Mobilna Android aplikacija za multimodalno upravljanje pametnom kućom za osobe s invaliditetom

Nađ, Andreja

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:923456

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij

MOBILNA APLIKACIJA ZA MULTIMODALNO UPRAVLJANJE PAMETNOM KUĆOM ZA OSOBE S INVALIDITETOM

Završni rad

Andreja Nađ

Osijek, 2021.

SADRŽAJ

1. UVOD

Internet stvari (eng. Internet of Things, IoT) predstavlja mogućnost povezivanja i komunikacije raznih uređaja putem interneta. U ovakvoj mrežnoj infrastrukturi nalaze se fizičke, ali i virtualne „stvari“ koje imaju sposobnost međusobno komunicirati. Pametna kuća je kuća kontrolirana sustavom automatizacije, a primjeri za to su automatsko centralizirano upravljanje grijanjem, klimatizacijom, osvjetljenjem i ostalim sustavima koji bi se inače ručno uključivali i isključivali. Pametna kuća daje nove opcije upravljanja vlastitim domom. Uz mogućnost namještanja uključivanja i isključivanja određenih sustava ovisno o određenom vremenu, postoji i mogućnost upravljanja svih sustava s jednog mjesta ili preko uređaja kao što su mobitel ili tablet. Danas, uz pomoć interneta stvari i pametnih kuća, osobe s invaliditetom mogu preuzeti dobar dio životnih aktivnosti pod svoj nadzor.

Cilj ovog rada je prikazati jednu jednostavniju izvedbu pametne kuće pristupačnu svima. Pristupačnost se ogleda u modularnosti sustava pomoću koje svaki korisnik može odabrati jednu od više ponuđenih postavki upravljanja pametnom kućom. U modeliranju sustava pametne kuće nastoji se da pametna kuća bude prilagođena najprije osobama s invaliditetom. Modelirani sustav inspiriran je već postojećim primjerima koje će biti navedeni i pojašnjeni.

Drugo poglavlje se bavi pojmom pametne kuće s osvrtom na potrebe osoba s invaliditetom i neka od već postojećih rješenja. Treće poglavlje opisuje zamišljeni model pametne kuće i mobilne aplikacije, dok četvrto poglavlje detaljno opisivati sklopovsko rješenje, a peto poglavlje programsko rješenje mobilne aplikacije za upravljanje pametnom kućom. Šesto poglavlje prikazuje rezultate ispitivanja i njihovu analizu za razne načine korištenja pametne kuće.

1.1. Zadatak završnog rada

U završnom radu potrebno je opisati probleme koje mogu imati osobe s određenim oblikom invalidnosti pri upravljanju pametnom kućom s naglaskom na probleme kretanja i slabovidnosti. Na temelju sposobnosti osobe, kao i sklopovskih i programskih mogućnosti upravljanja pametnom kućom, treba analizirati oblike upravljanja na više načina, definirati zahtjeve na mobilnu aplikaciju, te razraditi model i arhitekturu aplikacije. Nadalje, treba opisati potrebne programske tehnologije, jezike i razvojne okvire za razvoj mobilne aplikacije i izvršnog dijela sustava pametne kuće. U praktičnom dijelu rada, treba razviti mobilnu aplikaciju koja, ovisno o invalidnosti,

omogućuje multimodalno i prilagodljivo upravljanje pametnom kućom, pamćenje početnih postavki korisnika, te stvaranje osnovnih preporuka koristeći prikladan postupak obradbe prirodnog jezika. Mobilnu aplikaciju potrebno je ispitati i analizirati za odgovarajuće oblike invalidnosti, okolinu pametne kuće i potrebe korisnika.

2. UPRAVLJANJE PAMETNOM KUĆOM PRILAGOĐENOM OSOBAMA S INVALIDITETOM I PREGLED STANJA U PODRUČJU

U ovom poglavlju opisuju se i analiziraju problemi osoba s invaliditetom i potreba za upravljanje pametnom kućom prilagođeno osobama s invaliditetom. Također, dan je prikaz stanja u području i opisana slična rješenja.

2.1. Problemi osoba s invaliditetom

Prema Hrvatskoj enciklopediji [1], invalidnost je „stanje organizma pri kojem je trajno, potpuno ili djelomice smanjena sposobnost za rad i samostalan život, kao posljedica bolesti, ozljede ili prirođenog nedostatka, a to se stanje nikakvim postupcima liječenja ni rehabilitacije više ne može poboljšati.“ Invalidnost se dijeli na tjelesnu, senzornu, mentalnu i kombiniranu. Svaka od ovih podjela utječe na osobu tako da ju ograničava u izvedbi jedne ili više životnih aktivnosti.

Prema izvješću o osobama s invaliditetom u Republici Hrvatskoj koje je objavio Hrvatski zavod za javno zdravstvo u svibnju 2019. godine, u Republici Hrvatskoj prevalencija invaliditeta je 12.4%, od kojih je 30,4% u dobnoj skupini iznad 65 godina [2]. Pet glavnih vrsta oštećenja kod osoba s invaliditetom su: oštećenje lokomotornog sustava, oštećenje drugih organa, mentalna oštećenja, oštećenje središnjeg živčanog sustava te oštećenje glasovno govorne komunikacije, od kojih oštećenje lokomotornog sustava čini 28.8% od ukupnog broja osoba s invaliditetom, dok oštećenje glasovno govorne komunikacije čini 6% [2]. Invaliditet u većini slučajeva je ozbiljan problem koji utječe na svakodnevan život osobe. Osobe s invaliditetom se prilagođavaju svojim novim okolnostima kako bi nastaviti živjeti normalno, no, u nekim slučajevima prilagodba je vrlo teška. U slučaju teškog oštećenja lokomotornog sustava gdje se osoba ne može kretati, osoba je prinuđena imati pomoćnika za svoje osnovne potrebe.

U izvedbi pametne kuće moraju se uzeti u obzir mogući problemi osoba s invaliditetom tj. problemi sa sposobnošću upravljanja kućom. U slučaju oštećenja lokomotornog sustava osoba ne može ili se može djelomično kretati. Uzimajući to u obzir, upravljanje pametnom kućom mora imati mogućnost pristupa na području cijele kuće, u bilo kojem trenutku te bez potrebe kontakta. U slučaju oštećenja vida, pametna kuća i njeno upravljanje bi se trebalo prilagoditi tako da se sve obavlja preko naredbi prirodnog govora. U slučaju oštećenja govorno-glasovne komunikacije, upravljanje pametne kuće bi trebalo imati upravljanje putem gesta ili putem pristupačne mobilne aplikacije.

2.2. Pametna kuća

Pametna kuća koncept je kuće koji svojom automatizacijom sustava omogućuje poboljšanje standarda života stanara. Pod automatizaciju sustava smatraju se svi pametni uređaji i tehnologije koje se mogu automatizirati ili kontrolirati preko jednog univerzalnog upravljača. Programeri nastoje omogućiti ljudima upravljanje sa što više elemenata svog doma pa čak i kada su izvan svoje kuće, a na što lakši način [3]. U uobičajenom domu postoje tehnologije koje su žičano spojene. Ovakve tehnologije nameću ograničenja kao što su skupa instalacija, manjak prostora za žice, potreba za nadogradnjom kako bi se instalirali novi dodaci kući i sl. Današnje pametne kuće su dizajnirane u bežičnoj senzorskoj mreži koje zahtijevaju manje troškove instalacije, podržavaju veliku svestranost sustava te omogućuju lake nadogradnje [4].

Pametna kuća je još uvijek noviji koncept u određenom dijelu svijeta iako postoji već duže vremena, pogotovo s prisutnošću digitalizacije i automatiziranih uređaja [5]. Iako ovaj koncept nije bio popularan zbog raznih razloga, jedan od njih je bio zbog tehnofobije [6].

2.2.1. Sustavi pametne kuće

Među glavnim sustavima svake kuće nalaze se: rasvjeta, klima, protuprovalna zaštita, brojilo, grijanje, kućni aparati i multimedija [7]. Pametna rasvjeta podrazumijeva vrstu rasvjete koja može ili automatizirati uključivanje i isključivanje ili može upravljati preko grafičkog sučelja uz standardnu sklopku na zidu. Cilj pametne rasvjete je lako upravljanje prema vlastitoj potrebi. Uz senzore prisutnosti koji mogu očitavati prisutnost ili manjak prisutnosti osobe u prostoriji, bolje se osigurava automatizacija utemeljena na uštedi energije. Klimatizacija i grijanje kao dio sustava pametne kuće osigurava kontrolu nad temperaturom u vlastitoj kući. Danas postoje klime koje mogu i zagrijavati i rashlađivati prostor, no većina domova svejedno ima ili radijatore ili podno grijanje. Ovakvi pojedini sustavi ako se spoje na jedno grafičko sučelje daju optimalno rješenje promjene temperature svake prostorije ovisno o njenoj namjeni ili potrebi korisnika. Uz dodatne konfiguracije može se namjestiti da u slučaju kada sustav pametne kuće očita paljenje klime, hlađenje ili grijanje, a dok su otvoreni prozori zatvori iste, ako su automatizirani, ili javi korisniku. Neki domovi već imaju sustav centralnog grijanja preko kojega se može namjestiti korištenje po vremenu. Ovakav sustav je lako nadograditi kako bi pripao dijelu pametne kuće.

Za sigurnost kuće se ugrađuju sigurnosne kamere, alarmi, pametne brave i sl [7]. Sve to putem sustava pametne kuće može biti pristupačno na mobitelu kako bi se mogla nadzirati kuća čak i kad osoba nije u njoj. Pri slučaju provale ili detekcije neke čudne aktivnosti, korisnik bi dobio dojavu

na mobitel te bi lagano mogao reagirati sukladno situaciji. Može se narediti da ukoliko korisnik napusti kuću, a prvo iduće otvaranje vrata ne pripada korisniku i sustav nije informiran o njegovu dolasku, da automatski pali alarm i kontaktira autoritativne figure. Razne pametne brave omogućuju korisniku raznolik odabir od kojeg mogu odabrati onaj koji najviše odgovara korisniku, kao što su višestruke razine sigurnosti i pametne funkcije. Osim kamera, alarma i brava, neke od važnih komponenata potrebnih za sigurnost svakog doma su senzori za požar, dim, razne plinove [8], čak i senzori curenja ili poplave [7] ako se kuća nalazi u području gdje je takav događaj učestal ili se može postaviti senzor poplave u prostorijama gdje se nalaze većina cijevi kroz koje protječe voda. Senzori za dim nisu neuobičajeni, no senzori za plinove, tj. specifično detektor ugljičnog monoksida, baš i nije. Ugljični monoksid ili CO je tihi ubojica te simptomi trovanja mogu biti suptilni. Ukoliko se dogodi trovanje tijekom sna, velike su šanse da se osoba neće ni probuditi dok ne bude prekasno. CO je bezbojan plin bez mirisa i okusa te nastaje kao produkt izgaranja tvari koje sadrže ugljik bez dovoljne količine kisika u zraku. Tipično CO može nastati u kaminima pri gorenju drva ili kod plinskih pećnica kao i kod plinskih bojlera. Za osiguravanje sigurnosti, moguće je instalirati detektore plina CO-a koji bi odmah javili njegovu prisutnost i u kojoj količini se nalazi unutar koje prostorije. Uz dodatno programiranje detektor može očitavati do određenog praga nakon kojeg je preopasno te se automatski kontaktiraju autoritativne figure.

Prema Parku i suradnicima [9], osim navedenih stavki postoje i sljedeći sustavi: pametne memorije koje pamte korisnikove sklonosti rasvjete, zvukova, mirisa i sl.; pametna garderoba koja unaprijed provjerava stanje vremena i klime kako bi obavijestila korisnika o današnjoj garderobi, a ima i mogućnost pranja odjeće; pametni krevet koji lagano budi korisnika kako bi omogućio dobar početak dana; pametni frižider koji omogućuje korisniku preglednost kupljenih proizvoda i proizvoda koje treba kupiti, itd. Pametnih sustava ima puno više nego nabrojanih te svaka stvar koja se koristi se može napraviti u pametnoj verziji, kao npr. pametno ogledalo. Obično ogledalo naravno ima samo jednu funkciju a to je prikazivati odraz, no pametno ogledalo ima mogućnost prikazivanja datuma, vremena, klime, vijesti i drugih stvari [10].

2.2.2. Prednosti i nedostaci pametne kuće

Prednosti pametne kuće su: ušteda energije, pogodnost i upravljivost, zdravlje i podrška, udobnost, nadzor [11]. Naravno, glavne prednosti pametne kuće ovise o tome kako i za što je dizajnirana. Najistaknutija spomenuta prednost je sposobnost upravljanja energetskim uslugama pametne kućne tehnologije ili smanjenje potrošnje energije. Primjerice, automatizirano centralno grijanje ima postavke kada da se uključuje i isključuje, prema tome, moguće je namjestiti da se kuća ne

grije dok nema osoba u kućanstvu ili dok svi spavaju, za razliku da se ostavi grijanje uključeno cijeli dan. Uz odabir pravilnog gumba na aplikaciji korisnik može uključiti grijanje kada god poželi te tako smanjiti financijske troškove. Još jedna prednost koja se može istaknuti je praktičnost i kontrola nad kućanstvom. Kod mnogih pametnih kuća svrha nije ušteda energije, već olakšanje života te omogućavanje zabavnijeg i zanimljivijeg načina upravljanja. Najveća prednost za većinu ljudi je udobnost i laka kontrola koju nudi ovakav sustav [11].

Prema ispitivanju 31-og sudionika [11] o rizicima i barijerama u tehnologiji pametnih kuća, najveće rizike predstavljaju:

- privatnost, sigurnost, hakiranje
- tehnička pouzdanost, jamstva i zastarjelost
- korisnost, prihvaćanje i učenje korisnika.

Privatnost i sigurnost nije samo problem hakera već samog sustava pametne kuće. Neke pametne kuće dizajnirane su da znaju gdje je koji sustav u kući te da znaju gdje je koja osoba u kući. Bilo koja zlonamjerna osoba s dovoljno vještine i vremena može probiti svaku sigurnosnu prepreku i saznati sve informacije o tuđem domu. No, osim toga, neki sustavi koji rade na principu treće strane, ili oni koji koriste Google Home, Alexu i sl., omogućuju kompanijama koje posjeduju ove tehnologije pristup svim podacima pametne kuće. Pitanje je može li se vjerovati tvrtkama koje prikupljaju sve te podatke. Znači, glavni problem pametne kuće su zapravo sigurnost njenih podataka jer nitko ne želi da neka nepoznata osoba prikupi podatke o cijelom izgledu kuće, vremenu odlaska na počinak ili kada vlasnici nisu doma. Osim sigurnosti podataka, postoje problemi same tehničke izvedbe. Problem je, naime, oko napajanja, jer pametna kuća radi služeći se električnom strujom, što znači kada nestane struje, kuća će biti van upotrebe. Zato je važno pri dizajniranju i programiranju osposobiti kuću za sve mogućnosti. U slučaju da nestane struje, a vrata se otvaraju samo preko pametne kuće, npr. preko gesture, naredbe ili senzora, bilo bi pametno u tom slučaju ili otvoriti vrata ili pri samom pravljenju vrata napraviti kvaku ili mehanički gumb koji se mogu koristiti samo u ovakvim trenucima.

Iako manji nedostatak nego ostali, važan je za naglasiti, a to je korisnost. Pametne kuće nisu za svakoga te se ni ne treba svakoga prisiljavati na njih, no ako se netko planira odlučiti na takvu kuću, moraju biti svjesni svih mogućih nedostataka i vidjeti je li korisnost te kuće veća nego njeni mogući nedostaci. U velikom slučaju osoba s invaliditetom korisnost će prevladati.

Prema GfK [12], osobe koje više koriste tehnologiju, tj. mlađe generacije, su više privučene idealima koje donose pametne kuće. Prema Wilsonu i suradnicima [13], istraživanje tržišta otkriva da je najznačajnija prepreka za usvajanje trošak koji se mora platiti unaprijed te nedostatak informacija i zabrinutost za privatnost. Iako su studije od strane Balta-Ozkana i suradnika [14] i Paetza i suradnika [15] potvrdile interes za potencijalnim upravljanjem vlastite kuće i smanjenje troškova energije, također su identificirale već prije spomenute probleme: cijenu, privatnost, sigurnost, pouzdanost i interoperabilnost različitih tehnologija.

2.3. Izazovi upravljanja pametnom kućom za osobe s invaliditetom

Općenito upravljanje vlastitom kućom koja nije automatizirana predstavlja problem osobama s invaliditetom, a kako bi im uveli neovisnost i slobodu življenja, dizajn pametne kuće je iznimno važan za osobu s invaliditetom kako bi osigurali što pristupačniji dom. Pod time se podrazumijeva logičan, i lagan za upamtiti raspored sustava kojima se upravlja unutar određene prostorije. Kod pametnih kuća upravljane na glasovne naredbe, potrebno je paziti na jezik osobe i njihove mogućnosti izgovaranja određenih riječi. Stoga, naredbe bi morale biti lagane i kratke kako bi korisnik mogao naučiti u kratkom periodu sve naredbe. Treba izbjegavati kompleksne naredbe za sve sustave korištene unutar kuće. Pri prvoj uporabi pametne kuće, korisnik bi morao naučiti puno o načinu upravljanja, od naredbi za paljenje, gašenje i sl., do samih položaja sustava u kući.

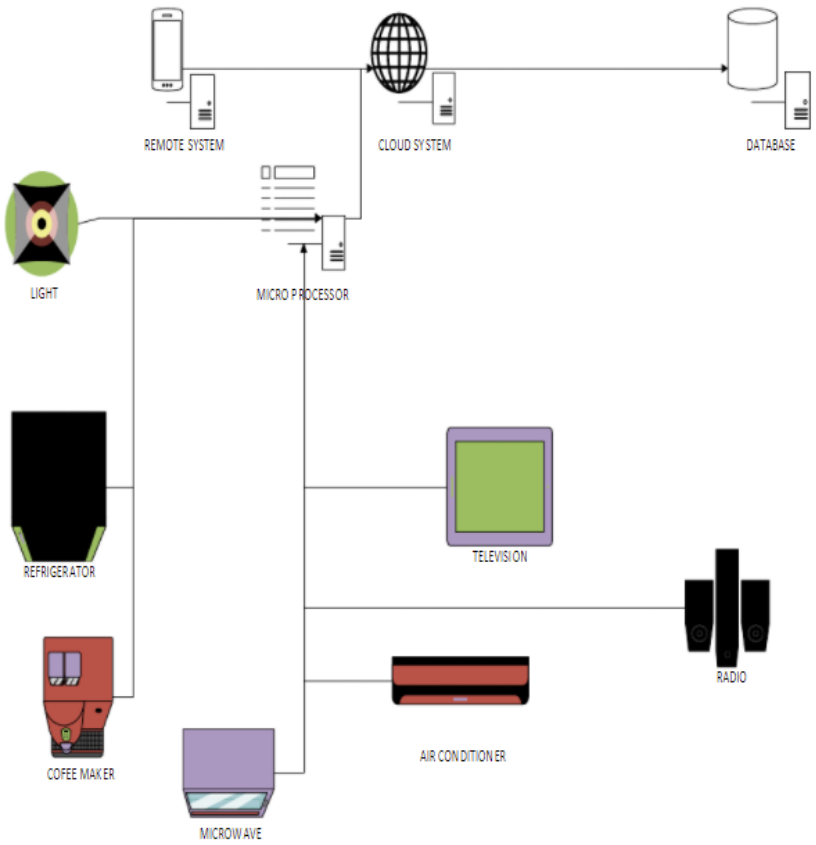
U slučaju sigurnosnih kamera ili senzora na pokret, oboje mogu registrirati pokret neke životinje kao nepoznatog uljeza. Osobe s teškom pokretljivosti ili one s teškim ozljedama vida neće moći provjeriti stanje te će takav događaj izazvati paniku. Nažalost, jedini način da se ovo izbjegne je ili pronaći dovoljno dobar sustav, koji će vjerojatno biti skup, ili kroz metodu pokušaja i pogrešaka naučiti svoju pametnu kuću ako ima tu mogućnost.

2.4. Trenutno stanje u području upravljanja pametnom kućom za osobe s invaliditetom

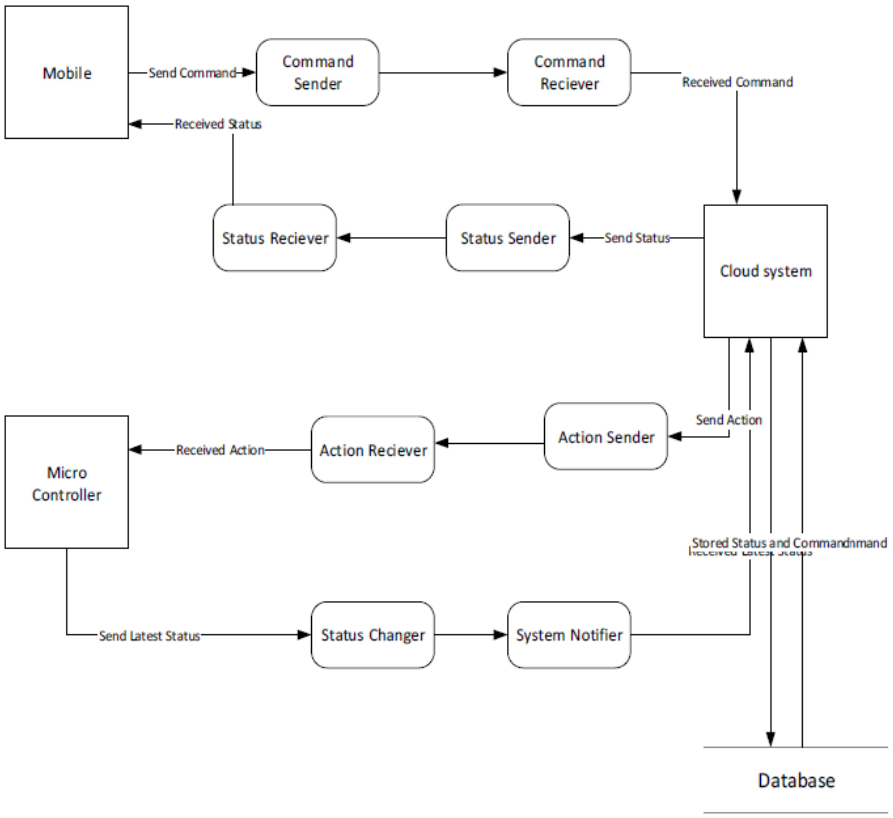
U sljedećim poglavljima navedena su neka od mogućih rješenja automatiziranja kuća. Svaka pametna kuća radi na drugačiji način ili nema sve iste sustave. Uzimajući to u obzir, mora se vidjeti što odgovara kojoj osobi te tako dalje modelirati pametnu kuću i njen sklopovski dio. Najvažnije za odabrati prije početka izrade pametne kuće je sklopovlje koje će se koristiti. Postoje razne inačice pametnih kuća koje su dizajnirane na drugačije načine kako bi odgovorili na određene zahtjeve.

2.4.1. Izvedba pametne kuće pomoću internet stvari

Slika 2.1 prikazuje sklopovsku arhitekturu sustava po uzoru na shemu Daviesa i suradnika [16]. Ovaj sustav sastoji se od mobitela putem kojeg se šalju naredbe. Naredbe dohvaća sustav oblaka računala te ih šalje na mikroupravljač koji, ako može, obavi radnju te vraća u oblak računala izvedenu radnju kao obavijest. Oblak računala tu obavijest potom vraća nazad na aplikaciju na mobitelu te tako obavješćuje korisnika o uspješnosti izvedene naredbe. Slika 2.2 prikazuje objašnjeni protok podataka u opisanom sustavu [16].



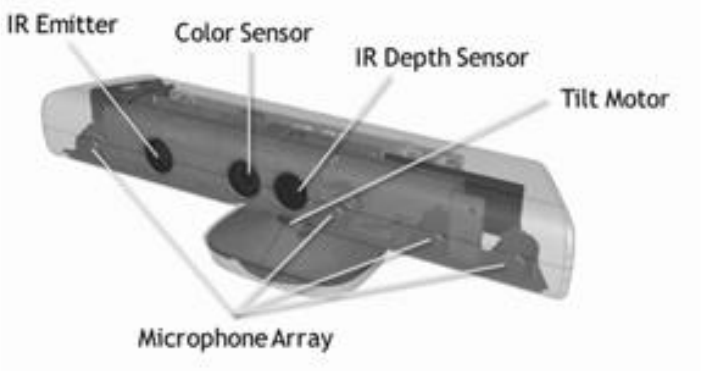
Slika 2.1. Shema sklopovske arhitekture sustava pametne kuće [16]



Slika 2.2. Protok podataka sustava [16]

2.4.2. Izvedba pametne kuće Kinect senzorom

Jedna od zanimljivijih primjena elemenata koji se inače ne koriste u kućanstvu već u svijetu videoigara je primjena uređaja zvanog Xbox Kinect senzor. Ovaj senzor ima mogućnost snimanja kretanja stvarnih pokretnih stvari u stvarnom životu. Mogućnost ovog senzora da snima kretanje stvarnih pokreta omogućuje upravljanje raznim uređajima tijelom, odnosno pomoću ruku ili nogu [17]. Prvo, Kinect emitira infracrveno svjetlo preko IR odašiljača prikazanog na slici 2.3, kao i ostale dijelove senzora. Refleksije laserskog snopa stvaraju se u polju dubine što predstavljaju piksele koje pokriva Kinect. Ova tehnika se naziva „light coding“ ili svjetlosno kodiranje. Senzor prima kao ulaz kodirano infracrveno svjetlo i proizvodi obojenu sliku, audio tok i dubinsku sliku.



Slika 2.3. Prikaz Kinect senzora sa označenim dijelovima

Sustav je programiran da prepoznaje pokrete pomoću položaja ruku i nogu [17]. U ljudskom tijelu 20 zglobova se prepoznaje pomoću Kinect senzora. Računanjem 3D koordinata svakog zgloba u prostoru u odnosu na glavu nastaje predodžba položaja svakog dijela tijela. Slika 2.4 prikazuje kako Kinect senzor prepoznaje geste ruke iznad glave, dok slika 2.5 prikazuje geste guranja.

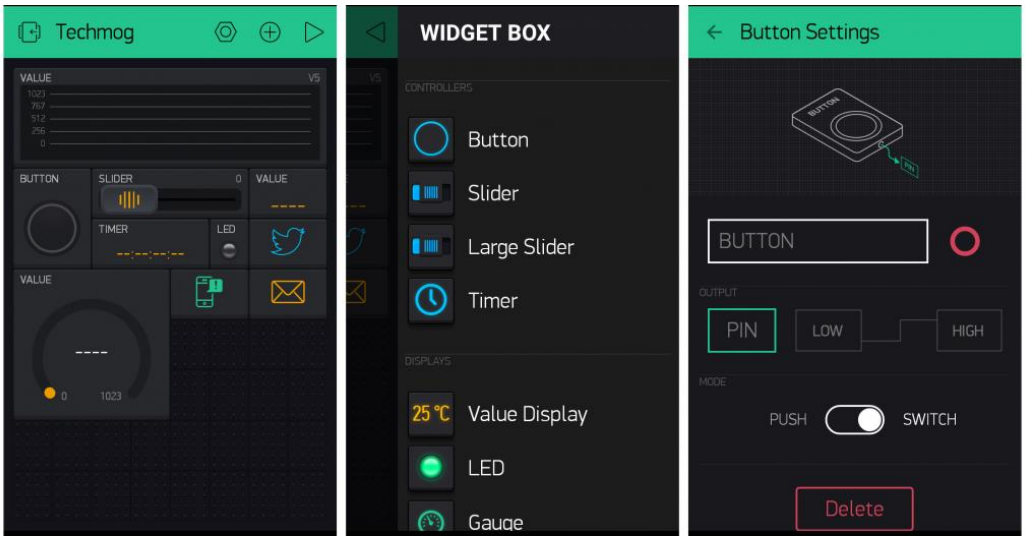
Slika 2.4. Prikaz prepoznavanja geste ruke iznad glave putem Kinect senzora

Slika 2.5. Prikaz prepoznavanja geste guranja putem Kinect senzora

Glavna uporaba ovakvog sustava bi bila za osobe s invalidnošću zbog upravljanja jednostavnim pokretima ruku ili nogu kojima kreiraju određene geste putem kojih mogu upravljati svojom pametnom kućom. Primjerice, u slučaju pada ili u slučaju hitnog slučaja, nakon što Kinect registrira gestu šalje upit korisniku je li sve u redu te ako korisnik ne reagira, sustav zove pomoć korisniku.

2.4.3. Aplikacije i digitalni asistenti

Aplikacije i programi koji pružaju samostalno i jeftino dizajniranje vlastite pametne kuće su Blynk, Particle, AWS IoT i sl. Među istaknutijima se nalazi Blynk. Blynk je aplikacija koja je dizajnirana za internet stvari. Može upravljati sklopovljem na daljinu te prikazivati, pohranjivati i vizualizirati razne podatke. Na slici 2.6 se nalazi prikaz ugrađenih programčića (eng. widget) te njihovih postavki unutar aplikacije Blynk.



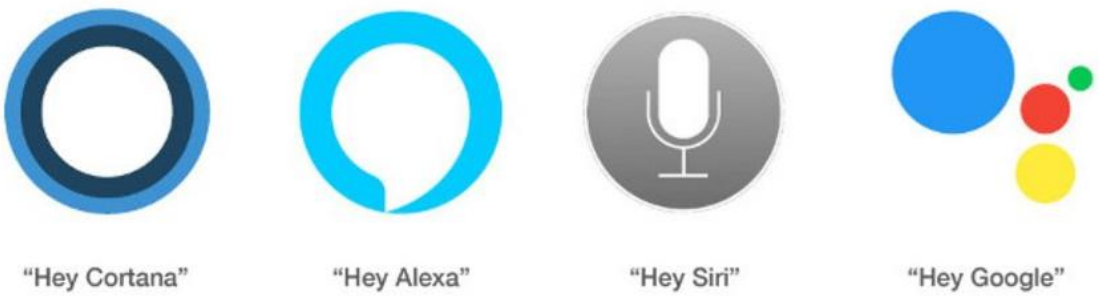
Slika 2.6. Blynk aplikacija - radna površine (lijevo), lista widgeta (sredina), postavke widgeta (desno)

Aplikacija Blynk omogućava stvaranje sučelja za programe koristeći ugrađene programčiće. Poslužitelj Blynk odgovoran je za svu komunikaciju između spojenog sklopovlja i pripadajućeg pametnog telefona. Biblioteka Blynk omogućava komunikaciju s poslužiteljem i obrađuje sve dolazne i odlazne naredbe [18].

Blynk je izuzetno jednostavan za korištenje te je kompatibilan sa modulima kao što su Arduino, Raspberry Pi i raznim drugim jeftinijim mikroupravljačima. Programiranje je svedeno na minimum što znači da i početnici mogu lagano naučiti koristiti ovaj program. Sve što osoba treba napraviti je spojiti željeni uređaj na mikroupravljač koji može komunicirati s Blynk aplikacijom te odabrati jedan od ugrađenih programčića koji je prikladan. Primjerice, za instalaciju pametne

rasvjete, svijetlo se spoji na nožicu mikroupravljača te se na Blynk aplikaciji odabere klizač putem kojega se određuje jakost osvjetljenja od 0 do 100% [19]. Na mikroupravljaču se treba programirati primanje vrijednosti s aplikacije Blynk i postavljanje iste na nožicu. Ovo je vrlo lako, jer Blynk ima već gotove i besplatne primjere programa.

Ako prema korisnikovim potrebama treba sustav s više upravljanja, jedna od opcija je koristiti se digitalnim asistentima. Pod digitalnim asistentima smatraju se pametni uređaji s umjetnom inteligencijom koje putem glasovnog upravljanja omogućuju obavljanje raznih stvari. Popularni digitalni asistenti su: Alexa, Siri i Google Assistant. Osim cijene izvornog uređaja, ovakav način rada pametne kuće je besplatan. No, kako se kaže, ako ne platite proizvod, proizvod ste [20]. Razlog zašto su ovi uređaji dobri je to što prikupljaju podatke kako bi dalje mogli obučiti svoje glasovne pomoćnike. Iako je privatnost kod ovakvih uređaja napadnuta, njihova kvaliteta nije upitna. Slika 2.7 prikazuje popularne digitalne asistente i njihove aktivacijske riječi putem kojih se dalje komunicira sa sustavom.



Slika 2.7. Digitalni asistenti i njihove aktivacijske riječi

3. PRIJEDLOG MODELA SUSTAVA PAMETNE KUĆE PRILAGOĐENE OSOBAMA S INVALIDITETOM

Ovo poglavlje daje opis zahtjeva na upravljački i korisnički dio sustava pametne kuće za osobe s invaliditetom, te prikazuje sklopovsku građu i programski model rješenja sustava. Također, opisan je multimodalni način rada sustava i postupak obrade prirodnog jezika.

3.1. Zahtjevi na sustav

3.1.1. Zahtjevi na upravljački dio sustava

Zahtjevi na upravljački dio sustava podrazumijevaju sljedeće zahtjeve [21]:

- tehničke
- vezane za sigurnost i privatnost
- jednostavnost
- ostalo

Glavni tehnički zahtjev je proširivost. Kako bi sustav radio kako treba i u budućnosti ili uz male izmjene, mora se osigurati da sustav može prihvatiti dodatna proširenja i nadogradnje. Proširivost podrazumijeva korištenje novih značajki koje nisu bile predviđene u vrijeme projektiranja. Za tu svrhu zahtijevaju proširive alate i metode za razvoj. Sigurnost procesa, kao i autentifikacija i autorizacija mora se valjano odraditi kako bi samo određena aplikacija mogla upravljati sustavom te samo određena osoba može upravljati tom aplikacijom. U slučaju da to nije osigurano, postoji mogućnost da osoba koja ima istu aplikaciju ili slično sklopovlje, može upravljati pametnom kućom drugog korisnika, a to se, naravno, ne želi. Jednostavnost opisuje složenost razvoja sustava te uključuje interakciju između cijelog sustava i programera. Sustav mora biti jednostavan za naučiti koristiti za bilo kojeg korisnika. Internetski prijenos videa, zvuka i podataka potreban je u više domena kao što je sigurnost (npr. prijenos videotoka nadzorne kamere) i zdravstvena zaštita (npr. daljinsko praćenje pacijenata). U ovom radu se koristi što jednostavnije sklopovlje na koje se uvijek mogu dodavati novi elementi. Glavni cilj je jednostavno i modularno korištenje.

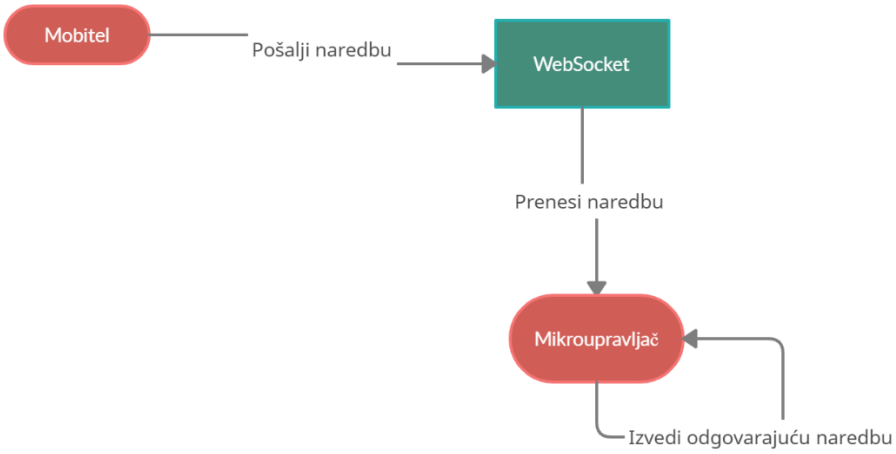
3.1.2. Zahtjevi na mobilnu aplikaciju

Zahtjevi na mobilnu aplikaciju podrazumijevaju zahtjeve kao i na upravljačkom dijelu, ali s dodatnim podskupinama specifikacija zahtjeva. Proširivost nije primjenjiva samo za upravljački dio, već i za razvoj aplikacije gdje se isto tako mora omogućiti dodavanje mogućih nadogradnji.

Manipulacija podataka je potrebna kako bi omogućila interakciju s uređajima i uslugama različitih dobavljača. U ovakvim slučajevima potrebno je mapiranje prikaza podataka i formata podataka te bavljenje konceptima koji su modelirani na različite načine u različitim područjima. To zahtijeva pretvorbu instance modela u drugi model ili pretvaranje primjeraka dva različita modela u model više razine. Sljedivost podrazumijeva praćenje određenih radnji kao što su praćenje procesa ili dozivanja nekih događaja. Sljedivost je obično potrebna za statistiku ili za pitanja odgovornosti. Nesigurne aplikacije negativno utječu na uređaje ili na okoliš na način koji programer ne predviđa te se mora predvidjeti kako bi se osigurala sigurnost procesa. Povjerljivost zahtjeva da podaci u sustavu nisu vidljivi nikome osim definiranoj skupini ljudi kojoj su te informacije nužne. Autentifikacija i autorizacija se podrazumijeva kao sljedeća stavka nakon integracije povjerljivosti jer samo ovim putem se može ograničiti ulaz neautoriziranim osobama u sustav. U ovom radu se želi projektirati što jednostavnija modularna aplikacija sa mogućnostima dodatnih proširivanja. Sama aplikacija bi morala biti jednostavna za naučiti i lagana za koristiti te bi se sve složenije funkcije trebale sakriti od korisnika. Razine apstrakcije su važne kako bi krajnjim korisnicima sakrili pojedinosti implementacije. Integritet procesa nam govori da istodobne radnje pametnih uređaja unutar pametne kuće ne bi trebale biti međusobno kontradiktorne (npr. otvaranje i zatvaranja vrata). Aplikacija mora pravilno upravljati sa svojim resursima. Uređaji s ograničenim resursima se oslanjaju na kompromis između funkcionalnosti i ograničenih resursa.

3.2. Sklopovska arhitektura sustava pametne kuće

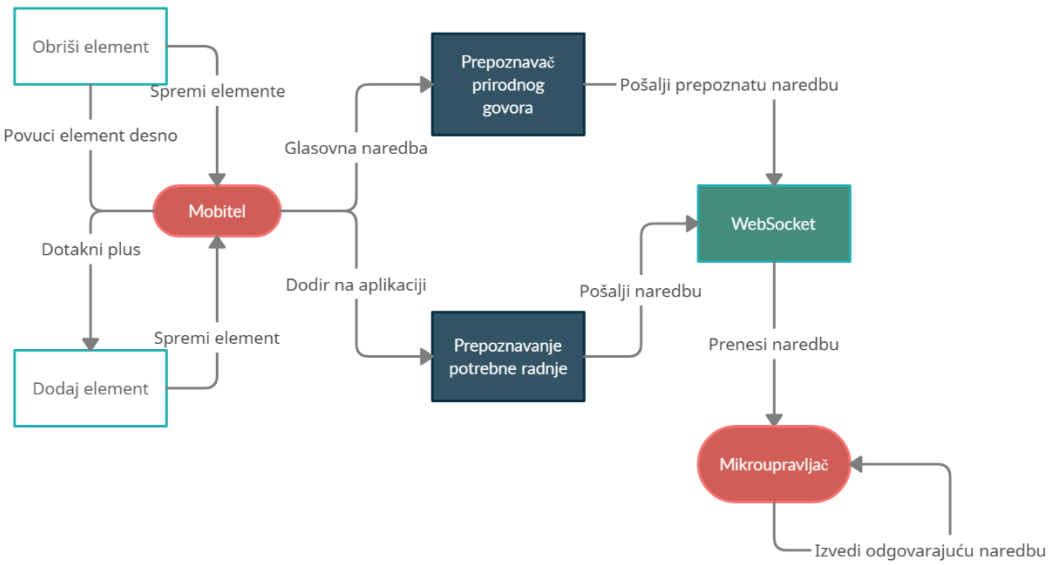
U ovom radu sklopovska arhitektura zasnovana je na ideji upravljanja putem jednog uređaja. Taj uređaj je mobilni telefon koji putem mobilne aplikacije šalje mikroupravljaču daljnje upute kako i što treba nešto učiniti. To znači da svi pametni uređaji moraju biti spojeni na sklop te da sklop i aplikacija moraju ostvariti povezanost preko nekog sustava. Taj vanjski sustav je stranica koju svaki sklop ima putem svoje pristupne točke. Putem stranice na kojoj se nalazi WebSocket, sklop prima naredbe aplikacije. Slika 3.1 prikazuje pojednostavljenu sklopovsku arhitekturu projektiranog sustava.



Slika 3.1. Projektirana sklopovska arhitektura sustava pametne kuće

3.3. Programska arhitektura sustava pametne kuće

Ideja arhitekture aplikacije je da korisnik može na jednostavan način odabrati neku od mogućih naredbi te se taj odabir šalje putem aplikaciju na vanjski sustav WebSocket, koji obrađuje taj zahtjev i šalje ga mikroupravljaču. Mikroupravljač iščitava naredbu i uspoređuje ju sa svojim predprogramiranim naredbama. Ako naredba odgovara nekoj od postojećih, izvodi naredbu. Slika 3.2 prikazuje programsku arhitekturu projektiranog sustava pametne kuće gdje se multimodalnost prikazuje putem dodira ili glasovne naredbe, koje šalju naredbu na WebSocket, kako bi ju primio i izveo mikroupravljač. Dodavanje novih ili brisanje elemenata je osnovno kako bi aplikacija omogućila korisniku lagano dodavanje novih uređaja u svoju pametnu kuću.



Slika 3.2. Projektirana programska arhitektura sustava pametne kuće

3.4. Načini multimodalnog upravljanja pametnom kućom

Multimodalno upravljanje pametnom kućom predstavlja ideju da se sa različitim radnjama dobije isti rezultat. To znači da se sa različitim radnjama izvodi ista radnja, kao paljenje svjetala putem pljeskanja ruku, glasovnom naredbom, jednoručnom gestom ili čak pametnim upravljanjem putem namještenog vremena.

Načini multimodalnog upravljanja:

- fizička naredba
- glasovna naredba
- geste rukama ili licem
- audio i video analize.

Pametna kuća dizajnirana specifično za ljude sa invaliditetom mora omogućiti sadržati multimodalnost, ali i prilagodljivu multimodalnost. Prilagodljivost multimodalnosti omogućuje da svaka osoba sa invaliditetom može postaviti određenu radnju za aktivaciju ovisno o njihovim sposobnostima. Osobe bez mogućnosti govora nemaju potreba za glasovnim upravljanjem, no geste im mogu pomoći, dok bi za osobe koje su izgubile dijelove tijela ili imaju ozljede od kojih ne mogu slobodno micati svoje tijelo, bilo od veće pomoći glasovno upravljanje. S gledišta audio i video analize smatra se da postoji dovoljno pametna aplikacija koja može prepoznati zvukove kao što su razbijanje vrata ili lomljenje prozora što su indikacije moguće provale. Osim zvuka, neke od pametnijih aplikacija imaju programirano svojstvo prepoznavanja ljudi te se tako može osigurati da i video tehnologijom aplikacija očita mogućeg provalnika. U radu, kao što je prikazano na slici 3.2 izradit će se sustav koji podržava fizičke i govorne naredbe.

3.5. Postupci obrade prirodnog jezika

Obrada prirodnog jezika kombinira umjetnu inteligenciju (eng. artificial intelligence, AI) i računalnu lingvistiku tako da računala i ljudi mogu neometano razgovarati. Obrada prirodnog jezika pokušava premostiti jaz između strojeva i ljudi dopuštajući računalima da analiziraju ono što je korisnik rekao (prepoznajući ulazni govor). Cilj obrade prirodnog jezika je „postići jezičnu obradu sličnu čovjeku“. Odabir riječi „obrada“ je namjeran i ne bi se trebao zamijeniti s „razumijevanje“ jer iako je cilj ostvariti razumijevanje, taj cilj još uvijek nije ostvaren. Da bi program razumio morao bi u potpunosti moći: parafrazirati, prevesti, odgovoriti na pitanja i izvesti zaključke iz konteksta [22]. Da bi razgovarao s ljudima, program mora razumjeti sintaksu

(gramatiku), semantiku (značenje riječi), morfologiju (vrijeme) i pragmatiku (dijalog) [23]. Broj pravila praćenja može se činiti suvislim i objašnjava zašto su rani pokušaju obrade prirodnog jezika u početku doveli do razočaravajućih rezultata. No, koristeći različite sustave, obrada prirodnog jezika se polako poboljšavala prelaskom s glomaznih pravila na metode računalnog programiranja temeljene na uzorcima. Obrada prirodnog jezika omogućuje računalnim programima korištenje umjetne inteligencije i strojnog učenja za razumijevanje nestrukturiranog sadržaja za izvršavanje jezika i pružanje konteksta za jezik, baš kao i ljudski mozak.

Osnovni zadaci predobrade prirodnog jezika koje znanstvenici trebaju obaviti kako bi alati obrade prirodnog jezika dobli smisao ljudskog jezika su: tokenizacija, označavanje dijelom govora, određivanje i lematizacija, zaustavljanje uklanjanja riječi [24]. Tokenizacija je zadatak koji se koristi za razbijanje niza riječi u semantičke korisne jedinice koje se nazivaju *tokeni*. Označavanje dijelom govora predstavlja zadatak u kojemu se svakom tokenu dodjeljuje kategorija govora. Neke od kategorija su: glagol, imenica, pridjev, zamjenica i sl. Ovaj zadatak je iznimno koristan za određivanje odnosa između riječi i stoga razumijevanje značenja rečenica. Određivanje i lematizacija predstavlja zadatak u kojem se svaka riječ svodi na osnovni, korijenski oblik. Zaustavljanje uklanjanja riječi predstavlja filtriranje potrebnih i nepotrebnih riječi kao što su prijedlozi i članci (na, do i sl.). Tek kada su svi ovi zadaci ispunjeni, alati obrade prirodnog jezika mogu pretvoriti tekst u nešto što stroj može razumjeti i dalje obraditi. Sljedeći postupak je izgradnja algoritma koji će izvoditi obradu prirodnog jezika i obavljati određene zadatke. Postoje dva tipa algoritama koji se koriste za rješavanje problema obrade prirodnog jezika, a to su: pristup temeljen na pravilima i algoritmi strojnog učenja. Sustavi temeljeni na pravilima se koriste ručno izrađenim gramatičkim pravilima stvorenih od strane stručnjaka za lingvistiku. Ovo je jedan od prvih pristupa te se koristi i danas. Problem je što se ovaj pristup temelji samo na poznatim gramatičkim pravilima koje je potrebno definirati samostalno. Kako bi zaobišli ovu prepreku, nastao je algoritam strojnog učenja čija najveća prednost je njegova sposobnost samostalnog učenja. Strojevi samostalno uče iz prethodnih podataka kako bi sami predviđali odgovor. Modeli strojnog učenja temelje se na statističkim metodama i uče izvršavati zadatke nakon unosa primjera koji zapravo predstavljaju podatke s kojima se obučavaju strojni algoritmi.

U svrhe obrade prirodnog jezika u radu će se koristiti sučelje SpeechRecognizer koje je besplatno uz Android Studio. SpeechRecognizer je sučelje za programiranje aplikacija (eng. Application Programming Interface, API) putem kojeg se primljeni zvuk šalje na udaljene poslužitelje radi prepoznavanja govora.

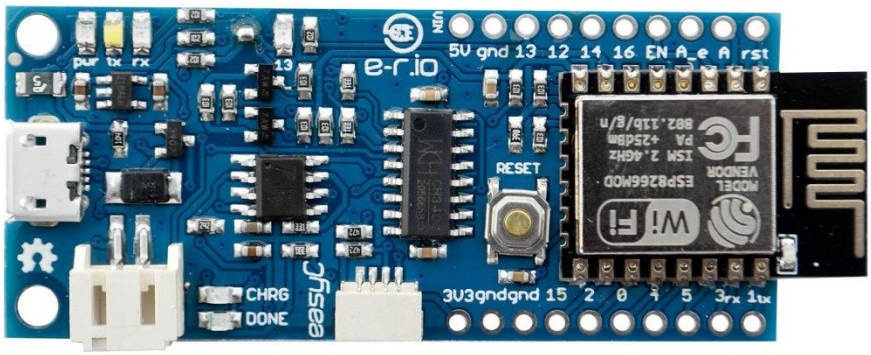
4. SKLOPOVSKO RJEŠENJE ZA UPRAVLJANJE PAMETNOM KUĆOM

Ovo poglavlje daje prikaz korištenih programskih tehnologija i okoline potrebnih za razvoj sustava pametne kuće, kao i sklopovski dio, način rada i programsko rješenje upravljačkog dijela sustava pametne kuće.

4.1. Korištene tehnologije, programski jezici i razvojna okolina za ostvarenje upravljačkog dijela sustava

4.1.1. Sklop Croduino NOVA2

Croduino Nova2 je mikroupravljačka pločica sa sposobnošću spajanja na Wi-Fi mrežu. Bazirana je na sklopu ESP8226 koji je jeftini Wi-Fi mikročip sa sposobnošću mikroupravljača i TCP/IP stoga, no vrlo je komplicirano koristiti ga samostalnog. Nova2 je izvedba ovog mikročipa sa svim potrebnim dijelovima za lagano reprogramiranje. Ima devet programibilnih nožica koje na slici 4.1 se može vidjeti da su: 0, 2, 4, 5, 12, 13, 14, 15 i 16. Nožice 0 i 2 nisu najsigurnije za uporabu zbog unutarnjeg spoja koji ponekad može učiniti ponovno postavljanje sklopa. Najvažnije, programibilna je iz Arduina bez potrebe za ručnim postavljanjem pločice.



Slika 4.1. Prikaz Croduino NOVA2 ESP8266 sklopa

4.1.1. Računalna platforma Arduino

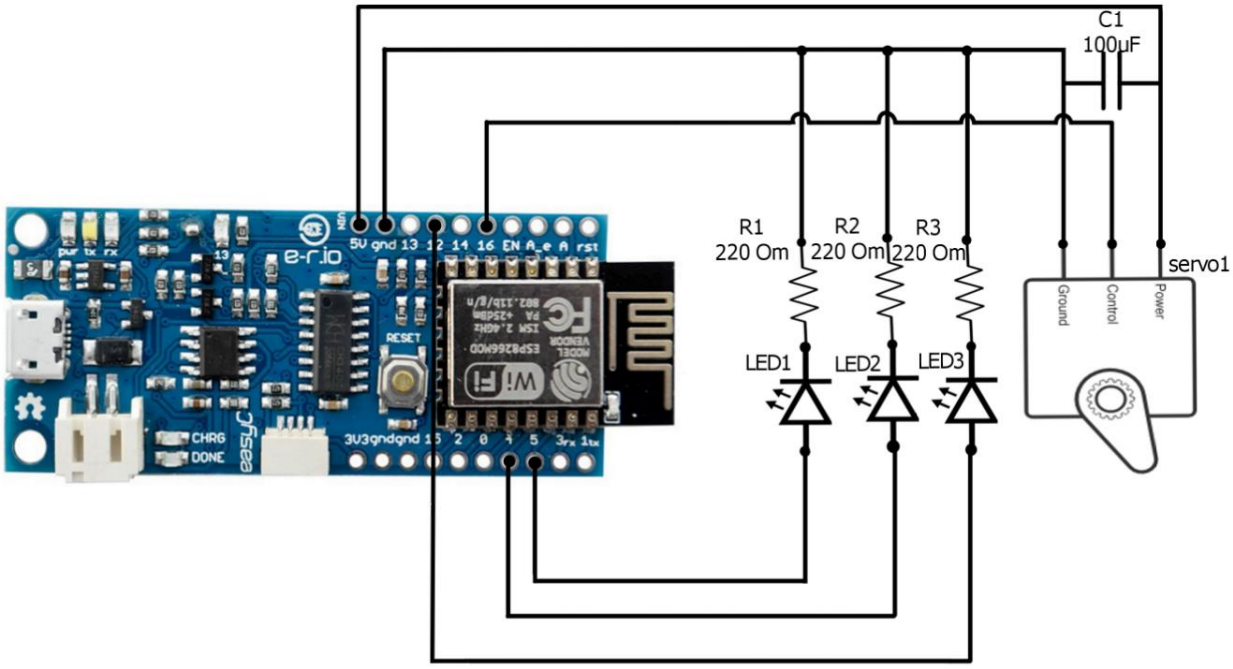
Računalna i programska platforma Arduino dizajnerima i graditeljima omogućuje stvaranje uređaja i naprava koji računalima omogućuju povezivanje s fizičkim svijetom, odnosno stvaranje internet stvari. Arduino je platforma za izradu elektroničkih prototipa otvorenog koda koja korisnicima omogućuje stvaranje interaktivnih elektroničkih objekata. Arduino ima puno besplatnih kodova, jer je glavni slogan ove platforme otvoreni kod, što znači da se na internetu mogu pronaći razni kodovi ili isječci koji mogu pomoći u izradi raznih projekata.

4.1.2. Biblioteke ESPAsyncWebServer i ESPAsyncTCP

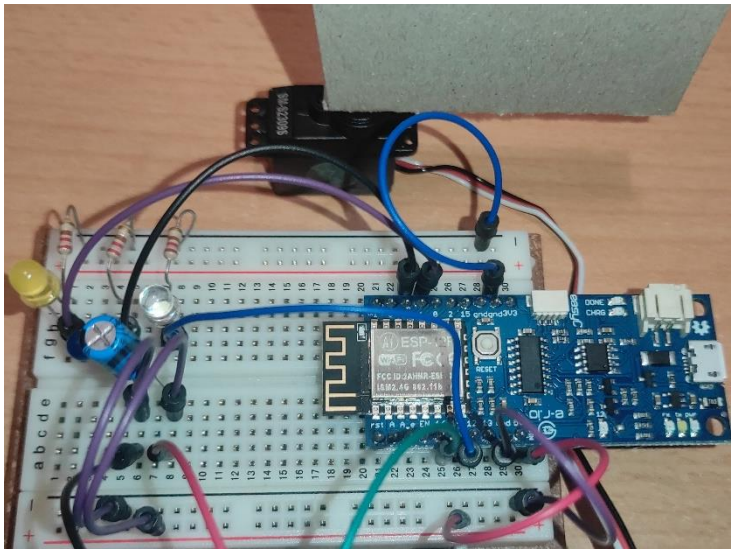
Biblioteka ESPAsyncWebServer koristi biblioteku ESPASyncTCP i predstavlja poslužitelj za WebSocket koji je potreban za izradu ovog projekta. ESPAsyncWebServer je zadužen za slušanje veze te upravljanjem rukovateljima (eng. handlers) i pridruživanjem rukovatelja odgovarajućim zahtjevima (eng. requests).

4.2. Opis korištenog sklopovlja

Za izradu ovog projekta korištene su tri svjetleće diode, tri otpornika od 220 Oma s 5% tolerancije i sklop ESP8266 Nova2. Svjetleće diode su spojene na nožice 4, 5 i 12. Za prikaz upravljanja vrata koristi se servo motor čiji je signal spojen na nožicu 16. Bez kondenzatora servo motor se može pomicati bez naredbe za to. Kondenzator od 100 µF se spaja paralelno na servo motor kako bi se filtrirala buka koja uzrokuje trzaje. Sljedeće što je potrebno je programiranje sklopa i spajanje s mobilnom aplikacijom kako bi se upravljalo pametnom kućom. Slika 4.2 prikazuje shematski prikaz spoja sklopa, dok slika 4.3 prikazuje laboratorijsku izvedbu sheme.



Slika 4.2. Prikaz sheme spoja



Slika 4.3. Izvedba shematskog spoja sklopa

4.3. Programsko rješenje upravljanja pametnom kućom

4.3.1. WebServer i WebSocket

Web poslužitelj postavlja se na pristup 80 te se *webSocket* stavi da sluša na pristupu 81 kao što se može vidjeti na slici 4.4. Metoda *webSocketEvent()*, vidljiva na slici 4.4, provjerava je li se *webSocket* pravilno spojio, odspojio ili je li primljena poruka koja se šalje preko aplikacije na web poslužitelj. U slučaju kada je primljena poruka, uspoređuje se tekst kroz *if* uvjete te se odrađuje ono što je zadano pod odgovarajućim *if* uvjetom. Taj dio programskog koda nalazi se u potpoglavlju 4.3.3.

```
ESP8266WebServer server = ESP8266WebServer(80);
WebSocketsServer webSocket = WebSocketsServer(81);

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
  switch (type) {
    case WStype_DISCONNECTED:
      Serial.printf("[%u] Disconnected!\n", num);
      break;
    case WStype_CONNECTED: {
        IPAddress ip = webSocket.remoteIP(num);
        Serial.printf("[%u] Connected from %d.%d.%d.%d url: %s\n", num, ip[0], ip[1], ip[2], ip[3], payload);

        // send message to client
        webSocket.sendTXT(num, "Connected");
      }
      break;
    case WStype_TEXT:

      // Print out raw message
      Serial.printf("[%u] Received text: %s\n", num, payload);
```

Slika 4.4. Metoda obrade stanja WebSocket-a

Socket poslužitelj nakon kreiranja se postavlja da svaki put kada se pojavi neki događaj, izvodi metodu *webSocketEvent()*. Slika 4.5 prikazuje pokretanje metode *webSocketEvent()* svaki put kada *webSocketa* primi neki događaj.

```
Serial.println("Starting Socket");
// start webSocket server
webSocket.begin();
webSocket.onEvent(webSocketEvent);
Serial.println("Socket Started");
```

Slika 4.5. Pokretanje WebSocket poslužitelja

4.3.2. Pristupna točka bežične mreže

Pristupna točka bežične mreže se postavlja metodom *softAP()*, vidljivom na slici 4.6 i predanim joj parametrima koji predstavljaju naziv i lozinku pristupne točke.

```
Serial.println("Starting AP");
WiFi.mode(WIFI_AP);
WiFi.softAP("Naziv", "lozinka");
Serial.println("AP Started");
```

Slika 4.6. Pokretanje i postavljanje pristupne točka bežične mreže

4.3.3. Upravljanje sustavom

Upravljanje paljenjem svjetala se izvodi tako da tekst koji se primi preko *webSocketa* testira s raznim slučajevima. U ovom radu to su testiranja traži li se paljenje svjetla sa „ON" ili gašenja sa „OFF", te koja svjetleća dioda je u pitanju, odnosno kojim nožicama je potrebno upravljati. Vrata se upravljaju putem ključnih riječi „OPEN" i „CLOSE" te nožicom na koju je spojen element. Tekst koji je sustav primio preko *webSocketa* uspoređuje se s ključnim riječima koje su određene unutar programa. Ako je dobiveni tekst jednak određenoj ključnoj riječi, program će izvesti odabranu radnju. Slika 4.7 prikazuje *if* uvjete paljenja i gašenja svjetleća dioda, dok slika 4.8 prikazuje uvjete za otvaranje i zatvaranje vrata pokretanjem servo motora.

```
// Toggle LED
if ( strcmp((char *)payload, "led1ON") == 0 ) {
  digitalWrite(led1, HIGH);
}
else if ( strcmp((char *)payload, "led1OFF") == 0 ) {
  digitalWrite(led1, LOW);
}
else if ( strcmp((char *)payload, "led2ON") == 0 ) {
  digitalWrite(led2, HIGH);
}
else if ( strcmp((char *)payload, "led2OFF") == 0 ) {
  digitalWrite(led2, LOW);
}
else if ( strcmp((char *)payload, "led3ON") == 0 ) {
  digitalWrite(led3, HIGH);
}
else if ( strcmp((char *)payload, "led3OFF") == 0 ) {
  digitalWrite(led3, LOW);
}
```

Slika 4.7. Dio uvjeta if za upravljanje paljenjem i gašenjem pametnom rasvjetom

```
else if ( strcmp((char *)payload, "OPEN16") == 0 ) {
  for (pos = 0; pos <= 180; pos += 5) {
      myservo16.write(pos);
      delay(15);
  }
}
else if ( strcmp((char *)payload, "CLOSE16") == 0 ) {
  for (pos = 180; pos >= 0; pos -= 5) {
      myservo16.write(pos);
      delay(15);
    }
}
else {
  Serial.println("[%u] Message not recognized");
}
break;
```

Slika 4.8. Dio uvjeta if za upravljanje otvaranja i zatvaranja vrata putem servo motora

5. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

U ovom poglavlju opisana je razvojna programska okolina i tehnologije za izradu mobilne aplikacije, programsko rješenje mobilne aplikacije na strani klijenta i ostvarenje komunikacije s upravljačkim dijelom sustava pametne kuće.

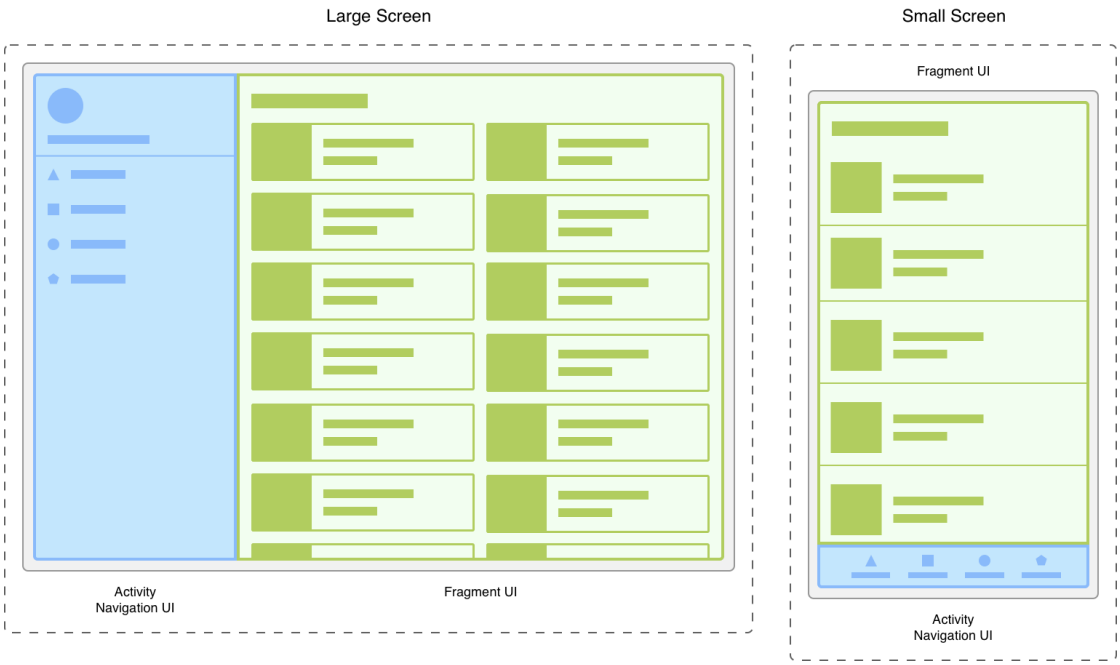
5.1. Korištene tehnologije, programski jezici i razvojna okolina

5.1.1. Program Android Studio

Korištena razvojna okolina za razvoj aplikacije je Android Studio koji je službeno integrirano razvojno okruženje Androida. Izgrađen je s JetBrains IntelliJ IDEA programom i dizajniran posebno za razvoj Androida. Program Android Studio podržava razne programske jezike kao što su: Java, C++ i Kotlin,

5.1.2. Fragmenti

Fragmenti, u okviru programiranja, predstavljaju klasu koja omogućuje modularno dizajniranje aktivnosti (eng. activity). To je dodatni sloj smješten između sučelja i aktivnosti. Svaki fragment ima vlastiti životni ciklus koji je usko povezan s aktivnosti te se preko njega može manipulirati pojedinim fragmentima. Rad s fragmentima nije jednostavan te prilikom njihovog korištenja potrebno je biti oprezan. Fragmenti unose modularnost i ponovnu uporabu u korisničko sučelje aktivnosti te omogućuje podjelu sučelja na zasebne dijelova kao što je vidljivo na slici 5.1[25].



Slika 5.1. Prikaz podjele sučelja putem fragmenata [25]

5.1.3. Klasa WebSocket

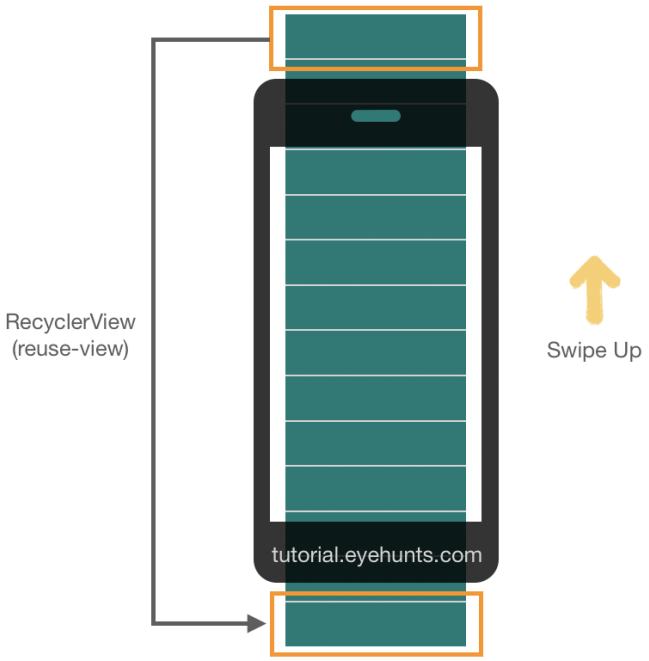
WebSockets API je napredna tehnologija koja omogućuje dvosmjernu interaktivnu komunikaciju između korisničkog preglednika i poslužitelja. Pomoću ovog API-ja moguće je slati poruke poslužitelju i primiti odgovore na temelju poslane poruke. Podaci se mogu proslijediti u bilo kojem trenutku što je korisno za razgovaranje u stvarnom vremenu.

5.1.4. Klasa AsyncTasks

Asinkroni zadatak je pozadinski zadatak koji se izvodi u pozadini niti i čiji je rezultat objavljen u niti korisničkog sučelja. To znači da će se ovakav zadatak obavljati u niti odvojene od glavne niti na kojem se izvodi program, ali će se njen rezultat prikazati u glavnoj niti.

5.1.5. Lista RecyclerView

Lista RecyclerView je komponenta koja se koristi pri prikazivanju velike veličine podataka te kada je potrebna dinamičnu izmjenu podataka sa strane korisnika. Ova komponenta prikazuje svoje elemente u obliku liste koja se može pomicati gore i dolje. Ima mogućnost recikliranja svojih elemenata te tako ubrzava svoj rad. Na slici 5.2 [26] prikazan je način na koji lista RecyclerView prikazuje svoje elemente. Lista učitava sve elemente koji su prikazani na aplikaciji te jedan element iznad i ispod tako da kada korisnik želi pomaknuti listu, element će već biti spreman. Svakim pomakom se učitavaju novi dodatni elementi dolje ili gore.



Slika 5.2. Prikaz liste RecyclerView [26]

5.1.6. Klasa SpeechRecognizer

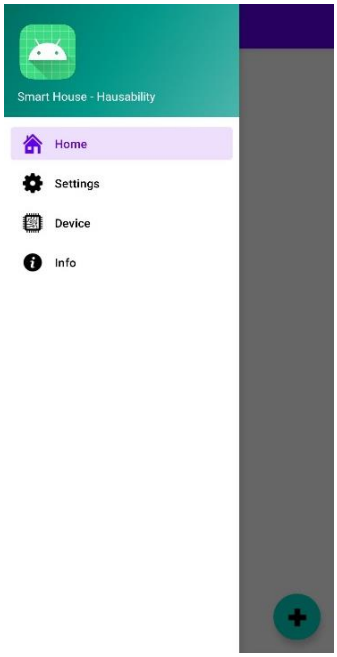
SpeechRecognizer je klasa koja pruža pristup usluzi prepoznavanja govora. S obzirom da ova implementacija API-ja šalje audio na udaljene poslužitelje radi prepoznavanja govora, API nije namijenjen za kontinuirano prepoznavanje govora.

5.1.7. Klasa TextToSpeech

TextToSpeech je klasa koja sintetizira govor iz teksta za trenutnu reprodukciju ili stvaranje zvučne datoteke. Može se koristiti tek nakon inicijalizacije te nakon toga postoji instanca putem koje se sintetizira uneseni tekst u govor.

5.2. Programsko rješenje na strani klijenta

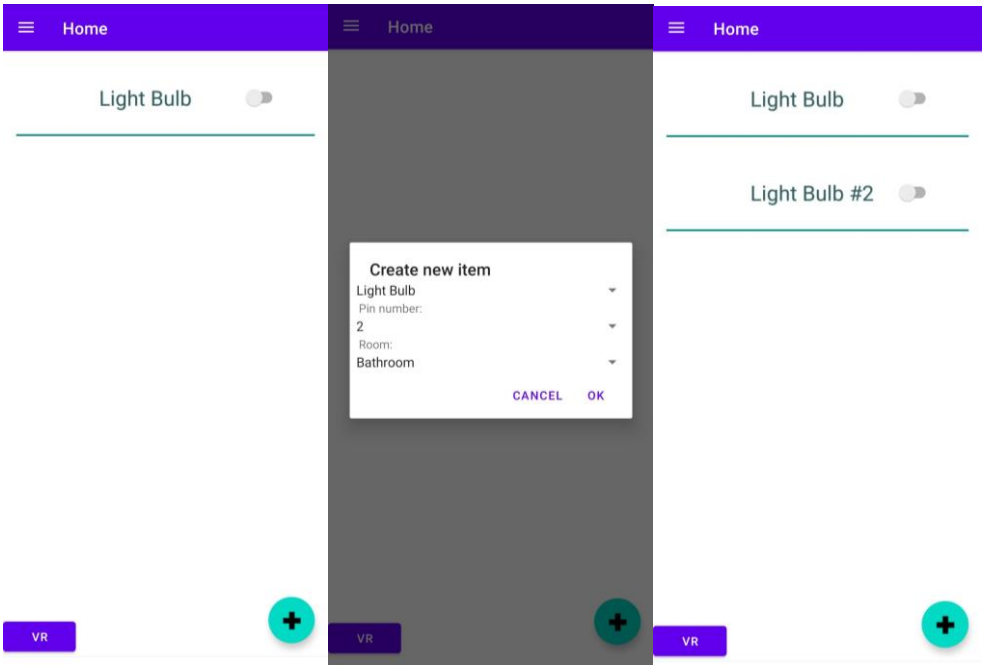
Aplikacija je izrađena koristeći „drawer layout“ tj. ladični prikaz. Ovakav prikaz radi na principu ladica tj. preko jedne glavne strukture se može odabrati pojedine ladice koje predstavljaju fragmente. Slika 5.3 prikazuje ladični prikaz unutar aplikacije.



Slika 5.3. Ladični prikaz tj. menu aplikacije

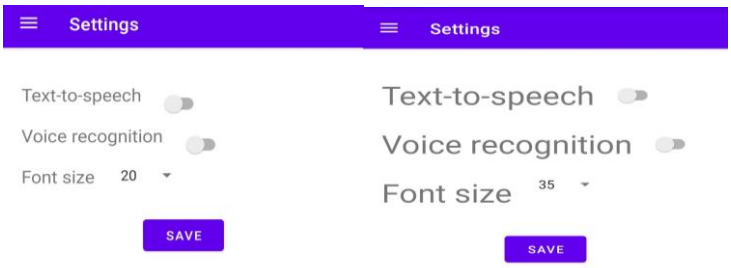
Odabirom na bilo koju od ladica, dobiva se njen pripadajući fragment. Tako odabirom na „Home” dobiva se fragment s istim nazivom. Ovaj odlomak sadrži listu pametnih stvari u sobi. Kao što je vidljivo na slici 5.4, dodirom ili čak samim riječima „add item” tj. dodaj novu stavku, aktivira se proces dodavanja stavki. Za dodavanje stavki se podrazumijeva poznavanje pozicije tj. prostorije u kojoj se stavka nalazi te nožica na samom uređaju na kojem je sve izrađeno. S obzirom da je ova aplikacija dizajnirana oko sklopa Nova2, samo neke nožice su slobodne za upotrebu te su navedene

u padajućoj listi. Moguće je više stavki spojiti na iste nožice te se tako stvara serijski spoj gdje uključivanjem jednog elementa uključuju se i ostali elementi. Isto vrijedi i za isključivanje. Svaka stavka ima svoj prekidač kojim se upravlja stanjem svake stavke. U donjem desnom kutu se nalazi gumb VR, odnosno „voice recognition" koji uključuje slušanje mikrofona te tako korisnik može upravljati aplikacijom i stavkama.



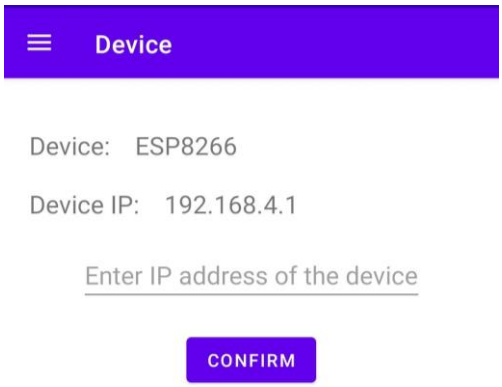
Slika 5.4. Prikaz dodavanja novog elementa u listi

U sljedećem fragmentu vidljivom na slici 5.5 postoje postavke koje su za sada samo postavke paljenja teksta u govor, prepoznavanja prirodnog govora i postavke veličine. Prve dvije postavke omogućuju upravo ono što i kažu tj. ako se ima uključen tekst u govor, aplikacija će proizvesti predprogramirani zvuk, kao primjerice predstavljanje imena određenog fragmenta i iščitavanje njegovog sadržaja. Prepoznavanje prirodnog govora omogućuje korisniku da upravlja aplikacijom preko programiranih ključnih riječi koje se uvijek mogu dopuniti kako bi odgovarale svakom korisniku. Promjenom veličine teksta mijenja se veličinu teksta u cijeloj aplikaciji te tako ljudi kojima je problem čitati manja slova mogu namjestiti veličinu koja im odgovara.



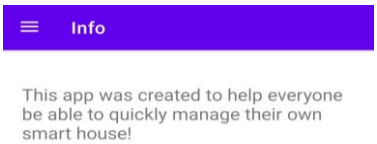
Slika 5.5. Prikaz promijene veličine teksta

Fragment uređaj je prvobitno dizajniran oko uređaja ESP8266 te su potrebne nadogradnje kada bi se krenuli koristiti drugi uređaju koji ne rade na istom ili sličnom principu. Ideja je da svaki sklop ESP8266 ima istu IP adresu kada je on pristupna točka te se od korisnika ne zahtijeva IP jer je sam po kodu programiran, no u slučaju da postoji takav sklop koji ima drugu IP adresu, ista bi se dala korisniku zajedno s uređajem pa bi instalacija bila lagana. Slika 5.6 prikazuje izgled device fragmenta.



Slika 5.6. Prikaz fragmenta device

Zadnji fragment informacija predstavlja odlomak koji ukratko govori korisniku o ideji aplikacije te bi tu za komercijalne svrhe stavili kontaktne informacije u slučaju nekih poteškoća kako bi korisnici mogli što lakše kontaktirati servis. Slika 5.7 prikazuje izgled informacijskog fragmenta.



Slika 5.7. Prikaz fragmenta info

5.3. Programsko rješenje na strani poslužitelja

5.3.1. Fragment Home

Početni fragment je zapravo i najvažniji fragment zato što sadrži *RecyclerView* koji ima sve stavke u sebi. Već prije spomenuti gumb VR preko instance *recyclerAdapter* poziva metodu *speechRecognize()* kao što je vidljivo na slici 5.8.

```
switchState.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        recyclerAdapter.speechRecognize();
    }
});
```

Slika 5.8. Pokretanje slušanja glasovnog ulaza

Metoda *speechRecognize()* se zapravo bavi obradom prirodnog jezika i to tako da se aktivira instanca *speechRecognizer*-a te putem nje se sluša korisnikov glasovni ulaz neko vrijeme. Nakon slušanja, aktivira se metoda *OnResults*() u kojoj se uspoređuje tekst koji je „čuo" *speechRecognizer*. Moguće usporedbe nisu konačne te kao što se može vidjeti na slikama 5.10 i 5.11, nisu dovršene. Zbog svrha testiranja za sada radi samo u slučaju dodavanja rasvjete i to samo prva dva elementa *RecyclerView* liste. Za dalju upotrebu bi se trebalo dodati za neograničeno dodavanje unutar liste te korištenje istoga za druge mogućnosti kao što su otvaranje vrata i slično. Slika 5.9 prikazuje inicijalizaciju instance *speechRecognizer*.

```
public void speechRecognize(){
    intentRecognizer = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intentRecognizer.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

    speechRecognizer = SpeechRecognizer.createSpeechRecognizer(recycler.getContext());
    speechRecognizer.setRecognitionListener(new RecognitionListener() {
```

Slika 5.9. Inicijalizacija speechRecognizer-a

Slike 5.10 i 5.11 prikazuju isječke nekih od mogućih usporedbi koje se nalaze u kodu. Naredbe se mogu nadodavati kako bi se obuhvatile razne radnje upravljanja pametnom kućom. Slika 5.10 prikazuje upravljanje rasvjetom glasovnim naredbama, dok slika 5.11 prikazuje upravljanje vratima.

```
@Override
public void onResults(Bundle results) {
    ArrayList<String> matches = results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    String string;
    SwitchCompat itemSwitch;
    if (matches != null) {
        string = matches.get(0);
        if (string.equals("light on") || string.equals("light one on")) {
            itemSwitch = getSwitch( string: "Light Bulb");
            itemSwitch.setChecked(true);
        } else if (string.equals("light off") || string.equals("light one off")) {
            itemSwitch = getSwitch( string: "Light Bulb");
            itemSwitch.setChecked(false);
        }
        else if (string.equals("light two on")) {
            itemSwitch = getSwitch( string: "Light Bulb #2");
            itemSwitch.setChecked(true);
        }
```

Slika 5.10. Prvi dio metode onResults() speechRecognizer-a

```
else if (string.equals("open door") || string.equals("open door one")) {
    itemSwitch = getSwitch( string: "Door");
    itemSwitch.setChecked(true);
}
else if (string.equals("close door") || string.equals("close door one")) {
    itemSwitch = getSwitch( string: "Door");
    itemSwitch.setChecked(false);
}
```

Slika 5.11. Drugi dio metode onResults() speechRecognizer-a

Metoda *getSwitch()* koja je korištena u slikama 5.10 i 5.11 prikazana je na slici 5.12. Ova metoda dohvaća određenu sklopku *RecyclerView* liste koja se pokušava dobiti kako bi joj promijenili stanje. Dohvaćanje sklopke se izvodi tako da se pronađe pozicija elementa čije je ime predano kao parametar metode.

```
public SwitchCompat getSwitch (String string){
    boolean test_flag;
    int item_position = 0;
    for(int i = 0; i< dataList.size(); i++){
        test_flag = dataList.get(i).getmName().equals(string);
        if(test_flag)item_position = i;
    }
    return (SwitchCompat) recycler.findViewHolderForAdapterPosition(item_position).
            itemView.findViewById(R.id.item_switch);
}
```

Slika 5.12. Metoda getSwitch()

Dodavanje novih stavki u dinamičnu listu se čini putem klika na donji desni gumb koji ima identifikaciju *addItem*. Kada aplikacija registrira taj klik, aktivira se metoda *createNewItem()*. Ta aktivaciju je vidljiva na slici 5.13.

```
binding.addItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { createNewItem(); }
});
```

Slika 5.13. Pokretanje metode createNewItem() putem klika

Metoda *createNewItem()* kreira novi prikaz putem kojeg se odabiru informacije o novoj stavki. Prvo se definira sve što je potrebno prikazati kao npr. *TextView* ili *Spinner* tzv. padajuća lista. Kada su svi elementi definirani, stave se u jedan prikaz te taj prikaz se izgradi putem *AlertDialog.Builder* instance koja izrađuje dizajniran iskočni prozor. Slika 5.14 prikazuje inicijalizaciju posebnih dijelova koji će biti vidljivi u iskočnom prozoru.

```
public void createNewItem(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivity());
    builder.setTitle("Create new item");

    LinearLayout layout = new LinearLayout (this.getActivity());
    layout.setOrientation(LinearLayout.VERTICAL);

    final Spinner nameInput = new Spinner(this.getActivity());

    ArrayAdapter<CharSequence> nameAdapter = ArrayAdapter.createFromResource(this.getContext(),
            R.array.items, android.R.layout.simple_spinner_item);
    nameAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    nameInput.setAdapter(nameAdapter);

    final TextView pinText = new TextView(this.getActivity());
    final Spinner pinInput = new Spinner(this.getActivity());
```

Slika 5.14. Metoda createNewItem() za inizijalizaciju dijelova iskočnog prozora

Iskočni prozor sadrži opciju „DA” i „NE” te stoga obje opcije se moraju programirati. Klikom na opciju „NE” se zatvara iskočni prozor bez ikakvih izmjena, no klikom na opciju „DA“ potrebno je dodati novi element u listu. Iskočni prozor je dizajniran sa dvije padajuće liste, jedna koja predstavlja elemente koji se dodaju u *RecyclerView* listu te druga padajuća lista koja prikazuje pripadajuće nožice svakog elementa. Pošto svaki element koji se nalazi u padajućoj listi ima drugačiju listu, potrebno je postaviti metodu *setOnItemSelectedListener()* koja kada očita promjenu u padajućoj listi, uspoređuje koji element je prikazan te po tome prikazuje njihove nožice. Slika 5.15 prikazuje metodu *setOnItemSelectedListener()* i postupak usporedbe. Slika 5.16 prikazuje izgradnju iskočnog prozora putem *AlertDialog.Builder* instance te kreiranje novog elementa nakon pozitivnog odabira.

```
pinText.setText("    Pin number:");
ArrayAdapter<CharSequence> pinAdapter = ArrayAdapter.createFromResource(this.getContext(),
        R.array.LED_pins, android.R.layout.simple_spinner_item);
pinAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
pinInput.setAdapter(pinAdapter);

nameInput.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        if(nameInput.getSelectedItem().toString().equals("Light Bulb")){
            ArrayAdapter<CharSequence> pinAdapter = ArrayAdapter.createFromResource(builder.getContext(),
                    R.array.LED_pins, android.R.layout.simple_spinner_item);
            pinAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            pinInput.setAdapter(pinAdapter);
        }
        else if(nameInput.getSelectedItem().toString().equals("Door")){
            ArrayAdapter<CharSequence> pinAdapter = ArrayAdapter.createFromResource(builder.getContext(),
                    R.array.door_pins, android.R.layout.simple_spinner_item);
            pinAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            pinInput.setAdapter(pinAdapter);
        }
    }
```

Slika 5.15. Metoda setOnItemSelectedListener() za padajuću listu iskočnog prozora

```
        layout.addView(nameInput);
        layout.addView(pinText);
        layout.addView(pinInput);
        layout.addView(roomText);
        layout.addView(roomInput);
        builder.setView(layout);

// Set up the buttons
        builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String mItemName = nameInput.getSelectedItem().toString();
                int pin = parseInt(pinInput.getSelectedItem().toString());
                //int pin = data.size()+1;
                int cnt = 0;
                for(int i = 0; i < mainActivity.getListData().size(); i++){
                    if(mainActivity.getListData().get(i).getmName().contains(mItemName)){
                        cnt++;
                    }
                }
```

Slika 5.16. Iskočni prozor kreiran metodom createNewItem() i njegove opcije

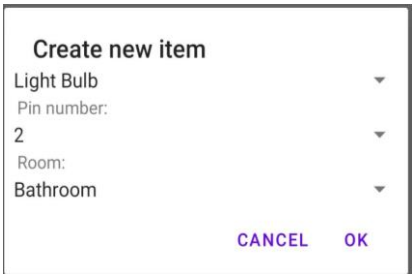
Lista je programirana tako da se automatski dodaje broj elementa tj. ako već postoji jedno svjetlo, sljedeće će imati „#2” do imena. To je napravljeno preko traženja najvećeg postojećeg broja u imenu svih stavki. Slika 5.17 prikazuje proces imenovanja sa dodatnim brojem.

```
            int max = 0;
            if(cnt!=0){
                String number =  String.format(Locale.ENGLISH, format: " #%d", ...args: cnt+1);
                mItemName = nameInput.getSelectedItem().toString() + number;
                max = cnt+1;
                for(int i = 0; i < mainActivity.getListData().size(); i++){
                    if(mainActivity.getListData().get(i).getmName().contains(mItemName)){
                        String substr = mItemName.substring(mItemName.indexOf("#")+1);
                        int number2 = parseInt(substr);
                        if(number2 > max) max = number2;
                        if(number2 == max) max++;
                    }
                }
                String maxstr =  String.format(Locale.ENGLISH, format: " #%d",max);
                mItemName = nameInput.getSelectedItem().toString() + maxstr;
            }

            if(recyclerAdapter.getItemCount() > 8){
                addItem.hide();
            }
            data.add(new ItemInRoom(mItemName, pin, mainActivity.getSwitchData()[recyclerAdapter.getItemCount()]));
            addCell(mItemName, pin);
            recyclerAdapter.writeToFile(requireContext());
        }
    });
    builder.setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) { dialog.cancel(); }
    });
    builder.show();
}
```

Slika 5.17. Postavljanje naziva novog elementa u listi

Slika 5.18 prikazuje iskočni prozora kreiran putem koda na slikama 5.15 i 5.16.



Slika 5.18. Prikaz iskočnog prozora kreiranog metodom createNewItem()

5.3.2. Fragment Settings

U ovom fragmentu se mijenjaju postavke aplikacije koje se primjenjuju tek klikom na gumb „SAVE”. Promjenom stanja sklopke se uključuje određenu postavku. Klikom na gumb „SAVE” spremaju se trenutna stanja svih elementa u privatni tekstualni dokument kako bi ih pri sljedećem paljenju aplikacije imali zapamćene. Ovaj tekstualni dokument može vidjeti samo aplikacija te je za potrebe ovoga rada dovoljno, no daljnje nadogradnje bi mogle sadržavati bolji sustav snimanja stanja kao što su baze podataka na vanjskom poslužitelju. Slika 5.19 prikazuje metodu koja se aktivira klikom na gumb „SAVE“. Slika 5.20 prikazuje metodu *writeToFile()* koja sprema informacije fragmenta u privatnu tekstualnu datoteku.

```
    saveSettings.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mainActivity.setData( i: 0, String.valueOf(TTS));
            mainActivity.setData( i: 1, String.valueOf(VR));
            FS = FS_spin.getSelectedItem().toString();
            mainActivity.setData( i: 2, FS);

            String data = TTS + "\n" + VR + "\n"+ FS;
            writeToFile(data);

            for(int i = 0; i < 3; i++){
                textViews[i].setTextSize(Integer.parseInt(FS));
            }
            if(mainActivity != null) TTS_status();
        }
    });

    if(mainActivity != null) TTS_status();

    return root;
}
```

Slika 5.19. Prikaz metode koja se aktivira klikom na gumb „SAVE“

```
public void writeToFile(String data){
    try {
        OutputStreamWriter outputStreamWriter = new OutputStreamWriter(getContext().openFileOutput( name: "settings.txt", Context.MODE_PRIVATE));
        outputStreamWriter.write(data);
        //Toast.makeText(getContext(), mainActivity.getFilesDir(), Toast.LENGTH_LONG).show();
        outputStreamWriter.close();
    } catch (IOException e) {
        Log.e( tag: "Exception", msg: "File write failed: " + e.toString());
    }
```

Slika 5.20. Prikaz principa ispisivanja informacija u privatnu tekstualnu datoteku

Iščitavanje spremljenih informacija pri kreiranju fragmenta se izvodi tako da se očita cijeli dokument u jedan string te se potom odvaja u više zasebnih cjelina putem zareza, oznake za novi red ili sličnih elemenata koji su postavljeni za takvu svrhu. U svim fragmentima se koriste varijacije funkcije koja piše i čita iz tekstualnih dokumenata koji su specificirani za njihove potrebne informacije. Slika 5.21 prikazuje čitanje dokumenta i njegovo odvajanje u smislene informacije koje se potom dalje koriste.

```
public String readFromSettingsFile(Context context){
    String ret = "";
    int i = 0;
    try {
        InputStream inputStream = context.openFileInput( name: "settings.txt");

        if ( inputStream != null ) {
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String receiveString = "";
            StringBuilder stringBuilder = new StringBuilder();

            while ( (receiveString = bufferedReader.readLine()) != null ) {
                stringBuilder.append(receiveString).append("\n");
            }

            inputStream.close();
            ret = stringBuilder.toString();
        }
    }
    catch (FileNotFoundException e) {
        Log.e( tag: "login activity", msg: "File not found: " + e.toString());
    } catch (IOException e) {
        Log.e( tag: "login activity", msg: "Can not read file: " + e.toString());
    }
    return ret;
}
```

Slika 5.21. Prikaz principa iščitavanja informacija iz tekstualne datoteke

U slučaju kada je uključena pretvorba teksta u govor aktivira se funkcija prikazana na slici 5.22, *TTS_status()*. Ovakva funkcija se nalazi u svakom fragmentu sa svojim specifičnim tekstom opisa fragmenta. Koristeći se *TextToSpeech.QUEUE_FLUSH* aplikaciji zna da mora prekinuti već aktiviranu tekstualnu pretvorbu, ako takva postoji, te zatim započinje reprodukcijom odabrane pretvorbe teksta u govor. Tako se omogućuje korisniku izmjenjivanje fragmenata bez da mu kasni objašnjenje trenutnog fragmenta.

```
public void TTS_status(){
    if(mainActivity.getSettingsData( i: 0) != null && mainActivity.getSettingsData( i: 0).equals("true")){
        textToSpeechSystem = new TextToSpeech(getContext(), new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if (status == TextToSpeech.SUCCESS) {
                    String VR;
                    String FS;
                    if( mainActivity.getSettingsData( i: 1).equals("true"))
                        VR = "on";
                    else
                        VR = "off";
                    FS = mainActivity.getSettingsData( i: 2);
                    String textToSay = "Settings Fragment. Text to speech is on. Voice recognition is " + VR + ". Font size is " + FS;
                    textToSpeechSystem.speak(textToSay, TextToSpeech.QUEUE_FLUSH,  params: null);
                }
            }
        });
    }
}
```

Slika 5.22. Metoda iščitavanja stanja postavki putem pretvorbe teksta u govor

Kako bi promijenili veličinu teksta u svakom fragmentu potrebno je primijeniti novu dimenziju na svaki *TextView* tj. tekstualni prikaz. Jedno od mogućih rješenja je da se unutar svakog fragmenta u jedno polje postave svi tekstualni prikazi te se na njih onda primjeni nova dimenzija. Takvo rješenje je prikazano na slici 5.23.

```
textViews = new TextView[]{root.findViewById(R.id.TTS_txt), root.findViewById(R.id.VR_txt), root.findViewById(R.id.FS_txt)};
for(int i = 0; i < 3; i++){
    textViews[i].setTextSize(mainActivity.getFontSize());
}
```

Slika 5.23. Primjer principa primjene veličine fonta na tekstualne prikaze unutar fragmenata

5.3.3. Fragment Device

Osim već pojašnjenih načina spremanja i dohvaćanja istih spremljenih informacija, u ovom fragmentu postoji izmjena IP adrese unosom tipkovnicom te klikom na „CONFIRM” gumb. Nova IP adresa se potom pohranjuje, te u slučaju kada je uključena pretvorba teksta u govor, izriče se nova IP adresa uređaja kao što je prikazano na slici 5.24.

```
enterIP.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        IP.setText(IP_input.getText().toString());
        Toast.makeText(mainActivity, IP_input.getText().toString(), Toast.LENGTH_SHORT).show();
        writeToFile(IP_input.getText().toString());
        if(mainActivity.getSettingsData( i: 0).equals("true")){
            textToSpeechSystem = new TextToSpeech(getContext(), new TextToSpeech.OnInitListener() {
                @Override
                public void onInit(int status) {
                    if (status == TextToSpeech.SUCCESS) {
                        String textToSay = "The Device IP address is now " + IP_input.getText().toString();
                        textToSpeechSystem.speak(textToSay, TextToSpeech.QUEUE_FLUSH,  params: null);
                    }
                }
            });
        }
    }
});
```

Slika 5.24. Metoda spremanja stanja nove IP adrese

5.3.4. Pametne preporuke

Na sljedećoj slici se mogu vidjeti *if* uvjeti kojima se uspoređuje koliko je sati te po tome bira koju treba odraditi metodu ili ju uopće ne odabire. Metoda *OFFnotification()* je metoda koja pita korisnika putem iskočnog prozora želi li ugasiti sva svjetla ako je poslije 22h, a prije 5h. Ova metoda se uključuje samo ako je upaljeno barem jedno svjetlo u tom periodu. Korisnik može odbiti ili prihvatiti preporuku, ako prihvati, sva svjetla će se ugasiti. Metoda *ONnotification()* radi na sličnom principu. Na slici 5.25 su prikazani uvjeti aktivacije obje metode.

```
currentTime = Calendar.getInstance().getTime().getHours();
if(currentTime >= 22 || currentTime <= 5){
    int flag = 0;
    for(int i = 0; i < recyclerAdapter.getItemCount(); i++){
        if(mainActivity.getListData().get(i).getSwitchState())
            flag++;
    }
    if(flag!=0)
        OFFnotification();
}
else if(currentTime >= 6 && currentTime <= 10){
    int flag = 0;
    for(int i = 0; i < recyclerAdapter.getItemCount(); i++){
        if(mainActivity.getListData().get(i).getSwitchState())
            flag++;
    }
    if(flag==0)
        ONnotification();
}
```

Slika 5.25. Prikaz dobivanja vremena i pokretanje metoda ovisno o koliko je sati

Ako je uključen tekst u govor, metoda će naglas pitati korisnika želi li uključiti sva svjetla ako ni jedno nije uključeno u periodu od 6h do 10h ujutro. Slika 5.26 prikazuje metodu *ONnotification()* gdje se kreira iskočni prozor i čeka se odabir korisnika. Ako korisnik odabere pozitivno, sva svjetla se pale, ako ne, iskočni prozor se gasi.

```
public void ONnotification(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivity());
    builder.setTitle("It's a new day, do you want to turn on all lights?");
    LinearLayout layout = new LinearLayout (this.getActivity());
    layout.setOrientation(LinearLayout.VERTICAL);
    if(mainActivity.getSettingsData( i: 0).equals("true")){
        textToSpeechSystem = new TextToSpeech(getContext(), new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if (status == TextToSpeech.SUCCESS) {
                    String textToSay = "It's a new day, do you want to turn on all lights?";
                    textToSpeechSystem.speak(textToSay, TextToSpeech.QUEUE_ADD,  params: null);
                }
            }
        });
    }
Set up the buttons
    builder.setPositiveButton( text: "YES", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            recyclerAdapter.turnOnAllLights();

            if(mainActivity.getSettingsData( i: 0).equals("true")){
                textToSpeechSystem = new TextToSpeech(getContext(), new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status == TextToSpeech.SUCCESS) {
                            String textToSay = "All lights have been turned off.";
                            textToSpeechSystem.speak(textToSay, TextToSpeech.QUEUE_ADD,  params: null);
                        }
                    }
                });
            }
        }
    });
```

Slika 5.26. Metoda ONnotification()

Ako je odabran pozitivan odabir na iskočnom prozoru kreiranom metodom *ONnotification()* priziva se metoda *turnOnAllLights()* koja radi na principu metode *getSwitch()* već prikazanom na slici 5.12. Slika 5.27 prikazuje metode *turnOnAllLights()* i *turnOffAllLights()* koje pretražuju cijelu *RecyclerView* listu i gase ili pale sva svjetla putem usporedbe sadrži li element naziv „Light Bulb“.

```
public void turnOffAllLights(){
    for(int i = 0; i < dataList.size(); i++){
        if(dataList.get(i).getmName().contains("Light Bulb")){
            dataList.get(i).setSwitchState(false);
            SwitchCompat itemSwitch;
            itemSwitch = getSwitch(dataList.get(i).getmName());
            itemSwitch.setChecked(false);
        }
    }
}

public void turnOnAllLights(){
    for(int i = 0; i < dataList.size(); i++){
        if(dataList.get(i).getmName().contains("Light Bulb")){
            dataList.get(i).setSwitchState(true);
            SwitchCompat itemSwitch;
            itemSwitch = getSwitch(dataList.get(i).getmName());
            itemSwitch.setChecked(true);
        }
    }
}
```

Slika 5.27. Metode turnOffAllLights() i turnOnAllLights()

5.4. Komuniciranje u sustavu

Za komunikaciju između aplikacije i sklopovlja Nova2 potrebno je koristiti instancu *WebSocketClient* koja se kreira putem URI (Uniform Resource Identifier) elementa. URI je kompaktni niz znakova koji predstavlja apstraktni ili fizički resurs. U ovom slučaju taj resurs predstavlja link do stranice kojoj se pristupa putem IP adrese pristupne točke sklopa i uz pristup 81 na koji je postavljen poslužitelj *WebSocket*-a. Slika 5.28 prikazuje metodu kreiranja novog *webSocketClient*-a putem poznatog URI elementa.

```
public void createWebSocketClient() {
    String IP;
    if(mainActivity != null)
        IP = mainActivity.getIP();
    else
        IP = "192.168.4.1";
    URI uri = null;
   // URI uri2 = null;
    try {
        uri = new URI( str: "ws://192.168.4.1:81/websocket");
        //uri = new URI("ws://"+IP+":81/websocket");
    } catch (URISyntaxException e) {
        e.printStackTrace();
        return;
    }
    webSocketClient = new WebSocketClient(uri) {
        @Override
        public void onOpen() {
            webSocketClient.send("Connected.");
        }
```

Slika 5.28. Metoda kreiranja WebSocket klijenta

Slika 5.29 prikazuje postavke kojima se izvršava spajanje *webSocketClient*-a.

```
    };
    webSocketClient.setConnectTimeout(10000);
    webSocketClient.setReadTimeout(60000);
    webSocketClient.enableAutomaticReconnection( waitTimeBeforeReconnection: 5000);
    webSocketClient.connect();
}
```

Slika 5.29. Postavke spajanja WebSocket-a

Kada je *webSocket* spojen, onda se putem naredbe *send()* mogu slati tekstovi koje tada sklop može dohvaćati i uspoređivati sa svojim kodom koji tekst prepoznaje kao naredbu a koji ne. Na slici 5.30 je prikaz slanja naredbe da se uključi određeno svjetlo ovisno o položaju uključivanja sklopa. Isti ovakav princip se primjenjuje na gašenje svjetla samo što bi poslane naredbe tada bile „led1OFF" umjesto „led1ON". Isto tako je prikazan način upravljanja vratima tako da se šalje „OPEN16" gdje broj predstavlja nožicu a riječ naredbu, po toj logici gašenje je „CLOSE16". Kako

bi se osiguralo da sklop ne prima neke krive poruke, *webSocketClient* se otvara samo pri slanju naredbi te se potom zatvara. Postavljen je vremenski brojač prije zatvaranja *webSocketClient*-a kako bi se osiguralo da je poruka poslana.

```
if(item_switch != null) {
item_switch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            createWebSocketClient();
                String command = "";
                Integer pin;
                mainActivity.changeSwitchState(isChecked, position);
                mainActivity.getListData().get(position).setSwitchState(isChecked);

                mTimer = new CountDownTimer( millisInFuture: 5000,  countDownInterval: 500) {
                    @Override
                    public void onTick(long l) {
                    }

                    @Override
                    public void onFinish() { webSocketClient.close(); }
                };

                if (isChecked) {
                    pin = dataList.get(position).getPin();
                    if (pin == 4) {
                        command = "led1ON";
                    } else if (pin == 5) {
                        command = "led2ON";
                    } else if (pin == 12) {
                        command = "led3ON";
                    } else if (pin == 14) {
                        command = "led4ON";
                    } else if (pin == 13) {
                       command = "OPEN13";
                    } else if (pin == 16) {
                       command = "OPEN16";
                    }

                    if (webSocketClient != null) {
                        webSocketClient.send(command);
                        mTimer.start();

                    } else {
                        Toast.makeText(holder.itemView.getContext(),  text: "IP address error",
                                Toast.LENGTH_SHORT).show();
                    }
```

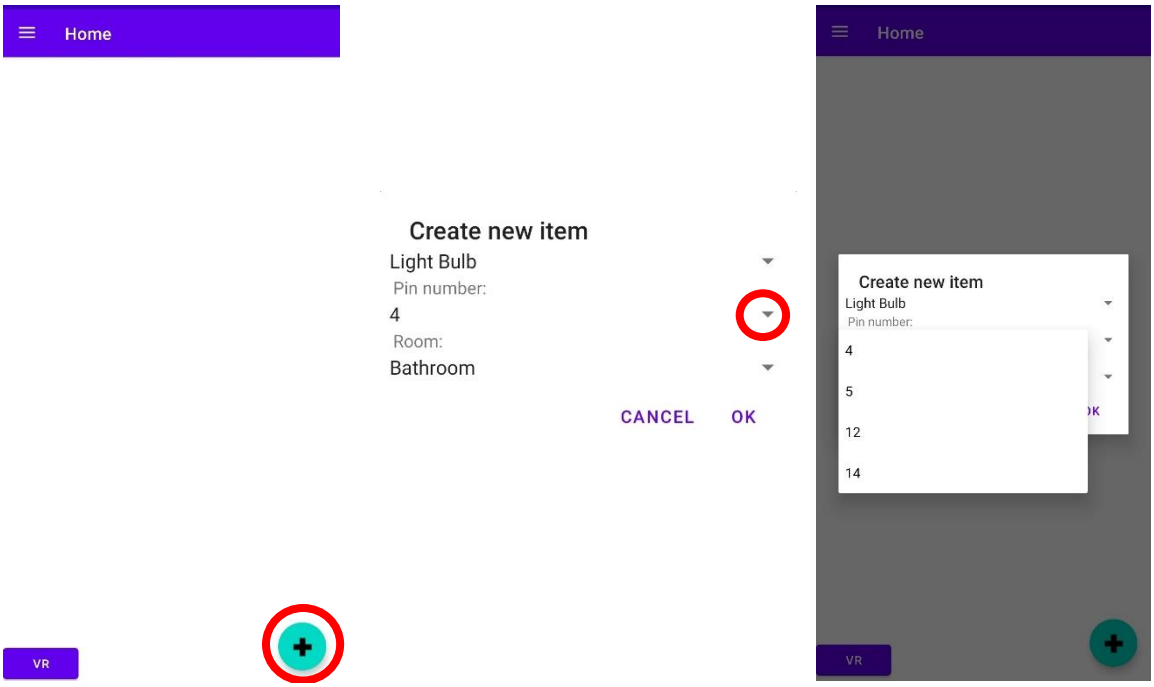
Slika 5.30. Metoda slanja stringa na WebSocket poslužitelj

6. OPIS NAČINA RADA I ISPITIVANJE SUSTAVA

Nakon izgradnje sustava, u poglavlju 6 opisan je način rada sustava pametne kuće s korisničke strane sustava, odnosno korištenje mobilne aplikacije. Zatim je mobilna aplikacija ispitana za nekoliko ispitnih slučajeva korištenja sustava, a rezultati ispitivanja analizirani.

6.1. Opis načina rada sustava

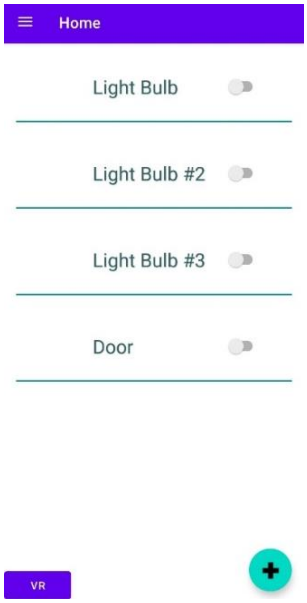
Glavni dio sustava je aplikacija koju korisnik može instalirati na bilo koji pametni uređaj kao što su mobitel ili tablet. Unutar nje, korisnik mora dodati pametne uređaje kao nove elemente u listi. Korisnik to može učiniti klikom na zaokruženi gumb na lijevoj slici slike 6.1. Klikom na taj gumb korisnik dobiva iskočni prozor kao na srednjoj slici slike 6.1. Klikom na padajuću listu, korisnik može odabrati broj nožice ili druge elemente kao što je vidljivo na desnoj slici slike 6.1.



Slika 6.1. Prikaz dodavanja novog elementa

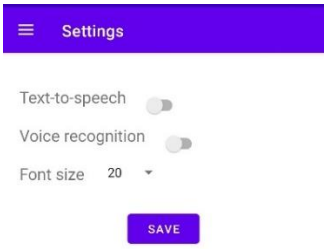
Nakon toga se dobije element unutar liste kao što se vidi na slici 6.2. Dodani element je postavljen na nožicu 4 te se ponavlja isti postupak kako bi se dodale još dvije rasvjete na nožicama 5 i 12, vrata se dodaju na nožicu 16. Kada je sve dodano, lista unutar aplikacije izgleda kao na slici 6.3. Korisnik može dodati elemente i glasovnim putem prateći isti redoslijed dodavanja. Dodavanje bi se tada izvelo ključnim riječima „Add new light. Pin 4.“ tj. „Dodaj novo svjetlo. Nožica 4.“

Slika 6.2. Rezultat dodavanja prvog elementa



Slika 6.3. Rezultat dodavanja dodatna tri elementa

Nakon što su postavljeni osnovni elementi, korisnik mora uključiti sklop i kada se pokaže pristupna točka koju otvori sklop, potrebno je prijaviti se na nju. Nakon prijave, korisnik može upravljati svojom novom pametnom kućom. Za dodatno upravljanje pametnom kućom, korisnik može uključiti bilo koju postavku koja omogućuje olakšano multimodalno upravljanje. Slika 6.4 prikazuje fragment postavki gdje korisnik ima mogućnost uređivanja upravljanja pametnom kućom i izgledom aplikacije.



Slika 6.4. Prikaz fragmenta postavki

6.2. Ispitivanja i analiza rezultata rada sustava

6.2.1. Ispitni slučaj 1

Za prvi ispitni slučaj uzet je primjer izlaska iz kuće. Kada se izlazi iz kuće, pogase se sva svjetla i vrata se zatvaraju na izlasku. Ovakva pametna kuća to može učiniti uz jednostavne naredbe kao što su „turn off all lights“ tj. „ugasi sva svjetla“ i naredba za zatvaranje vrata „close door“ ili „zatvori vrata“. Ispitivanje se započinje postavljanjem sklopa u spremno stanje i izgovaranjem naredbi u već spomenutom redoslijedu. Rezultati su prikazani na slici 6.4 gdje putem Arduino naredbe „Serial Monitor“ se mogu nadzirati primljene poruke. Sklop je pravilno primio naredbe u određenom redoslijedu i izvršio ih. U ovom slučaju, korisnik u postavkama ima uključenu opciju govora u tekst.

```
22:34:25.048 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:25.048 -> [2] Received text: OPEN16
22:34:25.755 -> [2] Disconnected!
22:34:27.588 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:27.588 -> [2] Received text: led1ON
22:34:27.869 -> [2] Disconnected!
22:34:28.390 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:28.390 -> [2] Received text: led2ON
22:34:28.438 -> [2] Disconnected!
22:34:29.143 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:29.143 -> [2] Received text: led3ON
22:34:29.378 -> [2] Disconnected!
22:34:30.606 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:30.606 -> [2] Received text: CLOSE16
22:34:31.357 -> [2] Disconnected!
```

Slika 6.5. Ispis na sklopu tijekom izvođenja prvog ispitnog slučaja

Tijekom slušanja naredbe za paljenje rasvjete, *speechRecognizer* se iznenada ugasio na prvom pokušaju te nije zabilježio naredbu. Drugi pokušaj je bio uspješniji te naredba za zatvaranje vrata nije imala problema pri aktivaciji. Glavni nedostatak sustava je servo motor koji ponekad otvori više, a ponekad manje. To bi značilo da se vrata ne otvore u cijelosti u svakom ispitivanju. U stvarnoj izvedbi ovoga sustava bi se svakako koristio drugačiji motor ili čak cilindar pa ovaj problem nema veliki utjecaj na ispravnost aplikacije.

6.2.2. Ispitni slučaj 2

Drugi ispitni slučaj predstavlja primjer povratka kući. Kada se ulazi u kuću, otvaraju se vrata, pale se sva svjetla i zatvaraju se vrata na ulasku. To se može učiniti naredbama „open doors“ tj. „otvori vrata“, „turn on all lights“ tj. „upali sva svjetla“ i naredba za zatvaranje vrata „close door“ ili „zatvori vrata“ nakon ulaska u kuću. Ispitivanje se započinje izgovaranjem naredbi u istom redoslijedu. Rezultati su prikazani na slici 6.5 gdje se vidi da je sklop pravilno primio naredbe u

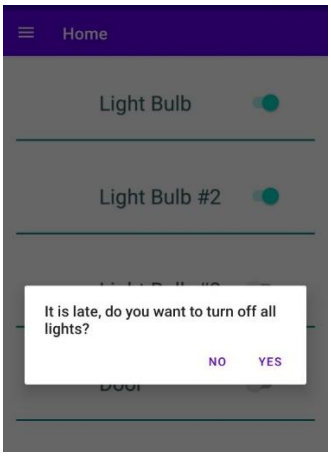
odgovarajućem redoslijedu i izvršio ih. U ovom slučaju, korisnik u postavkama ima uključenu opciju govora u tekst. Ovaj testni slučaj nailazi na slične probleme kao i prvi, osim glavnog problema u servo motoru, problem je u prepoznavanju riječi.

```
22:34:38.882 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:38.882 -> [2] Received text: OPEN16
22:34:39.593 -> [2] Disconnected!
22:34:40.958 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:40.958 -> [2] Received text: led1OFF
22:34:41.194 -> [2] Disconnected!
22:34:41.667 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:41.667 -> [2] Received text: led2OFF
22:34:41.667 -> [2] Disconnected!
22:34:42.468 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:42.468 -> [2] Received text: led3OFF
22:34:42.468 -> [2] Disconnected!
22:34:43.499 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:43.499 -> [2] Received text: CLOSE16
22:34:44.258 -> [2] Disconnected!
```

Slika 6.6. Ispis na sklopu tijekom izvođenja drugog ispitnog slučaja

6.2.3. Ispitni slučaj 3

Treći ispitni slučaj prikazuje gašenje i paljenje svjetla preko pametne preporuke ovisno o vremenu. Ovaj slučaj se izvodi bez korištenja glasovnih naredbi već samo dodira na aplikaciji. Ispitivanje je obavljeno nakon 22 sata tako da se korisnika pita želi li ugasiti sva svjetla. Ako korisnik odgovori pozitivno, sklop dobiva naredbu da ugasi sva svjetla koja su upaljena. Na slici 6.6 vidi se iskočni prozor koji se aktivira kada je kasno navečer, a primljene naredbe se vide na slici 6.7 Iskočni prozor sa slike 6.6. se aktivira samo od 22 do 5 sati te samo ako je barem jedno svjetlo ostalo upaljeno. Ovaj slučaj je odrađen bez ikakvih problema.



Slika 6.7. Prikaz iskočnog prozora za gašenje sva svjetla kada je kasno

```
22:34:06.493 -> [0] Connected from 192.168.4.2 url: /websocket
22:34:06.540 -> [0] Received text: led1OFF
22:34:06.540 -> [1] Connected from 192.168.4.2 url: /websocket
22:34:06.540 -> [2] Connected from 192.168.4.2 url: /websocket
22:34:06.540 -> [1] Received text: led2OFF
22:34:06.540 -> [2] Received text: led3OFF
22:34:06.540 -> [2] Disconnected!
```

Slika 6.8. Ispis na sklopu tijekom izvođenja trećeg ispitnog slučaja

6.3. Analiza rezultata ispitivanja sustava

U sva tri testna slučaja dobili su se pozitivni rezultati uz poneke greške. Prvi i drugi slučaj su koristili glasovno prepoznavanje putem klase SpeechRecognizer koja se pokazala kao manjkavo rješenje. U trenucima kada je naglasak bio malo drugačiji ili je izgovor bio nejasan, prepoznavanje riječi nije uspjelo ili je prepoznata riječ kriva. Osim samog prepoznavanja govora, najveći nedostatak je vidljiv u samostalnom neželjenom prekidanju slušanja koje se ne događa svaki put. Bolja način koji podržava kontinuirano slušanje i točniji je, jest Google Cloud Speech-To-Text, koji zahtijeva plaćanje za korištenje. Treći slučaj koji je koristio naredbu putem pametne preporuke je uspješno izveo svoju funkciju bez ikakvih problema.

Ispitivanjem je dokazano da aplikacija dobro radi u slučaju dodira. Aktiviranje pojedinog elementa, kao i njegovo gašenje se izvodi bez vidljive odgode te aplikacija bez problema izvodi zadane funkcije. Glasovne naredbe, iako imaju svoje probleme, odrađuju svoju funkciju kao što se može vidjeti na slikama prva dva testna slučaja. Duže naredbe imaju veću mogućnost da se ne zabilježe te je potrebno ponavljanje naredbe ili, kao već spomenuto, instanca *speechRecognizer* se ugasi prije nego li korisnik uspije reći i jednu riječ. Pretvorba teksta u govor radi bez ikakvih problema te brzom promjenom između fragmenata tekuća pretvorba se prekine bez odgode i odmah slijedi sljedeća pretvorba teksta u govor.

ZAKLJUČAK

Cilj ovog završnog rada bio je izraditi mobilnu Android aplikaciju za multimodalno upravljanje pametnom kućom za osobe s invaliditetom. U radu su analizirani načini na koje se može poboljšati kvaliteta života osobama s invaliditetom, ali i drugima. Upravo zbog toga je i potrebna multimodalnost, jer dvije osobe s invaliditetom ne moraju imati isti problem, niti iste poteškoće u upravljanju kućom. Upravo suprotno, jedna osoba može imati probleme s vidom, druga sa sluhom i slično. Glavni dio praktičnog dijela rada je izrada mobilne aplikacije s multimodalnim funkcijama. Aplikacija je kreirana s ciljem jednostavnog korištenja tako da od krajnjeg korisnika ne bi trebala zahtijevati nikakva znanja o kodu. Na jednostavan način moguće je upravljati raznim funkcionalnostima vlastitog doma s ciljem ugodnog življenja korisnika, a mogućnosti nadogradnje nisu ograničene. Aplikacija je najprije izrađena za osobe s invaliditetom, a iz tog razloga ima mogućnost multimodalnog upravljanja što je omogućeno pretvorbom teksta u govor, govora u tekst te klasičnim funkcijama koje se aktiviraju na dodir. Testiranje aplikacije pokazalo je da aplikacija radi prema postavljenim zahtjevima.

Aplikacija ima puno prostora za poboljšavanje, a dodatne nadogradnje bi mogle podržavati više elemenata od upravljanja vratima i rasvjetom, više sklopovskih mogućnosti koje korisnik može birati prema svojim potrebama i druge dodatne mogućnosti multimodalnosti kao što su geste.

LITERATURA

[1] Invalidnost, Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2021. Dostupno na: http://www.enciklopedija.hr/Natuknica.aspx?ID=27705 [24. lipnja 2021.]

[2] T. Benjak i sur, Izvješće o osobama s invaliditetom u Republici Hrvatskoj. Hrvatski zavod za javno zdravstvo, Zagreb, 2019. [24. lipnja 2021.]

[3] A. Tulenkov, Y. Yaramchenko, Anzhelika Parkhomenko, Y. Zalzubovskiy, Andriy Parkhomenko, M. Kalinina, Adaption of Smart House System for People with Special Needs Based on Wireless Technologies, Međunarodna konferencija o inteligentnom prikupljanju podataka i naprednim računalnim sustavima, Dortmund, Njemačka, rujan 2020. [30. lipnja 2021.]

[4] M. Li, W. Gu, W. Chen, Y. He, Y. Wu, Y. Zhang, Smart Home: Architecture, Technologies and Systems, Elsevier, 2018. [20. kolovoza 2021.]

[5] S. J. Darby, Smart technology in the home: time for some clarity, Building Research & Information, ožujak 2017. 7 [30. lipnja 2021.]

[6] S. Balakrishnan, H. Vasudavan, R. Kumar Murugesan, Smart Home Technologies: A Preliminary Review, Zbornik radova 6. međunarodne konferencije o informacijskim tehnologijama: IoT i pametni grad, prosinac, 2018. [20. kolovoza 2021.]

[7] D. Spivey, Home Automation for Dummies, John Wiley & Sons, New Jersey, 2015. [30. lipnja 2021.]

[8] K. Gill, S.-H. H. Yang, F. Yao, X. Lu, A ZigBee-Based Home Automation System, Transakcije o potrošačkoj elektronici, svibanj 2009. [30. lipnja 2021.]

[9] S. Park, S. Won, J. Lee, Smart home – digitally engineered domestic life, srpanj 2003. [30. lipnja 2021.]

[10] M. Jasinski, Smart Mirror, Smart Lab, Information Technology and Management Project, Tehnološki institut Illinois, 2016. Dostupno na: https://appliedtech.iit.edu/smart-lab-information-technology-and-management/projects/smart-mirror [30. lipnja 2021.]

[11] B. K. Sovacool, D. D. Furszyfer Fel Rio, Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies, Elsevier, 2019. [30. lipnja 2021.]

[12] GfK, Smart Home: A Global Perspective, CES, USA, 2016. Dostupno na: https://www.slideshare.net/GfK_en/ces-2016-gfk-smart-home-presentation [30. lipnja 2021.]

[13] C. Wilson, T. Hargreaves, R. Hauxwell-Baldwin, Benefits and risks of smart home tehnologies, Elsevier, 2017. [30. lipnja 2021.]

[14] N. Balta-Ozkan, B. Boteler, O. Amerighi, European smart home market development: Public views on tehnical and economic aspects across the United Kingdom, Germany and Italy, Elsevier, 2014. [30. lipnja 2021.]

[15] A. Paetz, E. Dutschke, W. Fichtner, Smart Homes as a Means to Sustainable Energy Consumption: A Study of Consumer Perceptions, Elsevier, 2011. [30. lipnja 2021.]

[16] E.I. Davies, Vincent Ike Anireh, Design and Implementation of Smart Home System Using Internet of Things, Digital Innovations, ožujak, 2019. [20. kolovoza 2021.]

[17] A. Ben, H. Mohamed, T. Val, L. Andrieux, A. Kachouri, Assisting people with disabilities through Kinect sensors into a smart house, siječanj 2013. [20. kolovoza 2021.]

[18] How Blynk Works, Dostupno na: https://docs.blynk.cc [24. lipnja 2021.]

[19] Blynk Board Project Guide, Dostupno na: https://learn.sparkfun.com/tutorials/blynk-board-project-guide/project-3-slide-dimming-leds [24. lipnja 2021.]

[20] M. Jon, Offline voice assistants - Can modern smart homes also be private?, svibanj 2021. Dostupno na: https://medium.com/product-ai/offline-voice-assistants-can-modern-smart-homes-be-private-too-36a8be3c7ea9 [24. lipnja 2021.]

[21] C. Beckel, H. Serfas, E. Zeeb, G. Moritz, F. Golatowski, D. Timmermann, Requirements for Smart Home Applications and Realization with WS4D-Pipes Box, Međunarodna konferencija o novoj tehnologiji i tvorničkim automatizacijama, Njemačka, listopad 2011. [20. kolovoza 2021.]

[22] E. D. Liddy, Natural Language Processing, Enciklopedija knjižničarske i informacijske znanosti, 2001. [20. kolovoza 2021.]

[23] D. Banerjee, Natural Language Processing (NLP) Simplified: A Step-by-step Guide, travanj 2020. Dostupno na: https://datascience.foundation/sciencewhitepaper/natural-language-processing-nlp-simplified-a-step-by-step-guide [20. kolovoza 2021.]

[24] R. Wolff, What Is Natural Language Processing, veljača 2020. Dostupno na: https://monkeylearn.com/blog/what-is-natural-language-processing/ [20. kolovoza 2021.]

[25]Fragments, Android developers. Dostupno na: https://developer.android.com/guide/fragments [20. kolovoza 2021.]

[26] Rohit, RecyclerView Android example with CardView in Kotlin, lipanj 2018. Dostupno na: https://tutorial.eyehunts.com/android/recyclerview-android-example-cardview-kotlin/ [20. kolovoza 2021.]

[27] N. Jayaweera, B. Gamaga, M. Samaraweera, S. Liyanage, S. Lokuliyana, T. Kuruppu, Gesture Driven Smart Home Solution for Bedridden People, Međunarodna konferencija o radionicama automatiziranog softverskog inženjeringa, 2020. [20. kolovoza 2021.]

[28] S. Guillet, B. Bouchard, A. Bouzouane, Designing smart homes dedicated to disabled people using modular Discrete Controller Synthesis, Radionica o diskretnim sustavima događaja, Francuska, svibanj 2014. [20. kolovoza 2021.]

[29] G. A. Araujo de Oliveira, R. Winickler de Bettio, A. P. Freire, Accessibility of the smart home for users with visual disabilities: an evaluation of open source, Brazil, listopad 2016. [20. kolovoza 2021.]

[30] A. Tulenkov, Y. Yaremchenko, A. Parkhomenko, Y. Zalyubovskiy, A. Parkhomenko, M.Kalinina, Adaptation of Smart House System for People with Special Needs Based on Wireless Technologies, Međunarodni simpozij o pametnim i bežičnim sustavima u okviru Međunarodne konferencije o inteligentnom prikupljanju podataka i naprednim računalnim sustavima, Njemačka, rujan 2020. [20. kolovoza 2021.]

[31] S. Guillet, B. Bouchard, A. Bouzouane, Correct by construction security approach to design fault tolerant smart homes for disabled people, Četvrta međunarodna konferencija o sveprisutnim sustavima u nastajanju i sveprisutnim mrežama, Elsevier, 2013. [20. kolovoza 2021.]

[32] Y. Zhong, Y. Suo, W. Xu, C. Yu, X. Guo, Y. Zhao, Y. Shi, Smart Home on Smart Phone, Kina, listopad 2011. [20. kolovoza 2021.]

[33] B. Kon, A. Lam, J. Chan, Evolution of Smart Homes for the Elderly, Zbornik radova s 26. međunarodne konferencije o pratiocu World Wide Weba, 2017. [20. kolovoza 2021.]

SAŽETAK

Ovaj završni rad bavi se razvojem mobilne Android aplikacije koja omogućuje multimodalno upravljanje pametnom kućom, a aplikacija je u prvom redu namijenjena osobama s invaliditetom. Opisani su problemi osoba s invaliditetom pri upravljanju pametnom kućom s naglaskom na probleme kretanja i slabovidnosti. Na temelju sposobnosti osobe, analizirani su oblici upravljanja na više načina te je modeliran sustav. Cilj aplikacije je olakšati upravljanje pametnom kućom za razne tipove invaliditeta. Predloženi sustav pametne kuće ima mikroupravljač koji može komunicirati s aplikacijom. Aplikacija omogućuje pretvorbu govora u tekst i teksta u govor kako bi ispunila uvjete multimodalnosti, a omogućuje upravljanje vratima i rasvjetom. Ispitivanje sustava pokazalo je ispunjavanje postavljenih zahtjeva funkcionalnosti.

Ključne riječi: Android, invaliditet, mikroupravljač, mobilna aplikacija, pametna kuća.

ABSTRACT

This final paper deals with the development of a mobile Android application that enables multimodal smart home management, and the application is primarily intended for people with disabilities. The problems of people with disabilities in smart home management are described with an emphasis on mobility problems and vision impairment. Based on the person's abilities, the forms of management in multiple ways were analyzed and the system was modeled. The aim of the application is to make smart home management easier for various types of disabilities. The proposed smart home system has a microcontroller that can communicate with the application. The application allows the conversion of speech into text and text into speech to meet the requirements of multimodality, and allows the management of doors and lighting. Testing of the system showed that the set functionality requirements were met.

Keywords: Android, disability, microcontroller, mobile application, smart home.

ŽIVOTOPIS

Andreja Nađ rođena je 19. lipnja 1999. godine u Osijeku, Hrvatska. Pohađala je Elektrotehničku i prometnu školu Osijek, gdje je uspješno završila smjer za tehničara za mehatroniku. U 2016. godini završila je stručno osposobljavanje u Šolskom centru Škofja Loka, u Sloveniji, za elektropneumatiku i električno upravljanje te je iz iste teme sljedeće godine osvojila deveto mjesto na državnom natjecanju. Isto tako, 2016. godine sudjeluje u Robotics Camp 2016. U 2018. godini je odradila praksu sa Erasmus+ projekom ponovno u Šolskom centru Škofja Loka. Iste te godine je sudjelovala na Erasmus+ projektu Wonderland gdje je u suradnji sa ostalim zemljama partnerima poboljšala svoje komunikacijske vještine i engleski jezik. Nakon završetka srednje škole, upisuje Preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija. Vješta je na području engleskog jezika, odgovorna i uporna.

_____

Potpis autora

PRILOZI (na DVD-u)

Prilog 1: Završni rad u formatu docx i pdf.

Prilog 2: Projekt mobilne aplikacije u Android Studiju

Prilog 3: Arduino studio kod za sklop ESP8266 (Nova2)