FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Parking senzor

Ana Šundov

Petra Šiklić

Smjer: Komunikacijska i informacijska tehnologija (114)

Split, lipanj 2018.

SADRŽAJ:

[1. UVOD 1](#_Toc516422630)

[2. ULTRAZVUČNI SENZOR HC-SR04 2](#_Toc516422631)

[2.1. Općenito 2](#_Toc516422632)

[2.2. Spajanje senzora s Arduinom 3](#_Toc516422633)

[3. PREDAJNIK/PRIJEMNIK 4](#_Toc516422634)

[3.1. Općenito 4](#_Toc516422635)

[3.2. Programski kod za predajnik 6](#_Toc516422636)

[3.2.1. Sensors.h 6](#_Toc516422637)

[3.2.2. Sensors.cpp 6](#_Toc516422638)

[3.2.3. Radio\_nRF.h 7](#_Toc516422639)

[3.2.4. Radio\_nRF.cpp 8](#_Toc516422640)

[3.2.5.Network\_ping.h 10](#_Toc516422641)

[3.2.6. Network\_ping.ino 10](#_Toc516422642)

[3.2.7. Printf.h 13](#_Toc516422643)

[3.3. Programski kod za prijemnik 14](#_Toc516422644)

[4. MQTT 15](#_Toc516422645)

[4.1.Općenito 15](#_Toc516422646)

[4.2.Postavke 15](#_Toc516422647)

[5. PYTHON 17](#_Toc516422648)

[6. ZAKLJUČAK 18](#_Toc516422649)

[LITERATURA 19](#_Toc516422650)

[POPIS SLIKA I TABLICA 20](#_Toc516422651)

# 1. UVOD

Senzor ili pretvornik je uređaj koji mjeri fizikalnu veličinu (npr. temperaturu, vlažnost zraka, tlak, broj okretaja motora) i pretvara je u signal pogodan za daljnju obradu.[1] Dijele se prema načinu rada, složenosti, vrsti izlaznog signala te prema načinu prikaza signala.

Prema načinu rada razlikujemo:

* Kontaktne koji moraju biti u kontaktu s objektom mjerenja (npr. termometar)
* Beskontaktne koji nisu u kontaktu s objektom mjerenja (npr.radar).

Prema složenosti senzori mogu biti:

* Samostalne jedinice kod kojih je odmah omogućeno očitanje sa izlaza
* Nesamostalne gdje izlaz senzora treba prilagoditi prije upotrebe.

S obzirom na izlazni signal senzora razlikujemo:

* Električni signal koji može biti binarni, pulsni ili analogni (sa ili bez pojačala)
* Neelektrični signal (npr. tlak, zrak, svjetlina).

Zadnja podjela odnosi se na način prikaza signala gdje razlikujem

* Binarni senzore koji pretvaraju mjerenu fizikalnu veličinu u binarni signal, primjerice senzor blizine, pritiska ili temperature.
* Analogne senzore koji pretvaraju fizikalnu veličinu u analogni signal, primjerice senzor sile, akustike ili udaljenosti.[1]

Senzori moraju biti podešeni prema unaprijed utvrđenim standardima.

Primjenjuju se u svakodnevnim predmetima, automobilima, letjelicama, medicini. [2]

# 2. ULTRAZVUČNI SENZOR HC-SR04

## 2.1. Općenito

Ultrazvučni modul HC-SR04 je senzor za mjerenje udaljenosti između njega i predmeta koji se nalazi ispred njega, pomoću ultrazvučnih valova.

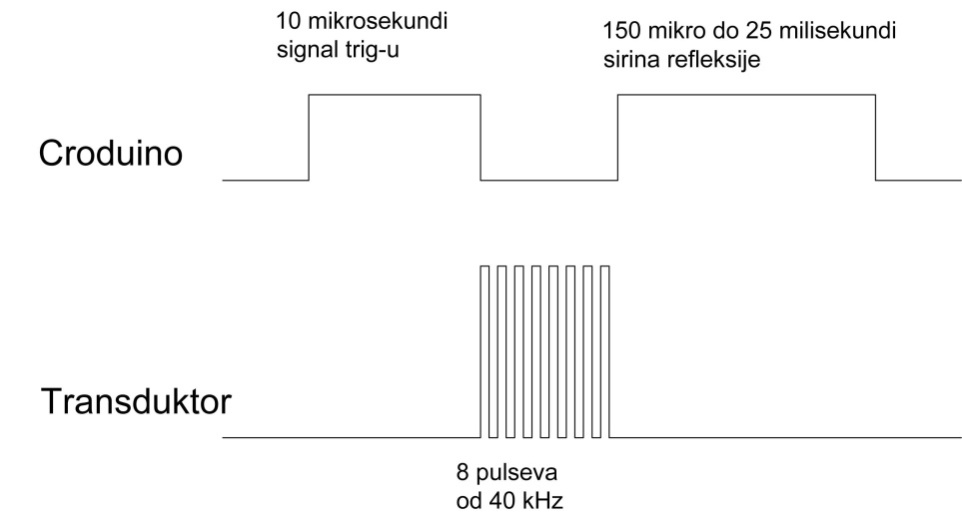


##### Slika 1. Ultrazvučni modul HC-SR04 [3]

Karakteristike ovog senzora su:

1. Domet: 2-200 cm
2. Preciznost: 3 mm
3. Efektivni kut mjerenja: 15**°**
4. Napon: 5V
5. Maksimalna mirna struja: 2mA
6. Radna struja: 15mA
7. Ultrazvučna frekvencija: 40kHz
8. Dimenzija: 45x20x15 mm

Dva osnovna dijela modula na kojima se temelji princip rada su prekidač ( trig ) i refleksija ( echo ). Mikrokontrolerom (Arduinom ili Croduinom) šaljemo 5V na trig pin modula u trajanju od minimalno 10 mikrosekundi. Na taj način aktiviramo ultrazvučni pretvarač koji odašilje 8 impulsa od 40 kHz i čeka njihovu refleksiju. Kada senzor registrira reflektirani impuls šalje podatke natrag mikrokontroleru preko echo pina. Trajanje reflektiranog impulsa je od 150 mikro do 25 milisekundi. Ako „jeka“ traje duže od 35 milisekundi, senzor registrira da je predmet izvan dosega. [5]



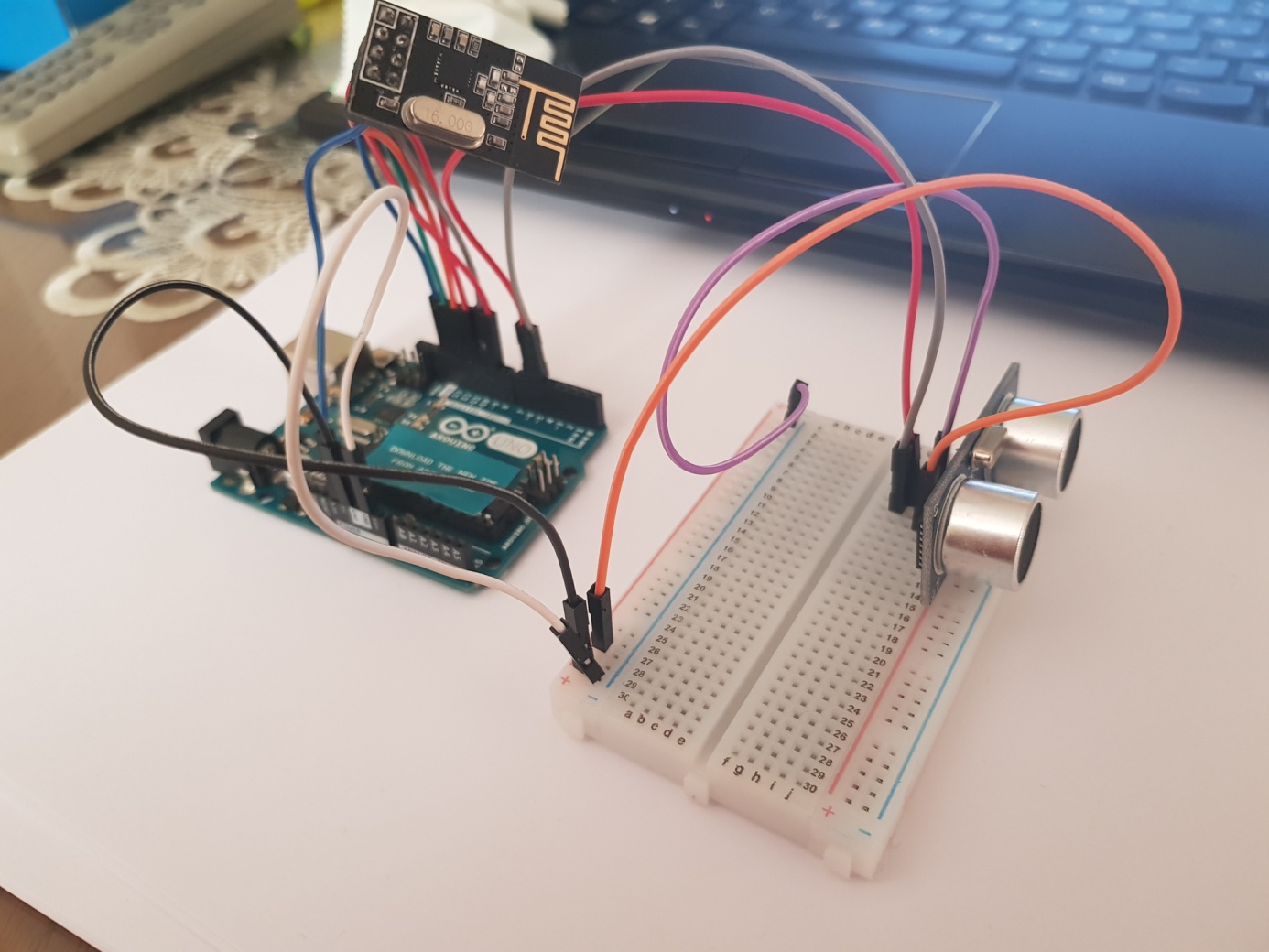
##### Slika 2. Način rada ultrazvučnog modela HC-SR04 [4]

## 2.2. Spajanje senzora s Arduinom

Arduino je ime za otvorenu računalnu i softversku platformu koja omogućava dizajnerima i konstruktorima stvaranje uređaja i naprava koje omogućuju spajanje računala s fizičkim svijetom. Arduino je stvorila talijanska tvrtka SmartProject 2005. godine, rabeći 8-bitne mikrokontrolereAtmel AVR.

Postoji nekoliko izvedbi Arduina među kojima su ArduinoDiecimila, Arduino Leonardo, Arduino Mega te Arduino Uno. [6]

Ultrazvučni modul sadrži 4 izlaza. Napajanje od 5V dovodi se na + pol, a GND na – pol pločice. Trig pin se spaja na Arduino, točnije na pin 5, a echo pin na pin 6.

Slika 3. Spajanje HC-SR04 senzora i transmittera

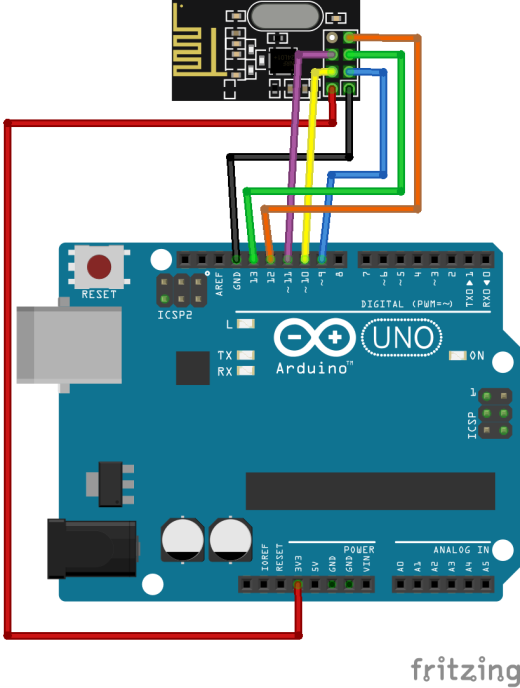
# 3. PREDAJNIK/PRIJEMNIK

## 3.1. Općenito

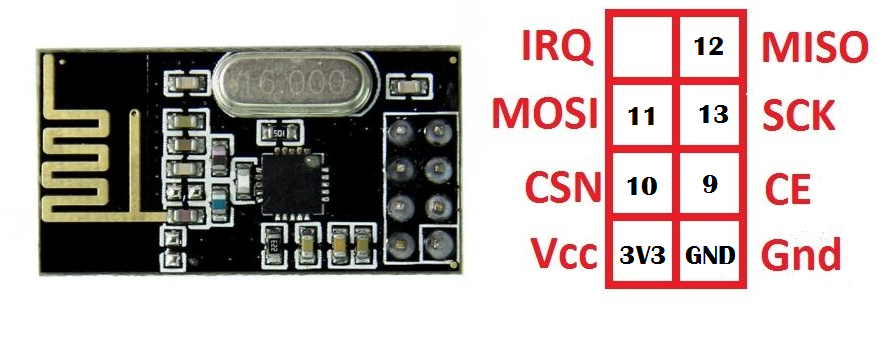
Predajnik ili odašiljač je elektronski uređaj koji stvara električne oscilacije visoke frekvencije, pojačava ih i modulira korisnim signalom informacije te zrači u slobodni prostor. [8]

Prijemnik je elektronski uređaj koji služi da primi poruku prenesenu preko nekog medija i izdvoji koristan signal informacije iz nje, odnosno provodi se postupak demodulacije. Sastoji se od pojačivača, oscilatora, filtara te demodulatora.

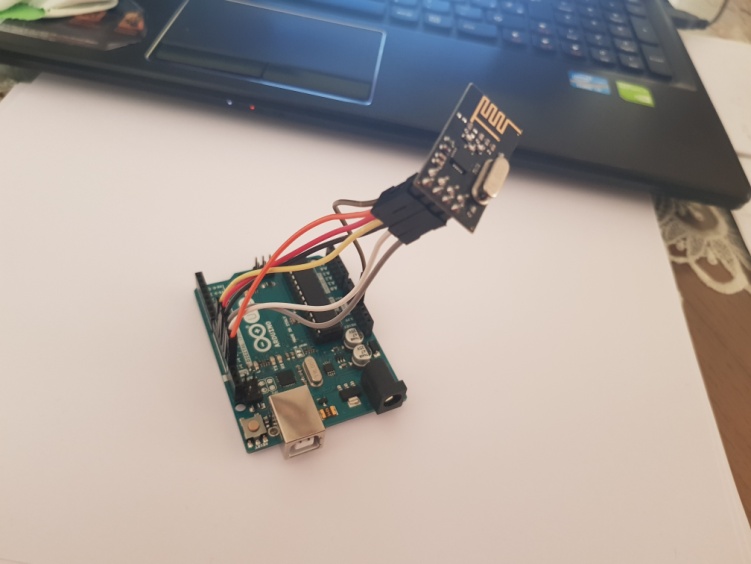
U ovom projektu korišten je nRF2401 primopredajnik koji radi na visokim frekvencijama od 2,4 do 2,5 GHz. Trenutna potrošnja je vrlo niska, oko 10,5 mA, pri izlaznoj snazi od -5dBm te 18 mA u postupku prijema.



##### Slika 4. Prikaz prijemnika, odnosno predajnika [11]



##### Slika 5. Prikaz spajanja pinova prijemnika/predajnika naArduino [12]

**

##### Slika 6. Spojeni prijemnik/predajnik

Kako bismo stvorili komunikaciju između predajnika i prijemnika, kao i napravili samu funkciju izračunavanja udaljenosti objekta, poslužili smo se unaprijed definiranim programima i bibliotekama koje smo modificirali kako bi funkcionirale za naš projekt. [15] [16]

Kod je pisan u C++ jeziku koji je objektno programski jezik i omogućuje bolju čitljivost odvajanjem deklaracije i implementacije klase tako da definiramo članske funkcije izvan same klase u .cpp datoteku istog imena kao i klasa kojoj pripadaju (Sensors.cpp). Isto radimo s definicijama klase koje spremamo u .h (header) datoteku istog imena kao klasa (Sensors.h). Ovakav pristup omogućava manju potrošnju memorije i olakšava izradu biblioteka.

## 3.2. Programski kod za predajnik

### 3.2.1. Sensors.h

Biblioteka stvorena za odvojeno definiranje klase Sensors i definiranja njene funkcije *distance* koja vraća float vrijednost izračunate udaljenosti objekta.

*#ifndefSensors\_h // provjeravamo je li još negdje definirana biblioteka*

*#defineSensors\_h // definiranje biblioteke Sensors\_h*

*#include<Arduino.h> //uključivanje biblioteke Arduino.h*

*#include<Adafruit\_Sensor.h> // uključivanje biblioteke Adafruit\_Sensor.h*

*#defineinputPin 2 // // definiranje ulaza 2*

*class SENSORS { // stvaranje klase SENSORS*

*public:*

*SENSORS(); //definiranje konstruktora SENSORS*

*~SENSORS(); // definiranje destruktora SENSORS*

*float distance(); //definiranje varijable distance tipa float*

*}*

*#endif // završetak definiranja*

### 3.2.2. Sensors.cpp

Poziva se funkcija *distance* klase Sensors i izračunava udaljenost objekta. Funkcija prvo čisti trigPin od prethodnih zaostalih vrijednosti, postavlja ga na HIGH stanje 10 mikrosekundi i nakon ga vraća na LOW stanje. Očitava vrijednost echoPin-a koji mjeri vremenski interval između slanja i primanja vala. Nakon toga se izračunava vrijednost udaljenosti objekta i ispisuje ga se u centimetrima na serijski monitor.

*#include "Sensors.h" //definiramo biblioteku*

*#define IN4 4*

*const int trigPin = 5;*

*const int echoPin = 6;*

*SENSORS::SENSORS(){}*

*SENSORS::~SENSORS(){}*

*float SENSORS::distance(){*

*long duration;*

*int distance = 0;*

*delayMicroseconds(10);*

*digitalWrite(trigPin, LOW); // Čisti trigPin od prethodnih vrijednosti*

*delayMicroseconds(2);*

*digitalWrite(trigPin, HIGH); // Postavlja trigPin na HIGH stanje na 10 mikrosekundi*

*delayMicroseconds(10);*

*digitalWrite(trigPin, LOW);*

*duration = pulseIn(echoPin, HIGH); // Očitava echoPin i vraća vrijednost vremenskog trajanja vraćanja vala*

*distance= duration\*0.034/2; // Izračunavanje udaljenosti*

*Serial.print("Distance: ");*

*Serial.println(distance); // Ispisuje udaljenost na serijski monitor*

*return distance; //Udaljenost je vraćena u tipu integer u centimetrima*

*}*

### 3.2.3. Radio\_nRF.h

Definiramo i uključujemo biblioteke potrebne za komunikaciju između predajnika i prijemnika. Postavljamo CE i CSN pin na željene vrijednosti i definiramo klasu RADIO koja sadrži potrebne funkcije za komunikaciju.

*#ifndef Radio\_nRF\_h // provjeravamo je li prethodno definirana*

*#define Radio\_nRF\_h // definiranje biblioteke Radio\_nRF\_h*

*#include<SPI.h>*

*#include<nRF24L01.h>*

*#include<RF24Network.h>*

*#include<RF24.h>*

*#define CE\_PIN 9 // definiranje pina 9*

*#define CSN\_PIN 10 // definiranje pina 10*

*class RADIO { //definiranje klase RADIO*

*public:*

*RADIO(); //konstruktor RADIO*

*~RADIO();// destruktor RADIO*

*voidnRF\_init();*

*voidnRF\_network\_init(uint16\_t);*

*bool RF\_send(uint16\_t, structSensorDatasensorData); // slanje*

*void RF\_receive(structSensorDatasensorDataRec, uint16\_t); //primanje*

*void RF\_powerUp(); //uključivanje*

*void RF\_powerDown(); // isključivanje*

*};*

*#endif*

### 3.2.4. Radio\_nRF.cpp

Kreiramo radio i niz aktivnih čvorova. Pokrećemo mrežu, povećavamo joj domet i postavljamo kanal. Pokrećemo funkciju *RADIO::RF\_send* koja, ukoliko je poslana informacija, ispisuje udaljenost. Nadalje pokrećemo funkciju *RADIO::RF\_receive* koja pregledava dostupne mreže i ukoliko je dostupna informacija, šalje je u predodređeni tip za očitavanje takve poruke. Provjeravaju se čvorišta te dodaju aktivnim čvorovima ukoliko je informacija poslana s druge lokacije a ne sami sebi. Ukoliko ne pronalazimo tip potreban za očitavanje poruke, ispisujemo poruku upozorenja.

*#include "Radio\_nRF.h"*

*#include "Network\_Ping.h"*

*RADIO::RADIO(){}*

*RADIO::~RADIO(){}*

*RF24 radio(CE\_PIN, CSN\_PIN); // Kreiramo radio*

*RF24Network network(radio); //Kreiramo mrežu*

*const short max\_active\_nodes = 10; // Max broj aktivnih čvorova*

*uint16\_t active\_nodes[max\_active\_nodes]; //Definiramo niz aktivnih čvorova*

*short num\_active\_nodes = 0; //Trenutni broj aktivnih čvorova*

*void RADIO::nRF\_init(){ //Poziv funkcije nRF\_init()*

*SPI.begin(); // Pokrećemo RF network*

*radio.begin(); //Pokrećemo radio*

*radio.setPALevel(RF24\_PA\_HIGH); //Povećavamo domet radija*

*}*

*void RADIO::nRF\_network\_init(uint16\_t this\_node\_){ //pokrećemo funkciju koja za varijablu uzima adresu našeg čvora*

*network.begin(/\*channel\*/ 101, /\*node address\*/ this\_node\_ ); //Postavljamo kanal i adresu čvora*

*}*

*bool RADIO::RF\_send(uint16\_t to, struct SensorData sensorData){ //pokrećemo funkciju koja uzima objekt sensorData iz strukture SensorData*

*RF24NetworkHeader header(/\*to node\*/ to, /\*type\*/ 'T' /\*Time\*/);*

*Serial.println(sensorData.distance); // Ukoliko je poslano, ispiši udaljenost*

*return network.write(header,&sensorData,sizeof(sensorData));*

*}*

*void RADIO::RF\_receive(SensorData sensorDataRec, uint16\_t this\_nodes) {*

*network.update(); // Konstantno provjeravanje mreže*

*while ( network.available() ) { // Provjeravamo postoji li dostupna mreža*

*RF24NetworkHeader header; // Ako postoji, pregledajemo je*

*network.peek(header);*

*switch (header.type){ //Odašiljemo poruku točnom tipu*

*case 'T': SensorData sensorDataRec; // T samo sadržava vrijeme*

*network.read(header,&sensorDataRec,sizeof(sensorDataRec)); //očitavamo poruku*

*printf\_P(PSTR("---------------------------------\n\r"));*

*Serial.println(sensorDataRec.distance); //ispisujemo udaljenost*

*// Dodati određeni čvor aktivnim čvorovima*

*if ( header.from\_node != this\_nodes || header.from\_node > 00 ) // Ako sami sebi odašiljemo, ne postavljaj čvor u aktivne čvorove*

*{*

*// Dodati određeni čvor aktivnim čvorovima*

*short i = num\_active\_nodes; // Provjera postoji li negdje čvor*

*while (i--) {*

*if ( active\_nodes[i] == header.from\_node )*

*break;*

*}*

*if ( i == -1 && num\_active\_nodes < max\_active\_nodes ){ //Ako ne, dodajemo ga*

*active\_nodes[num\_active\_nodes++] = header.from\_node;*

*printf\_P(PSTR("%lu: APP Added 0%o to list of active nodes.\n\r"),millis(),header.from\_node);*

*}*

*}*

*break;*

*default: printf\_P(PSTR("\*\*\* WARNING \*\*\* Unknown message type %c\n\r"),header.type); //Ako nismo pronašli tip čvora, ispisujemo upozorenje*

*network.read(header,0,0);*

*break;*

*};*

*}*

*}*

*void RADIO::RF\_powerUp() { //Pokreni radio*

*radio.powerUp();*

*delay(50);*

*}*

*void RADIO::RF\_powerDown() { //Ugasi radio*

*delay(50);*

*radio.powerDown();*

*delay(50);*

*}*

### 3.2.5.Network\_ping.h

Definiramo strukturu SensorData koja s float varijablom distance i radimo enumeraciju stanja mašine.

*#ifndef Network\_Ping\_h*

*#define Network\_Ping\_h*

*#include <avr/pgmspace.h>*

*#include <SPI.h>*

*struct SensorData {*

*float distance;*

*};*

*typedef enum {*

*READ\_SERIAL,*

*GOTO\_SLEEP,*

*READ\_SENSORS,*

*RECEIVING\_RADIO,*

*SENDING\_RADIO*

*} State;*

*#endif*

### 

### 3.2.6. Network\_ping.ino

Ovaj mrežni sloj koristi osnovnu sposobnost nRF24L01 (+) aktivnog slušanja do 6 drugih radija odjednom. Više čvorova nije međusobno povezano, tako da postoji samo jedan put do bilo kojeg čvora. Da bi realizirali mrežnu topologiju stablaste stukture potrebno je koristiti biblioteku *RF24Network.* Glavni čvor se može vidjeti kao pristupnik, s do 5 izravno povezanih čvorova. Svaki od tih čvorova stvara podmrežu ispod nje, s do 5 dodatnih child čvorova. Način numeriranja također se sličan IP adresiranju.

* Node 00 je osnovni čvor
* Čvorovi 01-05 su čvorovi čiji roditelj je osnovni čvor.
* Node 021 je drugi child čvora 01.
* Čvor 0321 treće je child čvora 021 itd.

Postavljamo echoPin i trigPin, brzinu slanja, adresu čvora i pokrećemo radio u Setup funkciji. U Loop-u postavljamo stanje radija i switch – case petljom prolazimo stanja radija. U *send\_T* funkciji šaljemo vrijednost SensorData.

*#include "printf.h"*

*#include "Sensors.h"*

*#include "Network\_Ping.h"*

*#include "Radio\_nRF.h"*

*// Oktalne adrese*

*const uint16\_t node\_address\_set[10] = { 00, 02, 05, 012, 015, 022, 025, 032, 035, 045 };*

*// 0 = Master*

*// 1-2 (02,05) = Children of Master(00)*

*// 3,5 (012,022) = Children of (02)*

*// 4,6 (015,025) = Children of (05)*

*// 7 (032) = Child of (02)*

*// 8,9 (035,045) = Children of (05)*

*uint16\_t this\_node; // Naša adresa čvora*

*uint8\_t NODE\_ADDRESS = 2; // Definiramo adresu čvora*

*const int trigPin = 5;*

*const int echoPin = 6;*

*unsigned long last\_time\_sent;*

*const unsigned long interval = 1000; // ms // Delay manager kako bi slali pingove redovno*

*bool send\_T(uint16\_t to); // Prototipovi funkcije za slanje i primanje poruka*

*SENSORS sensor;*

*RADIO radioNRF;*

*SensorData sensorData;*

*void setup(){*

*Serial.begin(115200);*

*pinMode(inputPin, INPUT);*

*pinMode(trigPin, OUTPUT); // postavlja trigPin kao Output*

*pinMode(echoPin, INPUT); // Postavlja echoPin kao input*

*pinMode(4, OUTPUT);*

*this\_node = node\_address\_set[NODE\_ADDRESS]; // Određujemo koji smo čvor*

*radioNRF.nRF\_init();*

*radioNRF.nRF\_network\_init(this\_node);*

*printf\_begin();*

*printf\_P(PSTR("\n\rRF24Network/examples/meshping/\n\r"));*

*}*

*void loop(){*

*State state = GOTO\_SLEEP;*

*for(;;) {*

*switch(state) {*

*case READ\_SERIAL:*

*state = GOTO\_SLEEP;*

*break;*

*case GOTO\_SLEEP:*

*//state = RECEIVING\_RADIO;*

*state = READ\_SENSORS;*

*break;*

*case READ\_SENSORS:*

*sensorData.distance = sensor.distance();*

*Serial.println(sensorData.distance);*

*delay(2000); // Delay*

*state = SENDING\_RADIO;*

*break;*

*case SENDING\_RADIO:*

*if ( millis() - last\_time\_sent >= interval ){ // Šaljemo sensordata idućem čvoru*

*last\_time\_sent = millis();*

*if ( this\_node > 00){*

*if (send\_T(00)){ // Obavijesti nas o rezultatu*

*printf\_P(PSTR("%lu: APP Send ok\n\r"),millis());*

*}else{*

*printf\_P(PSTR("%lu: APP Send failed\n\r"),millis());*

*last\_time\_sent -= 100; // Pokušaj poslaz u drugom vremenu u slučaju odugovlačenja*

*}*

*}*

*}*

*state = RECEIVING\_RADIO;*

*break;*

*case RECEIVING\_RADIO:*

*SensorData sensorDataRec;*

*radioNRF.RF\_receive(sensorDataRec, this\_node);*

*state = READ\_SERIAL;*

*break;*

*}*

*}*

*}*

*bool send\_T(uint16\_t to) //Šaljemo poruku što sadrži SensorData*

*{*

*return radioNRF.RF\_send(to, sensorData);*

*}*

### 3.2.7. Printf.h

Definira type struct printf\_info , makronaredbe i enum vrijednosti PA\_\*. Deklarira funkcije printf\_function, printf\_arginfo\_function, register\_printf\_function, parse\_printf\_format, printf\_size, printf\_size\_info.

*#ifndef \_\_PRINTF\_H\_\_*

*#define \_\_PRINTF\_H\_\_*

*#ifdef ARDUINO*

*int serial\_putc( char c, FILE \* )*

*{*

*Serial.write( c );*

*return c;*

*}*

*void printf\_begin(void)*

*{*

*fdevopen( &serial\_putc, 0 );*

*}*

*#else*

*#error This example is only for use on Arduino.*

*#endif // ARDUINO*

*#endif // \_\_PRINTF\_H\_\_*

## 3.3. Programski kod za prijemnik

Programski kod prijemnika izgleda poprilično slično kao i kod predajnika. Program se sastoji od Radio\_nRF.h i Network\_Ping.h biblioteka identičnih onima u predajniku, isto kao i identičnog Radio\_nRF.cpp. Jedina promjena događa se u Network\_Ping.ino u kojima se ne koristimo ispisivanjem na serijskim monitorima već običnim ispisivanje. Isto tako u switch – case petlji u Loop-u zakomentiramo stanje *READ\_SENSORS* i prebacujemo u stanje *RECEIVING\_RADIO* kao što je prikazano:

*void loop(){*

*State state = GOTO\_SLEEP;*

*for(;;) {*

*switch(state) {*

*case READ\_SERIAL:*

*state = GOTO\_SLEEP;*

*break;*

*case GOTO\_SLEEP:*

*state = RECEIVING\_RADIO;*

*//state = READ\_SENSORS;*

*break;*

*case READ\_SENSORS:*

*delay(100);*

*state = SENDING\_RADIO;*

*break;*

*case SENDING\_RADIO:*

*if ( millis() - last\_time\_sent >= interval ){*

*last\_time\_sent = millis();*

*if ( this\_node > 00){*

*if (send\_T(00)){*

*Serial.println("%lu: APP Send ok\n\r"),millis();*

*}else{*

*Serial.println("%lu: APP Send failed\n\r"),millis();*

*last\_time\_sent -= 100;*

*}*

*}*

*}*

*state = RECEIVING\_RADIO;*

*break;*

*case RECEIVING\_RADIO:*

*SensorData sensorDataRec;*

*radioNRF.RF\_receive(sensorDataRec, this\_node);*

*state = READ\_SERIAL;*

*break;*

*}*

*}*

*}*

# 4. MQTT

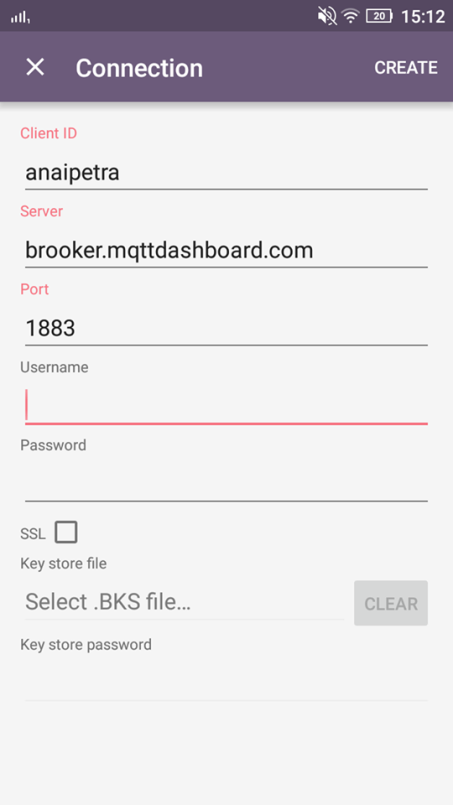
## 4.1.Općenito

MQTT protokol razvili su AndyStanford-Clark i ArlenNipper 1999. godine u svrhu praćenja naftovoda usred pustinje. Cilj im je bio razviti protokol koji bi omogućio efikasnu brzinu prijenosa podataka uz malu potrošnju baterije i jefitniju izvedbu. Protokol MQTT omogućava raspodijeljenu razmjenu poruka između krajnjih čvorova na principu objavi-pretplati. Za usporedbu objava i pretplata zadužen je posrednik MQTT koji prosljeđuje objavljene poruke i pamti pretplate klijenata. Početak komunikacije između MQTT klijenata je njihovo spajanje na posrednika MQTT koji predstavlja središte komunikacije. Klijenti moraju navesti adresu posrednika i port 1883 koji je rezerviran isključivo za komunikaciju protokolom MQTT.

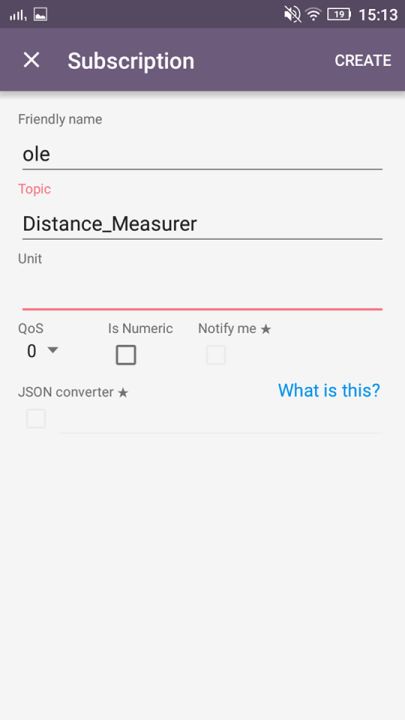
U ovom projektu koristi se aplikacija MQTT Dashboard koja nam daje vrijednosti udaljenosti koje očitava senzor.

## 4.2.Postavke

S obzirom da ćemo ispisivati rezultate udaljenosti na mobitel, skidamo s google Store Play-a aplikaciju naziva MQTT Dashboard i postavljamo ga na ove vrijednosti:



Slika 7. MQTT Dashboard



Slika 8. MQTT Dashboard

# 5. PYTHON

Unosimo mqtt klijenta i koristimo se bibliotekama time i serial. Definiramo oblik funkcije i enkriptiramo poruku, ispisujemo temu, kvalitetu poruke i postavljamo flag ukoliko nitko ne slusa poruku. Određujemo adresu broker-a i pozivamo funkciju serial za spajanje s arduinom. Očitavamo sadržaj poruke. Povezujemo se s novim klijentom, spajamo se s broker-om i započinjemo petlju. Povezujemo se s uređajem naziva Distance\_Measurer i ispisujemo sadržaj poruke. Čekamo 1 sekundu i zaustavljamo petlju.

*import paho.mqtt.client as mqtt*

*import time*

*import serial*

*def on\_message(client, userdata, message):*

*print("message received " ,str(message.payload.decode("utf-8")))*

*print("message topic=",message.topic)*

*print("message qos=",message.qos)*

*print("message retain flag=",message.retain)*

*broker\_address="broker.mqttdashboard.com"*

*arduino = serial.Serial('COM20', 115200, timeout=.1)*

*while True:*

*data = arduino.readline()[:-2]*

*if(data):*

*print("creating new instance")*

*client = mqtt.Client("anaipetra") #kreiramo novog klijenta*

*client.on\_message=on\_message #prilažemo funkciju za povratni poziv*

*print("connecting to broker")*

*client.connect(broker\_address) #spajamo s posrednikom*

*client.loop\_start() #započinjemo petlju*

*print("Subscribing to topic","Distance\_measurer")*

*client.subscribe("Distance\_measurer")*

*print("Publishing message to topic","Distance\_measurer")*

*client.publish("Distance\_measurer",data)*

*print(data)*

*time.sleep(1) # čekamo 1 sekundu*

*client.loop\_stop() #zaustavljamo petlju*

6. ZAKLJUČAK

U ovom kolegiju upoznali smo se s teorijom i načinima realizacije bežičnih senzorskih mreža i probleme koji mogu nastati prilikom izrade. Bežični senzorski uređaji su baterijski napajani i potrebna je dobra optimizacija potrošnje energije kako bi se povećala dugotrajnost baterije. Za komunikaciju između dva uređaja korišten je nRF24L01+ modul. Pri tome treba paziti da dva ili više uređaja istovremeno ne prenose poruku na istom kanalu jer je to jedan od uzroka kolizije. Kako bi svi čvorovi međusobno uspješno komunicirali treba odabrati adekvatnu topologije mreže.

Prilikom izrade projekta susreli smo se sa HC-SR04 senzorom za mjerenje udaljenosti objekata, kao i bibliotekama koje se koriste pri uporabi dotičnog. Senzor mikrokontrolerom šalje 5V na trig pin modula u trajanju od minimalno 10 mikrosekundi. Na taj način aktiviramo ultrazvučni transduktor koji odašilje 8 impulsa od 40 kHz i čeka njihovu refleksiju. Kada senzor registrira reflektirani impuls šalje podatke natrag mikrokontroleru preko echo pina. Odrađuje se vremenski interval odašiljanja i primanja vala i pomoću njega se izračunava udaljenost objekta.

S obzirom na izrazitu digitalizaciju svijeta u posljednih 10 godina, ovakvi senzori i način prenošenja podataka su izrazito potrebiti i sve češće korišteni u našoj okolini.

#### LITERATURA

[1] <https://hr.wikipedia.org/wiki/Senzori>

[2]<https://bs.wikipedia.org/wiki/Senzor>

[3] <https://e-radionica.com/hr/blog/2015/08/19/kkm-ultrazvucni-modul-hc-sr04/>

[4] <https://e-radionica.com/hr/blog/2015/08/19/kkm-ultrazvucni-modul-hc-sr04/>

[5]<https://e-radionica.com/hr/blog/2015/08/19/kkm-ultrazvucni-modul-hc-sr04/>

[6] <https://hr.wikipedia.org/wiki/Arduino>

[7]<http://www.instructables.com/id/Arduino-Parking-Sensor/>

[8] <https://sh.wikipedia.org/wiki/Oda%C5%A1ilja%C4%8D>

[9]<https://sh.wikipedia.org/wiki/Prijemnik>

[10]<https://www.sparkfun.com/datasheets/RF/nRF2401rev1_1.pdf>

[11]<https://github.com/toperkov/WiSe-2016-17-lab6>

[12]<https://www.google.com/search?q=nrf2401&source=lnms&tbm=isch&sa=X&ved=0ahUKEwijrvOE4MjbAhXKFZoKHVQFASIQ_AUICigB&biw=1517&bih=735#imgrc=eTFgMf8V_88Y2M>

[14] <https://bib.irb.hr/datoteka/768456.Final_0036472287_40.pdf>

[15] <https://github.com/toperkov/WiSe-2016-17>

[16] <http://www.instructables.com/id/Arduino-Parking-Sensor/>

#### POPIS SLIKA I TABLICA

[Slika 1. Ultrazvučni modul HC-SR04 [3] 2](#_Toc516422606)

[Slika 2. Način rada ultrazvučnog modela HC-SR04 [4] 3](#_Toc516422607)

[Slika 4. Prikaz prijemnika, odnosno predajnika [11] 5](#_Toc516422608)

[Slika 5. Prikaz spajanja pinova prijemnika/predajnika naArduino [12] 5](#_Toc516422609)

[Slika 6. Spojeni prijemnik/predajnik 5](#_Toc516422610)

[Slika 7. MQTT Dashboard 16](#_Toc516422611)

[Slika 8. MQTT Dashboard 16](#_Toc516422612)