Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/09BE07Sicar

VIZUELIZACIJA PODATAKA DOBIJENIH MJERENJIMA U VEB OKRUŽENJU

VISUALIZATION OF DATA OBTAINED FROM MEASUREMENTS IN A WEB ENVIRONMENT

Luka Šicar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U radu je opisana problematika i načini vizuelizacije podataka dobijenih mjerenjima sa senzora putem veb aplikacije. Komunikacija između senzora i servera je implementirana pomoću MQTT protokola, a vizuelizacija upotrebom jQWidgets biblioteke.*

Ključne reči *MQTT, senzor, jQWidgets*

Abstract – *This paper presents problematic and ways of visualizing data obtained from sensors via web application. Communication between sensors and server is implemented through MQTT protocol and visualisation using jQWidgets library.*

Keywords: *MQTT, sensor, jQWidgets*

1. UVOD

Senzori su svuda. *Internet of Things* (IoT) paradigma se bazira na svijedu ispovezanih objekata, koji imaju mogućnost komunikacije između sebe i sakupljanju podataka o njihovom kontekstu [1].

Razmatrajući podatke sakupljene sa senzora, stvara se potreba za dizajniranjem upravljačkih tabli kako sirovi podaci postaju beskorisni za korisnika [2].

Vizuelizacija je bitna kada se govori o podacima sa senzora jer ona umnogome olakšava korisniku rad sa podacima i podaci se mogu razumjeti brže i lakše [3].

Cilj ovog rada jeste da se daju smjernice za nalaženje ogovarajućeg rješenja za specifične podatke sa senzora. U okviru ovog rada prikazana je veb aplikacija za vizuelizaciju podataka dobijenih sa senzora sa posebnim osvrtom na MQTT komunikacijski protokol.

2. MQTT

MQTT (Message Queue Telemetry Transport) je protokol za slanje poruka koji pruža klijentima na mreži sa ograničenim resursima jendostavan način distribucije informacija dobijenim mjerenjima na nepristupačnim tačkama. Protokol koji koristi *publish subscribe* komunikacioni obrazac upotrebljava se za *machine-to-machine* komunikaciju i igra bitnu ulogu u IoT.
MQTT omogućava IoT uređajima sa ograničenim resursima da šalju ili objavljuju informacije o određenoj temi serveru koji radi na principut MQTT posrednika za poruke. Posrednik onda objavljuje informacije klijentima koji su se prethodno prijavili na tu temu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Branko Milosavljević

Tema izgleda kao hijerarhijska putanja do fajla. Klijent može da se pretplati na određen nivo u hijerarhiji teme ili da koristi *wild-card* karakter da se pretplati na više nivoa.

MQTT protokol je odličan izbor za bežične mreže koje imaju problem sa kašnjenjem zbog ograničenja u protoku ili nepouzdane konekcije. Ako se konekcija između pretplatnog klijenta i posrednika prekine, posrednik će i dalje generisati poruke i poslati ih klijentu kada se konekcija ponovo uspostavi. Ako se konekcija od klijenta do posrednika prekine bez razloga, posrednik može da zatvori konekciju i pošalje klijentima keširanu poruku sa instrukcijama.

2.1 Način rada

MQTT sesija je podijeljena u četiri faze: konekcija, autentifikacija, komunikacija i prekid. Klijent započinje kreiranjem TCP/IP konekcije ka posredniku koristeći ili standardni port ili poseban port definisan posrednikovim operatorima. Kada se kreira konekcija, važno je primjetiti da server može da nastavi staru sesiju ako pri kreiranju konekcije se koristi već korišćen klijentov id.

Svaka poruka koja se šalje po ovom protokolu se sastoji od fiksnog zaglavlja od dva bajta koji je opcion i porukom koja je ograničena na 256 MB informacija i QoS (quality of service) nivoom. QoS se odnosi na bilo koju tehnologiju koja upravlja prometom podataka da smanji gubljenje paketa, kašnjenje mreže i nestabilnost mreže. QoS kontroliše i upravlja mrežnim resursima određujući prioritete za određene tipove podataka na mreži.

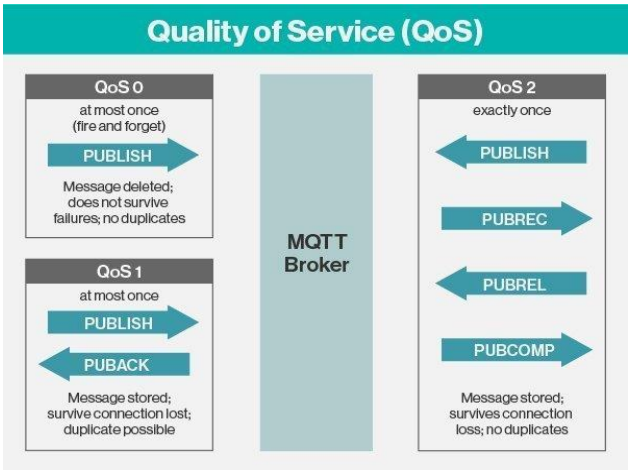
Postoje tri različita tipa *quality of service* nivoa koji određuju kako je sadržaj upravljan d strane MQTT protokola. Iako su viši nivoi QoS-a pouzdaniji, imaju veće kašnjenje i traže bolji protok, pa klijenti koji se pretplaćuju mogu da biraju najviši QoS nivo koji bi željeli da prime.

Najjednostavniji QoS nivo je nepriznata usluga. Ovaj QoS nivo koristi PUBLISH (objava) sekvencu paketa. Izdavač šalje poruku posredniku jednom i posrednik proljeđuje poruku pretplatnicima jednom. Ne postoji mehanizam koji bi se pobrinuo da je poruka primljena i posrednik ne čuva poruku. Ovaj QoS nivo se može još nazvati „najviše jednom“, QoS0 ili ispali i zaboravi.

Drugi QoS nivo je priznata usluga. Ovaj QoS nivo koristi PUBLISH/PUBACK sekvencu paketa između izdavača i njegovog posrednika, kao i između posrednika i pretplatnika.

Priznati paket provjerava da li je sadržaj primljen i mehanizam ponovnog pokušaja će poslati prvobitni sadržaj ponovo ako potvrda o dospijeću nije primljena u

određenom vremenskom roku. Ovo može da rezultuje da pretplatnik primi više kopija iste poruke. Ovaj QoS nivo se zove još i „barem jedanput“ ili QoS1.



Slika 1. – *MQQT nivoi kvaliteta servisa [4]*

Treći QoS nivo je uvjerena usluga. Ovaj QoS nivo dostavlja poruke u dva para paketa.

Prvi par je nazvan PUBLISH/PUBREC, a drugi par se zove PUBREL/PUBCOMP. Dva para obezbjeđuju da, će poruka biti dostavljena samo jednom, nebitno od broja pokušaja. Ovaj QoS nivo se još zove i „samo jednom“ ili QoS2.

Tokom faze komunikacije klijent može da izvrši objavu, pretplatu, ukidanje pretplate i ping operacije. Operacija objave šalje binarni blok podataka to jest sadržaj temi koja jedefinisana od strane izdavača.

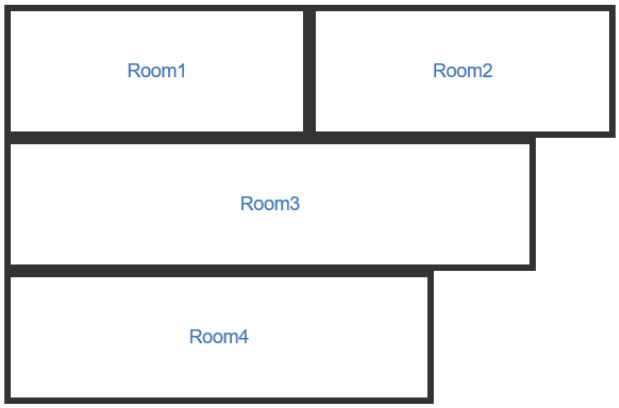
MQTT podržava BLOB (binary large object) do 256 MB veličine. Format sadržaja je specifičan od aplikacije do aplikacije. Pretplate na teme se obavljaju upotrebom SUBSCRIBE/SUBACK para paketa. Odjavljivanje sa pretplate se slično obavlja upotrebom UNSUBSCRIBE/UNSUBACK para paketa.

Stringovi tema formiraju prirodno stablo tema korišćenjem specijalnog karaktera odvajanja, kosom crtom (/). Klijent može da se pretplati i odjavi sa pretplate na cijele grane u stablu tema korišćenjem specijalnih *wild-card* karaktera. Postoje dva specijalna karaktera: specijalni karakter jednog nivoa (+) i specijalni karakter više nivoa (#). Specijalni tema karakter ($) se koristi za transport sistemskih ili poruka specifičnim za server.

Četvrta operacija koju klijent može izvršiti tokom faze komunikacije je da „pinguje“ prenosni server putem PINGREQ/PINGRESP sekvence paketa koja se grubo prevodi kao DA LI SI ŽIV/JESAM, ŽIV SAM. Ova operacija nema drugu funkciju nego da održava konekciju živom i osigura da TCP konekcija nije ugašena od strane *gateway*-a ili rutrea.

4. IMPLEMENTACIJA SISTEMA

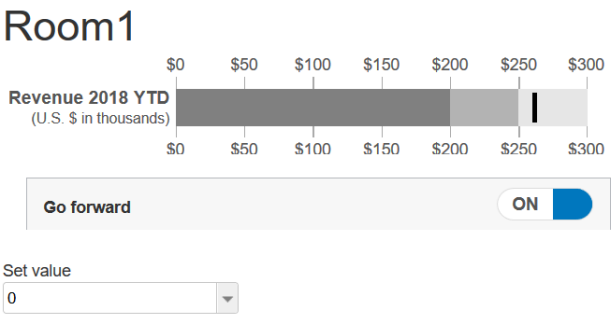
U ovom odjeljku će biti opisana implementacija sistema. Za ovaj projekat odabrano je .NET Core softversko razvojno okruženje [5]. Ideja projekta je bila da se prikažu mjerenja sa senzora putem veb aplikacije. Server je urađen kao ASP.NET Core MVC aplikacija sa jednostavnim interfejsom koji prikazuje raspored soba gdje u svakoj prostoriji postoje različiti senzori čija mjerenja se očitavaju i prikazuju putem prikladnih UI komponenti.



Slika 2. – *Jednostavan grafički interfejs aplikacije*

Pošto server treba da obavjesti klijenta da je došlo do novih očitavanja, komunikacija između servera i klijenta se odvija preko *WebSocket*-a. *WebSocket* predstavlja komunikacijski protokol, koji pruža kanale za dvosmjernu komunikaciju preko jedne TCP konekcije [6].

Kada korisnik otvori stranicu prostorije, odmah se šalje *WebSocket handshake* ka serveru da se uspostavi dvosmjerna komunikacija sa serverom da bi klijent bio ažuriran o eventualnim promjenama vrijednosti mjerenja sa senzora. Osim zahtjeva za uspostavljanje *WebSocket* konekcije, registruje se i funkcija koja će primati *WebSocket* poruke sa servera i ažurirati vrijednosti senzora na UI komponenti. Primjer prve prostorije koja ima samo jedan senzor dat je na slici 3.



Slika 3. – *Soba br.1*

Podaci koji se šalju putem *WebSocket*-a mogu da se protumače na tri načina koja su definisana putem enumeracije WebSocketMessageType.

Tri formata poruka su tekst, koji može da predstavlja čisti tekst, ili može da bude JSON, XML ili neki drugi tekstualni standard za razmjenu podataka, zatim binarni format i format *close* koji označava da je primanje poruke završeno.

Implementacija MQTT protokola za .NET Core uzeta je sa github-a, projekat je dostupan na https://github.com/chkr1011/MQTTnet. Da bi se kreirao MQTT server potrebno je da se putem *pipeline*-a doda na Kestrel aplikativni server koji koristi .NET Core što će biti prikazano na listingu 1.

```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }
    public static IWebHostBuilder
CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
         UseKestrel(o =>
        {
          // mqtt pipeline
           o.ListenAnyIP(1883, l => l.UseMqtt());
        })
        .UseStartup<Startup>();
}
```

Listing 1. – *Podrška za MQTT server*

Zatim se u Startup.cs klasi u ConfigureServices metodi postave svi servisi koje MQTT server koristi i takođe podese opcije MQTT servera.

Sledeći listing, pod brojem 2 pokazuje kako izgleda ConfigureServices metoda

```
// This method gets called by the runtime. Use this
method to add services to the container.
public void ConfigureServices(IServiceCollection
services)
{
    var mqttServerOptions = new
MqttServerOptionsBuilder()
        .WithoutDefaultEndpoint()
        .Build();
    services
        .AddHostedMqttServer(mqttServerOptions)
        .AddMqttConnectionHandler()
        .AddConnections();

    services.AddSingleton<IMqttServer>(s =>
s.GetService<MqttHostedServer>());
}
```

Listing 2. – *ConfigureServices metoda Startup klase*

Potom kod *application builder*-a registrujemo MQTT server. Tu *override*-ujemo metode potrebne za funkcionisanje servera. Za potrebe ove aplikacije definisano je ponašanje tri funkcije ali postoji veliki broj događaja koje server može da osluškuje.

Prvi događaj na koji server reaguje jeste kada se server podigne.

Drugi osluškivač implementiran je kada se MQTT klijent poveže prvi put sa MQTT serverom.

Ovdje se izvlači podatak o jedinstvenom identifikatoru klijenta i zatim se kreira skladište za njegova mjerenja. Treća metoda koja je implementirana hvata sve poruke koje dolaze kako sa MQTT klijenata tako i sa MQTT servera upućene klijentu. Pošto MQTT server djeluje kao čvorište preko njega idu sve poruke, pa tako i one koje on sam šalje.

Te poruke možemo razlikovati tako što one ne sadrže id klijenta. One će biti poslate ne direktno klijentu sa određenim identifikatorom već na određenu temu svim klijentima koji su pretplaćeni na istu.

Pošto nemamo prave senzore, koriste se konzolne aplikacije koje imitiraju senzore tako što na određeni interval generišu vrijednosti koje predstavljaju simulaciju mjerenja i zatim ih šalju ka serveru.

Pri kreiranju MQTT klijenta potrebno je podesiti neke parametre. Ti parametri uključuju putanju do servera, to jest njegova adresa i tema koja je registrova u *pipeline*-u. Specificiran je takođe i *WebSocket* komunikacijski protokol, postavlja se jedinstven klijentov identifikator. Potom se obavještava klijent da će se koristiti sigurna TCP veza putem TLS (transport layer security).

Ovo znači da će protokol biti prebačen sa ws:// na wss://. Kreiranje MQQT klijenta će biti prikazano na listingu 3.

```
var factory = new MqttFactory();
var options = new MqttClientOptionsBuilder()
    .WithWebSocketServer("localhost:44390/mqtt")
    .WithClientId("senzor1")
    .WithTls()
    .Build();
var mqttClient = factory.CreateMqttClient();
```

Listing 3. – *Kreiranje MQTT klijenta i podešavanje njegovih opcija*

Postoji i podrška za identifikaciju klijenta putem korisničkog imena i lozinke kao i podrška za rad sa sertifikatima. Nakon što su podešene opcije MQTT klijenta, konekcija može biti uspostavljena. Ako je veza sa serverom izgubljena, *Disconnected* događaj se okida. Ovaj događaj je takođe okinut i ako je poziv za konekciju propao jer je server nedostupan.

Obrada ovog događaja dozvoljava pozivanje *ConnectAsync* metode samo jednom i zatim odvojeno bavljenje ponovnim pokušajima preko *Disconnected* događaja.

Potrebno je i implementirati metodu koja obrađuje događaj kada klijent primi poruku. Korisnik ima opciju preko aplikacije da utiče na senzor kao na primjer da li će generisati vrijednosti veće za jedan od prethodne ili manje, ili da promjeni referentnu vrijednost.

Slanje poruka iz metode koja obrađuje poruke zahtjeva upotrebu Task.Run metode kada se koristi QoS nivo veći od 0.

Razlog ovome je što obrađivač poruka mora prvo da završi prije nego što je sledeća poruka primljena. Svrha ovog ponašanja je da bi se sačuvao red primljenih poruka.

Da bi klijent uopšte mogao da prima poruke sa servera, mora da pute pretplaćen na određenu temu. Pretplata na temu je moguća nakon što je veza sa serverom uspostavljena.

5. ZAKLJUČAK

U ovom radu rješavana je problematika i načini vizuelizacije podataka dobijenih mjerenjima sa senzora putem veb aplikacije.

Početni korak ka rješavanju ovog problema bio je istraživanje radova koji su se bavili sličnom tematikom. Kako je *Internet of Things* tema koja uživa veliku popularnost danas, postoji ogroman broj članaka koji se bave istom. Na osnovu ovih radova došlo se do zaključka da veb aplikacija mora da odgovori na visoku frekvenciju očitavanja senzora kao i da ima odgovarajuće komponente korisničkog interfejsa koje bi vjerno prikazale ta očitavanja. Takođe bilo bi poželjno da aplikacija bude *responsive* to jest da može jednako dobro da radi na svim veličinama uređaja od mobilnih telefona preko računara do video bimova.

Na osnovu ovih nalaza krenulo se u analizu *front-end* biblioteka koje bi odgovorile ovim zahtjevima, da imaju veliki broj raznovrsnih komponenti koje su prikladne za prikazivanje očitavanja senzora. Da te komponente budu iscrtane u vektorskoj grafici tako da povećavanjem rezolucije ili veličine ekrana ne bi došlo do gubljenja kvaliteta slike i da mogu da osvježavaju prikaz vrlo brzo zbog visoke frekvencije očitavanja senzora. Istraživanje je pokazalo da postoji samo par biblioteka koje odgovaraju na ove zahtjeve i pri tom imaju licencu za besplatno korišćenje. Između ovih biblioteka odabir je pao na jQWidgets koja ima najbolje dokumentovanu funkcionalnost.

Nakon odabira tehnologije u kojoj će biti prikazivani podaci mjerenja sa senzora ostalo je još da se odabere i implentira komunikacijski protokol između senzora i servera. Kako takođe komunikacija između senzora i servera mora da bude brza kao između servera i veb aplikacije, brzina slanja podataka je bila jedan od faktora pri odabiru odgovarajućeg protokola. Takođe protokol bi morao da odgovori na konekcije sa udaljenih lokacija odakle je potrebno malo koda da bi se ostvarilo slanje podataka i da poruke koje se šalju budu što manje, to jest da se protokol podredi senzorima koji imaju ograničene resurse. Sve ovo upravo podržava MQTT protokol koji je pravljen za komunikaciju *machine-to-machine* odnosno za *Internet of Things* komunikaciju.

Kako je tema i fokus ovog rada bio na uspješnoj vizuelizaciji očitavanja koja dolaze sa senzora, senzori su u ovoj implementaciji simulirani kao konzolne aplikacije koje su slale nasumično generisane podatke.

Jedan pravac u daljem razvitku ovog rješenja bio bi da se umjesto simuliranih senzora zaista iskoriste pravi senzori i implementira komunikacija između njih i servera.

6. LITERATURA

[1] Internet of things, https://en.wikipedia.org/wiki/Internet_of_things

[2] Ivan Logre, Sebastian Mosser, Phillipe Collet, Michel Riveill. Sensor Data Visualisation: A Composition-Based Approach to Support Domain Variability. European Conference on Modelling Foundations and Applications (ECMFA 2014), Jul 2014, York, United Kingdom.

[3] Christian Richter, Visualizing Sensor Data

[4] Margaret Rouse, MQTT (MQ Telemetry Transport) https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport

[5] .NET Core https://dotnet.microsoft.com

[6] WebSocket https://en.wikipedia.org/wiki/WebSocket

Kratka biografija:

Luka Šicar rođen je 03.08.1994. godine u Prijedoru. Osnovnu školu „Vuk Karadžić“ je završio 2009. godine, srednju školu „Petar Kočić“ u Novom Gradu, završio je 2013. godine. Iste te godine upisao je „Fakultet tehničkih nauka“ u Novom Sadu, odsjek računarstvo i automatika. Diplomirao je u roku 2017. godine i upisao master studije iste godine.

kontakt: sicluka@yahoo.com