Komparativna analiza IoT protokola

Sertić, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

KOMPARATIVNA ANALIZA IoT PROTOKOLA

Diplomski rad

Marko Sertić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 06.09.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Marko Sertić
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1269, 06.10.2019.
OIB studenta:	82231423228
Mentor:	Prof.dr.sc. Drago Žagar
Sumentor:	
Sumentor iz tvrtke:	Goran Kopčak
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Grgić
Član Povjerenstva 1:	Prof.dr.sc. Drago Žagar
Član Povjerenstva 2:	Mr.sc. Anđelko Lišnjić
Naslov diplomskog rada:	Komparativna analiza IoT protokola
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	S razvojem Internet of things tehnologije standardiziralo se nekoliko protokola koji se koriste u svrhu ostvarivanja komunikacije između različitih entiteta unutar mrežne infrastrukture. Zadatak ovog diplomskog rada je napraviti komparativnu analizu najpoznatijih protokola (npr. MQTT, CoAP, AMQP) te usporediti njihovu upotrebu u sklopu postojeće IoT platforme za scenarij praćenja mjerenja udaljenih senzora. Kroz različite slučajeve korištenja unutar ove domene potrebno je definirati prednosti i nedostatke upotrebe svakog od protokola unutar određenog korisničkog scenarija. Sumentor iz tvrtke Ericsson: Goran Kopčak
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	06.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.09.2021.

Ime i prezime studenta:	Marko Sertić
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1269, 06.10.2019.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom: **Komparativna analiza IoT protokola**

izrađen pod vodstvom mentora Prof.dr.sc. Drago Žagar

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja Marko Sertić, OIB: 82231423228, student/ica Fakulteta elektrotehnike,
računarstva i informacijskih tehnologija Osijek na studiju Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika', kao autor/ica ocjenskog rada pod naslovom: Komparativna analiza IoT protokola,
dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

Osijek, 13.09.2021. _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

U posljednjih 30 godina iznimno se povećala potreba za automatizacijom uređaja i automatiziranim načinom praćenja uređaja. Glavni cilj tehnološkog napretka je razvoj društva i gospodarstva te poboljšanje načina življenja. Stvaranjem i ugradnjom velikog broja senzora kojima je moguće slati različite telemetrijske podatke dobiva se automatizirani sustav praćenja. Povezivanjem objekata poput termostata, termometra, električnih žarulja, kućanskih aparata i slično, koji tradicionalno nisu povezani na Internet, dobivamo novu razinu inteligentnih sustava. Praćenjem i upravljanjem tim objektima u stvarnom vremenu dobivamo nove mogućnosti za poboljšanje životnih standarda. Ti senzori pripadaju internetu stvari (engl. *Internet of Things, IoT*) i povezani su na internet ili neku drugu mrežu. IoT opisuje tehnologiju povezivanja fizičkih uređaja, vozila i drugih stvari, koje prikupljaju, dijele i razmjenjuju podatke putem interneta. IoT omogućava povezivanje različitih stvari, uređaja iz svakodnevnog života u inteligentne informacijske mreže za prikupljanje i analiziranje informacija. Arhitektura IoT sustava se može podijeliti u tri cjeline, uređaje, rubne pristupnike i oblak. Uređaji poput senzora koriste neke od aplikacijskih protokola kako bi bili povezani s rubnim pristupnikom i tako prenosili prikupljene podatke. Uz pomoć ne kompliciranog računalstva, oblaka, mobilnih tehnologija i analitike, fizičke stvari mogu prikupljati i dijeliti podatke uz minimalnu pomoć čovjeka. Povoljni i pouzdani senzori čine IoT tehnologiju dostupnu svima. Posebni internetski protokoli olakšavaju povezivanje senzora s drugim uređajima i oblakom uz učinkovit prijenos podataka i minimalno zauzeće širine mobilnog pojasa. Tehnologija Interneta stvari se može primijeniti u raznim domenama našeg svakodnevnog života, a jedna od njih je i praćenje senzora na daljinu što je detaljnije objašnjeno u ovom radu.

Na početku diplomskog rada napravljen je uvod u internet stvari i dan je uvid u fizički i logički dizajn IoT sustava. Nakon toga su detaljno opisani MQTT, CoAP i AMQP protokoli, tri najpoznatija transportna protokola za prijenos podataka u internetu stvari. Detaljno su opisani struktura poruke koja se šalje tim protokolima i najčešći slučajevi primjene. Nakon toga se na temelju njihovih značajki analizirani i uspoređeni navedeni protokoli i prikazane su prednosti i nedostaci pojedinih protokola. U poglavlju slučajeva korištenja IoT protokola je za svaki protokol testiran korisnički slučaj te je implementiran na platformu za praćenje mjerenja udaljenih senzora.

1.1. Zadatak diplomskog rada

S razvojem tehnologije Interneta stvari (engl.*Internet of things*) standardiziralo se nekoliko protokola koji se koriste u svrhu ostvarivanja komunikacije između različitih entiteta unutar mrežne infrastrukture. Zadatak ovog diplomskog rada je napraviti komparativnu analizu najpoznatijih protokola (npr. MQTT, CoAP, AMQP) te usporediti njihovu upotrebu u sklopu postojeće IoT platforme za scenarij praćenja mjerenja udaljenih senzora. Kroz različite slučajeve korištenja unutar ove domene potrebno je definirati prednosti i nedostatke upotrebe svakog od protokola unutar određenog korisničkog scenarija.

2. UVOD U INTERNET STVARI

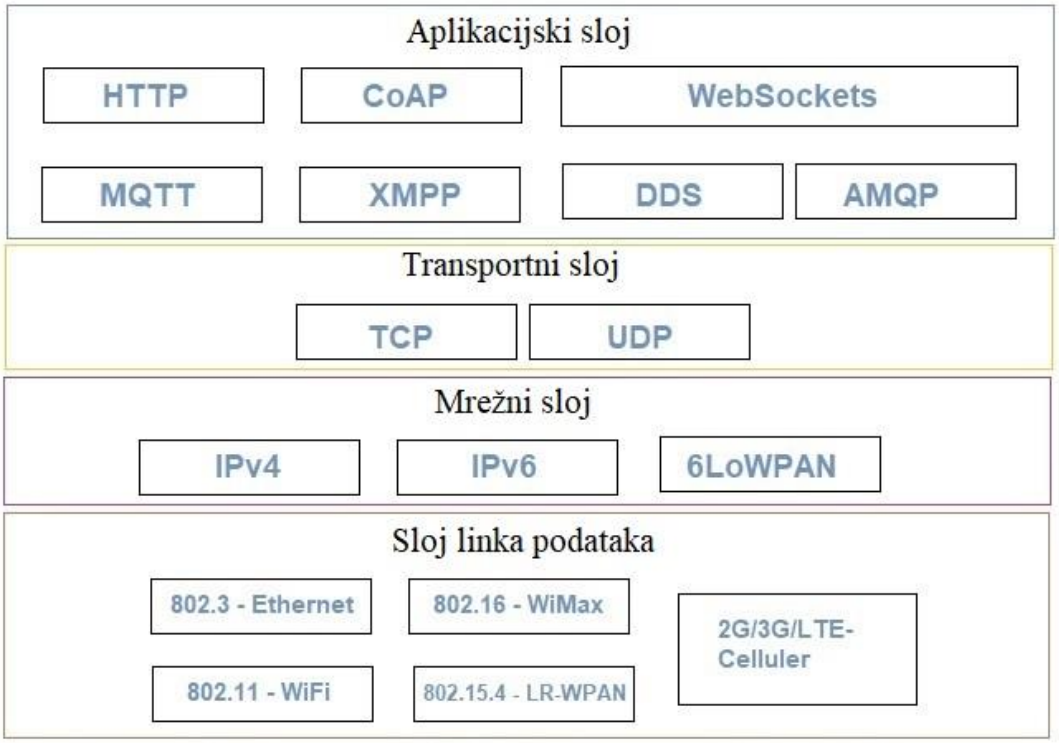
Internet stvari obuhvaća objekte koji imaju jedinstveni identitet i koji su povezani na Internet, žično ili bežično. Dok je u današnje vrijeme većina uređaja poput računala i mobilnih uređaja već povezana na internet, Internet stvari se više fokusira na konfiguriranje, kontroliranje i praćenje uređaja pomoću interneta koji po svojoj prirodi nisu povezani na internet kao što su termostati, brojila za plin i vodu te razni drugi senzori [1]. Prema literaturi iz [1], Internet stvari se može definirati kao dinamička globalna mrežna infrastruktura s mogućnosti samokonfiguriranja koja se temelji na standardiziranim interoperabilnim komunikacijskim protokolima gdje fizičke i virtualne „stvari“ imaju identitete, fizičke atribute, virtualne osobnosti, koriste inteligentna sučelja i integrirani su u informacijsku mrežu putem interneta gdje razmjenjuju podatke vezane za korisnike i njihovu okolinu.

Glavne karakteristike koje opisuju Internet stvari su:

- **Dinamičnost i adaptivnost**: IoT uređaji i sustavi imaju mogućnost da se dinamički prilagode i promjene ovisno o uvjetima i kontekstu u kojima se nalaze. Mijenjaju se prema unaprijed određenim pravilima ili uvjetima u kojima se trenutno nalaze.
- **Mogućnost samokonfiguriranja**: IoT uređaji imaju sposobnost samokonfiguriranja tako što omogućavaju da veliki broj uređaja radi zajedno kako bi se izvršila određena funkcionalnost. To uključuje automatsko mijenjanje funkcionalnosti koju uređaj obavlja ili automatsko ažuriranje sustava bez potrebe za čovjekovom intervencijom.
- **Interoperabilni komunikacijski protokoli**: IoT uređaji podržavaju veliki broj komunikacijskih protokola za međusobnu komunikaciju uređaja i infrastrukture.
- **Jedinstveni identiteti**: Svaki IoT uređaj ima jedinstveni identifikacijski broj, najčešće IP adresa uređaja ili URI (engl. *Uniform Resource Identifier*). Na taj način čovjek ili inteligentni sustavi mogu pristupiti određenom uređaju te pratiti njegov rad ili promijeniti funkcionalnost uređaja.
- **Integriranost u informacijsku mrežu**: IoT uređaji su najčešće integrirani u informacijsku mrežu koja im omogućava da međusobno komuniciraju i razmjenjuju podatke. Uređaji imaju sposobnost da ih mreža automatski prepozna kada se priključe na nju i mogućnost da mreži i uređajima u mreži daju informaciju o tome kakvi su uređaji, koja im je zadaća, koja im je funkcionalnost i slično.

2.1. FIZIČKI DIZAJN

IoT uređaje, sučelja i načine povezivanja definira njihov fizički dizajn infrastrukture kao i protokole TCP/IP mrežnog modela komunikacije kao što je prikazano na slici 2.1. IoT protokoli pomažu uspostaviti komunikaciju između IoT uređaja i IoT oblaka putem interneta.



Slika 2.1. TCP/IP model IoT protokola [31].

Sloj linka podataka određuje na koji način će se slati podaci unutar mreže na fizičkom sloju. Određuje na koji način će paketi biti kodirani i poslani s fizičkog dijela uređaja na medij kojim se šalju. Najčešći mediji za slanje podataka na fizičkom sloju koaksijalni kabel i radio valovi. Protokoli koji se nalaze u sloju linka podataka, a važni su za koncept IoT tehnologije su **802.3 Ethernet, 802.11 WiFi, 802.16 WiMax, 802.15.4 LR-WPAN i 2G/3G/4G Mobilne komunikacijske mreže.**

802.3 Ethernet se sastoji primarno sastoji od protokola i tehnologija koje se koriste u LAN-ovima (engl. *Local Area Network*). Definira fizički sloj i podsloj sloja podatkovne veze za kontrolu pristupa MAC sloju. Klasificiran je u dvije kategorije, klasičan Ethernet i komutirani Ethernet.

802.11 WiFi je dio 802.11 grupe LAN protokola koji je specificira MAC i fizičke protokole za implementaciju bežične komunikacije u WLAN-u (engl. *Wireless Local Area Network*). Komunikacija se odvija na frekvencijama od 2.4 GHz, 5 GHz i 60 GHz.

802.16 WiMax je standard za WMAN (engl. *Wireless Metropolitan Area Network*) mreže koje su dizajnirane za širokopojasni bežični pristup tipa točka na više točaka (engl. *point to multipoint*).

802.15.4 LR-WPAN grupa standarda za nisko potrošnu osobnu bežičnu mrežu (engl. *Low-Rate Wireless Personal Area Network*). Ovaj standard definira protokole kao što su Zigbee, 6LoWPAN, Thread WiSUN i MiWi protokole. Standard pruža komunikaciju pri malim brzinama i malom energetskom potrošnjom za uređaje ograničenih resursa.

2G/3G/4G Mobilne komunikacijske mreže su preže koje se koriste u mobilnim komunikacijama. IoT uređaji koji koriste protokole mobilnih mreža za međusobno komuniciranje mogu koristiti ćelijske mreže za ostvarivanje komunikacije.

Mrežni sloj je odgovoran za slanje IP (engl. *Internet Protocol*) datagrama od izvorišta do odredišta u mreži. Funkcija mrežnog sloja je adresiranje uređaja odnosno dodjeljivanje IP adresa i preusmjeravanje IP paketa. U ovom sloju se za IoT tehnologiju najčešće koriste IPv4, IPv6 i 6LoWPAN protokoli. Kod IPv4 protokola se za identifikaciju adresa koristi 32 bita dok se kod IPv6 protokola koriste 128- bitne adrese. 6LoWPAN (engl. *IPv6 over Low-Power Wireless Personal Area Network*) protokol je protokol temeljen na IP protokolu, ali se koristi za uređaje s ograničenom procesorskom moći [31].

Transportni sloj zadužen je za prijenos paketa s kraja na kraj. Funkcionalnosti koje pruža transportni sloj su kontrola pogreške, kontrola toka te kontrola zagušenja. Za IoT tehnologiju se iz transportnog sloja koriste TCP i UDP protokoli.

TCP (engl. *Transmission Control Protocol*) je konekcijski orijentirani protokol što znači da je zadužen za pouzdani prijenos paketa, odnosno osigurava njihovu dostavu. Protokol pruža detekciju pogrešaka što znači da može detektirati duplicirano poslane pakete i riješiti problem tako da samo jedan paket stigne na odredište. Njegova zadaća je da osigura odgovarajuću brzinu prijenosa paketa tako da odredišna strana ima dovoljno vremena za obraditi sve pristigle pakete.

UDP (engl. *User Datagram Protocol*) protokol je protokol koji radi na principu pošalji i zaboravi (engl. *Fire and Forget*). On je transakcijski orijentiran protokol i za razliku od TCP-a ne osigurava dostavu paketa već je njegova glavna karakteristika slanje manjih paketa zbog uštede vremena i povećanja brzine prijenosa.

Aplikacijski sloj ne pruža usluge drugim slojevima OSI modela već samo aplikacijama van OSI modela. Sloj definira sučelja za slanje podataka s nižih slojeva prema aplikacijskom i dalje prema korisniku kroz internet. Najčešće korišteni protokoli za IoT tehnologiju na ovom sloju su HTTP, MQTT, CoAP, XMPP, WebSockets, DDS i AMQP [30].

HTTP (engl. *Hypertext Transfer Protocol*) protokol je temeljni protokol svjetske mreže (engl. *World Wide Web,* WWW). Protokol se temelji na zahtjev/ odgovor modelu rada gdje šalje zahtjeve serveru koristeći HTTP naredbe. Naredbe koje HTTP koristi su GET, PUT, POST, DELETE, HEAD, TRACE, OPTIONS, itd. Zahtjevi kod HTTP protokola međusobno su neovisni pa tako klijent može biti mrežni pretraživač ili bilo koja aplikacija kod IoT uređaja koja koristi HTTP protokol.

WebSocket protokol je TCP orijentirani protokol koji omogućava potpunu dvosmjernu (engl. *full-duplex*) komunikaciju klijenta i servera. To postiže na način da jednu TCP vezu između klijenta i servera drži otvorenu i tako pruža komunikaciju u oba smjera.

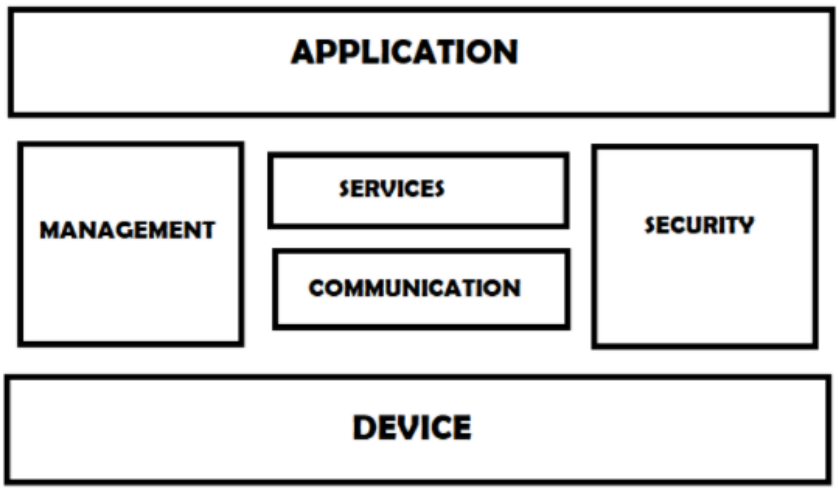
XMPP (engl. *Extensible Messaging and Presence Protocol*) protokol je aplikacijski protokol koji se koristi u aplikacijama gdje je bitno da se podaci prenose velikom brzinom odnosno u stvarnom vremenu. Aplikacije poput video poziva, telefonskih razgovora preko interneta, telekonferencije ili *online* igre šalju i dijele XML podatke između internetskih entiteta. U kontekstu IoT tehnologije, XMPP protokol pruža komunikaciju u stvarnom vremenu između uređaja i senzora u mreži.

Protokoli MQTT, CoAP i AMQP su detaljno opisani u nastavku rada.

2.2. LOGIČKI DIZAJN

Logički dizajn IoT sustava reprezentiraju entiteti i procesi koji sačinjavaju IoT logičku mrežu. IoT logička mreža se može podijeliti na 6 funkcionalnih blokova koji opisuju sva ponašanja u mreži. Blokovi opisuju funkcionalnosti poput identifikacije, komunikacije, dohvaćanja vanjskih podražaja, menadžmenta i slično. Slika 2.2. prikazuje strukturu funkcijskih blokova koji se koriste u IoT sustavima.

- **Uređaj:** dio IoT mreže koji je zadužen za osluškivanje i dohvaćanje podražaja iz okoline. Provode kontrolne funkcije, funkcije nadziranja i aktiviranja.
- **Komunikacija:** komunikacijski funkcijski blok je zadužen i opisuje komunikaciju u IoT sustavima.
- **Servisi:** servisi poput nadziranja, kontroliranja uređaja, objavljivanja podataka se koriste u IoT sustavima.
- **Menadžment:** funkcijski blok menadžmenta pruža funkcije za upravljanje IoT sustavima.
- **Sigurnost:** sigurnosni funkcijski blok pruža usluge poput autentifikacije, autorizacije i održava integritet podataka koji se šalju.
- **Aplikacija:** funkcijski blok aplikacija pruža sučelje korisniku s kojim korisnik može upravljati s prethodno navedenim blokovima. Aplikacija pruža uvid u prikupljene i obrađene podatke.



Slika 2.2. Logički dizajn IoT sustava [31].

3. MQTT PROTOKOL

Protokol prijenosa telemetrijskih poruka u redu čekanja (engl. *Message Queuing Telemetry Transport*, MQTT) je internetski protokol koji se koristi za prijenos poruka između krajnjih uređaja. Napravljen je tako da radi na principu objavi/pretplati se načinu rada. Koristi se za komunikaciju između krajnjih uređaja (M2M – engl. *Machine to machine*) te je jedan od najčešće korištenih protokola za takav način komunikacije. Podržava ISO i OASIS standarde. Protokol se nalazi na aplikacijskom sloju te se oslanja na TCP protokol u transportnom sloju te IP protokol u mrežnom sloju. U tablici 2.1. su prikazani protokoli na koje se oslanja MQTT protokol te na kojim se ISO/OSI slojevima nalaze [3].

ISO/OSI sloj	Protokol
5 do 7	MQTT
4	TCP
3	IP

Tablica 3.1. *Slojevi MQTT protokola.*

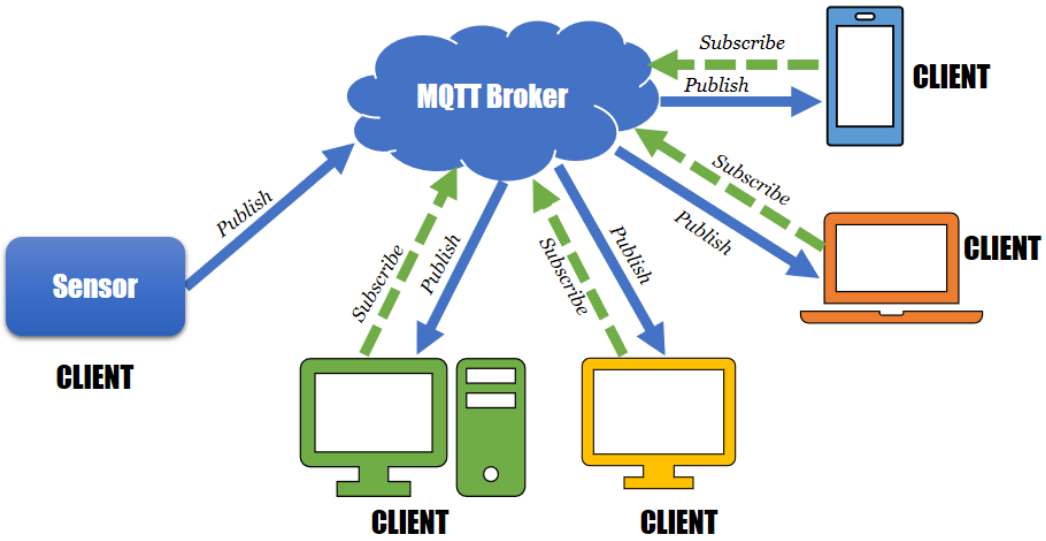
Protokol koristi klijent/server arhitekturu gdje se klijent koji predstavlja neki od uređaja spaja na server koji se zove MQTT Broker te tamo objavljuje poruke na određenu temu. Klijenti su najčešće senzori s ograničenim resursima poput radne memorije ili baterijskog napajanja. MQTT Broker je najčešće server spojen na internetsku mrežu koji upravlja podacima poslanima od strane senzora. Klijenti nisu nikada međusobno povezani i nemaju informacije o broju klijenata u mreži.

Akreditacija za uspostavu veze kod MQTT protokola se šalje u obliku običnog otvorenog teksta. Da bi se spriječila zloupotreba tih podataka kao što su prisluškivanje ili izmjena teksta, koristi se TLS protokol za šifriranje i zaštitu [3].

3.1. Klijent

MQTT klijent je svaki uređaj koji koristi MQTT protokol za povezivanje s MQTT brokerom pomoću internetske veze. Postoje dvije vrste klijenata, objavljivač (engl. *publisher*) i

pretplatitelj (engl. *subscriber*). Objavljivač objavljuje informaciju za određenu temu dok pretplatitelj bira na koju temu će biti pretplaćen. Klijent uređaj može u isto vrijeme biti i objavljivač i pretplatitelj. To ovisi koju radnju izvodi u datom trenutku. Klijent može biti server koji prikuplja podatke ili računalo koje grafički prikazuje obrađene podatke, a može biti i senzor za mjerenje temperature ili vlažnosti zraka koji u sebi sadrži mikrokontroler koji je povezan na internetsku mrežu. Da bi neki uređaj bio MQTT klijent, važno je samo da se podaci šalju MQTT protokolom. Na slici 3.1. su prikazani razni uređaji koji mogu predstavljati MQTT klijenta. Neki od njih su samo objavljivači, a neki su i objavljivači i pretplatitelji.



Slika 3.1. *Mogući MQTT klijenti* [2].

3.2. Posrednik

MQTT posrednik (engl. *broker*) je software koji se izvodi na računalu (pokrenut lokalno ili u oblaku), a može ga se samostalno izraditi ili koristiti kao gotov proizvod od drugih poslužitelja. Dostupan je u implementacijama otvorenog koda ili koda u vlasništvu [3].

Posrednik je središte svakog objavi/pretplati se protokola. Postoji mnogo izvedbi posrednika. Jednostavniji mogu od jednom imati 1000 aktivnih veza, dok složeniji mogu imati do 3 milijuna aktivnih veza. Zadaća posrednika je prihvaćanje svih poruka koje su poslane od strane objavljivača. Pristigle poruke posrednik obrađuje te prosljeđuje odgovarajućim pretplatiteljima koji su pretplaćeni na određenu temu. Jedna od zadaća posrednika je autentifikacija i autorizacija klijenta. Posrednik može biti bilo koje računalo čiji su resursi dovoljno snažni za obradu,

prikupljanje i prosljeđivanje poruka klijentima. Također je moguće i unajmiti ili besplatno koristiti web servis (oblak, engl. *cloud*) koji simulira rad posrednika pa je tako moguće testirati mrežna rješenja [4].

Neki od web servisa koji su dostupni su prikazani u tablici 3.2. Tablica prikazuje naziv servisa, podržane portove i web adresu.

Naziv	Portovi	Web adresa
Flespi	1883, 80, 443	mqtt.flespi.io
Mosquitto	1883, 8883, 8884, 8080, 8081	test.mosquitto.org
Verne	1883	mqtt.reserakt.io
Adafruit IO	1883, 8883, 443	adafruit.io
EMQ X	1883, 8883, 8884, 8083, 8084	broker.emqx.io

Tablica 3.2. *Prikaz web MQTT posrednika* [5].

3.3. MQTT paket

MQTT protokol radi na način međusobne izmjene MQTT poruka. Poruke koje se izmjenjuju ne moraju biti točno određene duljine, ali zato struktura poruke mora biti poredana na točno određen način. Takava jedana poruka se sastoji od tri dijela [6]:

- Fiksno zaglavlje
- Varijabilno zaglavlje
- Korisni teret (engl. *payload*)

3.3.1 Fiksno zaglavlje

MQTT paket na svom početku ima fiksno zaglavlje koje je najčešće veličine 2 bajta. Zaglavlje se sastoji od kontrolnog polja, zastavica kontrolnog paketa i preostale duljine paketa. Fiksno zaglavlje se uvijek nalazi u jednom MQTT paketu dok se varijabilno zaglavlje i korisni teret ne moraju nužno nalaziti u paketu. U tablici 3.3. je prikazana struktura fiksnog zaglavlja i način na koji je oblikovan.

Bit	7	6	5	4	3	2	1	0
bajt 1	MQTT kontrolni paket				Specifične zastavice za svaki MQTT kontrolni paket			
bajt 2	Preostala duljina paketa							

Tablica 3.3. *Struktura fiksong zaglavlja* [6].

Kontrolni paket čini prvi bajt fiksnog zaglavlja. Prva četiri najznačajnija bita (engl. *Most Significant Bit*, MSB) prvog bajta predstavljaju određenu naredbu koja se šalje. Popis naredbi i njihovih opisa prikazan je u tablici 2.4.

Naziv	Vrijednost	Smijer kretanja	Opis
Reserved	0	Zabranjeno	Rezervirano
CONNECT	1	Od klijenta prema serveru	Klijent zahtjeva spajanje na server
CONNACK	2	Od servera prema klijentu	Potvrda spajanja
PUBLISH	3	Od klijenta prema serveru ili od servera prema klijentu	Objava poruke
PUBACK	4	Od klijenta prema serveru ili od servera prema klijentu	Potvrda objave
PUBREC	5	Od klijenta prema serveru ili od servera prema klijentu	Objava primljena (osigurana dostava do 1)
PUBREL	6	Od klijenta prema serveru ili od servera prema klijentu	Objava otpuštena (osigurana dostava do 2)
PUBCOMP	7	Od klijenta prema serveru ili od servera prema klijentu	Objava završena (osigurana dostava do 3)
SUBSCRIBE	8	Od klijenta prema serveru	Klijent šanje zahtjev za pretplatu na server

SUBACK	9	Od servera prema klijentu	Pretplata potvrđena
UNSUBSCRIBE	10	Od klijenta prema serveru	Zahtjev za otkazivanje pretplate
UNSUBACK	11	Od servera prema klijentu	Otkazivanje pretplate potvrđeno
PINGREQ	12	Od klijenta prema serveru	PING zahtjev
PINGRESP	13	Od servera prema klijentu	PING odgovor
DISCONNECT	14	Od klijenta prema serveru	Klijent se odjavljuje
Reserved	15	Zabranjeno	Rezervirano

Tablica 3.4. *Tipovi kontrolnih paketa* [6].

Preostali dio prvog bajta fiksnog zaglavlja čine kontrolne zastavice. Jedina kontrolna zastavica koja se koristi u ovom dijelu je PUBLISH zastavica. PUBLISH zastavicu čine bitovi DUP, QoS i RETAIN. Ostale zastavice su označene oznakom „Rezervirano“ što znači da će one biti upotrebljene u nekoj novijoj inačici protokola pa stoga moraju biti spomenute. U tablici 3.5. su prikazane kontrolne zastavice.

Kontrolni paket	**Fiksne zastavice zaglavlja**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
CONNECT	Rezervirano	0	0	0	0
CONNACK	Rezervirano	0	0	0	0
PUBLISH	Korišteno u MQTT 3.1.1	DUP	QoS	QoS	RETAIN
PUBACK	Rezervirano	0	0	0	0
PUBREC	Rezervirano	0	0	0	0
PUBREL	Rezervirano	0	0	1	0
PUBCOMP	Rezervirano	0	0	0	0
SUBSCRIBE	Rezervirano	0	0	1	0
SUBACK	Rezervirano	0	0	0	0
UNSUBSCRIBE	Rezervirano	0	0	1	0
UNSUBACK	Rezervirano	0	0	0	0
PINGREQ	Rezervirano	0	0	0	0
PINGRESP	Rezervirano	0	0	0	0
DISCONNECT	Rezervirano	0	0	0	0

Tablica 3.5. *Kontrolne zastavice.*

DUP bit označava stanje slanja poruke. Ako je DUP bit postavljen na 0, poruka se šalje prvi puta. Ako je DUP bit postavljen na 1, to označava da je poruka poslana barem jednom ranije te da je ovo ponovljeno slanje iste. Sve dok je DUP postavljen na 1, slanje poruke će se ponavljati sve dok klijentu ne stigne PUBACK paket koji označava da je paket stigao primatelju. QoS bit određuje važnost odnosno osiguranje za isporuku poruke. Za važnost poruke koriste se 2 QoS bita. Važnost slanja poruke određena je na 3 načina. Poruka može biti poslana najviše jednom, poruka može biti poslana najmanje jednom i poruka može biti poslana točno jednom. Kombinacijom binarnih brojeva biramo koju važnost želimo.

Važno je da se ne dogodi kombinacija u kojoj su oba bita postavljena na 1. Ako se dogodi da pošiljatelj ili primatelj dobije takvu kombinaciju bitova, on tada zatvara vezu između njih. U tablici 3.6. su prikazane vrijednosti i opisi QoS zastavica [6].

QoS vrijednost	Bit 2	bit 1	Opis
0	0	0	Najviše jednom će se poslati
1	0	1	Najmanje jednom će se poslati
2	1	0	Točno jednom će se poslati
-	1	1	Rezervirano

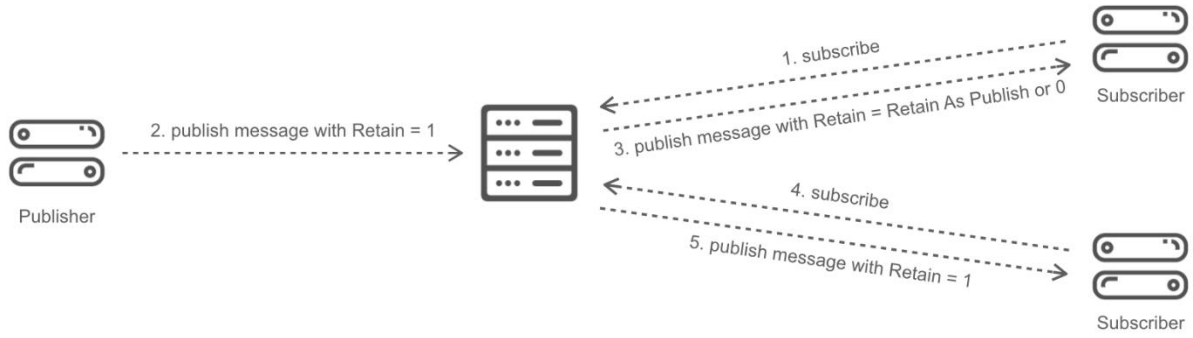
Tablica 3.6. *QoS zastavice.*

RETAIN je zastavica koja zadržava poruku. Ako je zastavica postavljena na 1, u PUBLISH paketu koji je klijent poslao poslužitelju, poslužitelj pohranjuje poruku i oznaku QoS koju ta poruka ima. Na taj se način poruka može proslijediti svim pretplatnicima koji se pretplate na temu čija poruka se čuva. Kada se napravi pretplata na novu temu, pretplatniku se šalje zadnja poruka koja se čuva za svaku temu na koju je prethodno pretplaćen. Ako poslužitelj primi poruku gdje su QoS postavljen na 0 i RETAIN postavljen na 1, odbacuje poruku koju je prethodno čuvao kako bi mogao sačuvati novu poruku s QoS postavljenim na 0. Poslužitelj takvu poruku može odbaciti u bilo kojem trenutku.

Kada se PUBLISH paket šalje klijentu, RETAIN zastavica postavljena na 1 ako je poruka koja je poslana rezultat nove pretplate klijenta. RETAIN zastavica se postavlja na 0 kada posrednik prosljeđuje poruku prethodno pretplaćenim pretplatiteljima. Ako se dogodi slučaj da se novi

korisnik pretplati na temu koja se čuva, posrednik šalje čuvanu poruku u kojoj je RETAIN bio postavljen na 1. Na taj će način novi pretplatitelj znati da je ta poruka bila čuvana.

Slika 3.2. prikazuje slučaj gdje je RETAIN postavljen na 1. Pretplatitelj se pretplaćuje na određenu temu. Posrednik dobiva poruku od objavljivača s RETAIN zastavicom postavljenom na 1. Tada uspoređuje temu pristigle poruke s temama na koje su pretplaćeni pretplatitelji. Budući da pretplatitelj postoji, posrednik prosljeđuje poruku te postavlja RETAIN prema vlastitim postavkama. Budući da je poruka bila s RETAIN = 1 posrednik sprema poruku za nove buduće pretplatitelje. Kada se pojavi novi pretplatitelj na tu temu čija se poruka čuva, posrednik će mu ju proslijediti sa sadržajem poruke i RETAIN = 1.



Slika 3.2. *Tok Publish poruke sa RETAIN = 1* [7].

Ako se dogodi situacija de je poslana poruka bez sadržaja, a ima RETAIN = 1, poslužitelj će takvu poruku poslati svima koji su pretplaćeni na tu temu. Posrednik ovakav paket neće čuvati za buduće pretplatitelje i izbrisat će sve prethodno spremljene poruke na tu temu.

Preostala duljina paketa započinje drugim bajtom fiksnog zaglavlja. Ona može biti dugačka najviše 4 bajta. Kod svakog se bajta 7 bitova koristi za prikazivanje duljine preostalog sadržaja u paketu to jest veličina varijabilnog zaglavlja i tereta. Prvi najznačajniji bit se koristi za upozoravanje je li idući bajt dio ovog paketa. Ako je prvi MSB = 1 tada je idući bajt dio ovog pakata, a ako je 1 MSB = 0 tada je trenutni bajt zadnji bajt u duljini paketa [8].

3.3.2. Varijabilno zaglavlje

Varijabilno zaglavlje smješteno je između fiksnog zaglavlja i korisnog tereta. Sadržaj samog varijabilnog zaglavlja ovisi o tipu paketa koji se šalje. Prvi bajt identifikatora paketa (engl. *Packet Identifier*) počinje od najznačajnijeg bita dok drugi bajt identifikatora paketa počinje od najmanje značajnijeg bita. U tablici 3.3. prikazani su svi tipovi kontrolnih paketa koji mogu označavati identifikator paketa.

Kontrolni paket	**Polje identifikatora paketa**
CONNECT	NE
CONNACK	NE
PUBLISH	DA (Ako je QoS > 0)
PUBACK	DA
PUBREC	DA
PUBREL	DA
PUBCOMP	DA
SUBSCRIBE	DA
SUBACK	DA
UNSUBSCRIBE	DA
UNSUBACK	DA
PINGREQ	NE
PINGRESP	NE
DISCONNECT	NE

Tablica 3.3. *Tipovi identifikatora.*

Jedni od bitnijih kontrolnih paketa su CONNECT, CONNACK te PUBLISH. CONNECT je polje u kojemu se nalazi kodirani naziv protokola koji se koristi. U ovom slučaju to je MQTT. Nakon toga dolaze zastavice koje određuju korisničko ime, lozinku, zadržavanje, QoS, oporuku, čistu sesiju i jedno rezervirano polje. Sve vrijednosti se postavljaju bitom 0 ili 1. Nakon zastavica dolazi polje koje određuje nakon koliko vremena će doći do ponovnog slanja.

CONNACK zaglavlje se koristi kada posrednik komunicira s pošiljateljem. Kada pošiljatelj pošalje zahtjev za povezivanje, a posrednik dobije taj zahtjev, tada on šalje nazad CONNACK odgovor. U tom zaglavlju će se nalaziti povratni kod za povezivanje te na temelju tog koda se može zaključiti jeli veza uspostavljena ili nije. Zaglavlje se sastoji od tri dijela. Zastavice potvrde povezivanja (engl. *Connect Acknowledge Flags*) moraju biti postavljene na 0 te je to polje namijenjeno za buduću upotrebu. Postojanje sesije (engl. *Session Present*) označava jesu li prijemnik i posrednik već imali uspostavljenu vezu. Ako jesu, sesija će biti postavljena na 1, a ako nisu, bit će postavljena na 0. Povratni kod povezivanje (engl. *Connect Return code*) šalje informaciju o tome što je posrednik napravio s CONNECT zahtjevom. U tablici 3.4 nalaze se mogući odgovori i njihovo značenje [6].

Vrijednost	**Povratni kod**	**Opis**
0	0x00	Povezivanje prihvaćeno
1	0x01	Povezivanje odbijeno, neprihvatljiva verzija protokola
2	0x02	Povezivanje odbijeno, identifikator odbačen
3	0x03	Povezivanje odbijeno, server ne dostupan
4	0x04	Povezivanje odbijeno, pogrešno korisničko ime ili lozinka
5	0x05	Povezivanje odbijeno, klijent nema autorizaciju za povezivanje
6-255	Rezervirano	Rezervirano

Tablica 3.4. *Povratni kod povezivanja* [6].

PUBLISH varijabilno zaglavlje se sastoji od 2 polja. Ime teme (engl. *Topick name*) i identifikatora paketa (engl. *Packet Identifier*).

Preostala varijabilna zaglavlja u sebi sadrže samo identifikator paketa dok PINGREQ, PINGRESP i DISCONNECT uopće ne posjeduju varijabilno zaglavlje.

4. CoAP PROTOKOL

CoAP (engl. *Constrained Application Protocol,* ograničeni aplikacijsku protokol) protokol je ograničeni aplikacijski protokol primarno namijenjen za uređaje s ograničenim kapacitetima kao što je definirano u IETF RFC 7252 standardu. Takvim uređajima protokol omogućava da dijele i prenose informacije međusobno. Ti uređaji se zovu čvorovi (engl. *Nodes*). Za ostvarivanje takve komunikacije, čvorovi moraju međusobno biti povezani na internet. Također je moguće ostvariti komunikaciju i s čvorovima iz druge mreže uz uvjet da su i ti čvorovi povezani na internetsku mrežu.

CoAP protokol je dizajniran tako da nesmetano radi u uvjetima gdje je zagušenost mreže velika i gdje je propusnost malena. Da bi to postigao, energetska potrošnja i pouzdanost rada protokola u takvim uvjetima mora biti na visokoj razini. Budući da je CoAP aplikacijski protokol, temelji se na UDP protokolu [9].

Zbog lakše integracije za rad na internetu, CoAP je dizajniran tako da se lako prevede na HTTP protokol s ciljem da zadovoljava zahtjeve kao što su višestruko slanje poruka (engl. *Multicast*), mala energetska potrošnja i jednostavnost [10]. Budući da se radi o M2M (engl. *Machine to Machine*) interakciji čvorova gde se uređaji nalaze u mreži interneta stvari (engl. *Internet of Things,* IoT) i nemaju električno napajanje kao uobičajni uređaji koji su povezani na internetsku mrežu, učinkovitost protokola je jako važna. Također, CoAP može raditi i na nekim uređajima koji su temeljeni i podržavaju TCP protokol [9].

4.1. CoAP Paket

CoAP paket započinje zaglavljem veličine 4 bajta u kojemu se nalaze Verzija, Tip, Duljina tokena, Zatraži/Odgovori kod i ID poruke.Nakon zaglavlja dolazi polje Token koje ne mora imati fiksnu veličinu. Veličina tokena može biti od 0 do 8 bajtova. Nakon tokena dolaze Opcije koje ne moraju nužno biti sadržane u paketu. Nakon Opcija slijedi korisni teret (engl. *Payload*) koji je također opcionalan dio paketa. Na slici 4.1. prikazan je izgled jednog CoAP paketa.

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>4</td><td>32</td><td colspan="2">VER</td><td colspan="2">Type</td><td colspan="4">Token Length</td><td colspan="8">Request/Response Code</td><td colspan="16">Message ID</td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="2">Token (0 - 8 bytes)</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Options (If Available)</td></tr>
<tr><td>20</td><td>160</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="24">Payload (If Available)</td></tr>
</table>

Slika 4.1. *CoAP paket* [9].

Polje Verzija označava o kojoj se verziji protokola radi. Verzija se određuje kombinacijom 2 bita. Ukoliko se koristi protokol prema standardu RFC 7252, verzija mora biti postavljena na kombinaciju „0 1“. Ostale kombinacije su rezervirane za novije buduće verzije. Ukoliko dođe kombinacija različita od „0 1“, poruka će biti ignorirana.

Polje Tip određuje o kojem tipu poruke se radi. Tip poruke određuje se kombinacijom 2 bita.

Razlikuju se četiri vrste poruke:

- 0 0: CON (Confirmable) – ova poruka očekuje odgovarajuću poruku potvrde
- 0 1: NON (Non- confirmalbe) – ova poruka ne očekuje poruku potvrde
- 1 0: ACK (Acknowledgment) – ova poruka je odgovor koji potvrđuje poruku potvrde (CON)
- 1 1: RST (Reset) – ova oznaka ukazuje da je poruka primljena, ali se ne može obraditi

Duljina tokena određuje se s četiri bita. Kombinacija tih bitova označava duljinu polja tokena promjenjive duljine koje može biti 0 – 8 bajtova. Duljine od 9 do 15 su rezervirane, te ukoliko dođe neka od tih kombinacija, poruka se mora označiti kao pogrešna poruka.

Zatraži/Odgovori kod je kod od 8 bitova podijeljen na dva dijela. Prva tri najznačajnija bita čine klasu koja je slična klasi HTTP statusnog koda. Preostali najmanje značajni bitovi tvore kombinacije kodova koje pokazuju detalje zahtjeva ili odgovora u komunikaciji.

Moguće kombinacije koje se mogu dobiti prikazane su u tekstu ispod:

- Method: 0.XX
    1. EMPTY
    2. GET
    3. POST
    4. PUT
    5. DELETE
    6. FETCH
    7. PATCH
    8. iPATCH

- Success: 2.XX
    1. Created
    2. Deleted
    3. Valid
    4. Changed
    5. Content

    31. Continue

- Client Error: 4.XX
    1. Bad Request
    2. Unauthorized
    3. Bad Option
    4. Forbidden
    5. Not Found
    6. Method Not Allowed
    7. Not Acceptable

    8. Request Entity Incomplete
    9. Conflict

    12. Precondition Failed
    13. Request Entity Too Large

    15. Unsupported Content-Format

- Server Error: 5.XX
    1. Internal Server Error
    2. Not Implemented
    3. Bad Gateway
    4. Service Unavailable
    5. Gateway Timeout
    6. Proxying Not Supported

- Signaling Codes: 7.XX
    1. Unassigned
    2. CSM
    3. Ping
    4. Pong
    5. Release
    6. Abort

ID poruke su dugačke 16 bita i koriste se za pronalaženje dupliciranih poruka i spajanje poruka tipa Potvrda/Reset sa porukama tipa Potvrdno/ne potvrdno. Poruke koje su tipa zahtjev će imati isti ID kao i poruke tipa odgovor.

Token polje je polje koje nije nužan dio svakog paketa i može po potrebi biti izostavljeno. Veličina ovog polja može biti od 0 do 8 bajtova. Veličina se može iščitati iz polja duljine tokena koje se nalazi u zaglavlju. Vrijednost ovog polja koristi se za povezivanje zahtjeva i odgovora.

Opcije su definirane CoAP protokolom i one pokazuju koliko opcija može biti uključeno u poruku. Svaka opcija u poruci je određena brojem opcije (engl. *Option Number*), duljinom opcije i samom vrijednosti opcije. Delta kodiranje se koristi za prikazivanje broja opcije. To se izračunava na način da se zbroje sve delte i broj opcije prethodne poruke [10].

Opcija Delta:

- 0 do 12: za deltu između 0 i 12: Predstavlja točnu vrijednost Delte između zadnjeg i željenog ID-a opcije, bez proširene vrijednosti Delte
- 13: za deltu od 13 do 268: Opcija proširene vrijednosti delte je 8 bitna vrijednost Delte minus 13
- 14: za delte između 269 i 65804: Opcija proširene vrijednosti Delte je 16 bitna vrijednost delte minus 269
- 15: Rezervirano za oznaku korisnog tereta gdje su Delta opcija i opcija duljine postavljene na 0xFF.

Opcija Duljina:

- Od 0 do 12: za opciju Duljine između 0 i 12 predstavlja točnu vrijednost duljine bez proširene vrijednosti Duljine
- 13: za opciju Duljine od 13 do 268: Opcija proširene vrijednosti duljine je 8 bitna vrijednost Duljine minus 13
- 14: za opciju Duljine od 269 do 65804: Opcija proširene vrijednosti duljine je 16 bitna vrijednost Duljine minus 269
- 15: Rezervirano za buduću upotrebu. Pogreška je ako je opcija Duljine postavljena da 0xFF.

Opcija Vrijednost:

- Veličina polja opcije Vrijednost definirana je opcijom Duljine u bajtovima.
- Semantika Vrijednosti i format oblikovanja ovise o odgovarajućoj opciji.

Korisni teret sadrži podatke koji se mogu obrađivati i koristiti. Duljina korisnog tereta se može izračunati iz veličine datagrama. Ako se na kraju korisnog tereta ne nalazi oznaka za kraj

tereta, to upućuje na to da teret ne postoji to jest da je duljina tereta 0. Ako se dogodi da je duljina tereta 0 i da se na kraju tereta nalazi oznaka za kraj tereta, tada se takva poruka smatra pogrešno definiranom porukom.

4.2. CoAP komunikacija

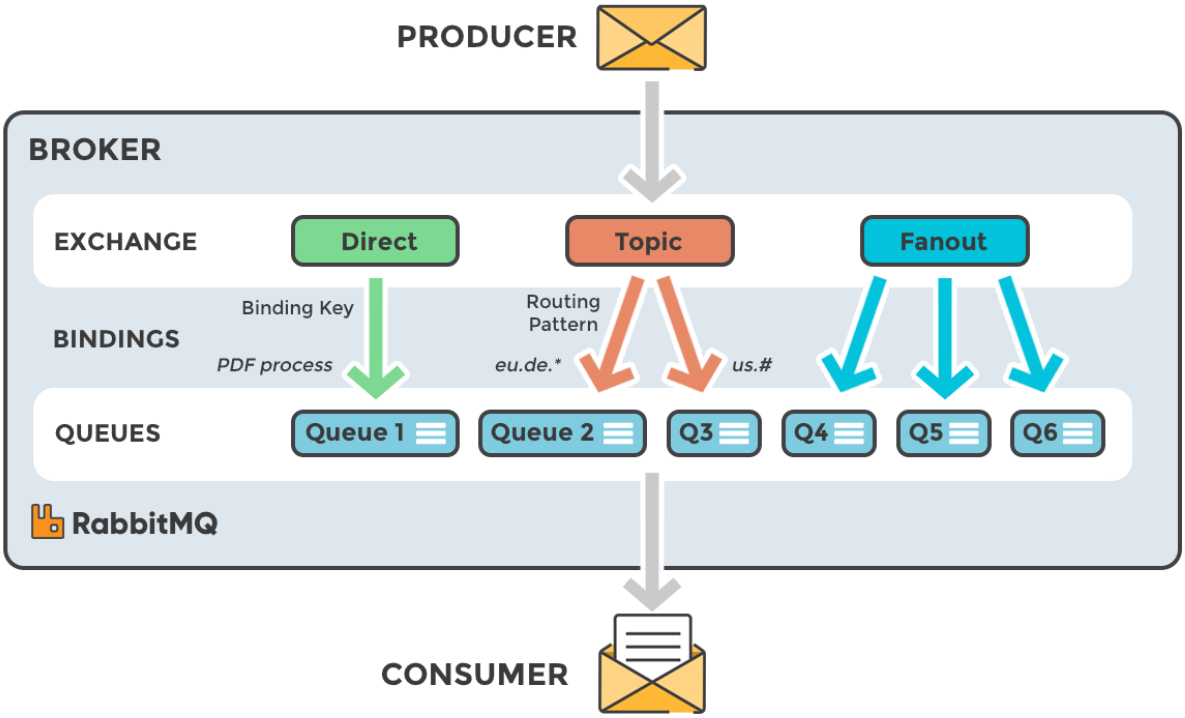
U situacijama gdje se u nekoj mreži nalazi puno čvorova koji obavljaju istu funkcionalnost i koji koriste CoAP protokol za komunikaciju, potrebno je sve čvorove adresirati kako bi im se moglo pristupiti. Da bi se izbjegao takav scenarij gdje svi čvorovi imaju vlastitu adresu, radna grupa *Group Communication for CoAP* je osmislila proširenje na već postojeći standard RFC-7252. Proširenje su nazvali RFC – 7390. Da bi se izbjegao scenarij gdje su svi čvorovi adresirani da bi im se moglo pristupiti, grupa je osmislila rješenje da se dio čvorova formira u grupu umjesto da se svakom čvoru pristupa pojedinačno. To se može izvesti tako da se koristi IP grupno razašiljanje (engl. *Multicast*) kako bi svi čvorovi bili uključeni u komunikaciju.

Ovakav način komuniciranja donosi prednost u tome što je smanjen broj paketa koji trebaju biti poslani svim članovima. Na taj način se produljuje životni vijek čvorova budući da imaju ograničene resurse u vidu napajanja električnom energijom. Nedostatak ovakve metode komuniciranja je loša pouzdanost dostave podataka, potreba za većim spektrom i mogućnost pojave zagušenja. Da bi se riješio taj problem, potrebno je uvesti posrednika u komunikaciju koji će oformiti grupe uređaja i tako spriječiti ne potrebno slanje dupliciranih paketa u *unicast* komunikaciji. Tako će klijent slati grupnu poruku posredniku koji će dalje slati jednostavnije poruke članovima u grupi. Kada čvorovi grupe budu slali informacije klijentu, prvo će ih posrednik prikupiti, oformiti u paket i poslati kao jednu poruku [11].

Primjer ovakve komunikacije je pametna kuća (engl. *Smart Home*) u kojoj se nalaze pametne žarulje. Ako korisnik želi upaliti sve žarulje, morat će svakoj žarulji poslati pojedinačan zahtjev za paljenje. Da bi se to izbjeglo, uvodi se posrednik u komunikaciji koji će od korisnika dobiti zahtjev da se upale ve žarulje, a on će onda svakoj pojedinačno poslati zahtjev da se upali. Na taj način se smanjuje dupliciranje i složenost poslanih paketa.

5. AMQP PROTOKOL

AMQP (engl. *Advanced Message Queuing Protocol*) protokol je binarni protokol otvorenog standarda koji radi na aplikacijskom sloju. Binarni protokol orijentiran je tako da bude razumljiv računalu ili uređaju koristeći binarne znakove, a ne čovjeku koristeći znakove iz ASCII tablice. Sastoji se od 8 bajtnog zaglavlja i korisnog tereta. Značajke koje opisuju AMQP protokol su orijentirane poruke (engl. *message orientation*), redovi (engl. *queuing*), pouzdanost i sigurnost. Podržava nekoliko vrsta orijentirane komunikacije za pouzdano dostavljanje paketa. Opcija najviše jednom (engl. *at most once*) osigurava da se poruka dostavi samo jednom ili da se uopće ne dostavi. Opcija barem jednom (engl. *at least once*) osigurava da će poruka biti dostavljena sigurno jednom ili više puta što znači da može doći do dupliciranja poruka. Opcija točno jednom (engl. *exactly once*) osigurava da se poruka sigurno dostavi do odredišta i to samo jednom bez ponavljanja. Protokol se oslanja na TCP (engl. *Transmission Control Protocol*) protokol i koristi ga kao siguran način dostave poruka [12].



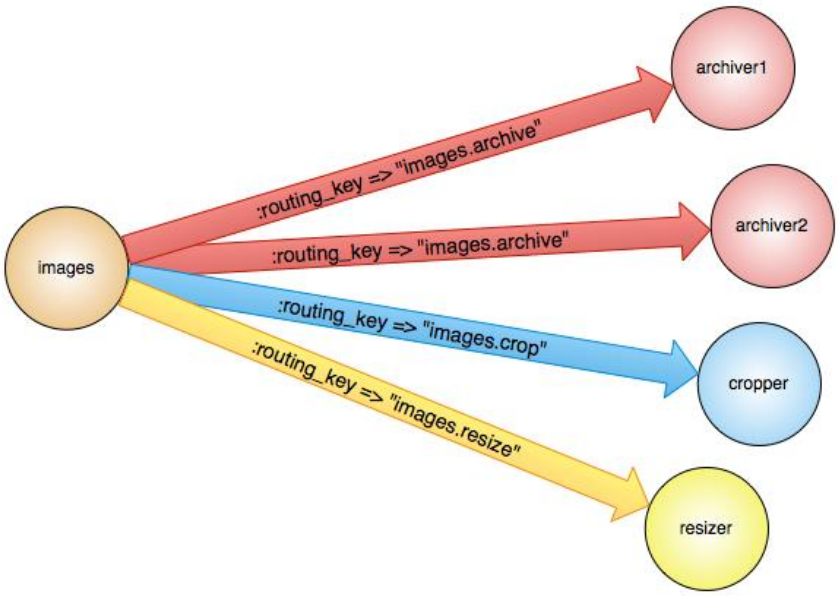
Slika 5.1. *Tok poruke AMQP protokolom* [14].

Protokol podržava dva načina rada, zahtjev / odgovor način rada i objavi / pretplati se način rada. Na slici 5.1. prikazan je tok poruke poslan AMQP protokolom. Za komunikaciju AMQP

protokolom potrebni su pošiljatelj (engl. *Producer*), broker i primatelj (engl. *Consumer*). Na početku komunikacije pošiljatelj kreira razmjenu (engl. *exchange*) s danim imenom pa zatim šalje to ime. Ime služi kako bi se pošiljatelj i primatelj mogli međusobno prepoznati. Nakon što su se prepoznali, primatelj kreira red čekanja i u isto vrijeme ga dodaje prethodnoj razmjeni. Zatim se stvara poveznica (engl. *binding*) između reda čekanja i razmjene. Svaki red čekanja ima svoju vezu, ali može biti povezan s više razmjena i više pošiljatelja. Poruke se mogu razmjenjivati na više načina, izravno, *fanout* metodom, prema temi, ili prema zaglavlju. Poruke se pohranjuju u redove sve dok ih primatelj ne preuzme. Broker čuva poruke sve dok ih primatelj ne preuzme. Da bi broker znao koliko dugo treba čuvati poruke, primatelj pošalje poruku potvrde da je preuzeo poruke. Tek nakon toga se poruke brišu iz reda čekanja [13].

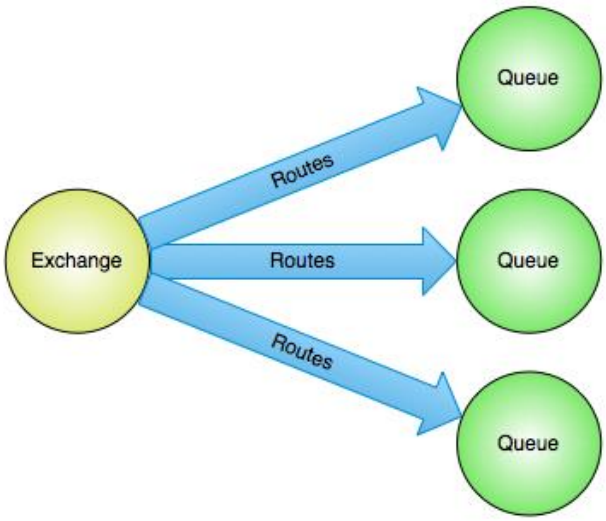
5.1. Tipovi usmjeravanja poruka

Izravna izmjena dostavlja poruke na temelju ključa za usmjeravanje poruka. Takav način prenošenja poruka pogodan je za jednosmjerno (engl. *unicast*) usmjeravanje poruka. Red čekanja se veže s razmjenom i s ključem usmjeravanja A. Kada jednosmjernom izmjenom stigne nova poruka s ključem B, razmjena preusmjerava poruku u red ako su ključevi A i B jednaki.



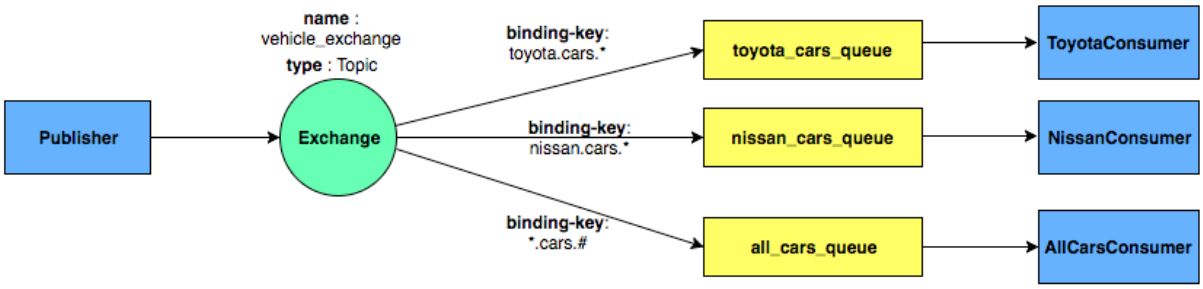
Slika 5.2. *Direktno usmjeravanje poruka* [14].

Fanout razmjena usmjerava poruke na sve redove koji su vezani na nju zanemarujući ključ usmjeravanja. Ako je N redova vezano za poruku A i pojavi se poruka B, kopija poruke A dostavlja se u sve N redove. Ovakav način je pogodan za difuzno emitiranje (engl. *broadcast routing*).



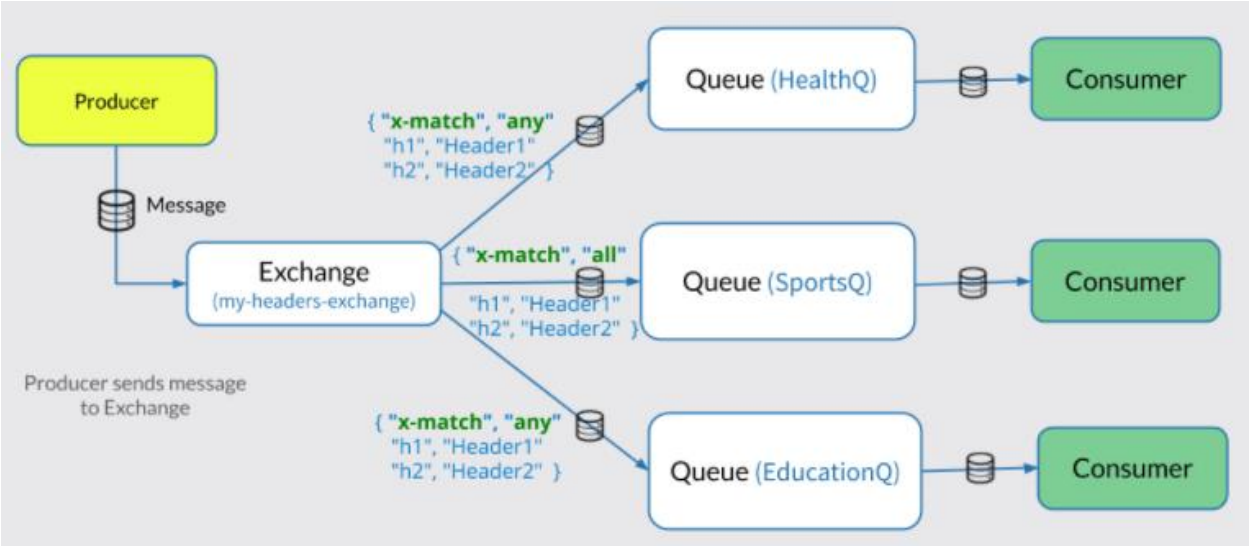
Slika 5.3. *Funout usmjeravanje poruka.*[14]

Razmjenjivanje po temi usmjerava jedan ili više redova na temelju podudaranja ključa usmjeravanja i uzorka koji je korišten za povezivanje reda za razmjenu. Ovaj način razmjenjivanja se često koristi kod implementacije različitih objavi / pretplati se uzoraka. Kada je problem koji način razmjenjivanja primatelj treba odabrati, razmjenjivanje po temi je najčešće pravi izbor. Primjer gdje se koristi ovakva distribucija je distribucija podataka relevantnih za određeno geografsko područje i ažuriranje cijena dionica ili bilo kakvih financijskih informacija [16].



Slika 5.4. *Razmjenjivanje po temi* [15].

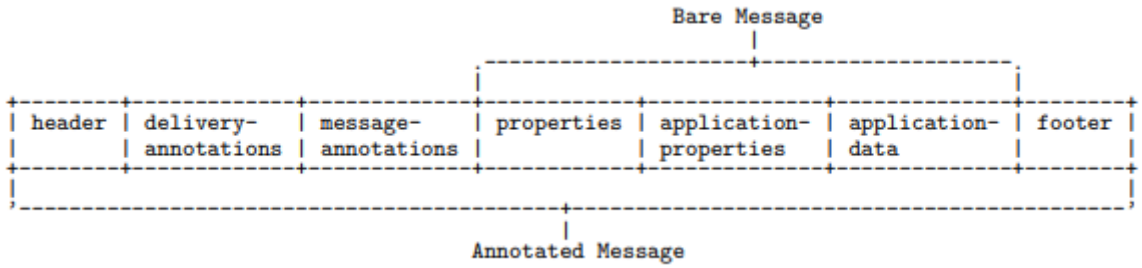
Razmjenjivanje po zaglavlju dizajnirano je tako da se usmjeravanje ne izvršava na temelju ključeva za usmjeravanje nego se u obzir uzimaju parametri iz zaglavlja. Poruka će biti pravilno usmjerena ako su parametri iz zaglavlja jednaki vrijednosti poveznice. Moguće je povezati razmjenjivanje po zaglavlju s više redova. Da bi to bilo moguće potreban je još jedan parametar koji će odrediti tok preusmjeravanja. Kada je „*x-match"* parametar postavljen na *any*, dovoljno je da se samo jedno zaglavlje podudara s tom vrijednosti. Kada je *x-match"* parametar postavljen na *all* svi parametri iz zaglavlja se moraju podudarati s tom vrijednosti. Na slici 5.5. prikazan je primjer usmjeravanja po zaglavlju.



Slika 5.5. *Usmjeravanje po zaglavlju* [17].

5.2. AMQP Paket

AMQP paket se može rastaviti na dva glavna dijela, ogoljena poruka (engl. *bare message*) i anotirana poruka (engl. *annotated message*). Ogoljena poruka se sastoji od tri dijela, standardne postavke (engl. *standard properties*), postavke aplikacije (engl. *application properties*) i korisnih podataka (engl. *application data*). Anotirana poruka sadrži zaglavlje (engl. *header*), bilješke dostave (engl. *delivery annotations*), bilješke poruke (engl. *message annotations*), ogoljenu poruku i podnožje (engl, *footer*).[18] Na slici 5.2.1. je prikazan izgled jednog AMQP paketa.



Slika 5.2.1. *AMQP paket* [18].

Zaglavlje sadrži detalje o isporuci paketa kroz mrežu kojom putuje AMQP paket. Ako se zaglavlje izostavi iz paketa, primatelj tada uzima u obzir postavke koje su postavljene za slučaj ako stigne paket bez zaglavlja. Detalji koji se postavljaju u zaglavlju su izdržljivost (engl. *durable*), prioritet (engl. *priority*), vrijeme trajanja (engl. *time to live*, TTL), prvi stjecatelj (engl. *first acquirer*) i brojač pokušaja dostave (engl. *delivery count*).

Bilješke dostave su opcionalan dio paketa koji sadrži nestandardne atribute koji se prenose od pošiljatelja do primatelja. Ako primatelj ne razumije atribute koje je poslao pošiljatelj, učinci koje ti atributi postižu neće biti ostvareni [19].

Bilješke poruke su opcionalni dio paketa koji sadrži ne standardne atribute koji se prenose od pošiljatelja do primatelja. Atributi koji se prenose su postavke namijenjene infrastrukturi, odnosno uređajima kojima prolazi paket.

Postavke su dio paketa namijenjen za dio koji se odnosi na postavljanje paketa u redove. U ovom dijelu se nalaze polja : ID poruke, ID korisnika, adresa primatelja, subjekt poruke, čvor kojem se šalje, korelacijski identifikator, tip sadržaja, sadržaj za dekodiranje, vrijeme koje

označava isticanje poruke, vrijeme kada je poruka napravljena, ID grupe kojoj poruka pripada i broj sekvence grupe.

Postavke aplikacije su dio paketa namijenjen za dio koji se odnosi na postavljanje paketa u redove i postavke poruke.

U korisnim podacima nalazi se koristan sadržaj namijenjem primatelju koji će te podatke dalje obrađivati i koristiti ih.

Podnožje je opcionalan dio paketa u kojem se nalaze podaci poruci ili isporuci koji se izračunavaju nakon što je konstruirana ogoljena poruka. To su hash poruke, razni ključevi i detalji za enkripciju [19].

6. USPOREDBA PROTOKOLA

U ovom poglavlju će biti prikazane prednosti i nedostatci pojedinih protokola. Protokoli će se usporediti po veličini poruke i troškovima, potrošnji energije i zahtjevima resursa, širini pojasa i kašnjenju, pouzdanosti i sigurnosti.

Na kraju poglavlja će biti prikazana tablica gdje će se na jednom mjestu moći vidjeti razlike u protokolima.

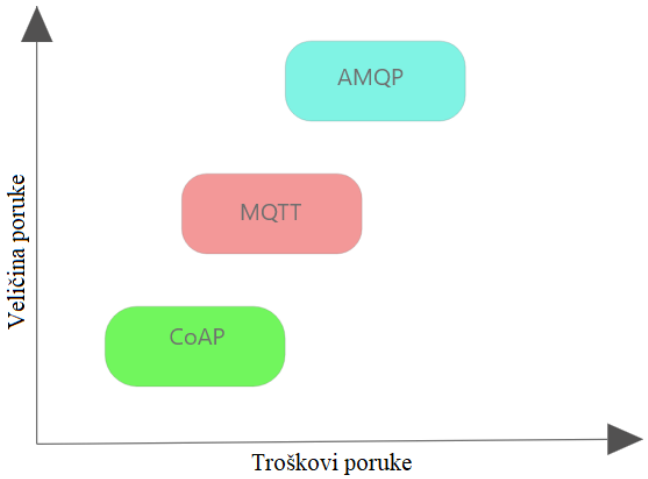
6.1. Veličina poruke i troškovi (overhead)

Protokoli MQTT i AMQP su protokoli koji se oslanjaju na TCP protokol pa stoga moraju imati i troškove za uspostavu i zatvaranje veze koji dolaze uz TCP protokol. MQTT protokol ima poruku veličine 2 bajta, ali zbog TCP načina komunikacije povećava se veličina poruke, a s time rastu i troškovi komunikacije. Za povezivanje i objavljivanje uz QoS = 1 potrebno je 58 bajtova što znači da 2 bajta zauzima MQTT zaglavlje, a 56 bajtova zauzima TCP. Maksimalna veličina MQTT poruke je 256 MB [20].

AMQP protokol je binarni protokol čije je zaglavlje veličine 8 bajtova. Njegova zadaća za sigurnost u komunikaciji, pouzdanost i učinkovito međusobno djelovanje s drugim sustavima povećava veličinu poruke, a samim time i troškove u komunikaciji. Maksimalna veličina AMQP poruke je jednaka veličini korisnog tereta.

CoAP protokol se oslanja na UDP protokolu. Za razliku kod TCP protokola gdje je potrebna povratna informacija da je veza uspostavljena, kod UDP protokola nema potvrde da je veza uspostavljena te se tako uštedjelo na veličini poruke. Samim time i troškovi su manji jer je potrebno poslati manje bajtova. Veličina poruke kod CoAP protokola mora biti dovoljno mala da stane u IP *datagram* kako ne bi došlo do fragmentacije poruke.

Slika 6.1. prikazuje odnos veličine poruke i troškova kod navedenih protokola.



Slika 6.1. *Omjer veličine i troškova poruke* [21].

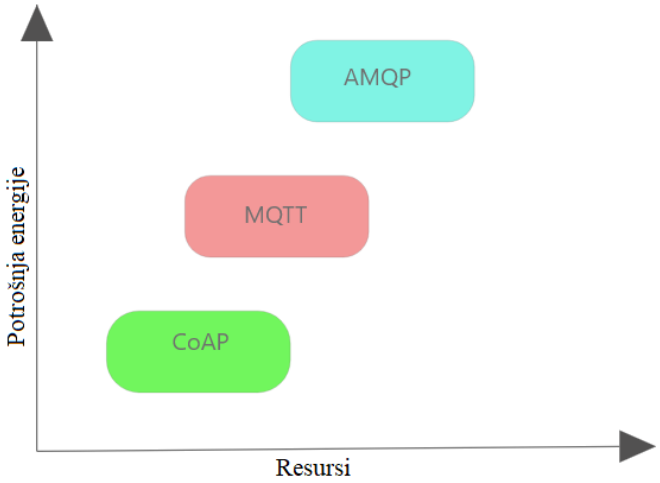
6.2. Potrošnja energije i zahtjevi resursa

Omjer potrošnje energije i zahtjeva resursa znatno ovisi o protokolu na kojem se temelje MQTT, CoAP i AMQP. Glavni razlog veće potrošnje energije kod MQTT i AMQP protokola je oslanjanje na TCP protokol. TCP protokol ne koristi svu raspoloživu širinu spektra, a zbog toga što koristi pristup sporog starta da bi se izbjegla zagušenja u mrežu, vrijeme povratnog putovanja (engl. *Round Trip Time,* RTT) se udvostručava. Zbog ugrađenog mehanizma kontrole toka kod TCP protokola, MQTT je pouzdaniji za prijenos a time i više energije troši.

CoAP protokol se oslanja na UDP protokol i zbog toga ne koristi mehanizme za kontrolu toka. Prema istraživanjima u [13], u istim uvjetima rada i uzimajući u obzir da se u prijenosu ne gube paketi, CoAP troši znatno manje energije u NON načinu rada nego MQTT s QoS = 0 načinu rada.

Zbog osiguravanja pouzdanosti u prijenosu AMQP koristi najviše energije u usporedbi s prethodna 2 protokola. Prema [22], AMQP protokolu je potrebno više resursa da bi mogao zadovoljiti sve potrebe pouzdanosti i sigurnosti.

Na slici 6.2. prikazan je omjer potrošnje energije i resursa potrebnih za obavljanje zadaća.



Slika 6.2. *Odnos potrošnje energije i resursa* [21].

6.3. Širina pojasa i kašnjenje

Prema istraživanjima koja su provedena u [23] gdje su promatrani MQTT i CoAP protokoli u testnom okruženju, manje kašnjenje poruka je imao MQTT u odnosu na CoAP kada je gubitak paketa bio na niskoj razini. Kada su se uspoređivala kašnjenja pri većem gubitku paketa, CoAP je imao veća kašnjenja u odnosu na MQTT. CoAP postiže bolje rezultate kod iskoristivosti širine pojasa i vremena odziva. U aplikacijama gdje su vremenski uvjeti ne savršeni i gdje se zahtjeva malo kašnjenje i mala potrošnja resursa, CoAP se pokazao kao najbolje rješenje.

Na ispitivanju u [23] gdje su testirani MQTT, CoAP i AMQP protokoli s porukama gdje je korisni teret u jednom slučaju iznosio 10, a u drugom 1000 poruka, najbolje rezultate je imao CoAP. Nakon njega dolazi MQTT pa AMQP. U slučaju kada je korisni teret bio 1000 poruka, najlošije rezultate je postigao MQTT. Nakon njega dolazi AMQP pa CoAP. Tablica 6.1. prikazuje rezultate dobivene nakon testiranja u slučajevima s 10 i 1000 poruka.

Mala veličina poruke (< 5 kilobajta)	
10 poruka, latencija manja -> veća	1000 poruka, latencija manja -> veća
CoAP -> MQTT -> AMQP	CoAP -> AMQP -> MQTT
Velika poruka (> 50 kilobajta)	
10 poruka, latencija manja -> veća	1000 poruka, latencija manja -> veća
Za 20 kilobajta: CoAP -> AMQP -> MQTT	Za 20 kilobajta: CoAP -> AMQP -> MQTT
Za 50 kilobajta: CoAP -> AMQP -> MQTT	Za 50 kilobajta: CoAP -> AMQP -> MQTT

Tablica 6.1. *Rezultati testiranja protokola* [23].

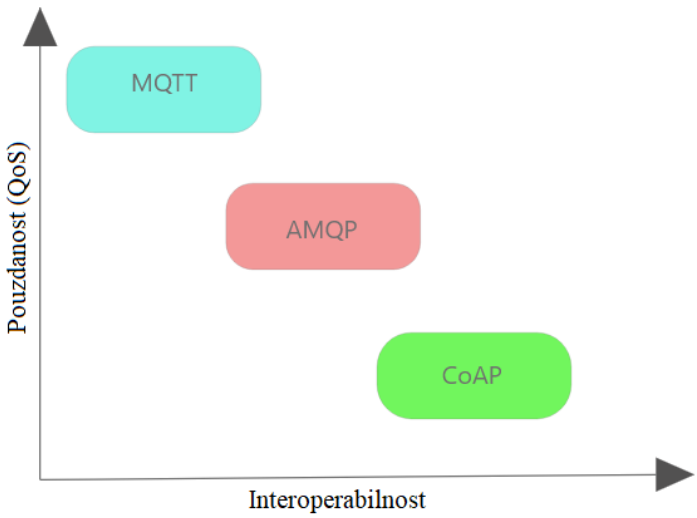
Kada se gleda iskoristivost širine pojasa, važan čimbenik je transportni protokol. Kod MQTT i AMQP protokola TCP povećava kašnjenje i smanjuje iskoristivost pojasa. TCP protokol ne koristi svu raspoloživu širinu spektra, a zbog toga što koristi pristup sporog starta da bi se izbjegla zagušenja u mrežu, vrijeme povratnog putovanja se udvostručava. Kod CoAP protokola transportni protokol UDP koristi samo 2 *datagrama* u uzlaznoj i silaznoj vezi te tako smanjuje vrijeme povratnog odgovora.

6.4. Pouzdanost

Razina pouzdanosti odnosno kvaliteta usluge (engl. *Quality of Service*, QoS) obrnuto je proporcionalna interoperabilnosti. TCP protokol osigurava povratnu informaciju je li poruka stigla do odredišta ili nije. Zbog toga MQTT i AMQP imaju veću razinu pouzdanosti od CoAP-a. MQTT protokol razlikuje tri razine QoS-a: 0 - najviše jednom,1 - najmanje jednom i 2 - točno jednom. AMQP razlikuje dvije razine QoS-a: određeni format koji je sličan QoS 0 kod MQTT i ne određeni format koji je sličan QoS 1 kod MQTT.

CoAP transport vrši na UDP protokolu pa stoga on nema mehanizam za potvrdu isporuke paketa. CoAP ima dvije razine QoS-a: NON koji je sličan QoS 0 kod MQTT i CON koji je sličan QoS 1 kod MQTT. UDP može uzrokovati problem kod IoT uređaja ako se dogodi situacija da je uređaj prešao iz jedne u drugu mrežu. IP adresa koja je bila u prvoj mreži dinamički će se promijeniti u drugoj mreži pa će prekinuti komunikacija jer sustav neće prepoznati da se radi o istom uređaju s novom IP adresom [23].

Najveći izazov u IoT protokolima je postići interoperabilnost odnosno dvosmjernu komunikaciju između čvora i korisnika. MQTT protokol jedini radi na način objavi / pretplati se te na taj način jedini zadovoljava dvosmjernu komunikaciju. CoAP i AMQP funkcioniraju na principu zatraži / odgovori. AMQP koristi međuspremnike i poveznice kako bi pravilno mogao poslati pakete.



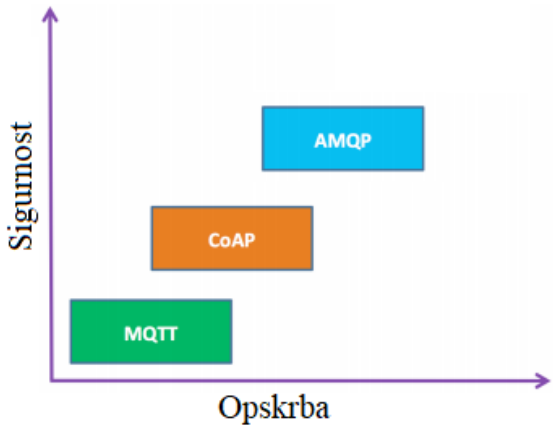
Slika 6.4. *Omjer QoS i interoperabilnosti* [23].

6.5. Sigurnost

Kao što je prikazano na slici 5.5. AMQP protokol ima najveću razinu sigurnosti. Sigurnosni mehanizam kod AMQP protokola je TLS (engl. *Transport Layer Security*). To postiže pomoću tri modela rada: model jedinstvenog porta TLS-a (engl. *Single port TLS model*), čisti TLS (engl. *Pure TLS*) i model TLS WebSockets.TSL je sigurnosni protokol koji je dizajniran za pružanje sigurne komunikacije u računalnim mrežama. TLS se može kombinirati zajedno sa SASL-om(engl. *Simple Authentication and Security Layer*). SASL je okosnica (engl. *framework*) za provjeru autentičnosti i sigurnost podataka kod protokola. SASL razdvaja mehanizme provjere autentičnosti od aplikacijskog protokola i dopušta da se koristi bilo koji mehanizam koji SASL podržava [24].

Metode koje CoAP koristi za autentifikaciju i enkripciju su DTLS (engl. *Datagaram Transport Layer Security*) i IPsec (engl. *Internet Protocol Security*). DTLS je komunikacijski protokol koji pruža sigurnost aplikacijama koje su temeljena na *datagramima*. Budući da je CoAP temeljen na UDP protokolu, on koristi DTLS kako bi se spriječilo prisluškivanje ili krivotvorenje poruke. IPsec je mrežni protokol namijenjen za autentifikaciju i u računalnoj mreži. Često se koristi u virtualnim mrežama (engl. *Virtual Private Network*, VPN)

MQTT protokol ima najmanju razinu sigurnosti u odnosu na prethodna dva protokola. Sigurnost kod MQTT protokola se postiže korisničkim imenom i lozinkom. MQTT protokol je transportni protokol i njegova primarna zadaća je prijenos podataka, zadaća pružatelja usluge koju koristi MQTT protokol je da osigura prikladnu zaštitu. To se najčešće postiže korištenjem TLS-a [6].



Slika 6.5. *Omjer sigurnosti i opskrbe* [13].

6.6. IoT iskorištenost i standardizacija

MQTT protokol je standardiziran OASIS standardom. Iako se koristi u mnogo implementacija i u raznim aplikacijama i dalje nije globalni standard. Prva verzija protokola je osmišljena 1999. godine, a značajnija upotreba je počela 2013. godine kada je IBM dodao preinake za već postojeći protokol. MQTT danas koriste brojne poznate tvrtke kao što su Facebook, IBM, Cisco i druge [3].

CoAP je protokol standardiziran IETF standardom kako bi integrirao IoT uređaje s internetom. Prva verzija protokola je standardizirana 2010. godine kako bi pojednostavio povezivanje uređaja na internet i time zamijenio HTTP protokol. Kako bi protokol postao što prilagodljiviji M2M aplikacijama naknadno su dodavane nove funkcionalnosti. CoAP koriste tvrtke kao što su Cisco, IoTivity i druge.

AMQP je najprošireniji i najkorišteniji protokol u IoT sustavima. Protokol je standardiziran OASIS ISO/IEC standardom 2003. godine. Protokol je dizajniran da podržava veliki broj aplikacija za razmjenu poruka sa sigurnošću da će se poruke dostaviti. AMQP protokol koriste tvrtke kao što su Microsoft, JP Morgan, Barclay i druge [13].

6.7. Komparativna analiza protokola

U tablici 6.1. prikazane su komparativna analiza protokola koji su opisani u ovom radu. Protokoli su uspoređeni po 13 kategorija pa ih je moguće usporediti po sličnostima i razlikama.

MQTT protokol osnovan je 1999. godine. Potrebni mrežni elementi za njegov rad su klijent i posrednik. Radi na principu objavi / pretplati se. Veličina zaglavlja mu iznosi 2 bajta. Maksimalna veličina poruke koju može prenositi je 256 MB. Metode rada koje se koriste u protokolu su Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close. Protokol nudi tri razine kvalitete usluge: QoS 0, QoS 1 i QoS 2. Protokol je standardiziran OASIS standardom. Transportni protokol kojim se obavlja prijenos podataka je TCP, a u rijetkim slučajevima to može biti i UDP. Sigurnost prijenosa se postiže TLS i SSL mehanizmima. Portovi na kojima se odvija komunikacija su 1883 i 8883. Za prijenos podataka se koristi binarno kodiranje. Protokol je otvorenog tipa i može se bez licenciranja slobodno koristiti.

CoAP protokol osnovan je 2010. godine. Potrebni mrežni elementi za njegov rad su klijent i server ili klijent i posrednik. Radi na principu zatraži / odgovori ili objavi / pretplati se. Veličina zaglavlja mu iznosi 4 bajta. Maksimalna veličina poruke mora biti dovoljno malena da stane u jedan IP datagram. Metode rada koje se koriste u protokolu su Get, Post, Put, Delete. Protokol nudi dvije razine kvalitete usluge: CON i NON. Protokol je standardiziran IETF standardom. Transportni protokol kojim se obavlja prijenos podataka je UDP. Sigurnost prijenosa se postiže DTLS i IPsec mehanizmima. Portovi na kojima se odvija komunikacija su 5683 i 5684. Za prijenos podataka se koristi binarno kodiranje. Protokol je otvorenog tipa i može se bez licenciranja slobodno koristiti.

AMQP protokol osnovan je 2003. godine. Potrebni mrežni elementi za njegov rad su klijent i server ili klijent i posrednik. Radi na principu zatraži / odgovori ili objavi / pretplati se. Veličina zaglavlja mu iznosi 8 bajtova. Maksimalna veličina poruke nije definirana pa ona iznosi onoliko koliko je velik koristan teret. Metode rada koje se koriste u protokolu su Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close. Protokol nudi dvije razine kvalitete usluge: Određeni i ne određeni format. Protokol je standardiziran OASIS i ISO/IEC standardom. Transportni protokol kojim se obavlja prijenos podataka je TCP. Sigurnost prijenosa se postiže TLS/SSL, IPsec i SASL mehanizmima. Portovi na kojima se odvija komunikacija su 5671 i 5672. Za prijenos podataka se koristi binarno kodiranje. Protokol je otvorenog tipa i može se bez licenciranja slobodno koristiti.

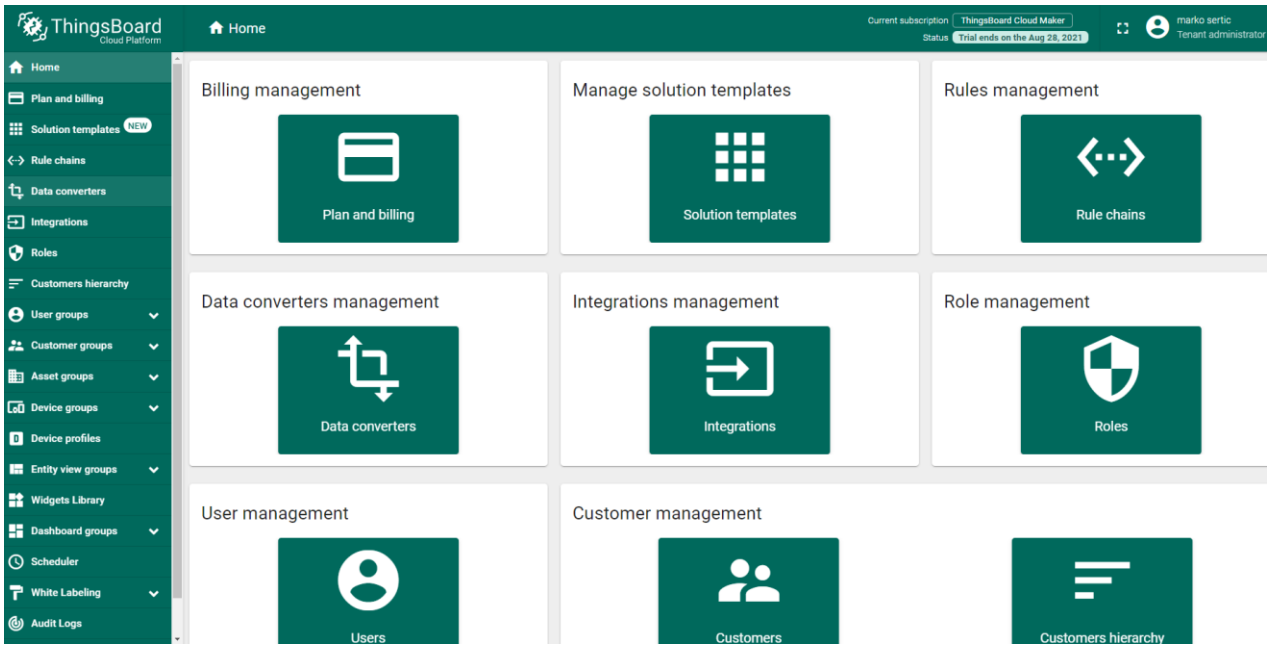
Kriterij	MQTT	CoAP	AMQP
1. Godina	1999	2010	2003
2. Arhitektura	Klijent / Broker	Klient / Server ili Klijent / Posrednik	Klient/Server ili Klijent / Posrednik
3. Način rada	Objavi / Pretplati se	Zatraži / Odgovori ili Objavi / Pretplati se	Zatraži / Odgovori ili Objavi / Pretplati se
4. Veličina zaglavlja	2 bajta	4 bajta	8 bajtova
5. Veličina poruke	Maksimalna veličina je 256 MB	Dovoljno mala da stane u jedan IP datagram	Ne definirana, jednaka veličini korisnog tereta
6. Metode rada	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Put, Delete	Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close
7. Kvaliteta usluge (QoS)/ Pouzdanost	QoS 0 – Najviše jednom, QoS 1 – Najmanje jednom, QoS 2 – Točno jednom	CON (slično kao QoS 0) i NON (slično kao QoS 1)	Određeni format (slično kao QoS 0) i Ne određeni format (slično kao QoS 1)
8. Standardi	OASIS	IETF	OASIS, ISO/IEC
9. Transportni protokol	TCP (u nekim slučajevima moguće i UDP)	UDP	TCP
10. Sigurnost	TLS/SSL	DTLS, IPSec	TLS/SSL, IPSec, SASL
11. Portovi	1883/ 8883 (TLS/SSL)	5683 (UDP port)/ 5684 (DLTS)	5671 (TLS/SSL), 5672
12. Način kodiranja	Binarno kodiranje	Binarno kodiranje	Binarno kodiranje
13. Licenciranje	Otvoreni kod	Otvoreni kod	Otvoreni kod

Tablica 6.1. *Komparativna analiza protokola.*

7. SLUČAJEVI KORIŠTENJA IoT PROTOKOLA

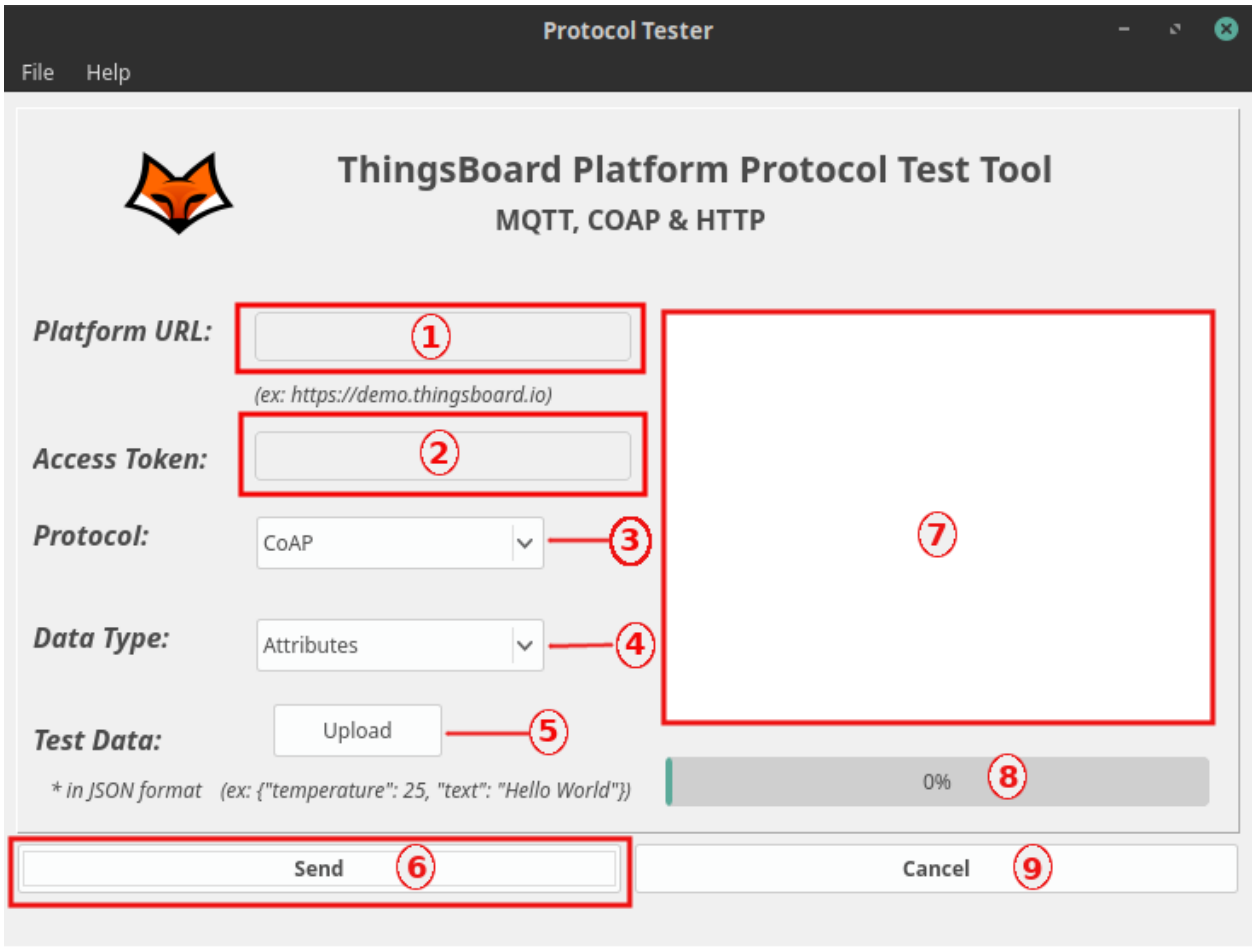
Za kreiranje različitih korisničkih scenarija prilikom izrade praktičnog rada korišteni je platforma ThingsBoard Cloud i aplikacija za testiranje protokola ThingsBoard-Platform-Protocol-Test-Tool.

ThingsBoard Cloud je potpuno upravljiva i skalabilna platforma koja se koristi za izradu IoT aplikacija. Platforma je namijenjena za korištenje svima koji ne žele imati vlastitu instancu platforme već ju koriste kao uslugu u oblaku (engl. *cloud*) [25]. Za pristup platformi potrebno je otvoriti korisnički račun s kojim se na temelju odabrane pretplate ostvaruje pravo na korištenje odgovarajućih ograničenja. Na slici 7.1. prikazan je izgled ThingsBoard sučelja nakon otvaranja korisničkog računa.



Slika 7.1. *ThingsBoard sučelje.*

ThingsBoard-Platform-Protocol-Test-Tool je aplikacija za testiranje i slanje telemetrijskih podataka CoAP, MQTT i HTTP protokolima [26]. Aplikaciju je moguće koristiti na Linux, Ubuntu ili Debian operacijskim sustavima. Aplikacija je pogodna za kreiranje korisničkih scenarija jer je sučelje aplikacije prilagođeno za jednostavno spajanje s ThingsBoard platformom. Sučelje aplikacije je prikazano na slici 7.2.

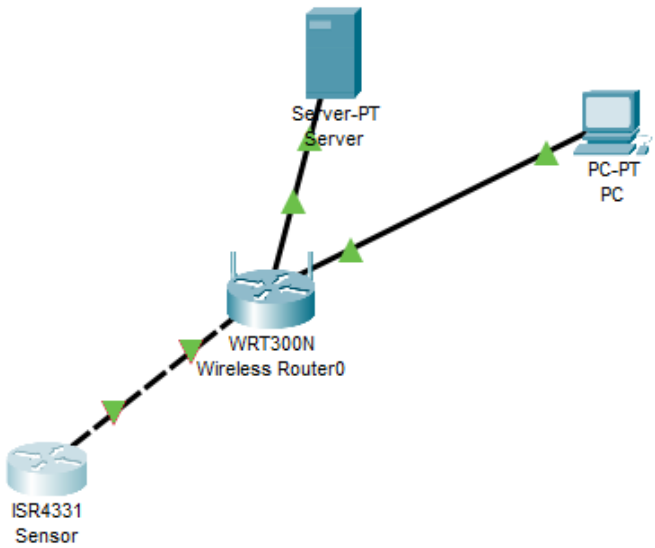


Slika 7.2. *Sučelje aplikacije.*

1. Mjesto za upisivanje URL adrese
2. Mjesto za upisivanje tokena za pristup uređaju (engl. *Device Access Token*)
3. Određivanje koji protokol se koristi za slanje podataka (CoAP, MQTT ili HTTP)
4. Određivanje tipa podatka koji se šalje (atributi klijenta ili telemetrijski podaci)
5. Dugme za učitavanje korisnih podataka (telemetrija)
6. Dugme za slanje
7. Prozor koji prikazuje zapisnik o poruci koja se šalje
8. Prozor koji prikazuje razinu izvršetka slanja
9. Dugme za prekidanje slanja

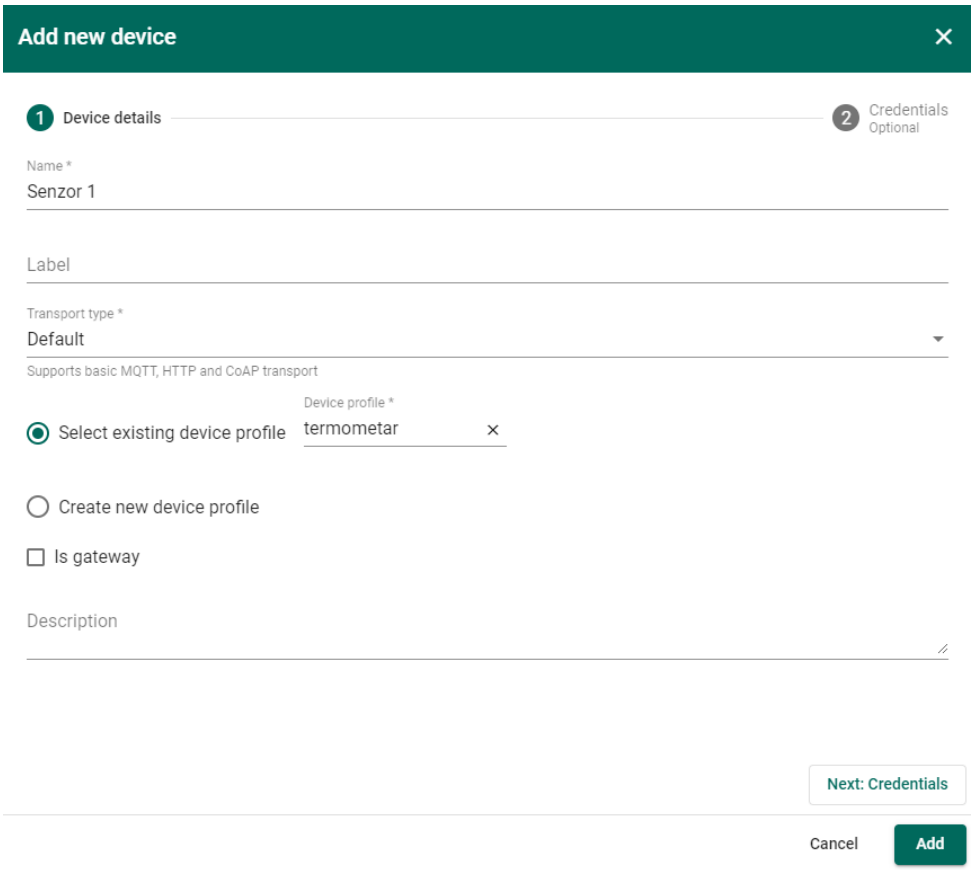
7.1. MQTT primjer korištenja

U ovom korisničkom scenariju koristi se senzor za mjerenje temperature i pH vrijednosti vode koji svaku sekundu šalje podatke o temperaturi i pH vrijednosti vode na ThingsBoard platformu. Senzor za mjerenje temperature i pH vrijednosti vode se nalazi na površini jezera. Senzor je povezan bežično na usmjerivač koji preusmjerava promet paketa. Paketi odlaze na server gdje se obrađuju, a sa računalom se može pristupiti podacima koje je senzor prikupio i poslao an sever. Slika 7.1.1. prikazuje shemu ovog korisničkog scenarija. Budući da se podatak šalje svake sekunde, korisniku nije jako važno da svaki paket s temperaturom i pH vrijednosti stigne do platforme. Korisniku nije nužno da baš svake sekunde zna kolika je temperatura i pH vrijednost vode jer se pretpostavlja da se temperatura vode ne mijenja učestalo pa je dopušteno da neki paketi ne stignu na svoje odredište. MQTT protokol je odličan izbor za ovakav scenarij jer se pomoću njega može izabrati određena kvaliteta usluge. U ovom primjeru se koristi QoS = 0 što znači da će se paket poslati najviše jednom (engl. *Fire and Forget*). Senzor će pakete slati svake sekunde te mu nije bitno je li paket stigao na odredište ili nije. Podatak će se obraditi kada paket stigne do platforme, ali neće biti poslana nikakva povratna informacija senzoru o tome jeli paket stigao ili nije, je li obrađen ili nije.



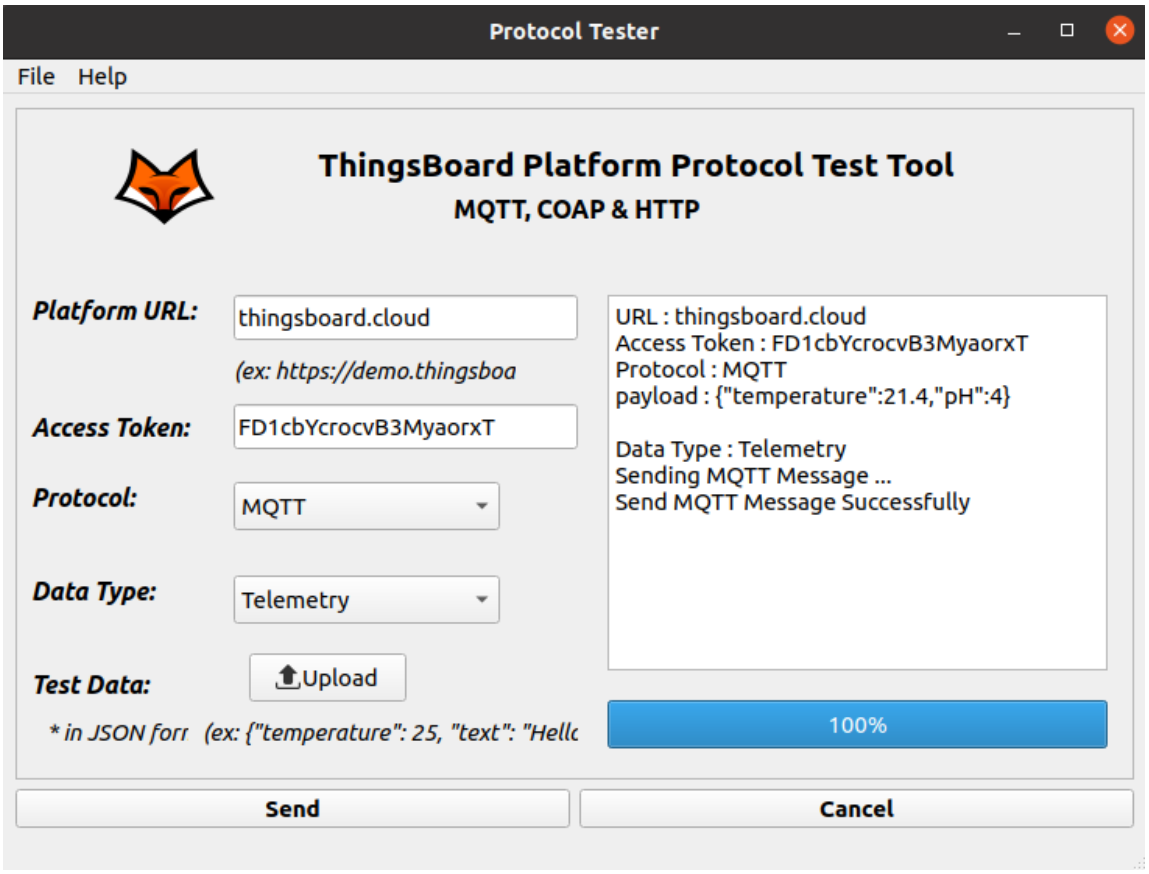
Slika 7.1.1. Shema korisničkog scenarija MQTT protokolm.

Za simulaciju ovog korisničkog slučaja potrebno je napraviti novi uređaj na ThingsBoardu koji će prikazivati podatke dobivene od aplikacije.



Slika 7.1.2. *Kreiranje senzora*.

Kada je senzor napravljen potrebno je u aplikaciju unijeti odgovarajuće podatke kako bi se uspješno mogli poslati podaci s aplikacije na platformu.



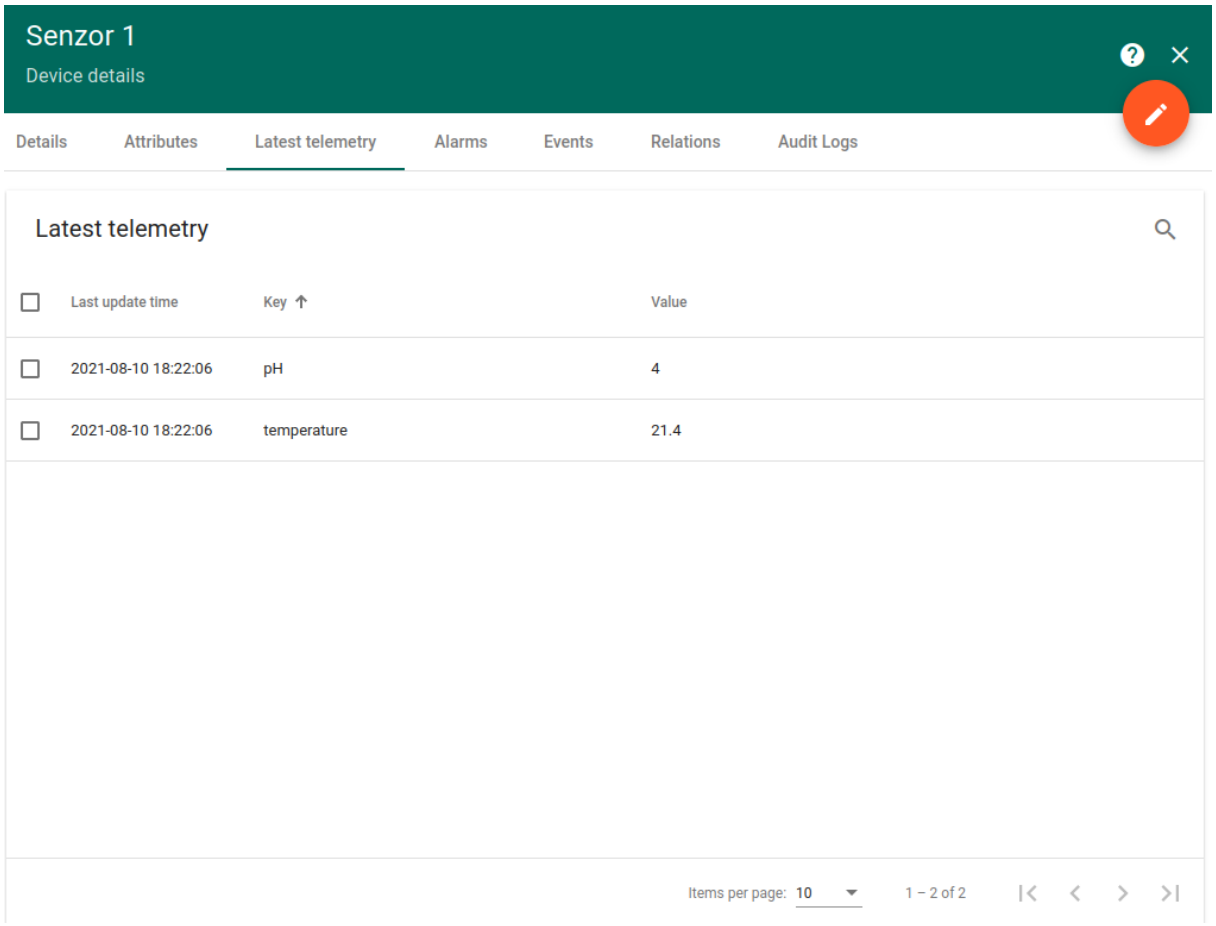
Slika 7.1.3. *Slanje poruke.*

U zapisničkom prozoru vidimo podatke koji su korišteni za slanje poruke. Odredišna adresa je URL *thingsboard.cloud* i na toj adresi se nalazi platforma koja simulira klijenta s pripadajućim tokenom koji ga jednoznačno određuje. Teret koji se šalje je temperatura i pH vrijednost vode te se on šalje u JSON formatu. JSON format za slanje telemetrije je oblika *„{ključ : vrijednost}“* što je u ovom slučaju *{„temperature“:21.4, „pH“:4}* Tip podataka koji se šalje je telemetrija i poruka je uspješno poslana.

Simulaciju slanja tereta je moguće obaviti i naredbom iz terminala naredbom:

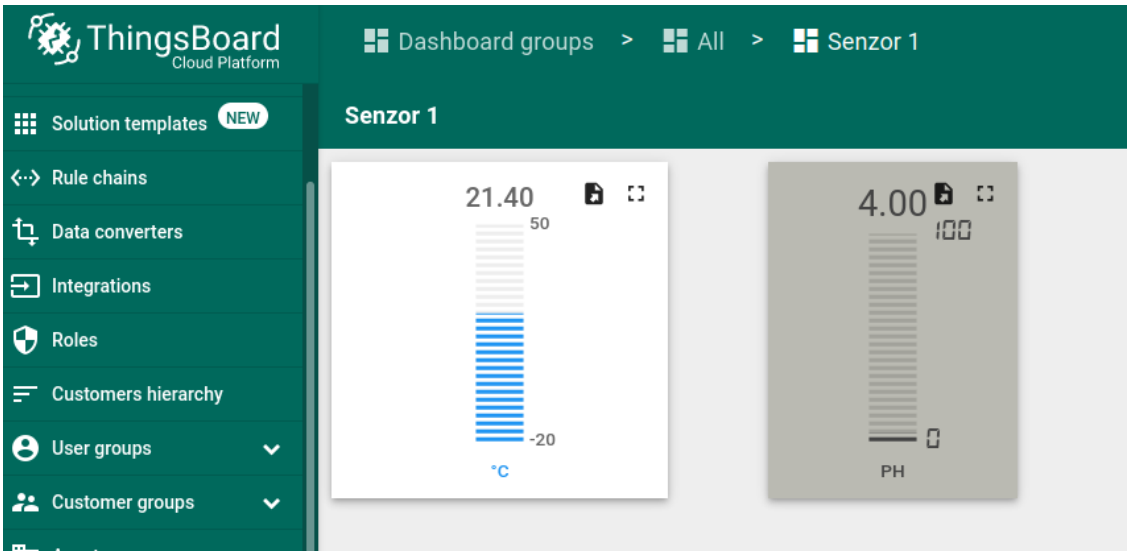
```
cat telemetry-data-as-array.json | mqtt pub -v -h
"thingsboard.cloud" -t "v1/devices/me/telemetry" -u
'$ACCESS_TOKEN' -s
```

gdje je potrebno navesti putanju do JSON dokumenta i pristupni token uređaja sa ThingsBoard platforme.

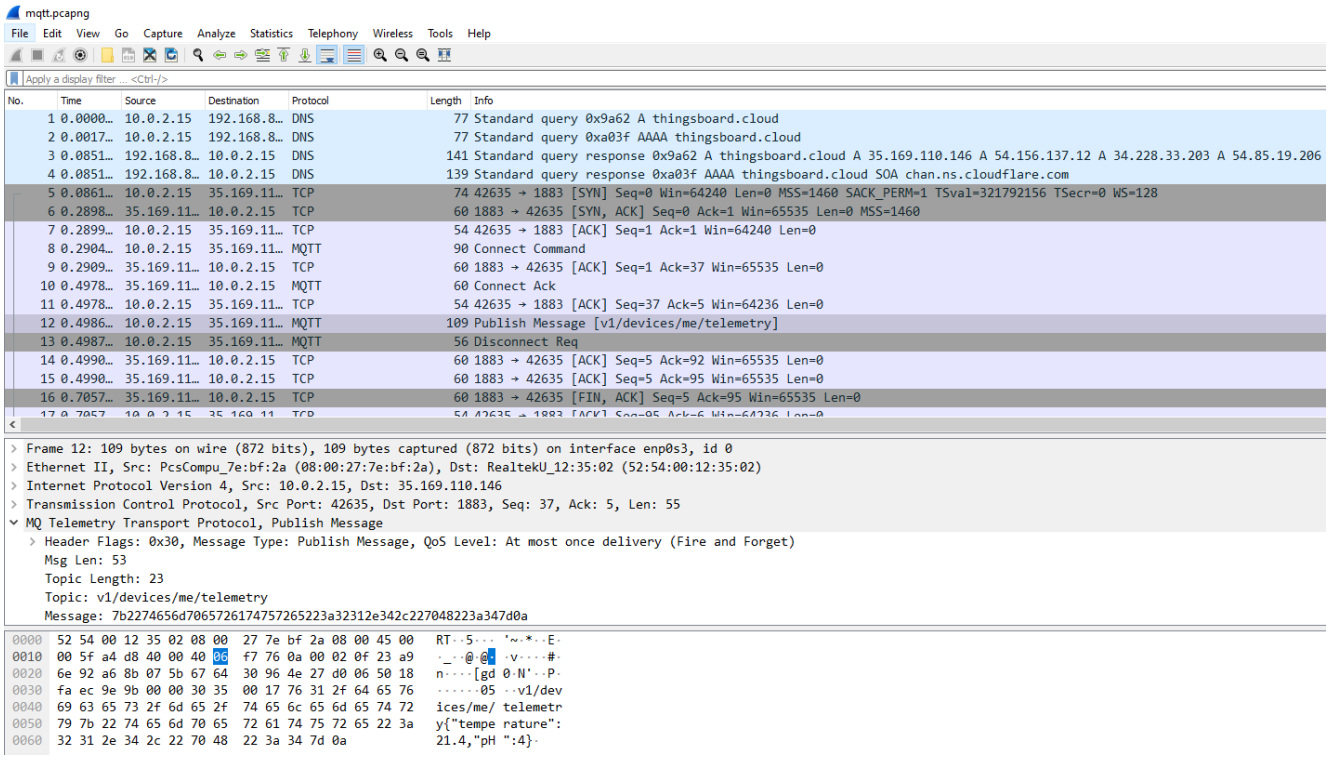


Slika 7.1.4. *Telemetrija na senzoru.*

Slika 7.1.4. prikazuje senzor na ThingsBoard platformi. Na slici je vidljivo kada su podaci primljeni, kako se zovu podaci koji su primljeni te koja im je vrijednost. Ti se podaci ovisno o potrebi mogu samo iščitavati kao telemetrija na senzoru ili se mogu prikazati na kontrolnoj ploči na ThingsBoard platformi kao što je prikazano na slici 7.1.5.



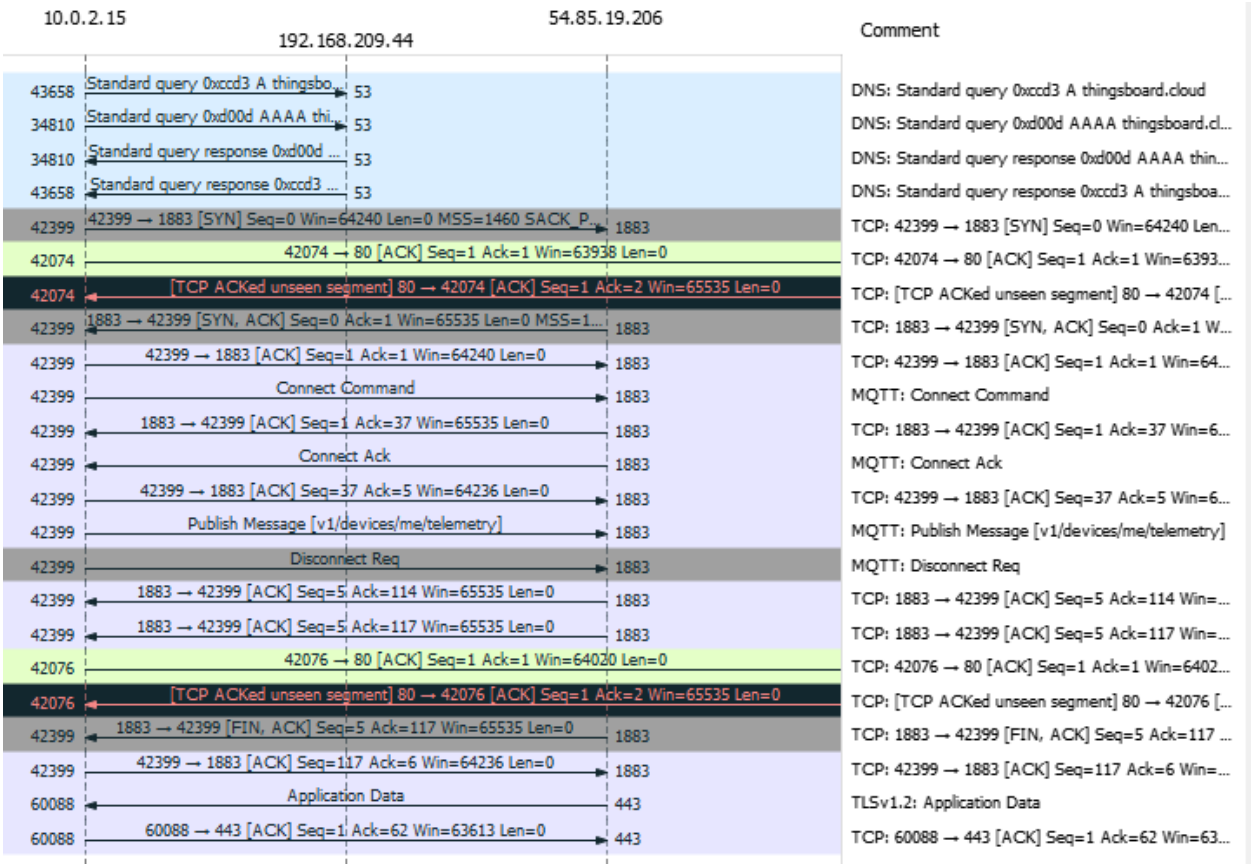
Slika 7.1.5. *Prikaz podataka na kontrolnoj ploči.*



Slika 7.1.6. *Promet u Wiresharku.*

U Programu Wireshark snimljen je internetski promet u trenutku slanja telemetrijske poruke iz Protocol Tester aplikacije prema platformi ThingsBoard. Za ovu komunikaciju MQTT protokol se oslanja na TCP protokol koji koristi za uspostavu veze s ThingsBoardom što je vidljivo u petom redu na slici 7.1.6. Nakon što je veza uspostavljena što se može vidjeti u desetom redu na slici, aplikacija šalje poruku sa teretom. U analizi tereta je vidljivo kakvi se telemetrijski podaci šalju, s kojom QoS razinom i s kojeg izvorišta na koje odredište. Na slici 7.1.7. prikazan je

redosljed slanja paketa. Vidljivo je kako izgleda provjera uspostave konekcije kod TCP protokola i sa kojeg odredišta na koje odrediše se šalju paketi.

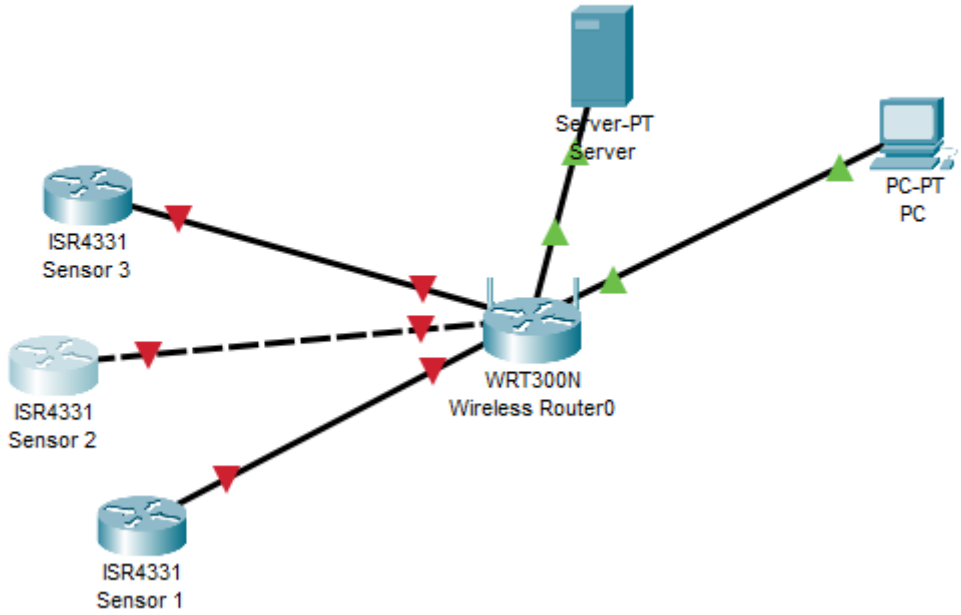


Slika 7.1.7. Dijagram toka MQTT prometa.

7.2. CoAP primjer korištenja

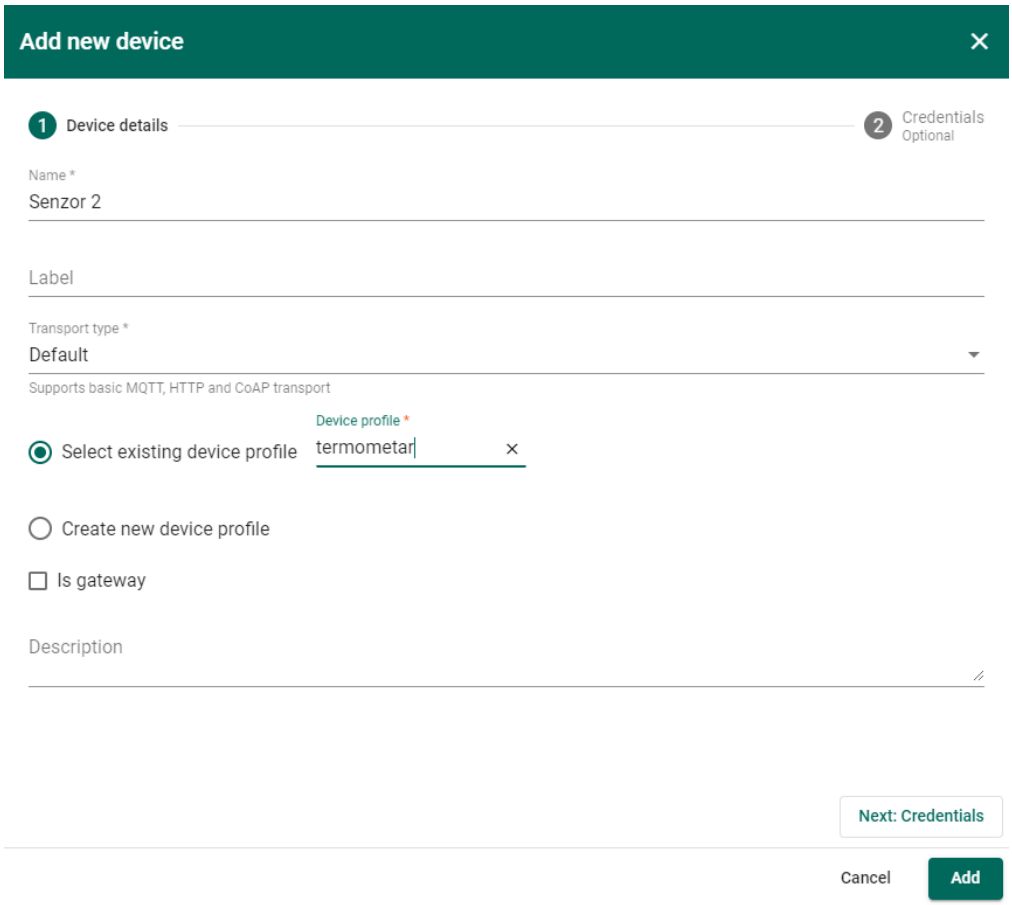
Kao u prethodnom scenariju i u ovom korisničkom scenariju koristi se senzor za mjerenje temperature i pH vrijednosti vode koji svaku sekundu šalje podatke o temperaturi i pH vrijednosti vode na ThingsBoard platformu. Za razliku od prethodnog slučaja gdje je zadatak senzora bio samo da šalje podatke na ThingsBoard u ovom slučaju senzor mora imati mogućnost komunicirati i sa susjednim senzorima. Da bi se to omogućilo, koristi se CoAP protokol [27].

U ovom korisničkom slučaju se na već postojeću senzorsku mrežu koja je povezana s internetom dodaje još jedan senzor. Razlog zašto se u ovom korisničkom slučaju koristi CoAP protokol jest ta što je CoAP kompatibilan sa HTTP protokolom i oslanja se na UDP transportni protokol. UDP je pogodan za višesmjerno odašiljanje jer nema povratnih paketa kontrolnog zbroja (engl. *checksum*) što omogućava uređajima da brzo međusobno razmjene poruke. Na slici 7.2.1. prikazana je shema korisničkog scenarija komunikacije CoAP protokolom.



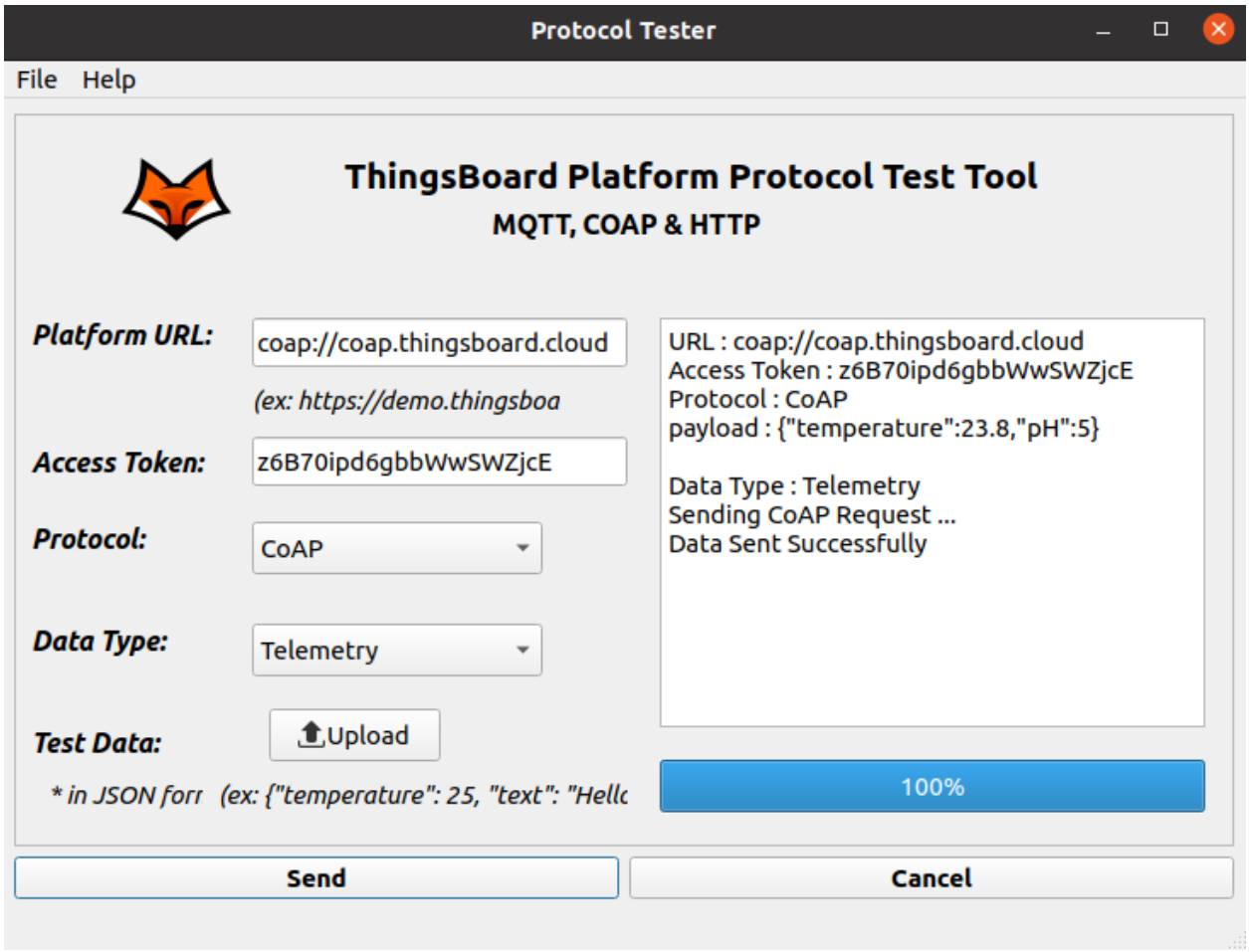
Slika 7.2.1. Shema korisničkog scenarija CoAP protokolom.

Za simulaciju ovog korisničkog slučaja potrebno je napraviti novi uređaj na ThingsBoardu koji će prikazivati podatke dobivene od aplikacije.



Slika 7.2.2. *Kreiranje senzora.*

Kada je senzor napravljen potrebno je u aplikaciju unijeti odgovarajuće podatke kako bi se uspješno mogli poslati podaci s aplikacije na platformu.



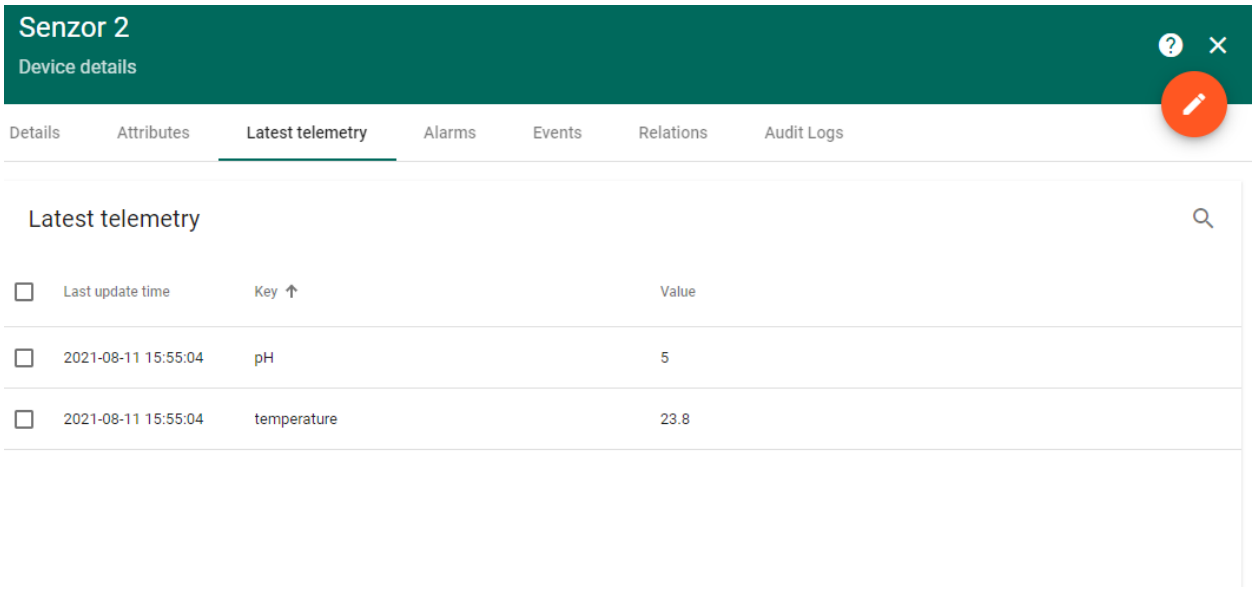
Slika 7.2.3. *Slanje poruke.*

U zapisničkom prozoru vidimo podatke koji su korišteni za slanje poruke. Odredišna adresa je URL *coap://coap.thingsboard.cloud* i na toj adresi se nalazi platforma koja simulira klijenta sa pripadajućim tokenom koji ga jednoznačno određuje. Teret koji se šalje je temperatura i pH vrijednost vode te se on šalje u JSON formatu. JSON format za slanje telemetrije je oblika *„{ključ : vrijednost}“* što je u ovom slučaju *{„temperature“:23.8, „pH“:5}* Tip podataka koji se šalje je telemetrija i poruka je uspješno poslana.

Simulaciju slanja tereta je moguće obaviti i naredbom iz terminala naredbom:

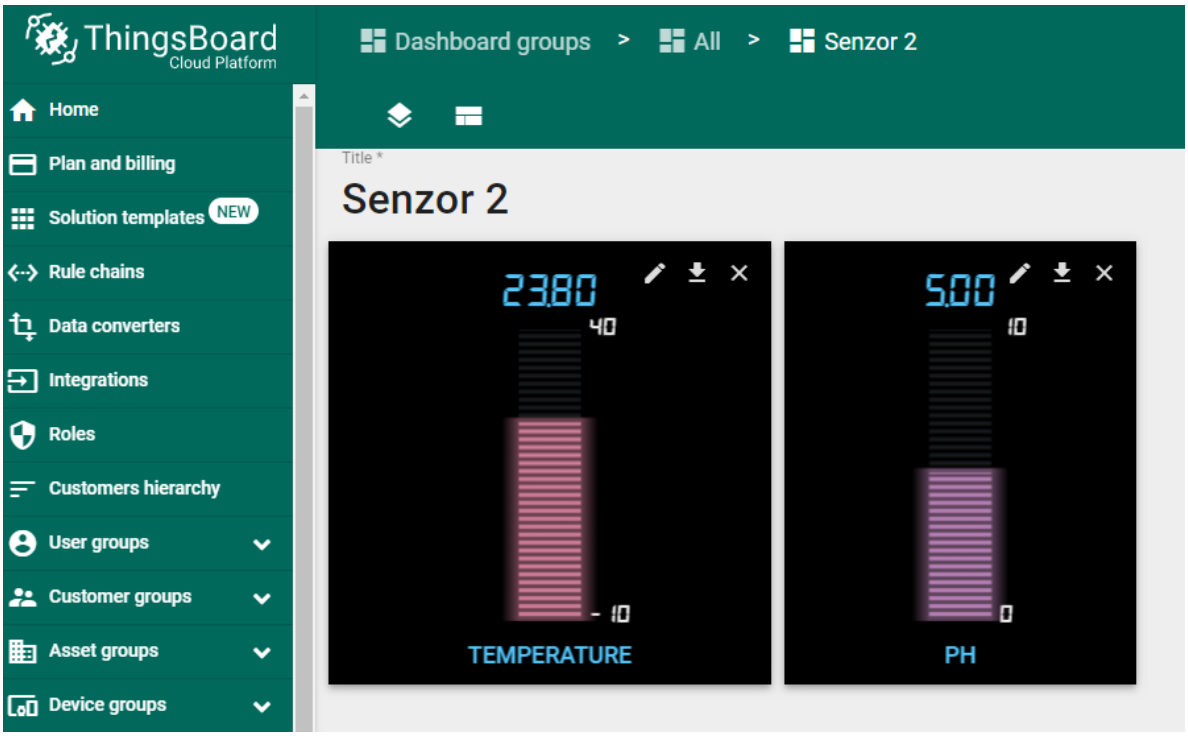
```
cat new-attributes-values.json | coap post
coap://coap.thingsboard.cloud/api/v1/$ACCESS_TOKEN/attributes
```

gdje je potrebno navesti putanju do JSON dokumenta i pristupni token uređaja sa Thingsboard platforme.

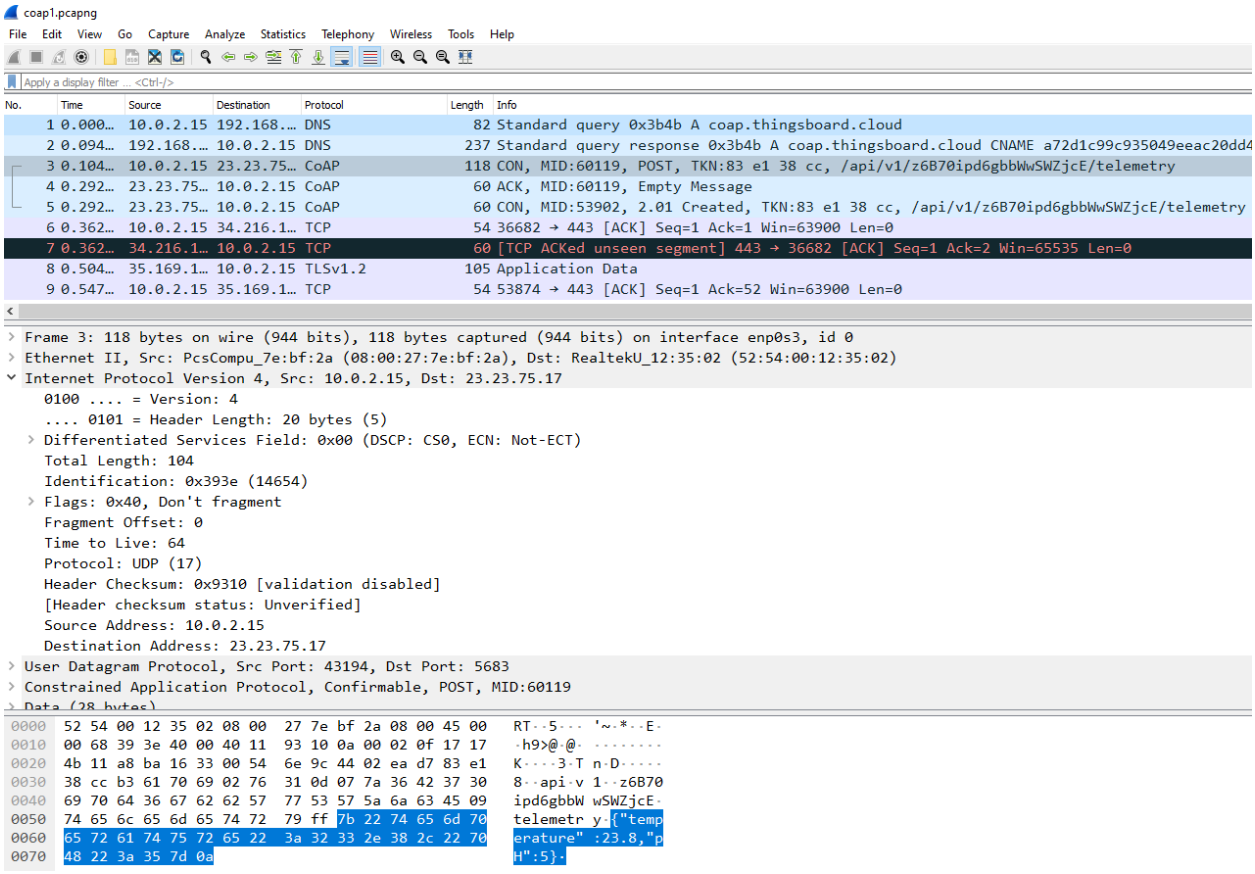


Slika 7.2.4. *Telemetrija na senzoru.*

Slika 7.2.4. prikazuje senzor na ThingsBoard platformi. Na slici je vidljivo kada su podaci primljeni, kako se zovu podaci koji su primljeni te koja im je vrijednost. Ti se podaci ovisno o potrebi mogu samo iščitavati kao telemetrija na senzoru ili se mogu prikazati na kontrolnoj ploči na ThingsBoard platformi kao što je prikazano na slici 7.2.5.



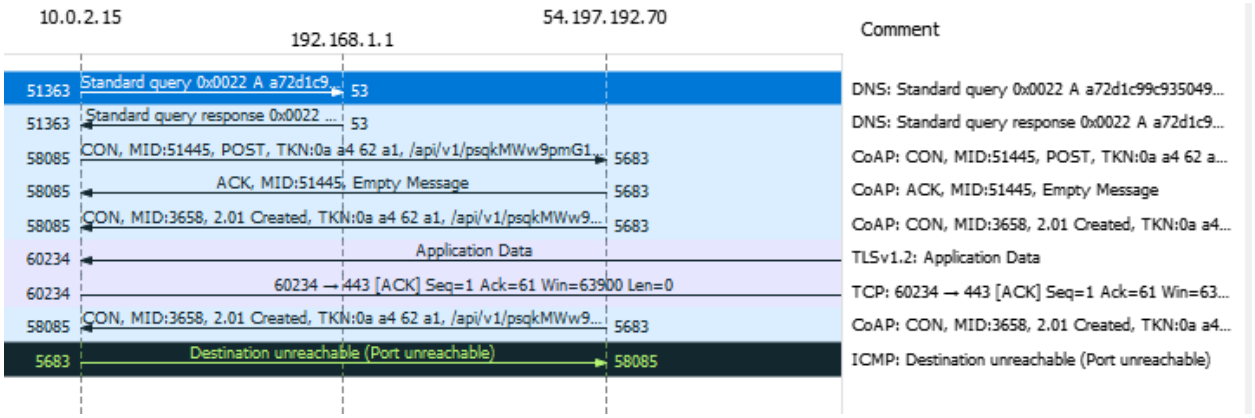
Slika 7.2.5. *Prikaz podataka na kontrolnoj ploči.*



Slika 7.2.6. *Promet u Wiresharku.*

U Programu Wireshark snimljen je internetski promet u trenutku slanja telemetrijske poruke iz Protocol Tester aplikacije prema platformi ThingsBoard. Za ovu komunikaciju CoAP protokol se oslanja na UDP protokol koji koristi za uspostavu veze sa ThingsBoardom što je vidljivo u prozoru za analiziranje paketa na slici 7.2.6. Kod UDP protokola nema uspostavljanja veze između pošiljatelja i primatelja kao što je to bio slučaj kod MQTT protokola. U analizi tereta je vidljivo kakvi se telemetrijski podaci šalju i s kojeg izvorišta na koje odredište.

Na slici 7.2.7. prikazan je redosljed slanja paketa. Vidljivo je kako nema provjere uspostave konekcije kao kod TCP protokola jer se CoAP oslanja na UDP transportni protokol. Vidljivo je s kojeg odredišta na koje odrediše se šalju paketi.

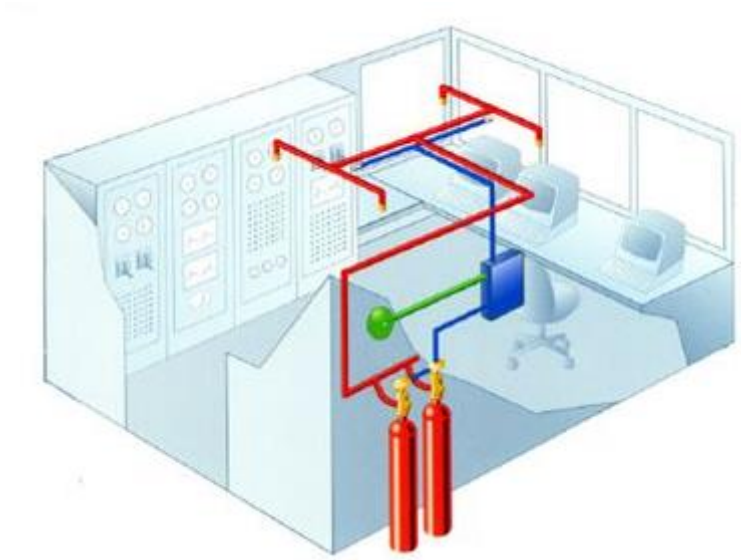


Slika 7.2.7. Dijagram toka CoAP protokola.

7.3. AMQP primjer korištenja

U ovom korisničkom slučaju imamo nadzornu sobu u kojoj su smješteni serveri koji prikupljaju podatke od senzora koji su prethodno spomenuti u prijašnjim slučajevima i drugih uređaja koji su povezani na istu mrežu. U nadzornoj sobi se nalaze 3 računala na kojima je moguće pratiti senzore pomoću ThingsBoard platforme. Soba je opremljena protu požarnim halon plinom.

ThingsBoard pruža podršku za prijenos podataka putem HTTP, MQTT i CoAP protokolima [29]. Stoga nije moguće implementirati korisnički slučaj u kojem se koristi AMQP protokol za prijenos telemetrije jer se AMQP koristi isključivo za slanje bitnih signalizacijskih poruka.



Slika 7.3.1. *Nadzorna soba [28].*

AMQP protokol je pogodan protokol za ovakve situacije jer je pouzdan. Pouzdan je zbog toga što će klijent dobiti povratnu informaciju o tome što će se dogoditi nakon što je poslao poruku brokeru. Dobit će informaciju je li poruka koju je poslao u red čekanja isporučena te je li ju netko preuzeo iz reda ili ju treba ponovno slati sve dok ju netko ne pročita. Dobit će informaciju i o tome što će se dogoditi nakon što je poruka pročitana ili što će se dogoditi ako poruka nije pročitana.

U slučaju da se dogodi požar u nadzornoj sobi ili iz bilo kojeg razloga nadzorni sustav procijeni da je sigurnost opreme ugrožena, pokrenut će se procedura za gašenje požara odnosno u prostoriju će biti ispušten halon plin, a vrata prostorije će se zatvoriti. Kada se plin ispusti u prostoriju, u prostoriji će nestati kisika. Budući da je čovjek uvijek važniji od opreme, za ovakve sustave se koristi AMQP protokol.

Na ulazu u prostoriju kod vrata se nalazi čitač kartica, klijent koji šalje informacije brokeru o tome jesu li ulazna vrata otvorena te ima li koga u prostoriji. U prostoriji se nalaze detektori plina koji mjere količinu kisika u prostoriji. Kada se dogodi situacija u kojoj je potrebno ispustiti halon plin u prostoriju prvo će se provjeriti ima li ljudi u prostoriji. Čitač kartice će poslati brokeru informaciju ima li ljudi ili ne. Na temelju povratne informacije čitač kartica će znati treba li zatvoriti vrata da se plin ispusti u prostoriju ili treba upaliti sirenu za uzbunu. Senzor za mjerenje kisika u prostoriji će nakon određenog vremenskog razmaka izmjeriti količinu kisika te poslati podatke brokeru koji će odlučiti mogu li se ulazna vrata otvoriti te je li razina kisika u prostoriji dovoljna za boravak čovjeka ili prostoriju treba dodatno provjetriti.

8. ZAKLJUČAK

Zbog ubrzanog tehnološkog napretka u svakodnevnom životu, razne tehnologije se razvijaju i primjenjuju u gospodarstvu sa ciljem poboljšanja životnog standarda, povećanja produktivnosti i efikasnosti. To je cilj i tehnologije Interneta stvari. Na moderan način pratiti i upravljati objektima iz svakodnevnog života te tako efikasnije i učinkovitije živjeti u doticaju s tehnologijom s ciljem poboljšanja svakodnevnog života.

MQTT protokol je protokol koji radi na principu objavi/ pretplati se. Služi za komunikaciju između klijenta i posrednika. Klijenti objavljivači šalju poruke s određenom temom posredniku, a klijenti pretplatitelji se pretplaćuju ne temu koja ih zanima. Protokol se najčešće koristi za prijenos telemetrijskih poruka.

CoAP protokol je ograničeni aplikacijski protokol primarno namijenjen za uređaje s ograničenim kapacitetima koji se nazivaju čvorovi. Protokol čvorovima omogućuje da međusobno komuniciraju, a da bi to mogli moraju biti povezani na internet. Služi za komunikaciju između klijenta i posrednika. Protokol se temelji na UDP protokolu i radi u uvjetima gdje je zagušenost mreže velika i gdje je propusnost malena. Komunikacija se odvija s kraja na kraj gdje svaki uređaj ima svoju IP adresu. Naprednija metoda komunikacije je da se određeni uređaji grupiraju pod jednu IP adresu te tako posrednik sprječava slanje dupliciranih paketa.

Značajke koje opisuju AMQP protokol su orijentirane poruke, redovi, pouzdanost i sigurnost. Protokol se oslanja na TCP protokol i koristi ga kao siguran način dostave poruka. Protokol podržava dva načina rada, zahtjev / odgovor način rada i objavi / pretplati se način rada. Za komunikaciju AMQP protokolom potrebni su pošiljatelj, broker i primatelj. Pošiljatelj stvara šalje poruke i stavlja ih u redove čekanja koje broker čuva dok ih primatelj ne pokupi.

U praktičnom dijelu rada implementirani su korisnički slučajevi za svaki protokol. U implementaciji korisničkih slučajeva korištena je platforma ThingsBoard Cloud i alat za simulaciju prijenosa podataka odabranim protokolom ThingsBoard-Platform-Protocol-Test-Tool. Paketi koji su poslani sa simulacijskog alata na ThingsBoard platformu snimljeni su Wireshark alatom za analizu internetskog prometa. Na temelju snimljenog prometa napravljena je analiza komunikacije i protokola koji su korišteni.

9. SAŽETAK

U ovom radu su opisani protokoli MQTT, CoAP i MQTT, najčešće korišteni aplikacijski protokoli u tehnologiji Interneta stvari. Struktura svakog protokola je detaljno objašnjena i način na koji funkcionira. Uspoređene su karakteristike, specifikacije i korisnički slučajevi svih navedenih protokola te su naglašene prednosti i nedostaci. Za svaki protokol je implementiran odgovarajući korisnički slučaj, te je napravljena analiza komunikacije i snimljenog prometa.

KLJUČNE RIJEČI: MQTT, CoAP, AMQP, IoT, ThingsBoard

COMPARATIVE ANALYSIS OF IoT PROTOCOLS

SUMMARY

This paper describes the MQTT, CoAP, and MQTT protocols, the most commonly used application protocols in IoT technology. The structure of each protocol is explained in detail and method how it works. The characteristics, specifications and use cases of all the mentioned protocols are compared and the advantages and disadvantages are highlighted. An appropriate user case was implemented for each protocol, and an analysis of communication and recorded traffic was performed.

KEY WORDS: MQTT, CoAP, AMQP, IoT, ThingsBoard

10. LITERATURA

[1] A. Bahga, V. Madisetti, Internet of Things: A Hands-On Approach, 2014., https://books.google.co.in/books/about/Internet_of_Things.html?id=JPKGBAAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false (kolovoz, 2021.)

[2] M. Kpizingui: Demystifying the MQTT maze: clients, servers, connection, publish, subscribe and its applications., 2019,

https://morioh.com/p/93ba2353480e (kolovoz, 2021.)

[3] Wikipedia: MQTT,

https://en.wikipedia.org/wiki/MQTT (kolovoz, 2021.)

[4] The HiveMQ Team: MQTT Client and Broker and MQTT Server and Connection Establishment Explained - MQTT Essentials: Part 3, 2019.,

https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/ (kolovoz, 2021.)

[5] Github: public_brockers,

https://github.com/mqtt/mqtt.org/wiki/public_brokers (kolovoz, 2021.)

[6] OASIS Standard: MQTT Control Packet type, 2014.,

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Figure_2.2_- (kolovoz, 2021.)

[7] EMQ: MQTT Retain Message, 2019.,

https://www.emqx.io/blog/mqtt5-features-retain-message (kolovoz, 2021.)

[8] OpenLab Pro: MQTT Packet Format,

https://openlabpro.com/guide/mqtt-packet-format/ (kolovoz, 2021.)

[9] Wikipedia: Constrained Application Protocol,

https://en.wikipedia.org/wiki/Constrained_Application_Protocol (kolovoz, 2021.)

[10] Z. Shelby, K. Hartke, C. Borman: RFC 7252, ožujak, 2014.,

https://datatracker.ietf.org/doc/html/rfc7252 (kolovoz, 2021.)

[11] I. Ishaq, J. Hoebeke, F. Van Den Abeele, J. Rossey, I. Moerman, P. Demeester: Flexible Unicast-Based Group Communication for CoAP-Enabled Devices, travanj 2014.,

https://www.mdpi.com/1424-8220/14/6/9833/htm (kolovoz, 2021.)

[12] OASIS Standard: OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0, listopad 2012.,

http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf (kolovoz, 2021.)

[13] N. Naik: Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP, listopad 2017.,

https://core.ac.uk/download/pdf/160743474.pdf (kolovoz, 2021.)

[14] RabbitMQ: AMQP 0-9-1 Model Explained,

https://www.rabbitmq.com/tutorials/amqp-concepts.html (kolovoz, 2021.)

[15] B. Kunudu: Topic Exchange in AMQP – RabbitMQ, ožujak 2020.,

https://jstobigdata.com/rabbitmq/topic-exchange-in-amqp-rabbitmq/ (kolovoz, 2021.)

[16] RabbitMQ: Direct exchane routing,

https://www.rabbitmq.com/tutorials/amqp-concepts.html (kolovoz, 2021.)

[17] B. Kundu: Headers Exchange in AMQP – RabbitMQ, ožujak 2020.,

https://jstobigdata.com/rabbitmq/headers-exchange-in-amqp-rabbitmq/ (kolovoz 2021.)

[18] OASIS Standard: OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0: Transport, listopad 2012.,

http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf (kolovoz, 2021.)

[19] IBM: Mapping AMQP and IBM MQ message fields, studeni 2021.,

https://www.ibm.com/docs/en/ibm-mq/8.0?topic=applications-mapping-amqp-mq-message-fields (kolovoz, 2021.)

[20] Steve: Understanding the MQTT Protocol Packet Structure, siječanj 2021.,

http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ (kolovoz, 2021.)

[21] Virtual Lab: Comparison of IoT Protocols,

https://docs.vlab.iotify.io/iot-protocols/comparison-of-iot-protocols (kolovoz, 2021)

[22] J. E. Luzuriaga, M. Perez, P. Boronat, J.C. Cano, C. Calafete, P. Manzoni: Testing AMQP protocol on unstable and mobile networks, 2014,

https://www.researchgate.net/publication/282914229_Testing_AMQP_Protocol_on_Unstable_and_Mobile_Networks (kolovz, 2021.)

[23] E. Al-Masri, K. R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, C. Yan: Investigating Messaging Protocols for theInternet of Things (IoT), srpanj 2020.,

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9090208 (kolovoz, 2021.)

[24] Wikipedia: Advanced Message Queuing Protocol,

https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol (kolovoz, 2021.)

[25] The ThingsBoard Authors: What is ThingsBoard Cloud?, 2021.

https://thingsboard.io/products/paas/what-is-thingsboard-cloud/ (kolovoz, 2021.)

[26] T. Shiyaz: ThingsBoard-Platform-Protocol-Test-Tool, ožujak 2019.,

https://github.com/shiyazt/ThingsBoard-Platform-Protocol-Test-Tool (kolovoz, 2021.)

[27] J. Fries: Why are IoT developers confused by MQTT and CoAP?, svibanj 2017.,

https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/Why-are-IoT-developers-confused-by-MQTT-and-CoAP (kolovoz, 2021.)

[28] https://www.foxvalleyfire.com/wp-content/uploads/2016/04/Kidde111.jpg (kolovoz, 2021.)

[29] The ThingsBoard Authors: ThingsBoard Monolithic architecture, 2021.,

https://thingsboard.io/docs/reference/monolithic/#transport-components (kolovoz, 2021.)

[30] A. Bahga, V. Madisetti: Internet of Things: A Hands – on Approach, 2015.,

https://books.google.co.in/books/about/Internet_of_Things.html?id=JPKGBAAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=false (kolovoz, 2021.)

[31] H. Mishra: Physical Design of IoT, 2020.,

https://iotbyhvm.ooo/physical-design-of-iot/