SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3907

SIMULACIJSKI MODEL PAMETNE KUĆE

Filip Rudan

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 6. ožujka 2015.

ZAVRŠNI ZADATAK br. 3907

Pristupnik: **Filip Rudan (0036468737)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Simulacijski model pametne kuće**

Opis zadatka:

Proučiti osnovne elemente koji su potrebni u modelu pametne kuće. Proučiti načine izgradnje trodimenzijskog modela objekata te načine upravljanja pojedinim aktivnostima dinamičkih elemenata u objektu. Realizirati simulacijski model te načiniti niz primjera koji će demonstrirati upravljanje simulacijskim modelom ostvarene virtualne pametne kuće. Načiniti ocjenu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Sadržaj

1. Uvod

Brzi napredak računalnih tehnologija otvorio je vrata sve lakšoj integraciji računalnih komponenti u svakodnevni život. Uređaji koji su se nekad isključivo ručno aktivirali i programirali danas sve više dolaze s ugrađenim računalnim komponentama koje mogu donositi odluke za nas. Na taj način ljudima omogućuju jednostavniju, fleksibilniju i napredniju uporabu istih. Pojava kućnih mreža jedan je od glavnih faktora koji su omogućili pojavljivanje takvih sustava i njihov brzi napredak. Već danas se mogu pronaći gotova rješenja koja integriraju više različitih uređaja u cjeline s naprednim mogućnostima koje nije moguće postići s istim uređajima koji su neovisni. Takvi sustavi iako napredni i dalje su zatvoreni u odnosu na interakciju s drugim takvim sustavima. Logični slijed je omogućiti interakciju više tih sustava u jedinstvenu cjelinu na razini kuće. Taj koncept zove se pametna kuća.

Ovaj rad počet će predstavljanjem osnovnih koncepata i elemenata pametne kuće. Nastavit će se s pregledom modela pametne kuće. Na kraju će se demonstrirati razvijeno rješenje pametne kuće.

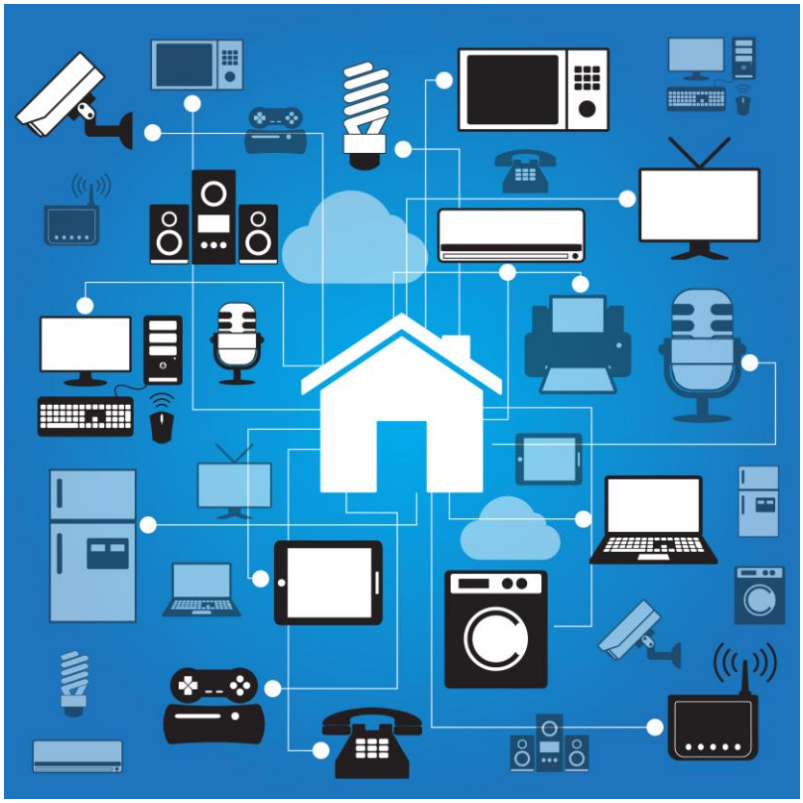
2. Pametna kuća

Prije ulaska u detalje izvedbe pametne kuće potrebno je definirati generalna svojstva sustava.

2.1. Definicija pametne kuće

Pametna kuća je definirana kao boravište koje inkorporira komunikacijsku mrežu u svrhu povezivanja ključnih električnih i elektroničkih uređaja i usluga, te omogućava da se njima daljinski upravlja, prati i pristupa (Slika 1.).

Pametne kuće koriste tehnologije kućne automatizacije koje korisnicima omogućavaju dobivanje naprednih povratnih informacija nadgledanjem raznih aspekata kuće. Na primjer, hladnjak u pametnoj kući može katalogizirati hranu, predložiti menije i zdrave alternative, naručiti zamjenu za potrošenu hranu, i tako dalje.



Slika 1. *Pametna kuća*

2.2. Elementi i koncepti u pametnoj kući

2.2.1. Elementi pametne kuće

Sustav pametne kuće sastoji se od kućne mreže i uređaja. Uređaji u sebi mogu imati razne kombinacije senzora, kontrolera i sučelja za komunikaciju s ljudima ili drugim uređajima.

Senzori služe za mjerenje vrijednosti u okolini kuće na temelju kojih se može donijeti odluka. Neke od, za sustav, zanimljivih vrijednosti koje se mogu mjeriti su primjerice: temperatura, atmosferski uvjeti, osvjetljenje, pokreti, buka, ...

Zadaća kontrolera je upravljanje stanjima sustava i prijelazima u druga stanja, na primjer, rad svjetla, klime uređaja, audio sustava...

Zadaća sučelja je premostiti jaz interakcije između ljudi i uređaja. Sučelja predstavljaju točku upravljanja nad mogućnostima sustava.

Uređaj, kao dio pametne kuće, može biti višestruka kombinacija prethodno navedenih elemenata u koherentnu cjelinu.

Kućna mreža je „ljepilo“ koje omogućava međusobnu interakciju između svih elemenata pametne kuće. Može imati aktivnu ili pasivnu ulogu ovisno o njezinom sudjelovanju u radu pametne kuće. Mrežna komunikacija može biti ostvarena preko različitih medija. Neki od važnijih medija za komunikaciju su žični Ethernet i optički kabeli, bežični Wi-Fi te eksperimentalna kućna energetska mreža kao medij.

2.2.2. Koncepti u pametnoj kući

Uslužno orijentirana arhitektura (Service oriented architecture - SOA) bazira se na konceptu usluge. Usluga predstavlja samostalnu jedinicu funkcionalnosti koju neka druga komponenta sustava može pozvati. Različiti sustavi na mreži nude različite usluge, te se mogu oslanjati na druge usluge za svoj pravilan rad. Kombinacijom raznih usluga mogu se stvarati komplicirane funkcionalnosti koje se onda mogu sakriti iza sučelja i predstavljati kao nove usluge. Usluge mogu razmjenjivati informacije preko mreže, a svaki uređaj može nuditi više usluga, ako je za to sposoban. Ovakva arhitektura omogućava sustavu fleksibilnu nadogradnju i dodavanje novih usluga kao i kombinaciju već postojećih mogućnosti u nove naprednije sposobnosti [1].

Događajima pokretana arhitektura (Event-driven architecture - EDA) je orijentirana na proces stvaranja, detekcije, obrade i reakcije na događaje. Događaj je definiran kao neka bitna promjena u stanju sustava. Poruka koja opisuje događaj može dalje biti proslijeđena ostalim dijelovima sustava na obradu. Ovakva arhitektura omogućava sustavu pametne kuće reaktivnost na promjene unutar sustava na vrlo fleksibilan način [2].

Pametna kuća predstavlja samo jedan mali podskup Interneta stvari (Internet of Things - IoT) to jest sustava u kojem gotovo svaki uređaj ima mikrokontroler i sposobnost komunikacije preko mreže. Internet stvari koristi koncepte uslužno orijentiranih i događajima pokretanih arhitektura kojima se postiže napredna komunikacija i interakcija između uređaja-ljudi te uređaja-uređaja [3].

2.3. Simulacijski model pametne kuće

Simulacijski model je baziran na softverskom arhitekturalnom obrascu MVC (model-view-controller) koji se sastoji od 3 glavne komponente.

Komponenta model MVC-a realizira simulaciju pametne kuće koja uključuje logiku rada i ponašanja konkretnih komponenti sustava. Ona se sastoji od komponenti koje simuliraju virtualne uređaje i njihovo ponašanje. Svaki virtualni uređaj unutar sebe može sadržavati više usluga koje predstavljaju neku funkcionalnost koju oni mogu izvršiti koja je dostupna vanjskim akterima na upotrebu. Ulogu upravljanja i administracije nad virtualnim uređajima ima upravitelj simuliranih uređaja.

Komponenta kontroler MVC-a realizirana je preko univerzalnog sučelja koje servisi nude prema van.

Uređaj ima sučelje preko kojeg se drugi alat može spojiti i dohvatiti stanje uređaja i napraviti konkretnu reprezentaciju njega. Korištenjem tog sučelja realizira se komponenta pogled MVC-a.

2.3.1. Model

Servis u sebi ostvaruje koncepte SOA i EDA. On nudi standardizirani način za dohvaćanje svojih ponuđenih mogućnosti i za izvršavanje tih usluga. Također, nudi i mogućnost slanja obavijesti o promjenama koje su se dogodile.

Uređaj predstavlja virtualnu simulaciju nekog stvarnog uređaja. Nudi sučelje kojim se može dohvatiti interno stanje uređaja, iz kojeg se dalje gradi virtualna reprezentacija uređaja. Uređaj na sebi može izvršavati više različitih usluga ako to ima smisla. Uređaj ima mogućnost dodavanja i uklanjanja usluga koje nudi.

Upravitelj uređaja je centralni sustav za upravljanje uređajima. On omogućava dodavanje novih, uklanjanje starih uređaja. Omogućava administraciju uređaja koja uključuje dodavanje, gašenje, brisanje usluga koje su dio tih uređaja.

2.3.2. Kontroler

Zadaća kontrolera je da omogući univerzalni način pristupa uređajima i uslugama. Kontroler može biti izveden na različite načine koji mogu ići sve od zadavanja naredbi preko naredbenog retka pa sve do nuđenja usluga preko web servisa.

2.3.3. Pogled

Pogled izvršava zadaću reprezentacije tih uređaja korisniku. On se spaja na definirana sučelja za pogled uređaja, iz njih vuče informacije o uređaju te ih reprezentira korisnicima na pristupačan način. Takvo reprezentacija može varirati od jednostavne koja samo ispisuje vrijednosti na ekran pa sve do kompliciranih kao grafička reprezentacija uređaja u stvarnom vremenu.

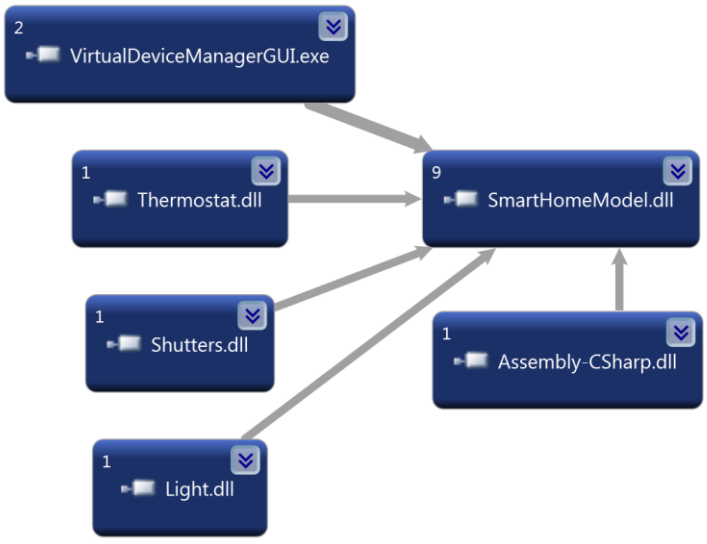
3. Realizacija modela pametne kuće

3.1. Microsoft .NET

.NET platforma, korijenima još iz davne 2000., jedan je od najpopularnijih radnih okvira za razvoj aplikacija na Windows platformi. Razlog njegove popularnosti je bogati skup biblioteka i mogućnosti koji olakšava razvoj programa brinući se o dosadnim i teškim dijelovima razvoja. Omogućava programeru da se posveti razvoju srži programa bez prevelike muke. Inspiriran ranim pokušajima Jave, objedinjuje više platformi kroz zajednički strojni međukod CIL i virtualni stroj CLR koji omogućava razvoj programa jedan put za više platformi. Također ima podršku za najpopularnije jezike koji se mogu prevesti u njegovu platformu što omogućava jednostavno kombiniranje više jezika u istom projektu. U svrhu ovog projekta korištene su osnovne mogućnosti radnog okvira .NET zajedno sa Javom inspiriranim jezikom C#.

3.2. Organizacija projekta modela pametne kuće

Rješenje modela pametne kuće smješteno je u projektu „SmartHomeModel“. Rješenje se kompajlira u prenosivu biblioteku radi prenosivosti projekta (Slika 2.).



Slika 2. *Organizacija projekta*

3.3. Upravljivost

Izvedba glavnih upravljačkih svojstava prisutna je u apstraktnom razredu „Controlable“. Razred definira metode za inicijalizaciju i izvođenje asinkronog zadatka. Također razred ima metode za pokretanje, gašenje i privremeno zaustavljanje asinkronim zadatkom. Razred omogućuje dohvaćanje imena i jedinstvenog identifikacijskog broja zadatka (Slika 3.).

Očekuje se da izvedeni razred implementira dohvaćanje imena i identifikacijskog broja, zajedno s metodama inicijalizacije i izvođenja konkretnog zadatka. Glavni izvedeni razredi su uređaj „Device“ i usluga „Service“, jer usluge i uređaji mogu imati različite zadatke koje neovisno izvode, a time se rješava postojeći problem i sprječava dupliciranje koda.

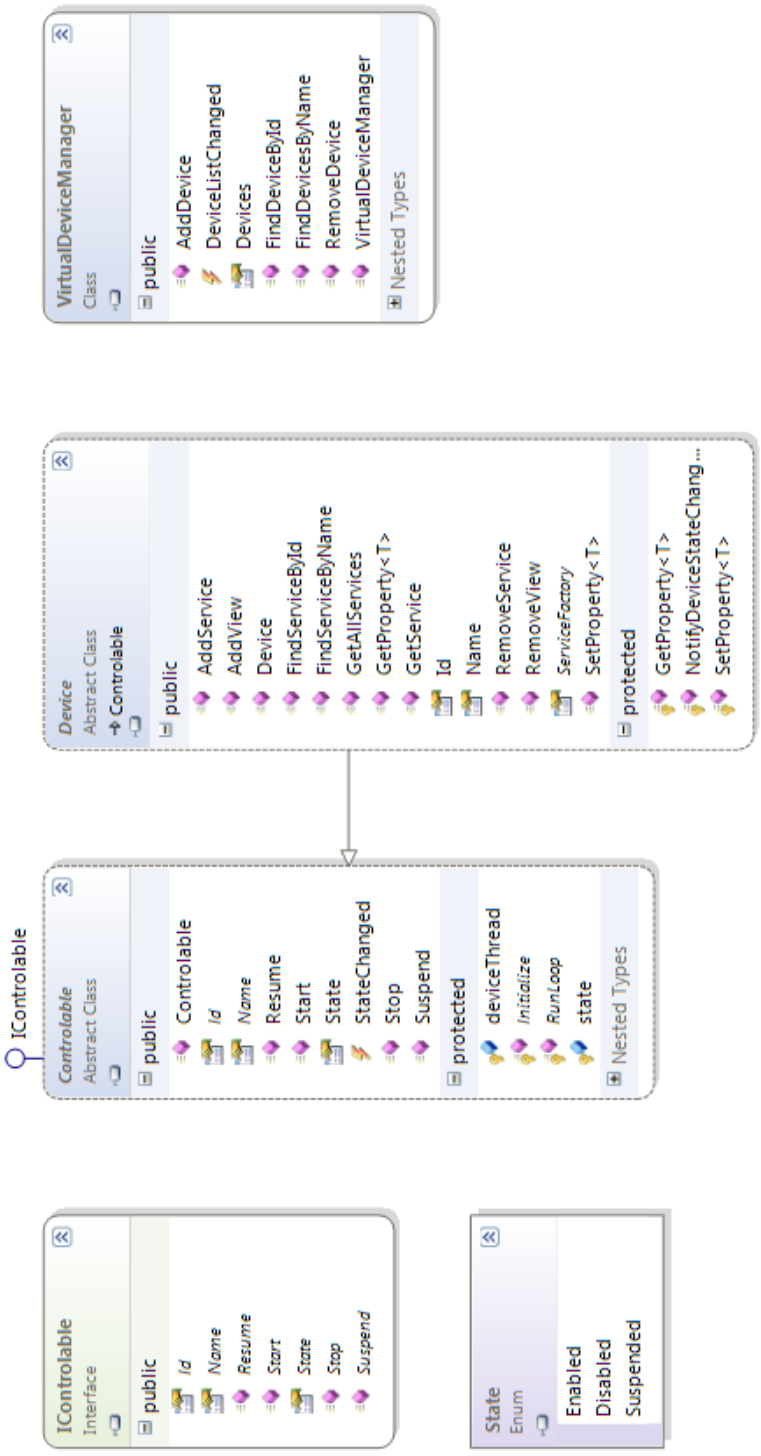
3.4. Uređaj

Simulirani uređaj modeliran je apstraktnim razredom „Device“ (Slika 3.) koji raspolaže kolekcijom svojstava, kolekcijom usluga „Service“ i omogućava prijavu zainteresiranih pogleda „View“ koji se preko oblikovnog obrasca „publisher-subscriber“ obavještavaju o promjenama svojstava uređaja.

Osnovna svojstva koja svaki uređaj sadrži su ime uređaja „DeviceName“ i njegov identifikacijski broj „DeviceId“. Uređaj je jedinstveno određen svojim identifikacijskim brojem. Uređaj skriva kolekciju svojstava i umjesto toga definira način upravljanja svojstvima preko već predefiniranih metoda.

Uređaj ima predefinirane metode kojima se manipulira kolekcijom usluga. Omogućava traženje, dodavanje i uklanjanje usluga, kao i dohvaćanje popisa svih usluga koje su pridijeljene uređaju. Usluge imaju pravo pristupa svojstvima konkretnog uređaja samo ako su pridružene tom uređaju, odnosno, uređaj provjerava da su zahtjevi koji manipuliraju njegovim svojstvima odaslani od usluga koje su pridružene tom uređaju. Time se sprječava neovlašteni, potencijalno maliciozni, pristup glavnim karakteristikama uređaja jer je predviđeno da se s njim komunicira preko usluga.

Uređaj preuzima funkcionalnost razreda „Controlable“ kojime je omogućena implementacija složenijih mogućnosti i funkcionalnosti.



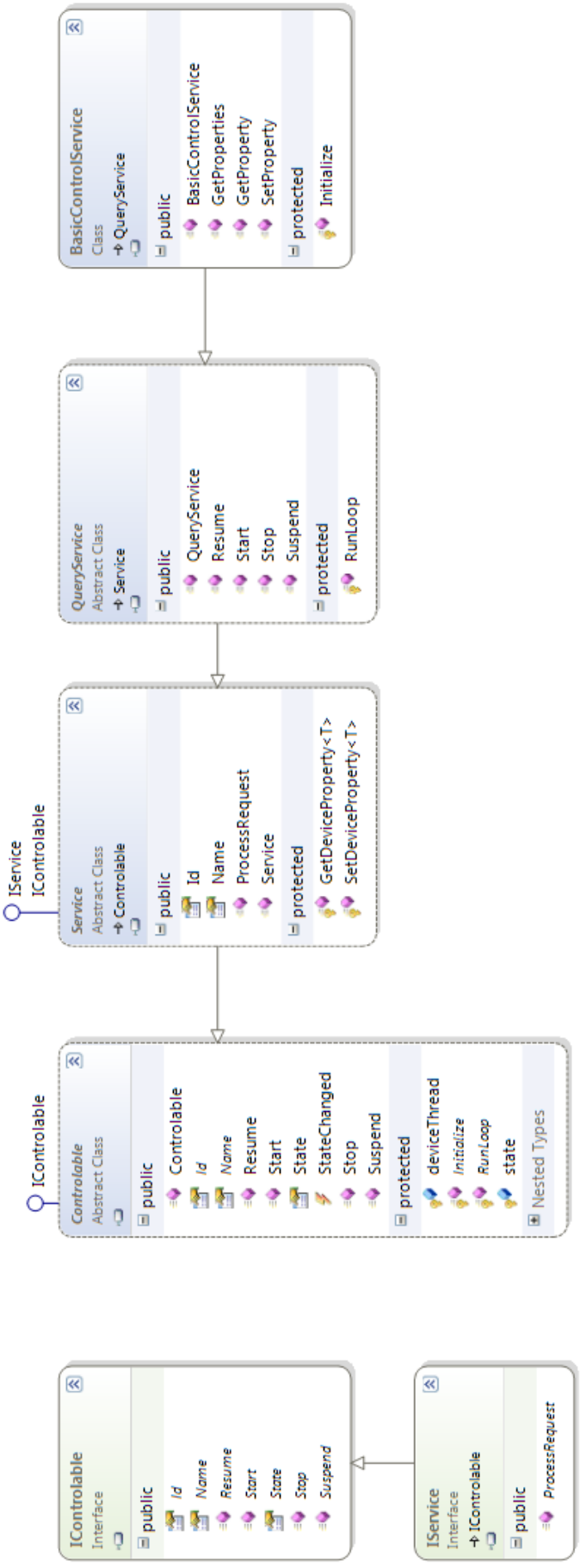
Slika 3. *Dijagram razreda uređaja*

3.5. Usluga

Simulirana usluga modelirana je apstraktnim razredom „Service“ (Slika 4.) koji preuzima funkcionalnost razreda „Controlable“ iz istog razloga kao i prethodno definirani uređaj. Svaka usluga ima svoje ime i jedinstveni identifikator kojime je određena.

Dodatno razred usluga definira univerzalnu metodu za obradu kojoj se predaju parametri zahtjeva, a vraća se odgovor na zahtjev. Obrada zahtjeva iskorištava napredne mogućnosti introspekcije, radnog okvira .NET, kojima dinamički zahtjeve prosljeđuje odgovarajućim javnim metodama. Na taj način olakšan je razvoj konkretnih usluga koji se svodi samo na definiciju metoda kojima je izvedena konkretna funkcionalnost neke takve usluge u izvedenom razredu.

U svrhu jednostavnijeg razvoja budućih usluga iz razreda usluge izvedeni su pomoćni razredi: „QueryService“ kojime se olakšava implementacija diskretnih, vremenski neovisnih, zahtjeva; i razred „BasicControlService“ koji nudi jednostavnu uslugu za dohvaćanje, postavljenje i pregled svojstava uređaja kojemu je usluga pridružena.

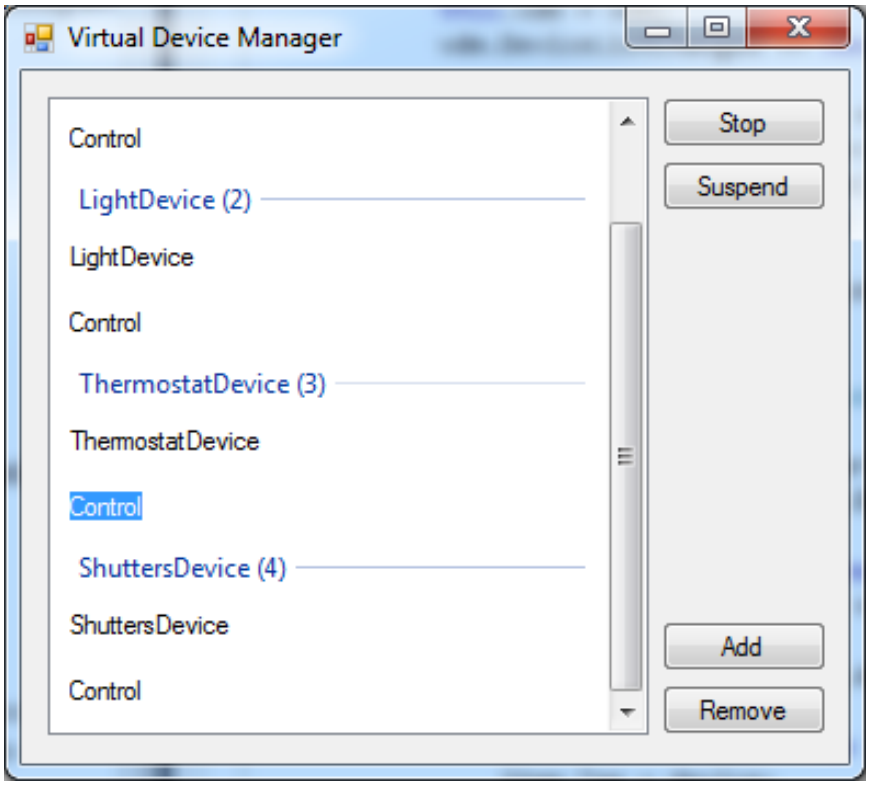


Slika 4. *Dijagram razreda usluge*

3.6. Upravitelj virtualnih uređaja

Upravitelj virtualnih uređaja modeliran je razredom „VirtualDeviceManager“. Njegova svrha je upravljanje kolekcijom virtualnih uređaja. Razred omogućava jednostavno dodavanje, uklanjanje uređaja te pretraživanje uređaja po imenu i identifikacijskom broju. Dodatne mogućnosti su mogućnost prijave praćenja promjena nad kolekcijom uređaja.

Grafičko sučelje upravitelja uređaja (Slika 5.) realizirano je pomoću biblioteke Windows.Forms radnog okvira .NET. Sučelje omogućava lagano dodavanje i uklanjanje novih uređaja i usluga u simulaciju. Omogućeno je lagano pokretanje, zaustavljanje i gašenje uređaja i usluga.



Slika 5. *Primjer upravitelja uređaja*

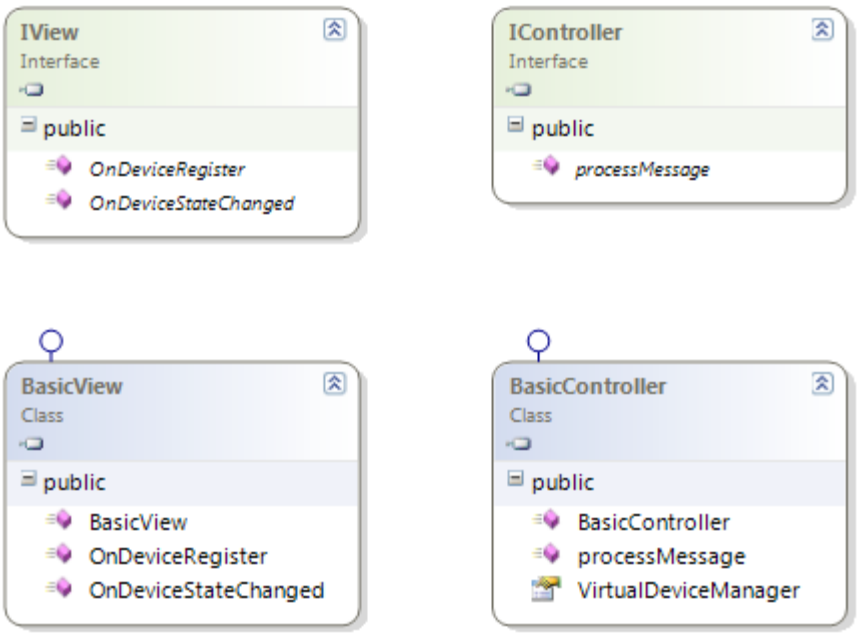
3.7. Sučelje pogled

Svaki uređaj ima upravitelj prijavljenih sučelja pogled „IView“ (Slika 6.) kojima se prate promjene svojstava uređaja. Sučelje deklarira metode koje se pozivaju kad se dogodi promjena u uređaju i kad se pogled prvi put registrira na uređaj. Kada dođe do promjene nekog od svojstva u uređaju poziva se metoda kojoj se predaje popis promjena u uređaju zajedno s imenom i identifikacijskim brojem uređaja. Konkretni pogled tim pozivom odgovarajuće reagira na promjenu u uređaju.

U svrhu jednostavnijeg budućeg razvoja dostupne su konkretne implementacije sučelja:

Razred „BasicView“ ispisuje promjene na konkretni u realnom vremenu tekstualni ponor, primjerice na standardni izlaz ili na neki drugi.

Adapter „NetAdapterView“ umjesto ispisa na tekstualni ponor, ostvaruje poslužitelj koji sluša na zadanom portu. Dolaskom obavijesti o promjeni, obavijest se proslijeđuje svim registriranim klijentima. Grafička simulacija u kombinaciji sa razredom „ProxyDevice“, koji se spaja kao klijent na poslužitelj u instanci razreda „NetAdapterView“, prima promjene o modelu i ažurira grafičku simulaciju.

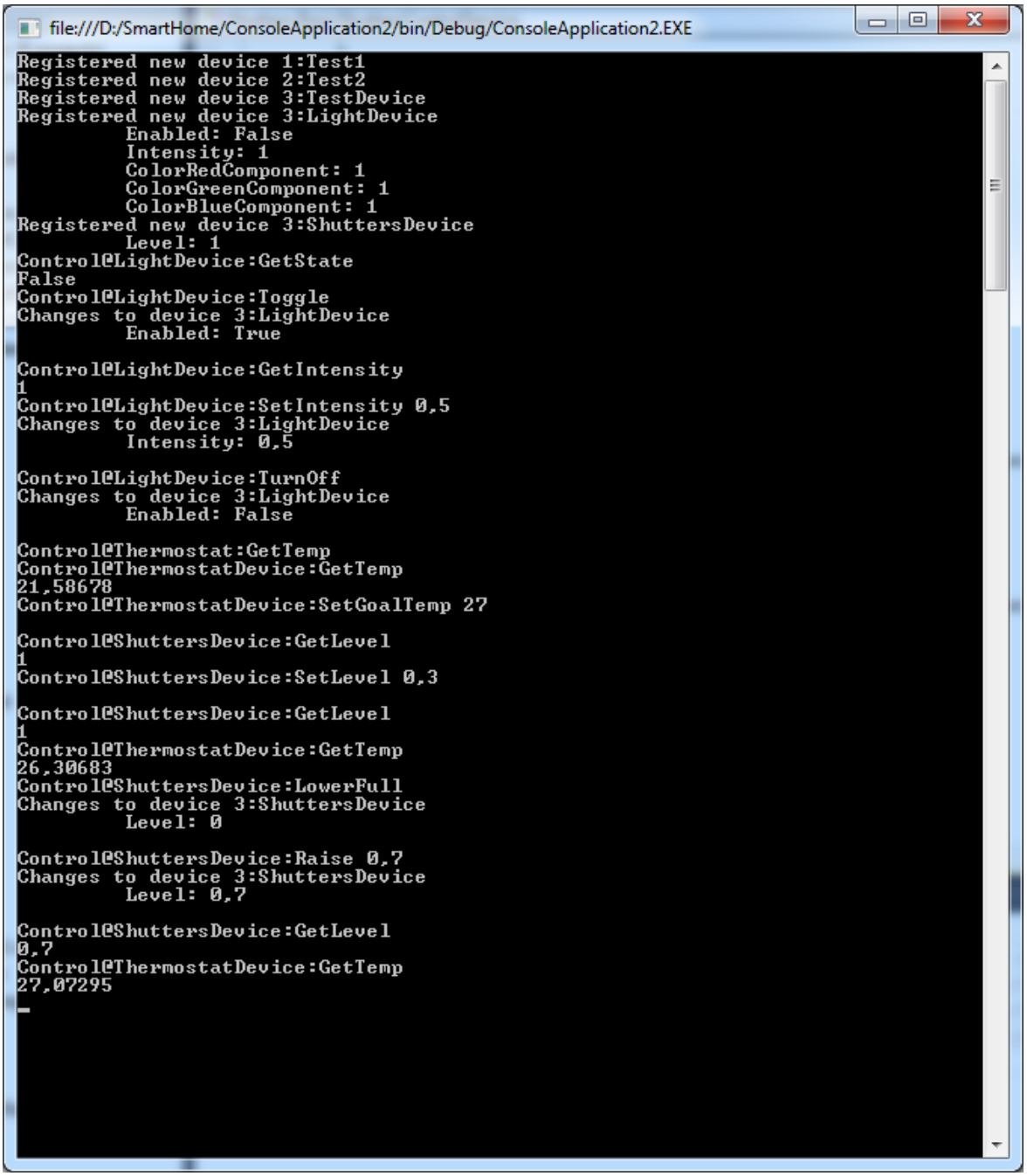


Slika 6. *Sučelja pogled i kontroler*

3.8. Sučelje kontroler

Kontroler je modeliran sučeljem „IController“ (Slika 6.) koje ima metodu za obradu poruka. Sustav dolazi sa konkretnom implementacijom jednostavnog kontrolera koji je modeliran razredom „BasicController“ (Slika 7.). On prima referencu na upravitelj uređaja kojeg koristi za pretragu uređaja i usluga kojima će proslijediti naredbu na obradu. Poruka ima format: *„usluga@uređaj: ime_naredbe [argument] ...“.* Prvi dio „usluga@uredaj“ koristi se za adresiranje usluge. Uređaje ili usluge može se adresirati imenom ili njihovim jedinstvenim identifikacijskim brojem. Drugi dio predstavlja naredbu s varijabilnim brojem argumenata koja se prosljeđuje usluzi na obradu.

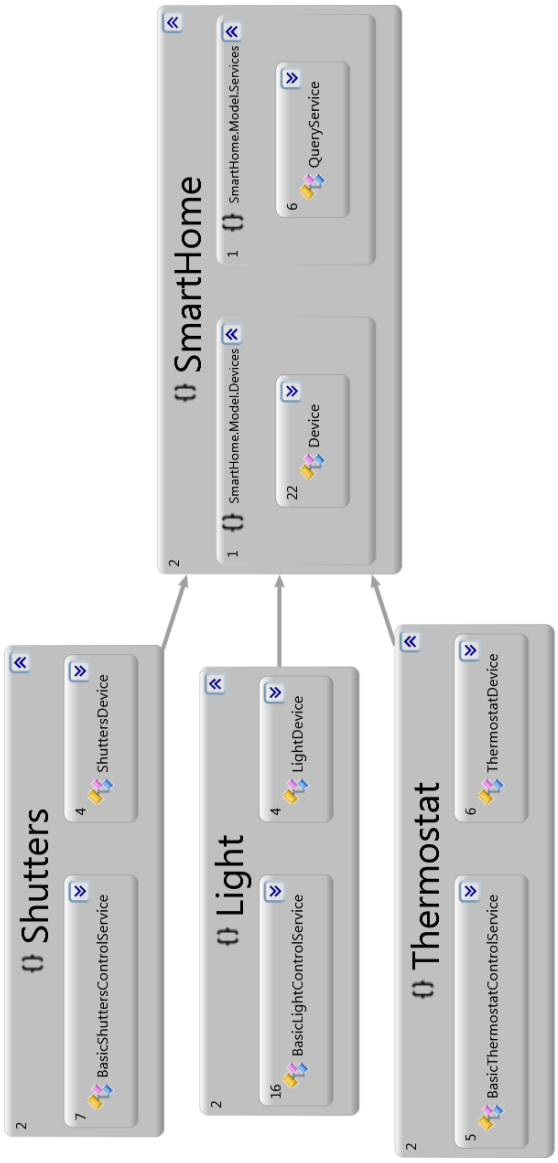
Internetska komunikacija ostvaruje se kroz adapter „NetAdapterController“ koji prima broj mrežnog porta i referencu na konkretnu instancu sučelja IController. Adapter održava više konkurentnih TCP veza preko kojih se prenose poruke. Poruke se dalje proslijeđuju zamotanom kontroleru. Format poruka koje klijenti šalju kontroleru identičan je zamotanom kontroleru. Kombinacijom razreda „NetAdapterController“ i „BasicController“ omogućeno je udaljeno upravljanje uređajima i uslugama. Klijent može implementirati bilo kakvo interaktivno sučelje dok god se komunikacija u konačnosti ostvaruje odabranim protokolom kontrolera [4].



Slika 7. *Kontroler u akciji*

3.9. Realizacija modela primjera

Svaka komponenta riješena je u zasebnom istoimenom projektu koji referencira projekt u kojem je riješen model. Svaki projekt komponente se kompajlira u premjestivu biblioteku (*.dll*) (Slika 8.) koja se može koristiti kao plugin i dinamički učitavati.



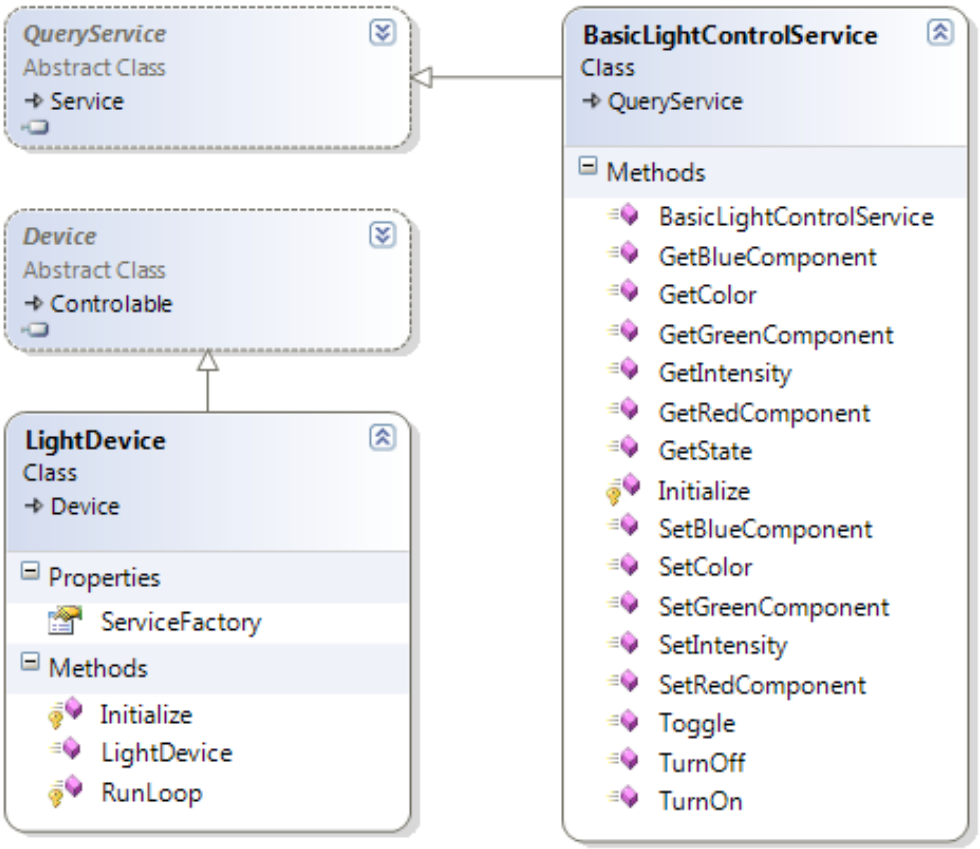
Slika 8. *Konkretni uređaji i usluge*

3.9.1. Komponenta Svjetlo

Komponenta svjetlo (Slika 9.) sastoji od uređaja „LightDevice“ koji sadrži svojstva:

- Upaljeno („Enabled“) – status svjetla
- Intenzitet („Intensity“) – jačina svjetla
- Komponenta boje („ColorComponent“) – komponenta boje (crvena, zelena, plava)

Usluga „BasicLigthControlService“ omogućava ispitivanje i postavljanje vrijednosti prethodno navedenih svojstava.



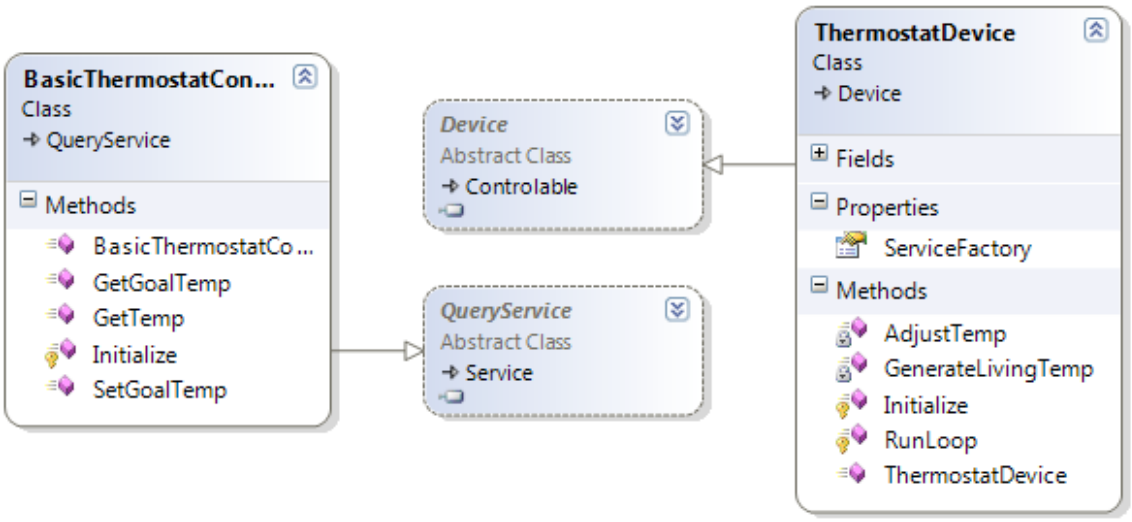
Slika 9. *Dijagram razreda svjetla*

3.9.2. Komponenta Termostat

Komponenta termostat (Slika 10.) sastoji se od jednostavnog uređaja „ThermostatDevice“ koji simulira temperaturu u sobi i utjecaj termostata na temperaturu. Uređaj sadrži svojstva:

- Trenutna temperatura („ActualTemp“) – simulirana temperatura prostorije
- Željena temperatura („GoalTemp“) – zadana temperatura koju termostat pokušava postići

Usluga „BasicThermostatControlService“ omogućava ispitivanje i postavljanje vrijednosti prethodno navedenih svojstava.



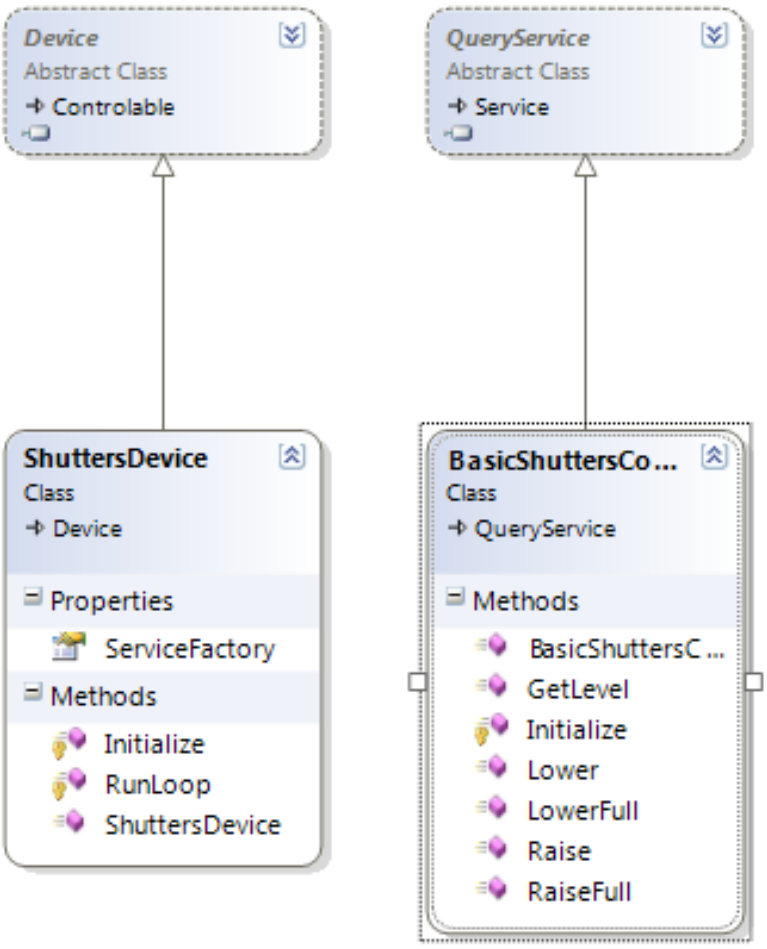
Slika 10. *Dijagram razreda termostata*

3.9.3. Komponenta Roleta

Komponenta roleta (Slika 11.) sastoji se od uređaja „ShuttersDevice“ koji simulira ponašanje rolete. Uređaj sadrži svojstvo:

- Razina („Level“) – Postotak podignutosti rolete

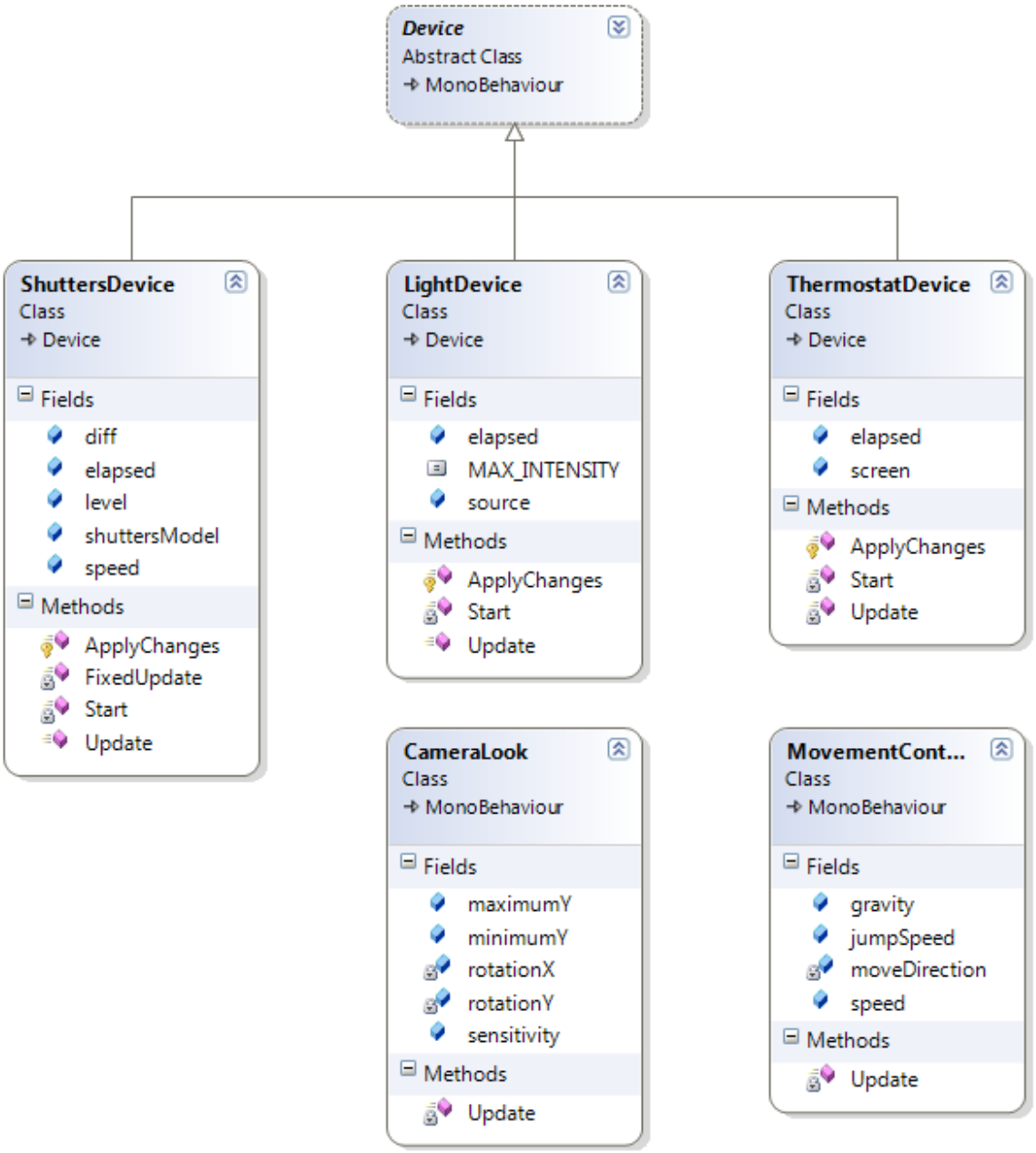
Usluga „BasicShuttersControlService“ omogućava namještanje razine podignutosti rolete.



Slika 11. *Dijagram razreda rolete*

4. Realizacija grafičke vizualizacije simulacije

Grafička vizualizacija smještena je u posebnom projektu i koristi referencu na model pametne kuće (Slika 12.).



Slika 12. *Glavne skripte vizualizacije*

4.1. Grafički engine – Unity

Grafički engine iznimno je popularni programski paket za razvoj igara i 3D simulacija. Izdan od razvojnog tima Unity Technologies još davne 2005. Omogućio je eksplozivan rast broja proizvoda, koji su razvijeni u malim timovima programera, za koje su inače potrebne veliki timovi. Jednostavno i efektivno razvojno okruženje za razvoj simulacija sa brzim pregledom prototipova, te podrška za multiplatformski razvoj su neki od razloga njegove velike popularnosti. Unity je baziran na programskom jeziku C/C++, a za skriptiranje se koriste jezici Javascript ili C#. Unity također koristi slobodnu reimplementaciju Microsoftove platforme .NET zvanu Mono što je jedan od razloga za korištenje dotičnog alata u svrhu razvoja ovog projekta [5].

4.2. Upravljanje grafičkom simulacijom

4.2.1. Upravljanje uređajima

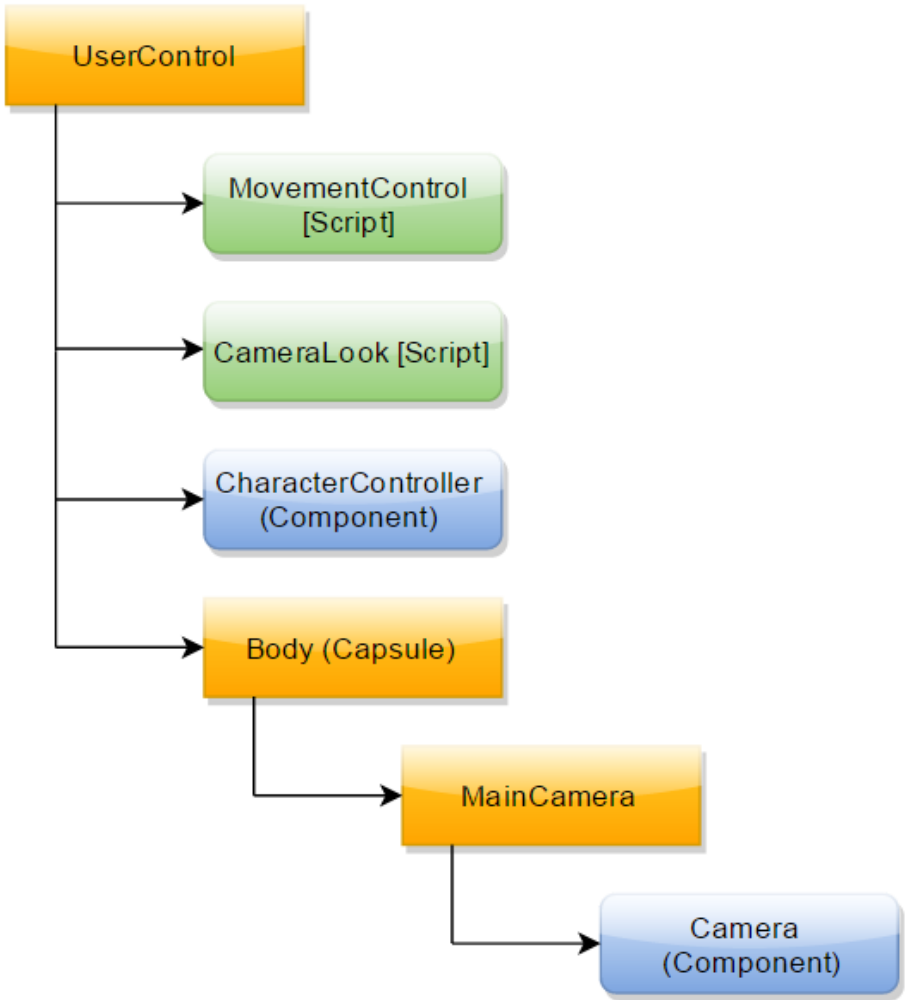
Uređaji se spajaju na poseban sustav za upravljanje događajima koji se dalje preko mreže spaja na već predodređeni „port“ odakle sustav prima poruke o promjenama uređaja. Poruke se prosljeđuju odgovarajućim uređajima koji se dalje vizualno prilagođavaju promjenama.

4.2.2. Dodavanje novih uređaja

Dodavanje uređaja u ovoj iteraciji trenutno nije dostupno dinamički, već se mora napraviti kroz Unity-ev uređivač simulacije. Projekt već dolazi s izrađenim komponentama (*„prefabs“*). Dovoljno je dodati novu komponentu u simulaciju i ispravno konfigurirati identifikacijski broj uređaja.

4.2.3. Upravljanje korisnikom

Korisnik simulaciju pametne kuće gleda iz perspektive prvog pogleda. S obzirom da korisnik sebe ne može vidjeti, on nema model, već je implementiran pomoću obične kapsule koja se koristi za računanje sudara sa okolinom. Za pomicanje korisnika koristi se standardna shema kontrole koja uključuje tipkovnicu za pokrete nogu i miša za pomicanje pogleda. Kontrola je implementirana u skriptama „CameraLook“ i „MovementControl“ (Slika 13.).



Slika 13. *Hijerarhija komponenti kontrole korisnika*

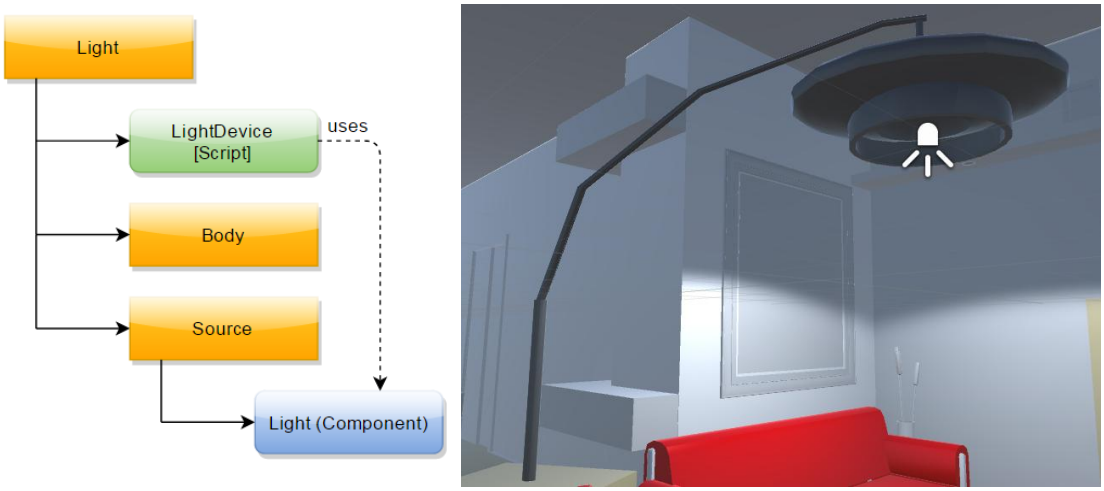
4.3. Realizacija vizualizacije primjera

U Unity grafičkom engine-u svaka objekt ima hijerarhiju koja se sastoji od više drugih objekata i komponenti. Kombinacijom željenih objekata i komponenti postiže se željeni rezultat. Komponente mogu biti razni elementi kao: modeli, primitivni oblici, razni kontroleri, izvori svjetla, skripte, elementi fizikalne simulacije...

4.3.1. Komponenta Svjetlo

Komponenta svjetlo sastoji se od praznog objekta kojem je pridružena skripta „LightDevice“ koja upravlja logikom svjetla. Glavnoj komponenti pridružen je objekt model koji predstavlja vizualnu reprezanticiju izvora svjetlosti (primjerice stolna ili stropna svjetiljka) i objekt točkastog izvora svjetlosti koji osvjetljava okolinu.

Skripti „LightDevice“ pridružuje se izvor svjetlosti kojem skripta direktno manipulira intenzitet, boju, i status rada.

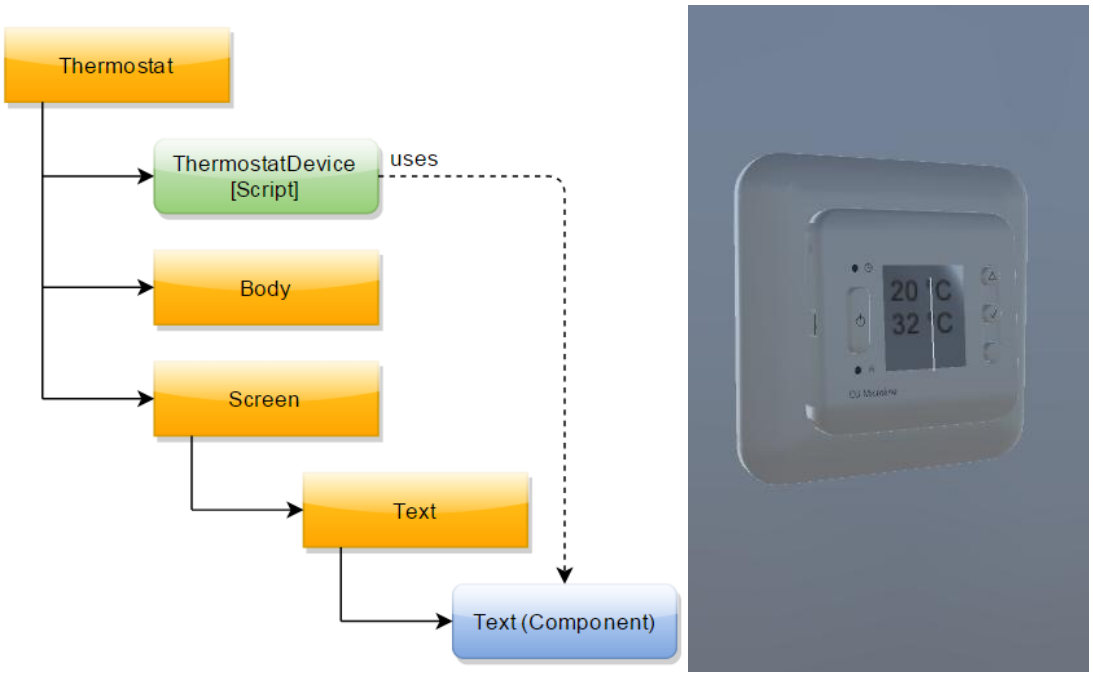


Slika 14. *Hijerarhija komponenti svjetla*

4.3.2. Komponenta Termostat

Komponenta termostat sastoji se od praznog objekta kojem je pridružena skripta „ThermostatDevice“ koja upravlja logikom termostata. Glavnoj komponenti pridružen je objekt model koji predstavlja vizualnu reprezentaciju uređaja termostat. Nadalje glavnoj komponenti je još pridružena UI platno s tekstualnom komponentom koja služi za ispis statusa termostata.

Skripti „ThermostatDevice“ pridružuje se tekstualna komponenta na kojoj skripta ispisuje stanje simulacije termostata.

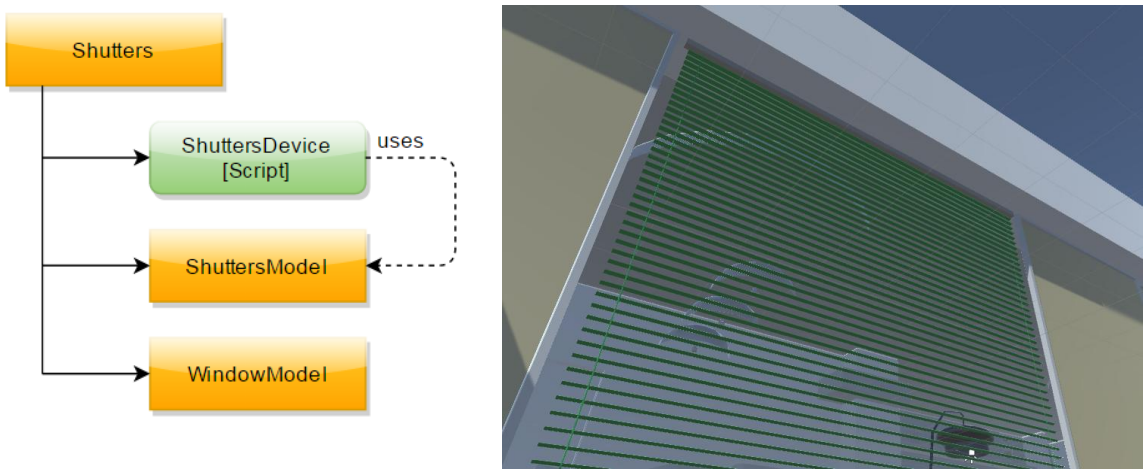


Slika 15. *Hijerarhija komponenti termostata*

4.3.3. Komponenta Roleta

Komponenta roleta sastoji se od praznog objekta koje je pridružena skripta „ShuttersDevice“ koja upravlja logikom rolete. Glavnoj komponenti pridružen je vizualni model prozora i rolete.

Skripti „ShuttersDevice“ pridružuje se transformacijska komponenta modela rolete. Skripta iz modela izvlači relevantne informacije o poziciji i veličini modela koje koristi za podizanje i spuštanje rolete na univerzalan način neovisno o stvarnoj veličini modela. U skripti se mogu namjestiti parametri razine rolete i brzine spuštanja i podizanja rolete.



Slika 16. *Hijerarhija komponenti rolete*

5. Zaključak

U ovom radu proučen je koncept pametne kuće i potencijalna realizacija simulacije navedenog. Pametna kuća je prirodna nastavak generalizacije postojećih kućnih pametnih sustava. Razvoj pametne kuće je vrlo popularno i istraživački aktivno područje u raznim granama informatičke industrije. Danas se već polako pojavljuju zasebni ekosustavi koji teže tom cilju. Pametna kuća je potencijalna revolucija kućnog života no još svi problemi nisu riješeni. Ideja ovog rada je da ponudi interaktivnu simulaciju pametne kuće u kojoj se kroz interakcije može istražiti što je potencijalno dobar pristup u razvoju pametne kuće, a što nije. Razvijeni rad može poslužiti kao osnova za daljnji razvoj simulacije pametne kuće. Rad nudi model simulacije s komponentama koje se dalje mogu lako razvijati u nove uređaje, kao i njihovu laku integraciju u postojeću simulaciju.

6. Literatura

[1] Service-Oriented Architecture (SOA). Retrieved June 2015, from https://msdn.microsoft.com/en-us/library/bb977471.aspx

[2] Brenda M. Michelson. Event-Driven Architecture Overview. Retrieved June 2015, from www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, D. Boyle: From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence

[4] Network Programming in the .NET Framework. Retrieved June 2015, from https://msdn.microsoft.com/en-us/library/4as0wz7t(v=vs.110).aspx

[5] Unity Manual. Retrieved June 2015, from http://docs.unity3d.com/

Internet of Things (IoT). Retrieved June 2015, from http://en.wikipedia.org/wiki/Internet_of_Things

Home automation. Retrieved June 2015, from http://en.wikipedia.org/wiki/Home_automation

Service-Oriented Architecture Retrieved June 2015, from http://en.wikipedia.org/wiki/Service-oriented_architecture

Event-Driven Programming Retrieved June 2015, from http://en.wikipedia.org/wiki/Event-driven_programming

Sažetak

Naslov: Simulacijski model pametne kuće

Ovaj rad predstavlja simulacijski model pametne kuće. Rad počinje predstavljanjem koncepta pametne kuće. Slijede koncepti i elementi na kojima je bazirana pametna kuća. Zatim je dan opis modela simulacije pametne kuće. Nakon toga slijedi opis realizacije modela pametne kuće zajedno s razvijenim programima za upravljanje modelom. Zadnje poglavlje predstavlja vizualizaciju modela pametne kuće u grafičkom engine-u Unity koji se spaja na prethodno realizirani model. Rad završava osvrtom na razvijen model i koncept pametne kuće.

Ključne riječi: pametna kuća, model simulacije pametne kuće, internet stvari, Unity

Abstract

Title: Smart home simulation model

This thesis represents smart home simulation model. Thesis begins with the introduction to the smart home concept. It is followed by concepts and elements that make up the basis of smart home, and the description of smart home simulation model. Following afterwards is the description of the realized smart home model, along with the developed programs for model control. Last chapter introduces smart home model visualization in Unity graphic engine which connects to formerly realized model. Thesis concludes with the remark on the developed model and smart home concept.

Keywords: smart home, smart home simulation model, internet of things, Unity