MQTT protokol

MQTT protokol

U današnjem međusobno povezanom svijetu, gdje uređaji neprimjetno komuniciraju kako bi olakšali automatizaciju i razmjenu podataka, razumijevanje MQTT protokola postaje sve vrednije.

Bez obzira da li ste programer koji se bavi IoT projektima ili ste jednostavno radoznali kako uređaji razgovaraju jedni sa drugima preko mreža, ovaj vodič će vas provesti kroz osnove MQTT-a, njegove ključne koncepte i njegove praktične primjene.

MQTT protokol

- Što je MQTT?
- Zašto je MQTT veoma popularan za IoT?
- Kako MQTT radi?
- MQTT radni tok.
- Početak rada sa MQTT: Brzi vodič.

Što je **MQTT?**

MQTT (Message Queuing Telemetry Transport) je lagan protokol za razmjenu poruka zasnovan na objavljivanju i pretplati.

Dizajniran je za uređaje sa ograničenim resursima, niske propusnosti, velikih kašnjenja ili nepouzdane mreže.

Široko se koristi u aplikacijama Interneta stvari (IoT), obezbjeđujući efikasnu komunikaciju između senzora, aktuatora i drugih uređaja.

Zašto je **MQTT** veoma popularan za IoT?

MQTT važi za jedan od najboljih IoT protokola zbog svojih jedinstvenih osobina i mogućnosti, prilagođenih specifičnim potrebama IoT sistema.

Neki od ključnih razloga uključuju:

- **Lagan:** IoT uređaji često su ograničeni u smislu procesorske snage, memorije i potrošnje energije. MQTT-ovo minimalno opterećenje i mala veličina paketa čine ga idealnim za ove uređaje, budući da troši manje resursa, omogućujući učinkovitu komunikaciju čak i sa/između uređaja s ograničenim mogućnostima.
- **Pouzdan:** IoT mreže mogu imati visoke latencije ili nestabilne veze. MQTT osigurava pouzdanu isporuku poruka čak i u izazovnim uslovima, što ga čini prikladnim za IoT aplikacije.

- **Sigurnost komunikacije**:
  - **Sigurnost**: Sigurnost je ključna u IoT mrežama, jer često prenose osjetljive podatke. MQTT podržava Transport Layer Security (TLS) i Secure Sockets Layer (SSL) enkripciju, osiguravajući povjerljivost podataka tokom prenosa. Dodatno, pruža mehanizme provjere autentičnosti i autorizacije putem akreditiva korisničkog imena/lozinke ili certifikata klijenta, štiteći pristup mreži i njenim resursima.
  - **Dvosmjernost**: MQTT-ov model objavljivanja i pretplate omogućuje besprekornu dvosmjernu komunikaciju između uređaja. Klijenti mogu objavljivati poruke na teme i pretplatiti se na primanje poruka o određenim temama, omogućujući učinkovitu razmjenu podataka u različitim IoT sistemima bez direktnog povezivanja između uređaja. Ovaj model također pojednostavljuje integraciju novih uređaja, osiguravajući jednostavnu skalabilnost.

Zašto je **MQTT** veoma popularan za IoT?

- **Sigurnost komunikacije**:
  - **Kontinualne sesije s praćenjem stanja**: MQTT omogućuje klijentima održavanje sesija s praćenjem stanja s brokerom, omogućujući sistemu da pamti pretplate i neisporučene poruke čak i nakon prekida veze. Klijenti također mogu, tokom povezivanja, odrediti interval održavanja, što od brokera traži da povremeno provjerava status veze. Ako se veza izgubi, broker pohranjuje neisporučene poruke (ovisno o QoS nivou) i pokušava ih isporučiti kada se klijent ponovno poveže. Ova osobina osigurava pouzdanu komunikaciju i smanjuje rizik od gubitka podataka.
  - **Podrška za IoT s velikim brojem uređaja**: IoT sistemi često uključuju veliki broj uređaja, zahtijevajući protokol koji može podnijeti implementacije velikih razmjera. MQTT-ova laganost, niska potrošnja propusnog opsega i efikasno korištenje resursa, čine ga prikladnim za velike IoT aplikacije. Obrazac objavljivanje-pretplata omogućuje efikasno skaliranje MQTT-a, jer odvaja pošiljaoca i primaoca, smanjujući mrežni saobraćaj i korištenje resursa.
  - **Jezična podrška**: IoT sistemi često uključuju uređaje i aplikacije razvijene pomoću različitih programskih jezika. MQTT-ova široka jezična podrška omogućuje jednostavnu integraciju s više platformi i tehnologija, podsstičući besprekornu komunikaciju i interoperabilnost u različitim IoT sistemima. MQTT se može koristiti u C++-u, PHP-u, Node.js, Python, Golang i drugim programskim jezicima.

Kako **MQTT** radi?

Da biste razumjeli kako MQTT radi, prvo se morate upoznati s konceptima:

- MQTT klijent,
- MQTT broker,
- metoda objave-pretplate,
- teme i
- QoS.

Kako **MQTT** radi?

MQTT klijent:

Svaka aplikacija ili uređaj koji pokreće biblioteku MQTT klijenta je MQTT klijent.

Na primjer:

- aplikacija za razmjenu trenutnih poruka, koja koristi MQTT, je klijent,
- različiti senzori koji koriste MQTT za dojavu podataka su klijent,
- razni alati za testiranje MQTT također su klijent.

Kako **MQTT** radi?

MQTT broker:

MQTT Broker upravlja klijentskim vezama, prekidom, zahtevima za pretplatu i otkazivanje pretplate i porukama za rutiranje.

Moćan MQTT broker može da podrži brojne veze i protok poruka na milionskom nivou, pomažući provajderima IoT usluga da se fokusiraju na posao i brzo kreiraju pouzdanu MQTT aplikaciju.

10 Free Public MQTT Brokers(Private & Public) - Mntolia.com

Kako **MQTT** radi?

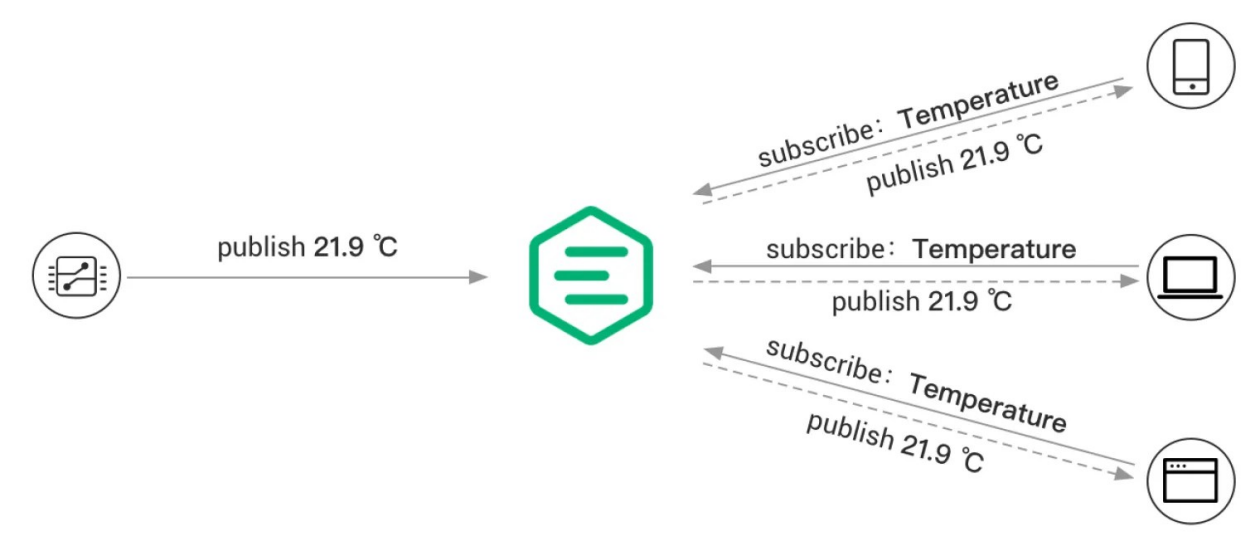
Obrazac objava-pretplata:

Obrazac objava-pretplata se razlikuje od obrasca klijent-server po tome što odvaja klijenta koji šalje poruke (izdavača) od klijenta koji prima poruke (pretplatnika). Izdavači i pretplatnici ne moraju da uspostavljaju direktnu vezu, a MQTT Broker je odgovoran za rutiranje i distribuciju svih poruka.

Kako **MQTT** radi?

Obrazac objava-pretplata:

Dijagram prikazuje MQTT proces objavljivanja/pretplate. Senzor temperature se povezuje sa MQTT serverom kao klijent i objavljuje podatke o temperaturi u temi (npr. Temperatura), a server prima poruku i prosleđuje je klijentu koji je pretplaćen na temu Temperatura.



Kako **MQTT** radi?

Teme:

MQTT protokol usmjerava poruke na osnovu teme. Tema razlikuje hijerarhiju kosom crtom /, što je slično URL putanjama, na primjer:

razgovor/soba/1

senzor/10/temperatura

senzor/+/temperatura

MKTT tema podržava sljedeće džoker znakove: + i #.

+: označava jedan nivo džoker znakova, kao što je a/+ koji odgovara a/k ili a/i.

#: označava više nivoa džoker znakova, kao što su a/# koji odgovaraju a/k, a/b/c/d.

Kako **MQTT** radi?

Quality of Service (QoS):

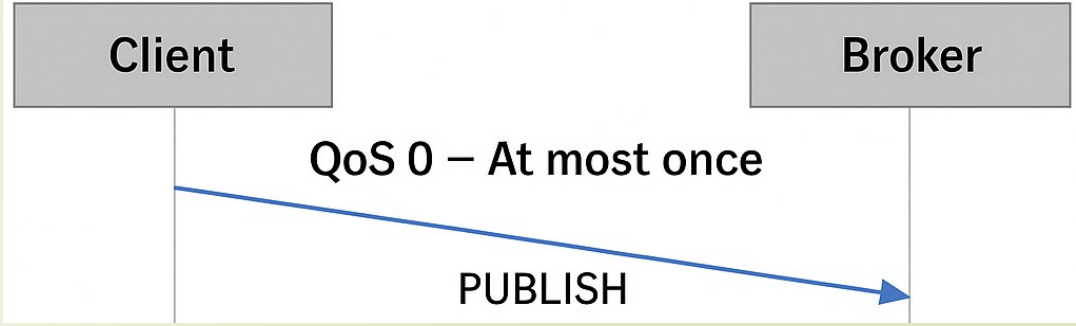
MKTT pruža tri vrste kvaliteta usluge i garantuje pouzdanost razmjene poruka u različitim mrežnim okruženjima.

QoS 0: Poruka se isporučuje najviše jednom. Ako klijent trenutno nije dostupan, izgubiće ovu poruku.

QoS 1: Poruka se isporučuje najmanje jednom.

QoS 2: Poruka se isporučuje samo jednom.

Kako **MQTT** radi?



QoS 0 – “At most once” (Najviše jednom)

Opis:

Poruka se šalje bez potvrde i bez ponovnog slanja.

Isporuka poruke nije zagarantovana – može se izgubiti ako klijent ili mreža zakažu.

Broker ne potvrđuje prijem poruke.

Korišćenje:

Kada je gubitak poruke prihvatljiv.

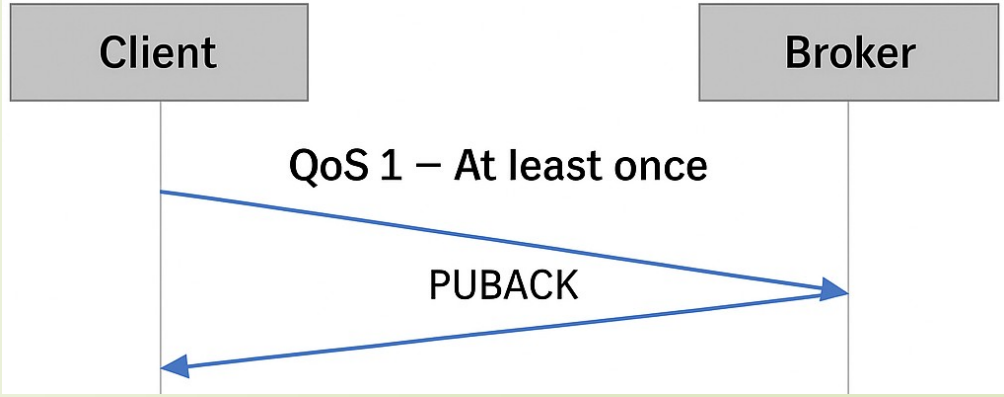
Primer: Temperaturni senzor koji šalje podatke svakih 10 sekundi – ako se jedan uzorak izgubi, nije kritično.

Prednosti:

Najmanje opterećenje na mrežu.

Najniža latencija.

Kako **MQTT** radi?



QoS 1 – “At least once” (Najmanje jednom)

Opis:

Poruka se šalje i mora biti potvrđena od strane brokera.

Ako klijent ne dobije potvrdu (PUBACK), ponavlja slanje poruke dok ne dobije potvrdu.

Moguća je duplirana isporuka (isti sadržaj više puta).

Korišćenje:

Kada je važna pouzdanost, ali se aplikacija može nositi s duplikatima.

Primer: Prekidač koji šalje komandu “uključi svetlo” – ako se poruka duplira, svetlo ostaje upaljeno.

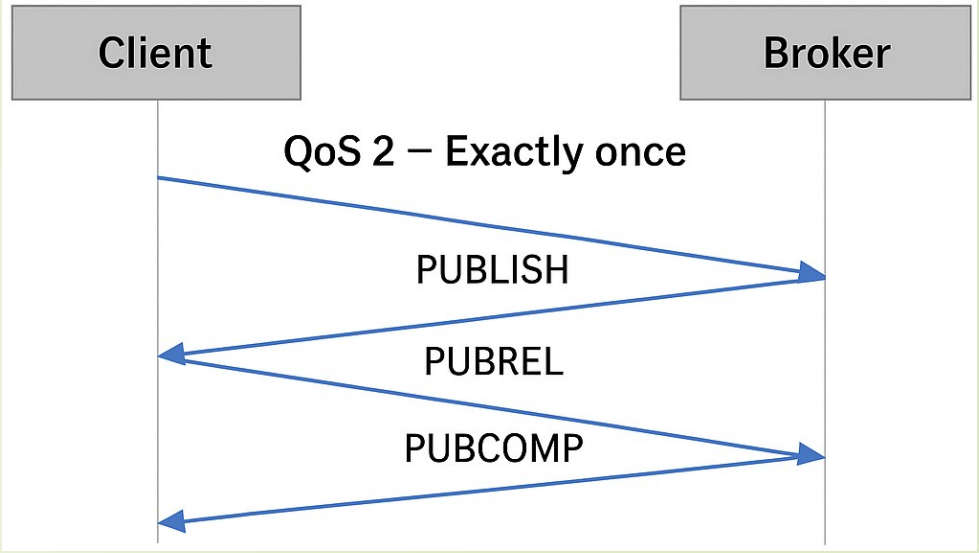
Prednosti:

Bolja pouzdanost.

I dalje relativno niska latencija.

Napomena: Klijent ili aplikacija treba da zna kako ignorisati duplikate ako je potrebno.

Kako **MQTT** radi?



QoS 2 – “Exactly once” (Tačno jednom)

Opis:

Najviši nivo pouzdanosti.

Poruka se sigurno isporučuje tačno jednom, koristeći složen četvorofazni handshake:

- **PUBLISH**
- **PUBREC (potvrda prijema)**
- **PUBREL (puštanje poruke)**
- **PUBCOMP (kompletirano)**

Eliminiše duplikate i gubitke.

Korišćenje:

Kada je kritično da poruka ne bude izgubljena niti duplirana.

Primer: Transakcija plaćanja – dvostruka isporuka bi izazvala dvostruko plaćanje.

Prednosti:

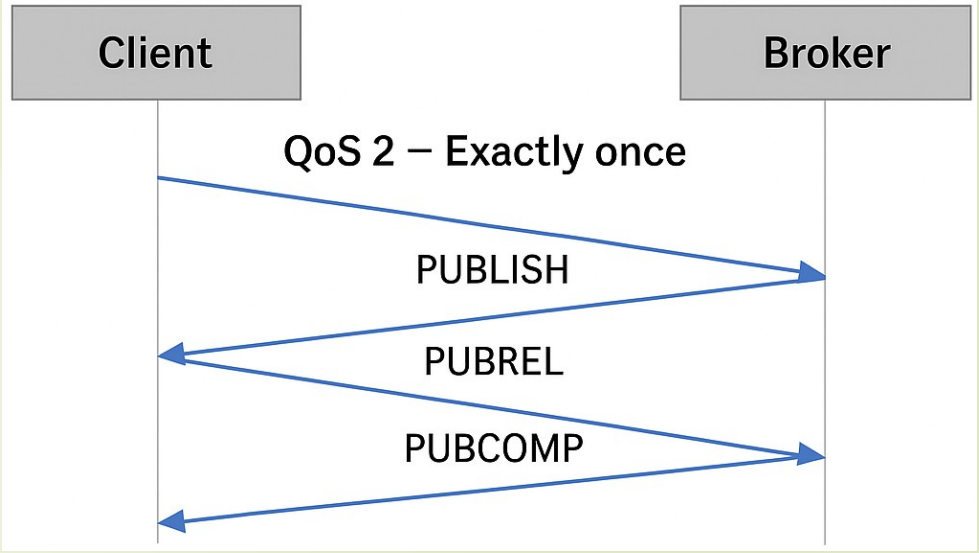
Najveća pouzdanost.

Nedostaci:

Najveće kašnjenje i mrežno opterećenje.

Koristi se samo kada je zaista neophodno.

QoS 2 – “Exactly once” (Tačno jednom)



Osnovni cilj QoS 2:

Obezbediti da se poruka isporuči tačno jednom čak i ako klijent ili broker padnu, veza pukne ili se pojave druge smetnje.

Detaljno po fazama:

1 PUBLISH

Klijent šalje poruku brokeru sa oznakom QoS 2.

Poruka se šalje sa jedinstvenim Packet Identifier-om.

Broker na osnovu toga zna da je u pitanju QoS 2 i čuva poruku u svom privremenom skladištu.

2 PUBREC (Publish Received)

Broker šalje potvrdu da je primio poruku — ali još nije isporučio ni jednom pretplatniku.

Ovo znači: “Primio sam poruku, ali je još nisam obradio u potpunosti.”

U ovom trenutku, klijent može zaboraviti sadržaj poruke, ali mora zapamtiti da čeka PUBCOMP.

3 PUBREL (Publish Release)

Klijent šalje poruku PUBREL brokeru.

Ovo je “dozvola za nastavak” – kaže brokeru: “U redu, sada možeš da isporučiš poruku pretplatnicima.”

PUBREL je signal za završetak transakcije sa strane klijenta.

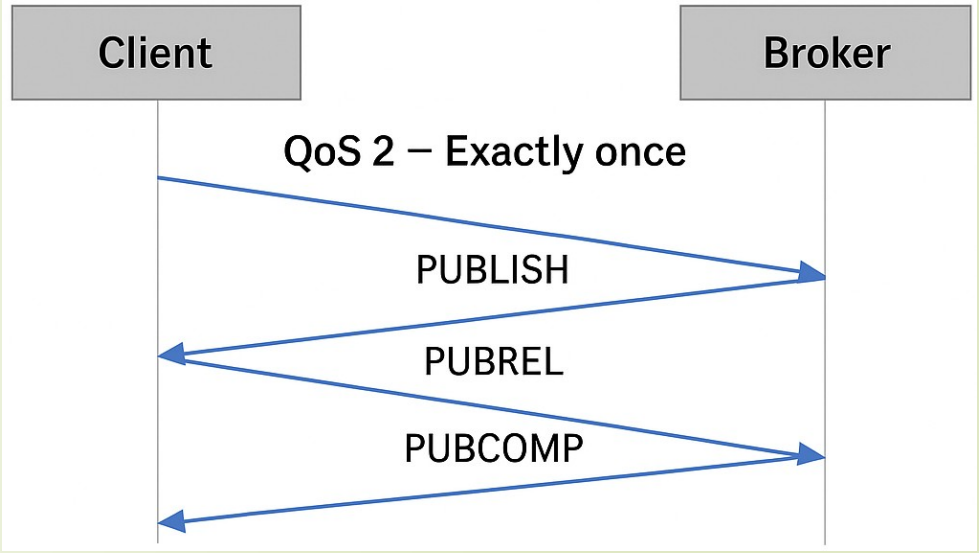
4 PUBCOMP (Publish Complete)

Broker šalje konačnu potvrdu klijentu.

Ovim broker kaže: “Završio sam, poruka je isporučena i transakcija je kompletna.”

Nakon ove poruke, i klijent i broker brišu sve podatke vezane za tu poruku.

QoS 2 – “Exactly once” (Tačno jednom)



REZIME:

Faza	Ko šalje	Značenje
`PUBLISH`	Klijent → Broker	Šaljem ti poruku.
`PUBREC`	Broker → Klijent	Primio sam poruku, još je ne isporučujem.
`PUBREL`	Klijent → Broker	Slobodno je isporuči dalje.
`PUBCOMP`	Broker → Klijent	Gotovo, možeš zaboraviti ovu poruku.

Zašto ovoliko komplikovano?

Zamislte da klijent padne nakon slanja PUBLISH, ali pre nego što primi PUBREC. Kad se ponovo poveže, ponovo pošalje PUBLISH, ali pošto broker već ima poruku i poslao PUBREC, zna da se ne radi o novoj poruci.

Na sličan način, ako broker padne između PUBREL i PUBCOMP, neće isporučiti poruku duplo jer zna da je već obradio PUBREL i poslaće samo PUBCOMP.

Kako **MQTT** radi?

Poređenje QoS nivoa

Karakteristika	QoS 0	QoS 1	QoS 2
Isporuka poruka	Najviše jednom	Najmanje jednom	Tačno jednom
Pouzdanost	Niska	Srednja	Visoka
Overhead	Minimalan	Srednji	Visok
Latencija	Najniža	Srednja	Najviša
Duplikati	Nema	Mogući	Nikada

Kako **MQTT** radi?

Ponašanje u praksi:

Klijent određuje QoS nivo kada šalje poruku.

Pretplatnik može zatražiti drugi QoS nivo, a efektivni QoS je minimum između pošiljaoca i primaoca.

Primjer:

Pošiljalac šalje sa QoS 2, pretplatnik se pretplatio na QoS 1 → poruka se isporučuje sa QoS 1.

MQTT radni tok

Kada se razumiju osnovne komponente MKTT-a, može se pogledati kako funkcioniše opšti radni tok:

- Klijenti pokreću vezu sa brokerom koristeći TCP/IP, sa opcionim TLS/SSL enkripcijom za bezbjednu komunikaciju. Klijenti obezbjeđuju akreditive za autentifikaciju i navode čistu ili trajnu sesiju.
- Klijenti ili objavljuju poruke na određene teme ili se pretplate na teme da bi primali poruke. Klijenti izdavači šalju poruke brokeru, dok pretplatnici izražavaju interesovanje za primanje poruka o određenim temama.
- Broker prima objavljene poruke i prosleđuje ih svim klijentima koji su pretplaćeni na relevantne teme. Osigurava pouzdanu isporuku poruka prema specificiranom nivou kvaliteta usluge (QoS) i upravlja skladištenjem poruka za isključene klijente, na osnovu tipa sesije.

Početak rada sa **MQTT**: Brzi vodič

Sada će se pokazati kako početi koristiti MQTT, sa nekoliko jednostavnih demonstracija.

Prije nego što se počne, potrebno je pripremiti MQTT brokera i MQTT klijenta.

Priprema MQTT brokera

EMQX je skalabilna, distribuirana MQTT platforma za razmenu poruka koja podržava neograničen broj konekcija, nudi besprekornu integraciju i može da se primeni bilo gdje.

Obezbjeđuje različite edicije koje zadovoljavaju različite zahtjeve korisnika.

Početak rada sa **MQTT**: Brzi vodič

Priprema MQTT brokera

EMQX Serverless je MQTT usluga za više zakupaca s cijenama prema korištenju i funkcijama automatskog skaliranja. Može se pokrenuti za nekoliko minuta i radi u 17 regija na AWS-u, Google Cloudu i Microsoft Azureu.

Isprobajte EMQX Serverless. Zauvijek besplatno za manje od 1 miliona minuta sesije mjesečno. Besplatni javni MQTT broker.

U ovom vodiču koristit ćemo se besplatnim javnim MQTT brokerom koji nudi EMQ, izgrađen na EMQX platformi.

Pojedinosti o pristupu serveru su sljedeće:

Adresa brokera: broker.emqx.io

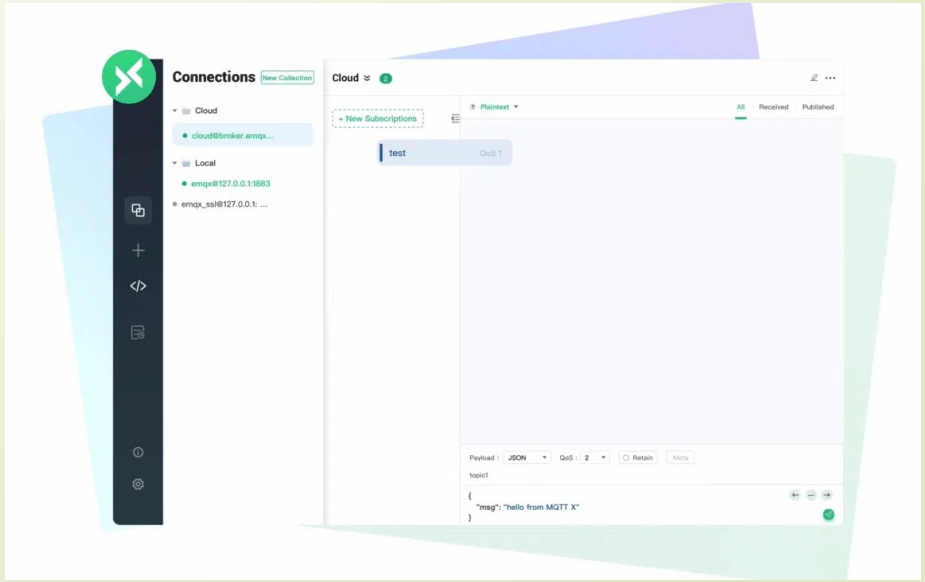
TCP priključak: 1883

WebSocket priključak: 8083

Početak rada sa **MQTT**: Brzi vodič

Priprema MQTT klijenta

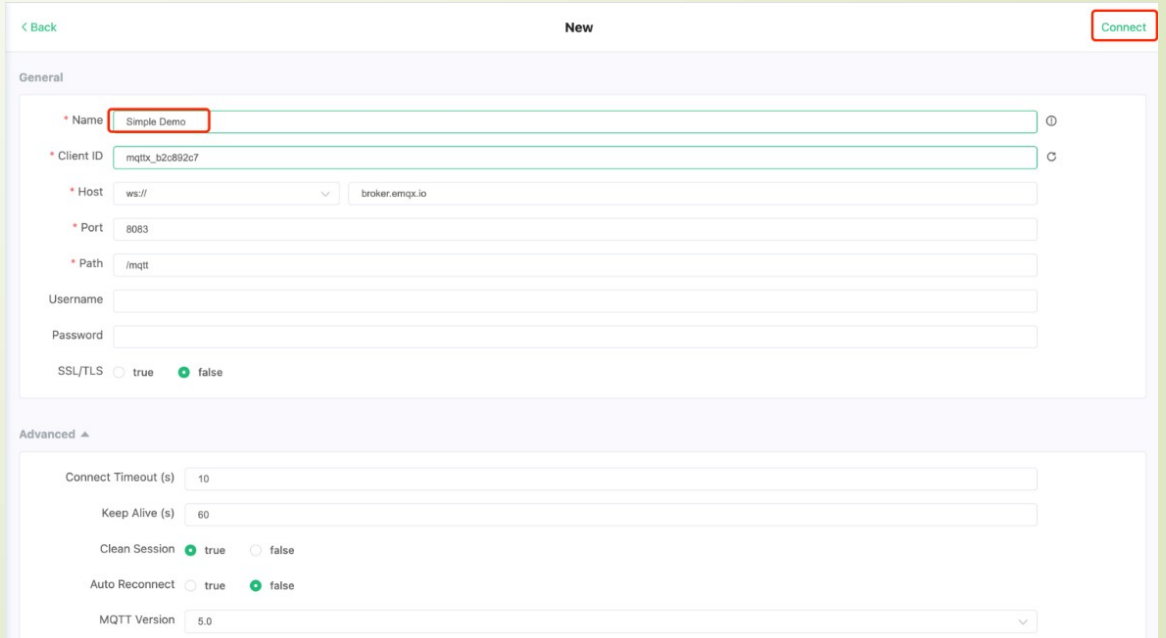
- Ovom prilikom će se koristiti MQTT klijentski alat koji nudi **MQTTX** i koji podržava pristup putem linka: **http://www.emqx.io/online-mqtt-client**. MQTT X takođe nudi **desktop client** i command line tool.
- MQTTX je elegantan višeplatformski MQTT **5.0** desktop klijent koji radi na macOS, Linux i Windows. Njegov interfejs u stilu ćaskanja omogućava korisnicima da lako kreiraju više MQTT veza i pretplata za MQTT poruke.



Početak rada sa **MQTT**: Brzi vodič

Kreiranje MQTT konekcije

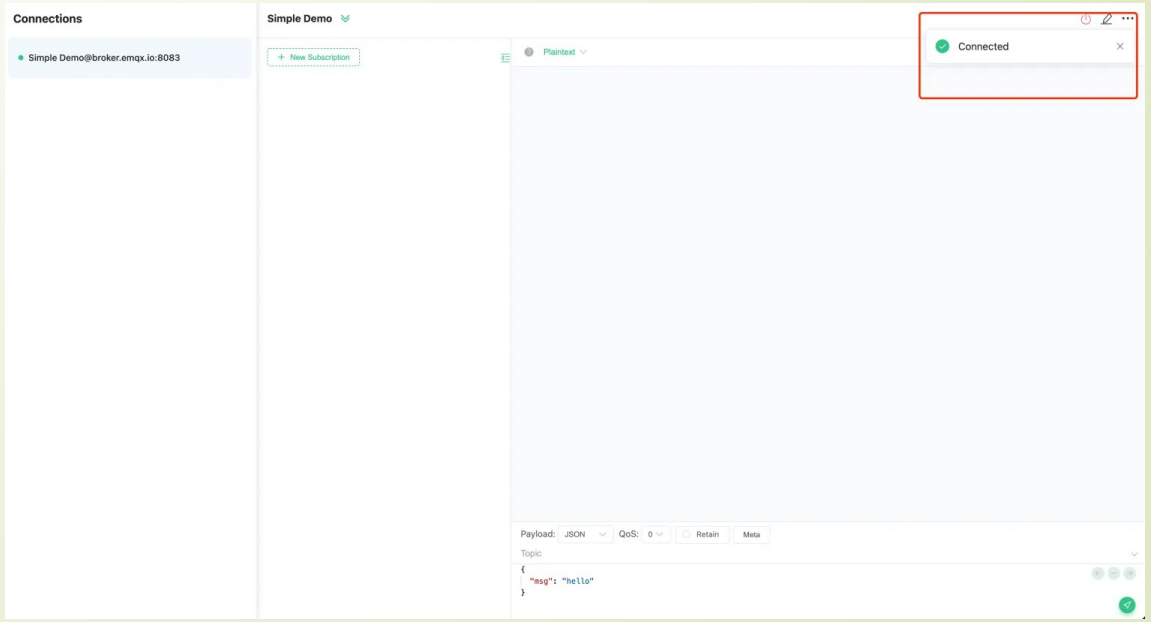
- Prije korištenja MQTT protokola za komunikaciju, klijent mora kreirati MQTT vezu za povezivanje s brokerom.
- Treba ići na: **http://www.emqx.io/online-mqtt-client** i kliknite na dugme New Connection na sredini stranice i vidjet ćete sljedeću stranicu.



Početak rada sa **MQTT**: Brzi vodič

Kreiranje MQTT konekcije

- Unesite **Proba** u Name i kliknite dugme Connect u gornjem desnom uglu za stvaranje MQTT veze. Sljedeće označava da je veza uspješno uspostavljena.

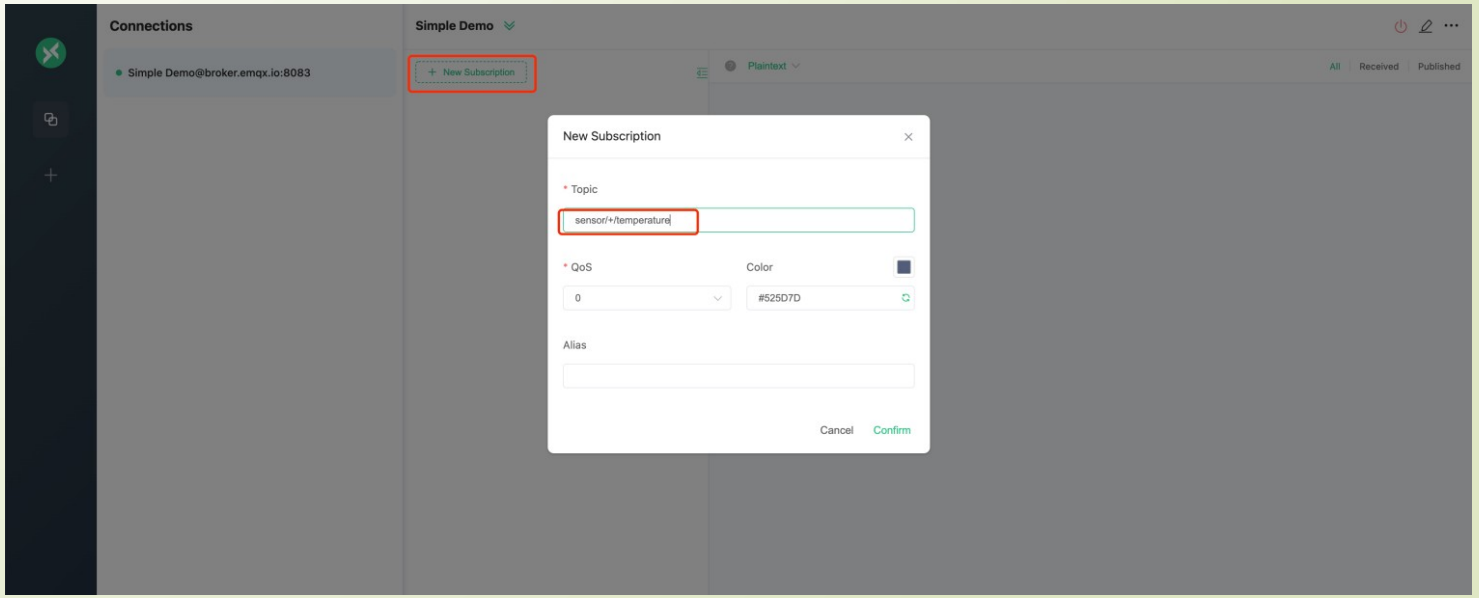


Kako biste saznali više o parametrima MQTT veze, pogledajte post na blogu:
How to Set Parameters When Establishing an MQTT Connection.

Početak rada sa **MQTT**: Brzi vodič

Pretplata se na temu:

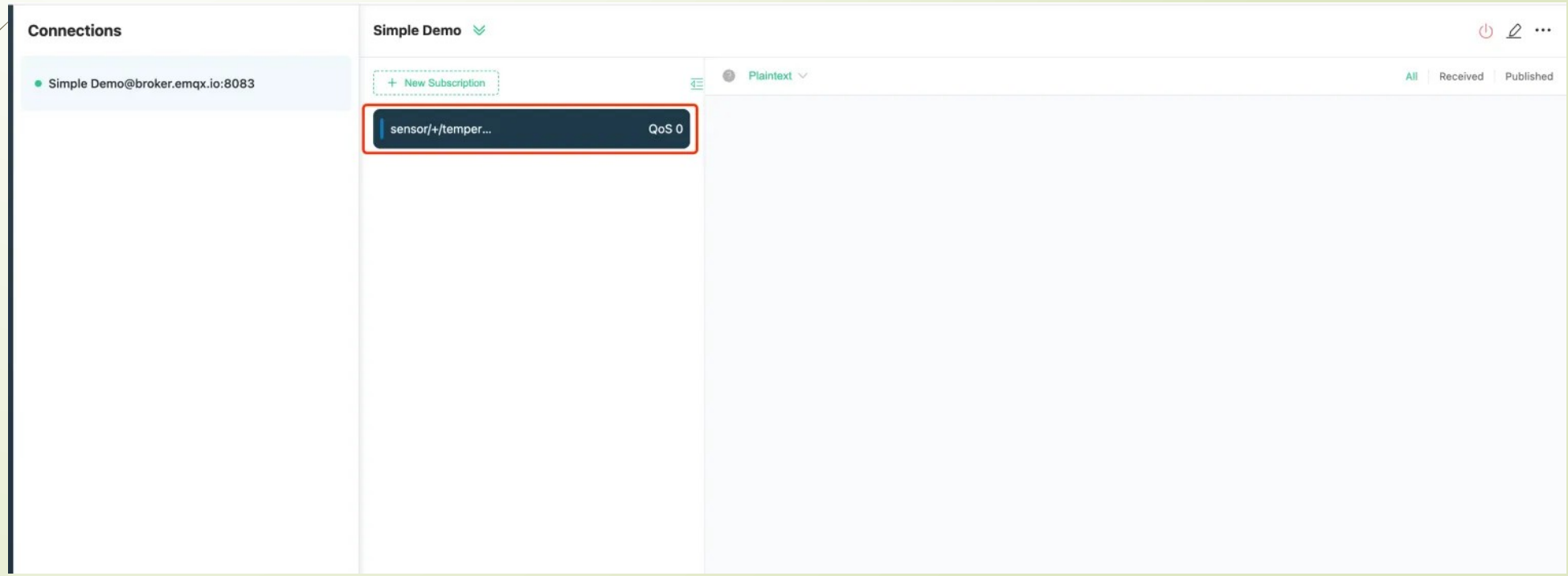
- Pretplaćivanje na temu sa zamjenskim znakom **senzor/+/temperatura** u **Proba** vezi stvorenoj ranije, koja će primati podatke o temperaturi koje objavljuju svi senzori.
- Kao što je prikazano u nastavku, kliknite na dugme New Subscription i unesite temu **senzor/+/temperatura** u polje Topic u iskačućem okviru, zadržavajući zadani QoS na 0.



Početak rada sa **MQTT**: Brzi vodič

Pretplata se na temu:

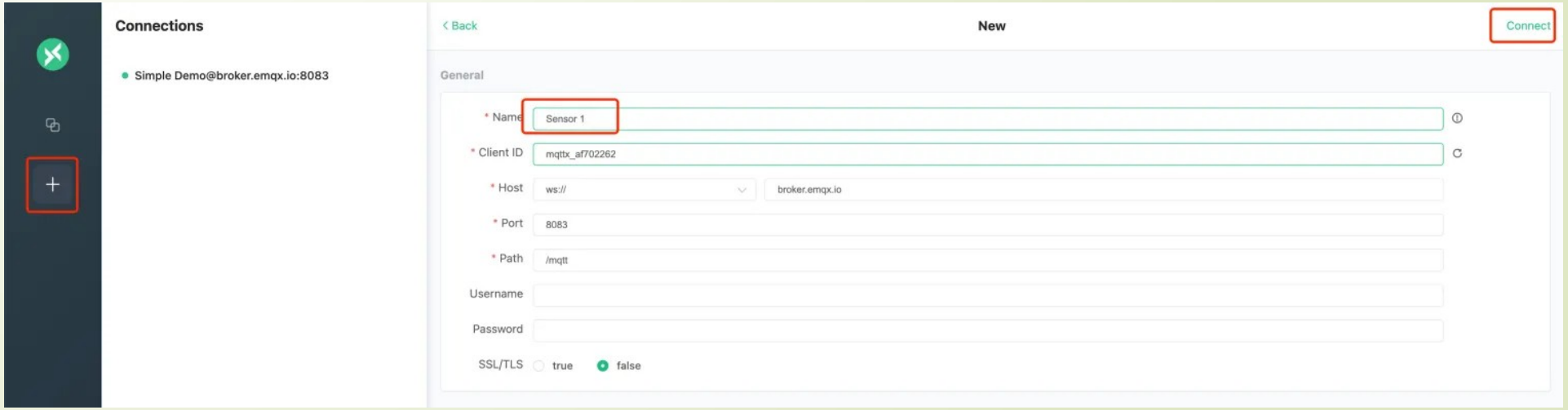
- Nakon uspješnog pretplaćivanja, vidijet ćete dodatni zapis u sredini liste pretplata.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

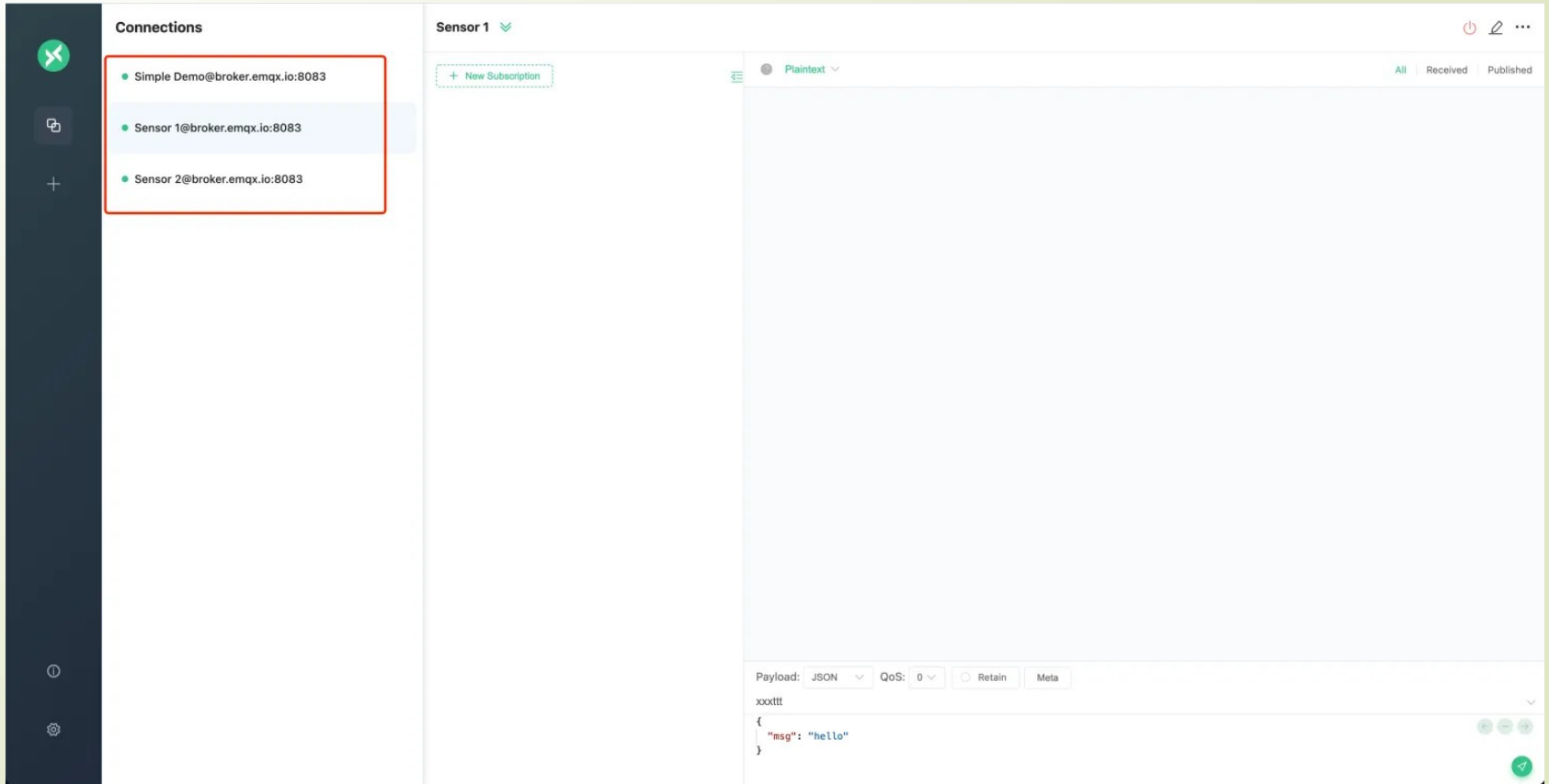
- Zatim treba kliknuti dugme + na lijevom meniju kako bismo stvorili dvije veze, senzor 1 i senzor 2, kako bismo simulirali dva temperaturna senzora.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

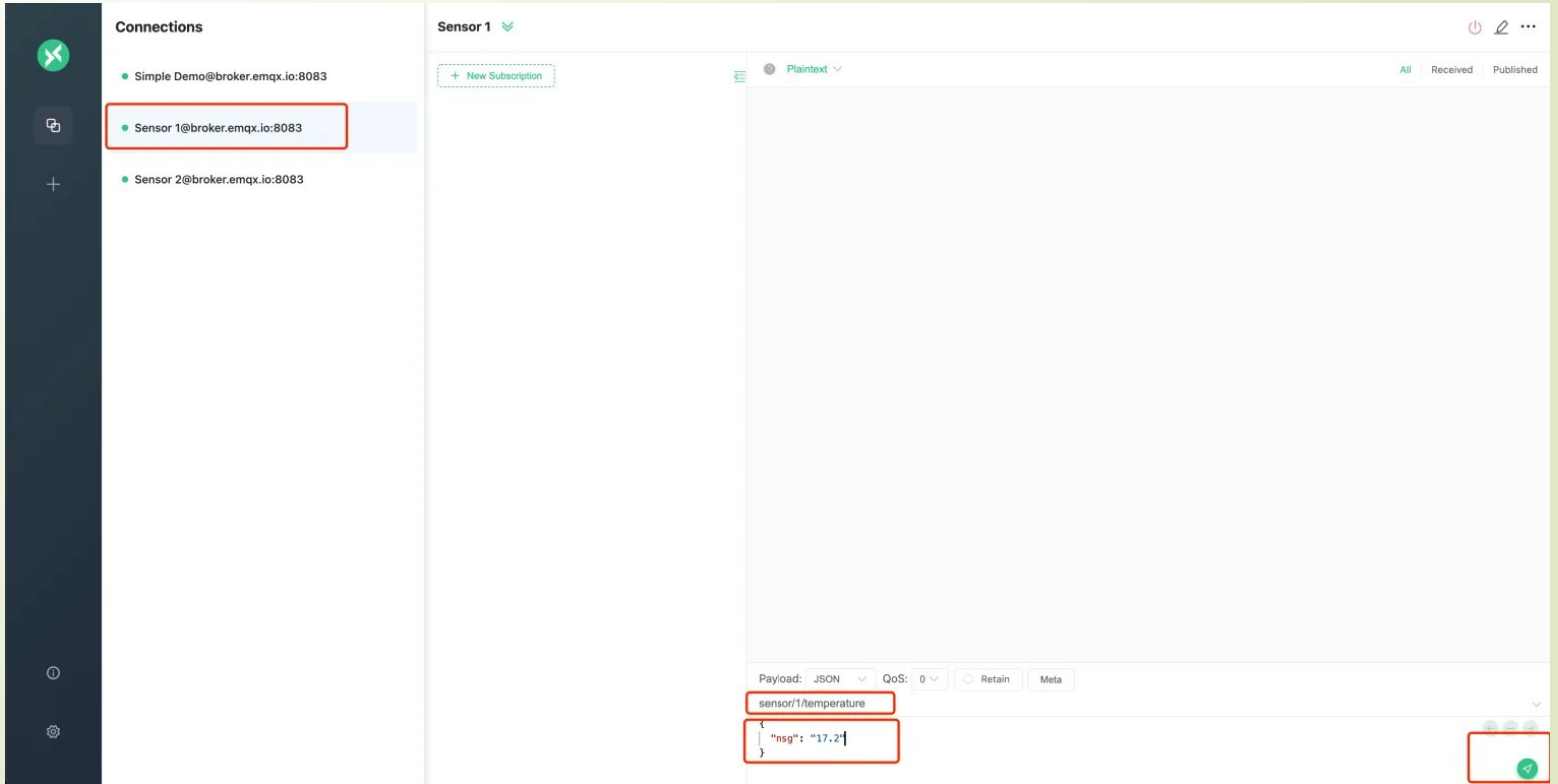
- Nakon što se veza kreira, vidjet ćete tri veze, a sve tačkice mrežnog statusa s lijeve strane veza bit će zelene.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

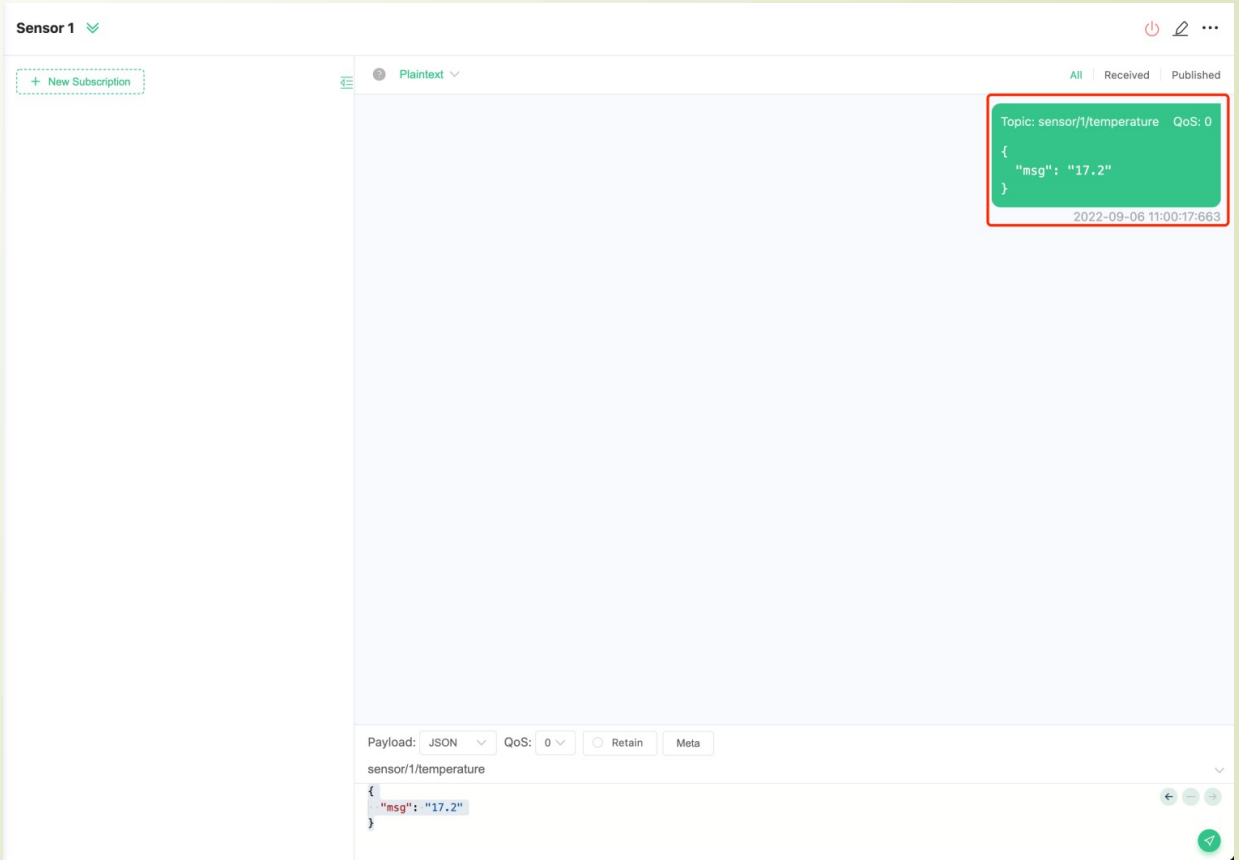
- Izaborite vezu **Senzor 1**, i unesite temu za objavljivanje **senzor/1/temperatura** u donjem levom delu stranice, unesite sledeću poruku, u formatu JSON, u okvir za poruke i kliknite na dugme za objavljivanje u donjem desnom uglu da biste poslali poruku.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

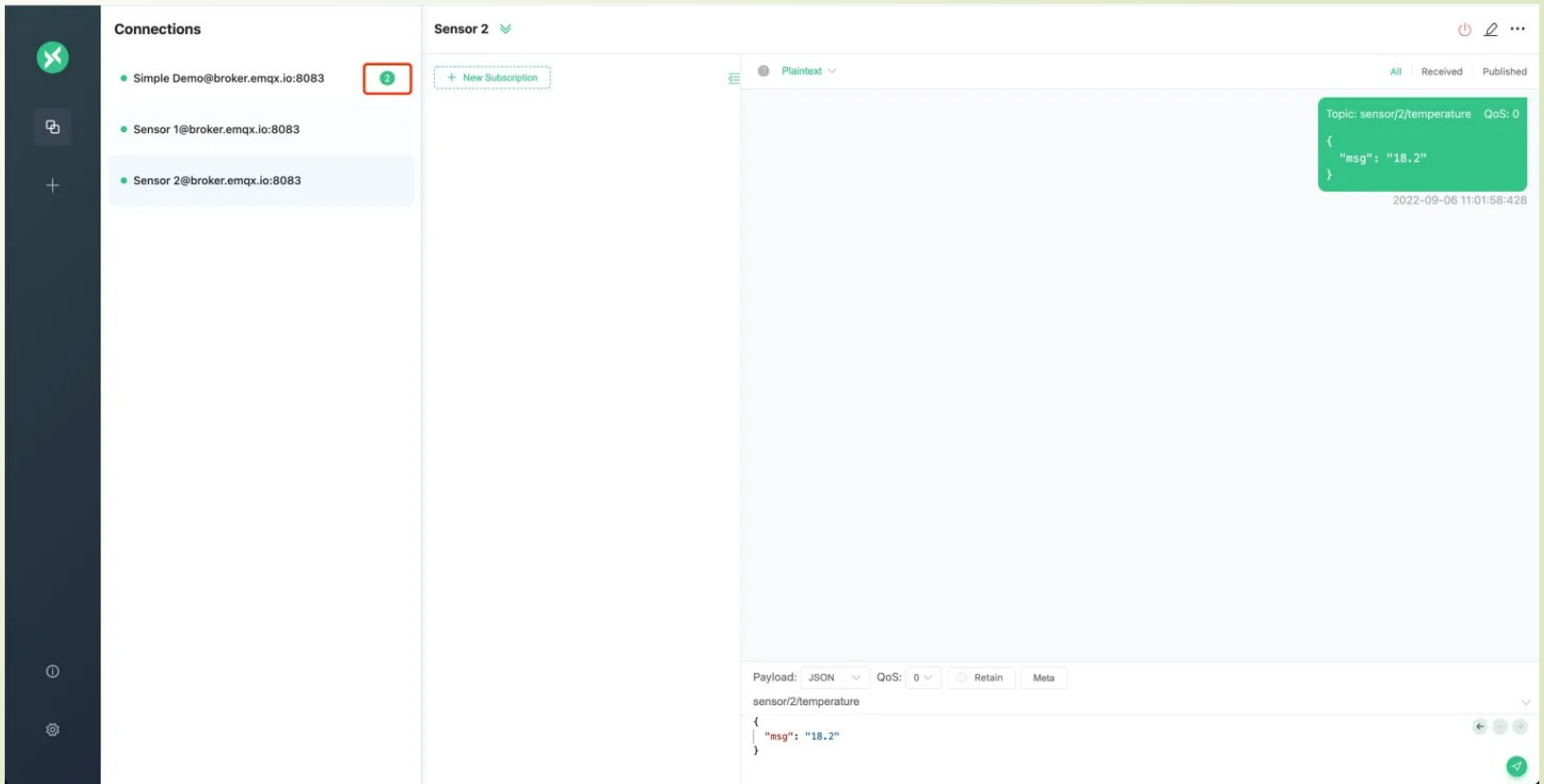
- Poruka je uspješno poslana na sljedeći način.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

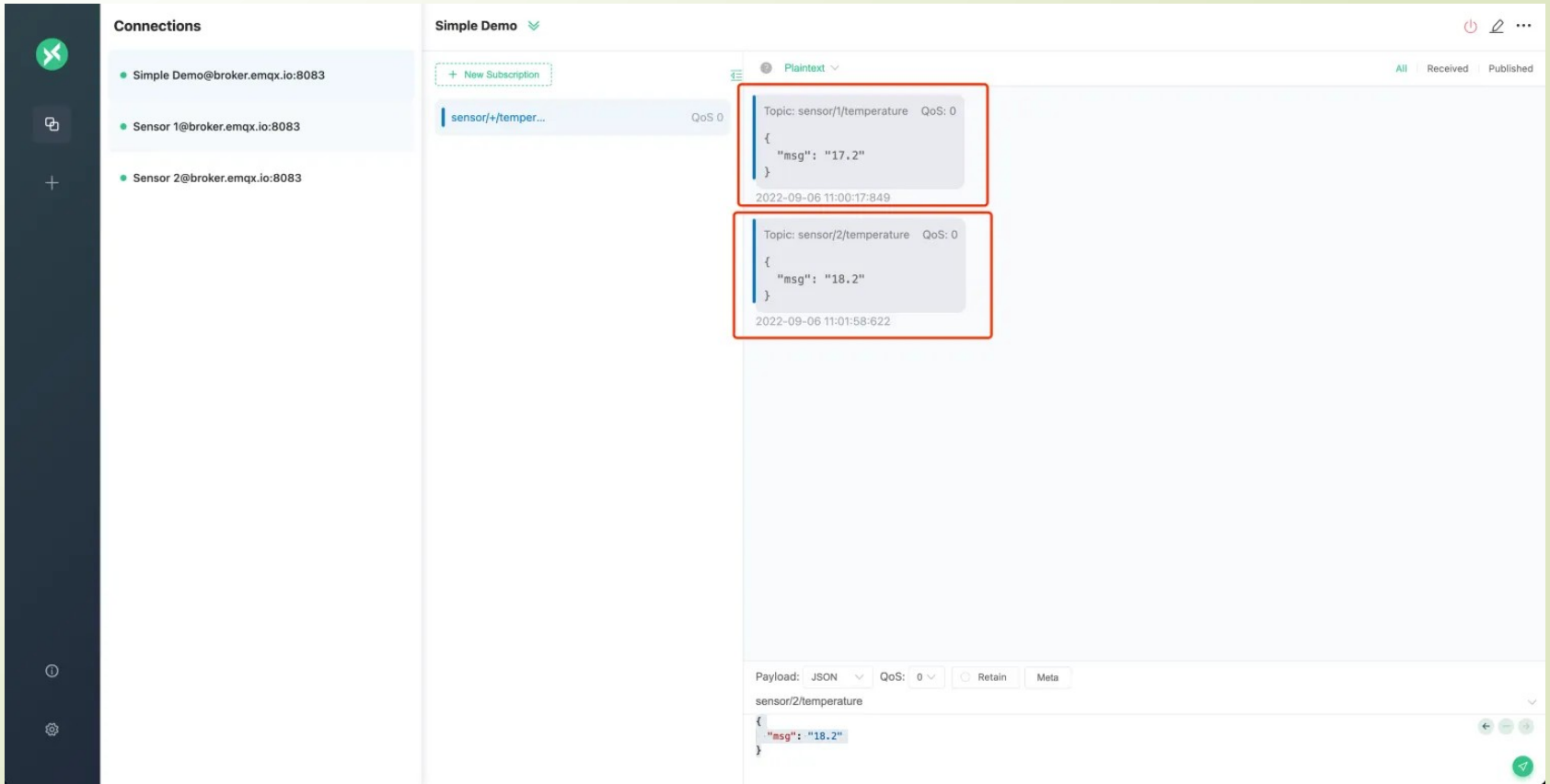
- Koristeći iste korake, objavite sljedeću JSON poruku u temi **senzor/2/temperatura** u vezi senzora 2.



Početak rada sa **MQTT**: Brzi vodič

Objavljivanje MQTT poruka:

- Kliknite vezu **Proba** i vidjet ćete dvije poruke koje su poslala dva senzora.



Arduino uređaj kao klijent

Upišite u Arduino Uno R4 WiFi skeč nazvan:

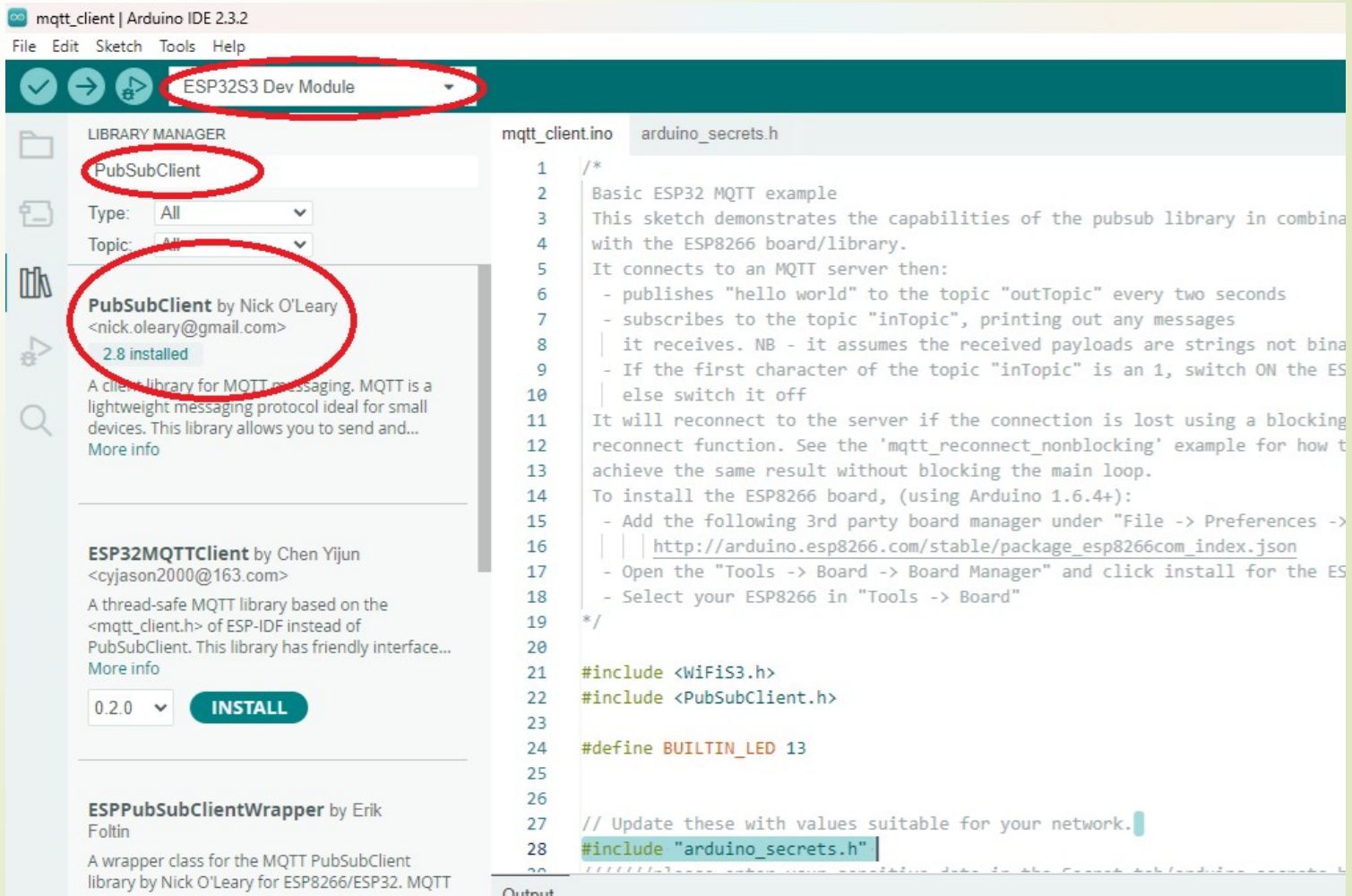
mqtt_cilent.ino

Skeč omogućuje da adruino uređaj funkcioniše kao MQTT klijent.

Da bi se skeč uspješno kompajlirao i izvršavao potrebno je instalirati biblioteku: **PubSubClient**

Arduino uređaj kao klijent

Instaliranje PubSubClient biblioteke:



Arduino uređaj kao klijent – uloga **loop()** funkcije

U biblioteci **PubSubClient**, funkcija **client.loop()** ključna je za održavanje veze s **MQTT brokerom** i rukovanje dolaznim porukama.

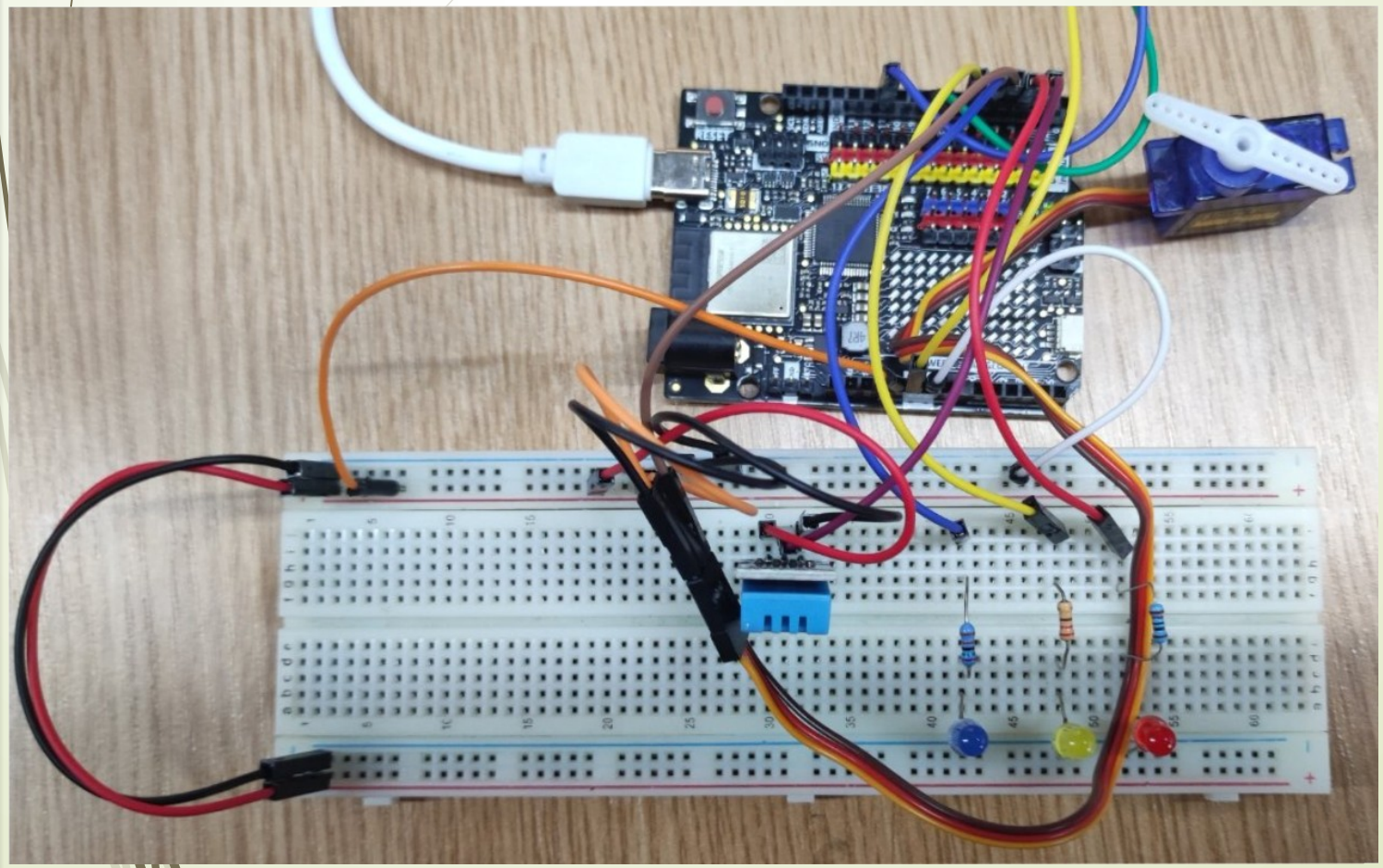
Kada se pretplatite na teme ili objavite poruke koristeći MQTT (Message Queuing Telemetry Transport), mikrokontrolerska ploča mora stalno komunicirati s MQTT brokerom kako bi primala poruke ili slala ažuriranja. Za ovu kontinuiranu komunikaciju odgovorna je funkcija client.loop().

Evo što radi:

- **Održavanje veze**: osigurava da se veza s MQTT brokerom održava živom. MQTT se temelji na trajnoj TCP vezi, a klijent mora slati periodične kontrolne pakete kako bi održao ovu vezu.
- **Rukovanje dolaznim porukama**: Provjerava sve dolazne poruke iz pretplaćenih tema. Kada stignu poruke, funkcija loop() ih obrađuje i pokreće odgovarajuće funkcije povratnog poziva koje ste možda definisali u svom kodu.
- **Obrada odlaznih poruka**: Ako postoje poruke u redu čekanja za objavljivanje, funkcija loop() se brine za njihovo slanje MQTT brokeru.
- **Keep-Alive Mechanism**: Također upravlja mehanizmom za održavanje aktivnosti MQTT protokola, gdje povremeno šalje ping zahtjeve brokeru kako bi se osiguralo da je veza još uvijek aktivna.

Sve u svemu, redovno pozivanje **client.loop()**, obično unutar glavne petlje vašeg programa, bitno je za nesmetan rad MQTT komunikacije. Održava komunikaciju živom, obrađuje dolazne poruke i osigurava da vaš uređaj i dalje reaguje na MQTT mrežu.

Priprema za vježbu



Mikrokontrolerski dizajn

Crvena LED je preko otpornika 220▶ spojena na pin 6.
Žuta LED je preko otpornika 220▶ spojena na pin 5.
Plava LED je preko otpornika 220▶ spojena na pin 3.

Servo motor je spojen na pin 4. Uz to VCC je spojeno na 5V, a GND na GND.

Signal DHT11 senzora povezan je na pin 2. VCC je spojeno na 5V, a GND na GND

Link za preuzimanje MQTT extenzije:
https://ullisroboterseite.de/android-AI2-PahoMQTT-en.html#down

Za vježbu

Zadatak 1

Omogućiti da Arduino uređaj šalje u temi TempHum_{Naziv_ekipe} informacije o temperaturi i vlagi, koju mjeri njegov T&H senzor, prikazati u formatu, kao što ilustruje primjer:

Temperatura: 19*C, 66.2*F, Vlaga: 27%

1 bod

Za vježbu

Zadatak 2

Omogućiti da se upravlja intezitetom sjaja LED na Arduino uređaju slanjem sljedećih u temu led_drive_{Naziv_ekipe}:

Primjer komende: LED:R20Y120G90

Po uspješnom izvršenju komande Arduino uređaj u temi led_set_{Naziv_ekipe} odgovori na sljedeći način:

CRVENA: 20, ŽUTA: 129, Zelena: 90

(2-1 bod)

Za vježbu

Zadatak 3

Omogućiti da se upravlja položajem servo motora na Arduino uređaju, kao i njegovom brzinom pomjeranja slanjem sljedećih u temu servo_drive_{Naziv_ekipe}:

Primjer komende: SET20SPEED10

Po uspješnom izvršenju komande Arduino uređaj u temi servo_set_{Naziv_ekipe} odgovori na sljedeći način:
Servo ugao: 20 stepeni, Brzina pomjeranja: 10 kor/s

Obezbijediti da je po uključenju uređaja servo mortor na poziciji 0 stepeni.

(2-1 bod)