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
Lektor(ica):	ime in priimek, naziv

KAZALO

KAZALO SLIK

Uporabljene kratice

TODO preveri ali so zajete vse kratice, razloži

IoT – Internet stvari (Internet of Things)

IP – Internet protocol

GPIO – General Purpose Input Output

SD

SSH

API

JSON

NAT

REST

PWM

BPM

git, svn, bzr, http in hg

GPS

RFID - Radio-Frequency Identification

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 vsega je povezovanje, upravljanje in nadzor nad porabniki, katerega cilj je zviševanja nivoja udobja, varčnosti in varnosti. Primer uporabe je, da se ob odhodu od doma avtomatsko izključijo vsa svetila in druge naprave v hiši, vključi se alarm, zniža nivo ogrevanja ter vključi simulacija prisotnosti [1].

Pri IoT gre za digitalizacijo fizičnega sveta, torej naprav in stvari, ki nas obdajajo. S tem pridobivamo različne podatke o stvareh in jih uporabimo v različne namene, koristne tako za uporabnike kot za podjetja, ki te naprave načrtujejo, proizvajajo in prodajajo. Pri IoT gre predvsem za tehnologije, ki ponujajo podporo pametnim rešitvam v panogah, ki niso običajne za IT. Na kratko, internet stvari precej vsakdanje »stvari« poveže med seboj s pomočjo interneta ali namenskih omrežij [2].

Cilj IoT je avtomatizem, ki prinaša boljšo izrabo obnovljivih in neobnovljivih virov, večje udobje in izboljšano varnost vseh. 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 pametne senzorske in aktuatorske naprave postale cenovno dostopne. Tudi dostop do interneta se smatra kot osnovna življenska potrebščina. Zaradi teh razlogov IoT naprave eksponentno pridobivajo na popularnosti ravno v današnjem času.

Na področju IoT obstaja več medseboj nekompatibilnih standardov. Nekateri sistemi so v celoti odprtokodni, spet drugi so lastniški in zaprtokodni. Uporabniška prednost takšnih 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

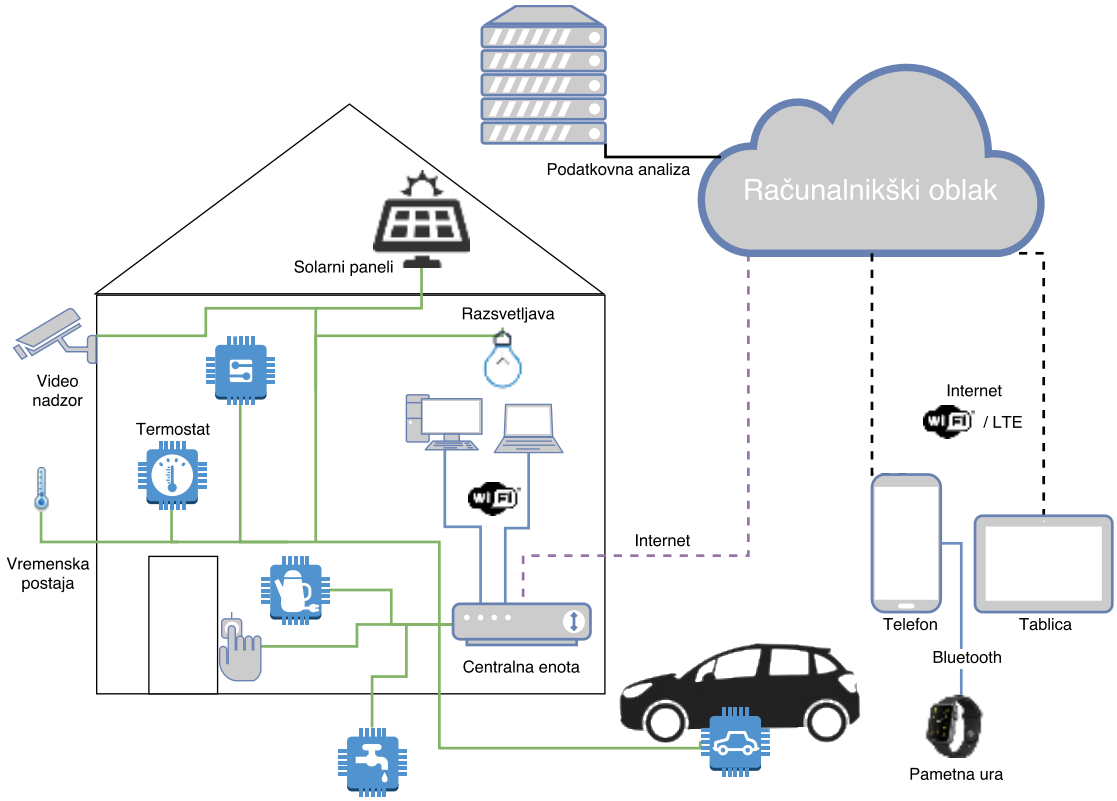
pomočjo računalnika Raspberry Pi in aplikacijo za Applove mobilne naprave z operacijskim sistemom iOS.

V prvem delu diplomskega dela bomo predstavili primere uporabe pametnih naprav v realnem svetu. V drugem delu bomo omenili princip odprte kode in omenili nekaj orodij in ponudnikov, ki podpirajo ta princip. V tretjem delu bodo predstavljeni sodobni komunikacijski protokoli, ki jih pametne naprave uporabljajo. V četrtem delu bomo opisali razvoj lastne rešitve za pametno hišo. Najprej bomo prikazali razvoj strojne opreme, za tem pa še 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. Peti del bo govoril o konfiguraciji lastno razvitega sistema v omrežje. Šesti del bo povzel poglavitne težave v IoT, v sklepu pa bo predlagan sistem za pametno hišo, ki je nastal na podlagi raziskav in testiranj za potrebe diplomskega dela.
TODO: preveri opis poglavij

2 PAMETNA HIŠA

Pod pojmom pametna hiša se skriva mnogo pametnih naprav, ki jih kontrolira inteligenten centralni sistem. Tipično se ta povezuje z računalniškim oblakom, nanj pa se prav tako 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
- Elektronske rolete
- Senzor vlage
- Senzor svetlosti v prostoru
- Vremenska postaja
- Polnilne postaje za električna prevozna sredstva
- Kontroler solarnih panelov



Slika 1: Primer: Shema pametne hiše

2.1 IoT

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

IoT je koncept povezovanja katerekoli naprave, ki omogoča povezljivost v internet. Lahko bi rekli, vse čemu lahko dodelimo IP naslov. To vključuje 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, 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 naprave narejene za specifične primere uporabe. Za boljšo izrabo, recimo za potrebe pametnih mest je nujno, da se naprave medsebojno povezujejo in vedo druga za drugo. Na podlagi podatkov iz senzorskih naprav ter inteligentnih algoritmov so lahko aktuatorji optimalno upravljani. S tem izboljšamo rabo virov, izboljšamo udobje in povečujemo človekovo varnost [6]. Nekaj problemov 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 mora IoT naprava imeti 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 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 miljard stvari, medtem ko naslovni prostor IP verzije 4 ponuja zgolj 4 miljarde 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 dovolj [7].

Rešitev tega 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 $3,4 \times 10^{38}$ naslovov. Za boljšo predstavo, to je približno 5×10^{28} naslovov za vsakega od približno 6,5 milijard ljudi ali še drugače pogledano $6,0 \times 10^{23}$ različnih naslovov na m^2 zemlje. Omogoča tudi večjo fleksibilnost in avtomatsko konfiguracijo, ki vključuje fizične naslove vmesnikov v naslovni prostor [8].

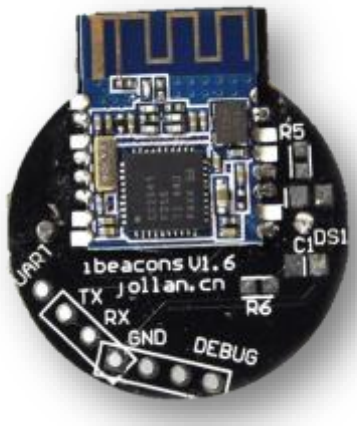
2.3.2 Bluetooth in iBeacon

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

iBeacon je protokol razvit s strani Appla in je bil predstavljen na Applovi mednarodni razvijalski konferenci WWDC leta 2013. 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 boljšo avtonomijo pametnih naprav [10].

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



Slika 3: iBeacon oddajnik [31]

Za delovanje potrebuje baterijo tipa 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 TODO

2.3.4 Z-wave TODO

2.3.5 Ostali TODO

2.4 Obstoječe rešitve

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

Na področju pametnih termostatov je najpopularnejši termostat Nest. Termostat se prilagaja urnikom in željenim temperaturam. Z njegovo uporabo bi uporabniki naj privarčevali na ogrevanju in hkrati imeli željeno temperaturo doma. Cena aktualnega modela tretje generacije je 249 ameriških dolarjev [11].

Slika 4: Termostat Nest

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

Podjetje August je znano po pametni ključavnici imenovani August Smart Lock. Podjetje trdi, da si ključavnico lahko vsak namesti sam. 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

Podjetje Samsung je naslovilo problem medsebojne nekompatibilnosti 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

Podjetje Apple sicer 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 delujejo z HomeKit.

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.

OpenHAB

Projekt OpenHAB se deli na dve dela in sicer openhab-runtime (izvaja se na strežniku) in openhab-designer (uporabniku prijazno konfiguracijsko orodje). Projekt je napisan v Javi in temelji ne OSGi. Odjemalski aplikaciji sta na voljo za iOS in Android ali pa kot spletna stran. Za razvoj in integracijo z lastnimi rešitvami je na razpolago tudi REST API [16].

TODO kratek opis ostalih odprtokodnih ogrodij:
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 z mesti. Trend sprememb se je začel s t. i. konceptom pametnih mest (Smart City), ki postaja del našega vsakdana. Pametno upravljanje mest vključuje tri bistvene sestavine:

- 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 samo ekonomsko učinkovitost podjetij, ki znotraj njega delujejo. Onesnaženost zraka, zvočna in svetlobna onesnaženost ter pretočnost prometa so glavni dejavniki tveganj, ki jih naslavljamo 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 [17].

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) [18].

Tabela 1: Primerjava odprtokodnih licenc [19]

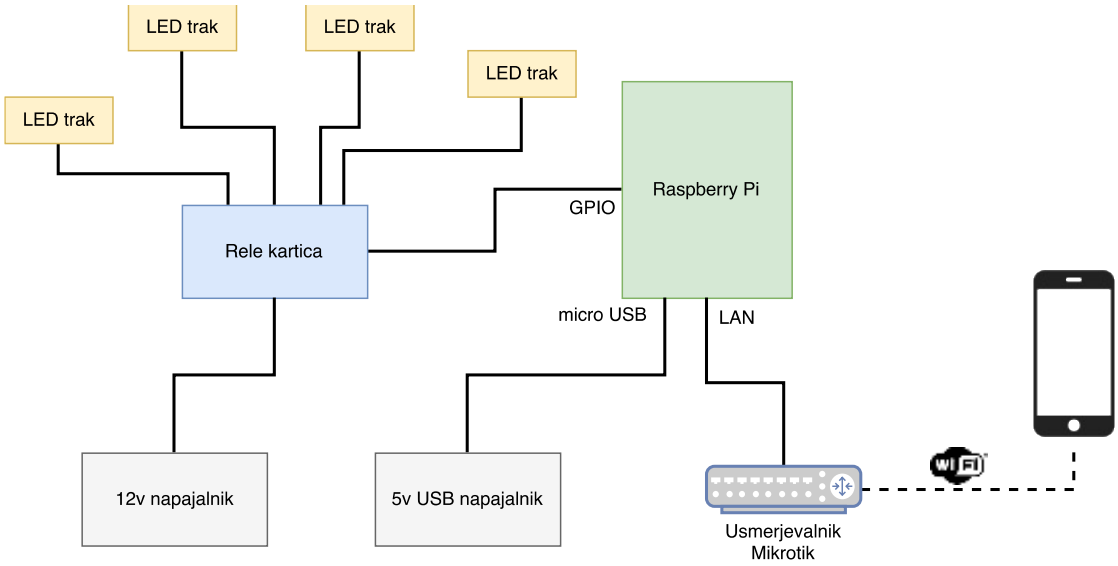
	Projektne		**Datotečne**		**Permisivne**	
Licenca	GPL 2	LGPL 2.1	ECLIPSE 1.0	APACHE 2.0	NEW BSD	MIT
Kodo lahko:						
Uporabiš	x	x	x	x	x	x
Spremeniš	x	x	x	x	x	x
Distribuiraš	x	x	x	x	x	x
Link to other programs under at least some circumstances without creating a derived work		x	x	x	x	x
Če distribuiraš moraš:						
Naredit izvorno kodo dosegljivo	x	x				
Prikazati obvestilo o avtorskih pravicah	x	x	x	x	x	x
Vključiti kopijo licence	x	x	x	x	x	x
Označiti spremembe	x	x	x	x		
Zavrniti garancijo	x	x	x	x	x	x
Zavrniti odgovornost	x	x	x	x	x	x
Licencirati spremenjene datoteke pod enakimi pogoji (“weak copyleft”)	x	x	x			
Licencirati večja izpeljana dela pod enakimi pogoji (“strong copyleft”)	x	x				
Dodeli dovoljenje za uporabo relavantnih patentov, ki jih lastiš			x	x		
Če distribuiraš ne smeš:						
Uveljavljati patentnih prijav za uporabljeno kodo	x	x	x	x		
Brez dovoljenja uporabljati imen originalnih avtorjev pri oglaševanje				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				

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 [20]. TODO: Zgodovina odprte kode

4 RAZVOJ LASTNE REŠITVE

Shema lastno razvite rešitve je na sliki 4.
TODO kratek opis.



Slika 8: Shema sistema pametne hiše

4.1 Strojna oprema

4.1.1 Raspberry Pi

Raspberry pi je mikroračunalnik razvit s strani neprofitne organizacije 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, ki se začne pri 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

Za potrebe naloge smo uporabili Raspberry Pi model B saj zaradi nizkih strojnih zahtev aplikacije boljše strojne opreme nismo potrebovali. Izbran mikroračunalnik ima ARM procesor proizvajalca Broadcom BCM2835, ki deluje s taktom 700 mHz in 512 MB delovnega pomnilnika. Od priključkov 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.

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 temperaturni senzor DS18B20.

Slika 10: Temperaturni senzor DS18B20

Testirali smo pa tudi senzor svetlosti in PIR senzor gibanja. TODO sliki

4.1.3 Aktuatorji

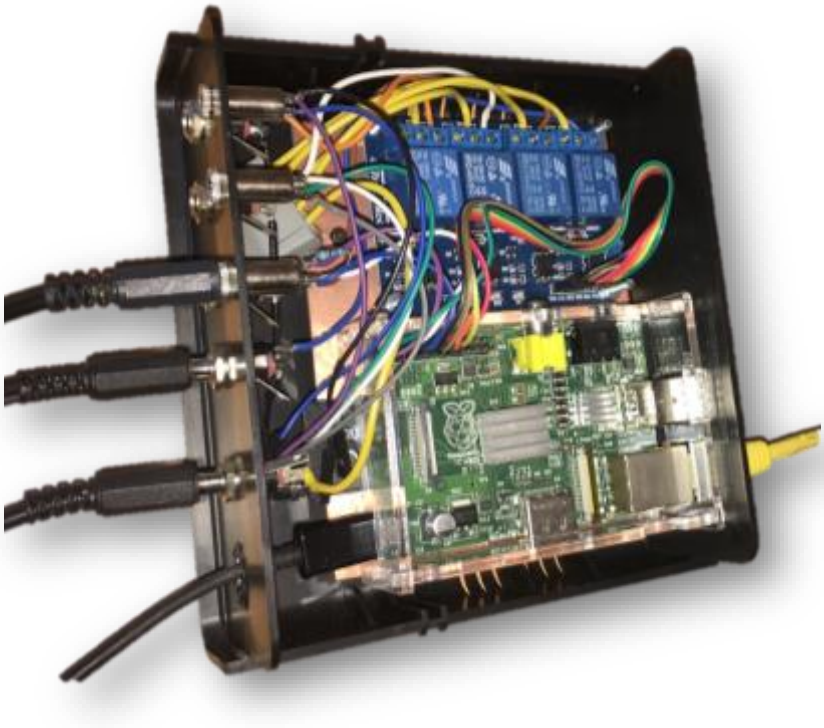
Na GPIO priključke mikro računalnika je moč priključiti različe tipe aktuatorjev. Za potrebe naloge smo izbrali kartico s štirimi releji. Ti omogočajo vklop in izklop naprav, kot so luč, črpalka, ventilator ipd. Sistem smo testirali z LED trakovi bele barve, ki delujejo na napetosti 12v. Samolepilne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

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. V ohišje smo vgradili tudi 4 priključke za LED trak, enega za 12v napajalnik za LED trak in 3 priključke za različne senzorje.



Slika 12: 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 [21]. Za oddaljen dostop smo uporabili spletni strežnik Nginx in programski jezik PHP. Ker novejše različice operacijskega sistema iOS aplikacije, ki uporabljajo spletne storitve brez SSL certifikata prisilno zapre smo spletni strežnik konfigurirali tako, da uporablja SSL povezavo. Podpisane certifikate smo brezplačno pridobili s pomočjo storitve Let's Encrypt [22].

Ob zagonu mikroračunalnika je potrebno najprej inicializirati izhodne GPIO priključke. To naredimo s sledečo 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 le ti privzeto vključijo jih moramo še izključiti, saj bi se v nasprotnem primeru po elektrikičnem izpadu luči prižgale. 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
```

To skripto poganja 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 v stopinjah celzija se nahaja 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, da mobilno napravo odklenemo. 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

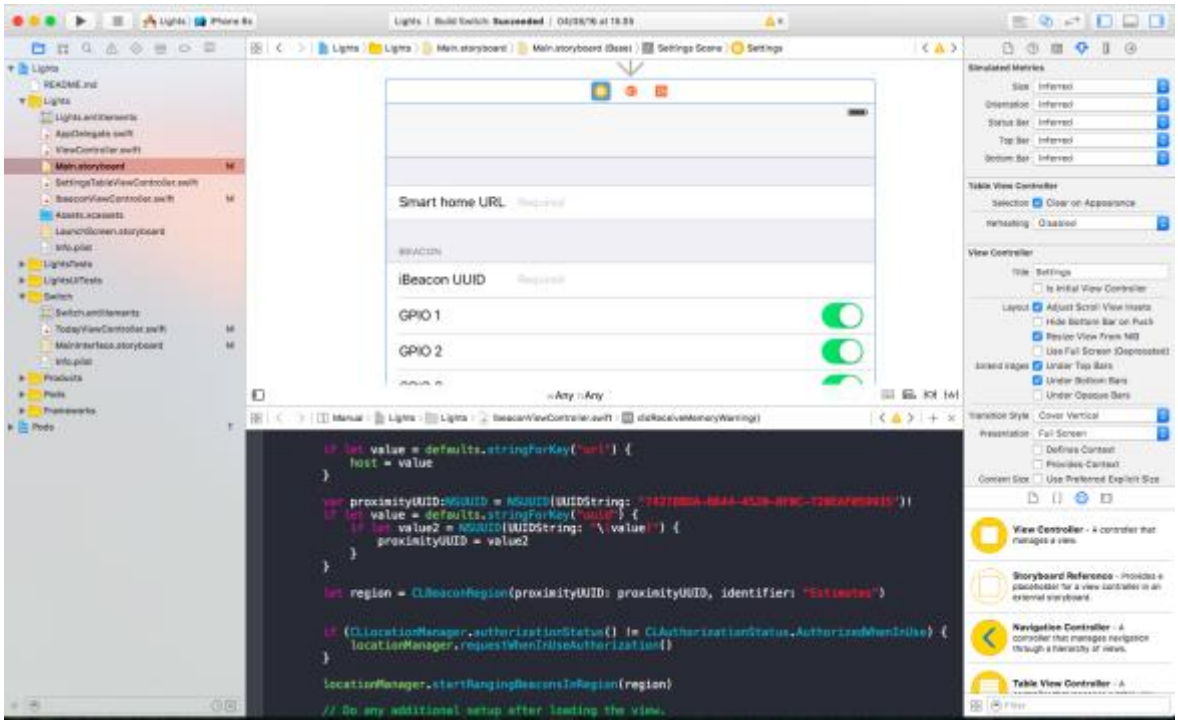
Nadzor različic je sistem, ki zapisuje spremembe v datoteko ali skupek datotek tekom časa, da lahko kasneje prikličete določeno različico. Uporabimo ga lahko za poljubne tipe datotek, še posebej pa je primeren za nadzor verzij programske kode. Ločimo dve kategoriji sistemov za nadzor različic in sicer 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 začela razvijalska skupnost Linux, največ k razvoju ideje pa je pripomogel ustvarjalec Linux-a Linus Torvalds [26]. Od njegovega izzida 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.

4.3.2 Xcode

Xcode (slika 13) je IDE (ang. Integrated Development Environment) razvit s strani Appla. 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 pa 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. Vgrajen ima tudi orodje za ustvarjanje grafičnih vmesnikov.



Slika 13: Zaslonska slika razvojnega okolja Xcode

4.3.3 Swift

Swift je programski jezik, ki je bil predstavljen na Applovi razvijalski konferenci leta 2014 [23]. Trenutno je zadnja stabilna različica 2.2.1, na voljo pa je tudi predogled za javnost različice 3.0. Od decembra 2015 je jezik pod Apache 2.0 licenco in je na voljo na Githubu. Namenjen je razvoju aplikacij za Applove operacijske sisteme iOS, OS X watchOS in tvOS, poganjamo ga pa lahko tudi na operacijskem sistemu Linux [24].

4.3.4 Cocoa pods

Cocoa pods je program za CI (ang. Continious Integration) in integracijo v Xcode projekte. Podpira več načinov pridobivanja izvorne kode kot so git, svn, bzr, http in hg. Program se namesti s pomocjo ukaza:

```
sudo gem install cocoapods
```

S pomocjo ukaza `pod init` v direktoriju projekta ga inicializiramo. Ustvari se datoteka z imenom Podfile. V to tekstovno datoteko vpišemo URL do izvorne kode, imena knjižnic ter verzije, ki jo potrebujemo. Sledi primer datoteke pod 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'
```

Nato z ukazom `pod install` namestimo vse knjiznice vključene v datoteki PodFile v projekt. Zgenerira se nam nova projektna datoteka s končnico xcworkspace, ki ima vkljucene željene 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

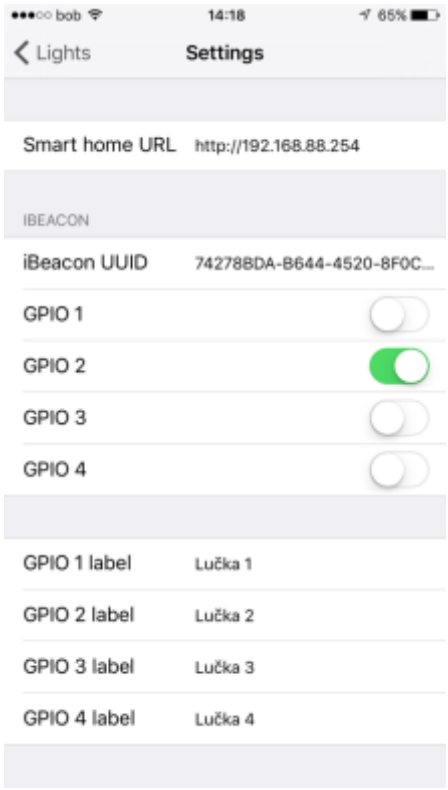
Kot je razvidno iz primerov poglavja 4.3.4 smo pri razvoju aplikacije pametne hiše uporabili programsko knjižnico Alamofire. Popularna programska knjižnica omogoča lažje in učinkovitejše delo s spletnimi storitvami. S pomočjo le te smo se povezovali na spletni strežnik na mikroračunalniku. Uporabo knjižnice v naslednjem izseku kode bomo prikazali na primeru metode za pridobitev 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 handler 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 torej poganja glavna nit, ki ima tudi najvišjo prioriteto. Vse časovno zahtevnejše operacij pa je potrebno izvajati v novo ustvarjenih nitih. V nasprotnem primeru operacijski sistem ob zamrznitvi grafičnega vmesnika za dalj časa aplikacijo zaustavi, kar pa je slabo za uporabniško izkušnjo. Delo z nitmi je vgrajeno v večino API-jev (tako operacijskega sistema kot knjižnih tretjih razvijalcev), zato je uporaba relativno preprosta.

Pri naši aplikaciji je večina klicev v lokalnem omrežju in se izvedejo relativno hitro. Vendar v primeru, ko mikroračunalnik ni dosegljiv mora vsak klic čakati na timeout. To lahko traja tudi nekaj sekund, zato je uporaba asinhronih klicev spletne storitve za dobro uporabniško izkušnjo nujna.

4.3.6 Nastavitve aplikacije in shramba NSUserDefaults



Slika 14: Zaslonska slika nastavitev aplikacije

Aplikacija ima tudi možnost nastavljanja nekaterih nastavitev (Slika 14). Te nastavitve je potrebno na nek način 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 [28]. 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 je to obvestilo avtomatsko prikazano s strani operacijskega sistema.

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 funkcionalnosti mora z uporabo vzorca delegiranje ustrezati protokolu CLLocationManagerDelegate.

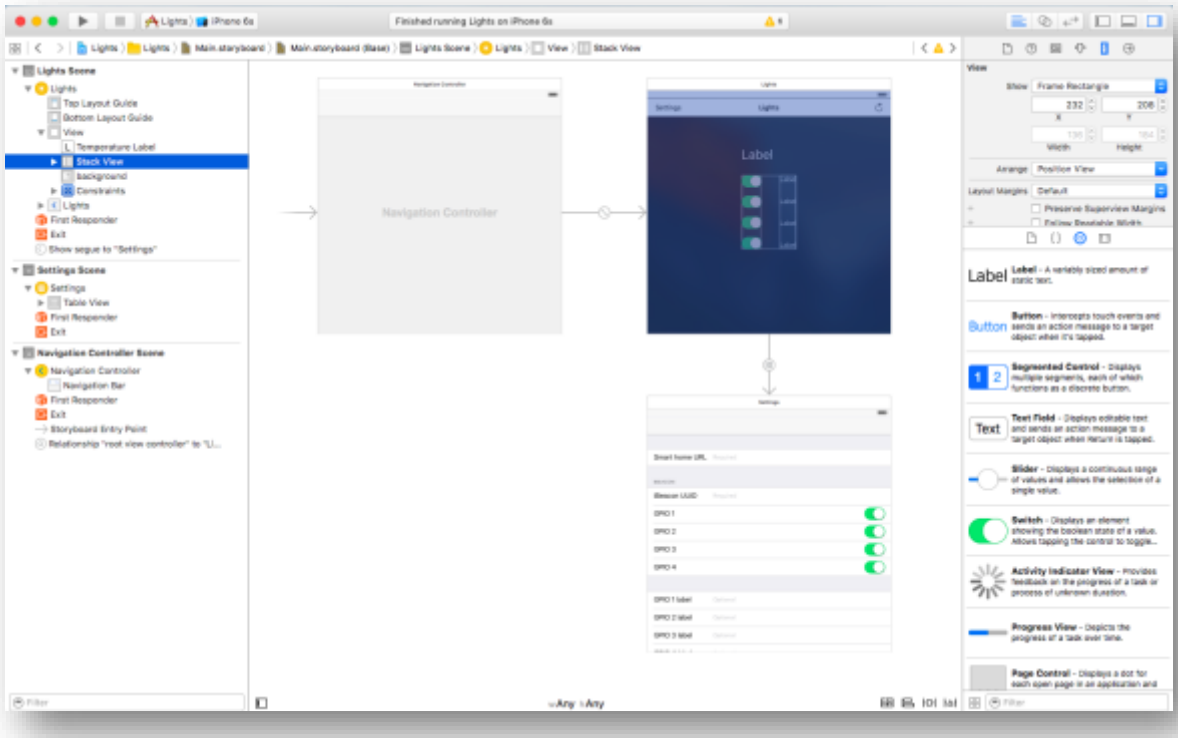
```
override func viewDidLoad() {
        let locationManager = CLLocationManager()
        locationManager.delegate = self;
        var proximityUUID:NSUUID = NSUUID(UUIDString: "74278BDA-B644-4520-8F0C-720EAF059935")!
        let region = CLBeaconRegion(proximityUUID: proximityUUID, identifier: "iBeacon")
        // 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. Gradnjo uporabniškega vmesnika za naš projekt lahko vidimo na sliki 15.

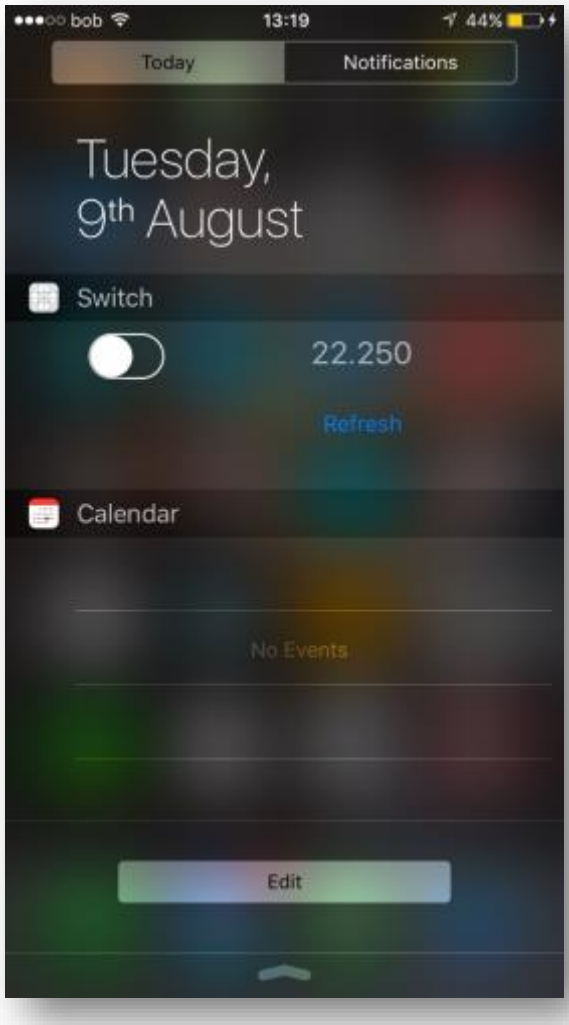


Slika 15: Gradnja grafičnega uporabniškega vmesnika v okolju Xcode

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). Ko ima uporabniški vmesnik željen izgled, ga moramo povezati s kodo. To storimo s potegom miške iz grafičnega elementa, do željene vrstice v kodi, med tem ko na tipkovnici držimo tipko ctrl. Da lahko to storimo moramo odpreti asistant editor, ki razdeli glavno okno na polovico. Na enem izberemo Storyboard, na drugema pa željen razred. Ko poteg zaključimo se nam odpre okno, kjer moramo izbrati 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.

4.3.9 Today extension

V operacijske sistemu iOS ima uporabnik vedno možnost s potegom navzdo iz vrha zaslona odpreti zaslon z obvestili in hitrimi dostopi 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. Za naš primer smo v tem metodi posodobili vrednost temperaturnega senzorja.



Slika 16: 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 x [29]). Podobno bi lahko naredili z USB

WIFI kartico, ki bi jo povezali v domače brezžično omrežje. Konfigurirali smo ga tako, da lahko dostopamo do njega lokalno ali preko spleta. V ta namen smo pripravili tudi poddomeno, ki kaže na naš zunanji IP naslov. Na mikroračunalniku smo pustili privzeto konfiguracijo dinamičnega pridobivanja IP naslova, na usmerjevalniku pa 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 naredili z naslednjim ukazom vnesenem preko ukazne vrstice usmerjevalnika [30].

```
/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 lokalen 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 pa je povezava tudi odzivnejša.

Slika 17: Usmerjevalnik Mikrotik hAP ac

5 SKLEP

Problemi zakamufliraj, TODO

IoT se še vedno srečuje z nekaterimi problemi. Največji je, da se ljudje ne zavedajo vseh dobrih lastnosti in primerov uporabe. Ljudje se zaradi filmov celo bojijo besed, kot so umetna inteligenca. Ves strah pa seveda ni zaman, saj je problem varnosti in zasebnosti

pri IoT ključnega pomena. Širok nabor naprav vključuje tudi naprave, kot so ključavnice, alarmne naprave in medicinske naprave, kot so merilci 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 nikoli zagotovo vedeti kdo jih vse lahko dostopa.

Drugi večji problem je nepoznavanje tehnologije s strani monterjev (električarjev). Le ti tipično ne znajo pravilno priključiti in konfigurirati sistema za pametno hišo. Za to je potreben razmeroma drag specialist.

Pri nalogi smo na podlagi raziskav in testiranj opravljenih za potrebe naloge smo prišli tudi do ideje za cenovno ugoden in preprost sistem za pametno hišo. Prednost je tudi, da za predlagan sisitem ne potrebujemo specializirane inštalacije. Mikroračunalnik bi bil vgrajen v obstoječo hišno elektro omarico. Povezan bi bil na električno omrežje, preko katerega bi tudi potekala 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 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 pa ne končajo pri stikalih in vtičnicah, vendar bi se lahko tretji razvijalci elektronskih naprav integrirali na ta 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 ne pozna omenjenega sistema. Sistem bi bil odporen na motilce signalov oziroma na motnje drugih brezžičnih sistemov v gosto naseljenih področjih.
TODO: idejna shema

Viri

[1] "Kaj je pametna hiša ali inteligentna hiša?" [Online]. Available: http://www.ps-promis.si/en/.

[2] V. Tomaž, "Internet stvari," *Finance*, vol. 5, 2015.

[3] "OpComm - Zakaj postaja Internet stvari največja globalna panoga?," 2013. [Online]. Available: 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. [Online]. Available: 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. [Online]. Available: https://securityintelligence.com/the-importance-of-ipv6-and-the-internet-of-things/.

[8] "IPv6 - Wikipedija, prosta enciklopedija." [Online]. Available: https://sl.wikipedia.org/wiki/IPv6.

[9] "Bluetooth - Wikipedija, prosta enciklopedija." [Online]. Available: https://sl.wikipedia.org/wiki/Bluetooth.

[10] "iBeacon." [Online]. Available: https://en.wikipedia.org/wiki/IBeacon. [Accessed: 17-Aug-2016].

[11] "Home | Nest." [Online]. Available: https://nest.com/.

[12] "Meet hue | en-XX." [Online]. Available: http://www2.meethue.com/en-xx/.

[13] "August Smart Lock | August." [Online]. Available: http://august.com/products/august-smart-lock/.

[14] "Smart Home. Intelligent Living. | SmartThings." [Online]. Available: https://www.smartthings.com/.

[15] "Samsung SmartThings Hub, 2nd Generation - - Amazon.com." [Online]. Available: https://www.amazon.com/dp/B010NZV0GE/?tag=thewire06-20&linkCode=xm2&ascsubtag=WC82592.

[16] "Source code of the open Home Automation Bus (openHAB)." [Online]. Available:

https://github.com/openhab/openhab. [Accessed: 13-Aug-2016].

[17] “Pametna mesta | FMC - Sistemski integrator.” [Online]. Available: http://fmc.si/resitve/pametna_mesta/.

[18] Berčič Boštjan, “Kako odprta je »odprta koda« | MonitorPro,” *Monitor PRO*, 2011. [Online]. Available: http://www.monitorpro.si/41846/praksa/kako-odprta-je-odprta-koda/.

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

[20] S. Lah, “Odprta koda - ne v ceni, v uporabni vrednosti je bistvo!,” *Organ. znanja*, vol. 15, no. 1–2, pp. 16–24, 2010.

[21] Henderson Gordon, “Raspberry Pi | Wiring | Gordons Projects,” 2012. [Online]. Available: https://projects.drogon.net/raspberry-pi/wiringpi/.

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

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

[24] “Swift.org - About Swift.” [Online]. Available: https://swift.org/about/#swiftorg-and-open-source.

[25] “Git - O nadzoru različic.” [Online]. Available: https://git-scm.com/book/sl/v2/Pri%C4%8Detek-O-nadzoru-razli%C4%8Dic.

[26] “Git - Kratka zgodovina Git-a.” [Online]. Available: 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] “NSUserDefaults Class Reference.” [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/.

[29] “RouterBoard.com : hAP ac,” 2016. [Online]. Available: http://routerboard.com/RB962UiGS-5HacT2HnT.

[30] “How to Port Forward in Mikrotik Router.” [Online]. Available: http://www.icafemenu.com/how-to-port-forward-in-mikrotik-router.htm.

[31] “Diymall Ibeacon Bluetooth Module 4 0 Ble Positioning Sensor Wireless | eBay.” [Online]. Available:

http://www.ebay.com/itm/181605556854?_trksid=p2057872.m2749.l2648&ssPageName=STRK%3AMEBIDX%3AIT.