



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



Иван Богдановић

# Реализација клијентске апликације за паметне куће на iOS платформи

ДИПЛОМСКИ РАД  
- Основне академске студије -

Нови Сад, 2014



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
21000 НОВИ САД, Трг Доситеја Обрадовића 6

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	Монографска документација
Тип записа, <b>ТЗ:</b>	Текстуални штампани материјал
Врста рада, <b>ВР:</b>	Завршни (Bachelor) рад
Аутор, <b>АУ:</b>	Иван Богдановић
Ментор, <b>МН:</b>	Др Иштван Пап
Наслов рада, <b>НР:</b>	Реализација клијентске апликације за паметне куће на iOS платформи
Језик публикације, <b>ЈП:</b>	Српски / латиница
Језик извода, <b>ЈИ:</b>	Српски
Земља публиковања, <b>ЗП:</b>	Република Србија
Уже географско подручје, <b>УГП:</b>	Војводина
Година, <b>ГО:</b>	2014
Издавач, <b>ИЗ:</b>	Ауторски репринт
Место и адреса, <b>МА:</b>	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО:</b> (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/50/9/4/26/0/0
Научна област, <b>НО:</b>	Електротехника и рачунарство
Научна дисциплина, <b>НД:</b>	Рачунарска техника
Предметна одредница/Кључне речи, <b>ПО:</b>	иОС, ОБЛО, паметне куће, графичка корисничка спрега
<b>УДК</b>	
Чува се, <b>ЧУ:</b>	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	У овом раду описано је једно решење клијентског програма који управља радом ОБЛО система. Решење је реализовано за иОС платформу са оперативним системом верзије 6.0 или новијим
Датум прихватања теме, <b>ДП:</b>	
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	Председник: Др Илија Башичевић, доцент
	Члан: Др Небојша Пјевалица, доцент
	Члан, ментор: Др Иштван Пап, доцент
	Потпис ментора



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	Monographic publication
Type of record, <b>TR</b> :	Textual printed material
Contents code, <b>CC</b> :	Bachelor Thesis
Author, <b>AU</b> :	Ivan Bogdanovic
Mentor, <b>MN</b> :	PhD Istvan Pap
Title, <b>TI</b> :	Smart home client application solution based on iOS platform
Language of text, <b>LT</b> :	Serbian
Language of abstract, <b>LA</b> :	Serbian
Country of publication, <b>CP</b> :	Republic of Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	2014
Publisher, <b>PB</b> :	Author's reprint
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, <b>PD</b> : <small>(chapters/pages/ref./tables/pictures/graphs/appendixes)</small>	7/50/9/4/26/0/0
Scientific field, <b>SF</b> :	Electrical Engineering
Scientific discipline, <b>SD</b> :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, <b>S/KW</b> :	iOS, OBLO, smart home, GUI
<b>UC</b>	
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, <b>N</b> :	
Abstract, <b>AB</b> :	This paper describes one solution of client application that controls the Oblo system. The solution was implemented for the iOS platform with operating system version 6.0 or later.
Accepted by the Scientific Board on, <b>ASB</b> :	
Defended on, <b>DE</b> :	
Defended Board, <b>DB</b> :	President: PhD Ilija Basicovic, docent
	Member: PhD Nebojsa Pjevalica, docent
	Member, Mentor: PhD Istvan Pap, docent
	Mentor's sign

## **Zahvalnost**

Posebnu zahvalnost dugujem svojoj porodici zbog pružene moralne, emotivne, kao i finansijske podrške tokom dosadašnjeg školovanja.

Takođe bih se zahvalio rodbini i prijateljima koji su uvek bili tu da isprate moje ideje, vizije i pruže bezrezervnu podršku.

## SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove .....	2
2.1 iOS platforma .....	2
2.2 Razvojno okruženje i alat za testiranje.....	4
2.2.1 Razvojno okruženje .....	4
2.2.2 Alat za testiranje .....	5
2.3 OBLO sistem.....	6
3. Koncept rešenja.....	8
3.1 Grafička korisnička sprega.....	9
3.1.1 Glavni prozor .....	9
3.1.2 Podešavanja .....	9
3.1.3 Kontrola kuće.....	10
3.1.4 Uređivač kuće .....	11
3.1.5 Nadzor.....	11
3.1.6 Bezbednost.....	12
3.2 Pozadinski mehanizmi.....	12
3.2.1 Modul za rukovanje podacima.....	12
3.2.2 Komunikacioni modul .....	12
4. Programsko rešenje.....	13
4.1 Realizacija modula za komunikaciju.....	13
4.2 Realizaciju modula za čuvanje podataka .....	15
4.3 Realizacija grafičke korisničke sprege.....	17
4.3.1 Realizacija glavnog prozora .....	17

---

4.3.2	Realizacija modula podešavanja.....	18
4.3.3	Realizacija kontrole sobe .....	20
4.3.4	Realizacija uređivača kuće .....	25
5.	Rezultati i testiranje .....	32
5.1	Ispitivanje funkcionalnosti .....	32
5.2	Automatski testovi.....	34
5.3	Testovi performansi.....	36
5.3.1	Testovi za osnovni režim rada aplikacije.....	36
5.3.2	Testovi prilikom preuzimanja konfiguracije kuće sa OHM .....	37
5.3.3	Testiranje aplikacije kada postoji više od 20 pogleda na sobu .....	38
6.	Zaključak .....	40
7.	Literatura.....	41

## SPISAK SLIKA

Slika 2.1 Arhitektura iOS platforme .....	3
Slika 2.2 Izgled razvojnog okruženja iOS platforme .....	4
Slika 2.3 Izgled alata <i>Instruments</i> .....	5
Slika 2.4 Arhitektura OBLO sistema .....	6
Slika 2.5 Sekvenca razmene podataka tokom komunikacije .....	7
Slika 3.1 Arhitektura programskog rešenja.....	8
Slika 3.2 Izgledi kontrole sobe.....	10
Slika 4.1 Realizacija komunikacionog modula.....	14
Slika 4.2 Izgled modela baze podataka .....	15
Slika 4.3 Izgled glavnog prozora .....	17
Slika 4.4 Izgled prozora za podešavanje aplikacije .....	18
Slika 4.5 Blok dijagram funkcionisanja SSDP protokola .....	19
Slika 4.6 Izgled sobe/sprata (Kontrola sobe) .....	20
Slika 4.7 Izgled sobe (Kontrola sobe).....	21
Slika 4.8 Izgled rasporeda uređaja u sobi.....	22
Slika 4.9 Izgled panela za kontrolu SmartOutlet uređaja.....	23
Slika 4.10 Izgled grafika .....	24
Slika 4.11 Izgled prozor za kontrolu regulatora svetla .....	25
Slika 4.12 Izgled sobe/sprata (uređivač kuće).....	26
Slika 4.13 Izgled pogleda sobe (uređivač kuće).....	27
Slika 4.14 Izgled rasporeda uređaja u sobi (uređivač kuće).....	28
Slika 4.15 Izgled dijaloga za dodavanje nove veze.....	28
Slika 4.16 Izgled dijaloga za dodavanje novog uređaja.....	29
Slika 5.1 Primer skripte korišćene za automatsko testiranje.....	35
Slika 5.2 Grafički prikaz rezultata testiranja za normalni režim rada aplikacije .....	37
Slika 5.3 Grafički prikaz rezultata testiranja tokom preuzimanja konfiguracije kuće .....	38
Slika 5.4 Grafički prikaz rezultata testiranja kada je prisutno više od 20 pogleda sobe.....	39

**SPISAK TABELA**

Tabela 5.1 Ručni testovi.....	34
Tabela 5.2 Rezultati zauzetosti memorije i procesora za normalni režim rada aplikacije ...	36
Tabela 5.3 Rezultati zauzetosti memorije i procesora tokom skidanja konfiguracije kuće .	37
Tabela 5.4 Rezultati zauzetosti procesora i memorije za slučaj više od 20 pogleda sobe ...	38

## **SKRAĆENICE**

**OHM** - **O**blo **H**ome **M**anager

**SSDP** – **S**imple **S**ervice **D**icover**y** **P**rotocol

**TCP** – **T**ransmission **C**ontrol **P**rotocol

**JSON** – **J**ava**S**cript **O**bject **N**otation

**IP** - **I**nternet **P**rotocol

**LAN** - **L**ocal **A**rea **N**etwork

**ACS** – **A**uto **C**onfiguration **S**erver

**HTTP** - **H**yper**T**ext **T**ransfer **P**rotocol

**SDK** - **S**oftware **D**evelopment **K**it

## 1. Uvod

Sve veća potreba za električnom energijom uz istovremeno očuvanje prirode usloвила je razvoj koncepta pametne kuće. Koncept pametne kuće predstavlja objekat u kome računarski sistem vrši kontrolu nadzora svih električnih uređaja kako bi se povećao komfor i energetska efikasnost objekta. Jedan od takvih sistema koji upravlja pametnom kućom predstavlja OBLO sistem.

U ovom radu opisano je jedno rešenje klijentskog programa koji upravlja radom OBLO sistema. Rešenje je realizovano za iOS platformu sa operativnim sistemom verzije 6.0 ili novijim.

Rad se sastoji iz sedam poglavlja. U drugom poglavlju opisane su teorijske osnove neophodne za razumevanje koncepta pametne kuće, arhitekture programske podrške iOS-a i OBLO sistema. Treće poglavlje čini koncept rešenja u kojem su analizirani zahtevi klijentskog programa i tehnologija potrebnih za efikasnu realizaciju klijentskog programa. Četvrto poglavlje daje uvid u programsko rešenje sa opisom modula koji su korišćeni za realizaciju klijentske aplikacije. Peto poglavlje daje na uvid u korišćene pristupe testiranju programa, kao i sažetak rezultata izvršenog testiranja. Šesto poglavlje pruža kratak rezime šta je urađeno i kakve primene ima realizovano rešenje. Spisak korišćene literature se može naći u poglavlju sedam.

## 2. Teorijske osnove

U ovom poglavlju izložene su teorijske osnove neophodne za razumevanje arhitekture programske podrške u okviru koga je realizovana klijentska aplikacija čiji je cilj upravljanje OBLO sistemom. Navedene su osnovne karakteristike: iOS platforme, OBLO sistema i razvojnog okruženja i alata za testiranje korišćenih za izradu programa.

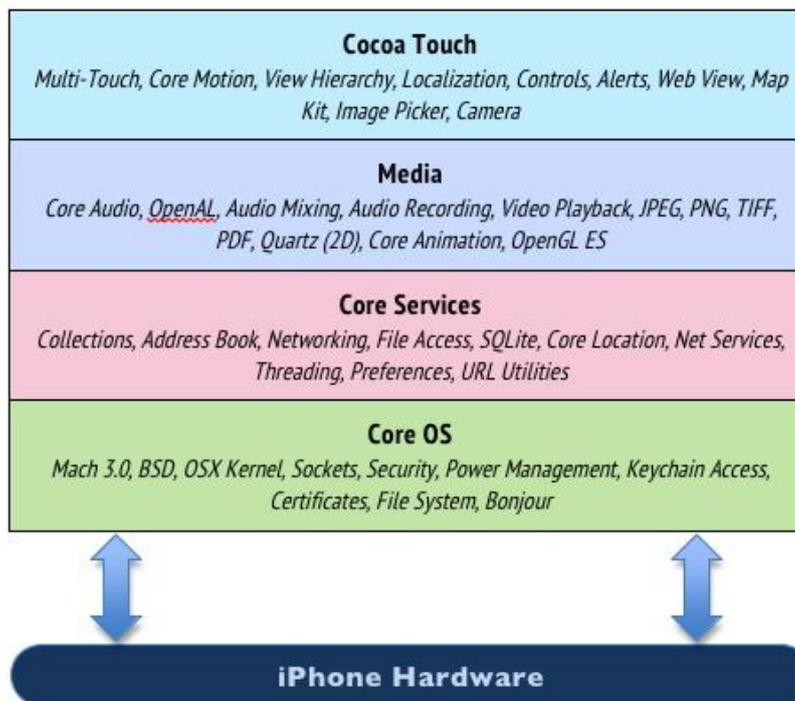
### 2.1 iOS platforma

iOS je operativni sistem razvijen od strane kompanije Apple. Namenjen je iPhone, iPad, iPod Touch i Apple TV uređajima. Koristi isključivo hardver koji je razvijen od strane Apple kompanije [1].

Glavne verzije iOS sistema plasiraju se na tržište svake godine. Poslednja objavljena verzija iOS sistema je 7.0. Trenutna verzija iOS sistema (iOS 7.1.1) koristi 1-1.5 GB memorijskog prostora što zavisi od datog uređaja. Pokreće se na iPhone uređajima četvarte generacije i novijim kao i iPad uređaja druge generacije i novijim, svim modelima iPad Mini, kao i iPod Touch uređaja pete generacije.

Arhitektura iOS operativnog sistema se sastoji iz više programskih slojeva od kojih svaki pruža sprege za programiranje i razvoj aplikacija.

Niži slojevi sadrže osnovne usluge i tehnologije. Slojevi viših nivoa su nadogradnja nižih i daju sofisticirane usluge i tehnologije. Apple kompanija većinu svojih sistemskih interfejsa isporučuje u posebnim paketima koji se nazivaju razvojna okruženja (engl. *frameworks*). U suštini razvojna okruženja su paketi dinamičkih biblioteka i resursnih datoteka (kao što su programska zaglavlja, slike, konfiguracione datoteke). Arhitektura iOS platforme [10] prikazana je na slici 2.1



Slika 2.1 Arhitektura iOS platforme

Opis standardnih iOS slojeva:

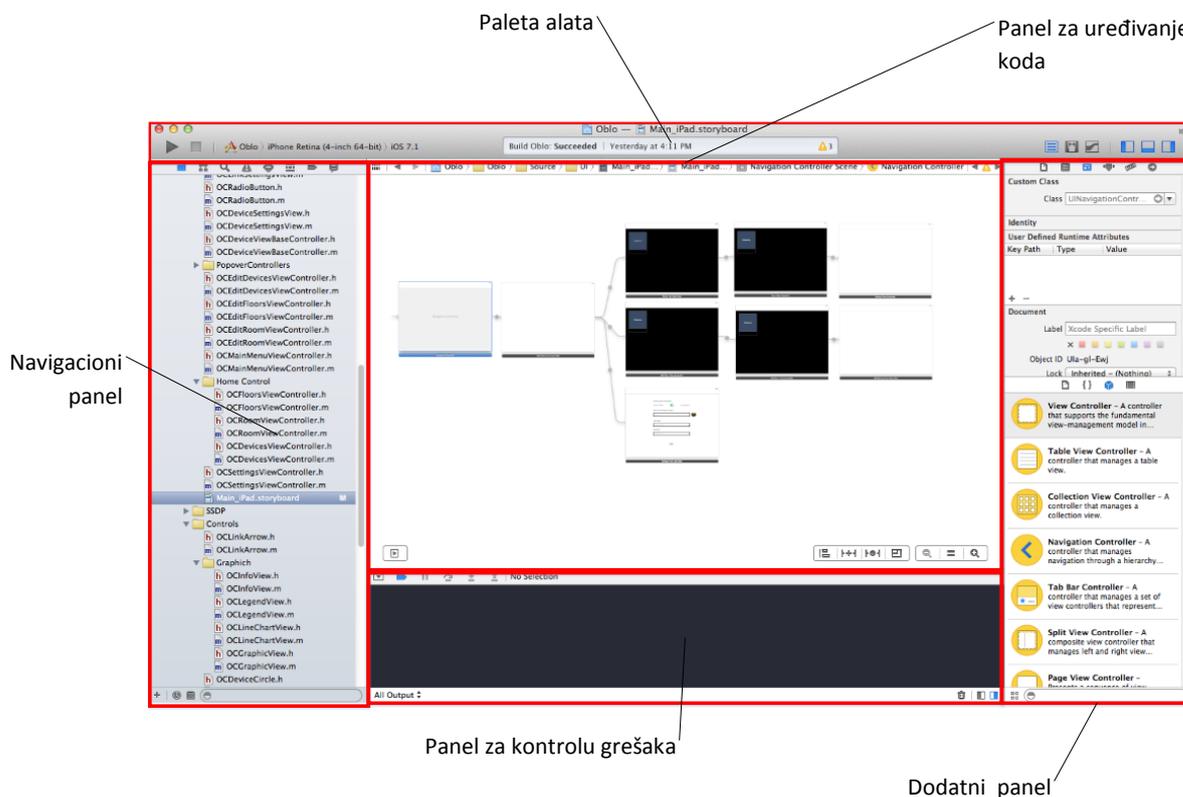
- **Cocoa touch**– sadrži ključna razvojna okruženja za iOS aplikaciju. Ova razvojna okruženja definišu izgled aplikacije. Oni takođe pružaju programsku strukturu i podršku za ključne tehnologije kao što su: rad sa više programa , rukovanje događajima interakcije ekranom osetljivim na dodir, sistemskim događajima i mnoge druge sistemske usluge visokog nivoa.
- **Media**– sadrži audio, video, grafičke tehnologije koje se koriste za realizaciju multimedijalnih aplikacija.
- **Core Service**– sadrži osnovne sistemske servise. Ključni među servisima su: *Core Foundation* i *Foundation* razvojna okruženja koji definišu osnovne tipove koje koriste aplikacije.
- **Core OS** – sadrži funkcije niskog nivoa kao što su rukovanje hardverom, protokolima nižih nivoa ili sertifikatima. Oni čine osnovu na koju se oslanjaju svi drugi slojevi. Uslugama ovog sloja se ne pristupa direktno već posredstvom viših slojeva.

Korisnička sprega za iOS platformu zasniva se na konceptu direktne manipulacije, koristeći više dodira (*engl. multi-touch*) ekrana. Elementi korisničke sprege sastoje se od klizača (*engl. slider*), prekidača (*engl. switch*), dugmića itd. Interakcija sa operativnim sistemom uključuje pokrete kao što su dodir ekrana, prebacivanje (*engl. swipe*) ekrana, od kojih svi imaju specifične definicije unutar konteksta iOS operativnog sistema.

## 2.2 Razvojno okruženje i alat za testiranje

### 2.2.1 Razvojno okruženje

Razvojno okruženje korišćeno za izradu ovog rada naziva se XCode. XCode je integrisano razvojno okruženje koje sadrži sve neophodne softverske pakete za razvoj programa na iOS i OS X operativnim sistemima. Okruženje je razvijeno od strane Apple kompanije i moguće ga je koristiti isključivo na uređajima sa OS X operativnim sistemom. Omogućuje uređivanje koda, podešavanje opcija projekta, podešavanje opcija za prevođenje koda, uređivanje izgleda korisničke sprege korišćenjem koncepta iOS pod nazivom *Storyboard*. XCode okruženje pruža mogućnost pokretanja aplikacije na softverskim uređajima - emulator ili na pravim iOS uređajima. Poslednja stabilna verzija ovog razvojnog okruženja je 5.1 [2]. Primer izgleda razvojnog okruženja prikazano je slici 2.2.



Slika 2.2 Izgled razvojnog okruženja iOS platforme

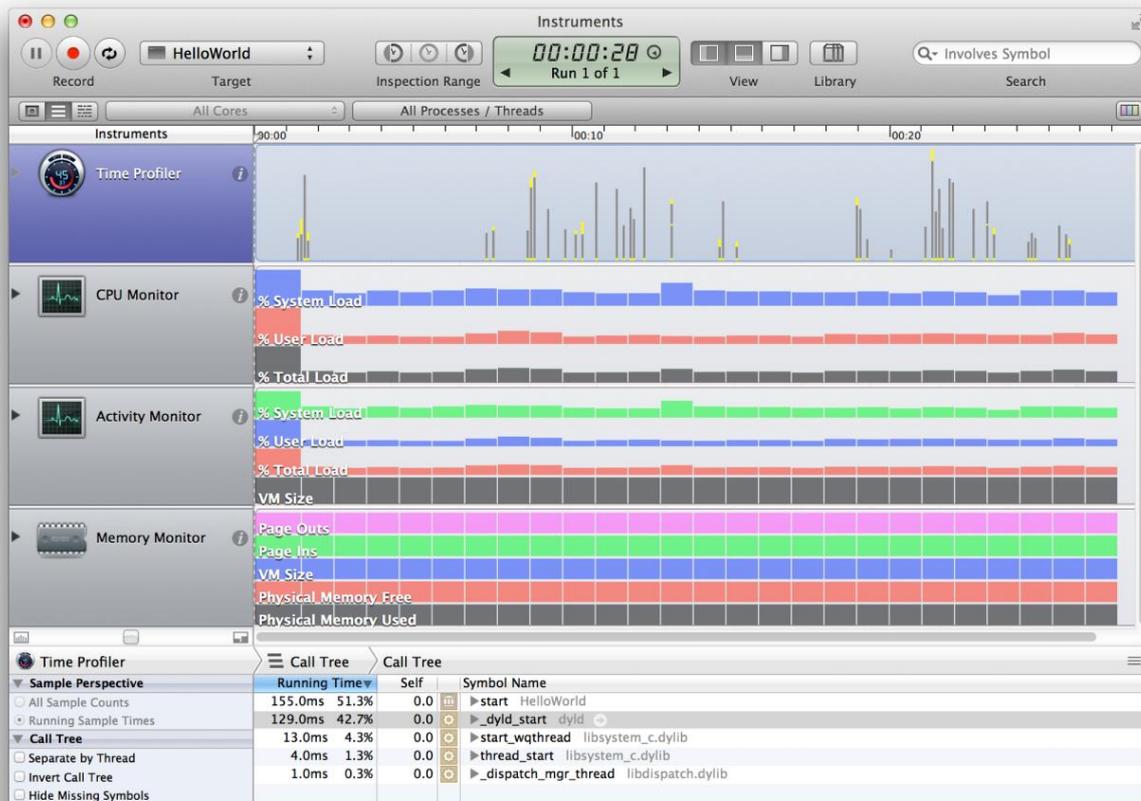
Izgled razvojnog okruženja iOS platforme se sastoji iz sledećih panela:

- Navigacioni panel – postoji više navigacionih panela koji mogu da se prebacuju. Neki od tih panela su: navigator projekta, navigator grešaka, navigator za pretraživanje projekta itd.

- Panel za uređivanje koda – na njemu se vrši pisanje koda kao i grafičko uređivanje izgleda aplikacije. Omogućava uređivanja koda i podešavanje stilova koda.
- Panel za kontrolu grešaka – sastoji se iz dva prozora: prozora za prikaz trenutnih vrednosti promenljivih i prozora konzole koja služi za ispis logova.
- Dodatni panel – daje informacije o datoteci istaknutoj u navigacijskom panelu. Sadrži i grafičke komponente koje mogu da se prevlače u panel za uređivanje koda
- Paleta alata - sadrži alate kao što su: alat za pokretanje, ispitivanje aplikacije, alat za različit prikaz panela za uređivanje koda, alat za izbor uređaja na kome želim da se izvršava aplikacija

## 2.2.2 Alat za testiranje

Alat za testiranje naziva se *Instruments*. *Instruments* pruža mogućnost profilisanja, analiziranja i testiranja rada aplikacija pisanih na iOS platformi i aplikacija pisanih na Mac OS X platformi. *Instruments* ima sposobnost da: ispita ponašanje jednog ili više procesa, snima i pauzira akcije korisnika, omogućava stvaranje niza različitih testova za analizu aplikacije, sačuva šablone rukovanja korisničkom spregom radi kasnije reprodukcije istih. Pomoću *Instruments* alata mogu se izvršiti sledeći vidovi testova: automatski testovi, stres testovi, testovi performansi [3]. Primer izgleda alata prikazan je na slici 2.3

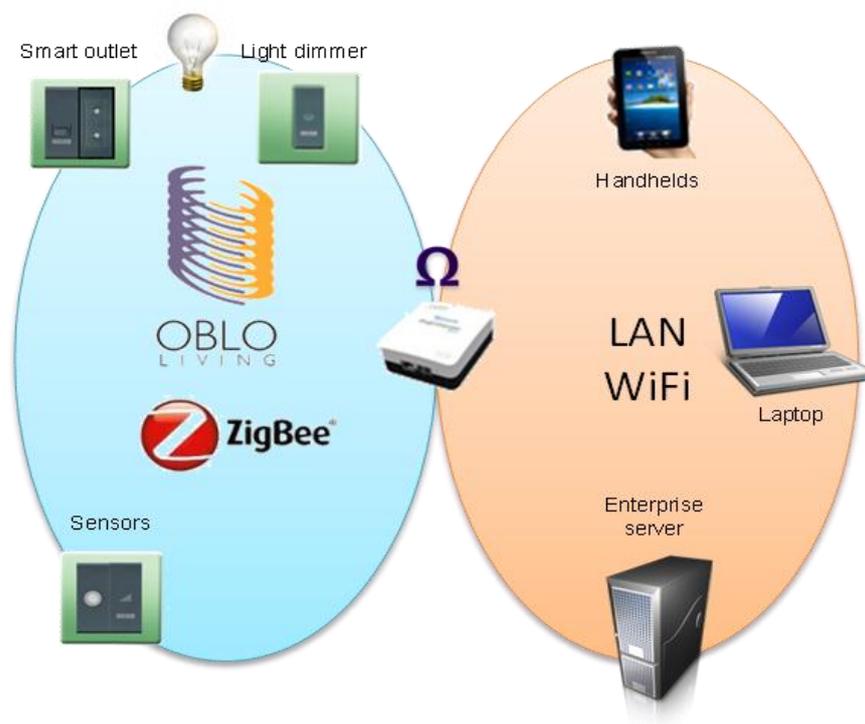


Slika 2.3 Izgled alata *Instruments*

## 2.3 OBLO sistem

Osnovi princip integracije OBLO sistema [4] u domaćinstvo jeste realizacija pametne kuće, u kojoj se preko bežične mreže kontrolišu uređaji priključeni na električnu mrežu. Sistem koristi *Zigbee* [6] 2.4GHz bežičnu mrežu za efikasno upravljanje i konfiguraciju uređaja. Centralni uređaj sistema predstavlja OHM (*engl. OBLO Home Manager*) koji predstavlja sponu između IP (LAN/WiFi) mreže i *Zigbee* uređaja, uz mogućnost da kontroliše sve uređaje u kući. OHM obrađuje naredbe koje prihvata od klijenta, dobavlja podatke o stanju uređaja

OHM služi kao posrednik između klijentskih aplikacija i uređaja. Arhitektura OBLO sistema prikazana je na slici 2.4.



Slika 2.4 Arhitektura OBLO sistema

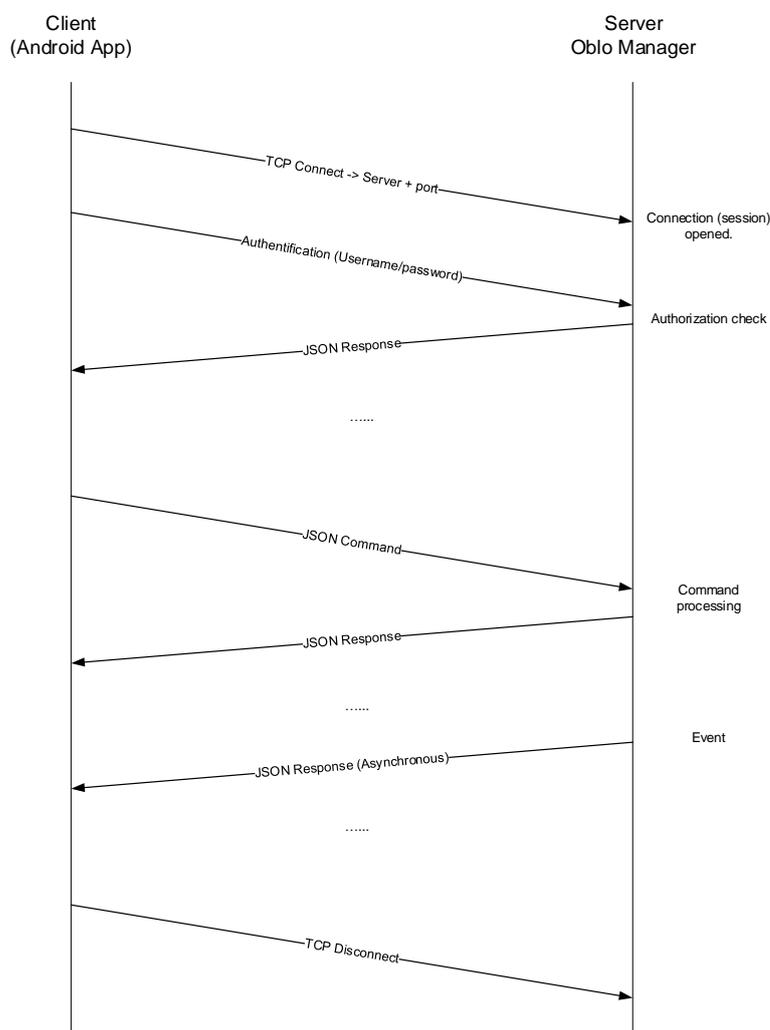
OBLO se sastoji iz nekoliko grupa uređaja koji se zajedno sa OHM-om integrišu u jedinstveni sistem:

- Pametni prekidači
- Pametne utičnice
- Daljinski upravljač
- Senzori pokreta

Kontrola svetala u kombinaciji sa perifernim OBLO modulima nudi fleksibilnost u radu. Moguće je upravljati svetlosnim grupama nezavisno od električnih instalacija. Sam sistem ostavlja

mogućnost integracije sa drugim uređajima/sistemima u kući ili zgradi takođe pružajući fleksibilnost i u njihovom korišćenju. Uređaji koji se prvi put uvode u sistem se mogu integrisati jednostavnom implementacijom programskih modula za njihovo upravljanje.

Unutar OHM je realizovan ručnovalac događajima koji na svaku promenu stanja u okruženju aktivira novi događaj. Programska podrška OHM je realizovan u C++ programskom jeziku. Programsko rešenje je prenosivo na Microsoft Windows i Linux operativne sisteme. Na strani OHM realizovan je TCP kontroler koji rukuje sa potencijalnim klijentima. Poruke koje se razmenjuju između OHM i klijentske aplikacije su formatirane kao JSON objekti. Komandovanje sistemom zahteva autorizaciju korisnika, koja se sastoji od razmene korisničkog imena i šifre. Način komunikacije aplikacije sa OHM prikazan je na slici 2.5.



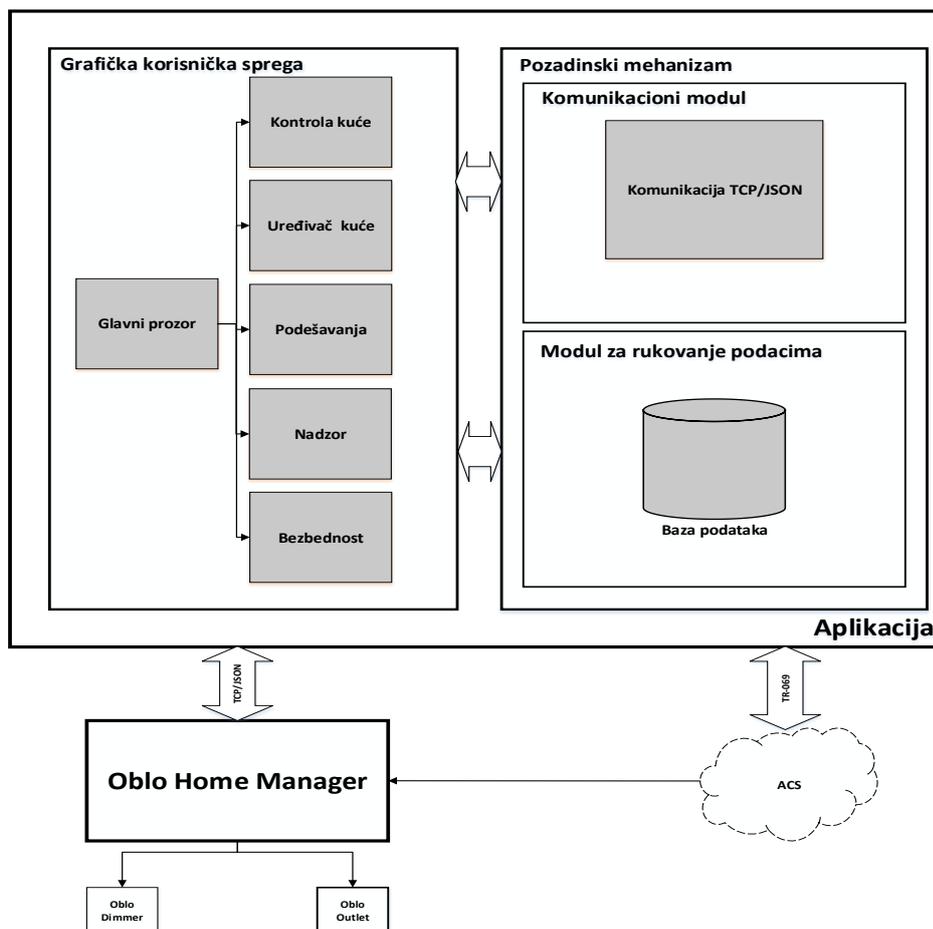
Slika 2.5 Sekvenca razmene podataka tokom komunikacije

Autorizovani klijenti su u mogućnosti da šalju komande i primaju odgovore i informacije o događajima. Ovo je slučaj kada se klijentska aplikacija direktno povezuje na OHM. Ovaj način povezivanja je pogodan za vezu klijenta sa jednim lokalnim OHM-om. Raskidanje TCP veze inicira klijent, a za svako ponovno povezivanje potrebno je izvršiti autorizaciju.

### 3. Koncept rešenja

U ovom poglavlju opisuje se analiza problema i predlog rešenja. Korisnička sprega ima predefinisani izgled koji mora biti sličan izgledu postojeće Android klijentske aplikacije. Izmene se odnose na sam izgled aplikacije zbog specifičnosti samog iOS sistema i uvođenjem nekih novih mogućnosti.

U poglavlju će biti analizirani zahtevi aplikacije sa objašnjenim konceptima realizacije aplikacije. Arhitektura aplikacije prikazana je na slici 3.1.



Slika 3.1 Arhitektura programskog rešenja

---

Sistem se sastoji iz klijentskog i poslužiteljskog dela. Klijentski deo sistema predstavlja iOS aplikacija koja je tema ovog rada, dok poslužilac je OHM koji predstavlja centralni deo OBLO sistema. Klijentski deo sastoji se iz dva glavna modula:

- Grafičke korisničke sprege i
- Pozadinskog mehanizma koji se sastoji od:
  - Komunikacionog modula
  - Modula za rukovanje podacima

### **3.1 Grafička korisnička sprega**

Moduli ove celine implementiraju grafički prikaz aplikacije. Podatke koje koristi aplikacija za prikaz svih grafičkih komponenti se mogu nalaziti na OHM u obliku datoteke ili sačuvane lokalno u bazi podataka. Da bi se dobavili podaci sa OHM potrebno je upotrebiti odgovarajuće komande u komunikaciji sa OHM. Nakon uspešnog dobavljanja datoteke sa OHM podaci iz datoteke se zatim parsiraju i smeštaju u lokalnu bazu podataka. Kada se podaci nalaze u lokalnoj bazi podatke potrebno je samo pročitati te podatke. Sastoji se iz šest pod modula:

- glavnog prozora,
- kontrole kuće,
- uređivača kuće,
- stranice sa podešavanjima,
- bezbednost, i
- nadzor.

Ovi moduli realizovani su korišćenjem grafičkih komponenti iOS platforme. Više značajnih grafičkih komponenti su namenski napravljeni.

#### **3.1.1 Glavni prozor**

Predstavlja ulaznu tačku aplikacije. U njemu je potrebno izvršiti konekciju sa OHM. Takođe potrebno je proveriti da li aplikacija ima pristup mreži putem WiFi. U slučaju da aplikacija nema pristup mreži, korisnika treba obavestiti o tome i upozoriti ga da za dalji rad sa aplikacijom mora da pristupi mreži. Na navigacionom panelu u gornjem desnom uglu treba da se nalazi meni koji sadrži opcije: *Connect, About, Quit*.

#### **3.1.2 Podešavanja**

Modul omogućava podešavanje parametara rada aplikacije. Parametri koji trebaju biti dostupni za podešavanje su: unos korisničkog imena i šifre potrebnih za autorizovanje na OHM, izbor načina konekcije prema OHM, mogućnost prikaza dostupnih OHM na mreži.

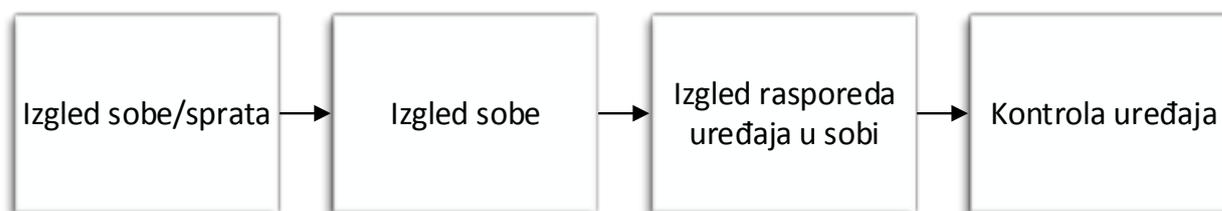
Postoje dva načina konekcije prema OHM a to su:

- konekcija putem HTTP protokola na ACS (*engl. Auto Configuration Server*).
- konekcija putem TCP protokola na OHM koji se nalazi u lokalnoj mreži.

Kada je u pitanju konekcija putem TCP protokola na OHM koji se nalazi u lokalnoj mreži potrebno je pomoću SSDP protokola pronaći OHM u lokalnoj mreži. Preko SSDP protokola se šalju poruke određenog formata ka svim uređajima na mreži, uređaji kada prime poruku šalju povratnu poruku preko koje se identifikuju. Aplikacija treba da bude sposobna da prepozna samo OHM uređaje i da pribavi korisne podatke koje će smesti u odgovarajuću listu iz koje korisnik može izabrati njemu odgovarajući OHM. Filtriranje samo OHM uređaja se vrši na osnovu njihovih IP adresa dobijenih u povratnoj poruci: ukoliko IP adresa ne postoji u listi OHM uređaja onda se dodaje novi OHM uređaj u suprotnom OHM uređaj se ne dodaje u listu. Sam SSDP mehanizam treba realizovati kao posebna nit da ne bi došlo do blokiranja rada aplikacije.

### 3.1.3 Kontrola kuće

Namenjen je kontroli uređaja i kompletan prikaz izgleda pametne kuće. Podaci koji se koriste za izgled kompletnog modula dobijaju se od strane OHM ili iz lokalne baze podataka.



Slika 3.2 Izgledi kontrole sobe

Sastoji se iz sledećih izgleda ekrana:

- **Izgled sobe/sprata** – služi za prikaz svih postojećih soba u pametnoj kući. Podaci koje treba prikazati su informacije o položaju soba po spratovima. Prikaz svih informacija treba da bude kolekcijski organizovan zbog lakšeg korišćenja i manipulacije sa samim izgledima.
- **Izgled sobe** – služi za prikaz niza pogleda odabrane sobe. Organizovan je takođe kao kolekcija. Informacije koje treba prikazati su nazivi pogleda na sobe i slike dodeljene pogledima sobe.
- **Izgled rasporeda uređaja u sobi** – preko celog ekrana treba prikazati sliku odabranog pogleda na sobu. Na određenim pozicijama potrebno je napraviti namenske kontrole koje označavaju poziciju uređaja u sobi. Potrebno je omogućiti lako kretanje kroz poglede sobe pomoću pomeranja pogleda ekrana klizanjem prsta levo ili desno po ekranu. Iz razloga lakšeg kretanje kroz ekrane potrebno je koristiti veze ka odabranom pogledu sobe. Veze su u obliku strelice koje se nalaze na

određenom mestu na ekranu i predstavljaju prečice koje povezuju različite poglede soba.

- **Kontrola uređaja** – treba da omogući kontrolu svih dostupnih uređaja na OHM. Treba da prikaže jedan prozor koji na sebi mora imati mogućnost kontrole uređaja zajedno sa statusom odabranog uređaja. Uređaji koji pružaju kontrolu potrošnje energije potrebno je da pored prikaza trenutne potrošnje energije grafički prikazuju potrošnju tokom vremena.

### 3.1.4 Uređivač kuće

Arhitektura modula je slična arhitekturi modula za kontrolu kuće. Modul deli i nadograđuje izgled i funkcionalnosti prethodno opisanog modula. Ovaj modul pruža mogućnost korisniku da konfiguriše izgled svoje pametne kuće, kao i izmenu već postojećeg izgleda. Proširenja koja treba dodati na već postojeće izgled u modulu za kontrolu kuće su :

- U izgled sobe/sprata treba dodati sledeće mogućnosti : izmenu već postojećeg imena sobe, skidanje konfiguracije pametne kuće sa OHM kao i slanje postojeće konfiguracije prisutne na aplikaciji
- U izgled sobe treba dodati sledeće mogućnosti: izmenu već postojećeg pogleda sobe, dodavanje novog pogleda sobe, brisanje već postojećeg pogleda. Kod dodavanja novog pogleda sobe treba omogućiti korisniku da unese naziv pogleda i izbor slike za pogled sobe. Treba omogućiti dva načina izbora slike: upotrebom kamere na iOS uređaju i korišćenje već postojeće slike iz galerije slika na iOS uređaju
- U izgledu rasporeda uređaja u sobi treba dodati sledeće mogućnosti: dodavanje novih uređaja u sobu kao i novih veza, promenu parametara već postojećih uređaja i veza i promena pozicije već postojećih uređaja i veza. Za dodavanje uređaja treba da se pojavi novi prozor koji treba da pruži korisniku: izbor tipa uređaja na osnovu koga treba da prikaže sve moguće identifikacione brojeve uređaja, izbor željene slike koja se dodeljuje uređaju. Za dodavanje novih veza treba da se pojavi prozor koji korisniku treba da pruži: izbor položaja veze (gore, dole, levo, desno), izbor pogleda sobe na koji želi da se pređe

### 3.1.5 Nadzor

Predstavlja modul za nadzor prostorije uz pomoć uređaja OHM, tj. nadzor se vrši uz pomoć *Zigbee* komunikacije sa OHM. Ovoj modul nije realizovan u ovom rešenju aplikacije.

---

### **3.1.6 Bezbednost**

Pružna mogućnost nadzora kuće uz pomoć IP kamera priključenih na kućnu mrežu. Ovaj modul nije realizovan u ovom rešenju aplikacije.

## **3.2 Pozadinski mehanizmi**

Pozadinske mehanizme čine dva osnovna modula: modul za rukovanje podacima i komunikacioni modul. Ovi moduli omogućuju pravilan rad same aplikacije i kao takvi predstavljaju osnovu kompletne aplikacije.

### **3.2.1 Modul za rukovanje podacima**

Modul treba da obezbedi čuvanje podataka o konfiguraciji kuće dobijenih od strane OHM, ti podaci su: podaci o sobi, podaci o spratu, podaci o uređajima u sobi, podaci o pogledima na sobe. Modul je potrebno proširiti sa podacima o vezama u sobama. Modul treba da omogući aplikaciji laku manipulaciju za čuvanje i korišćenje podataka.

### **3.2.2 Komunikacioni modul**

Predstavlja poseban modul koji služi za komunikaciju sa OHM. Komunikacija se vrši stalnim TCP komunikacionim kanalom, pa zbog toga sama komunikacija mora biti realizovana kao asinhroni zadatak. Aplikacija sa OHM razmenjuje poruke koje su formatirane kao JSON. Izgled jedne poruke zavisi od komande koju je potrebno poslati. Zbog velike količine različitih komandi potrebno je modularizovati implementaciju komandi, tj. svaka komanda treba da biti realizovana u okviru svoje klase koja nasleđuje jednu osnovnu klasu. Komunikacioni modul će znati da radi samo sa osnovnom klasom.

## 4. Programsko rešenje

Za realizaciju iOS aplikacije za kontrolu pametnih kuća korišćeno je razvojno okruženje XCode sa iOS SDK (*engl. Software Development Kit*) alatom. Programski jezik korišćen za realizaciju ovog rešenja je *Objective-C* [7].

### 4.1 Realizacija modula za komunikaciju

Glavni deo ovog modula predstavlja komunikacioni modul. Klasa koja realizuje ovaj modul naziva se *OCCommunicationManager*. Modul je dostupan iz svih modula aplikacije. Kada je neki modul aplikacije želi da pošalje komandu OHM on kreira objekat željene komande i prosleđuje ga komunikacionom modulu. Svaka komanda je implementirana kao posebna klasa. Neke od primera su: *OCDownloadSetup*, *OCUploadSetup*, *OCGetDevicePropertyValue*, *OCGetDeviceStatus*, itd. Svaka od ovih klasa treba da implementira različite komande u vidu poruke u JSON formatu. Jedna od JSON poruka prikazana je ispod:

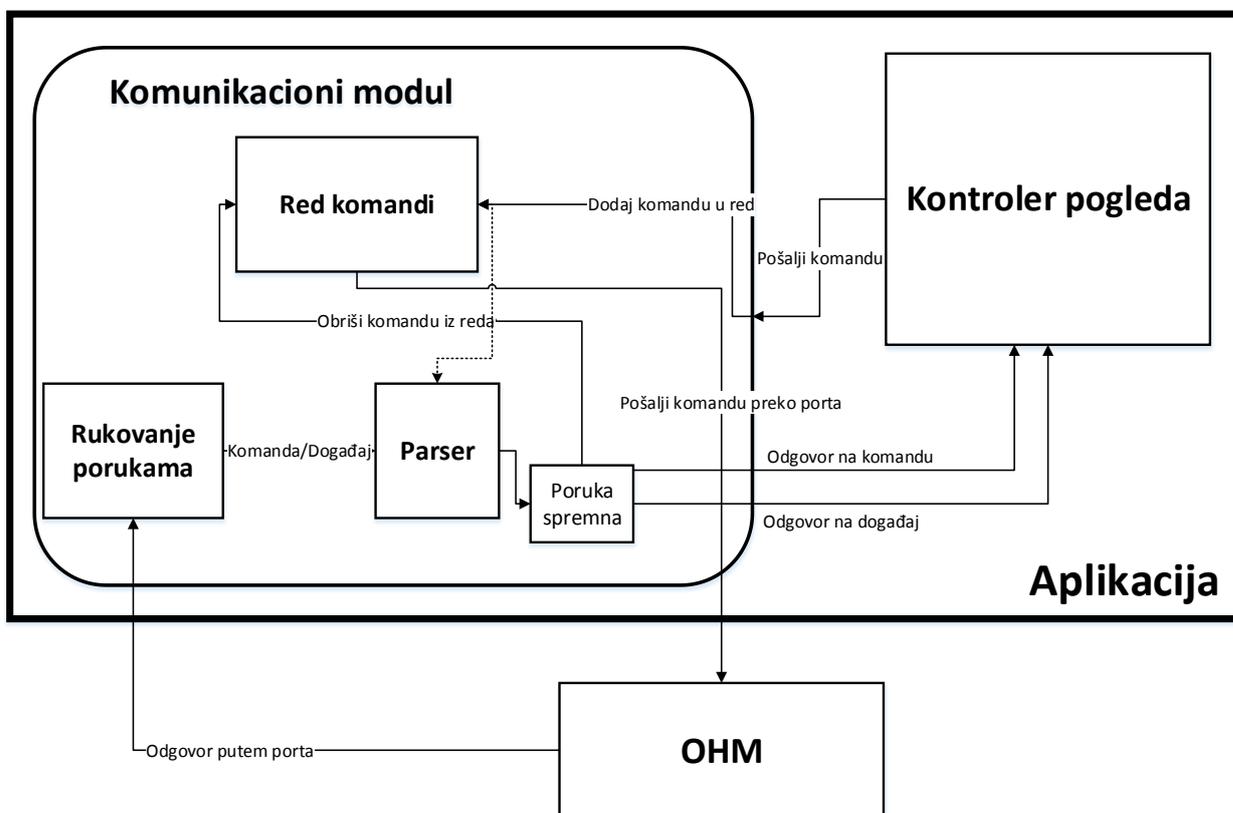
```
{
  "msg_type" : "command",
  "command": "get_device_status",
  "device_id" : 13
}
```

Koraci koje komunikacioni modul mora da obavi nakon prosleđene komande su:

- obrati se svom pod modulu **Red Komandi** koji objekat komande smesti u red i koji šalje poruku preko porta OHM-u u JSON formatu.
- OHM obradi poslatu poruku i na isti port vrati odgovor klijentskoj aplikaciji.

- **Modul za rukovanje porukama** vrši razvrstavanje poruke primljene od strane OHM. Ako je dobijena poruka u JSON formatu prosleđuju se modulu **Parser** a u suprotnom poruka se odbacuje.
- **Parser** modul vrši obradu prosleđene poruke. Vrsta obrade odlučuje se pristupom objektu sačuvanom u modulu Red Komandi.
- Obradena poruka se prosledi modulu **Poruka Spremna** koji preko funkcije *odgovor na komandu* obavesti trenutno aktivan kontroler pogleda o statusu poslate poruke. Poruka koja je uspešno obradena briše se iz reda poslatih poruka. Ova funkcija je delegat funkcija čija je uloga obaveštavanje kontrolera pogleda da se neki događaj desio u nekom drugom modulu. Da bi kontroler pogleda mogao da dobije obaveštenja od komunikacionog modula mora da implementira ovu funkciju.

Realizacija komunikacionog modula prikazana je na slici 4.1.



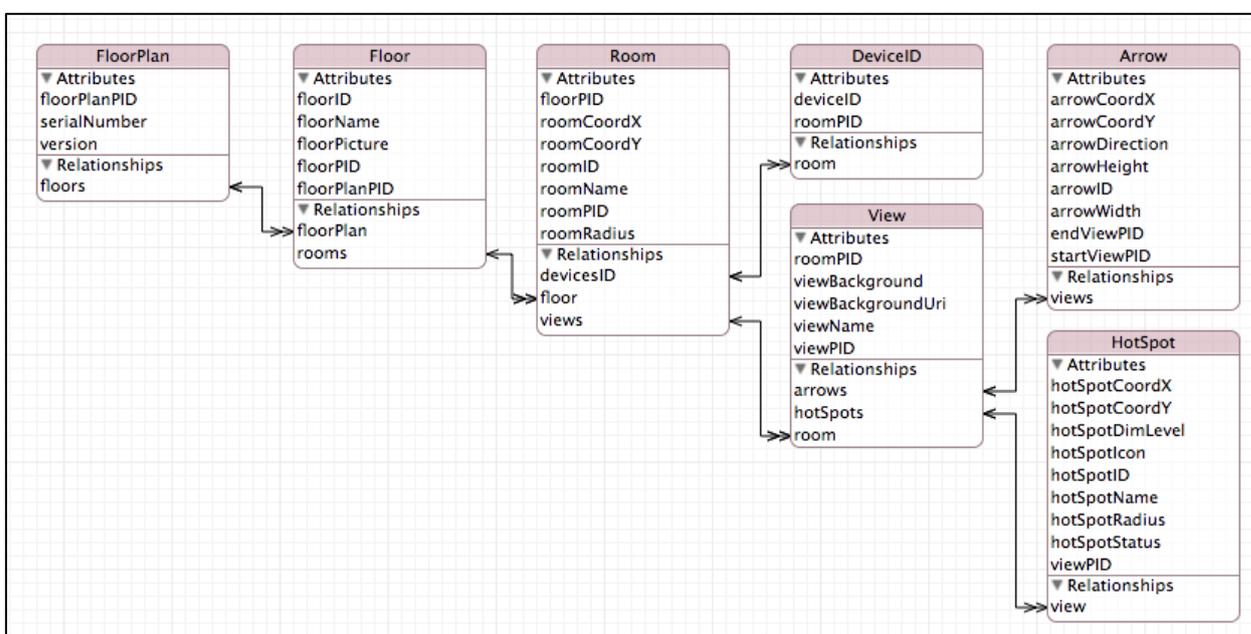
Slika 4.1 Realizacija komunikacionog modula

Od strane OHM mogu da se šalju poruke u obliku događaja. U tom slučaju komunikacioni modul pomoću svog pod modula za rukovanje porukama dobavi poruku. Prezetu poruku prosledi modulu za parsiranje koji tu poruku odradi. Obradena poruka se prosledi modulu Poruka Spremna koji obaveštava aktivan kontroler pregleda pomoću delegat funkcije *odgovor na događaj*. Da bi kontroler pregleda mogao da dobije poruku mora da implementira ovu delegat funkciju.

Kada bi trebalo da se doda nova poruka potrebno je jednostavno napraviti klasu koja će da implementirati potrebnu komandu.

## 4.2 Realizaciju modula za čuvanje podataka

Za realizaciju ovog modula korišćena je iOS tehnologija pod nazivom Core Data, a XCode razvojno okruženje pruža vizuelne alate za pravljenje modela podataka i klasa neophodnih za rukovanje tim podacima. Ova tehnologija se koristi iz više razloga, i to: 50% - 70% manji broj linija koda, jednostavna implementacija, mogućnost grafičkog prikaza modela baze podataka, optimizovano da koristi manje memorije od svih postojećih rešenja dostupnih za iOS platformu. Model baze podataka korišćen za ovu aplikaciju prikazan je na slici 4.2



Slika 4.2 Izgled modela baze podataka

Model se sastoji od sedam tabela:

- *FloorPlan* tabela – koristi se za smeštanje podataka o planu kuće. Klasa koja odgovara ovoj tabeli naziva se *OCFloorPlan*.
- *Floor* tabela – služi za smeštanje svih potrebnih parametara o spratu. Klasa koja odgovara ovoj tabeli naziva se *OCFloor*.
- *Room* tabela – služi za smeštanje potrebnih podataka o sobi. Klasa koja odgovara ovoj tabeli naziva se *OCRoom*.
- *DeviceID* tabela – služi za smeštanje identifikacionog broja uređaja za povezivanje. Klasa koja odgovara ovoj tabeli naziva se *OCDeviceID*.
- *View* tabela – služi za smeštanje podataka o jednom pogledu na sobu. Klasa koja odgovara ovoj tabeli naziva se *OCView*.

- *Arrow* tabela – služi za smeštanje podataka o strelici. Klasa koja odgovara ovoj tabeli naziva se *OCArrow*.
- *Hotspot* tabela – služi za čuvanje podataka o uređajima. Klasa koja odgovara ovoj tabeli naziva se *OCHotSpot*.

Neke od osnovnih funkcija ovog modula su :

- (void) *addHotSpot(OCHotSpot \*)hotspot forSelectedViewPID: (NSString \*)viewPID*

- **Opis:** Dodavanje novog uređaja u bazu
- **Parametri:**
  - *hotSpot* - objekat klase koji se dodaje u bazu. Sadrži sledeća polja: *hotspotID, hotSpotCoordX, hotspotCoordY, hotspotRadius, hotSpotIcon, hotSpotName*
  - *viewPID* - identifikacioni parametar pogleda

- (void) *deleteHotSpot(OCHotSpot \*)hotSpotForDelete*

- **Opis:** brisanje željenog uređaja iz baze
- **Parametar:** *hotSpotForDelete* – objekat uređaja za brisanje. Sadrži sledeće parametre: *hotSpotCoordx, hotSpotCoordY, hotspotRadius, hotSpotName, hotSpotIcon, hotSpotID*

- (NSArray \*) *getHotSpotsForViewPID: (NSString \*)viewPID*

- **Opis:** dobijanje uređaja na osnovu identifikacionog parametra pogleda sobe
- **Parametar:** *viewPID* – identifikacioni parametar pogleda
- **Povratna vrednost:** niz uređaja u željenom pogledu sobe

- (void) *addArrow: (OCLinkArrow \*)arrow*

- **Opis:** dodavanje nove strelice u bazu podataka
- **Parametar:** *arrow* - objekat klase *OCLinkArrow* koji sadrži sledeće parametre: *direction, currentViewID, nextViewID, frame, id*

- (NSArray \*) *updateArrowPosition: (OCLinkArrow \*)movedArrow*

- **Opis:** ažuriranje pozicije selektovane veze
- **Parametar:** *movedArrow* – objekat klase *OCLinkArrow*

- (NSArray \*) *getArrowsForViewPID: (NSString \*) viewPID*

- **Opis:** dobijanje željene veze iz baze na osnovu identifikacionog parametra pogleda
- **Parametar:** *viewPID* – identifikacioni parametar pogleda sobe
- **Povratna vrednost:** niz veza u željenom pogledu sobe

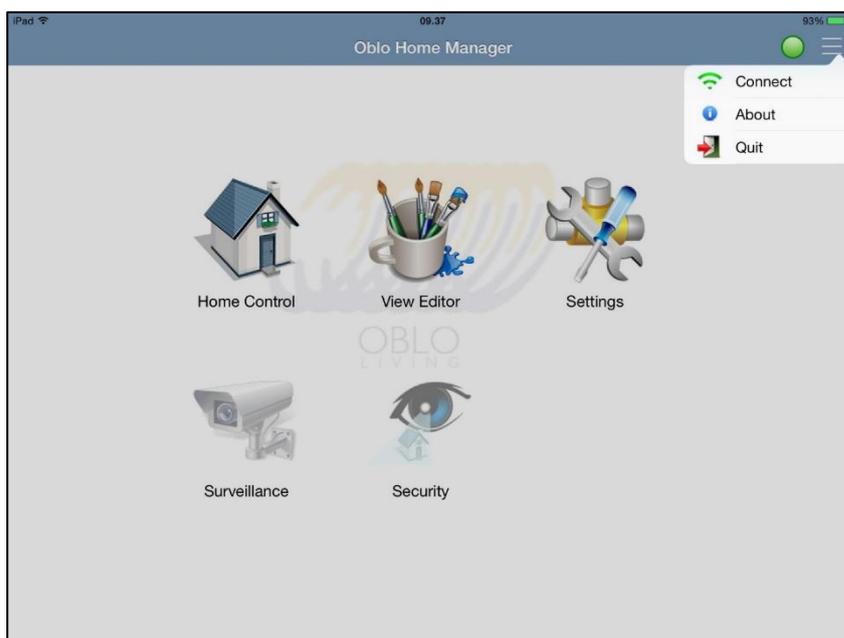
## 4.3 Realizacija grafičke korisničke sprege

Interakcija između korisnika i iOS aplikacije vrši se preko elementa iOS aplikacije koji se naziva kontrolor prozora (*engl. UIViewController*) koji ima kompleksni životni vek. Svim objektima u grafičkoj sprezi koji se nalaze u kontroleru prozora može se pristupiti tek kada je napravljen, a postoje sve dok je živ. Kontroleri prozora su međusobno povezani putem kontrolera navigacije (*engl UINavigationController*).

Zbog specifičnosti iOS uređaja koji imaju samo jedno fizičko dugme za izlaz iz aplikacije, mora se realizovati kretanje kroz kontrolere prozora primenom navigacionog kontrolera. Svaki kontroler prozora mora posedovati navigacionu traku na vrh ekrana. Navigacioni kontroler implementira dugme za povratka na prethodni kontroler. Navigacioni kontroler je dopunjen sa indikatorom konekcije koji predstavlja stanje OHM (da li je OHM dostupan ili nedostupan).

### 4.3.1 Realizacija glavnog prozora

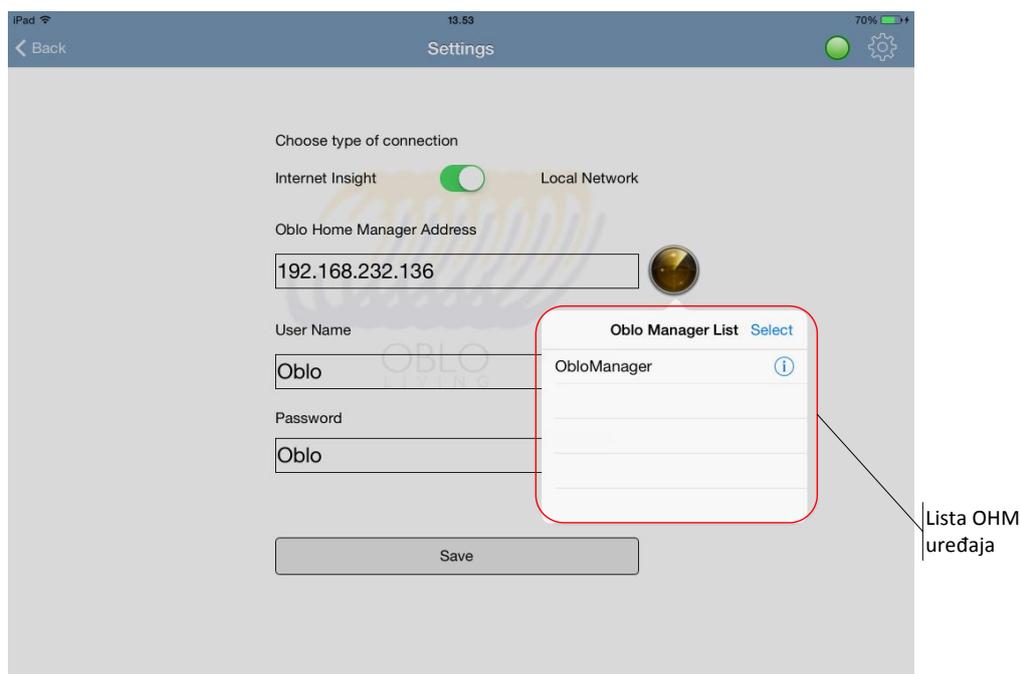
Izgled glavnog ekrana prikazan je slici 4.3. Sastoji se od pet dugmadi (*Home Control, View Editor, Settings, Surveillance, Security*). Pritiskom na određene dugme prelazi se u odgovarajući kontroler. Takođe postoji i dugme na navigacionoj traci koje kada se klikne prikazuje listu opcija: *Connect, About, Quit*. *Connect* opcija inicira uspostavljanje veze sa OHM otvaranjem utičnice (*engl. socket*) i zatim slanjem komande za autorizaciju. *About* opcija prikazuje osnovne informacije o verziji i proizvođaču aplikacije, dijalog koji se pojavljuje prilikom klika na ovu opciju je ugrađena komponenta iOS platforme pod nazivom *UIAlertDialog*. Opcija *Quit* pruža mogućnost korisniku da izađe iz aplikacije. Klasa koja implementira ovaj prozor se naziva *OCMainViewController*.



Slika 4.3 Izgled glavnog prozora

### 4.3.2 Realizacija modula podešavanja

Izgled modula prikazan je na slici 4.4.



Slika 4.4 Izgled prozora za podešavanje aplikacije

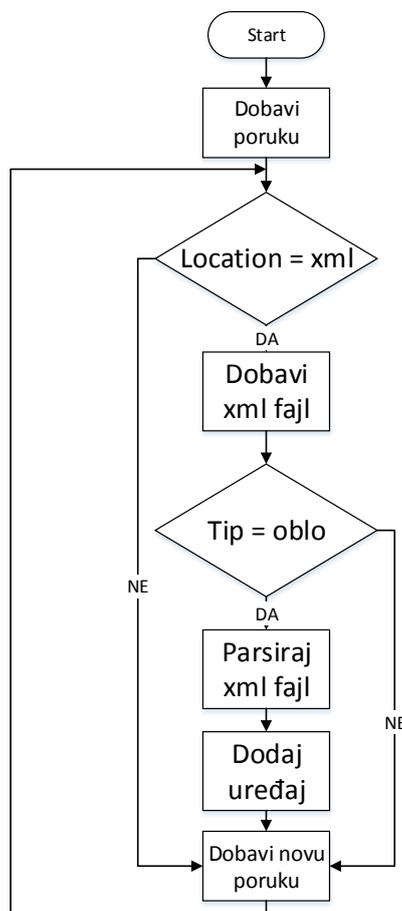
Realizacija ovog modula sastoji se iz dve celine: glavni prikaz i prozor za prikaz liste OHM.

Lista OHM uređaja je realizovana kao posebna klasa pod nazivom *OCObloInfoTableViewController* koja prikazuje listu OHM uređaja. Realizovana je uz pomoć komponente iOS platforme pod nazivom *UIPopoverController*. Info dugme daje korisniku detaljne informacije o svakom OHM. Podaci u info prozoru su informacije o samom OHM uređaju koji se nalaze u xml datoteci dobijenoj preko SSDP protokola.

Glavni prikaz – sastoji se iz sledećih grafičkih komponenti:

- prekidač (*engl. switch*) koji obezbeđuje biranje konekcije sa OHM
- polje za unos IP adrese OHM – ovo polje se popunjava IP adresom OHM koja se dobija klikom na element liste u Lista OHM uređaja.
- polje za unos korisničkog imena – polje služi za unos korisničkog imena potrebnog za autorizaciju na OHM.
- polje za unos šifre – polje služi za unos šifre potrebne za autorizaciju na OHM.
- dugme *Save* – klikom na ovo dugme svi prethodno uneti podaci se čuvaju u *NSUserDefaults*. *NSUserDefaults* je ugrađena klasa iOS platforme koja služi za čuvanje podataka u sistemsku bazu podataka.

SSDP protokol je realizovan kao poseban modul a koristi se u okviru ovog modula kao pozadinski modul. SSDP protokol je realizovan kao posebna klasa *OCSSDPCommunication*. Blok dijagram funkcionisanja SSDP modula prikazan je na slici 4.5.



Slika 4.5 Blok dijagram funkcionisanja SSDP protokola

Na početku rada pošalje se poruka koja je namenjena svim uređajima na mreži. Nakon toga dobavi se poruka koja je poslata od strane nekog od uređaja na mreži. Dobljena poruka se parsira i na osnovu dobijene informacije se proverava da li ona u sebi sadrži polje *Location* koje sadrži URL adresu. URL adresa sadrži adresu putanje do xml datoteke koja predstavlja opis uređaja i koja sadrži polja koja su definisana SSDP protokolom [9]. Bitna polja iz xml datoteke su: *friendlyName*, *manufacturer*, *manufacturer url*, *serial number*, *modelName*. Ako polje *Location* postoji onda se dobavi xml datoteka u suprotnom se dobavi sledeća poruka poslata od strane nekog uređaja sa mreže. Ako je uspešno dobavljena xml datoteka onda se ona parsira i na osnovu dobijenih informacija se proverava da li postoji polje koje sadrži ime OBLO, ukoliko to polje postoji dodaje se novi uređaj u listu u suprotnom prelazi se na dobavlja nova poruka poslate od strane drugog uređaja sa mreže. Kompletan ovaj proces se ponavlja periodično dokle god je korisnik u prozoru za podešavanje.

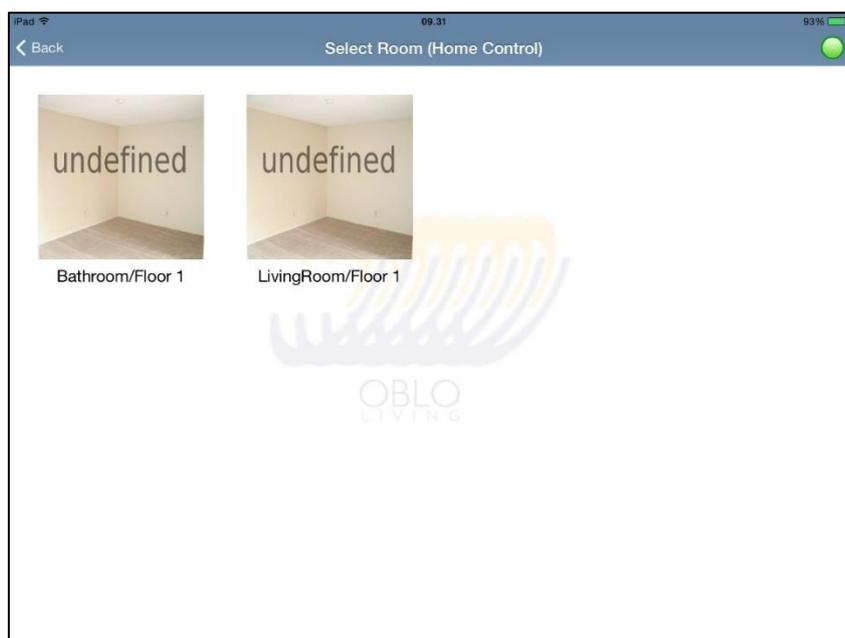
### 4.3.3 Realizacija kontrole sobe

Realizacija ovog modula sastoji se iz sledećih celina:

**Realizacija izgleda sobe/sprata** - izgled je realizovan kao kolekcija korišćenjem ugrađene komponente iOS sistema koja se naziva *UICollectionViewController* koja omogućava koleksijski prikaz podataka. Ova komponenta je dostupna na iOS uređajima verzije 6.0 i novijih. Klasa koja implementira ovaj izgled se naziva *OCFloorViewController*. Podaci koji su potrebni za ovaj izgled su sledeći:

- Podaci o spratovima – dobijaju se koristeći funkciju iz modula za čuvanje podataka - *(NSArray \*) getFloors*. Funkcija vraća niz objekata klase *OCFloor* koja u sebi sadrži podatke o spratu: ime, identifikacioni broj, slika sprata i jedinstveni string sprata. Iz niza objekata uzme se slika i ime sprata koji se prikazuju na ekran
- Podaci o sobama – na osnovu jedinstvenog stringa sprata funkcija iz modula za čuvanje podataka - *(NSArray\*) getRoomForSelectedFloorPID: (NSString \*)floorPID* koja dobavi sve sobe raspoređene po spratovima. Ova funkcija vraća niz objekata klase *OCRoom* koja sadrži informacije o sobi: ime, identifikacioni broj, jedinstveni string sobe. Iz niz objekata uzima se ime sobe koji se pridružuje imenu sprata i zajedno se prikazuju na ekranu.

Izgled je prikazan na slici 4.6.



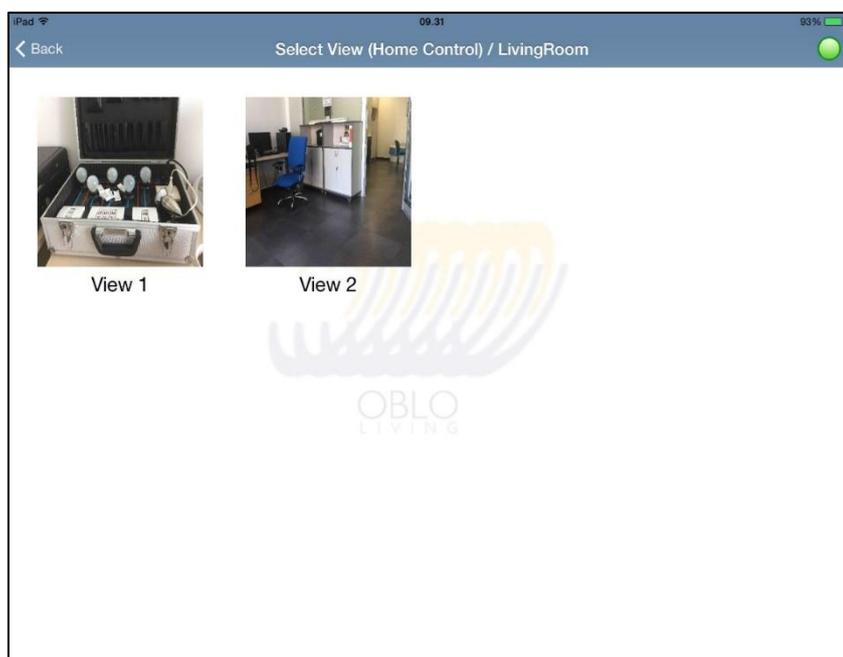
Slika 4.6 Izgled sobe/sprata (Kontrola sobe)

**Realizacije izgleda sobe** – realizovan je kao kolekcija koristeći prethodno navedenu komponentu iOS sistema. Klasa koja implementira ovaj izgled naziva se *OCRoomViewController*. Podaci potrebi za ovaj izgled su :

- Slika pogleda sobe
- Ime pogleda sobe

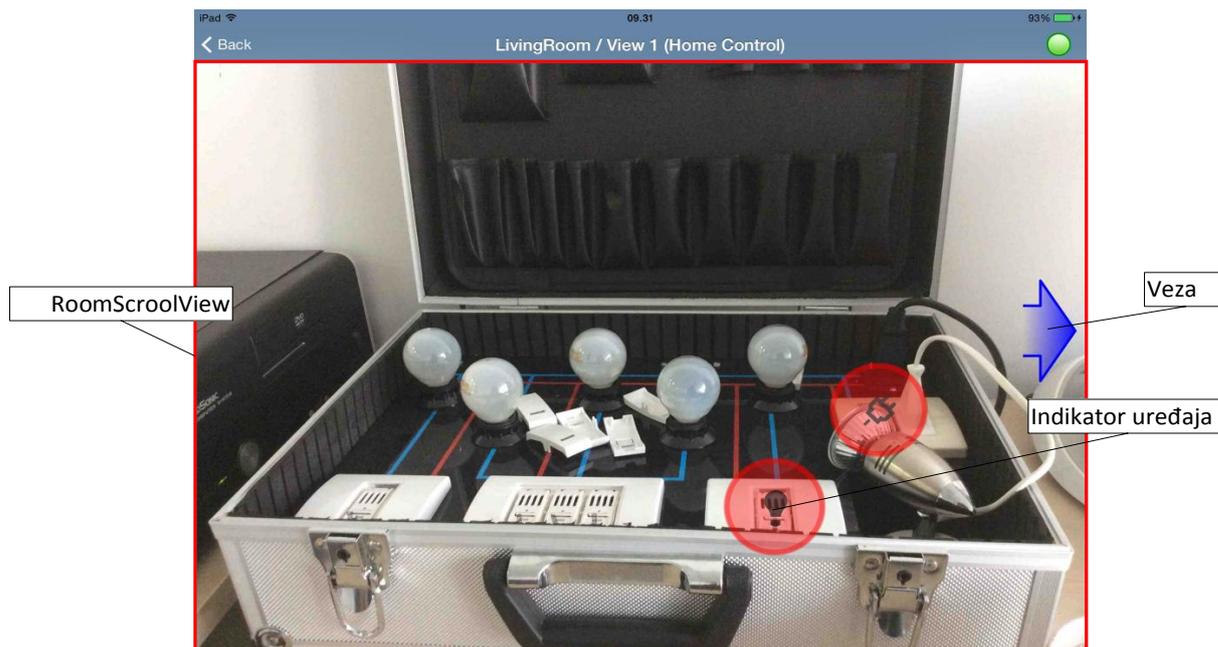
Ovi podaci dobijaju se iz modula za čuvanje podataka koristeći funkciju (*NSArray \**) *getViewsForRoomPID: (NSString \*)roomPID*. Funkcija vraća niz objekata klase *OCView* koja sadrži polja: ime pogleda, ime slike, uri slike. Uri slike predstavlja binarnu predstavu slike. Iz niza objekata uzimaju se podaci o imenu pogleda i uri slike koji se proslede komponenti *UICollectionViewViewController* koji zna da prikaže te podatke.

Izgled je prikazan na slici 4.7.



Slika 4.7 Izgled sobe (Kontrola sobe)

Realizacija izgleda rasporeda uređaja u sobi – Izgled je prikazan na slici 4.8.



Slika 4.8 Izgled rasporeda uređaja u sobi

Izgled se sastoji iz sledećih komponenti:

- *RoomScroolView* – Izveden je iz klase *UIScrollView*. Kao pozadinu koristi se slika pogleda sobe. Pruža mogućnost lakšeg kretanja kroz poglede sobe pomeranjem prsta levo/desno po ekranu. Na njemu se odjednom crtaju sve slike dostupnih pogleda sobe kao i sve veze i indikatori uređaja. Na ekranu je vidljiv samo odabrani pogled sobe
- *Veza* – služi za povezivanje različitih pogleda sobe. Klikom na nju prelazi se na odgovarajući pogled sobe. Klasa kojom je realizovana ova komponenta se naziva *OCLinkArrow*. Ova klasa sadrži sledeća polja:
  - *x, y* - koordinata strelice
  - *nextViewID* – identifikacioni broj pogleda na koje veza pokazuje
  - *direction* – orijentacija veze
  - *currentViewID* – identifikacioni broj pogleda na kome se veza nalazi
 Podaci koji definišu vezu čuvaju se u modulu podataka i mogu se dobiti koristeći funkciju *getArrowsForViewPID: (NSString \*) viewPID*
- *Indikator uređaja* - predstavlja poziciju uređaja u sobi. Klasa kojom je realizovana ova komponenta naziva se *OCDeviceIndicator*. Ova klasa sadrži sledeća polja:
  - *deviceName* – ime uređaja
  - *deviceID* – jedinstveni broj uređaja
  - *powerValue* – ovo polje se u slučaju uređaja za kontrolu svetla koristi za nivo osvetljaja a za smartOutlet uređaje za vrednost izmerene snage

- *myRect* – predstavlja okvir za crtanje indikatora
- *drawState* – predstavlja stanje indikatora (selektovan ili ne)
- *deviceIsOn* – stanje odabranog uređaja

Podaci koji definišu indikator uređaja čuvaju se u modulu podataka i mogu se dobiti koristeći funkciju *getHotSpotsForViewPID: (NSString \*)viewPID*

Klasa koja realizuje ovaj izgled se naziva *OCDeviceViewController*.

**Realizacija kontrole uređaja** – OHM trenutno podržava dva različita tipa uređaja: OBLO Dimmer i OBLO SmartOutlet [5]. OBLO Dimmer ima mogućnosti menjanja osvetljaja, dok Oblo SmartOutlet ima mogućnost merenja potrošnje energije. Za svaki tip uređaja se prikazuje odgovarajući panel.



Slika 4.9 Izgled panela za kontrolu SmartOutlet uređaja

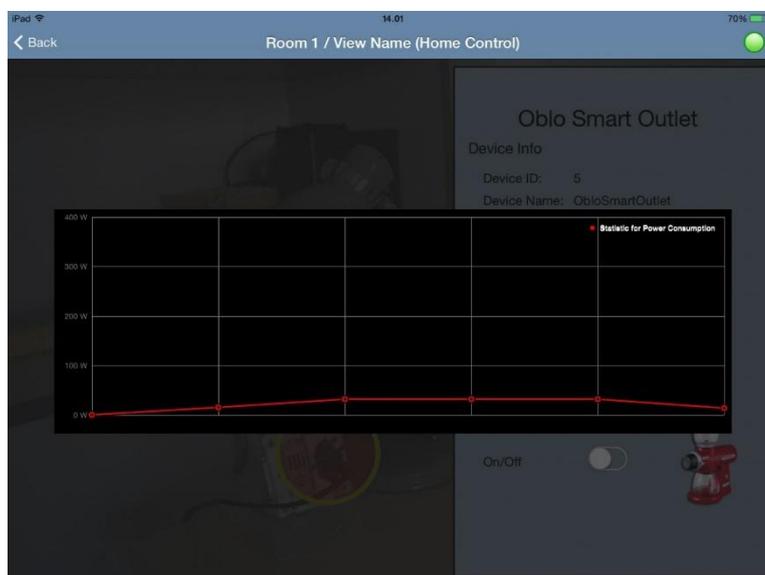
Izgled panela za prikaz SmartOutlet uređaja se može videti na slici 4.9. Na panelu se nalaze sledeće komponente:

- *deviceID* – predstavlja identifikacioni broj uređaja
- *deviceName* – predstavlja ime uređaja
- prekidač (*engl. switch*)– daje mogućnost uključivanja /isključivanja uređaja
- merač – koji daje informacije od trenutnoj potrošnji energije i predstavlja namensku kontrolu grafičke korisničke sprege
- slika – koja predstavlja status uređaja (uključen/isključen)

Klasa koja implementira ovaj panel naziva se *OCOutletView*. Podaci za *deviceID* i *deviceName* se dobavljaju iz klase *OCDeviceIndicator*. Kada se klikne na prekidač šalje se poruka OHM da uključi ili isključi uređaj. Komanda koja se koristi za slanje poruke implementirana je u klasi *OCSendCommandToDevice* komunikacionog modula. Namenska kontrola merač je implementirana u klasi *OCCGaugeControl* i sastoji se iz dva dela: kazaljke i

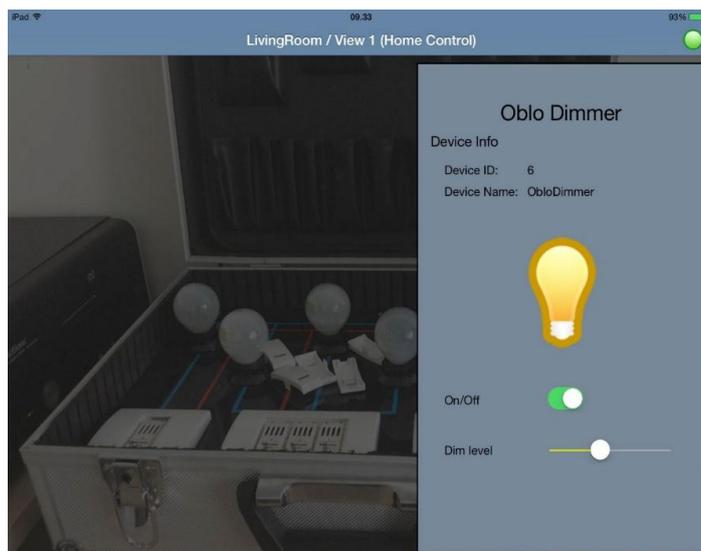
kruga sa podeocima. Klasa poseduje delegat funkciju *onGaugeClick()* koja obaveštava *OCOutletView* klasu da se desio klik. *OCOutletView* klasa preko funkcije *moveHand:(int)voltage* obaveštava *OCGaugeControl* klasu da se promenila potrošnja energija i da ona može da pomeri kazaljku na odgovarajuću poziciju prosleđenu preko parametra *voltage*.

Nakon što klasa *OCGaugeContol* obavesti *OCOutletView* klasu da se desio klik ona prikazuje grafik koji predstavlja potrošnju energije prikazanu u toku vremena. Grafik je implementiran pomoću klase koja se naziva *OCGraphich* koja sadrži funkciju *updateGraphichValue:(NSNumber\*)powerValue andTimeInterval:(int)timeInterval* preko koje je *OCOutletView* klasa obaveštava da se desila promena potrošnje energije. *OCGraphich* klasa nakon toga vrednost na *x* osi uveća za vrednost *timeInterval* parametra a na *y* osi iscrta vrednost parametra *powerValue*. Izgled grafika prikazan je na slici 4.10.



Slika 4.10 Izgled grafika

Na *y* osi grafika se nalaze podeoci koji označavaju potrošnju električne energije izražene u W (vat), a na *x* osi se nalaze podeoci koji predstavljaju vreme u sekundama.



Slika 4.11 Izgled prozor za kontrolu regulatora svetla

Na slici 4.11 prikazan je izgled prozor za kontrolu svetla. Na prozoru se nalaze sledeće komponente:

- *deviceID* – predstavlja identifikacioni broj uređaja
- *deviceName* – predstavlja ime uređaja
- slika - predstavlja stanje uređaja (uključen/isključen)
- prekidač (*engl. switch*) – daje mogućnost uključivanja/isključivanja uređaja
- klizač (*engl. slider*) – daje mogućnost menjanja osvetljaja sijalice

Klasa koja implementira ovaj panel naziva se *OCDimmerView*. Podaci za *deviceID* i *deviceName* se dobivljaju iz klase *OCDeviceIndicator*. Kada se klikne na prekidač šalje se poruka OHM da uključi ili isključi uređaj. Pomeranjem klizača šalje se komanda OHM da se promeni osvetlaj na vrednost pozicije na kom se nalazi klizač. Komanda koja se koristi za slanje poruke u oba slučaja implementirana je u klasi *OCSendCommandToDevice* komunikacionog modula. Prilikom inicijalizacije klase potrebno je postaviti određena polja klase a to su: *command* i *dimLevel*. U slučaju klika na prekidač *command* polje se postavlja na *On* ili *Off* u zavisnosti od stanja prekidača a *dimLevel* uvek na vrednost 0. U slučaju klizača polje *command* se postavlja na vrednost *dim* a *dimLevel* na vrednost pozicije klizača.

#### 4.3.4 Realizacija uređivača kuće

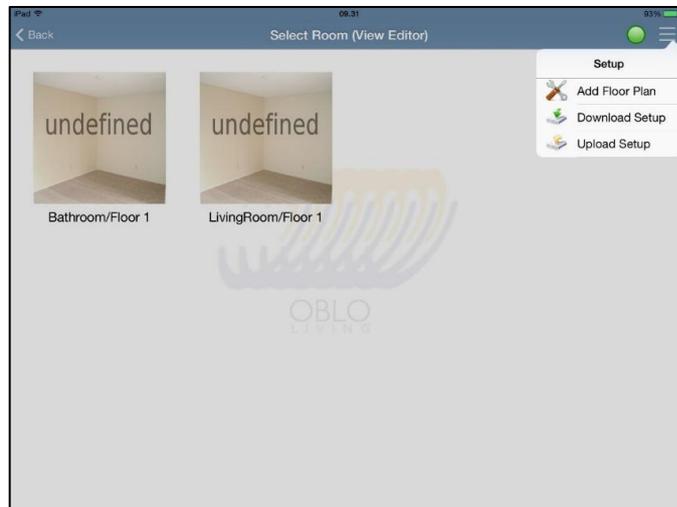
**Realizacija izgleda sobe/sprata** – za realizaciju ovog izgleda kao osnova je korišćena klasa *OCFloorViewController* modula za kontrolu kuće sa dodatim funkcionalnostima. Izgled ovog modula prikazan je na slici 4.12. Dodate funkcionalnosti su:

- Promena imena sobe – opcija omogućava da se promeni već postojeće ime sobe. Izmenjeni podaci se čuvaju u bazu podataka koristeći funkciju modula za podatke

*editRoom:(OCRoom\*)roomEdit withNewName:(NSString\*)newName* gde je *roomEdit* objekat odabrane sobe a *newName* je novo ime sobe.

- Postavljanje konfiguracije pametne kuće na OHM – opcija omogućava da se postavi nova konfiguracija kuće na OHM. Postavljanje nove konfiguracije na OHM vrši se pomoću dve klase *OCStartUpload* i *OCPrepareUpload* komunikacionog modula. Prilikom inicijalizacije klase *OCPrepareUpload* mora se postaviti ime konfiguracije kuće koje želimo da se nađe na OHM, a prilikom inicijalizacije klase *OCStartUpload* mora se postaviti polje *setupData* koje predstavlja podatke same konfiguracije. *setupData* polje se postavlja posredstvom modula podataka koristeći njegovu funkciju *getValueForSetup()* koja pročita podatke iz svake tabele baze podataka i spakuje ih u određeni format.
- Preuzimanje konfiguracije pametne kuće – opcija omogućava da se preuzme konfiguracija pametne kuće sa OHM. Funkcionalnost ove opcije implementirana je u klasi *OCPopoverSetupList*. Klasa prikazuje imena konfiguracija kuće koje su dobijena slanjem određene poruke OHM-u koja je implementirana u klasi komunikacionog modula *OCGetSetupList*. Klasa *OCPopoverSetupList* pruža mogućnost brisanja određene konfiguracije kuće.

Klasa koja realizuje ovaj izgled se naziva *OCEditFloorViewController*.

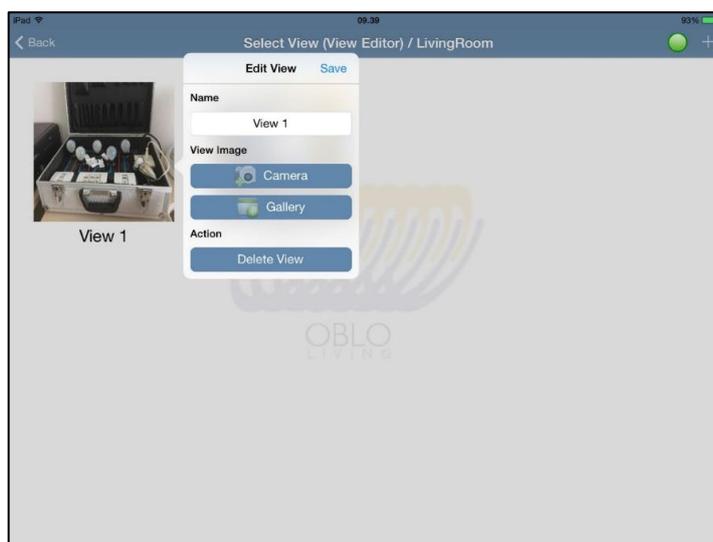


Slika 4.12 Izgled sobe/sprata (uređivač kuće)

**Realizacija izgleda sobe** – za realizaciju ovog izgleda za osnovu je korišćena klasa *OCRoomViewController* modula za kontrolu kuće koja je proširena sa sledećim funkcionalnostima:

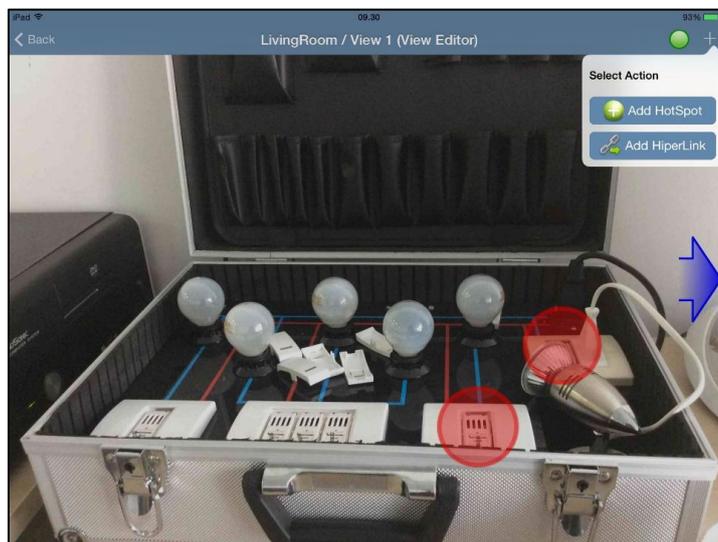
- Dodavanje novog pogleda na sobu - opcija omogućava dodavanje novog pogleda na sobu. Implementirana je u klasi *OCPopoverAddViewController* koja pruža sledeće mogućnosti :
  - Dodavanje slike za pogled sobe – vrši se na dva načina: koristeći kameru dostupnu na uređaju i koristeći slike iz galerija slika dostupnih na uređaju. Za oba slučaja koristi se ugrađena komponenta iOS platforme *UIImagePickerController*. Ova komponenta u slučaju dodavanja slike iz galerije slika omogućava prikaz galerija slika dostupnih na uređaju dok u slučaju da se koristi kamera omogućava da prikaz prozora na kom se može uslikati željena slika. Odabrana slika se u oba slučaja kompresuje radi manjeg zauzimanja memorije
  - Čuvanje odabranih podataka – vrši se posredstvom modula za čuvanje podataka koristeći njegovu funkciju *addView:(OCView\*)forSelectedRoomPID(NSString\*)roomPID*
- Promena već postojećih pogleda - na dugi klik na ćeliju kolekcije pojavljuje se dijalog sa slike 4.13. Dijalog je implementiran u klasi *OCPopoverEditViewController* koja pruža sledeće mogućnosti:
  - Brisanje odabranog pogleda sobe – vrši se posredstvom modula za čuvanje podataka pomoću njegove funkcije *deleteView:(OCView\*)viewForDelete*
  - Čuvanje izmenjenog imena pogleda - vrši se posredstvom modula za čuvanje podataka koristeći njegovu funkciju *editView:(OCView\*)view withNewName:(NSString\*)viewName*. Parametar *viewName* predstavlja novo ime pogleda sobe a parametar *view* predstavlja podatke odabranog pogleda sobe.

Klasa koja implementira ovaj izgled naziva se *OCEditRoomViewController*.



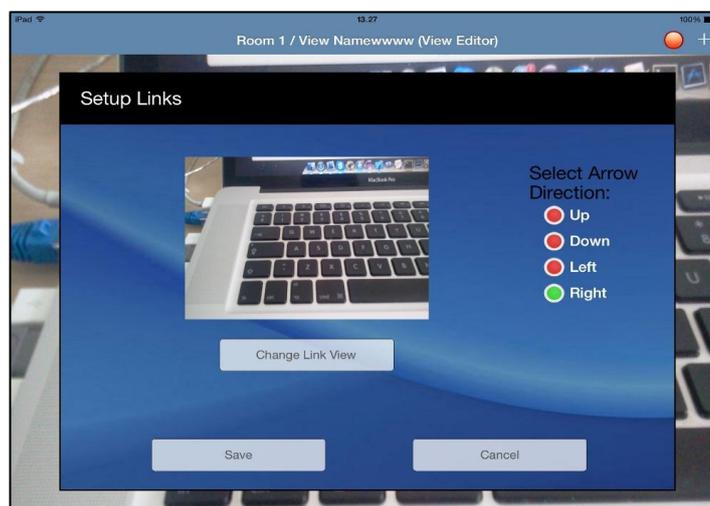
Slika 4.13 Izgled pogleda sobe (uređivač kuće)

**Realizacija izgleda rasporeda uređaja u sobi** – Za realizaciju ovog izgleda korišćena je klasa *OCDeviceViewController* modula za kontrolu sobe koja je proširena sa sledećim funkcionalnostima: dodavanje novog uređaja, dodavanje nove veze, promena parametra veze, promena parametar uređaja. Na slici 4.14. prikazan je izgled rasporeda uređaja u sobi.



Slika 4.14 Izgled rasporeda uređaja u sobi (uređivač kuće)

- **Dodavanje nove veze** – za dodavanje nove veze potrebno je odabrati opciju *AddHiperLink* sa slike 4.14 nakon koje se pojavljuje se dijalog sa slike 4.15.

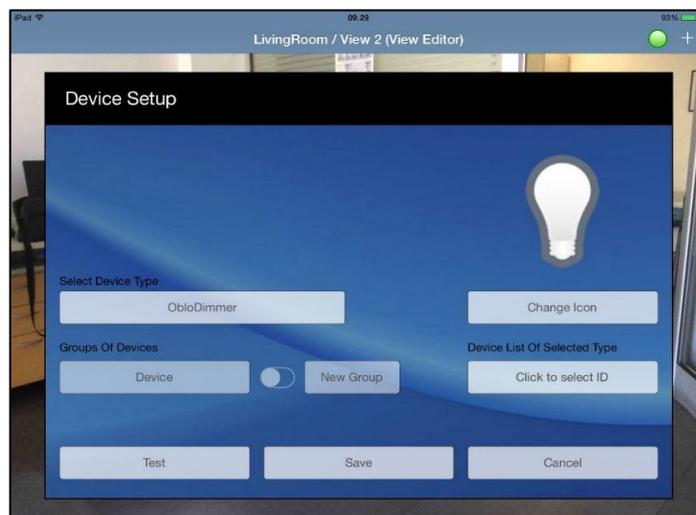


Slika 4.15 Izgled dijaloga za dodavanje nove veze

Dijalog je implementiran u klasi *OCLinkSettingsView* koja pruža sledeće mogućnosti:

- Odabir dostupnih pogleda sobe za odabranu sobu – ova mogućnost implementirana je u klasi *OCArrowCollectionControler*. Klasa omogućava kolekcijski prikaz podataka koristeći ugrađenu komponentu iOS platforme *UICollectionViewController*. Ova klasa podatke dobija posredstvom modula za

- čuvanje podataka pomoću njegove funkcije `-(NSArray *)getViewsForRoomPid:(NSString *)roomId` koja vraća niz pogleda za odabranu sobu
- Odabira orijentacije veze – sastoji se iz četiri radio-dugmeta koja predstavljaju namensku kontrolu grafičke sprege i implementirani su u klasi *OCRadioButton*. Klasa omogućava odabir samo jednog dugmeta.
  - Čuvanje odabranih podataka - sve prethodne izmene se čuvaju u bazu podataka posredstvom modula za čuvanje podataka koristeći njegovu funkciju `-(void)addArrow:(OCLinkArrow *)`
  - **Dodavanje novog uređaja** – za dodavanje novog uređaja potrebno je odabrati opciju *AddHotSpot* sa slike 4.14. nakon koje se pojavljuje se dijalog sa slike 4.16.



Slika 4.16 Izgled dijaloga za dodavanje novog uređaja

Dijalog je implementiran u klasi *OCDeviceSettingsView* koja pruža sledeće mogućnosti:

- Prikaz lista dostupnih tipova uređaja na OHM – lista dostupnih tipova dobija se posredstvom komunikacionog modula koristeći njegovu klasu *OCGetDeviceClasses*.
- Promena ikone uređaja – opcija omogućava da se pojavljuje kolekcija slika koju korisnik može da izabere. Nakon što korisnik izabere sliku na ekranu se menja već postojeća slika. Ova mogućnost implementirana je u klasi *OCCollectionPopover*
- Prikaz liste uređaja određenog tipa – nakon što je korisnik odabrao tip uređaja ovde se pojavljuje lista identifikacionih brojeva uređaja. Informacije kojim se popunjava lista dobavljaju se slanjem komanda OHM koja na osnovu tipa uređaja vraća identifikacione brojeve uređaja odabranog tipa. Komanda za dobijanje liste

uređaja određenog tipa implemetirana je u klasi *OCCGetDevicesByClass* komunikacionog modula

- Testiranje odabranog uređaja – šalje se komanda test za odbrani uređaj iz liste uređaja implementirana u klasi *OCTestDeviceCmd* komunikacionog modula.
- Čuvanje odabranih podataka – svi odabrani podaci se čuvaju u bazu podataka posredstvom modula za čuvanje podataka koristeći njegovu funkciju pod nazivom *addHotSpotForSelectedViewPID*.

- **Promena parametara uređaja** – postoje dva načina promena parametara uređaja:

Klikom na indikator uređaja sa slike 4.8. pojavljuje se dijalog sa tri opcije:

- Opcija *delete* – služi za brisanje selektovanog uređaja. Funkcija koja se koristi za brisanje uređaja je deo modula za čuvanje podataka i naziva se *deleteHotSpot:(OCHotSpot\*)hotSpotForDelete*.
- Opcija *configure* – služi za promenu konfiguracije odabranog uređaja. Odabirom ove opcije prikazuje se dijalog sa slike 4.16. Dijalog sadrži podatke o odabranom uređaju koje korisnik može menjati. Odabirom opcije potvrdi sva prethodna podešavanja se čuvaju u bazu podataka posredstvom modula za čuvanje podataka koristeći njegovu funkciju *editHotSpot(OCHotSpot\*)hotSpot forSelectedViewPID:(NSString \*)viewPID*
- Opcija *resize* – Izborom opcije korisniku se pružaju dodatne tri opcije: smanji krug, povećaj indikator uređaja i potvrdi izmene. Odabirom opcije smanji krug, selektovani indikator uređaja će se smanjiti što korisnik može videti na samom ekranu. Odabirom opcije povećaj krug selektovani indikator uređaja će se povećati što korisnik može videti na ekranu, tj. povećava se radijus odabranom. Odabirom opcije potvrdi sva prethodna podešavanja se čuvaju u bazu podataka posredstvom modula za čuvanje podataka koristeći njegovu funkciju *editHotSpot(OCHotSpot\*)hotSpot forSelectedViewPID:(NSString \*)viewPID*

Dugim klikom na indikator uređaja sa slike 4.8. Kada indikator uređaja postane zelen moguće je menjati njegov položaj tako što se prst pomera po ekranu. Kada se prst skloni sa ekrana vrši se provera da li okvir indikatora nema dodirnih tačaka sa ostalim komponentama na ekranu, ako indikator uređaja nema dodirnih tačaka onda je moguće promena pozicije selektovanog indikatora uređaja u suprotnom se vraća na svoju prvobitnu poziciju

- **Promena parametara veze** – postoje dva načina promene parametara veze.

Odabirom opcije Veza na ekranu sa slike 4.8. pojavljuje se dijalog sa dve opcije:

- 
- Opcija *delete* - služi za brisanje selektovane veze. Funkcija koja se koristi za brisanje veze je deo modula za čuvanje podataka i naziva se *deleteSelectedArrow:(OCLinkArrow\*)arrowForDelete*.
  - Opcija *configure* – služi za promenu konfiguracije selektovane veze. Odabirom ove opcije pojavljuje se dijalog sa slike 4.15. gde su sve komponente dijaloga popunjene sa parametrima selektovane veze. Odabirom opcije potvrdi sva prethodna podešavanja se čuvaju u bazu podataka posredstvom modula za čuvanje podataka koristeći njegovu funkciju *updateArrow:(OCLinkArrow\*)arrow*

Dugim klikom na jednu od veza na ekranu sa slike 4.8. Kada veza promeni boju moguće je menjati njen položaj tako što se prst pomera po ekranu. Kada se prst skloni sa ekrana vrši se provera da li okvir veze nema dodirnih tačaka sa ostalim komponentama na ekranu, u slučaju da ima dodirnih tačaka veze se vraća na svoju prvobitnu poziciju u suprotnom je moguće menjati poziciju selektovane veze. Ako je moguće promeniti položaj odabrane veze novi podaci se čuvaju u bazu podataka koriste funkciju modula za čuvanje podatka pod nazivom *updateArrowPostion:(OCLinkArrow\*)arrow*

## 5. Rezultati i testiranje

Da bi se utvrdila ispravnost i stabilnost aplikacije, ona mora biti podvrgnuta različitim vrstama testova. Nad realizovanim rešenjem su izvršeni sledeći testovi: ručni testovi, automatski testovi korišćenjem skripti i testovi za merenje performansi same aplikacije.

### 5.1 Ispitivanje funkcionalnosti

Ispitivanje i rezultati funkcionalnosti, navigacije i interakcije sa korisnikom u iOS aplikaciji svodi se na upotrebe aplikacije od strane korisnika. Treba ispitati da li se svaka komponenta, vidljiva na ekranu, može obeležiti i izabrati, da li je pritisak dugmeta praćen odgovarajućom reakcijom, odnosno kakve su posledice nasumičnog pritiskanja dugmadi.

Glavna ispitivanja fokusirala su se na funkcionalnosti korisničke sprege, odziv grafičke korisničke sprege na spoljašnje događaje, kao i otpornost aplikacije na greške prilikom korišćenja u dužem vremenskom periodu. Testovi su izvršavani ručno, na iPad uređaju sa ugrađenim operativnim sistemom iOS verzije 7.1. Primeri izvršenih testova su dati u nastavku. U tabeli 5.1 su prikazani sprovedeni ručni testovi.

Rb.	Opis testa	Očekivani rezultati	Uspešnost testa
1.	<b>Opis:</b> Promena stanja prekidača za uključivanje/isključivanje uređaja	Da se pojavi dijalog za kontrolu regulatora svetla uređaja sa svim potrebnim opcijama. Klikom na prekidač mora se promeniti	Uspešan
	<b>Početni uslovi:</b> OHM i iPad uređaj se moraju povezati na istu lokalnu mrežu. Aplikacija se mora nalaziti u prozoru za kontrolu regulatora svetla uređaja Slika 4.9		

	<b>Postupak:</b> potrebno je kliknuti prekidač na ekranu	trenutno stanje uređaja.	
2.	<b>Opis:</b> – promena već postojećeg pogleda sobe	Da se pojave svi dostupni pogledi za izabranu sobu. Nakon dugog klika da se pojavi dijalog sa opcijama za izmenu parametara pogleda sobe. Izmenom imena pogleda sobe na ekranu treba da se pojavi izabrani izgled sobe sa drugim imenom.	Uspešan
	<b>Početni uslovi:</b> OHM i iPad moraju biti povezani na istu lokalnu mrežu. Aplikacija se mora nalaziti u prozoru izgleda sobe modula za uređivanje kuće ( <i>OCEditRoomViewController</i> ), Slika 4.12. Mora postojati bar jedan pogled sobe.		
	<b>Postupak:</b> dugi klik na ćeliju kolekcije. Nakon što se pojavi dijalog potrebno je uneti novo ime pogleda i nakon toga kliknuti dugme <i>Save</i> na dijalogu		
3.	<b>Opis:</b> promena pozicije kruga (uređaja) u pogledu sobe	Da se pojave svi dostupni krugovi (uređaji) na ekranu. Nakon dugog klika da se pojavi zeleni okvir kruga i da se nakon toga može pomeranjem prsta krug dovesti u drugu poziciju. Nakon puštanja prsta krug mora da se ostane na željenoj poziciji na ekranu	Uspešan
	<b>Početni uslovi:</b> OHM i iPad moraju biti povezani na istu lokalnu mrežu. Aplikacija se mora nalaziti u prozoru za prikaz rasporeda uređaja u sobi modula za uređivanje kuće ( <i>OCEditDeviceViewController</i> ), Slika 4.13. Mora postojati bar jedan krug u okviru ovog izgleda		
	<b>Postupak:</b> zadržati klik na krug. Čim krug promeni boju u zelenu potrebno je krug pomeriti na željenu poziciju na ekranu		
	<b>Opis:</b> klik na vezu u izgledu kontrole uređaja	Da se pojave sve dostupne veze u	

4.	<b>Početni uslovi:</b> OHM i iPad moraju biti povezani na istu lokalnu mrežu. Aplikacija se mora nalaziti u prozoru za kontrolu uređaja ( <i>OCDeviceViewController</i> ), Slika 4.7. Mora postojati barem jedna veza na ekranu	izabranom pogledu sobe. Klikom na vezu potrebno je promeniti ekran na odabrani pogleda sobe.	Uspešan
	<b>Postupak:</b> potrebno je kliknuti na jednu od veza na ekranu		

Tabela 5.1 Ručni testovi

## 5.2 Automatski testovi

Automatski testovi izvršavaju se pomoću alata *Instruments*. Ovaj alat u sebi sadrži komponentu *UIAutomation* koja pravljenjem skripte u *JavaScript* [8] programskom jeziku može da oponaša korisnika bez ikakvog dodira uređaja. Ovaj alat podržan je od iOS 4.0 verzije i moguće je testirati aplikacije na samim uređajima, kao i na simulatorima dostupnim u XCode okruženju.

Glavna prednost ovakvih testiranja aplikacije jeste što jednom napisan test može da se ponovi na različitim uređajima kako bi se proverila podrška na različitim konfiguracijama i verzijama iOS operativnog sistema. Jednom napisan test može se ponoviti nekoliko puta u toku samog pravljenja aplikacije čime se može uštedeti na vremenu. Jedan primer izvršenog testa biće objašnjen u daljem tekstu zajedno sa primerom skripte koja je korištena za njegovo izvršenje.

Na slici 5.1 prikazana skripta predstavlja automatski test promene stanja prekidača (*engl. switch*) unutar kontrole uređaja. Sadrži četiri funkcije:

- Prva funkcija *mainScreenTest* predstavlja simulaciju pritiska dugmeta unutar glavnog menija.
- Druga funkcija *roomScreenTest* predstavlja simulaciju klika na jednu od ćelija kolekcije iz izgleda sobe/sprata.
- Treća funkcija *viewScreenTest* predstavlja simulaciju klika na jednu od ćelija kolekcije iz izgleda sobe.
- Četvrta funkcija *deviceTest* predstavlja simulaciju klika na jedan od krugova na izabranom prozoru. Vršiti se i simulacija promene vrednosti prekidača sa malim zakašnjenjem u promeni vrednosti.

```
var target = UIATarget.localTarget();
var app = target.frontMostApp();
var mainWindow = app.mainWindow();

function mainScreenTest() {
    var testName = "Main screen test";
    UIALogger.logStart(testName);
    UIATarget.localTarget().tap({x:250, y:200});
    UIALogger.logPass(testName);
}
function roomScreenTest()
{
    var testName = "Room screen test";
    UIALogger.logStart(testName);
    //get reference to collection view
    var rootCollection = mainWindow.collectionViews()[0];
    //click on second item of tableview
    rootCollection.cells()[1].tap();
    //wait 1 sec
    target.delay(1);
    UIALogger.logPass(testName);
}

function viewScreenTest()
{
    var testName = "View screen test";
    UIALogger.logStart(testName);
    var rootCollection = mainWindow.collectionViews()[0];
    //click on first item of collectionView
    rootCollection.cells()[0].tap();
    UIALogger.logPass(testName);
}
function deviceTest()
{
    var testName = "Device test";
    UIALogger.logStart(testName);
    mainWindow.logElementTree();
    //click on device dialog
    mainWindow.scrollViews()[0].tapWithOptions({tapOffset:{x:0.67, y:0.68}});
    target.delay(2);
    //set switch value to ON
    mainWindow.switches()[0].setValue(1);
    target.delay(2);
    //set switch value to OFF
    mainWindow.switches()[0].setValue(0);
    UIALogger.logPass(testName);
}
mainScreenTest();
roomScreenTest();
viewScreenTest();
deviceTest();
```

Slika 5.1 Primer skripte korišćene za automatsko testiranje

## 5.3 Testovi performansi

Testovi performansi odnose se na merenja iskorišćenosti procesora i radne memorije uređaja tokom rada aplikacije. Za potrebe ovih testova korišćen je takođe alat *Instruments*. Ovaj alat sadrži komponentu *Activity Monitor* koja vrši testove procesora i memorije. U svrhu testiranja korišćen je iPad Air uređaj sa ugrađenim operativnim sistemom iOS verzije 7.1, procesorom Dual-core 1,3GHz, radne memorije 1GB.

Merenja su izvršena za tri slučaja: normalan rad aplikacije, prilikom skidanja konfiguracije kuće sa OHM i kada konfiguracija kuće ima više od 20 pogleda na sobu.

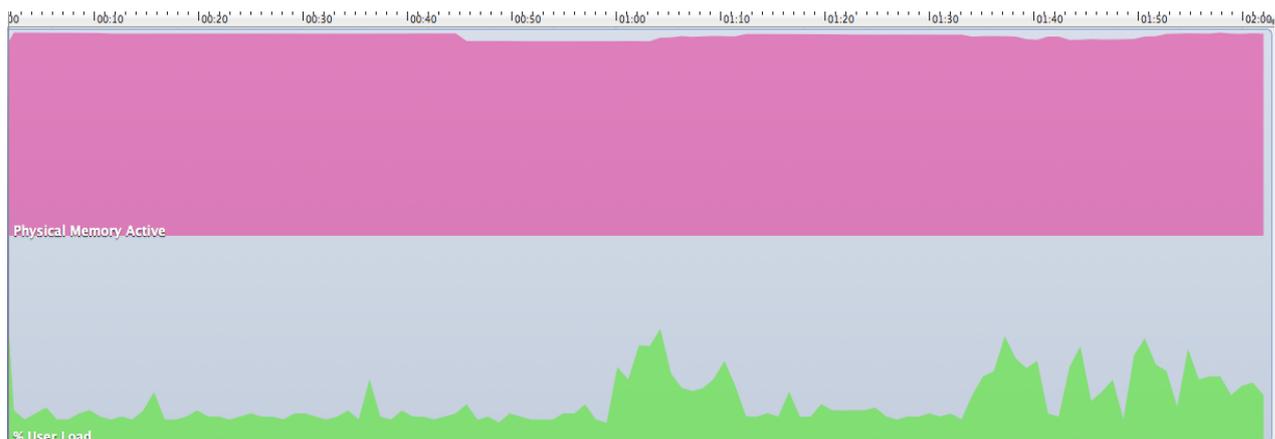
### 5.3.1 Testovi za osnovni režim rada aplikacije

Ovo testiranje je vršeno tokom korišćenja cele aplikacije, tačnije korišćene su sve moguće opcije koja daje aplikacija. U tabeli 5.2 može se videti zauzetost procesora i radne memorije tokom korišćenja aplikacije. Prikazani su minimalni, srednji i maksimalni rezultati zauzetosti procesora i memorije.

	<b>Minimalna vrednost</b>	<b>Srednja vrednost</b>	<b>Maksimalna vrednost</b>
<b>Iskorišćenost procesora [%]</b>	2	4.6	7
<b>Iskorišćenost memorije [MB]</b>	13	25	35

Tabela 5.2 Rezultati zauzetosti memorije i procesora za normalni režim rada aplikacije

Na slici 5.2 je prikazan grafički prikaz rezultata testiranja za normalni režim rada aplikacije. Ono što nam je bitno je *User Load* (tamno zelena boja na grafiku) koja predstavlja zauzetost procesora tokom vremena i *Physical Memory Active* (tamno ljubičasta boja) koja predstavlja zauzetost memorije tokom vremena.



Slika 5.2 Grafički prikaz rezultata testiranja za normalni režim rada aplikacije

Ono što treba primetiti je da kod normalnog režima rada aplikacije iskorišćenost procesora i memorije je jako mala što je trebalo i očekivati jer normalan režim rada aplikacije nije zahtevan po pitanju performansi.

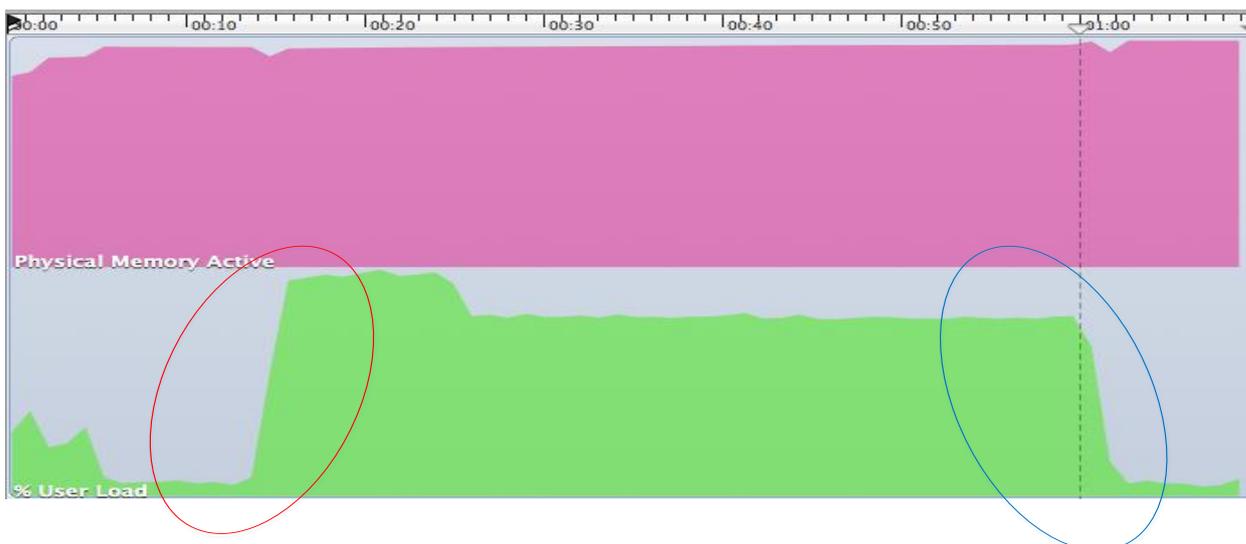
### 5.3.2 Testovi prilikom preuzimanja konfiguracije kuće sa OHM

Ovo testiranje vršeno je prilikom skidanja konfiguracije kuće sa OHM. Test je pokrenut u trenutku kada je počelo dobavljanje konfiguracije sa OHM i prekinut je nakon uspešnog preuzimanja. Rezultati ovog testiranja koji su trajali oko 3-5 minuta prikazani su u tabeli 5.3 gde se može videti minimalna, srednja i maksimalna vrednost zabeležena tokom testiranja.

	<b>Minimalna vrednost</b>	<b>Srednja vrednost</b>	<b>Maksimalna vrednost</b>
<b>Iskorišćenost procesora [%]</b>	100	100	100
<b>Iskorišćenost memorije [MB]</b>	20	25	35

Tabela 5.3 Rezultati zauzetosti memorije i procesora tokom skidanja konfiguracije kuće

Na slici 5.3 je prikazan grafički prikaz zauzetosti procesora i memorije za ovaj slučaj testiranja.



Slika 5.3 Grafički prikaz rezultata testiranja tokom preuzimanja konfiguracije kuće

U trenutku početka dobavljanja konfiguracije kuće pojavio se nagli porast zauzetosti procesora, crveni marker sa slike 5.3 Tokom trajanja dobavljanja zauzetost procesora je konstantno na 100%. Na kraju dobavljanja konfiguracije kuće zauzetost procesora se naglo smanjuje jer su svi potrebni podaci dobavljeni sa OHM-a i nema potreba za njihovom obradom, plavi marker sa slike 5.3. Zauzetost memorije raste od početka preuzimanja konfiguracije pa do njenog kraja. Te rezultate je trebalo očekivati.

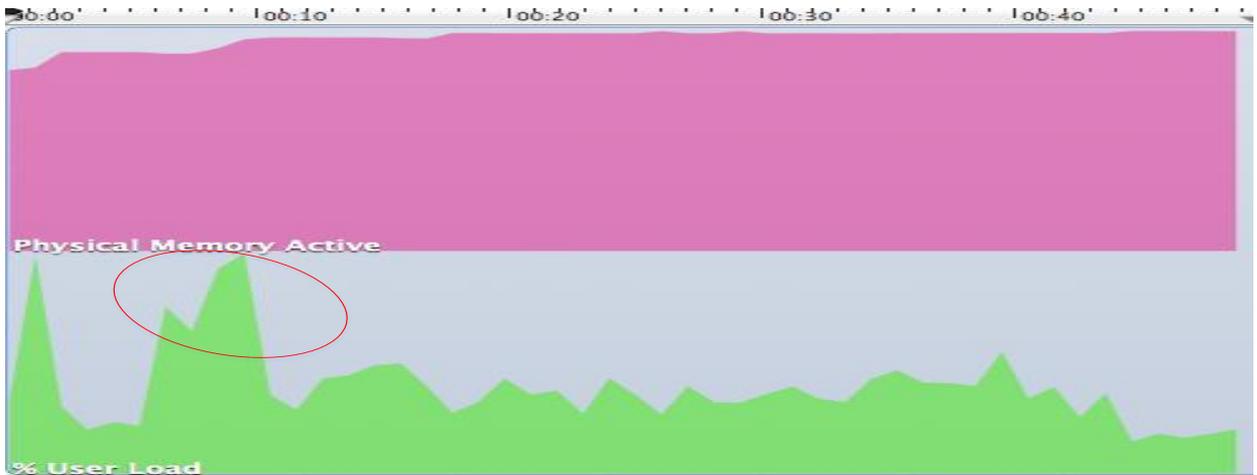
### 5.3.3 Testiranje aplikacije kada postoji više od 20 pogleda na sobu

Ovo testiranje je izvršeno kada je u konfiguraciji kuće prisutno više od 20 pogleda na sobu. Rezultati testiranja koje je trajalo oko 1 minut su prikazani u tabeli 5.4 gde se može videti minimalna, srednja i maksimalna vrednost zabeležena tokom testiranja.

	<b>Minimalna vrednost</b>	<b>Srednja vrednost</b>	<b>Maksimalna vrednost</b>
<b>Iskorišćenost procesora [%]</b>	10	15	25
<b>Iskorišćenost memorije [MB]</b>	27	31	36

Tabela 5.4 Rezultati zauzetosti procesora i memorije za slučaj više od 20 pogleda sobe

Na slici 5.4 je prikazan grafički prikaz zauzetosti procesora i memorije za ovo testiranje.



Slika 5.4 Grafički prikaz rezultata testiranja kada je prisutno više od 20 pogleda sobe

Ono što je potrebno primetiti je da iskorišćenost memorije i procesora nije znatno veće od normalnog režima rada aplikacije, što je jako dobro. Usled pojave velike količine podataka tokom učitavanja više od 20 pogleda na sobu pojavljuje se pik, crveni marker sa slike 5.4, dok su ostale vrednosti zauzetosti procesora u granicama normale.

## 6. Zaključak

Zadatak rada bio je realizacija iOS aplikacije za upravljanje pametnom kućom. Sve planirane funkcionalnosti aplikacije su implementirane i testirane. Aplikacija je zavisna od internet konekcije kao i od poslužioca OHM koji joj pruža usluge. Samim tim performanse same aplikacije zavise od karakteristike mreže i od samog poslužioca. Izgled korisničke sprege aplikacije je sličan već postojećem rešenju realizovanom za Android platformu.

iOS aplikacija je sposobna da iz bilo kog dela kuće gde iOS uređaj ima pristup lokalnoj mreži olakša korisniku upravljanje uređajima u sklopu OHM prisutnog u pametnoj kući.

Zadatkom je realizovan osnovni koncept iOS aplikacije. Aplikacija je realizovana za sve verzije iOS sistema veće od 6.0. Programsko rešenje je realizovano modularno da bi se aplikacija lako proširivala i menjala. Ono što aplikacija trenutno ne podržava, a bilo bi poželjno da ponudi u bliskoj budućnosti jeste:

- Realizacija modula za bezbednost i nadzor
- Konfiguracija samog izgleda pametne kuće, dodavanje i uklanjanje spratova i soba
- Upotreba više profila i automatski odabir na osnovu identifikatora bežične mreže

## 7. Literatura

- [1] About iOS, ac. 4.7.2014. <http://en.wikipedia.org/wiki/IOS>
- [2] About XCode, ac.4.7.2014,  
<https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode/Overview/LearnMoreAboutXcode/LearnMoreAboutXcode.html>
- [3] About Instruments, ac. 4.7.2014.  
<https://developer.apple.com/library/ios/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html>
- [4] OBLO, ac. 4.7.2014, <http://oblo.rt-rk.com/sr/>
- [5] M. Vukadinović Zbornik FTN, Jedno rešenje bežično kontrolisane priključnice za praćenje potrošnje električne energije
- [6] ZigBee Alliance official site, ac.4.7.2014, <http://www.zigbee.org/>
- [7] About Objective C, 4.7.2014, <http://en.wikipedia.org/wiki/Objective-C>
- [8] About JavaScript official site, ac. 4.7.2014  
<http://javascript.about.com/od/reference/p/javascript.htm>
- [9] Official documentation, page 15 <http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf>
- [10] About iOS architecture, ac. 4.7.2014.  
<https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html>