SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

FERIoT pametni sustav za upravljanje pametnim uređajima u blizini

Marko Bartaković, Karlo Ježić, Ivan Sarjanović, Josipa Šupe, Petra Završki, Leon Banko, Pero Drobac, Karlo Ivanov

Zagreb, lipanj 2022.

Ovaj rad izrađen je na Zavodu za telekomunikacije Fakulteta za elektrotehniku i računarstvo u Zagrebu pod vodstvom prof. dr. sc. Ivane Podnar Žarko i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022

Uvod

U današnje vrijeme tehnologija je jedan od neizostavnih faktora svakodnevnog života. Zbog sve veće potrebe za korištenjem pametnih telefona i uređaja, javlja se i veća potražnja za uslugama temeljenim na korisničkoj lokaciji. Sustavi za određivanje vanjske lokacije odnosno vanjsko pozicioniranje uglavnom koriste satelitske (npr. GPS) ili mrežne tehnologije (određivanje lokacije na temelju snage signala baznih stanica davatelja usluga mobilne telefonije, ili pristupnih točaka Wi-Fi). Upravo zbog činjenice da uvelike doprinosi kvaliteti brojnih ljudskih djelatnosti, GPS komponenta postala je neizostavan dio za gotovo sve pametne uređaje koji se proizvode i koriste u današnje vrijeme. Međutim, s obzirom da GPS i druge satelitske tehnologije u slučajevima lociranja u unutrašnjem prostoru nemaju dovoljnu preciznost ili u potpunosti ne zadovoljavaju korisničke zahtjeve, javlja se potreba za korištenjem drugačijih rješenja i tehnologija. Bolje rečeno, javlja se potreba za razvoj sustava za unutarnje pozicioniranje. Prema definiciji, sustav unutarnjeg pozicioniranja (engl. *Indoor Positioning System*, IPS) je sustav mrežno povezanih uređaja koji se koristi za bežično lociranje predmeta i osoba unutar zgrada i djelomično natkrivenih područja [5]. Budući da primjena tehnologija za pozicioniranje u zatvorenom uglavnom zahtjeva dodatnu infrastrukturu, izvedba je najčešće složenija od pozicioniranja na otvorenom. Primjena samih IPS sustava vrlo je široka te uključuje traženje predmeta ili grupe predmeta u dućanima, pronalaženje odgovarajućeg ureda u zgradama, orijentaciju unutar bolničkog kompleksa i pronalaženje određenih trgovina u trgovačkom centru, te mnoge druge.

Jednako tako, sve je brže rastuća i popularnost pametnih uređaja te se njihova primjena proteže kroz domaćinstva, fakultete, razne poslovne prostore te brojne druge. Kako bi uređaji mogli svojim funkcionalnostima pomoći ljudima da što manje vremena i brige provode na upravljanje prostorom, ljudi same uređaje moraju moći brzo i jednostavno koristiti. Za korištenje pametnih uređaja postoje brojne IoT platforme koje to omogućavaju, a jedna od njih je i Home Assistant.

Jedna od najčešćih slučajeva korištenja unutarnjeg pozicioniranja jest upravo pri implementaciji IoT aplikacija, a u okviru ovog rada opisano je rješenje pri kojemu aplikacija na temelju korisničke lokacije, ovisno o njegovim pravima, omogućava komunikaciju s pametnim uređajima koji se nalaze u njegovoj blizini uz pomoć platforme Home Assistant.

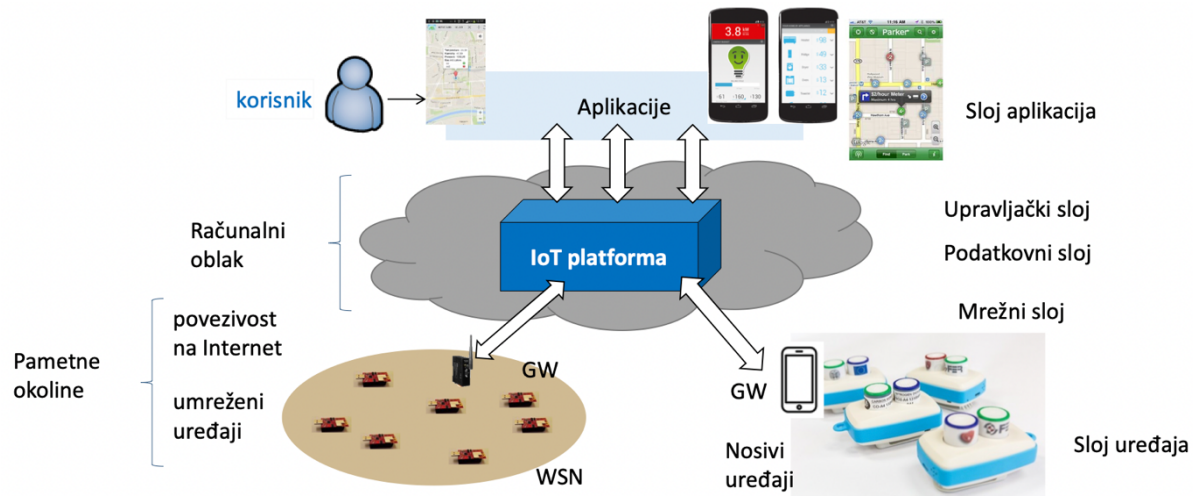
1. Internet stvari

Internet stvari[12] (engl. *Internet of Things*, IoT) je globalna infrastruktura za informacijsko društvo koja omogućuje usluge međusobnim (fizikalnim i virtualnim) „Stvarima“ baziranim na postojećim uslužnim i komunikacijskim tehnologijama. Internet stvari je sve više prisutan u svakidašnjem životima svakog čovjeka. Ubrzani svakodnevni život pojedinca zahtjeva što manje vremena provedeno na stvari o kojima ne bi nužno on trebao brinuti. S druge strane sigurnost i brzina koje Internet stvari donosi poboljšavaju život svakome tko se s njime susretne. Neka od područja primjene Interneta stvari:

- Pametni dom jedan je od primjera Interneta stvari u sklopu kojega možemo imati razne povezane uređaje. Pametna ulazna vrata, električne rolete, lampe i žarulje, detektore kvalitete zraka ili dima, upravljanje i praćenje energije.
- Pametni grad, u sklopu kojega možemo imati pametni parking, upravljanje i praćenje otpada, pametna rasvjeta i hitne intervencije.
- Praćenje okoliša: praćenje kvalitete zraka i zemlje, glasnoće ili senzori za prevenciju poplava i požara
- U prodaji, Internet stvari često ima ulogu praćenja skladišta ili provođenje bez kontaktnih plaćanja
- U logistici koristi se za praćenje pošiljaka, daljinskih vozila ili uređaja te za stvaranje ruta i planiranje vremena
- U industriji koristi se za dijagnozu mašinerije i praćenje proizvoda
- U agrikulturi za praćenje stanja određenih biljaka i njihovih potreba

1.1. Arhitektura Interneta stvari

Na slici 1 vidimo pojednostavljeni prikaz arhitekture Interneta stvari. U donjem lijevom kutu slike prikazana je pametna okolina koja se sastoji od uređaja i senzora koji generiraju i koriste podatke. Podatci se šalju računalnom oblaku, na slici prikazano kao IoT platforma, koja služi za njihovo procesiranje, spremanje i prosljeđivanje. Na kraju, u gornjem dijelu slike prikazan je sloj aplikacija koji generira zahtjeve za određenim podatcima, prikazuje ih korisniku i šalje naredbe određenim uređajima koji su povezani u sustavu.



Slika 1: Arhitektura Interneta stvari

2. Opis razvijenog rješenja

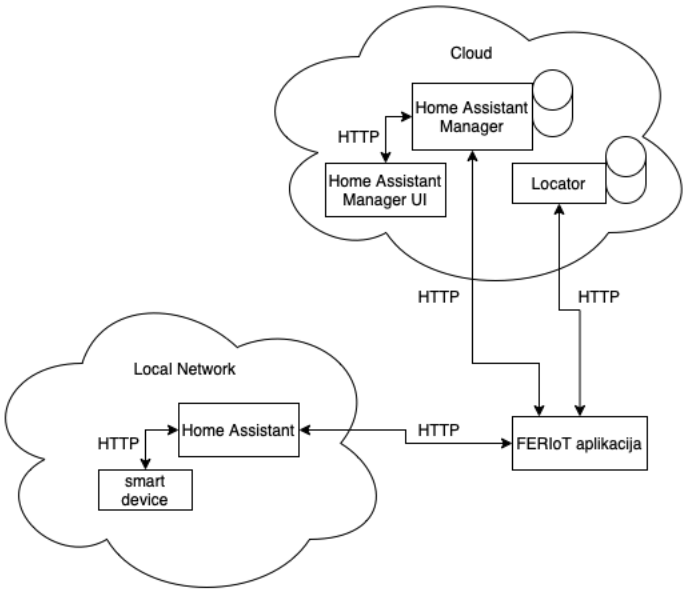
U sklopu ovog rada napravljen je i implementiran sustav za upravljanje i praćenje uređaja u okruženju Interneta stvari za primjenu pametnog doma, ureda, zgrade ili bilo kojeg većeg kompleksa. Nad sustavom je napravljen studijski primjer točnije sustav je testiran u prostoriji Fakulteta za elektrotehniku i računarstvo.

2.1. Arhitektura sustava

Arhitekturu sustava moguće je vidjeti na slici 2. Arhitektura se sastoji od tri glavna dijela:

- Računalni oblak
- Aplikacija
- IoT platforma

Na računalnom oblaku pokrenute su dvije aplikacije: aplikacija Locator i aplikacija Home Assistant Manager. Svaka od aplikacija ima svoju vlastitu bazu podataka. Softver koji ima ulogu IoT platofrme je usluga Home Assistant. Aplikacija kreirana u sklopu ovog rada koristi se IoT platformom preko kojega upravlja IoT uređajima te koristi računalni oblak za lociranje i autorizaciju. Više o implementaciji zasebnih usluga u sljedećim poglavljima.



Slika 2: Prikaz arhitekture FERIoT sustava

2.2. Aplikacija Locator

Aplikacija Locator je glavna točka orijentacije korisnika u prostoru. Aplikacija se temelji na tehnologiji Bluetooth te na nisko-energetskim Bluetooth (Bluetooth Low Energy, BLE) odašiljačima. BLE odašiljači imaju funkciju referentnih točaka u prostoru pomoću kojih aplikacija zna pozicionirati korisnike temeljem očitanja odašiljača u blizini korisnikovog uređaja.

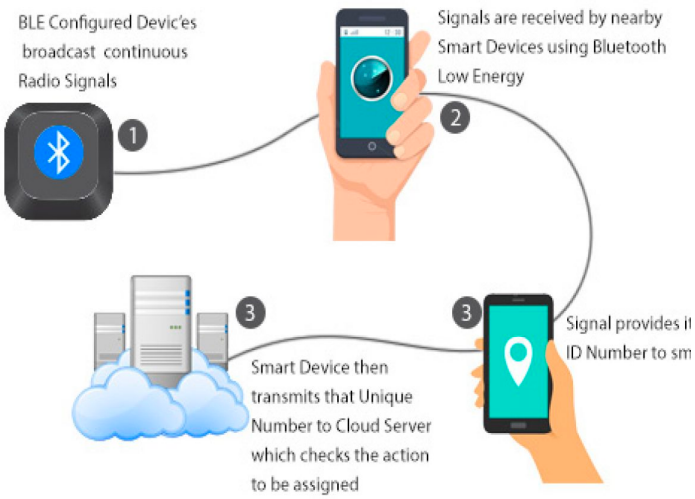
Princip rada funkcije pozicioniranja mobilnih uređaja u prostoru je sličan standardnom načinu rada GPS-a. Za ispravan rad sustav zahtjeva barem tri očitana odašiljača, a s većim brojem očitanih odašiljača raste i preciznost pozicioniranja.

2.2.1. Bluetooth Low Energy

Bluetooth tehnologija standard je bežične komunikacije koji se koristi za razmjenu podataka na relativno maloj udaljenosti [5]. Razvio ga je Ericsson 1994. godine, a 1998. godine Ericsson, IBM, Intel, Nokia i Toshiba osnovali su posebno nadležno tijelo, "Bluetooth Special Interest Group" (SIG). Uloga ovog tijela je poboljšati standarde, pravilnu provedbu i licenciranje Bluetooth tehnologije.

Glavne značajke Bluetooth tehnologije su jeftini Bluetooth uređaji, mala potrošnja energije, mali domet, robusnost i globalna upotreba. Bluetooth omogućuje brzinu prijenosa od 1 Mbit / s i koristi nelicencirani frekvencijski opseg od 2,4 do 2,485 GHz, odnosno koristi ISM (industrijsko, znanstveno i medicinsko) područje gdje je frekvencija globalno usklađena [5]. Također, Bluetooth nudi radijsku vezu s drugim sustavima. Sredinom 2010. godine Bluetooth SIG najavio je specifikaciju Bluetooth 4.0 koja je uključivala klasični Bluetooth, Bluetooth velike brzine i Bluetooth nisko-energetske protokole. Nisko-energetski Bluetooth (engl. *Bluetooth Low Energy*, BLE) karakteriziraju male dimenzije, niska cijena, mala potrošnja energije s mogućnošću nekoliko godina rada na malom izvoru energije (AAA baterije) i kompatibilnost s mobilnim uređajima, tabletima i računalima. Da bi se BLE tehnologija instalirala na uređaje, Bluetooth 4.0 uvodi dva načina rada: jednodimenzionalni i dvodimenzionalni [5]. Jednodimenzionalni rad uključuje samo integraciju BLE funkcionalnosti u kontroler, dok dvodimenzionalni rad omogućuje integraciju BLE i Bluetooth funkcionalnosti u standardni kontroler. Proizvođači uređaja imaju na raspolaganju obje mogućnosti te je bitno naglasiti da uređaji s jednodimenzionalnim radom ne mogu komunicirati s uređajima koji koriste klasični Bluetooth protokol. Danas se većina mobilnih uređaja

proizvodi s podrškom za standardni Bluetooth i BLE, odnosno u uređaje je instaliran Bluetooth mikrokontroler s dvodimenzionalnim načinom rada. BLE emitira signale s odašiljača koji rade na baterije. Koristi takozvani odašiljač (engl. *beacon*) i uređaje za lociranje koji su jeftini, male veličine, dugog trajanja baterije i ne zahtijevaju vanjski izvor napajanja. Uređaj detektira signal s odašiljača i može približno izračunati udaljenost do njega i na taj način procijeniti mjesto uređaja. Koristeći BLE tehnologiju, odašiljači obavještavaju obližnje uređaje o svojoj prisutnosti. Uređaji u blizini (mobilni uređaji, tableti) mogu se pretplatiti na obavijesti predajnika i mogu primati različit sadržaj (poput teksta, slika i URL-a). Istodobno, odašiljači se mogu koristiti za beskontaktna plaćanja na sličan način kao NFC. Životni vijek odašiljača procjenjuje se na 5-8 godina, a odašiljači su otporni na prašinu i vodu te im se točnost procjenjuje u rasponu od jednog do tri metra.



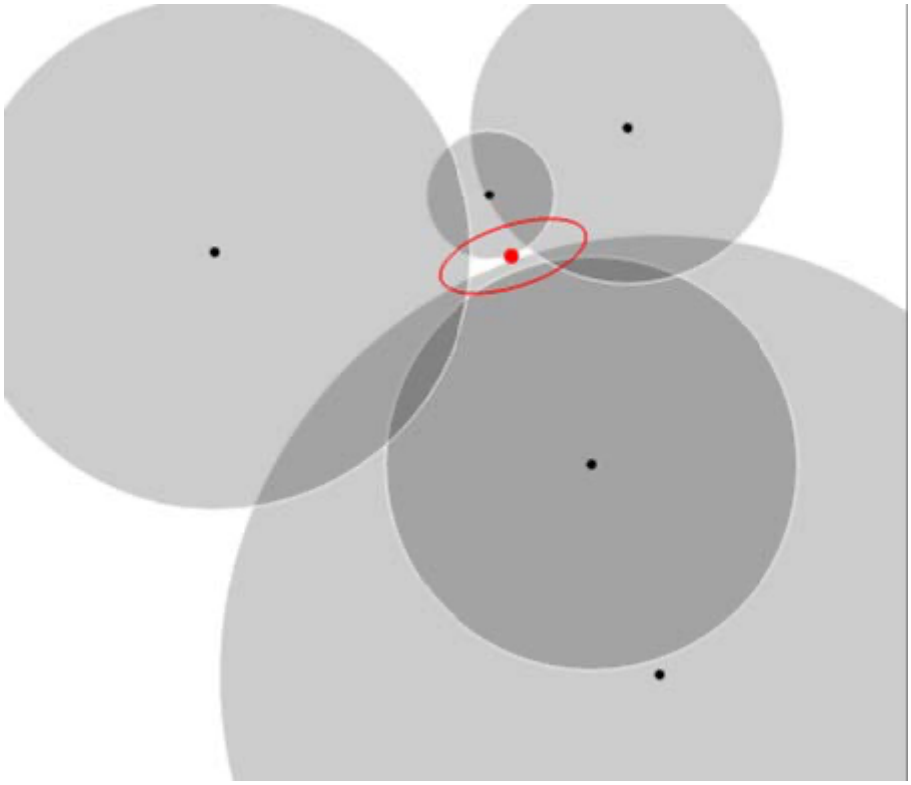
Slika 3 Prikaz BLE

2.2.2. Bluetooth odašiljači

Bluetooth odašiljači mali su bežični odašiljači koji koriste nisko-energetsku Bluetooth tehnologiju za slanje signala na druge pametne uređaje u blizini [6]. Oni su jedno od najnovijih dostignuća u tehnologiji lociranja. Princip rada ovih odašiljača vrlo je jednostavan: svaki uređaj sadrži procesor, radio i baterije te mu se funkcionalnost temelji na tome da više puta emitira jedinstveni identifikator. Ovaj identifikator bilježi korisnički uređaj, u većini slučajeva mobitel, te označava mjesto koje se nalazi u korisničkom okruženju. Identifikator je jedinstveni ID broj koji pametni telefon prepoznaje kao jedinstveni za odašiljač. Jednom kada se spoji, odašiljač može izvršavati bilo koju funkciju za koju je programiran.

2.2.3. Metoda izračunavanja lokacije

Brojne metode u svojim primjerima i dokazima koriste točno tri točke za određivanje lokacije, no u realnim uvjetima će uređaj očitavati više ili manje točaka. Ovakva situacija se rješava pomoću algoritama predviđenih za višebrojna očitanja kao u našem slučaju [6], a jedna od takvih je nelinearna metoda najmanjih kvadrata (engl. *non-linear least squares method*) s Levenberg-Marquardtovim algoritmom. Takav algoritam nam kao rješenje nudi centroid, odnosno mi koristimo njegovu središnju točku kao lokaciju korisnikovog uređaja, kao i samog korisnika. Slika 3 prikazuje primjer multilateracije.



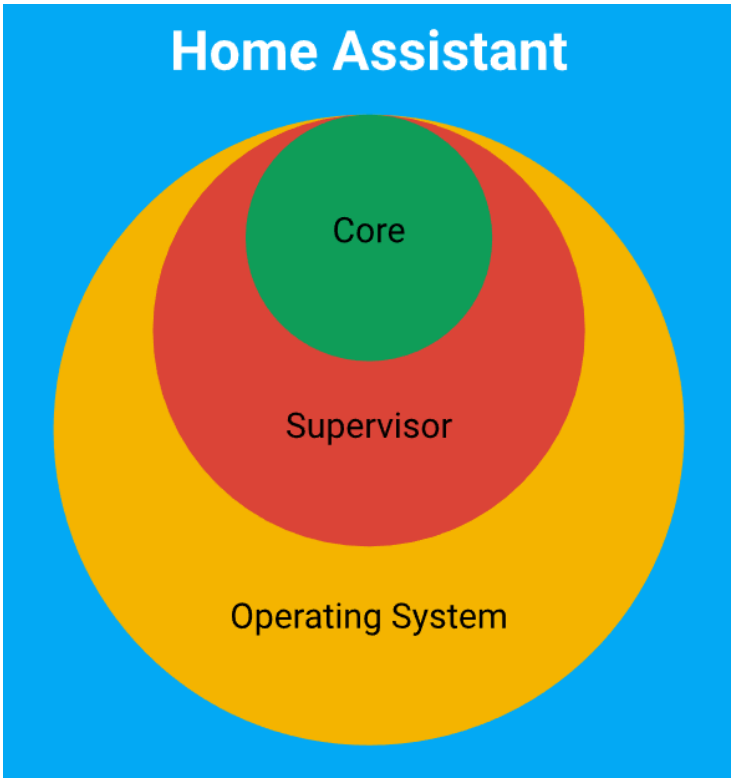
Slika 4: Primjer multilateracije kod lociranja

2.3. Integracija pametnih uređaja

Integracija pametnih uređaja u sustav ostvarena je uz pomoć platforme Home Assistant koji je alat namijenjen za upravljanjem pametnim uređajima unutar iste lokalne mreže. Zbog prirode zahtjeva na sustav, unutar platforme Home Assistant implementirano je više vrsta korisnika s različitim pravima pristupa. Spomenuta autorizacija korisnika platforme Home Assistant ostvarena je uz pomoć alata Node-RED. Od pametnih uređaja u sustav su integrirane Phillips Hue lampe te je implementirana automatizirana detekcija prekoračenja maksimalnog dozvoljenog broja ljudi u prostoriji.

2.3.1. Home Assistant

Home Assistant je alat otvorenog izvornog koda koji ima ulogu platforme u pametnom okruženju te omogućava upravljanje pametnim uređajima. Instalacija Home Assistant može se ostvariti unutar Docker kontejnera, na uređajima poput Raspberry Pi-a, ODROID-a, ASUS Tinkerboard-a, Generic x86-64 (Intel NUC), ali i na Windows, macOS te Linux platformama. Također, postoje i alternativne varijante instalacije. Pristup samom Home Assistant-u moguće je ostvariti putem web korisničkog sučelja, prikladnih iOS ili Android mobilnih aplikacija, ali i putem nekog virtualnog asistenta koji podržava glasovne naredbe, primjerice koristeći Google Assistant ili Amazon Alexu. Također, pruža i mnoge integracije s postojećim rješenjima za pametne uređaje, ali i omogućava ručnu integraciju s bilo kojom napravom koristeći web-hookove, krajnje točke kontakta koje primaju HTTP zahtjeve [2].



Slika 5 Prikaz arhitekture Home Assistant-a

Arhitektura Home Assistant-a sastoji od 3 dijela prikazanih na slici 5 :

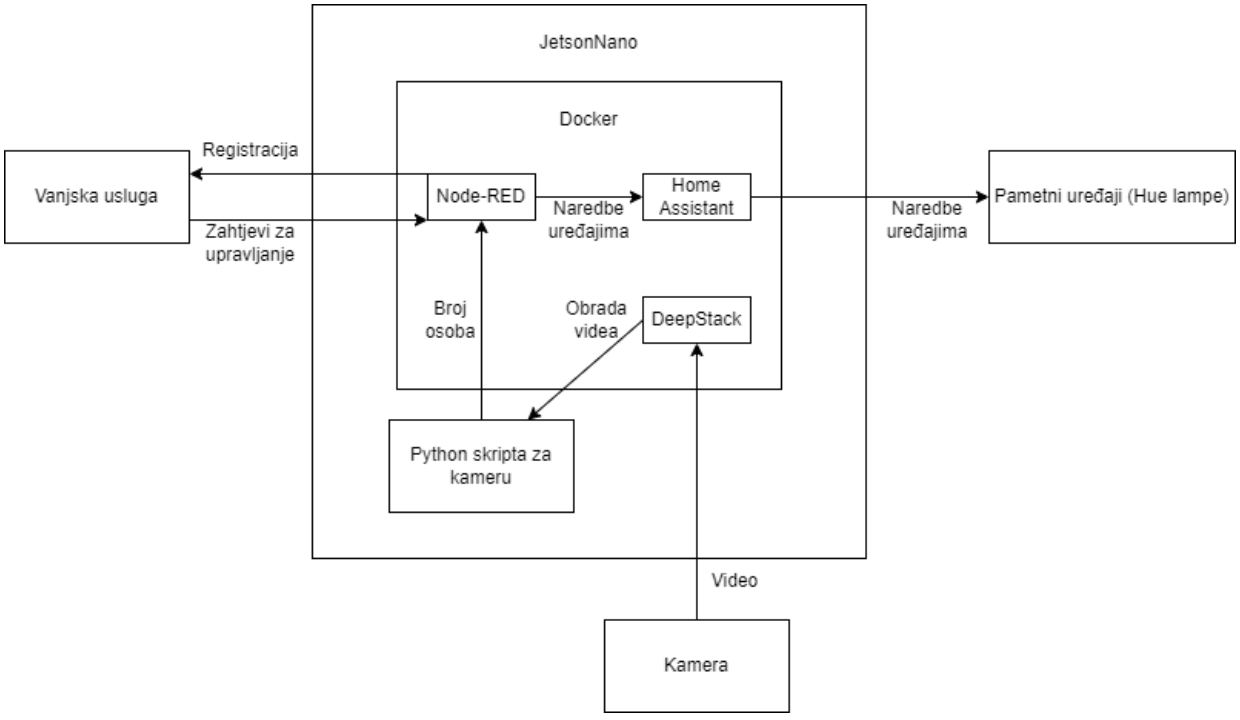
1. Operacijski sustav (engl. Operating System) koji pruža Linux okruženje potrebno za izvođenje Nadzornika i Jezgre. Preciznije, operacijski sustav Home Assistant-a koristi sustav ugradnje Buildroot koji nije klasična distribucija Linux sustava, već samo daje potporu za izgradnju Linux sustava. Zbog toga, moguće je pokretanje Home Assistant-a i na sustavima s manje resursa, kao što su sustavi temeljeni na ARM-u.

2. Nadzornik (engl. Supervisor) služi za upravljanje operacijskim sustavom te na taj način omogućuje i krajnjem korisniku upravljanje samim Home Assistant-om.
3. Jezgra (engl. Core) komunicira s Nadzornikom, korisnikom te IoT uređajima i uslugama. [11]

2.3.2. Node-RED

Node-RED je programski alat za spajanje hardverskih uređaja, aplikacijsko programskih sučelja i internetskih usluga. Alat nudi grafičko sučelje putem kojeg je moguće ostvariti automatizacije na nekompleksan način. Također, pruža potporu za integraciju s Home Assistant-om s kojim se zbog svoje jednostavnosti učestalo koristi [3].

2.3.3. Arhitektura i dizajn implementiranog rješenja



Slika 6 Prikaz arhitekture rješenja za integraciju s pametnim uređajima

Na slici 6 vidimo prikaz arhitekture implementiranog rješenja. Dakle, na računalu Nvidia JetsonNano[13] instaliran je alat Docker[14] u kojem su pokrenuti kontejneri za Home Assistant, Node-RED i DeepStack. Sustav omogućuje rad s Phillips Hue lampama, ali i očitanje trenutnog broja ljudi u prostoriji. Lampe su integrirane u sustav koristeći svoju nativnu

integraciju s Home Assistant-om, dok se komunikacija s Python skriptom za očitanje broja ljudi u prostoriji vrši putem HTTP zahtjeva. Više o samom brojaču prisutnosti, dostupno je u sljedećem poglavlju. Implementirano rješenje dizajnirano je kako bi moglo komunicirati s vanjskom uslugom. Vanjska usluga može biti bilo kakav sustav koji poštuje definirane krajnje točke, a u okviru ovog rada kao vanjska usluga korišteni su Home Assistant Manager te Android mobilna aplikacija. Autorizacija korisnika u Home Assistant-u ostvarena je uz pomoć alata Node-RED jer podržava programski jezik JavaScript kojim je moguće ostvariti enkripciju potrebnih tokena. Tokeni se generiraju prilikom registracije sustava na vanjskoj usluzi. Dodjeljuje se po jedan token svakoj vrsti korisnika ukoliko je unutar sustava definirano postojanje više vrsta korisnika. U ovom sustavu postoje dvije vrste korisnika, a to su profesor i student.

Sustav definira krajnje točke za dohvat popisa uređaja, dohvat stanja uređaja te promjenu stanja na uređajima. Prilikom pristizanja zahtjeva u sustav, vrši se autorizacija, tj. provjerava se je li korisnik ovlašten za korištenje željene usluge. U okviru ovog sustava, profesorima su sve mogućnosti sustava dozvoljene. Dakle, mogu paliti, gasiti lampe, ali i mijenjati boje. S druge strane, studenti imaju manje mogućnosti te, za razliku od profesora, nemaju ovlasti za mijenjanje boja na lampama. Osim toga, implementirana je i automatska detekcija prevelikog broja ljudi u prostoriji. Za svaku prostoriju moguće je odrediti maksimalan broj ljudi koji smije u istom trenutku boraviti u njoj. Međutim, ukoliko se definirani maksimum prijeđe, sustav automatizacijom pali pametne lampe i postavlja boju na crvenu [4].

2.3.4. Brojač prisutnih osoba u prostoriji

Brojanje osoba u prostoriji realizirano je pomoću python skripte i DeepStack poslužitelja [1]. DeepStack je poslužitelj otvorenog koda koji omogućuje upotrebu već istreniranih modela strojnog učenja za prepoznavanje različitih objekata sa slikovnih okvira. DeepStack nije ovisan o vrsti uređaja i operacijskom sustavu. Zbog toga je pogodan za korištenje na raznim uređajima na rubu mreže. Kamera koja je spojena na Jetson nano konstantno generira slikovne okvire koji se preko python skripte šalju DeepStack poslužitelju koji zatim pomoću istreniranog modela strojnog učenja na slici prepoznaje osobe. Na slikovnom okviru, Deepstack zatim iscrtava obrise oko osoba koje je prepoznao. Nakon toga u python skripti sumiramo sve obrise na slikovnom okviru i zatim ažuriramo vrijednost varijable u kojoj čuvamo informaciju o trenutnom broju osoba u prostoriji.

2.4. Aplikacija Home Assistant Manager

Aplikacija Home Assistant Manager centralna je aplikacija čija je zadaća upravljanje korisnicima u sustavu i na IoT platformi. Uz nju dolazi i korisničko sučelje za unošenje novih platformi Home Assistant te pregledavanje i uređivanje postojećih i registraciju, odobravanje, brisanje i pregledavanje novih korisnika.

2.4.1. Tehničke značajke

Home Assistant Manager aplikacija napisana je u programskom jeziku Java uz korištenje Spring Boot radnog okvira. Za komunikaciju s bazom podataka korištena je knjižnica Java Persistance API (JPA). Baza podataka korištena za spremanje podataka Home Assistant Manager aplikacije je relacijska baza PostgreSQL[15].

Korisničko sučelje napravljeno je koristeći programski jezik JavaScript i knjižnica React.

2.4.2. Programsko aplikacijsko sučelje (API)

Programsko aplikacijsko sučelje sastoji se od šest krajnjih točaka čije se metode i kratki opis mogu vidjeti na tablici 1.

Krajnja točka /authenticate koristi se za autorizaciju korisnika.

Krajnja točka /home-assistants koristi se za registraciju Home Assistant platformi u sustav. Kako bi se određeni Home Assistant uspješno registrirao u sustav potrebno je poslati POST zahtjev s tijelom poruke u JSON formatu s parametrima:

- token = token korisničkog profila platforme
- room = soba u kojoj se platforma nalazi
- baseUrl = virtualna (mrežna) lokacija platforme, točnije lokacija aplikacijsko programskog sučelja platforme
- role = uloga za koju je registracija obavljena

Platforma se registara na način da se napravi korisnički profil kojemu se dodijele određena prava i generira token. Dakle jedna platforma može imati više ulaza u samom sustavu, za svakog korisnika jedan. Razlog ovog dizajna je nedostatak ograničenja prava na samoj platformi Home Assistant gdje je za sada jedino omogućeno napraviti grupu kojoj su pridijeljena određena prava a onda se određeni korisnički profil dodjeljuje jednoj od grupa. Svaki korisnički profil ima svoj token te su tom tokenu dodijeljena prava grupe kojoj je korisnik predijeljen.Krajnja točka /home-assistants također ima i GET metodu koja dohvaća sve registrirane platforme. /home-assistants krajnja točka korištena je iz mobilne aplikacije dok su

mogućnosti krajnje točke /ha, koja ima slične funkcionalnosti, korištene unutar korisničkog sučelja.

Krajnja točka /ha sastoji se od tri metode: GET, POST i DELETE. GET metoda koristi se za dohvaćanje platformi i njihovih podataka te je korištena za prikazivanje i uređivanje podataka o platformama koristeći korisničko sučelje. POST metoda koristi se za registriranje novih platformi u sustav iz korisničke aplikacije. Metoda DELETE služi za brisanje platformi iz sustava.

Za dohvaćanje podataka o trenutnom korisniku koristi se krajnja točka /user i metoda GET. O kojem se točno korisniku radi saznaje se iz sjedničkog kolačića

Za pregled, dodavanje i brisanje korisničkih računa sustava koristi se krajnja točka /users i njene metode: GET, POST i DELETE

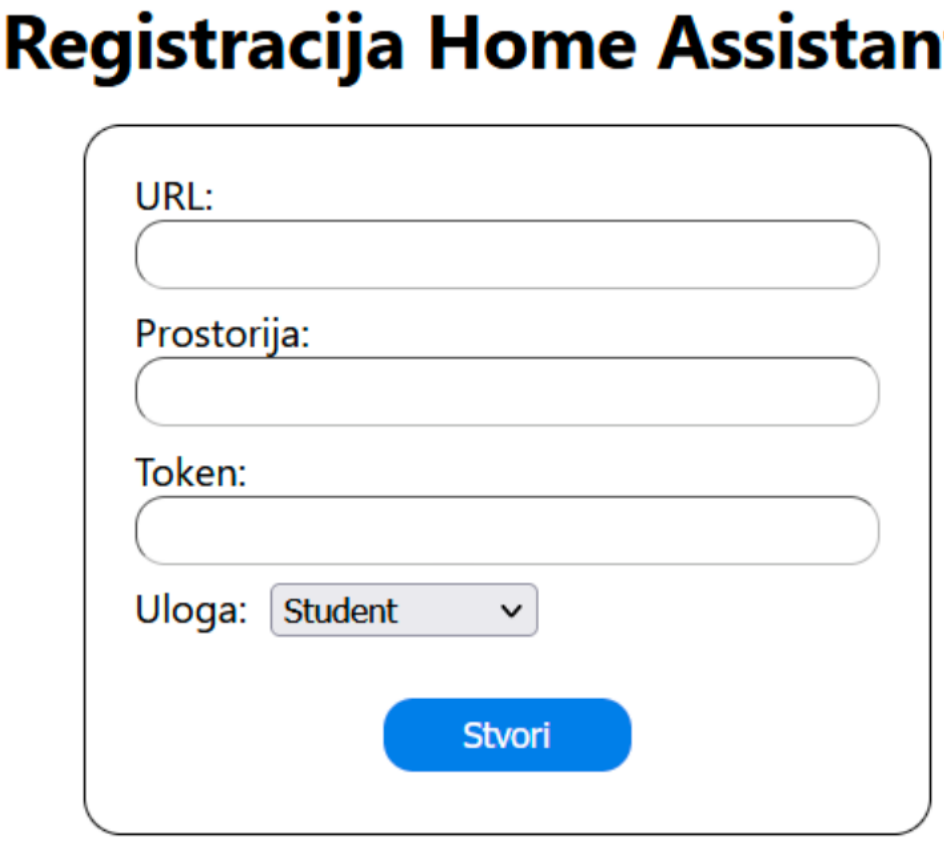
Tablica 1: Krajnje točke usluge Home Assistant Manager

KRAJNJA TOČKA	METODA	KRATKI OPIS
/home-assistants	POST	Registracija platforme za s tokenom za određena prava (mobilna aplikacija)
/home-assistants	GET	Dohvaćanje podataka o platformama i njenim tokenima i rolama (mobilna aplikacija)
/user	GET	Dohvaćanje podataka o vlastitom korisničkom računu
/users	GET	Dohvaćanje podataka o svim korisnicima registriranim u sustavu
/users	POST	Registriranje korisnika
/users	DELETE	Brisanje korisnika iz sustava

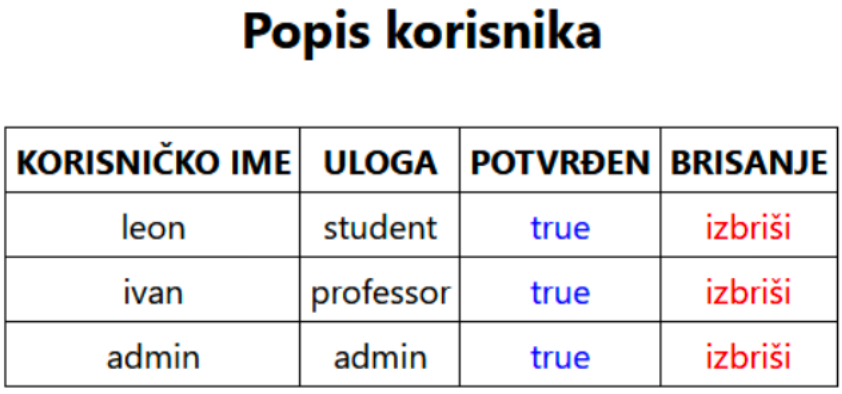
/users/confirm	POST	Brisanje korisničkog računa iz sustava
/ha	GET	Vraća popis platformi i njihove podatke (korisničko sučelje)
/ha	POST	Registrira platformu u sustavu (korisničko sučelje)
/ha	DELETE	Brisanje platformi iz sustava (korisničko sučelje)

2.4.3. Korisničko sučelje

Korisničko sučelje aplikacije Home Assistant Manger namijenjeno je za lakše održavanje podataka u sustavu. Korisnička aplikacija ima dvije svrhe: pregled podataka u sustavu, dodavanje novih podataka u sustav i odobravanje novih korisnika. Odobravanje novih korisnika i brisanje podataka iz sustava omogućeno je jedino korisničkom računu s admin pravima dok su sve ostale funkcionalnosti dozvoljene svima. Aplikacija od korisnika prije korištenja korisničkog sučelja zahtjeva prijavu na temelju koje zaključuje o kojem korisniku se radi i koja su prava dodijeljena tom korisniku. Komunikacija i autentifikacija se odvija preko sjedničkog kolačića koji se dobiva prilikom prijave u korisničko sučelje. Na slici 4 moguće je vidjeti primjer obrasca za registraciju korisnika dok je izgled obrasca za registraciju platforme Home Assistant preko korisničkog sučelja moguće je vidjeti na slici 5. Slika 6 prikazuje tablicu koja prikazuje korisnike te njihovo stanje u sustavu.



Slika 7 Prikaz registracije



2.4.4. Baza podataka

Baza podataka korištena u sklopu usluge Home Assistant Manager je relacijska baza PostgreSQL. Usluga u bazi podataka održava tri tablice:

- Feriot_user = tablica koja sadrži podatke o korisnicima sustava (tablica 2)

- Home_assistant = tablica za održavanje podataka o platformama Home Assistant (tablica 3)
- Role = tablica za održavanje podataka o ulogama i tokenima vezanim za tu rolu (tablica 4)

Tablica home_assistants napravljena je na način da odgovara rješenju uloga unutar platforme Home Assistant. Kako na platformi Home Assistant ne postoji standardni način definiranja uloga za potrebe ovog rada napravljene su grupe s ograničenjima te korisnički profil s vjerodajnicama. Svaki korisnički profil pridružen je određenoj grupi pa zato za svaku od platformi imamo više od jednog zapisa u tablici, za svaki korisnički profil platforme.

Tablica 2: Prikaz tablice feriot_user

atribut	**tip**	**opis**
id	broj	Identifikator korisnika
password	tekst	Lozinka korisnika
username	tekst	Korisničko ima korisnika
role	broj	Referenca na redak tablice role

Tablica 3: Prikaz tablice home_assistant

atribut	**tip**	**opis**
id	broj	Identifikator zapisa u tablici
base_url	tekst	Internetska lokacija platforme
token	tekst	Token kojemu su dodijeljena prava za upravljanje uređajima
room	tekst	Prostorija u kojoj se platforma nalazi

role_id	broj	Referenca na redak tablice role

Tablica 4: Prikaz tablice role

atribut	**tip**	**opis**
id	broj	Identifikator uloge
name	tekst	Ime uloge

2.5. FERIoT mobilna aplikacija

2.5.1. Android

Android je *open-source* operacijski sustav za uređaje poput mobilnih telefona i tableta. Razvijen je pod utjecajem američke tvrtke Google Inc.,a utemeljen na jezgri Linux i drugim softverima otvorenog kôda. U današnje vrijeme Android OS prisutan je i na drugim uređajima poput pametnih televizora i satova te automobila.

Većina Android uređaja automatski uključuje Googleove aplikacije za servise poput Gmail-a i Google tražilice, te Google Play trgovinu za aplikacije odnosno platformu za distribuciju digitalnog sadržaja.

U današnje je vrijeme, prema procjeni, Android najprodavaniji mobilni operacijski sustav za pametne telefone i za tablete. Prema procjeni iz 2017. godine, Android ima više od 2 milijarde mjesečno aktivnih korisnika što je više od bilo kojeg drugog operacijskog sustava. Također, na trgovini Google Play objavljeno je više od 3,5 milijuna aplikacija. [8].

2.5.2. Android Studio

Android studio Google-ov je preporučeni IDE za razvoj aplikacija namijenjenih za Android operacijski sustav. Napravljen je na temelju Intelij IDE-a za razvoj programa u Javi.

Android studio podržava više programskih jezika uključujući Javu i Kotlin koji su dobro integrirani i primarno se koriste za razvoj Android aplikacija. Također, sadrži i podršku za Gradle koji se također koristi u procesu razvoja aplikacija. [9]

2.5.3. Kotlin

Kotlin je programski jezik razvijen od JetBrainsa i Googlea za potrebe razvoja Android aplikacija. Donosi mnoga poboljšanja za programere kao što su sigurnost null-pointera, funkcije proširenja (engl. *extension functions*) i infix notacija. Interoperabilan je s Java- om i radi na Javinom virtualnom stroju (engl. *Java Virtual Machine*, JVM) . U današnje vrijeme, sve više se koristi kao glavni jezik razvoja Android aplikacija. [10]

2.5.4. Arhitektura sustava i dizajn rješenja

FERIoT mobilna aplikacija razvijena je za operacijski sustav Android.

Za razvoj korisničke aplikacije korišten je programski jezik Kotlin te je odabrana arhitektura MVVM (engl. Model-View-ViewModel).

Korisnicima se kao prvi ekran aplikacije prikazuje ekran za prijavu odnosno LoginActivity. Nakon što korisnik unese svoje korisničke podatke, aplikacija od njega traži privolu za lociranjem. Android zahtjeva da se od korisnika eksplicitno zatraži pristanak za očitavanje njegove lokacije. U protivnom taj se podatak ne smije koristiti. Ukoliko korisnik ne da odgovarajuću privolu, lociranje neće biti moguće te aplikacija neće moći korisniku prikazati dostupne uređaje vezane za njegovu trenutnu lokaciju. Ukoliko korisnik dade odgovarajuću privolu, uspješno će moći nastaviti s korištenjem aplikacije.

Lociranje se izvršava korištenjem BLE odašiljača. Signali odašiljača u korisničkom okruženju, očitavaju se korištenjem Bluetooth protokola koji koristi iBeacon standard.

2.5.5. iBeacon standard

Appleov iBeacon standard koji određuje kako trebaju biti formatirani zahtjevi za uspostavljanjem BLE veze, podržan je na iPhone 4S uređajima I novijim, iPadom treće generacije I novijim, iPad-om mini te sa uređajima iPod Touch pete generacije I novijim. iBeacon je također dostupan I na Android uređajima . [7]

iBecon standard sastoji se od 4 identifikatora u sklopu podatkovnog paketa koje odašilje beacon, a to su:

- univerzalni jedinstveni identifikator (engl. *universally unique identifier*, UUID) – 128-bitni podatak koji odgovarajućem korisniku pametnog telefona daje informaciju tko je osoba ili tvrtka koja koristi iBeacon

- The Major – predstavlja 16-bitni cijeli broj bez predznaka (engl. *unsigned integer*) koji se može iskoristiti za povezivanje beacona koji imaju isti UUID
- The Minor – predstavlja 16-bitni cijeli broj bez predznaka koji razlikuje beacone sa istim UUID I major vrijednostima
- Tx Snaga – predstavlja jačinu signala [7]

U aplikaciji je za samo lociranje BLE odašiljačima korištena android-beacon-library biblioteka koja olakšava komunikaciju između aplikacije i odašiljača te izračunavanje udaljenosti korisnika od samog odašiljača. Biblioteka radi na način da pruža procjene udaljenosti do određenog odašiljača, izražene u metrima.

2.5.6. Metode izračunavanja udaljenosti

Procjena udaljenosti utemeljena je na izmjerenoj jačini signala koje emitiraju odašiljači u blizini.

Kao što je navedeno, jačina signala određuje se pomoću vrijednosti Tx snage. Međutim, ta vrijednost sama za sebe nije dovoljna jer odašiljači mogu emitirati pakete s različitim jačinama odašiljanja. To znači da se može pojaviti slučaj u kojemu će jačine signala koje emitiraju odašiljači smješteni na istoj udaljenosti biti različite jer odašiljači emitiraju s različitim razinama TX snage.

Kako bi se riješio problem različitih jačina signala, odašiljači u svojim oglašivačkim paketima emitiraju I referentne jačine signala. Za iBeacon standard to je jačina signala izmjerena na 1 metar udaljenosti od odašiljači. Na taj način aplikacija koja osluškuje signale odašiljača može usporediti referentnu jačinu signala sa onom stvarno izmjerenom I na taj način izračunati moguću udaljenost od odašiljača.

Formule koje se koriste dio su AltBecon biblioteke koja je jedna od najčešće korištenih biblioteka za rad s odašiljačima sa iBeacon standardom za operacijski sustav Android.

Odabir formule ovisi o omjeru izmjerene I referentne jačine signala. U slučaju da je

$\frac{rssi}{txPower} < 1$ koristit će se formula a), au protivnom formula b).

Formule izračunavanja udaljenosti:

a) $(\frac{rssi}{txPower})^{10}$

b) $a * (\frac{rssi}{txPower})^b + c$

Vrijednost dobivena izračunom metoda predstavlja udaljenost u metrima, a varijable a, b i c u formuli b) su konstante. Njihove se vrijednosti određuju pri rješavanju funkcije najbolje aproksimacije (engl. *best fit curve*) s izmjerenim podacima, a izračunate konstante iznose 0.89976 za varijablu a, 7.7095 za varijablu b i 0.11 za varijablu c. [6]

Preciznost mjerenja udaljenosti navedenim metodama može varirati od 0.5 do 2 metra što odgovara potrebama aplikacije budući da nisu potrebne procjene velike preciznosti korisničke lokacije već samo detekcija najbližih odašiljača koji bi omogućili određivanje prostorije u kojoj se korisnik nalazi.

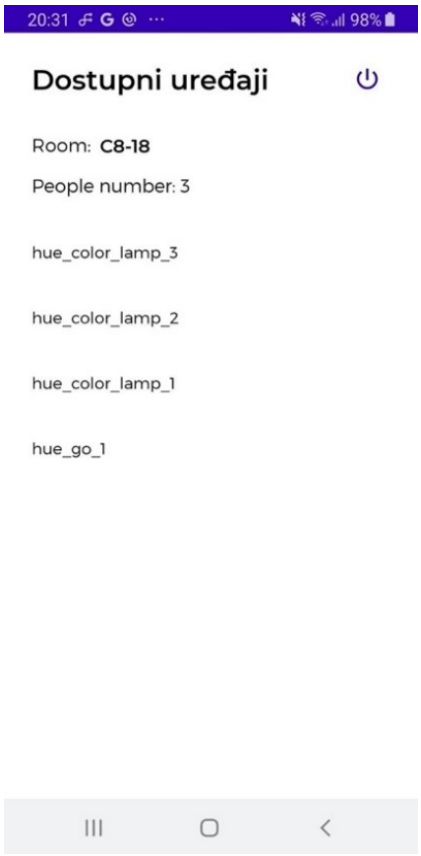
Sama komunikacija između mobilne aplikacije i servisa za lociranje u računalnom oblaku ostvarena je na način da klasa LocationService unutar aplikacije svakih 500 milisekundi prikuplja otkrivene BLE signale koje emitiraju odašiljači koji se nalaze u korisničkoj blizini te ih lokalno pohranjuje. Pri tome se za svako očitanje, s ciljem što preciznije procjene i otklanjanja šumova, primjenjuje Kalmanovo filtriranje.

Ukoliko se prikupe barem tri uspješna očitanja, aplikacija te podatke šalje aplikaciji Locator, koja na temelju dostavljenih očitanja procjenjuje u kojoj se prostoriji korisnik nalazi. Kako bi podatci bili što svježiji, provjera lokalno pohranjenih očitanja te njihovo slanje aplikaciji Locator odvija se periodički. To je postignuto korištenjem singleton obrasca za implementaciju LocationTimera, koji svake tri sekunde dohvaća pohranjena očitanja I ukoliko su ona ispravna, šalje ih aplikaciji Locator.

Nakon uspješnog lociranja korisnika, aplikacija nastavlja direktnu komunikaciju sa Home Assistant API-jem. Home Assistant API na temelju prostorije u kojoj se korisnik nalazi na fakultetu, vraća listu svih pametnih uređaja koji se u toj prostoriji nalaze. Korisnik tada ima mogućnost komunicirati s bilo kojim od tih uređaja putem mobilne aplikacije.

Izgled mobilne aplikacije moguće je vidjeti na slikama 8, 9, 10 i 11:

Slika 10 Prikaz ekrana za prijavu u sustav



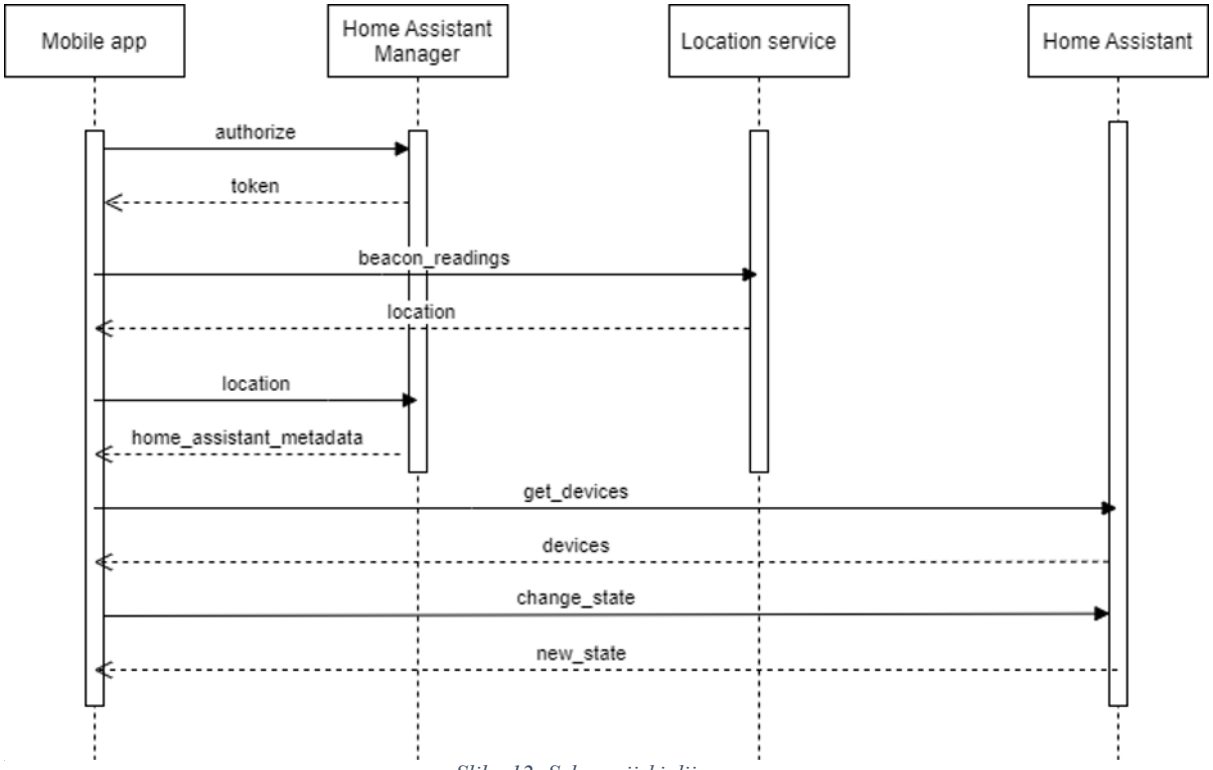
Slika 9 Prikaz ekrana za prikaz dostupnih uređaja

Slika 8 Prikaz ekrana za upravljanje Phillips Hue lampom

Slika 11 Prikaz ekrana za odabir boje na Phillips Hue lampi

3. Sekvencijski dijagram komunikacije komponenti u sustavu

Na slici 11 moguće je vidjeti sekvencijski dijagram koji prikazuje odvijanje komunikacije u sustavu prilikom pokretanja FERIoT aplikacije. Komunikacija započinje upisivanjem korisničkih podataka u aplikaciju na temelju čega mobilna aplikacija od usluge Home Assistant Manager dobiva token s određenim pravima. Aplikacija nakon toga šalje očitanja BLE doašiljača servisu Locator koji na temelju njih određuje u kojoj se prostoriji nalazi korisnik. Potom korisnička aplikacija dobivenu lokaciju šalje ponovno usluzi Home Assistant Manager koja na temelju lokacije i tokena kojeg je dobila od korisničke aplikacije iz baze podataka dohvaća lokaciju i token za onu platformu koja odgovara lokaciji uređaja i pravima dodijeljenim korisniku (tj njegovom tokenu) prilikom prijave u sustav. Ti podatci dovoljni su da korisnička aplikacija potom pošalje zahtjev za uređajima koji su registrirani na platformi zaduženoj na toj lokaciji. Nadalje komunikacija između korisničke aplikacije i platforme se događa direktno i moguća je sve dok korisnik ne izađe iz aplikacije, točnije odjavi se iz aplikacije ili ručno zahtjeva ponovno rekalibriranje.



Slika 12: Sekvencijski dijagram

Studijski primjer

Studijski primjer ovog rada napravljen je u prostorijama Fakulteta za elektrotehniku i računarstvo. U lokalnoj mreži IoT laboratorija pokrenuta je platforma Home Assistant na kojoj su registrirani okolni pametni uređaji. Ista ta platforma registrirana je u sustavu preko usluge Home Assistant Manager s dvije uloge, profesor i student. Profesoru su dodijeljena sva prava dok je ulozi učenik dodijeljeno pravo na samo određene uređaje. U prvom slučaju paljenja mobilne aplikacije sustav je testiran koristeći ulogu profesora. Nakon određenog vremena aplikacija je uspješno locirana koristeći uslugu Locator te je potom uspješno dohvatila uređaje sa platforme Home Assistant. U tom načinu rada korisnik je u mogućnosti promijeniti stanje svih navedenih uređaja što pokazuje ispravno ponašanje sustava. Nakon gašenja aplikacije i ponovnog paljenja aplikacija je pokrenuta u koristeći ulogu studenta. Nakon uspješne prijave i lociranja aplikacije u ovom načinu rada korisniku su se prikazali isti uređaji no prilikom pokušaja promjene stanje, za neke uređaje korisnik to nije mogao napraviti. Ovo pokazuje ispravno ponašanje sustava kod korištenja uloge s ograničenim pravima.

Zaključak

Ovaj rad obrađuje temu implementacije rješenja za pametno okruženje. Proširivanjem mogućnosti platforme Home Assistant napravljen je sustav pomoću kojega je moguće vrlo vješto upravljati pametnim uređajima u blizini. Rješenje je neovisno o okruženju, dakle može biti postavljeno u sklopu doma, fakulteta, kampusa, zagrade ili bio kojeg sličnog objekta. Rad također implementira i korisničku aplikaciju preko koje je moguće ostvariti interakciju s uređajima pametne okoline. Iako samo lociranje uređaja može biti poboljšano, ono je dovoljno dobro i precizno za sustav da odrađuje svoje zadaće. Uz mobilnu aplikaciju rad predstavlja i korisničko web sučelje pomoću kojega je moguće uređivati, brisati i dodavati korisnike i platforme u sustav.

Kao daljnji rad potrebno je unaprijediti sustav lociranja korisnika, povećati razinu sigurnosti i izbrusiti komunikaciju između aplikacije i uređaja.

Sažetak

FERIoT pametni sustav za upravljanje pametnim uređajima u blizini

U ovome je radu opisan FERIoT pametni sustav za upravljanje pametnim uređajima u blizini. Navedeni sustav implementiran je kao mobilna aplikacija koja locira korisnikov uređaj u prostoru koristeći BLE odašiljače, te na temelju lokacije i prava pojedinog korisnika u sustavu, omogućava korisniku upravljanje pametnim uređajima u njegovoj neposrednoj blizini. Rješenje je primjenjivo u svim pametnim okruženjima kao što su pametni dom, zgrada, kampus ili grad. Rad je strukturiran tako da obrađuje 4 glavne cjeline, a to su uvodni dio, razrada implementacije, okvirni plan za unaprjeđenje i održavanje sustava te zaključni dio.

Uvodni dio opisuje motivaciju za sami projekt te tehnologije koje su okosnica samog rješenja sustava.

U glavnom dijelu rada opisuje se cjelokupna arhitektura i dizajn rješenja sustava. Pri tome se postupak implementacije opisuje za svaku od komponenti sustava, a to su aplikacija Locator, aplikacija Home Assistant Manager, Home Assistant te FerIoT mobilna aplikacija za operacijski sustav Android.

U završnom dijelu rada opisali smo plan prema idejama koje smo definirali kao dobre potencijalne buduće značajke s ciljem unaprjeđenja postojećeg sustava.

Također, u posljednjem poglavlju ovoga rada izmiljeli smo zaključke o implementiranom rješenju na temelju rezultata studijskog primjera.

Ključne riječi: FERIoT, BLE, beacon, operacijski sustav Android, pametni sustav za upravljanje pametnim uređajima u blizini, Home Assistant, Locator, Home Assitant Manager, korisnička lokacija, korisnička prava

Abstract

FERIoT – the smart system for managing nearby smart devices

This paper describes the FERIoT smart system for managing nearby smart devices. This system is implemented as a mobile application that locates the user's device using BLE beacons, and based on the location and rights of each user in the system, allows the user to manage smart devices in his immediate vicinity. The proposed solution could be implemented in smart homes, buildings or cities. The paper is structured in such a way that it deals with 4 main units, namely the introductory part, the elaboration of the implementation, the framework plan for the improvement and maintenance of the system, and the concluding part.

The introduction describes the motivation for the project itself and the technologies that are crucial for system implementation.

The main part of the paper describes the overall architecture and design of the system solution. The implementation process is described for each of the system components, namely the Locator application, the Home Assistant Manager application, the Home Assistant application and the FerIoT mobile application for the Android operating system.

In the final part of the paper, we described the plan according to the ideas that we defined as good potential future features with the aim of improving the existing system.

Also, in the last chapter of this paper, we exchanged conclusions about the implemented solution based on the results of the study example.

Keywords: FERIoT, BLE, beacon, Android operating system, smart nearby device management system, Home Assistant, Locator, Home Assitant Manager, user location, user rights

Literatura

[1] DeepStack, https://www.deepstack.cc/, lipanj 2022.

[2] Službena dokumentacija Home Assistant-a, https://www.home-assistant.io/installation/, lipanj 2022.

[3] Službena dokumentacija alata Node RED, https://nodered.org/docs/, lipanj 2022.

[4] Drobac P., Umrežavanje i upravljanje uređajima pametnog kampusa pomoću platforme Home Assistant, Završni rad, lipanj 2022.

[5] Čabarkapa D., Grujić I., Pavlović P., Comparative Analysis of the Bluetooth Low-Energy Indoor Positioning Systems, Listopad 2015.

[6] Orečić I., Ispitivanje usluge pozicioniranja u prostorima Fer-a, Diplomski rad, lipanj 2019.

[7] iBeacon, https://www.techtarget.com/iotagenda/definition/Apple-iBeacon, 20.6.2022.

[8] What is Android, https://www.android.com/, 21.6.2022.

[9] Službena dokumentacija Android Studia, https://developer.android.com/studio, 21.6.2022.

[10] Kotlin Programming Language, Kotlin Programming Language (kotlinlang.org), 22.6.2022.

[11] Developer Home Assistant, https://developers.home-assistant.io/docs/architecture_index, lipanj 2022.

[12] Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. Information systems frontiers, 17(2), 243-259.

[13] Jetson nano, https://docs.nvidia.com/jetson/l4t/ , lipanj 2022.

[14] Docker, https://www.docker.com/, lipanj 2022.

[15] PostgreSQL, https://www.postgresql.org/, lipanj 2022.Službena dokumentacija programskog jezika Kotlin, https://kotlinlang.org/, 21.6.2022.