

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE

SEMINAR

Realizacija DHT22 senzora, te povezivanje
na OpenHAB

Ante Pupačić, Luka Svaguša, Ante Bakota

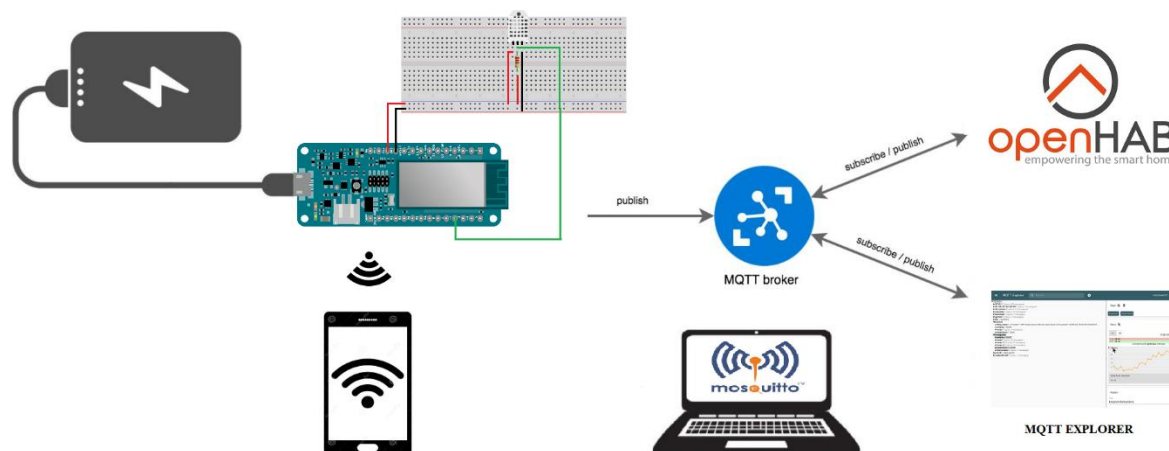
Split, siječanj 2021.

SADRŽAJ

| | |
|----------------------------------------------|-----------|
| 1. UVOD..... | 1 |
| 2. SENZORSKI ČVOR..... | 2 |
| 2.1. Arduino MKR WIFI 1010..... | 2 |
| 2.2. DHT22 senzor..... | 3 |
| 2.3. Povezivanje na WiFi..... | 5 |
| 3. MQTT PROTOKOL..... | 7 |
| 3.1. MQTT provjera autentičnosti..... | 8 |
| 3.2. Slanje podataka na broker..... | 9 |
| 3.3. MQTT Explorer..... | 11 |
| 4. OpenHAB..... | 12 |
| 4.1. Paper UI..... | 12 |
| 4.2. Basic UI..... | 14 |
| 5. ZAKLJUČAK..... | 16 |
| 6. LITERATURA..... | 17 |

1. UVOD

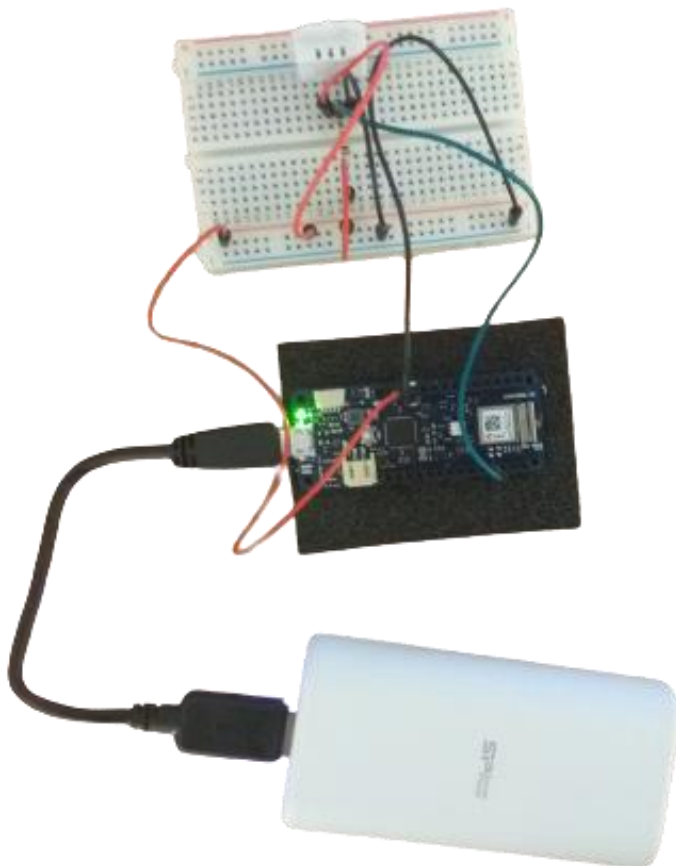
Unutar projekta realiziran je senzorski čvor s periodičkim očitavanje temperature i vlažnosti pomoću DHT22 senzora. Korišten je Arduino MKR WIFI 1010 mikrokontroler koji ima mogućnost povezivanja na WiFi mrežu. Arduino MKR WIFI 1010 je isprogramiran pomoću aplikacije platformia.io koji je ekstenzija unutar Visual studio code. Napajanje mikrokontrolera je realizirano pomoću powerbanka spojenog preko USB kabela. Podaci očitani na senzoru se šalju na MQTT broker preko WiFi mreže. MQTT broker se pokreće na računalu, za povezivanje na broker potrebno je korisničko ime (engl. *username*) i lozinka (engl. *password*) što daje dodanu zaštitu podacima. Podaci s MQTT brokera se šalju na OpenHAB, gdje se mogu vizualizirati i analizirati (*Slika 1.1. Cjelokupna shema projekta*). Cjelokupni kod projekta se nalazi na GitHub-u: <https://github.com/apupac00/SeminarIS>



Slika 1.1. Cjelokupna shema projekta

2. SENZORSKI ČVOR

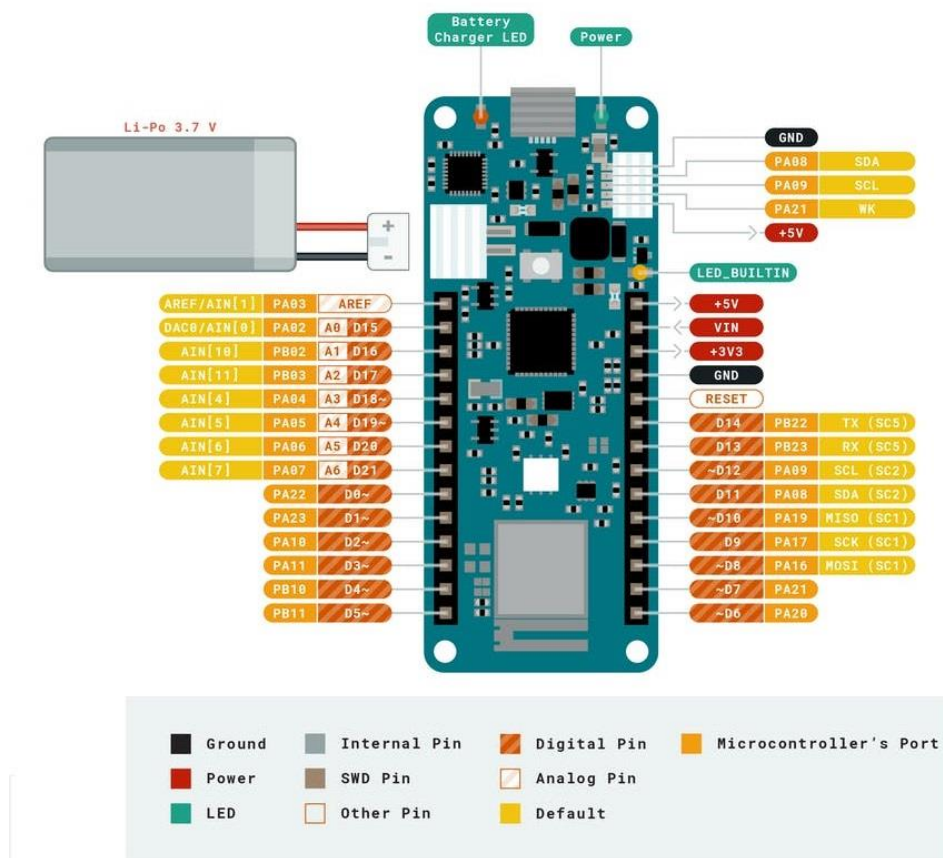
Senzorski čvor se sastoji od od mikrokontrolera Adrudino MKR WIFI 1010, DHT22 senzora, otpornika od $10k\Omega$ i napajanje pomoću powerbanka (*Slika 2.1. Shema senzorskog čvora*). DHT22 senzor periodički očitava vrijednosti temperature i vlažnosti koji se dalje šalju na MQTT broker.



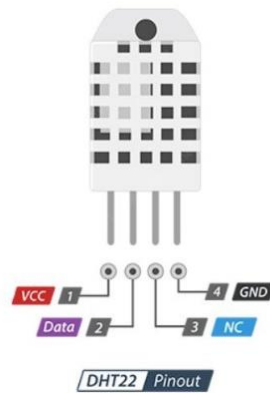
Slika 2.1. Shema senzorskog čvora

2.1. Arduino MKR WIFI 1010

Glavni procesor ploče je Arm Cortex-M0 32-bitni SAMD21 male snage, kao na ostalim pločama iz obitelji Arduino MKR (*Slika 2.2. Pinovi Arduina MKR WIFI 1010*).



potrebni analogni ulazni pinovi). Mana ovog senzora je što može dobiti podatke svako 2 sekunde (Slika 2.3. Pinovi DHT22).



Slika 2.3. Pinovi DHT22

Kod unutar programa koji inicijalizira DHT22 senzor i čita vrijednost temperature i vlažnosti:

```
//Biblioteke potrebne za rad s DHT22 senzorom
#include "DHT.h"
#include <Adafruit_Sensor.h>

//Struktura sa podacima temperature i vlažnosti
struct dataSensor
{
    float temperature = 0.0f;
    float humidity = 0.0f;
};

static dataSensor data;

#define DHTPIN 2 //Pin na koji se šalju podaci
#define DHTTYPE DHT22 //Tip DHT senzora
DHT dht(DHTPIN, DHTTYPE);

//Kod programa unutar void loop() funkcije
float t = 0.0F;
float h = 0.0F;

t = dht.readTemperature(); //Poziv metode koja čita vrijednost temperature
```

```

h = dht.readHumidity(); //Poziv metode koja čita vrijednost vlažnosti

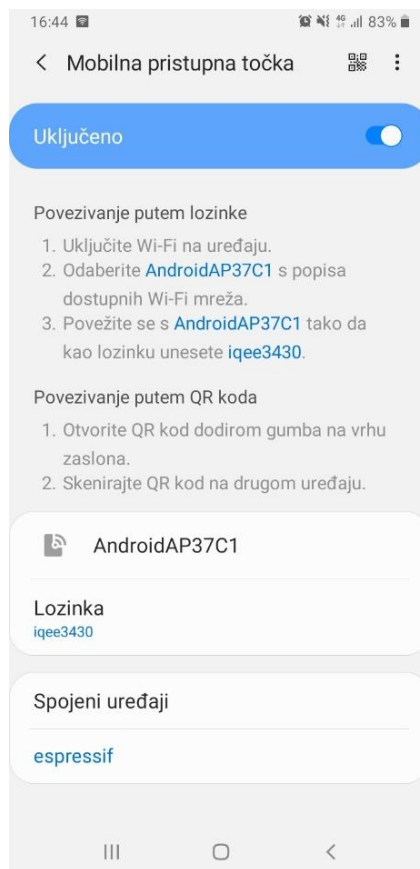
//Provjera da li je senzor ispravno očita vrijednosti
if (isnan(t) || isnan(h))
{
    //Serial.println("Failed to read sensor");
}

data.temperature = t; //Očitana vrijednost temperature se pohranjuje
unutar strukture
data.humidity = h; //Očitana vrijednost vlažnosti se pohranjuje unutar
strukture

```

2.3. Povezivanje na WiFi

Arduino MKR WIFI 1010 ima ugrađen modul za povezivanje na WiFi mrežu, te gotove biblioteke gdje se u nekoliko linija koda može povezati na WIFI mrežu, Arduino je povezan mobilnu pristupnu točku Samsung mobilnog uređaja (*Slika 2.4. Mobilna pristupna točka*).



Slika 2.4. Mobilna pristupna točka

Kod unutar programa koji povezuje Arduino na mobilnu pristupnu točku:

```
//Biblioteke potrebne za povezivanje na mobilnu pristupnu točku
#include <WiFiNINA.h>

const char *ssid = "AndroidAP37C1"; //SSID mobilne pristupne točke
const char *password = "iqee3430"; //Lozinka mobilne pristupne točke

setup_wifi();//Poziv funkcije unutar void setup() za povezivanje na
mobilnu pristupnu točku

//Funkcija koja povezuje Arduino na mobilnu pristupnu točku
void setup_wifi()
{
    delay(10);
    //Serial.println();
    //Serial.print("Connecting to ");
    //Serial.println(ssid);
    WiFi.begin(ssid, password);

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

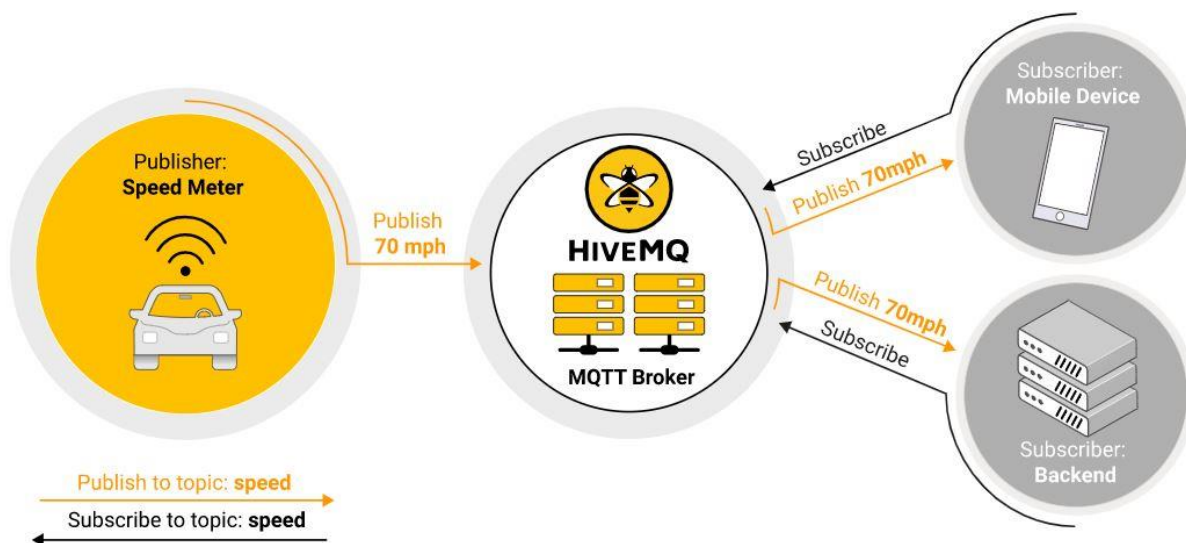
    randomSeed(micros());
    //Serial.println("");
    //Serial.println("WiFi connected");
    //Serial.println("IP address: ");
    //Serial.println(WiFi.localIP());
}
```


3. MQTT PROTOKOL

MQTT je protokol za slanje i objavljivanje poruka dizajniran za M2M (engl. *machine to machine*) telemetriju u okruženjima male propusnosti. Danas se najviše koristi kod internet stvari (engl. *Internet of Things*, skraćeno *IoT*). Lagan je, otvoren, jednostavan i dizajniran tako da se jednostavno implementira.

Publish/Subscribe:

Publish/Subscribe (poznat i kao pub/sub) nudi alternativu tradicionalnoj arhitekturi klijent poslužitelj. U modelu klijent-server klijent komunicira izravno s krajnjom točkom. Model pub/sub razdvaja klijenta koji šalje poruke (engl. *publisher*) od klijenta ili klijenata koji primaju poruke (engl. *subscribers*). Izdavači i pretplatnici nikada izravno ne komuniciraju. Već se između njih upravlja trećom komponentom (broker). Zadatak brokera je filtrirati sve dolazne poruke i pravilno ih distribuirati pretplatnicima (Slika 3.1. *MQTT protokol*).



Slika 3.1. *MQTT protokol*

Tema:

U MQTT, riječ tema odnosi se na niz UTF-8 koji broker koristi za filtriranje poruke za svakog povezanog klijenta. Tema se sastoji od jedne ili više tema. Svaka razina teme odvojena je

kosom crtom naprijed (separator razine teme). Klijentu nije potrebno stvoriti željenu temu prije nego objavi nešto ili se pretplati na temu. Broker prihvaća svaku valjanu temu bez ikakve prethodne inicijalizacije (*Slika 3.2. Primjer teme*).



Slika 3.2. Primjer teme

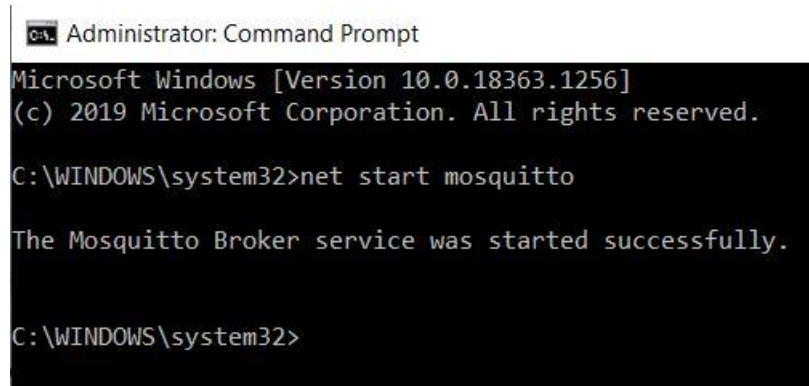
Razina kvalitete usluge (engl. *Quality of Service*, skraćeno *QoS*):

Razina kvalitete usluge je ugovor između pošiljatelja poruke i primatelja poruke koji definira garanciju isporuke za određenu poruku. Postoje tri razine QoS u MQTT-u:

- Najviše jednom (0)
- Barem jednom (1)
- Točno jednom (2)

3.1. MQTT provjera autentičnosti

MQTT broker korišten u ovome projektu je Mosquitto koji se pokreće s računala upisivanje naredbe "net start mosquitto" unutar command prompta pokrenutog kao administrator (*Slika 3.3. Pokretanje MQTT brokera*).



```
C:\> Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>net start mosquitto

The Mosquitto Broker service was started successfully.

C:\WINDOWS\system32>
```

Slika 3.3. Pokretanje MQTT brokera

Za povezivanje na broker potrebno je korisničko ime koje je “mqtt” i lozinka koja je “pasko123”, a host je IPv4 adresa WiFi mreže na koji je računalo povezano. Tako su podaci koji se šalju na MQTT broker zaštićeni, što nije u slučaju javnog MQTT brokera.

3.2. Slanje podataka na broker

Arduino MKR WIFI 1010 ima mogućnost slanja podataka na MQTT broker. Postoje već gotove biblioteke koje pojednostavljaju pisanje koda za slanje podataka na broker. Podaci koji se šalju na broker pretvaraju se u JSON (engl. *JavaScript Object Notation*) format. Tako da se na jednu temu može slati više parametara kao što je u ovome slučaju temperatura i vlažnost.

Kod unutar programa koji omogućava slanje podataka na MQTT broker

```
//Biblioteke za slanje podataka na MQTT broker
#include <SPI.h>
#include <PubSubClient.h>

//Biblioteke za pretvaranje podataka u JSON format
#include <ArduinoJson.h>

static char payload[256];
StaticJsonDocument<256> doc;
#define DEVICEID "804b7760-f511-11ea-bcda-a5152a32a9f6"

const char broker[] = "193.198.245.225";// Adresa brokera

const char publishTopic[] = "DHT22/sensors";// Tema
```

```

    mqtt.setServer(broker, 1883); // Definiranje brokera i porta unutar
funkcije void setup()

    PubSubClient mqtt(mkr1010Client);

    //Provjera da li je Arduino povezan na broker ako nije pokušaj opet
    if (!mqtt.connected())
    {
        reconnect();
    }

    mqtt.loop();

    //Pretvaranje podatka o temperaturi i vlažnosti u JSON format
    doc["temperature"] = data.temperature;
    doc["humidity"] = data.humidity;
    serializeJsonPretty(doc, payload);

    //Slanje podataka na MQTT broker svako 10 sekundi
    long now = millis();
    if (now - lastData > 10000)
    {
        lastData = now;
        //Serial.println(payload);
        mqtt.publish(publishTopic, payload);
    }

    //Funkcija koja provjerava da li je Arduino povezan na MQTT broker, ako
nije pokušaj opet nakon 5 sekundi
    void reconnect()
    {
        while (!mqtt.connected())
        {

            //Serial.print("Attempting MQTT connection ....");

            if (mqtt.connect(DEVICEID, "mqtt", "pasko123"))
            {

                //Serial.println("Connected to MQTT Broker");
            }

            else
            {

```

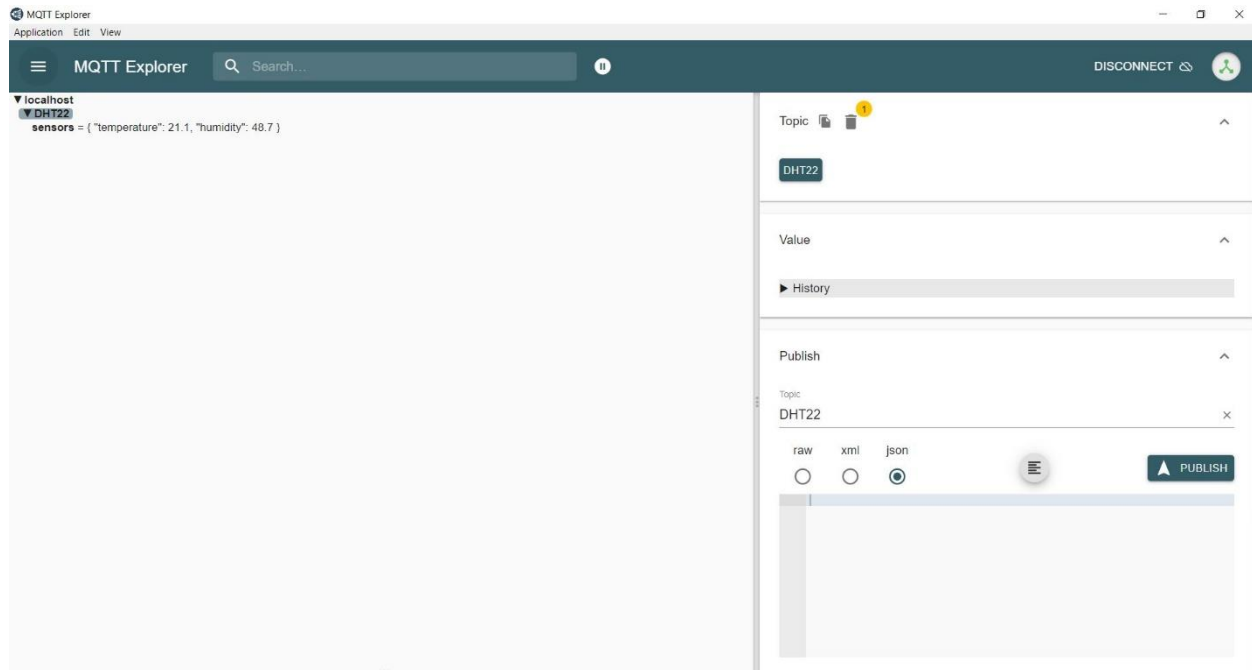
```

        //Serial.print("failed, rc=");
        //Serial.print(mqtt.state());
        //Serial.println("try again in 5 second");
        delay(5000);
    }
}
}

```

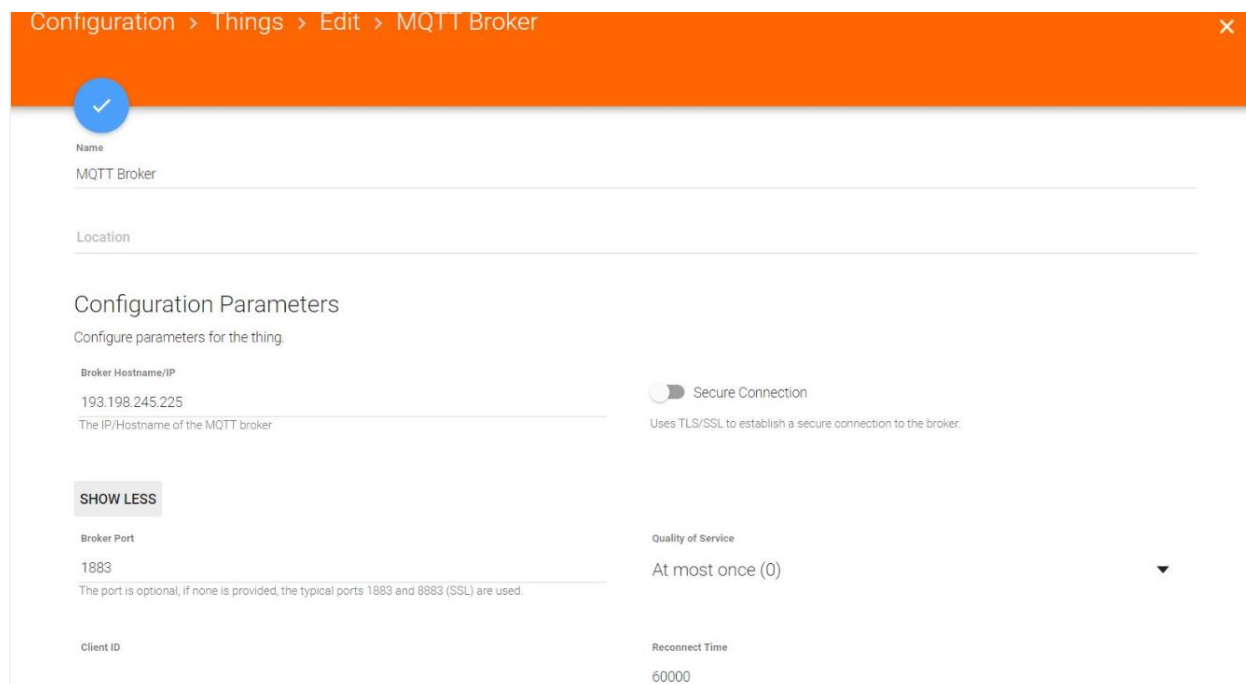
3.3. MQTT Explorer

MQTT-Explorer je desktop aplikacija za vizualizaciju podataka koji se šalju na MQTT broker, kod koje se pretplatimo na temu koja je “DHT22/sensors” (*Slika 3.4. MQTT-Explorer*). Podaci su prikazani u JSON format.



Slika 3.4. MQTT-Explorer.

konfiguracija provodi se uz pomoć OpenHAB Paper UI kojima korisnici mogu izmijeniti način prikaza podataka prema svojim specifikacijama.



Slika 4.2 Povezivanje s MQTT brokerom

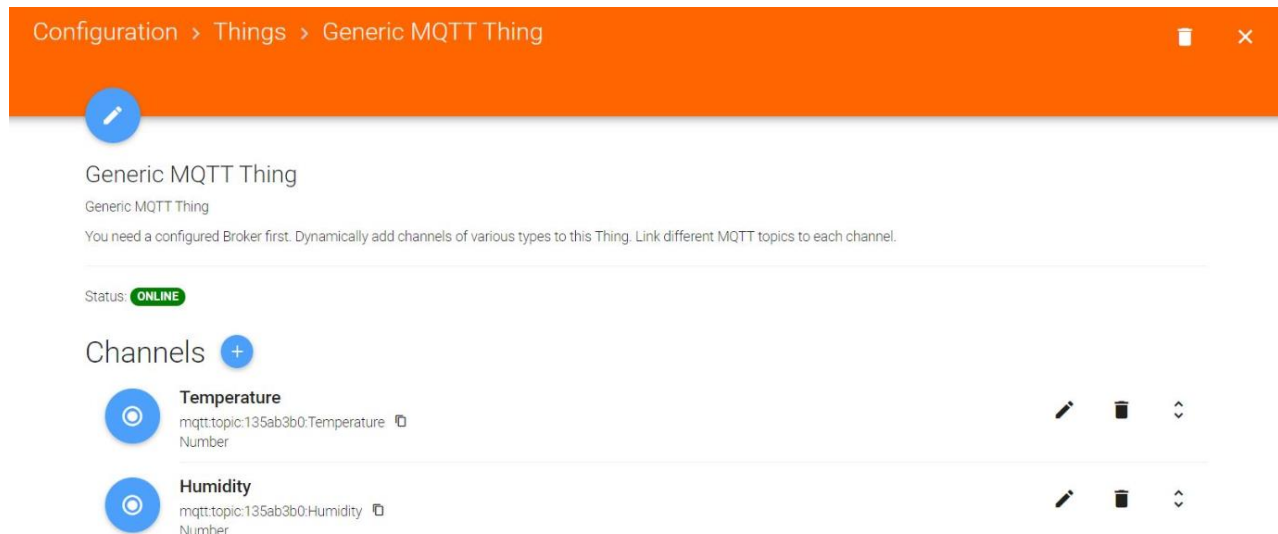
Bitno je uspostaviti vezu OpenHAB-a s lokalnim MQTT brokerom preko kojeg se izvršava prijenos zabilježenih podataka unošenjem IP (engl. *Internet Protocol*, *skraćeno IP*) adrese brokera u konfiguracijsku listu prikazano na slici 4.2.

Paper UI omogućuje:

- Poveznice (engl. *Bindings*), način povezivanja prikazanih podataka s izvorom gdje su podaci zabilježeni ili povezivanje kanala direktno s podacima koje želimo prikazati.
- Stvari (engl. *Things*), entiteti koji se mogu fizički dodati u sustav. To mogu biti uređaji, web protokoli ili drugi izvor informacije koji su bitni za uspostavu procesa.
- Kanali (engl. *Channels*), funkcija koja povezuje više entiteta iste vrste ili istog izvora koji opisuju funkcionalnost jednog sustava. Kanali su poveznice između

fizičkog i virtualnog sloja. U trenutku uspostavljanja veze podaci se prikazuju u realnom vremenu.

- Mostovi (engl. *Bridges*), vrsta entiteta potrebna za povezivanje s drugima entitetima. To mogu biti IP pristupnici (engl. *IP Gateway*) ili konfiguracije internet stranica s informacijama za autentikaciju.



Slika 4.3 Prikaz kanala u OpenHAB-u

Slika 4.3 prikazuje kanal koji povezuje dva entiteta: temperatura i vlažnost zraka.

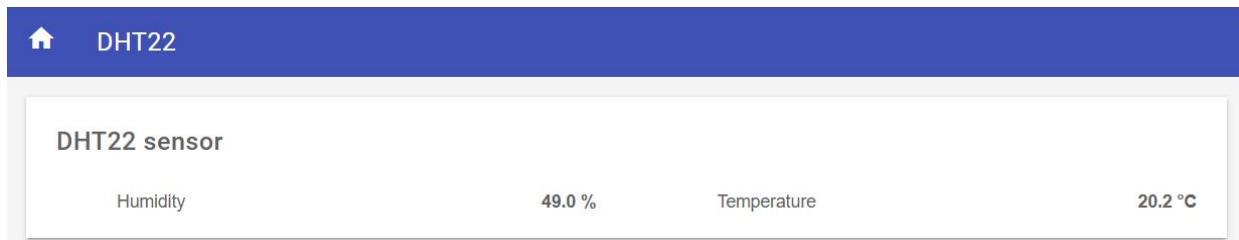
U projektu su temperatura i vlažnost navedeni u kanalu kao entiteti koji opisuju kvalitetu zraka u blizini senzora. Podatci o temperaturi i vlažnosti koji se šalju na OpenHab nalaze se u JSON formatu koje se pomoću JSON transformacije prebacuje u brojčanu vrijednost.

4.2. Basic UI

Basic UI dio je OpenHAB zaslužan za uređivanje sučelja internet stranice na kojoj se prikazuju poslani podaci. Za razliku od Paper UI, unosi promjene samo na vanjskom izgledu stranice gdje korisnici mogu po volji mijenjati estetski sadržaj. Basic UI omogućuje:

- Prilagodljiv izgled prikladan za različite veličine zaslona

- AJAX navigacija
- Ažuriranje



Slika 4.4 Basic UI izgled

Na slici 4.4 je prikazan izgled Basic UI koji omogućava prikaz vrijednosti podataka izmjerenih sa senzora.

Sitemaps u openHAB-u su zbirka Stvari i Predmeta koje predstavljaju fizičke ili logičke objekte u korisnikovim postavkama kućne automatizacije. Pomoću sitemap je kreiran izgled Basic UI, sitemap je bilo koji file s ekstenzijom .sitemap.

Kod unutar projekta koji se koristi za kreiranje Basic UI:

```
sitemap demo label="DHT22" {
  Frame label="DHT22 sensor" {
    Text item=GenericMQTTThing_Humidity label="Humidity [%.1f %%]"
    Text item=GenericMQTTThing_Temperature label="Temperature [%.1f °C"
  ]
}
```

5. ZAKLJUČAK

Prvo izdanje realiziranog projekta nije sadržavalo Arduino pločicu sposobnu za bežični prijenos podataka izmjerenih od strane DHT22 senzora već je bio potreban USB (engl. Universal Serial Bus) kabel za fizičko spajanje s uređajem kako bi se podaci mogli slati na OpenHAB. Zamjenom standardne Arduino pločice s pločicom s WiFi modulom, Arduino MKR 1010 WiFi, pokazalo se puno jednostavnije. Kabel je potreban jedino za napajanje pločice i proces prijenosa podataka odvija se jednostavnije.

Korištenjem privatnog MQTT servera, unosi novu razinu sigurnosti podataka. Za razliku od lokalnog (javnog) servera, privatni zauzima određeni dio memorije na uređaju no isplativo je radi očuvanja integriteta i povjerljivosti. Jednostavna vizualizacija očitanih podataka pomoću OpenHAB, gdje god da se senzor nalazi i mjeri podatke. Za slanje podataka na OpenHAB potreban je MQTT protokol jer ne postoji mogućnost izravnog povezivanja Arduina sa OpenHAB-om. Jedna od mana je što se Arduino mora nalaziti u radijusu neke WiFi mreže inače se neće moći povezati na WiFi, te neće slati podatke na MQTT broker.

6. LITERATURA

- [1] <https://www.openhab.org/docs/>
- [2] <http://www.steves-internet-guide.com/mqtt/>
- [3] <https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/>
- [4] <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>
- [5] <https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/>
- [6] <https://create.arduino.cc/projecthub/deodates/rssi-based-social-distancing-af0e26>
- [7] <https://store.arduino.cc/arduino-mkr-wifi-1010>
- [8] <https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/>
- [9] <https://www.adafruit.com/product/385>
- [10] <http://www.steves-internet-guide.com/mqtt-username-password-example/>