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

KAZALO TABEL

Uporabljene kratice

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

Veliko je govora o pametnih napravah, s katerimi se srečujemo dnevno. Pametni telefon, pametna omrežja, pametne hiše ipd. Kategoriji teh naprav rečemo tudi IoT (Internet of Things). Pametna hiša, je hiša v kateri naj bi večino naprav upravljal enoten sistem. Bistvo vsega je povezovanje, upravljanje in nadzor nad porabniki, s ciljem zviševanja nivoja udobja, varčnosti in varnosti. Primer tega bi bil, da ko zapustimo hišo, s pritiskom na eno tipko izključimo vsa svetila in druge naprave v hiši, vključimo alarm, znižamo nivo ogrevanja ter vključimo simulacijo prisotnosti [1].

Pri IoT gre za digitalizacijo fizičnega sveta, torej naprav in stvari okoli nas. S tem pridobivamo različne podatke o teh 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 tipična za IT. Poenostavljeno povedano, internet stvari precej vsakdanje »stvari« poveže med seboj prek interneta ali namenskih omrežij. Med te povezane stvari sodijo predvsem različne naprave in aparati kot so tiskalniki v podjetjih, avtomobili, stroji in oprema v tovarnah, obcestne luči in drugo, pa tudi vrata in okna v zgradbah, klimatske naprave, ventili na radiatorjih in oprema v naših domovih, od hladilnikov in pečic do pralnih, pomivalnih in sušilnih strojev. Tudi človek je del ekosistema – denimo rekreativni športniki s pametnimi urami, ki so dejansko merilni instrumenti – in celo živali, saj napredni kmetje svojo živino s pomočjo namenskih aplikacij že spremljajo tako v hlevu kot na paši [2].

Na področju IoT obstaja več medseboj nekompatibilnih standardov. Nekateri sistemi so v celoti odprtokodni spet drugi so lastniški in zaprtokodni. Nekateri odprtokodni sistemi imajo odprtokodno tudi strojno opremo pametnih naprav. Uporabniška prednost takšnih sistemov je, da uporabnik ni vezan na enega proizvajalca strojne opreme. V nalogi se bomo posvetili odprtokodnim sistemom. Razvili bomo tudi lasten sistem za pametno hišo s pomočjo računalnika Raspberry pi in aplikacijo za Applovo mobilno platformo iOS.

V prvem poglavju si bomo pogledali obstoječe rešitve za pametne hiše s poudarkom na odprtokodnih rešitvah.

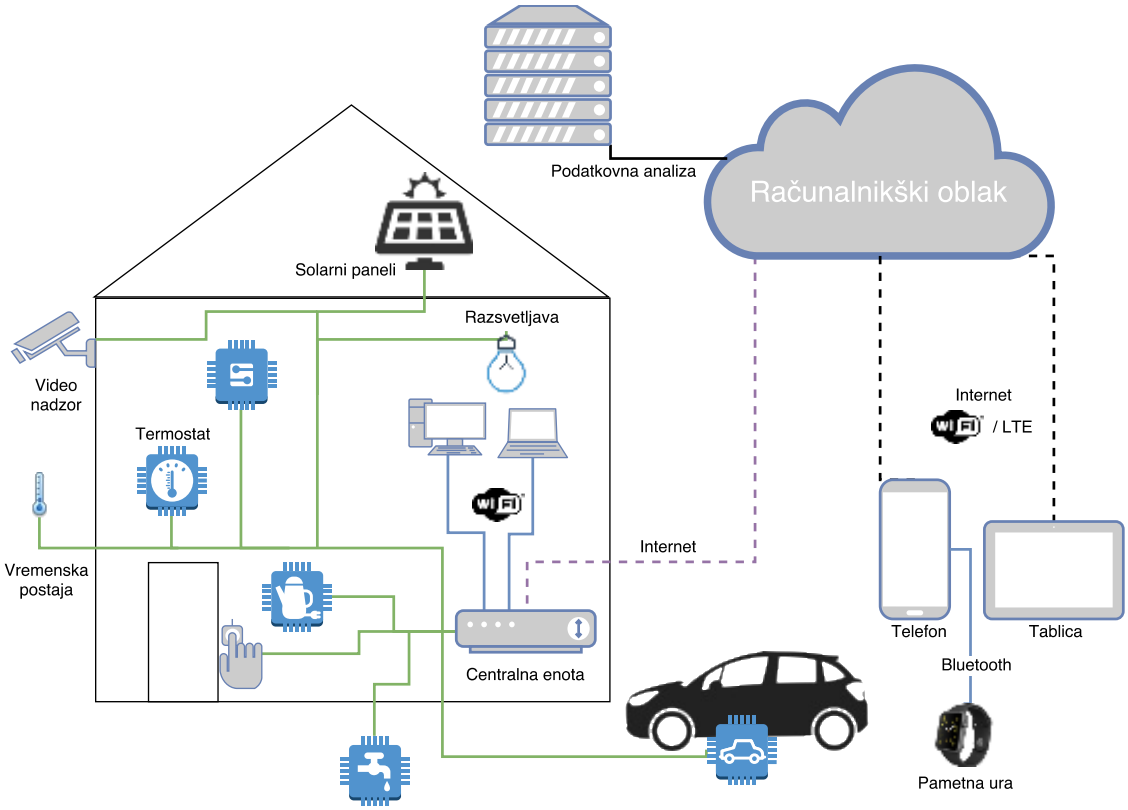
TODO opis poglavij

2 ????

2.1 Kaj je pametna hiša?

Pod pojmom pametna hiša se skriva množica pametnih naprav, ki jih kontrolira centralni sistem. Tipično se centralni sistem povezuje z računalniškim oblakom, nanj se pa prav tako povezujejo pametni telefoni in tablični računalniki. V oblaku se podatki shranjujejo in analizirajo z namenom optimiranja. Pametna hiša vključuje naprave kot so:

- Luči
- Senzor gibanja
- Kamera
- Senzor dima
- Elektronske ključavnice
- Napajalne vtičnice
- Termostat za centralno ogrevanje
- Elektronske roulete
- Senzor vlage
- Senzor svetlosti v prostoru
- Vremenska postaja
- Polnilne postaje za električna prevozna sredstva
- Kontroler solarnih panelov



Slika 1: Primer: Shema pametne hiše

2.2 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).

2.2.1 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. S pametnimi rešitvami na področju prometa posegamo predvsem na področje:

- Upravljanja in koordinacije klasičnih transportnih sredstev z inovativnimi prometnimi tehnologijami,
- urejanja mirujočega prometa,
- urejanja in optimizacije prometnih tokov.

2.2.2 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. Glavne prednosti rešitve energetskega upravljanja so:

- upravljanje s porabo energentov,
- zmanjšanje porabe energentov,
- optimizacija energetskih investicij,
- omogočanje novih storitev za ponudnike energetske oskrbe.

2.2.3 Upravljanje razsvetljave

Pametno upravljanje javne razsvetljave je pomemben steber pri uresničevanju ciljev 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 SmartCity koncepta. Glavne prednosti upravljanja razsvetljave so:

- nizka osnovna investicija za implementacijo rešitve,
- fleksibilnost postavitve,
- usmerjena v razvoj inovativnih rešitev, ki jih omogoča omrežje,
- enostavna uvedba.

2.2.4 Upravljanje odpadkov

Z vse večjo gostoto prebivalcev v mestih in potrošniško usmerjenostjo družbe postaja zbiranje, predelava in odlaganje odpadkov eno izmed ključnih meril uspešnosti delovanja mestnih uprav. Sodobne rešitve, poleg zaračunavanja storitve po dejanski količini odloženih odpadkov, omogočajo tudi optimizacijo urnikov in poti odvoza odpadkov ter spremljanje vonja v okolici zabojnikov. Glavne prednosti upravljanja z odpadki so:

- manjši stroški odvoza smeti,
- manjša onesnaženost okolice zabojnikov,
- spremljanje obnašanja potrošnikov in zaračunavanje storitve po dejanski količini odloženih odpadkov.

2.2.5 E-skupnost

Sodobne skupnosti vse bolj temeljijo na sodelovanju in neposrednem vključevanju v razvojne in družbene pobude. Državne, mestne in občinske oblasti lahko danes z uporabo naprednih tehnoloških rešitev omogočijo svojim volivcem sodelovanje pri oblikovanju skupne prihodnosti svoje skupnosti.

2.3 Odprta koda

Odprtokodne rešitve se pogosto označujejo s kratico FOSS (Free and Open Source Software). Pri tem beseda free ni oznaka za brezplačno programsko opremo. To ne pomeni, da z njo ni mogoče zaslužiti, temveč je oznaka za svobodo v uporabi programske opreme. Gre za izrecno poudarjanje svobode in pravice do uporabe izvorne kode programskih rešitev, njenega spreminjanja zaradi prilagajanja in izboljšav ter neomejenega razširjanja teh sprememb. 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 [3]. TODO: Zgodovina

2.4 IoT arhitektura

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.

Slika 2: IoT - A integracija interneta stvari

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. Nekaj problemov s katerimi se srečujemo ob integraciji IoT naprav:

- možnost posodabljanja in interoperabilnost
- zmogljivost in razširljivost
- zaupnost, varnost in zasebnost
- razpoložljivost in odpornost

2.5 Komunikacijski protokoli IoT naprav

2.5.1 IP

Večina sodobnih IoT naprav komunicira preko TCP/IP sklada. Pametna naprava je lahko v svet povezana izključno preko tega sklada ali pa je na drug način povezana z namenskim prehodom (recimo brezžično s protokolom ZigBee), ki je povezan s spletom. 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 [4].

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 [5].

2.5.2 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.

iBeacon je protokol razvit s strani Appla in je bil predstavljen na Applovi mednarodni razvijalski konferenci WWDC leta 2013 TODO:[1]. 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 [2][3]. TODO: viri iz wikipedije Tehnologija omogoča tudi navigacijo v zaprtih prostorih. V primerjavi z GPS-om ima boljšo natančnost in zagotavlja boljšo avtonomijo pametnih naprav.

Vsak beacon ima svoj unikaten identifikator UUID, major ter minor, pametne naprave pa omogočanje tudi merjenje oddaljenosti do oddajnika. [9] TODO
Za potrebe naloge smo uporabili iBeacon oddajnik, na sliki 3.

Slika 3: iBeacon oddajnik

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`.

2.5.3 RFID

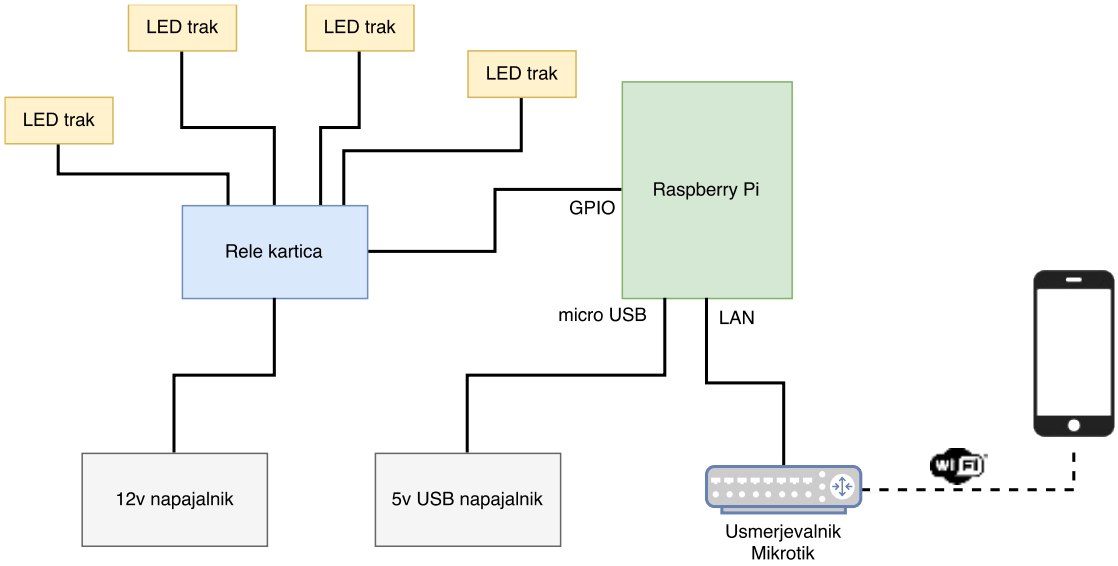
Mogoče lahko izpustim, saj ni povezan z nalogo. Vseeno treba nikje omeniti.

2.5.4 ZigBee

2.5.5 Ostali

3 RAZVOJ LASTNE REŠITVE

Shema lastno razvite rešitve je na sliki 4.



Slika 4: Shema sistema pametne hiše

3.1 Strojna oprema

3.1.1 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 5: 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.

3.1.2 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 6: Temperaturni senzor DS18B20

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

3.1.3 Aktuatorji

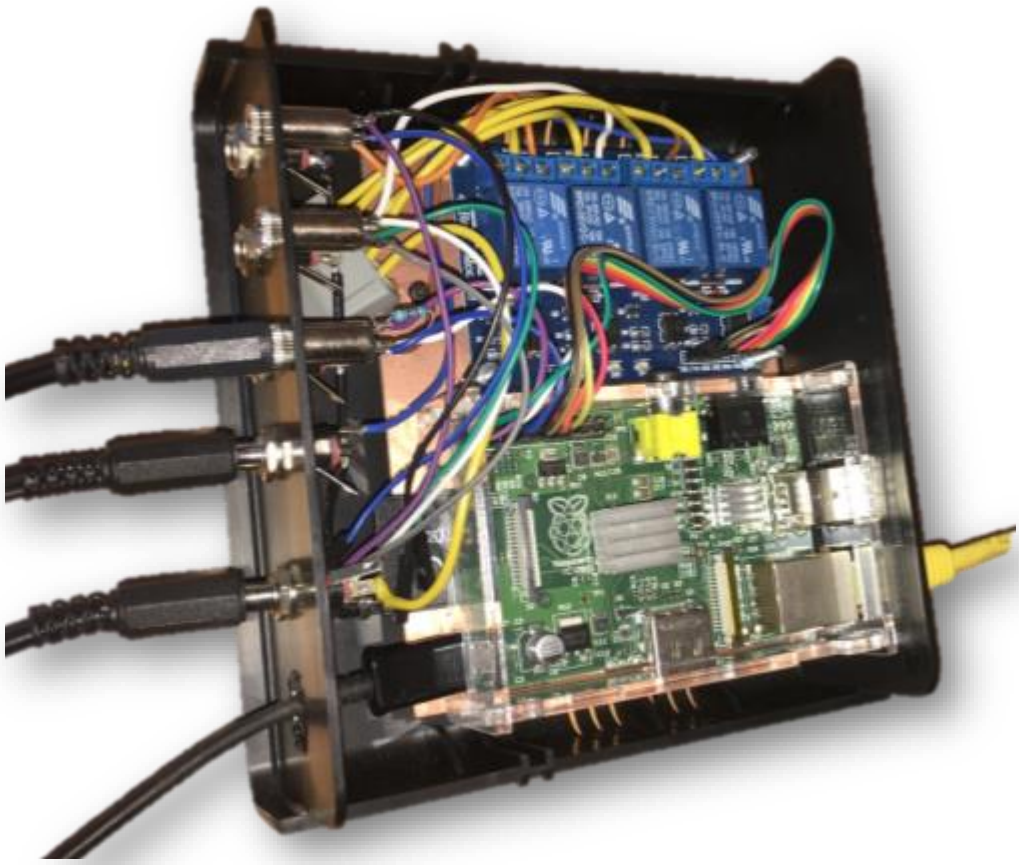
Na GPIO priključke mikro računalnika je moč priključiti različe tipe aktuatorjev. Za potrebe naloge smo izbrali kartico s štirimi releji. Ti omogočajo vklop in izklop naprav, kot so luč, črpalka, ventilator ipd. Sistem smo testirali z LED trakovi bele barve, ki delujejo na napetosti 12v. Samolepilne 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 7: Rele kartica

3.1.4 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 8: Sistem pametne hiše v ohišju

3.2 Server side

Na Raspberry pi smo namestili operacijski sistem Raspbian Jessie lite in programsko knjižnico za GPIO priključke imenovano WiringPi [6]. Za oddaljen dostop smo uporabili spletni strežnik Nginx in programski jezik PHP. Za večjo varnost smo spletni strežnik konfigurirali tako, da uporablja SSL povezavo. Tudi novejše različice operacijskega sistema iOS aplikacije, ki uporabljajo spletne storitve brez SSL certifikata prisilno zapre. Podpisane certifikate smo brezplačno pridobili s pomočjo storitve Let's Encrypt [7].

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 za izhode. Ker se pa ob inicializaciji le ti privzeto vključijo jih moramo še izključiti. V nasprotnem primeru bi se po izpadu elektrike prižgale luči. Pri tem 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 izvajamo 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 tako uporabniške račune, kot tudi pravice dostopa do posameznih vhodov in izhodov.

Do senzorja temperature lahko dostopamo podobno kot do datoteke odprte samo za branje. Za naš temperaturni senzor se je datoteka nahajala v mapi /sys/bus/w1/devices/28-0315040b1cff/w1_slave. Datoteka 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 se nahaja v drugi vrstici za enačajem, vendar ji manjka decimalna vejica. Za pravilen izpis skrbi naslednji izsek kode.

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

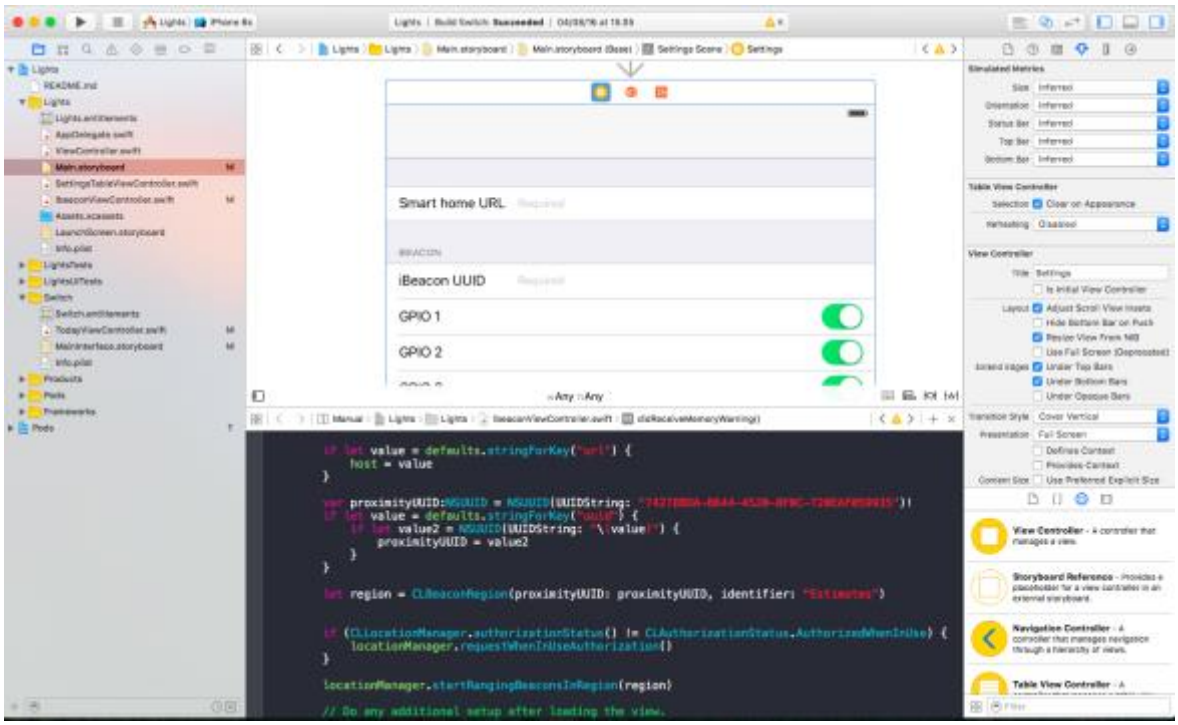
Z ukazom cat izpišemo vsebino senzorjeve datoteke.

3.3 Razvoj iOS 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 IDE (Integrated Development TODO) Xcode, ki obstaja samo za platformo Mac OS X. Razvijali smo v odprtokodnem programskem jeziku Swift. Za lažje delo z networkingom TODO smo s pomočjo CocoaPods vključili knjižnico Alamofire. Za hiter dostop 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 samodejno. Za nadzor verzij smo uporabili GIT in ponudnika github.

3.3.1 Xcode

Xcode (slika 7) je IDE 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 9: Zaslonska slika razvojnega okolja Xcode

3.3.2 Swift

Swift je programski jezik, ki je bil predstavljen na Applovi razvijalski konferenci leta 2014 [8]. Trenutno je zadnja stabilna različica 2.2.1, na voljo pa je tudi predogled za javnost različice 3.0. Od decembra 2015 je jezik odprtokodnega tipa 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 Linux platformi [9].

3.3.3 Nadzor različic GIT

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 [10].

GIT je nastal kot odgovor na spremembe pri licenciranju (postal je plačljiv) orodja za nadzor verzij BitKeeper. Razvoj je začela razvijalska skupnost Linux, največ k razvoju ideje pa je pripomogel ustvarjalec Linux-a Linus Torvalds [11]. Od njegovega izzida leta 2005 njegova popularnost raste in je trenutno najpopularnejše orodje za nadzor različic [12].

3.3.4 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 se ustvari datoteka Podfile. V to tekstovno datoteko vpišemo URL do izvorne kode, imena knjižnic ter verzije, ki jo potrebujemo. Primer:

```
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 v projekt. Zgenerira se nam nova projektna datoteka s končnico xcworkspace, ki ima vkljucene vse 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.

3.3.5 Programska knjižnica Alamofire

Kot je razvidno iz primerov poglavja 3.3.4 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

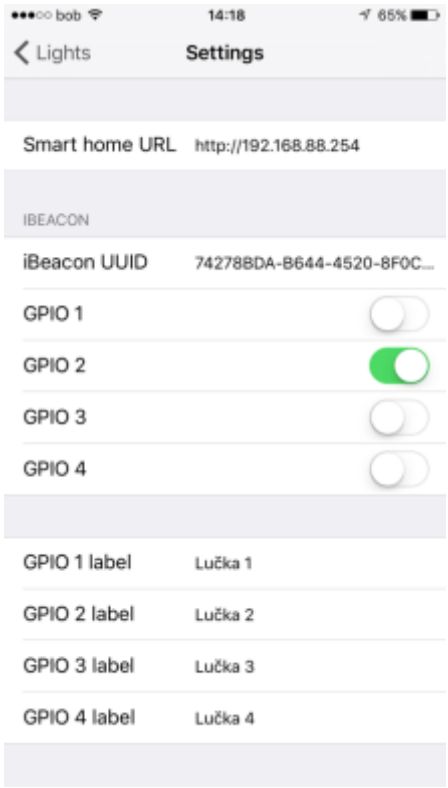
strežnik na mikroračunalniku. Uporabo knjižnice bomo prikazali na primeru metode za pridobitev vrednosti temperaturnega senzorja.

```
@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 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.

3.3.6 Nastavitve aplikacije in shramba NSUserDefaults



Slika 10: Zaslonska slika nastavitev aplikacije

Aplikacija ima tudi možnost nastavljanja nekaterih nastavitev (slika 9). Te nastavitve je potrebno na nek način hraniti na napravi, da tudi ob izhodu iz aplikacije uporabnik ne izgubi nastavitev. V ta namen smo uporabili razred NSUserDefaults. Omogoča hrambo parov ključev in vrednosti za nekatere osnovne podatkovne tipe [13]. 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
}
```

3.3.7 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 aplikacije mora uporabnik izbrati ali dovoli aplikaciji uporabo lokacijskih storitev in obvestil. Zaradi varovanja osebnih podatkov se vedenja ne da spremeniti. Proženje akcij v ozadju na podlagi lokacije je implementirano preko obvestil, zato mora uporabnik za to funkcionalnost odobriti tudi to.

V sobo smo namestili prej opisan iBeacon oddajnik. Uporabnik mora pred uporabo v nastavitve aplikacije vnesti njegov unikaten identifikator UUID in 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.

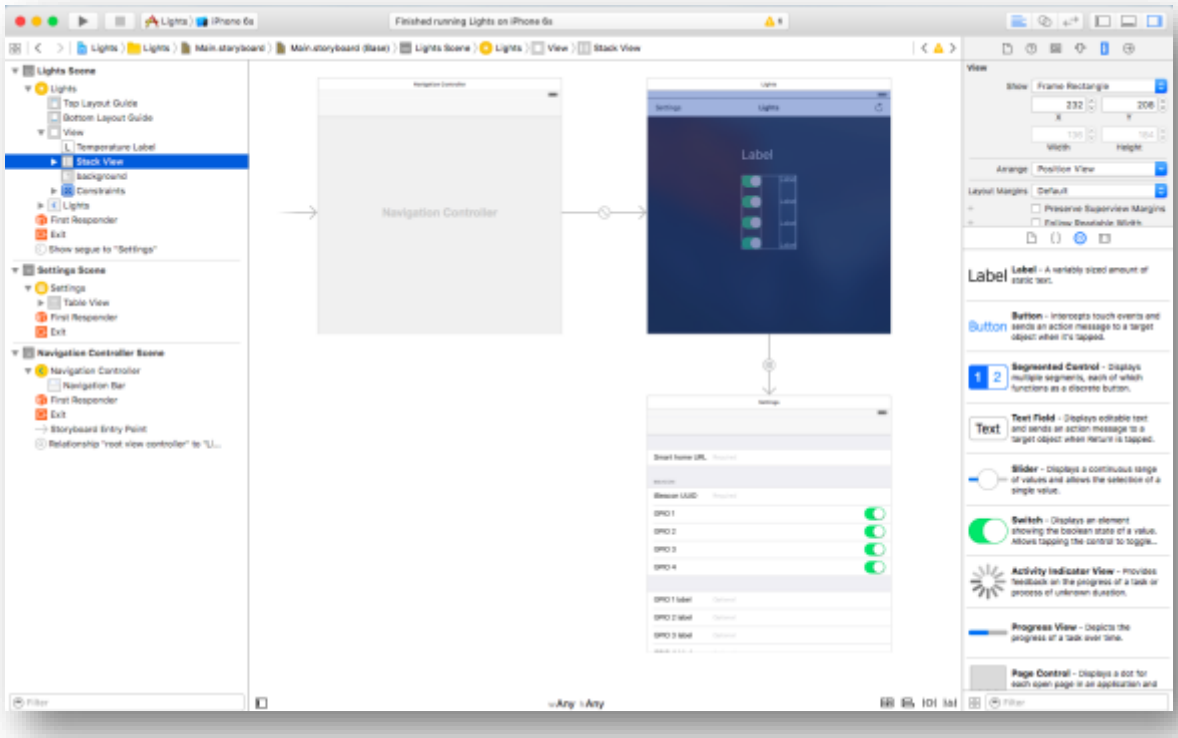
```
override func viewDidLoad() {
        let locationManager = CLLocationManager()
        locationManager.delegate = self;
        var proximityUUID:NSUUID = NSUUID(UUIDString: "74278BDA-B644-4520-8F0C-720EAF059935")!
        let region = CLBeaconRegion(proximityUUID: proximityUUID, identifier: "iBeacon")
        // 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 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.

3.3.8 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 10.

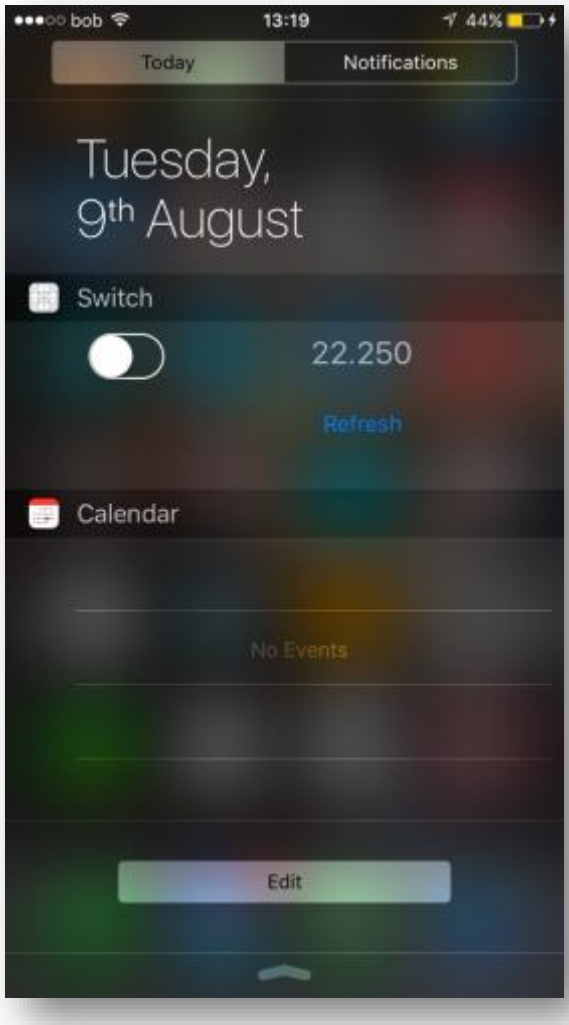


Slika 11: Gradnja grafičnega uporaniškega vmesnika

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.

3.3.9 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 12: Zaslonska slika hitrega dostopa Today extension

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 [14]). 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 [15].

```
/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 13: Usmerjevalnik Mikrotik hAP ac

5 TEŽAVE V IOT

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 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 pomembnejši 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.

6 SKLEP

Na podlagi raziskav in testiranj opravljenih za potrebe naloge smo prišli 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 hišno elektro omarico z varovalkami. Povezan bi bil zgolj na električno omrežje, preko katerega bi tudi potekala komunikacija. Trenutno so na tržišču že naprave, ki omogočajo deljenje internetne povezave preko omrežja.

Viri

[1] "VSE JE PAMETNO, TUDI HIŠE..."

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

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

[4] 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/.

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

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

[7] 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.

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

[9] "Swift.org - About Swift." [Online]. Available: https://swift.org/about/#swiftorg-and-open-source.

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

[11] "Git - Kratka zgodovina Git-a." [Online]. Available: https://git-scm.com/book/sl/v2/Pri%C4%8Detek-Kratka-zgodovina-Git-a.

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

[13] "NSUserDefaults Class Reference." [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/.

[14] "RouterBoard.com : hAP ac," 2016. [Online]. Available: http://routerboard.com/RB962UiGS-5HacT2HnT.

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