KOMUNIKAČNÍ PROTOKOL MQTT

Author Name (Tomáš Mičulka)

Faculty of Mechanical Engineering, Brno University of Technology
Institute of Automation and Computer Science
Technicka 2896/2, Brno 616 69, Czech Republic
Tomas.Miculka@vutbr.cz

Abstract: *Tato seminární práce řeší komunikační protokol MQTT. Cílem je informovat o jeho struktuře, zabezpečení a systému předávání zpráv. Zároven popisuje důležité paramaetry při navázání spojení a odesílání dat, jako je například QoS,Topic,Retain nebo Return code.*

Keywords: *mqtt, IoT, telecomunication, messaging, TCP, OASIS, broker, payload*

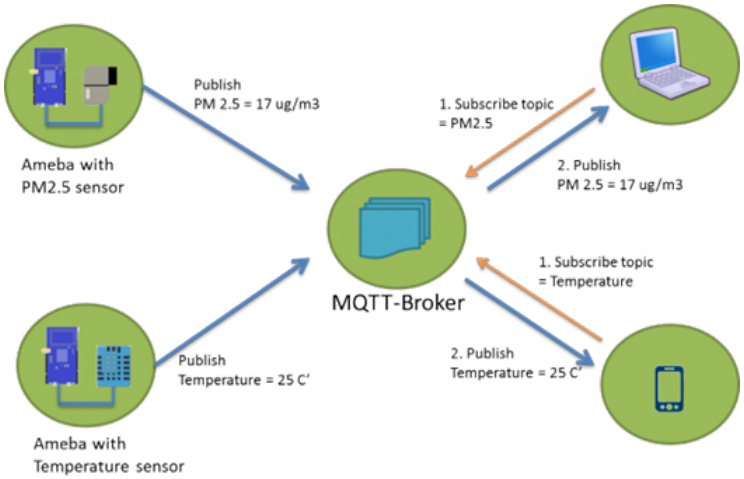
1 Úvod

MQTT (dříve: Message Queuing Telemetry Transport, dnes MQ Telemetry Transport) je jednoduchý a nenáročný protokol pro předávání zpráv mezi klienty prostřednictvím centrálního bodu – brokeru. Díky této nenáročnosti a jednoduchosti je snadno implementovatelný i do zařízení s „malými“ procesory a poměrně rychle se rozšířil. Původně byl navržen v IBM, ale dnes za ním stojí konsorcium "Eclipse foundation". Zároveň v roce 2014 proběhla standardizace OASIS [2, 5].

2 Základní informace o protokolu

Jde o centralizovaný protokol. Základem je systém typu zveřejnit/odebírat (publish/subscribe). Zařízení s funkcí zveřejnit odesílají zprávy zprostředkovateli (broker), který na základě přihlášených odběrů provede třídění a přeposlání správným uživatelům. Předávání je pouze jednosměrné (potvrzované), lze však využít QoS.

Od počátku byl protokol koncipován pro sítě TCP/IP. V současnosti, díky rozvoji nových technologií, je možné nalézt např. implementaci MQTT-SN (MQTT for Sensor Networks), nebo MQTT-WS (WebSocket). Důležitým předpokladem je přenos dat v blocích, nelze tedy použít datový proud (streamování).[5]



Obrázek 1: Princip centrálního bodu

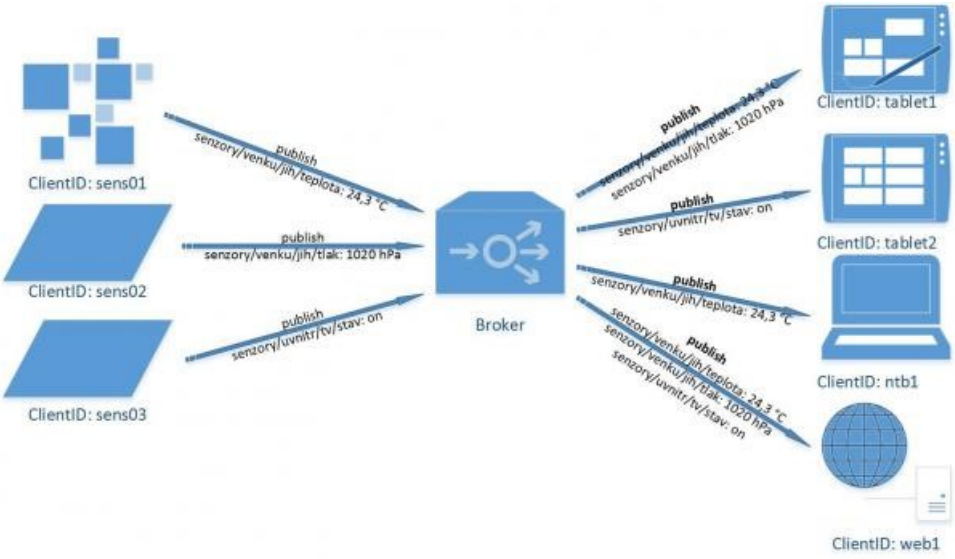
2.1 Topics (Témata)

Zprávy jsou zveřejňovány pod tématy (topic), které mají hierarchickou strukturu, a můžeme si je nezávisle definovat. Jednotlivé úrovně hierarchie jsou oddělovány znakem "/“. Příkladem může být téma obsahující informaci o venkovní teplotě:
`senzory/venku/jih/teplota 24,3 °C`

Při identifikaci tématu je možné použít zástupných znaků "+“ nebo "#“. Znak "+“ slouží jako zástupný pro celou jednu úroveň, např. senzory/+/+/teplota slouží pro odebírání dat ze všech teplotních senzorů. Znak "#“ lze využít jako zástupný znak pro všechny následující úrovně, tedy senzory/venku/# zaregistruje k odebírání všech dat z venkovních senzorů. Nelze jej však, na rozdíl od znaku "+“, umístit uprostřed téma.[9, 10]

Témata jsou v MQTT reprezentována v podobě řetězce v UTF-8 kódování, takže i pojmenování s diakritikou není problém. Hierarchie témat není nijak pevně dána a záleží jen na aplikaci a návrhu programátora. Jak správně navrhnout hierarchii může však být i netriviální úloha. Ne vždy je totiž ta nejsprávnější struktura ta „přirozená“. Důležité je už v rámci těchto úvah rozmyslet datový model a rozhraní – tedy vhodné uspořádání tak, aby bylo možné přihlásit takové topics, které logicky patří k sobě.[9]

Jediné téma, které je rezervované systémem je $SYS. To obsahuje informace o běhu brokeru[10].



Obrázek 2: Nahrávání a stahovaní podle témat [10]

2.2 Infrastruktura

Základem pro použití protokolu MQTT je kromě senzorů/jiných zdrojů dat a „spotřebitele“ také zařízení/aplikace, která se stará o řízení distribuce těchto dat. Jedná se o zprostředkovatele – broker.

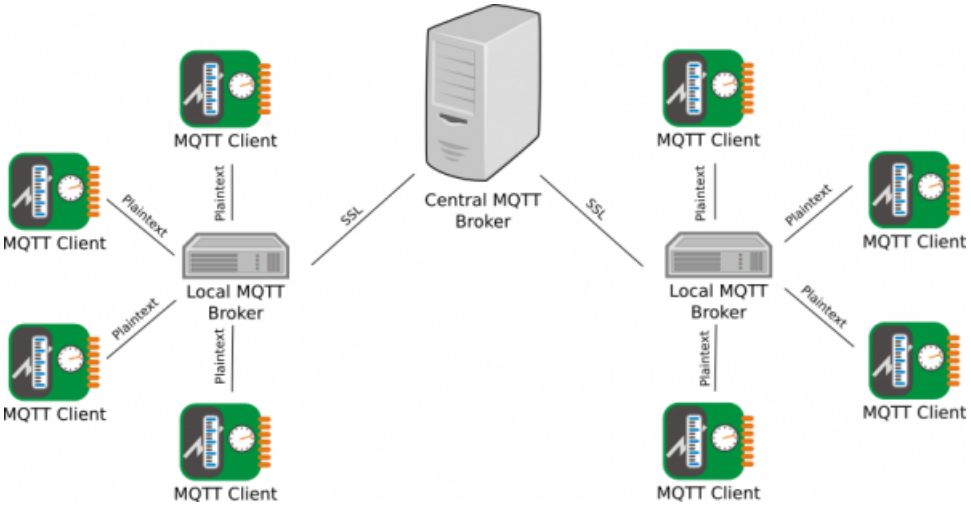
Broker je softwarová aplikace, která řídí autentizaci a autorizaci klientů, obsahuje kryptografické funkce pro zajištění ochrany obsahu před neoprávněným odposlechem a manipulací, a jak bylo uvedeno, distribuuje data na základě přihlášení k odběru jednotlivých témat. Kromě těchto základních funkcí umožňuje komunikaci s dalšími brokery, díky čemuž můžeme vytvářet složitější infrastrukturu.

Pokud se podíváme na klienty, kteří mohou spolupracovat s brokerem, existuje mnoho MQTT knihoven pro různé programovací jazyky. Mezi ty základní patří C, Perl, Python, Ruby, Java, Go, JavaScript, .NET, atd. Pro MQTT je zásadě důležitý pouze TCP/IP stack. Pro zařízení bez TCP/IP můžeme použít implementaci MQTT-SN.

2.3 Zabezpečení

V přihlašovací sekvenci se využívá identifikace každého klienta pomocí "ClientID"a pak volitelně i pomocí uživatelské jména "Username"a hesla "Password". Pokud nám to dovolují možnosti klienta, je vhodné nastavit "ClientID"jako jednoduchý řetězec, který jej co nejlépe identifikuje. Dále MQTT díky podpoře SSL/TLS umožňuje také přihlášení pomocí klientského SSL certifikátu. Podprovány jsou protokoly TLS v1.2, v1.1 nebo v1.0 s x509 certifikáty. Je důležité si uvědomit, že protokol MQTT je čistě textový, a proto bez použití SSL/TLS bude komunikace zcela nešifrovaná (nebezpečí se týká hlavně přenášeného hesla). Jistou možností je pak i využití vlastních klíčů (pre shared key – PSK), které slouží pro zašifrování komunikace jednotlivých klientů na základě jejich identit[10, 5].
Podle požadované úrovně šifrování komunikace pak MQTT protokol předepisuje následující TCP kanály[10, 5]:

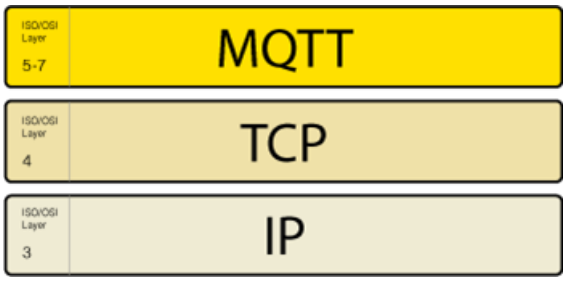
- 1883 = MQTT nešifrovaný přenos (unencrypted) - základní a nejběžnější MQTT komunikační kanál, kde komunikace je zcela nešifrovaná (nebezpečí se týká hlavně přenášeného hesla). Po tomto kanálu by tak neměla zasílat žádná citlivá data.
- 8883 = MQTT šifrovaný přenos (encrypted) - narozdíl od kanálu 1883 jsou data zde šifrována SSL/TLS protokolem a navázání komunikace tak vyžaduje podporu klienta.
- 8884 = MQTT šifrovaný přenos (encrypted) + certifikát klienta (client certificate) - toto je speciální a nejvyšší úroveň zajištění MQTT komunikace, protože nejen, že jsou data opět šifrována protokolem SSL/TLS, ale klient musí poskytnout i certifikát o autenticitě vydávaný brokerem. Tento kanál však zatím podporuje jen málo veřejných brokerů (např. Mosquitto - server "test.mosquitto.org").



Obrázek 3: Zabezpečeni komunikace v LAN a WAN [10]

2.4 Přenosový model

Protokol MQTT sám o sobě popisuje jen samotný popis struktury přenášených zpráv, ale nedefinuje způsob přenosu. K tomu využívá běžný TCP/IP protokol, tedy prakticky využívá běžnou lokální LAN ethernetovou i globální WAN internetovou síť. MQTT protokol tvoří tak pouze tzv. aplikační hladinu OSI modelu. Protože MQTT protokol má velmi jednoduchou strukturu a využívá běžné ethernetové komunikační rozhraní, je snadno implementovatelný i do zařízení s „malými“ procesory a rychle se rozšiřuje.[1]



Obrázek 4: Přenosový model [1]

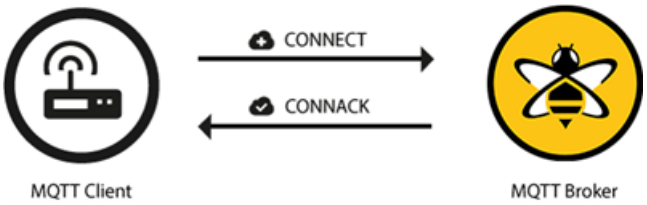
3 Výměna zpráv

3.1 Připojení klienta

Na počátku navazuje klient – (zařízení, node) spojení s brokerem pomocí TCP. Používá se nejčastěji port 1883, pro spojení s TLS pak 8883. Další používaná možnost je připojení přes WebSockets (ws / wss), nejčastěji na portech 8080 / 8081 (či přes reverzní proxy na portech 80 a 443), ale samozřejmě je možné nastavit si komunikaci jakkoli.

Po navázání spojení posílá zařízení zprávu CONNECT, nejčastěji s příznakem clean session, který zajistí, že se začíná s „čistým stolem“, žádná témata nejsou přihlášena k odběru. Připojení může probíhat i s nějakým základním ověřením jménem a heslem. Broker odpoví zprávou CONNACK, čímž potvrdí připojení.[8]

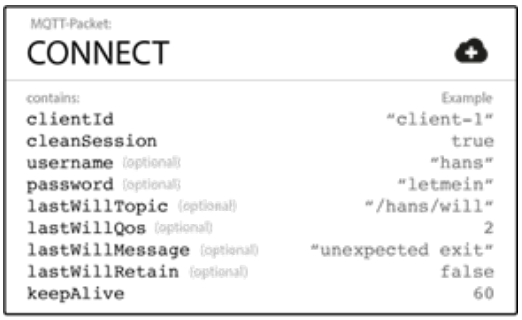
Nyní může následovat jedna či více zpráv SUBSCRIBE s názvy témat, které chce zařízení odebírat. Podrobnosti viz výše. Broker potvrzuje pomocí SUBACK. Samozřejmě kdykoli po připojení a potvrzení (CONNACK) může zařízení přihlásit odběr dalších témat kdykoli.[8]



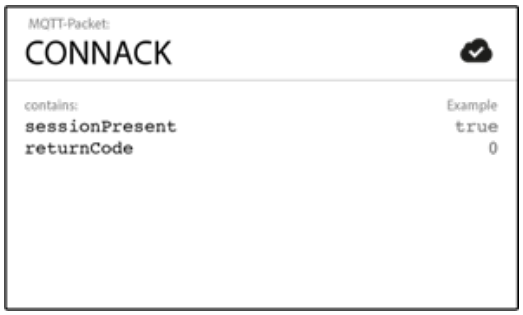
Obrázek 5: Přenosový model [1]

Od okamžiku připojení a potvrzení (CONNACK) je vše nastavené a jak zařízení, tak broker mohou posílat zprávy pomocí PUBLISH.[8]

Zařízení se může odhlásit od odběru určitého tématu pomocí zprávy UNSUBSCRIBE (broker potvrdí provedení zprávou UNSUBACK), a při ukončení práce posílá zprávu DISCONNECT.[1, 8]



(a) Connect



(b) Conncat

Obrázek 6: Parametry zpráv při navázání spojení [1]

3.2 Další parametry při připojeni klienta

Clean Session – určuje, zda má být navázané spojení mezi klientem a brokerem trvalé. V případě nastavení na false, budou všechny zprávy s QoS 1 a 2, které nebyly úspěšně doručeny od brokera klientovi, uloženy a po dalším navázání spojení předány. Pokud je nastaveno na true, broker žádné zprávy pro klienta neukládá.[1, 3]

Last Will – jedná se o trojici volitelných atributů Last Will Topic, Last Will QoS a Last Will Message, které definují téma, zprávu a související QoS (1 nebo 2) pro případ, že dojde k neočekávanému odpojení klienta. To může být z důsledku poruchy sítě, problémy na brokeru nebo při vypršení doby nastavené atributem Keep Alive. Pokud se taková situace stane, na brokeru se objeví definovaná zpráva. Tato vlastnost se často využívá pro zobrazení stavu klienta. [1, 3]

Keep Alive – definuje časový interval, kdy si klient a broker vymění informaci o svém stavu pomocí paketů PINGREQ a PINGRESP.[1, 3]

Return Code – tento příznak obsahuje návratový kód, který klientovi sděluje, zda byl pokus o připojení úspěšný nebo ne. [1, 3]

Tabulka 1: tabulka RC (ReturnCode) [1]

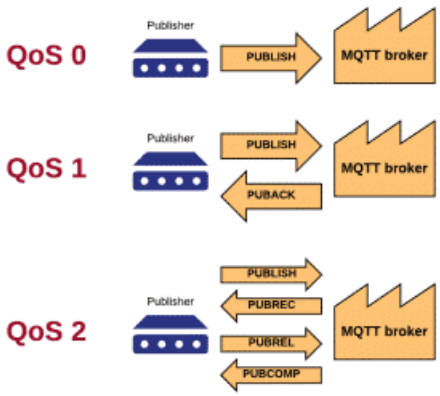
code	RC response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, indentifer rejected
3	Connection refused, server unavalible
4	Connection refused, bad user name or password
5	Connection refused, not authorized

3.3 QoS

V souvislosti se samotným přenosem zpráv pak MQTT protokol definuje tři úrovně potvrzování zpráv QoS (Quality of Service):

- V úrovni 0 (at most once, též fire and forget) pošle publisher zprávu PUBLISH brokeru a dál se o nic nestará. Broker ji stejným způsobem pošle subscriberům daného tématu.[4, 8]
- Na úrovni 1 (at least once) posílá publisher brokeru zprávu PUBLISH a čeká. Broker zprávu přijme a pošle ji odběratelům (opět PUBLISH). Odeslání proběhne buď s QoS 1, nebo s QoS 0, pokud zařízení QoS1 neumí. Ale obecně platí, že se odesílá s takovým QoS, s jakým byla přijata. V tomto případě pošle s QoS1 (PUBLISH) a čeká na potvrzení od příjemce. Jakmile odběratelé potvrdí přijetí zprávou PUBACK, broker zprávu odstraní a pošle PUBACK publisherovi zpět. Publisher ví, že zpráva prošla brokerem, a může ji zahodit. Broker může poslat PUBACK, aniž by měl potvrzení od všech příjemců. Přesné chování závisí na implementaci, většina to tak dělá a MQTT umožňuje oba scénáře, tj. čekat i nečekat.[4, 8]
- Na úrovni 2 (exactly once) posílá publisher brokeru zprávu PUBLISH. Ten ji, stejně jako v předchozím případě, přijme, posílá ji odběratelům, a publisherovi vrátí zprávu PUBREC (tedy potvrzení přijetí). Publisher odpoví zprávou PUBREL, broker zprávu smaže a potvrdí zprávou PUBCOMP. Tím je výměna uzavřena.[4, 8]

Pro QoS platí, že broker odesílá zprávu na té úrovni, na které zpráva přišla, s možností snížit úroveň, pokud klient umí pouze nižší.[4]



Obrázek 7: Type Qos [8]

3.4 Retain

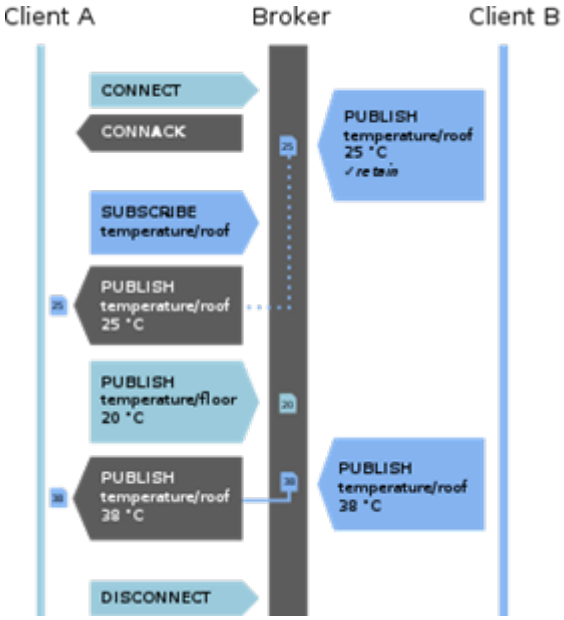
Kromě QoS se u zprávy nastavuje i retain flag, tj. příznak, který říká, že broker nemá zprávu zahazovat po rozeslání, ale uložit a poslat novým odběratelům daného topicu. Posílá se vždy poslední uložená zpráva s příznakem retain[8, 7]

3.5 Publish a Subscribe

Publish:
Pokud je klient úspěšně připojen k brokeru, může začít odesílat zprávy. Každá zpráva musí obsahovat název tématu, definici QoS (0–2), vlastní obsah zprávy a příznak uchování Retain Flag. Ten se používá v případě, že chceme na brokeru ponechat poslední zprávu pro nově připojené klienty. Zprávy jsou dle objednaných témat odesílány až po jejich opětovném příjmu brokerem. V tomto případě nový klient dostane právě tu aktuálně poslední.[6, 5]

Subscribe:
Každý odběr tématu musí být na brokeru správně zaregistrován. Při registraci je možné využít obou, již popsaných, zástupných znaků. V případě, že chceme u některých témat nastavit vlastní QoS, můžeme tak učinit. Broker následně bere v úvahu nastavení vyšší priority.

Po úspěšné registraci jednoho nebo více téma je, na základě nastavení QoS a případném nalezení v paměti brokera, přes funkci zveřejnění (subscribe) zaslán obsah klientovi.[6, 5]



Obrázek 8: Průbeh zprávy na brokeru

3.6 Nastavení relace

Pokud je spojení mezi klientem a zprostředkovatelem přerušeno během relace, kde není povoleno persistence session, dojde ke ztrátě těchto témat a při opětovném připojení se klient musí znovu přihlásit k odběru. Opětovné přihlášení k odběru při každém přerušení připojení je zátěží klienty s omezenými prostředky. Abychom se tomuto problému vyhnuli, může klient při připojení k brokeru požádat o persistent session (trvalou relaci). Trvalé relace ukládají na brokeru všechny informace, které jsou pro klienta relevantní. ClientId, které klient poskytne, které klient poskytne při navázání spojení identifikuje relaci.[3]

Co je uleženo v trvalé relaci:
V trvalé relaci broker ukládá následující informace (i když je klient offline). Když se klient znovu připojí, informace jsou okamžitě k dispozici

- Existence relace (i když neexistují žádné subscriptions)
- Všechny subscription klienta
- Všechny zprávy v rámci QoS 1-2, které klient dosud nepotvrdil
- Všechny nové zprávy (QoS1-2), které klient propásl během toho co byl offline
- Všechny přijaté QoS 2 zprávy od klienta, které nebyly doposud kompletně potvrzeny

Jak spustit nebo ukončit trvalou relaci
Když se client připojí k brokeru, může zažádat o trvalou (persistent) relaci, použitím příznaku CleanSession:

- Když je CleanSession nastaveno na true, klient nechce trvalou relaci. Při jakémkoliv odpojení klienta jsou veškeré informace a zprávy smazány
- Když je CleanSession nastaveno na false, broker vytvoří trvlou relaci pro klienta. Všechny informace a zprávy jsou zachovány dokud klient nezažádá o čisto relaci (clean session).

Podobně jako u brokeru každý klient musí ukládat trvalou relaci. Pokud klient zažádá server o uchování dat relace, je klient odpovědný za ukládání následujících informaci:

- Všechny QoS 1-2 zprávy, které broker zatím nepotvrdil
- Všechny Qos 2 zprávy přijaté od brokeru, ale doposud nebyly kompletně potvrzeny

Reference

[1] Client, broker / server and connection establishment - mqtt essentials: Part 3, © 202.
[2] Mqtt - the standard for iot messaging, ©2020.
[3] Persistent session and queuing messages - mqtt essentials: Part 7, ©2021.
[4] Quality of service 0,1 & 2 - mqtt essentials: Part 6, ©2021.
[5] Banks, A., and Gupta, R. Mqtt version 3.1.1 plus errata 01, © 2015.
[6] Cope, S. Mqtt publish and subscribe beginners guide, ©2021.
[7] Cope, S. Mqtt retained messages explained, ©2021.
[8] Malý, M. Protokol mqtt: komunikační standard pro iot, ©2021.
[9] Right, R. Mqtt — mq telemetry transport, 2021.
[10] Vojáček, A. Iot mqtt prakticky v automatizaci - 1.díl - úvod, 2014.