Primjena tehnologije Internet stvari (IoT) za udaljeno praćenje mjerenja senzora

Ivić, Slaven

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

Sveučilišni studij elektrotehnike

PRIMJENA TEHNOLOGIJE INTERNET STVARI (IOT) ZA UDALJENO PRAĆENJE MJERENJA SENZORA

Diplomski rad

Slaven Ivić

Osijek, 2019.

SADRŽAJ

1. UVOD

Glavni cilj tehnološkog napretka je razvoj društva, gospodarstva i općenito, poboljšanje životnog standarda. Tehnologija Interneta stvari je dobar primjer toga. Ona predstavlja revoluciju Interneta koja ubrzano raste potpomognuta napretkom tehnologije bežičnih komunikacija, mobilnih uređaja, senzora, računalnog oblaka i sl. **Internet stvari** (engl. *Internet of Things*, IoT) omogućava povezivanje različitih stvari, objekata iz našeg okruženja putem Interneta u inteligentne informacijske mreže za prikupljanje i analiziranje informacija vezanih za korisnika i njegovu okolinu. Povezivanje objekata koji tradicionalno nisu asocirani sa Internetom poput termostata, pumpi, kućanskih aparata, automobilskog motora i sl., te njihovo praćenje i upravljanje u stvarnom vremenu, omogućava novu razinu donošenja inteligentnih odluka na temelju obrade podataka. Rezultat je dakako optimizacija sustava, procesa i usluga, ušteda vremena te općenito poboljšanje kvalitete života. Tehnologija Interneta stvari se može primijeniti u raznim domenama svakodnevnog života, a jedna od primjena je i udaljeno praćenje mjerenja senzora što je detaljnije objašnjeno ovim radom.

Drugo poglavlje ovog rada daje teorijsku pozadinu korištenih tehnologija – pregled koncepta i značajki tehnologije Interneta stvari te Docker kontejnerske tehnologije. Treće poglavlje definira demo korisnički slučaj za koji će se primijeniti tehnologija Interneta stvari i pokriva pregled (analizu) dostupnih rješenja, na temelju čega se odabire IoT platforma za definirani IoT scenarij. Četvrto poglavlje pokriva korake primjene odabrane IoT platforme poput lokalne instalacije, stvaranja IoT modela te objašnjava interakciju sa kreiranom platformom – rezerviranje IoT entiteta, kreiranje relacija te slanje različitih zahtjeva specifičnih za demo *use-case*. Objašnjen je i postupak slanje i dohvaćanje podataka sa IoT platforme uz praktične primjere pri čemu su razmotreni različiti protokoli. Također je razmotrena i implementacija LoRa tehnologije za povezivanje uređaja te je objašnjen i testiran jedan od načina integracije LoRa mreže i odabrane IoT platforme.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je napraviti pregled koncepta i značajki tehnologije Interneta stvari (IoT), istražiti dostupna postojeća rješenja (IoT platforme) te primijeniti tehnologiju Interneta stvari kroz scenarij praćenja mjerenja udaljenih senzora korištenjem odabrane IoT platforme u okvirima *demo* korisničkog slučaja.

2. KORIŠTENE TEHNOLOGIJE

Ovim poglavljem se daje teorijska pozadina korištenih tehnologija pri rješavanju zadatka diplomskog rada. Prvenstveno, to je pregled koncepta i značajki tehnologije Interneta stvari, arhitekture, relevantnih komunikacijskih protokola i komunikacijskih rješenja za umrežavanje uređaja. Dodatno, objašnjena je i Docker tehnologija kao danas najpopularnija kontejnerske tehnologije koja se zbog svoje efikasnosti i praktičnosti često primjenjuje pri instalaciji IoT platformi.

2.1. Internet stvari

Internet stvari (engl. *Internet of Things*, skraćeno IoT) se može definirati kao dinamička globalna mrežna infrastruktura sa mogućnostima samokonfiguriranja koja se temelji na standardiziranim interoperabilnim komunikacijskim protokolima gdje fizičke i virtualne „stvari“ imaju jedinstvene identitete, fizičke atribute i virtualne osobnosti, koriste inteligentna sučelja i integrirani su u informacijsku mrežu putem Interneta pri čemu razmjenjuju podatke vezane za korisnike i njihovo okruženje, što omogućuje međusobnu interakciju između različitih sustava te razvoj novih usluga i inteligentnih aplikacija pomoću kojih je moguće učiniti različite domene ljudskog djelovanja „pametnijima“ [1]. Ovakvom definicijom tehnologije Interneta stvari uviđaju se temeljne karakteristike IoT-a poput:

- **Dinamičnost i adaptivnost** – IoT sustavi imaju mogućnost prilagodbe i poduzimanja određenih akcija na temelju promjene dinamičnih uvjeta u kojima se nalaze. Primjer: promjena načina rada sustava za video nadzor ovisno o dobu dana – normalno ili infracrveno noćno snimanje, promjena rezolucije pri detekciji pokreta.
- **Mogućnost samokonfiguriranja** – uređaji u IoT mreži imaju mogućnost samostalnog postavljanja mreže, ažuriranja i slično, bez intervencije korisnika.
- **Interoperabilni komunikacijski protokoli** – podrška različitih komunikacijskih protokola za razmjenu i korištenje informacija te komunikaciju uređaja sa IoT infrastrukturom.
- **Jedinstveni identiteti** – IoT uređaji jedinstveno su određeni svojim jedinstvenim identitetima – IP adresom ili URI-jem (*Uniform Resource Identifier*) putem kojeg je moguće vršiti upite, pratiti status i udaljeno upravljati uređajima u skladu sa konfiguracijom IoT infrastrukture.

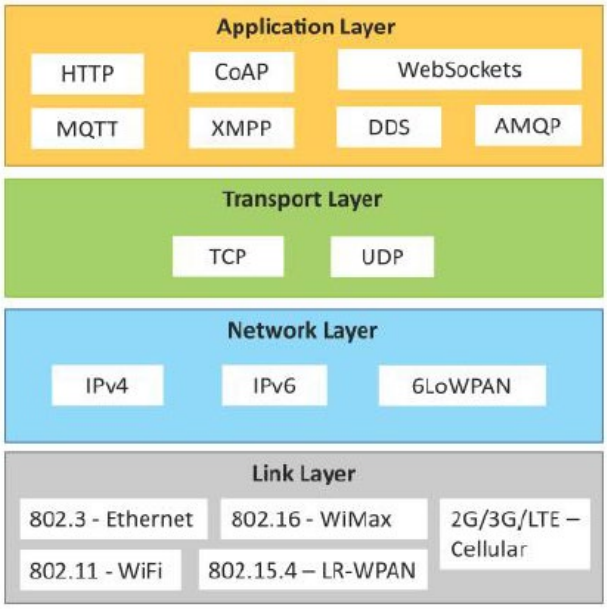
- **Integriranost u informacijsku mrežu** – uređaji su integrirani u informacijsku mrežu koja im omogućava komunikaciju i razmjenu informacija sa drugim uređajima i sustavima što IoT sustave čini „pametnijima“ budući da uređaji zajednički prikupljaju veliku količinu podataka koji se zatim mogu procesirati i analizirati (npr. vremenske stanice za predikciju vremenskih uvjeta). [1]

Dakle, osnovni koncept *Interneta stvari* je povezati stvari, fizičke objekte iz korisnikovog okruženja u inteligentnu, informacijsku mrežu, radi prikupljanja različitih podataka i njihove razmjene što uključuje slanje podataka putem Interneta na centralizirane servere gdje se vrši filtriranje, kategoriziranje i analiza podataka te se iz njih izvlače korisne informacije o različitim sustavima, operacijama, procesima i okruženju u cilju donošenja inteligentnih odluka i akcija, povećanja efikasnosti, optimizacije i automatizacije procesa, uvođenja novih usluga što sve generalno vodi ka poboljšanju kvalitete života.

Stvar ili objekt u navedenim definicijama i karakteristikama tehnologije označava uređaj, fizički objekt iz stvarnog svijeta, jedinstvenog identiteta, opremljen različitim senzorima, aktuatorima, upravljačem, softverom, I/O sučeljima, te mrežnim sučeljem (mrežna povezanost) što mu omogućuje slanje prikupljenih podataka (inputa) na Internet za skupljanje i procesiranje te razmjenu informacija sa drugim uređajima; uređaji sa aktuatorima (npr. relej) mogu, na temelju prikupljenih i analiziranih podataka izvršavati određene zadatke (outpute) i međudjelovati sa fizičkom okolinom.

IoT uređaje, njihova sučelja i načine povezivanja na centralizirani server za prikupljanje i obradu podataka, definira **fizički dizajn** IoT infrastrukture, kao i IoT protokole za različite slojeve TCP/IP mrežnog modela komunikacije (Slika 2.1) – aplikacijski, transportni, mrežni te sloj linka.

Na **sloju linka** (fizički sloj i sloj mrežnog pristupa – PHY i MAC) su definirani protokoli koji određuju na koji način se podaci šalju preko fizičkog sloja (bakrene žice, koaksijalnim kablom ili bežično radio valovima) odnosno kako su paketi kodirani i signalizirani određenim hardverom za slanje preko fizičkog medija i pružanje pristupa mreži. U kontekstu IoT-a, bitni su sljedeći protokoli sloja linka: **IEEE 802.3 Ethernet**, **802.11 WiFi**, **802.16 WiMax**, **802.15.4 LR-WPAN** (kojeg koristi i **ZigBee** tehnologija) ali i komunikacijski protokoli mobilnih tehnologija **2G** (GSM, CDMA), **3G** (UMTS, CDMA2000), **4G** (LTE) a uskoro i **5G**. Bitna tehnologija sloja linka je i **LoRaWAN** (*Long Range Wide Area Network*) tehnologija sa definiranom LoRa modulacijom za PHY sloj te LoRaWAN protokolom za MAC sloj. Više o LoRaWAN tehnologiji se nalazi u poglavlju 4.6.



Slika 2.1 *IoT protokoli prema TCP/IP mrežnom modelu [1]*

Mrežni/Internet sloj je zadužen za slanje **IP** (*Internet Protocol*) datagrama iz jedne u drugu mrežu pri čemu vrši adresiranje i rutiranje (usmjeravanje) paketa. Identifikacija adresa se temelji na **IPv4** (32-bitne adrese) i **IPv6** shemama (128-bitne). Osim IPv4 i IPv6 protokola, na ovom sloju se nalazi i **6LoWPAN** (*IPv6 over Low power Wireless Personal Area Network*) koji predstavlja IP protokol za uređaje male snage koji imaju ograničenu procesorsku moć.

Transportni sloj je zadužen za prijenos paketa s kraja na kraj neovisno o nižim slojevima mreže. Tu se izdvajaju:

- **TCP** (*Transmission Control Protocol*) – danas najupotrebljavaniji transportni protokol budući da je korišten kod HTTP-a za web promet, SMTP-a za email poštu, FTP-a za prijenos datoteka i sl. Konekcijski je orijentiran, osigurava pouzdan prijenos paketa u zadanom slijedu, nudi retransmisiju, detekciju pogreški, kontrolu toka i sl.
- **UDP** (*User Datagram Protocol*) – nije konekcijski orijentiran već transakcijski (nije potrebna uspostava veze), ne garantira dostavu ni redoslijed paketa ali je zbog malog zaglavlja koristan u stvarno-vremenskim aplikacijama gdje je kašnjenje bitnije od same dostave paketa.

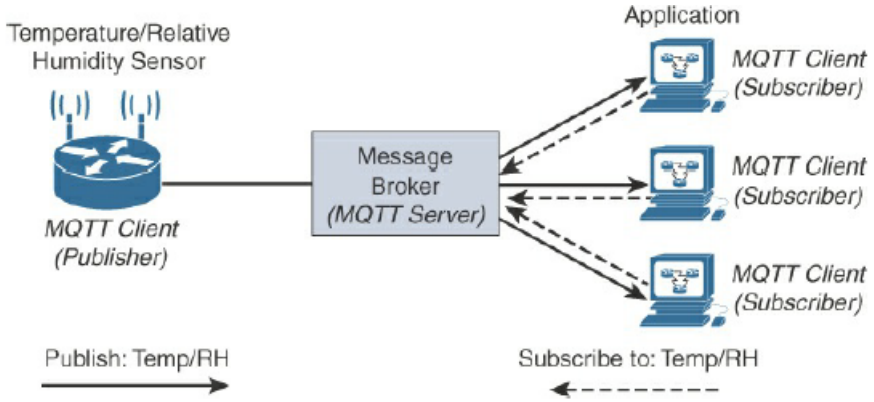
Aplikacijski sloj definira način na koji aplikacije komuniciraju sa protokolima nižih slojeva pri slanju podataka mrežom (između udaljenih komponenti IoT sustava). Protokoli aplikacijskog sloja šifriraju podatke aplikacija te ih enkapsuliraju u transportne protokole koji omogućavaju konekcijski ili transakcijski orijentiranu komunikaciju za prijenos podataka kroz

mrežu. Protokoli aplikacijskog sloja omogućavaju vezu između procesa koristeći portove. Neki od najbitnijih, IoT relevantnih protokola ovog sloja su:

- **HTTP** (Hypertext Transfer Protocol) – protokol aplikacijskog sloja koji je temelj *World Wide Web*-a (WWW). Uključuje naredbe poput: GET, POST, PUT, DELETE, TRACE, OPTIONS itd. Protokol se bazira na zahtjev-odgovor (engl. ***request-response***) modelu u kojem HTTP **klijent** (*Internet* pretraživač, aplikacija na IoT uređaju, mobilna aplikacija i sl.) šalje HTTP zahtjeve na **server**, koristeći prethodno spomenute HTTP naredbe. Za identifikaciju resursa, HTTP koristi jedinstvene identifikatore URI-je a za prijenos paketa koristi TCP kao transportni protokol. Standardni port za HTTP promet je port 80.

  HTTP kao protokol za komunikaciju sa IoT uređajima (naročito u ograničenim mrežama) nije optimalan budući da ne omogućava da se iste poruke distribuiraju većem broju komponenti IoT sustava te kao protokol unosi značajan višak pri razmjeni podataka u odnosu na samu dužinu korisnih podataka (*payload*). Stoga su se razvili drugi protokoli aplikacijskog sloja koji su bolje prilagođeni IoT tehnologiji.
- **CoAP** (Constrained Application Protocol) – protokol aplikacijskog sloja za M2M (*machine-to-machine*) aplikacije namijenjen ograničenim okruženjima – ograničene mreže sa uređajima ograničenih resursa (engl. *constrained networks, constrained devices*). Kao i HTTP, CoAP također koristi *zahtjev-odgovor* model klijenta i servera, URI-je za resurse, podržava naredbe kao što su GET, POST, PUT, DELETE, no na transportnom sloju ne koristi TCP već UDP - klijent komunicira sa serverom bezkonekcijskim datagramima i baziran je na razmjeni kratkih poruka - kratko zaglavlje (engl. *overhead*). Standardni port za CoAP protokol je port 5683. CoAP komunikacija u IoT infrastrukturi može biti direktna između klijenta (IoT uređaja) i servera, preko pristupnika ili putem HTTP-CoAP proxyja i slično.
- **WebSocket** – protokol aplikacijskog sloja koji omogućava potpunu dvosmjernu (*full-duplex*) komunikaciju poslužitelja i klijenta putem jedne TCP veze (*socket*) koja se uspostavlja i drži otvorenom što omogućava tok podataka u oba smjera. Radi na portovima 443 i 80 te je kompatibilan sa HTTP-om.
- **MQTT** (Message Queue Telemetry Transport) – jednostavni protokol aplikacijskog sloja za razmjenu podataka koji se temelji na *publish-subscribe* modelu (model objave i pretplate) gdje klijent (npr. IoT uređaj) objavljuje (*publish*) poruke na određenu temu (engl. *topic*) na poslužitelju koji se naziva MQTT *broker* (posrednik) i koji zatim

prosljeđuje te poruke klijentima koji su pretplaćeni (*subscribe*) na tu temu. Kao i CoAP protokol, MQTT se dobro uklapa u ograničenim okruženjima gdje je mrežna propusnost mala, a uređaji imaju ograničene memorijske resurse i procesorsku moć. Standardni port za MQTT protokol je 1883. Sljedeća slika ilustrira MQTT komunikaciju:



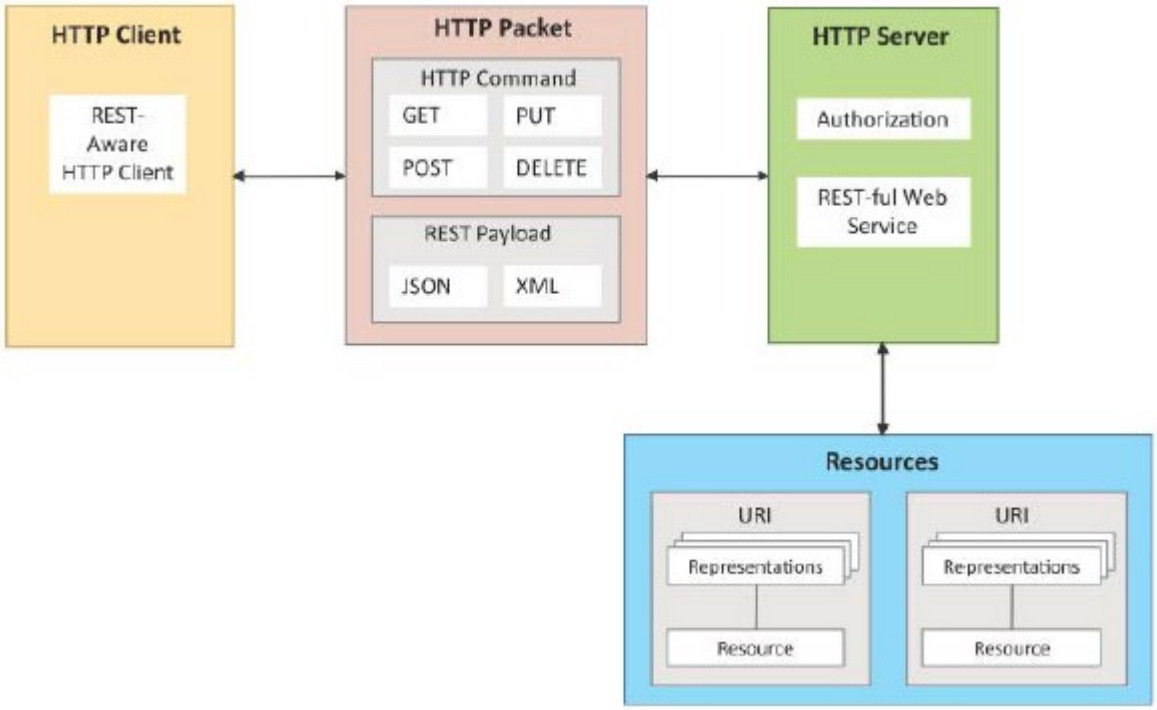
Slika 2.2 MQTT komunikacija [3]

- **AMQP** (Advanced Message Queuing Protocol) – otvoreni protokol aplikacijskog sloja za razmjenu poslovnih poruka ([1], str. 28). Podržava prijenos podataka od točke do točke i model objave i pretplate (*publish-subscribe*) sa rutiranjem i redovima za obradu (engl. *queues*). AMQP broker (server) sadrži centralu (engl. *exchange*) koja prima poruke od klijenta – *publisher* (različitih aplikacija koje generiraju podatke), te rutira (usmjerava) te podatke u odgovarajuće izlazne redove za obradu odakle se šalju pretplaćenim klijentima ili klijentima koji zatraže te podatke iz redova (*pull*). Usmjeravanje poruka se vrši na temelju veze (engl. *binding*) koju definira aplikacija a postoji nekoliko različitih tipova poput direktne, razgranate, veza na temelju teme (engl. *topic*), zaglavlja i sl. Standardni post za AMQP je port 5672.
- **XMPP, DDS** i dr.

Promatrajući sastavnice fizičkog dizajna Interneta stvari, jasno je da se IoT ne može promatrati kao jedna zasebna tehnologija već kao širi skup različitih koncepata, protokola i tehnologija. Tehnologiju Interneta stvari možemo promatrati i na višoj, apstraktnoj razini bez razmatranja detalja implementacije fizičkog dizajna IoT-a. Drugim riječima, IoT možemo razmatrati kroz **logički dizajn** kojeg definiraju različite funkcije u IoT sustavima. Prema tome, Internet stvari se sastoji od sljedećih komponenti: **uređaj**, **resursi**, **upravljačka usluga**, **baza podataka**, **web usluga**, **komponenta analize podataka** te **aplikacija**. [1]

Navedene komponente IoT-a uključuju i različite **komunikacijske modele** (poput već spomenutih *publish-subscribe* modela, *request-response* modela te *push-pull* komunikacijskog modela) te **komunikacijske API**-je koji se dijele na dvije vrste: API-ji temeljeni na **REST** arhitekturi te API-ji temeljeni na **WebSocket**-u.

REST (Representational State Transfer) – označava skup principa prema kojima se mogu dizajnirati web usluge i web API-ji, a temelje su na resursima sustava te načinu njihovog adresiranja (označavanja) i prijenosa. REST API-ji su u skladu sa *request-response* komunikacijskim modelom pri čemu svaki zahtjev klijenta na poslužitelju sadrži sve potrebne informacije za razumijevanje zahtjeva – nema čuvanja stanja tj. zahtjevi su međusobno nezavisni (engl. *stateless*). Na sljedećoj slici prikazana je komunikacija klijenta i poslužitelja koristeći REST API:



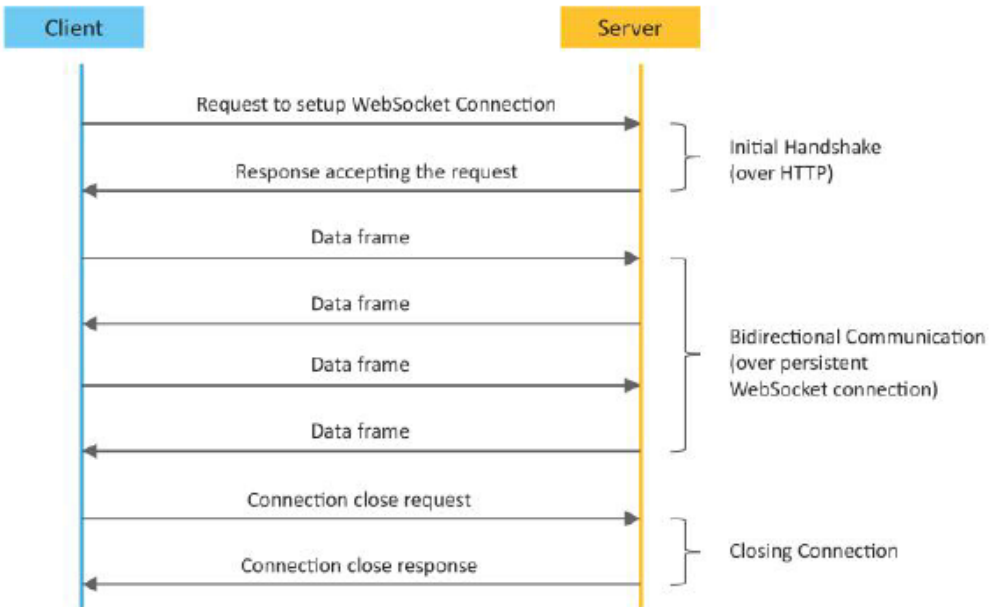
Slika 2.3 *Komunikacija klijenta i poslužitelja koristeći REST API [1]*

REST arhitektura također podrazumijeva i ograničenja poput slojevitog sustava (određena komponenta vidi samo trenutni sloj s kojim je u interakciji), ograničenje priručne memorije (engl. *cache)* i jedinstvenog sučelja (engl. *uniform interface*) kojim REST poslužitelj vraća reprezentacije resursa koje su konceptualno odvojene od samih resursa a sadrže sve potrebne informacije za izvršavanje određenih radnji nad traženim resursom (UPDATE, DELETE i slično).

RESTful web usluge, dakle, označavaju web API-je implementirane koristeći HTTP i REST principe i kao takve predstavljaju kolekcije resursa koji su određeni URI-jima na koje klijent

šalje zahtjeve koristeći metode definirane HTTP protokolom (GET, POST, PUT, DELETE). RESTful web usluge podržavaju različite formate podataka (*Internet media types*) među kojima su **XML** i **JSON** najpopularniji.

WebSocket API-ji imaju potpunu dvosmjernu komunikaciju (*full-duplex*) koja započinje inicijalnim HTTP zahtjevom tzv. WebSocket *handshake* kojim se uspostavlja WebSocket veza i ostaje uspostavljena za obostranu razmjenu poruka klijenta i servera pri čemu se smanjuje mrežni promet i kašnjenje (engl. *latency*). WebSocket API koristi *exclusive pair* model komunikacije.



Slika 2.4 *WebSocket model komunikacije [1]*

Skup tehnologija na koje se oslanja Internet stvari je velik a osim već navedenih ovim poglavljem, treba spomenuti i mreže bežičnih senzora, računalni oblak (engl. *cloud computing*), analitika velikih količina podataka (engl. *big data analytics*) te ugrađeni sustavi (engl. *embedded systems*). To su računalni sustavi koji se sastoje od mikroupravljača, memorije (ROM, RAM), mrežnog adaptera, I/O sučelja i skladišta (engl. *storage*), koji imaju određenu namjenu za razliku od osobnih računala opće namjene te su u odnosu na njih slabiji ali i znatno jeftiniji. Primjeri takvih uređaja su Arduino, Raspberry Pi, pcDuino, BeagleBone Black, Cubieboard te razni gotovi komercijalni IoT uređaji.

Tehnologija Interneta stvari se može primijeniti u širokom opsegu domena ljudskog života poput energetike, zaštite okoliša, logistike, industrije, poljoprivrede, zdravlja, u kućanstvima za „pametnu“ rasvjetu i kućanske aparate, u gradovima za „pametne“ sustave parkiranja, smanjenje gužvi u prometu i slično.

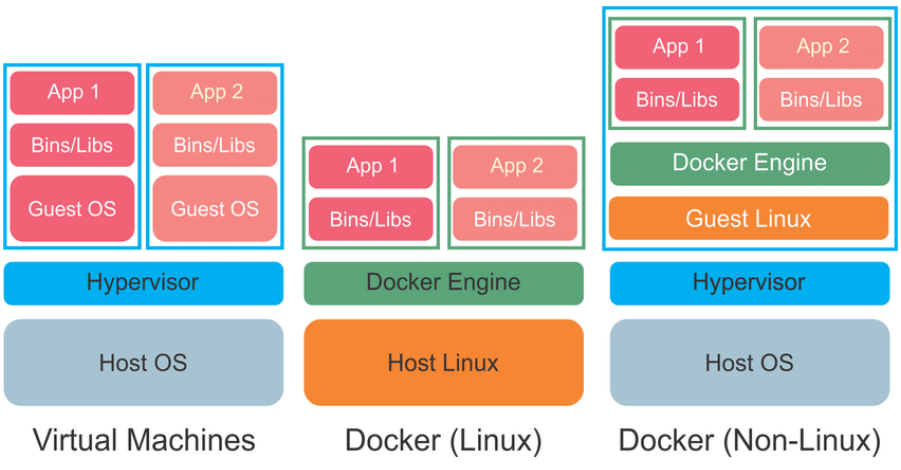
2.2. Docker tehnologija

Docker je najpopularnija i danas najviše korištena kontejnerska tehnologija i platforma za razvoj softvera te je de-facto standard za kreiranje, upravljanje i distribuciju softverskih kontejnera; u cilju objašnjavanja Docker (kontejnerske) tehnologije, potrebno je ponajprije objasniti što su zapravo *kontejneri*.

Slika 2.5 *Docker logo [4]*

Kontejneri (engl. *containers*) su izolirana računalna okruženja koja sadrže sve potrebne ovisnosti (engl. *dependencies*) i datoteke nužne za razvoj i pokretanje određene aplikacije (softvera) te predstavljaju oblik virtualizacije na nivou operacijskog sustava (OS). Docker kontejnerska tehnologija omogućava pokretanje više nezavisnih kontejnera unutar jedne Linux instance pri čemu kontejneri oslanjaju na funkcionalnost operacijskog sustava *hosta* (Linux jezgre) što omogućava njihovo brzo pokretanje i efikasnije trošenje resursa.

Kontejneri izgledaju kao „lagana“ virtualna računala iako je izvedba kontejnerske tehnologije znatno drugačija od klasične virtualizacije. Za **virtualna računala** (engl. *Virtual Machines*, *VMs*) potreban je odgovarajući hipervizor te instalacija (emulacija) operacijskog sustava za svako pojedino virtualno računalo (*Guest OS*). Zbog toga su virtualna računala više izolirana ali i znatno kompleksnija te zauzimaju znatno više resursa od kontejnera koje pokreće isti Linux operacijski sustav (OS) te su resursi OS-a dijeljeni između više kontejnera. Kontejnerima se dakle ne distribuira cijeli sustav već samo korisnička aplikacija/servis i minimalni skup ovisnosti (engl. *dependencies*) koja se onda može pokrenuti na bilo kojem računalu/serveru. Sljedeća slika prikazuje usporedbu klasične virtualizacije i Docker kontejnerske tehnologije:

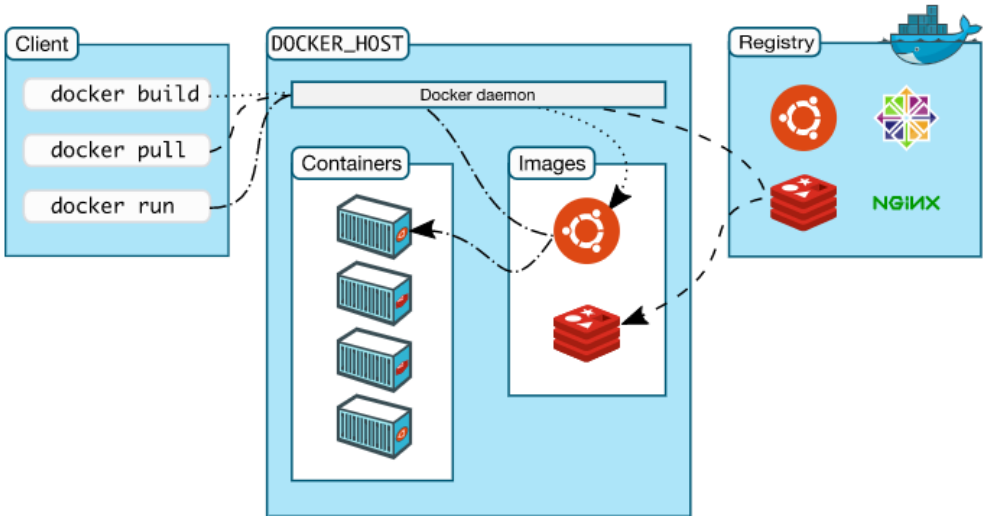


Slika 2.6 *Usporedba VM i Docker kontejnera [5]*

U slučaju da operacijski sustav domaćina nije Linux (npr. *Windows*), potrebna je virtualizacija (VM) mini-Linux *guest* OS-a budući da Docker radi sa Linux operacijskim sustavom. Taj virtualni Linux OS je onda domaćin (host) za pokretanje Docker kontejnera (Slika 2.6, treći stupac). Pokretanje Docker kontejnera omogućuje ***Docker Engine***. To je zapravo klijentsko-poslužiteljska (engl. *client-server*) aplikacija koja se sastoji od:

- **poslužitelja** (*Docker daemon*) koji stvara, pokreće i distribuira Docker kontejnere a zadužen je i za upravljanje Docker objektima poput Docker slika (engl. *Docker image*), mrežama i podatkovnim prostorom (engl. *data volumes*),
- **klijenta** koji je zapravo komandno-linijsko sučelje (engl. *command-line interface,* CLI) pomoću kojeg korisnik komunicira sa Dockerom (npr. naredba *docker run* za pokretanje kontejnera) putem **REST API**-ja za interakciju Docker klijenta i poslužitelja.

Dakle, klijent putem CLI-a šalje npr. naredbu *docker run* poslužitelju (Docker deamon) koji preko REST API-ja osluškuje naredbe klijenta; poslužitelj će zatim dohvatiti traženu Docker sliku i na temelju nje, pokrenuti kontejner. Docker slika je zapravo predložak za kreiranje kontejnera i obično se temelji na drugim slikama sa određenim izmjenama i konfiguracijama potrebnim za pokretanje određene aplikacije. Ukoliko slika nije dostupna lokalno, Docker deamon će dohvatiti (*docker pull*) sliku sa vanjskog registra za Docker slike; to je uobičajeno *Docker Hub* na kojem se nalazi velik broj Docker slika različitih sustava i aplikacija poput Ubuntu, Node.js, Redis, NGINX, MySQL, PostgreSQL i sl.



Slika 2.7 *Docker arhitektura [4]*

Zbog svoje efikasnosti, Docker kontejneri također omogućavaju znatno veće skaliranje aplikacija i predstavljaju budućnost razvoja softvera te su jedna od najznačajnijih inovacija u IT industriji u zadnjih nekoliko godina.

3. ODABIR IOT PLATFORME

Istraživanjem i analizom dostupnih postojećih rješenja, dobivena su dva kandidata odnosno dvije IoT platforme – **Thingsboard IoT** te **Mainflux IoT** – čije su karakteristike opisane u ovom poglavlju te je također napravljena i njihova usporedba u cilju odabira IoT platforme koja će se primijeniti za zadani IoT scenarij odnosno korisnički slučaj (engl. *use-case*) koji je definiran u nastavku.

3.1. Demo korisnički slučaj

Za primjenu IoT platforme odabran je sljedeći korisnički slučaj – praćenje kvalitete vode za različite izvore pomoću IoT tehnologije. Potrebno je konfigurirati određenu IoT platformu kako bi se na nju mogli slati, spremati te dohvaćati podaci mjerenja kvalitete vode – *fizikalna* svojstva vode: **temperatura**, **zamućenost**, **provodljivost** te **pH vrijednost**. Kvaliteta vode se obično iskazuje mjerenjem *kemijskih* svojstava vode, no, trenutni IoT senzori jednostavnije mjere i prate fizikalne parametre vode. Dakle, budući da su fizikalna svojstva jednostavnija za mjeriti tržišno-dostupnim IoT uređajima (senzorima), u sklopu ovog korisničkog slučaja, kao parametri kvalitete vode odabrana su njena fizikalna svojstva: temperatura, zamućenost, provodljivost i pH vrijednost.

Vlasnik sustava je određena tvrtka koja daje drugim klijentima, tvrtkama, organizacijama pristup monitoringu kvalitete vode – dohvaćanje podataka mjerenja. Klijenti mogu imati svoje korisnike koji kroz aplikacije i uređaje šalju podatke za različite točke mjerenja kvalitete vode (*water points*). To su zemljopisne točke određene lokacijom (koordinatama) te je moguće za svaku takvu točku slati podatke mjerenja kvalitete vode od strane korisnika. Ti podaci se onda mogu dohvatiti za svaki izvor (točku) te se time dobiva informacija o kvaliteti vode za pojedini izvor kroz određeno vremensko razdoblje. Dakle, krajnji korisnik ima uvid u podatke mjerenja za različite zemljopisne točke koje se prikazuju na karti, odnosno, korisnik može promatrati kvalitetu vode određenog zemljopisnog područja u određenom vremenskom razdoblju.

Točke mjerenja kvalitete vode pripadaju određenom širem području – gradu koji predstavlja skup više točaka mjerenja kvalitete vode. Grad također pripada određenom širem području – regiji koja obuhvaća više gradova nekog zemljopisnog područja.

Mjerenja kvalitete vode moguće je poslati za postojeće *water points* ali također, korisnik može i stvoriti/dodati novu točku mjerenja kvalitete vode te zatim poslati svoja mjerenja za tu točku.

3.2. Thingsboard

3.2.1. Opis i karakteristike platforme

Thingsboard je *open-source* IoT platforma za upravljanje uređajima te prikupljanje, obradu i vizualizaciju podataka. Omogućuje brzi razvoj, upravljanje i skaliranje IoT projekata. [6]

Postoje dva izdanja – Thingsboard *Community Edition* koje će se razmatrati u nastavku te Thingsboard *Professional Edition*.

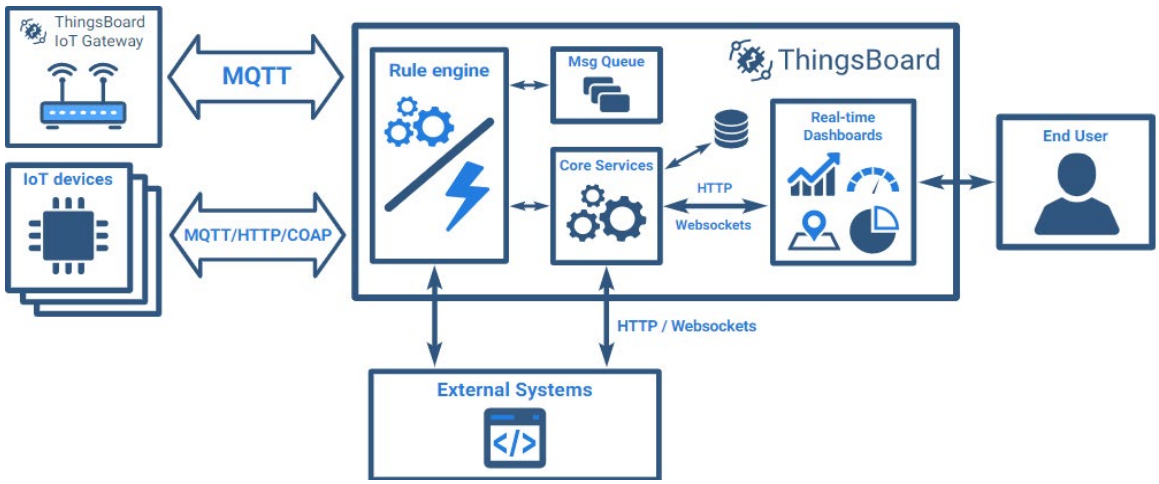
Slika 3.1 *Thingsboard logo [6]*

Povezivanje uređaja je omogućeno putem standardnih IoT protokola – **MQTT** (engl. *Message Queuing Telemetry Transport*), **HTTP** (engl. *Hypertext transfer protocol*) te **CoAP** (engl. *Constrained Application Protocol*).

Instalacija platforme moguća je na dva načina:

- **lokalno** na osobnom računalu - preporuka korištenja kontejnerske tehnologije – *Docker* (*CaaS – container as a service*) te
- u **oblaku** (engl. c*loud deployment*) – *Live Demo (PaaS – platform as a service)* kojem se može pristupiti putem korisničkog računa na stranici: https://demo.thingsboard.io

Arhitekture Thingsboard IoT platforme prikazana je na sljedećoj slici:



Slika 3.2 *Arhitektura Thingsboard IoT platforme [6]*

Thingsboard IoT platforma sadrži velik broj API-ja za upravljanje IoT entitetima i njihovim odnosima, prikupljanje i vizualizaciju podataka te nudi i mogućnost procesiranja i transformiranja ulaznih podataka i događaja u stvarnom vremenu definiranjem određenih pravila i funkcija *Rule*

Engine-a (*data processing rule chains*) kao i kreiranje alarma na temelju određenih aktivnosti i slično.

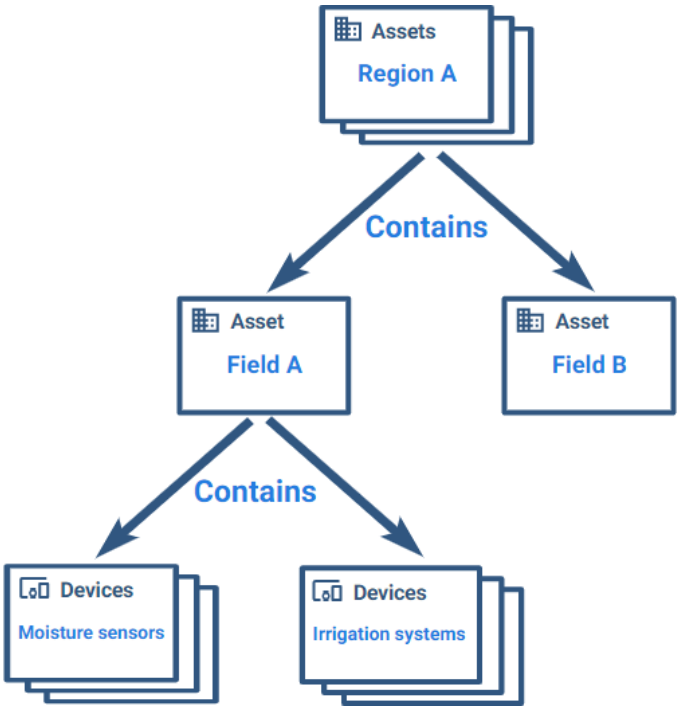
Navedene funkcionalnosti moguće je koristiti putem grafičkog sučelja Thingsboard **Web UI** (*user interface*) koje dolazi u sklopu platforme te putem **REST API**-a.

3.2.2. Upravljanje entitetima i njihovim odnosima

Thingsboard CE (*Community Edition*) pruža nekoliko različitih tipova entiteta:

- **Tenants** (stanari) – odvojeni poslovni entiteti – organizacija ili pojedinci koji posjeduju uređaje i mogu imati velik broj klijenata
- **Customers** (klijenti) – također mogu biti poslovni entiteti – organizacija ili pojedinci koji koriste sustav i uređaje stanara te mogu imati više korisnika
- **Users** (korisnici) – korisnici sustava koji konzumiraju podatke sa platforme – pregled kontrolnih ploča
- **Devices** (uređaji) – osnovni IoT entiteti koji proizvode podatke ili reagiraju na određene naredbe (senzori, aktuatori i slično)
- **Assets** (imovina) – apstraktni IoT entiteti koji se vežu za određene uređaje i/ili druge entitete. To mogu biti zgrade koje sadrže više stanova ili npr. automobil, zemljopisno područje i slično.
- **Alarms** (alarmi) – događaji koji ukazuju na određene aktivnosti ili probleme sa uređajima i ostalim entitetima sustava
- **Dashboards** (kontrolne ploče) – grafičko sučelje za vizualizaciju IoT podataka, mjerenja, poruka koja pruža i mogućnost upravljanja određenim uređajima
- **Rule Node** (čvor pravila) – jedinice za procesiranje ulaznih poruka, događaja i sl.
- **Rule Chain** (lanac pravila) – skup povezanih čvorova pravila koji definira ponašanje sustava

Za navedene entitete moguće je definirati **relacije** – usmjerene veze između entiteta (*contains, manages, owns* i sl.), **atribute** – parovi ključeva i vrijednosti (engl. *key-value pairs*) te poslati **telemetrijske podatke** – podatkovne točke vremenskog niza (engl. *time-series data points*) dostupne za pohranu, upite i vizualizaciju. Dodatno, *assets* i *devices* također mogu imati tip koji se definira i služi za međusobno razlikovanje. Sljedeća slika ilustrira primjer hijerarhije imovine i uređaja te njihovih odnosa:



Slika 3.3 *Primjer Thingsboard hijerarhije [6]*

Thingsboard CE nudi tri razine (uloge) korisnika platforme – **stanari** (engl. *tenants*), **klijenti** (engl. *customers*) te **korisnici** (engl. *users*). Entitet stanara ima jednog ili više administratora stanara (*tenant admin*) koji ima dozvolu stvarati nove *assets* na platformi te uređivati njihove odnose. Stoga, autorizacija pri stvaranju novih uređaja i assets-a se vrši putem tokena koji pripada korisniku sa ulogom **TENANT_ADMIN**.

Administratorski račun za stanare (engl. *tenant admin account*) se stvara od strane administratora sustava (engl. *system administrator*). Dakle, korisnik sa ulogom admina stanara (*TENANT_ADMIN role*) određen je mail adresom (*username*) i lozinkom (*password*). Te podatke koristi za prijavu a kao odgovor na prijavu dobiva se JSON Web Token (JWT). Taj token, koji će se koristiti u svim zahtjevima gdje je potrebna autorizacija korisnika, može se dobiti slanjem sljedećeg POST zahtjeva na **/api/auth/login**:

```
curl -X POST --header 'Content-Type: application/json' --
header 'Accept: application/json' -d
'{"username":"tenant@thingsboard.org", "password":"tenant"}'
'http://THINGSBOARD_URL/api/auth/login'
```

U navedenom zahtjevu *tenant@thingsboard.org* je demo račun admina stanara.

THINGSBOARD_URL se odnosi na vrstu instalacije Thingsboard platforme te za lokalnu instalaciju potrebno ga je zamijeniti sa ***localhost:broj_porta*** (npr. localhost:8080) a za live-demo server (*cloud* instalaciju) zamjenjuje se sa ***demo.thingsboard.io***.

Vrijeme isticanja autorizacijskog tokena (JWT *expiration*) se može podesiti u konfiguracijskoj datoteci platforme ***thingsboard.yml***, na način da se postavi vrijednost polja JWT_TOKEN_EXPIRATION_TIME na željeni broj (u sekundama).

Dakle, za autorizaciju zahtjeva potrebno je postaviti sljedeće zaglavlje (engl. *header*) pomoću dohvaćenog tokena pri prijavi:

```
X-Authorization: Bearer JWT_TOKEN
```

Za stvaranje novog *asset*-a potrebno je poslati POST zahtjev na **/api/asset**:

```
curl -v -X POST -d '{"name":"Field C","type":"field"}'
http://THINSGBOARD_URL/api/asset \
--header "Content-Type:application/json" \
--header "X-Authorization: Bearer JWT_TOKEN"
```

Kao rezultat slanja zahtjeva, u odgovoru platforme će biti jedinstveni identifikator kreiranog *asset*-a (ASSET_ID). Zahtjev za stvaranje novog uređaja je analogan prethodnom zahtjevu samo što se koristi druga krajnja točka (engl. *API endpoint*) na koju se upućuje zahtjev: **/api/device**.

Za stvaranje relacije između entiteta koji su određeni identifikatorima, potrebno je poslati POST zahtjev na /**api/relation**. Smjer relacije se definira sa ***from*** i ***to***:

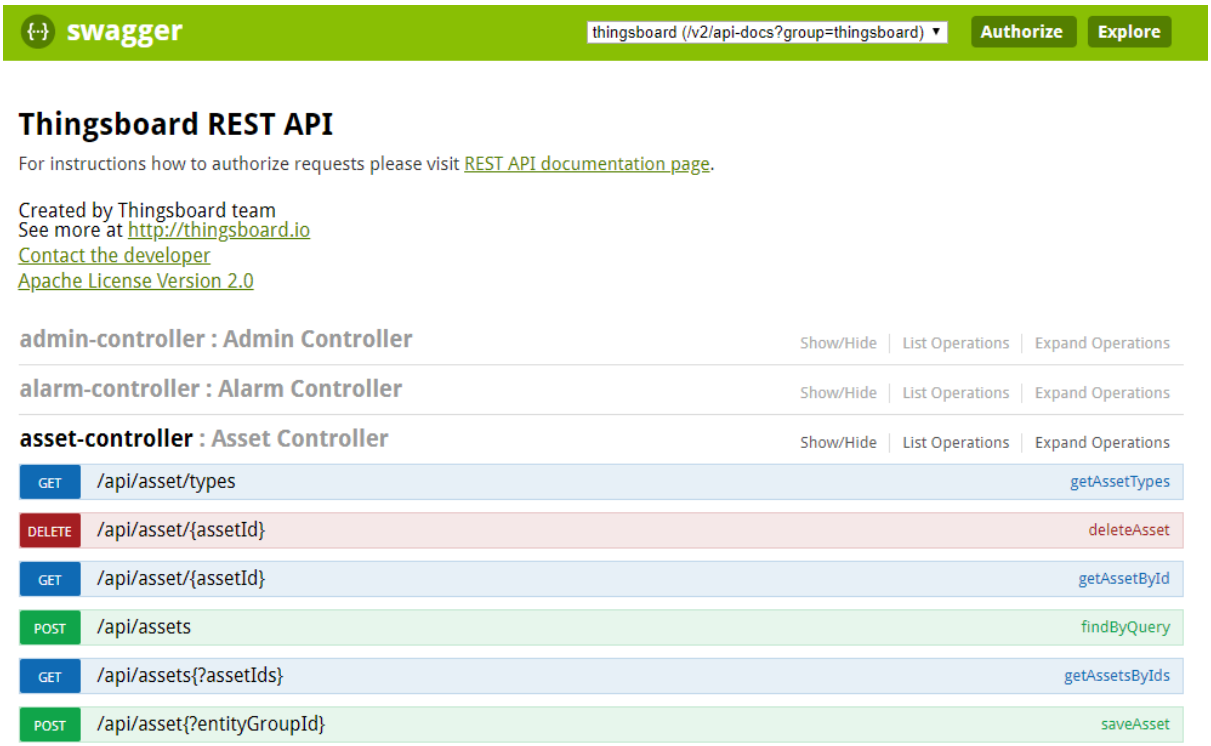
```
curl -v -X POST -d
'{"from":{"id":"$FROM_ASSET_ID","entityType":"ASSET"},"type":"
Contains","to":{"entityType":"ASSET","id":"$TO_ASSET_ID"}}'
http://THINSGBOARD_URL/api/relation \
--header "Content-Type:application/json" \
--header "X-Authorization: Bearer JWT_TOKEN"
```

U navedenom zahtjevu entitet sa identifikatorom $FROM_ASSET_ID će sadržavati (tip relacije: *Contains*) entitet $TO_ASSET_ID.

Za pregledavanje svih pristupnih točaka Thingsboard REST API-a može se koristiti *Swagger* UI – sučelje za vizualizaciju i interakciju sa dostupnim resursima i krajnjim točkama određenog API-a. Nakon instalacije Thingsboard servera, *Swagger* UI se može otvoriti koristeći sljedeći URL:

```
http://THINGSBOARD_URL/swagger-ui.html
```

Sljedeća slika prikazuje *Swagger* UI za Thingsboard REST API:



Slika 3.4 *Swagger UI za Thingsboard REST API [7]*

Na ovaj način moguće je kreirati zahtjeve za komunikaciju sa Thingsboard platformom te posložiti hijerarhiju uređaja i *assets*-a koja će odgovarati određenom IoT scenariju.

3.2.3. Slanje i dohvaćanje podataka

Thingsboard nudi bogat skup značajki vezanih za rad sa telemetrijskim podacima među kojima su:

- Prikupljanje podataka sa uređaja korištenjem MQTT, CoAP ili HTTP protokola
- Spremanje podataka vremenskog niza (engl. *timeseries data*) u bazu podataka (Cassandra, PostgreSQL)
- Vršenje upita i dohvaćanje zadnjih vrijednosti ili svih vrijednosti unutar specificiranog vremenskog intervala
- Preplata na promjene podataka pomoću *websockets*-a (za stvarno-vremensku vizualizaciju i analizu)
- Vizualizacija podataka vremenskog niza putem podesivih kontrolnih ploča i *widgets*-a
- Filtriranje i analiza podataka korištenjem fleksibilnog *Rule Engine*-a
- Generiranje alarma na temelju prikupljenih podataka itd.

Slanje podataka (i atributa) od strane uređaja na Thingsboard server odvija se tako da se šalju **parovi ključeva i vrijednosti** (engl. *key-value pairs*) unutar jednog ili više JSON objekata a vrijednosti mogu biti *string, boolean, double* i *long*. Primjer JSON objekta koji se može poslati kao telemetrijski podatak:

```
{"stringKey":"val1", "booleanKey":true, "doubleKey":5.1, "longKey":73}
```

API za slanje telemetrijskih podataka (engl. *Telemetry upload API*) se razlikuje u pristupnoj točki za različite protokole koji se koriste; u slučaju **HTTP** i **CoAP** protokola za slanje telemetrijskih podataka na Thingsboard server potrebno je poslati POST zahtjev na sljedeći *endpoint*:

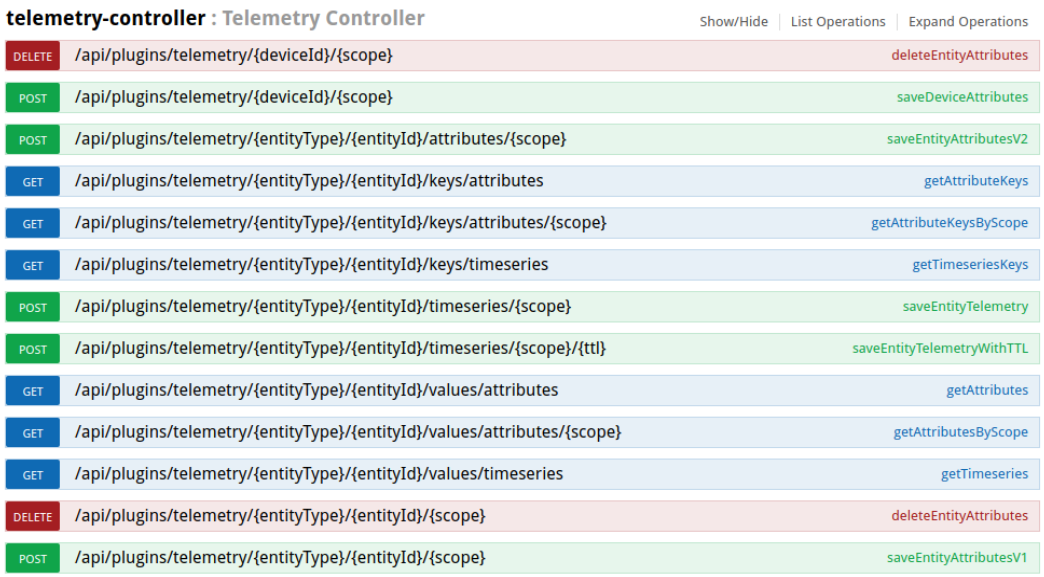
```
/api/v1/$ACCESS_TOKEN/telemetry
```

$ACCESS_TOKEN za navedeni URL se odnosi na autorizaciju uređaja pri čemu svaki uređaj ima svoj pristupni token za autorizaciju i potrebno ga je uključiti pri slanju telemetrijskih podataka uređaja. U slučaju **MQTT** protokola za slanje telemetrijskih podataka na Thingsboard server potrebno je poslati POST zahtjev na sljedeći *endpoint*:

```
v1/devices/me/telemetry
```

$ACCESS_TOKEN kod MQTT protokola se koristi pri slanju MQTT CONNECT poruke sa korisničkim imenom koje sadrži navedeni token.

Thingsboard telemetrijski servis također pruža REST API za slanje i dohvaćanje telemetrijskih podataka (i atributa) za bilo koji entitet:



Slika 3.5 *Thingsboard Telemetry Service REST API [7]*

Dakle, moguće je poslati telemetrijske podatke npr. za određeni *asset* (određen sa $ASSET_ID) slanjem sljedećeg POST zahtjeva:

```
curl -X POST -d '{"temperature":12.4,"humidity":76}'
http://THINGSBOARD_URL/api/plugins/telemetry/ASSET/$ASSET_ID \
/timeseries/{scope} \
--header "Content-Type:application/json" \
--header "X-Authorization: Bearer JWT_TOKEN"
```

Vidljivo je da se kod ovog načina slanja podataka autorizacija obavlja putem JWT tokena koji se dohvati pri prijavi administratorskog računa stanara u čijem je vlasništvu *asset* za kojeg šaljemo podatke.

Na sličan način moguće je dohvatiti spremljene podatke npr. za određeni *asset* ($ASSET_ID); u ovom slučaju potrebno je poslati sljedeći GET zahtjev:

```
curl -X GET
http://THINSGBOARD_URL/api/plugins/telemetry/ASSET/$ASSET_ID \
/values/timeseries \
--header "X-Authorization: Bearer JWT_TOKEN"
```

Navedeni zahtjev će dohvatiti zadnje vrijednosti svih ključeva za specificirani *asset*. Ukoliko se žele dohvatiti sve vrijednosti u određenom vremenskom intervalu, GET zahtjev ima sljedeći oblik gdje je potrebno kao parametre poslati nazive ključeva (*temperature, humidity*) te vrijednost početnog i krajnjeg trenutka (*start i end timestamp*):

```
curl -X GET
http://THINSGBOARD_URL/api/plugins/telemetry/ASSET/$ASSET_ID \
/values/timeseries?keys=temperature,humidity \
&startTs=$TS1&endTs=$TS2 \
--header "X-Authorization: Bearer JWT_TOKEN"
```

3.3. Mainflux

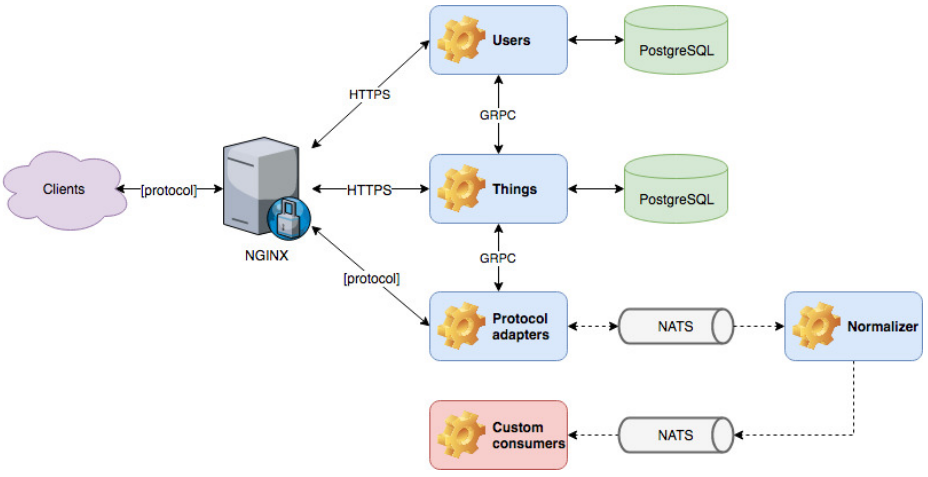
3.3.1. Opis i karakteristike platforme

Mainflux je moderna, skalabilna, sigurna i otvorenog izvora (*open-source*) IoT platforma napisana u programskom jeziku *Go*, a služi za povezivanje uređaja putem raznih mrežnih protokola, kao IoT posrednik pri razvoju složenih IoT sustava. [8]

Podržava različite mrežne protokole – **HTTP**, **MQTT**, **WebSocket** i **CoAP** te također podržava i premošćivanje navedenih protokola (engl. *protocol bridging*). Za pohranjivanje podataka nudi tri različite opcije odnosno baze podataka: ***Cassandra***, ***InfluxDB*** i ***MongoDB***. Kao i većina modernih IoT platformi, nudi instalaciju putem *Docker* kontejnerske tehnologije. Također podržava i integraciju sa LoRaWAN mrežama. **Arhitektura** Mainflux platforme se bazira na sljedećim uslugama odnosno servisima:

- *users*: za upravljanje korisnicima platforme i autentikaciju
- *things*: za upravljanje uređajima (*stvarima*), kanalima te pravom pristupa
- *normalizer*: za normaliziranje SenML (*Sensor Measurement List*) poruka
- *http-adapter*: HTTP sučelje za pristup komunikacijskim kanalima
- *mqtt-adapter*: MQTT sučelje za pristup komunikacijskim kanalima
- *ws-adapter*: WebSocket sučelje za pristup komunikacijskim kanalima
- *coap-adapter*: CoAP sučelje za pristup komunikacijskim kanalima
- *lora-adapter*: za prosljeđivanje i integraciju na LoRa server
- *mainflux-cli*: komandno-linijsko sučelje

Na sljedećoj slici prikazana je arhitektura Mainflux IoT platforme:



Slika 3.6 *Arhitektura Mainflux IoT platforme [8]*

Za instalaciju putem kontejnerske tehnologije potrebno je imati instaliran Docker i Docker Compose programski paket. Za pokretanje Mainflux platforme kao docker kompozicije potrebno je, zatim, unutar *mainflux* direktorija dohvaćenog sa GitHub-a (github.com/mainflux/mainflux) izvršiti sljedeću CLI naredbu:

```
make run
```

Nakon izvršenja naredbe, kreiraju se svi potrebni Docker kontejneri te je instanca Mainflux IoT platforme podignuta i može joj se pristupiti putem mainflux-cli komandno-linijskog sučelja ili putem REST API-a na adresi *http://localhost/* za upravljanje entitetima i njihovim odnosima te slanje i dohvaćanje podataka.

3.3.2. Upravljanje entitetima i njihovim odnosima

Mainflux platforma se temelji na tri glavna entiteta: ***users***, ***things*** i ***channels***. Entitet *user* odnosno korisnik, predstavlja stvarnog korisnika sustava putem korisničkog računa - e-mail i lozinka – koje koristi za dobivanje pristupnog tokena (engl. *access token*). Kada je korisnik prijavljen, može upravljati resursima – stvarima i kanalima (*things* i *channels*) putem CRUD metoda te definirati politike pristupa.

Entitet *thing* odnosno stvar, predstavlja uređaj ili aplikaciju, koji su spojeni na Mainflux platformu u cilju razmjene poruka sa drugim 'stvarima'. Entitet *channel* predstavlja komunikacijski kanal odnosno temu (engl. *topic*) za razmjenu poruka od strane *stvari* koje su spojene na taj kanal.

Za kreiranje korisničkog računa potrebno je na Mainflux API poslati sljedeći zahtjev:

```
curl -X POST -H "Content-Type: application/json" https://localhost/users -d
'{"email":"john.doe@email.com", "password":"123"}'
```

Zatim, za dohvaćanje pristupnog tokena (autorizacijskog ključa), potrebno je poslati sljedeći POST zahtjev na */tokens endpoint*:

```
curl -X POST -H "Content-Type: application/json" https://localhost/tokens -d
'{"email":"john.doe@email.com", "password":"123"}'
```

U odgovoru na poslani zahtjev nalazi se traženi token kojeg je potrebno ugraditi u autorizacijsko zaglavlje pri slanju zahtjeva za rezerviranje odnosno upravljanje entitetima IoT platforme.

```
{ "token": "<user_auth_token>" }
```

```
-H "Authorization: <user_auth_token>"
```

Za upravljanje entitetima *things* šalje se POST zahtjev na ***/things*** *endpoint* sa određenim JSON podacima:

```
curl -X POST -H "Content-Type: application/json" -H "Authorization: <user_auth_token>"
https://localhost/things -d '{"name":"weio"}'
```

U zaglavlju *Location* odgovora na poslani zahtjev nalazi se identifikator kreirane 'stvari' koja može predstavljati uređaj ili aplikaciju.

Dohvaćanje i uklanjanje entiteta je sukladno CRUD operacijama – GET, DELETE itd. Za upravljanje entitetima ***channels*** koriste se zahtjevi analogni zahtjevima za 'stvari' samo što se u ovom slučaju šalju na ***/channels*** *endpoint*.

Kontrola pristupa se određuje odnosom odnosno povezanošću stvari (uređaja i aplikacija) sa kanalima. Kanali se mogu smatrati komunikacijskim grupama stvari. Samo stvari koje su spojene na određeni kanal mogu slati i primati podatke od drugih stvari na tom kanalu. Stvari i određeni kanal povezuje korisnik koji je i kreirao taj kanal i to slanjem sljedećeg zahtjeva:

```
curl -X PUT -H "Authorization: <user_auth_token>"
https://localhost/channels/<channel_id>/things/<thing_id>
```

Za raskidanje veze potrebno je poslati DELETE zahtjev sa istim podacima.

3.3.3. Slanje i dohvaćanje podataka

Mainflux kao temelj razmjene poruka koristi **NATS** – *open-source* sustav za razmjenu poruka, a kao format poruka se preporučuje **SenML** format kako bi se poruke odnosno podaci mogli naknadno procesuirati i normalizirati.

Mainflux podržava slanje i dohvaćanje poruka putem različitih transportnih protokola – HTTP, WebSocket, MQTT te CoAP. Također nudi i podteme za razmjenu poruka (engl. *subtopic*) za određeni kanal kako bi razmjena podataka bila prilagodljivija određenim IoT scenarijima. Krajnja točka (engl. *endpoint*) za poruke je */channels/<channel_id>/messages* na koju se mogu nadovezati i podteme, poput: *messages/building1/temperature*.

Dakle, kada je rezerviran entitet kanala i kada su određene stvari spojene na njega, moguće je poslati podatke (poruke) na taj kanal putem različitih protokola. Za slanje podataka putem HTTP protokola potrebno je poslati sljedeći POST zahtjev:

```
curl -X POST -H "Content-Type: application/senml+json" -H "Authorization: <thing_token>"
https://localhost/http/channels/<channel_id>/messages -d
'[{"n":"voltage","u":"V","v":120.1}, {"n":"current","t":-5,"v":1.2}]'
```

U autorizacijskom zaglavlju navedenog zahtjeva šalje se pristupni token stvari koja šalje podatke odnosno objavljuje poruku na odabrani kanal (<*channel_id*>). Struktura odnosno format poruke je u skladu sa SenML formatom, gdje "n" označava naziv (engl. *name*) određene mjerene veličine, a "v" izmjerenu vrijednost (engl. *value*). Moguće je definirati i mjernu jedinicu, vrijeme (timestamp) i slično.

Mainflux podržava različite baze podatka za spremanje poslanih poruka: CassandraDB, MongoDB te InfluxDB. Također, implementira sustav pisača-čitača (engl. *writers-readers*) koji se zasebno instaliraju (kao dodaci, *add-ons*) prema želji korisnika. Pisači su servisi koji konzumiraju normalizirane Mainflux poruke (u SenML formatu) i pohranjuju ih u određene baze podataka. Tako na primjer, sljedeća naredba instalira i pokreće InfluxDB bazu podataka vremenskih nizova, InfluxDB pisač koji sprema poruke u spomenutu bazu te Grafana – alat za rad sa bazom podataka, njihovu vizualizaciju i analitiku:

```
docker-compose -f docker/addons/influxdb-writer/docker-compose.yml up -d
```

Za navedeni primjer moguće je instalirati i InfluxDB-reader – čitač, odnosno servis za konzumiranje Mainflux poruka iz baze podataka koji izlaže HTTP API za dohvaćanje spremljenih poruka odnosno podataka. Sljedećom naredbom instalira se spomenuti čitač te otvara HTTP API za dohvaćanje poruka na portu **8905**:

```
docker-compose -f docker/addons/influxdb-reader/docker-compose.yml up -d
```

Tada je moguće dohvatiti spremljene Mainflux podatke slanjem sljedećeg GET zahtjeva:

```
curl -H "Authorization: <thing_token>" http://localhost:8905/channels/<channel_id>/messages
```

Važno je napomenuti da stvar (npr. određena aplikacija) koja želi dohvatiti podatke sa određenog kanala (<*channel_id*>) mora biti spojena na taj kanal. Stvar koja dohvaća podatke je određena pristupnim tokenom (<*thing_token*>) u autorizacijskom zaglavlju navedenog zahtjeva.

3.4. Usporedba platformi

Analizirana dva IoT rješenja – **Thingsboard** i **Mainflux** – potrebno je usporediti kako bi se odabrala odgovarajuća platforma koja će se primijeniti za zadani demo IoT scenarij. Pri odabiru IoT platforme, kriterij nisu bolje karakteristike određene platforme, već bolja pokrivenost svih zahtjeva demo korisničkog slučaja, odnosno, koja platforma bolje odgovara zadanom IoT scenariju. Budući da je riječ o demonstrativnom scenariju, značajke poput broja mogućih korisnika ili povezanih uređaja na platformi ili učestalost poruka u jedinici vremena, ne igraju ključnu ulogu pri odabiru platforme. Ono što se razmatra pri odabiru su mogućnosti pri rezerviranju entiteta odnosno stvaranju hijerarhije entiteta i njihovih odnosa te način i format slanja i dohvaćanja podataka.

Pregledom opisanih karakteristika pojedine platforme dolazi se do zaključka da obje podržavaju glavne i najčešće korištene transportne protokole u IoT-u: HTTP i MQTT. Također, obje platforme nude Docker instalaciju koja olakšava i ubrzava proces podizanja instance platforme. Obje platforme pružaju i mogućnost integracije sa uređajima vanjskih mreža poput LoRaWAN mreže.

Obje platforme nude i suvremene baze podataka za spremanje poruka poput Cassandra baze podataka, pri čemu treba istaknuti da Mainflux ima donekle slobodniji način dohvaćanja podataka gdje je moguće instalacijom određenih čitača (*readers*) konfigurirati odnosno mijenjati način interpretacije spremljenih podataka tj. format dohvaćenih poruka. No, to znači i složeniji odnosno stroži format slanja (unosa) podataka kod Mainflux platforme koji treba biti u skladu sa SenML formatom; Thingsboard platforma kod spremanja podataka ima veću slobodu te je sam način slanja podataka jednostavniji, budući da je potrebno samo unijeti određenu izmjerenu veličinu kao par ključ-vrijednost (engl. *key-value pair*). Razlika je vidljiva na sljedećem primjeru za slanje izmjerene temperature:

Mainflux:

```
{
      "n": "temperature",
      "v": 12.4
}
```

Thingsboard:

```
{
      "temperature": 12.4
}
```

Autorizacija korisnika je slična kod obje platforme i obavlja se na standardni način dohvaćanjem određenog pristupnog tokena te zatim slanjem tog tokena u zaglavlju određenih zahtjeva za koje je potrebna autorizacija. Ono što predstavlja glavnu razliku između dvije odabrane platforme je rezerviranje entiteta i upravljanje njihovim odnosima na platformi (*provisioning*).

Mainflux IoT model se temelji na tri stavke: korisnici, stvari i kanali te međusobnoj povezanosti navedenih stavki. Razmatrajući mogućnosti pri rezerviranju entiteta na Mainflux platformi, s obzirom na zadani demo korisnički slučaj (3.1), dolazi se do sljedećeg mogućeg IoT modela odnosno hijerarhije entiteta:

- *users* ili entiteti korisnika bi definirali tvrtku koja daje pristup monitoringu kvalitete vode te eventualne klijente koji bi bili korisnici te usluge. Nedostatak je nemogućnost definiranja razine prava za klijente, što znači da bi se koristio jedan administratorski račun tvrtke pružatelja usluge (vlasnika sustava).
- *things* odnosno stvari bi predstavljale fizičke uređaje sa senzorima za mjerenje kvalitete vode za različite točke mjerenja, koji šalju svoja mjerenja u SenML formatu te se ta mjerenja (poruke) spremaju na Mainflux platformu.
- *channels* odnosno kanali bi služili za povezivanje stvari (uređaja) i određenih točaka mjerenja (lokacija) u smislu definiranja regije/grada/određene točke mjerenja kvalitete vode kao određene teme/podteme na kanalu što definitivno nije optimalan način za definiranje IoT modela za potrebe zadanog slučaja. Na primjer:
  `/channels/<channel_id>/messages/cityA/spring1`

Jasno je dakle, da Mainflux, budući da je generalno jednostavnija i 'lakša' platforma od Thingsboard-a, ima manje mogućnosti za definiranje odgovarajuće IoT modela za zadani demo korisnički slučaj. Thingsboard pruža više mogućnosti pri definiranju hijerarhije modela putem entiteta ASSETS, DEVICE i njihovim relacijama dok se kod Mainflux-a proces konfiguriranja IoT modela svodi samo na povezivanje uređaja (stvari) preko kanala sa odgovarajućim temama i podtemama (*topics*, *subtopics*).

Thingsboard platforma, dakle, nudi više slobode i mogućnosti pri konfiguriranju IoT modela odnosno stvaranju potrebne hijerarhije entiteta i odnosa koja bolje odgovara zadanom demo korisničkom slučaju – mogućnost definiranja regija, gradova, točaka mjerenja i njihove lokacije te uređaja za slanje mjerenja kvalitete vode. Također, format slanja i dohvaćanja podataka mjerenja je donekle jednostavniji i u potpunosti odgovara zadanom IoT scenariju.

Uzevši u obzir sve prethodno navedene stavke, **Thingsboard IoT platforma** je odabrana kao rješenje koje će se primijeniti za zadani demo IoT scenarij.

4. PRIMJENA IOT PLATFORME

Odabrana IoT platforma – **Thingsboard** će se primijeniti za prikupljanje podataka sa senzora u sklopu demo korisničkog slučaja koji uključuje mjerenje kvalitete vode, slanje mjerenja na IoT platformu te zatim dohvaćanje podataka o kvaliteti vode za različita područja - izvore vode (*water points*). Ponajprije potrebno je instalirati IoT platformu – odabrana je **lokalna** instalacija platforme koristeći ***docker*** kontejnere.

4.1. Instalacija Thingsboard IoT platforme

Instalacija Thingsboard IoT platforme će se izvršiti na Windows operacijskom sustavu, stoga je prije same instalacije potrebno skinuti i instalirati *Docker Desktop for Windows* instalacijski paket kojim se instalira Docker razvojno okruženje. Unutar Docker-a je moguće pokretanje *docker* kontejnera na temelju Thingsboard *docker* slike (engl. *image*) koja sadrži sve potrebne komponente (uključujući i bazu podataka) za pokretanje IoT platforme.

Ovisno o bazi podataka koja će se koristiti za IoT platformu postoje tri tipa *docker* slika za pokretanje Thingsboard platforme:

- **thingsboard/tb-cassandra** – Thingsboard instanca sa ***Cassandra*** bazom podataka – preporučena opcija sa najboljim performansama ali zahtijeva minimalno 6GB RAM memorije.
- **thingsboard/tb-postgres** – Thingsboard instanca sa ***PostgreSQL*** bazom podataka. Preporučeno za manje servere sa barem 1GB RAM memorije (dovoljno za nekoliko poruka u sekundi).
- **thingsboard/tb** – Thingsboard instanca sa ugrađenom HSQLDB bazom.

Odabran je drugi tip Thingsboard *docker* slike odnosno **thingsboard/tb-postgres** instanca sa PostgreSQL bazom podataka budući da ne zauzima previše resursa pri radu a nudi dovoljno dobre performanse za potrebe demo korisničkog slučaja.

Dakle, nakon instalacije Docker-a na Windows operacijskom sustavu, potrebno je pokrenuti jedan od komandno-linijskih klijenata (npr. *Powershell*) za unos sljedećih naredbi; za Windows korisnike se preporučuje kreiranje *docker* volumena za podatke i ispise logova (engl. *logs*) što se postiže sljedećim naredbama:

```
$ docker volume create mytb-data
$ docker volume create mytb-logs
```

Sljedeći korak je direktno pokretanje Thingsboard instance koji se sastoji od dohvaćanja (engl. *pull*) željene Thingsboard slike (engl. *image*) sa javnog repozitorija Docker slika ***DockerHub-***a te pokretanja *docker* kontejnera na temelju dohvaćene Thingsboard slike. Navedeno se postiže pokretanjem sljedeće naredbe u terminalu:

```
$ docker run -it -p 9090:9090 -p 1883:1883 -p 5683:5683/udp -v
mytb-data:/data -v mytb-logs:/var/log/thingsboard --name mytb
--restart always thingsboard/tb-postgres
```

Oznakom *–p* povezujemo lokalne portove sa izloženim lokalnim portovima ovisno o protokolu: za HTTP port 9090, za MQTT port 1883 te za COAP protokol port 5683.

Oznaka *–v* povezuje prethodno kreiranje volumene sa internim Thingsboard direktorijima.

Naziv kontejnera je **mytb** te pomoću oznake *–it* pridodajemo terminalu sesiju ispisa pokrenutog Thingsboard procesa.

Naredba pokreće Thingsboard instancu lokalno na računalu kojoj je moguće pristupiti putem web preglednika na adresi:

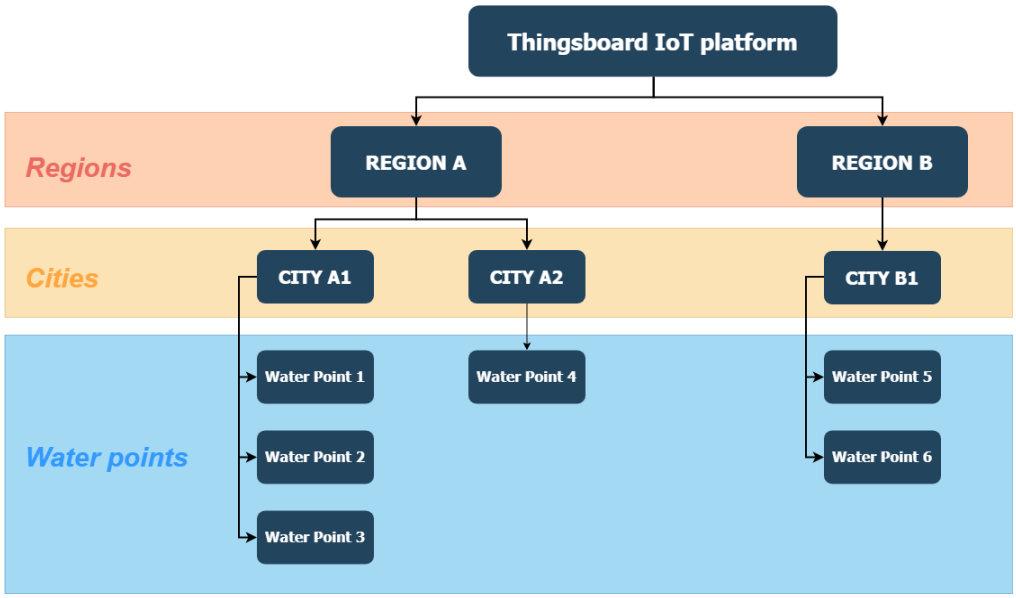
```
http://localhost:9090
```

Navedena adresa ujedno predstavlja i *base-url* za rad sa REST API-jem Thingsboard platforme za potrebe kreiranja i upravljanja IoT entitetima, stvaranje njihovih relacija te prikupljanje i dohvaćanje podataka za zadani korisnički slučaj.

4.2. Stvaranje Thingsboard IoT modela

4.4.1. Definiranje IoT hijerarhije

Na temelju zadanog demo korisničkog slučaja (3.1) potrebno je izraditi i konfigurirati hijerarhiju IoT entiteta na odabranoj Thingsboard IoT platformi koja će obuhvatiti sve zahtjeve zadanog korisničkog slučaja. **Rezerviranje** (engl. *provisioning*) IoT entiteta na Thingsboard platformi radi se putem REST API-a, a za slanje i rad sa REST zahtjevima koristi se REST klijent programskog paketa *Postman* te Thingsboard Web UI. Na sljedećoj slici prikazana je potrebna hijerarhija za opisani korisnički slučaj:



Slika 4.1 *Thingsboard IoT model (hijerarhija) za demo use-case*

Kao što je vidljivo na slici iznad, potrebni Thingsboard model se sastoji od više regija koje se sastoje od jednog ili više gradova; svaki grad (područje) može imati jednu ili više točaka mjerenja kvalitete vode – *water point*.

Regije se predstavljaju Thingsboard entitetima ASSET tipa REGION a **gradovi** Thingsboard entitetima ASSET tipa CITY; na taj način moguće ih je diferencirati. Tu treba napomenuti i postojanje relacija između navedenih entiteta, gdje određene regije **sadrže** (engl. *contains*) određene gradove – relacija između regije i grada je dakle *jedan prema više*.

Točke mjerenja kvalitete vode (*water points*) se predstavljaju Thingsboard entitetima DEVICE tipa WATERPOINT te će se za ove entitete slati podaci mjerenja kvalitete vode. Između entiteta koji predstavljaju gradove i entiteta koji predstavljaju točke mjerenja vode postoje relacije – određeni gradovi **sadrže** određene *water points* koje se, između ostalog, određuju svojom lokacijom (*latitude, longitude*).

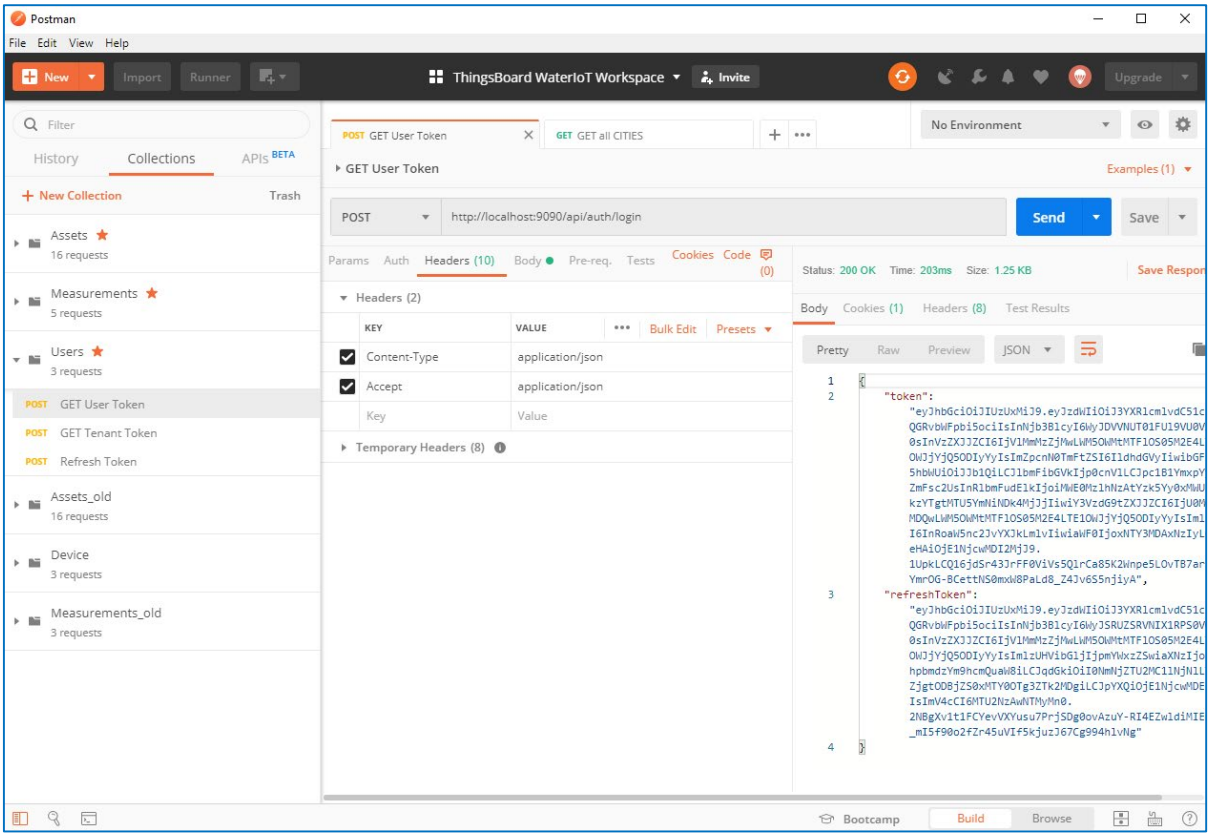
Kao što je opisano u poglavlju 3.1, za određenu točku mjerenja kvalitete vode potrebno je poslati i spremiti sljedeće podatke mjerenja kvalitete vode – **provodljivost**, **temperatura**, **zamućenost** te **pH vrijednost**. Na temelju toga definirani su sljedeći ključevi za podatke mjerenja (*telemetry keys*):

- ***conductivity*** *[µS/cm]*
- ***temperature*** *[°C]*
- ***turbidity*** *[NTU]*
- ***pH***

Za navedene ključeve će se slati vrijednosti mjerenja kvalitete vode za pojedinu *water point* a tip vrijednosti za sva mjerenja je *double* (decimalni zapis). Potrebni atributi za svaki *water point* su ***latitude*** i ***longitude*** – *double* vrijednosti WGS84 koordinata.

Nakon instalacije i pokretanja platforme te definiranja potrebne hijerarhije entiteta IoT modela prema unaprijed opisanom korisničkom slučaju, slijedi rezerviranje IoT entiteta te definiranje njihovih relacija slanjem REST zahtjeva na platformu, pri čemu, svaki zahtjev treba biti autoriziran.

Slanje zahtjeva je obavljeno putem *Postman* REST klijenta (Slika 4.2) unutar kojega se definiraju kolekcije HTTP zahtjeva za rad sa platformom u okvirima zadanog IoT scenarija. Klijentske aplikacije koje će koristiti IoT platformu oslanjaju se upravo na te kolekcije zahtjeva kako bi se adekvatno povezale sa odabranom platformom te slale i dohvaćale podatke prema određenoj logici. Tako npr. moguće je razviti klijentsku web aplikaciju koja implementira autorizaciju i prikaz točaka mjerenja kvalitete vode na zemljopisnoj karti te podatke mjerenja senzora za pojedinu točku – temperaturu, provodljivost, zamućenost i pH vrijednost.



Slika 4.2 Postman sučelje

4.4.2. Autorizacija korisnika

Kao što je navedeno u poglavlju 3.2.2, autorizacija pri stvaranju novih entiteta (*assets* i *devices*) se vrši putem JSON Web tokena koji se dohvaća prijavom korisnika sa ulogom **TENANT_ADMIN**. Račun navedenog korisnika predstavlja admina stanara platforme, odnosno, određenu tvrtku koja daje drugim klijentima, tvrtkama i organizacijama pristup monitoringu kvalitete vode; stvoren je od strane administratora sustava te je za ovu primjenu, određen sljedećim korisničkim imenom i lozinkom:

```
"username":"wateriot.tenant@domain.hr",
"password":"WaterIoT2019"
```

Ovi podaci se koriste za prijavu, pri čemu se kao odgovor, dobiva JSON Web Token.

Budući da je riječ o lokalnoj instalaciji Thingsboard platforme, ***baseUrl*** putem kojeg je moguće komunicirati sa podignutom instancom je ***http://localhost:9090***. Dakle, token za autorizaciju zahtjeva se može dohvatiti slanjem sljedećeg **POST** zahtjeva:

POST

http://localhost:9090/**api/auth/login**

zahtjev za dohvaćanja autorizacijskog JWT tokena

Headers

Content-Type	application/json
Accept	application/json

Body

```
{
    "username": "wateriot.user@domain.hr",
    "password": "WaterIoT2019"
}
```

Example Response 200 OK

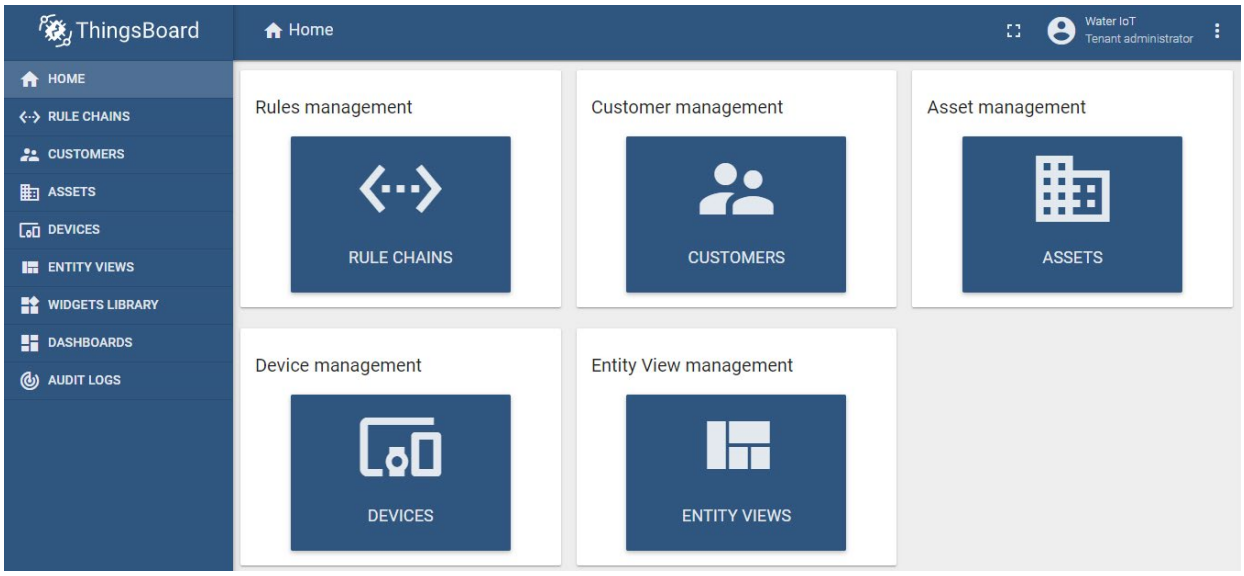
```
{
    "token": "YOUR_JWT_TOKEN",
    "refreshToken": "REFRESH_TOKEN"
}
```

Dohvaćeni JWT token potrebno je ugraditi u **X-Authorization** zaglavlje svih narednih zahtjeva za kreiranje i povezivanje IoT entiteta na sljedeći način:

X-Authorization: Bearer *JWT_TOKEN*

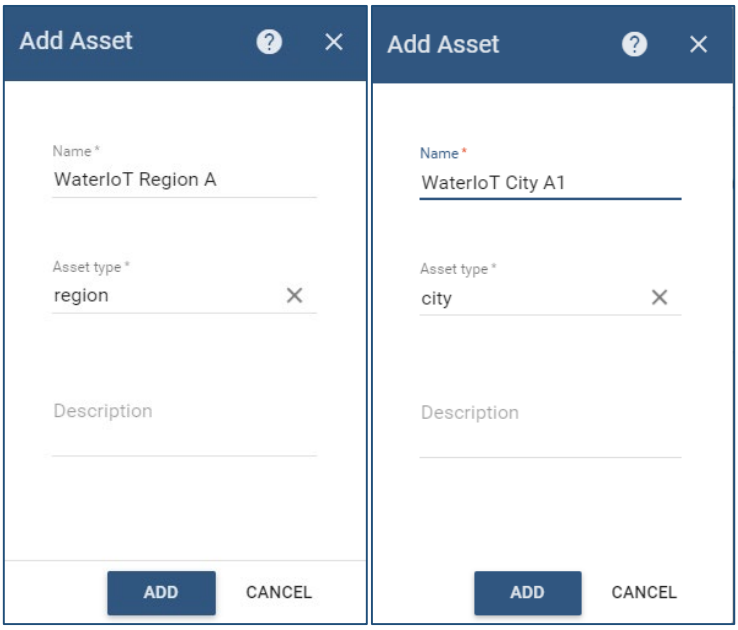
4.4.3. Rezerviranje IoT entiteta i stvaranje relacija

Na temelju postavljene IoT hijerarhije entiteta (Slika 4.1) kreiraju se Thingsboard entiteti (*assets* i *devices*) odgovarajućeg tipa te zatim stvaranje relacija između rezerviranih entiteta. Navedene stavke moguće je izvršiti pomoću Thingsboard Web UI – grafičko sučelje za brzo i jednostavno upravljanje IoT entitetima na platformi koje u pozadini šalje zahtjeve na Thingsboard REST API.



Slika 4.3 *Izgled Thingsboard grafičkog sučelja*

Na sljedećoj slici prikazano je kreiranje entiteta koji predstavlja *regiju* te entiteta koji predstavlja *grad*. Potrebno je unijeti **naziv** entiteta i **tip** (*asset type*).



Slika 4.4 *Stvaranje IoT entiteta putem Thingsboard Web UI*

U pozadini Web UI-a se kreira **POST** zahtjev, u ovom slučaju, za dodavanje novog *asset*-a na platformu. U tijelu zahtjeva se predaju naziv i tip *asset*-a a u zaglavlje autorizacijski token:

POST

http://localhost:9090/**api/asset**

zahtjev za stvaranje novog entiteta (asset)

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

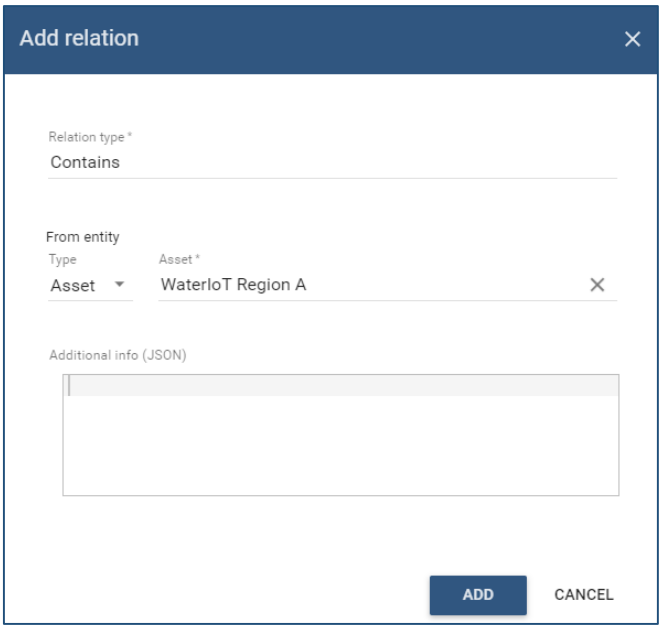
Body

```
{
    "name": "WaterIoT City A2",
    "type": "city",
    "additionalInfo": {
     "latitude": 45.555,
     "longitude": 18.6955
    }
}
```

Example Response 200 OK

```
{
    "id": {
        "entityType": "ASSET",
        "id": "2e041760-8252-11e9-b959-757c1eff178e"
    },
    "createdTime": 1559162508246,
    "additionalInfo": {
        "latitude": 45.555,
        "longitude": 18.6955
    },
    "tenantId": {
        "entityType": "TENANT",
        "id": "3223f4a0-613c-11e9-8010-a32539fe4f3c"
    },
    "customerId": {
        "entityType": "CUSTOMER",
        "id": "13814000-1dd2-11b2-8080-808080808080"
    },
    "name": "WaterIoT City A2",
    "type": "city"
}
```

Kao što je vidljivo u odgovoru na navedeni zahtjev, identifikator stvorenog *asset*-a se nalazi u objektu *„id“* pod ključem *„id“*. Ovako stvoreni *asset* koji predstavlja grad, povezuje se usmjerenom vezom (relacijom) sa odgovarajućom regijom (*asset* tipa region):



Slika 4.5 Stvaranje usmjerene veze grada i regije

Na slici je vidljivo da je tip relacije ***Contains*** kao što je i sugerirano korisničkim slučajem (regija **sadrži** jedan ili više gradova) a usmjerena je **od** (engl. *from*) – regije **prema** (engl. *to*) gradu. Usmjerenje *from* se još naziva *outbound relation*, a usmjerenje *to* se naziva *inbound relation.*

Dakle, za stvaranje usmjerene veze dva entiteta, potrebno je definirati **tip veze** te **usmjerenje veze** – koji entitet (objekt određen identifikatorom i tipom entiteta) je nadređen (*from*) a koji je podređen (*to*). Zahtjev je zatim moguće konstruirati na sljedeći način:

POST

http://localhost:9090/**api/relation**

zahtjev za stvaranje usmjerene veze dva entiteta

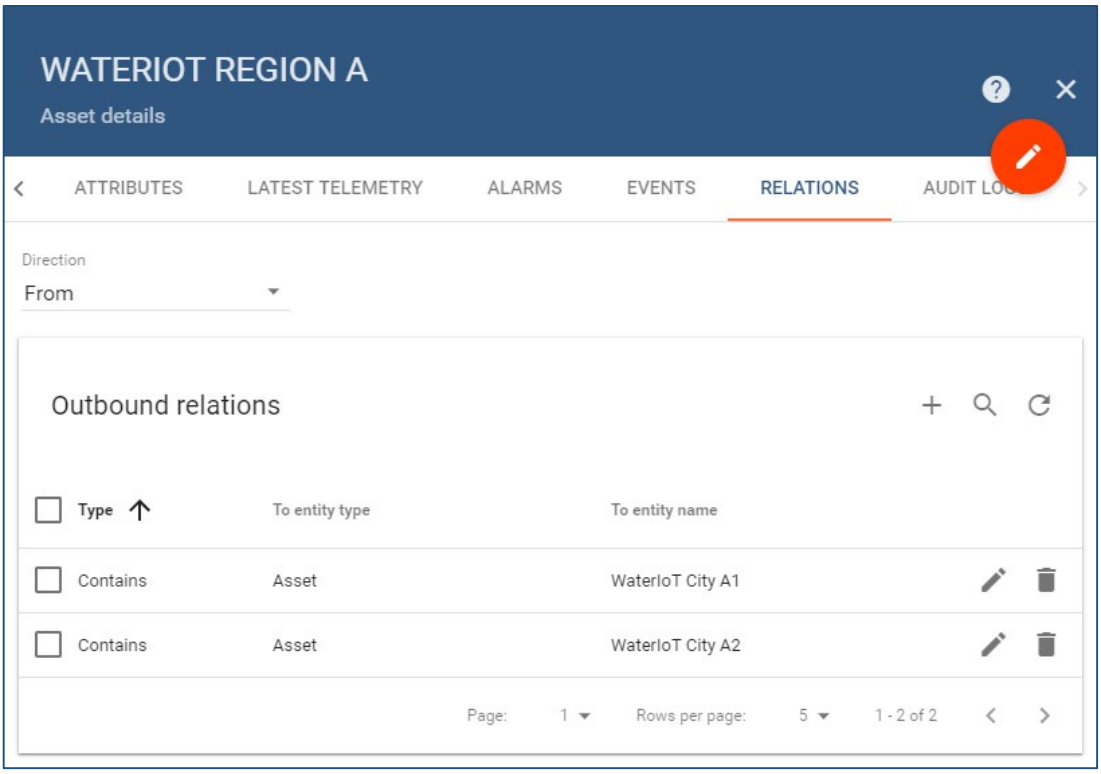
Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
{
    "from": {
        "id": "266c2900-613e-11e9-8010-a32539fe4f3c",
        "entityType": "ASSET"
    },
    "type": "Contains",
    "to": {
        "entityType": "ASSET",
        "id": "2e041760-8252-11e9-b959-757c1eff178e"
    }
}
```

Na sljedećoj slici vidi se rezultat stvaranja navedene relacije regije *WaterIoT Region A* i pripadnog grada *WaterIoT City A2*.



Slika 4.6 *Prikaz relacija za određenu regiju*

Analogno opisanim postupcima, stvaraju se te povezuju i ostali potrebni IoT entiteti prema hijerarhiji entiteta za opisani demo korisnički slučaj. Kao rezultat, na platformi se nalaze dvije regije – regija A koja sadrži gradove A1 i A2 te regija B sa gradom B1. Svaki od ovih gradova ima svoje točke mjerenja kvalitete vode (*water points*).

Sljedeći zahtjevi opisuju **dohvaćanje** entiteta. Za dohvaćanje svih regija, potrebno je poslati GET zahtjev za sve *assets* tipa *region* koji su u vlasništvu određenog stanara. Također, potrebno je kao parametar poslati i *limit* – broj dohvaćenih regija po zahtjevu.

GET

http://localhost:9090/**api/tenant/assets**

zahtjev za dohvaćanje svih regija

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Params

type	region

limit	*100*

Body

none

Example Response 200 OK


```
{
    "data": [
        {
            "id": {
                "entityType": "ASSET",
                "id": "266c2900-613e-11e9-8010-a32539fe4f3c"
            },
            "createdTime": 1555525517200,
            "tenantId": {
                "entityType": "TENANT",
                "id": "3223f4a0-613c-11e9-8010-a32539fe4f3c"
            },
            "customerId": {
                "entityType": "CUSTOMER",
                "id": "13814000-1dd2-11b2-8080-808080808080"
            },
            "name": "WaterIoT Region A",
            "type": "region"
        },
        {
            "id": {
                "entityType": "ASSET",
                "id": "39f1df10-613e-11e9-8010-a32539fe4f3c"
            },
            "createdTime": 1555525549953,
            "tenantId": {
                "entityType": "TENANT",
                "id": "3223f4a0-613c-11e9-8010-a32539fe4f3c"
            },
            "customerId": {
                "entityType": "CUSTOMER",
                "id": "13814000-1dd2-11b2-8080-808080808080"
            },
            "name": "WaterIoT Region B",
            "type": "region"
        }
    ],
    "nextPageLink": null,
    "hasNext": false
}
```

Ukoliko se žele dohvatiti svi gradovi, potrebno je samo promijeniti tip. U odgovoru se nalazi lista sa objektima koji predstavljaju pojedinu regiju, odakle je moguće dobiti jedinstveni identifikator određene regije – ***id*** (REGION_ID). Na temelju njega moguće je dohvatiti sve gradove za određenu regiju preko *endpoint*-a za informacije o relacijama:

GET

http://localhost:9090/**api/relations/info**

zahtjev za dohvaćanje gradova određene regije

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Params

fromType	ASSET
fromId	*REGION_ID*

Body

```
none
```

Example Response 200 OK

```
[
    {
        "from": {
            "entityType": "ASSET",
            "id": "266c2900-613e-11e9-8010-a32539fe4f3c"
        },
        "to": {
            "entityType": "ASSET",
            "id": "8c0cda70-613e-11e9-8010-a32539fe4f3c"
        },
        "type": "Contains",
        "typeGroup": "COMMON",
        "additionalInfo": null,
        "toName": "WaterIoT City A1"
    },
    {
        "from": {
            "entityType": "ASSET",
            "id": "266c2900-613e-11e9-8010-a32539fe4f3c"
        },
        "to": {
            "entityType": "ASSET",
            "id": "2e041760-8252-11e9-b959-757c1eff178e"
        },
        "type": "Contains",
        "typeGroup": "COMMON",
        "additionalInfo": null,
        "toName": "WaterIoT City A2"
    }
]
```

4.3. Stvaranje nove točke mjerenja kvalitete vode

Postupak stvaranja nove točke mjerenja kvalitete vode (*water point*) na platformi se sastoji od stvaranja entiteta DEVICE tipa WATERPOINT sa određenim atributima (lokacijom) te dodjeljivanja (engl. *assign*) tog uređaja odgovarajućem gradu (entitetu ASSET tipa CITY).

Prema opisanom korisničkom slučaju, korisnik pregledava kartu te može odabrati određenu lokaciju (*latitude, longitude*) za dodavanje nove *water point*. Slično stvaranju ostalih IoT entiteta, potrebno je specificirati **naziv** nove točke (uređaja), **tip** te spomenutu **lokaciju**:

POST

http://localhost:9090/**api/device**

zahtjev za stvaranje novog uređaja (water point)

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
{
    "name": "WaterPoint 1",
    "type": "waterpoint",
    "additionalInfo": {
        "latitude": 45.555,
        "longitude": 18.6955
    }
}
```

Example Response 200 OK

```
{
    "id": {
        "entityType": "DEVICE",
        "id": "ddc81d50-8071-11e9-9a0a-2518cdb354e4"
    },
    "createdTime": 1558956215205,
    "additionalInfo": {
        "latitude": 45.555,
        "longitude": 18.6955
    },
    "tenantId": {
        "entityType": "TENANT",
        "id": "3223f4a0-613c-11e9-8010-a32539fe4f3c"
    },
    "customerId": {
        "entityType": "CUSTOMER",
        "id": "13814000-1dd2-11b2-8080-808080808080"
    },
    "name": "WaterPoint 1",
    "type": "waterpoint"
}
```

Sljedeći korak je dodjeljivanje stvorene točke (uređaja) odgovarajućem gradu odnosno stvaranje relacije tipa ***Contains*** koja je usmjerena **od** grada *WaterIoT City A1* (određen sa ***id***) **prema** upravo kreiranom uređaju. Dakle, potrebno je konstruirati sljedeći zahtjev:

POST

http://localhost:9090/**api/relation**

zahtjev za dodjeljivanje točke mjerenja određenom gradu

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
{
    "from": {
        "id": "8c0cda70-613e-11e9-8010-a32539fe4f3c",
        "entityType": "ASSET"
    },
    "type": "Contains",
    "to": {
        "entityType": "DEVICE",
        "id": "ddc81d50-8071-11e9-9a0a-2518cdb354e4"
    }
}
```

Na ovaj način kreirani su i ostali uređaji i dodijeljeni pripadnim gradovima prema postavljenoj hijerarhiji entiteta opisanog korisničkog slučaja (Slika 4.1).

Određenu točku mjerenja (*water point*) moguće je obrisati slanjem **DELETE** zahtjeva na **/api/device/**{***waterPointId***} te dohvatiti slanjem **GET** zahtjeva na isti *endpoint*. Thingsboard REST API nudi i mogućnost **izmjene** (*update*) određenog uređaja (i *asset*-a) slanjem **POST** zahtjeva sličnog stvaranju novog uređaja uz određene izmjene u tijelu zahtjeva:

Body

```
{
    "id": {
        "entityType": "DEVICE",
        "id": "ddc81d50-8071-11e9-9a0a-2518cdb354e4"
    },
    "additionalInfo": {
        "latitude": 45.5549,
        "longitude": 18.6954
    },
    "name": "WaterPoint 1",
    "type": "waterpoint"
}
```

Stvorene točke mjerenja vode moguće je **dohvatiti** na više načina:

- Dohvaćanje određene točke slanjem **GET** zahtjeva na **/api/device/**{***waterPointId***}
- Dohvaćanje svih točki koje se nalaze trenutno na platformi (u vlasništvu stanara) slanjem **GET** zahtjeva na način opisan pri dohvaćanju svih regija (i gradova) samo što je u pitanju drugi tip (***type: waterpoint***) te drugi *endpoint* na koji je potrebno poslati zahtjev, budući da je u riječ o dohvaćanju uređaja (**/api/tenant/devices**)
- Dohvaćanje svih točki za određeni grad putem **upita** (engl. *query*) odnosno slanjem sljedećeg **POST** zahtjeva gdje je potrebno definirati: **identifikator grada** (*rootId*), **tip uređaja** (*deviceTypes*) te parametre za određivanje **relacije** (*direction, type*)

POST

http://localhost:9090/**api/devices**

zahtjev za dohvaćanje uređaja (water points) za određeni grad

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
{
    "deviceTypes": [
        "waterpoint"
    ],
    "parameters": {
        "rootId": "8c0cda70-613e-11e9-8010-a32539fe4f3c",
        "rootType": "ASSET",
        "direction": "FROM"
    },
    "relationType": "Contains"
}
```

Example Response 200 OK

```
[
    {
        "id": {
            "entityType": "DEVICE",
            "id": "ddc81d50-8071-11e9-9a0a-2518cdb354e4"
        },
        "additionalInfo": {
            "latitude": 45.5549,
            "longitude": 18.6954
        },...
        "name": "WaterPoint 1",
        "type": "waterpoint"
    },
    ... //WaterPoint 2, WaterPoint 3
]
```

4.4. Slanje podataka

Kada se korisniku prikažu određene točke mjerenja kvalitete vode na karti (*water points*), on može odabrati jednu od njih i za nju poslati ili dohvatiti podatke mjerenja kvalitete vode – **provodljivost**, **temperaturu**, **zamućenost** i **pH vrijednost**.

Kako bi se poslali i spremili podaci mjerenja za određenu točku, potrebno je konstruirati sljedeći **POST** zahtjev, gdje ***id*** predstavlja jedinstveni identifikator odabrane točke mjerenja kvalitete vode:

POST

http://localhost:9090/api/plugins/telemetry/DEVICE/{***id***}/timeseries/DEVICE

zahtjev za slanje podataka za određenu točku

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
{
      "conductivity": 25.33,
      "temperature": 14.5,
      "turbidity": 0.3,
      "pH": 7.01
}
```

Example Response 200 OK

Navedeni načinom se podaci šalju i spremaju putem zahtjeva autoriziranog administratorskim JWT tokenom. Postoji i drugi način slanja podataka gdje se autorizacija vrši **pristupnim tokenom** (***access token***) uređaja. Prije slanja podataka tim načinom, za određeni uređaj (*water point*), potrebno je dohvatiti i spremiti pristupni ključ (*access token*) za taj uređaj. Dakle, potrebno je poslati sljedeći **GET** zahtjev:

GET

http://localhost:9090/api/device/{***id***}/credentials

zahtjev za dohvaćanje pristupnog ključa određenog uređaja

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Body

```
none
```

Example Response 200 OK

```
{
    "id": {
        "id": "ddd6ea60-8071-11e9-9a0a-2518cdb354e4"
    },
    "createdTime": 1558956215302,
    "deviceId": {
        "entityType": "DEVICE",
        "id": "ddc81d50-8071-11e9-9a0a-2518cdb354e4"
    },
    "credentialsType": "ACCESS_TOKEN",
    "credentialsId": "KXE7rnlqQ0Xbvz9nUiV6",
    "credentialsValue": null
}
```

U odgovoru na navedeni GET zahtjev, dobiva se tajni ključ (*credentialsId*) koji se koristi za autorizaciju uređaja pri slanju podataka. Za slanje podataka putem dobivenog ključa, konstruira se sljedeći **POST** zahtjev:

POST

http://localhost:9090/api/v1/{***accessToken***}/telemetry

zahtjev za slanje podataka za određenu točku

Headers

Content-Type	application/json

Body

```
{
     "conductivity": 25.33,
     "temperature": 14.5,
     "turbidity": 0.3,
     "pH": 7.01
}
```

Example Response 200 OK

Opisani postupak slanja podataka mjerenja na IoT platformu podrazumijeva korištenje **HTTP** protokola. Međutim, u praksi, IoT uređaji koji šalju podatke, obično ne koriste HTTP protokol za prijenos podataka već protokole koji su puno bolje prilagođeni IoT-u i razmjeni podataka u takvom okruženju, čime se povećava brzina prijenosa a smanjuje veličina paketa i

potrošnja baterije IoT uređaja koji komunicira sa platformom. Riječ je o **MQTT** i **CoAP** protokolu. U nastavku su opisani primjeri slanja podataka mjerenja na Thingsboard IoT platformu putem spomenutih protokola.

4.4.1. Slanje podataka MQTT protokolom

MQTT (*Message Queuing Telemetry Transport*) je jednostavan protokol za razmjenu poruka s objavom i pretplatom (engl. *publish and subscribe*) što ga čini vjerojatno najprikladnijim protokolom za razne IoT uređaje. Zahtijeva postojanje MQTT brokera (posrednika) preko kojeg se poruke objavljuju i klijenti pretplaćuju.

U okvirima razmatranog IoT scenarija i primjene odabrane IoT platforme, MQTT broker, na koji se IoT uređaji spajaju MQTT protokolom, je Thingsboard server sa setom unaprijed definiranih tema (engl. *topics*):

- **v1/devices/me/telemetry** – za spremanje podataka mjerenja (telemetriju) slanjem PUBLISH poruke.
- **v1/devices/me/attributes** – za spremanje klijentskih informacija o uređaju slanjem PUBLISH poruke te za pretplatu na ažuriranje dijeljenih informacija o uređaju sa servera slanjem SUBSCRIBE poruke.

Za komunikaciju sa Thingsboard serverom putem MQTT protokola je potrebno definirati **host**, a budući da je instalacija platforme lokalna, host je ***localhost***. Port za MQTT protokol je uobičajeni ***1883***, što je vidljivo i u prethodno opisanoj instalaciji Thingsboard IoT platforme (4.1).

Nakon što su definirani svi parametri za komunikaciju IoT uređaja i Thingsboard servera, IoT uređaj može poslati podatke mjerenja (temperatura, zamućenost, provodljivost i pH vrijednost) putem MQTT protokola na određenu temu: **v1/devices/me/telemetry**. Kao primjer slanja podataka MQTT protokolom na Thingsboard server, korišten je MQTT klijent *Mosquitto.*

Dakle, za slanje podataka mjerenja kvalitete vode putem MQTT protokola, potrebno je poslati PUBLISH poruku sa sljedećim parametrima:

```
mosquitto_pub -d -h localhost -t v1/devices/me/telemetry
-u JkdaBsMo1wF7nVv9RTn0 -m '{"conductivity": 25.6, "temperature":
14.2, "turbidity": 0.34, "pH": 7.12}'
```

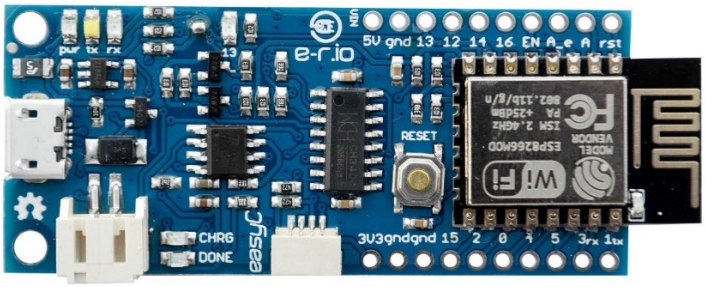
Parametar ***–u*** (*username*) se koristi za autorizaciju; ovdje je riječ o autorizaciji pristupnim ključem uređaja (***access token***).

U nastavku je prikazan praktičan primjer (Arduino program) opisanog slanja podataka MQTT-om sa IoT uređaja koji je baziran na Arduino mikroprocesoru sa ESP8266 Wi-Fi modulom:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

// MQTT parameters
const char* mqtt_server = "192.168.2.15"; // Local IP address of localhost
const char* topic = "v1/devices/me/telemetry"; // MQTT topic
const char* username = "JkdaBsMo1wF7nVv9RTn0"; // Device 'access token'
const char* deviceName = "WaterPoint 2";

const char* ssid = "WiFi_SSID";
const char* password = "WiFi_PASSWORD";
const int TIME_INTERVAL = 5000; // Send data every 5s (5000ms)
WiFiClient espClient;
PubSubClient client(espClient);
StaticJsonBuffer<300> jsonBuffer;
JsonObject& data = jsonBuffer.createObject();
long lastMsg = 0;
char msg[300];

void setup_wifi() { ... }
void callback(char* topic, byte* payload, unsigned int length) { ... }

void reconnect() {
  while (!client.connected()) {// Loop until we're reconnected
    Serial.println("Establishing MQTT connection...");
    if (client.connect(deviceName, username, "")) {
      Serial.println("Connected to MQTT server!");
      Serial.print("Sending data to MQTT topic: "); Serial.println(topic);
      Serial.print("Device name: "); Serial.println(deviceName);
    }
    else {
      Serial.print("failed, rc="); Serial.println(client.state());
      delay(5000);// Wait 5 seconds before retrying
    }
  }
}

void getData(){// Simulation of getting the real measurements from sensors
  float randNumber = random(100)/100.0;
  data["temperature"] = 10.0 + 3*randNumber;
  data["conductivity"] = 20.0 + 5*randNumber;
  data["turbidity"] = 0.3 + 0.1*randNumber;
  data["pH"] = 7.0 + 0.5*randNumber;
  data.printTo(msg);
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();

  long now =;// Sending data with specific time interval
  if (now - lastMsg > TIME_INTERVAL) {
    lastMsg = now;
    getData();
    Serial.print("Publishing message: "); Serial.println(msg);
    client.publish(topic, msg);
  }
}
```

Navedeni primjer koristi *PubSubClient* Arduino biblioteku za rad sa MQTT protokolom te je simulirano prikupljanje i slanje nasumično generiranih podataka mjerenja kvalitete vode svakih 5 sekundi na Thingsboard IoT platformu. U navedenom primjeru, MQTT server je i dalje lokalni host (poslužitelj) na kojem je pokrenuta instanca Thingsboard servera, no, budući da mu se pristupa sa drugog računala (uređaja) iz lokalne mreže, predstavljen je svojom lokalnom IP adresom (*localhost -> 192.168.2.15*). U nastavku je prikazan ispis serijskog terminala (Slika 4.7) IoT uređaja te korišteni uređaj *Croduino NOVA2* (Slika 4.8) na kojem je pokrenut opisani program:

Slika 4.7 *Slanje podataka MQTT protokolom na Thingsboard IoT platformu*



Slika 4.8 *Korištena razvojna pločica - Croduino NOVA2 [10]*

Thingsboard IoT platforma također nudi i mogućnost slanja podataka CoAP protokolom preko ugrađenog CoAP servera na UDP portu. U nastavku se nalazi kratki opis CoAP protokola, Thingsboard CoAP API-ja te primjeri slanja podataka mjerenja navedenim protokolom.

4.4.2. Slanje podataka CoAP protokolom

CoAP (*Constrained Application Protocol*) je specijalizirani web protokol za prijenos podataka u ograničenim IoT mrežama sa ograničenim čvorovima (engl. *nodes*) – uređaji niske potrošnje, male količine ROM i RAM memorije i slično. Dizajniran je za M2M (*machine-to-machine*) aplikacije pri čemu pruža zahtjev/odgovor (engl. *request/response*) tip interakcije. Poput HTTP-a, podržava ključne web koncepte poput URI-ja (jedinstveni identifikator resursa); prilagođen je REST arhitekturi, jednostavan je, ima vrlo kratko zaglavlje (engl. *overhead*) te koristi UDP (*User Datagram Protocol*) pri prijenosu paketa. [11]

U okvirima primjene odabrane IoT platforme za razmatrani IoT scenarij, CoAP server, s kojim IoT uređaji komuniciraju CoAP protokolom, je Thingsboard server sa unaprijed definiranim resursima (URI putanjama):

- **api/v1/ACCESS_TOKEN/telemetry** – za spremanje podataka mjerenja (telemetriju) slanjem CoAP POST zahtjeva.
- **api/v1/ACCESS_TOKEN/attributes** – za spremanje klijentskih informacija o uređaju slanjem CoAP POST zahtjeva te za njihovo dohvaćanje i dohvaćanje dijeljenih informacija o uređaju sa servera slanjem CoAP GET zahtjeva; moguća je i pretplata na izmjene dijeljenih atributa slanjem CoAP GET zahtjeva sa *Observe* opcijom.

Budući da je instalacija platforme lokalna, poslužitelj je ***localhost*** a port za komunikaciju sa Thingsboard CoAP API-jem je **UDP** port ***5683***, što je vidljivo i u prethodno opisanoj instalaciji Thingsboard IoT platforme (4.1).

Nakon što su definirani svi parametri za komunikaciju IoT uređaja i Thingsboard CoAP servera, IoT uređaj može poslati podatke mjerenja sa senzora slanjem CoAP POST zahtjeva sa definiranom putanjom: **api/v1/ACCESS_TOKEN/telemetry**. Kao primjer slanja podataka CoAP protokolom na Thingsboard server, korišten je komandno-linijski klijent *CoAP-CLI.*

Dakle, za slanje podataka mjerenja kvalitete vode putem CoAP protokola, potrebno je poslati sljedeći CoAP POST zahtjev:

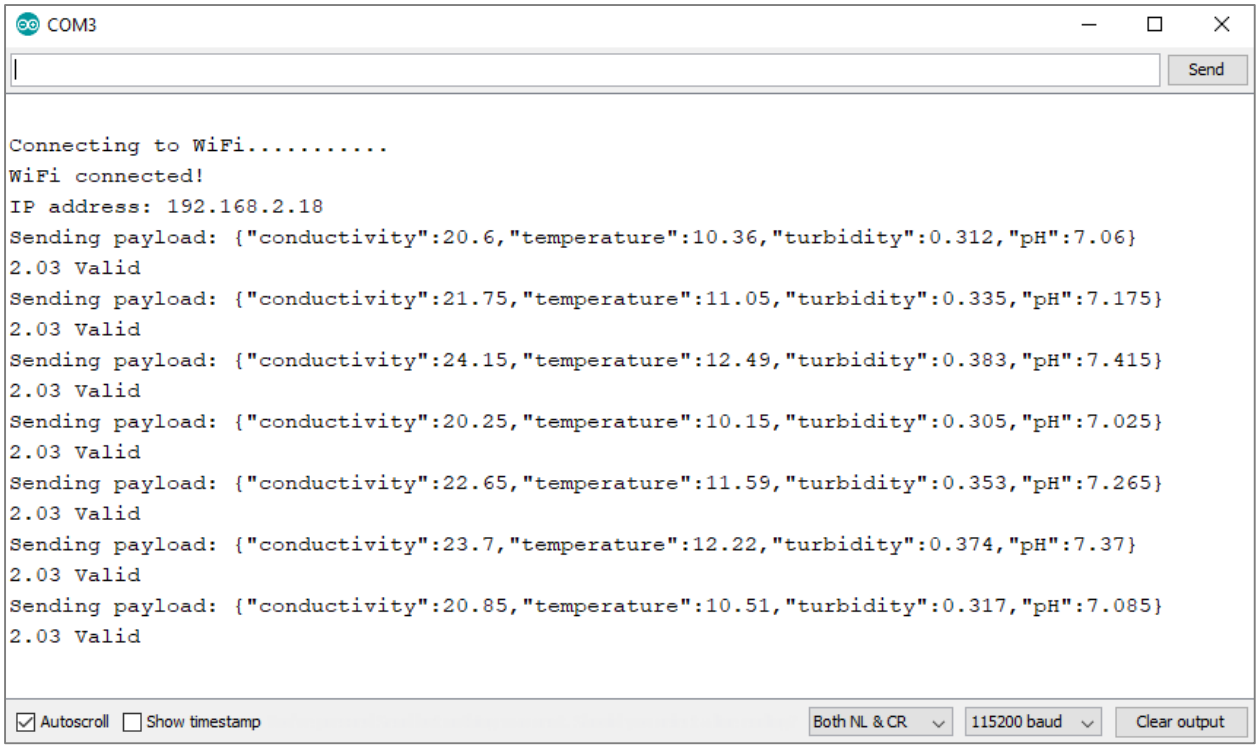
```
coap post coap://localhost/api/v1/JkdaBsMo1wF7nVv9RTn0/telemetry
-p '{"conductivity": 25.6, "temperature": 14.2, "turbidity": 0.34,
"pH": 7.12}'
```

U nastavku je prikazan praktičan primjer (Arduino program) opisanog slanja podataka CoAP protokolom sa već spomenute razvojne pločice *Croduino NOVA2* kao IoT uređaja:

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <coap-simple.h>
#include <ArduinoJson.h>

//WiFi settings
const char* ssid = "WiFi_SSID";
const char* password = "WiFi_PASSWORD";

//CoAP parameters
IPAddress ip(192,168,2,15);// Local IP address of localhost
int port = 5683;// Default CoAP port
char* path = "api/v1/ACCESS_TOKEN/telemetry";// Resource path

WiFiUDP udp;
Coap coap(udp);
StaticJsonBuffer<300> jsonBuffer;
JsonObject& data = jsonBuffer.createObject();

void setup_wifi() { ... }

void callback_response(CoapPacket &packet, IPAddress ip, int port) { ... }

String getData(){// Simulation of getting the real data from sensors
  float randNumber;
  randNumber = random(100)/100.0;
  data["conductivity"] = 20.0 + 5*randNumber;
  data["temperature"] = 10.0 + 3*randNumber;
  data["turbidity"] = 0.3 + 0.1*randNumber;
  data["pH"] = 7.0 + 0.5*randNumber;
  String msg;
  data.printTo(msg);
  return msg;
}

void setup() {
    Serial.begin(115200);
    setup_wifi();
    coap.response(callback_response);
    coap.start();
}

void loop() {
    coap.loop();
    delay(5000);// Send data every 5s
    String msg = getData();

    int msgid = coap.post(
        ip, port, path,
        (uint8_t *)msg.c_str(), //Payload
        msg.length()
    );
    if(msgid>0){
        Serial.print("Sending payload: "); Serial.println(msg);
    }
}
```

Navedeni primjer koristi *coap-simple* Arduino biblioteku za rad sa CoAP protokolom te je, kao i kod primjera za MQTT, simulirano prikupljanje i slanje nasumično generiranih podataka mjerenja kvalitete vode svakih 5 sekundi na Thingsboard IoT platformu.

U ovom primjeru, CoAP server je lokalni poslužitelj na adresi *192.168.2.15* sa CoAP (UDP) portom *5683*. Na Slika 4.9 je prikazan ispis serijskog terminala korištenog IoT uređaja na kojem je pokrenut navedeni program:



Slika 4.9 *Slanje podataka CoAP protokolom na Thingsboard IoT platformu*

U prilozima diplomskog rada se nalaze snimljeni paketi pri slanju podataka mjerenja na IoT platformu različitim protokolima (HTTP, MQTT, CoAP); paketi su snimljeni pomoću programskog alata za analizu mrežnih paketa - ***Wireshark***. Na temelju uzorka od 10 poslanih mjerenja za svaki protokol, dobivena su prosječna vremena potrebna za slanje jednog mjerenja.

Za svaki protokol definirana je *vremenska referenca* od koje se računa vremenska razlika do trenutka kada je primljen paket potvrde o uspješno spremljenim podacima mjerenja; za HTTP vremenska referenca je trenutak slanja POST paketa a potvrdu uspješnog spremanja predstavlja primitak HTTP paketa sa kodom *200 OK*; za MQTT referenca je „*Connect Command*“ paket a uspješan završetak slanja označen je „*Disconnect Req*“ paketom; za CoAP referenca je CoAP POST paket a uspješno spremanje označava (prvi) primljeni paket sa kodom *67* (*2.03 Valid*). U navedenoj analizi, u odnosu na HTTP (~**39.6** ms), MQTT protokol ima bolji rezultat (~**29.3** ms) vezano za prosječno vrijeme slanja, kao i CoAP protokol, čije je prosječno vrijeme slanja najmanje (~**16.3** ms) što potvrđuje dobru prilagođenost MQTT i CoAP protokola IoT mrežama ali i brzinu UDP protokola za prijenos paketa kojeg koristi CoAP.

4.5. Dohvaćanje podataka

Za **dohvaćanje podataka** mjerenja kvalitete vode za određenu točku (*water point*), potrebno je konstruirati sljedeći **HTTP GET** zahtjev, gdje ***id*** predstavlja jedinstveni identifikator odabrane točke mjerenja kvalitete vode a *JWT_TOKEN* prethodno već objašnjeni JSON Web Token odnosno autorizacijski ključ prijavljenog korisnika platforme:

GET

http://localhost:9090/api/plugins/telemetry/DEVICE/{***id***}/values/timeseries

zahtjev za dohvaćanje podataka za određenu točku

Headers

Content-Type	application/json
X-Authorization	Bearer *JWT_TOKEN*

Params

keys	conductivity,temperature,turbidity,pH

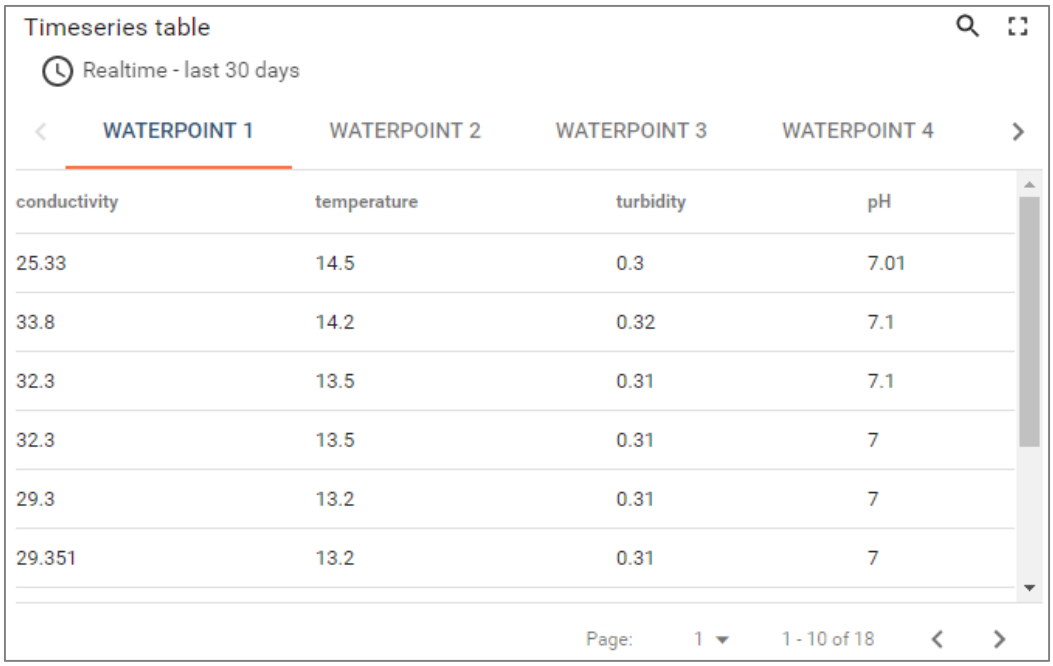
Example Response 200 OK


```
{
    "conductivity": [
        {
            "ts": 1559217600315,
            "value": "25.33"
        }
    ],
    "temperature": [
        {
            "ts": 1559217600315,
            "value": "14.5"
        }
    ],
    "turbidity": [
        {
            "ts": 1559217600315,
            "value": "0.3"
        }
    ],
    "pH": [
        {
            "ts": 1559217600315,
            "value": "7.01"
        }
    ]
}
```

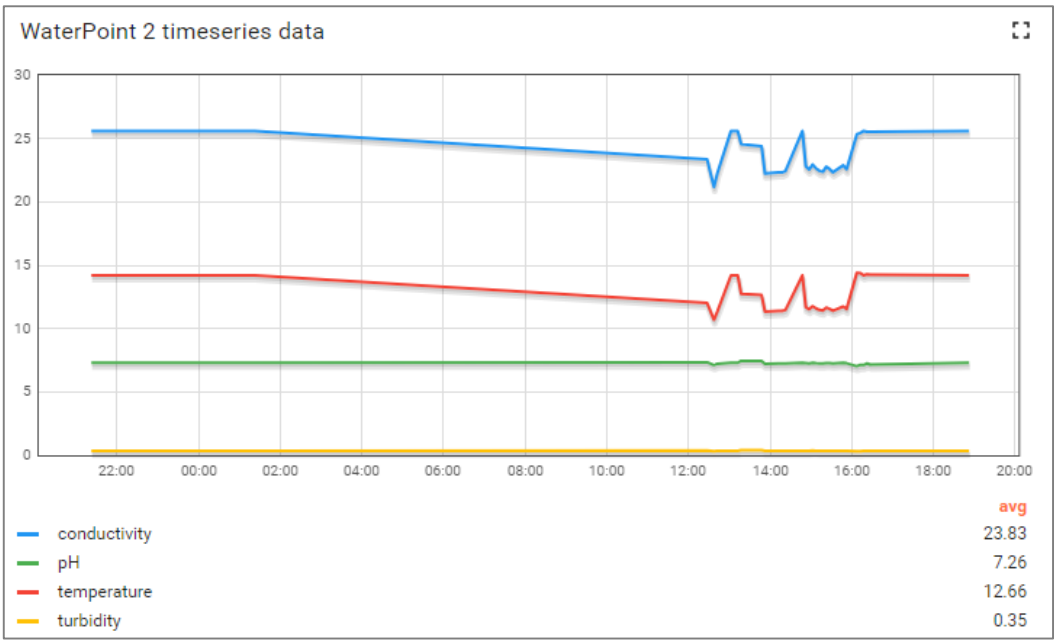

Pri dohvaćanju podataka potrebno je u parametrima zahtjeva specificirati barem jedan telemetrijski *ključ*. Navedeni zahtjev dohvaća **zadnje vrijednosti** (engl. *latest telemetry*) mjerenja za odabranu točku.

Ukoliko se žele dohvatiti **svi podaci** mjerenja unutar određenog vremenskog intervala potrebno je u parametrima zahtjeva specificirati početno vrijeme intervala (***startTs***) i krajnje vrijeme intervala (***endTs***) u *Unix Epoch Timestamp* formatu (milisekunde).

Na sljedećim slikama su primjeri prikaza IoT podataka za odabranu točku mjerenja kvalitete vode pomoću Thingsboard kontrolne ploče (engl. *dashboard*):



Slika 4.10 *Prikaz podataka tablicom pomoću Thingsboard kontrolne ploče*



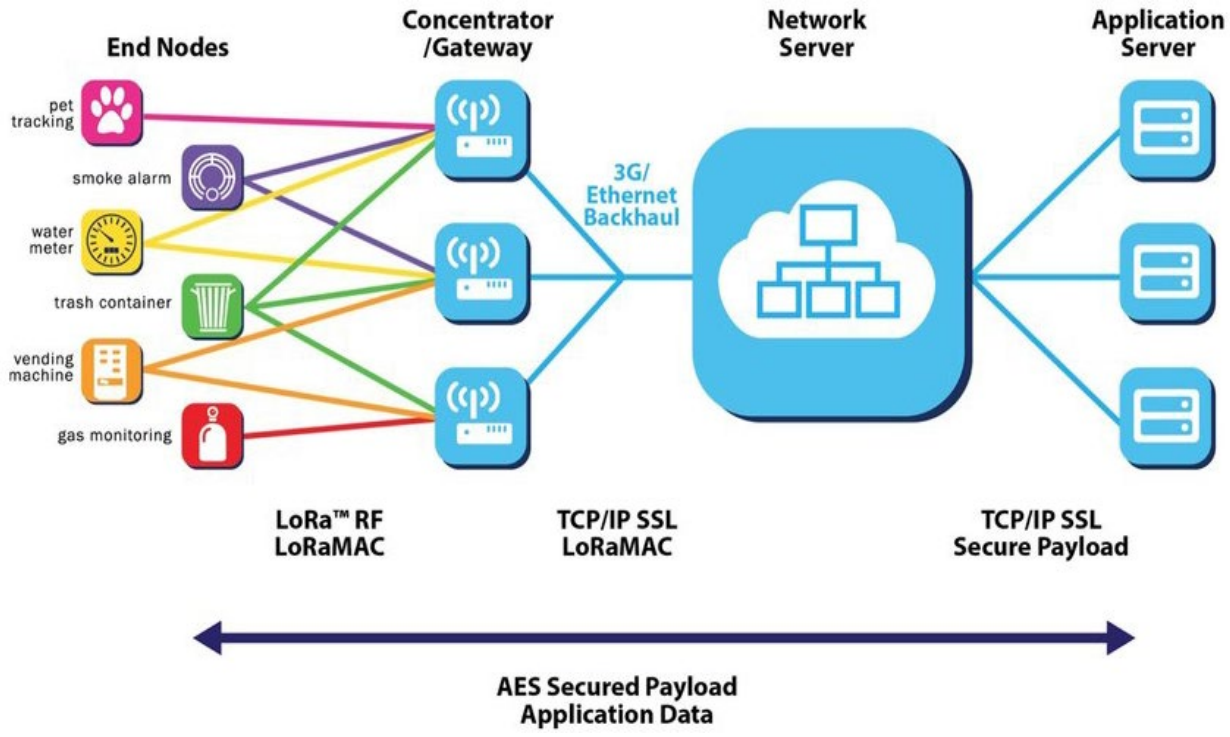
Slika 4.11 *Prikaz podataka grafom pomoću Thingsboard kontrolne ploče*

4.6. Integracija LoRa mreže

LoRa (skraćeno za *Long Range*) je tehnika modulacije raspršenog spektra **velikog dometa** i **male potrošnje** (engl. *low power*) koja se koristi kao radio-frekvencijska komunikacijska tehnologija za bežični prijenos podataka u različitim IoT mrežama.

Sama tehnologija podrazumijeva opisanu tehnologiju prijenosa na fizičkom sloju – **LoRa** tehnologiju te **LoRaWAN** (*Long Range Wide Area Network*) – komunikacijski protokol koji predstavlja gornje slojeve LoRa mreže – sloj pristupa mediju te mrežni sloj za upravljanje komunikacijom između LoRa uređaja/čvorova (engl. *end-node devices*) i LoRa pristupnika (engl. *gateways*). LoRaWAN upravlja brzinom prijenosa (51 byte po poruci – *low bit rate*), frekvencijama prijenosa te potrošnjom uređaja koji su obično napajani baterijski.

LoRa uređaji su IoT uređaji sa određenim senzorima i LoRa primopredajnicima, male potrošnje koji podržavaju LoRa prijenos velikog dometa (5-15 km) i prenose podatke na **LoRa pristupnike** (*gateways*). Podaci koje šalje određeni LoRa uređaj može primiti više LoRa pristupnika koji onda prosljeđuju primljene pakete putem Interneta – širokopojasni prijenos (engl. *high bandwidth*) na centralizirane **mrežne servere** koji filtriraju dupliranje paketa, vrše sigurnosne provjere i upravljaju mrežom. Ti podaci se onda šalju na **aplikacijski server**.

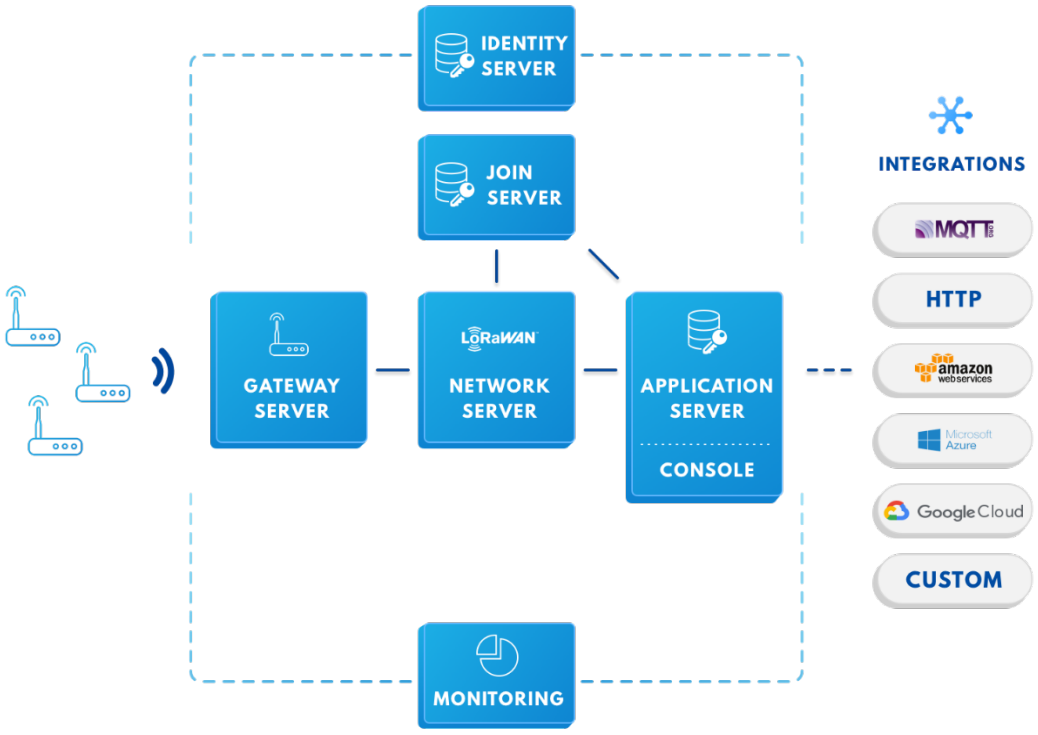


Slika 4.12 *Arhitektura LoRa mreže [13]*

Mnoge tvrtke i mrežni operatori su se uključili u tzv. LoRa savez (engl. ***LoRa Alliance***) – otvorena, neprofitna udruga sa preko 500 članova diljem svijeta, koji rade na stvaranju nisko-budžetnih, jednostavnih LoRa uređaja, standardizaciji LoRaWAN protokola te širenju i poboljšanju LoRa tehnologije, omogućavajući pritom njenu primjenu u velikom broju IoT scenarija.

Za upravljanje LoRa uređajima, pristupnicima te prikupljanje podataka dostupno je nekoliko otvorenih/besplatnih LoRa IoT platformi koje uključuju LoRa mrežni i aplikacijski server; jedan od takvih projekata je i LoRa Server koji nudi *open-source* komponente za kreiranje LoRaWAN mreža – podršku za LoRa pristupnike, mrežni server, aplikacijski server, geolokacijski server itd.

Jedna od popularnijih *open-source* decentraliziranih platformi za kreiranje i upravljanje IoT LoRa mreža je **The Things Network**; besplatna je za korištenje i nudi bogat set alata te otvorenu, globalnu IoT mrežu za povezivanje uređaja i realizaciju različitih IoT aplikacija.



Slika 4.13 *The Things Network arhitektura [12]*

Kao primjer integracije LoRa mreže u sklopu primjene Thingsboard IoT platforme za korisnički slučaj mjerenja kvalitete vode, odabrana je **The Things Network** platforma (*Community Edition*) za konfiguriranje i upravljanje demonstrativnom LoRa mrežom.

4.4.1. Thingsboard integracije

Thingsboard nudi mogućnost integriranja vlastite IoT platforme sa IoT uređajima koji su spojeni na sustave trećih strana (engl. *third party systems*). Thingsboard trenutno podržava dva glavna integracijska protokola – HTTP i MQTT putem kojih povezuje postojeće NB IoT, LoRaWAN, SigFox i druge uređaje koji imaju specifičan format korisnih podataka (*payload*) sa Thingsboard IoT platformom, pretvarajući pritom podatke sa uređaja u format koji odgovara formatu podataka na Thingsboard IoT platformi. Dvije su raspoložive opcije integracije na Thingsboard platformi:

- **Thingsboard Platform Integrations** (*Professional Edition*) i
- **Thingsboard IoT Gateway** (*Community Edition*)

Thingsboard Platform Integrations je značajka profesionalnog izdanja Thingsboard IoT platforme koja omogućuje spajanje uređaja sa drugih platformi osiguravajući siguran i pouzdan API most (engl. *API bridge*) između temeljnih značajki platformi – telemetrijski podaci, atributi te RPC poziva (engl. *Remote Procedure Calls*) i specifičnih API-ja platformi trećih strana. Podržava integraciju sa IoT platformama poput AWS IoT, IBM Watson, The Things Network, Microsoft Azure, SigFox, ThingPark i dr. Dohvaćeni podaci se zatim pretvaraju u oblik pogodan za Thingsboard platformu. Konfiguracija integracije se odvija putem Thingsboard GUI-a.

Thingsboard IoT Gateway je *open-source* rješenje koje je dio Thingsboard IoT platforme - *Community Edition* te također omogućava spajanje uređaja drugih platformi na Thingsboard te pretplatu na podatke koje ti uređaji šalju. Thingsboard Gateway je usluga koja se spaja na vanjski MQTT posrednik (engl. *external MQTT broker*) određene platforme te na Thingsboard MQTT server i služi kao *proxy* ili API most između njih. Konfiguracijom Thingsboard IoT Gateway-a omogućava se preplata na podatke određene MQTT teme (engl. *topics*) vanjske platforme te konverzija i spremanje dobivenih podataka na Thingsboard platformu.

Značajke opcije *IoT Gateway* i *Platform Integrations* se donekle preklapaju a ključne razlike su sljedeće:

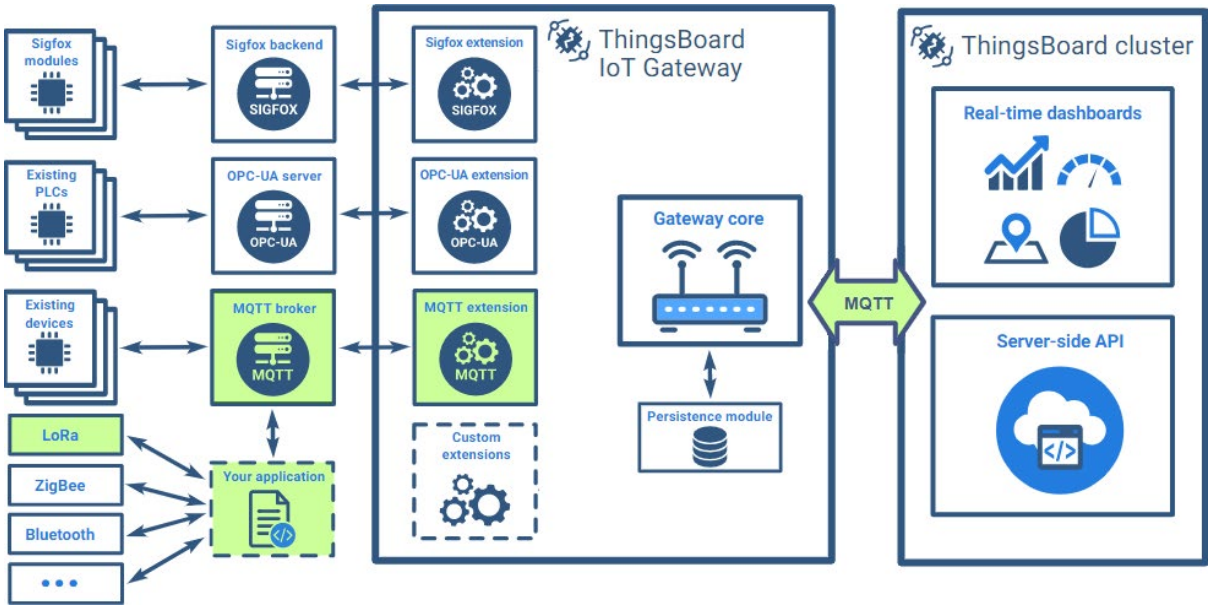
- Thingsboard IoT Gateway je dizajniran za implementaciju u lokalnim mrežama; Platform Integrations su dizajnirane za integraciju između dva poslužitelja.
- IoT Gateway je dizajniran da podrži <1000 uređaja, dok opcija Platform Integrations podržava visoku propusnost i skalabilnost.

- Za dodavanje/mijenjanje prilagođenog dekodera podataka kod IoT Gateway-a potrebno je ponovno pokretanje, dok je kod Platform Integrations *Integration Converter* JavaScript funkcija koja se može mijenjati u stvarnom vremenu.

Za potrebe integracije Thingsboard IoT platforme i LoRa mreže (*The Things Network*) u sklopu korisničkog slučaja mjerenja kvalitete vode, odabrana je druga opcija – **Thingsboard IoT Gateway** koja svojim opsegom funkcionalnosti pokriva zahtjeve zadanog korisničkog slučaja te spada u dostupno *Community* izdanje Thingsboard platforme što ju čini pogodnom za implementaciju u već kreirani Thingsboard IoT model (4.2).

Thingsboard IoT Gateway je, u suštini, Java aplikacija koja se na određenim platformama može instalirati kao usluga čija se funkcionalnost određuje konfiguracijskim datotekama te koja je u mogućnosti slati podatke sa različitih izvora na Thingsboard server putem MQTT-a.

U nastavku je opisano odabrano **rješenje za integraciju LoRa mreže sa Thingsboard platformom** – konfiguracija Thingsboard IoT Gateway-a (uz prethodnu instalaciju aplikacije kao Windows servisa) za uspješno spajanje na Thingsboard platformu i vanjski MQTT *broker* odabrane platforme za konfiguriranu LoRa mrežu – *The Things Network*. Odabrano rješenje integriranja LoRa mreže sa postojećim IoT modelom na Thingsboard platformi je ilustrirano na Slika 4.14 (IoT Gateway shema).



Slika 4.14 *Odabrana integracija LoRa mreže i Thingsboard platforme [6]*

Vidljivo je da je IoT Gateway u ulozi posrednika odnosno da spaja Thingsboard platformu (MQTT server koji sluša na *localhost:1883*) te MQTT *broker* vanjske platforme koja upravlja LoRa mrežom – u ovom slučaju MQTT API The Things Network platforme na koji je Gateway

preplaćen. Da bi MQTT veza između navedenih komponenti bila moguća, potrebno je konfigurirati MQTT ekstenziju Thingsboard IoT Gateway-a, koja osim spajanja na vanjski MQTT *broker*, ima ulogu formatiranja tj. *mapiranja* (engl. *mapping*) ulaznih podataka iz LoRa mreže u format pogodan za Thingsboard IoT platformu (*ključ – vrijednost* telemetrijskih podataka).

U nastavku slijedi prvi korak integracije LoRa mreže prema smjernicama opisanog rješenja – konfiguracija LoRa mreže na The Things Network platformi.

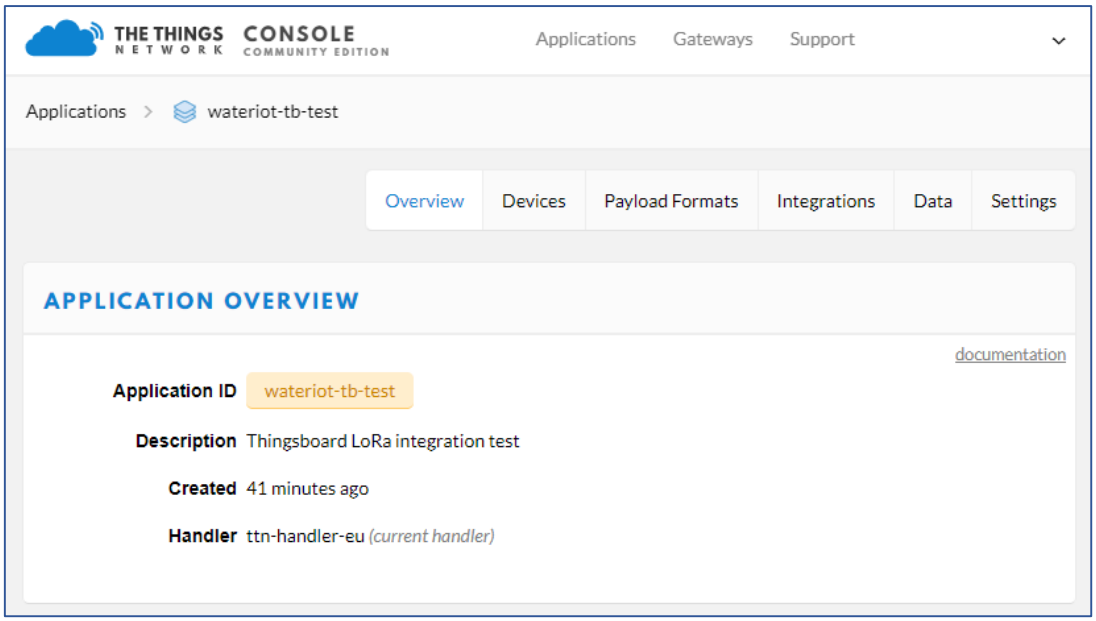
4.4.2. Konfiguracija LoRa mreže

Kao što je vidljivo na Slika 4.13 koja prikazuje arhitekturu *The Things Network* (*TTN*) platforme, LoRa uređaji šalju podatke LoRa bežičnom tehnologijom na LoRa *gateways* – pristupnike koji putem Interneta prenose podatke puno većim brzinama na TTN mrežni server.

Podaci sa senzora – korisni podaci (*payload*) se šalju u binarnom obliku kao niz bajtova (engl. *byte*) pa se poslani *payload* sa uređaja obično zapisuje u heksadekadskom obliku (npr. 0F 1A 28 4B, što odgovara prijenosu od 4 bajta). Svaka poruka sadrži određene identifikatore pomoću kojih ih LoRaWAN pristupnici usmjeravaju na određeni mrežni server. Jednom kada podaci uspješno stignu na mrežni server, mogu se konzumirati od strane aplikacijskog servera. Rad s TTN aplikacijskim serverom je omogućen putem TTN konzole (*The Things Network Console – Community Edition*).

Preko aplikacijskog servera, TTN nudi niz mogućih integracija sa IoT platformama poput *Amazon Web Services, Microsoft Azure* te *Google Cloud*, putem HTTP i MQTT sučelja. TTN nudi i **Data API** putem kojega je moguće direktno dohvatiti događaje i poruke sa LoRaWAN mreže koristeći **MQTT** protokol. Dakle, putem MQTT posrednika (*Data API*) podaci će se konzumirati od strane IoT Gateway-a te proslijediti na Thingsboard IoT platformu. No ponajprije, potrebno je stvoriti demo LoRaWAN mrežu odnosno testnu aplikaciju na The Things Network platformi unutar koje je zatim moguće registrirati LoRaWAN uređaje te simulirati slanje podataka na mrežni server.

Aplikacija se kreira u The Things Network konzoli pri čemu je potrebno specificirati identifikator aplikacije (Application ID): *console.thethingsnetwork.org/**applications/add***. Sljedeća slika prikazuje informacije o kreiranoj aplikaciji. ID aplikacije je ***wateriot-tb-test***.

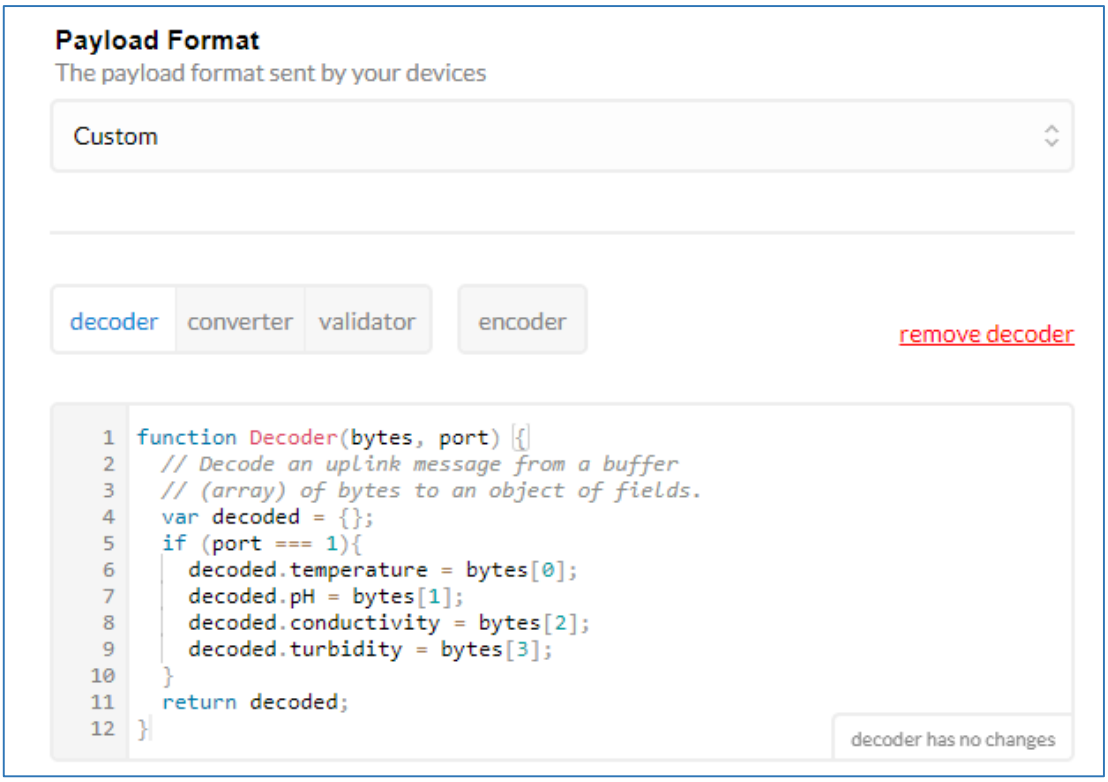


Slika 4.15 *The Things Network demo aplikacija*

Pri stvaranju aplikacije dobiva se **pristupni ključ** (*Access Key*) koji će se koristiti za autentikaciju preko MQTT-a, gdje će korisničko ime biti određeno identifikatorom aplikacije (*wateriot-tb-test*) a lozinka dobivenim ključem.

Sljedeći korak je dodavanje odnosno registracija LoRaWAN uređaja: (*/app_id/devices/register*) pri čemu je potrebno unijeti naziv uređaja (Device ID) – za ovaj primjer odabran je naziv: *waterpoint-lora-1*, te jedinstveni identifikator (DevEUI). Nakon stvaranja uređaja putem konzole moguće je slati poruke na uređaj (*downlink*) ili simulirati slanje podataka sa uređaja (*uplink*). Podaci odnosno *payload* se šalje u binarnom formatu kao niz bajtova.

Nakon registracije aplikacije i uređaja potrebno je konfigurirati dekoder podataka koji će pretvoriti podatke iz binarnog formata u JSON. To je moguće napraviti na The Things Network platformi pomoću *payload* funkcija ili na Thingsboard strani pomoću *uplink/downlink* pretvarača. Odabrana je prva opcija za dekodiranje, a za procesiranje podataka (mapiranje u Thingsboard format podataka) će se koristiti IoT Gateway. Na The Things Network platformi potrebno je odabrati kreiranu aplikaciju, odjeljak *Payload Formats* te konfigurirati funkciju za dekodiranje (*decoder function*) kao na Slika 4.16. Kao primjer slanja podataka odnosno jednog mjerenja, šalju se **4 bajta** pri čemu svaki bajt odgovara jednoj od veličina koje uređaji mjere – temperatura, pH, provodljivost te zamućenost.



Slika 4.16 *Formatiranje ulaznih podataka*

Također, moguće je konfigurirati i *converter* funkciju koja će dohvaćane vrijednosti staviti unutar određenog raspona vrijednosti ili im promijeniti tip i slično. Kao rezultat ovih funkcija, *payload* koji je poslan sa uređaja unutar LoRaWAN mreže, sada se može konzumirati putem MQTT-a sa The Things Network platforme od strane Thingsboard IoT Gateway-a koji će *mapirati* ulazne podatke i proslijediti ih na Thingsboard platformu.

Za dohvaćanje podataka potrebno je preplatiti se na MQTT *broker* The Things Network platforme sa sljedećim postavkama:

- **Host:** *<Region>.thethings.network*, gdje je <Region> oznaka regije u kojoj se nalazi mreža (zadnji dio *handlera* za koji je registrirana aplikacija, Slika 4.15). U ovom slučaju to je *eu,* te host ima sljedeću adresu: ***eu.thethings.network***
- **Port:** 1883 (8883 za TLS – autorizacija putem certifikata)
- **Vrsta autorizacije:** *basic (username:password)*
- **Korisničko ime:** *wateriot-tb-test*
- **Lozinka:** *application_access_key* (base64)
- **MQTT tema (*uplink*):** <AppID>/devices/<DevID>/up; u ovom slučaju MQTT tema (*topic*) je: ***wateriot-tb-test/devices/waterpoint-lora-1/up***

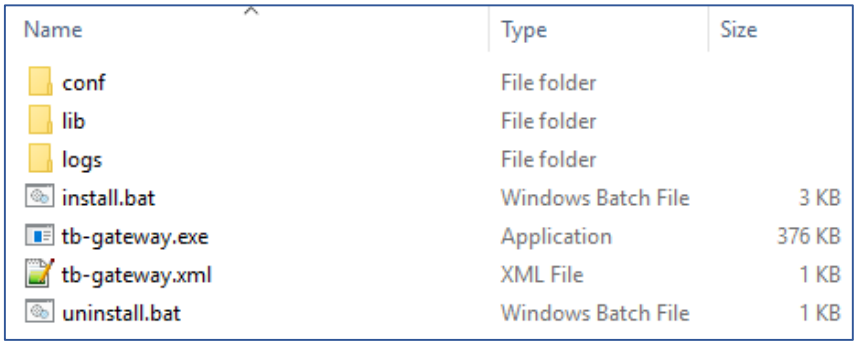
Dohvaćeni podaci odnosno poruka sa The Things Network MQTT API-ja ima sljedeći format:


```
{
    "app_id":"wateriot-tb-test",
    "dev_id":"waterpoint-lora-1",
    "hardware_serial":"001356EBB7CFD432",
    "port":1,
    "counter":0,
    "payload_raw":"GlpXCA==",
    "payload_fields":{
        "conductivity":34.12,
        "pH":7.06,
        "temperature":8.24,
        "turbidity":0.31
    },
    "metadata":{
        "time":"2019-06-03T16:10:02.289899704Z"
    }
}
```


4.4.3. Konfiguracija Thingsboard IoT Gateway-a

Thingsboard IoT Gateway je, dakle, Java aplikacija čija se funkcionalnost određuje konfiguracijskim datotekama te je u mogućnosti slati podatke sa različitih izvora na Thingsboard server putem MQTT-a. Prvi korak je instalacija IoT Gateway-a pri čemu je odabrana Windows platforma gdje se Thingsboard IoT Gateway može instalirati kao Windows usluga (engl. *Windows Service*) koja je spojena na MQTT *broker* konfigurirane TTN LoRa mreže te u pozadini prosljeđuje dohvaćene podatke na Thingsboard IoT platformu.

Za instalaciju IoT Gateway aplikacije nužna je **Java 8** te ju je potrebno imati instaliranu na odabranoj platformi. Na sljedećoj slici prikazane su preuzete i raspakirane datoteke potrebne za instalaciju Thingsboard IoT Gateway-a (*C:\tb-gateway*):



Slika 4.17 *Sadržaj tb-gateway mape*

Za instalaciju Gateway-a kao Windows usluge, potrebno je pokrenuti *install.bat* skriptu. Usluga će se pokretati automatski pri podizanju Windows sustava.

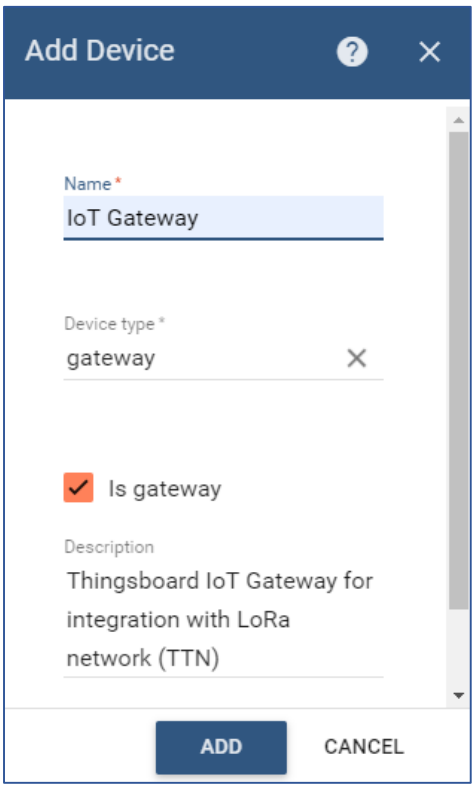
Za pokretanje ili zaustavljanje ***tb-gateway*** aplikacije (usluge) moguće je koristiti *Windows Services* aplikaciju ili putem komandno-linijskih klijenata (administratorski način rada). U drugom slučaju, za pokretanje Thingsboard IoT Gateway usluge potrebno je izvršiti sljedeću naredbu:

```
net start tb-gateway
```

Za zaustavljanje usluge izvršava se sljedeća naredba:

```
net stop tb-gateway
```

Sljedeći korak je dodavanje IoT Gateway-a na Thingsboard platformu. Na pokrenutoj instanci Thingsboard servera (*localhost:9090*) potrebno je prijaviti se sa administratorskim računom te otići na odjeljak sa uređajima (*/devices*). Zatim je potrebno dodati novi uređaj *IoT Gateway* pri čemu je potrebno naznačiti da je riječ od Gateway-u (označiti kućicu *Is Gateway*). Na sljedećoj slici prikazano je dodavanje novog IoT Gateway-a na Thingsboard platformi:



Slika 4.18 *Dodavanje IoT Gateway-a na platformu*

Novostvoreni *pristupnik* – IoT Gateway će komunicirati sa prethodno pokrenutom Thingsboard IoT Gateway uslugom putem MQTT protokola no ponajprije ih je potrebno povezati uređivanjem konfiguracijskih datoteka Thingsboard IoT Gateway aplikacije, pri čemu će biti potreban pristupni token (engl. *access token*) novostvorenog *gateway*-a.

U direktoriju ***conf*** instalirane Thingsboard IoT Gateway usluge (*C:\tb-gateway\conf*) se nalaze konfiguracijske datoteke koje je potrebno urediti za spajanje sa Thingsboard platformom: ***tb-gateway.yml***, te vanjskim MQTT *brokerom* The Things Network platforme za LoRa mrežu: ***mqtt-config.json***.

Za povezivanje sa Thingsboard platformom, u spomenutoj datoteci *tb-gateway.yml*, potrebno je izmijeniti svojstva **gateway.connection.host** i **gateway.connection.port** prema postavkama podignutog Thingsboard (MQTT) servera: **host:** localhost, **port:** 1883.

```
# Gateway configuration
gateways:
  tenants:
    -
      label: "Tenant"
      reporting:
        interval: 60000
      persistence:
        type: file
        path: storage
        bufferSize: 1000
      connection:
        host: "${GATEWAY_HOST:localhost}"
        port: 1883
        retryInterval: 3000
        maxInFlight: 1000
        security:
          accessToken: "${GATEWAY_ACCESS_TOKEN:bOsDIfEXuxaXnRnPPtYT}"
      remoteConfiguration: false
      extensions:
        -
          id: "mqtt"
          type: "MQTT"
          extensionConfiguration: mqtt-config.json

# Server bind address and port
server:
  address: "0.0.0.0"
  port: "9191"

updates:
  enabled: "${UPDATES_ENABLED:false}"
```

Također, napravljene su i izmjene svojstva **server.port** na 9191 budući da je port 9090 na adresi 0.0.0.0 već alociran od strane pokrenute lokalne Thingsboard instance te svojstva **gateway.remoteConfiguration** na *false* budući da će se izmjene konfiguracije odvijati lokalno uređivanjem konfiguracijskih datoteka (tb-gateway nudi i *remote* uređivanje putem Thingsboard Web GUI-a).

Pod poljem **gateway.extensions** navedena je MQTT ekstenzija – ***mqtt.config.json*** (*C:\tb-gateway\conf\mqtt-config.json*), koju je potrebno konfigurirati na sljedeći način (prema podacima navedenim u poglavlju 4.4.2) kako bi se IoT Gateway uspješno povezao sa vanjskim MQTT *brokerom* TTN platforme:

```
{
  "brokers": [{
      "host": "eu.thethings.network",
      "port": 1883,
      "ssl": false,
      "retryInterval": 3000,
      "credentials": {
        "type": "basic",
        "username": "wateriot-tb-test",
        "password": "ttn-account-v2.BXS896cB5kLtRCkIictmHg4ECXz7BSpnPc7fAyDpp0A"
      },
      "mapping": [{
          "topicFilter": "wateriot-tb-test/devices/+/up",
          "converter": {
            "type": "json",
            "filterExpression": "$",
            "deviceNameJsonExpression": "${$.dev_id}",
            "deviceTypeJsonExpression": "${$.hardware_serial}",
            "attributes": [{
                "type": "string",
                "key": "deviceType",
                "value": "${$.hardware_serial}"
            }],
            "timeseries": [
              {
                "type": "double",
                "key": "temperature",
                "value": "${$.payload_fields.temperature}"
              },
              {
                "type": "double",
                "key": "pH",
                "value": "${$.payload_fields.pH}"
              },
              {
                "type": "double",
                "key": "turbidity",
                "value": "${$.payload_fields.turbidity}"
              },
              {
                "type": "double",
                "key": "conductivity",
                "value": "${$.payload_fields.conductivity}"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

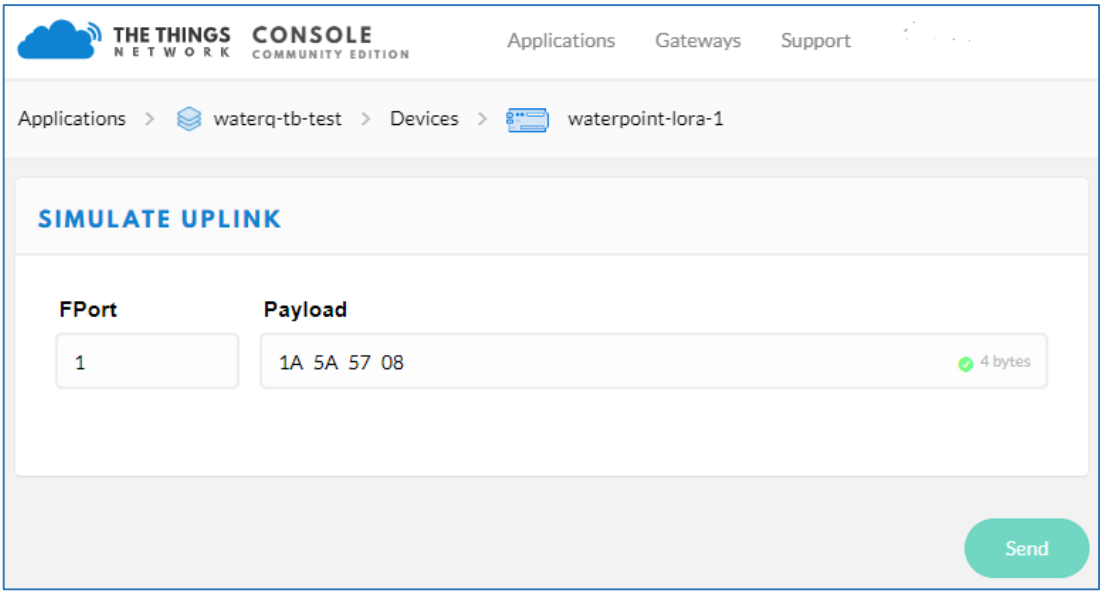
U datoteci *mqtt-config.json*, pod poljem ***mapping***, konstruiran je JSON pretvarač koji podatke sa vanjskog MQTT *brokera* pretvara u format podataka s kojima radi Thingsboard IoT platforma, odnosno, preslikava vrijednosti dohvaćane poruke (JSON objekta) sa TTN platforme u Thingsboard vrijednosti – *attributes* i *timeseries*. Također, ID uređaja se preslikava u naziv uređaja na Thingsboard platformi (DevID -> *deviceName*) te serijski broj uređaja u tip/model uređaja.

Ovako konfiguriran Thingsboard IoT Gateway predstavlja integraciju LoRa mreže sa Thingsboard platformom, odnosno, može se uspješno spojiti na MQTT posrednik The Things Network platforme za pristup podacima sa LoRa uređaja te ih proslijediti/spremiti na Thingsboard IoT platformu u odgovarajućem formatu. Sljedeći korak je **simulacija/testiranje** odabranog rješenja integracije.

4.4.4. Testiranje integracije

Za **testiranje kreirane integracije** LoRa mreže (The Things Network) sa Thingsboard IoT platformom, koristi se *web konzola* za upravljanje LoRa mrežom na The Things Network platformi, odnosno, pomoću nje ćemo simulirati slanje podatka sa LoRa uređaja na mrežni server te bi se ti podaci trebali zatim proslijediti i spremiti na Thingsboard IoT platformu.

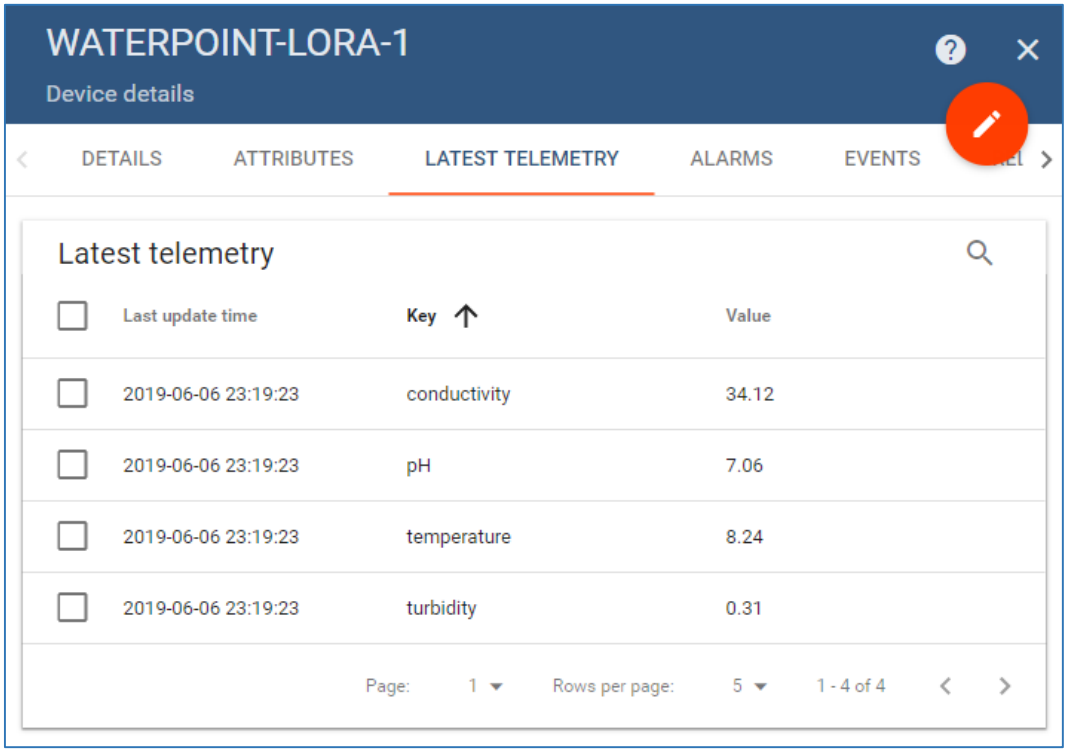
Dakle potrebno je otvoriti spomenutu konzolu, odabrati registriranu testnu aplikaciju *wateriot-tb-test* te otići na odjeljak *Devices* i odabrati registrirani uređaj *waterpoint-lora-1*. Na Slika 4.19 prikazan je odjeljak za simuliranje *uplink*-a odnosno za simuliranje slanja podataka od strane registriranog LoRa uređaja:



Slika 4.19 *Simulacija slanja podataka na LoRa mrežu*

Kao što je spomenuto u konfiguriranju LoRa mreže (4.4.2), *payload* se sastoji od **4 bajta** odnosno šalje se po jedan bajt za svaku mjerenu veličinu kvalitete vode – temperaturu (heksadekadski zapis vrijednosti poslanog bajta): $\boldsymbol{1A_{(16)}}$, provodljivost: $\boldsymbol{5A_{(16)}}$, zamućenost: $\boldsymbol{57_{(16)}}$ te pH vrijednost: $\boldsymbol{08_{(16)}}$.

Funkcije za formatiranje *payload*-a na TTN platformi pretvaraju dobivene vrijednosti za pojedinu veličinu mjerenja u decimalni zapis, konstruira se poruka TTN formata na MQTT brokeru koju dohvaća Thingsboard IoT Gateway putem MQTT ekstenzije, prebacuje podatke u Thingsboard format (*mapping*) te se podaci konačno spremaju na Thingsboard IoT platformu za registrirani LoRa uređaj *waterpoint-lora-1*:



Slika 4.20 *Dohvaćani podaci na Thingsboard platformi*

Ovim tokom podataka je pokazana uspješnost integracije LoRa mreže sa Thingsboard IoT platformom putem odabranog integracijskog rješenja – Thingsboard IoT Gateway. U okvirima demo IoT scenarija za mjerenje kvalitete vode, moguće je dakle postaviti LoRa uređaje velikog dometa i male potrošnje koji će svojim senzorima mjeriti kvalitetu vode na određenim mjestima, slati mjerenja preko LoRaWAN pristupnika (*gateways*) na određeni LoRa mrežni server (poput The Things Network servera) koji zatim omogućava preplatu na pristigle podatke; ti podaci se zatim dohvaćaju putem MQTT protokola (Thingsboard IoT Gateway) i spremaju na odabranu Thingsboard IoT platformu.

Primjer LoRa uređaja koji se uklapa u opisani scenarij razmjene podataka LoRa mrežom je **The Things Uno** – uređaj baziran na Arduino Leonardo razvojnoj pločici sa dodanim modulom za LoRaWAN komunikaciju (*Microchip LoRaWAN*). Na njega je moguće spojiti razne senzore (npr. senzor temperature), konfigurirati *payload*, spojiti i aktivirati ga na The Things Network serveru (određeni *appEUI* i *appKey*) te zatim slati podatke mjerenja senzora. Uređaj se konfigurira i potpuno je kompatibilan sa *Arduino IDE* razvojnim okruženjem.

Slika 4.21 *The Things Uno LoRa uređaj [12]*

Kao primjer mogu poslužiti i bilo koji drugi certificirani LoRaWAN uređaji poput Sentrius RS1xx senzora za mjerenje temperature i vlažnosti zraka tvrtke Laird:

Slika 4.22 *Laird Sentrius RS1xx senzor [14]*

Navedene LoRa uređaje je moguće konfigurirati i postaviti kako bi mjerili određene veličine (u opisanom demo-korisničkom slučaju to su *temperatura, provodljivost, zamućenost i pH vrijednost*) te zatim, te podatke mjerenja, u određenom *payload* formatu, slali LoRa bežičnom tehnologijom do LoRaWAN pristupnika (*gateways*). Primjer jednog od komercijalno-dostupnih

LoRaWAN pristupnika je Sentrius RG1xx LoRa Gateway koji predstavlja sigurno, skalabilno i robusno LoRa rješenje za privatne LoRaWAN mreže, nudeći LoRa domet oko 15km te je kompatibilan za jednostavnu instalaciju sa raznim LoRa mrežnim serverima uključujući The Things Network. Gateway je prikazan na sljedećoj slici:

Slika 4.23 *Laird Sentrius RG1xx LoRa Gateway [15]*

Podaci mjerenja koji dolaze od LoRa uređaja prolaze kroz pristupnike poput Sentrius RG1xx LoRa Gateway-a koji na temelju primljenih podataka sa uređaja, prosljeđuje mjerenja u odgovarajuću LoRa mrežu tj. na odgovarajući LoRa server, odakle ih je moguće konzumirati na odabranoj Thingsboard IoT platformi (pomoću integracija ili IoT Gateway-a).

5. ZAKLJUČAK

Pod utjecajem ubrzanog tehnološkog napretka današnjice, razne tehnologije se razvijaju i primjenjuju u gospodarstvu i svakodnevnom životu u cilju poboljšanja životnog standarda, povećanja produktivnosti, efikasnosti, olakšavanja određenih zadataka, modernizacije usluga i slično. To je cilj i tehnologije Interneta stvari. Stručnjaci (Cisco Systems, [3]) procjenjuju da će do 2020. godine biti preko 50 milijardi uređaja (stvari) spojeno na Internet, što će znatno promijeniti način na koji ljudi i poduzeća djeluju u svojoj okolini. Praćenjem i upravljanjem objektima iz našeg okruženja stvara se uža veza fizičkog svijeta i računala što omogućava nove usluge i poboljšanja u vidu bolje efikasnosti, preciznosti i automatizacije u raznim domenama ljudskog djelovanja i svakodnevnog života.

Tema ovog rada je jedna od mogućih primjena tehnologije Interneta stvari (IoT) – za udaljeno praćenje mjerenja kvalitete vode za različite lokacije (izvore vode) što u konačnici omogućava krajnjem korisniku praćenje kvalitete vode određenog zemljopisnog područja u određenom vremenskom razdoblju. U radu su objašnjene sve potrebne tehnologije i koncepti (tehnologija Interneta stvari, slojevi IoT-a, IoT komunikacijski modeli i protokoli, RESTful i WebSocket web usluge, kontejnerska tehnologija Docker) za realizaciju zadanog IoT sustava i navedeni svi potrebni koraci za primjenu tehnologije Interneta stvari za udaljeno praćenje mjerenja senzora – instalacija odabrane IoT platforme, kreiranje modela, rezerviranje (engl. *provisioning*) IoT entiteta i njihovih relacija na platformi, dodavanje novih uređaja na platformu, slanje podataka sa IoT uređaja različitim protokolima koji su najčešće korišteni u IoT-u poput MQTT i CoAP protokola uz praktične primjere Arduino programa te dohvaćanje i prikaz podataka mjerenja kvalitete vode. S obzirom na rastuću popularnost LoRa tehnologije i uređaja, rad pokriva i načine integracije odabrane IoT platforme sa LoRa mrežom te je objašnjena i implementirana integracija putem IoT Gateway-a koji dohvaća podatke pretplatom na vanjski MQTT broker LoRa mrežnog servera. Radom su, dakle, pokrivene sve osnovne cjeline za razumijevanje tehnologije Interneta stvari te je ista primijenjena za realizaciju određenog IoT sustava za udaljeno praćenje mjerenja senzora.

LITERATURA

[1] A. Bahga; V. Madisetti: „Internet of Things: A Hands-On Approach“, Bahga & Madisetti, 2017.

[2] A. McEwen; H. Cassimally: „Designing the Internet of Things“, John Wiley and Sons, Ltd., 2014.

[3] D. Hanes; G. Salguiero; P. Grossetete; R. Barton; J. Henry: „IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for Internet of Things“, Cisco Press, 2017.

[4] Docker, https://docs.docker.com/engine/docker-overview (25.6.2019)

[5] Comparison of Docker and Virtual Machines, https://www.researchgate.net/figure/A-comparison-of-the-architecture-of-virtual-machines-and-Docker-software (28.8.2019)

[6] Thingsboard, https://thingsboard.io/docs (24.6.2019)

[7] Thingsboard Swagger UI, https://demo.thingsboard.io/swagger-ui.html (27.6.2019)

[8] Mainflux, https://mainflux.readthedocs.io/en/latest (27.6.2019)

[9] MQTT, http://mqtt.org/ (30.8.2019)

[10] Croduino NOVA2, https://e-radionica.com/en/croduino-nova2.html (31.8.2019)

[11] CoAP, https://tools.ietf.org/html/rfc7252 (1.9.2019)

[12] The Things Network, https://www.thethingsnetwork.org/docs (28.6.2019)

[13] LoRaWAN, https://iot.ttu.ee/lora-wan-ttu-linnakus (29.6.2019)

[14] Laird Sentrius RS1xx LoRa, https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rs1xx-lora-enabled-sensors (28.8.2019)

[15] Laird Sentrius RG1xx LoRa, https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rg1xx-lora-enabled-gateway-wi-fi-bluetooth-ethernet (28.8.2019)

SAŽETAK

Tema ovog rada je primjena tehnologije Interneta stvari (IoT) za udaljeno praćenje mjerenja senzora u okvirima definiranog demo korisničkog slučaja – praćenje kvalitete vode za različite izvore pomoću IoT tehnologije pri čemu je potrebno konfigurirati određenu IoT platformu kako bi se na nju mogli slati, spremati te dohvaćati podaci mjerenja (*fizikalna* svojstva vode, temperatura, zamućenost, provodljivost i pH vrijednost) za različite lokacije odnosno točke mjerenja kvalitete vode (*water points*) koje hijerarhijski pripadaju određenom gradu/regiji. Na temelju analize i usporedbe dostupnih postojećih rješenja (Thingsboard i Mainflux), odabrana je odgovarajuća IoT platforma za zadani IoT scenarij – Thingsboard IoT. Platforma je instalirana lokalno pomoću Docker kontejnerske tehnologije. Na temelju opisanog korisničkog slučaja kreiran je IoT model te su rezervirani IoT entiteti i kreirane su odgovarajuće relacije između entiteta prema zadanom modelu koristeći Thingsboard REST API i Postman REST klijent. Objašnjen je postupak stvaranja nove točke mjerenja kvalitete vode te načini slanja podataka za određenu točku mjerenja kvalitete vode, različitim protokolima – HTTP, MQTT i CoAP uz praktične primjere za Arduino razvojnu pločicu. Kreirani IoT sustav je također integriran sa LoRa mrežom (za simulaciju LoRa mreže i LoRa uređaja je korištena platforma The Things Network) pomoću Thingsboard IoT Gateway-a koristeći vanjski MQTT broker za prosljeđivanje podataka mjerenja te je zatim testirana kreirana integracija.

Ključne riječi: Internet stvari, IoT platforma, komunikacijski protokoli, HTTP, MQTT, CoAP, WebSocket, REST, IoT uređaj, klijent, poslužitelj, Arduino, Thingsboard, Mainflux, Docker, The Things Network, LoRaWAN, senzori

ABSTRACT

Application of technology of the Internet of things (IoT) for remote monitoring of sensor measurements

The topic of this paper is the application of Internet of Things (IoT) technology to remotely monitor sensor measurements within a defined demo use-case - monitoring water quality for different sources using IoT technology, with the need to configure a specific IoT platform to send, store and retrieve measurement data (physical properties of water - temperature, turbidity, conductivity and pH) for different locations or points of water quality measurement (*water points*) that hierarchically belong to a particular city/region. Based on the analysis and comparison of available existing solutions (Thingsboard and Mainflux), the appropriate IoT platform was selected for the defined IoT scenario - Thingsboard IoT. The platform was installed locally using Docker container technology. Based on the described use-case, an IoT model was created, IoT entities provisioned and corresponding relations created between entities based on the created model, using the Thingsboard REST API and the Postman REST client. The process of creating a new water point and ways of sending measurements for a particular water point, using different protocols (HTTP, MQTT and CoAP) are explained, along with practical examples for the Arduino development board. The created IoT system was also integrated with the LoRa network (The Things Network is used to simulate LoRa network and LoRa devices) using the Thingsboard IoT Gateway with an external MQTT broker to pass the measurements from LoRa network, after which the created integration was tested.

Keywords: Internet of things, IoT platform, communication protocols, HTTP, MQTT, CoAP, WebSocket, REST, IoT device, client, server, Arduino, Thingsboard, Mainflux, Docker, The Things Network, LoRaWAN, sensors

ŽIVOTOPIS

Slaven Ivić rođen je 09.10.1995. godine u Münchenu, Njemačka. Osnovnoškolsko i srednjoškolsko obrazovanje završio je u Odžaku, Bosna i Hercegovina, gdje je sudjelovao na raznim izvannastavnim aktivnostima i osvajao nagrade na raznim školskim i županijskih natjecanjima iz matematike i fizike te je bio proglašen i učenikom generacije po završetku osnovnoškolskog obrazovanja. Svoje obrazovanje nastavlja u Osijeku gdje 2014. godine upisuje preddiplomski studij Elektrotehnike na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija (FERIT) te se nakon prve godine studija opredjeljuje za smjer Komunikacije i informatika. Kao student FERIT-a, sudjeluje više puta na međunarodnom natjecanju studenata elektrotehnike – Elektrijada, predstavljajući Fakultet u znanju i sportu, drži demonstrativnu nastavu na Fakultetu iz različitih kolegija te 2016. godine dobiva i dekanovu nagradu kao najbolji student na svoje smjeru (za uspješnost u studiranju). Također, tijekom studiranja, radi na raznim projektima, među kojima je i projekt fakultetskog natječaja Pro-Student: „*Feritoskop* – Android bežični osciloskop s pomoću Arduino mikroprocesora“ kojeg uspješno realizira za što dobiva nagradu Fakulteta. 2017. godine završava preddiplomski studij Elektrotehnike sa završnim radom na temu „Kontrola kvalitete pruženih usluga u turizmu putem informacijsko komunikacijskih tehnologija“, u sklopu kojeg izrađuje Android aplikaciju za anketiranje posjetitelja Nacionalnog parka Paklenica, čime uspješno stječe titulu prvostupnika Elektrotehnike. Iste godine upisuje sveučilišni diplomski studij Elektrotehnike, smjer Komunikacije i informatika, izborni blok Mrežne tehnologije. 2018. godine se prijavljuje na Ljetni kamp (*Summer Camp*) tvrtke Ericsson Nikola Tesla d.d. u Zagrebu, gdje u profesionalnoj radnoj atmosferi i timskom okruženju, u trajanju od 5 tjedana, uspješno završava projekt kampa vezan za prikupljanje različitih podataka mjerenja iz okoliša pomoću IoT platforme. Obaveznu stručnu praksu na posljednjoj godini diplomskog studija također odrađuje u spomenutoj tvrtki, gdje radi na razvoju Android aplikacije za određenu geografsko-informacijsku primjenu. 2019. godine dobiva Rektorovu nagradu za izvrstan seminarski rad iz kolegija Internet objekata pod nazivom „*SmartLock*“ – sustav pametne brave za dijeljenje gradskih bicikala pomoću Android aplikacije. 2019. godine ponovno dobiva nagradu Fakulteta za uspješnost u studiranju.

Slaven Ivić

PRILOZI

- CD
  - Elektronička verzija rada (dokument u *.docx* i *.pdf* formatu)
  - Thingsboard REST API (Postman collections)
  - Thingsboard IoT *backup* spremljenih podataka (*backup.tar*)
  - Thingsboard IoT Gateway sa konfiguracijskim datotekama
  - Arduino primjeri (*.ino* datoteke) i biblioteke za slanje podataka MQTT i CoAP protokolom

- Snimljeni paketi u *Wireshark*-u pri slanju podataka na Thingsboard IoT platformu za:
  - MQTT protokol:

mqtt

No.	Time	Source	Destination	Protocol	Length	Info
11	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
12	0.019379	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
14	0.022112	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
15	0.022113	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
34	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
35	0.014901	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
37	0.038633	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
38	0.038634	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
47	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
48	0.019779	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
50	0.027059	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
51	0.027767	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
60	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
61	0.016298	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
63	0.020147	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
64	0.020703	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
72	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
73	0.023498	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
75	0.029575	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
76	0.030668	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
87	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
88	0.019113	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
90	0.021892	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
91	0.021892	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
98	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
99	0.018276	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
101	0.024195	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
102	0.024196	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
111	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
112	0.033021	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
114	0.043151	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
115	0.043152	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
131	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
132	0.020179	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
134	0.027754	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
135	0.027755	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
161	*REF*	192.168.2.9	192.168.2.15	MQTT	96	Connect Command
162	0.021491	192.168.2.15	192.168.2.9	MQTT	58	Connect Ack
164	0.035698	192.168.2.9	192.168.2.15	MQTT	155	Publish Message [v1/devices/me/telemetry]
165	0.035699	192.168.2.9	192.168.2.15	MQTT	56	Disconnect Req
AVG:	0.029258					

Prilog 1 *Slanje podataka MQTT protokolom - snimljeni paketi*

– CoAP protokol:

coap

No.	Time	Source	Destination	Protocol	Length	Info
79	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:4841, POST, TKN:c3 ed 6e 34, /api/v1/wp2/telemetry
80	0.007456	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:4841, Empty Message
81	0.023810	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27603, 2.03 Valid, TKN:c3 ed 6e 34, /api/v1/wp2/telemetry
91	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:46739, POST, TKN:15 70 e1 e3, /api/v1/wp2/telemetry
92	0.006864	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:46739, Empty Message
93	0.010850	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27604, 2.03 Valid, TKN:15 70 e1 e3, /api/v1/wp2/telemetry
98	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:16154, POST, TKN:3e 08 f0 c4, /api/v1/wp2/telemetry
99	0.005481	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:16154, Empty Message
100	0.016581	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27605, 2.03 Valid, TKN:3e 08 f0 c4, /api/v1/wp2/telemetry
107	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:17062, POST, TKN:cb f0 09 87, /api/v1/wp2/telemetry
108	0.006781	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:17062, Empty Message
109	0.015797	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27606, 2.03 Valid, TKN:cb f0 09 87, /api/v1/wp2/telemetry
114	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:41782, POST, TKN:07 39 83 8a, /api/v1/wp2/telemetry
115	0.002846	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:41782, Empty Message
116	0.018213	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27607, 2.03 Valid, TKN:07 39 83 8a, /api/v1/wp2/telemetry
127	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:48934, POST, TKN:31 34 d2 5c, /api/v1/wp2/telemetry
128	0.002493	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:48934, Empty Message
129	0.006475	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27608, 2.03 Valid, TKN:31 34 d2 5c, /api/v1/wp2/telemetry
134	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:24979, POST, TKN:89 84 b4 c5, /api/v1/wp2/telemetry
135	0.006058	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:24979, Empty Message
137	0.018812	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27609, 2.03 Valid, TKN:89 84 b4 c5, /api/v1/wp2/telemetry
142	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:1355, POST, TKN:b3 b2 2a d6, /api/v1/wp2/telemetry
143	0.006618	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:1355, Empty Message
144	0.020424	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27610, 2.03 Valid, TKN:b3 b2 2a d6, /api/v1/wp2/telemetry
150	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:28549, POST, TKN:98 fe 66 8a, /api/v1/wp2/telemetry
151	0.005382	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:28549, Empty Message
152	0.016063	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27609, 2.03 Valid, TKN:89 84 b4 c5, /api/v1/wp2/telemetry
162	*REF*	192.168.2.9	192.168.2.15	CoAP	145	CON, MID:33351, POST, TKN:00 53 9f 13, /api/v1/wp2/telemetry
163	0.004871	192.168.2.15	192.168.2.9	CoAP	46	ACK, MID:33351, Empty Message
164	0.015735	192.168.2.15	192.168.2.9	CoAP	50	CON, MID:27612, 2.03 Valid, TKN:00 53 9f 13, /api/v1/wp2/telemetry
AVG:	0.016276					

Prilog 2 *Slanje podataka CoAP protokolom - snimljeni paketi*

– HTTP protokol:

http

No.	Time	Source	Destination	Protocol	Length	Info
9	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
10	0.032717	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
21	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
23	0.059175	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
34	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
35	0.030852	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
50	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
52	0.050631	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
62	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
63	0.044827	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
76	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
77	0.024646	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
90	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
92	0.043428	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
103	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
104	0.037142	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
117	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
120	0.043899	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
137	*REF*	192.168.2.9	192.168.2.15	HTTP	279	POST /api/v1/wp2/telemetry HTTP/1.1 (application/json)
138	0.029316	192.168.2.15	192.168.2.9	HTTP	286	HTTP/1.1 200
AVG:	0.039663					

Prilog 3 *Slanje podataka HTTP protokolom - snimljeni paketi*