Edge-Gateway:

Das IoT-Edge-Gateway kann eine Hardware oder eine Software sein. Es dient als Schnittstelle für die IoT-Geräte am Rand einer IT-Infrastruktur und die Cloud. Die Daten, die zwischen einem IoT-Gerät und der Cloud transferiert werden, verlaufen durch das IoT Edge Gateway, worin sie in lokaler Umgebung verarbeitet werden können. Die lokale Analyse minimiert die Menge an Daten, die in die Cloud wandern. Dadurch werden wiederrum die Bandbreitenauslastung und Latenzzeit verringert [27].

[4] Bildquelle: IDC´s Data Age 2025 study, sponsored by Seagate, [Online]. Available: https://www.industry-of-things.de/iot-basics-so-funktioniert-edge-computing-a-678225/ [Aufgerufen am 24. Dezember 2018]

Das IoT Edge Gateway bietet auch zusätzlichen Schutz an, indem es "Sicherheitsfunktionen für das IoT-Netz und den Datentransport"[5] übernimmt. Es gibt zwei verschiedene Gateway-Arten:

Field Gateway:
Geringenergie- oder einfache Geräte bieten evtl. keine Sicherheit für IoT. Beispielsweise kein SSL/TLS Unterstützung. Für diese Sicherheit ist das Field Gateway gedacht [28].

Protocol Gateway:
Das Protocol Gateway übersetzt ggf. Protokolle, falls das IoT-Gerät die Sprache der Cloud nicht unterstützt. Zum Beispiel wenn firmeneigene Protokolle oder veraltete Protokolle bei IoT-Geräten verwendet werden und die Cloud nur MQTT oder AMQP unterstützt [29].

Diese zwei Gateway-Arten können bei Bedarf miteinander kombiniert werden.

Die Cloud / Das Datenzentrum:
Die empfangenen Daten werden in die private Cloud / das Datenzentrum gesendet, analysiert und archiviert. [2] Sie fungiert als „zentrale Datensammelstelle für Business Analytics, Machine Learning und Prozesssteuerung“[6].

1.4.2 Fog Computing

Die Bezeichnung Fog Computing, die vom Unternehmen Cisco geprägt sein soll [30], stellt eine Erweiterung des herkömmlichen Cloud Computing-Modells her. Wie die Cloud (z. Dt. „Wolke“) soll der Fog (z. Dt. „Nebel“) ein Sinnbild für die Netzschicht in der Nähe des Bodens (der Rand des Netzwerks) sein und zugleich entfernter von den Wolken („Cloud“) [31]. Statt die von einem Gerät generierte Daten direkt in die Cloud oder das Datenzentrum zu senden, können die Daten durch Fog Computing schneller an sogenannte „Fog Nodes“ gesendet und dort weiterverarbeitet werden. Diese Fog Nodes

[5] itwissen.info (19.11.2017): *IoT-Gateway*. URL: https://www.itwissen.info/IoT-Gateway-IoT-gateway.html [Aufgerufen am 26. November 2018]
[6] Groß, Bernd (2018, 25. Juni): *Kanten, Wolken und das IoT*. URL: https://www.it-production.com/industrie-4-0-iot/edge-computing-cloud/ [Aufgerufen am 26. November 2018]

sind den IoT-Geräten physikalisch näher als die Cloud / das Datenzentrum. Abbildung 5 veranschaulicht das Prinzip von Fog Computing.

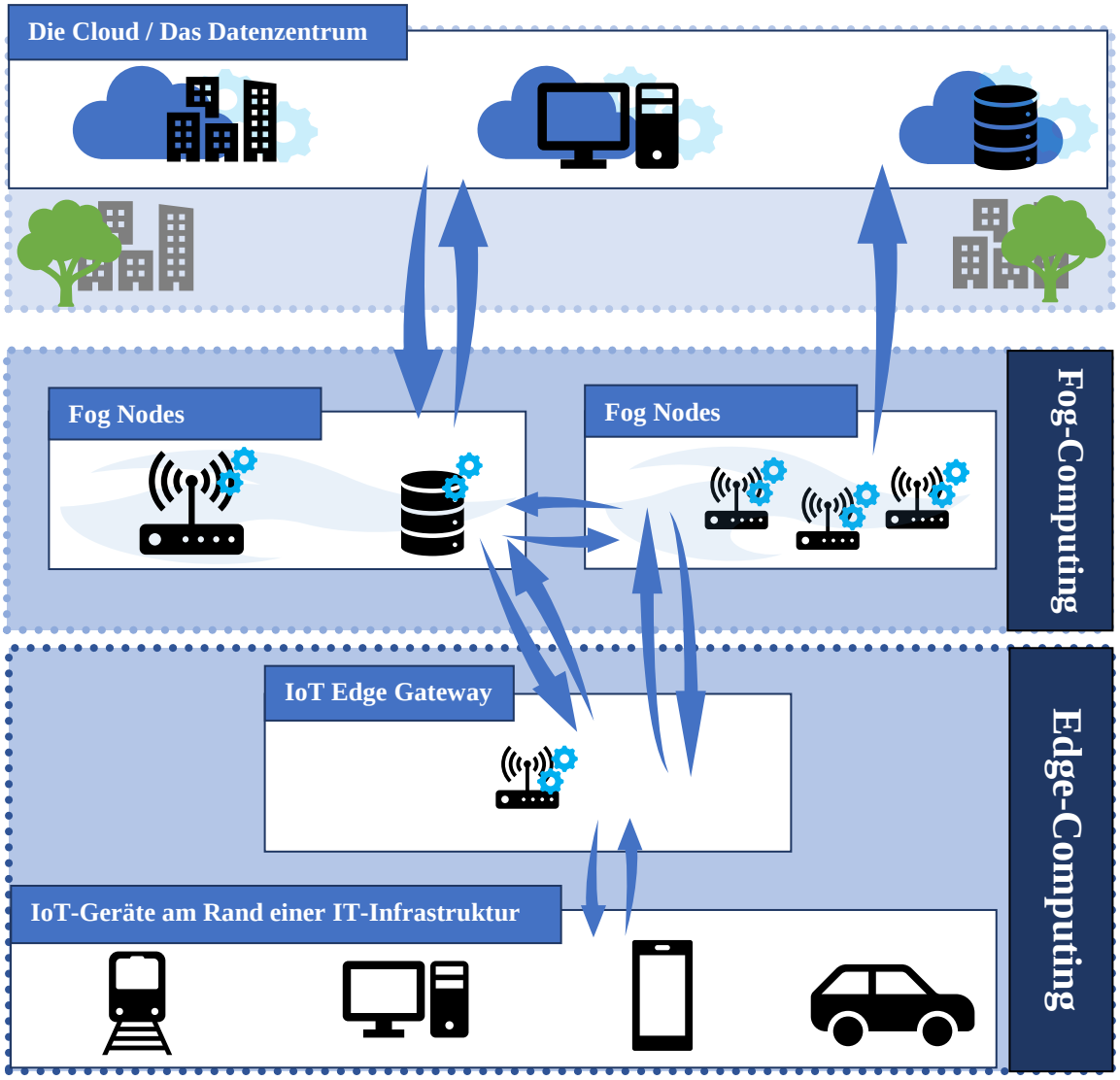


Abbildung 5: Fog-Computing-Architektur

Die unteren Pfeile auf Abbildung 5 verdeutlichen den Weg der Daten von IoT-Geräten, die am Rand einer IT-Infrastruktur sind, zum nächstliegenden Fog Node. Fog Nodes können z. B. Switches, Router, Überwachungskameras etc. sein. Die notwendigen Voraussetzungen für ein Gerät, welches als Fog Node fungieren kann, sind Datenverarbeitungsfähigkeit, Speicherkapazitäten und Netzwerk-Anschlussmöglichkeiten. Die Daten werden in den Fog Nodes analysiert und der Wichtigkeit entsprechend weitergeleitet. Die Wichtigkeit wird nach der Zeit, für die man eine Rückmeldung benötigt, beurteilt. Sind die Daten dringlich, werden sie in den Fog Nodes selbst analysiert, verarbeitet und wieder zurück an die IoT-Geräte gesendet.

Haben die Informationen jedoch keine Dringlichkeit, werden sie entweder zu anderen Fog Nodes oder zur Cloud / zum Rechenzentrum weitergeleitet und dort verarbeitet. In der Cloud können die Daten längerfristig gespeichert werden, wenn sie nicht relevant für aktuelle Geschehnisse in der IoT-Umgebung sind. Durch das Fog Computing wird die Auslastung der Bandbreite sowie die Latenzzeiten reduziert / minimiert. Zudem werden zügige Entscheidungen mittels den Fog Nodes erreicht. Auch die Sicherheit sensibler Daten wird gewährleistet, da sie im Umfeld des lokalen Netzwerks bleiben können [32].

1.4.3 Unterschied zwischen Edge Computing und Fog Computing

Der Unterschied zwischen Edge- und Fog-Computing liegt darin, dass die Verarbeitung der Daten bei Edge Computing am Gerät selbst oder physikalisch in der Nähe abläuft, während sich das Fog Computing auf eine Schnittstelle bezieht, die zwischen dem Gerät und der Cloud positioniert ist. Fog Computing stellt eine Erweiterung der Cloud dar, die sich in die Nähe der Geräte positioniert. Laut OpenFog wird beim Fog Computing auch immer Edge Computing genutzt, andersrum ist das nicht der Fall [33].

2 Edge Computing-Lösungen

Im Kapitel Edge Computing-Lösungen werden die zwei Edge Computing-Softwares Microsoft Azure IoT Edge und AWS Greengrass analysiert, tabellarisch miteinander verglichen und einzeln beschrieben.

2.1 Übersicht über die Unterschiede und Gemeinsamkeiten der Edge Computing Lösungen Azure IoT Edge und AWS Greengrass

Tabelle 3: Vergleich von Microsoft Azure IoT Edge und AWS IoT Greengrass.

	Microsoft Azure IoT Edge	AWS IoT Greengrass
API	• IoT-Edge-Laufzeit auf dem RPi • https://portal.azure.com/ • Tutorials und API-Referenzen unter https://docs.microsoft.com/de-de/azure/iot-edge/	• Diverse SDK's auf dem RPi • https://console.aws.amazon.com/ • Greengrass API-Referenz unter https://docs.aws.amazon.com/greengrass/latest/apireference/api-doc.html
Lizenzen	Kostenlose und nutzungsbasierte Lizenzen	Kostenlose und nutzungsbasierte Lizenzen
Preise (Stand: 13. Dezember 2018)	**Kostenlos**: Kostenloses Microsoft-Konto. 12 Monate kostenloser Test mit 170€ Startguthaben (Guthaben Gültigkeitszeitraum: 1 Monat) **Nutzungsbasierter Preis**: Upgrade von kostenlos auf diverse Pläne möglich. „Developer Plan": 24,46€/Monat. "Standard plan": 84,33€/Monat	**Kostenlos**: Kostenloses Konto. Das kostenlose Konto beinhaltet Produkte, die immer verfügbar sind und Produkte, die ab dem Registrierungsdatum nach 12 Monaten ablaufen. Bis zu drei Geräte kostenlos für 12 Monate. Ab drei Geräte nutzungsbasierter Preis. **Nutzungsbasierter Preis:** Für EU (Frankfurt) Kosten von 0,22 USD (Stand: Dezember 2018,

	"Professional Direct Plan": 843,30€ / Monat - IoT Hub Preislisten [34] - App-Service-Pläne [35] - Angebote an Produkten, die stündlich kosten (z. B. Virtuelle Maschinen, Azure SQL Datenbank, Azure Kubernetes Service…) - Container Registry Preise [36] - Bandbreiten Preise [37]	ca. 0,19 €) pro Monat und pro aktives AWS IoT Greengrass Gerät. Jahresvertrag möglich. Jährliche Verträge erhalten einen Rabatt von 22%. Zusätzliche Gebühren für andere AWS-Services oder Datenübertragungen. [38]
Programmiersprachen	• C • Java • Python • .NET Core 2.0 • Node.js [42]	• Sprachen mit C-Bibliotheken • C • C++ • Python 2.7 • Node.js 6.10 • Java 8 [39]
Kompatibilität mit dem Raspberry Pi 3 Modell B+	Ja	Nein Zurzeit nur Raspberry Pi 3 Model B (Stand 11/2018) [39]
Kompatibilität mit dem Raspberry Betriebssystem Raspbian Stretch	Ja (Raspbian Stretch Veröffentlichungsdatum 13.11.2018)	Nein (vgl. Anhang S. 66-67, Tabelle 8 und 9)

Für diese Bachelorarbeit wurde die Software-Lösung Microsoft Azure IoT Edge gewählt, da sie mit dem neuesten Betriebssystem für das Raspberry Pi 3 Model B + (Raspbian Stretch, Erscheinungsdatum 13.11.2018) kompatibel ist.

2.2 Microsoft Azure IoT Edge

Azure IoT Edge ist eine Ende-zu-Ende IoT-Plattform in der Public Cloud der Firma Microsoft [40]. Die Dienstleistungen von Azure IoT Edge sind in den Arten SaaS, PaaS und IaaS kategorisiert [41]. Microsoft Azure IoT Edge ist kompatibel mit den Betriebssystemen Linux und Windows. Die unterstützen Programmiersprachen sind C, Java, Python, .NET Core 2.0 und Node.js. Azure IoT Edge gibt es im kostenlosen Tarif und im Standardtarif. Der kostenlose Tarif ist für Tests und Bewertungen vorgesehen [42]. Abbildung 6 veranschaulicht die Architektur von Azure IoT Edge.

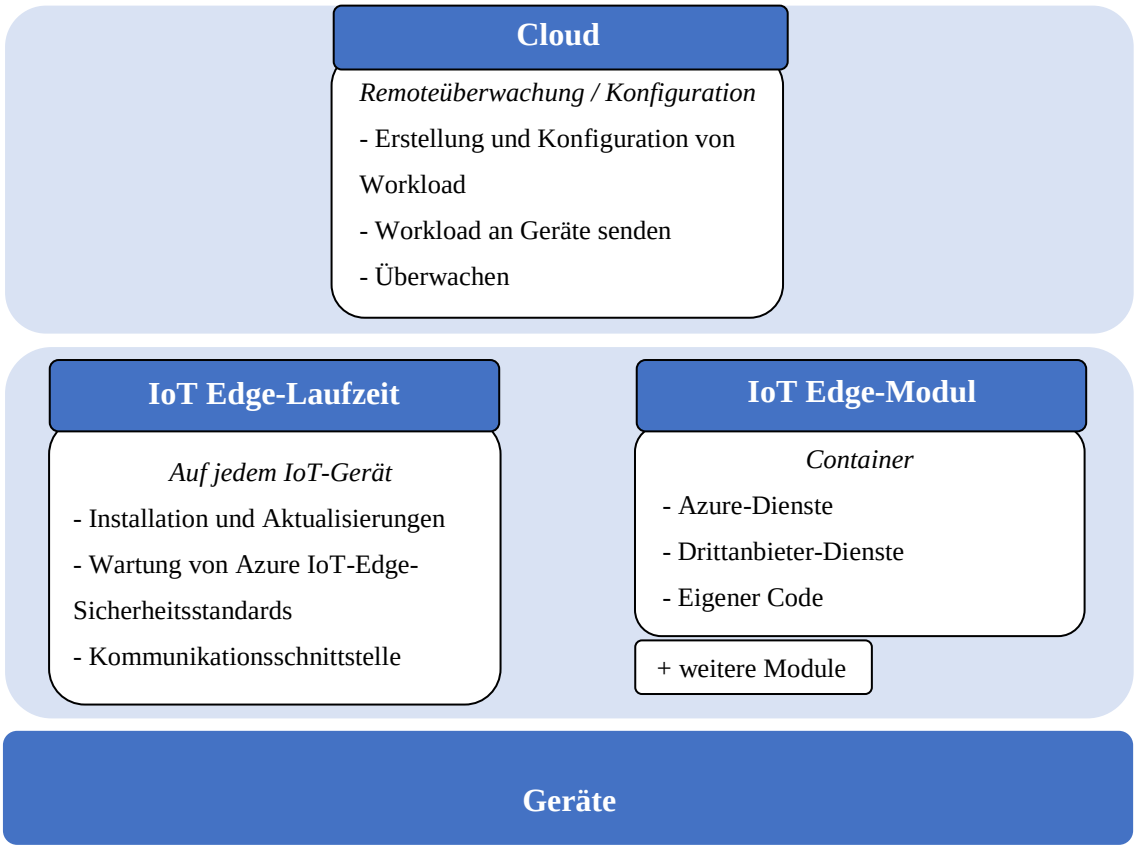


Abbildung 6: Azure IoT Edge Architektur

Die Cloud:

Die Cloud überwacht und verwaltet die IoT Edge-Geräte. Arbeitsvorgänge können in der Cloud für bestimmte Gruppen erstellt, konfiguriert und auch an die Geräte gesendet werden. Zusätzlich werden die Arbeitsvorgänge über die Cloud überwacht.

IoT Edge-Laufzeit:

Die IoT Edge-Laufzeit wird auf jedem IoT Edge-Gerät installiert. In der Laufzeit werden Arbeitsvorgänge installiert und aktualisiert, Sicherheitsstandards überprüft, Module verwaltet und überwacht. Sie fungiert als Kommunikationsschnittstelle zwischen nachgeschalteten Blattkonten und dem IoT Edge-Gerät, zwischen Modulen auf einem IoT Edge-Gerät oder zwischen einem IoT Edge-Gerät und der Cloud.

IoT Edge-Modul:

Ein IoT Edge-Modul führt Funktionen der Geschäftslogik aus. Module können von Azure, Drittanbietern oder einem selbst stammen. Dazu werden Code und Logik in ein solches Modul verpackt. Die Module können untereinander kommunizieren, auch wenn keine Internetverbindung besteht. Zusätzlich bietet Microsoft Azure IoT Edge die Nutzung von künstlicher Intelligenz an. Die KI-Module können von Azure oder von anderen Benutzern bezogen werden. Eigene KI-Module können auch erstellt und für andere Benutzer zur Verfügung gestellt werden. Beispiele für KI: Ereignisverarbeitung, Machine Learning oder Bilderkennung [42].

2.3 AWS (Amazon Web Services) Greengrass

AWS Greengrass ist eine Edge Computing-Plattform vom Cloud Computing-Anbieter Amazon Web Services. Greengrass ist in der Public Cloud positioniert [43] und ist in den Kategorien SaaS, PaaS und IaaS eingeordnet. Abbildung 7 zeigt die Architektur von AWS Greengrass.

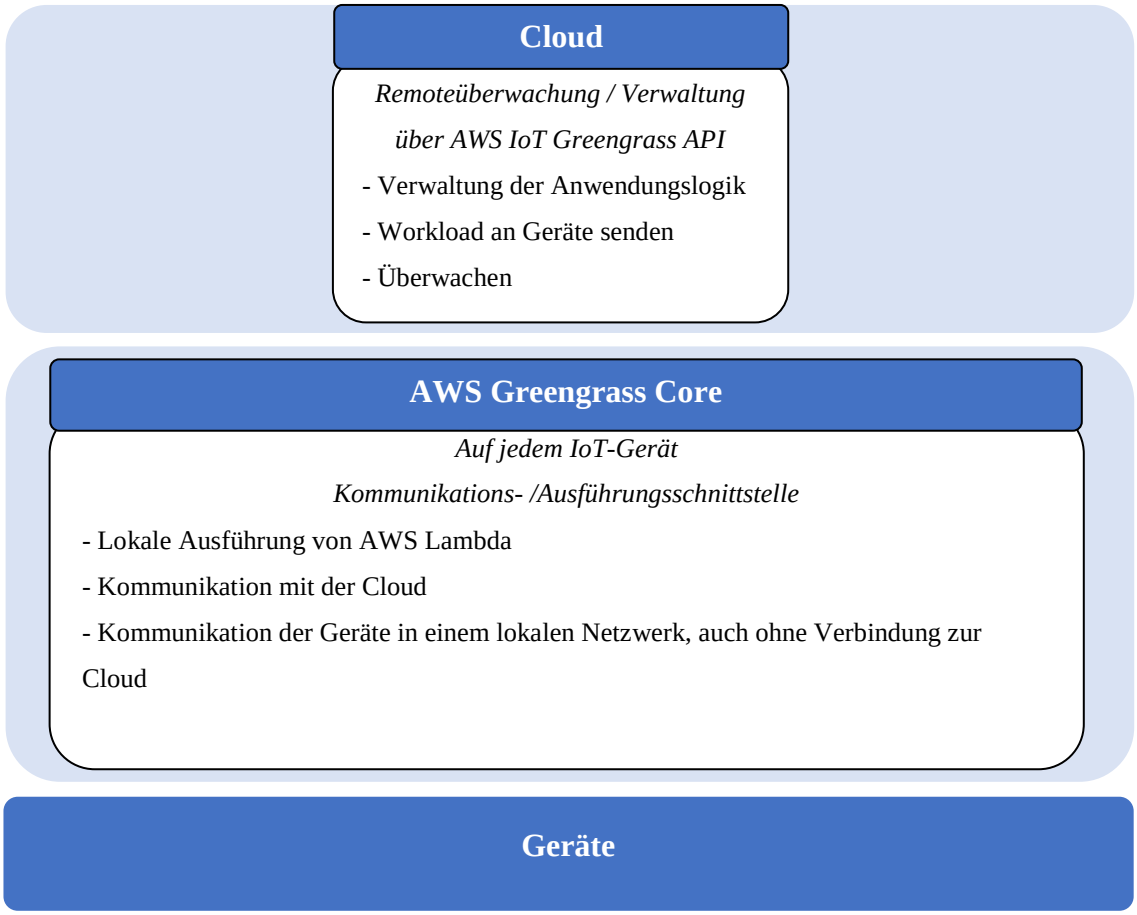


Abbildung 7: AWS Greengrass Architektur

Die Cloud:

In der AWS Greengrass API werden Anwendungslogik erstellt und von der Cloud an die Geräte gesendet / bereitgestellt [44].

AWS Greengrass Core:

Der AWS Greengrass Core dient als Kommunikationsschnittstelle zwischen der Cloud und den IoT-Geräten. Geräte, auf denen AWS IoT Greengrass Core und die AWS IoT Device SDK (Geräte-SDK) installiert sind, können miteinander kommunizieren und bilden eine Greengrass-Gruppe [43]. Der AWS IoT Greengrass Core führt sogenannte Lambda-Funktionen aus. Lambda-Funktionen sind, analog zu den Modulen bei Azure IoT Edge, Funktionen, die die eigene Geschäftslogik beinhalten und ausführen können. Diese Lambda-Funktionen sind ereignisabhängig, d. h. sie werden durch lokale Ereignisse, Aufträge aus der Cloud oder anderen Ereignissen ausgelöst [44].

AWS Greengrass Softwares und SDKs

AWS Greengrass bietet die Softwares „AWS IoT Greengrass-Core-Software“ und „AWS IoT Greengrass-Core-SDK“ an. Zusätzlich gibt es noch zwei weitere Arten von SDKs: AWS-SDK und AWS IoT-Geräte-SDK. Die AWS-SDK wird benötigt, um Anwendungen zu erstellen oder verfügbare Lambda-Funktionen zu nutzen. Die AWS IoT-Geräte-SDK wird verwendet, damit die Geräte Informationen voneinander erhalten und Verbindungen mit AWS-IoT- oder AWS-Greengrass-Services herstellen können. Die lokalen Anwendungen werden in der AWS Greengrass Core-Software implementiert und über die API ausgeführt. Die AWS Greengrass Kern-Software verwaltet den lokalen Nachrichtenaustausch zwischen Geräten über das MQTT-Protokoll. Über die AWS Greengrass-Gruppen können Geräte innerhalb einer Gruppe konfiguriert werden. So sind Konfigurationseinstellungen wie z.B. Gruppen-Rollen, Protokollkonfigurationen, Zertifizierungsstelle und Konfiguration der lokalen Verbindung möglich. Zusätzlich können Geräte innerhalb einer Gruppe mit AWS-Greengrass-Verbindungsinformationen belegt werden und Nachrichten untereinander austauschen [44]. Zudem bietet AWS Greengrass ein Machine Learning SDK an, welche über die Greengrass Core von Lambda-Funktionen verwendet werden kann [45].

3 Fog-Computing-Lösung

Im Kapitel Fog-Computing-Lösung wird die Fog-Computing-Lösung FogLAMP aufgezeigt.

3.1 FogLAMP

FogLAMP steht als Open-Source-Plattform zur Verfügung [46]. Unterstützt wird das FogLAMP Projekt durch Firmen wie OSIsoft, Arm, Arrow Electronics, BTE Networks Inc., Dianomic, Panduit, Sprint und Toshiba [47]. FogLAMP bietet eine skalierbare Infrastruktur zum Sammeln der Daten von Sensoren. Daten können bereits am Rand der IT-Infrastruktur verarbeitet und/oder zur Cloud weitergeleitet werden [48]. FogLAMP besteht aus mehreren Microservices, die in Abbildung 8 dargestellt sind.

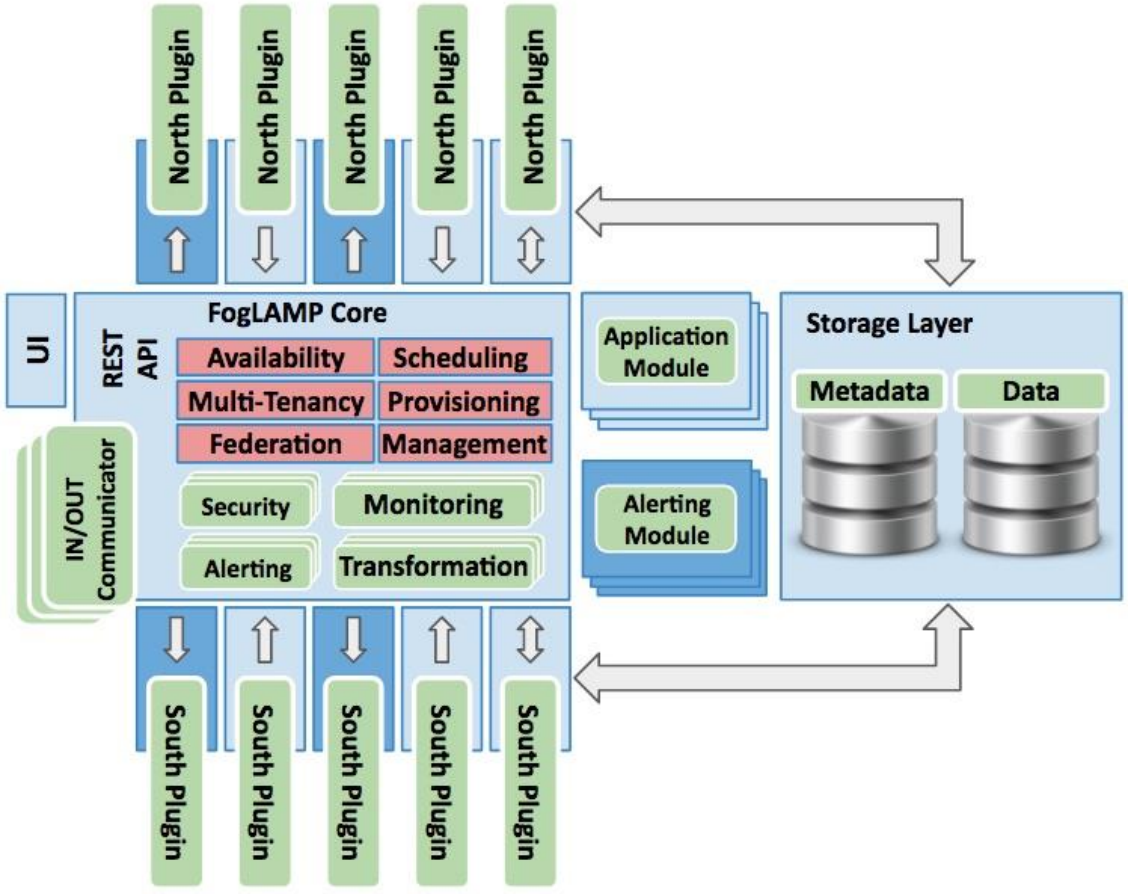


Abbildung 8: FogLAMP-Architektur[7]

[7] Bildquelle: Dianomic, „FogLAMP – Dianomic“, [Online]. Available: http://dianomic.com/platform/foglamp/

North Plugin:

Über die North Plugins werden Daten zu einem Server oder einer Dienstleistung in einer Cloud / einem Datencenter gesendet bzw. empfangen. Aktuell (Stand: Dezember 2018) gibt es zwei North Plugins: OSIsoft PI Server und OSIsoft Cloud Service (OCS) [49].

FogLAMP Core:

FogLAMP Core koordiniert die Abläufe und Ressourcen für FogLAMP Kernaufgaben [50].

South Plugin:

Über die South Plugins kommuniziert die Plattform mit Sensoren und Aktoren [51].

Storage Layer / Storage Plugin:

Über den Storage Plugin kann die Plattform die Storage Microservices nutzen. Dies ermöglicht die Speicherung transienter oder persistenter Daten für FogLAMP. Derzeit (Stand: Dezember 2018) gibt es zwei Storage Plugins: SQLite Plugin und PostgreSQL Plugin [52].

4 Installationsanleitung und Bedienung von Microsoft Azure IoT Edge

Im Kapitel Installation von Microsoft Azure IoT Edge wird eine detaillierte Installationsanleitung für Microsoft Azure IoT Edge auf den Einplatinencomputer Raspberry Pi 3 Model B+ dargelegt. Diese Anleitung lehnt sich an die Anleitung auf der offiziellen Webseite an [53][54]. Die in diesem Kapitel verwendeten Codes stammen ebenfalls aus der offiziellen Webseite und wurden für die vorliegende Arbeit lediglich zum Testen genutzt.

4.1 Vorbereitungen

4.1.1 Installation des Betriebssystems Raspbian Stretch

Für diese Anleitung wird das Betriebssystem Raspbian Stretch auf dem Raspberry Pi 3 Model B+ installiert.

In diesem Unterkapitel verwendete Hardware:

- Raspberry Pi 3 Model B+
- MicroSD-Karte 16 GB
- MicroSD-Kartenlesegerät
- Computer mit Betriebssystem Windows 10

Verwendete Software:

- SD Memory Card Formatter
- Betriebssystem Raspbian Stretch
- Win32 Disk Imager
- Packprogramm (WinRAR)

Schritt 1: MicroSD-Karte mit dem Computer verbinden.

Die MicroSD-Karte wird in das Kartenlesegerät eingesteckt und anschließend mit dem Computer verbunden.

Schritt 2: SD Memory Card Formatter.

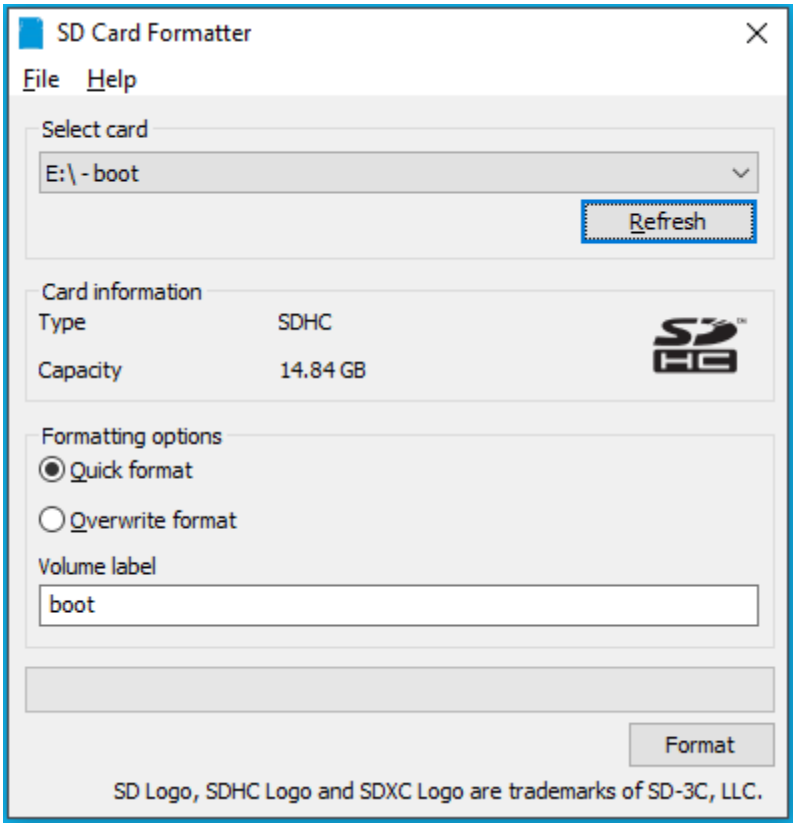


Abbildung 9: Bildschirmaufnahme SD Memory Card Formatter

Das Programm SD Memory Card Formatter wird gestartet und bei „Select card“ wird ausgewählt, welches Laufwerk zu formatieren ist. „Quick format“ bei „Formatting options“ wird ausgewählt, um die MicroSD-Karte zügig zu formatieren. Im Abschnitt „Volume label“ kann ein beliebiger Name für die Speicherkarte ausgewählt werden. Anschließend klickt man auf „Format“. Das Programm formatiert nun das ausgewählte Laufwerk.

Schritt 3: Aufspielen von Raspbian Stretch auf die Speicherkarte.

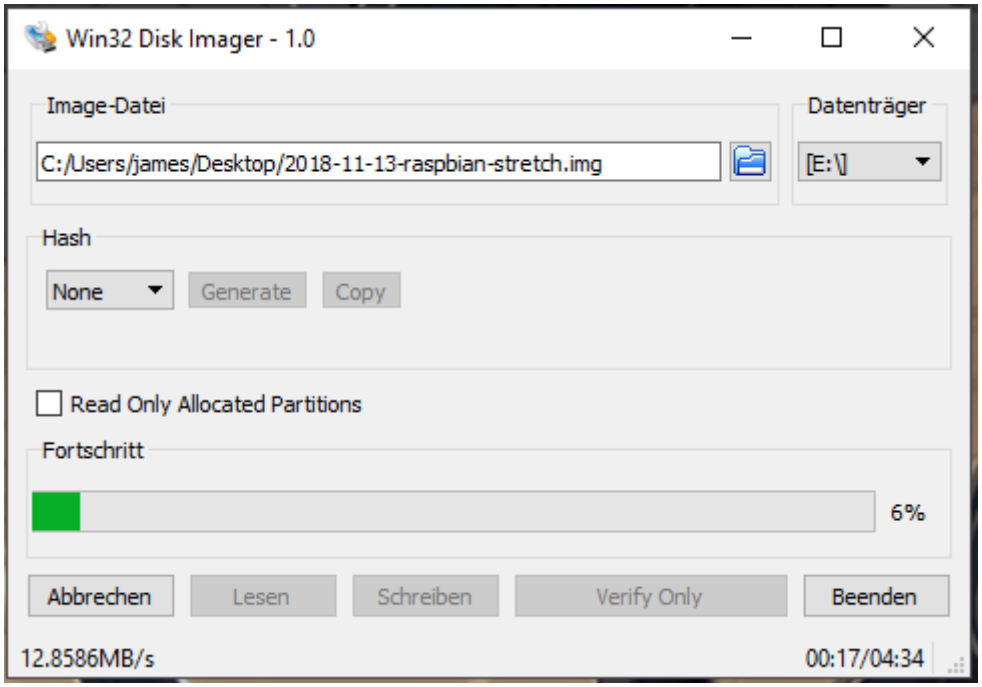


Abbildung 10: Bildschirmaufnahme Win32 Disk Imager - 1.0

Das Programm Win32 Disk Imager wird gestartet, um das Betriebssystem auf die MicroSD-Karte zu schreiben.

Bei „Image-Datei“ wird die heruntergeladene Datei `2018-11-13-raspbian-stretch.img` ausgewählt und anschließend überprüft, ob der richtige Datenträger selektiert wurde. Nun kann man auf „Schreiben“ klicken und die Überschreibung im nächsten Prompt bestätigen. Dieser Prompt warnt, dass das Schreiben auf ein physikalisches Gerät dieses beschädigen kann. Nach dem Schreibeprozess, der einige Minuten dauern kann, ist die Speicherkarte sicher zu entfernen.

Schritt 4: Betriebssystem auf den Raspberry Pi aufspielen.

Die Speicherkarte wird in den Raspberry Pi eingesteckt. An den Raspberry Pi wird eine Maus, eine Tastatur und ein Monitor angeschlossen. Anschließend wird er mit Strom versorgt, um die automatische Installation des Betriebssystems anzustoßen. Sobald das Betriebssystem hochgefahren ist, kann der Raspberry Pi konfiguriert werden.

4.1.2 Konfiguration des Raspberry Pi

Das Unterkapitel Konfiguration des Raspberry Pi beschreibt die Konfiguration für eine zweckmäßige Nutzung.

Nach Hochfahren des Betriebssystems wird zuerst auf das Raspberry Symbol geklickt und daraufhin auf *Preferences -> Raspberry Pi Configuration.*
Folgende Einstellungen werden ausgewählt:

Locale: "Set Locale" -> Language: de (German) -> Country: DE (Germany) -> `Character Set`: UTF-8
Timezone: „Set Timezone…" -> Area: Europe -> Location: Berlin
Keyboard: "Set Keyboard…" -> Country: German / Variant: German
WiFi Country: "Set WiFi Country..:" -> DE Germany

4.2 Installation von Azure IoT Edge

Im Unterkapitel Installation von Azure IoT Edge wird eine Anleitung zur Installation von Microsoft Azure IoT Edge auf den Raspberry Pi 3 Model B+ dargelegt.

In diesem Unterkapitel verwendete Programme:

- PuTTY für die Verbindung vom Windows Computer zum Raspi
- Webbrowser für die Anmeldung ins Azure Portal

Schritt 1: Anmeldung / Registration Azure Portal.

Für die Benutzung des Azure Portals[8] wird ein Microsoft-Konto benötigt.

Schritt 2: Verbindungsherstellung vom Computer zum Raspberry Pi via PuTTY.

Auf dem Raspi: Im Terminal wird der Befehl `hostname -I` eingegeben, um die IP-Adresse des Raspberry Pis auszulesen.
Auf dem Windows PC: Das Programm PuTTY wird geöffnet und die IP-Adresse des Raspberry Pis im Feld „Host Name (or IP address)" eingegeben. Port 22 wird eingetragen und SSH in „Connection type" ausgewählt. Die Verbindung wird mit dem

[8] https://portal.azure.com/

Schaltfeld „Open“ hergestellt. Der nächste Prompt mit der Sicherheitswarnung wird mit „Ja“ bestätigt. Daraufhin öffnet sich das Terminal. Standardmäßig sind folgende Daten einzugeben:

„login as“: pi

„pi@192.168.1.6's password“: raspberry

Schritt 3: Installation der Edge-Laufzeit auf dem Raspberry Pi.

Der Code muss kopiert und in das Terminal von Raspi eingefügt werden.

Herunterladen und Installation der Moby-Engine (Container-Laufzeit)

```
curl -L https://aka.ms/moby-engine-armhf-latest -o moby_engine.deb && sudo dpkg -i ./moby_engine.deb
```

Herunterladen und Installation der Moby-CLI

```
curl -L https://aka.ms/moby-cli-armhf-latest -o moby_cli.deb && sudo dpkg -i ./moby_cli.deb
```

Ausführung des apt-get Fixes

```
sudo apt-get install -f
```

Schritt 4: Installation des IoT Edge Security Daemons.

Herunterladen und Installation der Standard libiothsm-Implementierung

```
curl -L https://aka.ms/libiothsm-std-linux-armhf-latest -o libiothsm-std.deb && sudo dpkg -i ./libiothsm-std.deb
```

Herunterladen und Installation des IoT Edge Security Daemons

```
curl -L https://aka.ms/iotedged-linux-armhf-latest -o iotedge.deb && sudo dpkg -i ./iotedge.deb
```

Ausführung des apt-get Fixes

```
sudo apt-get install -f
```

Schritt 5: Erstellung einer Ressourcengruppe über das Azure Portal.

In diesem Schritt gibt es zwei Methoden, um eine Ressourcengruppe zu erstellen. Schritt **5a** beschreibt das Erstellen einer Ressourcengruppe über die GUI und Schritt **5b** über die Azure CLI.

5a) Erstellung einer Ressourcengruppe über die GUI

Über einen Webbrowser wird das Azure Portal (https://portal.azure.com) geöffnet.

Links auf *Ressourcengruppen −> Ressourcengruppe erstellen* klicken.

Ressourcengruppenname: Hier kann ein beliebiger Name für die Ressourcengruppe ausgewählt werden.

Abonnement: Free Trial

Ressourcengruppenstandort: Westeuropa

Anschließend auf *Erstellen* klicken.

5b) Erstellung einer Ressourcengruppe über die Azure CLI

Die Azure CLI kann über die GUI oben rechts geöffnet werden. (Abbildung 11)

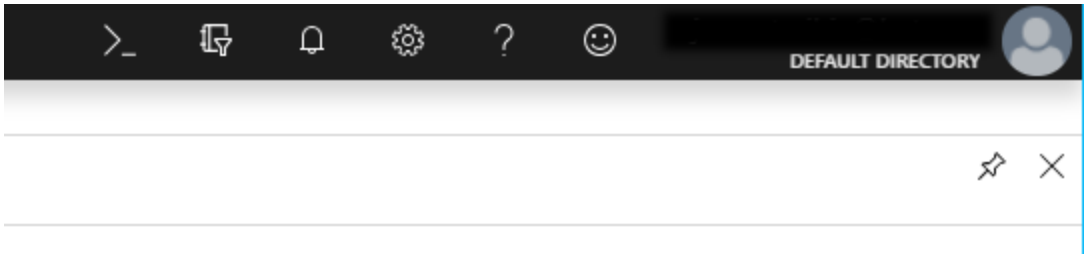


Abbildung 11: Bildschirmaufnahme Azure Portal: Azure CLI

Es besteht die Möglichkeit Bash oder PowerShell für die Kommandozeile zu benutzen. Bei Abonnement wird *Free Trial* gewählt und anschließend auf *Speicher erstellen* geklickt. Für eine Erweiterung der Azure IoT-Funktionalitäten muss folgender Code in die CLI eingegeben werden:

```
az extension add --name azure-cli-iot-ext
```

Anschließend wird in die Kommandozeile folgendes eingegeben, um eine Ressourcengruppe zu erstellen:

```
az group create --name <RESSOURCEN_GRUPPENNAME> --location <REGION>
```

Ein Beispiel, um eine Ressourcengruppe mit dem Namen „jt-ressource" in der Region „Westeuropa" zu erstellen:

```
az group create --name jt-ressource --location westeurope
```

Schritt 6: Erstellung eines IoT Hubs über das Azure Portal.

In diesem Schritt gibt es zwei Methoden, um ein IoT Hub zu erstellen. Schritt **6a** beschreibt das Erstellen eines IoT Hubs über die GUI und Schritt **6b** über die Azure CLI.

6a) Erstellung eines IoT Hubs über die GUI

Im Azure Portal wird *Ressource erstellen -> Internet der Dinge (IoT) -> IoT Hub gewählt*.

Folgende Einstellungen werden vorgenommen:

Basics

Subscription: Free Trial

Resource Group: jt-ressource

Region: Westeuropa

IoT Hub Name: Hier kann ein beliebiger IoT Hub Name eingegeben werden. In diesem Beispiel: jt-hub

Size and scale

Pricing and scale tier: F1: Free tier

Review + create

Überprüfen und anschließend auf *Create*, um den IoT Hub zu erstellen.

6b) Erstellung eines IoT Hubs über die Azure CLI

In der Azure CLI wird der folgende Code eingegeben, um ein IoT Hub zu erstellen:

```
az iot hub create --resource-group <RESSOURCEN_NAME> --name <HUB_NAME>
--sku F1
```

Ein Beispiel, um einen IoT Hub mit dem Namen „jt-hub", in der Ressourcengruppe „jt-ressource" und der Preiskategorie F1 (kostenlos) zu erstellen:

```
az iot hub create --resource-group jt-ressource --name jt-hub --sku F1
```

Schritt 7: Registration eines IoT Edge-Geräts.

In diesem Schritt gibt es zwei Methoden, um ein IoT Edge-Gerät im IoT Hub zu registrieren. Schritt **7a** beschreibt die Registration eines IoT Edge-Geräts über die GUI und Schritt **7b** über die Azure CLI.

7a) Registration eines Geräts über die GUI

Im Azure Portal auf *Alle Ressourcen -> <HUB_NAME> / jt-hub -> IoT Edge -> Add an IoT Edge device* gehen.

Device ID: Hier kann ein beliebiger Name für die Identität eines Geräts festgelegt werden. In diesem Beispiel für das Raspberry Pi: jt-raspi.

Zum Schluss auf *Save* klicken. Nach der Speicherung wird das Gerät angezeigt. Anschließend wird auf das Gerät geklickt und der „*Connection string (primary key)“* muss extern gespeichert werden. Es muss sichergestellt werden, dass „*Connect this device to an IoT hub“* auf „*Enable“* geschaltet ist.

7b) Registration eines Geräts über die Azure CLI

In der Azure CLI wird folgendes eingegeben, um ein Gerät zu registrieren:

```
az iot hub device-identity create --hub-name <HUB_NAME> --device-id
<GERÄTE_NAME> --edge-enabled
```

Ein Beispiel, um ein Gerät mit dem Namen „jt-raspi“ im IoT Hub „jt-hub“ zu registrieren:

```
az iot hub device-identity create --hub-name jt-hub --device-id jt-
raspi --edge-enabled
```

Nach der Registration muss der „connection string“ (Verbindungsschlüssel) des Gerätes ausgelesen, kopiert und gespeichert werden. Folgender Befehl liest den Schlüssel für das obige Beispiel aus:

```
az iot hub device-identity show-connection-string --device-id jt-raspi
--hub-name jt-hub
```

Schritt 8: Verbindungsaufbau des Geräts zum IoT Hub.

Auf dem Raspberry Pi:

Die Datei, die geändert werden muss, liegt im Pfad /etc/iotedge/config.yaml. Es kann ein beliebiger Texteditor für diesen Schritt benutzt werden. In diesem Beispiel wird der VI-Editor verwendet. Im Terminal wird folgendes eingegeben:

```
sudo vi /etc/iotedge/config.yaml
```

Mit den Pfeiltasten bis zum folgenden Punkt navigieren:

```
provisioning:
  source: "manual"
  device_connection_string: "<ADD DEVICE CONNECTION STRING HERE>"
```

Bei "`<ADD DEVICE CONNECTION STRING HERE>`" den oben kopierten Verbindungsschlüssel einfügen: Die Zeichen `<ADD DEVICE CONNECTION STRING HERE>` mit der Taste `x` löschen. Dann `i` drücken, um in den Eingabemodus zu gelangen und mit der rechten Maustaste, zwischen den beiden Anführungszeichen, den kopierten Verbindungsschlüssel einfügen. Anschließend ESC-Taste drücken und `:wq!` eingeben, um die Änderungen zu Speichern und den Editor zu verlassen.

Nach dieser Änderung muss der Daemon neugestartet werden:

```
sudo systemctl restart iotedge
```

Nützliche Befehle zur Überwachung:

Status des IoT Edge Daemons anzeigen lassen:

```
systemctl status iotedge
```

Daemon logs anzeigen lassen:

```
journalctl -u iotedge --no-pager --no-full
```

Laufende Module anzeigen lassen:

```
sudo iotedge list
```

Damit ist das Gerät bereit, Module aus der Cloud auszuführen.

4.3 Bereitstellung eines Moduls aus der Cloud

Im Unterkapitel Bereitstellung eines Moduls aus der Cloud wird ein Test-Modul zur Simulation eines Temperatursensors aus dem Azure-Marktplatz [55] heruntergeladen und über den IoT Hub (die Cloud) auf dem Gerät bereitgestellt.

Zuerst wird der Azure-Marktplatz in einem Webbrowser geöffnet und anschließend in der Suchleiste nach der Anwendung „Simulated Temperature Sensor“ gesucht.

Anschließend wird diese Anwendung mit der Schaltfläche „Get it now“ zum Account hinzugefügt. Die Nutzungsbedingungen müssen akzeptiert werden, um ins Azure Portal weitergeleitet zu werden.

Bei „Target Devices for IoT Module” wird folgendes ausgewählt und eingetragen:

Abonnement: *Free Trial*

IoT-Hub*: jt-hub*

Deploy to a device

IoT Edge Device Name: *jt-raspi* -> Find Device -> Available IoT Edge Devices: *jt-raspi* -> Select -> Create.

Im Anschluss auf *Next* klicken und verifizieren, ob folgender JSON-Code vorhanden ist:

```
{
  "routes": {
    "route": "FROM /messages/* INTO $upstream"
  }
}
```

Danach auf *Next* -> *Submit.*

Das Gerät erhält nun neue Bereitstellungsinformationen aus der Cloud, startet die Container-Prozesse und gibt der Cloud eine Rückmeldung über seinen aktuellen Status. Dieser Vorgang kann einige Minuten dauern. Zur Statusüberprüfung ist auf den Gerätenamen zu klicken.

4.4 Überwachung der Daten

Im Unterkapitel Überwachung der Daten werden Befehle und ein Programm (Visual Studio Code, auch VS Code oder VSC genannt) [56] für die Überwachung der vom Raspberry Pi erzeugten Daten aufgezeigt.

Befehle auf dem Raspberry Pi (Terminal):

Anzeigen, welche Module aktiv sind:

```
sudo iotedge list
```

Anzeigen, welche Nachrichten vom SimulatedTemperatureSensor-Modul in die Cloud versendet wird:

```
sudo iotedge logs SimulatedTemperatureSensor -f
```

Falls in dem Log die letzte Zeile „Using transport Mqtt_tcp_only“ länger angezeigt wird, kann es helfen das Modul neuzustarten.

Neustarten eines Moduls:

```
sudo docker stop <Modul_Name>
```

Beispiel:

```
sudo docker stop <SimulatedTemperatureSensor>
```

Das Modul SimulatedTemperatureSensor wird beendet und nach einem kurzen Moment erneut automatisch neugestartet.

Befehle auf der Azure CLI:

Überwachung der empfangenen Nachrichten:

```
az iot hub monitor-events --device-id jt-raspi --hub-name jt-hub
```

Die vom Gerät erhaltenen Nachrichten werden in der Konsole angezeigt.
Zum Beenden der Überwachung ist Strg + c zu drücken.

Windows-Programm zur Überwachung von empfangenen Daten aus dem IoT Edge-Gerät:

Für die Überwachung mittels eines Programms wird Visual Studio Code verwendet. Zusätzlich wird noch die Erweiterung Azure IoT Hub Toolkit für Visual Studio Code benötigt. [57]

Visual Studio Code starten, in der Explorer-Ansicht unten links auf Azure IoT Hub Devices klicken und anschließend Select IoT Hub wählen. Es öffnet sich ein kleines Pop-Up, in dem der Benutzer sich mit seinem Microsoft-Konto anmelden muss. Die nächsten Schritte der Anweisung des Anmeldefensters befolgen, um die Anmeldung erfolgreich abzuschließen.

In der Mitte von VSC erscheint ein Eingabefeld, in dem nacheinander folgende Daten eingegeben bzw. ausgewählt werden:

Select Subscription: Hier das Abonnement auswählen -> *Free Trial*.

Select IoT Hub: Hier das erstellte IoT Hub auswählen -> *jt-hub*.

Das Gerät ist nun unter Azure IoT Hub Devices zu finden. Um die Überwachung zu starten, ist dazu mit der rechten Maustaste auf den Gerätenamen zu klicken und „Start Monitoring D2C Message“ zu wählen.

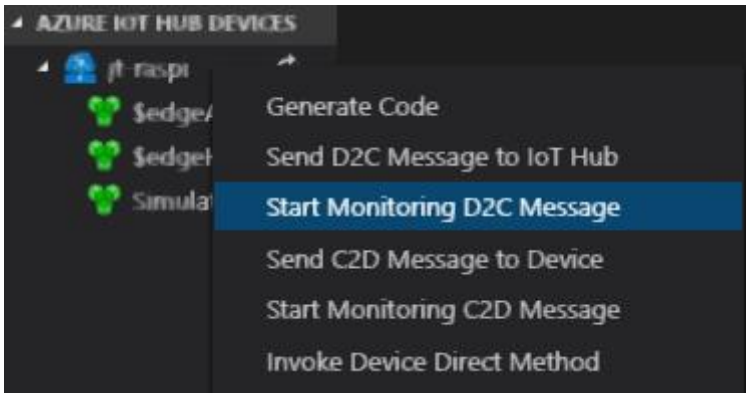


Abbildung 12: Bildschirmaufnahme VSC: Überwachung von D2C-Nachrichten

In der Output-Konsole „Azure IoT Hub Toolkit“ werden die empfangenen Daten angezeigt (Abbildung 13):

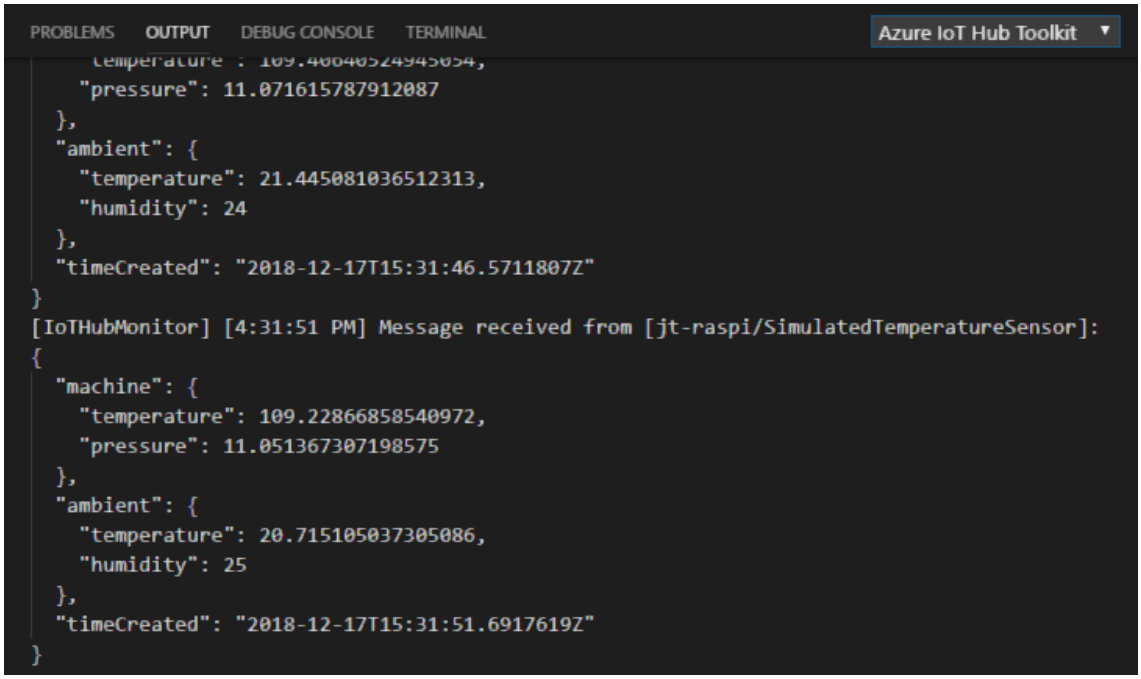


Abbildung 13: Bildschirmaufnahme VSC: Ausgabe empfangener Daten

Der Benutzer kann die Überwachung durch einen Rechtsklick in der Konsole und die Auswahl des Menüpunktes „Stop Monitoring D2C Message“ stoppen.

4.5 Entwickeln und Bereitstellen eines eigenen Python IoT Edge Moduls

In den vorherigen Kapiteln wurde aufgezeigt, wie ein IoT Edge-Gerät Daten generiert und diese anschließend in die Cloud weiterleitet. Im Kapitel 1.4.1 wurde erläutert, dass das Prinzip des Edge Computings, die Daten direkt am Edge-Gerät selbst oder am Rand der IT-Infrastruktur zu verarbeiten ist und gefilterte Daten an die Cloud weiterzuleiten. Dieses Unterkapitel 4.6 knüpft an die vorherigen Unterkapitel an und legt ein Beispiel zur Verarbeitung und Filterung von simulierten Luftfeuchtigkeits- und Temperatur-Daten dar. Die folgende Anleitung lehnt sich an die Anleitung der offiziellen Webseite [58] an.

Zusätzlich benötigte Programme:

- Python-Erweiterung für Visual Studio Code
- Docker CE
- Pip

4.5.1 Eine Container-Repository / Containerregistry erstellen

In einer Container-Repository werden die Containerimages gespeichert und verwaltet. Die Images können dann von dem Repository auf einem oder mehreren IoT Edge-Geräten bereitgestellt werden. Um eine Container-Repository zu erstellen, muss im Azure-Portal auf *Ressource erstellen -> Container -> Container Registry* geklickt werden.

Hinweis: Für die Containerregistry entstehen, je nach Abonnement, Kosten. Um entstehende Kosten möglichst gering zu halten, sollten Ressourcen, die anschließend nicht mehr benötigt werden, entfernt werden.

Folgende Werte werden in den Feldern eingegeben:

Registry name: Hier kann ein beliebiger, einzigartiger Name gewählt werden. Zum Beispiel: jtreg

Subscription: Das Abonnement, welches man benutzt. Zum Beispiel: Free Trial.

Resource group: Eine Ressourcengruppe wählen. In dem Beispiel: jt-ressource

Location: Die gewünschte Region auswählen. Beispielsweise: Westeuropa.

Admin user: Bei Admin user muss *Enable* ausgewählt werden.

SKU: Für diesen Test wird Basic ausgewählt. Standard und Premium bieten höhere Leistung und Skalierbarkeit, steigen jedoch preislich je nach Angebot an.

4.5.2 Eine neue Lösung erstellen

Im VSC ist auf View -> Terminal zu gehen und im Terminal folgender Befehl einzugeben:

```
pip install --upgrade --user cookiecutter
```

Es muss sichergestellt werden, dass Cookiecutter in der PATH-Umgebung installiert ist. In Windows mit der Python Version 3.6 ist dies unter C:\Users\<BENUTZERNAME>\AppData\Roaming\Python\Python36\Scripts. Ist dies sichergestellt, kann in VSC auf *View -> Command Palette* navigiert und der Befehl *Azure IoT Edge: New IoT Edge Solution* eingegeben werden. Die nachfolgenden Prompts müssen entsprechend ausgefüllt werden:

Select folder: Einen Ordner auswählen, in dem der Code und andere Dateien für die Lösung gespeichert werden soll.

Provide a solution name: Einen beschreibenden Namen für die Lösung eintragen. Für diesen Test wurde der die Lösung *tempTestSolution* benannt.

Select module template: In dieser Arbeit wird die Sprache Python ausgewählt. Also ist hier *Python Module* zu selektieren.

Provide a module name: Das Modul benennen. Für diesen Test *tempTestModule*.

Provide Docker image repository for the module: Hier ist der Pfad zur Containerregistry / Container-Repository anzugeben. Der Anmeldeserver ist im Azure Portal unter **Alle Ressourcen** -> *jtreg (Containerregistry Name)* -> **Zugriffsschlüssel** -> **Anmeldeserver** zu finden. Abbildung 14 stellt die Form des Pfades dar.

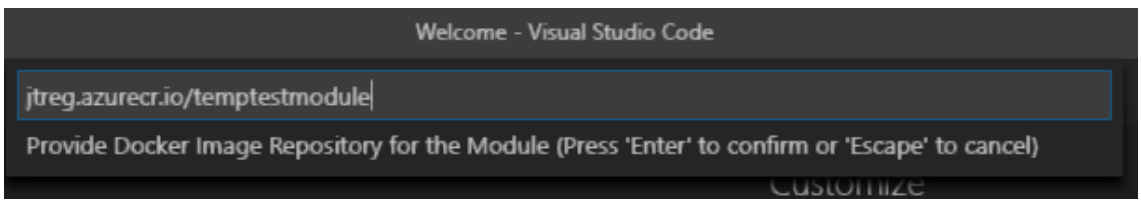


Abbildung 14: Bildschirmaufnahme VSC: Pfad zur Containerregistry / Container-Repository

Die Position der Daten, die im weiteren Verlauf noch benötigt werden, sind in Abbildung 15 zu sehen.

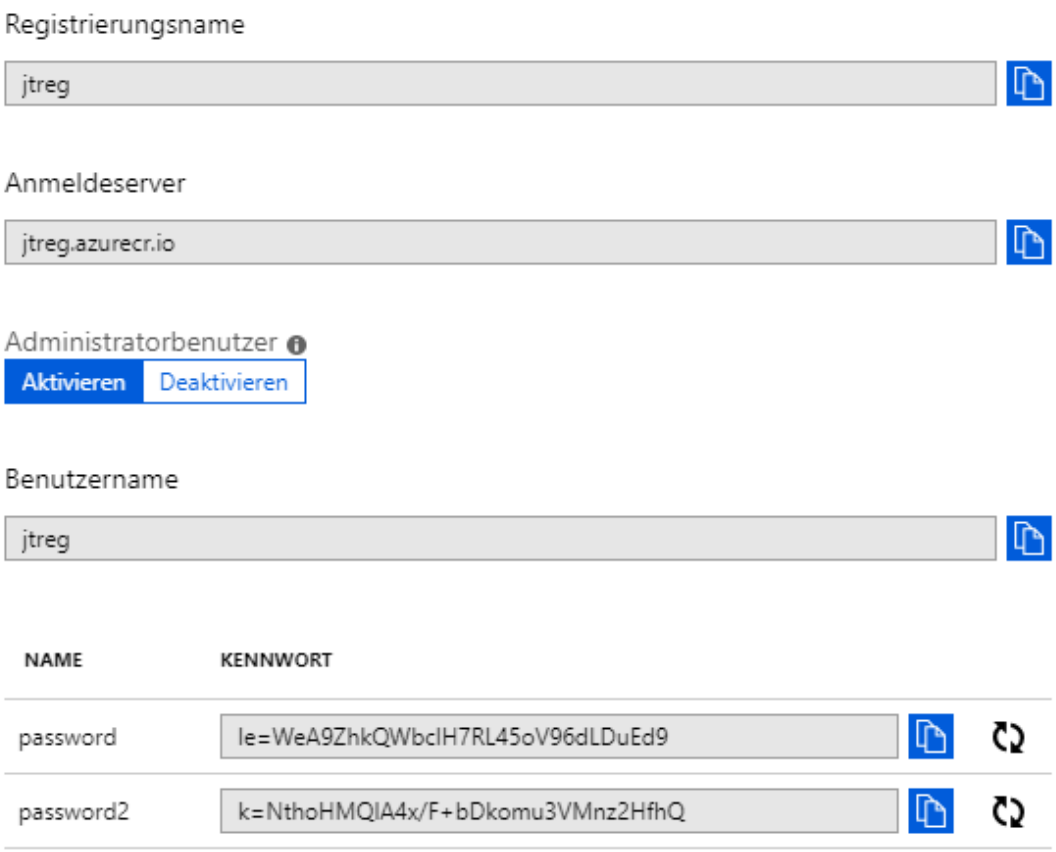


Abbildung 15: Bildschirmaufnahme Azure Portal: Zugriffsschlüssel der Containerregistry / Container-Repository

Im nächsten Schritt ist im VSC in der Explorer-Ansicht die .env-Datei zu öffnen und, wenn nicht bereits vorhanden, die folgenden Daten einzutragen:

```
CONTAINER_REGISTRY_USERNAME_jtreg=<REGISTRIERUNGSNAME>
CONTAINER_REGISTRY_PASSWORD_jtreg=<KENNWORT von "password">
```

Als nächstes ist **modules** -> *tempTestModule (MODULNAME)* -> **main.py** zu öffnen. Dieser Muster-Code muss modifiziert werden, damit nur bestimmte Daten zum IoT Hub gesendet werden.

Im Code muss die JSON-Bibliothek importiert werden.

```
import json
```

Im Anschluss muss der Grenzwert, also wann eine Warnung gesendet wird, bestimmt werden.

```
TEMPERATURE_THRESHOLD = 25
TWIN_CALLBACKS = 0
```

Daraufhin muss die gesamte Funktion **receive_message_callback(message, hubManager)** durch den folgenden Code ersetzt werden:

```
def receive_message_callback(message, hubManager):
    global RECEIVE_CALLBACKS
    global TEMPERATURE_THRESHOLD
    message_buffer = message.get_bytearray()
    size = len(message_buffer)
    message_text = message_buffer[:size].decode('utf-8')
    print ( "    Data: <<<%s>>> & Size=%d" % (message_text, size) )
    map_properties = message.properties()
    key_value_pair = map_properties.get_internals()
    print ( "    Properties: %s" % key_value_pair )
    RECEIVE_CALLBACKS += 1
    print ( "    Total calls received: %d" % RECEIVE_CALLBACKS )
    data = json.loads(message_text)
    if "machine" in data and "temperature" in data["machine"] and
data["machine"]["temperature"] > TEMPERATURE_THRESHOLD:
        map_properties.add("MessageType", "Alert")
        print("Machine temperature %s exceeds threshold %s" %
(data["machine"]["temperature"], TEMPERATURE_THRESHOLD))
    hubManager.forward_event_to_output("output1", message, 0)
    return IoTHubMessageDispositionResult.ACCEPTED
```

Anschließend ist eine neue Funktion **module_twin_callback** anzulegen:

```
def module_twin_callback(update_state, payload, user_context):
    global TWIN_CALLBACKS
    global TEMPERATURE_THRESHOLD
    print ( "\nTwin callback called with:\nupdateStatus = %s\npayload
= %s\ncontext = %s" % (update_state, payload, user_context) )
    data = json.loads(payload)
    if "desired" in data and "TemperatureThreshold" in
data["desired"]:
        TEMPERATURE_THRESHOLD =
data["desired"]["TemperatureThreshold"]
    if "TemperatureThreshold" in data:
        TEMPERATURE_THRESHOLD = data["TemperatureThreshold"]
    TWIN_CALLBACKS += 1
    print ( "Total calls confirmed: %d\n" % TWIN_CALLBACKS )
```

In die HubManager-Klasse muss folgende Methode eingefügt werden:

```
self.client.set_module_twin_callback(module_twin_callback, self)
```

Anschließend muss main.py gespeichert werden.

Der ganze Code ist im Anhang S.68-71 in der Tabelle 10: main.py-Code zu finden.

Als nächstes ist die **deployment.template.json**-Datei zu öffnen. Die Architektur des RPis in dieser Arbeit lautet **ARM32v7**. Dementsprechend muss in der Statusleiste „**arm32v7**“ ausgewählt werden. Standardmäßig ist dies auf **amd64** eingestellt.

In der **deployment.template.json**-Datei ist in den unteren Zeilen, nach **$edgeHub**, folgender Code einzutragen und die Datei daraufhin abzuspeichern:

```
"PythonModule": {
        "properties.desired":{
            "TemperatureThreshold":25
        }
    }
```

Für eine bessere Übersicht, ist auf Abbildung 16 die Positionierung des Codes zu sehen.

Der ganze Code ist im Anhang S.72-74 in der Tabelle 11: deployment.template.json zu finden.

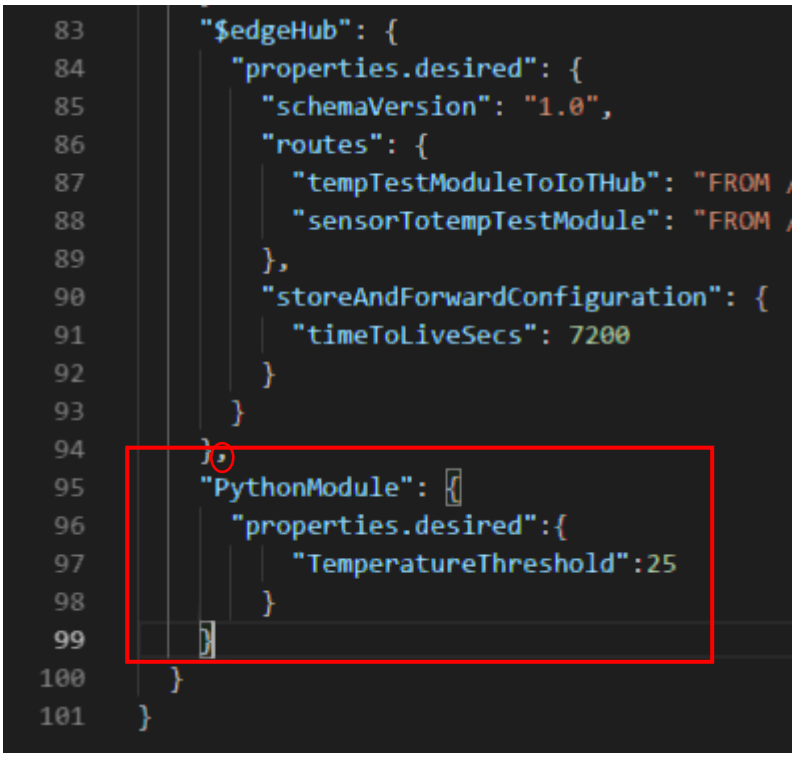


Abbildung 16: Bildschirmaufnahme VSC: deployment.template.json-Datei

Als nächstes ist das Terminal im VSC unter **Terminal** -> **New Terminal** zu öffnen. Dort ist folgender Befehl einzugeben, um sich anzumelden:

```
docker login -u <ACR Benutzername> -p <ACR Kennwort> <ACR
Anmeldeserver>
```

Der ACR Benutzername, das ACR Kennwort sowie der Anmeldeservername ist auf der Seite der Zugriffsschlüssel für die Containerregistrierung, wie in Abbildung 15 zu sehen, auszulesen.

In diesem Beispiel sieht der Befehl dann folgendermaßen aus:

```
docker login -u jtreg -p [KENNWORT von password] jtreg.azurecr.io
```

Mit dem Befehl meldet man sich bei Docker an, um nachher eine Modul-Image-Datei zu erstellen und anschließend in die Container-Registry / Repository im IoT Hub zu pushen.

Um die Lösung zu erstellen und in das Repository zu pushen, muss der Benutzer ein Rechtsklick auf das **deployment.template.json** tätigen und daraufhin auf **Build and Push IoT Edge Solution** klicken.

4.5.3 Bereitstellen und Ausführen der Lösung

Nun ist das Modul bereit, auf dem IoT-Edge-Gerät von der Cloud aus bereitgestellt zu werden. Im VSC *Azure IoT Hub Devices* aufklappen und mit der rechten Maustaste auf das IoT Edge-Gerät gehen. Zum Bereitstellen auf das Gerät auf *Create Deployment for Single Device* klicken. Im Anschluss ist im **config**-Ordner die **deployment.<ARCHITEKTUR>.json**-Datei zu wählen und auf **Select Edge Deployment Manifest** zu klicken. In diesem Beispiel heißt die Datei **deployment.arm32v7.json**.

Mit dem folgenden Befehl kann die Log-Datei eingesehen werden:

```
sudo iotedge logs tempTestModule -f
```

Anmerkung: Es kann passieren, dass der Raspberry Pi nicht mit dem Hub kommunizieren kann. Die Behebung dieses Fehlers wird im folgenden Abschnitt erläutert. Wenn der Fehler nicht auftritt, ist die Installationsanleitung hier fertig.

Fehlerbehebung [58]:

Im Azure Portal auf *Alle Ressourcen -> jt-hub -> IoT Edge -> jt-raspi -> set modules -> Configure advanced Edge Runtime settings* gehen. Bei *Edge Hub* folgendes einzutragen:

Environment Variables

Name: `OptimizeForPerformance`

Values: `false` (Abbildung 17)

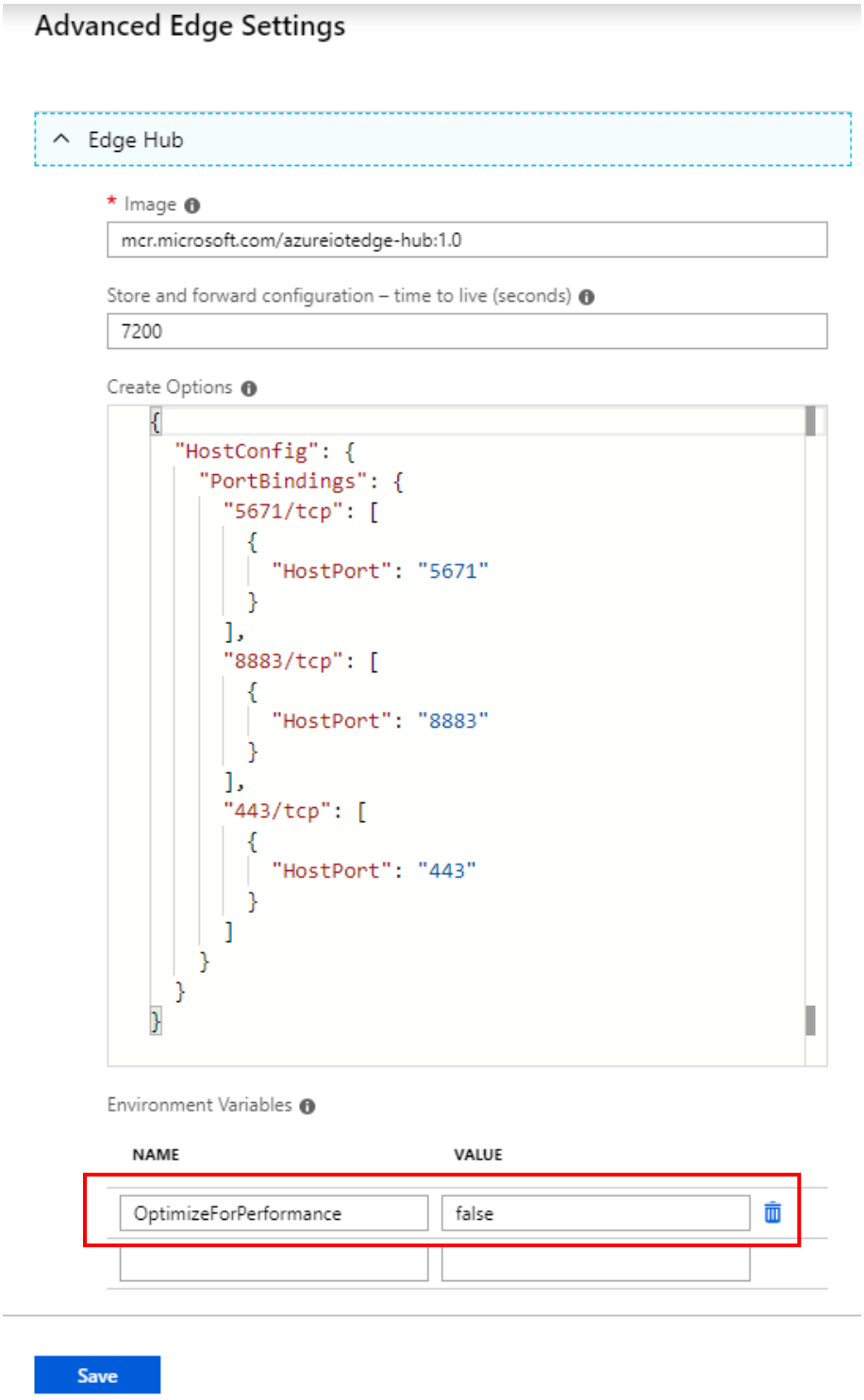


Abbildung 17: Bildschirmaufnahme Azure Portal: Umgebungsvariablen für Edge Hub

4.6 Kostenüberwachung / Kostenanalyse

Im Azure Portal unter *Kostenverwaltung –> Meine Abonnements: Free Trial* können entstandene Kosten analysiert werden.

In dem Tortendiagramm auf Abbildung 18 sind die einzelnen Kosten während der Tests dieser Arbeit zu sehen. Relevant ist hier **jtreg** mit **1,76 EUR**. Wird auf das Feld des Tortendiagramms geklickt, öffnet sich eine Liste mit den einzelnen Ressourcen (Abbildung 19). Wird hier auf **jtreg** geklickt, öffnet sich eine Grafik mit dem Kostenverlauf seit Erstellung (Abbildung 20). Hier ist zu erkennen, dass die Containerregistry **jtreg** pro Tag 0,14 € kostet.

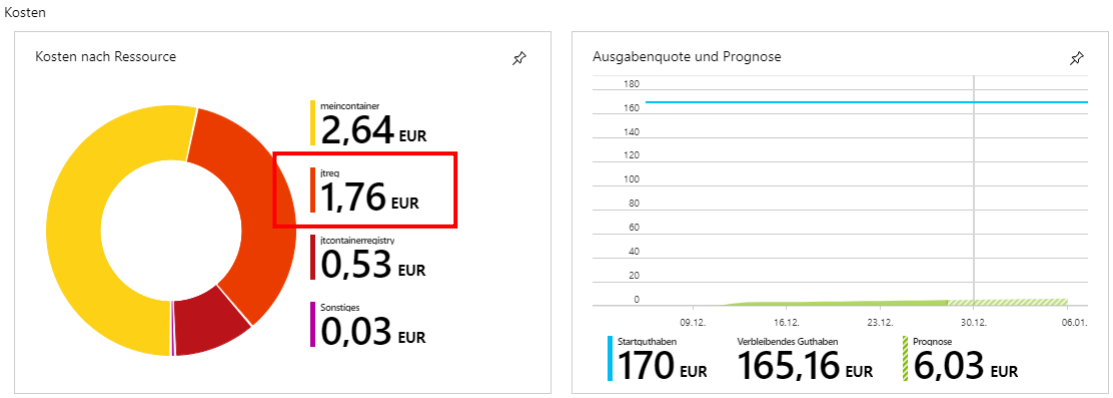


Abbildung 18: Bildschirmaufnahme Azure Portal: Kostenübersicht

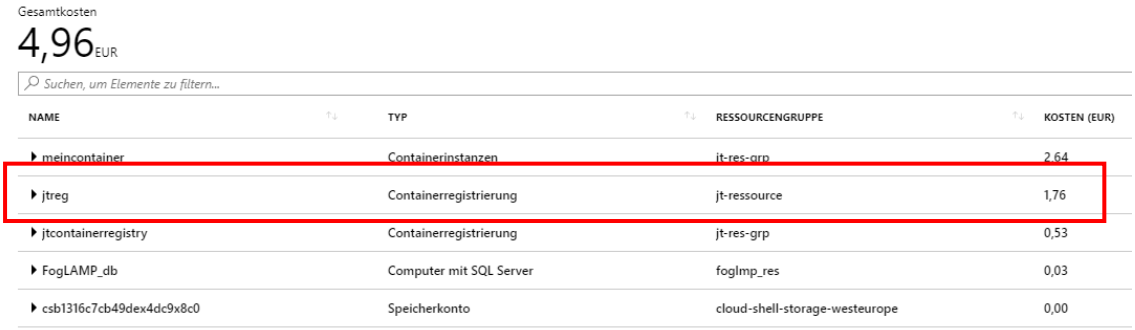
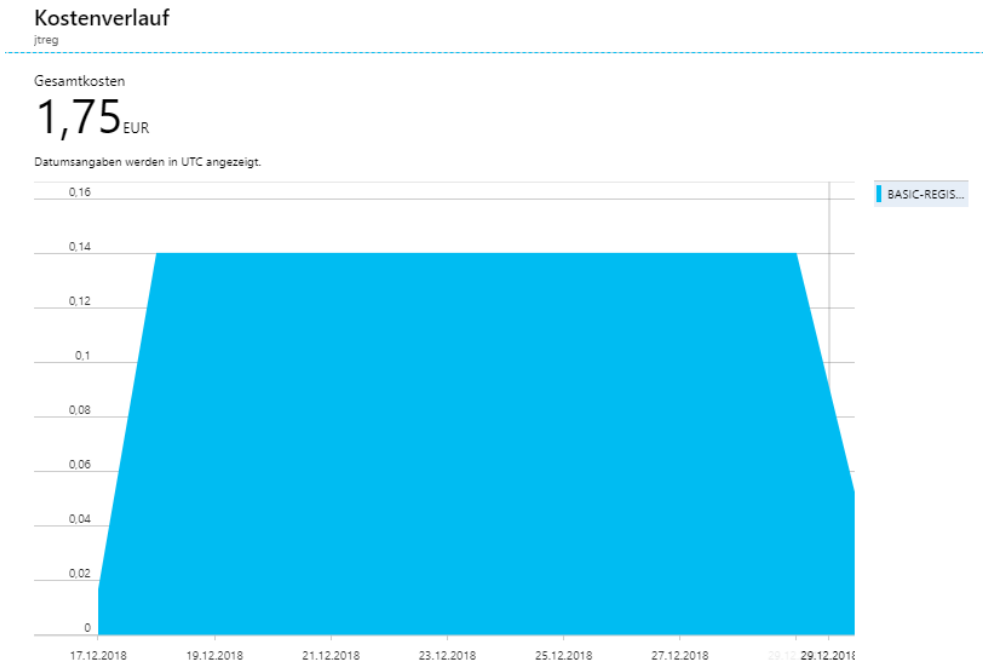


Abbildung 19: Bildschirmaufnahme Azure Portal: Kostenübersicht - Liste



DATUM	MENGE (EUR)
17.12.2018	0,02
18.12.2018	0,14
19.12.2018	0,14
20.12.2018	0,14

Abbildung 20: Bildschirmaufnahme Azure Portal: Kosten von jtreg pro Tag

4.7 Auswertung

Im Unterkapitel Auswertung wird das Verhältnis der generierten Daten und der vom IoT Hub empfangenen und verwendeten Daten ausgerechnet.

Ermitteln der erzeugten Nachrichten während einer Session vom 12. Dezember 2018.

Auf dem RPi:

```
iotedge logs tempSensor | grep -c 12/22/2018
```

862

Ermitteln der Anzahl von erhaltenen Nachrichten im IoT Hub, gefiltert durch das tempTestModul am 22. Dezember 2018.

Im Azure Portal:

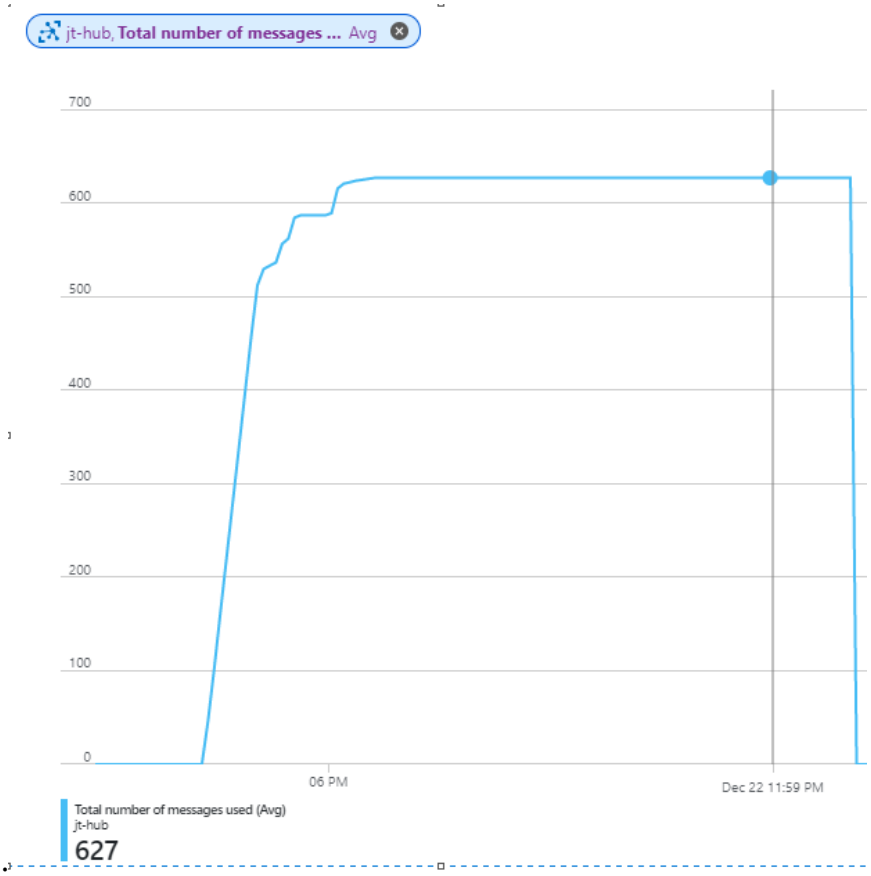


Abbildung 21: Bildschirmaufnahme Azure Portal: Gesamtanzahl der verwendeten (empfangenen) Nachrichten im jt-hub im Zeitraum von einem Tag (22. Dezember 2018).

Von 862 generierten Nachrichten wurden 627 Nachrichten an den IoT Hub gesendet. Somit wurden in diesem Beispiel durch Edge Computing 235 Nachrichten (27,26 % der Nachrichten), nicht unnötig in den Hub / die Cloud gesendet.

Kosten für die Breitbandverbindung

Die Benutzung der Breitbandverbindung für eingehende Daten in Azure-Datencentern sind kostenlos. Bei ausgehenden Daten aus den Azure-Datencentern entstehen Kosten ab 5 GB. Die Kostenersparnis konnten für diesen Test nicht errechnet werden, da die ersten 5 GB/Monat kostenfrei sind und Kosten von 0,074€ erst ab 5 GB – 10 GB/Monat [37] entstehen.

5 Installationsanleitung und Bedienung von FogLAMP

Im Kapitel Installation von FogLAMP wird das Sense HAT auf dem Raspberry Pi 3 Modell B installiert und die Fog-Computing-Lösung FogLAMP aufgespielt. Diese Installationsanleitung lehnt sich an die Anleitung in der offiziellen Webseite an [60].

Für die Installation werden folgende Pakete benötigt.

- FogLAMP Core
- FogLAMP User Interface
- Eine oder mehrere FogLAMP South Services (z. B. das RPi Sense HAT)
- Eine oder mehrere FogLAMP North Services (z. B. OSI PI oder OCS)

Anmerkung: Für den North Service OSI PI wird ein OSIsoft Konto mit bestimmten Berechtigungen benötigt, um die nötigen Softwares für ein North Service herunterzuladen. Sind diese Berechtigungen / Lizenzen nicht vorhanden, kommt es zu einer Fehlermeldung beim Versuch die benötigten Softwares herunterzuladen [61]. In dieser Anleitung wird der North Service nicht mit installiert.

5.1 Vorbereitungen

5.1.1 Installation von Raspbian

Analog zu Unterkapitel 4.1

5.1.2 Konfiguration des Raspberry Pi 3

Analog zum Unterkapitel 4.1.2

5.1.3 Sense-HAT-Installation

Schritt 1: Update der APT-Paketlisten.

Auf dem RPi (Terminal) oder über PuTTY:

```
sudo apt-get update
```

Schritt 2: Aktivierung von I2C.

Auf dem RPi ist die I2C-Einstellung zu aktivieren. Dazu auf dem RPi auf das RPi-Symbol klicken und dann auf Einstellungen -> Raspberry-Pi-Konfigurationen -> Schnittstellen -> gehen und I2C aktivieren. (Abbildung 22)

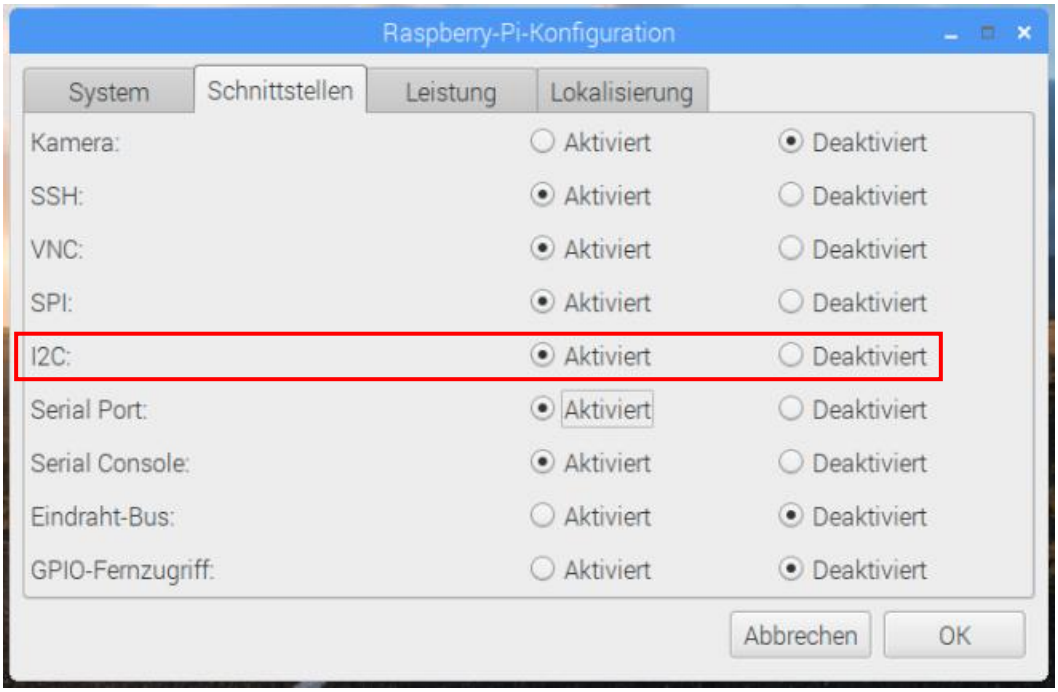


Abbildung 22: I2C-Einstellung

Schritt 3: Installation der Sense-HAT-Pakete.

```
sudo apt-get install sense-hat
```

Schritt 4: Neustart des Raspberry Pis, um Änderungen zu übernehmen.

```
sudo reboot
```

Schritt 5: Herunterladen der Debian-Pakete.

Die FogLAMP-Core-Software für die Prozessor-Architektur ARM, die FogLAMP GUI und die FogLAMP-Sense-HAT-Software werden von der offiziellen Webseite [62] heruntergeladen:

- `foglamp-1.4.0-armhf.deb (Core)`
- `foglamp-gui-1.4.0.deb (GUI)`
- `foglamp-south-sensehat-1.1.0-armhf.deb (Sense-HAT-Software)`

Schritt 6 (Optional, falls die Pakete auf einem Windows-Computer heruntergeladen wurden): Übertragen der Dateien auf das RPi.

Die IP-Adresse vom RPi einholen:

Auf dem RPi (Terminal):

```
hostname -I
```

Auf dem Windows-PC:

Mit der Windows-Taste + R wird das Ausführfenster geöffnet und cmd eingegeben. Darauf öffnet sich ein Windows-Terminal. In der Kommandozeile wird in den Ordner der heruntergeladenen Pakete navigiert

```
cd Downloads
```

und anschließend die Datei `foglamp-1.4.0-armhf.deb` auf den RPi übertragen:

```
scp foglamp-1.4.0-armhf.deb <RPi-Benutzername>@<RPi-IP-Adresse>:
```

Zum Beispiel:

```
scp foglamp-1.4.0-armhf.deb pi@192.168.1.10:
```

Nachdem dies mit der Eingabetaste bestätigt wurde, erscheint eine Meldung zur Authentizität der angegebenen IP-Adresse. Diese Meldung ist mit „yes“ zu bestätigen und anschließend ist das Passwort vom Benutzer „pi“ einzugeben. Standardmäßig ist das Passwort `raspberry`.

Das gleiche auch mit der `foglamp-gui-1.4.0.deb`-Datei und der `foglamp-south-sensehat-1.1.0-armhf.deb`-Datei durchführen:

```
scp foglamp-gui-1.4.0.deb pi@192.168.1.10:
scp foglamp-south-sensehat-1.1.0-armhf.deb pi@192.168.1.10:
```

5.2 Installation von FogLAMP

Im Unterkapitel FogLAMP Installation wird die Installation von FogLAMP auf das Raspberry Pi 3 Modell B durchgeführt.

Schritt 1: Installation der Pakete

Im Raspberry Pi Terminal:

Nach dem Übertragen der Daten vom Windows-PC zum RPi, liegen die Dateien im Verzeichnis */home/pi*. Es wird zum Verzeichnis navigiert und anschließend die Pakete installiert.

Installation des FogLAMP-Cores:

```
sudo apt -y install ./foglamp-1.4.0-armhf.deb
```

Installation der FogLAMP GUI:

```
sudo apt -y install ./foglamp-gui-1.4.0-dev.deb
```

Installation des South-Plugins Sense HAT:

```
sudo apt -y install ./foglamp-south-sensehat-1.1.0-armhf.deb
```

5.3 Bedienung von FogLAMP

Das Dienstprogramm für FogLAMP ist standardmäßig im Verzeichnis */usr/local/foglamp/bin* installiert. Tabelle 4 zeigt die verfügbaren Befehle an.

Befehl	Ausführung
`/usr/local/foglamp/bin/foglamp start`	Startet das FogLAMP-System
`/usr/local/foglamp/bin/foglamp stop`	Stoppt das FogLAMP-System
`/usr/local/foglamp/bin/foglamp status`	Zeigt die laufenden FogLAMP-Prozesse an
`/usr/local/foglamp/bin/foglamp reset`	Löscht alle Daten und Konfigurationseinstellungen und setzt FogLAMP zurück in die Werkeinstellung
`/usr/local/foglamp/bin/foglamp kill`	Terminiert FogLAMP-Prozesse, die auf das Stopp-Befehl nicht reagiert haben
`/usr/local/foglamp/bin/foglamp help`	Beschreibt FogLAMP-Optionen
`grep -a 'foglamp' /var/log/syslog | tail -n 20`	Gibt aus dem syslog die letzten 20 Zeilen aus

Tabelle 4: FogLAMP-Befehle

FogLAMP GUI

Die FogLAMP GUI ist über den FogLAMP-Server erreichbar. Hierzu ist die IP-Adresse, in diesem Beispiel vom Raspberry Pi (*192.168.1.10*), in einen Webbrowser einzugeben. Damit öffnet sich die FogLAMP GUI (Abbildung 23).

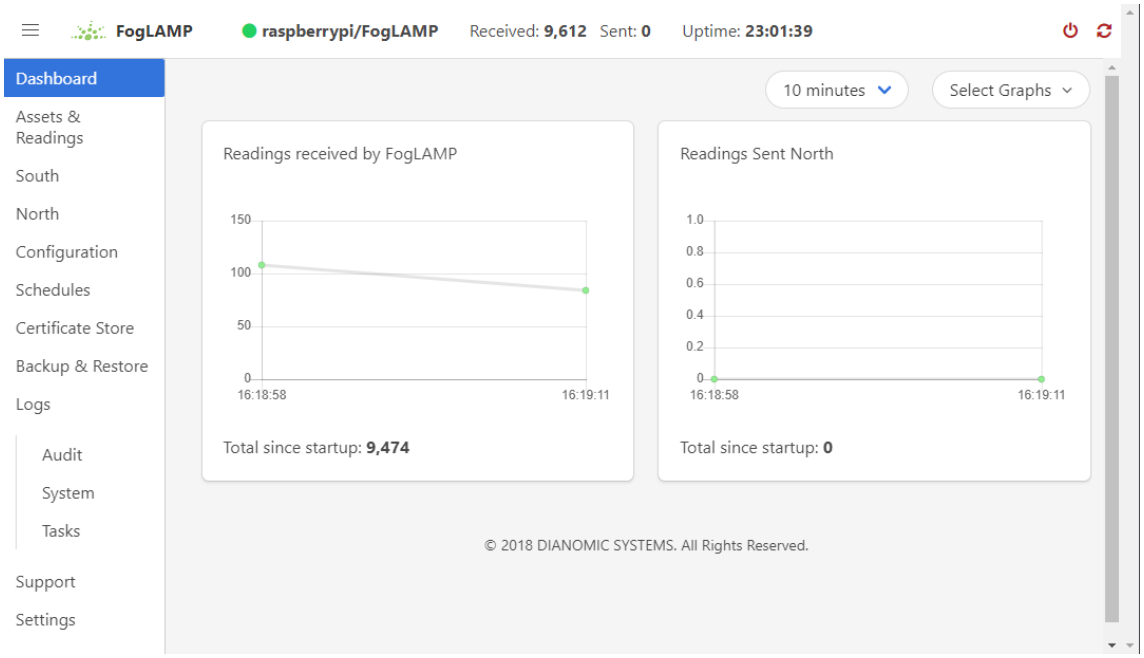


Abbildung 23: Bildschirmaufnahme: FogLAMP GUI

Hinzufügen vom South Service Sense HAT

In diesem Abschnitt wird das eingebaute Sense HAT aus Kapitel 1.1.4 als South Plugin verbunden. Dazu muss in der GUI *South* und anschließend *Add* + geklickt werden.

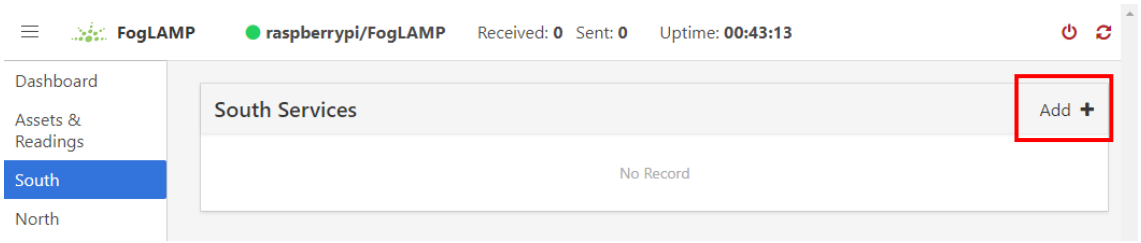


Abbildung 24: Bildschirmaufnahme FogLAMP GUI: Hinzufügen von South Services

FogLAMP erkennt automatisch die angeschlossenen Geräte / Erweiterungen. Um das Sense HAT hinzuzufügen, wird auf das South Plugin *sensehat* geklickt und ein Name für das Plugin ausgewählt. In diesem Beispiel wurde *sensehatSouth* ausgewählt. (Abbildung 25)

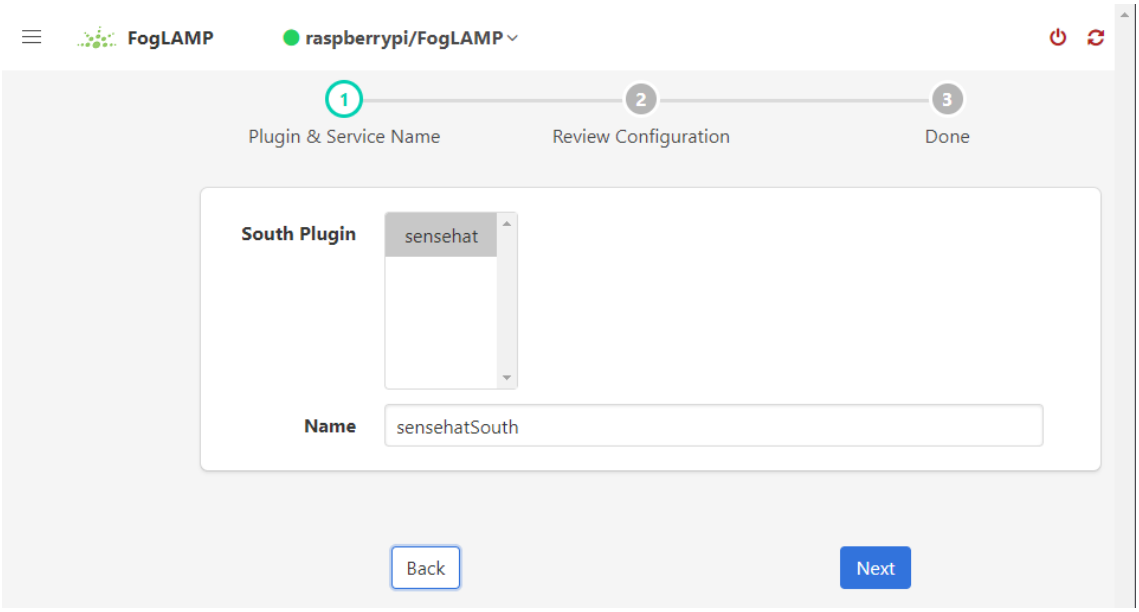


Abbildung 25: Bildschirmaufnahme FogLAMP GUI: Hinzufügen vom Sense Hat

Als Nächstes wird *Next* ausgewählt (Abbildung 25) und anschließend die Konfigurationseinstellungen überprüft (Abbildung 26).

Abbildung 26: Bildschirmaufnahme FogLAMP GUI: Überprüfung der Konfiguration

Zum Schluss auf *Next* klicken und auf der nächsten Seite ein Häckchen bei „*Enabled*“ setzen. Mit der Schaltfläche *Done* wird das South Plugin erstellt.
Klickt man nun auf South, ist der sensehatSouth-Service mit den aktuellen Messdaten zu sehen. (Abbildung 27)

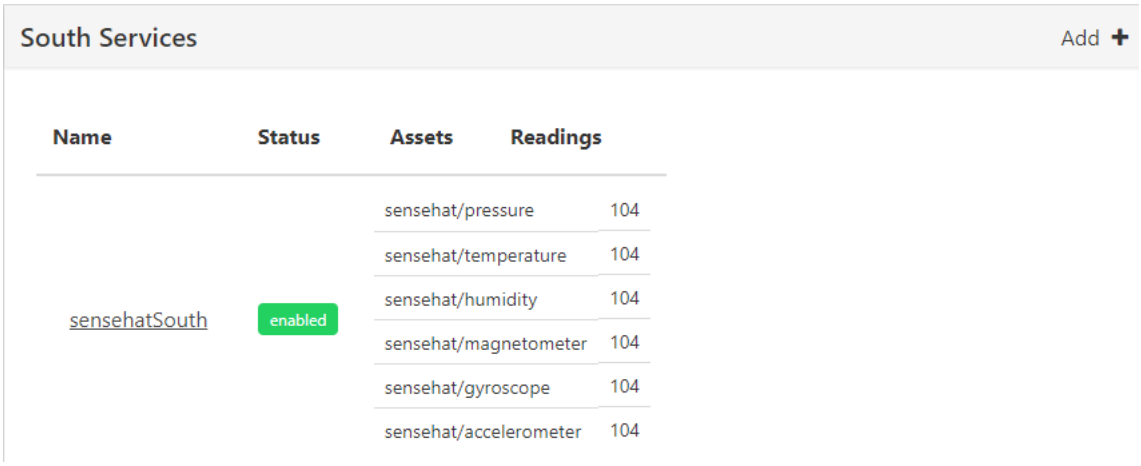


Abbildung 27: Bildschirmaufnahme FogLAMP GUI: sensehatSouth Sensordaten

Unter Assets & Readings sind die einzelnen Sensoren mit der Anzahl der Messungen gelistet (Abbildung 28). Wird auf den blauen Graphen geklickt, öffnet sich für den jeweiligen Sensor ein detaillierter Graph. (Abbildung 29)

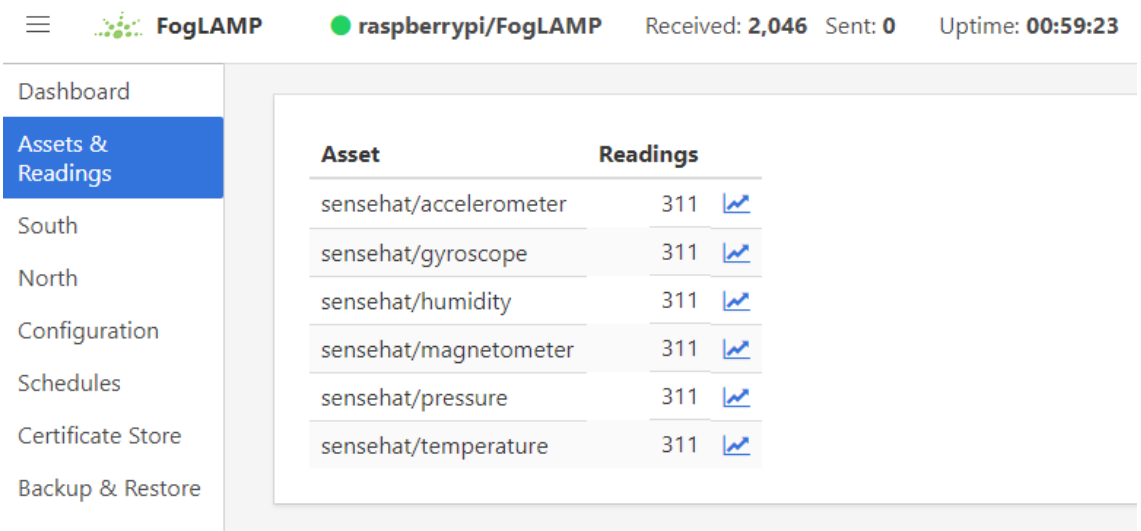


Abbildung 28: Bildschirmaufnahme FogLAMP GUI: Assets & Readings

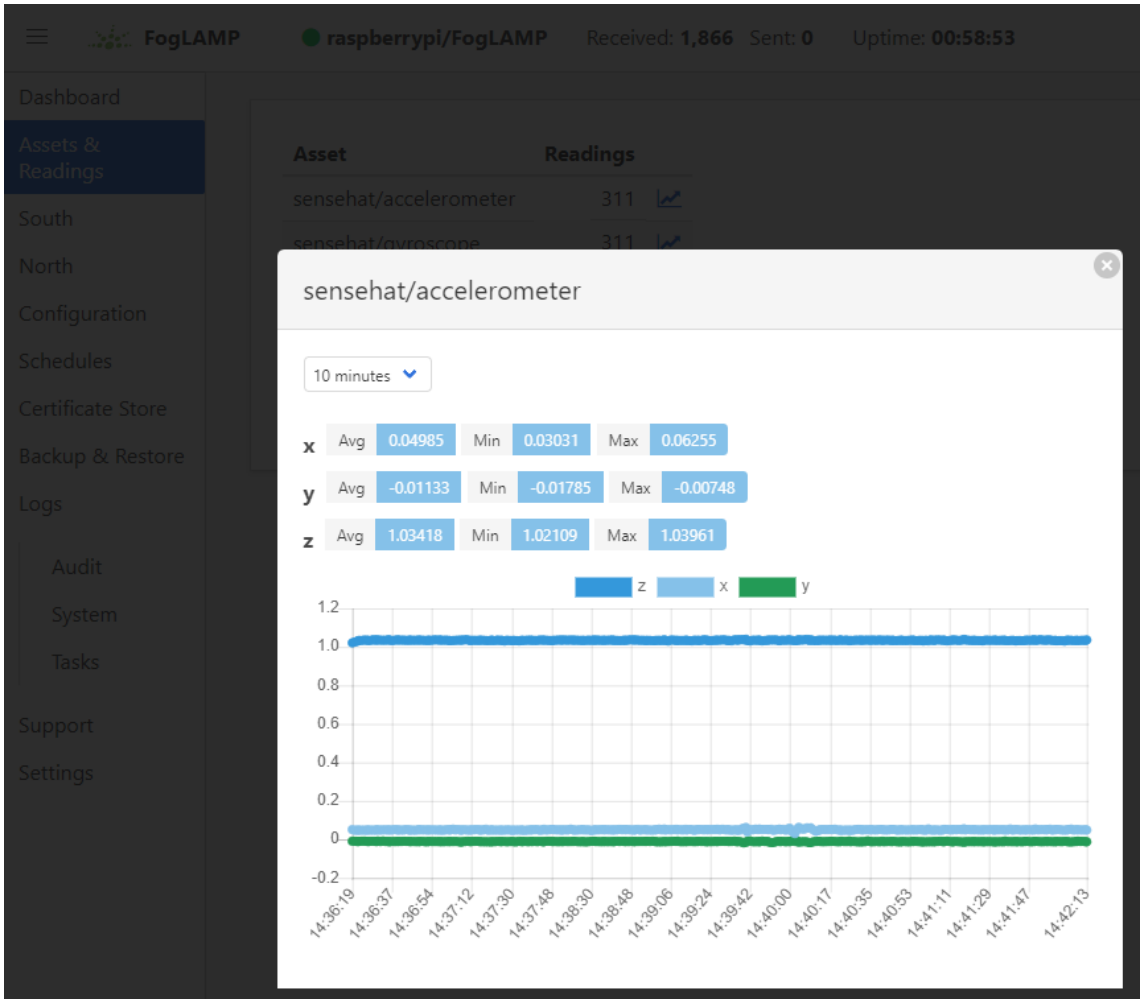


Abbildung 29: Bildschirmaufnahme FogLAMP GUI: Assets & Readings - sensehat/accelerometer

6 Fazit

Das Ziel dieser Bachelorarbeit war es, Edge- und Fog-Computing-Lösungen auf dem Einplatinencomputer Raspberry Pi zu realisieren. In der Anfangsphase dieser Thesis (11/2018) war AWS Greengrass auf dem Raspberry Pi nur mit dem Betriebssystem Raspbian Jessie kompatibel. Weil Azure IoT Edge jedoch das aktuellste Betriebssystem Raspbian Stretch unterstützt, wurde sie in dieser Arbeit als Lösung bevorzugt. Im Laufe der Thesis (12/2018) wurde für AWS Greengrass die Unterstützung einer bestimmten Version von Raspbian Stretch hinzugefügt. Dies zeigt, dass an den Lösungen ständig weiterentwickelt wird.

Die Installation von Microsoft Azure IoT Edge gestaltete sich einfach, da es auf der offiziellen Webseite viele Anleitungen, Tutorials und Hilfestellungen gibt. Es gibt viele Beispielprogramme, die man zum Testen verwenden kann. Zuvorkommend ist auch, dass Azure ein Guthaben von 170 EUR für 30 Tage zum Testen zur Verfügung stellt. Das Guthaben wird u. a. für eine Containerregistry genutzt. Es ist zu beachten, unnötige Ressourcen nach den Tests zu löschen, da nach den 30 Tagen Kosten entstehen können.
Das Azure Portal ist übersichtlich und leicht zu bedienen. Über die Suchleiste kann der Benutzer im Portal zügig navigieren. Die Tests mit dem Modul brachten die erhofften Ergebnisse. Dadurch, dass nicht alle Nachrichten in das IoT-Hub gesendet wurden, sondern die Verarbeitung bereits am Gerät stattfand, konnten irrelevante Informationen in der Cloud eingespart werden. Dies setzt an das Problem mit der Bandbreitenauslastung und den Datenmengen in der Cloud an.

Die Installation von FogLAMP erwies sich ebenfalls als weitgehend unkompliziert. Auf der offiziellen Webseite sind viele Erklärungen und Anleitungen verfügbar. Der Server war leicht zu installieren und das South Plugin war ohne Schwierigkeiten einzurichten. Die GUI ist übersichtlich und intuitiv. Die Einbindung des North Plugins erwies sich jedoch als schwierig, da man dazu ein OSIsoft-Konto mit bestimmten Lizenzen benötigt. Außerdem muss der Benutzer Kenntnisse mit den OSIsoft-Programmen besitzen.

Der Test mit den Sense HAT Sensordaten zeigte das Prinzip, dass Daten in den Server gelangen, um dort überwacht oder verarbeitet zu werden. Über den North Plugin sollen diese Daten dann analysiert bzw. persistent gespeichert werden.

7 Ausblick

Bei Azure IoT Edge könnte man testen, Funktionen als Module einzusetzen. Es wäre auch interessant, mehrere Module auf einem Gerät zu installieren und die Kommunikation zwischen den Modulen durch Routing (z. Dt. Routen) zu konfigurieren und zu überwachen.

Für FogLAMP könnte man die Lizenz für die Software OSI PI beim OSIsoft-Support beantragen, um den North Plugin erfolgreich installieren zu können und zu testen. Zusätzlich könnte das Storage Plugin installiert werden.

Literaturverzeichnis

[1] Bedner, Mark: „*Cloud Computing: Technik, Sicherheit und rechtliche Gestaltung*“, Institut für Wirtschaftsrecht an der Universität Kassel, Band 14, Kassel: kassel university press GmbH 2013

[2] Manuel, „*Raspberry Pi 3B+ und 3B im Vergleich*“, URL: https://www.datenreise.de/raspberry-pi-3b-vs-3b-unterschied/ [Aufgerufen am 29.12.2018]

[3] itwissen.info (19.11.2017): „*IoT-Gateway*“. URL: https://www.itwissen.info/IoT-Gateway-IoT-gateway.html [Aufgerufen am 26. November 2018]

[4] Bildquelle: „*IDC´s Data Age 2025 study, sponsored by Seagate*“, [Online]. Available: https://www.industry-of-things.de/iot-basics-so-funktioniert-edge-computing-a-678225/ [Aufgerufen am 24. Dezember 2018]

[7] Bildquelle: Dianomic, „*FogLAMP – Dianomic*“, [Online]. Available: http://dianomic.com/platform/foglamp/ [Aufgerufen am 20. Dezember 2018]

[8] Microsoft Azure, *Azure Portal*, [Online]. Available: https://portal.azure.com/ [Aufgerufen am 20. November 2018]

[9] AWS Greengrass, „Unterstützte Plattformen und Anforderungen“, URL: https://docs.aws.amazon.com/en_us/greengrass/latest/developerguide/what-is-gg.html [Aufgerufen am 15. November 2018]

[1] www.cloudflare.com, „*What Is Edge Computing? | Cloudflare*“, [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/ [Aufgerufen am 24 Dezember 2018]

[2] Filipe Martins und Anna Kobylinska, „*IoT-Basics: So funktioniert Edge Computing*“, [Online]. Available: https://www.industry-of-things.de/iot-basics-so-funktioniert-edge-computing-a-678225/ [Aufgerufen am 24 Dezember 2018]

[3] www.raspberrypi.org, „*Raspberry Pi Foundation - About us*“, [Online]. Available: https://www.raspberrypi.org/about/ [Aufgerufen am 18 November 2018]

[4] www.raspberrypi.org, „*Raspberry Pi Downloads – Software for the Raspberry Pi*“, [Online]. Available: https://www.raspberrypi.org/downloads/ [Aufgerufen am 18 November 2018]

[5] ELEKTRONIKPRAXIS - Wissen. Impulse. Kontakte, „*Alle Raspberry-Pi-Modelle auf einen Blick*“, [Online]. Available: https://www.ip-insider.de/alle-raspberry-pi-modelle-auf-einen-blick-v-39812-13275/ [Aufgerufen am 18. November 2018]

[6] Eben Upton, „*Raspberry Pi 3 onsale now at $35 -Raspberry Pi*“, [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/ [Aufgerufen am 18 November 2018]

[7] www.terraelectronica.ru, „*Raspberry Pi 3 Model B*“, [Online]. Available: https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fds%252Fpdf%252FT%252FTechicRP3.pdf [Aufgerufen am 19 November 2018]

[8] www.reichelt.de, „*Sense HAT: Erweiterungsplatine macht den Raspberry zur Raumklima-Messstation*“, [Online]. Available: https://www.reichelt.de/magazin/how-to/sense-hat-erweiterungsplatine-macht-den-raspberry-zur-raumklima-messstation/ [Aufgerufen am 29. Dezember 2018]

[9] Michael Radtke, „*Was ist Cloud Computing?*“, [Online]. Available: https://www.cloudcomputing-insider.de/was-ist-cloud-computing-a-563624/ [Aufgerufen am 23 November 2018]

[10] Microsoft Azure, „*Was ist Cloud Computing? Leitfaden für Einsteiger*“, [Online]. Available: https://azure.microsoft.com/de-de/overview/what-is-cloud-computing/ [Aufgerufen am 23 November 2018]

[11] tutanch, „*Was ist eine Public Cloud?*“, [Online]. Available: https://www.clou dcomputing-insider.de/was-ist-eine-public-cloud-a-633184/ [Aufgerufen am 23 November 2018]

[12] Microsoft Azure, „*Was ist eine private Cloud? - Definition*“, [Online]. Available: https://azure.microsoft.com/de-de/overview/what-is-a-private-cloud/ [Aufgerufen am 24 November 2018]

[13] tutanch, „*Was ist eine Hybrid Cloud?*“, [Online]. Available: https://www.cloudcomputing-insider.de/was-ist-eine-hybrid-cloud-a-633168/ [Aufgerufen am 24 November 2018]

[14] www.itwissen.info, „*Community Cloud*“, [Online]. Available: https://www.itwissen.info/Community-Cloud-community-cloud.html [Aufgerufen am 24 November 2018]

[15] Margaret Rouse, „*Was ist Community-Cloud? - Definition von WhatIs.com*“, [Online]. Available: https://www.searchstorage.de/definition/Community-Cloud [Aufgerufen am 24 November 2018]

[16] www.ibm.com, „*What is cloud computing?*“, [Online]. Available: https://www.ibm.com/cloud/learn/what-is-cloud-computing [Aufgerufen am 25.November 2018]

[17] Heinrich Seeger, „*Markübersicht Infrastructure as a Service: IaaS - vergleichen lohnt sich*“, [Online]. Available: https://www.computerwoche.de/a/iaas-vergleichen-lohnt-sich,3060832,2 [Aufgerufen am 23 November 2018]

[18] Klaus Manhart, „*Cloud-Modell "Platform as a Service": PaaS-Anbieter im Vergleich*“, [Online]. Available: https://www.computerwoche.de/a/paas-anbieter-im-vergleich,3066351 [Aufgerufen am 23 November 2018]

[19] Klaus Manhart, „*Cloud-Modell "Platform as a Service": PaaS-Anbieter im Vergleich*“, [Online]. Available: https://www.computerwoche.de/a/paas-anbieter-im-vergleich,3066351,2 [Aufgerufen am 23 November 2018]

[20] www.softselect.de, „*Definition SaaS (Software as a Service)*“, [Online]. Available: http://www.softselect.de/business-software-glossar/saas [Aufgerufen am 22 November 2018]

[21] www.ionos.de, „*SaaS (Software as a Service im Überblick)*“, [Online]. Available: https://www.ionos.de/digitalguide/server/knowhow/saas-software-as-a-service-im-ueberblick-vor-und-nachteile/ [Aufgerufen am 22 November 2018]

[22] Simon Lohmann, „*Saas: Was ist Software as a Service?*“, [Online]. Available: https://www.computerwoche.de/a/was-ist-software-as-a-service,3332266 [Aufgerufen am 22 November 2018]

[23] Simon Lohmann, „*Saas: Was ist Software as a Service?*“, [Online]. Available: https://www.computerwoche.de/a/was-ist-software-as-a-service,3332266,2 [Aufgerufen am 22 November 2018]

[24] RoSch, „*Was ist Edge Computing?*“, [Online]. Available: https://www.cloudcomputing-insider.de/was-ist-edge-computing-a-742343/ [Aufgerufen am 26 November 2018]

[25] www.hpe.com, „*Was ist Edge Computing?*“, [Online]. Available: https://www.hpe.com/de/de/what-is/edge-computing.html [Aufgerufen am 24 November 2018]

[26] Margaret Rouse, „*What is IoT gateway?*“, [Online]. Available: https://whatis.techtarget.com/definition/IoT-gateway [Aufgerufen am 26 November 2018]

[27] www.itwissen.info, „*IoT-Gateway*“, [Online]. Available: https://www.itwissen.info/IoT-Gateway-IoT-gateway.html [Aufgerufen am 26 November 2018]

[28] Chris Pietschmann, „*Introduction to Building IoT Solutions with Microsoft Azure*“, [Online]. Available: https://www.youtube.com/watch?v=Pxj9fYgcwV0 (10:14) [Aufgerufen am 27 November 2018]

[29] Chris Pietschmann, „*Introduction to Building IoT Solutions with Microsoft Azure*“, [Online]. Available: https://www.youtube.com/watch?v=Pxj9fYgcwV0 (10:56) [Aufgerufen am 27 November 2018]

[30] David Linthicum, „*Edge computing vs. fog computing: Definitions and enterprise uses*“, [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html [Aufgerufen am 26 November 2018]

[31] Margaret Rouse, „*What is fog computing (fog networking, fogging)?*“, [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/fog-computing-fogging [Aufgerufen am 27 November 2018]

[32] Cisco, „*Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*“, [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf [Aufgerufen am 27. November 2018]

[33] www.cbinsights.com, „*What is Edge Computing?*“, [Online]. Available: https://www.cbinsights.com/research/what-is-edge-computing/ [Aufgerufen am 27 November 2018]

[34] Microsoft Azure, „*IoT Hub – Preise*“, [Online]. Available: https://azure.microsoft.com/de-de/pricing/details/iot-hub/ [Aufgerufen am 27. November 2018]

[35] Microsoft Azure, „*App Service - Preise*“, [Online]. Available: https://azure.microsoft.com/de-de/pricing/details/app-service/linux/ [Aufgerufen am 15 Dezember 2018]

[36] Microsoft Azure, „*Container Registry – Preise*“, [Online]. Available: https://azure.microsoft.com/de-de/pricing/details/container-registry/ [Aufgerufen am 27. November 2018]

[37] Microsoft Azure, „*Preisübersicht Bandbreite*“, [Online]. Available: https://azure.microsoft.com/de-de/pricing/details/bandwidth/ [Aufgerufen am 30. Dezember 2018]

[38] AWS Amazon, „*AWS IoT Greengrass – Preise*“, [Online]. Available: https://aws.amazon.com/de/greengrass/pricing/ [Aufgerufen am 29. Dezember 2018]

[39] AWS Amazon, „*Häufig gestellte Fragen zu AWS Greengrass*“, [Online]. Available: https://aws.amazon.com/de/greengrass/faqs/ [Aufgerufen am 15 Dezember 2018]

[40] Janakiram MSV, „*Azure IoT Edge: A Technology Primer*“, [Online]. Available: https://thenewstack.io/azure-iot-edge-a-technology-primer/ [Aufgerufen am 28 November 2018]

[41] www.wikipedia.org, „*Microsoft Azure*“, [Online]. Available: https://de.wikipedia.org/wiki/Microsoft_Azure [Aufgerufen am 28 November 2018]

[42] Microsoft Azure, „*Was ist Azure IoT Edge?*“, [Online]. Available: https://docs.microsoft.com/de-de/azure/iot-edge/about-iot-edge [Aufgerufen am 28 November 2018]

[43] Margaret Rouse, „*What is AWS Greengrass?*“, [Online]. Available: https://searchaws.techtarget.com/definition/AWS-Greengrass [Aufgerufen am 23 Dezember 2018]

[44] AWS Amazon, „*What Is AWS IoT Greengrass?*“, [Online]. Available: https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html [Aufgerufen am 15 November 2018]

[45] AWS Amazon, „*Lambda-Funktionen auf dem AWS IoT Greengrass-Core ausführen*“, [Online]. Available: https://docs.aws.amazon.com/de_de/greengrass/latest/developerguide/lambda-functions.html [Aufgerufen am 15 November 2018]

[46] www.dianomic.com, „*The Most Powerful Platform to Simplify Data Management in the Fog*“, [Online]. Available: http://dianomic.com/platform/foglamp/ [Aufgerufen am 24 Dezember 2018]

[47] www.dianomic.com, „*Members - Dianomic*“, [Online]. Available: http://dianomic.com/members/#bte [Aufgerufen am 24 Dezember 2018]

[48] FogLAMP, „*Quick Start Guide - Introduction to FogLAMP*“, [Online]. Available: https://foglamp.readthedocs.io/en/latest/quick_start.html#introduction-to-foglamp [Aufgerufen am 24 Dezember 2018]

[49] FogLAMP, „*North Plugins*“, [Online]. Available: https://foglamp.readthedocs.io/en/latest/plugin_developers_guide/04_north_plugins.html [Aufgerufen am 24 Dezember 2018]

[50] FogLAMP, „*FogLAMP Architecture*“, [Online]. Available: https://foglamp.readthedocs.io/en/latest/foglamp_architecture.html [Aufgerufen am 24 Dezember 2018]

[51] FogLAMP, „*South Plugins*“, [Online]. Available: https://foglamp.readthedocs.io/en/latest/plugin_developers_guide/03_south_plugins.html [Aufgerufen am 24 Dezember 2018]

[52] FogLAMP, „*Storage Plugins*“, [Online]. Available: https://foglamp.readthedocs.io/en/latest/plugin_developers_guide/05_storage_plugins.html [Aufgerufen am 24 Dezember 2018]

[53] Microsoft Azure, „*Quickstart: Deploy your first IoT Edge module to a Linux x64 device*“, [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/quickstart-linux [Aufgerufen am 03 Dezember 2018]

[54] Microsoft Azure, „*Install Azure IoT Edge runtime on Linux (ARM32v7/armhf)*“, [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge-linux-arm [Aufgerufen am 03 Dezember 2018]

[55] Microsoft Azure, „*Simulated Temperature Sensor*“, [Online]. Available: https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft.edge-simulated-temperature-sensor-ga?tab=Overview [Aufgerufen am 03 Dezember 2018]

[56] Visual Studio Code, „*Visual Studio Code – Code Editing. Redefined*“, [Online]. Available: https://code.visualstudio.com/ [Aufgerufen am 03 Dezember 2018]

[57] Visual Studio Code, „*Azure IoT Hub Toolkit*“, [Online]. Available: https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.azure-iot-toolkit [Aufgerufen am 03 Dezember 2018]

[58] Microsoft Azure, „*Tutorial: Develop and deploy a Python IoT Edge module to your simulated device*“, [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-python-module [Aufgerufen am 04 Dezember 2018]

[59] Microsoft Azure IoT Edge, „*Stability issues on resource constrained devices*“, [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/troubleshoot#stability-issues-on-resource-constrained-devices [Aufgerufen am 17.12.2018]

[60] FogLAMP, „*Quick Start Guide*“, [Online]. Available: https://foglamp.readthedocs.io/en/master/quick_start.html#introduction-to-foglamp [Aufgerufen 30.12.2018]

[61] SSimonelli, „*Error downloading the 2 kits 'PI Connector Relay Setup Kit' and 'PI Data Collection Manager'*“, [Online]. Available: https://pisquare.osisoft.com/thread/36970-error-downloading-the-2-kits-pi-connector-relay-setup-kit-and-pi-data-collection-manager [Aufgerufen am 22.12.2018]

[62] FogLAMP, „*Downloads*“, [Online]. Available: https://foglamp.readthedocs.io/en/master/92_downloads.html [Aufgerufen am 14.12.2018]

Quellenverzeichnis

[5] itwissen.info (2017, 19. November): *IoT-Gateway*. URL: https://www.itwissen.info/IoT-Gateway-IoT-gateway.html [Aufgerufen am 26. November 2018]

[6] Groß, Bernd (2018, 25. Juni): *Kanten, Wolken und das IoT*. URL: https://www.it-production.com/industrie-4-0-iot/edge-computing-cloud/ [Aufgerufen am 26. November 2018]

8 Anhang

Raspberry Pi Modell	SoC-Typ (Broadcom)	CPU-Familie	Kern-Architektur	Kern-Anzahl / Takt	GPU: Broadcom Video-Core IV	LPDDR2-SDRAM	Veröffentlichung
Raspberry Pi Modell B	BCM 2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/700 MHz	250 MHz 1080p30	512 MB (400 MHz)	Februar 2012
Raspberry Pi Modell A	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/700 MHz	250 MHz 1080p30	256 MB (400 MHz)	Februar 2013
Compute Modul und Development Kit	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/700 MHz	250 MHz 1080p30	512 MB (400 MHz)	April 2014
Raspberry Pi Modell B+	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/700 MHz	250 MHz 1080p30	512 MB (400 MHz)	Juli 2014
Raspberry Pi Modell A+	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/700 MHz	250 MHz 1080p30	256 MB (400 MHz)	November 2014

Tabelle 5: Tabelle aktuelle Raspberry Pi Modelle (Stand: Oktober 2018), sortiert nach Veröffentlichung. (1 von 3) [5]

Raspberry Pi Modell	SoC-Typ (Broadcom)	CPU-Familie	Kern-Architektur	Kern-Anzahl / Takt	GPU: Broadcom Video-Core IV	LPDDR2-SDRAM	Veröffentlichung
Raspberry Pi 2 Modell B	BCM2836	ARM Cortex-A7	ARMv7-A (32-Bit)	4/à 900 MHz	250 MHz 1080p30	1024 MB (400 MHz)	Februar 2015
Raspberry Pi Zero	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/1 GHz	400 MHz 1080p60	512 MB (450 MHz)	November 2015
Raspberry Pi 3 Modell B	BCM2837	ARM Cortex-A53	ARMv8-A (64 Bit)	4/à 1,2 GHz	400 MHz 1080p60	1024 MB (450 MHz)	Februar 2016
Raspberry Pi Zero mit Kamera-Port	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/1 GHz	400 MHz 1080p60	512 MB (450 MHz)	Mai 2016
Raspberry Pi 2 Modell B v.1.2	BCM2837	ARM Cortex-A53	ARMv8-A (64-Bit)	4/à 900 MHz	250 MHz 1080p30	1024 MB (400 MHz)	September 2016

Tabelle 6: Tabelle aktuelle Raspberry Pi Modelle (Stand: Oktober 2018), sortiert nach Veröffentlichung. (2 von 3) [5]

Raspberry Pi Modell	SoC-Typ (Broadcom)	CPU-Familie	Kern-Architektur	Kern-Anzahl / Takt	GPU: Broadcom Video-Core IV	LPDDR2-SDRAM	Veröffentlichung
Compute Modul 3 und Development Kit	BCM2837	ARM Cortex-A53	ARMv8-A (64-Bit)	4/à 1,2 GHz	400 MHz 1080p60	1024 (500 MHz)	Januar 2017
Raspberry Pi Zero W (Wireless)	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/1 GHz	400 MHz 1080p60	512 MB (450 MHz)	Februar 2017
Raspberry Pi Zero WH (Header)	BCM2835	ARM1176JZF-S	ARMv6Z (32-Bit)	1/1 GHz	400 MHz 1080p60	512 MB (450 MHz)	Januar 2018
Raspberry Pi 3 Modell B+	BCM2837B 0	ARM Cortex-A53	ARMv8-A (64 Bit)	4/à 1,4 GHz	400 MHz 1080p60	1024 MB (500 MHz)	März 2018

Tabelle 7: Tabelle aktuelle Raspberry Pi Modelle (Stand: Oktober 2018), sortiert nach Veröffentlichung. (3 von 3) [5]

- Unterstützte Plattformen:
  - Architektur: ARMv7l; OS: Linux; Distribution: Raspbian Jessie, 2017-03-02
  - Architektur: x86_64; Betriebssystem: Linux; Distribution: Amazon Linux (amzn-ami-hvm-2016.09.1.20170119-x86_64-ebs)
  - Architektur: x86_64; Betriebssystem: Linux; Bereitstellung: Ubuntu 14.04 – 16.04
  - Architektur: ARMv8 (AArch64); OS: Linux; Distribution: Ubuntu 14.04 – 16.04 (Annapurna Alpine V2)
- Die folgenden Elemente sind erforderlich:
  - Mindestens 128 MB RAM, zugewiesen zum AWS Greengrass-Kern-Gerät.
  - Linux-Kernel Version 4.4 oder höher: Es können zwar verschiedene Versionen mit AWS Greengrass funktionieren, für optimale Sicherheit und Leistung empfehlen wir jedoch Version 4.4 oder höher.
  - Glibc-Bibliothek, Version 2.14 oder höher.
  - Das Verzeichnis /var/run muss auf dem Gerät vorhanden sein.
  - AWS Greengrass erfordert die Aktivierung des Hardlink- und Softlink-Schutzes auf dem Gerät. Ohne diesen kann AWS Greengrass nur im unsicheren Modus mit dem Marker -i ausgeführt werden.
  - Die Benutzer ggc_user und die Gruppe ggc_group müssen beide auf dem Gerät vorhanden sein.
  - Die folgenden Linux-Kernel-Konfigurationen müssen auf dem Gerät aktiviert sein:
    - Namespace: CONFIG_IPC_NS, CONFIG_UTS_NS, CONFIG_USER_NS, CONFIG_PID_NS
    - CGroups: CONFIG_CGROUP_DEVICE, CONFIG_CGROUPS, CONFIG_MEMCG
    - Sonstiges: CONFIG_POSIX_MQUEUE, CONFIG_OVERLAY_FS, CONFIG_HAVE_ARCH_SECCOMP_FILTER, CONFIG_SECCOMP_FILTER, CONFIG_KEYS, CONFIG_SECCOMP

Tabelle 8: Tabelle technische Voraussetzungen für AWS Greengrass[9]. (1 von 2)

[9] AWS Greengrass, *Unterstützte Plattformen und Anforderungen*, URL: https://docs.aws.amazon.com/en_us/greengrass/latest/developerguide/what-is-gg.html [Aufgerufen am 15. November 2018]

- /dev/stdin, /dev/stdout und /dev/stderr müssen aktiviert sein.
- Der Linux-Kernel muss cgroups unterstützen.
- Der *Speicher* cgroup muss aktiviert und gemountet sein, damit AWS Greengrass die Speichergrenze für Lambda-Funktionen festlegen kann.
- Das Stammzertifikat für Amazon S3 und AWS IoT muss im vertrauenswürdigen Speicher des Systems vorhanden sein.

- Die folgenden Elemente sind optional:
  - Die *Geräte* cgroup müssen aktiviert und gemountet sein, wenn Lambda-Funktionen mit LocalResource Access (LRA) zum Öffnen von Dateien auf dem AWS Greengrass-Kern-Gerät verwendet werden.
  - Python Version 2.7 ist erforderlich, wenn Python Lambda-Funktionen verwendet werden. Wenn dies der Fall ist, stellen Sie sicher, dass es Ihrer PATH-Umgebungsvariablen hinzugefügt ist.
  - NodeJS Version 6.10 oder höher ist erforderlich, wenn Node.JS Lambda-Funktionen verwendet werden. Wenn dies der Fall ist, stellen Sie sicher, dass es Ihrer PATH-Umgebungsvariablen hinzugefügt ist.
  - Java Version 8 oder höher ist erforderlich, wenn Java Lambda-Funktionen verwendet werden. Wenn dies der Fall ist, stellen Sie sicher, dass es Ihrer PATH-Umgebungsvariablen hinzugefügt ist.

OpenSSL 1.01 oder höher ist für Greengrass OTA-Agent sowie für die
Befehle: wget, realpath, tar, readlink, basename, dirname, pidof, df, grep und umounterfor derlich.

Tabelle 9: Tabelle technische Voraussetzungen für AWS Greengrass[9]. (2 von 2)

Tabelle 10: main.py-Code

```
# Copyright (c) Microsoft. All rights reserved.
# Licensed under the MIT license. See LICENSE file in the project root
for
# full license information.

import json # 1. Schritt: json-Bibliothek einfügen
import random
import time
import sys
import iothub_client
# pylint: disable=E0611
from iothub_client import IoTHubModuleClient, IoTHubClientError,
IoTHubTransportProvider
from iothub_client import IoTHubMessage, IoTHubMessageDispositionResult,
IoTHubError

# 2. Schritt: Grenzwert bestimmen
TEMPERATURE_THRESHOLD = 25
TWIN_CALLBACKS = 0

# messageTimeout - the maximum time in milliseconds until a message
times out.
# The timeout period starts at IoTHubModuleClient.send_event_async.
# By default, messages do not expire.
MESSAGE_TIMEOUT = 10000

# global counters
RECEIVE_CALLBACKS = 0
SEND_CALLBACKS = 0

# Choose HTTP, AMQP or MQTT as transport protocol.  Currently only MQTT
is supported.
PROTOCOL = IoTHubTransportProvider.MQTT

# Callback received when the message that we're forwarding is processed.
def send_confirmation_callback(message, result, user_context):
    global SEND_CALLBACKS
    print ( "Confirmation[%d] received for message with result = %s" %
(user_context, result) )
    map_properties = message.properties()
    key_value_pair = map_properties.get_internals()
    print ( "    Properties: %s" % key_value_pair )
    SEND_CALLBACKS += 1
    print ( "    Total calls confirmed: %d" % SEND_CALLBACKS )
```

```
# 3. Schritt: receive_message_callback-Funktion ersetzt
# receive_message_callback is invoked when an incoming message arrives
on the specified
# input queue (in the case of this sample, "input1").  Because this is a
filter module,
# we forward this message to the "output1" queue.
def receive_message_callback(message, hubManager):
    global RECEIVE_CALLBACKS
    global TEMPERATURE_THRESHOLD
    message_buffer = message.get_bytearray()
    size = len(message_buffer)
    message_text = message_buffer[:size].decode('utf-8')
    print ( "    Data: <<<%s>>> & Size=%d" % (message_text, size) )
    map_properties = message.properties()
    key_value_pair = map_properties.get_internals()
    print ( "    Properties: %s" % key_value_pair )
    RECEIVE_CALLBACKS += 1
    print ( "    Total calls received: %d" % RECEIVE_CALLBACKS )
    data = json.loads(message_text)
    if "machine" in data and "temperature" in data["machine"] and
data["machine"]["temperature"] > TEMPERATURE_THRESHOLD:
        map_properties.add("MessageType", "Alert")
        print("Machine temperature %s exceeds threshold %s" %
(data["machine"]["temperature"], TEMPERATURE_THRESHOLD))
    hubManager.forward_event_to_output("output1", message, 0)
    return IoTHubMessageDispositionResult.ACCEPTED

# 4. Schritt: Die Funktion module_twin_callback einfügen
# module_twin_callback is invoked when the module twin's desired
properties are updated.
def module_twin_callback(update_state, payload, user_context):
    global TWIN_CALLBACKS
    global TEMPERATURE_THRESHOLD
    print ( "\nTwin callback called with:\nupdateStatus = %s\npayload =
%s\ncontext = %s" % (update_state, payload, user_context) )
    data = json.loads(payload)
    if "desired" in data and "TemperatureThreshold" in data["desired"]:
        TEMPERATURE_THRESHOLD = data["desired"]["TemperatureThreshold"]
    if "TemperatureThreshold" in data:
        TEMPERATURE_THRESHOLD = data["TemperatureThreshold"]
    TWIN_CALLBACKS += 1
    print ( "Total calls confirmed: %d\n" % TWIN_CALLBACKS )

class HubManager(object):

    def __init__(
            self,
```

```
            protocol=IoTHubTransportProvider.MQTT):
        self.client_protocol = protocol
        self.client = IoTHubModuleClient()
        self.client.create_from_environment(protocol)

        # set the time until a message times out
        self.client.set_option("messageTimeout", MESSAGE_TIMEOUT)

        # sets the callback when a message arrives on "input1" queue.
Messages sent to
        # other inputs or to the default will be silently discarded.
        self.client.set_message_callback("input1",
receive_message_callback, self)

        # 5. Schritt: Methode client.set_module_twin_callback einfügen
        # Sets the callback when a module twin's desired properties are
updated.
        self.client.set_module_twin_callback(module_twin_callback, self)

    # Forwards the message received onto the next stage in the process.
    def forward_event_to_output(self, outputQueueName, event,
send_context):
        self.client.send_event_async(
            outputQueueName, event, send_confirmation_callback,
send_context)

def main(protocol):
    try:
        print ( "\nPython %s\n" % sys.version )
        print ( "IoT Hub Client for Python" )

        hub_manager = HubManager(protocol)

        print ( "Starting the IoT Hub Python sample using protocol
%s..." % hub_manager.client_protocol )
        print ( "The sample is now waiting for messages and will
indefinitely.  Press Ctrl-C to exit. ")

        while True:
            time.sleep(1)

    except IoTHubError as iothub_error:
        print ( "Unexpected error %s from IoTHub" % iothub_error )
        return
    except KeyboardInterrupt:
        print ( "IoTHubModuleClient sample stopped" )
```

```
if __name__ == '__main__':
    main(PROTOCOL)
```

Tabelle 11: deployment.template.json

```
{
  "$schema-template": "1.0.0",
  "modulesContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.0",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v1.25",
            "loggingOptions": "",
            "registryCredentials": {
              "jtreg": {
                "username": "$CONTAINER_REGISTRY_USERNAME_jtreg",
                "password": "$CONTAINER_REGISTRY_PASSWORD_jtreg",
                "address": "jtreg.azurecr.io"
              }
            }
          }
        },
        "systemModules": {
          "edgeAgent": {
            "type": "docker",
            "settings": {
              "image": "mcr.microsoft.com/azureiotedge-agent:1.0",
              "createOptions": {}
            }
          },
          "edgeHub": {
            "type": "docker",
            "status": "running",
            "restartPolicy": "always",
            "settings": {
              "image": "mcr.microsoft.com/azureiotedge-hub:1.0",
              "createOptions": {
                "HostConfig": {
                  "PortBindings": {
                    "5671/tcp": [
                      {
                        "HostPort": "5671"
                      }
                    ],
                    "8883/tcp": [
                      {
                        "HostPort": "8883"
```

```
                    }
                  ],
                  "443/tcp": [
                    {
                      "HostPort": "443"
                    }
                  ]
                }
              }
            }
          }
        },
        "modules": {
          "tempSensor": {
            "version": "1.0",
            "type": "docker",
            "status": "running",
            "restartPolicy": "always",
            "settings": {
              "image": "mcr.microsoft.com/azureiotedge-simulated-
temperature-sensor:1.0",
              "createOptions": {}
            }
          },
          "tempTestModule": {
            "version": "1.0",
            "type": "docker",
            "status": "running",
            "restartPolicy": "always",
            "settings": {
              "image": "${MODULES.tempTestModule}",
              "createOptions": {}
            }
          }
        }
      }
    },
    "$edgeHub": {
      "properties.desired": {
        "schemaVersion": "1.0",
        "routes": {
          "tempTestModuleToIoTHub": "FROM
/messages/modules/tempTestModule/outputs/* INTO $upstream",
          "sensorTotempTestModule": "FROM
/messages/modules/tempSensor/outputs/temperatureOutput INTO
BrokeredEndpoint(\"/modules/tempTestModule/inputs/input1\")"
```

```
        },
        "storeAndForwardConfiguration": {
          "timeToLiveSecs": 7200
        }
      }
    },
    "PythonModule": {
      "properties.desired":{
          "TemperatureThreshold":25
        }
    }
  }
}
```