

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE

IZVJEŠTAJ
SENZOR UDALJENOSTI OD ODABRANE LOKACIJE

Bruno Rudan, Josip Sanković

Split, svibanj, 2020

Sadržaj

UVOD.....	3
KORIŠTENA OPREMA.....	4
NEO6M gps senzor.....	4
Arduino MKRWAN1300	5
LoRa radio modul	6
Korišteni softver	7
HEMA SPAJANJA.....	8
IZVEDBA KODA	9
Arduino kod.....	9
The Things Network (TTN)	13
Docker	15
REZULTATI I ZAKLJUČCI	17
LITERATURA	17

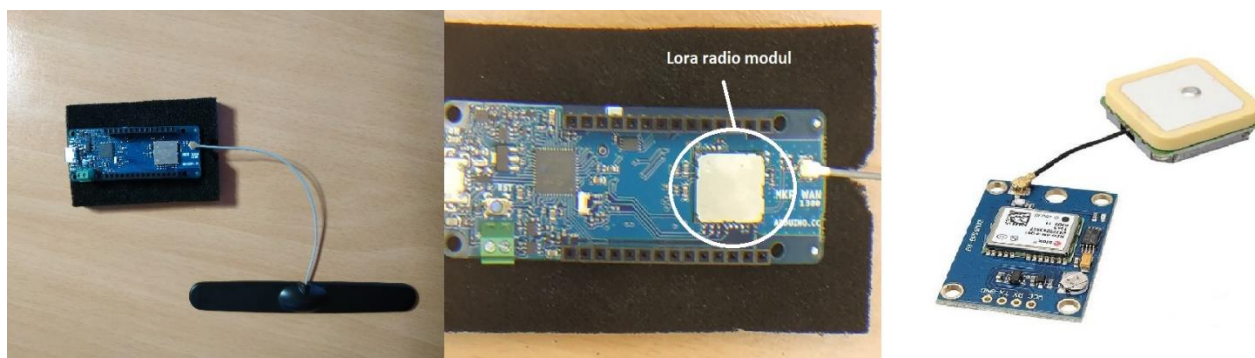
UVOD

Cilj ovog projekta je bila uspostava sustava obavijesti o udaljenosti između dvije osobe. U prvoj polovici izvještaja dajemo svoj osvrt na korištenu opremu, njihove prednosti i mane ali i izazove s kojima smo se suočili u njihovom osposobljavanju. U drugoj polovici dajemo konkretne primjere koda koji je korišten za projekt i dajemo objašnjenje njegove svrhe, pored toga dajemo pregled rada TTN servisa i obrade podataka. Projekt je prošao kroz mnoge iteracije ali smo se naposljetku odlučili za sljedeću generalnu izvedbu. Koristimo GPS senzor za položaj jedne osobe i pred unesene koordinate za položaj druge. Obavlja se izračun udaljenosti. Zatim, obavijest se šalje LoRa modulom ugrađenim u Arduino serije MKR na FESB Gateway, koji prosljeđuje podatke na The Things Network. Korištenjem Docker aplikacije i njenih servisa kao što su Node-RED, InfluxDB i Grafana prikazuje se graf udaljenosti GPS senzora od pred unesene koordinate.

KORIŠTENA OPREMA

NEO6M gps senzor

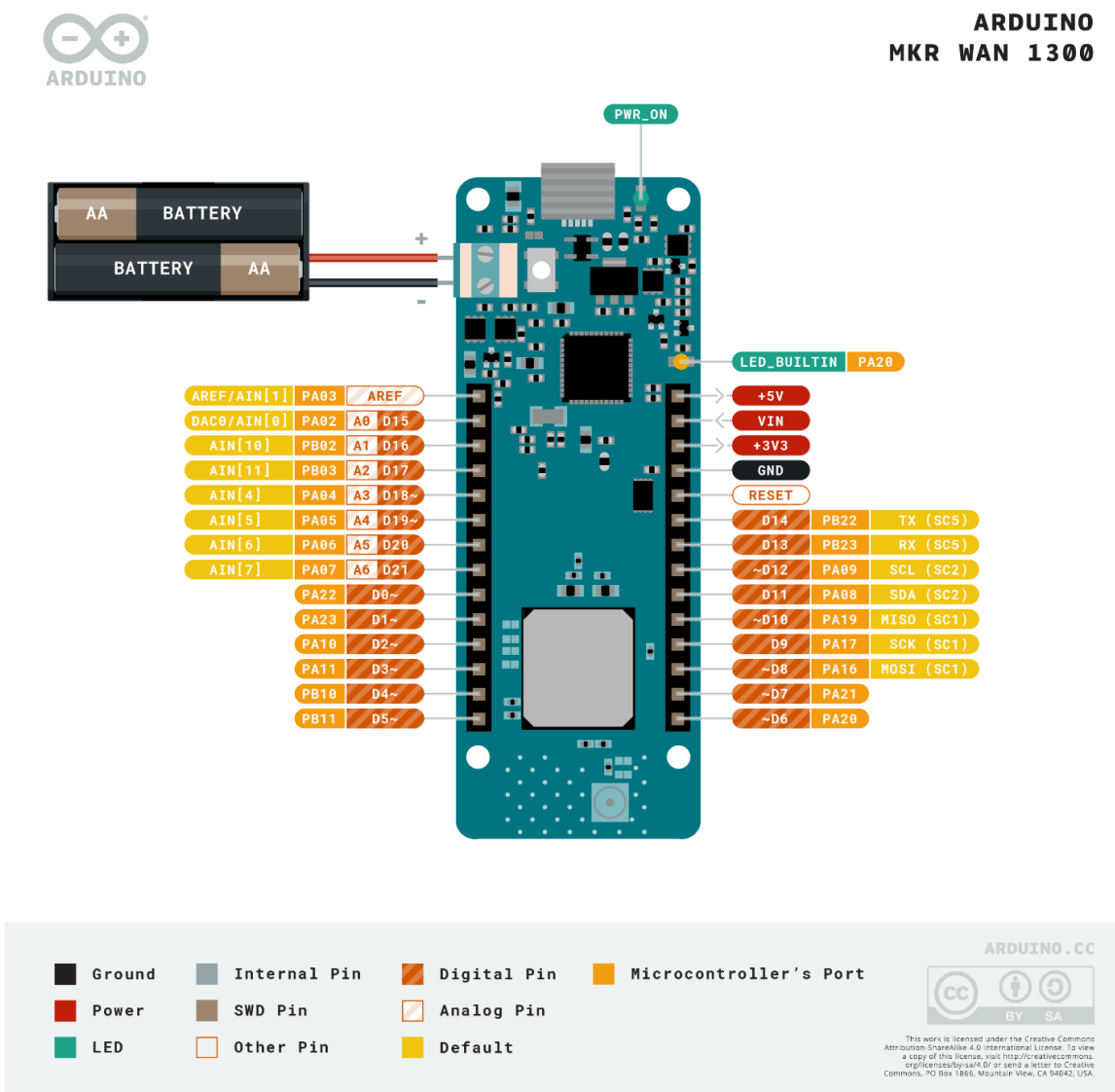
Kao GPS senzor je korišten NEO6M proizvođača Ublox. GPS tehnologija pretpostavlja pristup nebeskom svodu tj. satelitima. Najmanje 4 su potrebna za triangulaciju položaja. Zbog toga, mjerenja je poželjno obavljati na vedrije dane zbog opće funkcionalnosti ali i zbog manje mogućnosti refleksije signala tj. krivog očitavanja položaja. Položajna preciznost NEO6M senzora je 2.5m. To znači da je za vrijeme testiranja utvrđeno da se položajne koordinate nalaze u krugu od 2.5m, 50% vremena(CEP 50%). Takva razina preciznosti znači da je GPS senzorima nemoguće pouzdano računati manje udaljenosti, jer će senzor u mirovanju za pola podataka kojih pošalje pogriješiti za 2.5m. Za usporedbu, GPS tehnologija koja se koristi za geodetske svrhe ima približnu točnost od 2cm što je 100 puta točnije od NEO6M. Na prvi pogled to možda ne zvuči ohrabrujuće za NEO6M, ali taj nedostatak preciznosti je ono što zamjenjujemo za lakše rukovanje i mobilnost, jer precizne uređaje bi bilo nemoguće upotrijebiti za ovakvu svrhu. Usput je potrebno naglasiti i cijene tih preciznijih uređaja koje su po prosjeku 400 puta skuplje od NEO6M.



Slika 1. MKRWAN1300(lijevo), LoRa radio modul(sredina), NEO6M Gps senzor(desno)

Arduino MKRWAN1300

Za slanje podataka i upravljanje GPS senzorom korišten je Arduino MKR WAN 1300 koji ima ugrađen LoRa modul. MKR WAN 1300 je napravljen da pruži praktična i jeftina rješenja za izrađivače koji žele povezivost pomoću Lo-Ra modula. Moguće ga je napajati sa dvije 1.5 AA ili AAA baterije ili iz vanjskog napajanja od 5V. Arduino MKR WAN 1300 se spaja na računalo pomoću USB-a, a s Arduino strane preko COM 12 porta. Preko računala prebacimo kod s računala na Arduino. Na Arduino možemo spojiti senzore, diode, releje i još dodatne module koji su poznatiji pod nazivom shields.



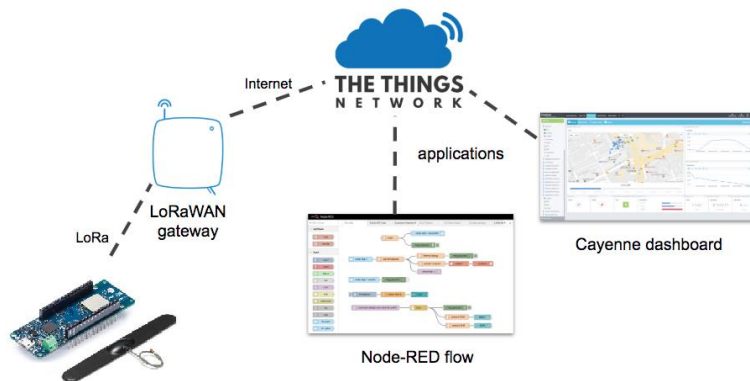
Slika 2.Shema pinova mikrokontrolera

LoRa radio modul

LoRa(Long Range) predstavlja fizički sloj radio komunikacije preko velikih udaljenosti, koristeći LPWAN(low-power wide-area network) protokol za komunikaciju. Podatke koje šalje modul sakuplja „Gateway“ te se podaci dalje šalju na „TheThingsNetwork“ koji pruža LPWAN mrežni server. Ugrađeni LoRa radio modul(CMWX1ZZABZ) posjeduje certificirana radio-regulatorna odobrenja za rad u 868 i 915 MHz ISM spektru. Stoga možemo pretpostaviti manju zagušenost komunikacijskog kanala. Procjenjeni domet modula je 2-3km u urbanoj sredini iako domet veoma ovisi o okruženju. Pokušaj spajanja u četvrti „Plokite“, približno 1.5km od „Gateway-a“ na FESB-u je bio neuspješan, stoga se može zaključiti da u spajanju obitava i element sreće kako se uređaj nebi našao u „mrtvoj zoni“.



Slika 3. Gateway(FESB)

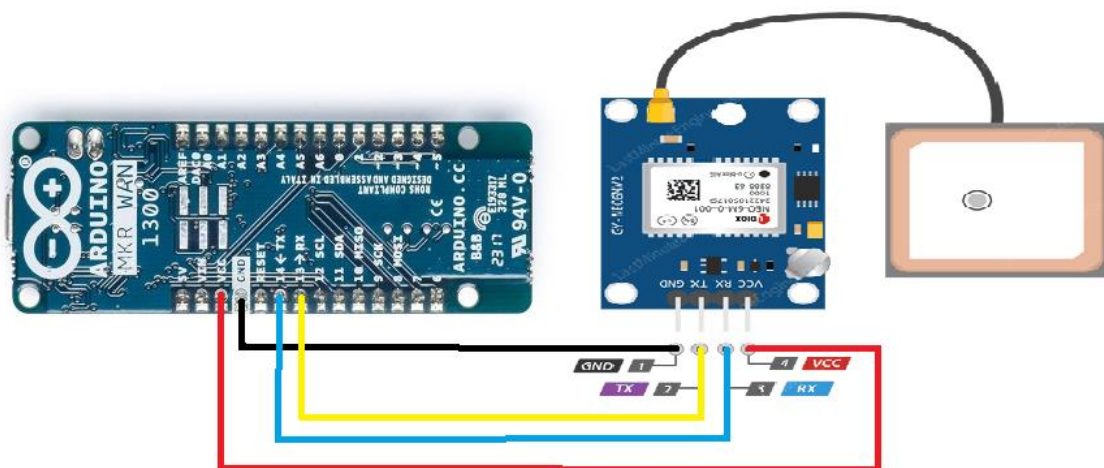


Slika 4.Arhitektura sustava

Korišteni softver

Visual Studio Code s Platformio IDE razvojnim okruženjem. Korišten za pisanje, kompajliranje i učitavanje koda na Arduino. Operativni sustav iz prve nije prepoznavao Arduino, stoga je uz to bilo potrebno instalirati Arduino IDE kako bi se mogao instalirati driver softver za MKR WAN 1300. Nakon toga problem sa spajanjem se još uvijek pojavljivao, ali s umanjenom učestalošću. Uz to korišten je i Docker container sa NodeRed, influxDB i Grafana servisima.

SHEMA SPAJANJA



Slika 4. Spajanje senzora

Vcc pin Arduina se spaja na Vcc pin NEO6M senzora, prikazano crvenom žicom. Taj pin dovodi 3.3V, što je unutar dozvoljenog napona napajanja senzora(2.7V-3.6V). GND pin Arduina se spaja na GND pin senzora, prikazano crnom žicom. Serijska komunikacija se obavlja preko serijskog porta broj 1 Arduina(pinovi 13 i 14). TX(14) pin arduina se spaja na RX pin senzora odnosno RX(13) na TX, prikazano plavom i žutom žicom.

IZVEDBA KODA

Arduino kod

Za izradu projekta koristili smo Visual Studio Code te smo instalirali odgovarajuće biblioteke da možemo upravljati našim mikrokontrolerom i GPS modulom.

```
#include <MKRWAN.h>
#include <mainGPS.h>
#include <TinyGPS++.h>
#include "arduino_secrets.h"
```

Svako spajanje na TTN(TheThingsNetwork) se vrši preko OTAA. Aktivacija preko zraka (OTAA) je najsigurniji način povezivanja s TTN serverom. Uređaj izvodi postupak spajanja s mrežom pri čemu se dogovaraju sigurnosni ključevi. Ti ključevi su pohranjeni u „arduino_secrets.h“ header datoteci.

```
#define SECRET_APP_EUI "70B3D57ED002F16B"
#define SECRET_APP_KEY "11FEAF9B62C323BF77308AF2BADFF947"
```

U Setup() funkciji prvo postavljamo brzinu slanja podataka na Serial Monitor te zatim postavljamo brzinu slanja i primanja podataka sa GPS modula pomoću Serial1. GPS modul smo spojili na pinove 13 i 14 jer su to Serial1 pinovi za RX i TX. Nakon postavljanja GPS modula postavljamo LoRa modul na odgovarajuću frekvenciju(u našem slučaju EU868) ,te varijablu connected koja nam označava dali smo se spojili na Fesb-ov gateway postavljamo na false.

```
void setup() {
  // Start the Arduino hardware serial port at 9600 baud
  Serial.begin(9600);

  // Start the software serial port at the GPS's default baud
  Serial1.begin(GPSBaud);
  modem.begin(EU868);
  delay(1000); // apparently the murata dislike if this tempo is removed...
  connected=false;
  err_count=0;
}
```

U glavnom djelu koda odnosno loop() funkciji prvo provjeravamo dali je GPS poslao ikakve informacije na Serial1 port te ako je pozovemo funkciju displayInfo() . Ako je prošlo 5 sekundi a ništa nije primljeno na Serial1 portu tada nam prikaže da GPS nije pronađen.

```
// This sketch displays information every time a new sentence is correctly encoded.
while (Serial1.available() > 0)
  if (gps.encode(Serial1.read()))
    displayInfo();

// If 5000 milliseconds pass and there are no characters coming in
// over the software serial port, show a "No GPS detected" error
if (millis() > 5000 && gps.charsProcessed() < 10)
{
  Serial.println("No GPS detected");
  while(true);
}
```

Ako je GPS poslao neke podatke Arduinou poziva se funkcija displayInfo() koja nam sprema geografsku širinu i dužinu GPS modula odnosno osobe koja ga nosi.

```
void displayInfo()
{
  if (gps.location.isValid())
  {
    osoba.duzina=gps.location.lng();
    osoba.sirina=gps.location.lat();

    distance(osoba,lab);

  }
  else
  {
    Serial.println("Location: Not Available");
  }
}
```

Zatim pozivamo funkciju za udaljenost kojoj šaljemo argumente lokacije osobe i lokacije naše odabrane točke. Udaljenost između te dvije točke izračunamo jednostavno preko Pitagorinog poučka, a udaljenost na kraju dobijemo u metrima.

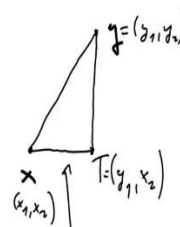
Udaljenost

Dvije točke u \mathbb{R}^2 :

$$\mathbf{x} = (x_1, x_2)$$

$$\mathbf{y} = (y_1, y_2)$$

Njihova udaljenost:



$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$= \sqrt{\sum_{i=1}^2 (x_i - y_i)^2}$$

Slika 5. Formula za udaljenost dviju točaka

Udaljenost dviju točaka spremamo u varijablu msg pomoću koje ćemo slati podatke na TTN tako da u msg[0] stavljamo pola varijable udaljenost a u msg[1] drugu polovicu.

```
void distance(location osoba, location lab){

    double sirina=(osoba.sirina-lab.sirina)*100000;
    double duzina=(osoba.duzina-lab.duzina)*100000;
    uint32_t udaljenost=sqrt(duzina*duzina+sirina*sirina);
    msg[0]=highByte(udaljenost);
    msg[1]=lowByte(udaljenost);

    Serial.println(udaljenost);

}
```

Kad smo učitali podatke sa GPS modula, u glavnom dijelu koda vrši se potvrda o spajanju u kojoj se pozivaju sigurnosni ključevi. U ovom djelu koda postavljamo data Rate pomoću kojeg govorimo modemu da nam šalje podatke brzinom od 5 470 bit/s, a Spreading factor(SF) je 7. Povećavanjem SF-a podatci se sporije šalju ali mogu prijeći veću udaljenost.

```

if ( !connected ) {
    int ret=modem.joinOTAA(appEui, appKey);
    if ( ret ) {
        connected=true;
        modem.minPollInterval(60);
        modem.dataRate(5);    // switch to SF7
        delay(100);           // because ... more stable
        err_count=0;
    }
}

```

Ako smo se spojili sa Fesb gateway-om pocinjemo slati poruku odnosno varijabliu msg pomoću funkcije modem.write(msg,2) i zatim čekamo potvrdu dali je paket primljen. Ako paket nije primljen u više od 50 pokušaja ponovno uspostavljamo vezu sa Fesb-om, a ako je primljen idemo sve iz početka.

```

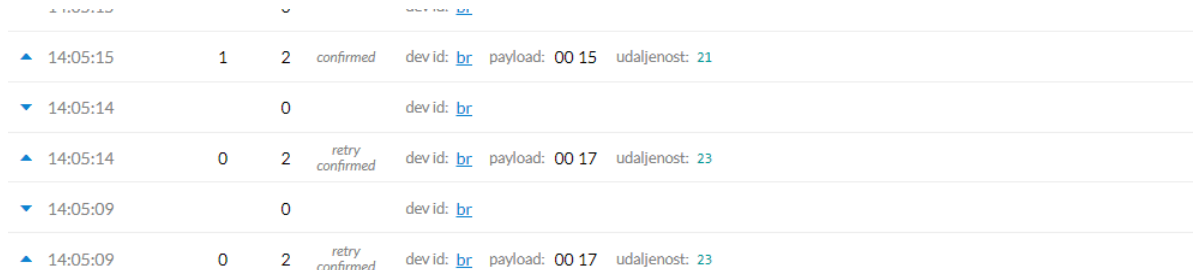
if ( connected ) {

    int err=0;
    modem.beginPacket();
    modem.write(msg,2);
    err = modem.endPacket(true);
    if ( err <= 0 ) {
        // Confirmation not received - jam or coverage fault
        err_count++;
        if ( err_count > 50 ) {
            connected = false;
        }
        // wait for 2min for duty cycle with SF12 - 1.5s frame
        // for ( int i = 0 ; i < 120 ; i++ ) {
        //     delay(1000);
        // }
    } else {
        err_count = 0;
        // wait for 10s for duty cycle with SF7 - 55ms frame
        delay(10000);
    }
}
}

```

The Things Network (TTN)

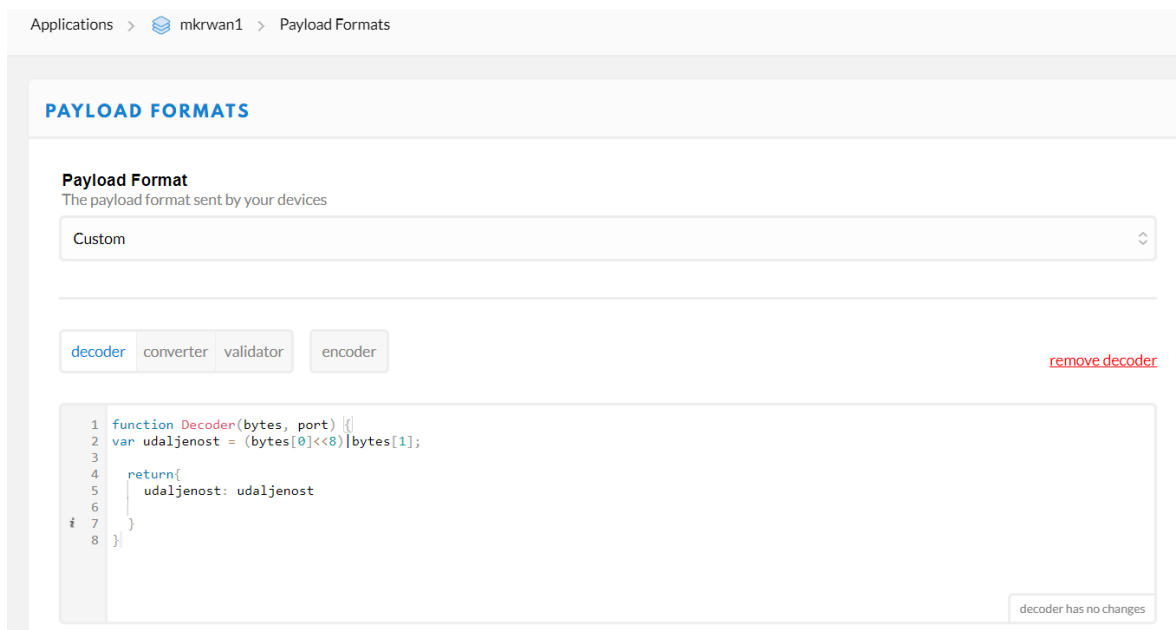
Podatci poslani preko Arduina na Fesb-ov gateway šalju se na TTN gdje se primaju kao binarni podatci te ih TTN već dekrptira tako da prikaže brojke gdje je svaki byte predstavljen sa dvije hexadecimalne znamenke.



▲ 14:05:15	1	2	confirmed	dev id: br	payload: 00 15	udaljenost: 21
▼ 14:05:14	0			dev id: br		
▲ 14:05:14	0	2	retry confirmed	dev id: br	payload: 00 17	udaljenost: 23
▼ 14:05:09	0			dev id: br		
▲ 14:05:09	0	2	retry confirmed	dev id: br	payload: 00 17	udaljenost: 23

Slika 6. Prikaz primljenih podataka u hexadecimalnom zapisu

Taj payload moramo sada dekodirati iz hexadecimalnog zapisa u nama čitljivi a to radimo pomoću funkcije unutar TTN servisa koja se zove Payload Formats.



Applications > mkrwan1 > Payload Formats

PAYLOAD FORMATS

Payload Format
The payload format sent by your devices

Custom

decoder converter validator encoder

[remove decoder](#)

```
1 function Decoder(bytes, port) {
2   var udaljenost = (bytes[0]<<8)|bytes[1];
3
4   return {
5     udaljenost: udaljenost
6   }
7 }
8 }
```

decoder has no changes

Slika 7. Funkcija dekodera

Ovdje uzimamo svaki pojedini byte koji smo poslali sa arduina i pretvaramo ga u jednu varijablu koju vraćamo kao JSON objekt, a program pišemo u JavaScript jeziku. Valjanost napisane

funkcije možemo provjeriti tako što upišemo proizvoljni hexadecimalni broj (u našem slučaju sa 4 znamenke odnosno 2 byte-a), a ispod nam ispiše rezultat naše konverzije.

Payload

01 14 2 bytes 1 Test

```
{  
  "udaljenost": 276  
}
```

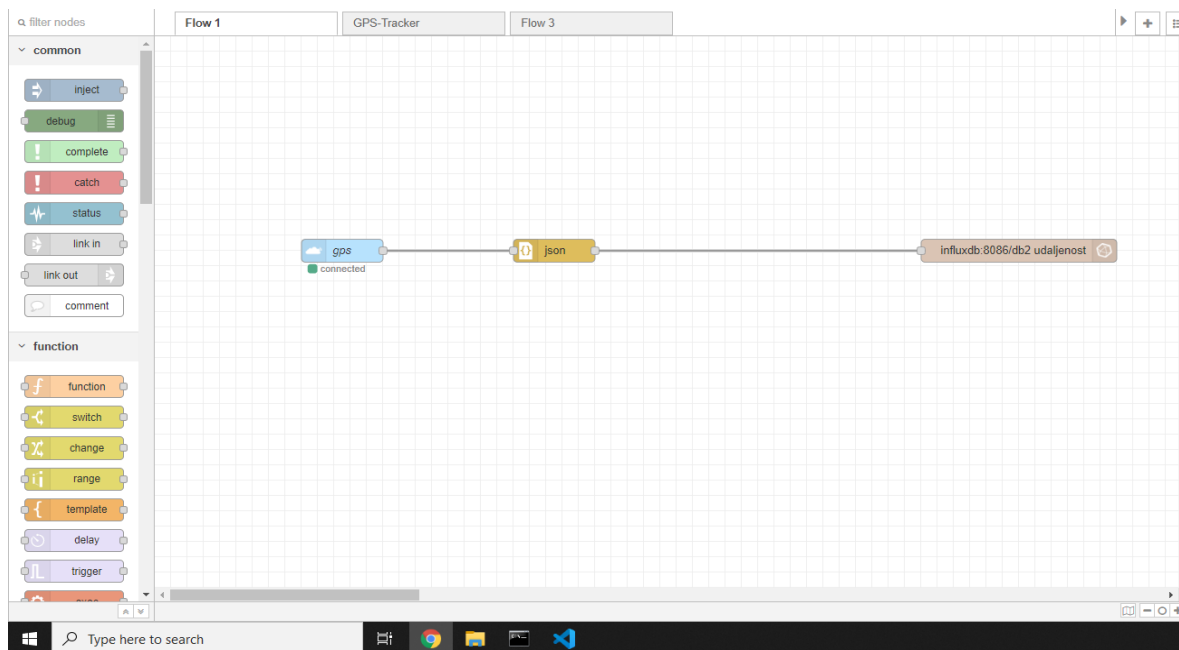
● Payload was valid

Cancel no changes to save Activate

Slika 8. Test funkcije za određeni payload

Docker

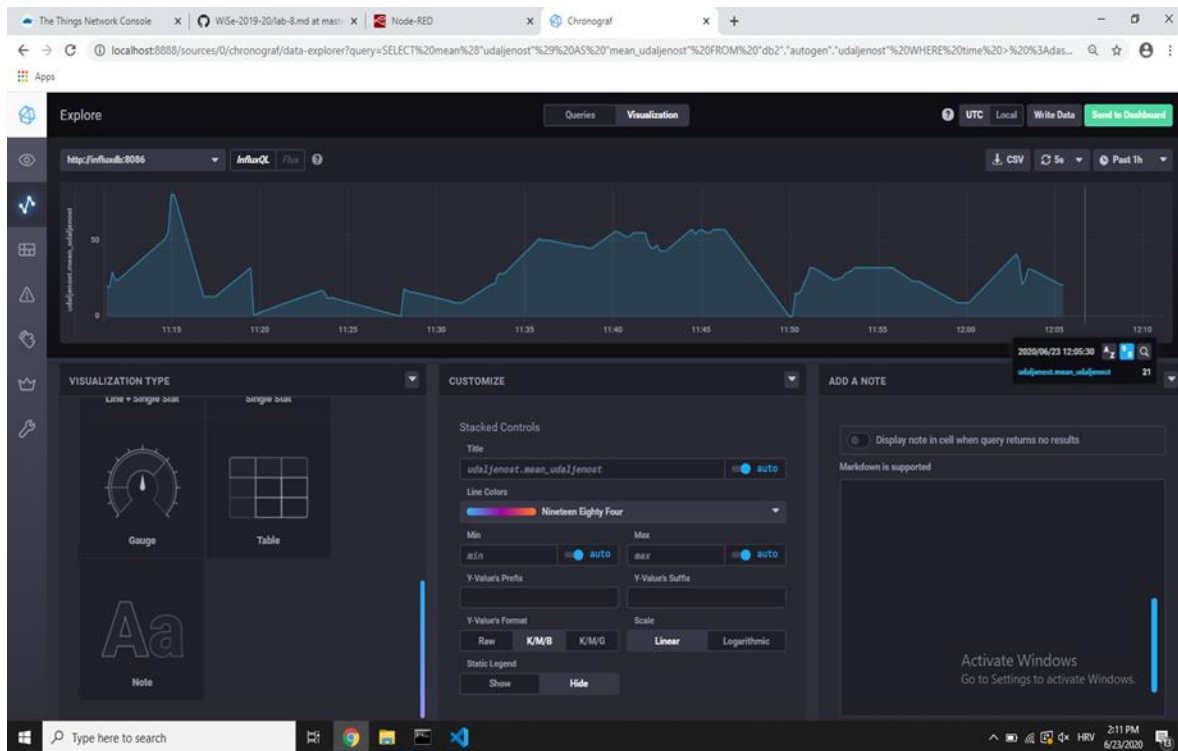
U Dockeru koristimo NodeRed da bi smo preuzeli podatke, odnosno JSON objekt koji sadrži varijablu udaljenost te je konvertiramo u JavaScript objekt koji možemo slati u našu bazu podataka db2 unutar InfluxDb-a.



Slika 9. NodeRed

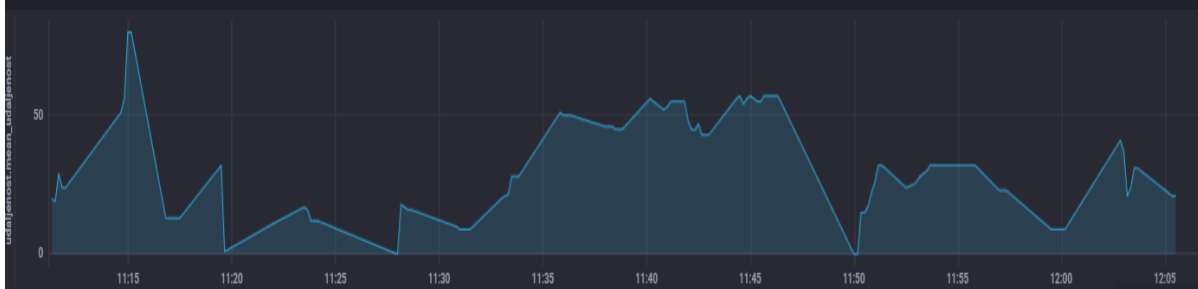
Za spajanje na TTN i InfluxDb moramo instalirati posebne pluginove.

Kad smo spojili sve nodove u NodeRed-u idemo na Chronograf i kreiramo novu bazu podataka pod nazivom db2 na koju će se slati udaljenost. Za vizualizaciju udaljenosti u ovisnosti o vremenu kliknemo na karticu unutar menu-a pod nazivom explore i u submenu-u odaberemo db2 i varijablu udaljenost te ih vizualiziramo.



Slika 10. Vizualizacija primljenih podataka

REZULTATI I ZAKLJUČCI



Slika 11. Grafički prikaz udaljenosti gps senzora od odabrane točke

Promatrajući graf vidimo da senzor ne očitava uvijek točnu vrijednost i ako smo u pokretu točnost mu se smanjuje (npr. Vrijeme od 11:15 do 11:20). Ako smo stacionarni, gps dosta preciznije očitava lokaciju.

Na grafu možemo primjetiti da u određenim trenucima vrijednost udaljenosti padne na nula. Pretpostavljamo da je razlog tomu gubitak signala sa satelitom ili kada se gps senzor resetira. Senzor nije baš pozdan ali zbog svoje niske cijene i jednostavnog korištenja njegove usluge su na prihvatljivoj razini. Također iz nepoznatih razloga nakon određenog vremena senzor prestane slati podatke te tada moramo restartati Arduino.

LITERATURA

<https://store.arduino.cc/arduino-mkr-wan-1300-lora-connectivity-1414>

<https://www.thethingsnetwork.org/docs/lorawan/addressing.html>

<https://wireless.murata.com/type-abz-078.html>

[https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf)

<https://discourse.nodered.org/t/passing-json-from-mqtt-into-influxdb/14563/8>