Fakultet elektrotehnike i računarstva

Raspodijeljeni sustavi

**Radni okvir gRPC**

Studenti:

Mislav Bajan, xxx

Matija Dadanović, 0036498794

Fran Borčić,xxx

Zagreb, 20. siječnja 2020.

**1. Zadatak**

Zadatak je bio proučiti radni okvir grpc.io i napraviti primjer jednostavne aplikacije za Internet stvari koja demonstrira osnovne funkcionalnosti navedenog radnog okvira.

Voditelj: Ivana Podnar Žarko

**2. Tehnologije**

Budući da se programska aplikacija sastoji od nekoliko dijelova, od kojih će svaki biti objašnjen detaljnije u arhitekturi, zbog istog razloga projekt će se sastojat od više različitih tehnologija. Korisničko sučelje aplikacije će koristiti programski jezik JavaScript i knjižnice React i Bootstrap kako bismo postigli željeni izgled. Također, kako bi se uspjela ostvariti veza sa gRPC serverom preko web-preglednika koristi se službena knjižnica *gRPC-Web.* Poruke koje se prenose između klijenta i servera definirane su prema googlovom standardu Protocol Buffers u jeziku protobuf3. Server je implementiran u Pythonu te koristi gRPC. Relevantne tehnologije, odnosno one koje se tiču projektnog zadatka će biti opisane detaljnije u nastavku.

**2.1 Protocol Buffers**

Budući da knjižnjica gRPC koristi Protocol Buffers prije objašnjavanja gRPC-a potrebno je shvatiti što je Protocol Buffers. Protocol Buffers je metoda za serilizaciju strukturiranih podataka. Zapravo, sastoji se od dva dijela :

* Jezik za definiranje strukture podataka koji će se prenositi
* Programa koji iz jezika za definiranje strukture može generirati kod za stvaranje tih struktura u željenom jeziku (*protoc*)

Struktura podataka se definira u *.proto* datotekama. Sljedeći odsječak koda pokazuje kako se definira jednostavna poruka :

syntax = "proto3";

message Point {

int32 latitude = 1;

int32 longitude = 2;

}

Jednom kada se definiraju strukture poruka koristi se već navedeni program *protoc* koji generira kod za stvaranje tih struktura u željenom jeziku. Sljedeći odsječak koda pokazuje naredbu koja će generirati klase u Pythonu preko koji je moguće stvarati definiranu poruku.

python -m grpc\_tools.protoc -I./protos --python\_out=. ./fileName.proto

Bitno je napomenuti kako kompajler *protoc* omogućava korištenje dodataka (*engl. plugin*) koji omogućavaju dodavanje dodatnih funkcionalnosti poput generiranje dodatnog koda. Ova mogućnost se najčešće koristit kako bi generirao kod koji omogućava RPC (engl. Remote procedure call) kada se Protocol Buffers koriste sa radnim okvirom gRPC.

Inicijalno, Google je razvio Protocol Buffers sa ciljem ostvarivanja boljeg načina prijenosa podataka od XML-a te su taj cilj ostvarili jer podaci pokazuju kako su Protocol Buffers poruke 3-10 puta manje te parsiranje poruka je 20-100 brže od XML. Unatoč tome, ulogu najpopularnijeg načina prijenosa podataka u današnjici nema Protocol Buffers, ta titula pripada JSON-u (engl. JavaScript Object Notation). Međutim, korištenje Protocol Buffers-a donosi brojne prednosti naspram JSON-a. Neke od njih su :

* Za podatke se može definirati kojeg su tipa
* Podaci su u binarnom obliku stoga su manji
* Mogućnost definiranje sheme u *.proto* datotekama
* Sheme su kompatibilne sa prijašnjim i budućim verzijama shema
* Kompajler omogućava generiranje koda iz *.proto* datoteka u brojne podržane jezika

**2.2 gRPC**

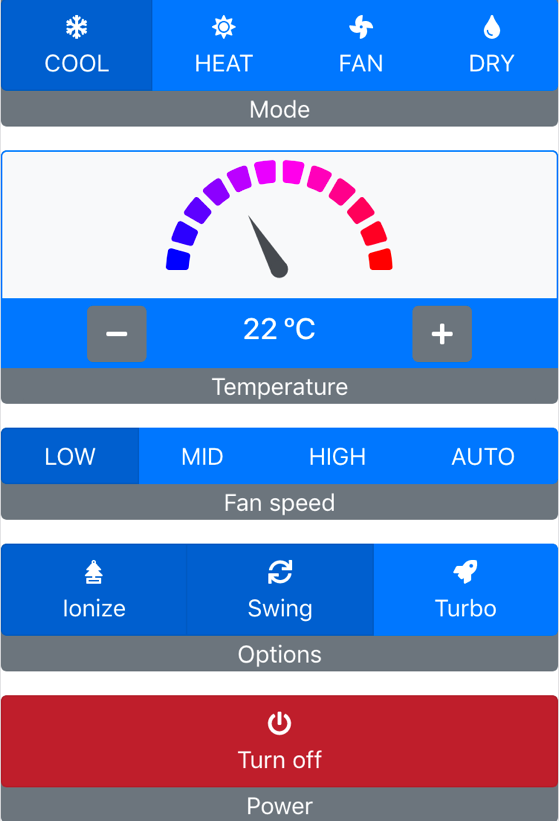
gRPC je razvojni okvir otvorenog koda razvijen od strane Google-a. gRPC je izgrađen na aplikacijskom protokolu HTTP2 te koristi upravo objašnjenu tehnologiju Protocol Buffers. Zbog činjenica da koristi te dvije tehnologije, automatski dobiva njihove beneficije. Na primjer, zbog korištenja Protocol Buffers dobiva mogućnost generiranja koda u različite jezike te je stoga gRPC pogodno rješenje ako sustav koji se razvija je pisan u različitim jezicima. Zbog korištenja HTTP2 dobiva prednosti poput efikasnijeg prijenosa, korištenja jedne veza za više zahtjeva te mogućnost streaminga. gRPC se najčešće koristi kod distributiranih sustava koji imaju veliki broj korisnika i potrebnu za malim vremenom odgovora. Najbolji primjer je Google, koji koristi gRPC za gotovo sve svoje unutarnje sustave. Također, gRPC se zbog svoje kompaktnosti često koristi u IOT (*engl. internet of things*) sustavima.

Uz tradicionalni model klijent-poslužitelj, gRPC podržava još 3 različita načina komunikacije

* Server streaming – server može poslat više poruka klijentu, korisno za slušanje događaja
* Client streaming – korisnik serveru šalje više poruka, korisno ako klijent šalje puno podataka prema serveru
* Bidirectional streaming – obje strane mogu slati više poruka, korisno ako je promet u oba smjera čest

**3. Namjena rješenja**

Sustav zamjenjuje daljinski upravljač starog klima uređaja marke Quadro i pruža mogućnost udaljenog upravljanja uređajem putem web sučelja. Pritom zadržava sve mogućnosti originalnog daljinskog upravljača poput podešavanja temperature, načina rada, snage ventilatora i slično.



Slika 1. Korisničko sučelje

Kroz sustav nastojimo demonstrirati korištenje gRPC standarda poziva udaljene procedure koristeći ga za komunikaciju između web preglednika korisnika i poslužitelja koji upravlja uređajem.

**4. Opis arhitekture**

Sustav se sastoji od tri glavne komponente – klijentske web aplikacije, poslužiteljske aplikacije i ugradbenog sklopovskog sustava.

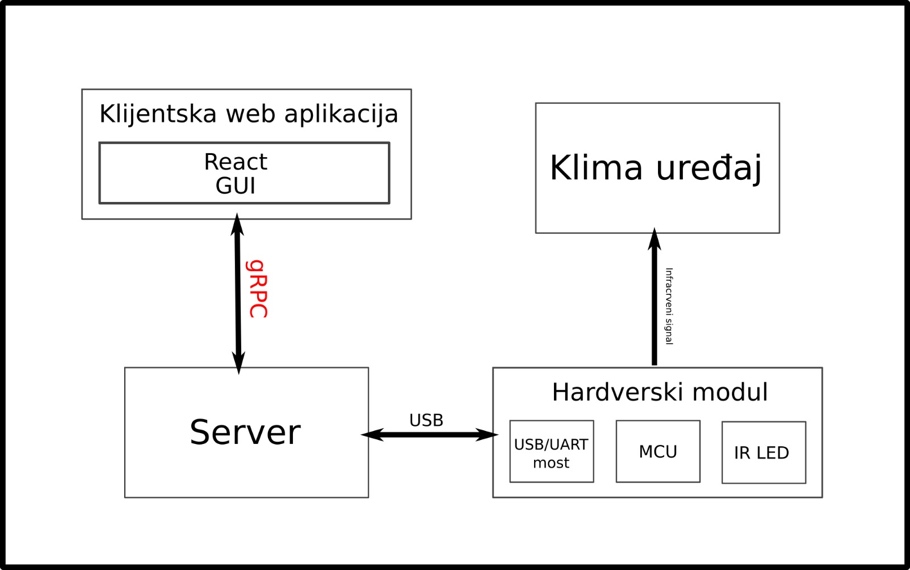
Klijentska web aplikacija ostvarena je koristeći radne okvire React i Bootstrap i prvenstveno namijenjena izvršavanju na mobilnim uređajima. Sučelje te aplikacije prikazuje posljednje postavljeno stanje klima uređaja i omogućava korisniku da mijenja stanje.

Svaki puta kad korisnik izrazi želju za promjenom stanja, klijentska aplikacija preko gRPC komunikacijskog protokola od poslužitelja zatraži izvršenje te promjene. Po potvrdnom odgovoru poslužitelja, u klijentskoj aplikaciji mijenja se prikaz stanja koji korisnik vidi.

Poslužiteljska aplikacija prima zahtjev za promjenom stanja, pretvara zahtjev u odgovarajuću poruku za klima uređaj i putem virtualnog serijskog sučelja preko USB veze tu poruku šalje sklopovskom modulu.

Sklopovski modul prema primljenoj poruci modulira signal nosioc i preko infracrvene LE diode odašilje željenu poruku klima uređaju.

Nakon odašiljanja, sklopovski modul poslužitelju potvrđuje provedbu odašiljanja, a poslužitelj putem gRPC odgovora promjenu potvrđuje klijentu.



Slika 2. Arhitektura sustava

**4. Implementacija**

Korisničko sučelje (klijentska aplikacija) ostvareno je koristeći JavaScript knjižnice React i Bootstrap. Koncipirano je tako da pri otvaranju aplikacije u korisničkom web pregledniku ono od gRPC poslužitelja zahtjeva zadnje postavljeno stanje uređaja te ga prikaže na ekranu, a potom pri svakoj interakciji korisnika s aplikacijom od istog poslužitelja zatraži promjenu stanja. Korisnički prikaz osvježit će se novim stanjem tek kada poslužitelj javi da je promjena stanja bila uspješna.

Kako je već spomenuto, klijentska aplikacija i poslužitelj komuniciraju prema gRPC standardu razmjenjujući Protocol Buffer poruke. Format tih poruka i procedure definirani su sljedećom specifikacijom u jeziku proto3:

service AirConditioner {

rpc SetState (StateRequest) returns (StateResponse);

rpc GetState (Void) returns (StateResponse);

}

message Void {};

message StateRequest {

bool power = 1;

int32 temperature = 2;

enum Speed{

LOW = 0;

MID = 1;

HIGH = 2;

AUTO = 3;

}

Speed speed = 3;

enum Mode{

COOL = 0;

HEAT = 1;

FAN = 2;

DRY = 3;

}

Mode mode = 4;

bool swing = 5;

bool ionizer = 6;

bool turbo = 7;

}

message StateResponse {

bool power = 1;

int32 temperature = 2;

enum Speed{

LOW = 0;

MID = 1;

HIGH = 2;

AUTO = 3;

}

Speed speed = 3;

enum Mode{

COOL = 0;

HEAT = 1;

FAN = 2;

DRY = 3;

}

Mode mode = 4;

bool swing = 5;

bool ionizer = 6;

bool turbo = 7;

int32 maxTemperature = 8;

int32 minTemperature = 9;

}

Procedura GetState služi za dohvaćanje zadnje postavljenog ili pretpostavljenog stanja, a procedura SetState za izražavanje namjere za postavljanjem novog stanja. Poruka StateRequest predstavlja zahtjev za promjenom stanja, a poruka StateResponse postavljeno stanje.

gRPC poslužitelj ostvaren je u programskom jeziku Python. Njegova je zadaća zaprimati zahtjeve za promjenom stanja, preračunati željeno stanje u binarni kod paketa koji treba proslijediti uređaju te proslijediti taj kod mikrokontroleru koji će uređaju putem infracrvenog signala poslati paket. Dodatno, poslužitelj poslužuje zadnje postavljeno stanje za potrebe inicijalizacije klijentskog sučelja.

Mikrokontrolerski modul zasniva se na AVR mikrokontroleru kompatibilnom s Arduino razvojnim okruženjem. Program mikrokontrolera ostvaren je u Arduino programskom jeziku, a njegova je zadaća primati gore navedeni binarni kod u heksadekadskom obliku te prema njemu modulirati signal nosioc frekvencije 38 kHz na jednom od izlaznih priključaka.

Na izlazni priključak mikrokontrolera priključena je infracrvena LE dioda koja klima uređaju odašilje poruku stanja.

**5. Upute za pokretanje i korištenje**

Kako bi se pokrenula aplikacija, potrebno je klonirati repozitorij aplikacije, instalirati potrebnu programsku podršku, pokrenuti gRPC poslužitelj i pokrenuti poslužitelj klijentske aplikacije.

Repozitorij se može klonirati sa servisa github pokretanjem naredbe:

git clone <https://github.com/fborcic/klima_react>

Potom je potrebno osigurati da su na računalu instalirani alati protoc, python3 i node.js. Postupak instalacije tih alata ovisi o platformi, pa ga ovdje nećemo navoditi.

Dodatno, potrebno je neke instalirati Python knjižnice što se može učiniti naredbom:

pip3 install grpcio-tools pyserial

Potom treba konfigurirati proxy servis koji će prevoditi HTTP/1.1 pakete na vratima 8080 u HTTP/2 pakete na vratima 50051. Kao i gore, postupak se razlikuje ovisno o platformi pa ga nećemo navoditi.

Kad je proxy poslužitelj uključen, treba pokrenuti poslužiteljsku skriptu naredbom:

python3 grpc\_server.py

Poslužitelj klijentske aplikacije pokreće se pozicioniranjem u poddirektorij frontend/acremote i pokretanjem naredbi:

npm install

npm run

Nakon što je aplikacija pokrenuta, klijentsko je sučelje dostupno na vratima 3000. Otvaranjem stranice u web pregledniku korisniku se prikazuje intuitivno grafičko sučelje nalik na ono na uobičajenom daljinskom upravljaču.

Kako bismo olakšali demonstraciju, preusmjerili smo heksadekadske pakete na standardni izlaz umjesto serijskog sučelja, pa se tako pri interakciji s korisnikom paketi ispisuju na izlazu Python procesa.

1. **Literatura**
2. Službena dokumentacija - <https://grpc.io/docs/>
3. gRPC Introduction, Stephane Maarek - <https://www.youtube.com/watch?v=XRXTsQwyZSU&t=45s>
4. gRPC Adoption and Working Architecture - <https://www.xenonstack.com/insights/what-is-grpc/>