PRIMJENA MQTT PROTOKOLA U INTERNETU STVARI

Šušnja, Marijana

Master's thesis / Specijalistički diplomski stručni

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:047974

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Elektrotehnike

MARIJANA ŠUŠNJA

Z A V R Š N I R A D

PRIMJENA MQTT PROTOKOLA U INTERNETU STVARI

Split, rujan 2020.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Elektrotehnike

Predmet: Senzorske mreže

Z A V R Š N I R A D

Kandidat: Marijana Šušnja

Naslov rada: Primjena MQTT protokola u Internetu stvari

Mentor: Tonko Kovačević

Split, rujan 2020.

SADRŽAJ

1. UVOD ... 3

2. SOFTVERSKE PLATFORME ... 5

2.1. Arduino IDE ... 5

2.1.1. Instalacija programa ... 5

2.1.2. Radni prostor Arduino IDE ... 5

2.2. Node.js ... 7

2.2.1. Node-RED ... 8

2.2.2. Node-RED instalacija ... 8

2.2.3. Node-RED radni prostor ... 10

2.3. ThingSpeak ... 13

2.3.1. Uporaba ThingSpeak ... 13

3. MQTT protokol ... 15

3.1. Klijent ... 15

3.2. Posrednik ... 16

3.3. MQTT paket ... 16

3.4. Fiksno zaglavlje ... 16

3.4.1. Tip MQTT kontrolnog paketa ... 17

3.4.2. Zastavice ... 18

3.4.3. Preostala duljina paketa ... 20

3.5. Varijabilno zaglavlje ... 21

3.5.1. CONNECT varijabilno zaglavlje ... 21

3.5.2. CONNACK varijabilno zaglavlje ... 22

3.5.3. PUBLISH varijabilno zaglavlje ... 23

3.5.4. Varijabilno zaglavlje ostalih tipova poruka ... 23

3.6. Korisni sadržaj ... 24

3.6.1. CONNECT korisni sadržaj ... 24

3.6.2. PUBLISH korisni sadržaj ... 24

3.6.3. SUBSCRIBE korisni sadržaj ... 25

3.6.4.SUBACK korisni sadržaj ... 25

3.6.5. UNSUBSCRIBE korisni sadržaj ... 25

3.7. Kvaliteta usluge QoS ... 26

3.7.1. Kvaliteta usluge razine 0 ... 26

3.7.2. Kvaliteta usluge razine 1 27
3.7.3. Kvaliteta usluge razine 2 28
3.8. MQTT veza (uspostava, održavanje, prekid) 29
3.9. MQTT teme 31
3.10. Usporedba HTTP i MQTT protokola 32
4. IMPLEMENTACIJA MQTT PROTOKOLA 34
4.1. DHT11 34
4.2. ESP32 35
4.3. Implementacija 36
4.3.1. Arduino IDE implementacija 37
4.3.2. Node-RED implementacija 41
4.3.3. ThingSpeak Implementacija 48
5. ZAKLJUČAK 52
LITERATURA 53
POPIS SLIKA 56
POPIS TABLICA 57
PRILOZI 58
Arduino IDE programski kod s objašnjenjem 58
Node-RED programski kod 64

Sažetak:

Primjena MQTT protokola u Internetu stvari

U ovom radu opisan jei implementiran MQTT protokol koji ima veliku upotrebu u IoT sustavima. Jedan MQTT sustav sastoji se od jednog posrednika i obično većeg broja klijenata. Posrednik je centar sustava. Sve poruke koje klijenti međusobno razmjenjuju imaju temu. Poruke prvo preuzima posrednik. Pregledava njihovu temu, filtrira ih i usmjerava prema klijentima pretplaćenim na tu temu.

U radu su objašnjeni tipovi paketa koji se razmjenjuju ovim protokolom. Opisano je koja polja posjeduju, koju vrijednost mogu imati ta polja, njihova veličina i svrha.

Za implementaciju MQTT protokola korišten je DHT11 senzor i ESP32 modul od elektroničkih elemenata, a za softversku implementaciju Arduino IDE, Node-RED i ThingSpeak.

Implementacija prikazuje izmjerenu vrijednost senzora u sva 3 softverska programa. Arduino IDE sa senzorom i pločom ne ostvaruje MQTT komunikaciju, ali Node-RED i ThingSpeak ostvaruju.

Ključne riječi:MQTT protokol, Arduino IDE, Node-RED, ThingSpeak, ESP32, DHT11, MQTT posrednik, MQTT klijent

Summary:

Application of MQTT protocol in the Internet of Things

In this graduation thesis, MQTT protocol which is widely used in IoT systems is described and implemented. MQTT system usually consists of one broker and larger number of clients. MQTT broker is the center of the system. Messages which are sent between the clients have their own topic. First to receive the messages is the MQTT broker. After the messages are received by the broker, messages are filtered and routed according to their topic and are sent to the clients which are subscribed to the corresponding topic.

This thesis covers the type of packets which are exchanged over the network by MQTT protocol. Different packet fields along with their values, sizes and purpose are described. DHT11 sensor and ESP32 electronical board are used for the hardware implementation of the MQTT protocol, while for the software implementation Arduino IDE, Node-RED

and ThingSpeak have been used.

Implementation is displaying the measured sensor values in all three software tools.

Arduino IDE is not communicating with the sensor and electronical board over the MQTT protocol, while the Node-RED and ThingSpeak are using MQTT for communication.

Key words: MQTT protocol, Arduino IDE, Node-RED, ThingSpeak, ESP32, DHT11, MQTT broker, MQTT client

1. UVOD

U posljednjih 30 godina izrazito se povećala potreba za automatizacijom uređaja. Stoga se razvojem starih uređaja i nastankom novih stvorila potreba za ugradnjom velikog broja senzora, mogućnošću daljinskog upravljanja uređajima kao i pregled njihovih vrijednosti putem mreže. Ti uređaji pripadaju IoT (engl. *Internet of Things*) uređajima. Povezuju se preko interneta ili neke druge mreže. Omogućuju prikaz dobivenih i/ili izračunatih parametara ili upravljanje preko mreže.

Postoji veliki spektar protokola za povezivanje uređaja ili senzora na mrežu. Oni se razlikuju prema sloju TCP/IP modela na kojem ostvaruju vezu. Protokol koji je potrebno koristiti u ovom radu je MQTT protokol koji pripada aplikacijskom sloju TCP/IP modela.

MQTT protokol je specifičan jer međusobnu vezu ne mogu ostvariti klijenti direktno. Sve poruke koje senzor ili uređaj šalju stižu prvo na broker. On je posrednik u ovom sustavu. Mosquitto je najpoznatiji MQTT posrednik, ali u ovom radu zbog korištenja Node-RED platforme odabran je MOSCA posrednik koji je ugrađen u program.

U svijetu postoje 4 IoT modela komunikacije. Poredani po složenosti to su:

- Uređaj s uređajem (engl. *Device-to-Device*)
- Uređaj s oblakom (engl. *Device-to-Cloud*)
- Uređaj s posrednikom (engl. *Device-to-Gateway*)
- Backend dijeljenje podataka (engl. *Backend Data Sharing*)

Prema dobivenom zadatku za implementaciju MQTT protokola potrebno je koristiti *Device-to-Gateway* model. Između uređaja i oblaka nalazi se posrednik (engl. *gateway*). Posrednik sustavu pruža sigurnost ali i druge funkcionalnosti kao što su prijevod podataka i protokola.

Za uspješno izvršavanje zadatka potrebno je napraviti sljedeće korake. ESP32 ploču žično povezati sa senzorom DHT11 koji mjeri dva parametra (temperaturu i vlažnost). Ploču povezati USB kabelom preko kojeg će se vršiti napajanje ploče i senzora, ali i ugradnja koda na ploču pomoću Arduino IDE programa. Kod će narediti ploči povezivanje na WiFi mrežu, uzimanje vrijednosti sa senzora u određenim vremenskim intervalima dodajući im teme i prosljeđivati ih WiFi mrežom do posrednika.

Posrednik se nalazi u Node-RED programu. Potrebno mu je postaviti IP adresu i TCP port i također navesti u Arduino IDE kodu. Node-RED ima vlastiti dashboard u kojem je omogućeno prikazivanje različitih widgeta. Oni omogućuju slikovito i numerički prikazivanje vrijednosti senzora. Node-RED se prikazuju samo u lokalnoj mreži. Za prikaz vrijednosti senzora na internetu koristit će se ThingSpeak aplikacija. Ona omogućuje slanje i preuzimanje vrijednosti IoT uređaja koristeći HTTP ili MQTT protokol. Radi u stvarnom vremenu, a implementacija ove aplikacije je jednostavna. Vrijednosti koje pristignu u Node-RED bit će objavljenje na linkovima koje pruža ThingSpeak.

2. SOFTVERSKE PLATFORME

Softverske platforme predstavljaju okruženje u kojem se izvršava dio softvera. Pružaju različite mogućnosti, ali imaju i ograničenja. Stoga će se za ovaj rad koristiti 3 platforme.

2.1. Arduino IDE

Arduino ima vlastito sklopovlje (engl. *Hardware*) i programsku podršku (engl. *Software*).

Postoje različite vrste sklopovlja po specifikacijama te su im pridodijeljena različite imena: Uno, Leonardo, Micro, Nano, Mini, Mega, Due, M0, Esplora, Yun Mini i dr. Svaki modul posjeduje programirani mikrokontroler i omogućuje prihvaćanje koda.

Arduino IDE predstavlja programsku podršku otvorenog koda dostupan na Mac, Windows i Linux operativnim sustavima. Programsku podršku moguće je koristi i na drugim sklopovljima koja nisu Arduino, a biraju se zbog boljih performansi ili niže cijene. Prilikom implementacije u ovom radu koristitiće se ESP32 ploča.

2.1.1. Instalacija programa

Arduino IDE je program otvorenog koda stoga je njegova instalacija besplatna. Može preuzeti s Arduino službene stranice https://www.arduino.cc/en/main/software. U dijelu na kojem se nalaze linkovi za preuzimanje, potrebno je prepoznati operacijski sustav i verziju koja se nalazi na računalu na kojem se želi pokrenuti preuzimanje i pokrenuti instalaciju.

2.1.2. Radni prostor Arduino IDE

Na slici 2.1 prikazan je Arduino IDE otvoreni prozor. Sastoji se od nekoliko dijelova koji su na slici označeni različitim bojama.

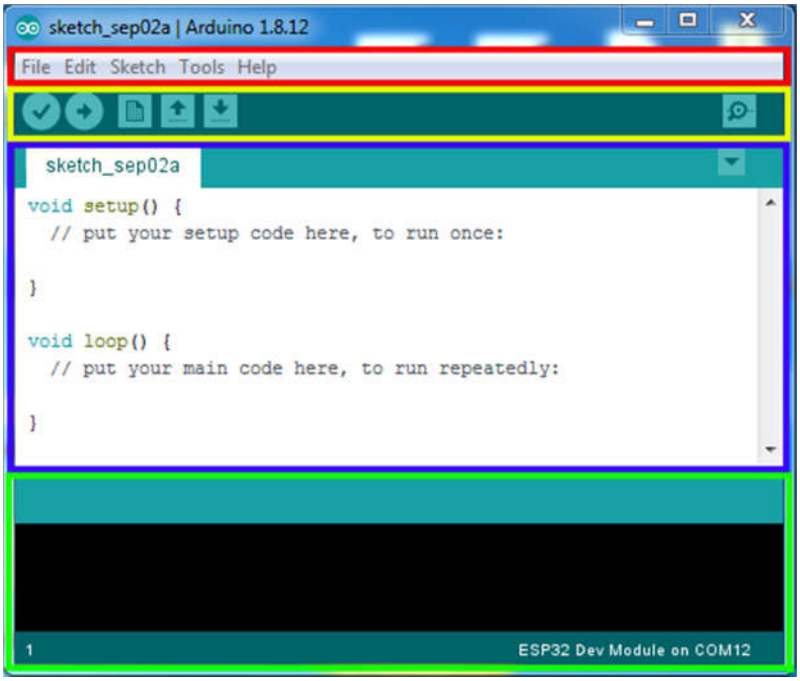
Crvenom bojom ocrtan je izbornik u kojem se mogu pronaći sve mogućnosti koje program pruža.

Žutom bojom označen je brzi izbornik sačinjen od prečica mogućnosti koje se najčešće koriste. S lijeva na desno to su:

- Verify – provjerava greške u kodu
- Upload – provjerava greške u kodu i učitava kod u ploču
- New – stvara novu datoteku za novu skicu (engl. *sketch*)
- Open – prikazuje sve primjere kodova koje pruža program i vlastite spremljene skice
- Save – sprema kod
- Serial Monitor – skočni prozor (engl. *pop-up window*), predstavlja terminal preko kojeg se šalju i primaju serijski podatci između ploče i Arduina IDE

Plavom bojom označen je tekstualni editor unutar kojeg se piše programski kod u C i C++ jeziku. Na slici je prikazano da se sastoji se od dvije funkcije (setup i loop). Unutar setup funkcije inicijaliziraju se i postavljaju početne vrijednosti. Ova funkcija se izvršava samo jednom, na početku pokretanja programa. Unutar loop funkcije upisuje se kod koji se neprekidno izvršava dok je modul uključen.

Zelenim okvirom ocrtana je konzola koja daje informacije o kodu. Ako je zeleni dio prozora nakon verifikacije ili uploada zelene boje kod je ispravan. Ako je narančaste boje nakon verifikacije, kod je neispravan, a ako je narančaste boje nakon uploada neispravan je kod ili veza s pločom. U donjem desnom kutu nalazi se naziv porta i ploče s kojim je program trenutno povezan.



Slika 2.1. Početni Arduino IDE prozor

2.2. Node.js

Node.js je platforma otvorenog koda. Omogućuje programiranje dinamičnih i brzih aplikacija. Dizajniran je za izgradnju skalabilnih mrežnih aplikacija. Po dizajnu sličan je Ruby's Event Machine i Python's Twisted sustavima.

Iz naziva je vidljivo da kao programski okvir (engl. *framework*) koristi JavaScript. Funkcionalnost programa čine moduli koji su napisani kao JavaScript datoteke. Node.js posjeduje veliki broj datoteka i omogućuje stvaranje vlastitih.

Node.js je sačinjen od terminala CLI (engl. *command-line interface*) i udaljenog repozitorija. Unutar terminala se upisuju naredbe za dohvaćanje JS spremljenih modula iz repozitorija.

NPM(engl.*Node Package Manager*) je aplikacija i repozitorij za razvoj i dijeljenje JavaScript modula. Veliki broj takvih modula napisan je za Node.js.

Node.js je „*cross platform*“ stoga je dopuštena instalacija i upotreba na svim operativnim sustavima kao što su GNU/Linux, Windows, Mac i dr. Može se besplatno preuzeti na ovlaštenoj stranici https://nodejs.org/en/. Potrebno je odabrati posljednju verziju

operativnog sustava koji se koristi na računalu, odabrati zadane vrijednosti i pokrenuti instalaciju. Ova instalacija uključuje NPM.

2.2.1. Node-RED

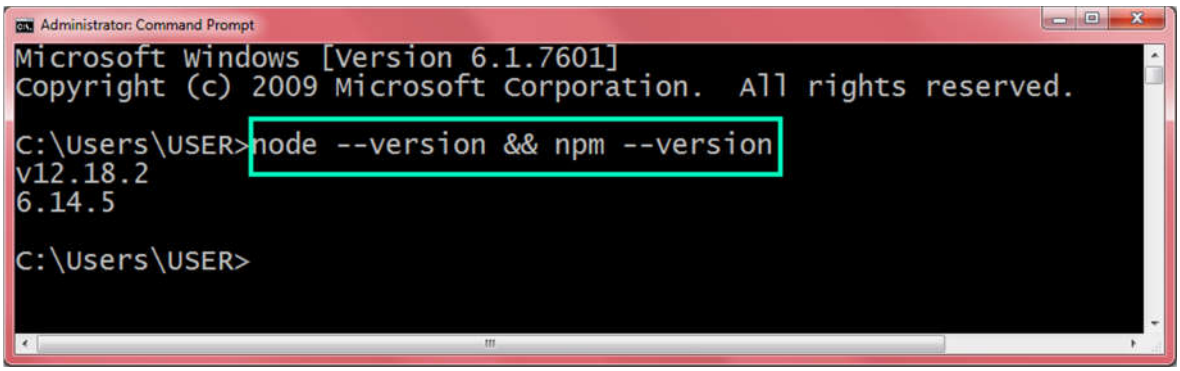
Node-RED je programski alat koji se koristi za vizualno programiranje. Dizajniran je da poveže hardverske uređaje sa sučeljem za programiranje aplikacija i internetom. Programiranje ostvaruje umetanjem čvorova u radni dio prozora i međusobnim povezivanjem nitima (vezama) kojima se ostvaruje povezanost i protok podataka. Unutar svakog čvora definirano je što čvor treba učiniti s pristiglim podatcima ili kada generirati nove zadane podatke.

2.2.2. Node-RED instalacija

Za instalaciju Node-RED potrebno je prvo preuzeti Node.js s prethodno navedenog linka. Verzija Node.js i NPM može se provjeriti unutar cmd (engl. *command prompt*) upisivanjem naredbe:

```
node --version && npm –version
```

Na slici 2.2 prikazan je cmd prozor s upisanom i ocrtanom naredbom te izlazom u kojem je prikazana verzija Node.js programa i NPM repozitorija.



Slika 2.2. Provjera Node.js i NPM verzije u cmd prozoru

Iduća naredba instalira Node-RED i također se unosi u cmd.

```
npm install -g --unsafe-perm node-red
```

Instalacija može potrajati par minuta, a nakon njenog završetka, otvaranje programa vrši se unosom naredbe:

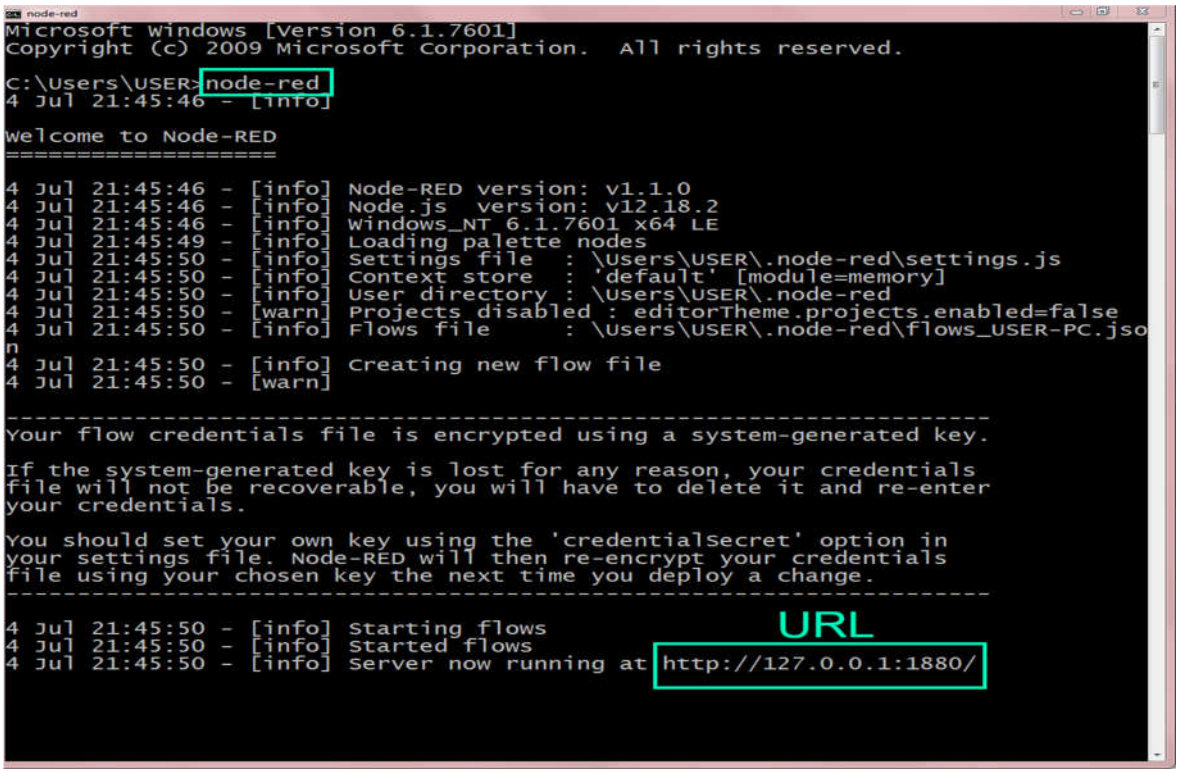
```
node-red
```

Ovu naredbu u cmd potrebno je unositi svaki put kada se želi pristupiti Node-RED programu. Cmd prozor u kojem se pokrene navedena naredba treba biti otvoren cijeli vremenski period u kojem koristi Node-RED.

Na slici prikazana je naredba za pokretanje Node-RED programa i izlazi:

- Verzija Node.js i Node-RED
- Lokacija datoteke s postavkama i korisničkim direktorijem
- Stvorena nova datoteka
- URL servera koji se unosi u pretraživač za pristup programu

Na slici 2.3 Posebno je istaknuta naredba i URL na slici.



Slika 2.3. Pokretanje Node-RED programa iz cmd

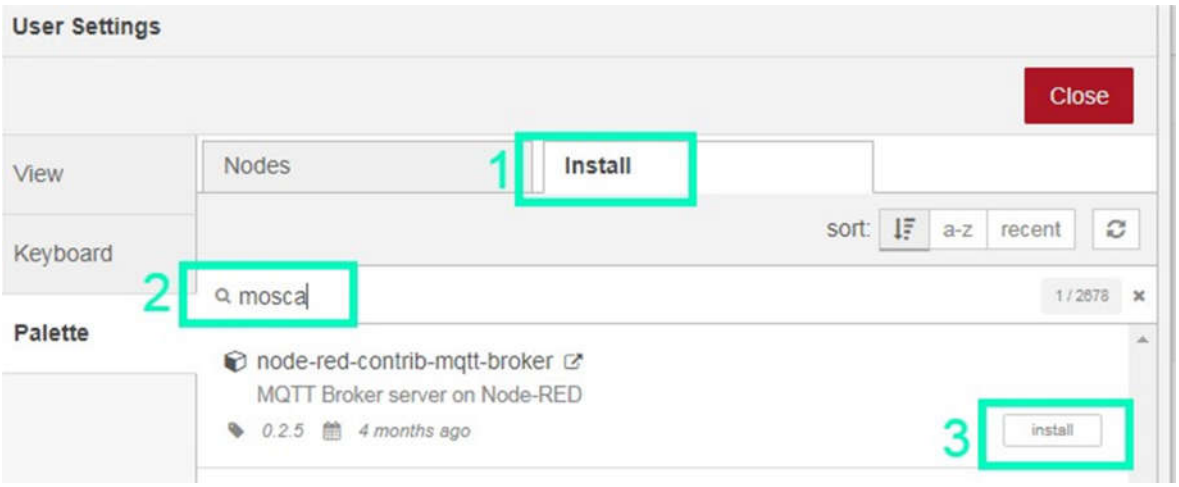
2.2.3. Node-RED radni prostor

Program se otvara upisivanjem URL-a iz cmd-a u pretraživač.

S lijeve strane otvorenog prozora nalazi se paleta koju čine čvorovi (engl. *nodes*), razvrstani u kategorije. Čvorovi su osnovni blokovi za izgradnju toka.

Nakon instalacije programa u paleti čvorova nalaze se samo osnovni, ako su potrebni dodatni čvorovi paletu je moguće proširiti. Klikom na izbornik u desnom gornjem kutu i odabirom upravljanja paletom (engl. *Manage palette*) otvara se pomoćni prozor. Unutar prozora potrebno je označiti palette i install. Novu skupinu čvorova moguće je pretraživati ručno ili upisivanjem u pretraživač čvorova. Kada se pronađe odgovarajući potrebno je samo kliknuti install i nakon nekoliko trenutaka novi čvorovi će biti instalirani i nalazit će se na lijevoj strani glavnog prozora zajedno s ostalim čvorovima.

Kao primjer instalacije čvorova na slici 2.4 prikazana je instalacija mosca čvora koji će se koristiti prilikom implementacije MQTT protokola. Ocrtan je redoslijed kojeg treba pratiti prilikom instalacije.



Slika 2.4. Preuzimanje čvorova za Node-RED platformu

Radni prostor Node-RED aplikacije nalazi se u sredini. Čvorovi koji će se koristiti pri programiranju mogu se postaviti u radni prostor na 3 načina:

- Povlačenjem čvorova iz palete u radni prostor

- Korištenjem dijaloga za brzo dodavanje (engl. *quick-add dialog*) koji se otvara unutar radnog prostora lijevim klikom dok je pritisnuta Ctrl ili Command tipka
- Dohvaćanjem s računala ili neke memorije povezane s računalom (engl. *clipboard*) ili biblioteke (engl. *library*)

Nakon pozicioniranja čvorova unutar radnog prostora potrebno je povezati čvorove prema logici i postaviti im potrebne parametre.

Čvorovi se međusobno povezuju na označenim priključcima (engl. *port*), držeći pritisnutu lijevu tipku miša od jednog do drugog priključka. Čvorovi se mogu povezati i korištenjem Ctrl ili Command tipke koja je pritisnuta cijelo vrijeme od odabira početka do kraja veze.

Svaki čvor uređuje se pojedinačno. Dvostrukim lijevim klikom na čvor otvara se pomoćni prozor unutar kojeg se nalaze 3 kartice:

- Svojstva – različita su za svaku vrstu čvora, postavke unutar ove kartice su iznimno važne za željeni tok podataka
- Opis – u ovoj kartici je moguće opisati ulogu čvora i bitne značajke za pravilan tok podataka, koristi se kao podsjetnik, a upisane informacije prikazuje u desnoj bočnoj traci kada su odabrane informacije, a čvor označen
- Izgled – ova kartica omogućuje urediti izgled čvora koji se prikazuje u programu, pa se određeni čvorovi izgledom mogu istaknuti veličinom od drugih ili drugom bojom

Osnovni čvorovi koje je neophodno upoznati za korištenje programa su:

- Inject
- Debug
- Function
- Change
- Switch
- Template

Inject čvor može se koristiti za ručno pokretanje toka klikom na gumb pored čvora, ali i za postavljanje vremenskog intervala u kojem će slati podatke i aktivirati tok. Inject poruci može se dodati tema i korisni sadržaj koji može biti različitih tipova podataka. Mogući tipovi podataka navedeni su u svojstvima čvora, potrebno je odabrati odgovarajući prema podacima koji se žele prenijeti.

Debug čvor se koristi za lakše uočavanje grešaka i za prikazivanje poruke u samom programu. Pristigle poruke prikazuju se u desnoj bočnoj traci koja pruža strukturirani prikaz primljene poruke. Uz poruku je naveden datum, vrijeme i koji debug čvor je primio poruku.

Function čvor omogućuje pokretanje JavaScript koda na porukama koje prolaze kroz čvor.

Change čvor omogućuje izmjenu svojstava poruke. U nekim situacijama može se koristiti umjesto Function čvora. Operacije koje se mogu primijeniti unutar ovog čvora su: postavi, promijeni, pomakni i izbriši neko od svojstva poruke. Unutar jednog change čvora može se postaviti veći broj operacija.

Switch čvor usmjerava poruke u određene grane toka prema postavljenim svojstvima.

Template čvor se koristi za generiranje teksta tako da se dijelovima primljene poruke ispunjava predložak. Ako se unutar predloška upiše npr. Trenutna temperatura u sobi je: {{payload}} °C. Payload će se zamijeniti s korisnim sadržajem pristigle poruke. Ako pristigla poruka ima vrijednost naprimjer 25, ispunjeni predložak će izgledati: Trenutna temperatura u sobi je: 25 °C.

Desni rubni stupac (engl. *sidebar*) sadrži korisne alate, to su: informacije, pomoć, debug, konfiguracijske čvorove i sadržaj spremljenih podataka.

Iznad stupca na desnoj strani nalaze se deploy i izbornik.

Deploy gumb se koristi kako bi se pokrenuo tok podataka nakon što su postavljene postavke čvorova i ostvarene veze. Nakon klika deploy gumba u vrhu prozora pojavit će se skočni prozor koji obavještava korisnika da li je uspješno pokrenut tok i da li ima čvorova koji su neispravno postavljeni.

Unutar izbornika nalaze se razne mogućnosti pa je moguće: uvoziti i izvoziti tokove, pretraživati tokove, dodavati i brisati tokove ili dijelove tokova, upravljati paletom, postaviti postavke za cijeli prozor, pogledati prečace i jednostavno pristupiti Node-RED web stranici.

2.3. ThingSpeak

ThingSpeak je IoT aplikacija i aplikacijsko programsko sučelje (APi) za pohranu, objavljivanje i razmjenu podataka uređaja koji koriste HTTP i MQTT protokol putem interneta ili lokalne mreže. Nudi mogućnost stvaranja komunikacijskih kanala za razmjenu podataka.

Za korištenje aplikacije potrebno je posjedovati Matlab korisnički račun ili ga napraviti na službenoj stranici https://thingspeak.com/. Prilikom stvaranja korisničkog profila može se odabrati standardna verzija koja se plaća ili kućna koja je besplatna i ima više ograničenja.

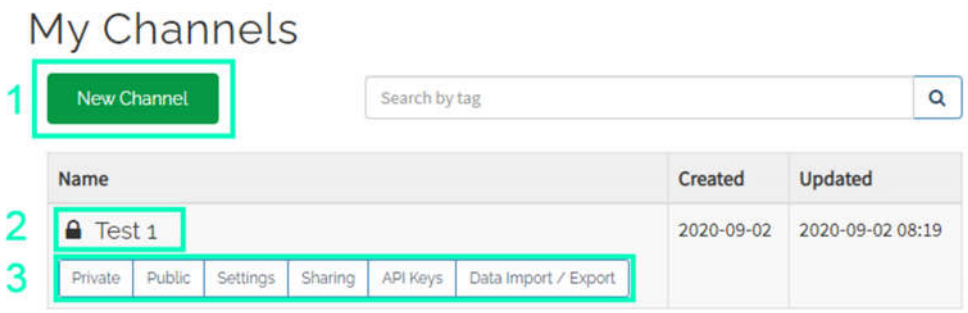
Besplatna verzija dozvoljava korištenje 4 kanala i 3 milijuna poruka godišnje odnosno 8200 poruka po danu. Između slanja poruka mora proći najmanje 15 sekundi.

Standardna verzija se koristi za komercijalnu upotrebu. Kapacitet se kupuje u jedinicama. Jedna jedinica dozvoljava uporabu 33 milijuna poruka godišnje odnosno 90 tisuća po danu i 250 kanala. Optimalna je za velike sustave.

2.3.1. Uporaba ThingSpeak

Unutar aplikacije moguće je pregledavati javne kanale i bez prijave, ali za izradu osobnih potrebna je prijava. Odabirom putanje Channels→My Channels→Channel otvorit će se predložak koji se sastoji od brojnih polja: ime, opis, mjesto, URL, video i oznake. Upisivanjem željenih vrijednosti u polja i spremanjem, stvorit će se kanal. On će dobiti vlastiti identifikator i moći će se dalje postavljati odabirom Channels→My Channels putanje. U prozoru će se pojaviti nazivi svih stvorenih kanala s karticama kao na slici 2.5. Označeni dijelovi s brojevima na slici 2.5 predstavljaju:

- 1 – stvaranje novog kanala
- 2 – naziv stvorenog kanala i lokot (zaključani označava privatni kanal a otključani javni)
- 3 – mogućnost uređivanja postavki postojećeg kanala



Slika 2.5. Prikaz stvorenog kanala u ThingSpeak platformi

Private kartica prikazuje informacije o kanalu koje su vidljive samo vlasniku računa, dok public prikazuje informacije o kanalu koje su vidljive drugima.

Unutar settings kartice moguće je promijeniti sve parametre kanala koji su postavljeni prilikom stvaranja kanala.

Unutar sharing kartice potrebno je odabrati hoće li kanal biti privatan (po zadanom), javan za one pretplatnike koji su se pretplatili na račun objavitelja ili javan za sve.

API Keys kartica prikazuje ključeve kanala. Koriste se za čitanje i upisivanje podataka na kanalu. Svaki kanal ima druge ključeve.

Data import / export kartica omogućuje uvoz i izvoz podataka kanala.

3. MQTT protokol

MQTT (engl. *Message Queuing Telemetry Transport*) je klijent/server protokol za prijenos poruka. Ostvaruje komunikaciju između uređaja (M2M – engl. Machine to machine) i jedan je od najčešće korištenih protokola za internet stvari (IoT). Omogućuje dvosmjernu komunikaciju. Podržava OASIS i ISO standarde. Kao što je prikazano u tablici 3.1. radi na aplikacijskom sloju ISO/OSI modela. Na transportnom sloju se mora nalaziti TCP protokol, a na mrežnom sloju IP protokol.

Tablica 3.1. ISO/OSI slojevi MQTT protokola

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

Koristi se obično za senzore odnosno IoT uređaje s ograničenim resursima. MQTT protokolom ostvaruje se veza između klijenata i posrednika (MQTT brokera). Klijenti se međusobno ne povezuju nikada direktno i nemaju podatke o IP adresama uređaja koji dohvaćaju poruke.

3.1. Klijent

Klijenti su krajnji uređaji MQTT protokola. Mogu biti objavljivači i pretplatnici. Razlikuju se po tome da li objavljuju poruke (objavljivači) ili se pretplaćuju na poruke (pretplatnici). Klijenti mogu biti razni uređaji od mikrokontrolera do servera. Neki uređaji istovremeno mogu biti objavljivači i pretplatnici. Objavljivači objavljuju informacije na određenu temu, a pretplatnici odabiru na koju temu će biti pretplaćeni.

3.2. Posrednik

Posrednik (engl. *broker*) se nalazi u središtu svakog objavi/pretplati protokola. Čini ga softver koji radi samostalno. Na jednog posrednika može biti povezan veliki broj klijenata, a na najprostijim je moguće ostvariti najmanje 1000 aktivnih veza klijenata i posrednika. Odgovoran je za prihvaćanje svih poruka generiranih od objavljivača. Prihvaćene poruke filtrira i prosljeđuje onim pretplatnicima koji su pretplaćeni na temu jednakoj temi poruke. Sadrži informacije o svim sesijama s klijentima. Obavlja funkcije autentifikacije i autorizacije klijenata.

3.3. MQTT paket

MQTT paketi se prenose između klijenata i MQTT posrednika u oba smjera i varijabilne su duljine. Paket se sastoji od 3 dijela koji su unutar paketa poredani po istom redoslijedu, a to su:

- Fiksno zaglavlje
- Varijabilno zaglavlje
- Korisni sadržaj (engl. *Payload*)

3.4. Fiksno zaglavlje

Svaki MQTT paket sastoji se od obaveznog fiksnog zaglavlja. Ovo zaglavlje ima veličinu 2-5 bajta i sačinjeno je od 3 polja. Polja su fiksne duljine osim polja koje se odnosi na preostalu duljinu paketa. U tablici 3.2 prikazan je najmanji oblik fiksnog zaglavlja.

Tablica 3.2. Fiksno zaglavlje MQTT poruke

Bitovi od MSB do LSB	7	6	5	4	3	2	1	0
Prvi bajt	Tip MQTT kontrolnog paketa				Zastavice specifične za svaki tip MQTT kontrolnog paketa			
Drugi bajt	Preostala duljina paketa							

3.4.1. Tip MQTT kontrolnog paketa

Tip MQTT kontrolnog paketa je polje veličine 4 bita. Definira tip paketa u kojem se nalazi. Najvažniji je dio MQTT paketa jer identificira cijeli paket. Raspored i značenje vrijednosti ovog polja navedene su u tablici 3.3.

Tablica 3.3. Vrijednosti koje tip kontrolnog paketa može imati s objašnjenjem

Vrijednost		Naziv	Smjer kretanja	Objašnjenje
Binarna	Decimalna			
0000	0	Rezervirano	Zabranjeno	Rezervirano
0001	1	CONNECT	Klijent → Server	Klijent zahtjeva spajanje na server
0010	2	CONNACK	Server → Klijent	Potvrda spajanja
0011	3	PUBLISH	Klijent ↔ Server	Objava poruke
0100	4	PUBACK	Klijent ↔ Server	Potvrda poruke
0101	5	PUBREC	Klijent ↔ Server	Objavljivanje primljeno (osigurana isporuka dio 1)
0110	6	PUBREL	Klijent ↔ Server	Puštanje objave (osigurana isporuka dio 2)
0111	7	PUBCOMP	Klijent ↔ Server	Objava je kompletna (osigurana isporuka dio 3)

1000	8	SUBSCRIBE	Klijent → Server	Klijent zahtjeva pretplatu na server
1001	9	SUBACK	Server → Klijent	Obavijest da je server potvrdio pretplatu
1010	10	UNSUBSCRIBE	Klijent → Server	Klijent zahtjeva otkazivanje pretplate
1011	11	UNSUBACK	Server → Klijent	Server potvrđuje otkazivanje pretplate
1100	12	PINGREQ	Klijent → Server	PING zahtjev
1101	13	PINGRESP	Server → Klijent	PING odgovor
1110	14	DISCONNECT	Klijent → Server	Klijent prekida vezu
1111	15	Rezervirano	Zabranjeno	Rezervirano

3.4.2. Zastavice

Zastavice (engl. *Flags*) se nalaze u području od 4 najmanje značajna bita u prvom bajtu fiksnog zaglavlja. Iako postoji 16 mogućih zastavica one se rijetko koriste. Zastavice koje se koriste kod objave od najznačajnije do najmanje značajne su: DUP, QoS i RETAIN, prikazane su u tablici 3.4. U polju paketa na kojem se trebaju nalaziti zastavice ako piše „reserved“ znači da nema zastavica namijenjenih paketu.

Tablica 3.4. Zastavice fiksnog zaglavlja MQTT protokola

Bitovi MSB → LSB	3	2	1	0
Nazivi zastavica	DUP	QoS		RETAIN

DUP - zastavica zauzima jednobitno polje. Ako je vrijednost ovog bita postavljena u 0, PUBLISH poruka šalje se prvi put. Za sve poruke koje imaju QoS 0, vrijednost DUP zastavice mora biti postavljena u 0. Ako je ova zastavica PUBLISH paketa postavljena u 1 označava da je ta poruka ponovo poslana, odnosno duplicirana. Pošiljatelj će iznova slati poruku nakon isteka vremenskog intervala, sve dok ne dobije PUBACK paket koji posjeduje ID poslane poruke. PUBACK paket je potvrda da je paket stigao primatelju.

QoS – zastavica zauzima 2 bitno polje i obavještava o kvaliteti usluge koja je korištena za objavu poruke, ima 3 moguće vrijednosti. U tablici 3.5 prikazane su moguće QoS vrijednosti s opisom.

Tablica 3.5. Vrijednosti QoS zastavice s opisom za MQTT protokol

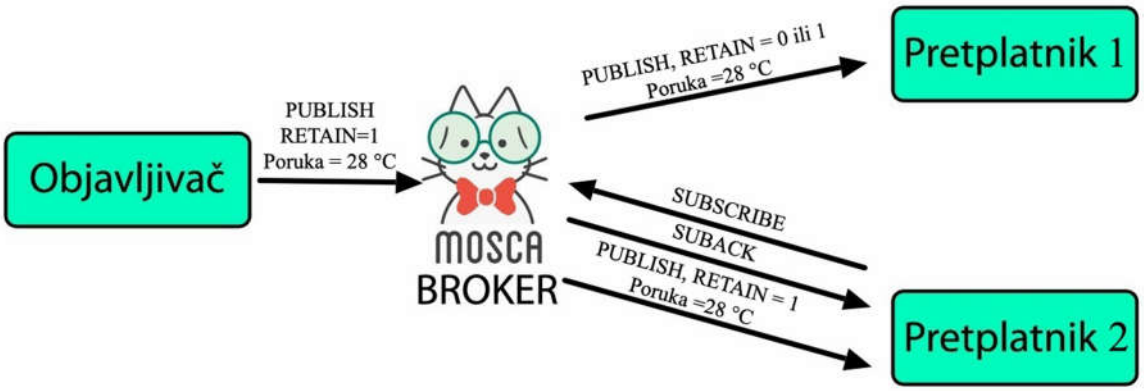
QoS Vrijednost	Bit 2	Bit 1	Opis
0	0	0	Najviše jedanput će se pokušati poslati poruka
1	0	1	Barem jedanput će se poslati poruka sa zahtijevanom potvrdom
2	1	0	Točno jedanput će se poslati poruka koristeći rukovanje u 4 koraka
-	1	1	Rezervirano

RETAIN je zastavica zadržavanja, duljine jednog bita. Ovu zastavicu prvo postavlja klijent pošiljatelj prilikom slanja PUBLISH poruke prema posredniku. Ako je ova zastavica postavljena u 0 posrednik prihvaća poruku i prosljeđuje je svim pretplatnicima koji su u tom trenutku pretplaćeni na temu. Kada klijent pošalje posredniku paket u kojemu vrijednost 0 postvaljena na mjestu RETAIN zastavice daje izbor posredniku da zamijeni s prethodno spremljenim paketom za navedenu temu ili da je zanemari.

Za poruke koje stižu do posrednika čija je vrijednost RETAIN bita postavljena u 1, posrednik će je pohraniti iako je QoS=0. Ako postoji spremljena poruka s temom jednakoj temi nove poruke koja ima RETAIN = 1, prethodno spremljenu poruku briše, a novu pohranjuje. Kada posrednik prosljeđuje poruku prethodno pretplaćenim primateljima RETAIN bit je postavljen u 0. Međutim kada se pretplati novi korisnik na temu jednakoj temi spremljene poruke posrednik šalje spremljenu poruku u kojoj je vrijednost RETAIN zastavice postavljena u 1. Novi pretplatnik će prema tome znati da je poruka najvjerojatnije neki vremenski period stajala pohranjena na posredniku.

Na slici 3.1 nalazi se prethodno navedeni slučaj kada objavljivač pošalje poruku s RETAIN zastavicom čiji je bit postavljen u 1. Poruka prenosi vrijednost temperature (28°C). Posrednik prihvaća poruku od objavljivača. Uspoređuje temu pristigle poruke s temama

pretplatnika. Pronalazi da pretplatnik 1 ima temu jednaku temi objavljene poruke. Postavlja vrijednost RETAIN zastavice prema vlastitim postavkama i prosljeđuje poruku.Posrednik je pohranio poruku za nove pretplatnike budući da je RETAIN = 1. Kada se pretplatnik 2 pretplati na posrednika s temom jednakoj temi pohranjene poruke, posrednik će proslijediti poruku pretplatniku 2 s RETAIN zastavom postavljenom u 1.



Slika 3.1. Prikaz toka PUBLISH poruke kojoj je RETAIN zastavica postavljena u 1

PUBLISH poruka kojoj je RETAIN zastavica postavljena u 1, a polje korisnog sadržaja veličine 0 bajtova tj. polje ne postoji. Posrednik taj paket ne pamti i ako ima prethodnu spremljenu poruku na istu temu obrisat će je.

3.4.3. Preostala duljina paketa

Ovo polje započinje s drugim bajtom fiksnog zaglavlja. Može biti duljine 1-4 bajta. Unutar svakog bajta 7 najmanje značajnih bitova (LSB) prikazuju preostali broj bajtova u paketu tj. veličinu varijabilnog zaglavlja i korisnog sadržaja. Najznačajniji bit u bajtu definira da li postoji idući red koji se odnosi na duljinu paketa.

3.5. Varijabilno zaglavlje

Varijabilno zaglavlje nalazi se između fiksnog zaglavlja i korisnog sadržaja. Veliki broj tipova poruka MQTT protokola sadrži ovo zaglavlje, međutim DISCONNECT, PINGREQ i PINGRESP poruke ne posjeduju varijabilno zaglavlje.

Ovisno o tipu poruke koja se šalje MQTT protokolom razlikuju se polja koja se mogu pronaći u varijabilnom zaglavlju. U nastavku će biti navedena i objašnjena polja varijabilnog zaglavlja prema tipu poruke.

3.5.1. CONNECT varijabilno zaglavlje

Ime protokola je polje koje se sadrže samo CONNECT poruke. U njemu se nalazi UTF-8 kodirani naziv protokola.

Verzija protokola je odvojeno polje kojim se definira verzija protokola.

U tablici 3.6 nalazi se dio varijabilnog zaglavlja CONNECT poruke u kojem su prikazane sve zastavice za ovo zaglavlje.

Tablica 3.6. Zastavice varijabilnog zaglavlja CONNECT poruke

Bitovi	7	6	5	4	3	2	1	0
Nazivi zastavica	Korisničko ime	Lozinka	Zadržavanje oporuke (*Will Retain*)	Kvaliteta usluge (*Will QoS*)		Zastavica oporuke (*Will Flag*)	Čista sesija (*Clean Session*)	Rezervirano

Zastavica korisničko ime (engl. *Username Flag*) zauzima jedan bit varijabilnog zaglavlja. Ako je vrijednost bita ove zastavice 1, unutar korisnog sadržaja nalazi se korisničko ime.

Lozinka (engl. *Password Flag*) zauzima također jedan bit unutar varijabilnog zaglavlja. Ako je bit zastavice postavljen u 1, unutar korisnog sadržaja nalazi se lozinka.

Zadržavanje oporuke (engl. *Will Retain*) upućuje da li posrednik treba zadržati poruku koja se objavljuje u slučaju da klijent neočekivano prekine vezu. Ako je bit zastavice oporuke (engl. *Will flag*) postavljen u 0, bit zastavice za zadržavanje poruke također mora biti postavljen u 0. Kada je zastavica oporuke postavljena u 1, bit zastavice za zadržavanje oporuke može imati vrijednost 0 (ne zadržavana poruka) ili 1 (zadržavana poruka).

Zastavica kvalitete usluge (engl.*Will QoS*) označava koju od 3 moguće kvalitete usluge posjeduje Will poruka.

Zastavica oporuke (engl. *Will flag*) obavještava da se u dijelu poruke koji se odnosi na korisnisadržaj nalazi Will poruka.

Zastavica čiste sesije (engl. *Clean Session*) ako je postavljena u 1 posrednik briše sve informacije o vezi s tim klijentom. A ako je postavljena u 0 poslužitelj nakon nove prijave klijenta šalje pohranjene poruke klijentu koje odgovaraju njegovim pretplatama. Pohranjene poruke mogu biti samo QoS 1 i QoS 2.

Nakon zastavica sljedeće polje prikazuje maksimalni vremenski interval između slanja poruke klijenta (engl. *Keep alive timer*). Izražen je u sekundama. Ima duljinu 16 bita. Klijent je obavezan unutar ovog perioda poslati poruku ili PINGREQ paket u slučaju nedostatka ostalih upravljačkih paketa.

3.5.2. CONNACK varijabilno zaglavlje

Varijabilno zaglavlje MQTT CONNACK poruke sastoji se od 3 polja:

- Zastavice potvrde povezivanja (engl. *Connect Acknowledge Flags*) je polje namijenjeno za buduću upotrebu. Bitovi ovog polja moraju biti postavljeni u 0.
- Zastavica postojanja sesije (engl. *Session present flag*) ako ima vrijednost 1 klijent i posrednik su već imali zapamćenu interakciju. Ako ima vrijednost 0 ostvaruje se čista sesija između klijenta i posrednika.
- Connect povratni kod (engl. *Connect Return Code*) definira da li je posrednik prihvatio CONNECT zahtjev. U tablici 3.7 su prikazane moguće vrijednosti Connect povratnog koda.

Tablica 3.7. Moguće vrijednosti povratnog koda CONNECT poruke s opisom

Vrijednost	Opis
0	Posrednik potvrđuje vezu s klijentom
1	Veza odbijena – posrednik ne podržava zahtijevanu verziju MQTT protokola klijenta
2	Veza odbijena – identifikator je ispravan, ali posrednik ne odobrava vezu s tim identifikatorom
3	Veza odbijena – posrednik nedostupan (uspostavljena veza, ali MQTT usluga nije dostupna)
4	Veza odbijena – neispravno korisničko ime ili lozinka
5	Veza odbijena – klijent nije autoriziran, nije mu dopušteno ostvarivanje veze
6	Veza odbijena – rezervirano za upotrebu u budućnosti

3.5.3. PUBLISH varijabilno zaglavlje

Varijabilno zaglavlje ovog paketa sastoji se od 2 polja:

- Ime teme (engl. *Topic Name*) – prvi dio (duljine 2 bajta) definira duljinu teme, a drugi naziv teme.
- Identifikator paketa (engl. *Packet Identifier*) prisutan je samo u PUBLISH paketima koji imaju QoS 1 ili QoS 2.

3.5.4. Varijabilno zaglavlje ostalih tipova poruka

PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE i UNSUBACK su tipovi poruke koji u varijabilnom zaglavlju imaju samo jedno polje – identifikator poruke. Polje je duljine 2 bajta. Prvi bajt je MSB, a drugi LSB. Između para klijent – posrednik ne može biti više od 65 535 poruka u jednom trenutku. Identifikatori se dodjeljuju porukama obično prema redoslijedu.

PINGREQ, PINGRESP i DISCONNECT su tipovi poruka koji ne posjeduju ni varijabilno zaglavlje ni korisni sadržaj.

3.6. Korisni sadržaj

Korisni sadržaj (engl. *Payload*) je posljednji dio MQTT poruke. Sastoji se od različitih polja prema tipu poruke.Nema određenu duljinu.

CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, UNSUBACK, PINGREQ, PINGRESP,DISCONNECT poruke nemaju ovaj dio MQTT poruke.

3.6.1. CONNECT korisni sadržaj

CONNECT korisni sadržaj sastoji se od 5 polja:

- Identifikator klijenta (engl. *Client Identifier*) identificira svakog klijenta koji se povezuje s poslužiteljem. Svaki klijent trebao bi imati jedinstven ID s posrednikom. Posrednik ga koristi za identifikaciju i trenutno stanje klijenta.
- Will tema (engl. *Will Topic*) predstavlja temu Will poruke. Ovo polje postoji ako je u CONNECT varijabilnom zaglavlju zastava oporuke postavljena u 1.
- Will poruka (engl. *Will Message*) je polje ispunjeno sadržajem Will poruke ako je zastava oporuke postavljena u 1.
- Ime korisnika (engl. *Username*) kao polje postoji samo ako je unutar varijabilnog zaglavlja zastavica za ime korisnika postavljena u 1. Posredniku služi za autentifikaciju i autorizaciju.
- Lozinka (engl. *Password*) polje postoji ako bit zastavice lozinka ima vrijednost 1.

3.6.2. PUBLISH korisni sadržaj

PUBLISH korisni sadržaj ima samo jedno polje PUBLISH poruku. Specifično je po tome što omogućuje prijenos raznih tipova podataka. Pošiljatelj tako može poslati unutar ovog polja binarne i tekstualne podatke, XML ili JSON.

3.6.3. SUBSCRIBE korisni sadržaj

Ovaj dio paketa SUBSCRIBE poruke sastoji se od 2 dijela: imena teme na koju se želi pretplatiti klijent i kvaliteti usluge koju zahtjeva. Kvaliteta usluge koju odabere klijent je najviša kojom će mu se isporučivati paketi. Ako se pretplatnik želi pretplatiti na više tema to može učiniti preko jednog SUBSCRIBE paketa. Nakon teme i zahtijevanog QoS može se dodati niz tema i zahtijevanih usluga.

3.6.4.SUBACK korisni sadržaj

Korisni sadržaj SUBACK poruke sačinjen je od jednog polja – povratnog koda (engl. *Return code*). On obavještava klijenta kako je posrednik primio SUBSCRIBE paket. Tablica 3.8 prikazuje dopuštene povratne kodove. Ostali kodovi se ne koriste. Rezervirani su za buduću upotrebu.

Tablica 3.8. Prikaz povratnih kodova u SUBACK porucis opisom

Povratni kodovi	Opis
0x00	Najveći QoS 0
0x01	Najveći QoS 1
0x02	Najveći QoS 2
0x80	Neuspješna pretplata na temu

3.6.5. UNSUBSCRIBE korisni sadržaj

Sastoji se samo od naziva teme s koje se klijent želi odjaviti. Prva 2 bajta definiraju duljinu polja naziva teme. Može se navesti veći broj tema čije poruke klijent ne želi primati.

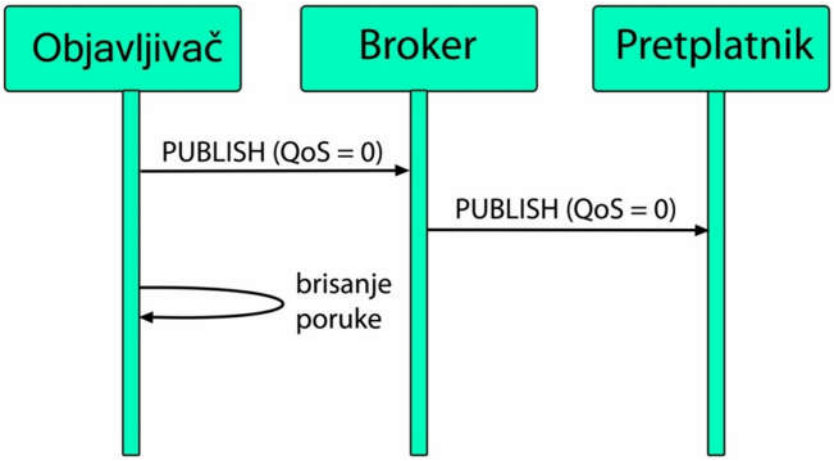
3.7. Kvaliteta usluge QoS

Razina kvalitete usluge je ugovor između pošiljatelja i primatelja. Definira koliko puta pošiljatelj šalje istu poruku i da li traži potvrdu od primatelja da je primio poruku. Protokol je simetričan što znači da klijent i server mogu biti i pošiljatelji i primatelji. Ako pošiljatelj treba poslati poruku s istim sadržajem većem broju klijenata, poruku svakom klijentu šalje posebno s različitom kvalitetom usluge ako je tako definirano.Postoje dvije vrste poruka: poruke koje objavljivač šalje posredniku i poruke koje šalje posrednik pretplatnicima.U ovim porukama vidljive su razlike. Objavljivač šalje brokeru poruku određene razine kvalitete usluge. Tu poruku prosljeđuje posrednik pretplatnicima pomoću razine kvalitete usluge koju jeklijent definirao prilikom pretplate na posrednika i određenu temu. Poruka koju je objavio objavljivač posredniku s QoS razine 2, a pretplatnik je pretplaćen s QoS 2, tu poruku će posrednik proslijediti s QoS 2. Međutim ako objavljivač objavi poruku QoS razine 2, a pretplatnik je pretplaćen s QoS 0, primljenu poruku posrednik će proslijediti s QoS 0.

MQTT protokol posjeduje 3 razine kvaliteta usluge. Povećanjem broja razine kvalitete usluge povećava se i vjerojatnost da će poslanu poruku primiti prijemnik točno jedanput, ali također se povećava količina podataka koji se prenose i složenost isporuke je veća.

3.7.1. Kvaliteta usluge razine 0

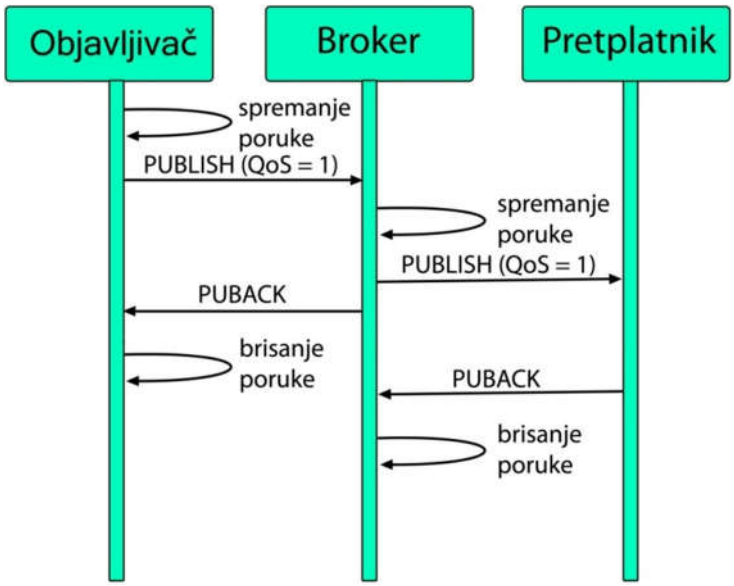
Najniža razina usluge je razine 0. Kvaliteta usluge u ovom slučaju ovisi o mogućnostima mreže, osigurava relativno nepouzdanu uslugu prijenosa jer ne postoji nikakva potvrda da će poslani paket stići do odredišta (engl. *Best effort*). Pošiljatelj (objavljivač ili posrednik) šalje poruku prijemniku samo jednom. U zaglavlju poruke vrijednost parametara je QoS=0 i DUP=0. Od prijemnika se ne zahtijeva potvrda, a pošiljatelj ne pokušava ponovo poslati poruku. QoS 0 često se naziva i pošalji i zaboravi (engl. *fire and forget*) i pruža jamstvo isporuke kao osnovni TCP protokol. Pošiljatelj nakon slanja poruke primateljimasprema poruku nego je zaboravlja. Primatelj ne odgovara na poruku. Primjer QoS 0 prikazan je PUBLISH paketom na slici 3.2.



Slika 3.2. Primjer toka PUBLISH poruke s kvalitetom usluge 0

3.7.2. Kvaliteta usluge razine 1

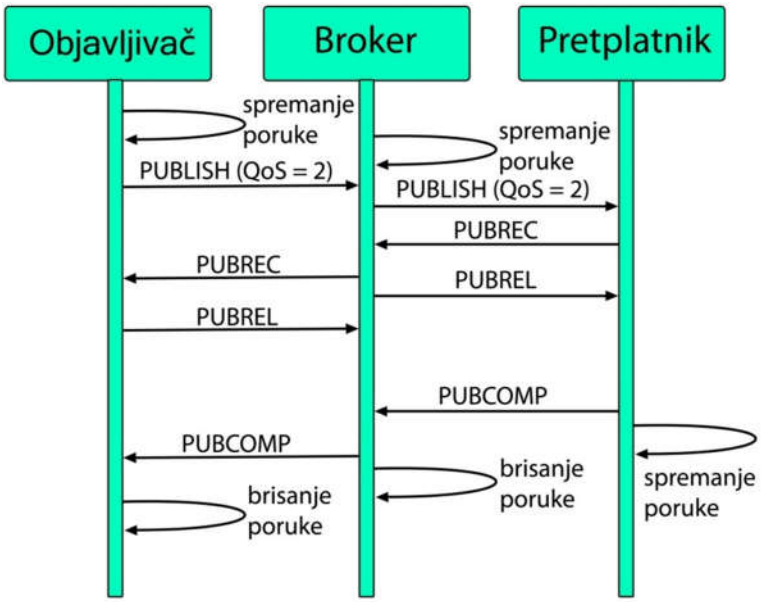
Razina 1 omogućuje dobru kvalitetu usluge jer osigurava da poruka stigne na odredište barem jedanput, ali postoji mogućnost dupliciranja poruke. Pošiljatelj ovu poruku sprema prije slanja kako bi istu poruku mogao ponovo poslati ako ne dobije potvrdu o primljenoj poruci od primatelja. Pošiljatelj šalje PUBLISH paket u kojem su vrijednosti QoS=1, DUP=0 i do tada neupotrebljeni identifikator (ID oznaka). Ako nakon isteka određenog vremena pošiljatelj ne dobije ACK potvrdu (PUBACK porukom). On šalje ponovo prethodno poslanu poruku s DUP=1 sve dok ne dobije potvrdu za tu poruku. Kada dobije potvrdu o primitku poruke briše je iz spremnika. Na slici 3.3 prikazano je slanje PUBLISH paketa s QoS 1.



Slika 3.3. Primjer toka PUBLISH poruke s kvalitetom usluge 1

3.7.3. Kvaliteta usluge razine 2

Najviša razina usluge koju MQTT protokol podržava je razina 2. Najsigurnija je ali i sporija od svih ostalih razina usluge koje pruža. Ostvaruje se kroz dvije sesije između pošiljatelja i primatelja. Ne dolazi do gubljenja poruka, ali ni dupliciranja. Pošiljatelj šalje PUBLISH paket koji ima vrijednosti QoS=2, DUP=0 i neupotrebljeni identifikator. On čuva pohranjenu poruku dok ne dobije potvrdu (PUBREC paket) od prijemnika. Kada primi PUBREC paket pošiljatelj briše prethodno spremljenu poruku jer je siguran da je primatelj dobio prethodnu poruku. PUBREL paket s istim identifikatorom kao u PUBLISH paketu sada objavljivač šalje primatelju. On odgovara pošiljatelju PUBCOMP paketom. Identifikator korišten za ove pakete nakon primitka PUBCOMP paketa postaje dostupan za neki od budućih paketa. Primjer slanja PUBLISH poruke koja ima QoS 2 prikazan je na slici 3.4.



Slika 3.4. Primjer toka PUBLISH poruke s kvalitetom usluge 2

3.8. MQTT veza (uspostava, održavanje, prekid)

Da bi se mogla ostvariti MQTT veza između klijenta i posrednika, svi uređaji koji žele sudjelovati u razmjeni kratkih poruka koristeći MQTT protokol moraju koristiti TCP/IP protokol na transportnom/mrežnom sloju. Prilikom uspostavljanja veze s posrednikom neophodno je ostvarivanje TCP veze iznad koje se stvara MQTT sesija.Veza se ostvaruje između klijenta i posrednika. Klijenti se ne povezuju nikada direktno međusobno, obavezno se svi povezuju na posrednik jer se preko njega odvija sva komunikacija.

Za uspostavljanje veze klijent koji je inicijator mora poslati CONNECT MQTT upravljački paket posredniku. Paket sadrži korisni sadržaj u kojem su postavljeni parametri za pokretanje veze i provjeru autentičnosti i autorizacije. Posrednik odgovara svakom klijentu sa CONNACK MQTT paketom i statusnim kodom koji definira uspješno povezivanje klijenta. Time je ostvarena veza koju posrednik pamti. Veza ostaje dostupna sve dok klijent ne zatraži prekid ili do nestanka konekcije.

Ako klijent pošalje posredniku nevažeći CONNECT paket, posrednik će odmah zatvoriti vezu.

Klijent mora održavati vezu s posrednikom. Ako je neaktivan duže vrijeme mora poslati PINGREQ poruku. Posrednik odgovara sa PINGRES kojim potvrđuje otvorenu vezu te da

nema greški na vezi i posredniku. Ako prođe dulji period od 1.5 vremenskog intervala za održavanje veze posrednik će odspojiti klijenta.

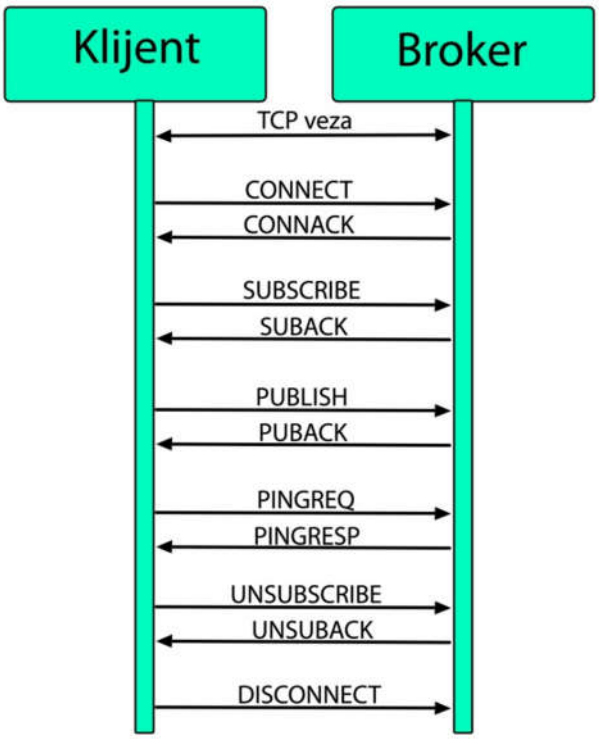
Ako klijent ne dobije PINGRES potvrdu od posrednika dužan je također prekinuti vezu s posrednikom.

Klijent koji želi objaviti poruku šalje PUBLISH poruku posredniku. Unutar poruke se nalazi blok podataka koji se želi prenijeti. Budući da podatci mogu biti napisani u različitim oblicima, šalju se u bajtovima. Potrebno je navesti i temu kako bi posrednik znao kojim pretplatnicima treba proslijediti poruku. Ako tema nije postavljena, posrednik će postaviti temu.

Kada klijent uspostavi vezu s posrednikom ako se želi pretplatiti na temu ili više njih, šalje SUBSCRIBE paket. Pretplatiti se može na svaku temu pojedinačno ili koristiti zamjenske znakove koji omogućuju pretplatu na cijelu granu teme ili dio. Ako je sve uredu posrednik odgovara klijentu SUBACK paketom. Klijentu se šalju zadržane poruke na temu ako ih ima.

UNSUBSCRIBE paket šalje pretplatnik posredniku kada se želi odjaviti s pretplate na neku temu. Kao potvrdu da je primio zahtjev za prekidom pretplate, posrednik šalje UNSUBACK paket.

DISCONNECT poruku šalju objavljivač i pretplatnik kada žele prekinuti vezu s posrednikom. Kada se klijent odjavi na ovaj način ima mogućnost povezivanja s posrednikom u budućnosti koristeći isti ID. Za slučaj prekida bez ove poruke, pretplatnicima se šalje Will oporuka.



Slika 3.5. Primjer toka poruka između klijenta i posrednika korištenjem MQTT protokola

3.9. MQTT teme

Sve PUBLISH poruke moraju imati temu (engl. *Topic*).Prilikom komunikacije MQTT protokolom objavljivač ne zna IP adresu krajnjeg korisnika. Kada poruka stigne posredniku on na osnovu teme pregledava kojim klijentima treba poruku proslijediti. Kada provjeri koji klijenti su pretplaćeni na temu pristigle poruke, šalje im poruku.

PUBLISH poruka može imati jednu ili više razina tema. Ako tema posjeduje više razina one su poredane u hijerarhijskom nizu i međusobno su odvojene kosom crtom (/).

Znakovi + i # služe kao zamjenske oznake (engl. *Wildcards*) za pretplatu pretplatnika na veći opseg tema.

Znak + zamjenjuje jednu razinu teme sa svim mogućim nazivima koji se mogu naći na tom mjestu.

Tablica 3.9.Primjeri tema koje može zamijeniti znak + i koje ne

Tema sa znakom +	Primjer tema na koje podržava zapis u prvom stupcu	Primjer tema na koje ne podržava zapis u prvom stupcu
home/room/+	home/room/humidity home/room/temperature	apartment/room/humidity home/kitchen/humidity
home/+/humidity	home/room/humidity home/kitchen/humidity	apartment/room/humidity home/humidity

Višerazinski zamjenski znak (#) može mijenjati veći broj razina teme ali se mora nalaziti na kraju naziva teme iza kose crte.

Tablica 3.10. Primjeri tema koje može zamijeniti znak # i koje ne

Tema sa znakom #	Primjer tema na koje podržava zapis u prvom stupcu	Primjer tema na koje ne podržava zapis u prvom stupcu
home/#	home home/room home/kitchen home/room/humidity home/kitchen /temperature	apartment apartment/room apartment/home

U tablici 3.9 i 3.10 navedeno je kako zamjenskim znakovima povećati opseg tema.

Prilikom dodjeljivanja teme korisnik odabire željeni naziv. Teme koje započinju sa znakom $ potrebno je izbjegavati jer se takve teme koriste za internu statistiku posrednika.

3.10. Usporedba HTTP i MQTT protokola

Budući da se MQTT nalazi na aplikacijskoj razini ISO-OSI modela može se usporediti s HTTP protokolom koji se nalazi na istoj razini. HTTP je najpoznatiji protokol na ovoj

razini. Odgovoran je za pristup web stranicama (WWW). Najčešće se koristi tako da se upiše URLu pretraživač, tada protokol šalje zahtjev poslužitelju. Poslužitelj odgovara HTTP odgovorom natrag u pretraživač.

U tablici 3.11 predstavljene su sličnosti i različitosti ovih protokola.

Tablica 3.11. Usporedba HTTP i MQTT protokola

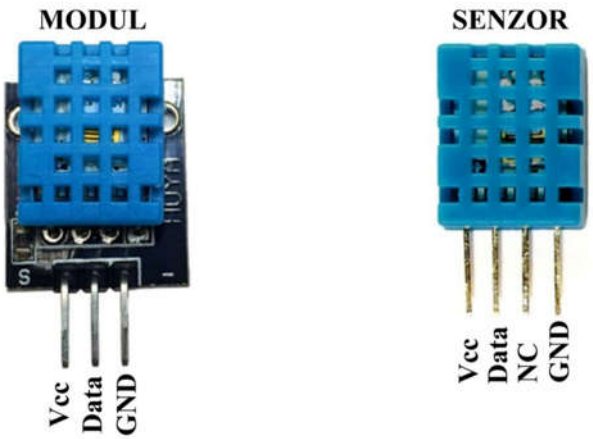
	HTTP	MQTT
Glavna primjena	Razmjena dokumenata	Razmjena podataka
Model	zahtjev/odgovor	objavi/pretplati
Veličina poruke	velike	male poruke do 256 MB
Kvaliteta usluge	ograničena	QoS 0, QoS 1 i QoS 2
Transportni protokol	TCP	TCP
Zadani port	80/443 (TLS/SSL)	1883/8883 (TLS/SSL)
Distribucija podataka	1 na 1	1 na 0, 1 ili n
Složenost	više složen	jednostavan
Metode	Get, Post, Head, Put, Patch, Options, Connect i Delete	Connect, Disconnect, Publish, Subscribe, Unsubscribe i Close

4. IMPLEMENTACIJA MQTT PROTOKOLA

Za implementaciju MQTT protokola osim navedenih platformi u 2 poglavlju, potrebne su 2 komponente (senzor i ploča s mikrokontrolerom) i ožičenje. U prvom dijelu ovog poglavlja opisane su komponente, a nakon njih implementacija.

4.1. DHT11

Senzori su uređaji koji mjere određenu fizikalnu veličinu i daju izlaznu vrijednost te veličine. Jedan od jednostavnijih senzora je DHT11.Ovaj senzor mjeri dvije fizikalne veličine (temperaturu i vlagu).Posjeduje kalibrirane digitalne izlazne signale. Senzor ima NTC otpornik kojem se promjenom temperature mijenja otpornost i dvije elektrode između kojih se nalazi podloga za zadržavanje vlage na kojem se u slučaju dolaska vlage oslobađaju ioni pa se povećava vodljivost između elektroda. 8-bitni mikrokontroler iskazuje dobivene vrijednosti kao serijske podatke. Ovisno o tome da li se radi o senzoru ili senzorskom modulu različit je broj pinova. SenzorDHT11 ima 4 pina, a modul DHT11 ima 3 pina. Na slici 4.1 prikazani su s označenim pinovima. Jednak broj pinova se koristi pri spajanju u oba slučaja. U tablicama 4.1 i 4.2 prikazani su pinovi modula i senzora s pojašnjenjem.



Slika 4.1. DHT11 modul i senzor s označenim pinovima

Tablica 4.1. Pinovi DHT11 senzora s opisom

Broj pina	Naziv pina	Opis
1	Vcc	Predstavlja napon napajanja senzora koji treba iznositi 3 - 5.5 V (preporučeni je 5.5 V, a u slučaju korištenja 3.3V kabel mora biti manji od 1 m)
2	Data	Vrijednosti vlage i temperature prosljeđuju se kao serijski podatci drugim uređajima
3	NC	Ne posjeduje vezu stoga se ne koristi
4	GND	Predstavlja uzemljenje strujnog kruga

Tablica 4.2. Pinovi DHT11 modula s opisom

Broj pina	Naziv pina	Opis
1	Vcc	Predstavlja napon napajanja senzora koji treba iznositi 3 - 5.5 V (preporučeni je 5.5 V, a u slučaju korištenja 3.3V kabel mora biti manji od 1 m)
2	Data	Vrijednosti vlage i temperature prosljeđuju se kao serijski podatci drugim uređajima
3	GND	Predstavlja uzemljenje strujnog kruga

DHT11 senzor može mjeriti vrijednosti temperature i vlage svake sekunde ili rjeđe. Napon napajanja za senzor treba iznositi u opsegu 3-5.5 V. Struja u stanju rada senzora treba iznositi0.2-1 mA, a u stanju pripravnosti 100-150 μA.

Senzor može mjeriti vrijednosti temperature u opsegu 0-50 ºC točnošću±2 ºC i vrijednosti vlažnosti zraka 20 - 80 % točnošću ±5 %.

4.2. ESP32

Nasljednik ESP8266 mikrokontrolera je ESP32 mikrokontroler. Mnogo kompleksniji i napredniji od prethodnika. Dizajniran je za korištenje u IoT svijetu. Ostvaruje dvije vrste bežičnog povezivanja integrirani WiFi (802.11 b/g/n) i dvostruki način rada Bluetooth-a (*Bluetooth Low Energy* i *Legacy Bluetooth*). Kada je u ulozi klijenta može komunicirati s drugim ruterima, ali kada je u ulozi servera ima mogućnost stvaranja pristupne točke. Na ESP32 integriran je čip ESP-WROOM-32 (ulazni napon postavljen je na 3.3V) i CP2102 čip koji omogućuje povezivanje na računalo bez programatora. Ima također dva gumba

ENABLE i BOOT. Kada se u Arduino IDE odabere Upload koda potrebno je držati na pločici BOOT tipku sve dok ne bude gotov prijenos koda. Kada je gotov prijenos otpusti se BOOT tipka te kratko pritisne ENABLE tipka kako bi procesor ploče počeo izvršavati kod.

ESP WROOM 32 ploča ima dvije verzije koje se razlikuju u broju pinova. Verzija s 30 pinova je jednostavnija jer verzija s 36 pinova zahtjeva veću preciznost pri programiranju zbog povećane vjerojatnosti nastanka nepravilnosti. Ploča korištena u ovom radu ima 30 pinova.

4.3. Implementacija

Za implementaciju MQTT protokola koristit će se prethodno analizirane komponente i platforme. Implementacija se može podijeliti u 3 dijela jer će se vrijednosti senzora objavljivati u svim platformama pod različitim okolnostima.

Unutar Arduino IDE platforme izmjereni podatci senzora stizat će USB kabelom kojim je ostvarena veza za utiskivanje koda i napajanje. MQTT protokol neće biti korišten i vrijednosti će biti vidljive samo u Serial Monitoru.

U Node-RED platformu podatci senzora stizat će putem lokalne mreže. Unutar Arduino IDE napisan je kod za povezivanje ploče na WiFi mrežu, slanje izmjerenih podataka s temom na mosca posrednik koji se nalazi u Node-RED te prosljeđuje vrijednosti na dashboard. Budući da se koristi posrednik i porukama su dodijeljene teme kojima se prenose podatci senzora implementiran je MQTT protokol.

Da bi vrijednosti bile dostupne svima na internetu koristit će se ThingSpeak aplikacija. Početni tok podataka jednak je kao i za Node-RED. Podatci se šalju WiFi mrežom do Node-Red gdje stižu na mosca posrednik,. nakon kojeg se podatci prosljeđuju na kanale ThingSpeak-a. Prilikom ove komunikacije korišten je MQTT protokol. Vrijednosti senzora dostupne su svima ako su kanali javni. U nastavku će biti detaljno objašnjena komunikacija na svim platformama.

Za početak implementacije neophodno je ožičiti komponente koje se koriste. Unutar ovog poglavlja opisani su pinovi DHT11 senzora i modula. Kako bi se ispravno povezali na ploču najbolje je koristiti tablicu podataka (engl. *datasheet*) ploče budući da su tamo

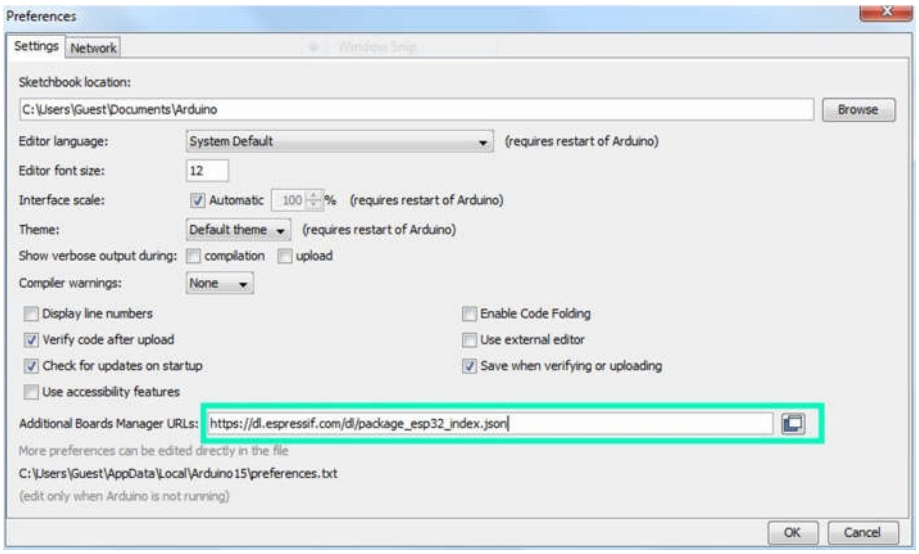
jasnije označeni pinovi u odnosu na oznake sa samoj ploči. Postoje razne verzije modula stoga je potrebno obratiti pozornost pri pretraživanju.

Za napajanje ploče i senzora te razmjenu podataka ploča se može povezati s računalom USB kabelom koji ima 2 konektora (Micro USB koji se povezuje na ploču i npr. USB3.0 koji se povezuje na računalo).

4.3.1. Arduino IDE implementacija

Kod koji će se utisnuti u mikrokontroler ESP32 ploče napisan je unutar Arduino IDE. Stoga je neophodno postaviti odgovarajuće parametre programu koji odgovaraju ploči kako bi se mogla uspostaviti komunikacijska veza između ploče i programa.

U otvorenom Arduino IDE prozoru odabirom File>Preferences putanje otvorit će se pomoćni prozor. U Additional Board Manager URLs polje potrebno je zalijepiti https://dl.espressif.com/dl/package_ESP32_index.json kao na slici 4.2.



Slika 4.2. Dodavanje URL-a za ESP32 u Arduino IDE

ESP32 ploča nije hardverski Arduino proizvod stoga je potrebno preuzeti instalaciju za ovu ploču. Odabirom Tools>Board>Board Manager putanje otvorit će se pomoćni prozor. Ploče se mogu pretraživati ručno ili upisivanjem naziva unutar pretraživača. Pronalaskom

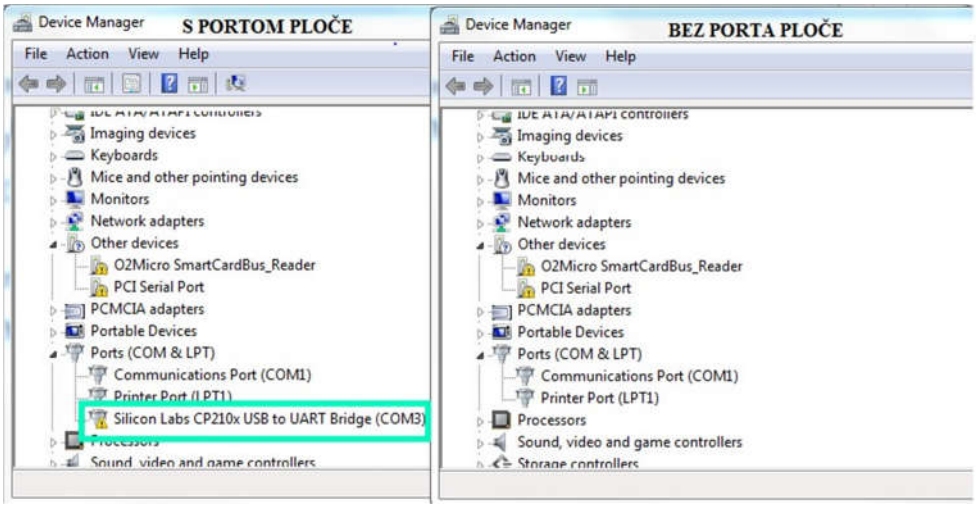
odgovarajuće verzije klikom na Install pokreće se instalacija. Na slici 4.3 prikazana je instalacija ESP32 ploče.

Slika 4.3. Instalacija biblioteke ESP32

Nakon instalacije potrebno je označiti ploču odabirom Tools>Board putanje. Za ploču koja se koristi u ovom radu potrebno je odabrati ESP32 Dev Module.

Za ostvarivanje veze potrebno je odabrati i priključak na koji je povezana ploča. U Windows operativnom sustavu broj priključka (engl. *port*) provjerava se na sljedeći način. Klikom na start izbornik u pretraživač unosom Device Manager ili u Control Panel-u se može pronaći Device Manager i odabrati.U Windows 7 operativnom sustavu u odjeljku Ports (COM & LPT) navedeni su nazivi i broj COM ili LPT priključka. Računalo i ploča trebaju biti povezani kabelom. Neophodno je pogledati koji se priključci nalaze unavedenom odjeljku unutar prozora i zapamtiti njihove nazive, zatim prekinuti vezu (odspojiti kabel) te provjeriti koji sepriključcipojavljuju nakon prekida veze. Priključak koji nedostaje ostvaruje vezu s pločom. Kada se ponovo poveže ploča s priključkompotrebno ga je označiti unutar Arduino IDE, odabirom Tools>Port.

Na slici 4.4 prikazan je izgled Device Manager-a s priključenim portom ploče na lijevoj strani i isključenim portom s kojim se ostvaruje veza s pločom na desnoj strani.



Slika 4.4. Prikaz Device Manager prozora s uključenim i isključenim priključkom kojim se ostvaruje veza s pločom

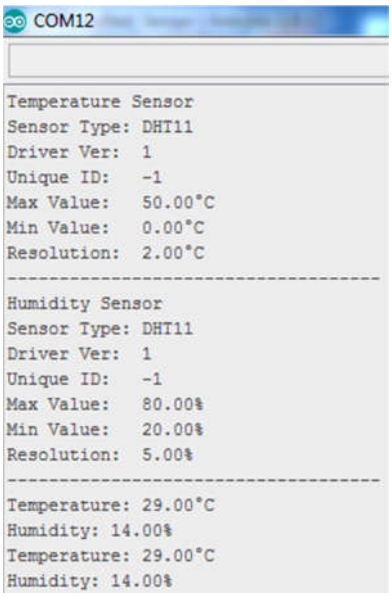
Za provjeru ispravne komunikacije između ploče i programa najjednostavnije je koristiti neki od postojećih kodova koji se nalaze u primjerima npr. kod u kojem treperi LED dioda na modulu. Može se pronaći odabirom File>Examples>Basics>Blink putanje. Potrebno je inicijalizirati izlazni digitalni pin unutar koda. Za ESP32 ploču to je pin 2.

Nakon utiskivanja koda u pločicu i pokretanja izvršavanja koda LED dioda će se paliti i gasiti naizmjenično nakon isteka postavljenih vremenskih intervala.

Naredni korak je preuzeti biblioteku za senzor. Odabirom Tools>Manage Libraries u pomoćnom prozoru u pretraživač unosnom DHT prikažu se biblioteke koje se odnose na DHT senzor. Odabirom DHT sensor library i njene verzije može se započeti s instalacijom. Nakon gotove instalacije odabirom nekog od postojećih primjera koda za DHT senzor može se provjeriti komunikacija. Odabirom File>Examples>DHT sensor library>DHT_Unified_Sensor putanje otvorit će se primjer koda unutar kojeg je potrebno provjeriti i promijeniti muodređene parametre. Tip senzora treba postaviti (DHT11 u ovom slučaju)i također inicijalizirati digitalni pin koji se nalazi na ploči kojim je ostvarena serijska veza između senzora i ploče.

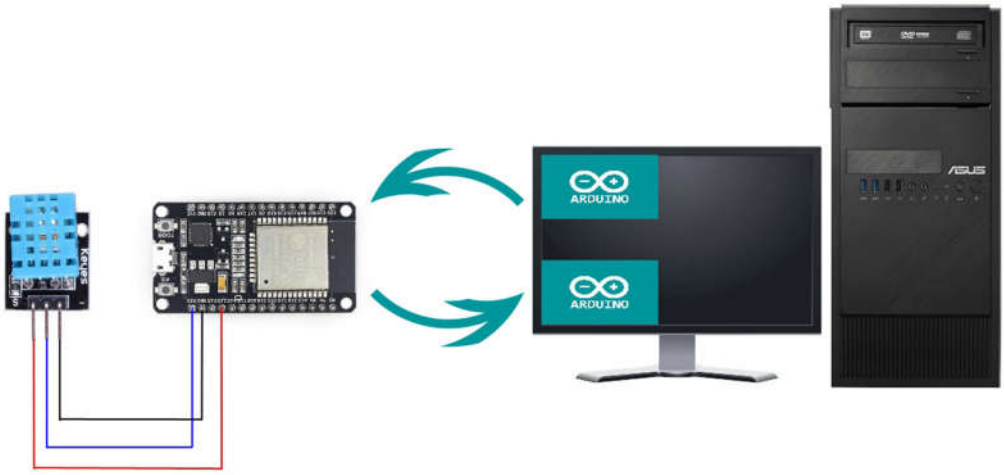
Kada je kod ispravno zapisan klikom na upload kod se sprema na mikrokontroler ploče, ali za ESP32 ploču potrebno je držati BOOT tipku za vrijeme učitavanja koda u mikrokontroler. Izvršavanje programa započinje klikom na ENABLE tipku. U Serial

Monitoru osim vrijednosti temperature i vlažnosti prikazat će se i opsezi u kojima mjeri senzor i njegova točnost kao na slici 4.5.



Slika 4.5. Serial monitor nakon pokretanja Arduino IDE koda na mikrokontroleru ploče

Slika 4.6 prikazuje prvi dio implementacije koji ne koristi MQTT protokol i mrežu za prijenos podataka. Koristi kabel kojim je utisnut kod u pločicu za prijenos izmjerenih podataka do računala i prikazuje ih u Serial Monitoru-u. Plave strelice na slici stoga predstavljaju USB kabel koji će prenositi podatke u oba smjera. Na slici je prikazana žična veza senzora i ploče.



Slika 4.6. Izgled sustava prilikom Arduino IDE implementacije

4.3.2. Node-RED implementacija

Za prikaz vrijednosti senzora u Node-RED potrebno je urediti prethodni Arduino IDE programski kod.

Osim DHT biblioteke potrebno je instalirati PubSubClient i WiFi biblioteke. PubSubClient biblioteka dopušta slanje i primanje MQTT poruka i podržava posljednju 3.1.1 MQTT verziju, a WiFi biblioteka dopušta ploči povezivanje na internet.

Za provjeru WiFi modula na ploči može se koristiti gotovi primjer odabirom File>Examples>WiFi>WiFiScan putanje. Pokretanjem ovog koda u Serial Monitoru ispisat će se mreže koje modul dohvaća. Ako se prilikom pokretanja koda na ploči ugasi priključak računala s kojim je ploča povezana potrebno je zamijeniti kabel.

Da bi se napisao složeniji kod u kojem se vrijednosti senzora šalju kao MQTT poruke do MQTT posrednika korištenjem WiFi mreže najbolje je pogledati primjere kodova novih biblioteka.

U Arduino IDE kod iz prethodne implementacije potrebno je dodati:

- nazive biblioteka
- ime i lozinku mreže
- dvije vremenske varijable

- IP adresu MQTT posrednika i njegov port
- Inicijalizaciju klijenta
- 2 varijable u kojima će se pohranjivati vrijednosti temperature i vlage
- petlju koja omogućuje povezivanje WiFi modula na pristupnu točku
- funkciju za ponovno ostvarivanje WiFi veze u slučaju prekida
- funkciju koja uzima vrijednosti senzora svakih 30 sekundi
- 2 nove varijable u koje se pohranjuju izmjerene vrijednosti
- Funkcije za pretvaranje vrijednosti
- MQTT teme porukama

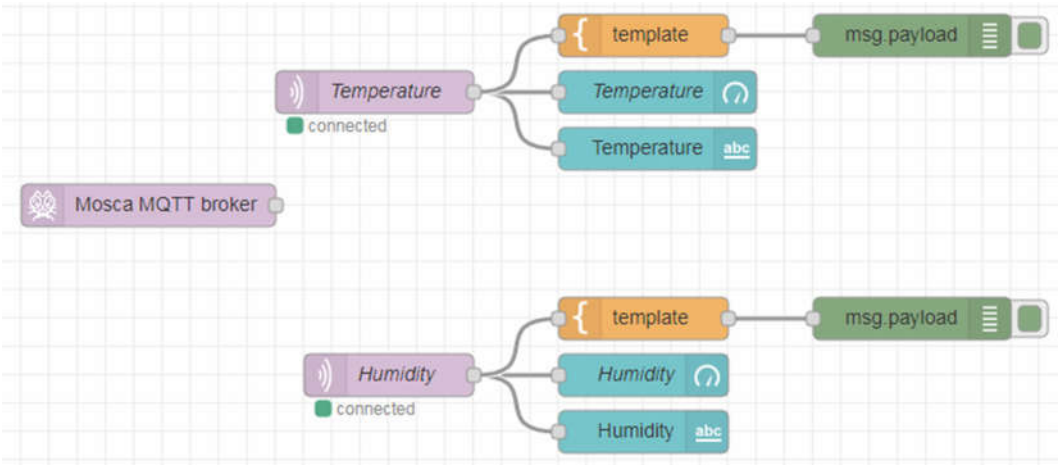
Cijeli Arduino programski kod, koji se utiskuje u ploču, s detaljnim opisom naredbi nalazi se u prilozima.

U Node-RED aplikaciji za korištenje mosca posrednika neophodno je instalirati grupu čvorova pod nazivom node-red-contrib-MQTT-broker, a za prikazivanje vrijednosti u dashboard-u potrebno je instalirati node-red-dashboard grupu čvorova.

Nakon što su izvršene sve dodatne instalacije unutar programa može se započeti s ubacivanjem čvorova u glavni radni prostor.

Početni čvor je mosca in koji se nalazi u kategoriji input. Predstavlja posrednika MQTT protokola. Zadužen je za prihvaćanje svih poruka od klijenata koji objavljuju MQTT poruke s IP adresom i portom koji odgovara posredniku. Poruke zatim filtrira te ih prosljeđuje klijentima prema temama koji se nalaze unutar zaglavlja poruka

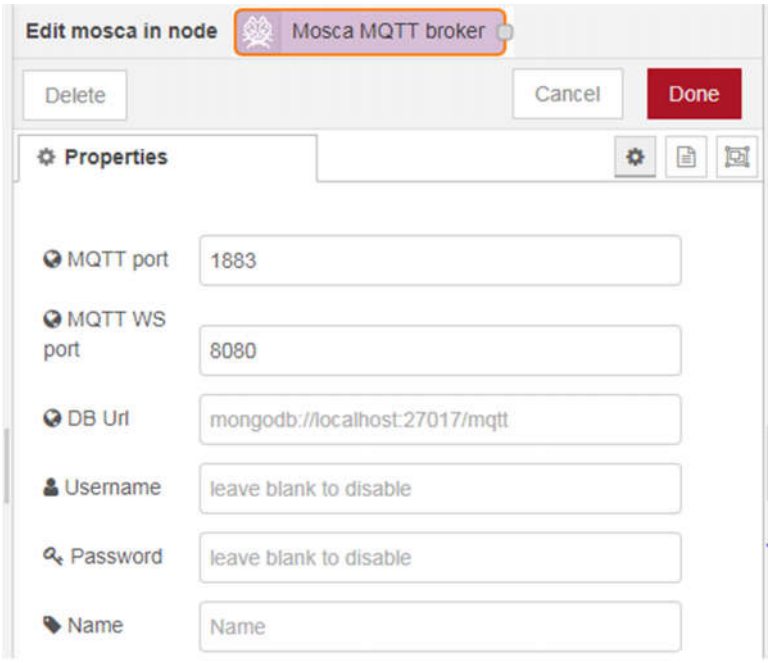
Na slici 4.7 je prikazan MQTT čvor koji predstavlja mosca posrednik i MQTT čvor koji prihvaća poruke s posrednika i prosljeđuje ih čvorovima koji omogućuju pregled prenesenih poruka. U nastavku će biti opisani korišteni čvorovi s postavljenim parametrima.



Slika 4.7. Izgled čvorova i njihovih veza prilikom Node-RED implementacije

Početni čvor je mosca in koji se nalazi u kategoriji input. Predstavlja posrednika MQTT protokola. Zadužen je za prihvaćanje svih poruka od klijenata koji objavljuju MQTT poruke s njegovom IP adresom i portom. Poruke zatim filtrira te ih prosljeđuje klijentima prema temama koji se nalaze unutar zaglavlja poruka. Port mosca posrednika najčešće se postavlja na vrijednost 1883. Vrijednost IP adrese posrednika je jednaka lokalnoj IP adresi jer je pokrenut u lokalnoj mreži.

Na slici 4.8 prikazan je izgled mosca in čvora s postavljenim vrijednostima.

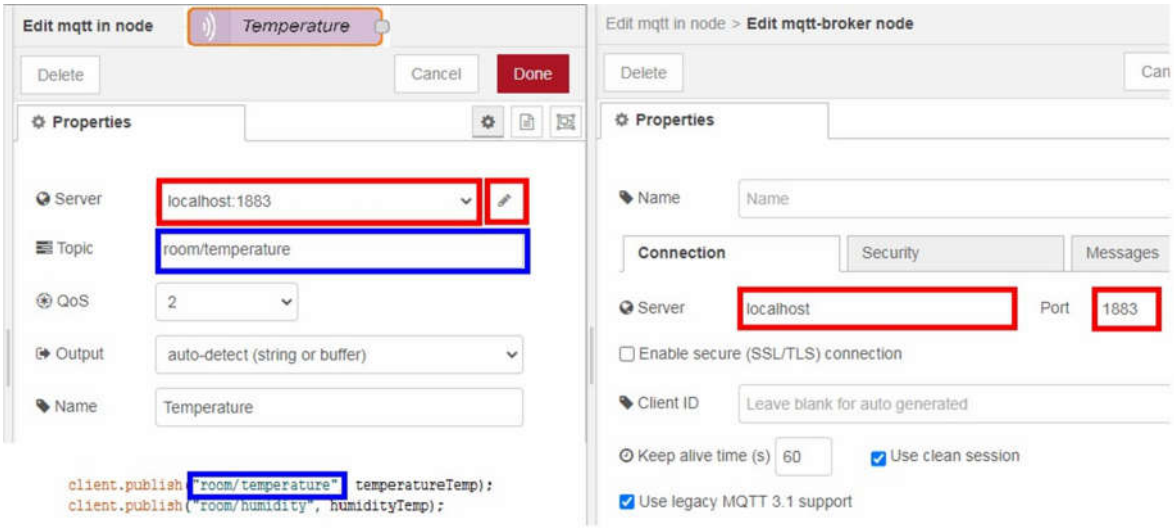


Slika 4.8. Postavke mosca in čvora

Sljedeća 2 mqtt in čvora su iz palete network. Predstavljaju dodatak prethodnom čvoru. Prihvaćaju MQTT poruke koje odgovaraju temi na koju su pretplaćeni.

Na slici 4.9 prikazan je čvor koji prihvaća vrijednost temperature. Crvenim okvirom ocrtana su polja koja trebaju imati vrijednosti mosca in čvora. Na isti način trebaju biti ispunjeni mqtt in čvorovi koji žele primati poruke s istog posrednika. Plavim okvirom ocrtana je tema na koju se čvor pretplaćuje. Ovaj parametar je različit za mqtt in čvor koji želi prihvaćati MQTT poruke s drugom temom.

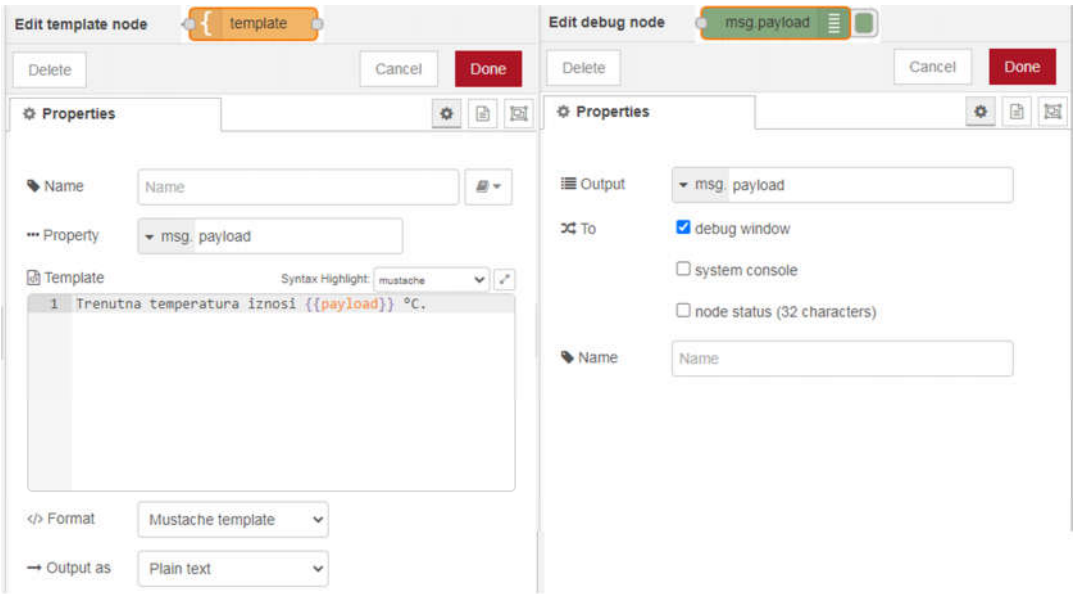
U humidity mqtt in čvoru unutar plavog okvira treba postaviti odgovarajuću temu kao i naziv čvora. Ostali parametri trebaju biti jednaki.



Slika 4.9. Postavke mqtt in čvora koji prihvaća poruke s temom room/temperature

Na mosca in čvorove u obje grane nastavlja se template čvor a na njega debug. Oni pripadaju osnovnim čvorovima i već su spomenuti u 2. Poglavlju. Pomoću template u ovom slučaju se opisuje se poruka i dodjeljuje joj se mjerna jedinica. Debug čvor prihvaća uređenu poruku i ispisuje u desnom bočnom dijelu prozora.

Na slici 4.10 prikazane su postavke čvorova koji se unutar koda nalaze nakon mqtt in čvora temperature.

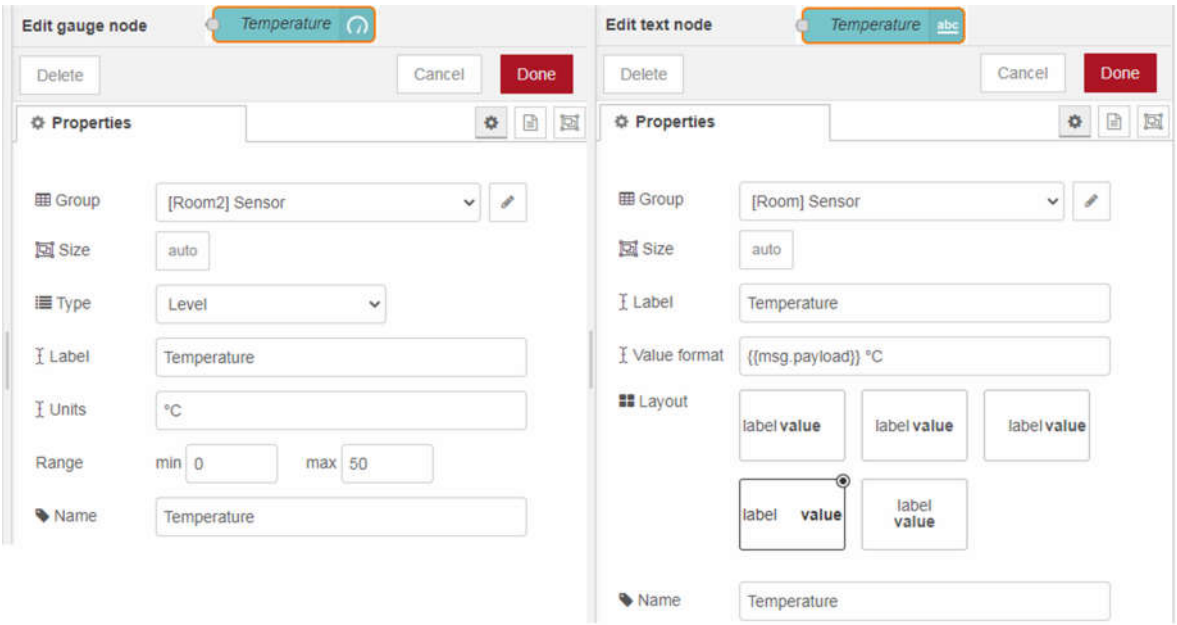


Slika 4.10. Postavke template i debug čvora

Za prikaz vrijednosti u Node-RED može se koristiti dashboard koji se ispunjava widget-ima. Postoji više vrsta widget-a koji prikazuju vrijednosti na različite načine.

Pri korištenju dashboard-a potrebno je urediti postavke stranice na kojoj će biti prikazane vrijednosti. Klikom na dashboard ikonu koja se nalazi u desnom bočnom dijelu prozora pojavit će se 3 kartice. U kartici koja se naziva layout klikom na tab stvorit će se nova kartica čiji se naziv može promijeniti odabirom edit-a. Klikom na group dodaje se nova grupa. Nakon kreirane grupe može se dashboard prikaz dodatno urediti u karticama pod nazivom site i theme. U ovom radu kartice nose naziv Room i Room2, dok je naziv grupe Sensor u obje kartice.

Na slici 4.11 prikazani su text i gauge čvorovi koji se nalaze u dashboard skupini čvorova. Na svaki mqtt in network čvor dodan je text i gauge čvor. Za prikaz podataka obavezno je odabrati grupu. Label se odnosi na naziv koji će se nalaziti u prikazu i opisuje svrhu vrijednosti. Unutar value format nalazi se {{msg.payload}}, jer je to dio poruke koji se želi prikazati. Ako je potrebna mjerna jedinica i slični dodatci oni se mogu dodati odmah nakon, pa će taj parametar izgledati npr.{{msg.payload}} °C. U layout dijelu se odabire pozicija u koju će se smjestiti widget prilikom prikaza na dashboard stranici.



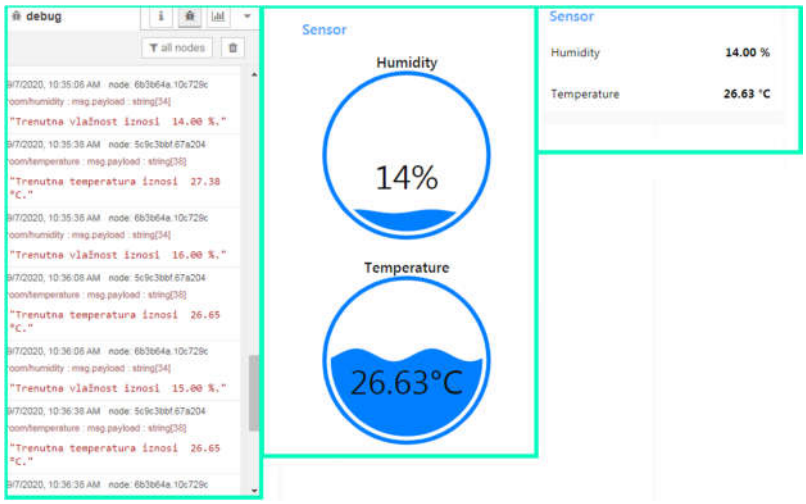
Slika 4.11. Prikaz text i gauge čvora s postavkama

Gauge čvor predstavlja widget veoma lijepog dizajna. Za njega je također potrebno odabrati grupu. Postoji više tipova za prikaz vrijednosti: Gauge, Donat, Compass i Level, a za rad je odabran Level. Label se kao i u prethodnom čvoru odnosi na naziv odnosno opis vrijednosti koja će se prikazivati. Unit omogućuje postavljanje mjerne jedinice. Obavezno je postaviti opseg (engl. *range*) kako bi dobili što bolji vizualni doživljaj. Opseg se postavlja prema opsegu vrijednosti koje može mjeriti senzor. Opseg čvora koji prikazuje vrijednost temperature postavljen je 0 - 50, a čvora koji prikazuje vlažnost od 0 - 100.

Nakon završenih postavki na svim čvorovima na dashboard stranici mogu se pogledati widgeti. U dashboard kartici na desnoj strani pored Layout, Site i Theme nalazi se ikona s okvirom i strelicom. Klikom na ikonu otvara se dashboard stranica u novoj kartici. Stranicu se može otvoriti i dodavanjem nastavka /ui na URL pomoću kojeg je otvoren Node-RED. Stranica prikazuje widgete, a nakon pokretanja koda i uspostavljanjem svih veza prikazuje vrijednosti poruka unutar widget-a.

Klikom na deploy spremaju se umetnuti čvorovi i postavljene veze. Pokretanjem mikrokontrolera na ploči u Node-RED svakih 30 sekundi, kao što je postavljeno u Arduino IDE kodu, uzimaju se vrijednosti sa senzora, dodjeljuju im se teme i šalju se na mqtt posrednik. Na izlazu iz posrednika poruke se dijele prema temama i odlaze u čvorove koji ih ispisuju uz opis i mjernu jedinicu.

Slika 4.12 prikazuje ispis primljenih vrijednosti u debug kartici i dashboard-u nakon pokretanja koda.



Slika 4.12. Primljene poruke u debug-u i dashboard-u Node-RED platforme

Cjeloviti prikaz sustava u drugom koraku implementacije prikazan je na slici 4.13. Plava strelica označava kabelsku vezu između Arduino IDE programa i ploče, a crvena strelica WiFi vezu između ploče i mosca posrednika koji je pokrenut u Node-RED platformi.

Slika 4.13. Prikaz sustava pri Node-RED implementaciji

4.3.3. ThingSpeak Implementacija

Da bi vrijednosti senzora bile dostupne svima na internetu nije potrebno mijenjati Arduino IDE programski kod koji je utisnut u ploču prilikom prethodne implementacije. Potrebno je u Node-RED dodati čvorove koji omogućuju vezu s ThingSpeak aplikacijom te u ThingSpeak aplikaciji kreirati kanale.

Budući da senzor mjeri dva parametra potrebno je stvoriti dva kanala. Prijavom na službenu stranicu mogu se kreirati kanali odabirom Channels→My Channels→Channel putanje. U otvorenom prozoru postavljaju se osnovni parametri, a zatim spremaju. U My Channels je vidljiv novonastali kanal. Ispod njegovog naziva vidljivo je nekoliko kartica koje omogućuju dodatne postavke kanala.

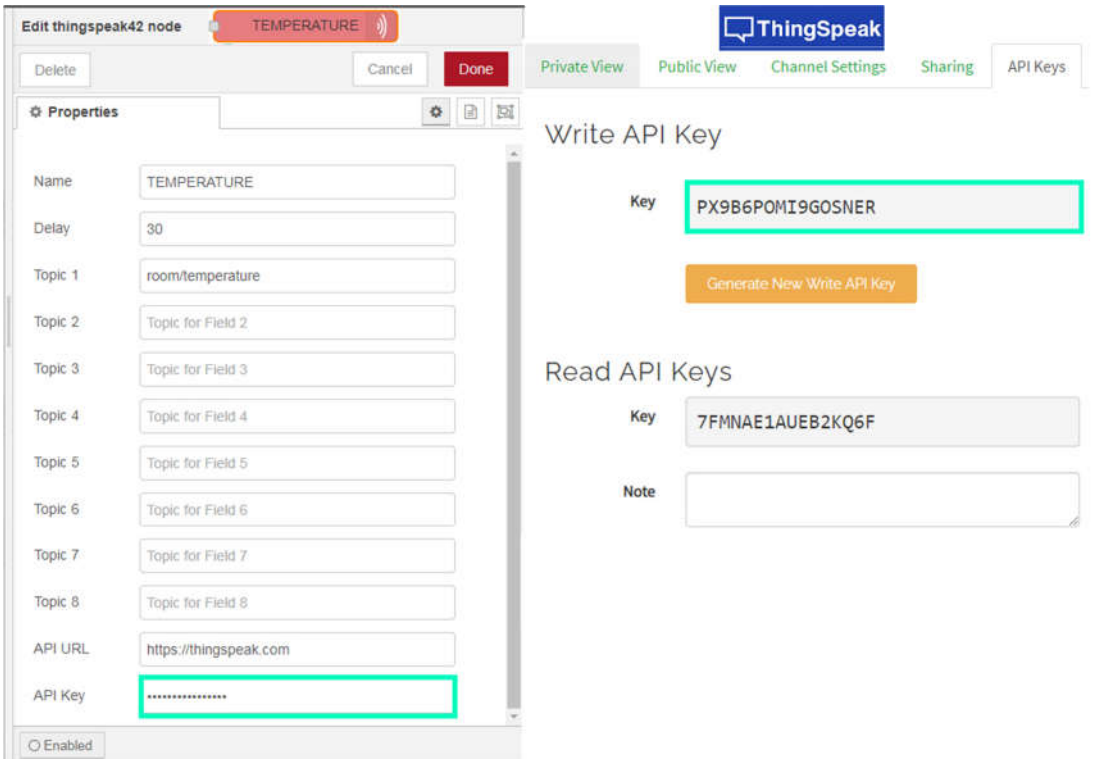
Kanali za ovaj rad trebaju biti dostupni svima na internetu, stoga je u kartici dijeljenje (engl. *Sharing*) važno odabrati dijeljenje kanala sa svima (engl. *Share channel view with everyone*). Kada je kanal javan, public view kartica je dostupna. Unutar te kartice odabire

se način na koji će biti prikazane vrijednosti. Za ovaj rad odabran je grafički prikaz. Podešen je da prikazuje posljednjih 10 primljenih vrijednosti i da označene vrijednosti u grafu povezuje linijom.

U Node-RED potrebno je instalirati novi čvor pod nazivom node-red-contrib-thingspeak42. Nakon instalacije čvor se nalazi u function grupi pod nazivom thingspeak42. Svakom mqtt in čvoru potrebno je pridružiti po jedan thingspeak42 čvor.

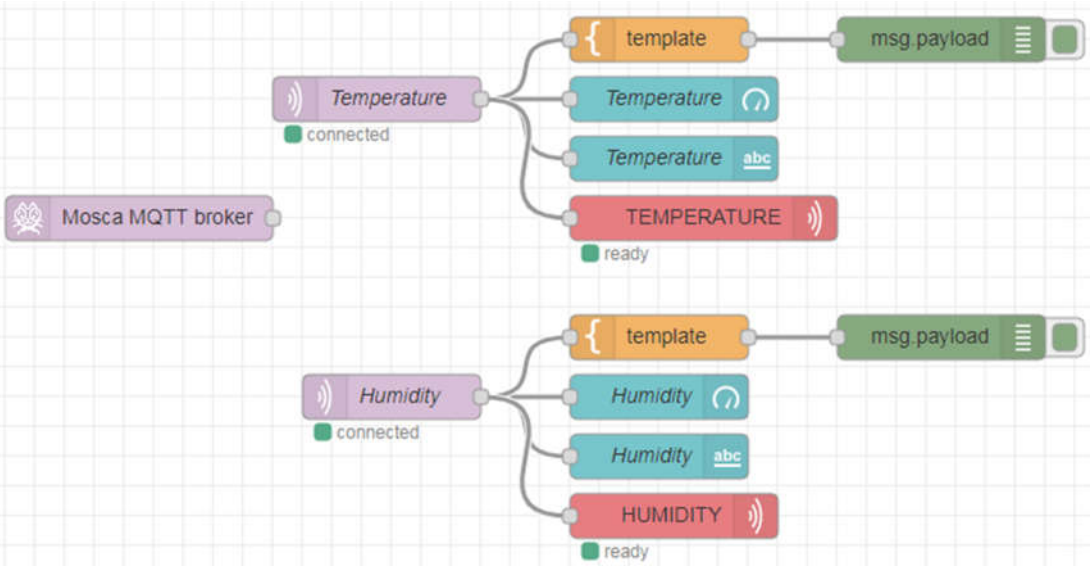
Novim čvorovima u Node-RED potrebno je postaviti parametre. Najbitniji parametri su: tema koja odgovara temi mqtt in čvora na koji je povezan, API URL koji se odnosi na URL ThingSpeak stranice i API ključ. API ključ se nalazi istoimenoj kartici kanala u ThingSpeak aplikaciji. Svaki kanal posjeduje dva API ključa. U ovom slučaju potrebno je odabrati onaj za pisanje i kopirati ga u thingspeak42 čvor.

Na slici 4.14 prikazan je thingspeak42 čvor koji prenosi vrijednost temperature i API ključ koji čini jednu od kartica u ThingSpeak.



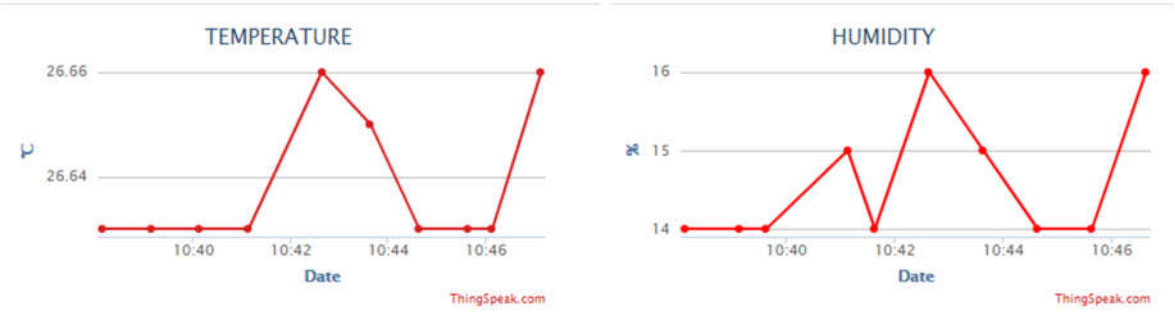
Slika 4.14. Postavke thingspeak42 čvora i APi ključeva ThingSpeak platforme

Nakon postavljenih thingspeak42 čvorova Node-RED prozor izgleda kao na slici 4.15.



Slika 4.15. Prikaz čvorova i veza u Node-RED aplikaciji prilikom ThingSpeak implementacije

Pri pokretanju koda izmjerene vrijednosti senzora stižu u kanale ThingSpeak aplikacije, filtrirane prema temama. Svaki kanal prikazuje vrijednosti koji pripadaju jednoj temi i ima vlastiti link na kojem objavljuje vrijednosti. Prikaz objavljenih vrijednosti u ThingSpeak platformi nalazi se na slici 4.16.



Slika 4.16. Prikaz temperature i vlažnosti javnim kanalima ThingSpeak-a

Cjeloviti prikaz sustava u trećem koraku implementacije prikazan je na slici 4.17. Plava strelica označava kabelsku vezu između Arduino IDE programa i ploče, a crvena strelica

WiFi vezu između ploče i mosca posrednika koji je pokrenut u Node-RED platformi. Iz mosca posrednika filtriraju se 2 teme koje imaju zasebne kanale u ThingSpeak platformi.

Slika 4.17. Prikaz sustava za ThingSpeak implementaciju

Sva tri koraka implementacije integrirana su u jednu cjelinu. U Arduino IDE i Node-RED platformi nalazi se po jedan programski kod koji objedinjuje zahtjeve za sve implementacije. Arduino IDE programski kod koji se ugrađuje u ploču nalazi se u prilozima kao i Node-RED programski kod.

5. ZAKLJUČAK

MQTT protokol obično se koristi za IoT aplikacije. Funkcionira na objavi/pretplati protokolu. Sustav je sastavljen od klijenta i posrednika. Klijenti objavljivači šalju poruke s određenim temama posredniku, a klijenti pretplatnici se pretplaćuju na teme koje ih zanimaju. Sve poruke prvo stižu posredniku i svaka veza u ovom protokolu je između klijenta i posrednika. Protokol podržava prijenos više tipova poruka koje su male duljine i ne opterećuju mrežu koju koriste za prijenos podataka.

U praktičnom dijelu rada demonstriran je prijenos vrijednosti DHT11 senzora koji se upravlja mikrokontrolerom ESP32 ploče. Mikrokontroler usmjerava vrijednosti senzora prema platformama. Izmjereni podatci su tako vidljivi u Arduino IDE programu unutar kojeg je napisan programski kod za mikrokontroler. WiFi modul šalje MQTT poruke s vrijednostima senzora na MQTT posrednik. Posrednik filtrira poruke i usmjerava prema klijentima Node-RED aplikacije koji objavljuju vrijednosti u dashboardu i debug bočnom prozoru. Ove vrijednosti su vidljive samo u lokalnoj mreži. Za objavljivanje vrijednosti na internet, na MQTT posrednik su povezani čvorovi koji usmjeravaju podatke prema temama na ThingSpeak platformu koja objavljuje primljene vrijednosti. Vrijednosti senzora tako su vidljivesvima na internetu.

LITERATURA

[1] https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/ [30.8.2020.]

[2] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718021 [30.8.2020.]

[3]https://learn.adafruit.com/dht [30.8.2020.]

[4] http://www.steves-internet-guide.com/node-red-overview/ [30.8.2020.]

[5] https://randomnerdtutorials.com/esp32-troubleshooting-guide/ [30.8.2020.]

[6] http://www.esp32learning.com/code/use-a-esp32-to-display-dht11-readings-on-a-web-page.php [30.8.2020.]

[7] https://nodered.org/docs/getting-started/windows [30.8.2020.]

[8]https://flows.nodered.org/node/node-red-dashboard [30.8.2020.]

[9] http://www.steves-internet-guide.com/install-Mosca-mqtt-broker-node-red/ [30.8.2020.]

[10] https://learn.adafruit.com/dht [30.8.2020.]

[11] https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf [30.8.2020.]

[12] https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/ [30.8.2020.]

[13] https://en.wikipedia.org/wiki/ESP32 [30.8.2020.]

[14] http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ [30.8.2020.]

[15] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718025 [30.8.2020.]

[16] https://medium.com/@emqtt/introduction-to-mqtt-5-0-protocol-qos-quality-of-service-e6d9b0aaf9fb [30.8.2020.]

[17] https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105 [30.8.2020.]

[18] https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/ [30.8.2020.]

[19] https://hivemq.github.io/hivemq-mqtt-client/docs/mqtt-operations/publish/#topic [30.8.2020.]

[20] https://docs.solace.com/Open-APIs-Protocols/MQTT/MQTT-Topics.htm [30.8.2020.]

[21] https://www.emqx.io/blog/mqtt5-features-retain-message [30.8.2020.]

[22] https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/ [30.8.2020.]

[23] https://www.embedded.com/mqtt-essentials-clients-servers-and-connections/ [30.8.2020.]

https://fiware-tutorials.readthedocs.io/en/latest/iot-over-mqtt/index.html [30.8.2020.]

[24] https://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT_311_Prtl_Conformance_Spec.htm [30.8.2020.]

[25] https://www.ietf.org/proceedings/103/slides/slides-103-maprg-evaluating-the-performance-of-coap-mqtt-and-http-in-vehicular-scenarios-jaime-jimenez-00 [30.8.2020.]

[26] https://www.geeksforgeeks.org/difference-between-mqtt-and-http-protocols/ [30.8.2020.]

[27] https://whttps://www.journaldev.com/7397/introduction-to-node-js-basics#nodejsww.paessler.com/it-explained/mqtt [30.8.2020.]

[28] https://nodered.org/docs/user-guide/nodes#debug [30.8.2020.]

[29] https://en.wikipedia.org/wiki/MQTT [30.8.2020.]

[30] https://en.wikipedia.org/wiki/ThingSpeak [30.8.2020.]

[31] https://www.mediabuzz.com.sg/research-analysis-trends-mar2018/communication-models-in-iot [30.8.2020.]

[32] http://www.inetservicescloud.com/the-four-internet-of-things-connectivity-models-explained/ [30.8.2020.]

[33] https://en.wikipedia.org/wiki/Node.js [30.8.2020.]

[34] http://mef-lab.com/praktikum-2017/book/Node.html#moduli [30.8.2020.]

[35] http://blog.king-ict.hr/hello-nodejs [30.8.2020.]

[36] https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/ [30.8.2020.]

[37] https://en.wikipedia.org/wiki/ThingSpeak [30.8.2020.]

[38] https://e-radionica.com/hr/blog/2015/08/19/arduino-ide-i-prvi-kod/ [30.8.2020.]

POPIS SLIKA

Slika 2.1. Početni Arduino IDE prozor 7
Slika 2.2. Provjera Node.js i NPM verzije u cmd prozoru 8
Slika 2.3. Pokretanje Node-RED programa iz cmd 9
Slika 2.4. Preuzimanje čvorova za Node-RED platformu 10
Slika 2.5. Prikaz stvorenog kanala u ThingSpeak platformi 14
Slika 3.1. Prikaz toka PUBLISH poruke kojoj je RETAIN zastavica postavljena u 1 20
Slika 3.2. Primjer toka PUBLISH poruke s kvalitetom usluge 0 27
Slika 3.3. Primjer toka PUBLISH poruke s kvalitetom usluge 1 28
Slika 3.4. Primjer toka PUBLISH poruke s kvalitetom usluge 2 29
Slika 3.5. Primjer toka poruka između klijenta i posrednika korištenjem MQTT protokola 31
Slika 4.1. DHT11 modul i senzor s označenim pinovima 34
Slika 4.2. Dodavanje URL-a za ESP32 u Arduino IDE 37
Slika 4.3. Instalacija biblioteke ESP32 38
Slika 4.4. Prikaz Device Manager prozora s uključenim i isključenim priključkom kojim se ostvaruje veza s pločom 39
Slika 4.5. Serial monitor nakon pokretanja Arduino IDE koda na mikrokontroleru ploče 40
Slika 4.6. Izgled sustava prilikom Arduino IDE implementacije 41
Slika 4.7. Izgled čvorova i njihovih veza prilikom Node-RED implementacije 43
Slika 4.8. Postavke mosca in čvora 44
Slika 4.9. Postavke mqtt in čvora koji prihvaća poruke s temom room/temperature 45
Slika 4.10. Postavke template i debug čvora 45
Slika 4.11. Prikaz text i gauge čvora s postavkama 46
Slika 4.12. Primljene poruke u debug-u i dashboard-u Node-RED platforme 47
Slika 4.13. Prikaz sustava pri Node-RED implementaciji 48
Slika 4.14. Postavke thingspeak42 čvora i APi ključeva ThingSpeak platforme 49
Slika 4.15. Prikaz čvorova i veza u Node-RED aplikaciji prilikom ThingSpeak implementacije 50
Slika 4.16. Prikaz temperature i vlažnosti javnim kanalima ThingSpeak-a 50
Slika 4.17. Prikaz sustava za ThingSpeak implementaciju 51

POPIS TABLICA

Tablica 3.1. ISO/OSI slojevi MQTT protokola 15
Tablica 3.2. Fiksno zaglavlje MQTT poruke 17
Tablica 3.3. Vrijednosti koje tip kontrolnog paketa može imati s objašnjenjem 17
Tablica 3.4. Zastavice fiksnog zaglavlja MQTT protokola 18
Tablica 3.5. Vrijednosti QoS zastavice s opisom za MQTT protokol 19
Tablica 3.6. Zastavice varijabilnog zaglavlja CONNECT poruke 21
Tablica 3.7. Moguće vrijednosti povratnog koda CONNECT poruke s opisom 23
Tablica 3.8. Prikaz povratnih kodova u SUBACK porucis opisom 25
Tablica 3.9.Primjeri tema koje može zamijeniti znak + i koje ne 32
Tablica 3.10. Primjeri tema koje može zamijeniti znak # i koje ne 32
Tablica 4.1. Pinovi DHT11 senzora s opisom 35
Tablica 4.2. Pinovi DHT11 modula s opisom 35

PRILOZI

Arduino IDE programski kod s objašnjenjem

Potrebne su 3 biblioteke. WiFi biblioteka koja omogućuje povezivanje WiFi modula na mrežu. PubSubClient biblioteka omogućuje MQTT vezu, a DHT biblioteka čitanje vrijednosti senzora DHT11.

```
#include <WiFi.h>
```

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

```
#include "DHT.h"
```

Postoje tri vrste DHT senzora (DHT11, DHT22 i DHT21). Stoga je u ovoj liniji koda definiran tip senzora koji se koristi. U ovom slučaju je DHT11.

```
#define DHTTYPE DHT11
```

Pin (engl. *data pin*) na ploči ESP32 kojim se ostvaruje podatkovna veza između senzora i ESP32 također je potrebno inicijalizirati.

```
uint8_t DHTPin = 4;
```

Dvije pomoćne vremenske varijable koje će omogućiti da se mjerenja vrše nakon određenog vremenskog perioda.

```
long now = millis();
```

```
long lastMeasure = 0;
```

Inicijalizacija DHT senzora (objektno orijentirano programiranje).

```
DHT dht(DHTPin, DHTTYPE);
```

Ssid predstavlja ime mreže i upisuje se unutar navodnika prema vlastitim vrijednostima mreže. Password se odnosi na zaporku mreže koje je također potrebno navesti unutar navodnika.

```
const char* ssid = "Marijana";
```

```
const char* password = "21112111";
```

MQTT_server pokazivač odnosi se na IP adresu na kojoj je pokrenut MQTT server. IP adresa ukoliko je postavljena kao localhost potrebno ju je pronaći. Najjednostavnije je u cmd unijeti ipconfig. Obično postoje dvije IPv4 adrese. Ukoliko je računalo povezano na usmjerivač (engl. *router*) žično tj. Ethernet kabelom tražena adresa se nalazi u dijelu koji se naziva „*ethernet adapter Local area Connection*“. Ukoliko je veza računala sa usmjerivačem ostvarena bežično tada se tražena adresa nalazi u dijelu podnazivom„*wireless lan adapter wireless network connection*“.

```
const char* MQTT_server = "192.168.43.163";
```

MQTT_port odnosi se na proizvoljni port koji je korišten za MoscaMQTT server.

```
int MQTT_port = 1883;
```

Inicijalizacija ESP32 modula kao klijenta.

```
WiFiClient espClient;
```

```
PubSubClient client(espClient);
```

Dvije float varijable u koje će se pohranjivati trenutne vrijednosti temperature i vlažnosti.

```
float Temperature;
```

```
float Humidity;
```

U idućoj funkciji DHT senzor započinje sa radom i dodjeljuju se prethodno inicijalizirane vrijednosti servera i ispisuju se u Serial Monitoru. Također se ispisuje IP adresa WiFi modula koji se nalazi na ESP32.

```
void setup() {
Serial.begin(9600);
DHT.begin();
setup_wifi();
client.setServer(MQTT_server, MQTT_port);
Serial.println("Connected ");
Serial.print("MQTT Server ");
Serial.print(MQTT_server);
Serial.print(":");
Serial.println(String(MQTT_port));
Serial.print("ESP32 IP ");
Serial.println(WiFi.localIP());
}
```

Sljedećom funkcijom ostvaruje se povezivanje WiFi modula na pristupnu točku, prema prethodno navedenim podatcima. U Serial Monitoru će se ispisati IP adresa WiFi modula koji se nalazi na ESP32.

```
void setup_wifi() {
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("Connecting to WiFi..");
   }
```

```
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
```

Prilikom izvršavanja programa ukoliko nije ostvarena veza sa klijentom program će ući u ovu funkciju i pokušavati ostvariti vezu sa klijentom svakih 5 sekundi dok ne uspije.

```
void reconnect() {
Loop until we're reconnected
while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect()) {
          Serial.println("connected");
       }
    else {
          Serial.print("failed, rc=");
    Serial.print(client.state());
          Serial.println(" try again in 5 seconds");
          delay(5000);
       }
   }
}
```

Glavna petlja Arduino ide.

```
void loop() {
```

Naredna petlja omogućava da ukoliko klijent nije povezan da se pokrene funkcija reconnect kako bi se ponovo pokušala ostvariti veza.

```
if (!client.connected()) {
    reconnect();
   }
```

```
if(!client.loop())
    client.connect();
now = millis();
```

Objavljivanje vrijednosti vlage i topline sa senzora svakih 30 sekundi ostvaruje se pomoću if uvjeta.

```
if (now - lastMeasure > 30000) {
    lastMeasure = now;
```

Funkcijama readHumidity() i readTemperature() se čitaju vrijednosti temperature i vlage na senzoru i spremaju u varijable h i t.

```
    float h = DHT.readHumidity();
    float t = DHT.readTemperature();
```

Iduće linije koda služe ukoliko DHT senzor nije ispravno učitao vrijednosti temperature i vlažnosti da ispiše da je došlo do greške.

```
    if (isnan(h) || isnan(t)) {
          Serial.println("Failed to read from DHT sensor!");
          return;
```

```
        }
```

Pretvaranje vrijednosti temperature u stupnjeve Celzijusa vrši se narednim naredbama.

```
    float hic = DHT.computeHeatIndex(t, h, false);
    static char temperatureTemp[7];
    dtostrf(hic, 6, 2, temperatureTemp);
```

Pretvaranje vrijednosti vlažnosti u postotke vrši se sljedećim naredbama.

```
    static char humidityTemp[7];
    dtostrf(h, 6, 2, humidityTemp);
```

Vrijednostima temperature i vlage pridružuju se nazivi tema. Teme mogu biti jednake ako svi klijenti koji primaju podatke o vrijednostima senzora su jednako zainteresirani za oba podatka. U radu su odabrane dvije različite teme iako isti klijent primatelj prima oba podatka. Međutim ukoliko bi dodali još klijenata primatelja jednostavno bi im mogli dati pristup samo temperaturi ili vlažnosti.

```
    client.publish("room/temperature", temperatureTemp);
    client.publish("room/humidity", humidityTemp);
```

Ispisivanje vrijednosti vlage i temperature u Serial Monitoru Arduino IDE. Da bi se mogle usporediti vrijednosti sa onima Dashboard-u Node-RED-a.

```
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t Temperature: ");
    Serial.print(t);
    Serial.print(" °C ");
    Serial.print('\n');
```

```
    }
}
```

Node-RED programski kod

Kod se odnosi na sliku 4.15 iz rada.

```
[
    {
        "id": "49424c8f.951274",
        "type": "tab",
        "label": "dht11",
        "disabled": false,
        "info": ""
    },
    {
        "id": "fed46ff2.cb05",
        "type": "mqtt in",
        "z": "49424c8f.951274",
        "name": "Temperature",
        "topic": "room/temperature",
        "qos": "2",
        "datatype": "auto",
        "broker": "8e369e08.12fdc8",
        "x": 270,
        "y": 360,
```

```
        "wires": [
            [
                "51aaac78.ec3664",
                "40b452f4.f1cb0c",
                "ea43814f.6b97b",
                "9dd98bc6.df3f88"
            ]
        ]
    },
    {
        "id": "72ff518b.5f52b",
        "type": "ui_gauge",
        "z": "49424c8f.951274",
        "name": "Humidity",
        "group": "5c9b527a.cf68e4",
        "order": 1,
        "width": 0,
        "height": 0,
        "gtype": "wave",
        "title": "Humidity",
        "label": "%",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
```

```
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 460,
        "y": 560,
        "wires": []
    },
    {
        "id": "50c74bba.87a044",
        "type": "mqtt in",
        "z": "49424c8f.951274",
        "name": "Humidity",
        "topic": "room/humidity",
        "qos": "2",
        "datatype": "auto",
        "broker": "8e369e08.12fdc8",
        "x": 280,
        "y": 560,
        "wires": [
            [
                "72ff518b.5f52b",
                "6b07dc7b.7b0d84",
                "e891f355.cd96b",
                "d869687.a5e0798"
```

```
            ]
        ]
    },
    {
        "id": "6b3b64a.10c729c",
        "type": "debug",
        "z": "49424c8f.951274",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 650,
        "y": 520,
        "wires": []
    },
    {
        "id": "6b07dc7b.7b0d84",
        "type": "ui_text",
        "z": "49424c8f.951274",
        "group": "7e2fb6ea.bffe1",
        "order": 0,
        "width": 0,
        "height": 0,
```

```
        "name": "Humidity",
        "label": "Humidity",
        "format": "{{msg.payload}} %",
        "layout": "row-spread",
        "x": 460,
        "y": 600,
        "wires": []
    },
    {
        "id": "51aaac78.ec3664",
        "type": "ui_text",
        "z": "49424c8f.951274",
        "group": "7e2fb6ea.bffe1",
        "order": 0,
        "width": 0,
        "height": 0,
        "name": "Temperature",
        "label": "Temperature",
        "format": "{{msg.payload}} Â°C",
        "layout": "row-spread",
        "x": 470,
        "y": 400,
        "wires": []
    },
    {
        "id": "40b452f4.f1cb0c",
```

```
        "type": "ui_gauge",
        "z": "49424c8f.951274",
        "name": "Temperature",
        "group": "5c9b527a.cf68e4",
        "order": 1,
        "width": 0,
        "height": 0,
        "gtype": "wave",
        "title": "Temperature",
        "label": "Â°C",
        "format": "{{value}}",
        "min": 0,
        "max": "50",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 470,
        "y": 360,
        "wires": []
    },
    {
        "id": "c34a37a4.e63928",
```

```
        "type": "mosca in",
        "z": "49424c8f.951274",
        "mqtt_port": 1883,
        "mqtt_ws_port": 8080,
        "name": "",
        "username": "",
        "password": "",
        "dburl": "",
        "x": 110,
        "y": 440,
        "wires": [
            []
        ]
    },
    {
        "id": "5c9c3bbf.67a204",
        "type": "debug",
        "z": "49424c8f.951274",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 650,
```

```
        "y": 320,
        "wires": []
    },
    {
        "id": "ea43814f.6b97b",
        "type": "template",
        "z": "49424c8f.951274",
        "name": "",
        "field": "payload",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "Trenutna temperatura iznosi {{payload}}
Â°C.",
        "output": "str",
        "x": 460,
        "y": 320,
        "wires": [
            [
                "5c9c3bbf.67a204"
            ]
        ]
    },
    {
        "id": "e891f355.cd96b",
        "type": "template",
```

```
        "z": "49424c8f.951274",
        "name": "",
        "field": "payload",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "Trenutna vlaĹľnost iznosi {{payload}}
%.",
        "output": "str",
        "x": 460,
        "y": 520,
        "wires": [
            [
                "6b3b64a.10c729c"
            ]
        ]
    },
    {
        "id": "9dd98bc6.df3f88",
        "type": "thingspeak42",
        "z": "49424c8f.951274",
        "name": "TEMPERATURE",
        "delay": "30",
        "topic1": "room/temperature",
        "topic2": "",
        "topic3": "",
```

```
        "topic4": "",
        "topic5": "",
        "topic6": "",
        "topic7": "",
        "topic8": "",
        "endpoint": "https://thingspeak.com",
        "x": 490,
        "y": 440,
        "wires": []
    },
    {
        "id": "d869687.a5e0798",
        "type": "thingspeak42",
        "z": "49424c8f.951274",
        "name": "HUMIDITY",
        "delay": "30",
        "topic1": "",
        "topic2": "room/humidity",
        "topic3": "",
        "topic4": "",
        "topic5": "",
        "topic6": "",
        "topic7": "",
        "topic8": "",
        "endpoint": "https://thingspeak.com",
        "x": 470,
```

```
        "y": 640,
        "wires": []
    },
    {
        "id": "8e369e08.12fdc8",
        "type": "mqtt-broker",
        "z": "",
        "name": "",
        "broker": "localhost",
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "willTopic": "",
        "willQos": "0",
        "willPayload": ""
    },
    {
```

```
        "id": "5c9b527a.cf68e4",
        "type": "ui_group",
        "z": "",
        "name": "Sensor",
        "tab": "f394212d.be50a8",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "7e2fb6ea.bffe1",
        "type": "ui_group",
        "z": "",
        "name": "Sensor",
        "tab": "808ba514.5bf558",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "f394212d.be50a8",
        "type": "ui_tab",
        "z": "",
        "name": "Room2",
```

```
        "icon": "dashboard",
        "order": 1,
        "disabled": false,
        "hidden": false
    },
    {
        "id": "808ba514.5bf558",
        "type": "ui_tab",
        "z": "",
        "name": "Room",
        "icon": "dashboard",
        "order": 1,
        "disabled": false,
        "hidden": false
    }
]
```