UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

IGOR KRIVEC

# MOBILNA APLIKACIJA ZA UPRAVLJANJE NAPRAV IN NADZOR NJIHOVE PORABE

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Rok Rupnik

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00362/2013
Datum: 07.01.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **IGOR KRIVEC**

Naslov: **MOBILNA APLIKACIJA ZA UPRAVLJANJE NAPRAV IN NADZOR NJIHOVE PORABE**

**MOBILE APPLICATION TO CONTROL DEVICES AND THEIR CONSUMPTION**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Izdelajte načrt z mobilno aplikacijo na upravljanje elektronskih naprav, ki so priključene v vtičnico. Pri izdelavi načrta temeljite na uporabi komunikacijske naprave Plugwise in njihove protokole.
Na podlagi načrta aplikacijo tudi razvijte na Android platformi.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Igor Krivec, z vpisno številko **63070112**, sem avtor diplomskega dela z naslovom:

*Mobilna aplikacija za nadzor naprav in merjenje električne energije*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- So elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 14. junija 2013 Podpis avtorja: _____

**Zahvala**

Za podporo pri izdelavi diplomske naloge se zahvaljujem staršem, prijateljem, predvsem Urbanu za izdatno pomoč ter nasvete in punci Urški. Zahvala gre tudi Laboratoriju za telekomunikacije na fakulteti za elektrotehniko, kjer so mi prijazno posodili strojno opremo Plugwise. Zahvaljujem se tudi sodelavcem iz tega laboratorija, ker so mi bili vedno pripravljeni priskočiti na pomoč pri reševanju nekaterih problemov pri razvoju aplikacije. Še posebej pa bi se rad zahvalil mentorju doc. dr. Roku Rupniku.

## Kazalo

## Povzetek

Evropska unija si želi zaradi ekonomskih in ekoloških razlogov zmanjšati rabo primarne električne energije. Prav tako so v zadnjih letih razcvet doživeli pametni mobilni telefoni. Cilj diplomske naloge je izdelava rešitve za nadzor porabe električne energije ter upravljanje elektronskih naprav v gospodinjstvu za domačega uporabnika, s pomočjo modulov skandinavskega podjetja Plugwise. Rešitev je sestavljena iz strežniškega in odjemalskega dela. V prvem delu diplomske naloge so na kratko predstavljena pametna omrežja in pametne hiše. Opisani so tudi moduli Plugwise in njihov komunikacijski protokol. V nadaljevanju je predstavljen operacijski sistem Android in uporabljene tehnologije ter razvojno okolje. Za tem je opisan razvoj same rešitve in sicer najprej strežnika, ki je razdeljen na tri dele: strežnik SOAP, podatkovna baza in storitev Windows. Strežniški del krmili module Plugwise, zapisuje porabo energije v podatkovno bazo ter iz nje tudi bere. Sledi predstavitev odjemalca, ki je mobilna aplikacija za naprave z operacijskim sistemom Android. Aplikacija komunicira s strežniškim delom in mu pošilja ukaze, s katerimi strežnik nato krmili module Plugwise. Na zaslonu aplikacije je prikazan tudi graf porabe električne energije za posamezne naprave. Proti koncu je ponazorjena še praktična uporaba rešitve, čisto na koncu pa še možnosti izboljšav.

**Ključne besede:** mobilni telefon, Android, pametno omrežje, pametna hiša, SOAP, Plugwise, ZigBee

## Abstract

One of the goals of European Union is to reduce the amount of primary electrical energy consumed by household users, a large part of which is wasted due to inefficiencies that occur at various stages. The goal of this thesis is to develop a software solution for monitoring consumption of electrical energy and management of electrical devices for household users. Solution is to be developed to work with modules by the scandinavian company Plugwise and is to be composed of server and client side. To begin with, smart grids and smarth houses are briefly introduced, followed by the description of Plugwise modules and their communication protocols. That will be followed by the description of Android operating system, its development enviornment and technologies used during the development. In the chapters afterwards we look at the solution itself, starting with server side and proceeding to the client side afterwards. Server side is composed of a SOAP server, relational database and Windows service and is used for controling Plugwise modules and communicating with database. Client side is primarily Android application used for sending commands via server to Plugwise modules and visualizing data for users. In the end practical implementation is shown and possible options for future modification are presented.

**Key words:** smartphone, Android, smart grid, smart house, SOAP, Plugwise, ZigBee

## 1. Uvod

Uporaba električne energije iz leta v leto narašča, zalog energetskih virov pa je čedalje manj. Evropska komisija je leta 2008 sprejela podnebno-energetski sveženj[1], ki vsebuje med drugim tudi zmanjšanje rabe primarne energije za dvajset odstotkov do leta 2020, glede na pričakovano raven, skozi izboljšanje energetske učinkovitosti. Za zmanjšanje porabe električne energije lahko v veliki meri poskrbi tudi potrošnik sam. Njegov motiv pa je, če ne okoljevarstveni, stroškovni, saj lahko z varčevanjem z električno energijo zmanjša svoje mesečne stroške.

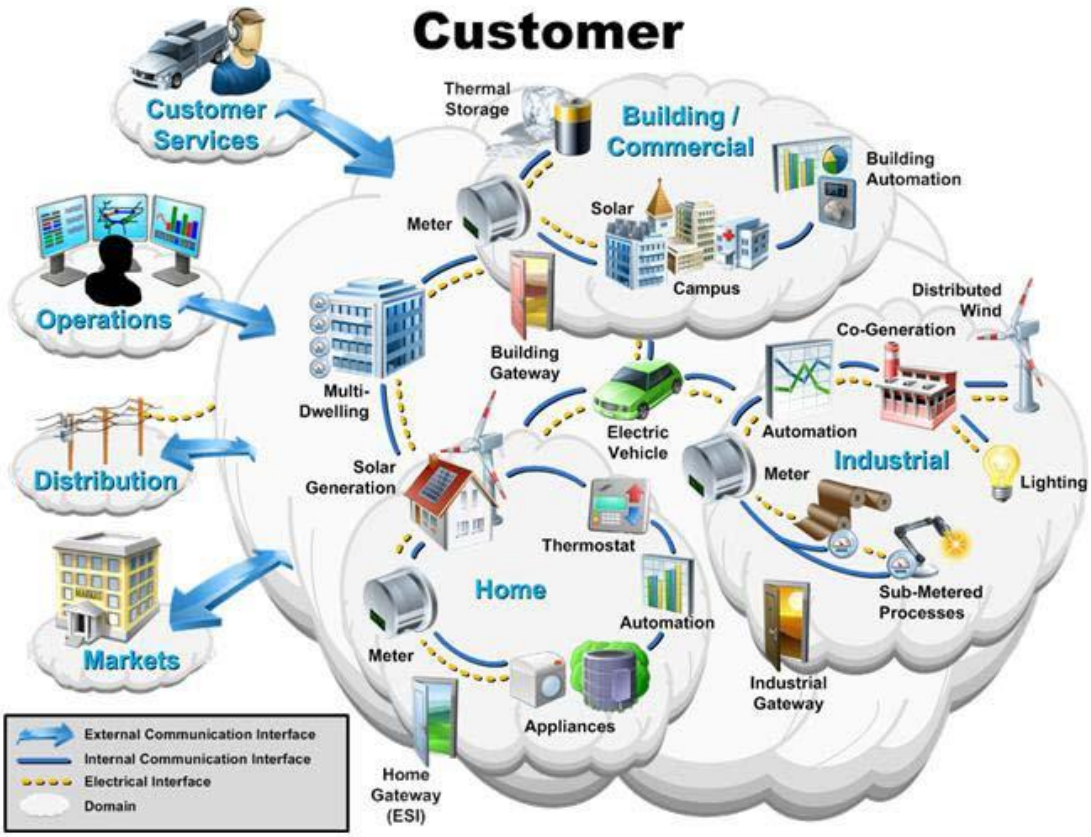
V zadnjih letih smo bili priča bliskovitemu razvoju mobilne industrije in v razvitem svetu skoraj ne najdemo več odraslega človeka, ki nima v lasti vsaj enega mobilnega telefona. Velik delež teh telefonov so tako imenovani pametni telefoni, ki lahko postanejo priročno orodje za spremljanje stroškov gospodinjstva ter tudi za upravljanje elektronskih aparatov v gospodinjstvu. Še več, preko pametnega, v splet povezanega mobilnika, lahko uporabnik sodeluje z distributerjem električne energije in mu pošilja podatke o porabi električne energije, distributer pa lahko te podatke uporabi ter si z njimi pomaga bolj smotrno razporejati električno energijo.

S tem v mislih je bila v okviru te diplomske naloge izdelana aplikacija za pametne telefone z operacijskim sistemom Android, ki končnemu uporabniku omogoča upravljanje elektronskih naprav v gospodinjstvu in nadzor nad porabo le-teh. Aplikacija uporabniku omogoča več možnosti vklopa in izklopa posamezne naprave, grafični prikaz meritev porabe energije ter stroškovni prikaz porabe izbrane naprave.

## 2. Pametna omrežja

Kaj so pametna omrežja? Enotne definicije tega pojma ni. Povzeto po združenju električne industrije Eureletric, je pametno omrežje elektroenergetsko omrežje, ki lahko stroškovno učinkovito vključuje delovanje in obnašanje vseh uporabnikov, ki so povezani v njem. Združenje inženirjev elektronike in elektrotehnike IEEE pametna omrežja definira kot integracijo primarnih, komunikacijskih in informacijskih tehnologij v izboljšano infrastrukturo elektroenergetskega sistema, s ciljem zadovoljevanja potreb po energiji in hkratnim kontinuiranim razvojem uporabniških aplikacij. Definicija mednarodne energijske agencije pa se glasi: pametna omrežja so elektroenergetska omrežja, v katerih se uporabljajo digitalne in ostale napredne tehnologije za spremljanje in upravljanje transporta električne energije iz vseh proizvodnih virov. Skupna definicija, združena iz zgornjih definicij se glasi: pametno omrežje je elektroenergetsko omrežje, ki lahko stroškovno učinkovito vključuje delovanje in obnašanje vseh uporabnikov, ki so povezani v njem – proizvodnih virov, odjemalcev in tistih, ki so oboje, s ciljem ekonomsko učinkovitega, trajnostnega sistema z nizkimi izgubami ter visokim nivojem zanesljivosti, kakovosti in varnosti dobave električne energije[2].

Primer pametnega omrežja, ki mora biti fleksibilno, dostopno, zanesljivo in ekonomično, prikazuje slika 1.

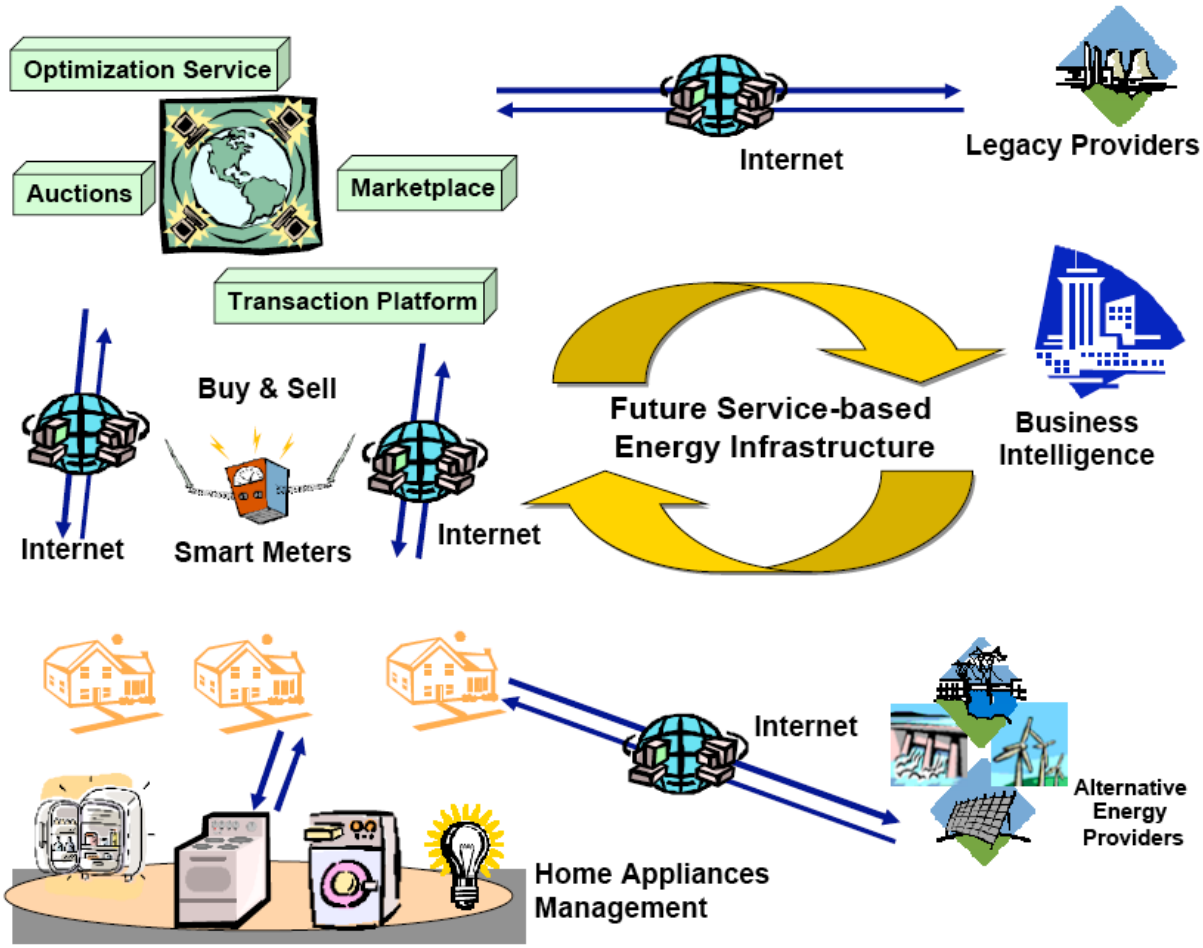



**Slika 1 - pametno omrežje**

## 2.1. Vloga pametnih hiš in končnega uporabnika

Po podatkih iz leta 2010 so gospodinjstva v Evropi odgovorna za 26,65 odstotkov vse porabljene električne energije[3]. Trenutni električni sistem pa ne vsebuje aktivnega sodelovanja domov in stavb. To precej omejuje učinkovitost celotnega električnega sistema in truda za bolj smotrno porabo električne energije. Če želimo zagotoviti energetsko učinkovitost in obstojnost naslednje generacije, potrebujemo tudi električni sistem nove generacije. Takšno omrežje temelji na interakciji pametnih hiš s pametnim omrežjem. Takšna arhitektura omrežja bi omogočala združevanje pametnih hiš kot inteligentno, omreženo sodelovanje in ne kot izolirane, pasivne enote v energetski mreži[4]. Na sliki 2 je ilustrirana vključitev pametnih hiš v pametno omrežje.

Dokler pa se to ne zgodi, lahko končni uporabnik za nekatere stvari poskrbi sam. Na tržišču je kar nekaj strojne opreme, ki omogoča hišno avtomatizacijo, kot tudi spremljanje porabe in stroškov v gospodinjstvu porabljene električne energije. V tej diplomski nalogi smo za komplet takšne strojne opreme razvili rešitev za končnega uporabnika, s katero lahko stori korak v smeri proti električnemu sistemu nove generacije.




**Slika 2 - Vključitev pametnih hiš v pametno omrežje**

## 2.4. Moduli Plugwise

V tem poglavju so predstavljeni v diplomski nalogi uporabljeni moduli podjetja Plugwise. Module smo si sposodili v Laboratoriju za telekomunikacije na fakulteti za elektrotehniko.

### 2.4.1. Sestavni deli

Aplikacijo za nadzor porabe in krmiljenje naprav smo izdelali za krmiljenje strojne opreme home starter kit, ki ga izdeluje podjetje Plugwise. Strojna oprema je sestavljene iz glavnega krmilnika, ki so ga v podjetju poimenovali 'stick', mrežnega krmilnika (poimenovan je 'circle+'), ter ostalih vozlišč v omrežju('circles'). V našem primeru je navadno vozlišče le eno, isto vlogo pa opravlja tudi mrežni krmilnik, ki ima poleg osnovne, še funkcijo komuniciranja z glavnim krmilnikom. Deli strojne opreme so prikazani na slikah 3, 4 in 5.

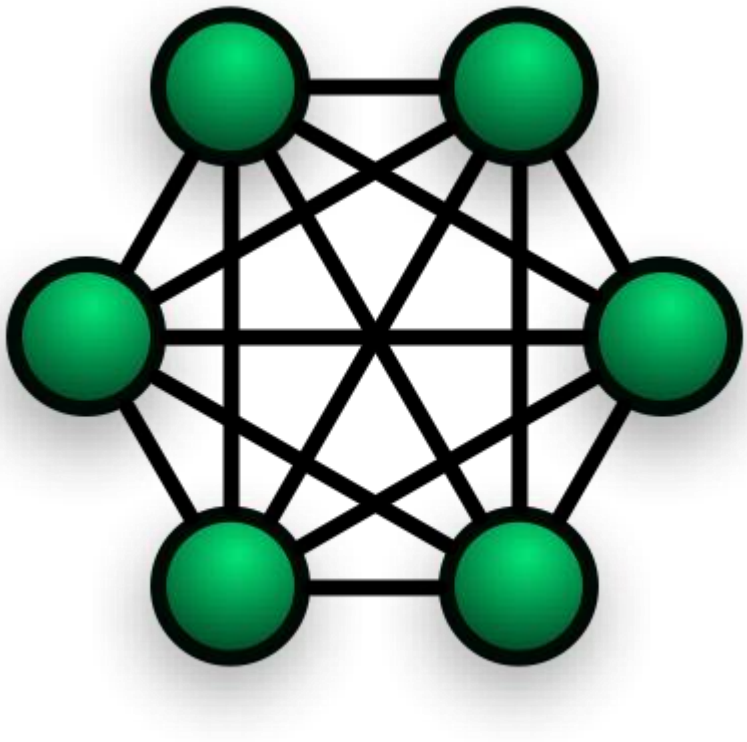
**Slika 3 - vozlišče**

**Slika 4 - mrežni krmilnik**

**Slika 5 - glavni krmilnik**

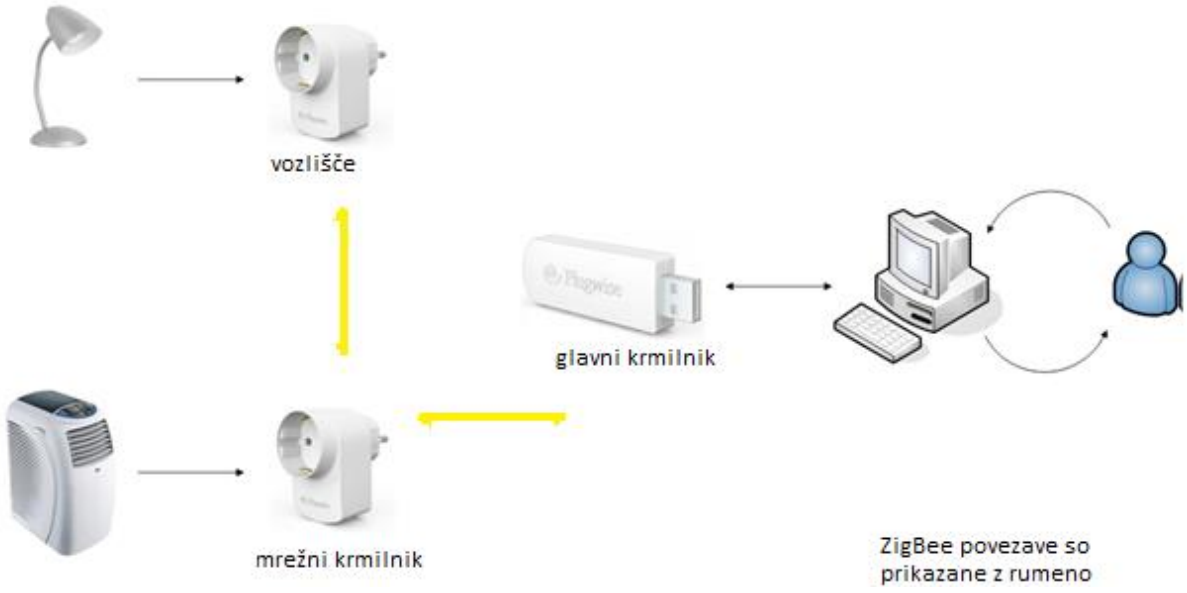
### 2.4.2. Komunikacija

Plugwise naprave se med sabo sporazumevajo preko protokola ZigBee, ki temelji na standardu IEEE 802. ZigBee je zanesljiv in porabi malo energije, zaradi česar je prava izbira za komunikacijo med tovrstnimi napravami. Vozlišča in omrežni krmilnik so med sabo povezani v zankasto omrežje. Zankasto omrežje je omrežje, kjer vsako vozlišče poleg tega da pošilja svoje podatke, tudi usmerja promet drugih vozlišč (slika 6).



**Slika 6 - zankasto omrežje[5]**

Vsa vozlišča v omrežju pošiljajo svoje podatke na mrežni krmilnik, ta pa jih pošlje glavnemu krmilniku ki je priključen v serijska vrata računalnika. Komunikacija med glavnim krmilnikom in računalnikom pa poteka preko protokola, razvitega s strani podjetja Plugwise. Ta protokol je z obratnim inženiringom uspel razvozlati Maarten Daamen ter ga predstavil v dokumentu z naslovom Plugwise Unleashed, ki je dostopen na spletu. Slika 7 prikazuje komunikacijo med vozlišči, mrežnim krmilnikom, glavnim krmilnikom in računalnikom.



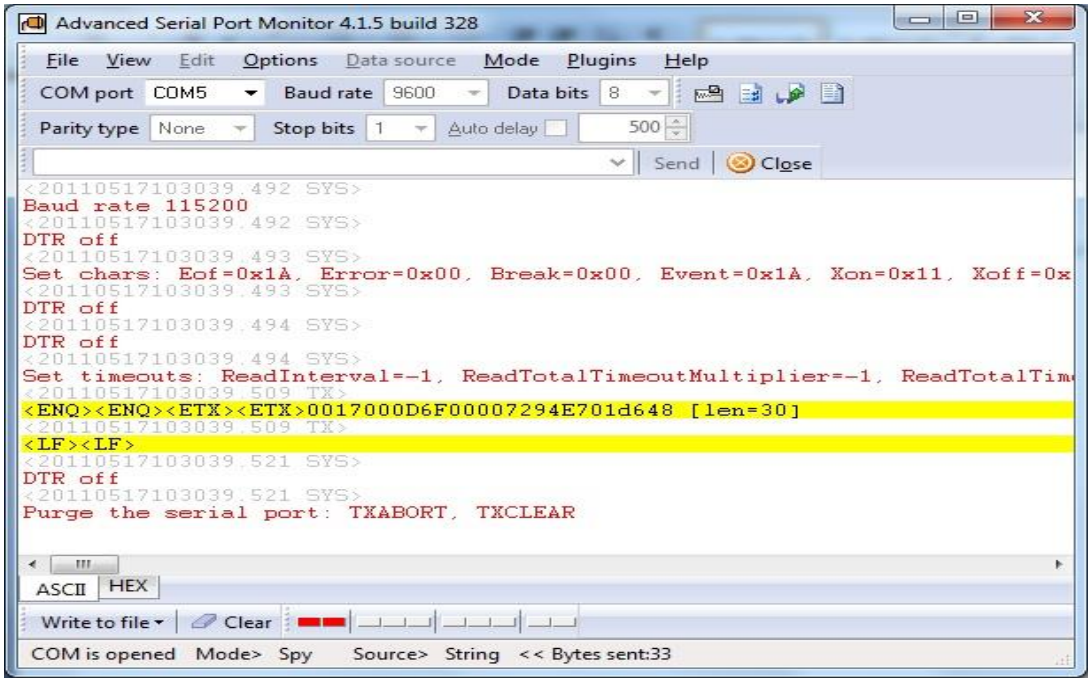

**Slika 7 - komunikacija Plugwise**

### 2.4.3 Zahtevki komunikacije

V nadaljevanju so opisani zahtevki komunikacije med računalnikom in glavnim krmilnikom Plugwise, ki jih je v svojem dokumentu razložil Maarten Damen. Priložene slike od 8 do 11 prikazujejo stanja na serijskem portu ob izvedbi zahtevka.

#### 2.4.3.1 Vklop naprave

Računalnik pošlje zahtevo: 0017000D6F00007294E701d648. Zahteva (vklop) se izvrši, nazaj ne dobimo potrditve in ne rezultata.



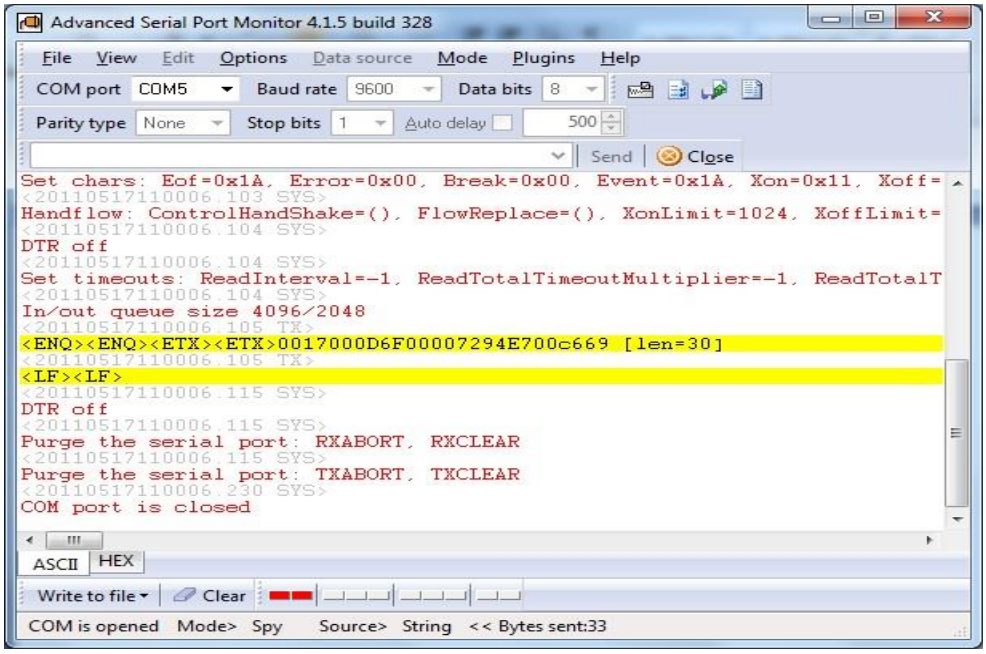

**Slika 8 - zahteva po vklopu naprave**

Razlaga zahteve:

- 0017: ukaz za vklop naprave (integer)
- 000D6F00007294E7: MAC naslov modula (nepredznačen 64-bitni integer)
- 01: status modula. Modul je vklopljen (boolean)
- d648: CRC kontrolna vsota

### 2.4.3.2 Izklop naprave

Računalnik pošlje zahtevo: 0017000D6F00007294E700d648. Zahteva (izklop) se izvrši, nazaj ne dobimo potrditve in ne rezultata.



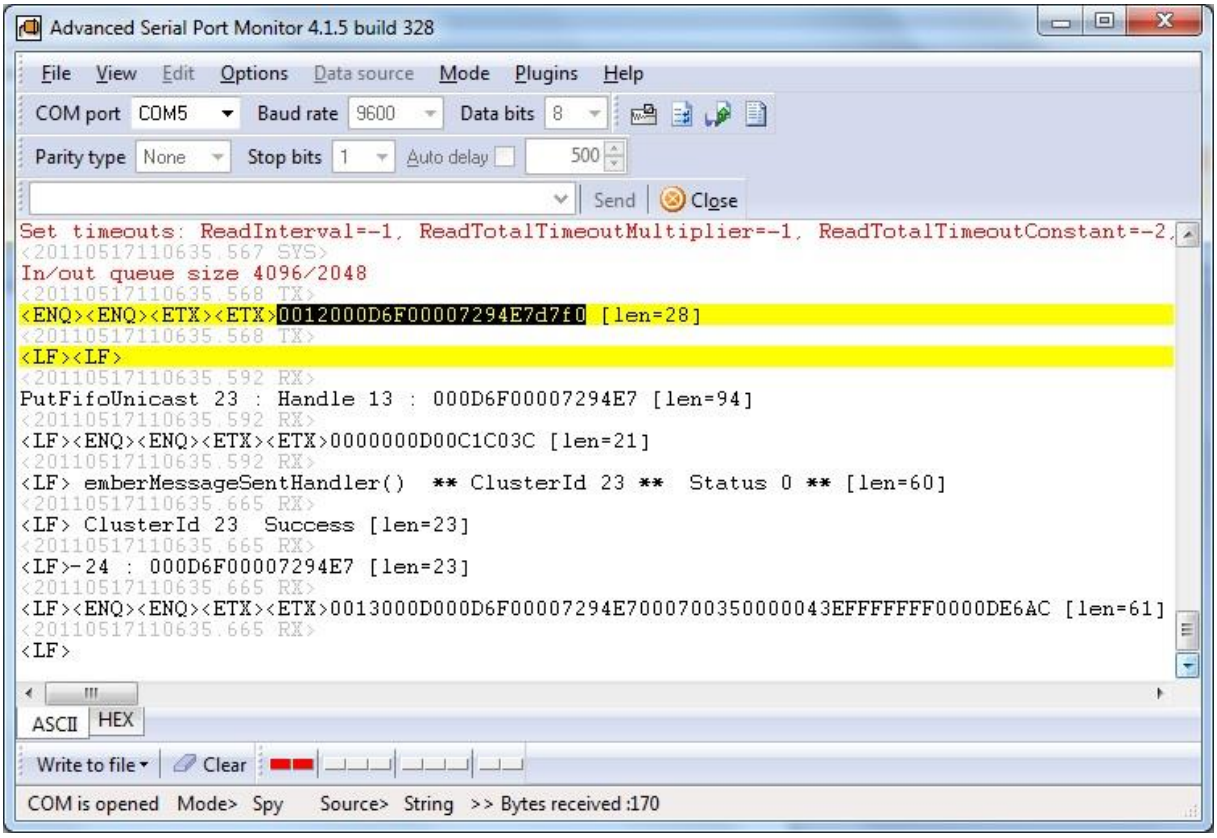

**Slika 9 - zahteva po izklopu naprave**

Razlaga zahteve:

- 0017: ukaz za vklop naprave (integer)
- 000D6F00007294E7: MAC naslov modula (nepredznačen 64-bitni integer)
- 00: status modula. Modul je izklopljen (boolean)
- C669: CRC kontrolna vsota

### 2.4.3.3 Trenutna moč naprave

Računalnik pošlje zahtevo: 0012000D6F00007294E7d7f0. Zahteva se izvrši, nazaj pa dobimo potrditev in rezultat zahteve.




**Slika 10 - zahteva po izklopu naprave**

Razlaga zahteve:

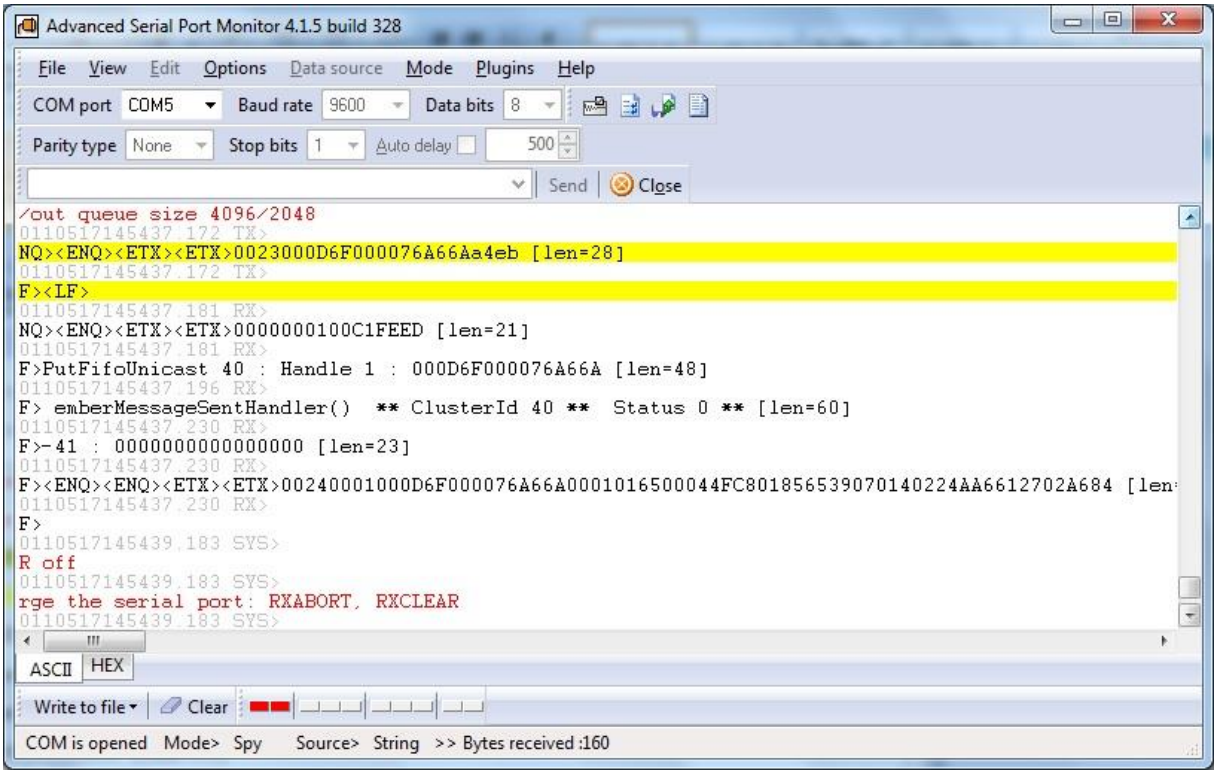
- 0012: koda ukaza za zahtevo po trenutni moči (integer)
- 000D6F00007294E7: MAC naslov modula (nepredznačen 64-bitni integer)
- C669: CRC kontrolna vsota

Razlaga rezultata zahteve:

- 0013: odzivna koda za zahtevo po informaciji o trenutni moči
- 000D: sekvenčna številka povezana z zahtevo ali odgovorom
- 000D6F00007294E7: MAC naslov modula
- 0007: število impulzov, ki temelji na uporabi enosekundnega intervala
- 0035: število impulzov, ki temelji na uporabi osemsekundnega intervala
- 0000043EFFFFFFF0000: neznano
- E6AC: CRC kontorolna vsota

### 2.4.3.4 Status

Program pošlje zahtevo po informaciji naprave: 0023000D6F000076A66Aa4eb. Nazaj dobimo rezultat:
00240001000D6F000076A66A0001016500044FC80185653907014022 4AA6612702A684




**Slika 11 - zahteva po statusu naprave**

Razlaga zahteve:

- 0023: koda zahteve
- 000D6F000076A66A: MAC naslov modula
- CRC kontrolna vsota

Razlaga odgovora:

- 0024: odzivna koda za zahtevo po informaciji naprave
- 0001: sekvenčna številka povezana z zahtevo ali odgovorom
- 000D6F000076A66A: MAC naslov naprave
- 00: leto modulove interne ure šestnajstiškem formatu. Če želimo dobiti trenutno leto, moramo prišteti 2000.
- 01: mesec modulove interne ure šestnajstiškem formatu
- 0165: število minut modulove interne ure v šestnajstiškem formatu
- 00044FC8: trenutni log naslov iz katerega pridobimo podatke o medpomnilniku porabe
- 01: naprava je trenutno vklopljena (če bi bilo 00 bi bila naprava izklopljena)
- 85: frekvenca na kateri deluje vtikač v šestnajstiškem formatu
- 653907014022: verzija strojne opreme modula
- 4AA66127: verzija programske opreme modula
- 02: neznano
- A684: CRC kontrolna vsota

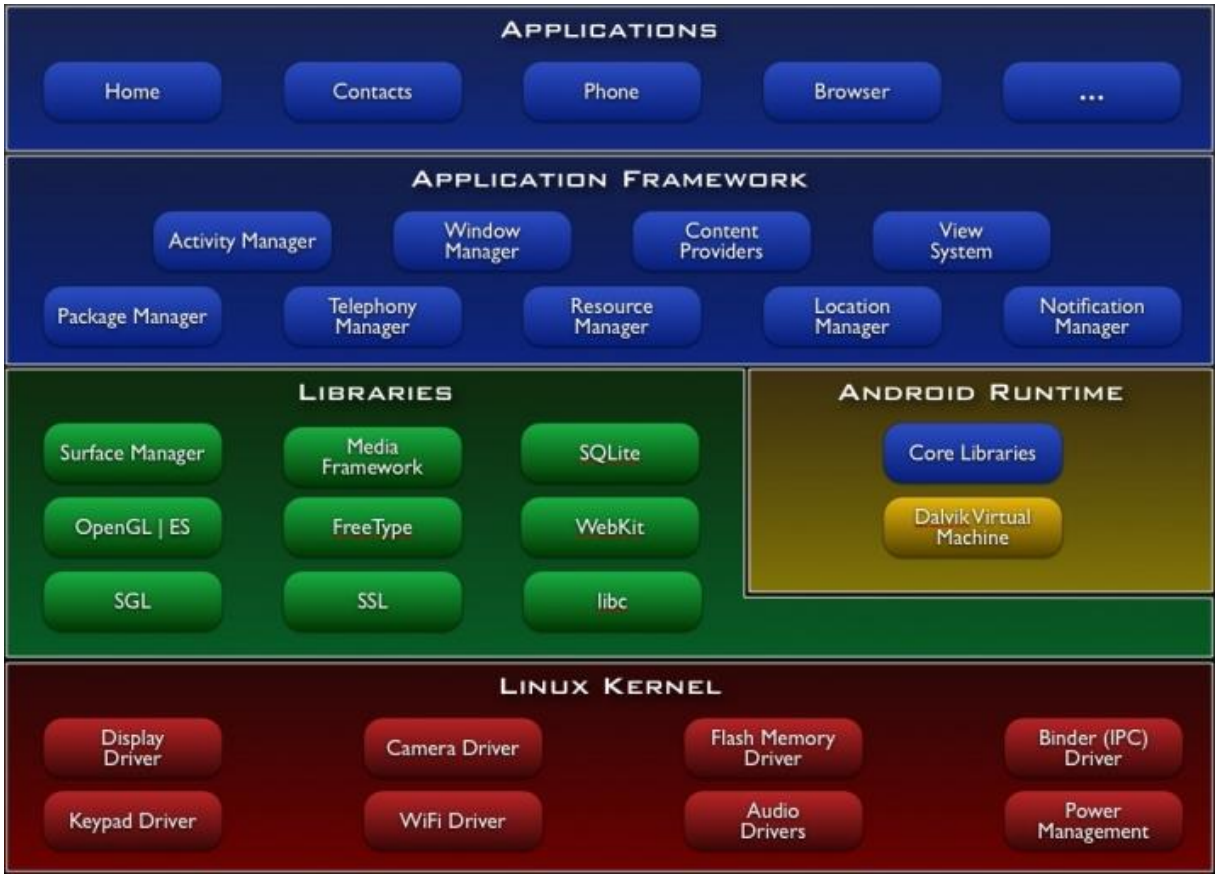
# 3. Operacijski sistem Android

## 3.1 Zgodovina

Podjetje Android so leta 2003 z željo razviti pametnejše mobilne naprave, ki se zavedajo uporabnikove lokacije in želja ustanovili Andy Rubin, Rich Miner, Nick Sears in Chris White[6]. Na začetku se je podjetje ukvarjalo z razvojem operacijskega sistema za digitalne kamere in fotoaparate, a so se, ko so ugotovili, da je to premajhen trg preusmerili v razvoj operacijskega sistema, ki bi bil konkurenčen Symbianu in Microsoftovemu operacijskemu sistemu Windows Mobile. 17. avgusta leta 2005 je podjetje Android kupil Google in s tem nakazal da želi prodreti na trg mobilnih naprav. Novembra 2006 je bila ustanovljena zveza Open Handset Alliance, ki pod vodstvom Googla združuje mobilne operaterje, proizvajalce mobilnih naprav, proizvajalce čipov, proizvajalce programske opreme in komercializacijska podjetja. Cilj združenja je razvijati odprte standarde za mobilne naprave. Leto dni kasneje, novembra 2007 je bil operacijski sistem Android razkrit kot prvi produkt združenja. Prvi komercialno dostopen telefon z Androidom je bil HTC Dream, ki je prišel v prodajo 22. oktobra 2008[6, 7, 8].

## 3.2 Arhitektura

Slika 12 prikazuje arhitekturo operacijskega sistema Android. Posamezni sestavni deli so razloženi v nadaljevanju.




**Slika 12 – Arhitektura operacijskega sistema Android**

### 3.2.1 Aplikacije

Aplikacije so napisane v programskem jeziku Java. Orodja razvojnega okolja prevedejo izvorno kodo v arhivsko datoteko s končnico .apk. Vsa koda v eni datoteki .apk je smatrana kot ena aplikacija in je datoteka, ki jo naprave z operacijskim sistemom Android uporabijo, da namestijo aplikacijo.

Ko je nameščena, vsaka aplikacija teče v svojem peskovniku (angleško: sandbox), kar pomeni:
- Android je večuporabniški operacijski sistem, v katerem je vsaka aplikacija en uporabnik.
- Sistem privzeto vsaki aplikaciji dodeli uporabniško identifikacijsko številko (user id) (to številko pozna le sistem, aplikaciji pa je neznana). Sistem dodeli vsem datotekam, ki jih aplikacija uporablja, dovoljenja tako, da samo uporabniška številka dodeljena tej aplikaciji lahko dostopa do njih.
- Vsak proces teče v svojem navideznem stroju, torej vsaka aplikacija teče izolirana od drugih aplikacij.

- Vsaka aplikacija privzeto teče v svojem Linux procesu. Android zažene proces, ko mora biti posamezna komponenta aplikacije zagnana in ugasne proces, ko komponenta ni več potrebna ali ko mora sistem sprostiti spomin za druge aplikacije.

Na ta način je pri Androidu poskrbljeno za varnost, saj lahko vsaka aplikacija dostopa le do komponent, ki so potrebne, da aplikacija opravi svoje delo in ne more dostopati do delov operacijskega sistema, za katere nima dodeljenih pravic. Vseeno pa je možno, da aplikacije delijo podatke z drugimi aplikacijami. Možno je ročno dodeliti dvema aplikacijama isto uporabniško identifikacijsko številko in v tem primeru ti aplikaciji lahko dostopata do datotek v drugi aplikaciji. Da varčujemo s sistemskimi viri, lahko aplikacijama z isto identifikacijsko številko dodelimo, da tečeta v istem Linux procesu in si delita isti navidezni stroj. Aplikacija lahko tudi zahteva dostop do podatkov kot je uporabnikov telefonski imenik, sporočila SMS, podatki na kartici SD in še več. Ta dovoljena se nahajajo v Android manifestu (podrobneje je razložen v nadaljevanju) in jih mora uporabnik pred namestitvijo aplikacije potrditi.

### 3.2.2 Aplikacijsko ogrodje

Aplikacijsko ogrodje so deli operacijskega sistema s katerimi aplikacija neposredno komunicira. Ti programi upravljajo osnovne naloge operacijskega sistema, kot so upravljanje z viri, upravljanje s klici in podobno. Pomembni deli aplikacijskega ogrodja so razloženi v spodnji tabeli 1.

| | |
|---|---|
| Activity Manager | Upravljanje z življenjskim ciklom aktivnosti aplikacije. |
| Content Providers | Upravljajo deljenje podatkov med aplikacijami. |
| Telephony Manager | Upravlja vse glasovne klice. |
| Location Manager | Upravlja z lokacijo telefona, bodisi preko satelitov GPS ali brezžičnih in mobilnih omrežij. |
| Resource Manager | Upravlja z različnimi tipi virov, ki jih uporabljamo v aplikaciji. |

**Tabela 1 - aplikacijsko ogrodje**

### 3.2.3 Knjižnice

Naslednja plast aplikacijskega ogrodja so knjižnice operacijskega sistema. Ta plast napravi omogoča, da obvladuje različne podatke. Knjižnice so napisane v jezikih C in C++ in so specifične za posamezno strojno opremo. Sledi opis nekaj pomembnejših knjižnic operacijskega sistema.

- Surface manager: uporabljen je za komponiranje upravljavca oken z slikovnim medpomnjenjem. To pomeni, da lahko rišemo po ekranu, ampak naše risbe se shranijo v slikovni medpomnilnik. Tam so združene z drugimi risbami in tvorijo končen zaslon, kot ga vidi uporabnik. Ta pomnilnik omogoča transparentna okna v operacijskem sistemu Android.

- Media framework: priskrbi različne multimedijske kodeke, ki omogočajo snemanje in predvajanje različne multimedijske vsebine.

- SQLite: SQLite je pogon podatkovne baze uporabljen v operacijskem sistemu za shranjevanje podatkov.

- WebKit: je pogon iskalnika uporabljen za prikaz HTML vsebine.

- OpenGL: knjižnica uporabljena za upodabljanje dvodimenzionalne in tridimenzionalne grafike.

### 3.2.4 Android izvajalno okolje

Izvajalno okolje Android sestoji iz navideznega stroja Dalvik in javanski jedrnih knjižnic.

Navidezni stroj Dalvik je vrsta navidezna stroja Java (JVM) in je uporabljen v napravah Android za poganjanje aplikacij in je prilagojen za nizko procesorsko moč in majhno porabo spominskih virov. V nasprotju z javanskim navideznim strojem, navidezni stroj Dalvik ne zaganja datotek s končnico .class, ampak datoteke s končnico .dex. Te datoteke so med prevajanjem zgrajene iz datotek .class in omogočajo večjo učinkovitost v okoljih z malo spominskimi in procesorskimi viri. Navidezni stroj Dalvik dovoljuje ustvarjenje več primerkov navidezna stroja sočasno, kar zagotavlja varnost, upravljanje s spominom in podporo nitenju.

Javanske jedrne knjižnice so različne od Java SE in Java ME knjižnic, a vseeno zagotavljajo večino funkcionalnosti definiranih v Java SE knjižnicah.

### 3.2.5 Jedro Linux

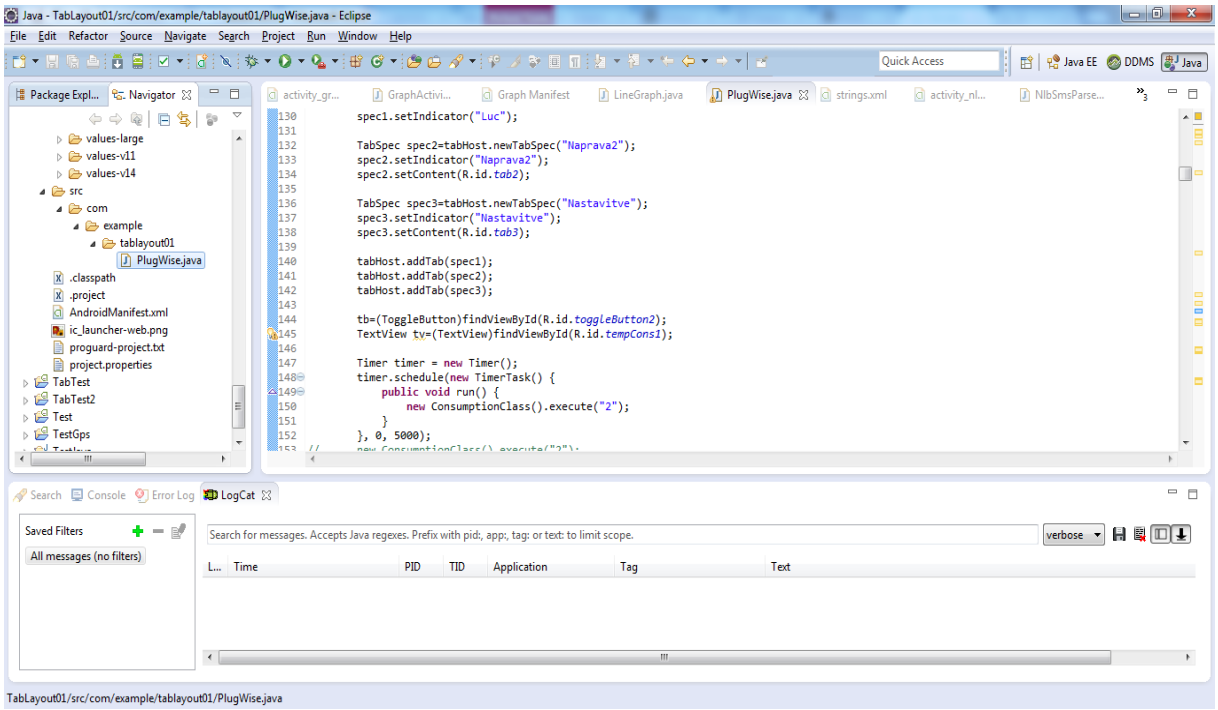
Najosnovnejša plast operacijskega sistema Android pa je jedro Linux. Celoten operacijski sistem je, z nekaj popravki s strani Googla, zgrajen na jedru Linux 2.6. Jedro komunicira s strojno opremo in vsebuje vse bistvene gonilnike strojne opreme. Jedro Linux prav tako igra vlogo abstrakcijske plasti med plastmi strojne in programske opreme. Android uporablja Linux za vse svoje temeljne funkcionalnosti, kot so upravljanje s spominom, upravljanje s procesi, mreženje in varnostne nastavitve. Ker je Android zgrajen na jedru Linux, je prenašanje le-tega na različno strojno opremo relativno lahko opravilo[].

## 4. Tehnologije in razvojno okolje

Za izdelavo aplikacije za nadzor porabe energije smo uporabili razvojno okolje Eclipse in programski jezik Java za izdelavo aplikacije za pametne telefone, ter razvojno okolje Microsoft Visual Studio in programski jezik C# za strežniški del aplikacije. Pri izdelavi strežniškega dela smo si pomagali z odprtokodno knjižnico PlugwiseLib, ki omogoča komuniciranje z moduli Plugwise mimo uradne programske opreme podjetja. Knjižnica je napisana v programskem jeziku C#, kar je tudi razlog, da smo razvoj strežniškega dela izbrali ta jezik. Za hrambo podatkov je bila uporabljena podatkovna baza MySql, za komunikacijo med strežniškim in odjemalskim delom pa protokol Soap, ki je bil implementiran s pomočjo tehnologije Asp.NET Web Services. Vse tehnologije in razvojni okolji so podrobneje opisani v nadaljevanju.

### 4.1. Eclipse in ADT

Eclipse je razvojno okolje, ki podpira razvoj v večih programskih jezikih, med drugim tudi v Javi, ki se uporablja za razvoj mobilnih aplikacij za operacijski sitem Android. ADT(Android Development Tools – razvojna orodja android) pa je vtičnik za Eclipse, ki omogoča prevajanje, testiranje, razhroščevanje in pakiranje aplikacij Android. Na sliki 13 je zaslonski posnetek razvojnega okolja Eclipse z vtičnikom ADT:

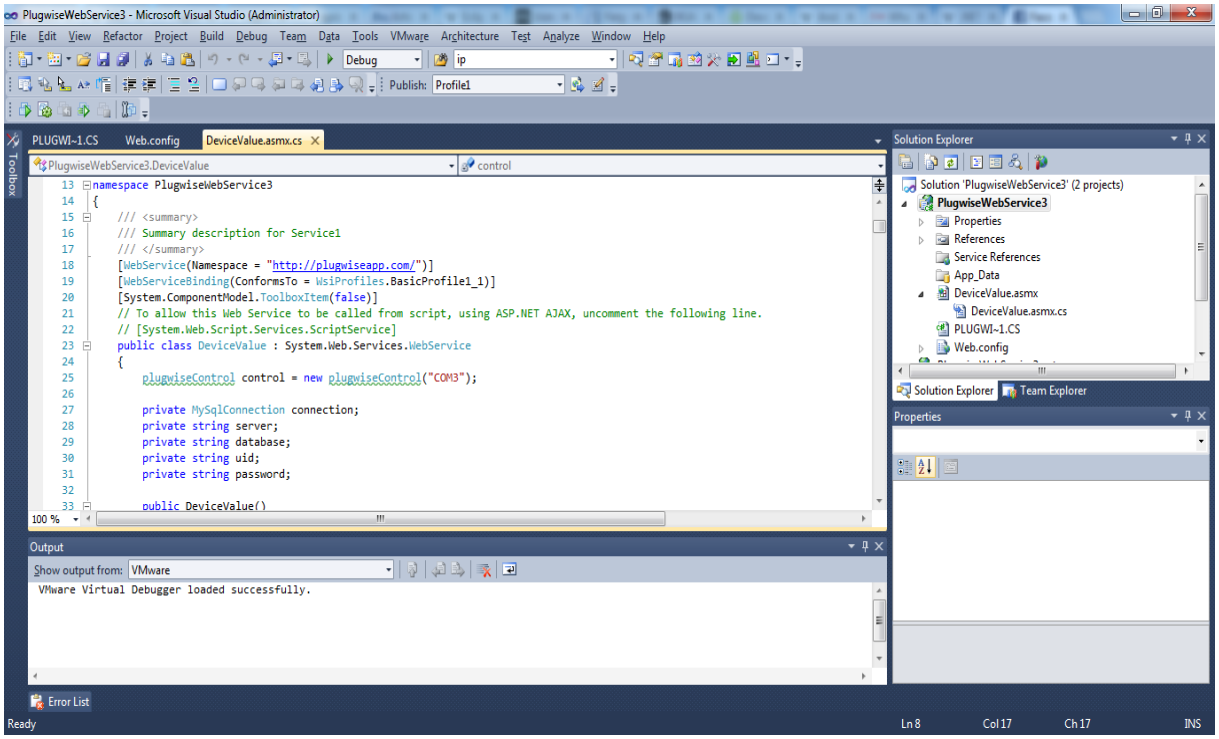



**Slika 13 - Eclipse z vtičnikom ADT**

## 4.2. Microsoft Visual Studio in C#

Za razvoj strežniškega dela rešitve smo uporabili Microsoft Visual Studio (slika 14), ki je s strani podjetja Microsoft razvito razvijalno okolje, namenjeno razvoju v okolju .NET. Okolje, podobno kot Eclipse, vsebuje urejevalnik kode, razroščevalnik in grafični urejevalnik. Na voljo je več različic razvijalne okolja, mi smo v diplomski nalogi uporabili Visual Studio 2010, ki je dostopen preko programa MSDNAA.

C# je objektno orientiran programski jezik, ki ga je razvilo podjetje Microsoft v okviru iniciative .NET in naj bi povzemal prednosti programskih jezikov C, C++ in Java. Njegova sintaksa je z nekaj razlikami podobna sintaksi jezika C++.



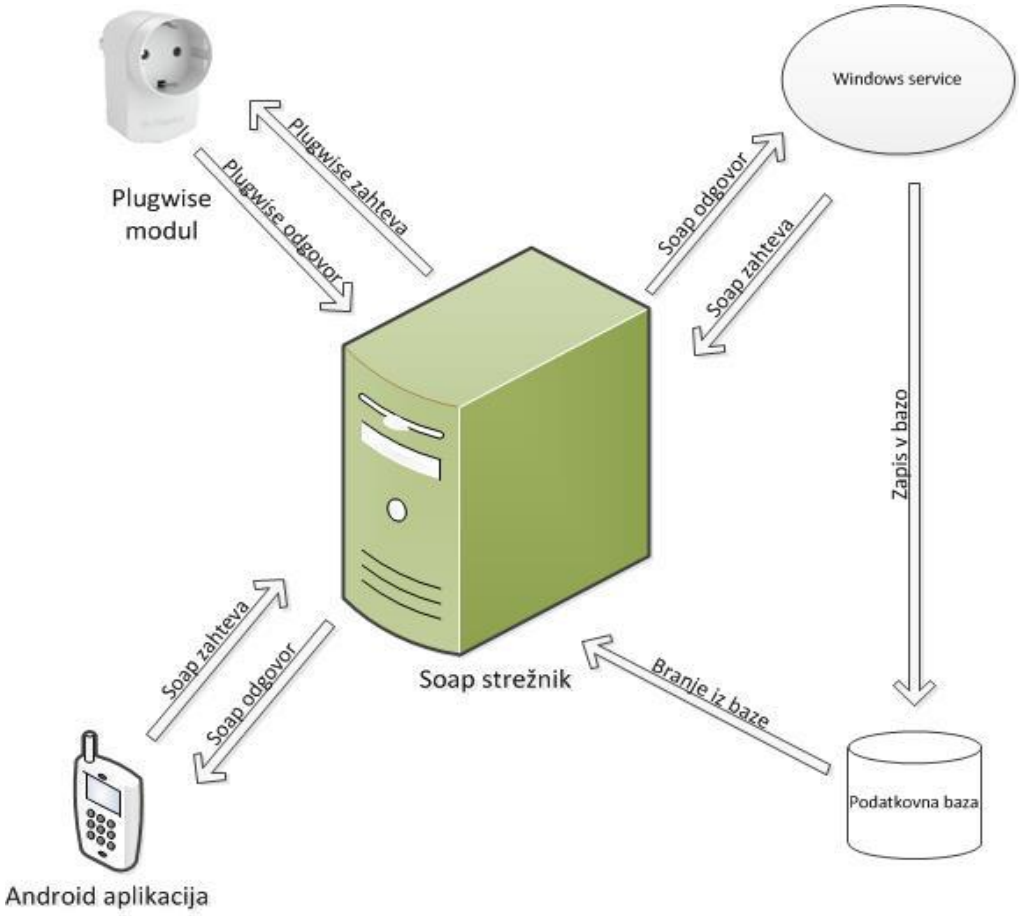

**Slika 14 - Microsoft Visual Studio**

# 5. Razvoj aplikacije

V tem poglavju je predstavljen razvoj rešitve za nadzor naprav in merjenje porabe električne energije, ki je sestavljena iz strežniškega dela in aplikacije za mobilne naprave z operacijskim sistemom Android. Najprej bomo predstavili zasnovo rešitve in pokazali shemo povezljivosti, nato pa podrobneje predstavili razvoj obeh delov, strežniškega in odjemalskega.

## 5.1. Zasnova in arhitektura rešitve

Zaradi omejitve s strojno opremo, zaradi katere nujno potrebujemo računalnik z usb vhodom v katerega priključimo glavni krmilnik Plugwise, je nujno, da aplikacija na nek način komunicira z računalnikom, ki nato ukaze preko glavnega krmilnika pošlje na mrežni krmilnik, le-ta pa naprej na posamezna vozlišča. Za način komunikacije smo si izbrali protokol soap, ki je zanesljiv, varen in relativno enostaven za implementacijo, tako v okolju .NET, kot tudi pri programiranju za Android. Ker moduli Plugwise hranijo zgodovino o porabi električne energije le za 4 ure, smo implementirali tudi svojo podatkovno bazo, v katero soap strežnik vpisuje rezultate meritev. Ker strežnik SOAP deluje po principu zahteva-odgovor in zaradi varčevanja z energijo nismo želeli pošiljati pogostih zahtev po trenutni porabi električne energije, smo za ta namen implementirali storitev Windows, ki vsakih nekaj sekund pošlje poizvedbo po trenutni porabi in le-to zapiše v že omenjeno podatkovno bazo. Arhitektura rešitve je predstavljena na sliki 15.




**Slika 15 - arhitektura rešitve**

Strežniški del skrbi za posredovanje ukazov modulom Plugwise, obdelovanje podatkov in zapisovanje le-teh v bazo. Mobilna aplikacija pa uporabniku omogoča, da prek uporabniškega vmesnika krmili električne porabnike priključene na module Plugwise in prikazuje podatke, ki jih prek strežniškega dela ti moduli vračajo. Aplikacija s strežnikom komunicira preko protokola Soap. V nadaljevanju je posebej opisan razvoj vsakega od obeh delov – strežnika in aplikacije.

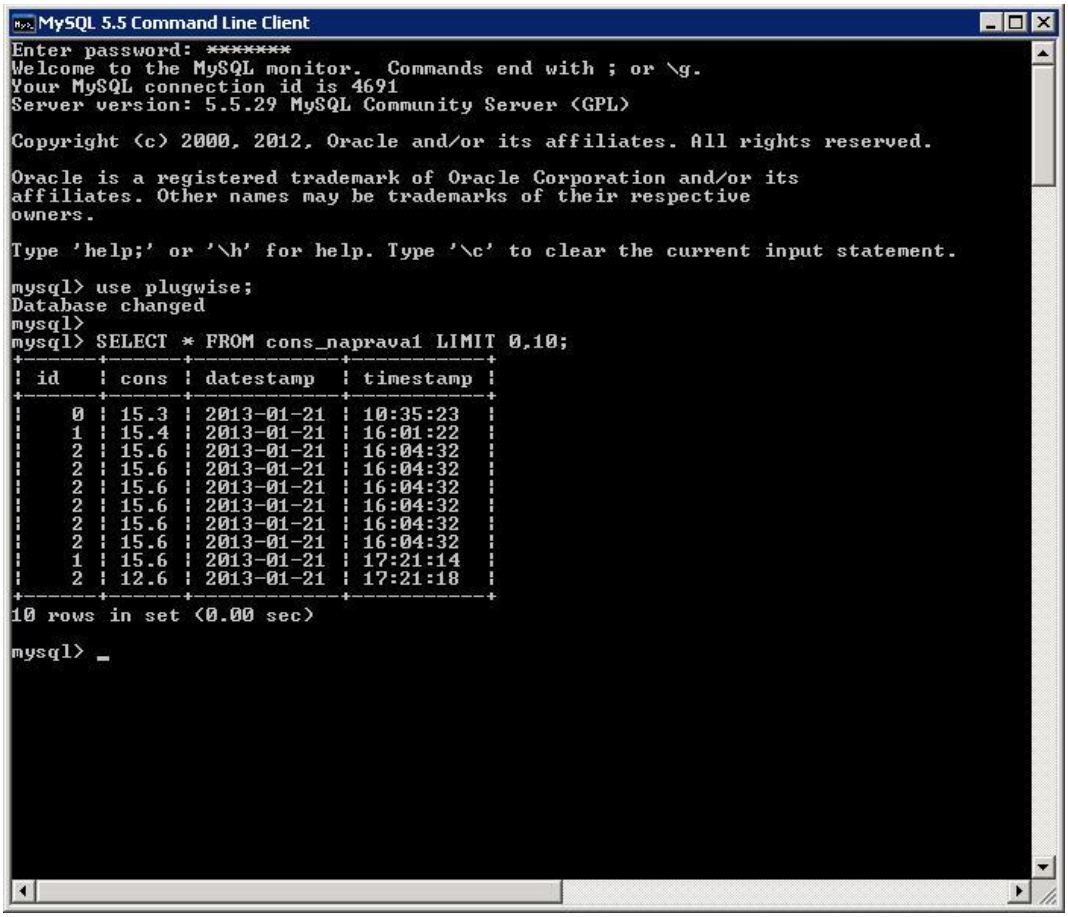
## 5.1 Strežniški del

Pri razvoju strežniškega dela smo si pomagali z odprtokodno knjižnico PlugwiseLib, ki je prosto dostopna na internetu, napisana pa je programskem jeziku C#, zato smo tudi mi strežniški del razvili v tem jeziku. Knjižnica omogoča komuniciranje z napravami Plugwise mimo uradne programske opreme podjetja. V pomoč nam je bila tudi konzolna aplikacija, ki jo je za svojo diplomsko nalogo leta 2010 razvil Blaž Oštrek.

Strežniški del je sestavljen iz strežnika SOAP, podatkovne baze MySql in storitve Windows.

### 5.1.1. Podatkovna baza

Za podatkovno bazo smo izbrali podatkovno bazo MySQL, ker je odprtokodna, dobro dokumentirana in lahko dostopna.




**Slika 16 - odjemalec za podatkovno bazo mysql**

Ker je bila podatkovna baza, tako kot ves strežniški del, postavljena na starejšem računalniku, smo izbrali odjemalca brez grafičnega vmesnika (slika 16).

Podatkovna baza ima dve tabeli. Po eno za vsako napravo, oziroma vozlišče. Iz slike 14, na kateri je tabela za napravo 1, je razvidno da ima tabela 4 polja. In Sicer: id vnosa (id), porabo naprave v vatih (cons), datum vnosa (datestamp) ter čas vnosa (timestamp).

### 5.1.2. Strežnik soap

Strežnik SOAP služi za komunikacijo med uporabniškim vmesnikom aplikacije in moduli Plugwise. Torej uporabnik preko uporabniškega vmesnika aplikacije pošilja ukaze na strežnik soap, ta pa jih nato pošlje naprej strojni opremi Plugwise. Strežnik ima tako implementirane funkcije za krmiljenje naprav Plugwise ter vnašanje podatkov v podatkovno bazo in branje iz nje.

Strežnik SOAP smo implementirali s pomočjo v okolje .NET vključenih spletnih storitev(web service). Spletne storitve razširjajo infrastrukturo svetovnega spleta. Zagotavljajo možnost, da različni programi komunicirajo med sabo. Ti programi lahko dostopajo do spletne storitve preko različnih protokolov in podatkovnih formatov, brez da bi morali skrbeti, kako je spletna storitev implementirana[10].

Sledi opis funkcij in delov kode strežnika SOAP:

```
[WebService(Namespace = "http://plugwiseapp.com/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
```

Ta del kode je avtomatsko generiran s strani okolja Microsoft Visual Studio. To so atributi spletne storitve. Deklarirani morajo biti pred razredom spletne storitve. Le-ta je deklariran tako:

```
public class DeviceValue : System.Web.Services.WebService
```

Naslednji izsek kode vsebuje konstruktor razreda, ki kliče funkcijo za inicializacijo podatkovne baze:

```
public DeviceValue() {
     Initialize();
 }
 private void Initialize() {
     server = "10.0.6.199";
     database = "plugwise";
     uid = "root";
     password = "ltfe123";
                 string connectionString = "SERVER=" + server + ";" +
   "DATABASE=" + database + ";" + "UID=" + uid + ";" + "PASSWORD=" +
    password + ";";
     connection = new MySqlConnection(connectionString);
 }
```

Spremenljivke server, database, uid, password in connection so predhodno deklarirane v kodi.

#### 5.1.2.1. Metoda za vklop in izklop naprav

Sledi metoda za vklop in izklop naprav.

```
[WebMethod]
public int setDeviceValue(int deviceID, int parameterValue)
{
	plugwiseControl control = new plugwiseControl("COM3");

	string mac = "";
	if (deviceID == 1)
	{
		mac = "000D6F00007294E7";
	}
	else if(deviceID==2)
	{
		mac = "000D6F000076A66A";
	}

	if (parameterValue == 1)
	{
		control.Open();
		control.Action(mac, PlugwiseLib.BLL.BC.PlugwiseActions.On);
		control.Close();
	}
	else if (parameterValue == 0)
	{
		control.Open();
		control.Action(mac, PlugwiseLib.BLL.BC.PlugwiseActions.Off);
		control.Close();
	}
	control.Close();
	return parameterValue;

}
```

Ta metoda sprejme dva argumenta, in sicer ID naprave (deviceID) ter vrednost parametra (parameterValue). ID naprave nam pove katero napravo bomo upravljali. Glede na ID naprave nastavimo naslov mac vozlišča v katerega je naprava, ki jo želimo upravljati, vključena. Vrednost parametra pa nam pove ali bomo napravo vključili ali izključili. Glede na vrednost parametra pokličemo funkcijo: `control.Action()`, in sicer, če je vrednost parametra parameterValue enaka 0, s parametrom `PlugwiseLib.BLL.BC.PlugwiseActions.Off`, če pa je parameterValue nastavljen na 1, pa funkcijo `control.Action()` kličemo s parametrom `PlugwiseLib.BLL.BC.PlugwiseActions.On`. V obeh primerih pa dotični funkciji kot parameter dodamo tudi naslov mac, ki smo ga dodelili glede na ID naprave. Pred klicem funkcije `control.Action()` s funkcijo `control.Open()` odpremo dostop do serijskih vrat na katera je priključen glavni krmilnik. Podobno nato dostop do serijskih vrat s klicem funkcije `control.Close()` zapremo.

Na primer, s takšnim klicem funkcije:

```
DeviceValue(1, 0);
```

bi izključili napravo 1, ki je vključena v vozlišče z naslovom mac 000D6F00007294E7. Funkcija `DeviceValue()` na koncu vrne vrednost parametra parameterValue.

Uporabljene funkcije `control.Open(),control.Action()` in `control.Close()` so del razreda plugwiseControls, ki je del zgoraj omenjene odprtokodne knjižnice.

#### 5.1.2.2. Metoda za trenutno porabo in status

Na podoben način delujeta tudi metodi `public string consumption(int deviceID),` ki vrne trenutno porabo izbrane naprave in `public int status(int deviceID)`, ki vrne status izbrane naprave, torej ali je naprava vključena ali izključena. Obe metodi sprejmeta le argument deviceID in na njegovi podlagi izbereta naslov mac vozlišča, na katerega pošljeta ukaz bodisi za poizvedbo po trenutni porabi ali po statusu naprave.

Metoda `public string consumption(int deviceID)` pa kliče še eno pomembno metodo in sicer metodo, ki iz odgovora o moči, ki ga glavni krmilnik plugwise pošlje nazaj računalniku (glej poglavje 2.4.3.3), izračuna moč v vatih.

Koda:

```
        private string izracunPorabe(string mac)
        {
            control.Open();
            control.Action(mac, PlugwiseLib.BLL.BC.PlugwiseActions.powerinfo);
            //delay 2sec
            System.Threading.Thread.Sleep(2000);
                  string text1 =
      System.IO.File.ReadAllText(@"C:\\inetpub\\wwwroot\\webservice\\port_podatki.txt"
      );
            float RealWatt=-1;
            string sRealWatt="-1";

            string[] lines = Regex.Split(text1, "\r\n");
            foreach (string podatki in lines)
            {
                if (podatki.Length > 59)
                {
                    string watt = podatki.Substring(28, 4);
                    int watti = int.Parse(watt,
System.Globalization.NumberStyles.HexNumber);
                    float wattf = (float)watti;
                    string dWatt = podatki.Substring(32, 4);
                    int dWatti = int.Parse(dWatt,
System.Globalization.NumberStyles.HexNumber);
                    float dWattf = (float)dWatti;
                    RealWatt = (wattf * 3) - (dWattf / 10);
                    sRealWatt = RealWatt.ToString("0.00");
                    control.Close();
                    return sRealWatt;
                }
            }
            return sRealWatt;
        }
```

Ta funkcija prebere porabo moči, ki jo je poslalo vozlišče, jo razdeli po vrsticah in nato upošteva le vrstico daljšo od 59 znakov (glej poglavje 2.4.3.3), ter iz nje izlušči uporabne dele v šestnajstiškem zapisu, ter iz njih izračuna moč v vatih.

#### 5.1.2.3. Metoda drawChart

Pri risanju grafa porabe električne energije smo si pomagali s knjižnico GoogleChartSharp, ki omogoča oblikovanje grafa z Googlovim orodjem Google charts v okolju .NET.

Metoda na podlagi datuma in dneva v tednu s pomočjo metode Select, ki je opisana v naslednjem podpoglavju, iz podatkovne baze pridobi podatke o porabi električne energije za vsak dan v tednu za izbrano napravo. Iz pridobljenih podatkov izračuna povprečno porabo za vsak dan in jo shrani v tabelo, ki predstavlja množico podatkov za prikaz na grafu. Primer grafa je na sliki 17.

Kreiranje grafa:

```
            float[] data1 = { (float)15.4, (float)14.5, (float)0.0, (float)20.1,
(float)17.3, (float)0.0, (float)15.7 };
            List<float[]> datasets = new List<float[]>();
            datasets.Add(data1);

            ChartAxis bottomAxis = new ChartAxis(ChartAxisType.Bottom, new string[] {
"pon", "tor", "sre", "cet", "pet", "sob", "ned" });
            ChartAxis leftAxis = new ChartAxis(ChartAxisType.Left);


            BarChart barChart = new BarChart(300, 150, BarChartOrientation.Vertical,
BarChartStyle.Grouped);

            barChart.SetTitle("Poraba tekoci teden");

            barChart.AddAxis(bottomAxis);
            barChart.AddAxis(leftAxis);
            barChart.SetData(datasets);

            barChart.SetDatasetColors(new string[] { "FF0000" });
```

Orodje Google charts smo uporabili, ker je preprosto za uporabo in takoj kreira naslov URL grafa, ki ga lahko nato uporabimo v naši aplikaciji.


**Slika 17 - graf kreiran z Google charts**

#### 5.1.2.4. Metoda select

Sledi še opis zadnje pomembnejše metode strežnika SOAP. To je metoda `public List<string>[] Select(string date)`, ki iz podatkovne baze mysql prebere porabo izbrane naprave za datum, ki je podan kot argument funkcije.

Koda:

```
public List<string>[] Select(string date)
{
    string query = "SELECT * FROM cons_naprava1 WHERE datestamp='"+date+"'";

    //Create a list to store the result
    List<string>[] list = new List<string>[4];
    list[0] = new List<string>();
    list[1] = new List<string>();
    list[2] = new List<string>();
    list[3] = new List<string>();

    //Open connection
    if (this.OpenConnection() == true)
    {
        //Create Command
        MySqlCommand cmd = new MySqlCommand(query, connection);
        //Create a data reader and Execute the command
        MySqlDataReader dataReader = cmd.ExecuteReader();

        //Read the data and store them in the list
        while (dataReader.Read())
        {
            list[0].Add(dataReader["id"] + "");
            list[1].Add(dataReader["cons"] + "");
            list[2].Add(dataReader["datestamp"] + "");
            list[3].Add(dataReader["timestamp"] + "");
        }

        //close Data Reader
        dataReader.Close();

        //close Connection
        this.CloseConnection();

        //return list to be displayed
        return list;
    }
    else
    {
        return list;
    }
}
```

Pri tej funkciji nam je bila v pomoč knjižnica Connect/NET, ki je prosto dostopna na spletu. Najprej si v spremenljivko query shranimo poizvedbo mySql za preprostejšo nadaljnjo uporabo. Nato si pripravimo polje seznamov, v katerega bomo kasneje shranili rezultate poizvedbe. Nato odpremo povezavo do podatkovne baze ter če je uspela, izvršimo poizvedbo. Za branje rezultatov ustvarimo instanco objekta MySqlDataReader in rezultate zapišemo v

polje seznamov, ki smo si ga prej pripravili. Na koncu zapremo povezavo do podatkovne baze in vrnemo polje seznamov z rezultati. Če nam povezava s podatkovno bazo ni uspela, samo vrnemo polje seznamov, ki je v tem primeru prazno.

### 5.1.3 Storitev Windows

Storitev Windows skrbi, da se trenutna poraba ob določenih časovnih intervalih zapisuje v podatkovno bazo. Časovni interval je za enkrat določen v kodi in je trenutno 30 sekund. Tudi v tem delu rešitve smo uporabili knjižnico Connect/NET, saj je glavni naloga storitve Windows zapis trenutne porabe v datoteko. Razlog, da smo za ta del izvedli z storitvijo Windows in ne kar neposredno v spletni storitvi (strežniku soap), je da smo potrebovali storitev, ki teče ves čas, ko je računalnik oziroma strežnik prižgan in ob določenih časovnih intervalih kliče spletno metodo `Consumption`(), ki vrača trenutno porabo (glej sliko 13).

Storitev Windows ima torej za delo z bazo podobne metode kot spletna storitev, s to razliko da podatke v podatkovno bazo zapisuje, spletna storitev pa jih iz nje bere. Storitev Windows ima dve metodi, ki sta ključni za njeno delovanje in jih po pravilu mora imeti. To sta `OnStart`() in `OnStop`(). Ko že imeni metod povesta, se prva izvede ob zagonu storitve, druga pa ob njenem koncu.

`OnStart`():

```
protected override void OnStart(string[] args)
{
    eventLog1.WriteEntry("In OnStart plugwise windows service");

    timer = new Timer();
    timer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
    timer.Interval = 300000;
    timer.Enabled = true;
}
```

Ko zaženemo storitev, se to najprej zapiše v dnevnik dogodkov operacijskega sistema Windows. Sledi nastavitev merilnika časa. To storimo s pomočjo razreda Timer. S stavkom `timer.Elapsed += new ElapsedEventHandler(OnTimedEvent);` določimo kaj se bo zgodilo po preteku določenega časa. Mi pokličemo metodo OnTimedEvent, ki jo bomo opisali v nadaljevanju. Za tem določimo časovni interval ter na koncu še poženemo merilnik časa.

OnStop():

```
protected override void OnStop()
{
    eventLog1.WriteEntry("In onStop.");
    timer.Stop();
}
```

V tej metodi najprej v dnevnik dogodkov zapišemo da se storitev končuje. Nato pa še ugasnemo merilnik časa.

Sledi predstavitev metode OnTimedEvent:

```
private void OnTimedEvent(object source, ElapsedEventArgs e)
{
    string cons = webService.consumption(2);
    double dCons = Convert.ToDouble(cons);

    try
    {
      Insert(counter, dCons, DateTime.Now.ToString("yyyy-MM-dd"),
      DateTime.Now.ToString("HH:mm:ss"));
         sw.WriteLine(DateTime.Now + " Poraba: " + cons + "\n");

    }
    catch (Exception ex)
    {
        eventLog1.WriteEntry("Exception caught (writing): "+ex);
    }
    counter++;
}
```

To je kot rečeno metoda, ki se izvrši vedno ko se časovnik izteče. Vse kar ta metoda naredi je, da pošlje zahtevo po trenutni moči naprave (glej Consumption()) in odgovor skuša zapisati v podatkovno bazo. V primeru, da ji to ne uspe, v dnevnik dogodkov operacijskega sistema Windows zapiše do kakšne izjeme je prišlo.

## 5.3. Odjemalski del

Odjemalski del predstavlja aplikacija Android. Ker komunikacija med strežniškim in odjemalskim delom poteka preko protokola SOAP, nam je bila pri izdelavi aplikacije v pomoč odprtokodna knjižnica ksoap2, odjemalska knjižnica SOAP za platformo android.

### 5.3.1. Zasnova

Aplikacija je zasnovana tako, da ja kar najlažje uporabna za povprečnega končnega uporabnika. Vsako napravo upravljamo v svojem zavihku, zadnji zavihek pa je rezerviran za splošne nastavitve. Na zavihku naprave je vidno ali je naprava vključena ali izključena, koliko je njena poraba v vatih in kar je povprečnemu uporabniku morda še bolj pomembno, kakšna je poraba v centih, oziroma evrih. Ceno v evrih smo dobili tako, da smo moč najprej pretvorili v kilovatno uro po formuli: moč v vatih pomnožena z eno uro, deljeno z tisoč. Nato pa smo dobljeno kilovatno uro pomnožili z 0.08879, kolikor je bila v času nastajanja diplomske naloge povprečna cena za kilovatno uro na slovenskem trgu. Večji del zavihka obsega graf porabe energije v tekočem tednu, na dnu zavihka pa so še različne možnosti vklopa in izklopa naprave. Te možnosti so: preko gumba, glede na lokacijo uporabnika, nočni način (uporabnik napravo potrese za vklop ali izklop) in časovnik, kjer lahko uporabnik nastavi po kakšnem času se bo naprava vklopila in izklopila.

### 5.3.2. Komunikacija s strežnikom

Glavni deli aplikacije so trije razredi, ki skrbijo za posredovanje ukazov do strežnika soap. Ti razredi so `SetDeviceValueClass`, ki skrbi da se naprava vključi ali izključi, `ConsumptionClass`, ki pridobiva podatke o porabi naprave, `StatusClass`, ki pridobiva podatke o tem ali je naprava vključena ali izključena in `DrawChartClass`, ki pridobi naslov url grafa porabe električne energije. Ti trije razredi kličejo ustrezne metode spletne storitve (strežnika SOAP) (glej poglavje 5.1.2.). V tabeli 2 je prikazano kateri razredi kličejo katere metode ter kaj metode sprejmejo kot parameter in kaj vračajo.

| Razred aplikacije | Parametri | Metoda spletne storitve | Odgovor spletne storitve |
|---|---|---|---|
| SetDeviceValueClass | id naprave, vrednost | setDeviceValue | vrednost |
| ConsumptionClass | id naprave | consumption | trenutna poraba |
| StatusClass | id naprave | status | status naprave |
| DrawChartClass | id naprave | drawChart | naslov URL grafa |

**Tabela 2 - povezava med razredi odjemalca in metodami strežnika**

Ker so si vsi trije omenjeni razredi podobni, je v nadaljevanju razloženo njihovo delovanje na primeru razreda `SetDeviceValueClass`.

Koda:

```
class SetDeviceValueClass extends AsyncTask<Integer, Void, String> {

	private static final String SOAP_ACTION2 =
			"http://plugwiseapp.com/setDeviceValue";
	private static final String METHOD_NAME2 = "setDeviceValue";

	private static final String NAMESPACE = "http://plugwiseapp.com/";
	private static final String URL =
			"http://10.0.6.199/webservice/DeviceValue.asmx";

	@Override
	protected String doInBackground(Integer... params) {
		SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME2);

		int switchValue=params[0];
		int deviceId=params[1];
		if(deviceId==2){
			request.addProperty("deviceID", "2");
		} else if(deviceId==1){
			request.addProperty("deviceID", "1");
		}

		if(switchValue==1){
			request.addProperty("parameterValue", "1");
		}
		else if(switchValue==0){
			request.addProperty("parameterValue", "0");
		}
```

```
                SoapSerializationEnvelope envelope =
                            new SoapSerializationEnvelope(SoapEnvelope.VER11);
                envelope.dotNet = true;
                envelope.setOutputSoapObject(request);

                HttpTransportSE ht = new HttpTransportSE(URL);
                ht.setXmlVersionTag
                ("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
                try {
                      ht.call(SOAP_ACTION2, envelope);
                      SoapPrimitive response = (
                                  SoapPrimitive)envelope.getResponse();

                      if(response.toString().equals("1"))
                            return "vkljucena";
                      else if(response.toString().equals("0"))
                            return "izkljucena";
                      else
                            return "error";

                } catch (Exception e) {
                      e.printStackTrace();
                }
                return "error";
          }
          @Override
          protected void onPostExecute(String result) {

                if(result.equals("vkljucena")){
                      statusIntern=1;
                } else if (result.equals("izkljucena")) {
                      statusIntern=0;
                }
          }
    }
```

V prvi vrstici vidimo, da razred razširja abstraktni razred AsyncTask. AsyncTask omogoča pravilno in lahko uporabo niti uporabniškega vmesnika. Omogoča izvajanje operacij v ozadju in objavlja rezultate v niti uporabniškega vmesnika, brez da bi morali mi kot programerji ročno upravljati z nitmi[11]. Namenjen je kratkim, nekaj sekundnim operacijam in za takšne smo ga uporabili tudi mi pri svoji rešitvi.

Vsi zgoraj omenjeni razredi za komunikacijo s spletno storitvijo prepišejo vsaj eno od metod: `doInBackground` in `onPostExecute`. Kot že imeni povesta, se prva metoda izvede v ozadju, da lahko nit uporabniškega vmesnika nemoteno teče dalje, druga pa se izvede po izvajanju in vpliva na nit uporabniškega vmesnika. V našem primeru se v metodi doInBackground kreira zahteva SOAP, ki je objekt tipa SoapObject. SoapObject je del zgoraj omenjene knjižnice ksoap2. Zahteva vsebuje imenski prostor, ime metode, ki jo želimo klicati ter parametre ki smo jih podali (v konkretnem primeru id naprave in vrednost

vključeno/izključeno). Po tem se prav tako s pomočjo razreda iz knjižnice ksoap2, kreira ovojnica soap, ki vsebuje prej kreirano zahtevo. Nato pa se kreira še instanca razreda HttpTransportSE (prav tako član omenjene knjižnice), ki poskrbi za pošiljanje ovojnice SOAP in sprejem odgovora. Glede na odgovor pa metoda `doInBackground` vrne znakovni niz »vključena«, »izključena« ali »napaka«. Gre seveda za stanji izbrane naprave in obvestilo v primeru napake.

Metoda `onPostExecute` pa je krajša in bolj preprosta. Njena naloga je samo, da glede na rezultat metode `doInBackground` nastavi interno spremenljivko za spremljanje statusa naprave na nič ali ena.

Razreda `StatusClass` in `ConsumptionClass` pa sta zelo podobna, le da še malo preprostejša saj metodi spletne storitve `status` in `consumption` (glej tabelo) sprejmeta le en argument in ne dveh, kot metoda `setDeviceValue`. Razen tega se razreda od prvega razlikujeta še po metodi `doInBackground`. In sicer pri teh dveh razredih ta metoda, glede na odgovor spletne storitve osvežuje elementa uporabniškega vmesnika, ki obveščata o statusu naprave ter o njeni porabi.

Razred `DrawChartClass` pa metodi spletne storitve `drawChart` poda argumenta datum in dan v tednu, metoda pa mu vrne naslov url do grafa porabe električne energije v tekočem tednu.

Instanca razreda `StatusClass` se v programu ustvari le enkrat, in sicer samo ob zagonu aplikacije, v nadaljevanju pa se zaradi hitrejšega delovanja za preverjanje statusa uporablja interna spremenljivka, ki jo ob vsaki spremembi statusa naprave popravimo. Instanci preostalih razredov pa se ustvarjata večkrat. Razred `setDeviceValueClass` se uporabi vsakič, ko želimo napravo vključiti ali izključiti, razred `ConsumptionClass` pa vedno ko preverimo porabo naprave, to pa počnemo s pomočjo javanskega razreda Timer vsakih 5 sekund.

### 5.3.3. Možnosti vklopa in izklopa

Kot je omenjeno že v poglavju 5.3.1., ima uporabnik na voljo več možnosti vklopa in izklopa naprave. V nadaljevanju je opisana programska izvedba teh možnosti.

#### 5.3.3.1. Gumb za vklop in izklop

Najosnovnejša je možnost vklopa in izklopa izbrane naprave preko gumba za vklop in izklop. Zaradi boljše uporabniške izkušnje je ta gumb večji od ostalih gumbov. Gumb za vklop in izklop je preklopni, kar pomeni da ga z enim pritiskom aktiviramo (vključimo napravo), z drugim pritiskom pa deaktiviramo (izključimo napravo). Ta gumb služi kot indikator, ali je naprava vključena ali izključena, saj kadar je aktiviran, na njem gori zelena lučka, spreminja pa se tudi napis na samem gumbu, ki je ob vključeni napravi »Naprava vključena«, ob izključeni pa »Naprava izključena«.

#### 5.3.3.2. Vklop in izklop glede na lokacijo

Druga možnost vklopa naprave je glede na lokacijo na kateri so uporabnik nahaja. V ta namen smo implementirali skočni meni, ki se pojavi na zaslonu ko uporabnik pritisne gum za upravljanje glede na lokacijo. Za ta namen smo uporabili razred AlertDialog.Builder, ki ustvari novo preprosto pogovorno okno, v katerem uporabnik vnese kdaj naj se naprava vključi in kdaj naj se izključi (več o tem v poglavju uporaba). Princip delovanje je, da se naprava vključi ko je uporabnik oddaljen od doma za manj kot število metrov, ki ga je vnesel v prej omenjenem pogovornem oknu in izključi, ko je oddaljen za več kot število metrov, ki ga je prav tako vnesel v pogovornem oknu. Za zajem lokacije smo uporabili razred LocationManager, ki je del razvojnega paketa Android. Ob vsaki spremembi lokacije kličemo metodo `onLocationChanged`, ki za argument sprejme zadnjo znano lokacijo. V tej metodi nato izračunamo oddaljenost lokacije od našega doma. Koordinate doma so v tej verziji aplikacije vnesene direktno v programsko kodo in jih uporabnik še ne more spreminjati. Razdaljo izračunamo z metodo `distanceTo`, ki vrne razdaljo med dvema paroma koordinat v metrih.

Primer klica metode distanceTo:

```
float razdalja=location.distanceTo(location2);
```

Tak klic nam v spremenljivko razdalja shrani razdaljo med lokacijo1 in lokacijo2. To razdaljo nato primerjamo z razdaljama, ki ju je uporabnik vnesel v pogovorno okno in glede na to ali smo znotraj ali zunaj vnesenega polmera, napravo vklopimo ali izklopimo s pomočjo razreda `SetDeviceValueClass`.

#### 5.3.3.3. Nočni način

Nočni način smo poimenovali možnost vklopa ali izklopa naprave s tem da mobilni telefon potresemo. Za implementacijo te možnosti smo uporabili razreda `SensorManager`, ki omogoča dostopanje do senzorjev mobilne naprave in `SensorEventListener`, ki sprejema obvestila razreda `SensorManager`, ko se vrednost senzorja spremeni. Ko SensorManagerju registriramo poslušalca (`SensorEventListener`), mu tudi povemo za kateri tip senzorja gre. V našem primeru je to pospeškomer. Vedno ko se vrednost pospeškomera spremeni, primerjamo ali je bil pospešek večji od vnaprej določenega pospeška, ki je meja za vklop ali izklop naprave. Če je bil večji napravo glede na njeno stanje vklopimo ali izklopimo. Na primer, če je naprava predhodno vključena, jo ob prekoračitvi mejnega pospeška izključimo in obratno.

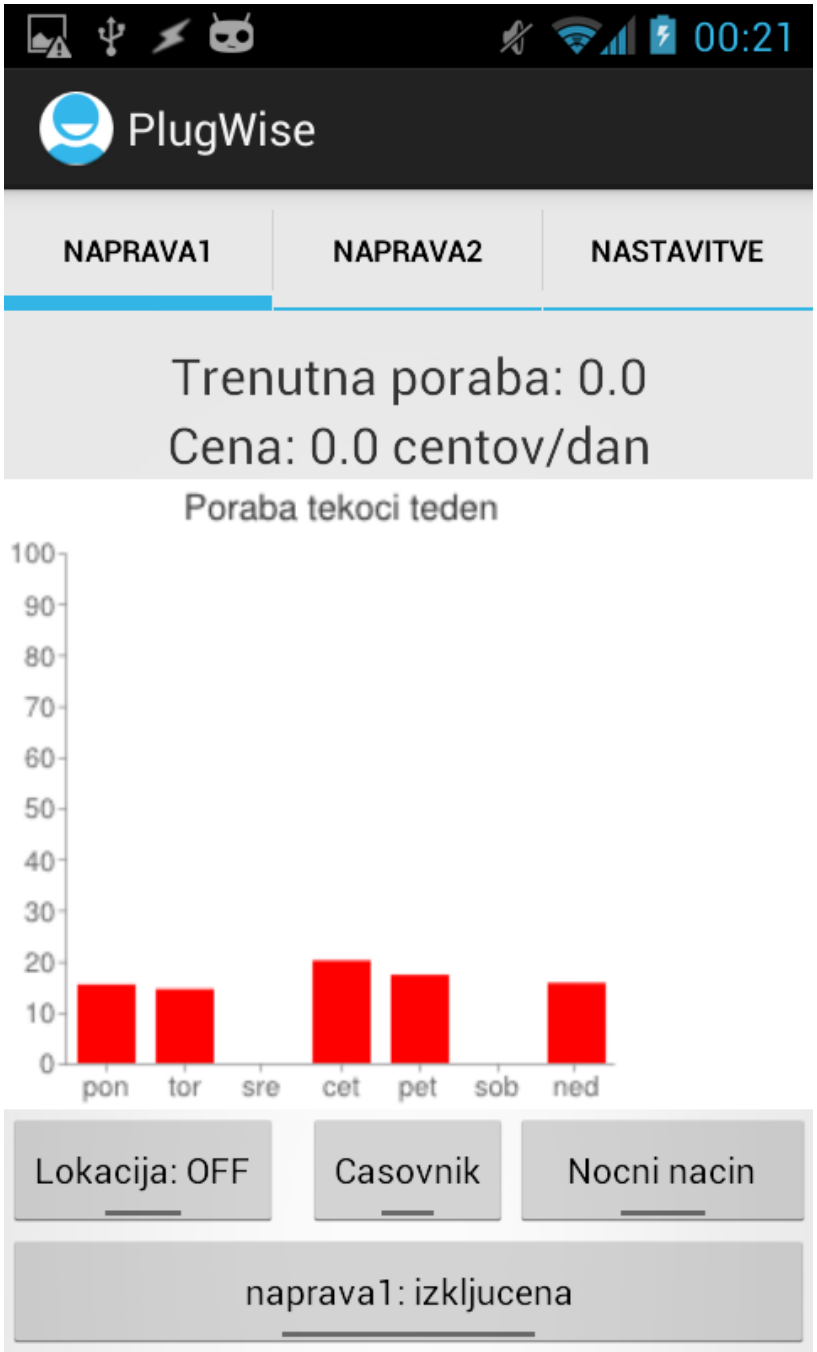
#### 5.3.3.4. Časovnik

Zadnja možnost vklopa in izklopa naprave je uporaba časovnika. Tudi za to možnost smo, podobno kot pri možnosti vklopa glede na lokacijo, implementirali skočni meni, v katerem uporabnik izbere čas vklopa in izklopa naprave. Pri implementaciji časovnika smo si pomagali z javanskim razredom Timer, s pomočjo katerega ob izteku časa do vklopa ali

izklopa naprave kličemo Razred `SetDeviceValueClass` in preko njega izbrano napravo vključimo ali izključimo.

## 6. Uporaba aplikacije

Uporaba aplikacije je preprosta in pregledna. V tem poglavju bomo predstavili vse možnosti njene uporabe. Na sliki 18 je uporabniški vmesnik aplikacije. V nadaljevanju bomo razložili vse njegove funkcije.



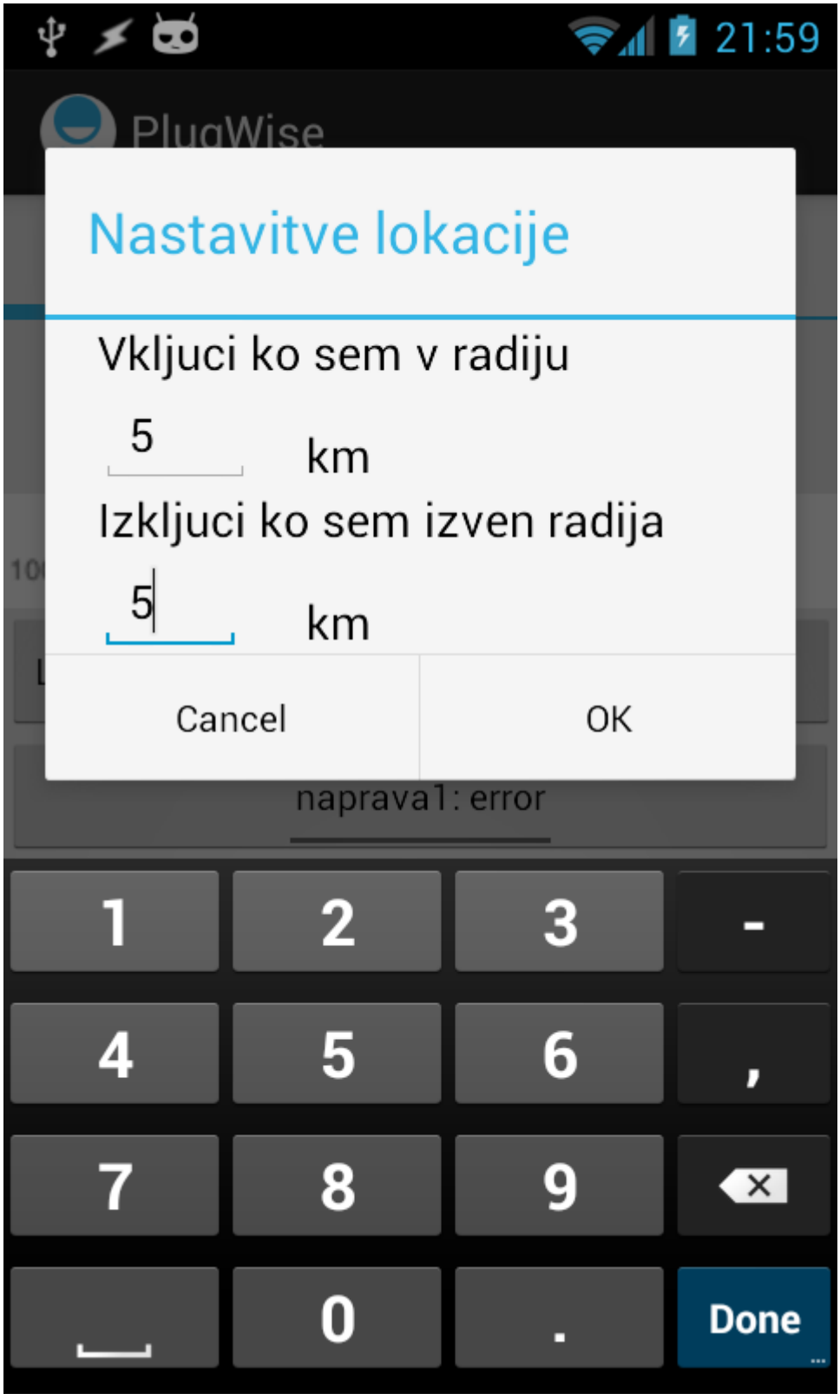

**Slika 18 - Uporabniški vmesnik aplikacije**

Na vrhu je vrstica z zavihki. Prva dva zavihka predstavljata vsak svojo napravo in sta na videz enaka. Tretji zavihek pa so nastavitve aplikacije. Pod zavihki sta polja, ki uporabnika

obveščata o trenutni porabi naprave. Najprej je prikazana naprava v vatih, nato pa še približno koliko uporabnika stane, če je ta naprava prižgana cel dan. Glavnino zaslona zaseda graf porabe aplikacije za tekoči teden. Iz tega grafa je razvidno koliko električne energije je naprava porabila v posameznih dnevih v tednu. Pod grafom pa se nahajajo različne možnosti vklopa in izklopa naprave. Te možnosti so štiri: s pritiskom guma za vklop in izklop, glede na lokacijo na kateri se uporabnik nahaja, s tresenjem telefona (nočni način) in s časovnikom.

Vklop ali izklop s pritiskom na gumb je trivialen in ne potrebuje posebne razlage. Uporabnik le pritisne gumb za vklop in izklop in če je v času pritiska naprava izključena, se le-ta vključi in obratno.

Naslednja možnost je možnost vklopa ali izklopa glede na lokacijo uporabnika. Ob pritisku na gumb »Lokacija: OFF«, se uporabniku, če nima vklopljenega pridobivanja lokacije preko satelitov GPS, pokaže obvestilo, da je to pridobivanje izključeno in možnost da ga vključi. Uporabnik lahko to stori s pritiskom na gumb OK ali pa zavrne s pritiskom na gumb Cancel, saj lahko za pridobivanje lokacije uporabi tudi omrežje ponudnika storitev. Ko to stori pa ga pričaka novo pogovorno okno (slika 19), v katerem izbere koliko kilometrov mora biti oddaljen da se naprava vključi ali izključi. Če uporabnik ne želi prikazovanja obvestila o satelitih GPS, lahko to izključi v zavihku nastavitve ali pa direktno v pogovornem oknu.




**Slika 19 - nastavitve lokacije**

Nastavljanje časovnika je podobno nastavljanju lokacije. Ob pritisku na gumb »Časovnik« se odpre pogovorno okno, kjer uporabnik izbere kdaj se bo naprava vključila in kdaj izključila.

Zadnja možnost naprave pa je nočni način, ki ga uporabnik vključi s pritiskom na gumb »Nočni način«. Ko je ta način vključen, je dovolj, da uporabnik za vklop oziroma izklop naprave mobilni telefon le potrese in naprava se bo vključila ali izključila.

# 7. Možnosti izboljšav

Čeprav rešitev zadošča potrebam povprečnega uporabnika, je možnosti prostora za izboljšave še veliko. V tem poglavju je predstavljenih nekaj teh možnosti.

## 7.1. Izboljšave uporabniškega vmesnika

Pri izboljšavah uporabniškega vmesnika najprej pade na misel izboljšava izgleda, da bi aplikacijo še bolj približali uporabniku. Trenutno aplikacija še nima pozdravnega zaslona. Prav tako nismo sami izdelali grafik za gumbe, ampak smo uporabili privzete sloge grafičnega vmesnika za platformo Android.

Naslednja izboljšava oziroma razširitev uporabniškega vmesnika bi bila možnost za dodajanje več naprav. Pri velikem številu naprav za izbiro naprave najbrž ne bi več uporabili zavihkov za izbiranje naprave, saj bi to hitro postalo nepregledno. Namesto tega bi uporabili spustni meni, v katerem bi uporabnik izbral s katero napravo želi upravljati. Uporabniku bi lahko ponudili tudi možnost, da svoje naprave shrani po skupinah na isti način, kot jih ima doma priključene po prostorih. Na primer: vozlišče v katerega je priključen televizijski sprejemnik in vozlišče z glasbenim stolpom bi shranil v skupino »dnevna soba«, vozlišče s hladilnikom in vozlišče z mikrovalovno pečico pa v skupino kuhinja.

Uporabniku bi lahko tudi ponudili več možnosti grafičnih prikazov. Recimo, poleg tedenskega prikaza, tudi prikaz po dnevih in mesecih. Lahko bi tudi prikazali več naprav na istem grafu za lažjo primerjavo.

## 7.2. Možnost povezovanja z distributerjem električne energije

Da bi rešitev postala skladna z idejo o pametnih omrežjih, bi jo bilo potrebno razširiti na način, da se povezovala z distributerji električne energije. To bi lahko rešili na enostaven način. Bodisi bi aplikacija sama ob določenih časovnih intervalih porabo, namesto da jo le zapiše v podatkovno bazo uporabnika, pošiljala tudi na strežnik distributerja ali pa bi za ta namen v aplikacijo dodali gumb »Pošlji distributerju« in bi uporabnik sam poslal podatke, ko bi to želel. Na ta način bi distributerji izvedeli kje so viški električne energije in bi takrat, ko določen uporabnik tem energije ne potrebuje, le-to usmeril drugam.

## 8. Zaključek

V okviru diplomske naloge je bila razvita rešitev za spremljanje porabe električne energije v gospodinjstvu, ki pa hkrati uporabniku tudi olajša rokovanje z električnimi napravami. Uporabnik ima tako več možnosti vklopa, kot le s pritiskom na gumb na napravi sami.

Spremljanje porabe energije v gospodinjstvu je pomembno, saj lahko le tako prispevamo k manjši porabi le-te. Ta aplikacije je zato korak v pravo smer, k električnemu omrežju nove generacije, kjer gospodinjstva niso le pasivne enote, ampak inteligentna vozlišča, ki distributerjem električne energije nudijo vpogled v porabo električne energije.

Da bi ta zgodba resnično zaživela, bi bilo potrebno sodelovanje tako države, kot distributerjev samih, ki bi uporabnika spodbujali k uporabi takšnih rešitev in jim celo brezplačno posojali strojno in programsko opremo, kot je to praksa pri ponudnikih internetnih storitev in ip televizije. Uporabnike, ki bi želeli sodelovati, bi lahko za njihovo sodelovanje nagradili z nižjo ceno električne energije. Na ta način bi vsi, uporabniki in distributerji pripomogli k postavitvi električnega omrežja nove generacije in posledično zmanjšanje rabe primarne energije, kar je ne nazadnje tudi cilj Evropske unije.

## 9. Viri in literatura

[1] (2008) Ozadje sprejemanja podnebno-energetskega svežnja. Dostopno na:

http://www.evropa.gov.si/si/energetika/podnebno-energetski-svezenj/

[2] A. Kljun, Pametna omrežja, predstavitev, prosojnici 3 in 4

[3] (2012) Energy Efficiency Status Report 2012. Dostopno na:

http://iet.jrc.ec.europa.eu/energyefficiency/sites/energyefficiency/files/energy-efficiency-status-report-2012.pdf

[4] (2011) Field Testing Smart Houses for Smart Grid. Dostopno na:

http://www.smarthouse-smartgrid.eu/fileadmin/templateSHSG/docs/publications/CIRED_FieldTesting.pdf

[5] Mesh networking. Dostopno na:

https://en.wikipedia.org/wiki/Mesh_networking

[6] Google Buys Android for Its Mobile Arsenal. Dostopno na:

http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal

[7] Industry Leaders Announce Open Platform for Mobile Devices. Dostopno na:

http://www.openhandsetalliance.com/press_110507.html

[8] Google is working on a mobile OS, and it's due out shortly. Dostopno na:

http://www.engadget.com/2007/08/28/google-is-working-on-a-mobile-os-and-its-due-out-shortly/

[9] Android Architecture – The Key Concepts of Android OS. Dostopno na:

http://www.android-app-market.com/android-architecture.html

[10] Web Services. Dostopno na:

http://msdn.microsoft.com/en-us/library/ms950421.aspx

[11] AsyncTask. Dostopno na:

http://developer.android.com/reference/android/os/AsyncTask.html