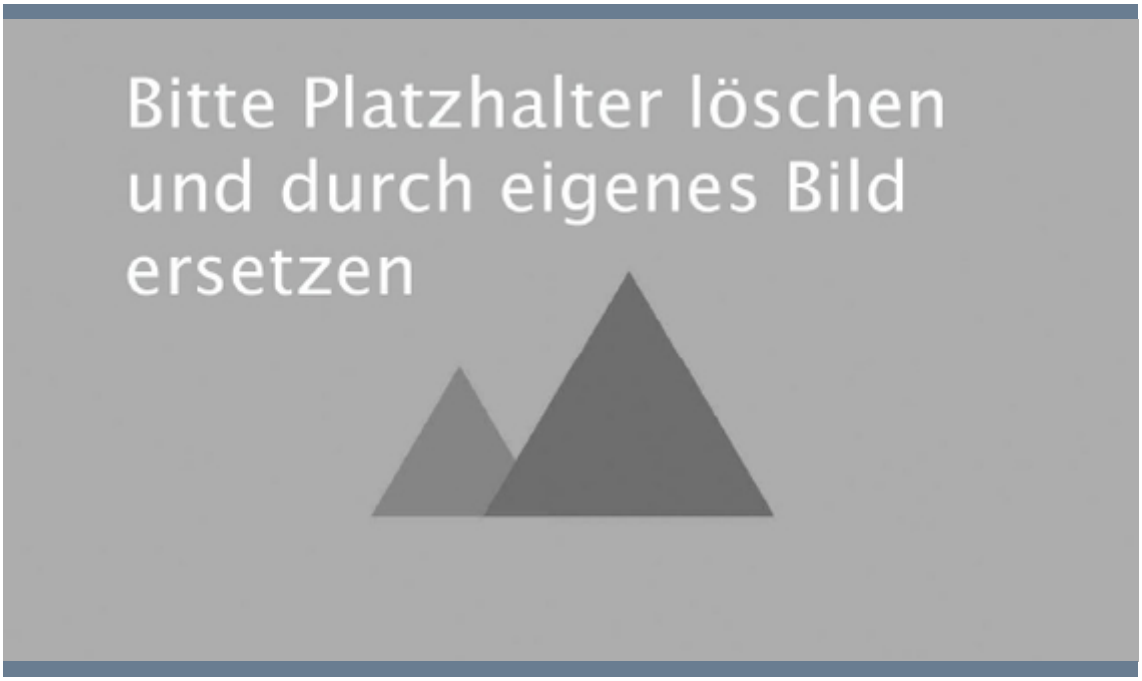

Titel des Berichts

Hier steht ein Untertitel

Bachelorthesis

Studiengang:	Informatik
Autor:	Adrian Bärtschi
Betreuer:	Prof. Dr. Ing. Reto E. Koenig
Experte:	Dr. Federico Flueckiger
Datum:	23.08.2015

Versionen

Version	Datum	Status	Bemerkungen
0.1	23.08.2015	Entwurf	Dokument erstellt

1. Management Summary

- Was ist das Ziel der Thesis
- Kurzbeschrieb des Themas
- Was wurde gemacht, erreicht

Inhaltsverzeichnis

2. Einleitung

Bei Systemen im Internet of Things (IoT) Umfeld sind sehr viele und auch unterschiedliche Geräte in einen Netzwerk miteindander verbunden. Für diese IoT - Machine-To-Machine Kommunikation werden andere Netzwerkprotokolle eingesetzt als im 'klassischen' Internet. Dies ist nötig, weil die Geräte stark eingeschränkte Resourcen haben und die Netzwerke geringe Bandbreiten aufweisen.

MQTT ist ein Protokoll, welches die Anforderungen für IoT Systeme erfüllen soll. Beim Entwurf des Protokolls wurde auf Einfachheit und Leichtgewichtigkeit grossen Wert gelegt. Mit MQTT ist es möglich, beliebige Daten in beliebiger Codierung zu versenden. Dies bietet den Entwicklern der Systeme grosse Freiheiten.
Es ist aber ersichtlich, dass die fehlende Struktur und Beschreibung der Daten gewisse Schwierigkeiten mit sich bringen kann. Eine Anwendung, welche Daten per MQTT erhält, muss wissen wie diese vom Absender codiert wurden und was sie bedeuten.

Beispielsweise wird eine Messung eines Temperatursensors via MQTT versendet werden. Der Sensor liefert den Wert 22° Celsius. Der ganzzahlige Wert 22 wird als binärer Wert 10110 versendet. Der Empfänger erhält nun die MQTT Nachricht 10110. Er hat aber keine Ahnung, was mit diesem Wert anzufangen ist. Wird dieser Wert als Fliesskommazahl interpretiert (Float), so wird der Wert in 3.0828 konvertiert. Der Empfänger müsste wissen, dass es sich um einen Integer Wert (hier 32 bit signed) handelt, damit die Daten in das richtige Format gebracht werden können. Ausserdem muss der Empfänger jetzt noch wissen, dass in welcher Einheit (Celsius, Fahrenheit, etc.) die Temperatur übermittelt wird.

Um diese Information vom Anbieter der Daten resp. des Diestes oder Geräts den Entwicklern einer Anwedung zur Verfügung zu stellen, werden zurzeit Beschreibungen in Dokumentenform verwendet. Diese sogenannte out-out-band Dokumentation ist aber aufwändig in der Nachführung und bekanntegabe von Änderungen. Auch ist es schwierig die Struktur der Daten einheitlich und klar zu erklären.

Ziel dieser Thesis ist es, dass Geräte welche ihre Daten per MQTT versenden, die Möglichkeit erhalten, sich selbst inkusive ihrer Daten und Möglichkeiten zur Interkation zu beschreiben. Dies soll so getan werden, dass die Beschreibung für Mensch und Maschine les- und verstehbar ist, die Eigenschaften der eingeschränkten Geräte und Netze berücksichtig wird und der MQTT Standard weiterhin eigehalten wird.

3. Project Managament

3.1. Organisation

TODO: formatieren

Name	Rolle	Aufgaben
Adrian Bärtschi	Studierender	Selbständiges Projektmanagement während der Thesis. Setzt die Aufgaben gemäss Aufgabenstellung und Vorgaben Betreuer um. Organisiert Kommunikation mit Betreuer und Experte.
Prof. Dr. Ing. Reto E. König Berner Fachhochschule	Betreuer	Hauptansprechsperson für Studierenden, verantwortlich für den Ablauf der Thesis. Beurteilung aufgrund von Aufgabenstellung und abgegebenen Artefakten.
Dr. Federico Flueckiger Eidg. Finanzdepartement	Experte	Beurteilung aufgrund der Aufgabenstellung und abgelieferten Artefakten sowie mindestends ein bis zwei Sitzungen mit dem Studierenden.

Tabelle 3.1.: Involvierte Personen und deren Aufgaben

3.2. Meilensteine

Datum	Meilenstein	Bemerkungen
25.09.2015	Initialisierung, Vorgehen geklärt	-
01.10.2015	Ziele definiert	-
15.10.2015	Prototyp erstellt, Konzept der Lösung skizziert	-
30.11.2015	Implementation System abgeschlossen	-
15.12.2015	Implementation Demo Applikation abgeschlossen	-
18.01.2016	Dokumentation inhaltlich abgeschlossen	-
21.01.2016	Dokumentation fertiggestellt	-

Tabelle 3.2.: Meilensteine der Thesis

3.3. Resourcen

Kosten, etc.

3.4. Ziele

Die zu entwickelnde Lösung soll folgendes beinhalten:

- asdf

3.5. Aufwände

Task	Soll	Ist	Bemerkungen
asdf	2	2	df

Tabelle 3.3.: Planung der Tasks und Auswetung der Aufwände

3.6. Deliverables / Termine

Datum	Thema
	Abgabe Texte/Grafiken/etc. für das Book
	Abgabe Elektronische Version des Posters
21.01.2016	Abgabe Projektdokumentation an Betreuer, Experte, Sekretariat
22.01.2016	Final Day Bern, Präsentation und Ausstellung
	Verteidigung

Tabelle 3.4.: Termine und Fristen

4. Technologie

4.1. MQTT

Aus Proj2 übernehmen

4.2. Bestehende Konzepte

Die verschiedenen Hersteller von MQTT Anwendungen entwickeln jeweils ihre eigenen Ansätze, um die Daten zu strukturieren.

4.2.1. IBM Internet of Things Foundation

IBM hat unter dem Brand 'IBM IoT Foundation' [2] einen Dienst entwickelt, mit dem vernetzte Geräte verwaltet werden können. Als Kommunikationsprotokoll wird MQTT eingesetzt. Die Plattform verwendet folgende konzeptionelle Ideen:

- Organizations: Eindeutige Identifikation der Kunden der Plattform
- Devices: Beliebiges vernetztes Gerät. Versendet Events und reagiert auf Commands.
- Applications: Anwendung, welche mit den Daten der Devices interagiert.
- Events: Daten, welche von den Devices an die Plattform gesendet werden
- Commands: Applications können mittels Commands mit den Devices kommunizieren.

Events
Events müssen an ein definiertes Topic nach folgendem Schema gesendet werden:
`iot-2/evt/<event_id>/fmt/<format_string>`

Beispiel: `iot-2/evt/temperature_outdoor/fmt/json`

Eine Anwendung, welche Events empfangen möchte, muss sich auf ein Topic in der Form `iot-2/type/<device_type>/id/<device_id>/evt/<event_id>/fmt/<format_string>` registrieren. Die Teile `device_type`, `device_id`, `event_id` und `format_string` des Topics können auch mit dem Wildcard Charakter '+' ersetzt werden, um jeweils alle Events der Komponenten zu erhalten.

Beispiel: `iot-2/type/temp/id/+/evt/temperature_outdoor/fmt/+`

Commands
Um einen Command zu erzeugen, sendet eine Anwendung eine MQTT Message mit Topic gemäss folgenden Schema: `iot-2/type/<device_type>/id/<device_id>/cmd/<command_id>/fmt/<format_string>`

Beispiel: `iot-2/type/temp/id/sensor1/cmd/setInterval/fmt/json`

Das Device `sensor1` würde damit eine Message auf Topic `iot-2/cmd/setInterval/fmt/json` erhalten.

Payload Format

Grundsätzlich unterstützt IBM IoT Foundation ein beliebiges Payload Format. Es wird jedoch empfohlem, JSON zu verwenden. Um alle Funktionen der Platform nutzen zu können, müssen die JSON Dokumente zusätzlich nach den Vorgaben [3] von IBM strukturiert sein.

```
{
  "d": {
    "host": "IBM700-R9E683D",
    "mem": 54.9,
    "network": {
      "up": 1.22,
      "down": 0.55
    },
    "cpu": 1.3,
  }
}
```

Listing 1: JSON Beispiel im IBM IoTF Payload Format

4.2.2. Tinkerforge MQTT Proxy

Tinkerforge hat ein modulares System von Sensoren und Aktoren (so gennate Bricklets) entwickelt, die u. A. für Prototyping und in der Ausbildung (auch an der BFH) eingesetzt werden. Um die Mudule zu steuern, wird klassischerweise das bereitstellete SDK in der gewünschten Programmiersprache verwendet. Tinkerforge ausserdem eine Anwendung entwickelt und die Bausteine per MQTT ansprechen zu können [1].

Topics

Die Tinkerforge Devices senden ihre Daten an ein MQTT Topic nach Schema `tinkerforge/<prefix>/<uid>/<suffix>`.

Ein Temperatur Bricklet mit Unique Identifier (UID) `xf2` würde also den gemessenen Wert an das Topic `tinkerforge/bricklet/temperature/xf2/temperature` senden.

Die Bricklets reagieren auf Messages die an ein passendes Topic mit Siffix `/set` gesendet werden. Sollen beispielsweise die LEDs des Dualbutton Bricklets mit UID `mxg` eingeschaltet werden, muss eine Message an das Topic `tinkerforge/bricklet/dual_button/mxg/led_state/set` gesendet werden mit folgendem Payload:

```
{
    "led_l": 2,
    "led_r": 2
}
```

Listing 2: JSON Beispiel Tinkerforge Format

Payload Format

Die Tinkerforge MQTT Komponente verwendet JSON als Datenformat für die Messages. Jede Message, die von einem Bricklet gesendet wird, enthält unter dem Key `_timestamp` den Zeitpunkt der Erzeugung als UNIX Timestamp.

```
{
  "_timestamp": 1440083842.785104,
  "temperature": 2343
}
```

Listing 3: JSON Beispiel Tinkerforge Format

Die Beschreibung, unter welchen Topics Daten publiziert werden und wie die Bricklets angesprochen werden können, ist in der Dokumentation von Tinkerforge [1] beschrieben.

4.3. Datenbeschreibung

4.3.1. JSON Schema

4.3.2. Protocol Buffer

4.3.3. DFDL

Aus Proj2 übernehmen

4.3.4. Vergleich

4.4. Konzept aus der nicht-IoT Welt

SOAP-WSDL, REST, etc. HATEOAS

5. Konzept

5.1. Allgmein

Bei der Beschreibung eines Devices werden folgende drei Bereiche unterschieden:

State:
Die Menge aller Eigenschaften des Devices und deren Werte wird als State bezeichnet. Beispielsweise Name, Firmwareversion, etc. Der State eines Devices ist beständig, d.h. solange keine Interaktion geschieht, verändert sich der State nicht.

Events:
Tritt auf dem Device ein Ereignis ein, wird dadurch ein Event erzeugt. Ein Event wird grundsätzlich vom Device selbst ausgelöst. Beispielsweise erzeugt ein Temperatursensor alle 5 Sekunden ein Event welches den Messwert enthält. Applikationen registrieren sich, um bestimmte Events von den Devices zu erhalten.

Commands:
Eine Anwendung interagiert mit einem Device, indem Commands an eines oder mehrere Devices gesendet werden. Die Devices empfangen die Commands und reagieren entsprechend. Ein Command ist bestimmt durch einen Namen und Parameter mit dazugehörigen Werten. Beispielsweise kann bei einem Temperatursensor über einen Command `SetInterval: 10s` der Abstand der Messungen eingestellt werden. Ein Device muss bekanntgeben, auf welche Commands mit welchen Paramtern es reagiert.

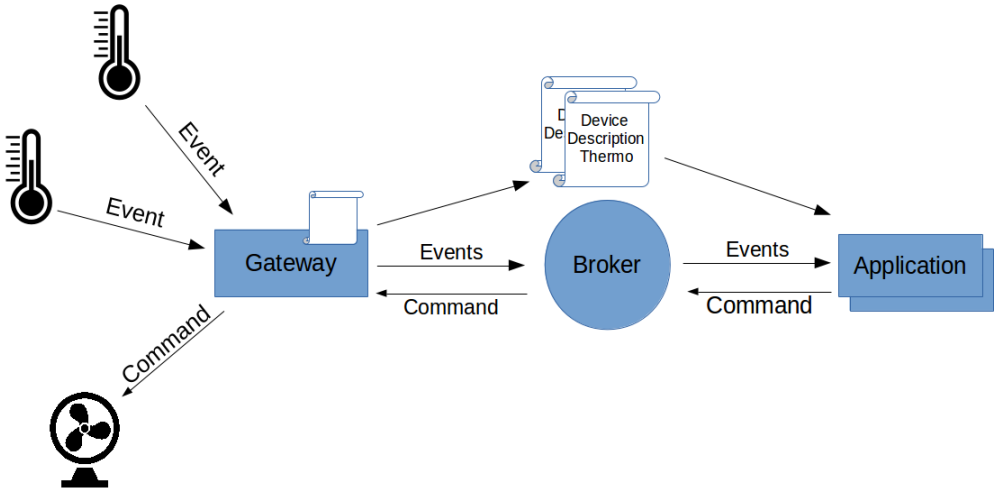

Abbildung 5.1.: Übersicht

5.2. Hierarchie Topics

Die Topic Hiearachie wird nach folgenden Muster aufgebaut:

Level	Beschreibung	Beispiel
0	**Identifikation Anwendung** Eindeutige identifikation der Anwendung.	ch.bfh.barta3.myApp
1	**Master Host** asdf	-
2	**Gateway (Stack)**	-
3	**Device Typ** Bezeichnung, um was für einen Typ von Sensor oder Aktor es sich handelt.	Temperatursensor
4	**Device ID** Da mehrere Devices vom selben Typ im Einsatz sein können, wird auf dieser Stufe mit einer eindeutigen ID der konkrete Sensor resp. Aktor angegeben.	mxg
4	**Device Description** d	-
5	**Kategorisierung Devicedaten** Die Eigenschaften und Daten werden in die nachfolgenden drei Teile aufgegliedert.	Events, State, Commands
6	**Events** TODO	Temperatur
6	**State** TODO	Interval
6	**Commands** TODO	setInterval

Tabelle 5.1.: Topic Hierarchie

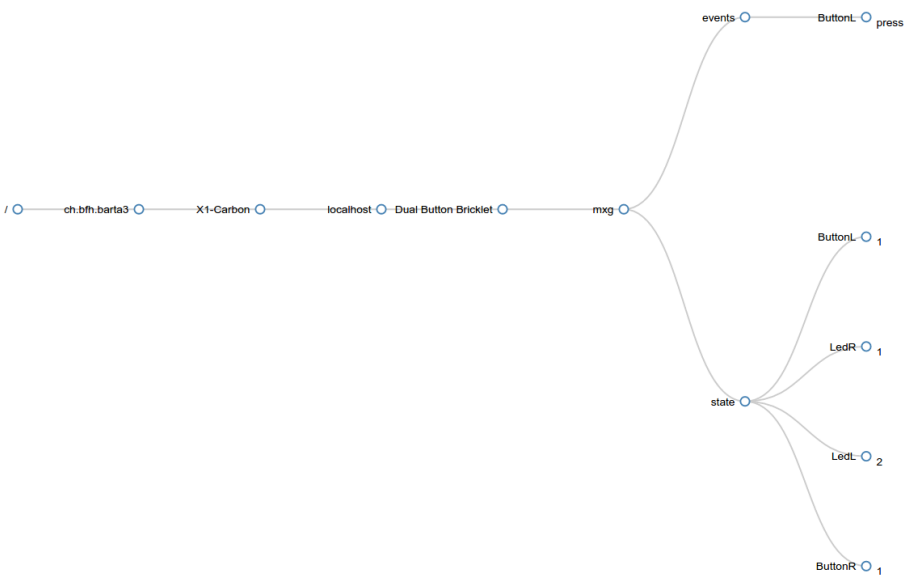

Abbildung 5.2.: Visualisierung MQTT Topics Tinkerforge IR Temperatur Sensor

TODO: better diagram

5.3. Device Description

Die Beschreibung eines Device enthält die drei Elemente State, Events und Commands.

State
Die State Informationen des Devices werden als Key-Value Paare abgebildet. Der Name der Keys muss eindeutig sein.

Events
Ein Device muss beschreiben, welche Events es versendet und was darin enthalten ist. Ein Event ist folgendermassen aufgebaut: Name

Level	Beschreibung	Beispiel
0	**Identifikation Anwendung** Eindeutige identifikation der Anwendung.	ch.bfh.barta3.myApp
1	**Master Host** asdf	-
2	**Gateway (Stack)**	-
3	**Device Typ** Bezeichnung, um was für einen Typ von Sensor oder Aktor es sich handelt.	Temperatursensor
4	**Device ID** Da mehrere Devices vom selben Typ im Einsatz sein können, wird auf dieser Stufe mit einer eindeutigen ID der konkrete Sensor resp. Aktor angegeben.	mxg
4	**Device Description** d	-
5	**Kategorisierung Devicedaten** Die Eigenschaften und Daten werden in die nachfolgenden drei Teile aufgegliedert.	Events, State, Commands
6	**Events** TODO	Temperatur
6	**State** TODO	Interval
6	**Commands** TODO	setInterval

Tabelle 5.2.: Topic Hierarchie

6. Umsetzung

6.1. Tinkerforge Teil

Enumeration

6.2. MQTT Teil

Topic Tree

6.3. Thing Description

7. Schlussfolgerungen/Fazit

Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum: Bern, 23.08.2015

Namen Vornamen: Bärtschi Adrian

Unterschriften: .....

Acronyms

IoT Internet of Things. 1

UID Unique Identifier. 6

Literaturverzeichnis

[1] Brick MQTT Proxy – Tinkerforge. Accessed: 2015-11-02. [Online]. Available: http://www.tinkerforge.com/en/doc/Software/Brick_MQTT_Proxy.html

[2] IBM Internet of Things Foundation. Accessed: 2015-11-02. [Online]. Available: https://internetofthings.ibmcloud.com

[3] Message Payload — IBM IOT Foundation 1.0 documentation. Accessed: 2015-11-02. [Online]. Available: https://docs.internetofthings.ibmcloud.com/messaging/payload.html

Abbildungsverzeichnis

Tabellenverzeichnis

A. Beliebiger Anhang

Phasellus eget velit massa, sed faucibus nisi. Etiam tincidunt libero viverra lorem bibendum ut rutrum nisi volutpat. Donec non quam vitae lacus egestas suscipit at eu nisi. Maecenas non orci risus, at egestas tellus. Vivamus quis est pretium mauris fermentum consectetur. Cras non dolor vitae nulla molestie facilisis. Aliquam euismod nisl eget risus pretium non suscipit nulla feugiat. Nam in tortor sapien. Nam lectus nibh, laoreet eu ultrices nec, consequat nec sem. Nulla leo turpis, suscipit in vulputate a, dapibus molestie quam. Vestibulum pretium, purus sed suscipit tempus, turpis purus fermentum diam, id cursus enim mi a tortor. Proin imperdiet varius pellentesque. Nam congue, enim sit amet iaculis venenatis, dui neque ornare purus, laoreet porttitor nunc justo vel velit. Suspendisse potenti. Nulla facilisi.