SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

SEMINAR

MJERENJE UDALJENOSTI I OSVIJETLJENOSTI POMOĆU HC-SR04 I LDR

Ante Pupačić

Mentori: **prof. dr. sc. Mario Čagalj**

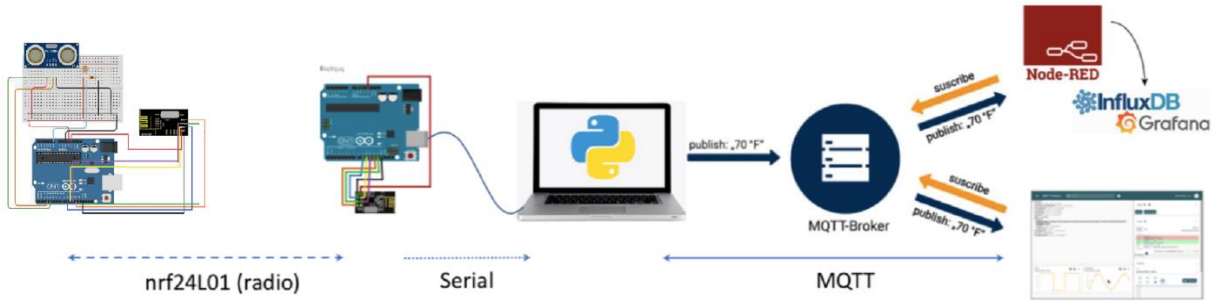
prof. dr. sc. Toni Perković

Split, srpanj 2020.

SADRŽAJ

1. UVOD

U ovom projektu je realiziran senzorski čvor s periodičkim očitavanjem udaljenosti i osvijetljenosti pomoću senzora HC-SR04 i LDR. LDR ne može očitavati sa analognog ulaza vrijednosti u lux-ima već treba te podatke izračunati poznavajući otpor i napon LDR. Slanjem podataka pomoću radioprijamnika te odlaskom u stanje mirovanja. Radioprijamnik šalje podatke samo u slučaju ako očitani podatak nije jednak prethodnome, izvedeno pomoću state mašine. Prikupljanje podataka te prosljeđivanje na serijski port. Čitanje tih podataka sa serijskog porta putem Python skripte te slanje podataka prema MQTT brokeru. Primanje podataka sa MQTT brokera (Node-RED), spremanje u Influx bazu te njihova vizualizacija (Chronograf, Grafana). Pretplaćivanje (engl. subscribe) na temu te njihova vizualizacija pomoću MQTT-explorera (*Slika 1.1. Cjelokupna shema projekta*). Cjelokupni kod projekta se nalazi na GitHub-u: https://github.com/apupac00/izvjestaji_WiSe_2019_20/tree/master/Seminar.



Slika 1.1. Cjelokupna shema projekta

2. SENZORSKI ČVOR

Senzorski čvor se sastoji od ultrazvučnog senzora HC-SR04 koji mjeri udaljenost i LDR koji mjeri osvijetljenost. Podaci o udaljenosti i osvijetljenosti se učitavaju te šalju pomoću nRF24L01+ radioprijamnika. U slučaju da je došlo do pogreške u hardveru ili softveru Watchdog Timer resetira procesor. Nakon toga Arduino odlazi u stanje mirovanja, realizirano pomoću state mašine. Arduino Uno ne može odraditi sve radnje odjednom već izvršava jednu po jednu radnju.

Kod za state mašinu.

```
switch (state) {

    case READ_SERIAL:
      state = READ_SENSORS;
      break;

    case READ_SENSORS:

      dataToSend.distance = sensor.readDistance();
      dataToSend.lightLevel = sensor.readLight();

      state = RADIO_TX;
      wdt_reset();
      break;

    case RADIO_TX:

      compareWithPreviousValues();
      wdt_reset();

      break;
    case RADIO_RX:

      radioNRF.RF_receive(rslt);
      state = SLEEP_STATE;
      break;
    case SLEEP_STATE:

      for(int i = 0; i < N; i++){
        delay(50);
        LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
        delay(50);
      }
```

```
      state = READ_SERIAL;
      break;

    default:

      break;
  }
```

2.1. Arduino Uno

Arduino Uno je mikrokontrolerska ploča koji se temelji na 8-bitnom ATmega328P mikrokontroleru. Zajedno s ATmega328P, on se sastoji od drugih komponenti poput kristalnog oscilatora, serijske komunikacije, regulatora napon itd (*Slika 2.1.1. Arduino Uno*).

.

Slika 2.1.1. Arduino Uno

Arduino Uno ima 14 digitalnih ulazno/izlaznih pinova (od kojih se 6 može koristiti ka PWM izlazi), 6 analognih ulaznih pinova, USB priključak, priključnice za napajanje, ICSP zaglavlje i gumb za resetiranje. Arduino Uno napaja se preko USB kabela koji je spojen na laptop, na pinovima su spojene komponente projekta HC-SR04, otpornik, LDR i nRF24L01+ te su isprogramirane.

Tablica 2.1.1. Opis pinova

Kategorija pina	Ime pina	Detalji
Napajanje	Vin, 3.3V, 5V, GND	Vin: Ulazni napon u Arduino kada koristi vanjski izvor napajanja. 5V: Regulirano napajanje koje se koristi za napajanje mikrokontrolera i drugih komponenti na ploči. 3.3V: Napajanje generirano pomoću ugrađenog regulatora napona. Maksimalni strujni tok je 50mA. GND: uzemljeni pin.
Reset	Reset	Resetira mikrokontroler
Analogni pinovi	A0 - A5	Koristi se za osiguravanje analognog ulaza u rasponu od 0 - 5V.
Ulazno/Izlazni pinovi	Digitalni pinovi	Može se koristiti kao ulazni ili izlazni pin.
Serijski	0(Rx), 1(Tx)	Koristi se za primanje i prijenos TTL serijskih podataka.
Vanjski prekid	2, 3	Za aktiviranje prekida.
PWM	3, 5, 6, 9, 11	Koristi 8-bitni PWM izlaz.
SPI	10 (SS), 11 (MOSI), 12 (MISO) i 13 (CSK)	Koristi se za SPI komunikaciju.
Ugrađeni LED	13	Koristi se za uključivanje ugrađene LED-ice.
TWI	A4 (SDA), A5 (SCA)	Koristi se za TWI komunikaciju.
AREF	AREF	Osigurava referentni napon za ulazni napon.

2.2. Ultrazvučni sensor HC-SR04

HC-SR04 ultrazvučni sensor koristi sonar za određivanje udaljenosti od objekta. Nudi izvrsno beskontaktno otkrivanje dometa s visokom preciznošću i stabilnim očitanjima u paketu koji se lako koristi (*Slika 2.2.1. HC-SR04*).

Slika 2.2.1. HC-SR04

Pinovi:

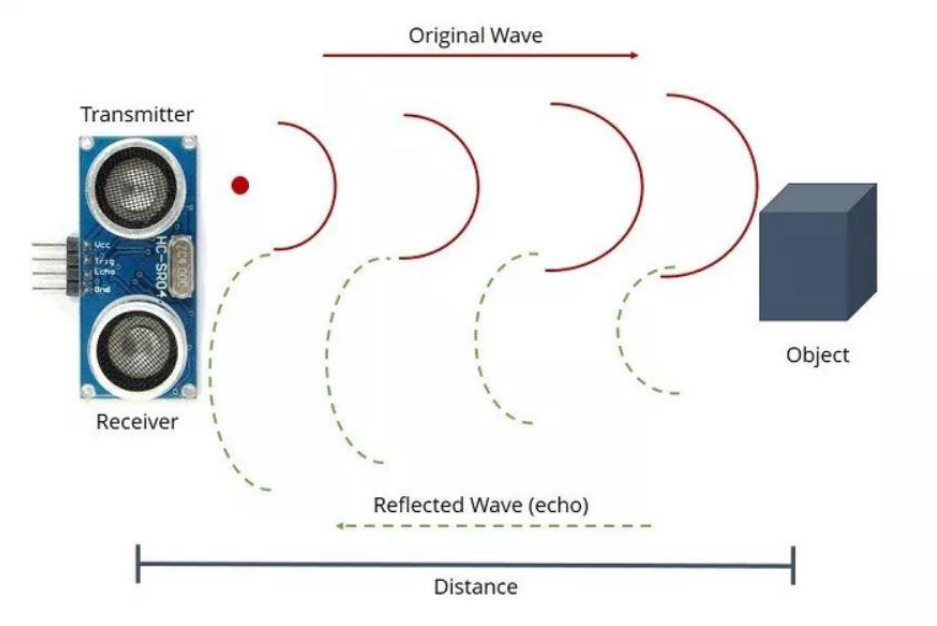
- Vcc: +5VDC
- Trig: Trigger (INPUT)
- Echo: Echo (OUTPUT)
- GND: GND

Popis nekih karakteristika i specifikacija ultrazvučnog senzora HC-SR04:

- Napajanje: +5V DC
- Mirna struja: < 2mA
- Radna struja: 15mA
- Učinkoviti kut: < 15°
- Raspon udaljenosti: 2cm – 400 cm
- Rezolucija: 0.3cm
- Mjerni kut: 30 stupnjeva
- Širina ulaznog impulsa okidača: 10us
- Dimenzija: 45mm x 20mm x 15mm

Ultrazvučni senzor koristi sonar za određivanje udaljenosti od objekta (*Slika 2.2.2. Određivanje udaljenosti*). Evo što se događa:

1. Odašiljač (trig pin) šalje signal: visokofrekventni zvuk.
2. Kad signal pronađe objekt, refelektira se i …
3. Prijamnik (echo pin) prima ga.



Slika 2.2.2. Određivanje udaljenosti

Kod unutar funkcije readDistance() u kojem ultrazvučni senzor mjeri udaljenost od objekta u centimetrima deset puta te računa aritmetičku sredinu, te se ta vrijednost vrača pomoću naredbe return. Radi stabilnosti kod čitanja vrijednosti dodan je mali delay od dvadeset milisekundi između svakog čitanja vrijednosti. Delay pauzira program za iznos vremena (u milisekundama) navedenog kao parameter.

```
float SENSORS::readDistance(){

    float dist;
    float sum = 0.0;

    for(int i = 0; i < 10; i++){
        delay(20);
        float d = hcsr04.dist();
        delay(20);
        sum = sum + d;
```

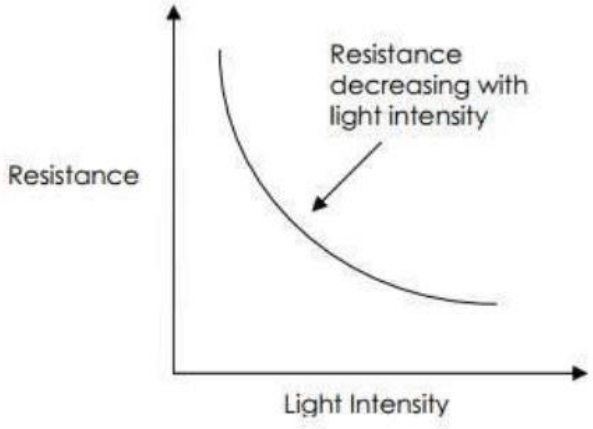
```
    }
    dist = sum / 10;

    Serial.print(F("Distance: "));
    Serial.print(dist);
    Serial.print(F(" cm "));

    return dist;
}
```

2.2. LDR

LDR je svjetlosno ovisni otpornik, odnosno otpor LDR se mijenja kako se količina svjetlosti koja pada na njega mijenja (*Slika 2.2.1. LDR*).

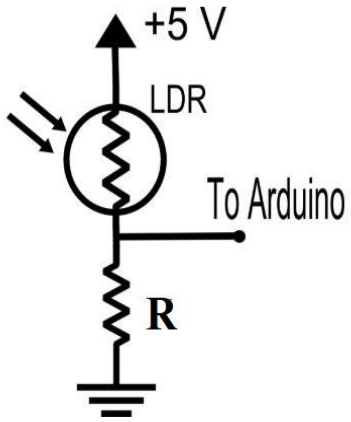
Slika 2.2.1. LDR

Mjerenjem otpornosti LDR i poznavanje karakteristika LDR-a može se kvantificirati količina lux-a koja pada na LDR. Općenito, što je svjetlost svjetlija, to je niži otpor. Odnos otpornosti i lux-a za LDR je eksponencijalan (*Graf 2.2.2. Ovisnost otpora o osvijetljenosti*).



Graf 2.2.1. Ovisnost otpora o osvijetljenosti

Za izradu lux metra LDR je spojen na 5V, otpornik je spojen na zemlju, a točka između njih spojena je s analognim ulazom 0 (*Slika 2.2.3. Shema spoja lux metra*).



Slika 2.2.2. Shema spoja lux metra

Jednadžba dobivena u 1. dijelu odnosi se na osvjetljenje na otporu, ali analogni ulazni pin mjeri samo napon, tako da treba napraviti nekoliko izračuna da bi se dobio otpor. Prvo se digitalni prikaz analognog napona mora dobiti pomoću analogRead funkcije (to je napon preko otpornika, a ne napon preko LDR).

$$RawData = analogRead(LDR\_PIN) \quad (2.2.1)$$

Drugo, digitalni prikaz mora se pretvoriti natrag u napon povećanjem vrijednosti analogno-digitalnog pretvarača na referentni napon. MAX_ADC_READING je 1023, a ADC_REF_VOLTAGE je 5.

$$ResistorVoltage = (float)ldrRawData/MAX\_ADC\_READING*ADC\_REF\_VOLTAGE \quad (2.2.2)$$

Treće, treba odrediti napon LDR. Budući da je 5V podijeljeno između otpornika i LDR, jednostavno oduzmemo napon otpornika od 5V.

$$LdrVoltage = ADC\_REF\_VOLATGE - resistorVoltage \quad (2.2.3)$$

Konačno, otpor LDR-a može se izračunati na temelju napona (jednostavan izračun otpora za krug djelitelja napona). REF_RESISTANCE je 216.4ohm

$$LdrResistance = ldrVoltage / resistorVoltage * REF\_RESISTANCE \quad (2.2.4)$$

Jednom kada se dobije otpor, osvjetljenje (u lux-ima) se može izračunati. LUX_CALC_SCALAR i LUX_CALC_EXPONENT određene su u Excelovoj proračunskoj tablici. U mom su primjeru postavljeni na 12518931 odnosno -1.405

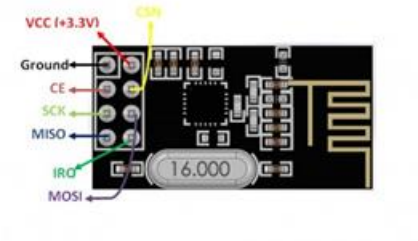
$$ldrLux = LUX\_CALC\_SCALAR * pow(ldrResistance, LUX\_CALC\_EXPONENT) \quad (2.2.5)$$

Kod unutar funkcije readLight() pomoću koje se učitava vrijednost analognog ulaza deset puta te se računa aritmetička sredina, te jednadžbe pomoću kojeg se prebacuje u lux. Vrijednost u lux-ima se vraća pomoću naredbe return.

```
uint16_t SENSORS::readLight(){

    uint16_t ldrRawData;
    uint16_t sum = 0;

    for(int i = 0; i < 10; i++){
        delay(20);
        uint16_t l = analogRead(A0);
        delay(20);
        sum = sum + l;
    }
    ldrRawData = sum / 10;

    uint16_t MAX_ADC_READING = 1023;
    uint8_t ADC_REF_VOLTAGE = 5;
    float REF_RESISTANCE = 216.4;
    uint32_t LUX_CALC_SCALAR = 12518931;
    float LUX_CALC_EXPONENT = -1.405;
    float resistorVoltage = (float)ldrRawData / MAX_ADC_READING * ADC_REF_VOLTAGE
;
    float ldrVoltage = ADC_REF_VOLTAGE - resistorVoltage;
    float ldrResistance = ldrVoltage / resistorVoltage * REF_RESISTANCE; // REF_R
ESISTANCE is 216.4 ohm

    uint16_t ldrLux = LUX_CALC_SCALAR * pow(ldrResistance, LUX_CALC_EXPONENT);

    Serial.print(F("Light: "));
    Serial.print(ldrLux);
    Serial.println(F(" lux"));

    return ldrLux;
}
```

3. BEŽIČNA KOMUNIKACIJA - nRF24L01+

Bežična komunikacija je izvedena pomoću primopredajnika nRF24L01+ gdje se jedan koristi kao predajnik podataka očitanih sa senzora a drugi kao prijamnik tih podataka (*Slika 3.1. nRF24L01+*).



Slika 3.1. nRF24L01+

nRF24L01+ je modul bežičnog primopredajnika, što znači da svaki modul može i slati i primate podatke. Radi u frekvenciji 2.14 GHz, što spade u raspon ISM-a i stoga je legalno koristiti u gotovo svim zemljama za inženjerske primjene. Kada se učinkovito upravlja modulima signal može prijeći udaljenost od 100 metara što ga čini odličnim izborom za sve bežično daljinsko upravljane projekte.

Modul djeluje na 3.3V, stoga se može lako koristiti s 3.2V ili 5V sustavima. Svaki moduli ima raspon adresa od 125 i svaki modul može komunicirati sa šest drugih modula, stoga je moguće u više područja komunicirati više bežičnih jedinica.

NRF24L01 modul djeluje uz pomoć SPI komunikacije. Serijsko periferno sučelje (engl. Serial Peripheral Interface – SPI) je specifikacija sinkronog serijskog komunikacijskog sučelja koje se koristi na kratke udaljenosti, prije svega u ugrađenim sustavima.

Ako modul povezujemo s Arduinom, tada su na raspolaganju spremne biblioteke poput biblioteke R24. Pomoču ovih biblioteka može se jedonstavno sučeliti nRF24L01 s Arduinom pomoču nekoliko linija koda.

Tablica 3.1. nRF24l01+ konfiguracija pina

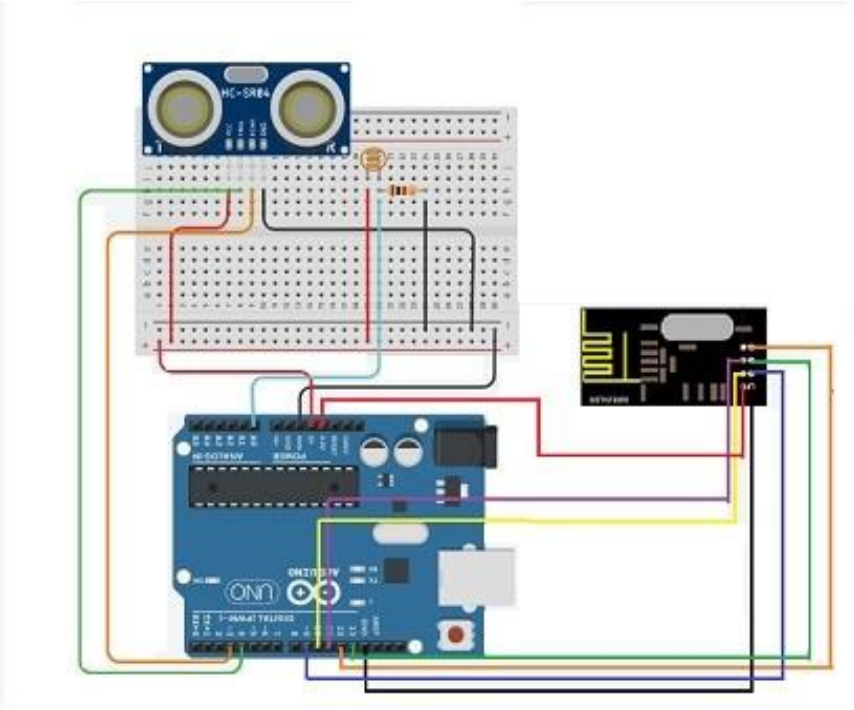
Broj pina	Ime pina	Skraćenica	Funkcija
1	Ground	Ground	Povezan s masom sustava
2	Vcc	Power	Napaja modul pomoću 3.3V
3	CE	Chip Enable	Koristi se za omogućavanje SPI komunikacije
4	CSN	Ship Select Not	Taj pin uvijek mora biti u visokom stanju, jer će u suprotnome onemogućiti SPI
5	SCK	Serial Clock	Omogućuje puls sata koristeći SPI komunikaciju
6	MOSI	Master Out Slave In	Spojen na MISO pin MCU-a, kako bi modul primio podatak iz MCU-a
7	MISO	Master In Slave Out	Spojen na MISO pin MCU-a, kako bi modul posla podatak iz MCU-a
8	IRQ	Interrupt	To je aktivni niski pin i koristi se samo ako je potreban prekid

Značajke nRF24L01+:

- 2.4GHz RF modul primopredajnika
- Radni napon: 3.3V
- Nominalna struja: 50mA
- Raspon: 15 – 30 metara
- Radna struja: 250mA (najveća)
- Komunikacijski protocol: SPI
- Brzina prijenosa: 250kbps – 2Mbps
- Raspon kanala: 125
- Maksimalni broj cjevovoda/čvorova: 6
- Niska cijena bežičnog rješenja

3.1. Odašiljač

Odašiljač je realiziran pomoću nRF24L01+ koji šalje podatke o udaljenosti i osvijetljenosti te odlazi u stanje mirovanja (*Slika 3.1.1. Shema spoja odašiljača*).



Slika 3.1.1. Shema spoja odašiljača

Kod funkcije nRF_init() u kojoj je inijaliziran nRF24L01+.

radio.setDataRate(RF24_250KBPS) predstavlja brzinu prijenosa podataka. Što je veća brzina prijenosa podatak, više se energija troši.

radio.setChannel(113) je kanal pri kojem nRF24L01+ komunicira.

radio.setPALevel(RF24_PA_MIN) je pojačalo snage emitiranog signala. Ova vrijednost utječe na udaljenost koju će se slati signal.

radio.setRetries(3, 5) predstavlja broj pokušaja ponavljanja komunikacije i vrijeme koje prođe između pokušaja.

```
void RADIO::nRF_init()
{
    radio.begin();
    radio.setDataRate(RF24_250KBPS);
    radio.setChannel(113);
    radio.setPALevel(RF24_PA_MIN);
```

```
    radio.setRetries(3, 5);
    radio.openWritingPipe(address);
}
```

Unutar strukture SensorData kod koje imamo dvije varijable, spremaju se vrijednosti o udaljenosti i osvjetljenosti. Ti podaci se šalju pomoću funckije RF_send () koja prima podatke strukture te vraća istinu ili laž ovisno o tome dali je podatak uspješno poslan.

```
bool RADIO::RF_send(struct SensorData sensorData)
{
    bool rslt;

    delay(50);
    rslt = radio.write(&sensorData, sizeof(sensorData));
    delay(50);

    Serial.print(F("Data Sent "));
    Serial.print(sensorData.distance);
    Serial.print(F(" "));
    Serial.print(sensorData.lightLevel);

    return rslt;
}
```

Kod funkcije RF_recieve. Ispisuje na serijskom portu dali je poruka uspješno poslana ili nije.

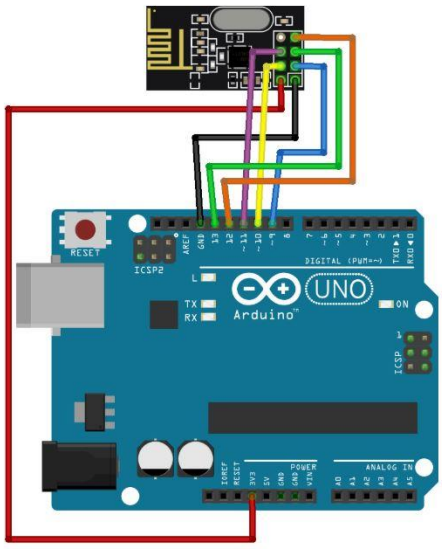
```
void RADIO::RF_receive(bool rslt)
{
    if(rslt) {
        Serial.println(F(" Acknowledge received"));
    } else{
        Serial.println(F(" Tx failed"));
    }
}
```

Kod funkcije compareWithPreviousValues(). Ako očitani podaci na senzorima nisu se promijenili u odnosu na prethodne s odstupanjem od 2%, podatak se neće poslati. Inicijaliziran je brojač koji povećava vrijednost za jedan svaki put kada se podataka ne pošalje. Ako se podataka ne pošalje deset puta za redom (brojač ima vrijednost deset), podatak će se poslati tako da prijamnik zna da je odašiljač "živ". Nakon toga vrijednost brojača odlazi na nulu i sve se to ponavlja.

```
void compareWithPreviousValues(){

  if (!((preData.preDist <= (dataToSend.distance + (DEV*dataToSend.distance))) &&
 (preData.preDist >= (dataToSend.distance - (DEV*dataToSend.distance)))) || !((pr
eData.preLight <= (dataToSend.lightLevel + (DEV*dataToSend.lightLevel))) &&(preDa
ta.preLight >= (dataToSend.lightLevel - (DEV*dataToSend.lightLevel)))))
  {
    rslt = radioNRF.RF_send(dataToSend);
    preData.preDist = dataToSend.distance;
    preData.preLight = dataToSend.lightLevel;
    counter = 0;
    state = RADIO_RX;
  } else if (counter >= N_max){
    rslt = radioNRF.RF_send(dataToSend);
    counter = 0;
    state = RADIO_RX;
  } else{
    Serial.println(F("Transmitter will not send data!!!"));
    counter++;
    state = SLEEP_STATE;
  }

}
```

3.1. Prijamnik

Prijamnik je izveden pomoću nRF24l01+ koji stalno čita podatke te u trenutku kad odašiljač pošalje podatke on ih prihvati (*Slika 3.1.1. Shema spoja prijamnika*).



Slika 3.1.1. Shema spoja prijamnika

Kod funkcije nRF_init() koja služi za inicijalizirat nRF24l01+.

```
void RADIO::nRF_init()
{
    Serial.println(F("SimpleRx Starting"));
    radio.begin();
    radio.setDataRate(RF24_250KBPS);
    radio.setChannel(111);
    radio.setPALevel(RF24_PA_MAX);
    radio.openReadingPipe(1, address[0]);
    radio.openReadingPipe(2, address[1]);
    radio.openReadingPipe(3, address[2]);
    radio.openReadingPipe(4, address[3]);
    radio.openReadingPipe(5, address[4]);
    radio.openReadingPipe(6, address[5]);
    radio.startListening();
}
```

Kod funkcije getData() koji dohvaća podatke.

```
void RADIO::getData(){
    uint8_t pipeNum;
    if (radio.available(&pipeNum))
    {
        radio.read(&dataReceived, sizeof(dataReceived));
        newData = true;
    }
}
```

Kod funckije showData() koji šalje podatke na serijski port.

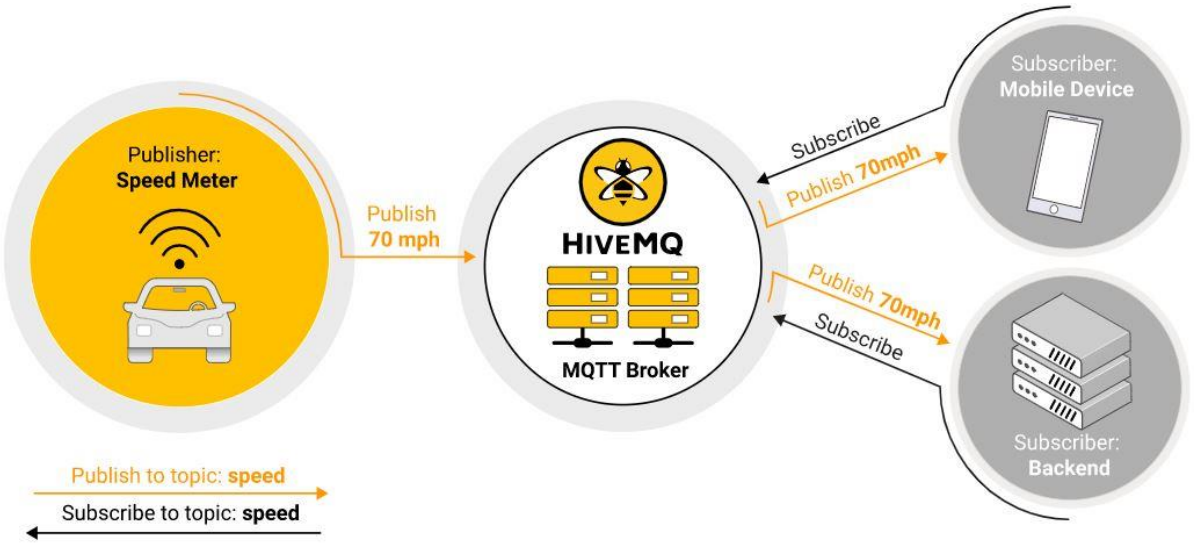
```
void RADIO::showData()
{
    if (newData == true)
    {
        Serial.println(F("Received data"));
        Serial.print(F("D: "));
        Serial.println(dataReceived.distance);
        Serial.print(F("L: "));
        Serial.println(dataReceived.lightLevel);
        newData = false;
    }
}
```

4. MQTT PROTOKOL

MQTT je protokol za slanje i objavljivanje poruka dizajniran za M2M (machine to machine) telemetriju u okruženjima male propusnosti. Danas se najviše koristi kod internet stvari (engl. Internet of Things - 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 4.1. MQTT protokol*).



Slika 4.1. 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 ili se pretplati na temu. Broker prihvaća svaku valjanu temu bez ikakve prethodne inicijalizacije (*Slika 4.2. Primjer teme*).

Slika 4.2. Primjer teme

Razina kvalitete usluge (engl. Quality of Service - 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)

4.1. Python skripta

Python skripta čita podatke o udaljenosti i osvijetljenosti sa serijskog porta, te podatke šalje preko MQTT protokola na broker koji je mqtt.eclipse.org. Prijamnik mora slati podatke na serijski port u formatu “D: 3.58” i “L: 127“ da bi ih Python skripta mogla čitati.

```
import serial
import paho.mqtt.client as mqtt

BROKER = "mqtt.eclipse.org"
```

```
CLIENTID = "MQTTExample"
TOPIC_ONE = "A507/sensors/distance"
TOPIC_TWO = "A507/sensors/light"
COMPORT = "COM6" # please replace this with your port number
QOS = 1

import time
flag_connected = 0

def on_connect(client, userdata, flags, rc):
    logging.debug("Connected result code "+str(rc))
    client.loop_stop()

def on_disconnect(client, userdata, rc):
    logging.debug("DisConnected result code "+str(rc))
    client.loop_stop()

def on_publish(client,userdata,result):             #create function for callback
    print("data published \n")
    pass

mqttc = mqtt.Client(CLIENTID)
mqttc.on_connect = on_connect
mqttc.on_disconnect = on_disconnect
mqttc.on_publish = on_publish
mqttc.connect(BROKER)
mqttc.loop_start()

ser = serial.Serial(COMPORT, 115200, timeout=5)
while True:
    message = ser.readline()
    print (message)
    if b'D:' in message:
        string, distance = message.split(b' ')
        print(distance.decode('utf-8'))
        mqttc.publish(TOPIC_ONE, payload=distance.decode('utf-
8'), qos=QOS, retain=False)
    if b'L:' in message:
        string, light = message.split(b' ')
        print(light.decode('utf-8'))
        mqttc.publish(TOPIC_TWO, payload=light.decode('utf-
8'), qos=QOS, retain=False)
    time.sleep(0.01)
mqttc.disconnect()
time.sleep(1)
```

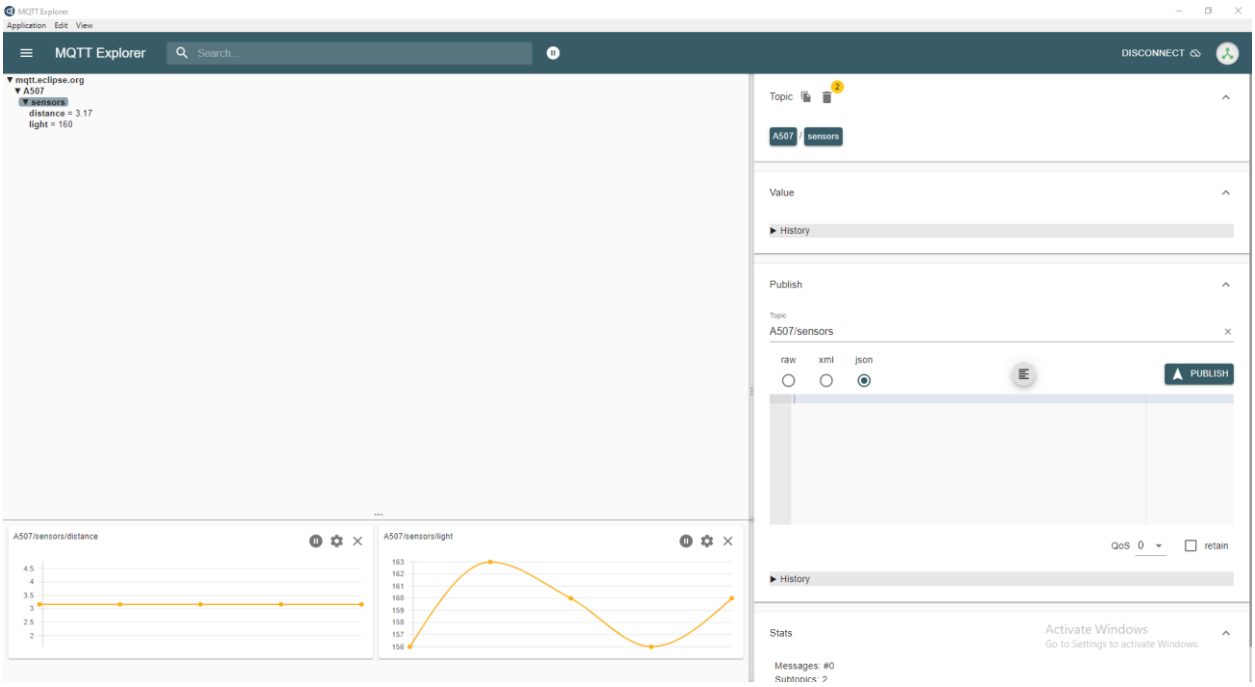
5. VIZUALIZACIJA PODATAKA

Prvi način vizualizacije podataka pomoću destop aplikacije MQTT-Explorer gdje se pretplatimo na temu u ovom radu to su A507/sensors/distance i A507/sensors/light.

Drugi način vizualizacije podataka ralizirano tako da ćemo pokrenuti doker-compose.yml u naredbenom retku koji sadrži Node-RED, Grafanu, Chronograf i influxDB. Gdje se pomoću Node-RED povezujem na određenu temu ta podatke o udaljenosti i osvijetljenosti šaljemo na influxDB bazu podataka i podatke s baze uzimamo i prikazivamo pomoću Chronografa i Grafane.

5.1. MQTT-explorer

MQTT-Explorer je desktop aplikacija za vizualizaciju podataka. Kod koje pretplatimo na temu (*Slika 5.1.1. MQTT-Explorer*).



Slika 5.1.1. MQTT-Explorer

5.2. Docker kontejner

Docker je alat dizajniran da olakšava stvaranje, implementaciju i pokretanje aplikacije pomoću kontejnera. Kontejneri omogućuju programeru da pakira aplikaciju sa svim dijelovima, kao što su bibliteke i druge ovisnosti, i raspoređuje je kao jedan paket. Na taj način, zahvaljujući kontejneru, programer može biti siguran da će se aplikacija pokrenuti na bilo kojem drugom Linux računulu bez obzira na prilagođene postavke koje stroj može imati, a koje bi se mogle razlikovati od uređaja koji se koristi za pisanje i testiranje koda.

Docker-compose.yml pomoću kojeg se pokreće Node-Red, Influxdb, Chronograf i Grafana.

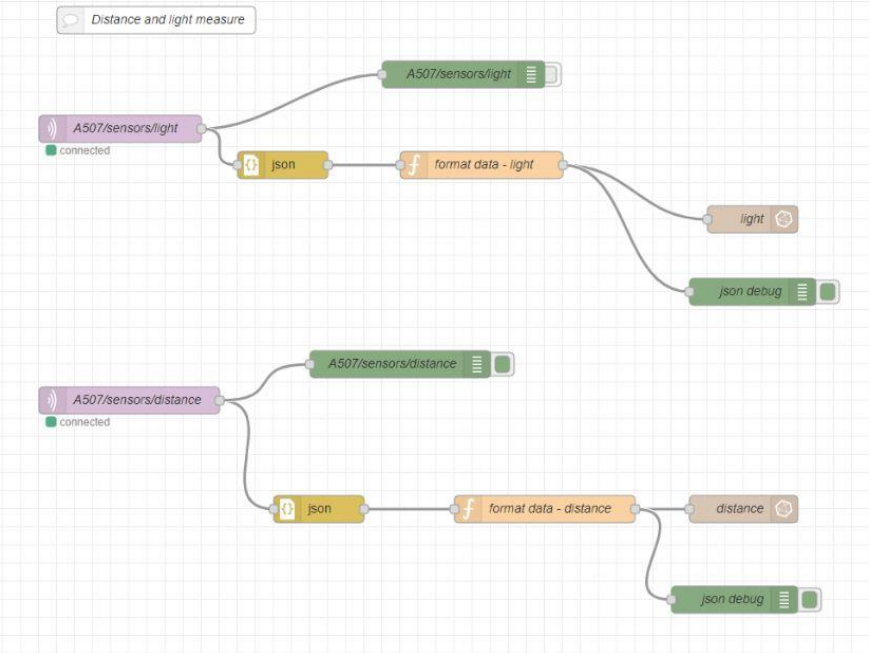
Servisi i aplikacije rade na sljedećim portovima.

Tablica 5.2.1. Servisi i portovi

Host Port	Service
1880	Node-RED
3000	Grafana
8086	InfluxDB
127.0.0.1:8888	Chronograf

5.2.1. Node-RED

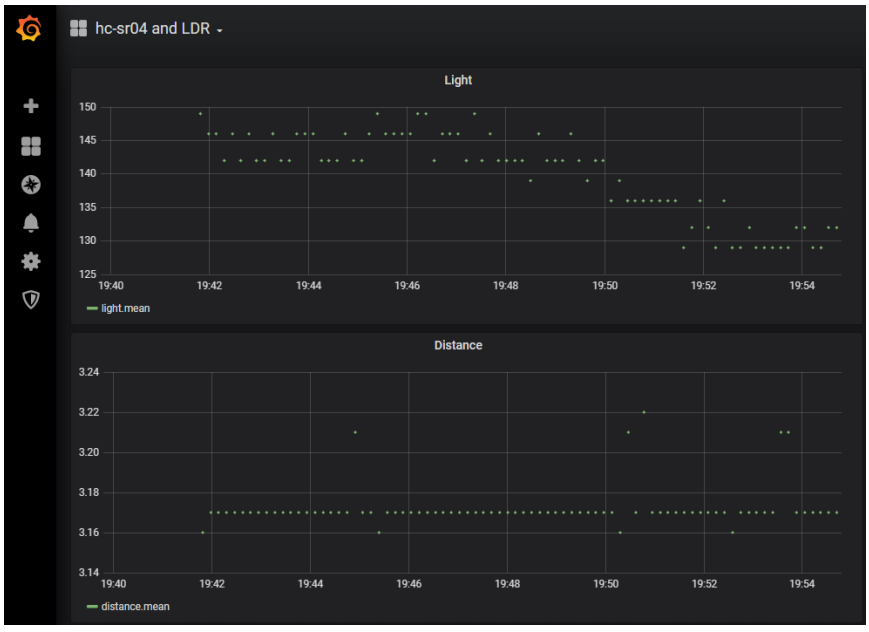
Node-Red je programski alat za spajanje hardverskih uređaja, API-ja i mrežnih usluga na nove i zanimljive načine. Omogućuje uređivač temeljen na pregledniku koji olakšava povezivanje protoka koristeći širok raspon čvorova u paleti koji se mogu uvesti u njegovo vrijeme izvršavanja (engl. runtime) u jednom kliku. Node-RED spaja se na MQTT-temu koje su A507/sensors/distance i A507/sensors/light te podatke očitane šalje na InfluxDB bazu podataka (*Slika 5.2.1.1. Node-RED*).



Slika 5.2.1.1. Node-RED

5.2.2 Grafana

Grafan je softver za vizualizaciju i analitiku otvorenog koda. Omogućuje ispisivanje, vizualiziranje, upozorenje i istraživanje mjernih podataka bez obzira na to gdje su pohranjeni (*Slika 5.2.2.1. Vizualizacija udaljenosti i osvijetljenosti pomoću Grafane*).



Slika 5.2.2.1. Vizualizacija udaljenosti i osvijetljenosti pomoću Grafane

6. ZAKLJUČAK

HC-SR04 je senzor za mjerenje udaljenosti, kod kojeg dolazi do razbacivanja mjerenih vrijednosti na većim udaljenostima, zato što se odaslani signal ne može najbolje odbiti od nekih predmeta. LDR je jeftino rješenje za lux metar, ali ne sasvim precizno. Komunikacija pomoću radioprijamnika nRF24L01+ je moguća na manjim udaljenostima, te troši mnogo energije tijekom slanja. Taj se problem rješava tako da Arduino Uno odlazi u stanje mirovanja, budi se samo kada je potrebno očitati vrijednosti na senzorima i poslati podatak. Vrlo jednostavan rješenje za vizualizaciju u realnom vremenu pomoću Grafane, Chronografa i MQTT-explorer, gdje u svakom trenutku možemo promatrati i analizirati očitane vrijednosti.

7. LITERATURA

[1] https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

[2] https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

[3] https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/

[4] https://components101.com/wireless/nrf24l01-pinout-features-datasheet

[5] http://www.steves-internet-guide.com/mqtt/

[6] https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/

[7] https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/

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

[9] https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[10] https://opensource.com/resources/what-docker

[11] https://www.docker.com/resources/what-container

[12] https://nodered.org/

[13] https://grafana.com/docs/grafana/latest/getting-started/what-is-grafana/

[14] https://www.influxdata.com/time-series-platform/chronograf/

[15] https://components101.com/microcontrollers/arduino-uno