Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Podgoršek

Integracija multi-senzorskih naprav v okviru pametnega doma

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomskega dela preučite produkte in rešitve, ki so na voljo za podporo pametnega doma. Poskusite najti multi-senzorske naprave, ki so odprte z vidika integracije s poljubnimi drugimi napravami in s pomočjo drugih odprto-kodnih rešitev sestavite arhitekturo rešitve za podporo različnih scenarijev pametnega doma.

Zahvaljujem se svoji družini, mentorju za vodenje in koristne napotke pri pisanju diplomskega dela ter podjetju L-TEK elektronika d.o.o. za predstavitev podjetja, razvoja in proizvodnje tiskanih vezij.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IoT	Internet of Things	Internet stvari
MQTT	Message Queuing Telemetry Transport	Protokol za prenos telemetrijskih sporočil
CoAP	Constrained Application Protocol	Protokol za prenos dokumentov
URL	Uniform Resource Locator	Enolični krajevnik vira
REST	Representational state transfer	Način medsebojne komunikacije med računalniškimi sistemi
API	Application programming interface	Aplikacijski programski vmesnik
BLE	Bluetooth Low Energy	Nizko potratni Bluetooth
OTA	Over-the-air programming	Brezžičen prenos programske kode
AWS	Amazon Web Services	Amazonove spletne storitve
HTTP	Hypertext Transfer Protocol	Protokol za prenos hipertekста
HTTPS	Hyper Text Transfer Protocol Secure	Protokol za varen prenos hiperteksta
UDP	User Datagram Protocol	Nepovezavni protokol transportnega sloja v protokolnem skladu TCP/IP
IR	Infrared	Infrardeče
IP	Internet Protocol	Internetni protokol
IPv6	Internet Protocol version 6	Internetni protokol verzije 6
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks	IPv6 prek nizko energijskega brezžičnega osebnega omrežja

kratica	angleško	slovensko
DOM	Document Object Model	Dokumentni objektni model
JSON	JavaScript Object Notation	Notacija JavaScript objekta
DBaaS	Database-as-a-Service	Podatkovna baza kot storitev
UTC	Coordinated Universal Time	Univerzalni koordinirani čas
CLI	Command-line interface	Vmesnik programske vrstice
QoS	Quality of Service	Kakovost storitve

Povzetek

Naslov: Integracija multi-senzorskih naprav v okviru pametnega doma

Avtor: Luka Podgoršek

Internet stvari predstavlja pomemben element četrte industrijske revolucije. Na voljo so številni senzorji, aktuatorji in druge naprave, ki so posredno ali neposredno dostopne preko interneta in so zmožne pošiljanja, prejemanja in procesiranja podatkov. Realizacija različnih scenarijev s področja pametnih hiš, mest, tovarn itn. še nikoli ni bila tako izvedljiva, kot je danes. Vseeno pa se pri implementaciji takšnih scenarijev soočamo z izzivi. Eden teh je interoperabilnost oziroma nezmožnost medsebojne komunikacije med „stvarmi interneta“. Uporabljajo se namreč različni komunikacijski protokoli in standardi, zato naprave, ki jih kupimo, niso vedno povezljive med seboj. V tej diplomski nalogi sem se posvetil uporabi enostavnih multi-senzorjev, ki zmorejo meriti več podatkov iz okolja. V nalogi pokažem, kako lahko z relativno preprosto arhitekturo razvijemo podporo za enostavne scenarije s področja pametnih hiš.

Ključne besede: IoT, senzor, multi-senzor, Firefly, Bluemix, MQTT, OpenHab.

Abstract

Title: Integration of multisensor devices for smarthome scenarios

Author: Luka Podgoršek

Internet of things is an important part of fourth industrial revolution. There are many different sensors, actuators and other devices which can be accessed through internet and are able to send, receive and process data. Today we can realize many different IoT scenarios on the field of smart houses, cities, factories, etc. but we face with different challenges when implementing such systems. One of these challenges is connecting different things of internet of things between each other. Devices use different communication protocols and standards therefore devices that we buy can't connect and communicate with each other. In this thesis, I've used simple multisensors FireFly that can measure different environment states. I demonstrate how we can use relatively simple architecture to support different scenarios in the field of smart houses.

Keywords: IoT, sensor, multisensor, Firefly, Bluemix, MQTT, OpenHab.

Poglavje 1

Uvod

Dandanes je veliko govora o četrti industrijski revoluciji, ki digitalizira industrijo. Internet stvari (v nadaljevanju IoT) pri tem igra ključno vlogo, saj predstavlja povezavo med fizičnim in digitalnim svetom podatkov. Na trgu je na voljo veliko različnih senzorskih naprav, aktuatorjev, platform in drugih rešitev, ki omogočajo realizacijo različnih IoT scenarijev.

Na področju pametnih domov obstajajo platforme, ki dobro delujejo z družinami produktov. Primeri takšnih platform so Apple HomeHit, Amazon Echo, Philips Hue in ostali. Takšne platforme omogočajo uporabniku avtomatizacijo določenih delov pametnega doma, kot je nadzor razsvetljave, senčil na oknih, klimatskih naprav in še kaj. Vse te platforme podpirajo določeno število naprav, s katerimi lahko realiziramo različne scenarije. Vendar se uporabnik hitro sooči s problemom, kaj storiti v primeru, ko določena naprava ni podprta s strani platforme. Takšne naprave ne more uporabiti, zato je neuporabna, dokler ne kupi dodatne platforme, ki podpira takšno napravo.

Tukaj se uporabnik sreča z dodatnim problemom pametnih domov. Vse naprave za pametne domove so občutno dražje kot naprave, ki jih uporabljamo danes v naših domovih. Pri nakupu pametne žarnice, bomo odšteli med 11 in 50 EUR za posamezno žarnico. Povprečni lastnik pametnega doma porabi okrog 1000 EUR za nakup naprav [24]. Na podlagi tega vidimo, da

sprememba standardnega doma v pametni dom, predstavlja dokaj visoko investicijo.

Sami se nismo želeli omejiti na takšne platforme in družine produktov, zato smo poiskali cenovno dostopen nizkonivojski multi-senzor, s katerim bi lahko zajeli čim več koristnih podatkov za potrebe pametnega doma. Z uporabo izbranih multi-senzorjev smo pokazali, kako lahko takšen sistem programsko podpremo.

Kot IoT naprave smo izbrali multi-senzorje Firefly[1], ki jih izdeluje slovensko podjetje L-TEK elektronika d.o.o.[2]. Multi-senzor Firefly vsebuje žiroskop, magnetometer, pospeškometer, svetlobni senzor, senzor za relativno vlago in temperaturo. Te smo razporedili doma in jih preko prehodnega vmesnika, ki teče na Raspberry Pi 3, povezali z IBM Watson IoT Platform v oblaku Bluemix. Razvili smo spletno aplikacijo, katera uporabniku omogoča vizualizacijo zajetih podatkov in možnost spremljanja stanja domačega okolja.

[1] https://firefly-iot.com/
[2] http://www.l-tek.si/

Poglavje 2

Pregled področja pametni dom

Pred zasnovo arhitekture sistema smo pregledali obstoječe družine produktov in platforme, multi-senzorske naprave in komunikacijske protokole uporabljene na področju interneta stvari. Začeli smo s pregledom obstoječih rešitev, nadaljevali pa s pregledom multi-senzorskih naprav. Na koncu poglavja smo opisali pogosto uporabljene komunikacijske protokole ter izbrali multi-senzorske naprave in tehnologije za našo rešitev.

2.1 Družine produktov

Pregledali smo obstoječe rešitve in družine produktov na področju pametnega doma. Te rešitve uporabniku omogočajo postavitev različnih naprav in sestavljanje različnih scenarijev pametnega doma. Vendar komercialna uporaba in dokaj enostavna namestitev teh sistemov terja svoj davek. Uporabnik občuti to kot zaprtost sistemov in visoko ceno uporabljenih rešitev. Opisali smo osnovno funkcionalnost, nadzor platform oziroma sistema in ceno.

2.1.1 Philips Hue

Philips Hue je sistem za nadzor hišne razsvetljave. Glavni del sistema predstavlja Philips Hue bridge. To je centralna naprava, ki nadzira delovanje svetil, stikal za prižig svetil in senzorjev gibanja. Nanj lahko povežemo do 50

različnih svetil, stikal in senzorjev. Celotni sistem nadziramo z uporabo mobilne aplikacije. Nadziramo lahko svetilnost in barvnost svetil, ter prižigamo in ugašamo svetila [35, 36]. Vse Philips Hue naprave med seboj komunicirajo s protokolom Zigbee. Cena posamezni naprav se razlikuje. Za Philips Hue bridge mora uporabnik odšteti okrog 50 EUR. Cena bele žarnice je okrog 30 EUR, barvne pa okrog 70 EUR. Philips ponuja tudi zagonske pakete, ki vsebujejo Philips Hue bridge in dve beli žarnici za 70 EUR.

2.1.2 Amazon Echo

Amazon Echo je pametni zvočnik, ki ga je razvilo podjetje Amazon. Danes obstaja že nova različica produkta imenovana Amazon Echo Dot. Ta ne vsebuje zvočnikov, zato jo je možno povezati s samostojnimi zvočniki. Skupina produktov Amazon Echo omogoča izvajanje glasovnih ukazov. Uporabnik aktivira zvočnik z besedo „Alexa“, nato pa izreče glasovni ukaz. Amazon Echo je povezan z Amazon-ovim servisom imenovan Alexa Voice Service in različnimi servisi za predvajanje glasbe, kot so Spotify, TuneIn. Poleg tega omogoča branje novic, vremenske napovedi, povezavo z Philips Hue in nekaterimi ostalimi napravami. Za konfiguracijo sistema uporabnik potrebuje mobilno aplikacijo in Amazonov račun. Cena Amazon Echo je okrog 180 EUR, cenejšo različico Echo Dot pa lahko kupimo po ceni 50 EUR [3, 4].

2.1.3 Google Home

Google Home je pametni zvočnik, katerega poganja Google Assistant. Google Assistant je virtualna oseba zmožna pogovora z drugo osebo z določenimi omejitvami. Oba produkta je razvilo podjetje Google. Uporabnik nadzoruje delovanje zvočnika z uporabo glasovnih ukazov. Vsi glasovni ukazi se začnejo z besedama „Ok Google“. Uporabnik lahko povpraša Google Home o svojem dnevu, zvočnik pa bo uporabniku povedal, kaj ima v koledarju. Uporabimo ga lahko kot spletni iskalnik Google, zvočnik pa nam bo zadetke povedal. Poleg tega podpira prostoročno telefonijo, vendar je zaenkrat ome-

jena na Združene države Amerike in Kanado. Google Home podpira številne ostale aplikacije in naprave za pametne domove. Z uporabo glasovnih ukazov lahko na televiziji predvajamo video vsebino, nadziramo razsvetljavo, predvajamo glasbo, dodamo dogodke v koledar in še marsikaj. Seveda ne moremo nadzirati naprav, ki niso podprte s strani Google Home. Zaradi uporabe glasovnih ukazov ne potrebujemo dodatne aplikacije za nadzor sistema. Cena pametnega zvočnika se giblje okrog 110 EUR, kar je ceneje kot Amazon Echo [23, 21].

2.1.4 Apple HomeKit

Apple HomeKit je družina produktov za pametni dom. Za postavitev sistema potrebujemo Apple TV. Apple TV deluje kot centralna naprava, ki skrbi za nadzor celotnega sistema. Z Apple TV lahko povežemo ostale produkte HomeKit. Podobno kot Amazon Echo lahko nadziramo delovanje HomeKit-a z uporabo glasovnih ukazov in Siri. Siri je servis, ki teče na napravah z operacijskim sistemom IoS. Zmožen je interpretirati človeški govor in razbrati ukaze. Tako lahko uporabnik z glasovnimi ukazi krmili delovanje sistema HomeKit. Poleg glasovnih ukazov ima uporabnik na voljo mobilno aplikacijo s katero lahko nadzoruje celoten sistem. Apple HomeKit se lahko poveže z različnimi napravami za nadzor razsvetljave, kot je na primer Philips Hue, nadzira določene pametne vtičnice, termostate, hladilne naprave in senzorje. Trenutno je na voljo omejeno število različnih naprav, vendar se število podprtih naprav povečuje. Za uporabo Apple HomeKit-a mora uporabnik kupiti Apple TV, za kar bo odštel med 180 in 230 EUR odvisno od pomnilnih kapacitet. Če uporabnik ne želi upravljati svojega pametnega doma pred televizijskim zaslonom, potrebuje mobilno napravo z operacijskim sistemom IoS, katera ni ravno poceni. Tako je Apple HomeHit eden dražjih sistemov za nadzor pametnega doma [8, 9].

2.1.5 OpenHab2

OpenHab2 (angl. Open Home Automation Bus) je odprtokodna platforma, ki omogoča uporabniku priklop različnih naprav v enovit sistem. Zgrajena je nad ogrodjem Eclipse SmartHome [34, 17]. OpenHab2 ima veliko skupnost razvijalcev, ki razvijajo vtičnike, s katerimi lahko priključimo naprave različnih proizvajalcev v OpenHab2 platformo. Zaradi tega ima veliko število podprtih naprav. Tako kot Apple HomeKit in Amazon Echo lahko z platformo OpenHab2 povežemo Philips Hue. Uporaba platforme OpenHab2 je popolnoma brezplačna. Na voljo je tudi mobilna aplikacija za operacijske sisteme Android in IoS. Izmed ostalih platform se razlikuje po tem, da je potrebno precej konfiguracije za vzpostavitev sistema in povezavo z ostalimi napravami. Del konfiguracije lahko opravimo preko spletne aplikacije. Določene specifične nastavitve pa moramo opraviti ročno z urejanjem datotek. Preden lahko uporabimo platformo, jo moramo namestiti. Podprti so operacijski sistemi Windows, Mac OS X in določene distribucije Linuxa. Uporaba platforme je brezplačna [34].

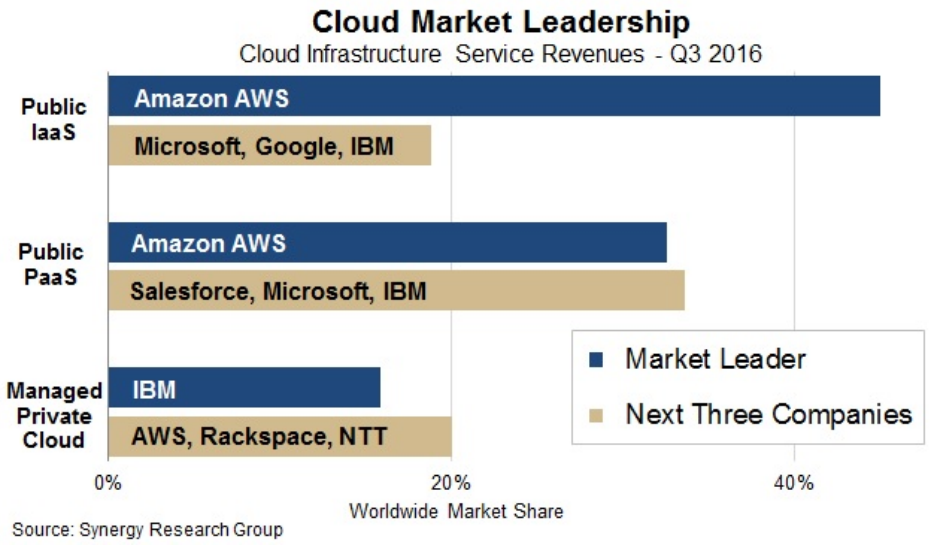
2.2 Oblačne platforme

Veliko rešitev na področju interneta stvari uporablja oblak. Ker smo želeli razviti svoj sistem, smo pregledali obstoječe ponudnike oblačnih storitev na področju interneta stvari.
Pregledali smo naslednje oblačne platforme:

- Amazon Web Services [5],
- Google Cloud Platform [22],
- ARM mbed [10],
- IBM Bluemix [12].

2.2.1 Amazon Web Services

Amazon Web Services je oblačna platforma podjetja Amazon in ima največji tržni delež (slika 2.1).



Slika 2.1: Tržni delež oblačnih platform [11]

Za dostop do AWS storitev se moramo registrirati. Po uspešni registraciji lahko 12 mesecev brezplačno uporabljamo storitve v določenih omejitvah. AWS IoT platforma nam omogoča 250.000 poslanih sporočil brezplačno, vsakih naslednjih 5 milijonov sporočil pa stane 5 $.

Na platformi lahko uporabimo protokol MQTT, MQTT z uporabo WebSocket-a in HTTP. Za pošiljanje in prejemanje sporočil skrbi posrednik sporočil (angl. message broker). Omogoča nam določanje pravil, s katerimi lahko nastavimo določene akcije glede na podatke, ki jih pošiljajo stvari interneta stvari. Zaradi varnosti moramo na sami platformi definirati naprave, ki jih bomo uporabljali in njihov tip.

2.2.2 Google Cloud Platform

Google Cloud Platform je oblačna platforma podjetja Google. Ob registraciji uporabnik dobi brezplačni račun za 12 mesecev in 300 $ denarja, katerega lahko porabi na oblačni platformi. Brezplačni račun je aktiven, dokler ne porabimo dobljenega denarja, vendar največ 12 mesecev.

Za komunikacijo skrbi posrednik sporočil, ki nam omogoča 120.000 sporočil na minuto za 25 $ mesečno. Kot posrednika sporočil uporabljajo RabbitMQ, ki je odprtokodna rešitev. Komunikacija med napravami poteka z uporabo protokola MQTT. Zaradi varnosti moramo za vsako napravo registrirati in definirati njen tip. Preden se naprava lahko poveže, se mora avtenticirati z uporabniškim imenom in geslom [22].

2.2.3 ARM mbed

Oblačno platformo mbed je razvilo podjetje ARM. Platforma je dostopa partnerjem, kar pomeni, da moramo za dostop poslati prošnjo. Od ostalih oblačnih platform se razlikuje v načinu komunikacije in povezavi naprav z oblakom.

Naprava, ki želi komunicirati z oblakom, potrebuje operacijski sistem mbed ali pa mbed odjemalca. Ta skrbita za varno komunikacijo z oblakom z uporabo protokola CoAP. Vse nastavitve oblaka lahko nastavimo s pomočjo spletne platforme. Poleg tega omogoča posodabljanje programske opreme na samih napravah. Servise, ki jih uporabimo v oblaku lahko povežemo z zunanjimi servisi z uporabo programskega vmesnika REST. Tako lahko na primer povežemo oblak mbed z Bluemix-om [10].

2.2.4 IBM Bluemix

IBM Bluemix je oblačna platforma podjetja IBM. Ob registraciji dobimo brezplačni račun aktiven 30 dni. V preizkusnem obdobju lahko uporabimo do 10 oblačnih servisov ter ne smemo prekoračiti uporabe prostora in procesorskih ur. Po izteku preizkusnega obdobja lahko uporabljamo večino oblačnih

storitev brezplačno, vendar moramo izbrati način plačila. V primeru prekoračitve brezplačne porabe, dobimo mesečni račun za porabo storitev.

Na voljo imamo več kot 130 različnih oblačnih storitev, katere lahko med sabo povežemo. Bluemix se najbolj razlikuje od ostalih oblačnih platform, saj ima na voljo Watson-a. Watson je ime računalnika, ki je leta 2011 sodeloval na televizijski oddaji Jeopardy. Sposoben je bil interpretirati vprašanja in odgovoriti nanje. Danes uporabljajo to tehnologijo za medicinske diagnoze. Na oblačni platformi Bluemix ime Watson predstavlja skupino storitev, ki izhaja iz Watson računalnika. Z uporabo teh storitev lahko pretvorimo govor v tekst in obratno. Omogoča prepoznavanje slik, obrazov in objektov, analizo psihološke lastnosti osebe z uporabno družabnih omrežij in ostalih podatkov.

Bluemix ima na voljo IoT platformo. Podobno kot ostale oblačne platforme ima Bluemix posrednika sporočil, ki uporablja protokol MQTT. V brezplačnem načinu lahko mesečno pošljemo do 200 MB sporočil in registriramo do 500 različnih naprav. Ob prekoračitvi omejitev nam uporabo zaračunajo na mesečni ravni. Zaradi varnosti moramo naprave, ki jih želimo povezati v Bluemix registrirati. Vsaki napravi moramo dodeliti tip, enolični identifikator in geslo. Vsaka naprava, ki se želi povezati, se mora avtenticirati.

2.3 Multi-senzorske naprave

Senzor je električna komponenta ali modul, ki je sposobna zaznavati spremembe v fizičnem okolju, kjer se nahaja in pošiljati podatke o spremembi drugim elektronskim elementom [37]. Najbolj splošna delitev senzorjev je delitev v analogne in digitalne senzorje. Delimo jih tudi glede na tip senzorja.

Multi-senzor je naprava, ki sestoji iz več različnih senzorjev. Takšne naprave so uporabne na področju interneta stvari, saj lahko spremljajo več različnih stanj.

2.3.1 SensorTag

SensorTag je multi-senzor podjetja Texas Instruments. Proizvajajo tri različice multi-senzorja, ki se razlikujejo v načinu komunikacije. Prva podpira Bluetooth low energy (BLE), 6LoWPAN in ZigBee, druga podpira BLE in Sub-1 GHz long range wireless, tretja pa nizko potraten WiFi [40].
Vsebuje senzorje za [38]:

- IR senzor za merjenje temperature,
- svetilnost,
- tlak,
- relativno vlažnost,
- magnetometer,
- pospeškometer,
- žiroskop,
- kompas.

Celotni multi-senzor napaja baterija CR2032. Podpira nadgradnjo programske opreme preko brezžične komunikacije (angl. Over The Air download, OTA). To je zelo priročna funkcionalnost, saj uporabnik ne potrebuje vsakega modula povezati z računalnikom v primeru nadgradnje programske opreme. Vse to lahko stori preko brezžične komunikacije, kar pride še posebej prav pri velikem številu naprav.

Baterija lahko napaja multi-senzor med 48 in 240 urami [42], odvisno od števila prižganih senzorjev. V primeru uporabe vseh senzorjev, moramo vsake 2 dni zamenjati baterijo. To predstavlja problem, saj mora uporabnik neprestano menjati baterijo, kar predstavlja dodaten strošek in nepraktično uporabo. Uporabni so za testne aplikacije, testiranje različnih prototipov in testiranje IoT scenarijev. Cena različic se giblje med 25 in 35 EUR.

2.3.2 ARM® mbed™-enabled FRDM-K64F

ARM® mbed™-enabled FRDM-K64F je multi-senzor namenjen enostavni povezavi z IBM-ovim oblakom. Poganja ga 32 bitni procesor ARM® Cortex™-M4 Core in ima 256KB delovnega spomina. Za povezavo z internetom skrbi priključek ethernet, napaja pa ga USB priključek.
Multi-senzor vsebuje:

- majhen LCD monitor,
- zvočnik,
- dva potenciometra,
- temperaturni senzor,
- pospeškometer,
- igralno palico (angl. joystick).

Multi-senzor razvijalcu omogoča povezavo z IBM-ovim oblakom Bluemix in uporabo platforme ARM mbed. Namenjen je predvsem za razvijalce, ki razvijajo prototipe na področju interneta stvari. ARM® mbed™-enabled FRDM-K64F lahko kupimo v začetnem paketu, katerega cena je 55 EUR [18, 28, 27].

2.3.3 FireFly

FireFly je multi-senzor, katerega proizvaja slovensko podjetje L-TEK Elektronika [29]. Podjetje smo tudi obiskali, kjer so nam predstavili razvoj tiskanih vezij, proizvodnjo in testiranje proizvedenih tiskanih vezij.

Multi-senzor poganja 32 bitni ARM Cortex -M0 procesor. Ima 16 kB delovnega spomina in nizko potratni Bluetooth modul (angl. Bluetooth Low Energy, BLE) [19]. Za napajanje skrbi baterija CR2032 ali pa priključek micro USB.
Vsebuje senzorje za:

- temperaturo,
- svetlost,
- relativno vlažnost,
- pospeškometer,
- žiroskop,
- magnetometer.

Senzorji v tiskanem vezju so povezani z **I2C** vodilom, preko katerega poteka vsa komunikacija z napravami v modulu. Na tiskanem vezju je prost I2C priključek, kar omogoča dodajanje poljubnega zunanjega senzorja ali pa krmiljenje zunanjih naprav. To pomeni, da lahko uporabimo multi-senzor FireFly kot krmilnik.

V podjetju so definirali tudi svoj komunikacijski protokol z multi-senzorjem Firefly [20]. Vsa komunikacija poteka z uporabo nizko potratnega Bluetooth-a. FireFly ima 2 načina delovanja. V standardnem načinu delovanja moramo poslati ukaz, FireFly pa nam odgovori s podatki. V drugem načinu FireFly pošilja podatke v določenem časovnem intervalu. Način delovanja lahko spreminjamo s pošiljanjem ukazov. Prav tako lahko nastavimo hitrost pošiljanja podatkov, prižigamo in ugašamo senzorje.

FireFly pošlje zaporedje znakov. Prejete znake je potrebno razčleniti in izračunati dejanske vrednosti. Na primer, če želimo prebrati podatek o temperaturi moramo prejeto vrednost pretvoriti. Za izračun temperature v stopinjah Celzija moramo uporabiti formulo 2.1.

$$Temp = (175.72 * vrednostSenzorja)/65536 \tag{2.1}$$

Podobno moramo pretvoriti vrednosti ostalih senzorjev.

Podjetje L-TEK elektronika

Podjetje L-TEK elektronika d.o.o. (v nadaljevanju L-TEK) locirano v Šentjerneju in ima zaposlenih 60 ljudi. Je slovensko podjetje specializirano za izdelavo prototipov, majhnih serij in izdelavo srednje velike serije visoko-tehnoloških izdelkov [29]. Razvili in izdelali so multi-senzor FireFly.

Razvoj in proizvodnja

Ob ogledu so nam predstavili razvojno ekipo ter potek razvoja. Razvoj vezja se začne z načrtovanjem vezja. Za ta namen uporabljajo programsko orodje, kjer izrišejo načrt vezja. Načrt je sestavljen iz več plasti, ki se jih na koncu združi v tridimenzionalni model vezja. Izdelani načrt je možno uporabiti za načrt po katerem stroji namestijo komponente na tiskano vezje. Pri načrtovanju morajo paziti na število plasti, število navpičnih vodil, kakšna vodila bodo uporabili, napajanje, kako poteka komunikacija s posameznimi komponentami na tiskanem vezju, itd. Ko končajo načrt, naročijo tiskano vezje izdelano po načrtu in ga preizkusijo. V prvi iteraciji izdelave prototipa redko deluje vse po pričakovanjih, zato ob testiranju identificirajo probleme in z novim znanjem popravijo načrt. Zopet naročijo novo tiskano vezje in ga preizkusijo. Ko tiskano vezje prestane začetni preizkus, ga pošljejo v proizvodnji obrat, kjer stroji namestijo elektronske komponente.

Komponente na tiskano vezje namestijo tako, da v začetek proizvodnje linije namestijo tiskano vezje. Prvi stroj nanj namaže prevodno pasto na delih, kjer bodo stiki. Naslednja naprava preveri nanos paste. V primeru slabega nanosa se tehnik odloči ali je potrebno nanos popraviti ali gre za napačno zaznan problem. Tehnik po potrebi spremeni nastavitve stroja, ki nanaša prevodno pasto. Nepopolna tiskana vezja tehnik očisti in postavi na začetek proizvodnje linije. Na takšno vezje nato naslednji stroj namesti elektronske komponente. Stroj je sestavljen iz tekočega traku, dveh glav za pobiranje in nameščanje elektronskih komponent in vhodnih trakov, ki vsebujejo elektronske komponente. Robotski glavi pobirata elektronske komponente z vhodnih trakov in jih nameščata na točno določena mesta na tiskanem vezju. Vse po-

zicije kamor mora glava postaviti komponente so vnaprej določene v načrtu, katerega je narisal inženir v fazi razvoja. Ko so vse elektronske komponente nameščene na vezje, pošlje stroj vezje v peč. Peč stopi prevodno pasto in zapeče vse komponente na vezje. Po končani peki vezja prestavijo v oddelek za kakovost.

Vezja položijo v naslednji stroj, kjer kamera pregleda vezje in poišče anomalije. Vse anomalije javi tehniku, kateri jih pregleda in se odloči ali gre res za napako ali je kamera narobe zaznala napako. Največkrat kamera napačno zazna napako, kadar napisi na senzorjih niso popolnoma enaki. Vezja, ki niso popolna zavržejo, v primeru manjše napake pa popravijo. Vezja nato preidejo v naslednji korak, kjer jih izrežejo in pobrusijo robove. Za rezanje vezij uporabijo posebno orodje, s katerim odščipnejo odvečno plastiko ali pa nož za rezanje vezij.

Celotna proizvodnja ni avtomatizirana. Za skoraj vsakim strojem stoji tehnik, ki skrbi za nemoten potek proizvodnje linije. Tehniki, med samo proizvodnjo preverjajo kakovost izdelka in lahko že med samo proizvodnjo določene komponente zavrnejo oz. popravijo, da je končni izdelek delujoč.

Zadnji korak je programiranje vezij. Na vezja naložijo mikrokodo in preizkusijo njihovo delovanje. Ko vezje prestane vse teste, ga zapakirajo in shranijo v skladišče. Imajo tudi avtomatizirana skladišča z regulacijo temperature in vlage za komponente, ki so občutljive na vlago in temperaturo.

Za svoja vezja imajo vnaprej pripravljeno mikrokodo. To kodo sprogramira programer v razvojnem oddelku. Programsko opremo razvijajo s programom Keil microvision. Pri razvoju uporabljajo posebno vezje za razvojno okolje. Na to vezje lahko namestijo različne komponente in tako simulirajo okolje končnega izdelka. Programer razvije programsko opremo, jo preizkusi in na koncu pretvori v binarno kodo, katero programator naloži na končano tiskano vezje.

2.4 Komunikacijski protokoli

Vse IoT naprave morajo imeti možnost komunikacije. V nasprotnem primeru ne moremo zajemati podatkov, pošiljati ukazov in nadzorovati naprav. Pregledali smo pogosto uporabljene protokole na aplikacijski in fizični ravni.

HTTP

HTTP (angl. Hypertext Transfer Protocol) je protokol na aplikacijski ravni namenjen komunikaciji preko svetovnega spleta. Deluje po principu zahtevka in odgovora. Odjemalec pošlje zahtevek, strežnik pa odgovori na zahtevo s podatki. Danes se uporablja za izmenjavo vsebin na svetovnem spletu in v mnogih aplikacijah pri aplikacijskih programskih vmesnikih REST. Pri aplikacijskih programskih vmesnikih se uporabljajo HTTP metode kot so *GET, POST, PUT, DELETE* za izvajanje različnih akcij. Obstaja tudi varna različica protokola imenovana HTTPS (angl. Hyper Text Transfer Protocol Secure), ki šifrira komunikacijo med odjemalcem in strežnikom [25, 26].

MQTT

MQTT (angl. Message Queuing Telemetry Transport) je enostaven protokol namenjen komunikaciji med napravami preko strežnika. Za delovanje ne potrebuje širokopasovnega internetnega dostopa, zato je primeren za vse naprave, ki nimajo dostopa do hitrega internetnega omrežja. Deluje po principu poslušanja in pošiljanja podatkov na določeno temo (angl. topic). Uporaben je za mobilne aplikacije, multi-senzorje in ostale IoT naprave, saj ima nizko porabo energije in majhne podatkovne pakete [31].

Protokol MQTT sta razvila dr. Andy Stanford-Clark in Arlen Nipper leta 1999. Leta 2014 je postal OASIS standard [33]. TPC/IP vrata (angl. port) 1833 so rezervirana pri agenciji IANA za protokol MQTT.

MQTT deluje po modelu odjemalca in strežnika [30]. Vsak senzor predstavlja odjemalca, ki se povezuje na strežnik. Strežnik v tem modelu ime-

nujemo posrednik virov (angl. broker). Največkrat je to oblak, zato te posrednike imenujemo posrednik v oblaku. Vsa komunikacija poteka po protokolu TCP v obliki pošiljanja sporočil. Vsako sporočilo je poslano na točno določen naslov, imenovan tema (angl. topic). Posrednik v oblaku skrbi za posredovanje sporočil tistim napravam, ki poslušajo na določeni temi.

Protokol MQTT nam omogoča nastavitev kakovosti storitve (angl. Quality of Service). Izbiramo lahko med tremi:

- Pošlji in pozabi,
- dostavi vsaj enkrat,
- dostavi točno enkrat.

Poleg tega lahko nastavimo sporočilo, katerega pošlje posrednik v oblaku v primeru prekinitve povezave. Tako lahko posrednik sporoči poslušajočim napravam, da je povezava z napravo prekinjena.

Če želimo varno povezavo med odjemalci in posrednikom lahko TCP povezavo kriptiramo z uporabo SSL-a ali TLS-a. Poleg tega veliko posrednikov zahteva, da se naprava avtenticira z uporabo uporabniškega imena in gesla. Protokol MQTT lahko uporabimo tudi preko WebSocket-a, kar nam omogoča direktno povezavo med dvema napravama.

CoAP

CoAP (angl. Constrained Application Protocol) je protokol za prenos dokumentov. Razumemo ga lahko kot alternativo HTTP-ju. Komunikacija poteka po protokolu UDP, zgrajen pa je po modelu odjemalca in strežnika, kjer se za zahteve uporablja model REST [30].

Tako kot protokol MQTT ima CoAP različne nastavitve kakovosti storitev. Poslana sporočila morajo biti potrjena ali pa nepotrjena, odvisno od nastavitve kakovosti storitev. Če želimo zagotoviti, da so poslana sporočila dostavljena, mora prejemnik odgovoriti s potrditvijo. V primeru nepotrjenih sporočil se sporočilo pošlje in nanj pozabi.

Ker CoAP uporablja protokol UDP za komunikacijo, povezave ne moremo šifrirati z uporabo TLS-a. Zato se uporablja DTLS, vendar naprave, ki podpirajo CoAP, pogosto podpirajo šifriranje podatkov z uporabo RSA in AES algoritmov [30].

2.4.1 Bluetooth

Bluetooth je brezžična tehnologija namenjena prenosu podatkov na kratki razdalji. Deluje na frekvenci 2.4 GHz. Prenos podatkov poteka tako, da se podatke razbije v manjše podatkovne pakete, te pa se nato pošlje drugi napravi. Danes je na trgu več različic protokola. Posebej zanimiva je različica Bluettoth 4.0 imenovana tudi Bluetooth Low Energy (v nadaljevanju BLE). Glavna prednost BLE je v manjši porabi energije. Zaradi tega je primerna za IoT naprave. BLE je večino časa v spalnem načinu, razen, ko se vzpostavi povezava za prenos podatkov. Poleg tega so intervali za prenos podatkov krajši zaradi višje hitrosti prenosa podatkov. Danes je razvit že Bluetooth 5.0, ki omogoča dvakrat hitrejši prenos podatkov kot prejšnja različica in štirikratno razdaljo pošiljanja podatkov [13, 14, 15].

2.4.2 Z-wave

Z-Wave je brezžični komunikacijski protokol za povezavo naprav. Razvit je bil za uporabo v pametnih domovih. Z-Wave deluje na frekvenci 900 MHz. Zaradi tega ne povzroča motenj na domačem brezžičnem omrežju. Ima tudi prednost pred protokolom Bluetooth, ki podpira omejeno število povezanih naprav. Z-wave protokol uporablja veliko različnih naprav. Leta 2005 je bila ustanovljena skupina Z-wave Alliance, ki je sestavljena iz več kot 300 podjetij [45]. Ta podjetja izdelujejo rešitve, ki komunicirajo z uporabo protokola Z-Wave. Poleg tega ima Z-Wave dodatno prednost. Za uporabo ne potrebujemo centralne enote, ampak najmanj 2 napravi. Vse naprave, ki komunicirajo z uporabo protokola Z-Wave med seboj ustvarijo omrežje. Zaradi tega ne potrebujemo centralne enote za nadzor omrežja, saj naprave

za to poskrbijo same.

2.4.3 ANT

ANT je protokol za brezžično komunikacijo namenjen komunikaciji med senzorji. Deluje na frekvenci 2.4 GHz podobno kot BLE. Naprave, ki uporabljajo ANT, lahko delujejo kot pošiljatelji, prejemniki in posredniki podatkov. Tako lahko naprave vzpostavijo svoje omrežje. Ima nizko porabo energije, pogosto pa je uporabljen v senzorjih za šport, kot so ure, merilniki srčnega utripa in drugi [6].

2.4.4 3G/4G

3G predstavlja tretjo generacijo brezžične telekomunikacijske tehnologije. Uporabljajo jo predvsem mobilne naprave za dostop do interneta in video klicev. 4G je četrta generacija in naslednik tretje generacije brezžične telekomunikacijske tehnologije. Omogoča predvsem hitrejši prenos podatkov. V urbaniziranih delih sveta je pokritost s to tehnologijo zelo dobra zaradi česar je zanimiva na področju interneta stvari. Z uporabo te tehnologije lahko IoT napravo postavimo skoraj kjerkoli in bo imela dostop do interneta. Podjetje SORACOM izdeluje SIM kartice, katere lahko uporabimo v IoT napravah. Prednost teh SIM kartic je v tem, da delujejo pri večini ponudnikov mobilnih omrežij na svetu. Tako imajo naprave dostop do interneta, vse prenešene informacije pa so šifrirane [1, 2, 39].

2.4.5 ZigBee

Zigbee je protokol za brezžično komunikacijo med napravami. Danes ga uporabljajo telekomunikacijska podjetja za nadzor uporabnikovih naprav, kot so satelitski sprejemniki, prehodni vmesniki in ostale naprave. Ima nizko porabo energije, zato je primeren za manjše naprave in naprave uporabljene v pametnih domovih. Zaradi tega ga uporabljajo različne pametne naprave kot so pametni termostati, pametne vtičnice, Philips Hue in ostali. [43, 46].

2.4.6 Thread

Thread je internetni protokol narejen na osnovi IPv6. Namenjen je predvsem internetu stvari in napravam za avtomatizacijo pametnega doma, ki komunicirajo preko lokalnega omrežja. Za delovanje uporablja 6LoWPAN (angl. IPv6 over Low power Wireless Personal Area Networks). Naprave, ki komunicirajo s protokolom Thread, ne potrebujejo posebne naprave za nadzor komunikacije. Zaradi tega je takšen protokol primeren za avtomatizacijo pametnih domov, saj se ne more pokvariti centralna naprava in s tem celotno omrežje. Vsako napravo, ki uporablja Thread protokol, lahko naslovimo preko IP naslova. V trenutni različici v enem Thread omrežju protokol podpira do 250 povezanih naprav [41, 44].

2.5 Izbira multi-senzorskih naprav in platform

Po pregledu področja pametnih domov smo izbrali naprave in rešitve, ki smo jih uporabili v naši rešitvi. Odločili smo se, da bomo kot IoT naprave izbrali multi-senzorje FireFly. Te multi-senzorji nam omogočajo zajem različnih podatkov, imajo dobro dokumentirano komunikacijo in izdeluje pa jih slovensko podjetje. Arhitektura multi-senzorjev FireFly nam omogoča nadgradnjo le teh. Zaradi uporabljenega I2C vodila lahko na FireFly namestimo dodatne module in tako dodamo želeno funkcionalnost. Ker nismo želeli razviti prototipa zaprtega sistema, smo se odločili, da bomo uporabili oblak Bluemix in odprtokodno platformo OpenHab2. V oblak lahko povežemo katerokoli napravo, ki je zmožna komunikacije preko protokola MQTT. V primeru, da naprava nima dostopa do interneta, izkoristimo prehodni vmesnik. Prehodni vmesniki so pogosto uporabljeni v scenarijih interneta stvari, saj se povezujejo z različnimi napravami z uporabo različnih protokolov, kot je na primer Bluetooth. Prehodni vmesniki prejemajo podatke z naprav, te pa pošiljajo v internet. Zaradi tega smo izbrali odprtokodno platformo OpenHab2. OpenHab2 ima veliko število podprtih naprav za pametne domove, poleg tega pa omogoča razvijalcem, da podprejo nove naprave. Z izbiro teh tehnologij smo

sestavili sistem, ki omogoča dodajanje različnih naprav, vendar smo se za demonstracijo delovanja omejili na multi-senzorje FireFly.

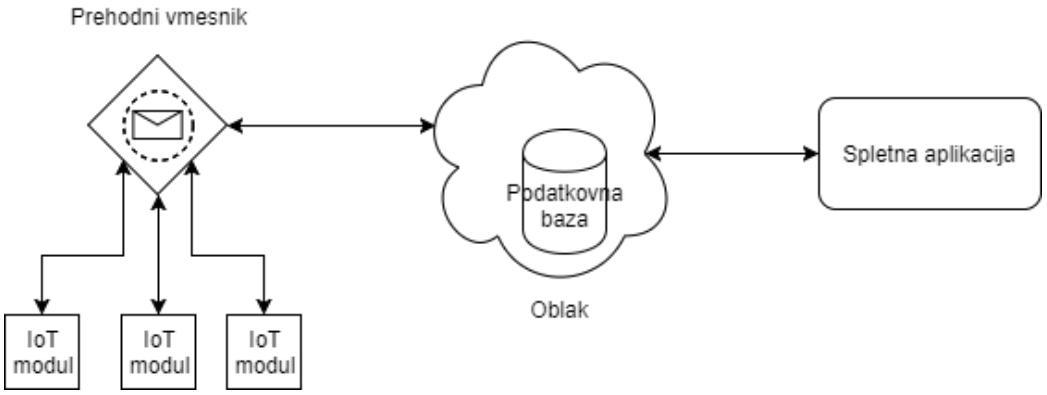
Poglavje 3

Zasnova sistema

Glavni gradniki rešitev v internetu stvari so IoT naprave. Te zajemajo podatke iz okolja in jih pošiljajo drugim komponentam sistema za analizo in prikaz. Ker smo želeli, da naš sistem dopušča realizacijo različnih scenarijev, smo pregledali možne komponente za preostanek sistema. Praksa na področju interneta stvari je, da so IoT naprave povezane s prehodnim vmesnikom (angl. gateway), kateri skrbi za komunikacijo med IoT napravami in internetom. Povezavo IoT naprav z internetom in poslane podatke lahko dodobra izkoristimo z uporabo oblaka. Oblak nam omogoča realizacijo različnih scenarijev, saj lahko uporabimo različne servise. Ti servisi lahko skrbijo za komunikacijo, shranjevanje in analizo podatkov. Poleg tega nam oblak omogoča gostovanje različnih aplikacij, enostavno skaliranje in enostavno dodajanje novih funkcionalnosti.

3.1 Arhitektura in opis komponent

Arhitektura sistema je sestavljena iz treh glavnih komponent (slika 3.1). Iz **prehodnega vmesnika**, **oblaka** in **spletne aplikacije**. Vsaka komponenta igra pomembno vlogo za nemoteno delovanje celotnega sistema.



Slika 3.1: Osnovna arhitektura

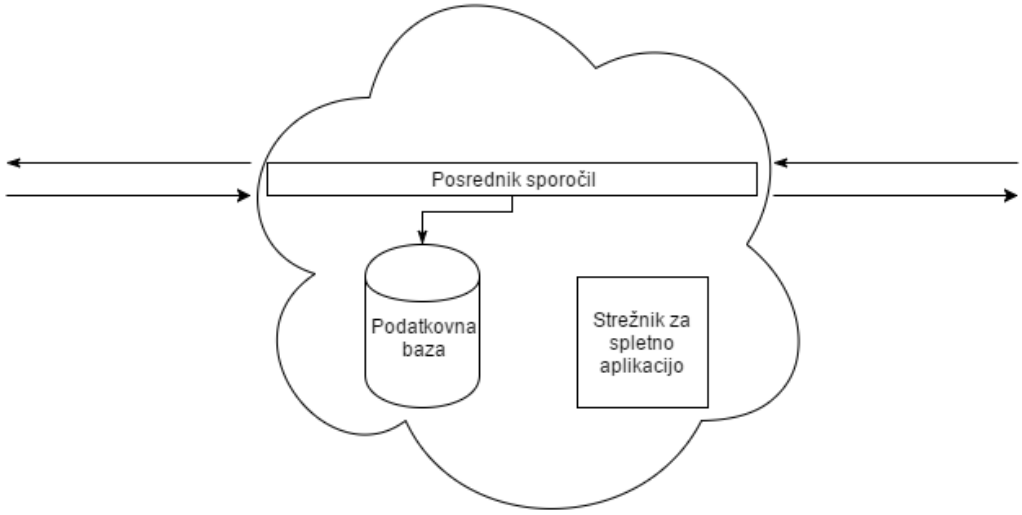
3.1.1 Prehodni vmesnik

Prehodni vmesnik je vhodna komponenta sistema. Različne IoT naprave imajo različne načine povezave in komunikacije. Nekatere se lahko povežejo z uporabo brezžičnega omrežja WiFi, druge z uporabo Bluetooth-a. Poleg tega imajo različne standarde za komunikacijo in pošiljanje zajetih podatkov. Zaradi tega na področju interneta stvari uporabljamo prehodne vmesnike. Ti skrbijo za povezavo med IoT napravami in oblakom.

Prehodni vmesnik se poveže z vsako IoT napravo, ki jo imamo v nekem okolju. Naprava zajema podatke iz okolja in jih pošilja prehodnemu vmesniku. Ta podatke sprejme, jih po potrebi pretvori in pošlje oblaku. Zmožen je dvosmerne komunikacije, kar pomeni, da lahko sprejema sporočila in jih pošlje ustrezni napravi. Na takšen način lahko nadgradimo programsko opremo povezanih IoT naprav, jim spremenimo način delovanja ali pa ugasnemo določene senzorje. V oblak zaradi varnosti ne moremo povezati naprav brez, da se te avtenticirajo. Tako prehodni vmesnik ne skrbi samo za komunikacijo, ampak tudi za varnost in avtentikacijo IoT naprav.

3.1.2 Oblak

Oblak je osrednja komponenta sistema. Arhitektura naše rešitve v oblaku je sestavljena iz **posrednika sporočil**, **strežnika** spletne aplikacije in **podatkovne baze** (slika 3.2). Posrednik sporočil skrbi za komunikacijo med prehodnimi vmesniki, IoT napravami in odjemalci, kot je na primer spletna aplikacija. Povezan je s podatkovno bazo, kar nam omogoča shranjevanje prejetih sporočil. Shranjeni podatki so dostopni preko programskega vmesnika. To izkorišča spletna aplikacija za pridobitev in prikaz shranjenih podatkov. Spletno aplikacijo servira strežnik, ki teče v oblaku. Oblak nam omogoča dodajanje in odvzem resursov po potrebi. To pomeni, da plačujemo toliko, kolikor porabimo in zagotovimo nemoteno delovanje sistema.



Slika 3.2: Komponente v oblaku

3.1.3 Spletna aplikacija

Spletna aplikacija je izhodna komponenta sistema namenjena uporabniku. Prikazuje stanje vseh registriranih naprav v IoT platformi in omogoča vizu-

alizacijo podatkov poslanih z IoT naprav, kar omogoča uporabniku pregled nad želenim okoljem. Servira jo strežnik v oblaku. Povezana je s posrednikom sporočil, kar nam omogoča prejemanje sporočil poslanih iz IoT naprav. Z uporabo programskega vmesnika podatkovne baze Cloudant, spletna aplikacija dostopa do shranjenih podatkov.

Ker je spletna aplikacija dostopna na internetu, je vsa komunikacija šifrirana. Povezava med strežnikom in odjemalcem uporablja HTTPS, vsa poslana MQTT sporočila pa so šifrirana z uporabo TLS-a.

3.2 Izbor tehnologij

Preden smo se lotili razvoja rešitve po opisani arhitekturi, smo pregledali različne tehnologije. Začeli smo z izbiro načina komunikacije med komponentami sistema. Želeli smo hitro komunikacijo med IoT napravami in odjemalci. Po pregledu različnih protokolov smo izbrali protokol MQTT. Izbrali smo ga zaradi njegove enostavne uporabe, hitrosti pri pošiljanju sporočil ter podpore na oblačnih platformah.

Sledila je izbira ponudnika oblaka in oblačne platforme. Po pregledu oblačnih platform smo se odločili za IBM-ovo platformo Bluemix. Izbrali smo jo zaradi IoT platforme, ki omogoča komunikacijo preko protokola MQTT ter avtomatskega nadzora varnosti podatkovne baze in šifrirane komunikacije. Poleg tega je IBM objavil knjižnico za uporabo protokola MQTT. Ta knjižnica nam omogoča zelo hitro komunikacijo med komponentami sistema, kar smo izkoristili pri vizualizaciji podatkov v spletni aplikaciji. Pri izbiri smo se oprli tudi na tehnologijo NodeRed [32] (slika 3.3).

3.2.1 NodeRed

NodeRed je programsko orodje, ki ga je razvilo podjetje IBM. Omogoča nam razvoj funkcionalnosti brez pisanja kode tako, da uporabimo posebne bloke. Z uporabo blokov lahko med seboj povežemo servise v oblaku, posrednika sporočil s podatkovno bazo, razvijemo programski vmesnik in razvijemo po-

ljubno funkcionalnost. Vse to pripomore k hitremu razvoju prototipa, kar je danes ključnega pomena.

Slika 3.3: Primer povezave IoT platforme s podatkovno bazo z uporabo NodeRed

Tehnologija NodeRed deluje kot strežnik, kar smo izkoristili za serviranje spletne aplikacije. Moderne spletne aplikacije dinamično prikazujejo vsebino glede na uporabnikovo interakcijo. Izbrali smo tehnologijo React.js.

3.2.2 React.js

React je odprtokodna knjižnica napisana v jeziku JavaScript, namenjena grajenju uporabniških vmesnikov. Omogoča enostavno grajenje interaktivnih uporabniških vmesnikov z uporabo komponent. Komponenta predstavlja opis posameznega dela aplikacije, React pa jo pretvori v HTML kodo z uporabo virtualnega DOM (angl. Document Object Model) drevesa. To omogoča izredno hitro izrisovanje elementov, saj React primerja izrisano vsebino z novo virtualno vsebino. Tako zamenja samo del DOM drevesa, kar povzroči izris dela spletne aplikacije namesto celotne. Izris sprememb se zgodi avtomatsko, ko shranimo ali spremenimo podatke v stanju (angl. state) aplikacije.

Izbrali smo ga zaradi hitrega izrisovanja elementov v spletnem brskalniku, kar smo izkoristili za vizualizacijo podatkov z IoT naprav.

3.3 Opis rešitve

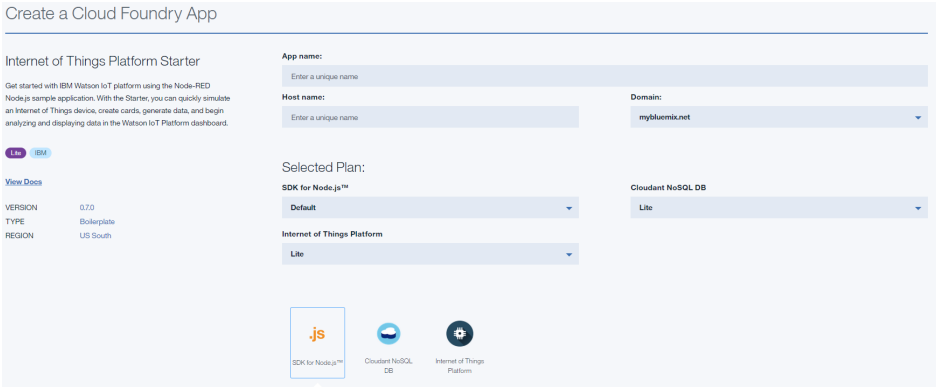
Po načrtovanju sistema in izboru tehnologij, smo začeli z implementacijo sistema. Pri razvoju je pomembno vmesno testiranje posameznih komponent.

Zato smo na mobilno napravo z operacijskim sistemom Android naložili dve aplikaciji. Prva, z imenom IoT Starter, uporablja senzorje v mobilnem telefonu in pošilja podatke v Bluemix. Druga aplikacija, z imenom FireFly-IoT, se preko Bluetootha poveže s FireFly modulom in pošilja podatke v oblak preko protokola MQTT. Z uporabo teh dveh aplikacij smo imeli možnost testiranja komunikacije med IoT napravo in oblakom. Nato smo se lotili postavitve okolja v oblaku.

3.3.1 Postavitev okolja v oblaku

Preden smo lahko uporabili oblačno platformo Bluemix, smo se morali registrirati. Po uspešni registraciji in prijavi v sistem smo morali izbrati lokacijo strežnikov. Izbirali smo lahko med Nemčijo, Avstralijo, Veliko Britanijo in jugom Združenih držav Amerike. Izbrali smo jug ZDA, saj so to najbolj oddaljeni strežniki. Te smo izbrali, saj smo želeli pokazati hitrost MQTTja.

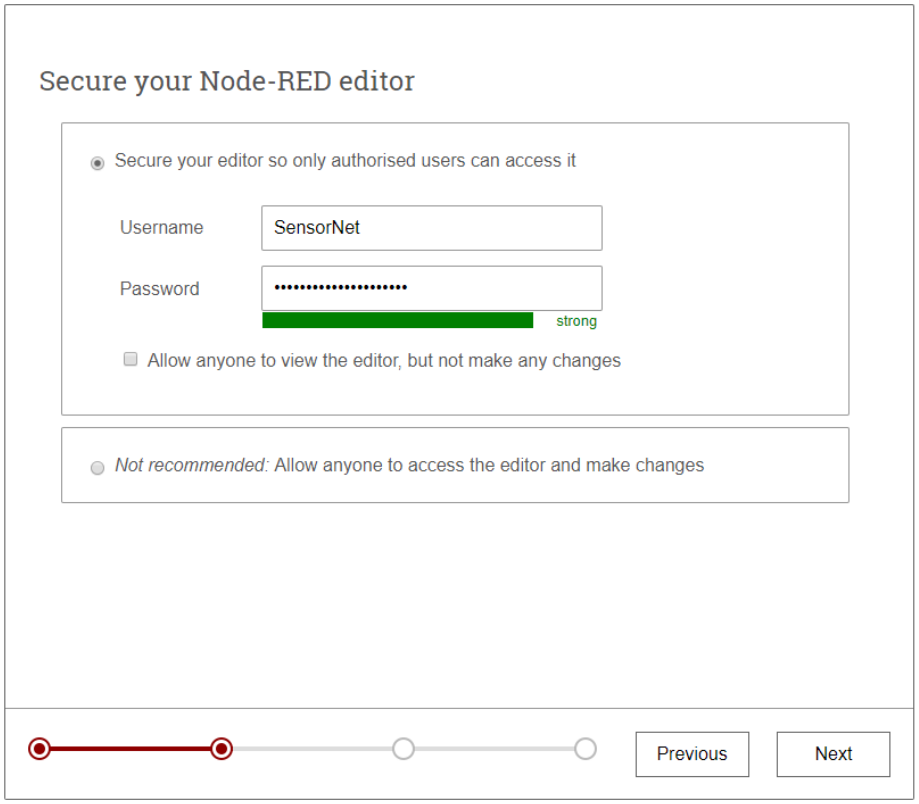
Za izbrano regijo smo morali kreirati prostor (angl. space) za aplikacijo. V izbrani regiji imamo lahko več različnih prostorov. Izberemo lahko poljubno ime za prostor. Po kreaciji prostora pa nam spletna platforma ponudi različne začetne pakete, s katerimi lahko zaženemo osnovne aplikacije. Izbrali smo IoT platformo.



Slika 3.4: Postavitev IoT platforme

Po vnosu imena aplikacije in domenskega imena Bluemix zažene IoT plat-

formo. Prvi zagon aplikacije potrebuje nekaj časa, saj Bluemix postavi aplikacijski strežnik NodeRed in IoT platformo. Ko se aplikacija postavi, je dostopna na URL naslovu, ki smo ga definirali pri izbiri imena aplikacije. V tej fazi je aplikacijski strežnik dostopen vsem, ki poznajo URL naslov. Zato je priporočljivo, da odpremo URL naslov aplikacije in nastavimo uporabniške pravice.

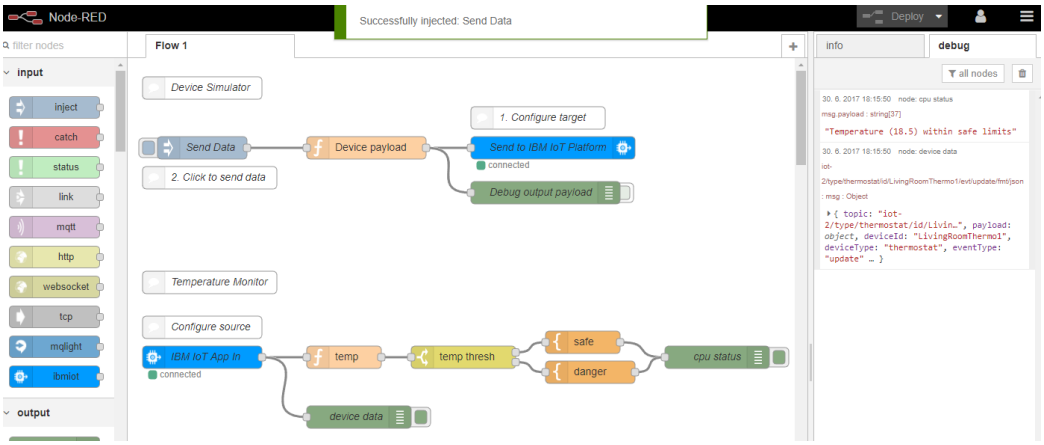


Slika 3.5: Zaščita aplikacijskega strežnika

Na ta način zaščitimo dostop do strežnika pred nepooblaščenimi uporabniki in zaključimo s postavitvijo aplikacijskega strežnika in IoT platforme. Po zaključeni konfiguraciji smo se vpisali v urejevalnik NodeRed. Dostopen je na istem URL naslovu kot aplikacija s pripono **/red**. Druga možnost pa je, da po zaključeni konfiguraciji kliknemo gumb, ki nas odpelje v urejevalnik.

V naslednjem koraku smo preizkusili delovanje postavljene aplikacije z

uporabo urejevalnika NodeRed. Urejevalnik ima ob prvem odprtju že prikazan primer uporabe, s katerim lahko preizkusimo pošiljanje MQTT sporočil. Na desni strani urejevalnika smo odprli okno za razhroščevanje aplikacije (angl. debug) in pritisnili na gumb *Send Data*. Ob kliku se pošlje sporočilo in se prikaže v desnem oknu.



Slika 3.6: Testiranje aplikacije v oblaku

Preizkusiti smo želeli, če deluje povezava z zunanjimi IoT napravami. Zato smo izkoristili nameščeni Android aplikaciji. Preden lahko povežemo katerokoli napravo v oblak, jo moramo definirati (opisano v poglavju 3.7). V nasprotnem primeru se IoT naprava ne more povezati z platformo, saj ji ta ne pusti povezave brez avtentikacije. Poleg tega moramo poznati podatke o IoT platformi, saj v nasprotnem primeru ne moremo povezati IoT naprav.

Na pregledni plošči Bluemix-a smo izbrali IoT servis. To nas je pripeljalo na stran, kjer smo lahko odprli spletno aplikacijo za nadzor IoT platforme. Z uporabo spletne aplikacije IBM Watson IoT Platform smo definirali nove IoT naprave. Poleg tega potrebujemo za povezavo IoT naprav poznati naslov, kamor povežemo naprave. Te podatki so dostopni na pregledni plošči Bluemixa. Izberemo našo aplikacijo in nato povezave. Med povezavami vidimo vse servise, ki so povezani z našo aplikacijo. Vsak servis ima svoje prijavne podatke. Ker želimo povezati naše naprave z IoT platformo moramo zanjo

pridobiti prijavne podatke. To storimo tako, da kliknemo na gumb *View credentials*, ki nam prikaže vse prijavne podatke. Poznati moramo ime organizacije, tip naprave, ki smo ga definirali v prejšnjem koraku in geslo. S temi podatki lahko povežemo aplikacijo IoT Starter z IoT platformo. Ob uspešni povezavi s platformo se v urejevalniku NodeRed prikažejo poslana sporočila s telefona.

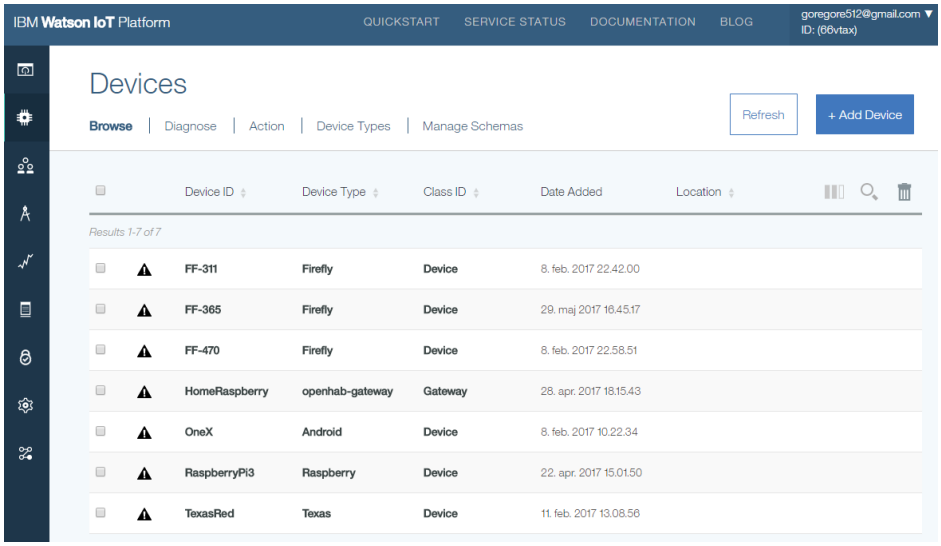
IBM Watson IoT Platform

IBM Watson IoT Platform (v nadaljevanju IoT platforma) je platforma, namenjena aplikacijam na področju interneta stvari. Pri postavitvi okolja in aplikacije je Bluemix avtomatično povezal IoT platformo in podatkovno Cloudant NoSQL kot servisa z našo aplikacijo. V IoT platformi teče posrednik sporočil. Poleg tega skrbi za varno komunikacijo, registracijo in avtentikacijo naprav in shranjevanje sporočil v podatkovno bazo. Delovanje in nastavitve IoT platforme lahko spreminjamo preko spletne aplikacije. Dostopna je preko Bluemix pregledne plošče, kjer izberemo povezani IoT servis. To nas pripelje do strani, kjer lahko zaženemo spletno aplikacijo IoT platforme. Ob zagonu spletne aplikacije nas pripelje na stran, kjer lahko definiramo IoT naprave (slika 3.7).

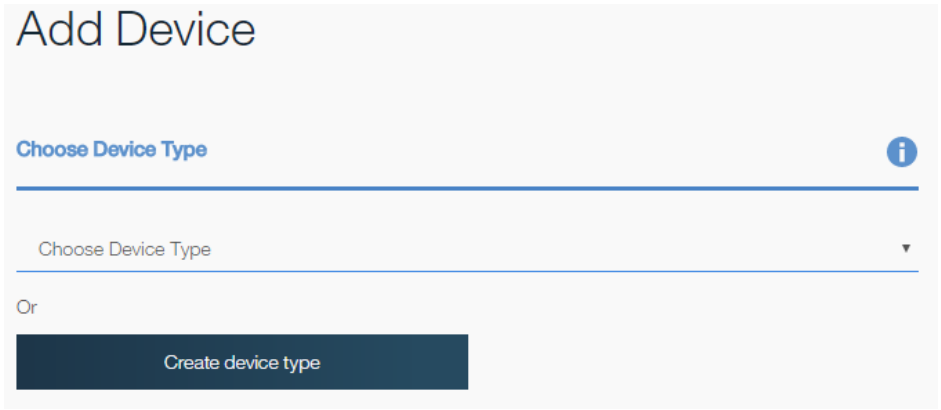
Vsaka IoT naprava mora imeti svoj enolični identifikator in tip, drugače se ne more povezati z IoT platformo. Zato moramo vsako napravo, ki jo želimo povezati definirati. To lahko storimo ročno ali pa uporabimo aplikacijski programski vmesnik, preko katerega lahko to storimo programsko. Sami smo naprave definirali ročno, saj smo imeli majhno število naprav. Kliknili smo na gumb *Add a device*, kar nam je odprlo novo okno (slika 3.8).

Preden smo lahko registrirali napravo, smo morali napravam dodeliti tip. Definirali smo tri različne tipe naprav:

- Android,
- Firefly,
- Raspberry.



Slika 3.7: IBM Watson IoT Platform



Slika 3.8: Registracija IoT naprav

S tipom naprave povemo za kakšen tip naprave gre. To nam omogoča, da naprave lažje ločimo med sabo. Ko smo definirali tipe, smo lahko registrirali naše naprave. Vsaki napravi moramo dodeliti enolični identifikator in geslo s katerim se naprava avtenticira. Namesto gesla lahko uporabimo certifikate, kar predstavlja dodaten varnostni nivo. Tako se naprave avtenticirajo brez gesla z uporabo certifikatov.

Poleg registracije naprav na IoT platforma omogoča pregled porabe, prikazuje povezane naprave, brisanje naprav, spreminjanje varnostnih nastavitev, določanje pravil in priključkov z zunanjimi storitvami. Uporabili smo priključek z imenom Historical Data Storage. Ta priključek nam omogoča avtomatično shranjevanje sporočil poslanih preko IoT platforme v našo podatkovno bazo.

Podatkovna baza Cloudant

Cloudant NoSQL je nerelacijska podatkovna baza za shranjevanje dokumentov tipa JSON. Na oblačni platformi Bluemix je na voljo kot storitev (angl. DataBase as a Service, DBaaS) [16]. Zgrajena je na osnovi odprto-kodne platforme Apache CouchDB™ [7].

Povezana je z našo aplikacijo in IoT platformo. Uporabljamo jo za shranjevanje poslanih sporočil. Do podatkovne baze dostopamo preko pregledne plošče Bluemix, kjer se nahaja povezava do spletne aplikacije IBM Cloudant. Ta nam omogoča vpogled v posamezne podatkovne baze, kreiranje in brisanje podatkovnih baz, nastavitve dostopa, grajenje shem in indeksov. Tako kot IoT platforma ima aplikacijski programski vmesnik, katerega lahko izkoristimo za programsko grajenje indeksov, branje podatkov in spreminjanje določenih nastavitev. Ta vmesnik izkorišča priključek Historical Data na IoT platformi. Priključek vsak mesec naredi novo podatkovno bazo, v katero shranjuje poslana sporočila za tekoči mesec. Preden se sporočilo shrani v podatkovno bazo, se mu doda časovni žig v UTC formatu.

Vsako sporočilo je tipa JSON in se shrani v podatkovno bazo kot svoj dokument. Cloudant nam omogoča brskanje po teh dokumentih z uporabo poizvedb imenovanih Cloudant Query.

Primer shranjenega dokumenta:


```
{
  "_id": "0031b720-5e8c-11e7-bad1-1187caee0f54",
  "_rev": "1-848f563d1a51077198327a175c6d7df3",
  "deviceType": "Android",
```

```
  "deviceId": "OneX",
  "eventType": "accel",
  "format": "json",
  "timestamp": "2017-07-01T18:35:04.978Z",
  "data": {
    "d": {
      "acceleration_x": -2.1395874,
      "acceleration_y": 7.2100525,
      "acceleration_z": 4.3945465,
      "roll": 0.45309123,
      "pitch": -0.9750539,
      "yaw": -0.06882584,
      "lon": 46.050194,
      "lat": 14.468963
    }
  }
}
```


Postavitev aplikacije na Bluemix-u

Po preizkusu postavljenega okolja smo želeli naložiti na aplikacijski strežnik testno spletno aplikacijo. Bluemix uporablja tehnologijo Cloud Foundry za postavitev kontejnerjev, ki tečejo v oblaku. Zaradi tega se vse aplikacije postavi z uporabo vmesnika ukazne vrstice (angl. Command-line interface, CLI). Z uporabo določenih ukazov naložimo programsko kodo aplikacije na Bluemix. Vendar smo morali naprej pridobiti programsko kodo strežnika. Na pregledni plošči Bluemixa smo izbrali našo aplikacijo in omogočili uporabo Continuous delivery-ja. S tem smo povezali servis Continuous Delivery Toolchain z našo aplikacijo. Ta servis nam omogoča povezavo z git repozitorijem in avtomatsko postavitvijo aplikacije. Nastavili smo integracijo z našim GitHub računom, kar je omogočilo servisu postavitev repozitorija. Tako smo prišli do programske kode. Poleg tega smo nastavili, da se zadnja verzija apli-

kacije samodejno naloži na Bluemix, ko zazna servis spremembe na določeni git veji. Tako smo si olajšali delo s postavitvijo nove verzije aplikacije.

Tako smo imeli postavljeno aplikacijo v oblaku, v katero smo lahko povezali zunanje IoT naprave, videli poslana sporočila in jih shranili v podatkovno bazo. V naslednjem koraku smo se lotili razvoja spletne aplikacije, saj smo imeli postavljeno okolje in dostop do programske kode strežnika.

3.3.2 Spletna aplikacija SensorNet

Razvoja spletne aplikacije smo se lotili lokalno. Z uporabo tehnologije Node.js in vmesnika ukazne vrstice npm smo naložili modul, ki nam je omogočil generacijo strukture projekta za spletno aplikacijo. Uporabili smo modul *create-react-app*, ki nam zgradi osnovno strukturo projekta in aplikacije v tehnologiji React.js. Ta modul smo izbrali, saj nam omogoča gradnjo projekta in lokalni razvojni strežnik. Prednost uporabe takšnega strežnika je v tem, da ob spremembah programske kode ne potrebujemo ponovno naložiti spletne strani v brskalniku. Omogoča nam tudi generacijo produkcijskih paketov, kjer se vsa programska koda strne v eno samo datoteko.

Razvoj spletne aplikacije SensorNet

Ko smo imeli postavljen projekt in lokalno razvojno okolje, smo definirali želeno funkcionalnost spletne aplikacije. To smo pretvorili v risbe (angl. mockup), katere so nam služile kot načrt spletne aplikacije. Nato smo se lotili implementacije ogrodja spletne aplikacije. Želeli smo razviti svojo rešitev, zato nismo uporabili nobenega programskega ogrodja za komponente. Razvoj ogrodja smo začeli z osnovno postavitvijo spletne aplikacije, nadaljevali pa smo z razvojem posameznih elementov. Razvili smo svoje komponente za:

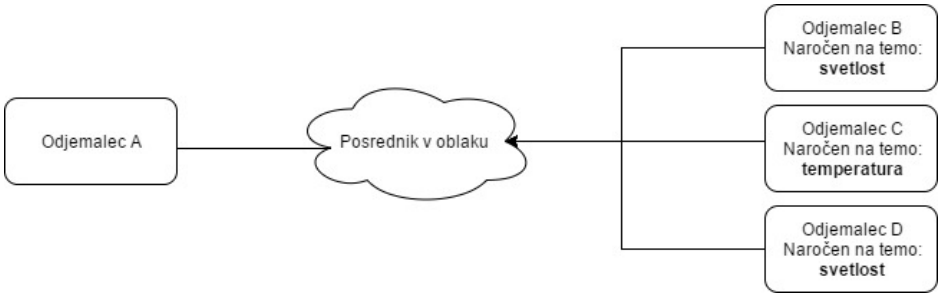
- gumbe,
- sezname in elemente seznama,
- kartice,

- povezave,
- menije,
- tabele,
- naslove.

Tako smo definirali osnovno obnašanje elementov in sam izgled aplikacije. Nadaljevali smo z implementacijo osnovne strukture spletne aplikacije. Spletno stran smo razdelili na glavo in vsebino. Glava spletne aplikacije služi za prikaz navigacije, menijev, stanja povezave in imena aplikacije. Ta del aplikacije je statičen in se med samo uporabo ne spreminja. Vsebina spletne aplikacije se dinamično prikazuje glede na uporabnikovo interakcijo.

Ko smo definirali osnovno strukturo spletne aplikacije, smo dodali knjižnice. Sprva smo dodali knjižnico *ibmiotf*, ki skrbi za MQTT komunikacijo.

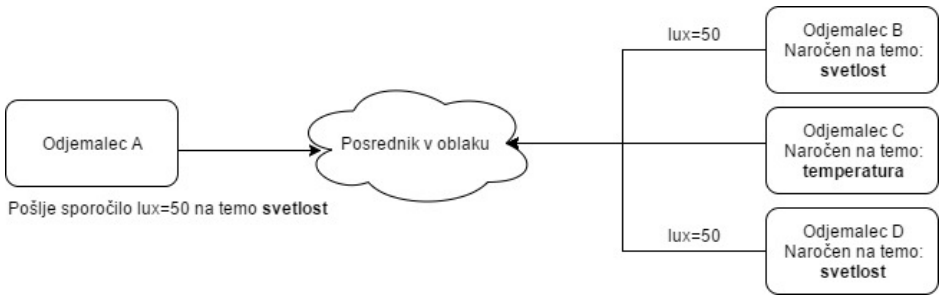
Za posredovanje MQTT sporočil skrbi posrednik v oblaku. Sporočila posreduje tistim napravam, ki poslušajo na določeni temi (glej sliko 3.9).



Slika 3.9: Naročnina odjemalcev

Ko odjemalec A pošlje sporočilo *lux=50* na temo **svetlost**, posrednik v oblaku dostavi sporočilo ustreznim poslušalcem (glej sliko 3.10). Ker odjemalec C ne posluša na temi **svetlost**, ne prejme sporočila, ki ga je poslal odjemalec A.

Z uporabno naročnin spletna aplikacija pridobiva podatke o stanju sistema. Ob zagonu spletne aplikacije se ta naroči na temo **iot-2/type/+/id/+/mon**.



Slika 3.10: Odjemalec A pošlje sporočilo na temo svetlost

Na tej temi posrednik sporočil vrne stanje vseh registriranih naprav. Poleg tega, ob spremembi stanja katerekoli naprave pošlje novo sporočilo, ki vsebuje novo stanje naprave. Primer prejetih sporočil:


```
{
        "Action": "Connect",
        "Time": "2017-07-06T09:03:02.386Z",
        "ClientAddr": "188.64.25.43",
        "ClientID": "d:66vtax:Firefly:FF-311",
        "Port": 1883,
        "Secure": false,
        "Protocol": "mqtt4-ws",
        "Durable": false
}
{
        "Action": "Disconnect",
        "Time": "2017-07-05T13:12:48.840Z",
        "ClientAddr": "212.85.163.183",
        "ClientID": "d:66vtax:Android:OneX",
        "Port": 1883,
        "Secure": false,
        "Protocol": "mqtt3",
        "ConnectTime": "2017-07-05T13:00:39.314Z",
```

```
        "CloseCode": 160,
        "Reason": "The connection timed out",
        "ReadBytes": 149009,
        "ReadMsg": 634,
        "WriteBytes": 32,
        "WriteMsg": 0
}
```


Na podlagi teh podatkov spletna aplikacija izriše pregledno ploščo, kjer so vidne vse registrirane naprave in njihovo stanje (slika 3.11).

Slika 3.11: Stanje naprav na pregledni plošči spletne aplikacije SensorNet

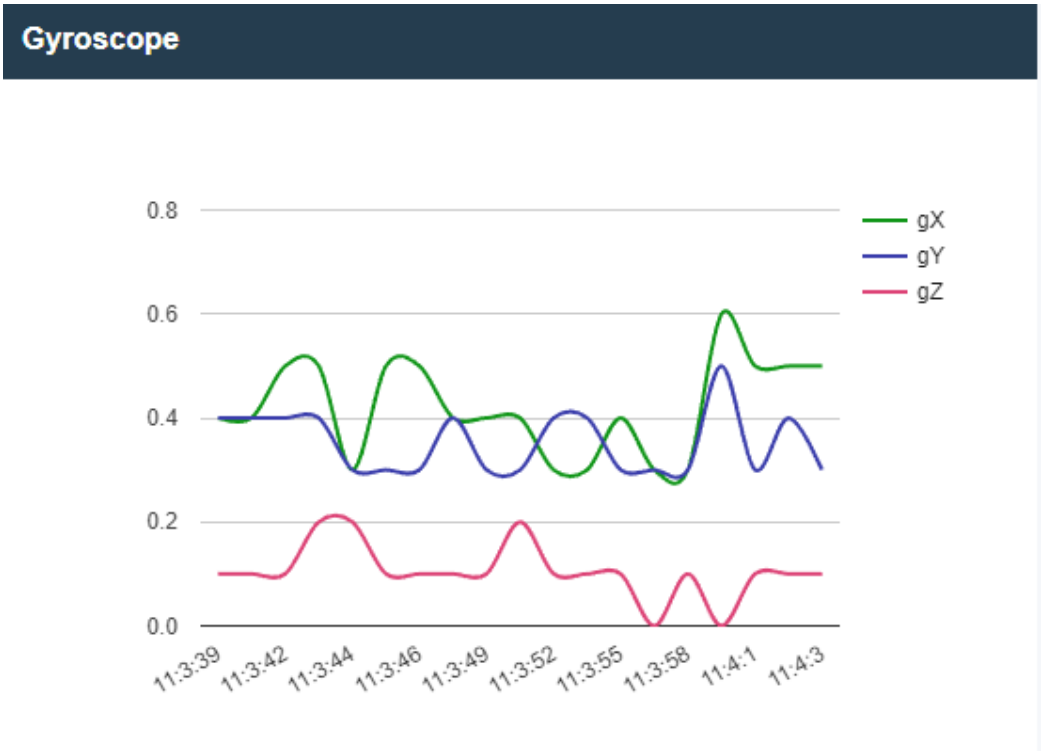
Watson IoT platforma ima v naprej določen format za MQTT teme. Začne se z *iot-2/*, nato pa dodamo **tip naprave**, **enolični identifikator**, **tip dogodka** in **format** poslanih podatkov. To smo izkoristili za vizualizacijo podatkov na spletni aplikaciji. Ko uporabnik izbere želeno napravo, se spletna aplikacija naroči na temo naprave. Z uporabo naročnine spletna aplikacija prejema sporočila poslana z IoT naprave in vizualizira prejete podatke. Če uporabnik izbere napravo FF-311 (slika 3.11), se spletna aplikacija naroči na *iot-2/type/Firefly/id/FF-311/evt/+/fmt/json*. Pri tipu dogodka smo uporabili poseben znak +, kar pomeni, da smo naročeni na vse tipe dogodka.

Nato smo želeli vizualizirati prejeta sporočila. Za vizualizacijo podatkov smo uporabili knjižnico podjetja Google, saj deluje dobro pri izrisu dinamičnih podatkov. Omogoča nam uporabo različnih grafov in zemljevida. Ker so vsa prejeta sporočila tipa JSON, smo morali pretvoriti podatke v format razumljiv grafom, imenovan *DataTable*. Ta format predstavlja dvodimenzionalno tabelo, saj vsak stolpec predstavlja atribut, vrstica pa vrednosti podatkov. Pretvorjena sporočila smo nato shranili v stanje spletne aplikacije, ki shrani zadnjih 20 sporočil. Ko spletna aplikacija prejme 21-to sporočilo, zavrže prvo prejeto sporočilo, doda novo sporočilo na konec tabele in shrani nove podatke v stanje aplikacije. Sprememba stanja aplikacije povzroči ponoven izris komponente s spremenjenimi podatki. To smo izkoristili za vizualizacijo podatkov. Ko shranimo novo sporočilo, aplikacija izriše prejete podatke. Na ta način smo dinamično vizualizirali prejete podatke, kot prikazuje slika 3.12.

Poleg dinamične vizualizacije smo želeli izkoristiti shranjene podatke v podatkovni bazi in omogočiti uporabniku vpogled v shranjene podatke. Preden smo lahko povezali spletno aplikacijo s podatkovno bazo, smo morali na podatkovni bazi nastaviti pravice za dostop. Dostop smo omejili na domeno, kjer teče spletna aplikacija. Poleg tega smo kreirali uporabniško ime in geslo za dostop do podatkovne baze. Te podatke smo uporabili na spletni aplikaciji za povezavo s podatkovno bazo.

Spletna aplikacija dostopa do podatkov v podatkovni bazi z uporabo REST API-ja. Za pridobitev podatkov spletna aplikacija zgradi HTTP zahtevek. V glavo zahtevka doda polji *Authorization*, kjer pošljemo prijavne podatke zakodirane v formatu base64 in *Content-Type*, kjer povemo v kakšnem formatu pošiljamo podatke. V telesu zahtevka pošljemo Cloudant Query, kjer povemo podatkovni bazi, kakšne podatke želimo. Primer telesa zahtevka:

```
{
        "selector": {
                "$and": [{
                        "deviceId": "FF-311"
```



Slika 3.12: Vizualizacija podatkov

```
            }, {
                    "timestamp": {
                            "$gt": "2017-07-04T22:00:00.000Z"
                    }
            }, {
                    "timestamp": {
                            "$lt": "2017-07-06T20:00:00.000Z"
                    }
            }]
    },
    "fields": ["data", "timestamp"],
    "limit": 300
}
```

Prejete podatke pretvorimo v format berljiv grafom, jih shranimo v stanje aplikacije, ta pa izriše prejete zgodovinske podatke. Tako ima spletna aplikacija dva načina pridobivanja podatkov z uporabo naročnin in z branjem podatkov iz podatkovne baze. S tem smo zaključili z razvojem spletne aplikacije in se lotili razvoja prehodnega vmesnika.

3.3.3 Prehodni vmesnik

Pred razvojem programske opreme prehodnega vmesnika smo potrebovali napravo, ki bo služila kot prehodni vmesnik. Izbrali smo si Raspberry Pi 3 model B. Pri izbiri smo se oprli na Bluetooth modul, saj multi-senzorji FireFly komunicirajo z uporabo nizko potratnega Bluetootha. Poleg tega je Raspberry Pi 3 model B cenovno dostopen in za naše potrebe dovolj zmogljiv.

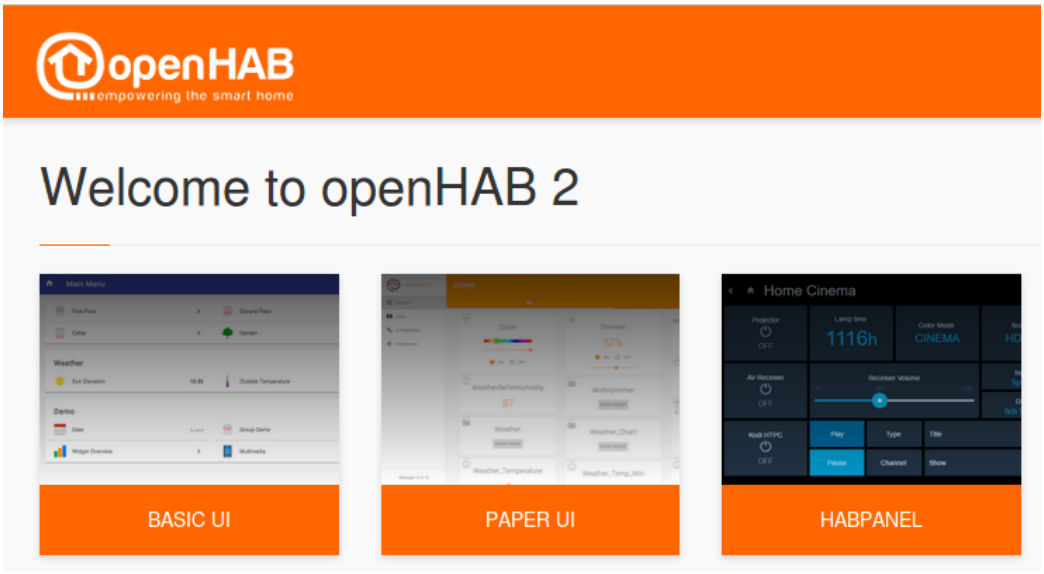
Raspberry Pi je majhen računalnik, na katerem teče operacijski sistem Raspbian. Raspbian je različica operacijskega sistema Linux, najbližje pa je distribuciji Debian. Poganja ga štiri jedrni 64-bitni procesor in ima 1 GB RAM-a. Vsebuje modula za brezžično komunikacijo WiFi in Bluetooth, štiri USB priključke, ethernet vhod, HDMI izhod in micro USB priključek za napajanje.

Na Raspberry Pi 3 smo namestili odprtokodno platformo OpenHab2. Po uspešni namestitvi smo izbrali način postavitve platforme. Odločili smo se za priporočeni paket imenovan Standard package. Ta paket namesti tri uporabniške vmesnike:

- Paper UI,
- Basic UI,
- HABPanel.

Basic UI in HABPanel sta namenjena mobilnim napravam in tabličnim računalnikom, mi pa smo uporabili Paper UI, ki je namenjen administraciji platforme in dodajanju vtičnikov. Po uspešni namestitvi lahko začnemo z uporabo platforme in uporabniških vmesnikov. Dostopni so na URL naslovu

http://localhost:8080, kjer lahko izberemo želen uporabniški vmesnik (slika 3.13).



Slika 3.13: Domača spletna stran platforme OpenHab2

Z uporabo Paper UI smo namestili vtičnik za MQTT komunikacijo imenovan MQTT Binding. Ker smo uporabili Watson IoT platformo, nismo mogli direktno povezati OpenHab platforme z Bluemixom. Zato smo na Raspberry Pi 3 namestili Mosquitto.

Mosquitto je produkt, ki nam omogoča postavitev MQTT posrednika sporočil, poleg tega pa ga lahko uporabimo kot prehodni vmesnik za povezavo z oblakom. Delovanje posrednika sporočil Mosquitto lahko nadzirano s spreminjanjem konfiguracijskih datotek. Napisali smo svojo konfiguracijo, katero posrednik sporočil uporabi za povezavo z Watson IoT platformo. Tako smo z uporabo Mosquittota povezali OpenHab2 z Watson IoT platformo. V naslednjem koraku smo se lotili povezave multi-senzorjev FireFly.

FireFly senzorji komunicirajo preko Bluetooth-a. Podjetje L-TEK elektronika d.o.o. je na spletni strani GitHub objavilo demo aplikacijo za povezavo s FireFly moduli. Njihovo rešitev smo izkoristili in predelali za naše potrebe. Programska oprema za povezavo s FireFly moduli je napisana v

Pythonu. Uporablja knjižnico *pygatt* za dostop do Bluetooth vmesnika. Z uporabo vmesnika program poišče vse FireFly module v dosegu in se z njimi poskuša povezati. V primeru neuspele povezave poskusi trikrat, nato preneha. Ko se vzpostavi povezava s FireFly modulom, se nastavi način delovanja.

FireFly moduli imajo svoj komunikacijski protokol. Z uporabo posebnih ukazov jim nastavimo način delovanja, beremo podatke s senzorjev ter senzorje prižigamo in ugašamo. V naši rešitvi smo uporabili način delovanja Continuous Response, kar pomeni, da FireFly zajema vrednosti senzorjev v določenem časovnem intervalu. Po izteku intervala FireFly pošlje programu na Raspberry Pi 3 vrednosti senzorjev.

Send this command for continuous measurement of sensors.

							Cmd Type	Time units	Time1	Time2
Cmd	ID1	ID2	ID3	ID1	ID2	ID3	1	U	T1	T2

Time units	U
Hours	1
Minutes	2
Seconds	3
100 Milliseconds	4

Time1	Time2
0-9	0-9

Slika 3.14: Primer ukaza za nadzor multi-senzorja FireFly **??**

Prejeti niz podatkov moramo pretvoriti tako, da dobimo vrednosti posameznih senzorjev. Določene vrednosti moramo pretvoriti v prave merske enote, saj drugače prejete vrednosti nimajo smisla. Po pretvorbi podatkov, program sestavi JSON objekt in ga pošlje v oblak.

Za povezavo in komunikacijo z Watson IoT platformo smo uporabili knjižnico mqtt-paho. Ker prejeti podatki vsebujejo enolični identifikator naprave, smo to izkoristili za nastavitev teme kamor program pošilja MQTT sporočila.

0	1	2	3	4	5	6	7	8	9	10	11
ID1	ID2	ID3	Active sensors	Gyro X1	Gyro X2	Gyro Y1	Gyro Y2	Gyro Z1	Gyro Z2	Acc X1	Acc X2

12	13	14	15	16	17	18	19	20	21	22	23
Acc Y1	Acc Y2	Acc Z1	Acc Z2	Mag X1	Mag X2	Mag Y1	Mag Y2	Mag Z1	Mag Z2	Temp H	Temp L

24	25	26	27	28	29	30	31	32			
Rel.H H	Rel.H L	Lum H	Lum L	BAT	IN/OUT	Ain	RSSI	CRC			

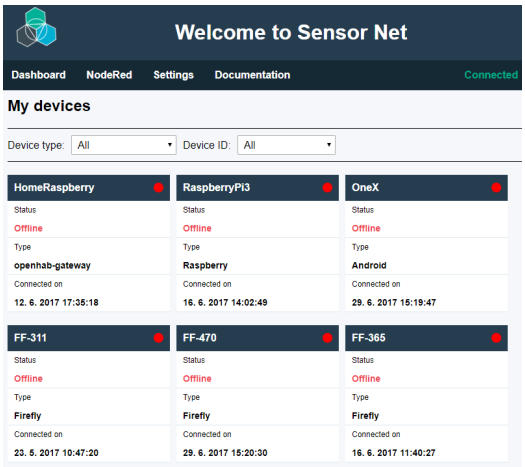
Slika 3.15: Predstavitev podatkov multi-senzorja FireFly

Poglavje 4

Demonstracija delovanja

Po končani implementaciji, smo imeli pripravljen celotni sistem. Preostala nam je postavitev multi-senzorjev FireFly v domačem okolju in povezava s prehodnim vmesnikom. Prvi multi-senzor smo zaradi napajanja priključili na Raspberry Pi 3, kjer teče prehodni vmesnik. Druge pa smo razporedili po hiši, napaja pa jih baterija.

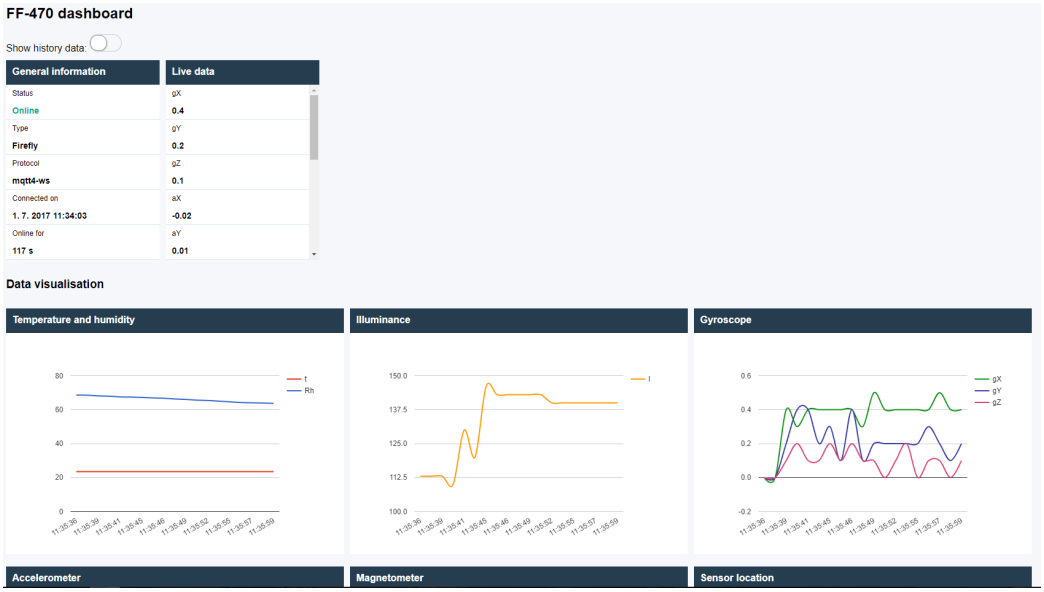
Spletna aplikacije SensorNet je dostopna na *https://sensor-net.mybluemix.net*. Pri odprtju spletne aplikacije nam ta prikaže domačo stran (slika 4.3). Domača



Slika 4.1: Domača stran spletne aplikacije SensorNet

stran prikazuje pregledno ploščo, kjer so prikazane vse naprave in prehodni

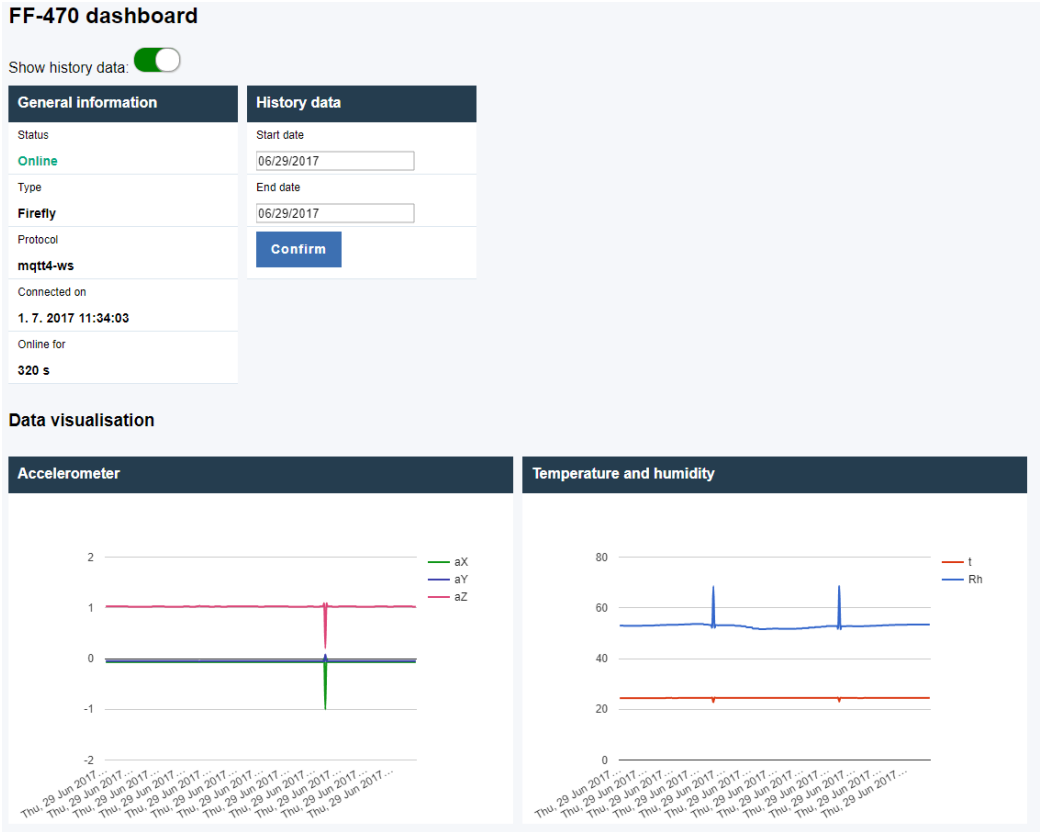
vmesniki, ki so se povezali z IoT platformo. Na pregledni plošči uporabnik vidi imena naprav (enolični identifikator), stanje in tip naprave, kdaj se je naprava nazadnje povezala ali pa čas vzpostavljen povezave. S klikom na napravo uporabniku prikažemo pregledno ploščo izbrane naprave. Tukaj lahko uporabnik izbira med vizualizacijo trenutnih in zgodovinskih podatkov. S kli-



Slika 4.2: Pregledna plošča naprave

kom na drsnik se prikaže nova kartica, kjer uporabnik izbere časovni interval prikaza zgodovinskih podatkov. Spletna aplikacija zgradi poizvedbo in preko programskega vmesnika podatkovne baze pridobi podatke. Podatki so tipa JSON zato jih spletna aplikacija pretvori v obliko, ki je berljiva grafom. Podatke shranimo v stanje aplikacije, ta pa vizualizira prejete podatke. Tako lahko uporabnik vizualizira stanje svojega doma. Preko spletne aplikacije ima uporabnik dostop do NodeRed urejevalnika. V urejevalniku NodeRed smo definirali pravila, na podlagi katerih sistem pošlje potisno sporočilo na uporabnikovo napravo. Tako sistem obvešča uporabnika o spremembi stanja doma.

Sami smo definirali pravila za obveščanje spremembe temperature pro-



Slika 4.3: Prikaz zgodovinskih podatkov

stora. Ko temperatura prostora preseže 25 stopinj Celzija sistem pošlje uporabniku potisno sporočilo. Poleg tega spremlja nadaljnje spremembe in ponovno pošlje potisno sporočilo, ko se temperatura poviša za dodatne 3 stopinje Celzija.

Poglavje 5

Možne razširitve

Razvili smo prototip sistema, ki pokaže, kako lahko hitro in enostavno povežemo različne IoT naprave z aplikacijo v oblaku, vizualiziramo podatke in opozarjamo uporabnika na določene dogodke. Kljub delujočemu prototipu ima sistem dokaj omejeno funkcionalnost, vendar sama arhitektura sistema dopušča enostavno razširitev in nadgradnjo sistema.

Prva možna razširitev je boljši izkoristek oblaka. Oblačna platforma Bluemix nam omogoča uporabo veliko različnih servisov. Če bi uporabili servise za analitiko, bi lahko analizirali podatke ter iz podatkov izluščili informacije in znanja. Ta bi lahko uporabili za gradnjo odločitvenih modelov. Pri demonstraciji delovanja smo prikazali primer, ko sistem javi uporabniku, da je temperatura prostora presegla določeno vrednost. Z uporabo analitike bi lahko napovedali, kdaj se bo naš prostor začel ohlajati. Če bi imeli v hiši pametni termostat, bi tega lahko povezali z aplikacijo v oblaku. Ta bi uporabila zgrajene odločitvene modele in preventivno začela ogrevati prostor. Tako privarčujemo pri porabi sredstev, saj je konstantno minimalno ogrevanje cenejše kot ogrevanje mrzlega prostora. V poletnem času, bi iste modele lahko uporabili za preprečitev segrevanje prostora. Sonce v določenem delu dneva sveti na določeni del hiše, kar povzroči, da se prostori v tem delu bolj segrejejo. Sistem bi spustil senčnike in tako preprečil pretirano segrevanje prostora. Z uporabo lokacijskih podatkov bi lahko sistem naučili, kdaj pridejo domov

družinski člani. Tako bi lahko sistem prevzel nadzor nad temperaturo hiše in jo optimalno prilagodil na željene temperaturne vrednosti. Na primer, v vročem poletnem dnevu bi uporabnik shladil domače okolje preden uporabnik prispe domov. Tako namesto v vroče stanovanje stopi v prostor prijetne temperature.

Z analitiko bi lahko povezali, servis za napovedovanje vremena. Tako bi sistem na podlagi vremenske napovedi, naučenih vzorcev in naših preferenc optimalno nadziral delovanje grelnih in hladilnih naprav. Uporabniku olajša delo, saj se s tem ne potrebuje ukvarjati in privarčuje denar.

Spletna aplikacija SensorNet v trenutni fazi omogoča vizualizacijo podatkov. Razširili, bi jo lahko z upravljanjem uporabljenih naprav. Vsaka naprava pošilja in prejema podatke na določeni temi. To bi lahko izkoristili za pošiljanje ukazov napravam. Spremenili bi lahko hitrost pošiljanja podatkov, izklopili ali prižgali določene naprave, kot so pametne luči. Vendar je lahko zamudno odpiranje aplikacije za vsako akcijo, ki jo želimo izvesti. Nadgradnja tega bi bila povezava pametnega telefona in pametne ure s sistemom. Tako bi lahko z uporabo glasovnih ukazov nadzirali celoten dom.

Opisali smo samo nekaj možnih scenarijev, s katerimi bi lahko nadgradili sistem. Področje pametnih domov in naprav interneta stvari se konstantno razvija, na trgu pa se pojavlja vse več različnih naprav. Menim, da bodo tekom par let naši domovi postali del interneta stvari, nadzirali pa jih bodo pametni sistemi. Tako bodo domovi skrbeli za nas, privarčevali na porabi, sami pa bomo odmislili del vsakdana in se lahko osredotočili na ostale stvari.

Literatura

[1] 3G. Dosegljivo: https://en.wikipedia.org/wiki/3G, 2017. [Dostopano 21. 7. 2017].

[2] 4G. Dosegljivo: https://en.wikipedia.org/wiki/4G, 2017. [Dostopano 21. 7. 2017].

[3] Amazon Echo. Dosegljivo: https://www.amazon.co.uk/Amazon-SK705DI-Echo-Black/dp/B01GAGVIE4#, 2017. [Dostopano 2. 7. 2017].

[4] Amazon Echo Family. Dosegljivo: https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011, 2017. [Dostopano 2. 7. 2017].

[5] Amazon Web Services. Dosegljivo: https://aws.amazon.com/, 2017. [Dostopano 12. 6. 2017].

[6] ANT. Dosegljivo: https://en.wikipedia.org/wiki/ANT_(network), 2017. [Dostopano 21. 7. 2017].

[7] Apache CouchDB. Dosegljivo: http://couchdb.apache.org/, 2017. [Dostopano 5. 7. 2017].

[8] Apple HomeKit. Dosegljivo: https://www.apple.com/lae/ios/home/, 2017. [Dostopano 2. 7. 2017].

[9] Apple HomeKit accessories. Dosegljivo: https://www.apple.com/lae/ios/home/accessories/, 2017. [Dostopano 2. 7. 2017].

[10] ARMmbed. Dosegljivo: https://www.mbed.com/en/, 2017. [Dostopano 17. 6. 2017].

[11] AWS still owns the cloud. Dosegljivo: https://techcrunch.com/2017/02/02/aws-still-owns-the-cloud/, 2017. [Dostopano 17. 6. 2017].

[12] Bluemix. Dosegljivo: https://www.ibm.com/cloud-computing/bluemix/, 2017. [Dostopano 7. 5. 2017].

[13] Bluetooth. Dosegljivo: https://en.wikipedia.org/wiki/Bluetooth, 2017. [Dostopano 21. 7. 2017].

[14] Bluetooth Low Energy. Dosegljivo: https://en.wikipedia.org/wiki/Bluetooth_Low_Energy, 2017. [Dostopano 21. 7. 2017].

[15] Bluetooth Vs. Bluetooth Low Energy: What's The Difference? Dosegljivo: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy, 2017. [Dostopano 21. 7. 2017].

[16] Cloudant NoSQL DB overview. Dosegljivo: https://console.bluemix.net/docs/services/Cloudant/cloudant.html#overview, 2017. [Dostopano 5. 7. 2017].

[17] Eclipse SmartHome. Dosegljivo: https://eclipse.org/smarthome/, 2017. [Dostopano 7. 5. 2017].

[18] Ethernet IoT Starter Kit. Dosegljivo: https://developer.mbed.org/platforms/IBMEthernetKit/, 2017. [Dostopano 21. 7. 2017].

[19] Firefly specifications. Dosegljivo: https://firefly-iot.com/product/ff1502-sensor-ble/, 2017. [Dostopano 2. 7. 2017].

[20] FlyTag – Communication Protocol. Dosegljivo: https://firefly-iot.com/wp-content/uploads/2016/05/FlyTag_Communication_V1.0.0.pdf, 2017. [Dostopano 6. 7. 2017].

[21] Google Assistant. Dosegljivo: https://en.wikipedia.org/wiki/Google_Assistant, 2017. [Dostopano 21. 7. 2017].

[22] Google Cloud. Dosegljivo: https://cloud.google.com/, 2017. [Dostopano 12. 6. 2017].

[23] Google Home. Dosegljivo: https://madeby.google.com/home/, 2017. [Dostopano 21. 7. 2017].

[24] How Much Does it Cost to Install a Home Automation System? Dosegljivo: http://www.homeadvisor.com/cost/electrical/install-or-repair-a-home-automation-system/, 2017. [Dostopano 21. 7. 2017].

[25] HTTP. Dosegljivo: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol, 2017. [Dostopano 21. 7. 2017].

[26] HTTPS. Dosegljivo: https://en.wikipedia.org/wiki/HTTPS, 2017. [Dostopano 21. 7. 2017].

[27] IBM-MBED IOT STARTER KIT). Dosegljivo: http://uk.farnell.com/arm/ibm-mbed-iot-starter-kit/dev-board-ethernet-iot/dp/2468176, 2017. [Dostopano 21. 7. 2017].

[28] IoT Starter Kit – Ethernet Edition Released. Dosegljivo: https://developer.mbed.org/blog/entry/IoT-Starter-Kit-Ethernet-Edition/, 2017. [Dostopano 21. 7. 2017].

[29] L-TEK Elektronika. Dosegljivo: http://www.l-tek.si/, 2017. [Dostopano 17. 6. 2017].

[30] MQTT and CoAP, IoT Protocols. Dosegljivo: https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php, 2017. [Dostopano 18. 6. 2017].

[31] MQTT.org. Dosegljivo: http://mqtt.org/, 2017. [Dostopano 17. 6. 2017].

[32] NodeRed. Dosegljivo: https://nodered.org/about/, 2017. [Dostopano 20. 6. 2017].

[33] OASIS MQTT version 3.1.1. Dosegljivo: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, 2017. [Dostopano 24. 5. 2017].

[34] OpenHab. Dosegljivo: https://www.openhab.org/, 2017. [Dostopano 7. 5. 2017].

[35] Philips Hue specifications. Dosegljivo: http://www2.meethue.com/en-us/productdetail/philips-hue-bridge#specifications, 2017. [Dostopano 2. 7. 2017].

[36] Philips Hue specifications. Dosegljivo: http://www2.meethue.com/en-us/productdetail/philips-hue-bridge#overview, 2017. [Dostopano 2. 7. 2017].

[37] Sensor. Dosegljivo: https://en.wikipedia.org/wiki/Sensor, 2017. [Dostopano 17. 6. 2017].

[38] SensorTag teardown. Dosegljivo: http://www.ti.com/ww/en/wireless_connectivity/sensortag/tearDown.html#main, 2017. [Dostopano 2. 7. 2017].

[39] SORACOM Overview. Dosegljivo: https://soracom.io/en/overview/, 2017. [Dostopano 21. 7. 2017].

[40] The SensorTag story. Dosegljivo: http://www.ti.com/ww/en/wireless_connectivity/sensortag/index.html#main, 2017. [Dostopano 2. 7. 2017].

[41] Thread (network protocol). Dosegljivo: https://en.wikipedia.org/wiki/Thread_(network_protocol), 2017. [Dostopano 21. 7. 2017].

[42] TI SensorTag 2 Power consumption analysys. Dosegljivo: http://mobilemodding.info/2015/06/ti-sensortag-2-power-consumption-analysys/, 2017. [Dostopano 2. 7. 2017].

[43] What is Zigbee? Dosegljivo: http://www.zigbee.org/what-is-zigbee/494-2/, 2017. [Dostopano 21. 7. 2017].

[44] Why we made thread. Dosegljivo: http://threadgroup.org/About, 2017. [Dostopano 21. 7. 2017].

[45] Z-Wave Alliance. Dosegljivo: http://z-wavealliance.org/z-wave-alliance-overview/, 2017. [Dostopano 2. 7. 2017].

[46] Zigbee. Dosegljivo: https://en.wikipedia.org/wiki/Zigbee, 2017. [Dostopano 21. 7. 2017].