UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA TELEKOMUNIKACIJE

Home-assistant platform

Projektni zadatak iz predmeta Telekomunikacijski softver inženjering
Teorijski dio

Ćutahija Zerina, 1685/17085
Hasanbegović Selma, 1574/17753
Mahovac Nerman, 1575/17919
Repeša Almin, 1684/17550
Velić Nejra, 1634/17313

Sarajevo, 2019. godina

Sadržaj

Opis problema

Klijent je kupio starter-set Xiaomi smart-home Internet of Things (IoT) uređaja (*motion detector*, *door/window sensor*, *smart socket plug*, *temperature/humidity sensor* i *wireless switch*). Ipak, zbog bojazni za privatnost i sigurnost informacija o korištenju uređaja, klijent insistira na konfiguraciji i instalaciji Home-assistant platforme. Prema zahtjevu klijenta, mreža koja se instalira u privatnom okruženju bi trebala blokirati sav odlazni IoT saobraćaj te bi se poptuna kontrola upravljanja i nadzora smart-home orkuženja trebala vršiti isključivo preko home-assistant platforme.

Internet of Things (IoT) uređaji imaju senzore i softver koji omogućava sakupljanje i razmjenu podataka putem interneta. IoT objekti mogu biti kontrolisani tako da dozvole direktnu integraciju sa računarskim sistemima, što dovodi do ekonomskih benefita i veće efikasnosti za same korisnike [1].

Koncept „Pametnog doma“ (*Smart home*) objedinjuje opremu sa senzorima koji su osjetljivi na optiku, temperaturu, pokret, vlagu, pritisak itd., različite sisteme (grijanje, osjvjetljenje, sigurnost...) te kućanske uređaje (mašina za veš, frižider, kućni roboti itd.) koji mogu biti pokrenuti i kontrolisani putem računara, pametnog telefona ili tableta unutar doma ili putem interneta [2].

Glavni zadatak tima jeste implementacija, odnosno instalacija smart-home okruženja korištenjem home-assistant platforme te, nakon povezivanja platforme sa Xiaomi starter-set uređajima, podešavanje osnovnih funkcionalnosti u svrhu demonstracije ispravnosti rada. Posebnu pažnju treba obratiti na klijentov zahtjev za blokiranjem odlaznog IoT saobraćaja i, pored implementacije navedenog, osigurati nadzor i potpunu kontrolu upravljanja smart-home okruženja isključivo preko navedene platforme.

Prije same izrade projekta izvedena je i studija izvodljivosti ("*Feasibility Study*"), kojim se ispitatuje mogućnost izrade projektnog zadatka. Istraživanjem i analiziranjem softwareskog rješenja je pokazano da je moguće implementirati isto u svrhu ispunjenja zahtjeva klijenta. Na kraju je potrebno izvršiti testiranja maksimalnog opterećenja platforme i broja priključenih uređaja.

1. Organizacija projekta

1.1. Organizacija tima

U ovom odjeljku dat je opis razvojnog tima i alata korištenih pri razvoju *software*-skog rješenja, način komunikacije unutar tima i odabir vođe istog. Prikaz načina organizacije tima dat je u tabeli 1.1.

Za vođu tima je jednoglasano izabran Mahovac Nerman zbog svojih komunikacijskih sposobnosti, samopouzdanja, kao i spremnosti na preuzimanja odgovornosti za cjelokupan projekat. Također, zadatak vođe tima jeste i ostvarivanje komunikacije sa ostalim članovima tima u svrhu dodjele zadataka i uvida u izradu istih. Komunikacija unutar tima se odvija kako usmeno, tj. putem sastavana uživo, tako i pismeno, tj. *online* vid komunikacija, što omogućava kontinuiran uvid u izradu zadataka.

U svrhu struktuiranja tima korišten je demokratski pristup, gdje se svakom od članova tima dodjeljuju ravnopravni zadaci tokom izrade projekta čime se smanjuje neravnopravno opterećenje po članu tima i omogućava uvid u rad drugih članova tima što, ne samo da pospješuje komunikaciju unutar tima, nego i olakšava izradu zadataka.

Alati koji će biti korišteni pri razvoju software-skog rješenja su Microsoft Excel, GitHub, StarUML, VisualParadigm (Online Diagrams, dostupno na: `https://online.visual-paradigm.com/diagrams/`).

Tablica 1.1: Organizacija tima

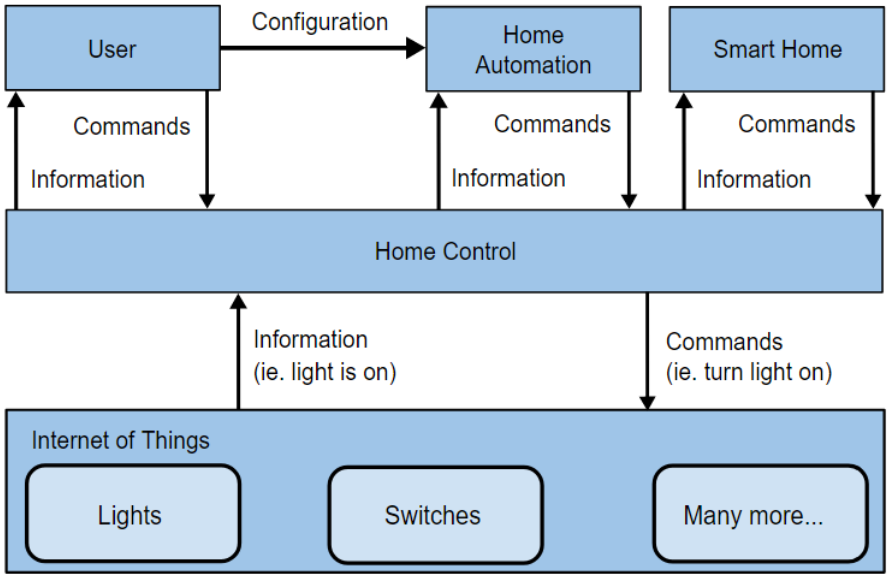
Uloga	**Član tima**	**Odgovornosti**
Vođa tima	Mahovac Nerman	Zadužen je za koordinaciju, komunikaciju unutar tima i odgovoran je za cjelokupni projekat. Vrši dodjelu zadataka i snosi odgovornost za njihovu izradu. Zajedno sa ostalim članovima tima vrši procjenu vremena i troškova potrebnih za izvršenje zadataka.
Programer	Repeša Almin	Bira razvojno okruženje za pisanje koda. Razvija dizajn i implementira proizvod.
Tester	Velić Nejra	Vrši testiranje i osigurava ispravnost konačnog proizvoda. Pri svakoj iteraciji utvrđuje da li proizvod zadovoljava potrebe klijenta.
Builder baze podataka	Ćutahija Zerina	Radi sa programerom na koordinaciji i izradi baze podataka. Radi sa dizajnerom na izradi dokumentacije baze podataka.
Dizajner	Hasanbegović Selma	Odlučuje o finalnom dizajnu i arhitekturi proizvoda. Surađuje sa ostalim članovima tima prilikom pisanja dokumentacije.

1.2. Analiza postojećih rješenja

U ovoj sekciji je odabrano i ukratko opisano nekoliko postojećih rješenja implementirana u različitim okruženjima. Posebna pažnja je posvećena Home-assistant platformi, koja će se kroristiti u nastavku projekta.

1.2.1. Home-assistant platform

Home-assistant platforma predstavlja open source rješenje za automatizaciju doma, koje se može konfigurisati i integrisati u mnoge uređaje kao što su Amazon Alexa, Google Assistant, Z-Wave kao i Zigbee uređaje (razni senzori, prekidači), te između ostalog i Xiaomi smart home uređaje koji će biti upravljani putem navedene platforme konfigurisane prema zahtjevima klijenta. Navedena platforma je napisana u Python programskom jeziku, verzija 3 [3].



Slika 1.1: Prikaz strukture kućne automatizacije

Platforma koristi Hass.io kao operativni sistem koji vodi brigu o održavanju i instalaciji Home Assistant-a, koji je upravljan preko Home Assistant User Interface-a te omogućava kreiranje i uspostavljanje konfiguracija. Ovaj operativni sistem je baziran na ResinOS i Docker-u [4]. Na slici 1.1 je dat prikaz strukture kućne automatizaje, a detaljan opis svih komponenti je dostupan na stranici Home-assistant - Developer documentation.

1.2.2. OpenHUB

OpenHUB (*open Home Automation Bus*) je platforma čiji je glavni cilj integracija i interakcija između različitih tehnologija i uređaja za kućnu automatizaciju. U potpunosti je napisan u Javi, te zbog toga ovisi samo o Java Virtualnoj Mašini [5]. Funkcionalni prikaz usredotočen je na to kako su informacije o uređajima, vezama i sl. predstavljene u korisničkim sučeljima. Također, uključuje fokusiranje na to kako pravila utiču na predstavljanje fizičkih uređaja u software-u.

OpenHAB radi na Linuxu, Windows-u i macOS-u, ali može se pokretati i na Raspberry PI, Pine64 i Docker-u, a kao takav koristi opetHABian koji predstavlja pandan Hass.io sistemu kod Home-assistant.a. Nudi mobilne aplikacije za Android uređaje i iOS aplikaciju za iPhone i iPad. OpenHAB može podržati Google Assistant, Amazon Alexa, IFTTT i Apple HomeKit [6].

1.2.3. Eclipse SmartHome

Eclipse SmartHome predstavlja rješenje koje je je po svojoj arhitekturi identično openHAB-u te je također napisan u Javi. Međutim konfiguraciju je moguće vršiti pomoću DSLs (*Domain specific languages*) kao i pomoću konfiguracijskih file-ova. User interface-i su identični onima kod openHAB-a.Navedena platforma omogućava izgradnju smart-home rješenja koja imaju snažan fokus na heterogenim okruženjima, tj. rješenja koja se bave integracijom različitih protokola ili standarda [7]. Podržava mnoge ugrdbene uređaje poput Raspberry Pi, BeagleBone, Intel Edison, Orange Pi, Rock64 i slično.

1.2.4. Calaos

Naredno rješenje koje ćemo obraditi jeste Calaos. Calaos je Linux bazirano rješenje za kućnu automatizaciju i njegov software stack je napisan u C++ programskom jeziku. Arhitektura je nešto drugačija u odnosu na prethodno navedena rješenja, te je predstavljena preko software stack-a koji se sastoji iz 6 cjelina: *Server*, *Home*, *WebApp*, *OS*, *Mobile* i *Installer*. Calaos nudi čitav niz aplikacija koje se mogu koristiti na više uređaja: Web, Android, iOS, Linux i još mnogo toga. Također, podržan je na različitim platformama poput Raspberry Pi, Zodianet's ZiBASE, Cubieboard, Squeezebox.

1.2.5. PiDome

PiDome je platforma za kućnu automatizaciju koju pokreće RaspberryPi. Cilj ove platforme je pružiti jednostavan sistem za korištenje ne-tehničkim osobama. PiDone podržava MySensors, RFXCOM, KODI multimedijski sistem. Pomoću piDome korisnik može kreirati i prilagoditi svoju kontrolnu ploču, koristiti monitore različitih rezolucija i pokretati piDome klijente na Windows, Android i macOS [6].

Navedena rješenja imaju pojedine mane. Neka su loše dokumentovana te je stoga teško razumjeti njihov rad, neka nemaju *user-friendly interface*, dok neka nemaju date upute za integraciju sa mnogim uređajima. Stoga Home Assistant predstavlja idealno rješenje koje nije teško integrisati i konfigurisati čak i za početnike koji ne posjeduju izuzetno tehničko predzananje.

1.3. Big Bang model

Nakon dužeg istraživanja i diskutovanja različitih modela za razvoj *software*-skog rješenja, model koji je u konačnici odabran je Big bang model, iz više razloga. Prije svega, zbog preopterećenja članova tima i nedovoljnog iskustva na polju software inženjeringa navedeni model je izuzetno pogodan jer ne zahtijeva predetaljno planiranje. Međutim, Big bang model nosi sa sobom određene nedostatke, kao na primjer visok rizik i troškovi ukoliko se zahtjevi klijenta pogrešno shvate.

Međutim, iako smo svjesti svih navedenih rizika koje nosi upotreba ovog modela prilikom razvoja software-skog rješenja, zbog jednostavnosti, mogućnosti eksperimentisanja i učenja na projektu za naš tim ovo je najpogodniji model.

1.4. MoSCoW

MoSCoW je tehnika prioritizacije koja se koristi u poslovnoj analizi, upravljanju projektima i razvoju softvera kako bi se postiglo zajedničko razumijevanje prioriteta pri ispunjavanju svakog zahtjeva. Sam termin MoSCoW je akronim izveden od prvih slova kategorija prioriteta, i to: *Must have*, *Should have*, *Could have* i *Won't have* [8]. Shodno tome, prioriteti zahtjeva u konkretno našem slučaju se određuje:

Must have predstavlja zahtjeve koji moraju biti implementirani prije dostave proizvoda. U našem slučaju se to odnosi na zahtjeve samog klijenta:

- Korištenje starter-set Xiaomi smart-home IoT uređaja
- Konfiguracija i instalacija Home-assistant platforme

Should have se odnosi na dodatne zahtjeve koji nisu kritični za samu dostavu proizvoda, ali su od izuzetne važnosti klijentu:

- Blokiranje odlaznog IoT saobraćaja
- Osiguranje potpune kontrole upravljanja i nadzora smart-home okruženja isključivo preko home-assistant platforme

Could have su funkcionalnosti koje bi bilo poželjno implementirati ukoliko vrijeme to dozvoli:

- Detektor dima/požara
- Pametni prekidač za uključivanje/isključivanje svjetala koji bi omogućio i proizvoljno podešavanje nivoa osvjetljenja (Wireless Dimmer Switch)

Wont have predstavljaju funkcionalnosti koje se ostavljaju za implementaciju u nekoj od idućih iteracija:

- Implementacija opcije kontaktiranja službenih lica (policija, vatrogasci) u slučaju detektovanja neregularnosti od strane nekog od senzora
- Automatsko uključivanje video nadzora prilikom aktivacije senzora pokreta

1.5. Procjena vremena i troškova

Za razvoj software-skog rješenja od izrazitog je značaja procijeniti vrijeme koje je potrebno za sam razvoj, kao i troškove projekta. Izuzetno dobar pristup prilikom razvoja

software-a, u cilju uštede vremena je podjela zadataka u aktivnosti i zadatke manjeg obima koji se planiraju na dnevnoj bazi ili u jedinici kalendarskih mjeseci. Za procjenu troška potrebno je razmatrati veličinu i kvalitet software-a, cijenu hardware-a, komunikaciju, vješto osoblje itd.

Za procjenu vremena i troškova razvoja software-a koristit će se tehnike empirijske procjene, koje se baziraju na dekompozitnim tehnikama: Putnam model i COCOMO (COnstructive COst MOdel).
S obzirom da je riječ o malom, jednostavnom timu koji razvija software-sko rješenje uz strogo definisane zahtjeve izabran je COCOMO 81 Semi-detached mod.

COCOMO 81 model definiše sljedeće parametre:

$$MM = a * (KLOC)^b \tag{1.1}$$

$$D = c * MM^d \tag{1.2}$$

Vrijednost $KLOC$ (*kilo line of code*) predstavlja procjenu broja linija koda i ona se koristi pri proračunu jedinice "*man/month*" (MM), dok vrijendost D iz jednačine (1.2) predstavlja procjenu broja mjeseci po projektu (*development time*).

Na osnovu analize koda kojim je opisano navedeno rješenje [9], procjenjeni broj linija koda iznosi 1 676, odnosno procijenjena vrijednost KLOC-a iznosi 1.676. Odabirom odgovarajćeg moda (Semi-detached) dobijamo vrijednosti potrebnih koeficijenata: $a = 3.0$, $b = 1.12$, $c = 2.5$ i $d = 0.35$.

Na osnovu formule (1.1), proračunata vrijednost jedinice "*man/month*" iznosi:

$$MM = 3 * (1.676)^{1.12} = 5.35\ 5"man/month" \tag{1.3}$$

dok je procjena broja mjeseci po projektu:

$$D = 2.5 * 5^{0.35} = 2.5 * 1.756465 = 4.39116\ 4.5mjeseca \tag{1.4}$$

Procjena troška razvoja software-a po COCOMO 81 modelu je data izrazom:

$$Tr = D * MP \tag{1.5}$$

gdje je MP prosječna cijena radnika, za koju je uzeta pretpostavljena vrijednost od 1500 KM, slijedi da je procjena troška:

$$Tr = D * MP = 4.5 * 1500 = 6750KM \tag{1.6}$$

1.6. Procjena rizika

U ovoj sekciji su definisane kritične tačke pri razvoju projekta, procijena vjerovatnoće dešavanja neočekivanih problema i definisane su odgovarajuće reakcije.

U tabeli 1.2 je dat prikaz procjene vjerovatnoće dešavanja neočekivanih problema [10], na osnovu koje je izvršena procjena rizika koji mogu nastati za vrijeme izrade projektnog zadatka (tabela 1.3).

Jedan od zadataka vođe tima jeste da prvi prepozna potencijalni rizik i, u razgovoru sa ostalim članovima tima, pokuša naći rješenje da ne dođe do njegovog pojavljivanja. Monitoring potencijalnih problema se vrši prilikom svakog sastanka tima.

Tablica 1.2: Vjerovatnoća rizika

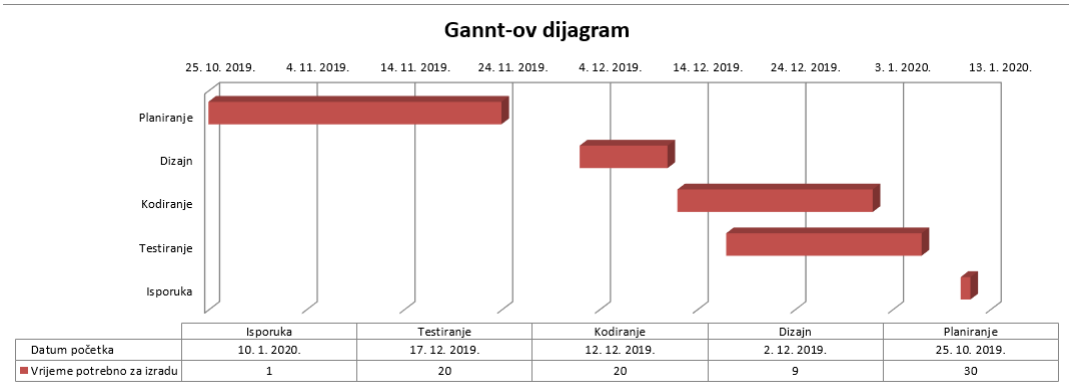
Vjerovatnoća	**Opis**
Velika	Događaj će se javiti u većini slučajeva
Srednja	Događaj se može dogoditi ponekad
Mala	Nastanak događaja nije vjerovatan

Tablica 1.3: Procjena rizika i odgovarajuće reakcije

Rizik	**Vjerovatnoća**	**Odgovarajuća reakcija**
Nemogućnost pronalaska optimalnog rješenja	Srednja	Povećanje interakcije sa klijentom. Konsultacije sa ekspertima iz domena zadanog problema.
Loša komunikacija unutar samog tima. Nedostatak motiva članova tima. Slaba radna atmosfera za izradu projekta.	Mala	Druženja članova tima van projektnih aktivnosti redi povećanja povezanosti tima. Kontinuirana motivacija članova tima od strane vođe tima.
Pogrešna procjena vremenskih okvira	Velika	Nova procjena vremena sa određenim overhead-om. Pridržavanje rasporeda sa novim vremenskim okvirom.
Izostanak člana tima i nemogućnost zamjene istog	Velika	Preraspodjela zadataka na ostale članove tima
Povlačenje Home-assistant platforme iz upotrebe	Mala	Odabir drugog smart-home open source rješenja.
Neispravan rad nekog od korištenih senzora	Srednja	Pravovremena zamjena neispravnog senzora i testiranje ispravnosti istog.
Sukobljeni zahtjevi	Srednja	Pronalazak kompromisa sa klijentom.

1.7. Gannt-ov dijagram

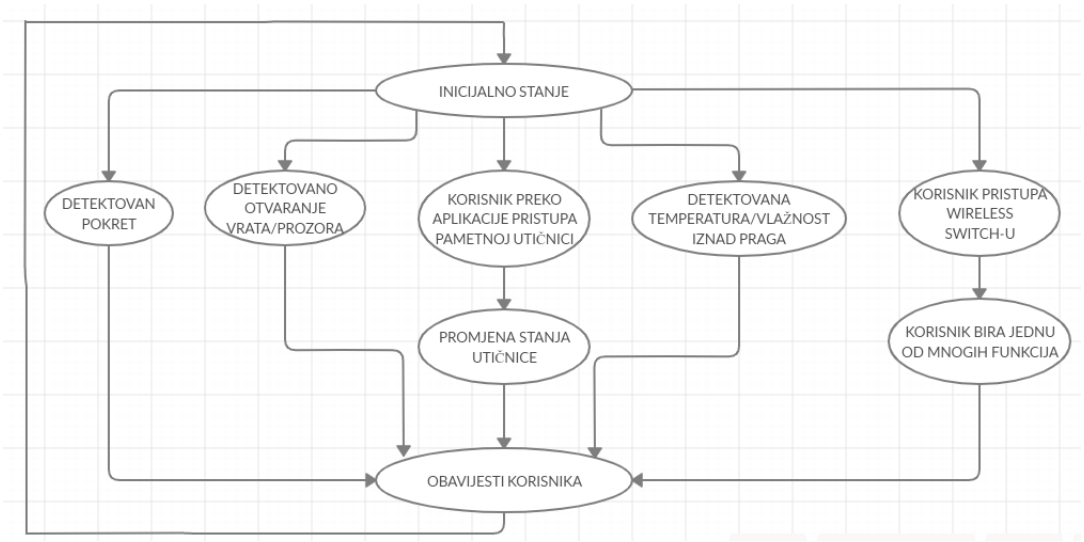
Gannt-ov dijagram služi za predstavljanje projektne aktivnosti u odnosu na vremenske periode, pri čemu je dužina aktivnosti proporcionalna planiranom trajanju te aktivnosti. Navedeni dijagram konkretno za naš projekat je prikazan na slici 1.2 gdje su definisane projekte aktivnosti i procjena vremena za njihovo izvršenje.



Slika 1.2: Gannt-ov dijagram

1.8. FSM dijagram software-skog rješenja

FSM (*Finite State Machines*) predstavlja apstraktnu mašinu koja se može naći samo u jednom stanju od nekoliko konačnih stanja u tačno određenom vremenskom momentu, te može prelaziti iz stanja u stanje u zavisnosti od internih ili eksternih pobuda kroz "tranzicije". Shodno tome, na slici 1.3 je prikazan FSM dijagram za naše konkretno software-sko rješenje, a u nastavku se nalazi kraći uopis istog.



Slika 1.3: FSM dijagram

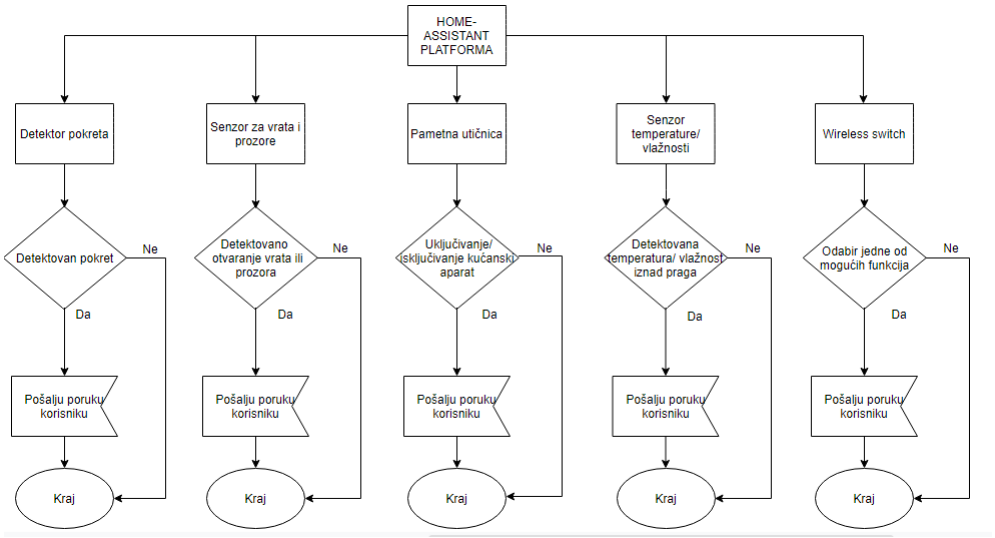
Kad je u pitanju FSM dijagram, sistem se nalazi u inicijalnom stanju sve dok ne

dođe do korisnikovog pristupa aplikaciji (provjera utičnica, pristup wireless switch-u) ili neočekivanih pojava (detektovan pokret, povišena temperatura,...). U tom slučaju sistem prelazi u stanje koje je direktno vezano za ponašanje korisnika i/ili očitanju senzora. U ovom stanju se nalazi vrlo kratko, nakon toga sistem prelazi u stanje obavještenja korisnika o nastalim promjenama u kojem se zadržava sve dok korisnik ne vidi iste. Na poslijetku se sistem vraća u inicijalno stanje i čeka nove pobude za ponavljanje postupka.

1.9. SDL dijagram software-skog rješenja

SDL (*Specification and Description Language*) predstavlja jezik za modeliranje koji se koristi za opisivanje sistema u stvarnom vremenu i baziran je na setu specijalnih simbola i pravila. Postoje tri dijela SDL dijagrama: definicija sistema, blok i proces.

Na slici 1.4 prikazan je SDL dijagram našeg software-skog rješenja. Osnovno stanje je sistem Home-assistant platforma, na kojoj je implementirano 5 funkcionalnosti. Za svaku od funkcija prikazana je akcija koja se preduzima u slucaju njenog aktiviranja, od pocetka do kraja.



Slika 1.4: SDL dijagram

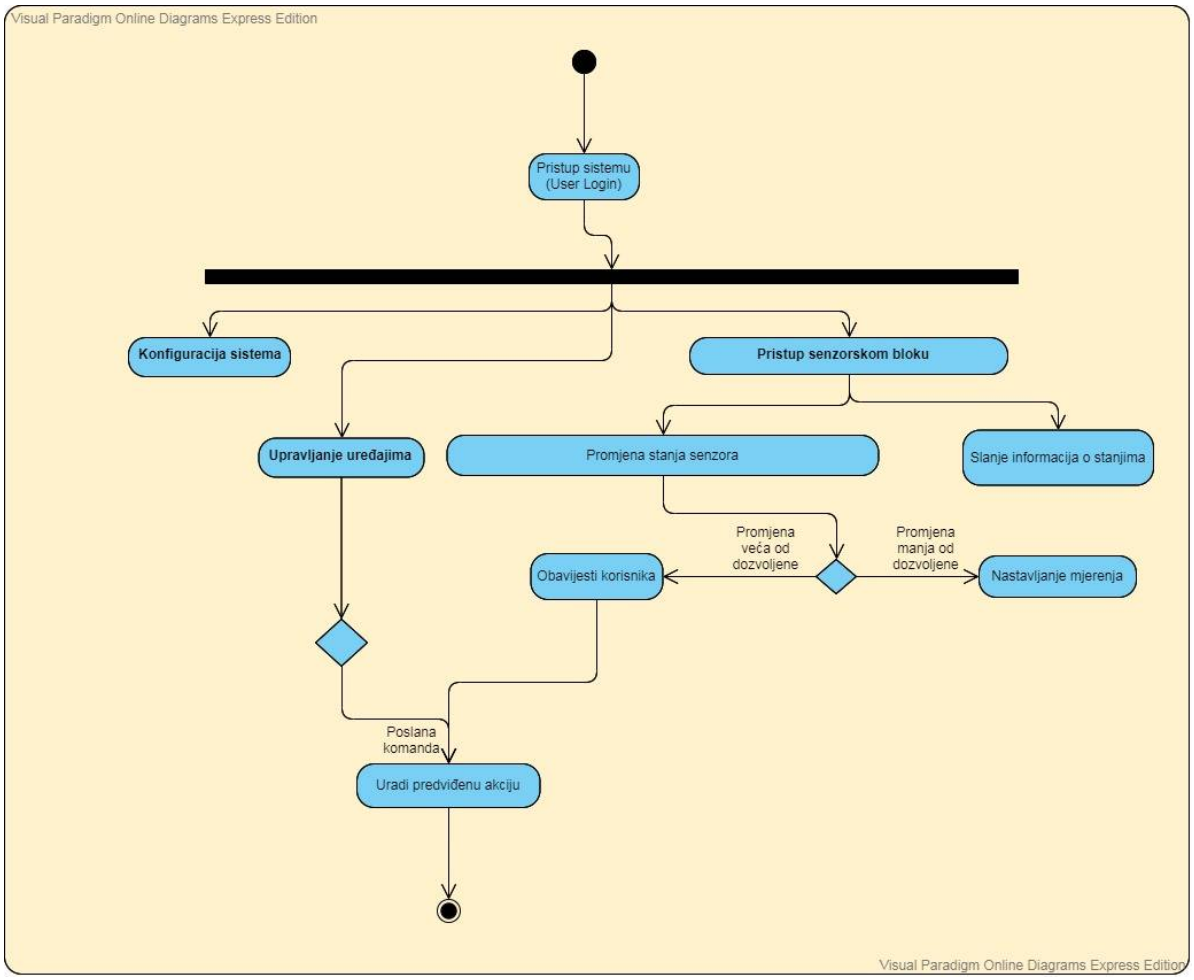
1.10. UML dijagrami software-skog rješenja

UML (*Unified Modeling Language*) predstavlja jezik opšte namjene za modeliranje i specifikacije u softverskom inženjerstvu. Koristi se za analizu, dizajn i implementaciju softversko zasnovanih sistema.

Kreirani su različiti UML dijagrami za potrebe opisa našeg rješenja, dok su odabrani dijagrami: dijagram aktivnosti, dijagram klasa i dijagram sekvence. U nastavu je dat kraći opis i prikaz pojedinačnih dijagrama.

UML dijagram aktivnosti

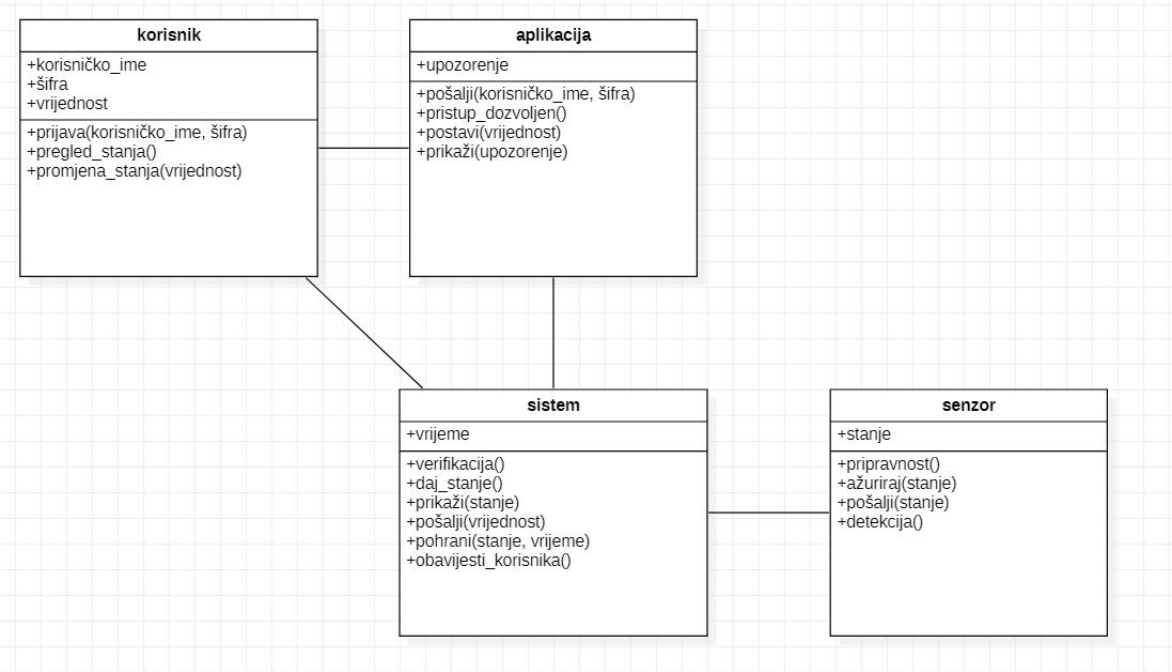
Dijagram aktivnosti je dijagram toka koji prikazuje odvijanje upravljanja od jedne aktivnosti do druge. Na slici 1.5 je prikazan dijagram aktivnosti za Home-assistant platformu. Prva aktivnost jeste pristup sistemu nakon čega se tok razdvaja u više aktivnosti kao što su konfiguracija sistema, upravljanje uređajima i pristup senzorskom bloku. Nadalje sistem izvršava određene akcije u ovisnosti o stanjima senzora te šalje informacije o određenim stanjima unutar doma.



Slika 1.5: UML dijagram aktivnosti

UML dijagram klasa

Dijagram klasa opisuje strukturu sistava objašnjavajući klase unutar sistava, njihove atribute i odnose. Dijagram prikazan na slici 1.6 prikazuje uopštenu komunikaciju unutar Home-assistant platforme između korisnika, aplikacije, sistema(servera) i generaliziranog senzora. U narednim koracima izrade projektnog zadatka će biti detaljno analizirani upotrijebljeni senzori, te njihova komunikacija sa ostatkom sistema, kao i ostali detalji o implementaciji.



Slika 1.6: UML dijagram klasa

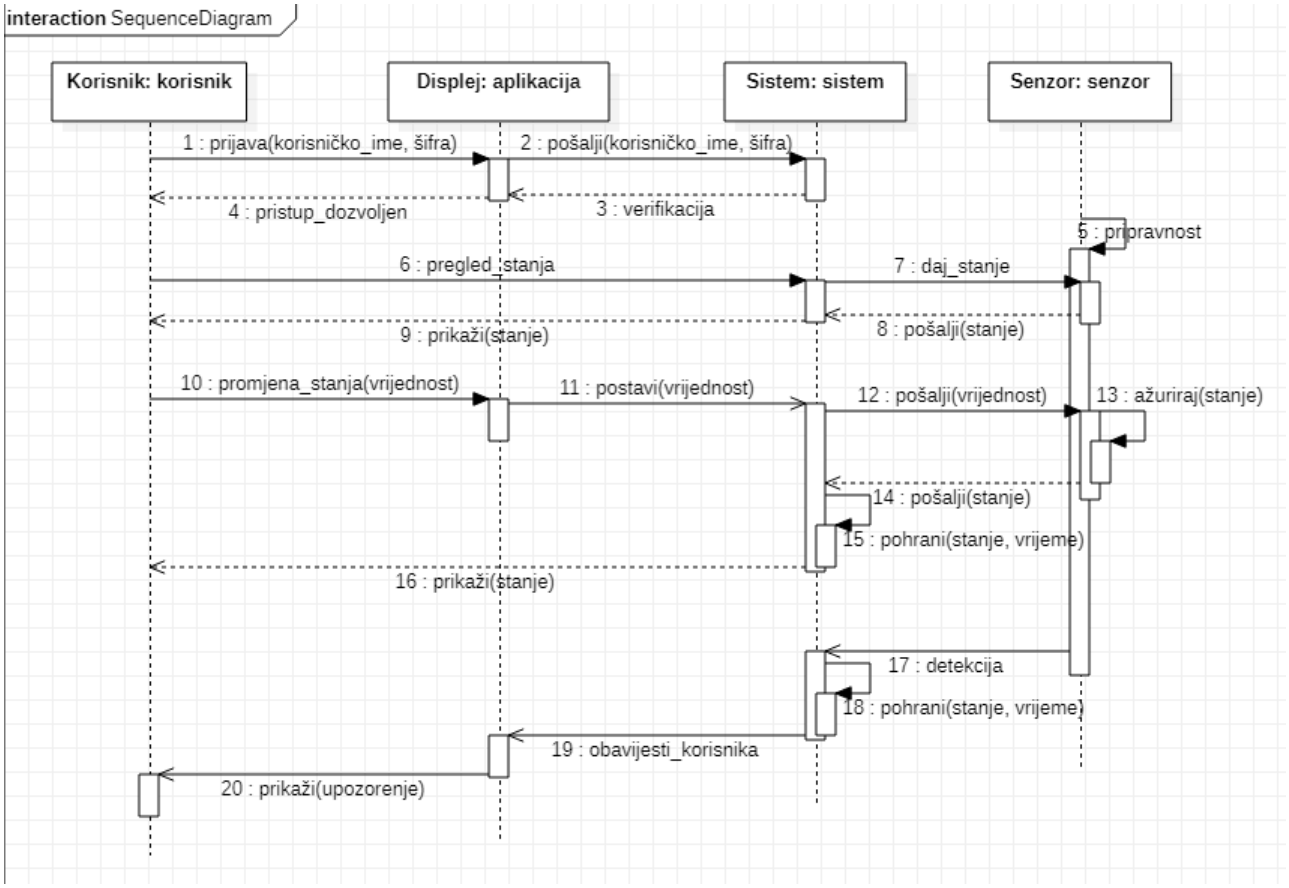
UML dijagram sekvence

Dijagram sekvence, odnosno sekvencijalni dijagram jeste jedan od dijagrama interakcija koji opisuju interakcije aktera i klasa u sistemu kao vremenski određen niz poruka koje oni razmjenjuju.

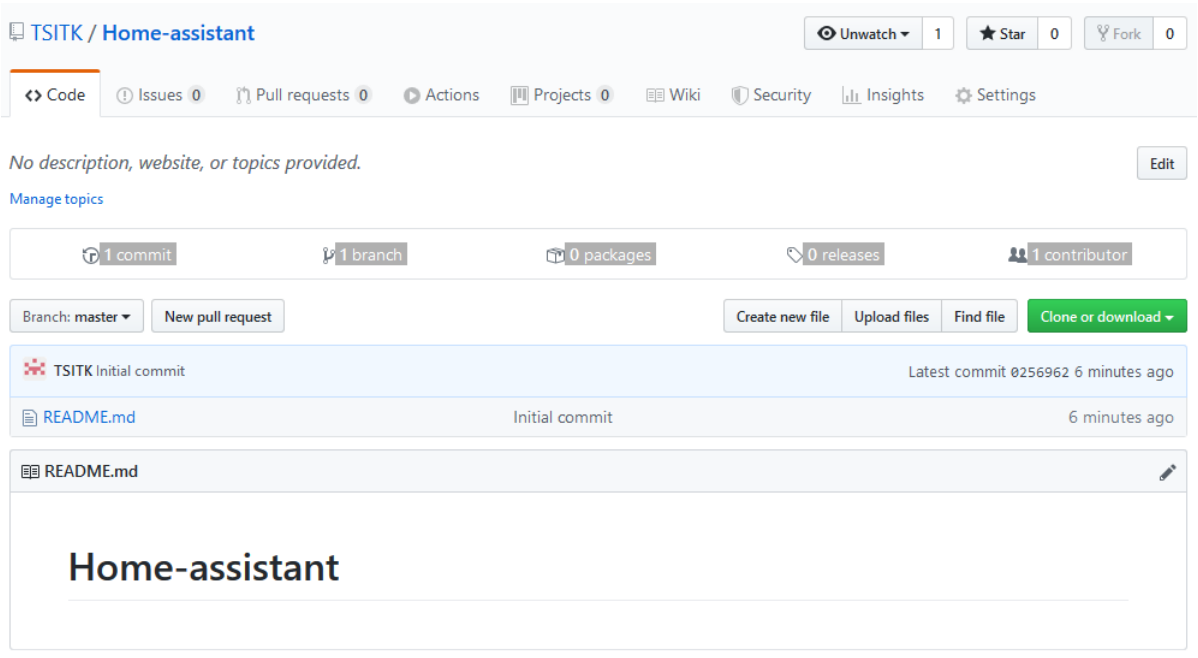
Na slici 1.7 prikazan je sekvencijalni dijagram našeg rješenja, gdje su akteri (objekti) isto kao i u slučaju dijagrama klasa, a za svaki objekat je definisana linija života (vertikalna isprekidana linija) i odgovarajuće akcije koji oni poduzimaju.
Potrebo je obratiti pažnju na aktivnost objekta senzor i akciju 'pripravnost'. Navedena akcija predstavlja činjenicu da je senzor sve vrijeme aktivan, tj. u stanju 'pripravnosti' i adekvatno ažurira stanja, sve dok ne detektuje odgovarajuću akciju, nakon čega se, naposlijetku, obavještava korisnik.

1.11. GIT repozitorij

Na stranici github.org kreiran je GIT repozitorij pod nazivom: TSITK/Home-assistant.



Slika 1.7: UML dijagram sekvence



Slika 1.8: GIT repozitorij TSITK/Home-assistant

Popis slika

Popis tablica

Bibliografija

[1] "The definition of Internet of Things: A simple explanation," 2019. [Online]. Available: https://www.expressvpn.com/blog/what-is-the-internet-of-things-iot/?offer=3monthsfree&offer_code=3c6hjoor69&gclid\=CjwKCAiA8K7uBRBBEiwACOm4dzgw9SQpkFSXdgQu-sojcBfiQiWR-\9G1nsejMWO45QxeatIjy7U0XhoChCYQAvD_BwE

[2] M. N. Anwar, M. N. Nazir, and K. Mustafa, "Security threats taxonomy: Smart-home perspective," *2017 3rd International Conference on Advances in Computing,Communication Automation (ICACCA) (Fall)*, pp. 1–4, 2017.

[3] "Home Assistant - Developer documentation." [Online]. Available: https://developers.home-assistant.io/en/

[4] "Introducing Hass.io." [Online]. Available: https://www.home-assistant.io/blog/2017/07/25/introducing-hassio/

[5] "Introduction | openHAB." [Online]. Available: https://www.openhab.org/docs/

[6] "17 Open-source Free Home Automation Systems." [Online]. Available: https://medevel.com/17-home-automation-open-source/

[7] "Eclipse SmartHome - A Flexible Framework for the Smart Home." [Online]. Available: https://www.eclipse.org/smarthome/documentation/index

[8] K. Waters, "Prioritization using moscow," *Agile Planning*, vol. 12, p. 31, 2009.

[9] "GitHub - home-assistant." [Online]. Available: https://github.com/home-assistant/home-assistant

[10] "Smjernice za Provođenje Procesa Upravljanja Rizicima u Institucijama BiH," 2015. [Online]. Available: https://www.mft.gov.ba/bos/images/stories/chj/Registri/2015/smjernice_bs.pdf

[11] "SDLC - Big Bang Model." [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_bigbang_model.htm

[12] "Calaos | MySensors." [Online]. Available: https://www.mysensors.org/controller/calaos