



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Уна Радосавац
Број индекса: РА 28/2013

Тема рада: Развој cloud сервиса за контролу паметне куће гласовним
командама

Ментор рада: проф. др Иштван Пап

Нови Сад, јул 2017.



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Уна Радосавац		
Ментор, МН:	проф. др Иштван Пап		
Наслов рада, НР:	Развој cloud сервиса за контролу паметне куће гласовним командама		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2017		
Издавач, ИЗ:	Ауторски репрингт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)			
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	ИоТ, паметне куће, гласовна контрола, cloud, Амазон Алекса, Гугл Асистент, производјач, потрошач, редови		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	Рад се базира на имплементацији сервиса задужених за контролу паметне куће коришћењем гласовних команда. Искоришћене су постојеће технологије за препознавање гласа попут Амазон Алексе и Гугл Асистента. Имплементиран је интерфејс за прихватање гласовне наредбе корисника и сервиси задужени за њено парсирање и извршавање. Решење је интегрисано унутар постојећег ОБЛО система.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	доц. др Миодраг Ђукић	
	Члан:	проф. др Илија Башичевић	Потпис ментора
	Члан, ментор:	проф. др Иштван Пап	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Una Radosavac	
Mentor, MN:	Istvan Papp, Phd	
Title, TI:	Implementation of cloud based voice control module for smart home automation	
Language of text, LT:	Serbian	
Language of abstract, LA:	Serbian	
Country of publication, CP:	Republic of Serbia	
Locality of publication, LP:	Vojvodina	
Publication year, PY:	2017	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: <small>(chapters/pages/ref./tables/pictures/graphs/appendices)</small>		
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	IoT, smart home, voice control, cloud, Amazon Alexa, Google Assistant, publisher, consumer, queues	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	This paper is based on the implementation of the services in charge of the smart home system voice control. Used voice recognition engine is Amazon's Alexa and Google Assistant. An interface for accepting the voice command of the users and services in charge of its parsing and execution has been implemented. The solution is integrated within the existing OBLO system.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	Miodrag Djukic, Phd
	Member:	Ilija Basicovic, Phd
	Member, Mentor:	Istvan Papp, Phd
		Menthor's sign

Zahvalnost

Ovim putem se zahvaljujem svojoj porodici i priateljima na pruženoj podršci i strpljenju tokom dosadašnjeg školovanja.

Takođe bih se zahvalila mentorima Ištvanu i Mariji, kao i kolegama iz tima na odličnoj saradnji tokom rada na projektu.



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



SADRŽAJ

1.	Uvod	7
1.1	Pojam pametnih kuća	7
1.2	Kontrola uređaja glasovnim komadama	8
1.3	Integracija u OBLO sistem	9
2.	Teorijske osnove	10
2.1	OBLO sistem	10
2.1.1	Veza između <i>Cloud</i> servisa i centralnog kontrolera	11
2.2	Arhitektura zasnovana na odnosu proizvođač – potrošač	12
2.2.1	<i>RabbitMQ</i>	12
2.3	<i>ANTLR</i> parser	14
2.4	Servisi zaduženi za prepoznavanje govora	15
2.4.1	Amazon Aleksa	15
2.4.2	Google Asistent	16
3.	Koncept rješenja	18
3.1	Arhitektura na kojoj se zasniva ponuđeno rješenje	19
3.2	Uvezivanje servisa za glasovnu kontrolu sa OBLO <i>cloudom</i>	19
3.3	Integracija parser generatora u postojeći sistem	20
4.	Programsko rješenje	21
4.1	Interfejs za prihvatanje glasovne komande	21
4.2	Implementacija komunikacije između servisa	23
4.3	Parsiranje komande	24
4.4	Izvršavanje korisničke naredbe	26
5.	Rezultati	28
5.1	Testiranje korišćenjem uređaja za prepoznavanje glasa	28

5.2	Testiranje uz pomoć <i>ApacheBench</i> alata	29
6.	Zaključak	31
7.	Literatura	32

SPISAK SLIKA

<i>Slika 2.1 Odnos između komponenti unutar OBLO sistema</i>	11
<i>Slika 2.2 Odnos između komponenti unutar RabbitMQ brokera</i>	13
<i>Slika 2.3 Primjer generisane gramatike</i>	14
<i>Slika 2.4 Amazon Echo, Dot i Tap uređaj.....</i>	16
<i>Slika 2.5 Google Home uređaj.....</i>	17
<i>Slika 3.1 Prikaz putanje glasovne komande korisnika</i>	18
<i>Slika 4.1 Primjer JSON poruke pristigle od AVS-a</i>	21
<i>Slika 4.2 Primjer JSON poruke pristigle od Google Asistenta</i>	22
<i>Slika 4.3 Komunikacija između implementiranih servisa</i>	23
<i>Slika 5.1 Ukupno vrijeme izvršenja glasovne komande korišćenjem Amazon Alekse i Google Asistenta</i>	29
<i>Slika 5.2 Ukupno vrijeme izvršenja tekstualne komande, sa i bez korišćenja servisa za parsiranje</i>	30

SPISAK TABELA

<i>Tabela 4.1 Potrebna polja u zavisnosti od tipa komande</i>	25
<i>Tabela 4.2 Rašlanjivanje komande na podsekvence.....</i>	25
<i>Tabela 5.1 Testiranje izgovaranjem glasovnih komandi</i>	28

SKRAĆENICE

- IoT** – Pojam uređaja povezanih na internet (eng. *Internet of Things*)
- ANTLR** – Alat za parsiranje teksta (eng. *Another Tool For Language Recognition*)
- RGB** – Model za prikaz boja (eng. *Red Green Blue*)
- IP** – Internet protokol (eng. *Internet Protocol*)
- MQTT** – Protokol za asinhronu razmjenu poruka (eng. *Message Queueing Telemetry Transport*)
- AMQP** – Protokol za razmjenu poruka korišćenjem redova i ruta (eng. *Advanced Message Queuing Protocol*)
- AVS** – Amazonov servis za prepoznavanje govora (eng. *Alexa Voice Service*)
- TCP** – Protokol za upravljanje prenosom (eng. *Transmission Control Protocol*)
- API** – Sprega za programiranje aplikacija (eng. *Application Programming Interface*)
- HTTP** – Protokol za ramjenu podataka (eng. *Hyper Text Transfer Protocol*)
- JSON** – JavaScript objektna notacija (eng. *JavaScript Object Notation*)
- VCI** – Interfejs za prihvatanje glasovne komande (eng. *Voice Command Interface*)
- VCP** – Servis za parsiranje glasovne komade (eng. *Voice Command Parser*)
- VCE** – Servis za izvršavanje glasovne komande (eng. *Voice Command Executor*)

1. Uvod

U sadašnjem periodu progresivnog razvoja civilizacije dolazi do stalnog napretka i usavršavanja kompjuterskih tehnologija. Potražnja za elektronskim uređajima na tržištu iz dana u dan ostvaruje konstantan porast što se ispoljava u pristupačnijim cijenama i većoj dostupnosti navedenih uređaja.

Pojavljivanjem mobilnih mreža sa velikim brzinama prenosa javlja se mogućnost lakše integracije elektronskih komponenti u jedinstvenu cjelinu. Cijeli proces razvoja doveo je do stvaranja pojma “Internet stvari” (eng. IoT – *Internet of Things*).

IoT predstavlja proces umrežavanja fizičkih ili virtualnih uređaja korišćenjem naprednih komunikacionih tehnologija. Osnovni cilj implementacije jeste prikupljanje i razmjena informacija između raznih perifernih uređaja u cilju povećanja efikasnosti i optimizacije potrošnje. Njegova primjena se ogleda na mnogim poljima poput medicine, transporta, arhitekture, energetike, prehrambene i vojne industrije. Korišćenjem postojeće mrežne strukture čovjek počinje da komunicira sa svojim fizičkim okruženjem, te samim tim utiče na poboljšanje kvaliteta svakodnevnog života.

1.1 Pojam pametnih kuća

Pametna kuća je sistem kućne automatizacije zasnovan na potebi i želji korisnika da vlastiti dom učini komforntijim i sigurnijim za život. Koncept njene implementacije se zasniva na kontroli potrošnje energije, korišćenju ekoloških materijala prilikom njene izgradnje i očuvanju životne sredine. Definicija je različita u zavisnosti od profesije osobe koja je zadužena za njenu implementaciju. Sa strane projektanata softverskih ili hardverskih komponenti, pametne kuće predstavljaju jedinstvenu mrežu manjih mikroprocesorskih

kontrolera koji omogućavaju upravljanje i očitavanje stanja uređaja koristeći informacije koje se prenose unutar mreže.

Elementi kućne automatizacije se mogu podijeliti na kontrolere, senzore i aktuator. Kontroleri upravljaju sistemom, senzori očitavaju stanja pojedinih uređaja, a aktuatori su zaduženi za promjenu njihovih stanja.

Bilo da je riječ o korišćenju nekih od postojećih električnih instalacija ili je riječ o bežičnoj vezi, razmjena informacija između komponenti omogućava korisniku da upravlja cijelim sistemom. Korisnik putem mobilne aplikacije, upotrebom računara ili nekom drugom vrstom kontrolera može da mijenja ili očitava stanje svih uređaja koji se nalaze u “pametnoj” kući. Ovakvi sistemi imaju mogućnost da obavjeste vlasnika o bilo kakvom nedostatku ili promjeni unutar vlastitog doma čime smanjuju njegovu potrebu za stalnim nadzorom.

Iako ideja pametne kuće postoji već decenijama, pojava jeftinijih računarskih komponenti i razvoj naprednih tehničkih rješenja doveli su do njenog procvata u savremenom društву. U narednom periodu se očekuje stalni napredak kućne automatizacije i veća pristupačnost različitim slojevima društva.

1.2 Kontrola uređaja glasovnim komandama

Sistem kućne automatizacije zasnovane na glasovnoj kontroli je realizovan tako da korisnik izdavanjem **glasovne komande** može da mijenja i kontroliše stanje uređaja unutar vlastitog doma. Osnovna komponenta sistema jeste “pametni zvučnik” koji konstantno osluškuje korisničke zahtjeve, te u skladu sa izdatom komandom šalje naredbe drugim komponentama sistema u cilju izvršenja akcije.

Danas imamo mnoštvo takvih uređaja razvijenih od strane “moćnih” kompanija. Pored najpoznatijih **Amazon Echo-a** i **Google Home-a**, tu su još i **Microsoft-ova Cortana**, **Apple -ov Siri** i mnogi drugi. Svi oni predstavljaju pametne uređaje sa velikim brojem mikrofona, opremljene dobrom zvučnim prenosnicima radi imitacije što boljeg čovjekovog sagovornika i asistenta. Uglavnom su implementirani u svrhu pružanja zabave i informisanja ukućana o raznim događajima. Njihova integracija sa sistemom kućne automatizacije predstavlja značajan korak na polju IoT industrije. Uvezivanjem ovih komponenti dobijamo skladan sistem koji pruža mnoštvo novih mogućnosti. Upravljanje na daljinu smanjuje čovjekovu potrebu za stalnom interakcijom i neposrednom prisutnošću, što predstavlja ogromnu prednost kako kod starijih osoba tako i kod osoba sa zdravstvenim poteškoćama.

1.3 Integracija u OBLO sistem

U daljem radu je izložena implementacija *cloud* servisa namenjenog kontroli pametne kuće korišćenjem glasovnih komandi. Postojeći **OBLO cloud** sistem proširen je modulima za prihvatanje, parsiranje i obradu glasovne komande pristigle od strane korisnika. Rješenje je realizovano korišćenjem postojećih servisa poput Amazon Alexa i Google asistenta koji su zaduženi za detekciju i razumjevanje korisničkog zahtjeva.

Unutar teorijskih osnova izložen je prikaz OBLO sistema sa detaljnim objašnjenjem sastavnih komponenti. Opisane su i tehnologije korišćene za implementaciju, poput *RabbitMQ* brokera i *ANTLR* parsera, i sam princip rada postojećih servisa za detekciju glasa. Dalji rad se bazira na detaljnem opisu i implementaciji rješenja i daje uvid u rezultate dobijene procesom testiranja.

2. Teorijske osnove

Teorijske osnove daju prikaz korišćenih tehnologija i same arhitekture programske podrške koji su nepohodni za dalje razumjevanje integracije servisa u postojeći sistem. Na samom početku čitalac se upoznaje sa OBLO sistemom koji predstavlja postojeću arhitekturu unutar koje je realizovan napomenuti servis. U daljem tekstu su opisane tehnologije koje su korićene prilikom izrade različitih komponenti sistema.

2.1 OBLO sistem

OBLO sistem predstavlja skup nezavisnih komponenti koje međusobno komuniciraju u cilju kontrolisanja i nadgledanja uređaja unutar pametne kuće. Glavnu ulogu u sistemu predstavlja centralni kontroler (eng. *gateway*). Pored njega, sastavne komponente sistema čine uređaji, klijentska aplikacija i *cloud* servis.

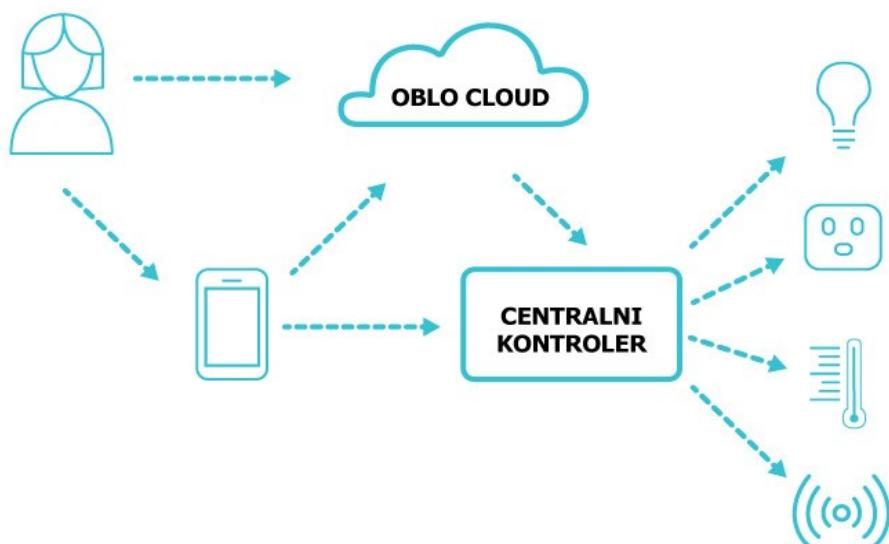
Centralni kontroler obavlja proces prihvatanja i obrade korisničke komande i upravlja događajima pristiglim od strane uređaja. On obezbjeđuje kako konfiguraciju i nadgledanje promjene stanja uređaja, tako i mogućnost njihove kontrole. Protokol koji se koristi prilikom razmjenjivanja poruka između kontrolera i uređaja je Zigbee [9] ili Z-Wave [10].

Komunikacija sa korisnikom se obavlja putem servisa za udaljeni pristup (eng. *cloud*). *Cloud* označava komponentu sistema koja se zasniva na poslužiocu koji ima mogućnost pristupa globalnoj mreži. Korisnik putem ove komponente u svakom trenuntku ima u vidu trenutno stanje uređaja kojim raspolaze. Takođe, u zavisnosti od raspoloživih funkcionalnosti uređaja korisnik može da mijenja njihova stanja.

Još jednu opciju za pristup korisnika sistemu nudi mobilna aplikacija. Mobilna aplikacija dobija raspoložive informacije o sistemu obavljajući komunikaciju sa centralnim kontrolerom ili pristupanjem *cloud-u*.

Posljednji element sistema čine uređaji. Oni se dijele u dve osnovne grupe koje čine aktuatori i senzori. Senzori predstavljaju komponente koje služe isključivo za očitavanje stanja, dok aktuatori daju mogućnost same kontrole. Trenutno podržani uređaji su:

- Utičnice
- Dimeri
- RGB svjetla
- Senzori za očitavanje temperature i vlažnosti vazduha
- Detektori poplave i dima
- IP kamere
- VOCO uređaj za reprodukciju zvučnog signala
- Roletne



Slika 2.1 Odnos između komponenti unutar OBLO sistema

2.1.1 Veza između *Cloud* servisa i centralnog kontrolera

Razmena poruka između *cloud* servisa i centralnog kontrolera se bazira na **MQTT** [6] protokolu. Ovaj protokol funkcioniše po principu “preplate” i “objave” na određene teme. Broker predstavlja centralnu komponentu sistema, čija uloga jeste isporuka sadržaja klijentima koji su pretplaćeni na poruke koje im pristižu. Klijent ostvaruje vezu sa brokerom korišćenjem TCP konekcije. Svako slanje zahtjeva korisnika propraćeno je odgovarajućim

periodom u okviru kog se očekuje odgovor. Takav način prenosa ne garantuje sigurnost prilikom isporuke. Moguća je pojava greške izazvane prekidom veze. Jedno od riješenja jeste mogućnost klijenta da se pretplati na obavještenja koja će mu dati informacije o promjeni unutar sistema.

2.2 Arhitektura zasnovana na odnosu proizvođač – potrošač

Koncept se zasniva na postojanju dva procesa, proizvođača i potrošača, koji dijele zajedničko parče memorije koje predstavlja red. Zadatak proizvođača je da generiše i proslijedi poruku u red, dok potrošač u isto vrijeme prihvata poruku iz reda i nastavlja sa daljim osluškivanjem.

Svrha korišćenja ovakve arhitekture jeste smanjenje opterećenja i perioda obrade unutar servisa. Prednost se ogleda u tome što potrošač može da počne sa obradom pristiglih informacija pritom ne znajući ništa o njegovim pošiljaocima, dok istovremeno proizvođač generiše nove poruke i proslijeđuje ih u red.

2.2.1 RabbitMQ

RabbitMQ je alat zadužen za rukovanje porukama (eng. *message broker*). Bazira se na AMQP [12] protokolu čiji zadatak jeste da obezbjedi prenos informacija između komponente sistema zadužene za slanje, i servisa koji obavlja prihvatanje poruka.

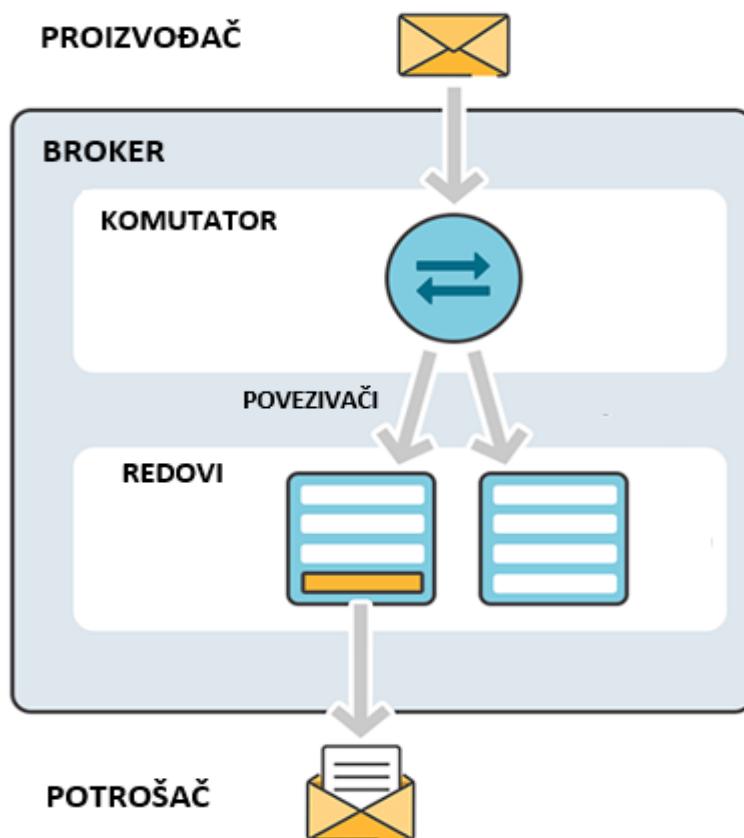
Rukovalac porukama, u ovom slučaju *RabbitMQ*, ima ulogu posrednika između proizvođača i potrošača. Informacije koje se razmjenjuju između ova dva servisa se prosleđuju brokeru koji je zadužen za njihovo rutiranje. Komponenta unutar brokera koja je odgovorna za njihov transfer u odgovarajuće redove naziva se **komutator** (eng. *exchange*). Pored komutatora, osnovne komponente brokera čine još i **red** i **povezivač**.

Komutator vrši isporuku informacija na osnovu ključa rutiranja. Ključ rutiranja predstavlja atribut koji proizvođač pridodaje poruci u cilju tačne isporuke. Postoji više vrsta komutatora na osnovu čijih uloga razlikujemo sledeće vrste:

- Direktni - omogućava isporučivanje poruka samo onim redovima čiji ključ povezivanja odgovara ključu rutiranja.
- *Fanout* - vrsta komuadora koja isporučuje poruke svim postojećim redovima
- Tematski - informacije se isporučuju jednom ili više redova u zavisnosti od podudarnosti ključa rutiranja i obrasca

- Zaglavje - bazira se na parametrima iz zaglavlja koji predstavljaju argument na osnovu kojih se vrši rutiranje

Da bi poruka bila poslata u red, povezivač mora da obezbjedi vezu reda sa komutatorom. Kada su ove dve komponente povezane tek tada je moguće proslijediti informacije u red. Po dospijevanju poruke u red potrošač dobija obavještenje i ukoliko je trenutno slobodan uzima poruku iz reda i počinje njenu obradu. Još jedna mogućnost koju *RabbitMQ* nudi jeste “*Dead Letter Exchange*”. Ova komponenta obezbeđuje hvatanje poruka koje nisu rutirane i tako daje mogućnost lakšeg uvida u problem.



Slika 2.2 Odnos između komponenti unutar RabbitMQ broker-a

Veza sa brokerom se ostvaruje posredstvom jedne od mnoštva biblioteka koje olakšavaju pristup funkcionalnostima RabbitMQ rukovaoca. Ukoliko se koristi programski jezik *JavaScript*, biblioteka koja se izdvaja je *amqplib* [11].

Kada je veza sa brokerom uspostavljena, potrebno je kreirati kanal koji je zadužen za prenos svih informacija u okviru navedenog rukovaoca. Unutar kanala deklarišemo redove, komutatore i druge komponente koje međusobno komuniciraju. Prilikom konfiguracije ovih komponenti, možemo postaviti dodatne parametre koji definišu njihovo trajanje, pravo pristupa, veličinu zauzete memorije i dr.

2.3 ANTLR parser

ANTLR (*Another Tool for Language Recognition*) [5] je alat koji se koristi za konvertovanje proizvoljnog teksta u format pogodan za dalju obradu.

Svrha korišćenja ovog parser generatora u okviru datog rješenja jeste pretvaranje slobodne forme teksta dobijene od strane nekog od servisa za prepoznavanje govora u strukturiranu formu pogodnu za izvršavanje određene akcije u sistemu pametne kuće.

Na samom početku, potrebno je generisati gramatički fajl u kom definišemo semantička pravila koja naša komanda mora da zadovolji.

```
grammar Html;

root : html EOF;
html : ( italic | normal ) *;
italic : '<i>' html '</i>';
normal : TEXT;
TEXT : ~[<>]+;
```

Slika 2.3 Primjer generisane gramatike

Svako *ANTLR* gramatičko pravilo se sastoji iz naziva i definicije. Ukoliko je naziv proizvoljan odnosno ne predstavlja definisano leksičko pravilo piše se malim slovom, nakon čega slijedi dvotačka i potom definicija. Razlikujemo terminalne i neterminalne simbole. Neterminalni simboli moraju da budu napisani malim slovima, i označavaju gramatičke elemente koji imaju sopstvenu strukturu i imena (sa slike 2.3 to su *html*, *italic*, *normal*). Terminalni simboli označavaju cifre ili riječi koje same po sebi nemaju definisanu strukturu (sa slike 2.3 to su '*<i>*', *TEXT*), pa se nalaze pod navodnicima ili su napisani velikim slovima.

Dodatni znakovi koji se takođe koriste su:

- * - označava da se sekvenca ne ponavlja nijednom ili se pojavljuje više puta
- ? - označava ponavljanje sekvence jednom ili više puta
- | - predstavlja alternativu između više ponuđenih solucija
- + - znak da se sekvenca ponavlja više puta
- . - zamjenjuje bilo koji znak
- .. - predstavlja interval
- ~ - znak za inverziju

Unutar slike 2.3 koja predstavlja primjer gramatičkog fajla, polje *root* označava ulaznu tačku programa. Tu je definisano pravilo u kom formatu treba da se nalazi ulazni tekst koji se dalje šalje na parsiranje. Na početku obrade, lekser na osnovu definisane gramatike vrši proces grupisanja gramatičkih elemenata u niz tokena pri čemu svako pravilo predstavlja jedan token. Sledeća komponenta koja se aktivira jeste parser.

Parser koristi ulazni tekst i predhodno generisane tokene i na osnovu toga formuliše sintaksno stablo. Svaki token označava jedan čvor, pri čemu su svi čvorovi vezani za korijen stabla koji sam po sebi nema nikakvo značenje. Pristupanjem elementima stabla, dobijaju se sve potrebne informacije nastale procesom parsiranja koje možemo iskoristiti u našoj daljoj implementaciji.

ANTLR se od mnoštva sličnih alata izdvaja po svojoj brzini, dobro realizovanom rukovanju greškama i velikoj raznolikosti na polju podržanosti od strane različitih programskih jezika.

2.4 Servisi zaduženi za prepoznavanje govora

Servisi koji nam nude mogućnost prepoznavanja govora i njegovog vraćanja u vidu tekstualne forme pretstavljaju osnovu za implementaciju datog rješenja. Iz mnoštva sličnih alata, dva koja se posebno izdvajaju na osnovu dodatnih funkcionalnosti koje nude i vještine prepoznavanja jesu **Amazon Alexa** i **Google Asistent**.

2.4.1 Amazon Alexa

Aleksa predstavlja Amazonov servis za glasovnu kontrolu. Ovaj servis nudi mogućnost razumjevanja glasovne komande korisnika, te u skladu sa prepoznatim, šalje odgovarajući odgovor. **AVS** (eng. *Alexa Voice Service*) [7] je implementiran tako da oponaša stvarnu konverzaciju sa korisnikom. Aleksa se osnjava na internet vezu što samim tim nudi mnoštvo mogućnosti u vidu informisanja, pružanja zabave i integracije unutar komponenti pametne kuće.

Da bi moglo da se pristupili datom servisu mora se ostvariti veza sa nekim od uređaja koji zadovoljavaju određene uslove neophodne za njegovu integraciju. Platforme razvijene u ovu svrhu jesu Amazonovi uređaji *Echo*, *Echo Dot* i *Tap*. Njihova moć prepoznavanja zavisi od sedam veoma osjetljivih mikrofona koji se nalaze unutar navedenih komponenti. Sistem je implementiran tako da konstatno osluškuje korisnika, a pokreće se izgovaranjem ključne riječi *Alexa*, *Amazon*, *Echo* ili *Computer*. Kada je izgovorena ključna riječ, mikrofon prikuplja sve podatke i šalje ih na Amazonov *cloud* gdje se izvršava navedena komanda.

Amazon nudi mogućnost integracije Aleksa servisa sa pojedinim komponentama pametne kuće. Ta funkcionalnost se odnosi samo na određene uređaje koje navedeni servis podržava. U podržane uređaje spadaju: *Philips* svjetla, *Nest* termostati, *SmartThing* utičnice, *Lifx* sijalice i mnogi drugi.

Ukoliko naša pametna kuća nije povezana sa Aleksom, Amazon nam omogućava kreiranje vlastite aplikacije (eng. *skill*), unutar koje ćemo implementirati željenu funkcionalnost. Prilikom izrade navedene funkcionalnosti potebno je definisati komunikacioni model za koji ćemo nagovjestiti format i moguće nazive svih pojedinačnih komponenti korisničke komande. Da bi aplikacija uspješno radila, osoba koja je projektuje mora pokriti sve primjere izrečenih zahtjeva koji omogućavaju izvršavanje različitih akcija. Ukoliko neka od naredbi nije definisana, a izgovorena je od strane korisnika, komanda se u startu odbacuje. Ovaj proces inicijalizacije čini ograničavajući faktor prilikom upotrebe i iziskuje mnogo posla sa programerskog stanovišta. Bez obzira na date mane, navedenom implementacijom se povećava razumljivost i omogućava se korišćenje što prirodnijih komandi, što i jeste osnovni cilj njene upotrebe.



Slika 2.4 Amazon Echo, Dot i Tap uređaj

2.4.2 Google Asistent

Google Asistent je virtuelni asistent razvijen od strane kompanije *Google* koji obavlja glasovnu komunikaciju sa korisnikom. U početku se primjenjivao isključivo u okviru aplikacije *Allo* koja služi razmjeni poruka. Ubrzo postaje dostopan kao alat za korisnike *Pixel* mobilnih telefona. Polje djelovanja se potom proširilo i na *Android* uređaje, a razvili su i sopstvenu platformu namjenjenu isključivo za interakciju sa korisnikom koju su nazvali *Google Home*.

Google Home predstavlja uređaj koji rukovodi glasovnim komandama oslanjajući se na dva mikrofona koja prikupljaju korisničke informacije. Korišćenjem ove platforme, korisnik

može da se informise o raznim događajima kao i da zahtjeva obavljanje određene akcije, poput reprodukcije muzičkog sadržaja ili kontrole uređaja unutar postojeće *cloud* strukture. Naredba se izgovara prozivanjem uređaja sa “*Ok, Google*” ili “*Hey, Google*”.

U decembru 2016. godine, *Google* je plasirao na tržište ***Actions on Google*** koji predstavlja razvojnu platformu za Google Asistenta [8]. U okviru ove platforme se nudi mogućnost projektantima da razviju sopstveni hardver na kojem će da se izvršava Google Asistent. Takođe, unutar postojeće platforme korisnik može da integriše sopstvenu aplikaciju (akciju) koja služi zabavi, pribavljanju informacija, ili kontroli uređaja, te da na taj način proširi mogućnosti uređaja na kojima se Google Asistent izvršava. Prilikom kreiranja akcije potrebno je navesti njen naziv, adresu izvršavanja i sve zajedno otpremiti na razvojni projekat. Funkcionalnost postaje dostupna njenim ubacivanjem u servis sekciiju na *Google Home* aplikaciji.

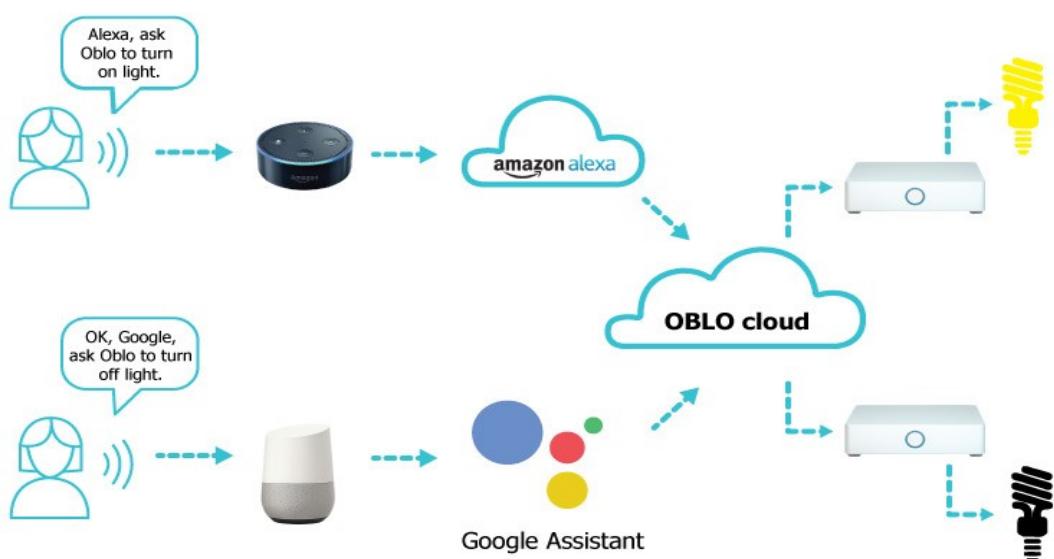


Slika 2.5 *Google Home* uređaj

3. Koncept rješenja

Unutar predhodnih cjelina rada izložene su osnove neophodne za razumjevanje ponuđenog rješenja. Intergracija servisa baziranog na obradi glasovne komande korisnika unutar OBLO sistema predstavlja još jednu dodatnu funkcionalnost koju je bilo važno implementirati. Uvezivanjem svih komponenti iz teorijskih osnova dobijamo funkcionalnu cjelinu koja nudi mogućnost korisniku da glasovnim komandama upravlja uređajima unutar vlastite pametne kuće.

Nastavak rada se bazira na iznošenju rješenja u vidu implementacije servisa zaduženih za prihvatanje, parsiranje i izvršavanje korisničke komande.



Slika 3.1 Prikaz putanje glasovne komande korisnika

3.1 Arhitektura na kojoj se zasniva ponuđeno rješenje

Rješenje je implementirano definisanjem tri servisa. Prvi koji se pokreće jeste interfejs zadužen za komunikaciju sa nekim od servisa koji obrađuju glasovnu komandu [7][8]. Pored ovog, imamo još i servis za parsiranje pristiglog zahtjeva korisnika i servis koji izvršava naredbu koja je pristigla. Svaka komponenta obavlja tačno određenu funkcionalnost i u potpunosti je nezavisna od druge dve.

Nezavisnost između modula se zasniva na arhitekturi koju čine **potrošač** i **proizvođač**. Zadatak proizvođača je da poruku proslijedi potrošaču, a da pri tome nema u vidu kako je on implementiran. Posrednik u prenosu informacija između ove dve komponente je broker. Veza između nase aplikacije i brokera se ostvaruje korišćenjem klijentske biblioteke *amqplib* koja predstavlja API za pristup *RabbitMQ* rukovaocu. Njegov zadatak je da rukuje redovima i drugim dijelovima neohodnih za obavljanje uspješne komunikacije.

Sistem je implementiran tako da svaki servis istovremeno ima ulogu i proizvođača i potrošača. Svaki od servisa ima poseban red u koji mu pristižu poruke. Potrošac se "pretplaćuje" na poruke unutar vlastitog reda i kada je slobodan kreće sa njihovom obradom. Nakon što je obradio poruku, servis je dalje prosljeđuje novom redu za koji je zadužena neka druga komponenta sistema. Veličina reda može da se mijenja u zavisnosti od raspoloživih resura.

Prednost korišćenja ovakve arhitekture je bolja izbalansiranost opterećenja između komponenti. Pokretanjem više instanci implementiranih servisa omogućuje se da više rukovalaca pristupa istom redu, što smanjuje proces čekanja da poruka bude obrađena.

3.2 Uvezivanje servisa za glasovnu kontrolu sa OBLO *cloudom*

Da bi korisnik mogao da koristi glasovne komande unutar OBLO sistema neophodno je da ima nalog na *Amazon* ili *Google* servisu. U tom slučaju, moguće je kreirati akciju koja će omogućiti dalju funkcionalnost. Konfiguracijom svih potrebnih parametara prilikom formiranja iste, dobija se funkcionalan glasovni servis koji može da se implementira unutar *cloud-a*. Veza između OBLO *clouda* i nekog od navedenih servisa se ostvaruje povezivanjem naloga.

Servis koji je zadužen za izvršavanje same akcije jeste **servis za izvršavanje korisničkog zahtjeva**. Unutar ovog modula je implementirana sva potrebna funkcionalnost

kojoj predhodi izvršavanje korisničke naredbe. Tu se ostvaruje veza sa centralnim kontrolerom i uređajima u okviru pametne kuće.

Na primjeru korišćenja uređaja *Amazon Echo Dot* pokretanje akcije se vrši izgovaranjem ključne riječi “**Alexa, ask Oblo to ...**”, nakon čega slijedi zahtjev koji korisnik očekuje da se izvrši. Primjer izgovorene komande glasi “*Alexa, ask Oblo to turn on light*”. Oblo predstavlja naziv implementirane akcije. Naredbe je moguće izdavati na engleskom ili njemačkom jeziku u slučaju Alekse, dok Google Asistent podržava isključivo engleski jezik. Prilikom obraćanja *Google Home* uređaju, princip izdavanja naredbe je u potpunosti isti, osim što mu se obraćamo sa “**Ok Google**”.

Nakon što je zahtjev poslat nekom od servisa zaduženim za obradu glasovne komande, odgovor u tekstualnom formatu se prosljeđuje nasem *cloud-u*. Poruka pristiže na servis koji obavlja prihvatanje korisničke naredbe. Zadatak ove komponente je da rutira poruke u zavisnosti od formata u kom su pristigle. Ukoliko je nosilac zahtjeva Amazon Alexa, poruka se prosljeđuje servisu zaduženom za njeno izvršavanje, dok se u suprotnom ona upućuje na modul koji vrši njeno parsiranje.

3.3 Integracija parser generatora u postojeći sistem

Poruka pristigla od *Google* servisa za prepoznavanje govora sadrži tekstualni prikaz korisničke naredbe u formi izvornog teksta. Sa strane implementacije, potrebno je obezbjediti format poruke u kojoj je komanda raščlanjena u manje cjeline. Strukturirana poruka treba da sadrži informacije o imenu uređaja, akciji koja treba da bude izvršena, kao i nazive dodatnih opisnih parametara poput imena sobe, sprata, vrijednost procenta, boje i dr. Iz ovih razloga upotreba parser generatora predstavlja logičan izbor unutar postojeće arhitekture.

Njegova implementacija je realizovana unutar **servisa zaduženog za parsiranje komande**. Poruke pristigle od Amazon Alekse se ne upućuju na ovaj servis jer stižu u formatu koji nije potrebno dalje obrađivati. Da bi imali što jedinstveniji kod unutar komponente sistema zadužene za izvršavanje komande, modul koji parsira korisnički zahtjev mora da generiše strukturu poruke koja u potpunosti odgovara onoj dobijenoj korišćenjem Amazonovog servisa.

4. Programsко rješenje

4.1 Interfejs za prihvatanje glasovne komande

Modul za prihvatanje korisničkog zahtjeva je zadužen za komunikaciju sa servisima za prepoznavanje glasa. Poruke poslane od strane ovih servisa pristižu na jednu od dve formirane rute unutar *cloud-a*. Korišćenjem *HTTP POST* metode, Amazon prosljeđuje tekstualni format glasovne komande na rutu *vci/alexa*.

```
{
  "intent": {
    "name": "ZoneIntent",
    "slots": {
      "Zone_Requests": {
        "name": "Zone_Requests",
        "value": "turn on"
      },
      "Zone_Things": {
        "name": "Zone_Things",
        "value": "all lights"
      },
      "Floors": {
        "name": "Floors",
        "value": "floor"
      },
      "Rooms": {
        "name": "Rooms",
        "value": "kitchen room"
      },
      "FloorDescriptor": {
        "name": "FloorDescriptor",
        "value": "1st"
      }
    }
  }
}
```

Slika 4.1 Primjer JSON poruke pristigle od AVS-a

Što se tiče *Google*-ovog servisa, ruta za slanje zahtjeva je *vci/google*.

```
{
  "inputs": [
    {
      "intent": "assistant.intent.action.MAIN",
      "raw_inputs": [
        {
          "input_type": 2,
          "query": "ask oblo to turn on all lights in kitchen at first floor",
          "annotation_sets": []
        }
      ]
    }
}
```

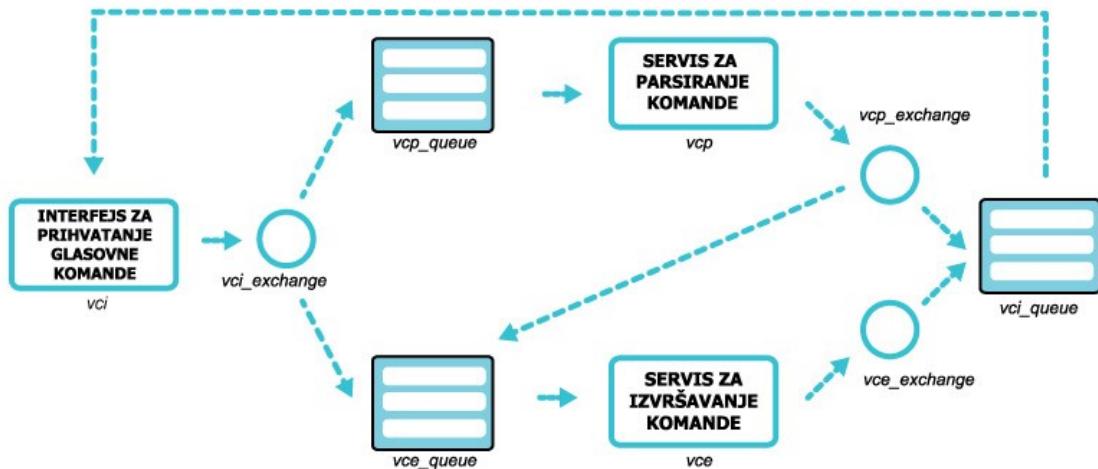
Slika 4.2 Primjer JSON poruke pristigle od Google Asistenta

Novopristigle poruke se šalju funkciji zaduženoj za njihovo dalje rutiranje u odgovarajući red. Pokretanjem funkcije, generiše se proizvoljni identifikator koji se kači na pristigli zahtjev. Formiranje identifikatora se realizuje da bi sistem znao na koju adresu da pošalje odgovor u skladu sa izrečenom naredbom.

Generisana poruka se šalje na dalju obradu u odgovarajući red u zavisnosti od rute na koju je pristigla. Ukoliko je nosilac naredbe Amazon Aleksa ona se direktno prosljeđuje servisu za obradu komande, dok se u suprotnom upućuje na modul zadužen za njenu parsiranje.

Implementirani interfejs ima zadatak i da konstantno osluškuje na odgovore koji pristižu od ova dva modula i na osnovu uspješnog ili neuspješnog izvršavanja obavjesti korisnika o ispravnosti izrečene naredbe.

4.2 Implementacija komunikacije između servisa



Slika 4.3 Komunikacija između implementiranih servisa

Na samom početku izrade ovog modula neophodno je unutar “*config*” fajla definisati nazive redova, komutatora i drugih sekvenci potrebnih za dalju implementaciju. Prilikom pokretanja *cloud-a*, vrši se uspostavljanje *TCP* konekcije između naše aplikacije i *RabbitMQ* brokera. Na osnovu uspostavljenje veze formira se kanal koji predstavlja glavni put za prenos informacija između servisa. Unutar navedenog kanala se vrši kreiranje i inicijalizacija komponenti navedenih u “*config*” fajlu. Odmah nakon što su formirani, redovi počinju da osluškuju na novopristigle poruke.

Nakon što je zahtjev pristigao na određenu rutu, servis za prihvatanje komande ima zadatku da ga proslijedi jednom od modula. U zavisnosti od rute na koju je zahtjev stigao, prenos se obavlja posredstvom drugog komutatora. Zadatak komutatora je da rutira poruke u odgovarajuće redove na osnovu ključa rutiranja. Ukoliko je naredba stigla na rutu *vci/alexa*, komutator koji je zadužen za njeno rutiranje je *vce_exchanger*, sa ključem rutiranja *vce*. Što se tiče zahtjeva pristiglog sa rute *vci/google*, komutator kojem se obraćamo je *vcp_exchanger*, a ključ rutiranja *vcp*. Uvezivanje reda sa komutatorom se definiše prilikom njegovog

formiranja. Tom prilikom se označava i ključ rutiranja na osnovu kojeg će komutator znati kojem redu da isporuči poruku.

Kada je poruka pristigla u odgovarajući red, potrošač, koji je u ovom slučaju servis za parsiranje ili izvršavanje datog zahtjeva, započinje dalju obradu. Ako je komanda proslijedena redu za parsiranje (*vcp_queue*), modul za njenu obradu će je prihvati i izvršiti njen raščlanivanje na podsekvence u skladu sa definisanim formatom. Ukoliko je proces uspješno završen, isparsirana naredba u **JSON formatu** se proslijeđuje vce_exchanger-u.

Servis za izvršavanje komande prihvata poruke pristigle u *vce_queue*. Nezavisno od toga da li je poruka došla direktno iz servisa zaduženog za prihvatanje glasovne naredbe ili iz parsera, modul za izvršavanje korisničke komande će obaviti proces komunikacije sa centralnim uređajem u cilju izvršenja akcije i na osnovu toga formirati dati odgovor. Odgovor se proslijedi redu za čije rukovanje je zadužen modul koji ostvaruje vezu sa servisima od kojih je naredba pristigla (Amazon Aleksa ili Google Asistent). Na osnovu identifikatora koji se nalazi u poruci dobijaju se informacije o tome na koji zahtjev odgovor treba da se proslijedi. Isto tako, poruka će dospjeti u navedeni red ukoliko se tokom pretrage uspostavi da je korisnička komanda neispravna, čime parser ili servis za izvršavanje prekidaju dalju pretragu i šalju podatak o njenoj neispravnosti. Ukoliko je neka od poruka zalutala, odnosno nije uspješno isporučena odgovarajućem redu, ona biva proslijedena *dead_letter_exchange* komponenti koja je zadužena za slanje obavještenja o mogućem problemu.

Kad je akcija izvršena, odgovor je poslat i jedna korisnička sesija je završena. Nove komande pristižu i obrada se nastavlja.

4.3 Parsiranje komande

Kada je poruka pristigla u red za parsiranje korisničkog zahtjeva, servis zadužen za njenu obradu se aktivira i započinjanje proces parsiranja. Poruka se proslijeđuje parser generatoru koji će u skladu sa definisanom gramatikom da je raščlani na manje komponente. Gramatički fajl sadrži informacije o očekivanom formatu naredbe. Svaka komanda u skladu sa svojim značenjem mora da ima tačno definisane komponente koje je opisuju.

Komande podržane u sistemu su:

- paljenje i gašenje uređaja
- mijenjanje nivoa osvjetljenosti
- promjena boje na RGB sijalici
- očitavanje stanja (temperature, vlažnosti) određenih senzora
- mijenjanje stanica na VOCO uređaju

- pomjeranje roletni
- aktiviranje određene scene

Polja unutar komande	Primjer komande
TurnOnOff_Request + Device_name	<i>Turn on plug.</i>
Dim_Request + Device_name + Percentage	<i>Set dim to fifty.</i>
Hue_Request + Device_name + Color	<i>Change lamp to red.</i>
Temperature_Request + Device_name	<i>Get temperature from sensor.</i>
Humidity_Request + Device_name	<i>Get humidity from sensor.</i>
Zone_Request + Zone_Device	<i>Turn on all lights.</i>

Tabela 4.1 Potrebna polja u zavisnosti od tipa komande

U tabeli 4.1 je navedeno koja polja svaka naredba mora da sadrži. Pored osnovnih zahtjeva, postoji još mnoštvo opisnih polja koja se definišu u zavisnosti od pozicije i imena uređaja koje je korisnik definisao. Ukoliko smo uređaj nazvali stono svjetlo i stavili ga u radnu sobu na prvom spratu, potrebno je da sve to navedemo prilikom izgovaranja komande za njegovo paljenje. U skladu sa tim dobijamo niz pojedinačnih komponenti koje zajedno čine jednu naredbu (tabela 4.2).

Polje	Vrijednost
Request	<i>Turn on</i>
Description name	<i>Desk</i>
Device name	<i>Light</i>
Room description	<i>Work</i>
Room name	<i>Room</i>
Floor description	<i>First</i>
Floor name	<i>Floor</i>

Tabela 4.2 Rašlanjivanje komande na podsekvence

Sva polja i mogući nazivi su definisani u gramatičkom fajlu. Kada je poruka poslata na obradu, parser prolazi kroz niz tokena koji su predhodno generisani od strane leksera, i provjerava da li sekvence izgovorene naredbe odgovaraju definisanim obrascima poruke. Svaka podudarnost rezultuje pokretanjem odgovarajuće funkcije unutar koje imamo pristup

sekvenci izgovorene komande koja odgovara određenoj vrijednosti polja. Ukoliko naredba koja je došla od korisnika nije u očekivanom formatu, tj. ne zadovoljava ni jednu od navedenih formi, pravi se *JSON* poruka u kojoj se nalaze podaci o njenoj neispravnosti. Ta informacija se šalje servisu zaduženom za prihvatanje komande koji će korisnika obavjestiti o nepostojanju izrečenog zahtjeva.

U suprotnom, ukoliko je komponenta unutar komande sadržana u dotoj definiciji, njena vrijednost se zapisuje u okviru *JSON* objekta koji će biti prosleđen servisu zaduženom za njeno izvršavanje. Format *JSON* poruke se formuliše tako da u potpunosti odgovara poljima ispasiranog zahtjeva koji dolazi direktno od Amazon servisa. Dodatni objekat poput broja sesije, identifikatora zahjeva, naziva pošiljaoca i drugih opisnih parametara koji se vezuju za korisnika se kače na novoformljeni *JSON* i bivaju prosljeđeni sledećoj komponenti sistema zaduženoj za njenu obradu.

4.4 Izvršavanje korisničke naredbe

Da bi naredba bila izvršena potrebno je provjeriti da li korisnik u svom sistemu ima navedeni uređaj i da li je izrečena akcija u skladu sa njegovom funkcionalnošću. Pristupanjem centralnom kontroleru dobija se uvid u cijelokupno stanje komponenti sistema kojim korisnik raspolaze. Prolaskom kroz datu listu, provjerava se da li je unutar korisničke komande spomenut jedan od uređaja iz liste. Ukoliko je pretraga uspješno obavljena, potrebno je provjeriti da li je izrečena akcija (npr. promjena nivoa osvetljenosti) podržana u okviru navedene komponente. O funkcionalnosti pojedinačnog uređaja saznajemo iz liste podržanih servisa. **Postojeći servisi** u sistemu su:

- *Outlet* servis – određuje stanje utičnice, ima vrijednost 0 ili 1
- *Light* servis – određuje stanje sijalice, ima vrijednost 0 ili 1
- *Hue* servis – zadužen za mijenjanje boje i zasićenja kod RGB sijalice
- *Dim* servis – omogućava promjenu nivoa osvetljenosti RGB ili dim sijalice
- *Temperature* servis – očitava temperaturu na senzoru
- *Humidity* servis – očitava vlažnost vazduha
- *Smoke* servis – očitava pristutnost gasa u prostoriji
- *VocoContent* servis – zadužen za mijenjanje stanica na VOCO uređaju
- *Volume* servis - kontroliše nivo zvuka na VOCO uređaju
- *RollerShutter* servis – upravlja pozicijom roletni

Ukoliko akcija unutar izrečene komande odgovara nekom od navedenih servisa, MQTT poruka sa novom vrijednošću uređaja se proslijeđuje centralnom kontroleru koji je zadužen za njeno izvršavanje.

Još neke od mogućih kontrola su i aktiviranje scene, kontrola grupe uređaja unutar prostorije, i dr. Koncept realizacije je u principu dosta sličan prethodnom. Kada je riječ o aktiviranju scene, prolazimo kroz listu naziva svih scena koje je korisnik oformio i šaljemo naredbu za njeno aktiviranje. Što se tiče kontrole grupe uređaja, ona se svodi na formiranje grupa na osnovu podržanih servisa. Tako je moguće paliti ili gasiti sva svjetla u jednoj zoni, kontrolisati njihov nivo osvjetljenosti ili istovremeno upravljati svim roletnama u prostoriji.

Ako je utvrđeno da je korisnik izrekao ispravnu komandu, generiše se odgovor o uspješnoj promjeni stanja uređaja. Formira se određeni *JSON* u zavisnosti od toga da li je zahtjev došao od Amazonovog ili Google-ovog servisa, i poruka se proslijeđuje redu kojim rukovodi sistem za prihvatanje komande.

Ukoliko u nekom od prethodnih koraka pretraga nije dala potvrđne rezultate, korisnik se obavljaštava o nepravilnosti izrečene naredbe.

5. Rezultati

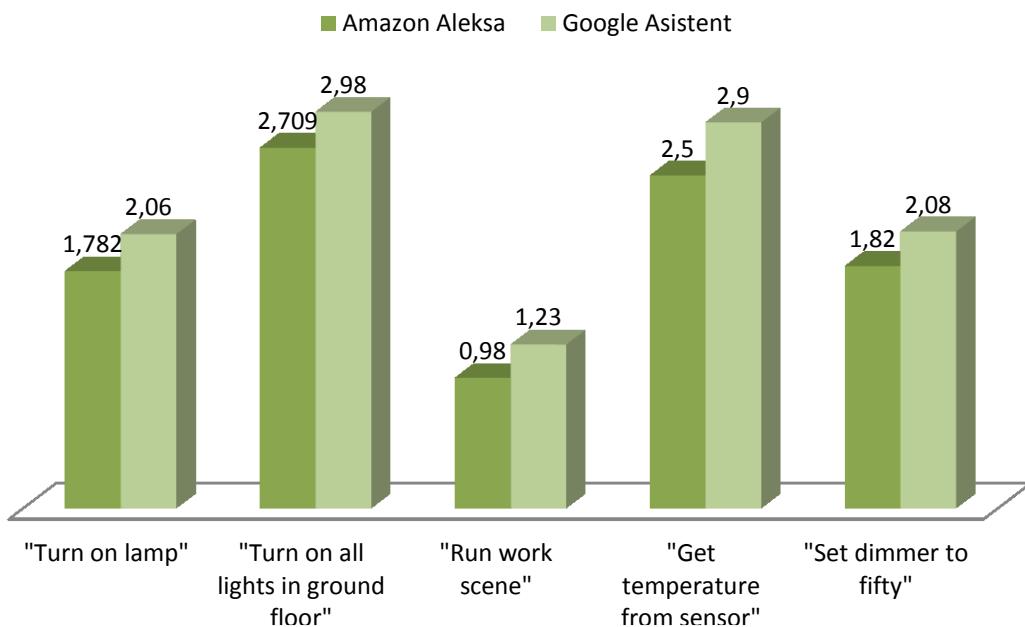
5.1 Testiranje korišćenjem uređaja za prepoznavanje glasa

Uređaji koji su korišćeni u svrhu testiranja su Amazon Echo i mobilni telefon na kom je podržan Google Asistent. Tri osobe izgovorile su oko pedeset komandi u cilju dobijanja rezultatata u vidu vremena odziva i potvrde uspješnosti prilikom promjene stanja uređaja. Vrijeme odziva je mjereno od trenutka nakon što je izgovorena komanda do **dobijanja odgovora** od strane uređaja sa kojim se komunicira.

Izgovorena komanda	Dobijeni odgovor	Uspješnost izvršavanja
<i>Turn on lamp.</i>	<i>The lamp is turned on.</i>	✓
<i>Turn on all lights in ground floor.</i>	<i>All lights are turned on.</i>	✓
<i>Run work scene.</i>	<i>Work scene is run.</i>	✓
<i>Get temperature from sensor.</i>	<i>Temperature from sensor is 26 degrees.</i>	✓
<i>Set dimmer to fifty.</i>	<i>Dimmer is set.</i>	✓

Tabela 5.1 Testiranje izgovaranjem glasovnih komandi

Oba servisa za prepoznavanje govora ostvarila su visok stepen razumjevanja koji iznosi preko 70%. Kada je komanda dobro prepoznata svaki put je generisan odgovarajući odgovor i akcija je uspješno izvršena.



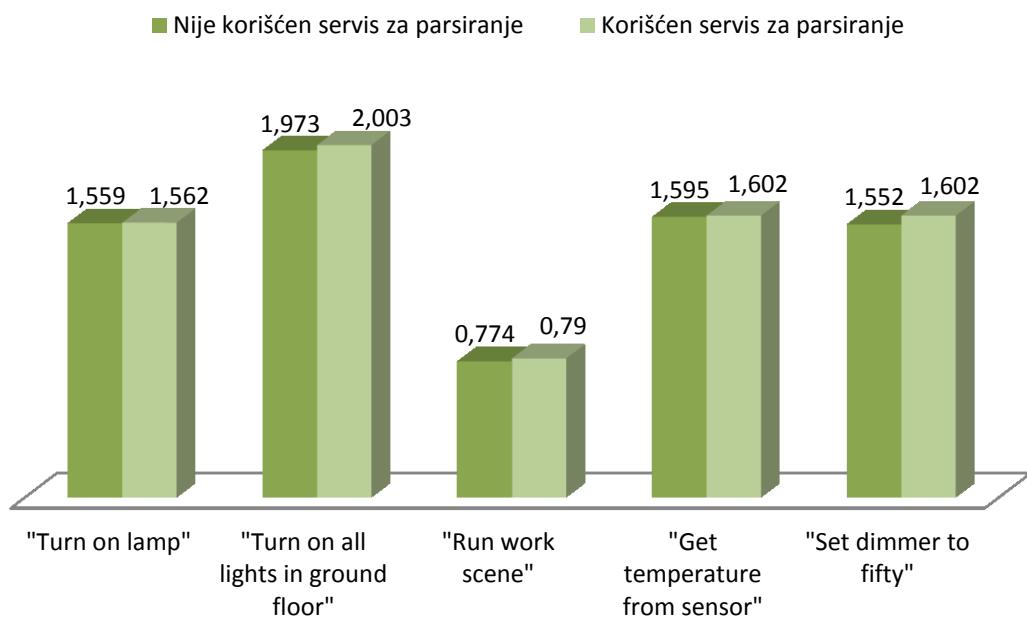
Slika 5.1 Ukupno vrijeme izvršenja glasovne komande korišćenjem Amazon Alekse i Google Asistenta

Rezultati sa slike 5.1 predstavljaju ukupno vrijeme izvršenja glasovne komande uračunavajući obradu od strane Amazon i Google servisa za prepoznavanje korisničkog zahtjeva tako i obradu od strane servisa zaduženih za njeno izvršavanje. Analiziranjem dobijenih podataka zaključujemo da je period odziva zadovoljavajući i da zavisi isključivo od broja komponenti unutar izgovorene naredbe.

5.2 Testiranje uz pomoć **ApacheBench** alata

ApacheBench [13] je softver zadužen za mjerjenje performansi HTTP web servera. Navedeni alat je korišćen u cilju dobijanja rezultata koji se odnose isključivo na svojstva implementiranih servisa. Zanemirivanjem glasovnih komponenti dobija se pravi uvid u performanse realizovanog sistema.

Testiranje je izvršeno slanjem deset uzastopnih komandi na rute unutar OBLO *cloud*-a u kojima se nalaze servisi zaduženi za obradu datih informacija. Poruke su slane u *JSON* formatu koji u potpunosti odgovara tekstualnoj formi obrađene komade dobijene uz pomoć servisa za prepoznavanje govora. Rezultati sa slike 5.2 su pokazali pozitivne strane korišćenja navedene arhitekture, te dokazali da upotreba parser generatora ne dovodi do značajnijeg kašnjenja.



Slika 5.2 Ukupno vrijeme izvršenja tekstualne komande, sa i bez korišćenja servisa za parsiranje

Treba uzeti u obzir da je u daljem radu neophodno izvršiti obimnije testiranje jer postoji mnoštvo parametara koji mogu narušiti funkcionalnost datog sistema. Na primjer, struktura mreže preko koje protokoli komuniciraju može dodatno usporiti komunikaciju.

6. Zaključak

Integracija postojećih glasovnih servisa unutar *cloud-a* predstavlja suštinu ovog rada. Cilj date implementacije jeste proširenje postojeće funkcionalnosti OBLO sistema pružanjem mogućnosti korisniku da kontroliše uređaje izdavanjem glasovnih komandi. Upravljanje uređajima na ovaj način je vrlo atraktivno i interesantno sa korisničke strane, a prije svega pruža mnoge olakšice osobama sa zdravstvenim poteškoćama.

Realizacija sistema se zasniva na implementaciji servisa koji vrše prihvatanje i obradu teksutalnog formata glasovne komande pristigne od strane korisnika. Prednost korišćenja datog rješenja se ogleda u nezavisnosti imlementiranih servisa što smanjuje vrijeme obrade prilikom korišćenja višeprocesorskih računara. Da bi data funkcionalnost bila ostvarena, neophodno je da korisnik posjeduje jedan od uređaja kao što je *Google Home* ili *Amazon Echo*.

Rezultati su pokazali da je period odziva zadovoljavajući od strane korisnika. Upotreba modula za parsiranje izaziva minimalno kašnjene u odnosu na poruke koje se direktno šalju servisu zaduženom za izvršavanje komande. Postojeće tehnologije zasnovane na detekciji glasa pokazuju visoku moć prepoznavanja ali i ostavljaju dosta prostora za dalji napredak. Očekuje se i proširenje na polju podržanih jezika.

Budući koraci rada na projektu će se fokusirati na **integraciju drugih servisa** koji nude mogućnost konvertovanja glasovne komande u tekstualni format. Takođe, postoji i mogućnost razvoja vlastitog uređaja koji će obavljati snimanje korisničkih zahtjeva i reprodukciju odgovora sistema.

7. Literatura

- [1] Richardson Alexis, “*RabbitMQ—An open source message broker that just works*”, Qcon, 2009
- [2] Videla Alvaro, J.W.Williams Jason, “*RabbitMQ in action*”, Manning, 2012
- [3] David Dossot, “*RabbitMQ Essentials*”, Packt Publishing, 2015
- [4] Sigismondo Boschi, Gabriele Santomaggio, “*RabbitMQ Cookbook*”, Packt Publishing, 2013
- [5] T.Parr, “*The definitive ANTLR 4 Reference*”, Pragmatic Bookshelf, 2013
- [6] Banks, Andrew i Rahul Gupta. "MQTT Version 3.1. 1." OASIS Standard (2014).
- [7] Alexa Voice Service, Available [Online]: <https://developer.amazon.com/alexa-voice-service>
- [8] Actions On Google, Available [Online]: <https://developers.google.com/actions>
- [9] Milan Tucić, “*Protokol za razmenu poruka u sistemima pametnih kuća*”, Master rad, 2016
- [10] Gomez, Carles i Josep Paradells. "Wireless home automation networks: A survey of architectures and technologies." IEEE Communications Magazine 48.6 (2010): 92-101.
- [11] amqlib, Available [Online]: www.squaremobius.net/amqp.node/channel_api.html
- [12] AMQP, Available [Online]: https://en.wikipedia.org/wiki/Advanced_Message_Queueing_Protocol
- [13] ApacheBench tool, Available [Online]: <https://httpd.apache.org/docs/2.4/programs/ab.html>