VELEUČILIŠTE U VIROVITICI
Preddiplomski stručni studij Elektrotehnika

HRVOJE ĐAKOVIĆ

PRAĆENJE VRIJEDNOSTI UDALJENIH SENZORA PRIMJENOM MQTT PROTOKOLA I VIZUALIZACIJA DOBIVENIH VRIJEDNOSTI PRIMJENOM WEB TEHNOLOGIJA

ZAVRŠNI RAD

VIROVITICA, 2021.

VELEUČILIŠTE U VIROVITICI
Preddiplomski stručni studij Elektrotehnika

PRAĆENJE VRIJEDNOSTI UDALJENIH SENZORA PRIMJENOM MQTT PROTOKOLA I VIZUALIZACIJA DOBIVENIH VRIJEDNOSTI PRIMJENOM WEB TEHNOLOGIJA

ZAVRŠNI RAD

Predmet: Osnove web programiranja

Mentor: Student:
Ivan Heđi, dipl. ing., v. pred. Hrvoje Đaković

VIROVITICA, 2021.

Veleučilište u Virovitici

Preddiplomski stručni studij Elektrotehnike - Smjer Telekomunikacije i informatika

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: **ĐAKOVIĆ HRVOJE** **JMBAG:** **0152209873**

Imenovani mentor: Ivan Heđi, dipl. ing., v. pred.

Imenovani komentor: -

Naslov rada:

Praćenje vrijednosti udaljenih senzora primjenom MQTT protokola i vizualizacija dobivenih vrijednosti primjenom web tehnologija

Puni tekst zadatka završnog rada:

Internet stvari je pojam koji opisuje tehnologije i načine povezivanja uređaja u jednu cjelinu koji prikupljaju i izmjenjuju podatke putem Internet mreže. Tako prikupljeni podaci mogu biti dostupni na bilo kojem uređaju koji je dio Internet mreže. Broj i različitost uređaja razvojem informacijsko – komunikacijskih tehnologija je sve veći i veći. Vaš zadatak je opisati koncept Internet stvari. Navesti i ukratko opisati uređaje, tehnologije prijenosa podataka te korištene protokole. Detaljnije se osvrnuti na MQTT protokol. Na praktičnom primjeru implementirati tehnologiju Internet stvari baziranu na MQTT protokolu u svrhu praćenja vrijednosti udaljenih senzora. Potrebno je izraditi programski modul koji će prikupljati podatke sa senzora i pohranjivati ih u bazu podataka. Podatke iz baze podataka vizualizirati koristeći web tehnologije kako bi se podacima moglo pristupiti bez ograničenja. Uz svako mjerenje u bazu podataka zapisati i vrijeme kako bi mogli detektirati da li je mjerenje relevantno.

Datum uručenja zadatka studentu/ici: 28.07.2021.
Rok za predaju gotovog rada: 06.09.2021.

Mentor:

Ivan Heđi, dipl. ing., v. pred.

PRAĆENJE VRIJEDNOSTI UDALJENIH SENZORA PRIMJENOM MQTT PROTOKOLA I VIZUALIZACIJA DOBIVENIH VRIJEDNOSTI PRIMJENOM WEB TEHNOLOGIJA

MONITORING VALUES OF REMOTE SENSORS USING MQTT PROTOCOL AND VISUALIZING THE OBTAINED VALUES WUSING WEB TEHNOLOGIES

SAŽETAK *– U ovom radu opisan je i implementiran „Message Queuing Telemetry Transport“ (skraćeno MQTT) protokol u svrhu praćenja temperature i vlažnosti zraka u prostoru. Cilj završnog rada je istaknuti mogućnosti i važnosti uključivanja „Internet stvari“ (eng. Internet of Things, skraćeno IoT) u svakidašnjem životu. IoT u današnje vrijeme koristi se u sve više sustava, a kako bi takvi sustavi bili pametni, moraju imati međusobnu komunikaciju koju nam omogućuje MQTT protokol. Jedan MQTT sustav sastoji se obično od posrednika (eng. broker) i klijenta (eng. client), na način da se klijent dijeli na pretplatnika (eng. subscriber) i objavljivača (eng. publisher). Pretplatnik se pretplati na određenu temu, objavljivač objavljuje nove vrijednosti senzora, filtrira ih po temama, te na koncu isporučuje pretplatniku samo informacije za onu temu na koju se isti pretplatio. Podaci prikupljeni sa senzora na ovakav način dostupni su za daljnju obradu, u ovom slučaju za pohranu u bazu podataka, te na taj način praćenje stanja temperature i vlažnosti zraka kroz period vremena.*

Za praktični dio završnog rada korišten je DHT11 senzor na platformi Raspberry Pi, koji je povezan na internet mrežu, te se putem mreže šalju paketi sa informacijama prema klijentima. Posrednička, objavljivačka i pretplatnička aplikacija napisane su u JavaScript programskom jeziku, koristeći Node.js, WebSocket i Firebase kako bi prikazali rezultate u realnom vremenu, ali isto tako ih i zapisali u bazu podataka zbog daljnje obrade rezultata mjerenja.

Ključne riječi*: internet stvari, MQTT protokol, web tehnologije, raspberry pi, baze podataka*

SADRŽAJ

1. UVOD 1
2. KORIŠTENE TEHNOLOGIJE 2
2.1. MQTT 2
2.1.1. Posrednik 2
2.1.2. Mosca 3
2.1.3. Klijent 3
2.2. HTML 4
2.3. CSS 5
2.4. JavaScript 6
2.4.1. Node.js 7
2.5. jQuery 8
2.6. Raspberry Pi 9
2.6.1. DHT11 senzor 10
2.7. Firebase 11
3. JEDNOSTAVNA PRIMJENA MQTT PROTOKOLA 13
3.1. Kreiranje Mosca posredničke aplikacije 13
3.2. Kreiranje objavljivačke aplikacije 14
3.3. Kreiranje pretplatničke aplikacije 15
4. PREGLED FUNKCIONALNOSTI APLIKACIJE 16
4.1. BACKEND DIO APLIKACIJE 16
4.1.1. Posrednik 16
4.1.2. Objavljivač 17
4.2. Frontend dio aplikacije 20
4.2.1. Tablični prikaz podataka učitanih iz Firebase baze podataka 21
4.2.2. „Live“ povezivanje sa MQTT posrednikom i pretplata na temu 23

5. ZAKLJUČAK 25
6. LITERATURA 26
7. POPIS ILUSTRACIJA 27

1. UVOD

Internet stvari (eng. Internet of Things, skraćeno IoT) je skup tehnologija koje su se prometnule u svakodnevni život posljednjih nekoliko godina prvenstveno jer je potrebna automatizacija uređaja, kuća i pogona. IoT se može koristiti na razne načine, uz razne protokole i razne tehnologije, a u ovom radu je opisan rad IoT uređaja putem MQTT protokola koristeći DHT11 senzor koji očitava vrijednosti temperature i vlažnosti zraka u prostoru, te Rasperry Pi kao platforma na kojoj se nalaze mjerni senzori i aplikacija za očitavanje vrijednosti senzora. IoT uređaji povezuju se preko interneta ili neke druge mreže, a omogućuju prikaz očitanih rezultata i upravljanje preko mreže.

Sve očitane vrijednosti senzora, točan datum i vrijeme očitavanja pohranit će se u Firebase bazu podataka.

U naslovu su spomenute i web tehnologije. Koristiti ćemo HTML, CSS, JavaScript i Node.js kako bi prikazali rezultate mjerenja pohranjene u bazi podataka, ali uz iste tehnologije ćemo prikazati u realnom vremenu povezivanje na posrednika, pretplatu na određenu temu, te prikaz vrijednosti senzora.

2. KORIŠTENE TEHNOLOGIJE

U nastavku rada, prije prezentacije same aplikacije će se detaljnije opisati protokol MQTT, način na koji rade senzori, spomenuti i opisati platforme i tehnologije koje su korištene pri izradi aplikacije. Tehnologije koje će se spominjati: Mosca, Node DHT senzor, HTML, CSS, JavaScript, Node.js.

2.1. MQTT

MQTT protokol (Message Queuing Telemetry Transport) su razvili Arlen Nipper (Eurotech) i Andy Stanford-Clark (IBM). Njihova prva verzija objavljena je 1999. godine i korištena je za praćenje vrijednosti naftovoda usred pustinje. Glavni cilj njihova razvijanja je bio osmisliti princip prijenosa podataka uz malu potrošnju električne energije sa efikasnom brzinom prijenosa podataka, te malom cijenom cjelokupnog sustava[1]. Protokol MQTT se sastoji od posrednika i klijenata. Posrednik je centralni dio sustava (zamislimo ga kao server) na koji se povezujemo i primamo podatke koje on prima od klijenata. Klijenti se dijeli na objavljivača i pretplatnika. Objavljivač je zapravo aplikacija koja učitava vrijednosti senzora, pohranjuje vrijednosti u određenu temu, te nakon pretplate klijenta na temu, poruke određene teme dolaze do pretplatnika. Nekakav standard je koristiti klijent-server model kod kojega je omogućena direktna komunikacija između servera i klijenta, dok se kod MQTT protokol radi na principu objavi-pretplati (eng. publish – subscribe) gdje objavljivač i pretplatnik nikada nisu u direktnom kontaktu, već se sve prenosi putem posrednika. Najpoznatiji MQTT posrednik je „Mosquitto“, ali zbog korištenja Node.js u ovom radu, odabran je „MOSCA“ posrednik kao komunikacijski protokol na dionici posrednik-objavljivač-pretplatnik.

2.1.1. Posrednik

Posrednik je srce MQTT objavi-pretplati protokola, predstavlja softver koji radi samostalno. Na jednog posrednika može biti povezana velika količina klijenata. Odgovoran je za prihvaćanje konekcije od strane objavljivača, prihvaćanje i filtriranje poruka od objavljivača, praćenje tko od klijenata je pretplaćen na koju temu, te prosljeđivanje poruka točno onim pretplatnicima koji su pretplaćeni na određenu temu. Sadrži i podatke o sesiji svih

[1] https://thenewstack.io/mqtt-protocol-iot/

klijenata sa aktivnim vezama, uključujući pretplate i nedostavljene poruke. Posrednik je centralna točka kroz koju prolaze sve informacije što se tiču komunikacije, zato je važno da je posrednik skalabilan, lak za nadgledanje, otporan na greške i lak za integraciju u backend sustave.

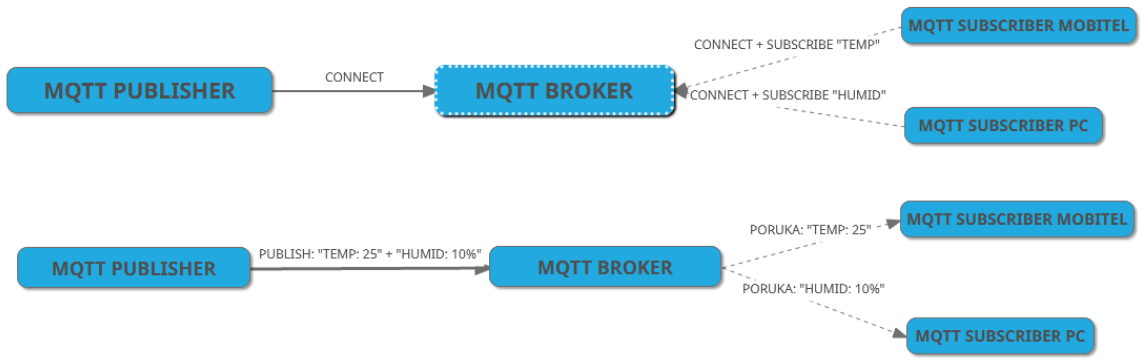
2.1.2. Mosca

„Mosca“ je Node.js MQTT posrednik koji može biti korišten samostalno ili ugrađen u Node.js aplikaciju. Za korištenje MQTT protokola može se koristiti javno dostupan posrednik, ali to bi značilo i izloženost svim podacima koji su u opticaju, često su ti serveri preopterećeni prometom, a s obzirom da je iznimno jednostavno kreirati svoj MQTT posrednik, za potrebe završnog rada kreirati ću svoj.

2.1.3. Klijent

MQTT klijent su krajnji uređaji protokola MQTT, mogu biti objavljivač i pretplatnik, a razlika je u njihovoj funkciji odnosno primaju li poruke ili ih odašilju. Objavljivač objavljuje informacije na njemu zadanu temu, dok pretplatnik odabire temu na koju se želi pretplatiti i koje poruke želi primati.

Slika 1. Struktura MQTT protokola



Izvor: Autor

Prethodna slika prikazuje jednostavan način rada MQTT protokola. Nakon podešavanja posrednika, važno je objavljivača spojiti na točnu IP adresu posrednika. Objavljivač u svom kodu ima točno određenu IP adresu na koju se spaja i točne teme, u ovom slučaju „TEMP“ i „HUMID“, na koje šalje poruke. Pretplatnik se spaja na istu IP adresu kao i objavljivač, to je

adresa posrednika, pretplaćuje se na određenu temu („TEMP“ ili „HUMID“) i nakon toga čeka do iduće objave objavljivača kako bi primio poruku.

2.2. HTML

HTML (HyperText Markup Language), prezentacijski jezik za izradu web stranica, stvara hipertekstualni dokument pomoću HTML jezika. „Hipertekstualni“ označava mogućnost povezivanja dokumenata hipervezama. Jednostavan je za uporabu i učenje, iz toga proizlazi njegova rasprostranjenost i popularnost. Osmišljen je kao besplatan i dostupan svima. Prikaz HTML dokumenta omogućuje web preglednik, a zapis HTML dokumenta možemo pisati u bilo kojem uređivaču teksta, primjer: Notepad, WordPad, Sublime, VisualStudio Code.. i tako dalje, a lako je prepoznati HTML dokument po ekstenziji datoteke „.html“ ili „.htm“.

Prva verzija HTML jezika objavljena je 1993. godine. Tada je pristup bio ograničen, pa tako nismo mogli dodavati niti fotografije. Danas se koristi HTML5 verzija, koja omogućuje prikazivanje i video zapisa, fotografija, „drag and drop“ manipulacija i još puno toga.

HTML dokument sadržava građevne blokove – HTML elemente. Svaki ima svoju oznaku ili tzv. tag, to su nazivi elemenata unutar znakova „<“ i „>“[2].

```
<div class="container">
    <p>Sadržaj stranice</p>
</div>
```

Prethodni izvorni HTML kod opisuje način korištenja oznaka. U ovome slučaju to je građevni blok „div“ sa nazivom klase „container“, te „p“ što označava naziv elementa (paragraf) u kojemu je zapisano „Sadržaj stranice“. Svaka oznaka se završava sa istim nazivom elementa kao i što se započinje, jedina razlika je što prije same oznake dodajemo „/“ kako bi označili kraj oznake.

[2] https://www.w3schools.in/html-tutorial/

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplikacija</title>
</head>
<body>
    <div class="container">
        <p>Tekst na stranici unutar klase "container"</p>
    </div>
</body>
</html>
```

U prethodno navedenom izvornom kodu, u prvoj liniji koda označavamo tip datoteke HTML, browser će sam odrediti o kojoj se točno verziji radi, odnosno u ovom slučaju o verziji HTML5. Svaki HTML dokument se sastoji od zaglavlja i tijela stranice. Zaglavlje stranice „<head>“ sadrži meta podatke koji korisniku nisu vidljivi, a tijelo stranice „<body>“ je sam sadržaj stranice.

2.3. CSS

CSS (**C**ascading **S**tyle **S**heets) je jezik stilova. Opisni jezik koji se oslanja na HTML. Style sheet je datoteka koja definira izgled, stil i interaktivnost web stranice. Cascading označava kaskadnu primjenu, odnosno možemo upotrijebiti određen kod na cijelom dokumentu ili na samo jednom segmentu.

U samim počecima weba, CSS je korišten u sklopu HTML dokumenta, na način da se svakom elementu dodjeli stil ili cijelom dokumentu koristeći oznaku „<style>“[3]. Kako je tehnologija napredovala, stvorio se problem miješanja sadržaja i kvarenja strukture dokumenta (HTMLa) koji služi samo za prezentaciju sadržaja. Danas, glavna ideja je odvojiti CSS od HTML dokumenta, to radimo na način da u HTML dokumentu u zaglavlju HTML dokumenta uključimo CSS koristeći oznaku „<link>“ i povežemo sa CSS datotekom koju prepoznajemo po ekstenziji „.css“.

[3] https://www.w3schools.com/css/

```css
.container {
    background-color: #FF0000;
}

p {
    font-weight: bold;
    color: white;
}
```

Prethodni CSS kod prikazuje primjer koristeći klasu „container“ i element „p“. Klasi „container“ dodjeljujemo boju pozadine u hex vrijednosti #FF0000 (crvena), a paragrafu dodjeljujemo podebljani font i bijelu boju.

2.4. JavaScript

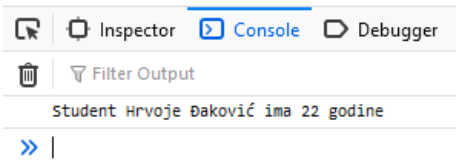
JavaScript kao skriptni programski jezik koristi se za kreaciju interaktivnih web stranica, npr. interakcija s korisnikom, promjena sadržaja u pregledniku klijenta i slično. JavaScript kod izvršava se na klijentskoj strani u web pregledniku. Podržan je od strane većine preglednika, stoga nije potrebna dodatna instalacija kako bi ga koristili. Za razliku od nekih drugih programskih jezika, JavaScript ne kompajlira svoj kod, nego ga izvršava „u hodu“ tako što ga pomoću interpretera prevodi u strojni jezik svaki puta kada se pokrene skripta[4].

JavaScript datoteka se označava ekstenzijom „.js“, te ju isto kao i CSS dodajemo ali na dno „<body>“ elementa HTML datoteke koristeći oznaku „<script src=“x.js“>“. Drugi način dodavanja JavaScript koda u HTML dokument je da jednostavno napišemo skriptu unutar „<script> kod </script>“ oznaka.

```javascript
let mojeIme = "Hrvoje"
let mojePrezime = "Đaković"
let mojeGodine = 22

console.log(`Student ${mojeIme} ${mojePrezime} ima ${mojeGodine} godine`)
```

Slika 2. Ispis JavaScript koda



Izvor: Autor

[4] https://www.w3schools.com/js/default.asp

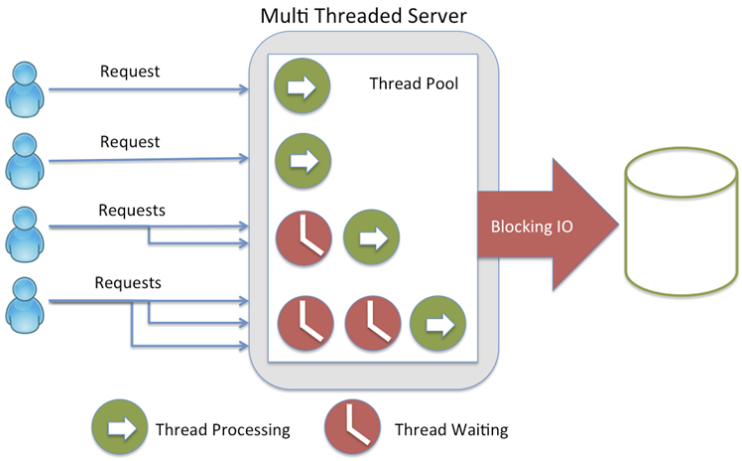
U prijašnjem izvornom JavaScript kodu i na slici br. 6 prikazan je primjer JavaScript koda sa ispisom u konzoli web preglednika. Zadali smo tri varijable sa imenom, prezimenom i godinama studenta, te smo funkcijom „console.log“ ispisali sve tri varijable koristeći *template strings* metodu ispisa.

2.4.1. Node.js

Node.js je besplatan alat za izradu brzih *real-time* aplikacija, napravljen je pomoću skriptnog programskog jezika JavaScript s ciljem skriptiranja na strani poslužitelja. JavaScript se koristi na svakom klijentskom pregledniku zasebno, dok Node.js kod se nalazi na serveru i korisnik učitava sadržaj sa servera. Omogućava korištenje jedinstvenog jezika između front-end i back-end-a, to znači da kompletnu aplikaciju možemo realizirati korištenjem JavaScript skriptnog programskog jezika.

Tradicionalno programiranje se obavlja sinkrono: linija koda se izvršava, sistem čeka rezultat, rezultat se procesuira i zatim se program nastavlja. Ukoliko učitavamo podatke sa baze podataka i imamo puno podataka za učitati, taj proces zahtjeva dugo čekanje.

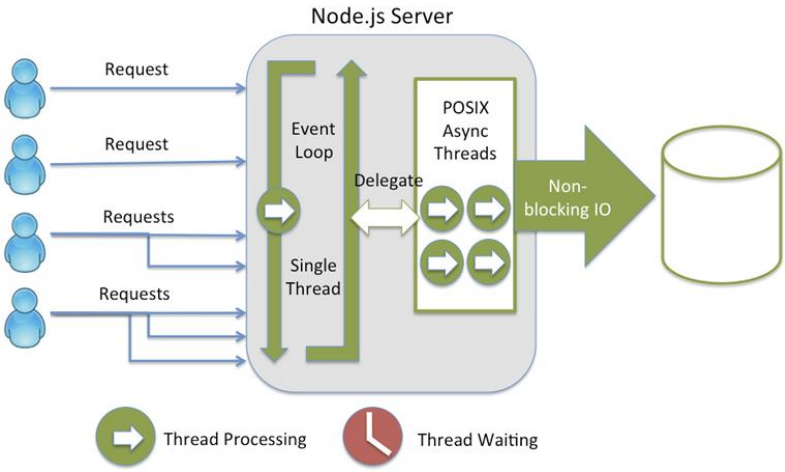
Slika 3 . Sinkrono učitavanje sa servera



Izvor: Šta je Node.js? https://www.popwebdesign.net/popart_blog/2015/06/sta-je-node-js/ (21.08.2021.)

JavaScript ima drugačiji pristup ovom problemu, program ne čeka na izvršenje operacije kao što bi bilo učitavanje iz baze podataka, nego nastavlja sa sljedećom linijom koda. Nakon što su podaci iz baze podataka učitani, pokreće se *callback* funkcija i rezultat se obrađuje. Node.js nudi jednostavan, brz način izrade modernih web aplikacija, koristeći ne blokirajući način rada gdje više zahtjeva može obraditi u isto vrijeme.

Slika 4 . Asinkrono učitavanje sa servera



Izvor: Šta je Node.js? https://www.popwebdesign.net/popart_blog/2015/06/sta-je-node-js/ (21.08.2021.)

Node.js dokazan je kao savršena biblioteka za korištenje u aplikacijama koje koriste ulaz/izlaz, prijenos podataka, real-time aplikacije, aplikacije zasnovane na JSON aplikacijskom programskom sučelju, web aplikacije koje koriste jednu stranicu…

2.5. jQuery

Razvojno okruženje jQuery, izvorno djelo Johna Resiga i tima programera koje predvodi Dave Methvin, daleko je najpopularnija JavaSscript biblioteka na klijentskoj strani. Lako ju je koristiti i zauzima malo resursa na računalu. Osnovne namjene jQuerya su DOM pristup i manipulacija, olakšavanje i ubrzavanje programiranja, proširivanje funkcionalnosti pomoću gotovih komponenti. jQuery nudi i odličnu mogućnost kreiranja vlastitih dodataka (eng. plugin). Bez njega bi određene stvari bilo teško ili gotovo nemoguće izvesti. Prepoznatljiv je u svijetu, tako da danas gotovo 65% od 10 tisuća najviše posjećenih stranica koristi jQuery. Prednosti alata su brzina i mogućnost razvoja boljih web usluga i aplikacija, što ga čini idealnim za početnike i napredne programere.

Oznaka jQuery funkcija je znak dolara ($), a najčešća sintaksa koju možemo vidjeti je[5]:

```
$(document.ready(() => {
  ...
}))
```

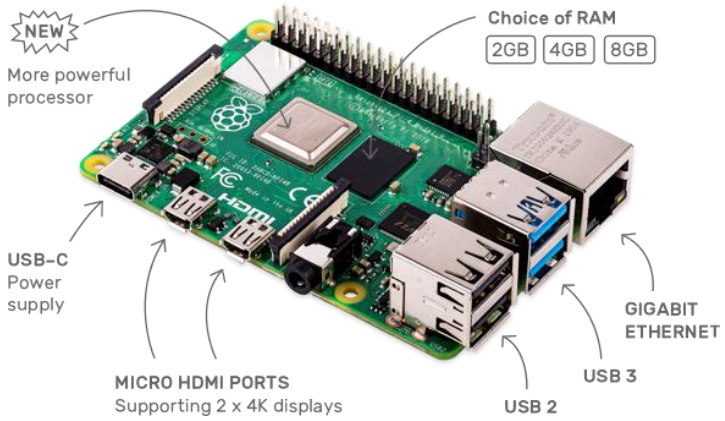
[5] https://www.w3schools.com/jquery/default.asp

Taj dio koda govori browseru da cijeli JavaScript unutar te funkcije može biti izvršen samo nakon što je stranica cijela učitana, bez čekanja na učitavanje slika, videa i sličnog.

2.6. Raspberry Pi

Raspberry Pi je jeftino računalo veličine kreditne kartice koje je u potpunosti smješteno na jednoj matičnoj ploči, a spaja se na monitor, moguće je spojiti razne USB uređaje poput tipkovnice, miša, prijenosnih medija. Sve to stvara jedan moćan uređaj koji koriste i mladi i stari, a najveća svrha u uređaju je učenje programiranja. U mogućnosti je raditi sve što i osobno računalo radi, poput pregledavanja interneta, igranja igrica, korištenja u uredske svrhe, mogućnosti su neograničene. Još fascinantnije je kada shvatite da to nije jedina namjena, možemo dodati hrpu senzora na Raspberry Pi, za razne digitalne projekte, poput MIDI klavijatura, vremenskih stanica, nadzornih kamera, automatsko zalijevanje biljaka, moguća je čak i potpuna automatizacija kuća.

Slika 5. Raspberry Pi 4 Model B - prikaz pločice



Izvor: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ (23.08.2021)

Raspberry Pi proizveden je u Ujedinjenom Kraljevstvu od strane Raspberry Pi zaklade, prvotno u svrhu promocije računalne znanosti u školama, ali je postalo puno više od toga. Zbog svoje niske cijene, a velikih mogućnosti, Raspberry Pi je najprodavanije Britansko računalo. Do svibnja 2021. prodano je više od 40 milijuna uređaja[6].

Uređaj ne dolazi sa pred instaliranim operativnim sustavom, instalira se na SD karticu, ali prije toga je potrebno sa Raspberry Pi stranica skinuti operativni sustav i potom instalirati na SD karticu.

[6] https://linuxhint.com/raspberry-pi-history/

Raspbian možemo smatrati glavnim operativnim sustavom. Dolazi od riječi Raspberry i Debian (linux optimiziran za raspberry platformu). Raspbian omogućuje preuzimanje više tisuća softverskih paketa, optimiziran je za procesor ARM11 koji koristi uređaj i lak je za integraciju novih stvari.

Slika 6 . Logo Raspbiana, najraširenijeg OS-a za Raspberry Pi

Izvor: https://www.ibtimes.co.uk/raspbian-os-raspberry-pi-debian-linux-eban-364401 (23.08.2021)

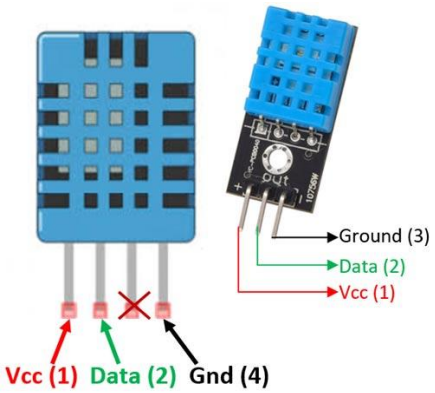
2.6.1. DHT11 senzor

DHT11 senzor je najjednostavniji senzor temperature i vlage kojega možemo pronaći. Spaja se na Raspberry vrlo jednostavno putem tri pina, te nam vraća dvije varijable, temperaturu i vlagu. To je uobičajeno jedan od prvih senzora sa kojima se početnici susreću.

Unutar samog modula nalaze se dvije vrste senzora, senzor temperature i senzor vlage. Senzor temperature je mali termistor zalemljen na pločicu. Senzor vlage je mala tiskana pločica nadolemljena na osnovnu. Ima otvorene bakrene vodove koji su međusobno vrlo blizu. Što je više vlage u zraku, više vlage dolazi i na same vodove te je otpor među njima manji.

Postoje dvije izvedbe, tri pinske i četvero pinske, razlike u konačnici nema jer se u četvero pinskoj izvedbi treći pin sa lijeva na desno ne koristi.

Slika 7. DHT11 - senzor temperature i vlage u zraku



Izvor: https://components101.com/sensors/dht11-temperature-sensor (21.08.2021)

Zbog same cijene izvedbe senzora, mjerne vrijednosti temperature variraju od 0 °C do 50 °C, a raspon mjerenja vlage od 20% do 90%.

Za spajanje na Raspberry Pi potrebne su nam 3 žice. VCC pin spajamo na plus izvor napajanja, GND pin spajamo na uzemljenje, DATA pin spajamo na GPIO pin 17.

Nakon spajanja potrebno je instalirati NPM paket kako bi mogli pomoću JavaScript skriptnog programskog jezika učitavati vrijednosti senzora u terminalu, to radimo u terminalu pomoću komande „npm install node-dht-sensor“.

Za ispis vrijednosti temperature i vlažnosti zraka možemo koristiti sljedeći kod:

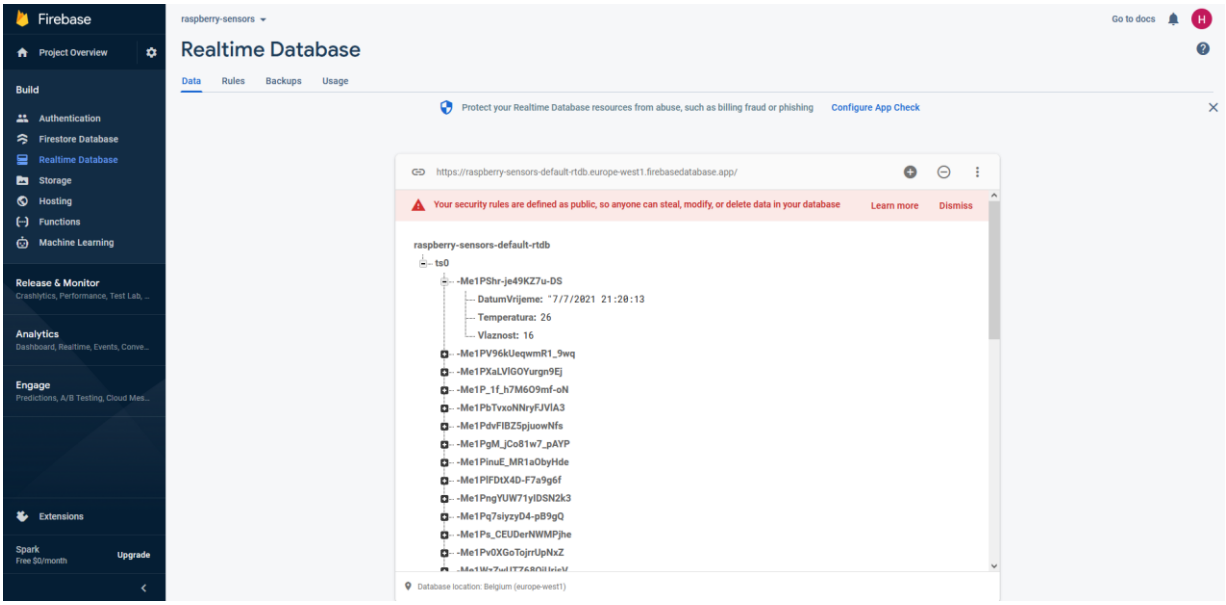
```
var sensor = require("node-dht-sensor");
sensor.read(11, 4, function(err, temperature, humidity) {
  if (!err) {
    console.log(`temp: ${temperature}°C, humidity: ${humidity}%`);
  }});
```

2.7. Firebase

Firebase je platforma razvijena od strane kompanije Google za kreiranje mobilnih i web aplikacija. Originalna kompanija osnovana je 2011. od strane Jamesa Tamplin-a i Andrew Lee-a, a preuzeta je 2014. od strane kompanije Google. Prvi produkt Firebase-a bila je *Firebase Realtime Database* (Firebase baza podataka u stvarnom vremenu), to je aplikacijsko programsko sučelje koje je sinkronizirano na iOS, Android i Web uređajima, te sve podatke sprema na Firebase oblak.

U ovoj aplikaciji se koristi funkcija Realtime Database u svrhu spremanja podataka izmjerenih na DHT11 senzoru – temperature i vlažnosti zraka.

Slika 8 . Firebase Realtime Database konzola



Izvor: Autor

3. JEDNOSTAVNA PRIMJENA MQTT PROTOKOLA

U ovom poglavlju bit će opisana jednostavna primjena MQTT protokola tako što ćemo kreirati posredničku aplikaciju, objavljivačku aplikaciju i pretplatničku aplikaciju. Aplikacije će raditi na lokalnoj računalnoj mreži.

Prvi korak je kreiranje direktorija Vašeg projekta, nazovimo ga „MQTT“

3.1. Kreiranje Mosca posredničke aplikacije

Kreiranjem Mosca posredničke aplikacije kreiramo „de facto“ server na koji ćemo spojiti objavljivački i pretplatnički dio aplikacije, odnosno senzore koji će objavljivati vrijednosti, te klijente koji će potraživati poruke o vrijednostima senzora.

Postupak:

1) Pokrenite terminal i pozicionirajte se unutar projektnog direktorija „MQTT“
2) Prvo moramo instalirati „Mosca“ modul. U terminalu pokrenite komande: „npm init“ i „npm install mosca --save“

   To će rezultirati u kreiranju „node_modules“ foldera i „package.json“ datoteke unutar projektnog direktorija, te instaliranje „Mosca“ modula.
3) Kreirajte novu datoteku „broker.js“ te zalijepite kod:

```
var mosca = require('mosca');
console.log("MQTT broker je spreman za otici online...");
var settings = {
    http: {
      port: 1883,
      bundle: true,
      static: './'
    }
  }
console.log("Setting port: " + settings.http.port + "...");
var server = new mosca.Server(settings);
server.on('ready', function () {
    console.log("MQTT broker je online!");
});
```

4) Ponovno pokrenite terminal, pozicionirajte se unutar projektnog direktorija, te komandom „node broker.js“ pokrenite posredničku aplikaciju. Ukoliko je uspješno, u komandnoj konzoli izaći će poruka „MQTT broker je online!“. Spajanje se vrši povezivanjem na lokalnu adresu + default port 1883. Posrednik kao takav je beskoristan bez klijenata, u idućem poglavlju ćemo proširiti naš MQTT sustav.

3.2. Kreiranje objavljivačke aplikacije

Kreiranjem objavljivačke aplikacije kreiramo objavljivača koji će trenutne vrijednosti senzora zapisivati u teme „TEMP“ i „HUMID“, te će poruke iz istih tema slati prema posredniku, a koji će te poruke prosljeđivati pretplaćenim klijentima.

Postupak :

1) Pozicionirati se u projektni direktorij, te kreirati „publisher.js“ datoteku
2) Zalijepiti sljedeći kod:

```
var mqtt = require('mqtt');
var client = mqtt.connect('mqtt://lokalna_adresa_vaseg_racunala');

var temperature = 25
var humidity = 10
client.on('connect', function () {
    console.log('Publisher je online!');
    setInterval(function () {
        if (!err) {
            client.publish('TEMP', 'Trenutna temperatura je: ' + temperature);
            client.publish('HUMID', 'Trenutna vlaznost zraka je: ' + humidity);
        } else {
            console.log('Desila se pogreska: ' + err);
        }
    }, 5000);
})
```

3) Pokrenimo aplikaciju komandom „node publisher.js“. Ukoliko je konekcija objavljivača na posrednika bila uspješna, prikazat će Vam se poruka „Publisher je online!“ i objavljivač će početi objavljivati poruke na dvije teme „TEMP“ i „HUMID“ u intervalu od 5 sekundi. Posrednik i objavljivač kao samostalne aplikacije nemaju određenu korist ukoliko se informacije ne dođu do pretplatnika.

3.3. Kreiranje pretplatničke aplikacije

Kreiranjem pretplatničke aplikacije kreiramo klijenta koji se pretplaćuje na dvije teme „TEMP“ i „HUMID“. Nakon što objavljivač objavi nove vrijednosti senzora, pošalje ih posredniku, posrednik će te iste poruke filtrirati po temama i proslijediti samo onim klijentima koji su pretplaćeni na određenu temu.

Postupak:

1) Potrebno je pozicionirati se u projektni direktorij, te kreirati datoteku „subscriber.js“
2) Zalijepiti sljedeći kod:

```
var mqtt = require('mqtt')
var client  = mqtt.connect('mqtt://lokalna_adresa_vaseg_racunala');
client.on('connect', function () {
    client.subscribe('TEMP')
    client.subscribe('HUMID')
});
client.on('message', function (topic, message) {
    context = message.toString();
    console.log(context)
});
```

3) Pokrenuti aplikaciju komandom „node subscriber.js“, ukoliko je konekcija uspješna, u idućih 5 sekundi pojaviti će nam se dvije poruke.

   „Trenutna temperatura je: 25“

   „Trenutna vlaznost zraka je: 10“

4. PREGLED FUNKCIONALNOSTI APLIKACIJE

U ovome dijelu završnog rada biti će opisana arhitektura aplikacije, te neke funkcije koje su se koristile unutar aplikacije. Web aplikacija za prikaz podataka očitanih senzorom DHT11 nam služi za prikaz temperature i vlage u prostoriji, te za daljnju obradu vrijednosti. Aplikacija je podijeljena u dva dijela – frontend sa tabličnim prikazom podataka spremljenih u bazi podataka i real-time spajanje na udaljeni Raspberry Pi uređaj i njegov DHT11 senzor, gdje možemo prikazati trenutne vrijednosti temperature i vlage u zraku, te backend sa MQTT posredničkim, objavljivačkim i pretplatničkim aplikacijama za dohvaćanje i objavljivanje vrijednosti DHT11 senzora.

4.1. BACKEND DIO APLIKACIJE

U procesu objavljivanja rezultata koristimo tri aplikacije – posrednik, objavljivač i pretplatnik. Već ranije spomenut je proces nastajanja sve tri aplikacije, ali sada će se prikazati točan primjer kako je izvedena ova aplikacija.

4.1.1. Posrednik

Posrednik u MQTT protokolu služi nam kao server, spajamo razne objavljivače na posrednika, te sa njegove IP adrese dohvaćamo teme koje želimo pratiti. Za potrebe ove aplikacije korišten je ranije spomenut dodatak MOSCA. Posrednik je izveden na lokalnoj IP adresi koristeći port 1884.

```
var mosca = require('mosca');
console.log("MQTT broker is ready to go online...");
var settings = {
    http: {
      port: 1884,
      bundle: true,
      static: './'
    }
  }
console.log("Setting port: " + settings.http.port + "...");
var server = new mosca.Server(settings);
server.on('ready', function () {
    console.log("MQTT broker is online!");
});
```

Slika 9 . Pokretanje MQTT Posrednika

```
PS C:\Users\Hrvoje\Documents\GitHub\zavrsni_rad\server-broker> node broker.js
MQTT broker is ready to go online...
Setting port: 1884...
MQTT broker is online!
```

Izvor: Autor

Slika br. 13 prikazuje princip pokretanja MQTT posrednika. Zadani port je 1884, te povratna potvrda o stanju MQTT posrednika, odnosno da je online.

4.1.2. Objavljivač

Objavljivački dio aplikacije sastoji se od tri dijela. Prvi dio služi za inicijalizaciju programa i dodatnih komponenti, drugi dio služi za povezivanje sa MQTT posrednikom, treći dio omogućuje pohranjivanje podataka u Firebase bazu podataka i slanje MQTT poruka.

U prvom dijelu inicijalizacije programa navodi se koji su nam dodatni moduli potrebni za rad, u ovoj aplikaciji potrebni su nam MQTT, Firebase/App, Firebase/Auth, Firebase/Database i NODE-DHT-SENSOR. Nakon same inicijalizacije potrebnih modula, potrebno je sa Firebase baze podataka povući „config“ podatke za spajanje na bazu podataka. Idući korak je određivanje putanje do podataka na bazi podataka, kao i pravljenje niza kako bi mogli pravilno pohraniti vrijednosti u bazu podataka.

```
var mqtt = require('mqtt');
var firebase = require('firebase/app');
require('firebase/auth');
require('firebase/database');
var client = mqtt.connect('mqtt://192.168.1.20');
var sensor = require('node-dht-sensor');

const firebaseConfig = {
    apiKey: "AIzaSyDyNdsLzJ14AaycXk0O0nuOgQH92Sog7u4",
    authDomain: "raspberry-sensors.firebaseapp.com",
    databaseURL: "https://raspberry-sensors-default-rtdb.europe-
west1.firebasedatabase.app",
    projectId: "raspberry-sensors",
    storageBucket: "raspberry-sensors.appspot.com",
    messagingSenderId: "465842400670",
    appId: "1:465842400670:web:85a4de277e5db767707a8f",
    measurementId: "G-P7RMGYRXE2"
};
firebase.initializeApp(firebaseConfig);

let oDb = firebase.database();
let aSenzor = [];
let oDbsenzor0 = oDb.ref('ts0');
oDbsenzor0.on('value', function (oOdgovorPosluzitelja) {
    aSenzor = [];
    oOdgovorPosluzitelja.forEach(function (oSenzorSnapshot) {
        let oSenzor = oSenzorSnapshot.val();
        aSenzor.push({
            DatumVrijeme: aSenzor.datetime,
            Temperatura: aSenzor.temperatura,
            Vlaznost: aSenzor.vlaznost
        })
    })
})
```

U drugom dijelu programa izvodimo pohranjivanje vrijednosti učitanih sa DHT11 senzora u bazu podataka. Interval za pohranu vrijednosti u bazu podataka iznosi 30 minuta (1800000ms). Podaci u bazu podataka pohranjuju se pod unikatnim ID-em koji dobijemo pomoću Firebase funkcije „.key“. U bazu podataka zapisujemo datum i vrijeme mjerenja, izmjerenu temperaturu i vlagu u prostoru. Funkcija „addZero“ se koristi za dodavanje nule ispred sata, minute i sekunde, ukoliko je broj jednoznamenkasti (npr. ukoliko je 13 sati i 4 minute, zapis je 13 sati i 04 minute).

```
setInterval(function () {
    sKey = firebase.database().ref().child('ts0').push().key;
    let currentdate = new Date();
    let datetime = currentdate.getDate() + "/" +
        (currentdate.getMonth() + 1) + "/" +
        currentdate.getFullYear() + " " +
        addZero(currentdate.getHours()) + ":" +
        addZero(currentdate.getMinutes()) + ":" +
        addZero(currentdate.getSeconds());
    sensor.read(11, 4, function (err, temperature, humidity) {
        if (!err) {
            client.publish('temp', 'Trenutna temperatura je: ' + temperature);
            client.publish('humid', 'Trenutna vlaznost zraka je: ' + humidity);
            oSenzor = {
                DatumVrijeme: datetime,
                Temperatura: temperature,
                Vlaznost: humidity
            };
            let oZapis = {};
            oZapis[sKey] = oSenzor;
            oDbsenzor0.update(oZapis);
            console.log(oZapis);
            console.log('Temperatura: ' + temperature + ' Vlaga: ' + humidity);
        } else {
            console.log('Desila se pogreska: ' + err);
        }
    });
    console.log('Message sent!');
}, 1800000);
function addZero(i) {
    if (i < 10) {
        i = "0" + i;
    }
    return i;
}
```

U trećem dijelu programa se izvodi konekcija na MQTT posrednika i objavljivanje rezultata po temama i slanje poruke prema posredniku. Interval slanja MQTT poruke je 5 sekundi (5000ms), tako svakih 5 sekundi možemo primiti trenutnu (eng. live) vrijednost temperature i vlažnosti zraka, pa u budućim nadogradnjama sistema možemo napraviti i alarm ukoliko se desi nekakva drastična promjena u vrijednostima.

```
client.on('connect', function () {
    console.log('Publisher is online!');
    setInterval(function () {
        sensor.read(11, 4, function (err, temperature, humidity) {

            if (!err) {
                client.publish('temp', 'Trenutna temperatura je: ' + temperature);
                client.publish('humid', 'Trenutna vlaznost zraka je: ' + humidity);

                console.log('Temperatura: ' + temperature + ' Vlaga: ' + humidity);
            } else {
                console.log('Desila se pogreska: ' + err);
            }
        })
    }, 5000);
})
```

Slika 10 . Pokretanje MQTT objavljivača

```
PS C:\Users\Hrvoje\Documents\GitHub\zavrsni_rad\mqtt> node publisher.js
Publisher is online!
Trenutna temperatura je: 20
Trenutna vlažnost je: 40
```

Izvor: Autor

Slika br. 14 prikazuje princip pokretanja MQTT objavljivačkog dijela aplikacije sa povratnom potvrdom o stanju aplikacije, odnosno poruka „Publisher is online!“.

4.2. Frontend dio aplikacije

U frontend dijelu aplikacije prikazati će se tablični prikaz podataka prikupljenih sa senzora, podaci se učitavaju iz Firebase baze podataka i prikazati će se mogućnost direktnog spajanja na MQTT posrednika i dohvaćanje vrijednosti tema za mjerenje temperature i vlažnosti zraka. Pretplatnički dio MQTT protokola je dio direktnog spajanja na MQTT posrednika u frontend dijelu aplikacije.

4.2.1. Tablični prikaz podataka učitanih iz Firebase baze podataka

Za potrebe tabličnog prikaza koristi se jQuery DataTables dodatak, zbog jednostavnosti i stilskog izgleda prikaza. Pozadina prikaza se odvija u JavaScriptu, tako što prvo moramo spojiti aplikaciju na Firebase bazu podataka koristeći „config“ podatke koje možemo pronaći u Firebase konzoli, nakon toga moramo učitati sve podatke pohranjene u bazi podataka u niz i potom funkcijom „UcitajPodatke“ prikazujemo sve podatke iz niza.

```
const firebaseConfig = {
apiKey: "AIzaSyDyNdsLzJ14AaycXk0O0nuOgQH92Sog7u4",
authDomain: "raspberry-sensors.firebaseapp.com",
databaseURL: "https://raspberry-sensors-default-rtdb.europe-west1.firebasedatabase.app",
projectId: "raspberry-sensors",
storageBucket: "raspberry-sensors.appspot.com",
messagingSenderId: "465842400670",
appId: "1:465842400670:web:85a4de277e5db767707a8f",
measurementId: "G-P7RMGYRXE2"
};
firebase.initializeApp(firebaseConfig);
let oDb = firebase.database();
let aSenzor = [];
let oDbsenzor0 = oDb.ref('ts0');
oDbsenzor0.on('value', function (oOdgovorPosluzitelja) {
aSenzor = [];
oOdgovorPosluzitelja.forEach(function (oSenzorSnapshot) {
let oSenzor = oSenzorSnapshot.val();
aSenzor.push({
datetime: oSenzor.DatumVrijeme,
temp: oSenzor.Temperatura,
vlaz: oSenzor.Vlaznost
})
})
UcitajPodatke();
})
function UcitajPodatke() {
aSenzor.forEach(function (oSenzor) {
$("#table_body").append("<tr><td>" + oSenzor.datetime + "</td><td>" + oSenzor.temp + " °C</
td><td>" + oSenzor.vlaz + " %</td></tr>");

})
$(document).ready(function () {
table = $('#TablicaSenzor0').DataTable();
})
};
```

Dio vidljiv korisnicima napisan je u HTML i CSS jezicima. Na početku dokumenta u „head“ dijelu važno je istaknuti korištenje jQuery DataTables dodatka, kao i CSS stilskog

dokumenta. Potom na jednostavan način koristeći gotovu HTML oznaku „table“ prikazujemo podatke. Na kraju HTML dokumenta važno je pozvati Firebase dodatke (app, auth, database), jQuery, jQuery DataTables, kao i samu JavaScript datoteku koja je zadužena za dohvaćanje podataka (app.js).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Raspberry Pi & MQTT & Websocket</title>
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.20/css/jqu
ery.dataTables.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <a href="websock.html" class="nav-link">Websocket Using Javascript MQTT Subscriber</a>
    <table id="TablicaSenzor0" class="display" style="width:100%">
        <thead>
            <tr>
                <th>Date</th>
                <th>Temperature</th>
                <th>Humidity</th>
            </tr>
        </thead>
        <tbody id="table_body"> </tbody>
    </table>
</body>
<script src="https://www.gstatic.com/firebasejs/8.2.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.2.0/firebase-analytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.2.0/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.2.0/firebase-database.js"></script>
<script src="jquery-3.2.1.min.js"></script>
<script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.20/js/jq
uery.dataTables.js"></script>
<script src="app.js"></script>
</html>
```

Slika 11 . Tablični prikaz podataka

Show 10 entries Search:

Date	Temperature	Humidity
7/7/2021 21:20:43	28 °C	17 %
7/7/2021 21:21:43	27 °C	15 %
7/7/2021 21:22:03	27 °C	16 %
7/7/2021 21:22:13	27 °C	15 %
7/7/2021 21:20:13	26 °C	16 %
7/7/2021 21:20:23	26 °C	16 %
7/7/2021 21:20:33	26 °C	16 %
7/7/2021 21:20:53	26 °C	16 %
7/7/2021 21:21:03	26 °C	16 %
7/7/2021 21:21:13	26 °C	16 %

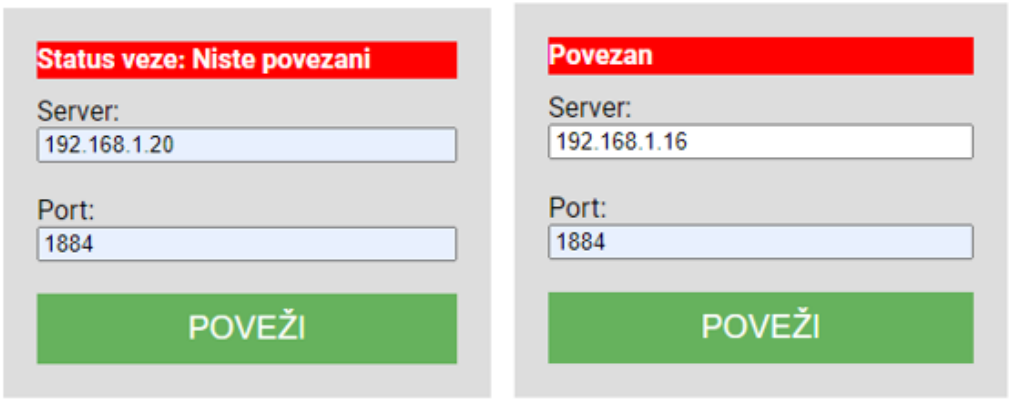
Showing 1 to 10 of 43 entries Previous 1 2 3 4 5 Next

Izvor: Autor

4.2.2. „Live“ povezivanje sa MQTT posrednikom i pretplata na temu

Drugi dio frontend aplikacije koristimo za povezivanje sa MQTT posrednikom, pretplaćivanjem na teme, dohvaćanje poruka sa pretplaćenih tema i slanje poruka prema objavljivaču na temu koju mi odredimo.

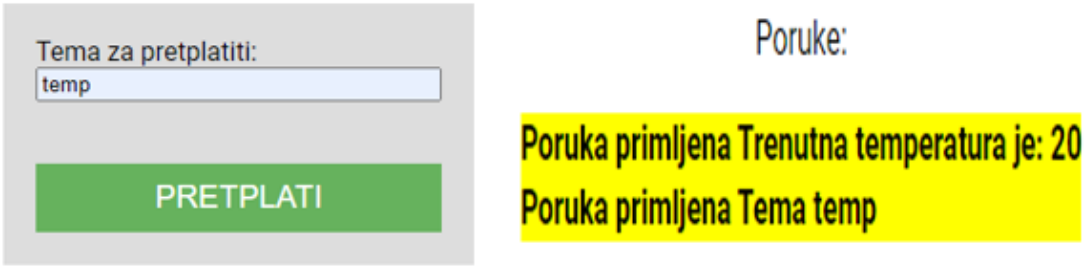
Slika 12 . Povezivanje na MQTT posrednika



Izvor: Autor

Slika br. 16 lijevo prikazuje status veze prilikom otvaranja stranice, unosa pogrešne IP adrese. Slika br. 16 desno prikazuje status veze „Povezan“ nakon unosa ispravne IP adrese i porta MQTT posrednika.

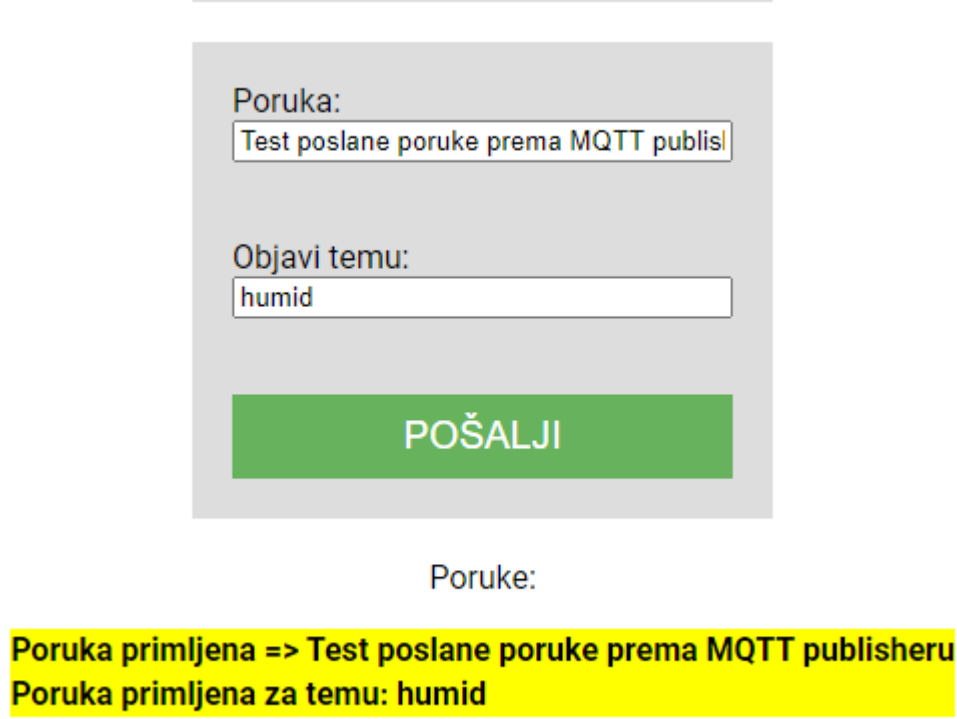
Slika 13 . Pretplata na temu i primitak poruke



Izvor: Autor

Slika br. 17 prikazuje pretplaćivanje na temu „temp“ i nakon određenog vremena primitak poruke od teme „temp“ sa tekstom „Trenutna temperatura je: 20“.

Slika 14 . Slanje poruke u drugu temu



Izvor: Autor

Slika br. 18 prikazuje mogućnost slanja poruke u temu po izboru. Tako možemo sami kreirati temu i samo određenim klijentima dopustiti pristup temi i njenom sadržaju.

5. ZAKLJUČAK

Kako bi aplikacija bila uspješno izrađena, važan je koncept rada aplikacija, pravilno učitavanje podataka i pravilna pohrana podataka za daljnju obradu. U izradi aplikacije mora se paziti na intervale očitavanja podataka kako ne bi došlo do preopterećenja samog Raspberry Pi uređaja i zagušenja mreže prilikom slanja podataka.

Aplikacija kao cjelina je spremna za uporabu, bilo u kućanstvu ili industriji, poslužit će svrsi za početak kako bi mogli odraditi „monitoring“ temperatura i vlage u zraku u pogonima, prostorijama i slično. Dakako ima mjesta za poboljšanja, kao što je dodavanje alarma ukoliko prilikom učitavanja temperature ili vlage u zraku se zabilježe veće razlike, što nam može dati signal kako je nešto na električnoj opremi u kvaru, klima uređaj u pogonu ne radi ili čak i požar. Alarm bi bio izvediv kao slanje email poruke prema osoblju ili poruka/poziv na mobitel.

Sami prikaz aplikacije (frontend) se isto tako može proširiti, pa tako možemo dodati statistički prikaz mjerenja, filtere, popis tema za pretplatu, popis posrednika i slično.

Prilikom izrade završnog rada i aplikacije, autor je naišao na probleme sa kojima se još nije susreo. Ovo je bio prvi susret sa MQTT protokolom i popratnim bibliotekama. Također aplikacija je poslužila za proširivanje znanja u području HTML, CSS, JavaScript, Node.js, Firebase tehnologija. Isto tako ponudila je širi pogled na cjelokupnu sliku IoT svijeta, pošto se u nekim situacijama moralo misliti „izvan kutije“ (eng. out of the box) kako bi se pronašlo rješenje. Znanje koje je autor stekao tijekom izrade rada podignulo je samopouzdanje i uvelike povećalo kompetencije za daljnje napredovanje i pridonijela boljoj kompetenciji prilikom traženja posla.

6. LITERATURA

Internetski izvori:

1. Naučite jQuery JavaScript library *http://www.kroativ.net/izdvojeno/naucite-jquery-javascript-library/ (21.08.2021)*
2. Šta je Node.js? *https://www.popwebdesign.net/popart_blog/2015/06/sta-je-node-js/ (20.08.2021)*
3. HTML: HyperText Markup Language *https://developer.mozilla.org/en-US/docs/Web/HTML (18.08.2021.)*
4. Mozilla CSS extensions *https://developer.mozilla.org/en-US/docs/Web/CSS/Mozilla_Extensions (18.08.2021.)*
5. JavaScript *https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.08.2021.)*
6. DHT11 senzor temperature i vlage *https://e-radionica.com/hr/dht11-senzor-temperature-i-vlage.html (21.08.2021.)*
7. Raspberry Pi *https://www.raspberrypi.org/ (21.08.2021.)*
8. *Raspberry Pi History https://linuxhint.com/raspberry-pi-history/ (25.08.2021.)*
9. *MQTT Version 5.0* https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html (25.08.2021.)
10. *Get to Know MQTT: The Messaging Protocol for the Internet of Things https://thenewstack.io/mqtt-protocol-iot/ (25.08.2021.)*

Knjige:

1. M. H. Asghar, A. Negi and N. Mohammadzadeh (2020): Principle application and vision in Internet of Things (IoT)
2. Joseph D. Booth (2018): W3.CSS Succinctly
3. Rick Hernandez (2016): The Beginners Guide To Node.js
4. Gaston C. Hillar (2017): MQTT Essentials – A Lightweight IoT Protocol
5. Baptiste Pesquet (2020): The JavaScript Way - A gentle introduction to an essential language
6. Damian Wielgosik (2016): How To Code In HTML5 and CSS3

7. **POPIS ILUSTRACIJA**

Slika 1. Struktura MQTT protokola 3
Slika 2. Ispis JavaScript koda 6
Slika 3 . Sinkrono učitavanje sa servera 7
Slika 4 . Asinkrono učitavanje sa servera 8
Slika 5. Raspberry Pi 4 Model B - prikaz pločice 9
Slika 6 . Logo Raspbiana, najraširenijeg OS-a za Raspberry Pi 10
Slika 7. DHT11 - senzor temperature i vlage u zraku 10
Slika 8 . Firebase Realtime Database konzola 12
Slika 9 . Pokretanje MQTT Posrednika 17
Slika 10 . Pokretanje MQTT objavljivača 20
Slika 11 . Tablični prikaz podataka 23
Slika 12 . Povezivanje na MQTT posrednika 23
Slika 13 . Pretplata na temu i primitak poruke 24
Slika 14 . Slanje poruke u drugu temu 24