Message Queuing Telemetry Transport

Implementierung einer IoT-Anwendung auf Basis von MQTT

Maximilian Gaul, Lukas Dorner

01.07.2019

MQTT

Control Packet

Besteht aus bis zu drei Teilen:

- **Fixed Header** - in allen MQTT Paketen vorhanden
- **Variable Header**
- **Payload**

MQTT

Fixed Header

Bit →	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
Byte 2	Remaining Length							

MQTT

Fixed Header - Byte 1

- *Control Packet Type*[7:4] gibt an, welche Art von Paket versendet wird:
  - *CONNECT* Client will sich mit dem Server verbinden
  - *CONNACK* Verbindungs-*ACK*
  - *PUBLISH* Sensor schickt neuen Wert an Server
  - *SUBSCRIBE* Client *abonniert* ein Thema, Server leitet *PUBLISH* weiter an Abonnenten
- *Flag*[3:0] sind spezifisch je nach Control Packet Type gesetzt
  ungültige Flags führen zu einem Schließen der Verbindung durch den Empfänger
  - *DUP* 0 := Erster Versuch, ein *PUBLISH* zu senden, 1 := Möglicherweise erneutes Senden eines *PUBLISH*
  - *QoS* Gibt an, wie oft ein *PUBLISH* maximal bzw. minimal gesendet wird
    0 := höchstens einmal versendet
    1 := Server *muss* Nachricht speichern & an *zukünftige* Abonnenten senden
  - *RETAIN* 0 := Server *darf nicht* 1

MQTT

Fixed Header - Byte 2

- Gibt die Anzahl der verbleibenden Bytes in diesem Paket an (beinhaltet den *variablen* Header und die *Payload*), maximal 4 Bytes lang
- Das *höchstwertige* Bit eines Bytes gibt an, ob noch ein weiteres Byte für die Kodierung der verbleibenden Länge verwendet wurde
  ⇒ Es können *Control* Packets bis zu einer Größe von 256 MB kodiert werden

MQTT

Variable Header

- Unterschiedliche Bedeutung je nach *Control Packet Type*
  - Zuordnung von *PUBLISH* Paketen mit *QoS* > 0 z.B. über einen 16 Bit Identifier
  - *PUBLISH* mit *QoS* = 0 darf keinen Packet Identifier enthalten
  - Bei einem *CONNECT* Paket definiert er wichtige Verbindungseigenschaften

MQTT

Payload

- *CONNECT*, *SUBSCRIBE* und *UNSUBSCRIBE* haben immer eine Payload
- *PUBLISH* Pakete auch ohne Payload

MQTT

Verbindungsaufbau

MQTT

Verbindungsabbau

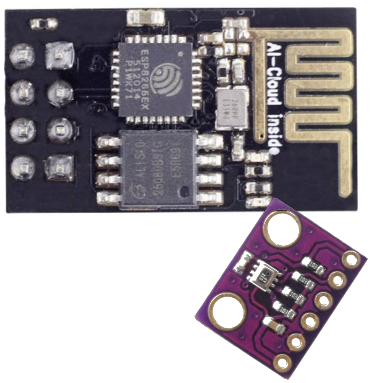
MQTT

Netzwerk & Sicherheit

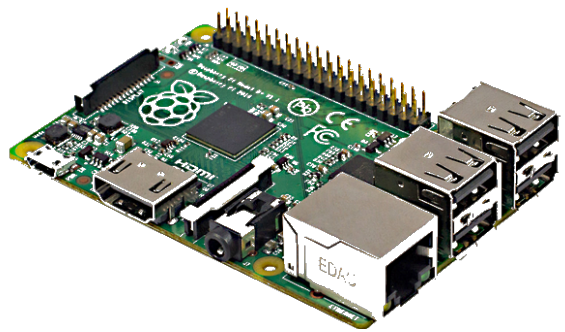
- Als Transportprotokoll kann z.B. *TCP*, *TLS* oder *WebSocket* verwendet werden
  - *UDP* ist ungeeignet, da *verbindungslos* - keine Garantie für Zustellung der Daten
- *MQTT* sieht sich selbst nur als Transportprotokoll, für Sicherheit zu sorgen liegt in der Verantwortung des Implementierenden
  - Empfiehlt *TLS*

Projekt

Grundlegender Aufbau



MQTT PUBLISH



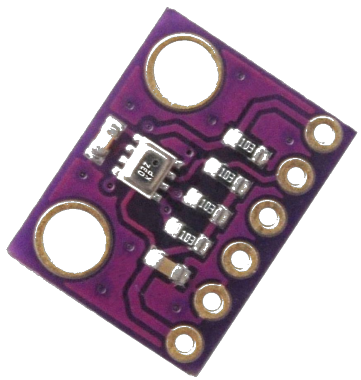
ESP8266 + BMP280

Raspberry Pi + Mosquitto

Projekt

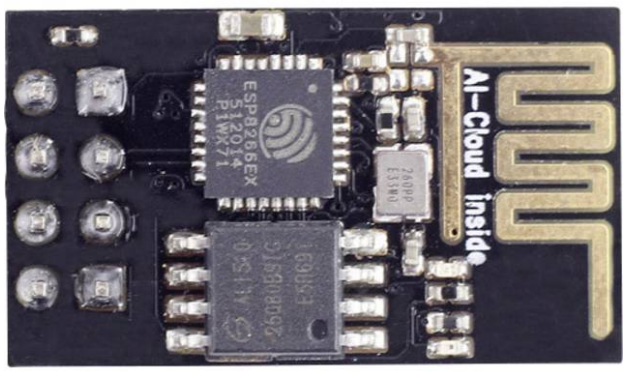
BMP280

- Anbindung über den I^2C-Bus (*Inter Integrated Circuit*)
- Temperatur- und Drucksensor mit einer Auflösung von bis zu 20 Bit
- Ausgelesene Werte verrechnet mit voreingestellten Kalibrationsparametern ergeben Temperatur



Projekt

ESP8266



- Unterstützt 802.11 b/g/n mit bis zu 72.2Mbps
- Antenne als Leiterbahn auf der Platine
- Tensilica L106 32-bit *RISC* mit bis zu 160 MHz (mit *PLL-OC* auch > 300 MHz)
- 96 KByte RAM
- 4 MB Flash Speicher, per *SPI* angebunden
- Ursprünglich als *dummer*, nur per *AT*-Kommandos bedienbarer WiFi-Chip verkauft
- Nach viel *Reverse-Engineering* der Community und schließlich Freigeben eines *SDK* durch den Hersteller heute frei programmierbar (z.B. mit *Arduino* oder in *C / Assembler*)

Projekt

ESP8266 Firmware

- Implementierung der *Master-*I^2C-Schnittstelle in Software (kein *HW-I2C*)
- Implementierung des *MQTT-PUBLISH*-Paketes
- 10s Software-Timer:
  - Temperatur messen
  - MQTT-PUBLISH Paket erstellen
  - Mit Raspi per TCP verbinden und Paket über WiFi verschicken

Projekt

Raspberry Pi

Demo