Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
FERI

Nejc Vidrih

ODPRTOKODNE REŠITVE ZA UPRAVLJANJE PAMETNIH HIŠ

Diplomsko delo

Maribor, september 2016

ODPRTOKODNE REŠITVE ZA UPRAVLJANJE PAMETNIH HIŠ

Diplomsko delo

Študent(ka):	Nejc Vidrih
Študijski program:	Univerzitetni študijski program
	Informatika in tehnologije komuniciranja
Smer:	Informacijski sistemi
Mentor:	doc. dr. Domen Verber

Sklep o diplomskem delu

Zahvala

Zahvaljujem se mentorju doc. dr. Domnu Verberju za vodenje in pomoč pri opravljanju diplomskega dela.

Prav tako se zahvaljujem družini, ki me je v času izobraževanja podpirala in vzpodbujala.

ODPRTOKODNE REŠITVE ZA UPRAVLJANJE PAMETNIH HIŠ

Ključne besede: IoT, odprta koda,pametna hiša ,iOS

UDK:

Povzetek

V diplomski nalogi predstavimo uporabo naprav interneta stvari za primer upravljanja pametne hiše. Razložen je princip odprte kode in povezan z izvorno kodo naprav interneta stvari. Pogledamo si obstoječe sisteme za upravljanje pametnih hiš in komunikacijske protokole s katerimi obstoječe naprave komunicirajo. Z uporabo računalnika Raspberry Pi razvijemo sistem pametne hiše upravljane z mobilno iOS aplikacijo. Razvijemo tudi razširitev za hiter dostop do izbranih funkcionalnosti in avtomatsko upravljanje z uporabo iBeacon tehnologije. Opisana so orodja uporabljena pri razvoju, podrobneje je razložen razvoj mobilne aplikacije za iOS v programskem jeziku Swift. Opisan je postopek priključitve in konfiguracije razvitega sistema. Povzamemo poglavitne težave v IoT in predlagamo idejo sistema pametne hiše, ki je nastala na podlagi raziskav in testiranj za potrebe diplomskega dela.

OPEN SOURCE SOLUTIONS FOR SMART HOME MANAGEMENT

Key words: IoT, open source, smart home ,iOS

UDK:

Abstract
Tu se zacne slo povezetek

KAZALO

KAZALO SLIK

KAZALO TABEL

Uporabljene kratice

IoT – (ang. Internet of Things)

GPIO – (ang. General Purpose Input Output)

API – (ang. Application Programming Interface)

NAT – (ang. Network Address Translation)

PWM – (ang. Pulse-width modulation)

BPM – (ang. Business process management)

GPS - Globalni sistem pozicioniranja

CVS - (ang. Concurrent Versions System)

1 UVOD

Internet, kot ga pozna večina, je »internet ljudi«. Vsebino kot so novice, videi ter slike ustvarjajo ljudje za druge ljudi. Internet je uporabljen zgolj kot medij za prenos informacij med ljudmi. V zadnjem času pa je veliko govora o internetu stvari (ang. IoT Internet of Things). Kaj je torej IoT in kako je povezan s pametno hišo?

Pametna hiša, je hiša v kateri bi naj večino naprav upravljal enoten inteligenten sistem. Bistvo enotnega inteligentnega sistema je povezovanje, upravljanje in nadzor nad porabniki. Cilj je zviševanja stopnje udobja, varčnosti in varnosti bivanja. Primer uporabe enotnega inteligentnega sistema bi bil naslednji: ob odhodu stanovalcev od doma se avtomatsko izključijo vsa svetila in druge električne naprave v hiši, vključi se alarm, zniža se nivo ogrevanja ter vključi simulacija prisotnosti [1].

Pri IoT gre za digitalizacijo fizičnega sveta, torej naprav in stvari, ki nas obdajajo. Z njim pridobivamo mnogo podatkov o stvareh ki jih lahko uporabimo v različne namene, koristne tako za uporabnike kot za podjetja, ki naprave načrtujejo, proizvajajo in prodajajo. Za IoT je značilno, da se uporablja v panogah, ki niso običajne za IT. Na kratko, IoT precej vsakdanje stvari poveže med seboj s pomočjo interneta ali namenskih omrežij [2].

Cilj IoT je avtomatizem, ki omogoča boljšo izrabo obnovljivih in neobnovljivih virov, večje udobje in izboljšano varnost ljudi. Pri nalogi smo se osredotočili na primer uporabe IoT za potrebe pametnega doma. Pojem pa je popularen tudi v večjih celotah, kot so pametna mesta.

Z razvojem računalništva so senzorske in aktuatorske naprave postale cenovno dostopne. Dostop do svetovnega spleta je postala osnovna življenjska potrebščina. Zaradi tega postaja ideja o internetu stvari popularna ravno v današnjem času.

Na področju IoT obstaja več, med seboj nekompatibilnih sistemov. Nekateri sistemi so v celoti odprtokodni, spet drugi so lastniški in zaprtokodni. Uporabniška prednost odprtokodnih sistemov je, da uporabnik ni vezan na enega proizvajalca strojne opreme. V nalogi se bomo torej posvetili odprtokodnim sistemom. Razvili bomo tudi lasten sistem za pametno hišo s pomočjo računalnika Raspberry Pi in aplikacijo za Applove mobilne naprave z operacijskim sistemom iOS.

V prvem delu diplomskega dela bomo predstavili primere uporabe pametnih naprav v realnem svetu. Pogledali si bomo tudi arhitekture IoT sistemov ter komunikacijske protokole s katerimi IoT naprave medsebojno komunicirajo.

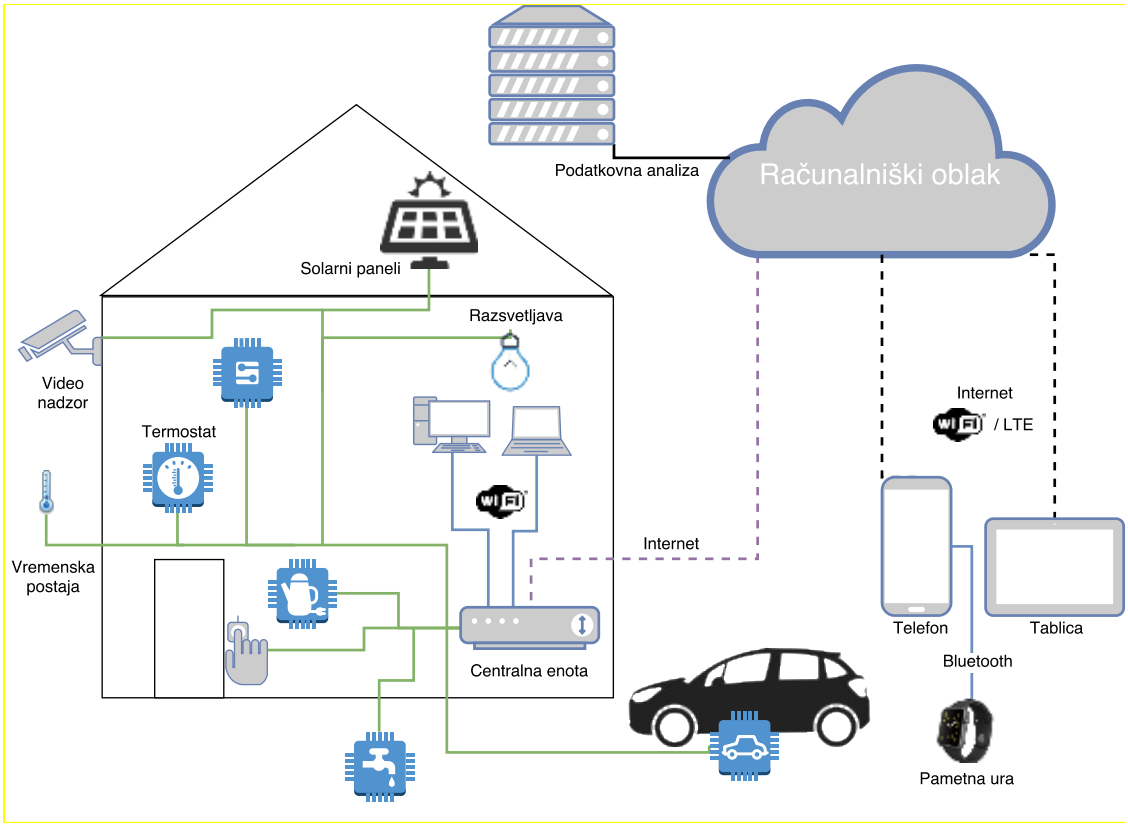
V drugem delu bomo opisali princip odprte kode in medsebojno primerjali nekatere popularnejše licence odprte kode.

V tretjem delu bomo opisali razvoj lastne rešitve za pametno hišo. Najprej bomo prikazali razvoj strojne opreme, za tem programski del, ki bo razdeljen na strežniški del in na odjemalca, t.j. mobilno aplikacijo za operacijski sistem iOS. Opisana bodo tudi nekatera uporabljena orodja. V poglavje bomo vključili še omrežno konfiguracijo lastno razvite rešitve. V sklepu bodo povzete poglavitne težave v IoT, predlagana bo ideja za sistem pametne hiše, ki je nastala na podlagi raziskav in testiranj za potrebe diplomskega dela.

2 PAMETNA HIŠA

Pod pojmom pametna hiša se skriva mnogo pametnih naprav v hiši, ki jih kontrolira inteligenten centralni sistem. Tipično se le-ta povezuje z računalniškim oblakom, nanj pa se prav tako lahko povezujejo pametni telefoni in tablični računalniki. V oblaku se podatki shranjujejo in analizirajo. Pametna hiša vključuje naprave kot so:

- Luči,
- Senzor gibanja,
- Kamera,
- Senzor dima,
- Elektronske ključavnice,
- Napajalne vtičnice,
- Termostat za centralno ogrevanje,
- Elektronsko vodena senčila,
- Senzor vlage,
- Senzor svetlosti v prostoru,
- Vremenska postaja,
- Polnilne postaje za električna prevozna sredstva,
- Pretvornik solarnih panelov.



Slika 1: Primer sheme pametne hiše

2.1 IoT

Besedno zvezo IoT je leta 1999 prvi uporabil Britanski inženir Kevin Ashton. Kevin Ashton je ustanovitelj laboratorija Auto-ID Center na MIT (Massachusetts Institute of Technology), ki se je pričel ukvarjati z identifikacijo RFID in sprožil to, kar danes imenujemo Internet stvari [3].

IoT je koncept povezovanja katerekoli naprav, ki omogočajo povezljivost v internet. Lahko bi rekli, obsega vse naprave ki jim lahko dodelimo IP naslov. Povezovanje vključuje na primer pametne telefone, avtomate za pripravo kave, pralne stroje, hladilnike, pečice, slušalke, luči ipd. Analiza podjetja Gartner pravi, da bo do leta 2020 na splet povezanih več kot 26 milijard naprav. Nekateri celo predvidevajo, da bo ta številka mnogo višja (preko 100 milijard) [4].

Internet stvari predstavlja enega izmed stebrov interneta prihodnosti, ki bo z uporabo standardiziranih komunikacijskih protokolov in omrežne infrastrukture, sposoben samodejne konfiguracije. Prav tako bo razširil Internet na heterogene fizične in navidezne stvari, ki nas obdajajo v vsakdanjem življenju. Te stvari bodo komunicirale med seboj ali s

končnimi uporabniki in tako z novo dodano vrednostjo v obliki informacije o stvarnem stanju postale aktivne udeleženke procesov na različnih področjih človeškega udejstvovanja. Ta koncept zahteva rešitev številnih izzivov na različnih področjih, hkrati pa odpira priložnosti za vrsto novih storitev, aplikacij in poslovnih modelov [5].

2.2 Arhitektura IoT sistemov

Na področju pametnih naprav se poskuša uveljaviti več standardov. Le ti pa med seboj niso kompatibilni. S področjem povezovanja različnih arhitektur obstoječih IoT naprav in snovanjem smernic za nove generacije naprav se ukvarja evropski projekt Internet of Things – Architecture. Nekateri popularni komunikacijski protokoli so WIFI, Bluetooth, ZigBee, Z-wave in RFID (ang. Radio Frequency IDentification).

Slika 2: Prikaz integracije različnih komunikacijskih protokolov [6]

Pogosto so IoT naprave namenjene za specifične primere uporabe. Za izboljšano uporabnost oziroma izkoristek naprav, na primer za potrebe pametnih mest, je nujno, da naprave medsebojno komunicirajo in so na nek način povezane v mrežo oziroma vedo druga za drugo. Na podlagi podatkov iz senzorskih naprav ter inteligentnih algoritmov so lahko aktuatorji optimalno upravljani. S povezovanjem izboljšamo rabo virov, povečamo udobje in varnost ljudi [6].

Nekaj izzivov s katerimi se srečujemo ob integraciji IoT naprav:

- Možnost posodabljanja in interoperabilnost,
- Zmogljivost in razširljivost,
- Zaupnost, varnost in zasebnost,
- Razpoložljivost in odpornost.

2.3 Komunikacijski protokoli IoT naprav

Kot je bilo omenjeno v neformalni definiciji, je IoT naprava, naprava, ki ima IP naslov. Pametna naprava je lahko v svet povezana izključno preko TCP/IP sklada ali pa je na drug način povezana z namenskim prehodom (recimo brezžično s protokolom ZigBee), ki je povezan s spletom.

2.3.1 TCP/IP

Naprave so lahko s spletom povezane žično z ethernet priključkom ali brezžično z uporabo WIFI-ja. Ethernet standarde določa IEEE 802.3, WIFI pa IEEE 802.11.

Protokol IP se v TCP/IP skladu nahaja na internetni plasti. Trenutno prevladuje IP verzije 4, katerega naslovni prostor je že izčrpan. Predvideva se, da bo do leta 2020 v splet povezanih 30 milijard stvari, medtem ko naslovni prostor IP verzije 4 ponuja zgolj 4 milijarde IP naslovov. Trenutno se za ohranjanje naslovov uporablja NAT, ki prepisuje IP naslove, na prehodu v lokalno omrežje. Ob hitrem razvoju te kategorije naprav pa tudi metoda prepisovanja spletnih naslovov ne bo več zadoščala [7].

Rešitev navedenega problema je IP verzije 6. Naslov IPv6 sestavlja 8 16 bitnih šestnajstiških števil ločenih z dvopičjem. IPv6 podpira naslovni prostor v velikosti 2^{128}, kar je približno 3,4 x 10^{38} naslovov. Za boljšo predstavo, to je približno 5 x 10^{28} naslovov za vsakega od približno 6,5 milijard ljudi ali še drugače pogledano 6,0 x 10^{23} različnih naslovov na m^2 zemlje. Omogoča tudi večjo fleksibilnost in avtomatsko konfiguracijo, ki vključuje fizične naslove vmesnikov v naslovni prostor [8].

2.3.2 Bluetooth in iBeacon

Bluetooth je brezžična tehnologija za povezovanje digitalnih elektronskih naprav na razdaljah do nekaj metrov. Njegova uporaba je zelo razširjena, velik je tudi nabor različnih tipov naprav, ki ga uporablja [9].

iBeacon je protokol, ki ga je razvil Apple in je bil letu 2013 predstavljen na Applovi mednarodni razvijalski konferenci WWDC. Različni proizvajalci ponujajo kompatibilne iBeacon oddajnike, pogosto imenovane beacons. Sodijo v kategorijo bluetooth oddajnikov z nizko močjo BLE (ang. Bluetooth low energy). Bluetooth uporablja frekvenčni pas med 2400 MHz in 2483.5 MHz. Tehnologija omogoča pametnim telefonom, tabličnim računalnikom in ostalim napravam z bluetooth vmesnikom, izvajanje akcij v bližini iBeacon oddajnika. Tehnologija omogoča tudi navigacijo v zaprtih prostorih. V primerjavi z GPS-om ima boljšo natančnost in zagotavlja izboljšano avtonomijo pametnih naprav [10].

Vsak beacon ima svoj unikaten identifikator UUID, 'major' ter 'minor' vrednosti. Pametne naprave pa omogočajo tudi merjenje oddaljenosti od oddajnika. Za potrebe naloge smo uporabili iBeacon oddajnik na sliki 3.

Slika 3: iBeacon oddajnik [35]

Za delovanje potrebuje baterijo tip CR2032. Preko aplikacije LightBlue za iOS naprave in AT ukazov lahko spreminjamo nastavitve iBeacon oddajnika. Spremenili smo interval oddajanja za boljšo odzivnost in natančnost na 100ms. Za spremembo te nastavitve smo vnesli ukaz `AT+ADVI0`.

2.3.3 ZigBee

ZigBee je brezžična tehnologija, razvita kot odprti globalni standard, opredeljen z IEEE 802.15.4 specifikacijami. V Evropi deluje na frekvenčnem območju 2,4 GHz. Naprave so cenovno ugodne, imajo nizko porabo ter majhno zakasnitev. Povezava je varna saj je šifrirana s 128 bitno AES enkripcijo. Maksimalna hitrost prenosa podatkov je 250kb/s. Tehnologija daje podporo različnim mrežnim topologijam: Point-to-Point, Point-to-Multipoint, mesh mreža (do 65.000 vozlišč), domet do 1,6 km. Za razvoj standarda skrbi ZigBee Alliance, aktualna je verzija 3.

2.3.4 Z-Wave

Z-wave za razliko od ostalih protokolov (WIFI, Bluetooth, ZigBee) Z-Wave ne deluje na 2,4 GHz območju, ampak za delovanje uporablja frekvenčno območje pod 1 GHz. Povezava je kot pri ZigBee šifrirana s 128 bitno AES enkripcijo. Maksimalna hitrost prenosa je 100kb/s. Podpira mrežno topologijo mesh. Za razvoj skrbi Z-Wave Aliance.

2.4 Obstoječe rešitve

Na trgu je možno kupiti veliko različnih pametnih naprav različnih kategorij.

Na področju pametnih hišnih termostatov je eden popularnejših termostat Nest. Termostat se prilagaja uporabnikovim urnikom in željenim temperaturam. Z njegovo uporabo bi uporabniki lahko privarčevali pri stroških ogrevanja in hkrati dosegli željeno temperaturo doma. Cena aktualnega modela tretje generacije je 249 ameriških dolarjev [11].

Slika 4: Termostat Nest [11]

Na področju razsvetljave podjetje Philips ponuja LED žarnice različnih oblik in velikosti. Žarnice se brezžično povezujejo s pametnimi napravami ali s centralno enoto. Serijo teh naprav tržijo pod imenom Philips Hue. Philips na svoji spletni strani promovira tudi aplikacije tretjih razvijalcev, ki omogočajo kontrolo njihove razsvetljave [12].

Slika 5: Sistem pametne razsvetljave Philips Hue [12]

Podjetje August je znano po pametni ključavnici imenovani August Smart Lock. Podjetje trdi, da je namestitev enostavna in da ključavnico lahko namestijo sami. Uporabnikom omogoča beleženje prehodov in dodeljevanje pravic drugim uporabnikom za odklep doma. Za odklep hiše uporabnik potrebuje pametni telefon in ustrezne pravice. Pametna ključavnica je na voljo za 199 ameriških dolarjev ali pa 229 dolarjev za pametno ključavnico z Applovo Home Kit podporo [13].

Slika 6: Pametna ključavnica August [13]

Podjetje Samsung se je lotilo problema nekompatibilnosti komunikacijskih protokolov z napravo Samsung SmartThings Hub. Naprava podpira komunicijske protokole Bluetooth, Wi-Fi, ZigBee in Z-Wave. Na omenjeno napravo lahko torej povežemo različne pametne naprave različnih proizvajalcev. Samsung pa ponuja tudi serijo senzorskih in aktuatorskih naprav za pametne domove. Naprave lahko upravljamo z uporabo mobilne aplikacije SmartThings Mobile [14]. Cena naprave SmartThings Hub je 99 ameriških dolarjev [15].

Slika 7: Samsung SmartThings Hub [14]

Podjetje Apple ne proizvaja lastne strojne opreme, nudi pa ogrodje za razvoj pametnih naprav in aplikacij imenovano HomeKit. Od iOS 10 bo na mobilne naprave privzeto nameščena tudi aplikacija Home, namenjena upravljanju pametnih naprav, ki imajo HomeKit podporo.

Na spletu lahko najdemo tudi veliko odprtokodnih projektov za pametne hiše. Večinoma jih lahko namestimo na mikroračunalnik Raspberry Pi. Nabor podprtih operacijskih sistemov je širok. Večino ogrodij ima tudi aplikacije za upravljanje za iOS in Android. Nekatera popularnejša ogrodja so:

- OpenHAB,
- Domoticz,
- Calaos,
- Home Assistant,
- OpenMotics,
- LinuxMCE.

2.5 Pametna mesta

Kar 70 % prebivalstva naj bi do leta 2050 živelo v mestih. Da bi človeštvo ob takih napovedih lahko živelo kakovostno, so nujno potrebne spremembe v pristopu upravljanja mest. Trend sprememb se je začel s t. i. konceptom pametnih mest (Smart City), ki postaja del našega vsakdana. Pametno upravljanje mest vključuje tri področja:

- Internet stvari (IoT),
- Obdelava množice podatkov (Big data),
- Upravljanje procesov (BPM).

Upravljanje prometa

Vse gostejši promet v urbanih centrih negativno vpliva na kvaliteto življenja prebivalcev in dnevnih migrantov, kot tudi na ekonomsko učinkovitost podjetij, ki znotraj njega delujejo. Onesnaženost zraka, zvočna in svetlobna onesnaženost ter pretočnost prometa so glavni dejavniki tveganj, ki se jih lotevamo s pametnimi rešitvami za upravljanje prometa.

Upravljanje energetske učinkovitosti stavb in pametnih energetskih omrežij

Energetsko upravljanje stavb in pametno upravljanje omrežij, sta ključna za zagotavljanje energetsko učinkovite rabe energije ter uvajanje novih storitev, tako za gospodinjstva kot za podjetja. Daljinsko odčitavanje porabe energentov, upravljanje energetskih porabnikov in večja zanesljivost oskrbe, so le nekatere prednosti pametnih energetskih omrežij.

Upravljanje razsvetljave

Pametno upravljanje javne razsvetljave je nujno za večjo energetsko učinkovitost in zmanjševanje svetlobne onesnaženosti mest. Mesto si lahko brez posega v obstoječo infrastrukturo zagotovi oddaljeno upravljanje z vsemi elementi javne razsvetljave, hkrati pa z vzpostavitvijo omrežja za upravljanje omogoči uvedbo drugih naprednih rešitev pametnega mesta [16].

3 ODPRTA KODA

Za odprto kodo oziroma prosto programje štejemo različna licencirana računalniška avtorska dela, za katera je značilno, da je koda v prosti uporabi, torej na voljo, pod enakimi pogoji, vsakomur. Odprto kodo zato imenujemo tudi prosto programje (free software). A pozor, to kar je dostopno pri odprti kodi, ni zgolj računalniški program v izvršljivi obliki, pač pa vedno tudi izvorna koda programa – zato izraz odprta koda (ang. Open source). Uporablja se več različic odprtokodnih licenc. V tabeli 1 so primerjane popularnejše odprtokodne licence.

Značilnost odprte kode je ta, da jo smemo predelovati, torej je izvorno kodo vsakomur dovoljeno spreminjati. Prav tako je značilno, da lahko kodo (nespremenjeno ali predelano) v primeru nadaljnjega razširjanja razširjamo le pod enakimi pogoji, pod katerimi smo jo pridobili. Vendar pa takšno redistribuiranje odprte kode ni obvezno – zmeraj lahko odprtokodne programe brez predelav ali pa z lastnimi spremembami uporabljamo tudi zgolj za lastne namene.

Ena od pogostih zmot v zvezi z odprto kodo je, da mora biti ta brezplačna oziroma je zanjo mogoče računati zgolj stroške distribucije. Imetnik licence lahko odprto kodo v resnici največkrat distribuira po kateri koli ceni, ki jo določi sam. Za to obstaja dober ekonomski razlog. Če gre za distribucijo kode v nespremenjeni obliki, potem zanjo verjetno tako in tako ne bo mogel zaračunati veliko, ker bo ta prosto dostopna iz drugih virov. Če pa gre za distribucijo predelane kode, mora cena odražati vrednost spremembe oziroma predelave. Če ta ni previsoka, se odjemalcem izplača plačati za dodatno funkcionalnost. Če pa je previsoka, je vsakdo še vedno pred izbiro, da vzame prosto dostopni izvirnik in ga sam predela oziroma doda novo funkcionalnost (ter morebiti to sam prodaja drugim) [17].

Odprtokodne rešitve se pogosto označujejo tudi s kratico FOSS (Free and Open Source Software). Z uporabo programskih rešitev FOSS pridobijo uporabniki nadzor nad programsko opremo, predvsem pa si zagotovijo trajnost programskih rešitev in se zavarujejo pred „prisilnimi“ nadgradnjami in podobnimi spremembami, v katere jih pogosto silijo ponudniki komercialnih zaprtokodnih programskih rešitev (npr. zamenjave proizvodov MS Office in celo datotečnih formatov verzij 2000, 2003 do 2007). Odprtokodni model v tem pogledu varuje uporabnike tudi pred morebitnimi skritimi funkcijami. Bistvo ideje odprte programske kode je torej v dostopnosti in razpoložljivosti, s čimer pridobijo predvsem uporabniki [18].

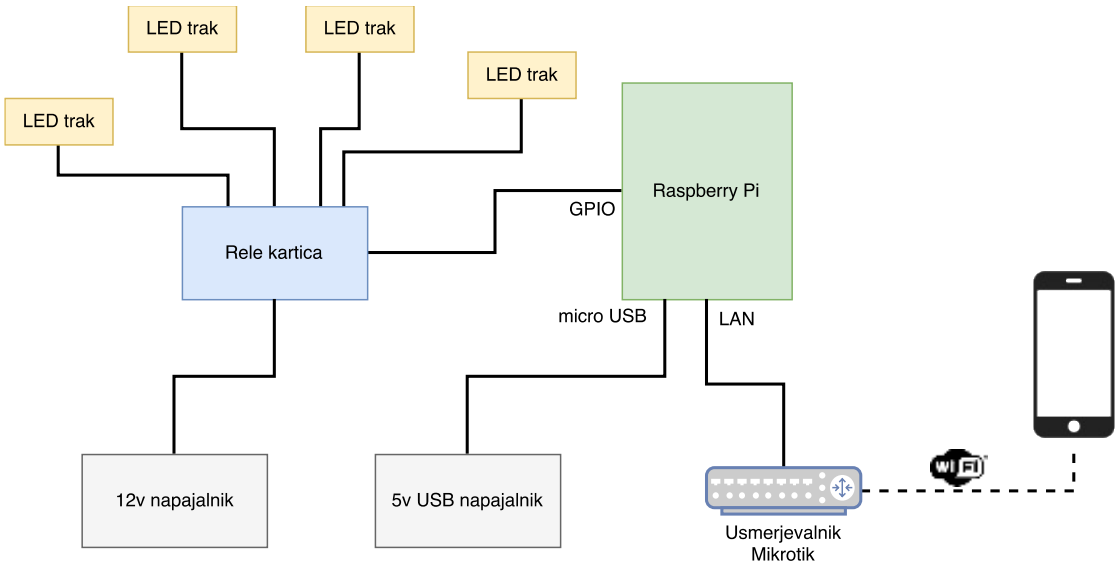
Tabela 1: Primerjava odprtokodnih licenc [19]

	Projektna licenca		Datotečna licenca		Permisivna licenca	
Ime licence	GPL 2	LGPL 2.1	ECLIPSE 1.0	APACHE 2.0	NEW BSD	MIT
Izvorna koda:						
Uporaba	x	x	x	x	x	x
Sprememba	x	x	x	x	x	x
Distribucija	x	x	x	x	x	x
Povezava z drugimi programi, brez ustvarjanja izpeljank dela		x	x	x	x	x
Distribucija						
Dosegljivost izvorne kode	x	x				
Prikaz obvestila o avtorskih pravicah	x	x	x	x	x	x
Vključena kopija licence	x	x	x	x	x	x
Označitev spremembe	x	x	x	x		
Zavrnitev garancije	x	x	x	x	x	x
Zavrnitev odgovornosti	x	x	x	x	x	x
Licenciranje spremenjene datoteke pod enakimi pogoji (“weak copyleft”)	x	x	x			
Licenciranje večjih izpeljanih del pod enakimi pogoji (“strong copyleft”)	x	x				
Dodelitev dovoljenje za uporabo relavantnih patentov			x	x		
Če distribuiraš, ni dovoljeno:						
Uveljavljati patentnih prijav za uporabljeno kodo	x	x	x	x		
Brez dovoljenja uporabljati imena originalnih avtorjev pri oglaševanju				x	x	
Distribuirati, če bi delo bilo pod licenco tretje osebe	x	x				
Distribuirati, če bi to bilo v nasprotju z zakonom ali drugim predpisom	x	x				

4 RAZVOJ LASTNE REŠITVE

Razvoja lastne rešitve smo se lotili z namenom spoznavanja tematike in reševanja izzivov pri vpeljevanju IoT.

Za razvoj lastne rešitve pametne hiše smo uporabili računalnik Raspberry Pi, s katerim smo z uporabo rele kartice upravljali luči. Luči v obliki LED traku za delovanje potrebujejo 12v napajalnik, mikroračunalnik se napaja preko USB napajalnika. Rele kartico in mikroračunalnik smo vgradili v plastično ohišje. Vgradili smo tudi vtičnice za senzorje in LED trakove. V ohišje torej priključimo dva napajalnika, na izhode pa lahko priključimo 4 ločene LED trakove, na vhode pa 3 senzorje. Priključiti moramo tudi LAN kabel, ki je povezan v posebej konfiguriran usmerjevalnik. Pametno hišo lahko tako upravljamo z iOS mobilno aplikacijo ali hitro dostopamo do nekaterih funkcionalnosti z iOS razširitvijo Today. Na sliki 8 je shema razvite rešitve.



Slika 8: Shema sistema pametne hiše

4.1 Strojna oprema

4.1.1 Raspberry Pi

Raspberry Pi je mikroračunalnik, ki ga je razvila neprofitna organizacija iz Velike Britanije. Je dobro orodje za učenje programiranja, saj je nanj možno preprosto priključiti senzorje in aktuatorje, kar naredi programiranje zanimivejše. Tudi njegova cena okrog 25$ za model B je ugodna. Na voljo so tudi močnejše različice, ki so zmožne poganjati tudi Microsoft Windows 10 IoT.

Slika 9: Mikroračunalnik Raspberry Pi model B [20]

Za potrebe naloge smo uporabili Raspberry Pi model B, saj zaradi nizkih strojnih zahtev aplikacije, boljše strojne opreme niti nismo potrebovali. Izbran mikroračunalnik ima ARM procesor proizvajalca Broadcom BCM2835, ki deluje s taktom 700 mHz in 512 MB delovnega pomnilnika. Med priključki lahko zraven etherneta, USB 2.0, HDMI, RCA ter izhoda za slušalke, ki so standardni priključki osebnih računalnikov, najdemo tudi GPIO (ang. General Purpose Input Output) priključke. Nanj lahko priključimo veliko različnih senzorjev, kot so temperaturni senzor, senzor vlage, PIR senzor gibanja, senzor svetlosti, senzor dima ter aktuatorje, kot so LED dioda, PWM gonilnik za elektro motorje, rele kartico ipd [20].

4.1.2 Senzorji

Na GPIO priključke je mogoče priključiti velik nabor senzorjev, katerih podatke lahko v pametni hiši koristno uporabimo. Kot primer smo uporabili digitalni enožični temperaturni senzor DS18B20 (Slika 10). Temperaturni senzor se napaja preko GPIO priključkov in 4,7K upora. Operacijskemu sistemu moramo najprej omogočiti enožično podporo. To naredimo tako, da datoteki /boot/config.txt dodamo naslednjo vrstico.

```
dtoverlay=w1-gpio
```

Nato smo s serijo naslednjih ukazov pridobili vrednost temperaturnega senzorja [21].

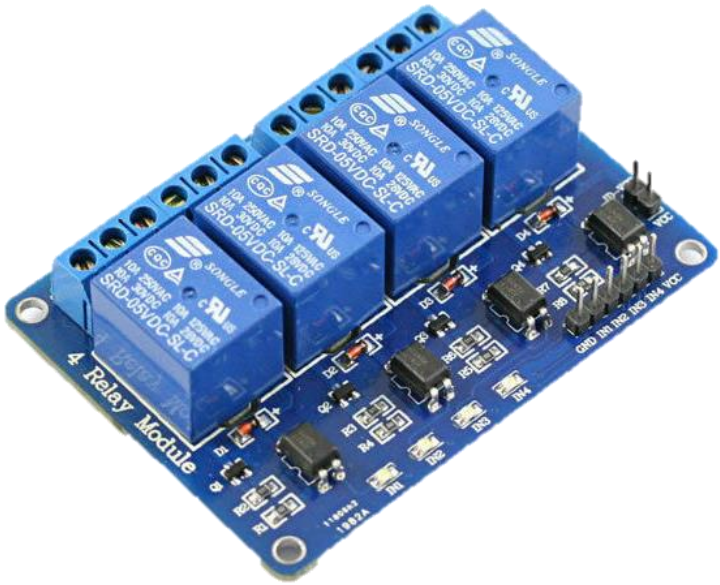
```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices
ls
cd 28-0315040b1cff
cat w1_slave
```

Slika 10: Temperaturni senzor DS18B20 [22]

4.1.3 Aktuatorji

Na GPIO priključke mikroračunalnika je moč priključiti tudi različne tipe aktuatorjev. Za potrebe naloge smo izbrali kartico s štirimi releji (Slika 11). Ti omogočajo vklop in izklop naprav, kot so luč, črpalka, ventilator ipd. Sistem smo testirali z LED trakovi hladno bele barve, ki delujejo na enosmerni napetosti 12v. Samolepilne trakove smo namestili na različne dele sobe in vsakega povezali na ločen izhod rele kartice. Tako smo lahko do želene mere uravnavali tudi intenziteto svetlosti. Če bi želeli nastavljati različne odtenke barv na RGB LED trakovih ali svetila zatemnjevati, bi lahko uporabili PWM krmilnik.

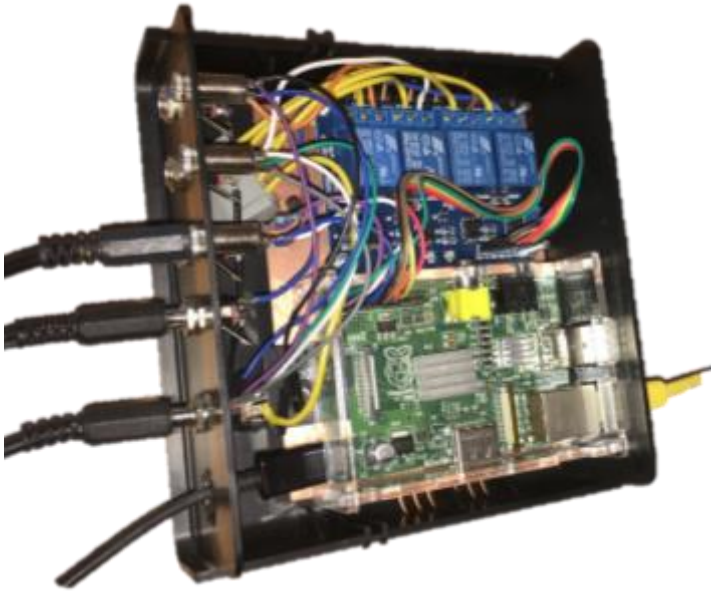
Kartica ima priključke, ki smo jih neposredno povezali na GPIO priključke mikroračunalnika.



Slika 11: Rele kartica

4.1.4 Sestavljanje

Sistem za pametno hišo smo vgradili v univerzalno plastično ohišje z režami za zračenje (Slika 13). V ohišje smo vgradili tudi 4 vtičnice za LED trak, enega za 12v napajalnik za LED trak in 3 vtičnice za različne senzorje Mikroračunalnik in rele kartico smo povezali z namenskimi povezovalnimi kabli, ki jih je potrebno le natakniti na priključke. V izhode rele kartice smo na priključne sponke priključili bakrene tokovodnike. Drug konec smo spajkali na izhodne vtičnice (Slika 12)..



Slika 12: Notranjost sistema pametne hiše

Slika 13: Sistem pametne hiše v ohišju

4.2 Programska oprema na strani strežnika

Na Raspberry Pi smo namestili operacijski sistem Raspbian Jessie lite in programsko knjižnico za GPIO priključke imenovano WiringPi [23]. Za oddaljen dostop smo uporabili spletni strežnik Nginx in programski jezik PHP. Ker novejše različice operacijskega sistema, ki uporabljajo spletne storitve brez SSL certifikata, iOS aplikacije prisilno zapre, smo spletni strežnik konfigurirali tako, da uporablja SSL povezavo. Podpisane certifikate smo brezplačno pridobili s pomočjo storitve Let's Encrypt [24].

Ob zagonu mikroračunalnika je bilo potrebno najprej inicializirati izhodne GPIO priključke. To smo storili z bin/bash skripto. Ker ob zagonu vse sistemske spremenljivke še niso nastavljene, moramo za klic ukaza gpio uporabiti tudi celotno pot. Priključke najprej inicializiramo s pomočjo opcije mode, ki ji sledi številka GPIO priključka. S parametrom out nastavimo priključek kot izhod. Ker se ob inicializaciji izhodi GPIO privzeto vključijo, jih moramo še izključiti. V nasprotnem primeru bi se po električnem izpadu luči ponovno prižgale, kar ni zaželjeno. To storimo z opcijo write, ki ji sledi številka izhoda in željen status izhoda. Pri tem ukazu je potrebno opozoriti, da število 1 izhod izključi, 0 pa vključi.

```
#!/bin/bash
/usr/local/bin/gpio mode 0 out
/usr/local/bin/gpio mode 1 out
/usr/local/bin/gpio mode 2 out
/usr/local/bin/gpio mode 3 out

/usr/local/bin/gpio write 0 1
/usr/local/bin/gpio write 1 1
/usr/local/bin/gpio write 2 1
/usr/local/bin/gpio write 3 1
```

Skripto poganjamo s skripta /etc/rc.local, ki se privzeto zažene ob zagonu operacijskega sistema.

Vklop in izklop luči na strani mikroračunalnika implementira naslednji izsek PHP kode.

```
<?php
if($_POST['geslo'] == "xxxxxxxxxxxxxxxx"){
	$pin = explode(',', $_POST['pin']);
	for($i=0 ; $i<count($pin) ; $i++){
		if(is_numeric($pin[$i])){
			if($_POST['on'] == '1'){
				exec("gpio write ".$pin[$i]." 1");
			} else if($_POST['on'] == '0'){
				exec("gpio write ".$pin[$i]." 0");
			}
		}
	}
}
```

S funkcijo exec se ukaz, ki je podan v obliki teksta izvede podobno kot v ukazni vrstici. Na ta način izvedemo klic programa gpio programske knjižnice Wiring Pi. Za potrebe naloge se uporabnik identificira s pomočjo gesla poslanega z metodo post. Za večuporabniške sisteme bi bilo potrebno sistem nadgraditi tako, da bi administrator lahko upravljal uporabniške račune in pravice dostopa do posameznih vhodov in izhodov.

Temperaturni senzor ima trenutno temperaturo zapisano v tekstovni datoteki. Ta datoteka se za naš senzor nahaja v direktoriju /sys/bus/w1/devices/28-0315040b1cff/w1_slave in ima naslednjo obliko.

```
aa 01 55 00 7f ff 0c 10 5e : crc=5e YES
aa 01 55 00 7f ff 0c 10 5e t=26625
```

Trenutna temperatura je navedena v stopinjah Celzija in je navedena v drugi vrstici za enačajem, vendar ji manjka decimalna vejica. Datoteko temperaturnega senzorja preberemo z ukazom cat. S funkcijo explode in number_format pa oblikujemo izpis v ustrezno obliko. V naslednjem izseku je koda, ki mobilni aplikaciji omogoča branje temperature v prostoru.

```
<?php
$s = exec("cat /sys/bus/w1/devices/28-0315040b1cff/w1_slave");
$polje = explode("t=", $s);
echo number_format($polje[1]/1000, 3);
```

4.3 Razvoj mobilne aplikacije

Aplikacijo smo razvili za Applovo mobilno platformo iOS. Namestimo jo lahko na mobilni telefon iPhone, tablične računalnik iPad in na iPod touch. Razvoj je potekal v razvojne okolju Xcode. Razvijali smo v odprtokodnem programskem jeziku Swift. Za lažje delo s spletnimi storitvami smo s pomočjo CocoaPods vključili knjižnico Alamofire. Za hiter dostop do osnovnih akcij smo razvili tudi Today extension, ki omogoča hiter dostop do nekaterih funkcionalnosti brez odklepanja mobilne naprave. Aplikacija se s pomočjo iBeacon tehnologije zaveda lokacije in lahko proži akcije, kot je vklop in izklop luči. Za nadzor verzij smo uporabili Git in ponudnika github.

4.3.1 Nadzor različic

Nadzor različic je sistem, ki sproti zapisuje spremembe v datoteko ali skupek datotek , da lahko kasneje prikliče določeno različico. Uporabimo ga lahko za poljubne tipe datotek, še posebej pa je primeren za nadzor verzij programske kode. Ločimo dve kategoriji sistemov za nadzor različic in sicer centralizirane, kot je Subversion in CSV in distribuirane, kot GIT [25].

Git

Git je nastal kot odgovor na spremembe pri licenciranju orodja za nadzor verzij BitKeeper (postal je plačljiv). Razvoj je pričela razvijalska skupnost Linux, največ je k razvoju ideje pripomogel ustvarjalec Linux-a Linus Torvalds [26]. Od izida leta 2005 njegova popularnost raste in je trenutno najpopularnejše orodje za nadzor različic [27]. Podpirajo ga različna razvijalska okolja, kot so Eclipse, Netbeans in Xcode. Obstajajo pa tudi programi namenjeni izključno za nadzor nad različicami, kot je ukaz git v konzoli ali pa grafični SourceTree. Kot Linux je Git licenciran z GPL licenco verzije 2.

Git zaznava podatke kot skupek posnetkov miniaturnega datotečnega sistema. Vsakič, ko je stanje projekta shranjeno, Git naredi sliko stanja datotek v določenem trenutku in shrani referenco na posnetek. V primeru, da se datoteke niso spremenile, jih Git ne shrani ponovno in s tem ohranja učinkovitost.. Shrani le povezavo do prejšnje identične datoteke, ki je že shranjena. Git podatke zaznava kot tok posnetkov.

Večina operacij v Git-u potrebuje za delovanje le lokalne datoteke in vire - v splošnem niso potrebne informacije drugega računalnika oziroma strežnika iz omrežja. V primerjavi z

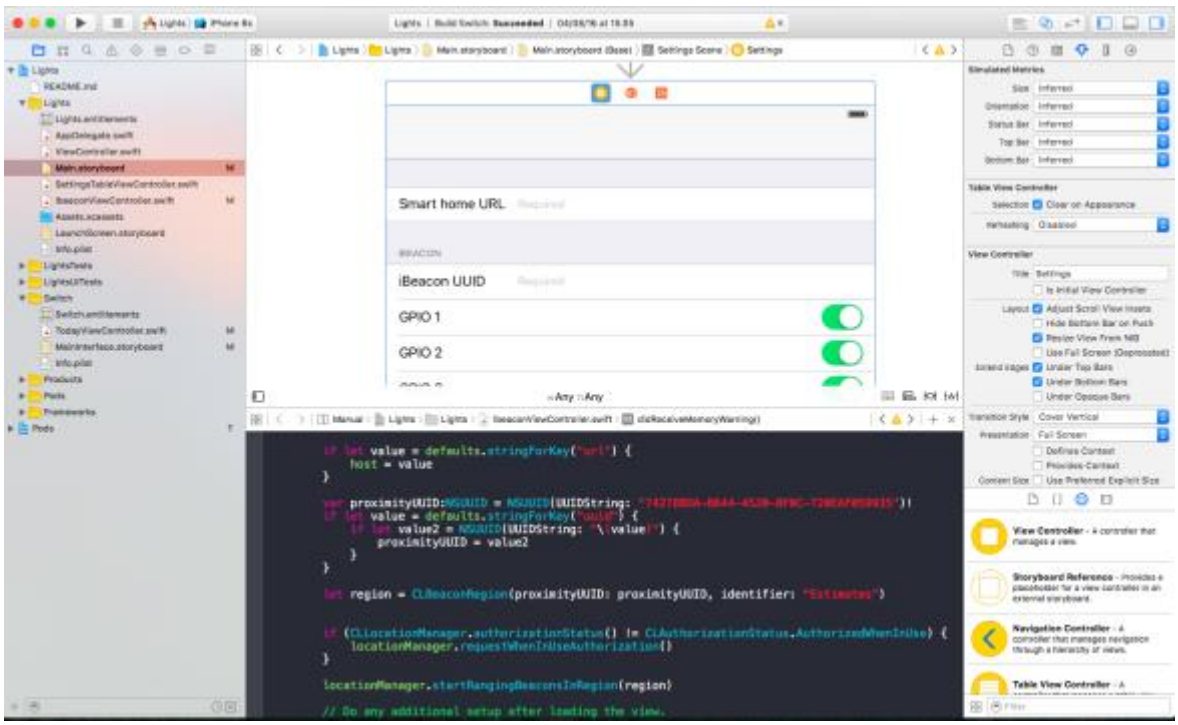
orodji, kot je SVN, ki za vse operacije uporablja centralni strežnik, je večina operacij veliko hitrejših. Vsa zgodovina projekta je shranjena lokalno, zato je večina operacij takojšnjih [25].

Včasih se zgodi, da poleg glavne veje razvoja potrebujemo tudi vzporedno oz. stransko vejo (angl. branch). Stranske veje pridejo prav predvsem pri razvoju delov programa, ki jih še nimamo namena takoj vključiti v glavno vejo [28].

4.3.2 Xcode

Xcode je integrirano razvojno okolje – IDE (ang. Integrated Development Environment) razvit s strani Appla (Slika 14). Okolje lahko uporabljamo samo na računalnikih, ki jih poganja operacijski sistem OS X. Na voljo je brezplačno v trgovini aplikacij App Store. Prva verzija je izšla leta 2003, aktualna različica je 7.3.1. Podpira velik nabor programskih jezikov, kot so C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), in nenazadnje tudi Swift. Okolje vključuje več orodij, ki so razvijalcem v pomoč pri razvoju. To so na primer orodje za izdelavo grafičnih vmesnikov, iOS simulator ter orodja za profiliranje.

V prejšnjih verzijah Xcode je za testiranje na dejanski napravi bila potrebna plačljiva iOS razvijalska licenca, od verzije Xcode 7 naprej, je testiranje brezplačno. Razvijalski račun je še vedno potreben za trženje aplikacij na trgovini App Store.

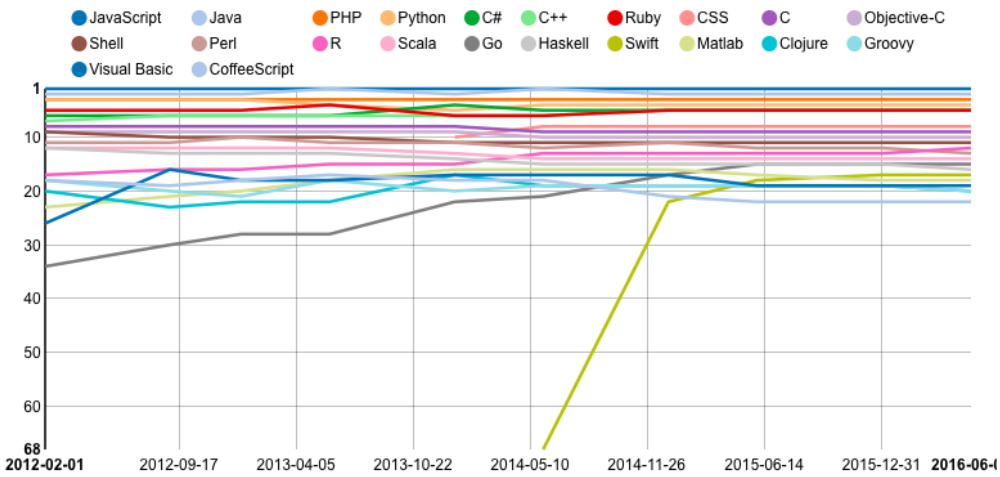


Slika 14: Zaslonska slika razvojnega okolja Xcode

4.3.3 Swift

Swift je programski jezik, ki je bil predstavljen na Applovi razvijalski konferenci leta 2014 [29]. Trenutno je zadnja stabilna različica 2.2.1, za javnost je na voljo predogled različice 3.0. Od decembra 2015 je jezik pod Apache 2.0 licenco, njegova izvorna koda je na voljo na Githubu. Namenjen je razvoju aplikacij za Applove operacijske sisteme iOS, OS X watchOS in tvOS, poganjamo ga lahko tudi na operacijskem sistemu Linux [30].

Jezik je za razliko od Objective-C hitrejši in bolj intuitiven. Po raziskavi podjetja RedMonk junija 2016 je Swift po popularnosti na 17. mestu in njegova popularnost raste. Na grafu (Slika 15) lahko vidimo hitro naraščanje popularnosti jezika Swift [31].



Slika 15: Graf popularnosti programskih jezikov [31]

4.3.4 Cocoa pods

Cocoa pods je upravitelj knjižnic za programska jezika Swift in Objective-C. Orodje knjižnice prenese iz sistema za verzioniranje in ustrezno konfigurira Xcode projekt. Pogosto se uporablja za vključevanje knjižnic tretjih razvijalcev v lastne projekte. Podpira več načinov pridobivanja izvorne kode, kot so git, svn, bzr, http in hg. Program je napisan v jeziku Ruby in se namesti s pomocjo ukaza:

```
sudo gem install cocoapods
```

Inicializiramo ga s pomočjo ukaza pod init v korenskem direktoriju projekta. Ustvari se datoteka imenovana Podfile. V to tekstovno datoteko vpišemo URL do izvorne kode, imena knjižnic ter verzije, ki jo potrebujemo. Sledi primer datoteke Podfile za naš projekt.

```
target 'Lights' do
end
target 'LightsTests' do
end
target 'LightsUITests' do
end
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '8.0'
use_frameworks!
pod 'Alamofire', '~> 3.0'
```

Ko v datoteko ustrezno vpišemo podatke o željenih knjižnicah, jih z ukazom pod install namestimo v projekt. Knjižnice se prenesejo v mapo Pods v korenskem direktoriju projekta, generira se nam tudi nova projektna datoteka s končnico .xcworkspace. Ta že ima vključene izbrane knjižnice. Za odpiranje projekta moramo od tega koraka naprej uporabljati novo projektno datoteko. Preostane še samo vključitev v razrede, kjer bomo knjižnico potrebovali z ukazom import.

4.3.5 Programska knjižnica Alamofire

Kot je razvidno iz primerov poglavja 4.3.4, smo pri razvoju aplikacije pametne hiše uporabili programsko knjižnico Alamofire. Popularna programska knjižnica omogoča lažje in učinkovitejše delo s spletnimi storitvami. S pomočjo le te, smo se povezovali na spletni strežnik na mikroračunalniku. Uporabo knjižnice, bomo v naslednjem izseku kode prikazali na primeru metode pridobitve vrednosti temperaturnega senzorja.

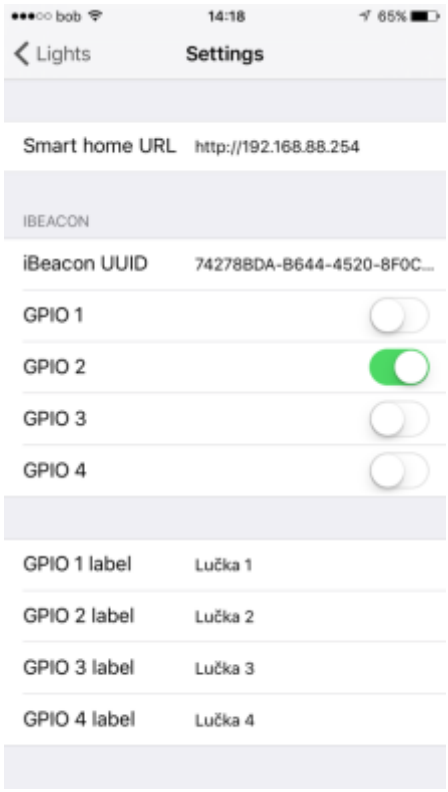
```
@IBAction func refreshTemp(sender: AnyObject) {
    Alamofire.request(.GET, "\(host)/temp.php").responseString { response in
        if let value = response.result.value {
            self.temperatureLabel.text = "\(value)"
        }
        print(response)
    }
}
```

Metodi request smo kot parameter podali HTTP metodo get in URL do želene spletne storitve. Ker nas zanima predvsem odgovor, smo dodali še responseString, ki vrne odgovor v obliki opcijskega besedila. Ker bi bila ob potencialni napaki vrednost odgovora nil, moramo s pomocjo if let sintakse vrednost odviti (ang. unwrap). Ker so vsi klici na spletne storitve

asinhroni, moramo za dostop do grafičnih elementov, kot je Label eksplicitno dodati besedico self.
Grafični vmesnik aplikacije poganja glavna nit, ki ima tudi najvišjo prioriteto. Vse časovno zahtevnejše operacij je potrebno izvajati izven te niti. V nasprotnem primeru lahko operacijski sistem ob zamrznitvi grafičnega vmesnika aplikacijo zaustavi. To je za uporabniško izkušnjo slabo. Delo z nitmi je vgrajeno v večino API-jev (tako operacijskega sistema kot knjižnih tretjih razvijalcev), zato je uporaba relativno preprosta.

Pri naši aplikaciji se ob normalni uporabi, večina klicev izvede v lokalnem omrežju. Ti klici so hitri. V primeru, ko mikroračunalnik ni dosegljiv mora nit čakati na timeout. To lahko traja tudi nekaj sekund. Če bi klic spletne storitve izvedli v glavni niti, bi aplikacija zamrznila in se morebiti tudi zaprla. Zato je uporaba asinhronih klicev spletne storitve naše aplikacije nujna.

4.3.6 Nastavitve aplikacije in shramba NSUserDefaults



Slika 16: Zaslonska slika nastavitev aplikacije pametne hiše

Aplikacija ima tudi možnost nastavljanja nekaterih nastavitev (Slika 14). Uporabnik si lahko za vsako GPIO izhod nastavi lastno oznako, nastaviti mora tudi URL do mikroračunalnika. Za avtomatski vklop in izklop luči glede na bližino iBeacon oddajnika, mora uporabnik vnesti

tudi unikaten identifikator svojega oddajnika in izbrati kateri izhodi se bodo avtomatsko krmilili. Vse nastavitve je potrebno hraniti na napravi tako, da tudi ob izhodu iz aplikacije uporabnik nastavitev ne izgubi. V ta namen smo uporabili razred NSUserDefaults. Ta omogoča hrambo parov ključev in vrednosti za nekatere osnovne podatkovne tipe [32]. Primer shranjevanja in branja vrednosti iz nastavitev aplikacije:

```
let defaults = NSUserDefaults(suiteName:"group.lights.vidrih.net")!
defaults.setObject("\(value)", forKey: "labelGpio1")

if let value = defaults.stringForKey("labelGpio1") {
   label1.text = value
}
```

4.3.7 CoreLocation

Tehnologija iBeacon spada v operacijskem sistemu iOS v kategorijo lokacijskih storitev. Za potrebe aplikacije smo jo uporabili za avtomatsko upravljanje z lučmi, glede na uporabnikovo lokacijo. Ob prvem zagonu mora uporabnik dovoliti aplikaciji uporabo lokacijskih storitev in obvestil. Zaradi varovanja osebnih podatkov, operacijski sistem avtomatsko prikaže obvestilo.

V sobo smo namestili prej opisan iBeacon oddajnik. Uporabnik mora pred uporabo v nastavitve aplikacije, vnesti unikaten identifikator UUID svojega beacona ter izbrati željene luči. Naslednji izsek kode prikazuje uporabo iBeacon tehnologije v aplikaciji. Razred, ki implementira to funkcionalnost, mora z uporabo vzorca delegiranje ustrezati protokolu CLLocationManagerDelegate.

```
override func viewDidLoad() {
        let locationManager = CLLocationManager()
        locationManager.delegate = self;
        var proximityUUID:NSUUID = NSUUID(UUIDString: "74278BDA-B644-4520-8F0C-720EAF059935")!
        let region = CLBeaconRegion(proximityUUID: proximityUUID, identifier: "iBeacon")
        // preverimo ali je uporabnik odobril uporabo lokacijskih storitev
        if (CLLocationManager.authorizationStatus() != CLAuthorizationStatus.AuthorizedWhenInUse) {
            locationManager.requestWhenInUseAuthorization()
        }
      locationManager.startRangingBeaconsInRegion(region)
}
```

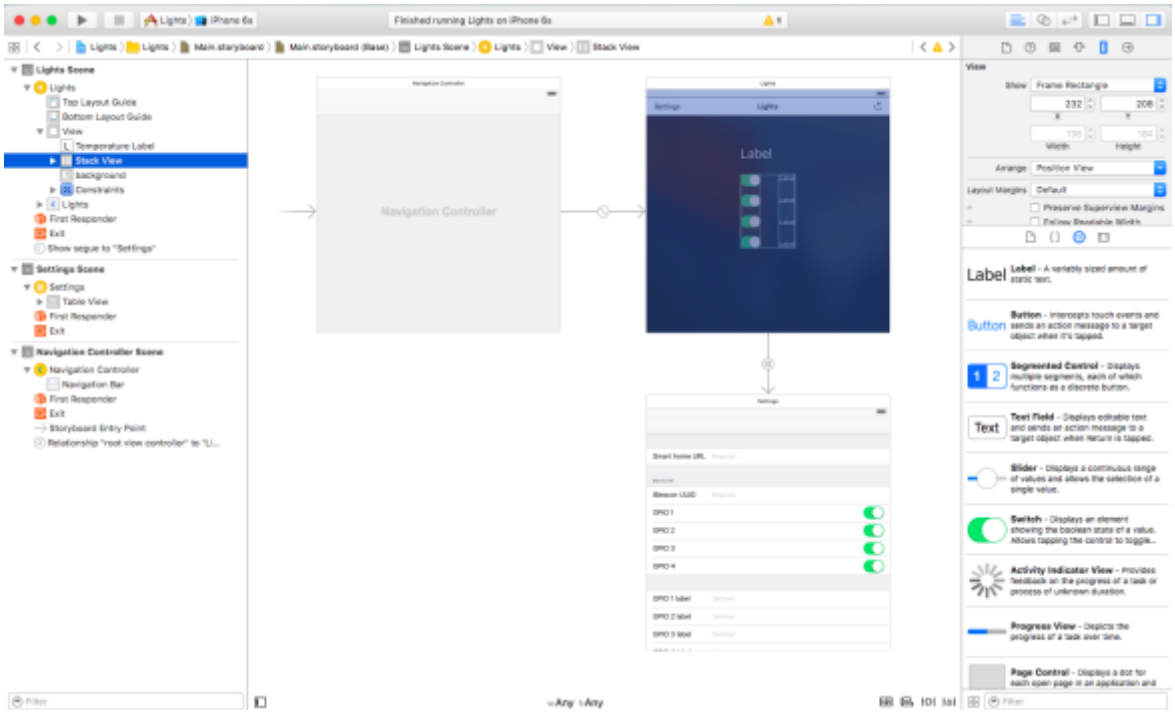
```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
   print(beacons)
}
```

Metoda locationManager se kliče približno vsako sekundo, ne glede na to ali je oddajnik v dosegu. Če je ta v dosegu, lahko preprosto preverimo z beacons.count. S pomočjo atributov

primerka razreda CLBeacon pa lahko ugotovimo tudi oddaljenost od oddajnika, natančnost oddaljenosti, moč signala, 'major' ter 'minor' vrednosti in UUID.

4.3.8 Izdelava uporabniškega vmesnika

Xcode za implementacijo grafičnega vmesnika uporablja Storyboard. Predloge zaslonskih slik razporedimo po platnu in jih med seboj povezujemo z različnimi tipi prehodov. Na Predloge zaslonskih slik nato razporedimo elemente, ki jih posamezen zaslon potrebuje. Primeri teh grafičnih elementov so label, button, text field, slider, switch, image view. To orodje je primerno tudi za prototipiranje in takojšnje testiranje na dejanski napravi (Slika 17).



Slika 17: Gradnja grafičnega uporabniškega vmesnika v okolju Xcode

Za naš projekt smo se odločili, da bomo razvili univerzalno iOS aplikacijo. To pomeni, da jo lahko poganjamo na tabličnih računalnikih iPad, telefonih iPhone in medijskih napravah iPod. Te naprave so različnih velikosti, različno pa je tudi njihovo razmerje zaslonov. Da je uporabniški vmesnik pravilno prikazan na različnih zaslonih moramo grafičnim elementom dodati še omejitve (ang. Constraints). Omejitve dodajamo s potegi miške in tipke ctrl na druge grafične elemente. Nato moramo izbrati željen tip omejitev. Ko ima uporabniški vmesnik željen izgled, ga moramo povezati s kodo. To storimo s potegom miške iz grafičnega elementa, do željene vrstice v kodi, medtem ko na tipkovnici držimo tipko ctrl. Da lahko to storimo, moramo odpreti asistant editor, ki razdeli glavno okno na polovico. Na

prvem izberemo Storyboard, na drugem željeni razred. Ko poteg zaključimo, se nam odpre okno, kjer izberemo tip povezave, ki ga želimo. Povezave lahko izvedejo klice funkcij ali pa jih imamo v kodi kot spremenljivke in lahko dostopamo do njihovih lastnosti. Primer kjer želimo, da dogodek proži klic funkcije, je klik na gumb, natančneje: ko uporabnik izpusti gumb v notranjosti elementa (ang. touch up inside). Primer, ko želimo dostopati le do lastnosti elementa, je element z besedilom (ang. Label). Temu elementu lahko nastavimo besedilo preko lastnosti text, nastavljamo pa lahko tudi barvo, velikost ipd.

Podobno kot pri povezavi grafičnih elementov s kodo, lahko naredimo tudi preproste prehode med različnimi zasloni. Pri naši aplikaciji smo naredili prehod pri kliku na gumb Settings. Tip prehoda, ki smo ga izbrali, je prikaži (ang. Show). Ta tip prehoda animira prehod med zasloni, tako da prikaže nov zaslon z leve strani. Avtomatsko se pojavi tudi delujoč gumb za prehod na prejšnji zaslon. Uporabnik lahko pride na prejšnji zaslon tudi s potegom prsta iz levega roba zaslona v desno. Drugi tipi prehodov so še: prikaži podrobnosti (ang. Show detail), modalni prikaz (ang. Present modally) in prikaz pojavnega okna (ang. Present as popover). Nekateri prehodi se razlikujejo le pri napravah z večjim zaslonom, modalni prikaz pa popolnoma zamenja trenuten zaslon in prejšnjega tipično uniči. Prehod lahko izvedemo programsko in sicer z metodo performSegueWithIdentifier in prehod na prejšnji zaslon z unwindForSegue.

4.3.9 Today extension – hiter dostop

V operacijske sistemu iOS ima uporabnik vedno možnost s potegom navzdol - z vrha zaslona, odpreti zaslon z obvestili in hitrim dostopom do aplikacij imenovanimi Today. Za potrebe naloge, smo razvili preprosto razširitev, za hiter dostop do stikala za luči ter do temperaturnega senzorja. Gradnja grafičnega uporabniškega vmesnika poteka podobno, kot pri aplikaciji. Predloga kontrolerja se razlikuje v metodi widgetPerformUpdateWithCompletionHandler, ki jo operacijski sistem kliče, ko želi vsebino posodobiti. Ko je naprava v načinu varčevanja z energijo, se lahko ta klic izvede manj pogosto. Za naš primer smo v metodi posodobili vrednost temperaturnega senzorja (Slika 18).

Slika 18: Zaslonska slika hitrega dostopa Today extension

4.4 Konfiguracija in namestitev

Sistem pametne hiše smo povezali v domače omrežje z LAN povezavo z usmerjevalnikom (ang. Router) hAP ac proizvajalca Mikrotik (Slika 19). Podobno bi lahko naredili z USB WIFI kartico, ki bi jo povezali v domače brezžično omrežje. Usmerjevalnik smo konfigurirali tako, da lahko dostopamo do njega lokalno ali prek spleta. V ta namen smo pripravili tudi poddomeno, ki kaže na naš zunanji IP naslov. Tako smo pri ponudniku DNS storitev dodali DNS tip zapisa A ter vpisali zunaji IP naslov in ime željene poddomene.

Na mikroračunalniku smo pustili privzeto konfiguracijo dinamičnega pridobivanja IP naslova, na usmerjevalniku smo rezervirali IP 192.168.88.254 za MAC naslov vmesnika mikroračunalnika. Nastaviti smo morali tudi mapiranje vrat 443, za zunanji dostop do HTTPS strežnika. To smo dosegli z naslednjim ukazom vnesenem preko ukazne vrstice usmerjevalnika [33].

```
/ip firewall nat
add action=dst-nat chain=dstnat disabled=no dst-port=443 in-interface=ether1-gateway
protocol=tcp to-addresses=192.168.88.254 to-ports=443
```

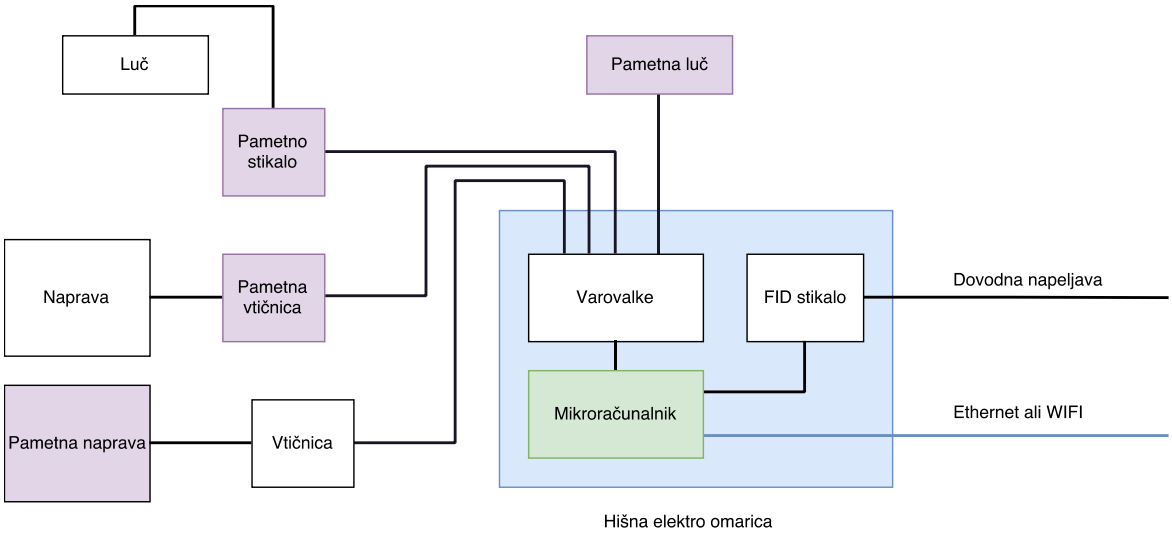
Za lokalni dostop smo dodali statičen DNS vnos za enako poddomeno, kot smo jo ustvarili prej in jo nastavili na IP naslov mikroračunalnika. Na ta način tudi ob izpadu povezave s spletom sistem deluje, zaradi manj vmesnih prehodov je povezava lokalno odzivnejša.

Slika 19: Usmerjevalnik Mikrotik hAP ac [34]

5 SKLEP

IoT se še vedno srečuje z veliko izzivi. Največji je, da se ljudje še ne zavedajo vseh pozitivnih lastnosti in primerov uporabe. Ljudje se zaradi filmov celo bojijo besednih zvez kot npr. umetna inteligenca. Delno je skrb upravičena, saj je problem varnosti in zasebnosti pri IoT ključnega pomena saj širok nabor naprav vključuje tudi zelo pomembne naprave, kot so ključavnice, alarmne naprave in medicinske naprave, kot merilec srčnega tlaka. Ti podatki so lahko relativno preprosto zlorabljeni, zato je zanesljiva kontrola dostopa nujna. Podatkov si tipično ne lastimo sami, vendar so v t.i. oblaku, kjer ne moremo zagotovo vedeti kdo vse lahko do njih dostopa. Drug problem je nepoznavanje tehnologije tehničnega osebja (elektrikarjev), saj klasični tehniki ne znajo pravilno priključiti in konfigurirati sistema za pametno hišo. Za priključitev je potreben razmeroma drag specialist.

Pri nalogi se nam je podlagi raziskav in testiranj porodila ideja za cenovno ugoden in preprost sistem za pametno hišo, ki ga bomo poskušali razviti v prihodnosti. Shema ideje je na sliki 20. Prednost idejnega sistema je, da ne potrebujemo namenske elektroinštalacije. Mikroračunalnik je lahko vgrajen v obstoječo hišno elektro omarico. Povezan je na električno omrežje, preko katerega bi potekala tudi komunikacija. Razvili bi pametna stikala in pametne vtičnice, ki bi se avtomatsko povezale z mikroračunalnikom. Mikroračunalnik bi bil priključen v internet preko žične ali brezžične povezave. Za konfiguracijo bi potrebovali le telefon ali tablični računalnik z nameščeno aplikacijo pametne hiše. Le ta bi ob prvi uporabi samodejno zaznala vse priključene pametne naprave in zahtevala kreiranje uporabniških računov. Administrator bi dodeljeval pravice posameznim uporabnikom za dostop do naprav. Možnosti uporabe te tehnologije se ne končajo samo pri stikalih in vtičnicah, saj bi se lahko tretji razvijalci elektronskih naprav integrirali na sistem. Napravo bi enostavno priključili v omrežje, kot do sedaj, vendar bi imeli dostop do njenih funkcionalnosti. Tako bi naprave, kot so elektronska senčila, pametni hladilnik, klimatska naprava in centralna kurjava še vedno imele popolnoma enak postopek namestitve, vendar veliko dodatnih funkcionalnosti. Naprave bi lahko namestila oseba, ki podrobno ne pozna omenjenega sistema. Sistem bi bil odporen na motilce signalov oziroma na motnje drugih brezžičnih sistemov v gosteje naseljenih področjih.



Slika 20: Idejna shema

Če povzamemo, IoT bo v bližnji prihodnosti doživel eksponentno rast. To razvijalcem ponuja veliko priložnosti za razvoj novih naprav in storitev, uporabnikom bo izboljšalo kvaliteto življenja, doprineslo pa bo tudi k manjši porabi obnovljivih in neobnovljivih virov.

Viri

[1] „Kaj je pametna hiša ali inteligentna hiša?“ [Na spletu]. Dostopno: http://www.ps-promis.si/en/.

[2] Vidonja Tomaž, „Internet stvari“, *Finance*, let. 5, 2015.

[3] „OpComm - Zakaj postaja Internet stvari največja globalna panoga?“, 2013. [Na spletu]. Dostopno: http://www.opcomm.eu/sl/medijsko-sredisce/blog/139-zakaj-postaja-internet-stvari-najveja-globalna-panoga.

[4] Babnik Matjaž, „Mi lahko nekdo že pove kaj je internet stvari (The Internet of Things – IoT)! – Konica Minolta Slovenija“, 2016. [Na spletu]. Dostopno: http://www.konicaminolta.si/sl/poslovne-resitve/blog-sl/2016/06/29/mi-lahko-nekdo-ze-pove-kaj-je-internet-stvari-the-internet-of-things-iot/.

[5] M. Mohorčič, „Internet stvari – izzivi in priložnosti“.

[6] S. A. Bauer Martin, Boussard Mathieu, Bui Nicola, Carrez Francois, Jardak Christine, Jourik De Loof, Magerkurth Carsten, Meissner Stefan, Nettsträter Andreas, Olivereau Alexis, Thoma Matthias, Walewski Joachim, Stefa Julinda, „Internet of Things – Architecture IoT-A“, 2013.

[7] Poulin Chris, „The Importance of IPv6 and the Internet of Things“, 2014. [Na spletu]. Dostopno: https://securityintelligence.com/the-importance-of-ipv6-and-the-internet-of-things/.

[8] „IPv6 - Wikipedija, prosta enciklopedija“. [Na spletu]. Dostopno: https://sl.wikipedia.org/wiki/IPv6.

[9] „Bluetooth - Wikipedija, prosta enciklopedija“. [Na spletu]. Dostopno: https://sl.wikipedia.org/wiki/Bluetooth.

[10] „iBeacon“. [Na spletu]. Dostopno: https://en.wikipedia.org/wiki/IBeacon. [Dostopano: 17-avg-2016].

[11] „Home | Nest“. [Na spletu]. Dostopno: https://nest.com/.

[12] „Meet hue | en-XX“. [Na spletu]. Dostopno: http://www2.meethue.com/en-xx/.

[13] „August Smart Lock | August“. [Na spletu]. Dostopno: http://august.com/products/august-smart-lock/.

[14] „Smart Home. Intelligent Living. | SmartThings“. [Na spletu]. Dostopno: https://www.smartthings.com/.

[15] „Samsung SmartThings Hub, 2nd Generation - - Amazon.com“. [Na spletu]. Dostopno: https://www.amazon.com/dp/B010NZV0GE/?tag=thewire06-20&linkCode=xm2&ascsubtag=WC82592.

[16] „Pametna mesta | FMC - Sistemski integrator“. [Na spletu]. Dostopno:

http://fmc.si/resitve/pametna_mesta/.

[17] Berčič Boštjan, „Kako odprta je »odprta koda« | MonitorPro“, *Monitor PRO*, 2011. [Na spletu]. Dostopno: http://www.monitorpro.si/41846/praksa/kako-odprta-je-odprta-koda/.

[18] S. Lah, „Odprta koda - ne v ceni, v uporabni vrednosti je bistvo!“, *Organ. znanja*, 2010.

[19] „Free and Open Source License Comparison (David Lee Todd, Unknown Product Manager)“, 2007. [Na spletu]. Dostopno: https://blogs.oracle.com/davidleetodd/entry/free_and_open_source_license.

[20] S. Sebastjan, „Informacijski sistem za pametni vrt“, 2015.

[21] „Adafruit’s Raspberry Pi Lesson 11. DS18B20 Temperature Sensing“, 2015.

[22] „Waterproof Digital Thermal Probe or Sensor DS18B20 Length 1M New | eBay“. [Na spletu]. Dostopno: http://www.ebay.com/itm/Waterproof-Digital-Thermal-Probe-or-Sensor-DS18B20-Length-1M-New-/281404601361?hash=item4185057411:g:98kAAOSwEK9T4O7M.

[23] Henderson Gordon, „Raspberry Pi | Wiring | Gordons Projects“, 2012. [Na spletu]. Dostopno: https://projects.drogon.net/raspberry-pi/wiringpi/.

[24] Anicas Mitchell, „How To Secure Nginx with Let’s Encrypt on Ubuntu 14.04 | DigitalOcean“, 2015. [Na spletu]. Dostopno: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-14-04.

[25] „Git - O nadzoru različic“. [Na spletu]. Dostopno: https://git-scm.com/book/sl/v2/Pri%C4%8Detek-O-nadzoru-razli%C4%8Dic.

[26] „Git - Kratka zgodovina Git-a“. [Na spletu]. Dostopno: https://git-scm.com/book/sl/v2/Pri%C4%8Detek-Kratka-zgodovina-Git-a.

[27] *Eclipse Community Survey 2014 results | Ian Skerrett*. Ianskerrett.wordpress.com, 2014.

[28] J. Mihelič, „Altruist svn in egocentrik git: dva sistema za vodenje izvedb Laboratorij za algoritme in podakovne strukture Fakulteta za računalništvo in informatiko Univerza v Ljubljani“, 2015.

[29] *Swift Has Reached 1.0*. Apple, 2014.

[30] „Swift.org - About Swift“. [Na spletu]. Dostopno: https://swift.org/about/#swiftorg-and-open-source.

[31] S. O’Grady, „The RedMonk Programming Language Rankings: June 2016“, 2016. [Na spletu]. Dostopno: http://redmonk.com/sogrady/2016/07/20/language-rankings-

6-16/. [Dostopano: 21-avg-2016].

[32] „NSUserDefaults Class Reference“. [Na spletu]. Dostopno: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/.

[33] „How to Port Forward in Mikrotik Router“. [Na spletu]. Dostopno: http://www.icafemenu.com/how-to-port-forward-in-mikrotik-router.htm.

[34] „RouterBoard.com : hAP ac“, 2016. [Na spletu]. Dostopno: http://routerboard.com/RB962UiGS-5HacT2HnT.

[35] „Diymall Ibeacon Bluetooth Module 4 0 Ble Positioning Sensor Wireless | eBay“. [Na spletu]. Dostopno: http://www.ebay.com/itm/181605556854?_trksid=p2057872.m2749.l2648&ssPageName=STRK%3AMEBIDX%3AIT.