Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksander Tomić

Prilagodljiv sistem za zajem podatkov iz zdravstvenih merilnih naprav

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge preučite področje zajema podatkov iz IoT medicinskih merilnih naprav. Podrobneje analizirajte delovanje protokola BTLE, ki se najpogosteje uporablja za komunikacijo z zdravstvenimi merilnimi napravami. Preučite vse vidike komunikacijskega protokola. Na osnovi tega izdelajte zasnovo in implementirajte programski vmesnik (API) za upravljanje z medicinskimi merilnimi napravami. Delovanje sistema demonstrirajte z uporabo poljubnega odjemalca, npr. OpenHab.

Zahvaljujem se mentorju prof. dr. Marku Bajcu za usmerjanje in podporo pri izdelavi diplomskega dela. Zahvaljujem se tudi moji družini, ki mi je tekom študija stala ob strani, me spodbujala in mi bila v podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IoT	Internet of Things	internet stvari
IT	Information Technology	informacijska tehnologija
RFID	Radio Frequency Identification	radiofrekvenčna identifikacija
WSN	Wireless sensor network	brezžična senzorska omrežja
BT	Bluetooth	Bluetooth
BLE	Bluetooth Low Energy	Bluetooth Low Energy
APH	Augmented Personalized Healthcare	obogateno osebno zdravstvo
(D)DoS	(Distributed) Denial Of Service	(porazdeljena) zavrnitev storitve
SoC	System on Chip	sistem na čipu
CPE	central processing unit	Centralna Procesna Enota
PDU	Protocol Data Unit	podatkovni paketi določene BLE plasti
MiTM	Man In The Middle attack	napad s posrednikom
OH2	openHAB 2	sistem za avtomatizacijo pametnega doma
API	Application Program Interface	programski vmesnik
SDK	Software Development Kit	paket za razvoj programske opreme
IOTA	IOTA	tehnologija porazdeljene evidence zapisov
MAM	Masked Authentication Messaging	protokol za overjeno sporočanje podatkov nad omrežjem IOTA

Povzetek

Naslov: Prilagodljiv sistem za zajem podatkov iz zdravstvenih merilnih naprav

Avtor: Aleksander Tomić

Na področju zdravstva se vse več pozornosti namenja razvoju osebnih zdravstvenih rešitev, ki temeljijo na kontinuiranem zajemanju zdravstveno relevantnih podatkov. Glavni vir podatkov posameznika predstavljajo zdravstvene merilne naprave in okoljski senzorji. V diplomskem delu smo spoznali osnovne koncepte IoT in trenutne trende na področju zdravstva. Ker večina zdravstvenih merilnih naprav podpira protokol BLE, smo ga v nadaljevanju podrobneje predstavili. Za lažje zbiranje in upravljanje podatkov smo razvili sistem HealthySync, ki omogoča integracijo naprav BLE v okolje pametnega doma OH2. Sestavljen je iz aplikacije AndroidThings in vtičnika OH2. Pri tem ima končni uporabnik popoln nadzor nad sistemom in podatki, ki jih lahko z izbranimi osebami varno deli po porazdeljenem omrežju IOTA.

Ključne besede: aplikacija Android Things, vtičnik Openhab 2, BLE, specifikacije GATT, osebno zdravstvo, IOTA, MAM.

Abstract

Title: Configurable system for acquiring data from medical measuring devices

Author: Aleksander Tomić

In healthcare, more and more attention is being paid to developing personalized healthcare solutions. They are primarily based on patient-generated health data (PGHD). The main source of those are medical measuring devices and environmental sensors. In this diploma thesis, we learn about the basic concepts of IoT and the current trends in health care. Since most medical devices support BLE technology, the thesis describes some important features that are part of the BLE core specification. In order to provide simple data collection and management, I have developed the HealthySync system, which enables the integration of BLE devices into the OH2 smart home environment. It consists of the AndroidThings application and the OH2 plugin. The end-user has complete control of the system and the data, which can be shared with selected parties safely through the distributed IOTA network.

Keywords: Android Things application, Openhab 2 plugin, BLE, GATT specifications, personalized healthcare, IOTA, MAM.

Poglavje 1

Uvod

Kot posledica tehnološkega napredka in razvoja je na trgu na voljo vse več pametnih naprav oziroma objektov (angl. smart objects). Ti predstavljajo osnovne gradnike interneta stvari (IoT), ki omogočajo interakcijo digitalnega in fizičnega sveta. To so v veliko primerih že znane, obstoječe naprave (v zdravstvu npr.: merilec krvnega tlaka, termometer, glukometer, spirometer, ipd.), ki jim je dodana zmožnost komuniciranja (upravljanje naprave, prenos podatkov) z uporabo različnih tehnologij (nekaj jih bomo predstavili v odlomku 2.2, kasneje pa se bomo bolj osredotočili na tehnologijo BLE, predstavljeno v poglavju 3). Med njimi so tudi nosljive naprave in okoljski senzorji, ki predstavljajo pomemben vir podatkov, iz katerih se je mogoče z uporabo sodobnih tehnik podatkovnega rudarjenja, strojnega učenja in statistike marsikaj naučiti.

Žal številna področja zaostajajo za aktualnimi trendi in ne izkoriščajo danih tehnoloških potencialov. Mednje zagotovo spada področje zdravstva, ki je predvsem zaradi ekonomskih dejavnikov deležno številnih sprememb. Namesto običajne, kurativne medicine (za katero je značilen reaktivni pristop, osredotoča se na zdravljenje bolezni), se vse več pozornosti namenja preventivni medicini (za katero je značilen proaktiven pristop, osredotoča se na pravočasno ukrepanje in preprečevanje nastanka bolezni). Cilj drugačnega pristopa in uporabe tehnologij IoT v zdravstvu je predvsem znižati stroške,

povečati kakovost življenja in podaljšati življensko dobo posameznika [24].

Izdatki zdravstva so po podatkih Eurostata ([11]) v Sloveniji leta 2016 znašali 3422,72 milijona evrov oziroma 8,47 odstotkov bruto domačega proizvoda (% BDP). Od sosednjih držav sta imeli več izdatkov (v % BDP) le Avstrija in Italija (10,44 in 8,94), medtem ko sta Madžarska in Hrvaška namenili zgolj 7,36 in 7,18 % BDP. Podatki kažejo, da so se izdatki zdravstvenih storitev v vseh zgoraj omenjenih državah v zadnjih nekaj letih (2014–2016) ves čas povečevali. S tovrstnim problemom se srečuje tudi ZDA, kjer so po ocenah CMS (centra za zdravstvene in socialne storitve, angl. „Center for Medicare and Medicaid Services“) izdatki za leto 2016 znašali 3,4 bilijona ameriških dolarjev oziroma 18,26% BDP ([10], [17]). Projekcije kažejo, da bodo stroški v letu 2025 dosegli 5,5 bilijona ameriških dolarjev oziroma približno 20% BDP. Prav tako se pričakuje, da bo leta 2019 kar 87% zdravstvenih organizacij uporabljalo tehnologije IoT, od tega kar 73 % z namenom zmanjšanja stroškov, 64 % pa za spremljanje pacientov ([24]).

Stroške bi lahko zmanjšali z digitalizacijo in optimizacijo zdravstvenih procesov, z uporabo tehnologij IKT/IoT in z razbremenitvijo javnega zdravstva (večji del bremena bi moral prevzeti pacient, a bi moral imeti ustrezno zdravstveno-informacijsko podporo). V stroki je tako vse bolj priljubljena vizija obogatenega osebnega zdravstva (APH), ki s pomočjo umetne inteligence iz velike množice podatkov izlušči zgolj tiste informacije (predvsem v obliki opozoril), na podlagi katerih je mogoče nadaljnje ukrepanje. Ključnega pomena pri tem sta vseprisotno računalništvo (angl. *ubiquitous computing*) in pacient, ki s pomočjo nosljivih naprav (angl. *wearables*, *wearable computing/technology*) ves čas, tudi izven nadzorovanega kliničnega okolja, zbira pomembne zdravstvene podatke (angl. *Patient Generated Health Data*, krajše PGHD). Da gre za res aktivno gibanje se kaže tudi v ocenah in finančnih naložbah. Pričakuje se, da bo globalna industrija pametnega zdravstva do leta 2020 dosegla 169,32 bilijona ameriških dolarjev, kar je v primerjavi z letom 2015, 112,82 bilijona ameriških dolarjev več ([19]).

V okviru diplomskega dela smo razvili sistem HealthySync, ki omogoča

preprost zajem strukturiranih podatkov in meritev naprav BLE. Sestavljata ga aplikacija AndroidThings, ki je namenjena prehodu (angl. *gateway*) z ustrezno strojno podporo, in vtičnik OH2, ki omogoča integracijo sistema v okolje pametnega doma ter možnost shranjevanja zajetih sej v decentralizirano omrežje IOTA. Pacient lahko zdravstvene meritve in okoljske podatke (kot so npr. temperatura, vlažnost, kakovost zraka) lažje upravlja in jih po želji deli z ostalimi pomembnimi institucijami.

Vsebina diplomskega dela je sestavljena iz uvodnega dela, ki je namenjen predstavitvi IoT in trenutnih trendov na področju osebnega zdravstva. Sledi drugi del, kjer je predstavljena tehnologija BLE, ki jo zaradi številnih prednosti uporablja veliko zdravstvenih merilnih naprav. V tretjem delu je predstavljena programska rešitev HealthySync, ki zajema predstavitev aplikacije, vtičnika OH2 in tehnologije porazdeljene evidence zapisov IOTA. Sledi zaključek, kjer so predstavljeni sklepne ugotovitve in problemi, s katerimi smo se srečevali med razvojem programske rešitve.

Poglavje 2

Internet stvari

„Internet stvari“ je na področju IT zagotovo ena od pogosto uporabljenih besednih zvez (angl. buzzwords). Namen tega poglavja je bralca seznaniti z definicijo oziroma pomenom IoT, predstaviti nekatere ključne tehnologije, aktualna področja, izive in trende.

2.1 Definicija in struktura IoT

Za izraz „internet stvari“ v literaturi ni mogoče najti splošno sprejete definicije. Termin je leta 1999 prvič uporabil Kevin Auston, takrat izvršni direktor laboratorija Auto-ID na MIT ([21]). Obstaja sicer veliko različnih definicij, vsem pa je skupna ideja, da so bistvo interneta prvotno predstavljali podatki, ki smo jih ustvarjali ljudje, kasneje pa podatki, ki jih ustvarjajo *stvari*. Stvari so lahko osebe ali poljubni objekti fizičnega sveta, ki jih je mogoče med seboj razlikovati, kar poleg tehnološko naprednih izdelkov in elektronskih naprav vključuje tudi druge (ne elektronske) stvari, kot so npr. oblačila, hrana, pohištvo, zdravstveni pripomočki, zdravila, živali, itd.

IoT (pravimo mu tudi medomrežje stvari) je globalna dinamična omrežna infrastruktura, ki se je s pomočjo standardov in interoperabilnih komunikacijskih protokolov zmožna samoorganizirati, kjer imajo fizične in virtualne *stvari* svoje identitete, fizične atribute in virtualne osebnosti in se preko in-

teligentnih vmesnikov povezujejo v informacijsko omrežje [2].

IoT lahko v osnovi razdelimo na tri dele. Prvi del zajema fizične objekte in tehnologije, ki omogočajo stik fizičnega in digitalnega sveta (npr.: RFID, kode QR, računalniški sistemi, tipala, aktuatorji, komunikacijski protokoli). Drugi del zajema ustrezno omrežno podporo, ki omogoča komunikacijo po mobilni in žični infrastrukturi. Tretji del predstavljajo storitve, ki omogočajo interoperabilno delovanje različnih sistemov IoT (npr.: zdravstva, prometa, oskrbovanih domov ipd.).

2.2 Ključne tehnologije

Ključnega pomena za razvoj IoT je tehnologija RFID, ki omogoča preprosto prepoznavo in sledenje posameznim fizičnim objektom. Zaradi številnih prednosti sledenja (boljšega nadzora, optimizacije procesov, zmanjšanja napak in stroškov) so tehnologijo RFID začeli množično uporabljati že v 80. letih prejšnjega stoletja, in sicer v logistiki, farmaciji, proizvodnji, trgovinah in pri upravljanju zalog [33]. Naslednji pomemben mejnik predstavlja razvoj brezžičnih senzorskih omrežij (WSN), ki s pomočjo inteligentnih senzorjev omogoča spremljanje okolice in zaznavanje različnih fizikalnih veličin. Uporabljajo se pri spremljanju okolja, zdravstva, industrijskih procesov, cestnega prometa, ipd. Iz praktičnih potreb so se razvila tudi ostala področja in tehnologije IoT, kot so npr.: črtne kode, pametni telefoni, družabna omrežja, računalništvo v oblaku, ipd. Poleg **prepoznavanja** (identifikacije) in **zaznavanja** (merjenja) je bistvenega pomena za IoT tudi **komunikacija**.

Komunikacijske protokole in standarde glede na način prenosa delimo na:

- žične (npr.: USB, Ethernet, RS-232, RS-485, UART, USART,...) in
- brezžične (npr.: Bluetooth, LE, WiFi, RF, ZigBee, Z-Wave, ANT+...).

V nadaljevanju bomo predstavili nekaj aktualnih brezžičnih tehnologij, ki so zaradi številnih prednosti množično prisotne v svetu IoT.

2.2.1 Radiofrekvenčna identifikacija

Radiofrekvenčna identifikacija (RFID) je cenovno ugodna in energijsko varčna brezžična tehnologija, ki se primarno uporablja za identifikacijo fizičnih objektov. Sistem RFID sestavljajo elektronske oznake (angl. tags), ki hranijo podatke in bralniki, ki z oznakami komunicirajo. Za komunikacijo se v praksi uporabljajo različne frekvence radijskega valovanja, ki jih predpisuje standard ISO/IEC 18000. Med njimi je tudi območje ultra visoke frekvence (krajše UHF), ki skupaj s številnimi izboljšavami elektronskih oznak (kot je zmožnost računanja in zaznavanja [8]) omogoča razvoj številnih, energijsko učinkovitih rešitev. Prednost tovrstnih sistemov UHF RFID sta večja pokritost (15-25 metrov) in nizka poraba energije v primerjavi s konkurenčnimi tehnologijami, kot je npr. WSN.

2.2.2 Brezžična senzorska omrežja

Brezžična senzorska omrežja (WSN) so množica avtonomnih, cenovno ugodnih naprav (tipal, ki jim rečemo tudi senzorska vozlišča), ki omogočajo zaznavanje in zbiranje različnih podatkov iz okolja, kot so npr.: temperatura, vlažnost zraka, zračni tlak, hitrost vetra ipd. [31]. Senzorska vozlišča se povezujejo v velika brezžična omrežja (tudi po več tisoč naprav hkrati), podatki (meritve) pa se običajno zbirajo v posebnih vozliščih (t.i. prehodih) na robu WSN. V primerjavi z UHF RFID imajo WSN bistveno večji doseg (do 100 metrov na prostem [8]), podpirajo različne topologije omrežij, a so energijsko bolj potratna in posledično zahtevnejša za vzdrževanje, saj se baterije hitreje izpraznijo.

2.2.3 Bluetooth

Bluetooth je cenovno ugodna in energijsko varčna tehnologija krajšega dosega (10–100 metrov [21]), ki je bila prvotno ustvarjena kot alternativa serijskim podatkovnim povezavam [18]. Je del skupine brezžičnih osebnih omrežij (angl. Wireless Personal Area Network, krajše WPAN), ki ga obravnava spe-

cifikacija standarda IEEE 802.15.1 in omogoča prenos podatkov do 1Mb/s (Basic Rate). Naprave se med seboj povezujejo v *ad-hoc* omrežja (*piko-omrežja*, angl. *piconets*), ki jih sestavlja primarna naprava (gospodar, angl. master) in do sedem sekundarnih naprav (sužnjev, angl. slaves).

2.2.4 ZigBee

ZigBee je brezžična tehnologija, ki je bila razvita z namenom nadgradnje in izboljšav WSN [21]. Zaradi številnih lastnosti (zanesljivosti, skalabilnosti, varnosti, prilagodljivosti, preprostosti uporabe in podpore različnih omrežnih topologij) je pogosto uporabljena pri avtomatizaciji pametnih domov, v zdravstvu, v industrijskih obratih, ipd. Temelji na specifikaciji IEEE 802.15.4 in je prav tako del skupine WPAN, z nekoliko nižjo hitrostjo prenosa (do 250 kb/s) in povprečnim dosegom od 10 do 20 metrov oziroma do 100 metrov na prostem (brez ovir) [32].

2.3 IoT na področju zdravstva

IoT predstavlja temelj digitalnega zdravstva [30], ki z uporabo številnih informacijsko-komunikacijskih tehnologij omogoča učinkovitejše zdravstvene storitve. Za zbiranje podatkov posameznika se pogosto uporabljajo nosljive naprave in senzorji. Z njimi lahko spremljamo aktivnosti posameznika in zajemamo pomembne okoljske podatke. Uporaba tovrstnih podatkov je zanimiva tudi za področje digitalnega zdravstva (preventivne medicine), saj skupaj z osebnimi genetskimi podatki omogočajo zgodnje odkrivanje posameznih zdravstvenih tveganj in preprečevanje potencialnega nastanka bolezni [4], [16].

Podatki pacienta se običajno hranijo in obdelujejo v okviru različih zdravstvenih ustanov (zdravstvenih domov, bolnic, specialističnih klinik, zavarovalnic ipd). Te organizacije opravljajo večino zdravstvenih procesov (zajem podatkov, preventivno ukrepanje, postavljanje diagnoz, opravljanje posegov, rehabilitacijo in zdravljenje kroničnih bolezni). Vse več znanja in podatkov

se tako nabira v centraliziranih IT sistemih, kar posledično vpliva na večjo obremenjenost institucij in zdravstvenega osebja.

Uporaba nosljivih naprav v okviru IoT omogoča, da se v središče porazdeljenega zdravstveno-informacijskega sistema postavi uporabnika oziroma pacienta, ki je glavni vir podatkov. Podatki, ki jih ustvarja pacient, so v primerjavi s podatki zdravstvenih ustanov bistveno bolj pogosti in raznoliki. S pomočjo teh podatkov lahko zmanjšamo število potrebnih zdravstvenih obiskov in s tem delno razbremenimo zdravstveni sistem, omogočimo boljši pregled nad dejanskim dogajanjem, na podlagi katerega lahko lažje ukrepamo in spremljamo napredek morebitnega zdravljenja. IoT omogoča tudi osnovo za razvoj rešitev APH.

APH je vizija [24], [5] obogatenega osebnega zdravstva, ki izkorišča številne podatke (elektronske zdravstvene zapise, podatke spletnih strani, mobilnih aplikacij in socialnih omrežij, aktivnosti ljudi in druge okoljske podatke), medicinska znanja in tehnike umetne inteligence za izboljšanje zdravja in kakovosti življenja. Primer ogrodja APH predstavlja platforma kHealth (Knowledge-enabled Healthcare [20]), ki so jo razvili v okviru raziskovalnega centra Knoesis. Omogoča dodajanje kontekstualnih anotacij, integracijo in interpretacijo senzorskih in mobilnih podatkov posameznika. Pri tem se uporabljajo specifične baze medicinskih znanj. Razvite so bile tudi namenske mobilne aplikacije za astmo, demenco, akutno poslabšanje srčnega popuščanja (ADHF) in cirozo jeter.

Poleg zbiranja in obdelave podatkov se v zdravstvu za podporo različnih procesov pogosto uporablja tehnike sledenja, identifikacije in avtentikacije.

2.4 Trenutni izzivi

2.4.1 Standardizacija

Dejstvo je, da se zaradi številnih tehnoloških napredkov v internet povezuje vse več raznolikih naprav. Zaradi različnih potreb so se v industriji oblikovale številne konkurenčne tehnologije, ki niso interoperabilne. Z namenom,

da končnim uporabnikom (ki niso več samo osebe, temveč tudi naprave in sistemi) omogočimo visoko kakovost storitev, je treba tudi na področju IoT razviti ustrezne standarde.

Standardizacija je za uspeh IoT ključnega pomena, saj omogoča interoperabilnost, kompatibilnost, zanesljivost in učinkovitost delovanja na globalni ravni ([33], [7]). Predvsem zaradi potencialnih ekonomskih koristi se za razvoj standardov poteguje veliko držav in organizacij. Pri razvoju številnih standardov na področju IoT sodelujejo naslednje organizacije ([33], [1], [6]):

- Mednarodna telekomunikacijska zveza (krajše ITU),
- Mednarodna komisija za elektrotehniko (krajše IEC),
- Mednarodna organizacija za standardizacijo (krajše ISO),
- Inštitut inženirjev elektrotehnike in elektronike (krajše IEEE),
- Evropski odbor za standardizacijo v elektrotehniki (krajše CENELEC),
- Kitajski inštitut za standardizacijo elektronike (krajše CESI) in
- Ameriški državni inštitut za standarde (krajše ANSI).

Prav tako je bilo v praksi oblikovanih več predlogov za arhitekturo IoT, kar je najverjetneje posledica različnih pogledov in znanj vpletenih deležnikov. Tudi tukaj se je treba uskladiti in s tem omogočiti dobro izhodišče za nadaljni razvoj aplikacij in storitev na področju IoT.

2.4.2 Sprejemanje kompromisov

Za naprave, ki delujejo na baterije, je energijska varčnost ključnega pomena. To so na primer nosljive naprave (ura, pedometer, glukometer, merilec srčnega utripa ipd.) in okoljski senzorji (merilnik kakovosti zraka, senzor gibanja, merilnik temperature ipd.). Programska rešitev, ki smo jo razvili v okviru diplomskega dela, se osredotoča predvsem na nizko porabo energije, pri čemer se uporablja protokol BLE. Pri tem smo omejeni glede možne topologije (ena centralna naprava, ki komunicira z perifernimi napravami), skalabilnosti (največ 7 perifernih naprav) in prepustnosti (do 1 Mb/s). Vendar pa te omejitve ne predstavljajo težave za občasno prenašanje manjših količin

podatkov (meritev). Poleg energijske varčnosti so v industriji pomembni tudi številni drugi dejavniki.

Spodaj bomo tabelarično predstavili nekaj ključnih dejavnikov, ki vplivajo na ceno razvoja industrijskih aplikacij. Zaradi teh se v praksi (na račun funkcionalnosti) pogosto sprejemajo številni kompromisi [33].

Dejavnik	**Opis (primer)**
Energija	Kako dolgo lahko naprava IoT deluje z omejenim napajanjem?
Latenca	Koliko časa je potrebnega za širjenje in obdelavo informacij?
Prepustnost	Kolikšna je maksimalna količina podatkov, ki jih lahko naenkrat prenesemo čez omrežje?
Skalabilnost	Koliko naprav je mogoče podpreti?
Topologija	Kakšne so možnosti poteka komunikacija med deležniki (kdo komunicira s kom)?
Varnost	Kako varna je uporaba?

Tabela 2.1: Pomembni dejavniki za razvoj industrijskih aplikacij IoT ([33], [12])

2.4.3 Varnost in zasebnost

Zagotavljanje varnosti in zasebnosti za področje IoT predstavlja velik problem, saj ga sestavlja veliko število raznolikih naprav, storitev, tehnologij in sistemov. Trenutno obstoječe storitve in sistemi so v veliki meri odvisni od ljudi, ki jih upravljajo (skrbijo za nemoteno delovanje in varnost), in imajo različne varnostne zahteve (mehanizme za overjanje in zaščito podatkov, zaščito in kontrolo dostopa, varovanje zasebnosti, uveljavljanje številnih pravil (npr. zasebnosti in varnosti) ipd.) [25]. V prihodnosti se pričakuje, da bodo naprave delovale avtonomno (brez človeške interakcije), za kar je treba razviti ustrezno infrastrukturo in poskrbeti za različne varnostne mehanizme, kot so zaščita pred napadi DoS/DDoS, zmožnost samoorganizacije in obrambe pred splošno znanimi napadi (uporaba umetne inteligence in strojnega učenja), možnost varnega oddaljenega nadzora, uporaba ustrezne kriptografije (tudi

za omejene naprave) ipd.

Veliko vsakdanjih naprav (npr. pametni telefoni in nosljive naprave) zajema in obdeluje osebne podatke, pogosto tudi brez naše vednosti. Zmožnost samodejne komunikacije pametnih objektov predstavlja potencialno nevarnost za našo prihodnost [6], saj omogoča osnovne mehanizme za nadzor in sledenje. Tak primer predstavljajo tudi naprave, ki z okolico komunicirajo po brezžičnih omrežjih in pri tem razkrivajo svoje identifikacijske podatke (npr. BT, BLE in RFID). Tudi če tehnologija omogoča varno komunikacijo (npr. šifriranje podatkov z uporabo simetrične kriptografije), je lahko zasebnost uporabnika ogrožena. To lahko predstavimo s primerom uporabe pametnega telefona v nakupovalnem centru. Uporabnik ima na pametnem telefonu vklopljen BT (v načinu oglaševanja), medtem ko so prostori nakupovalnega centra opremljeni s sprejemniki BT in pod video nadzorom. Javni MAC naslov pametnega telefona (oddajnik BT) je viden vsem napravam v okolici (sprejemnikom BT) in omogoča identifikacijo in sledenje s pomočjo ustrezne telekomunikacijske infrastrukture. Ko se za določen MAC naslov pridobijo podatki o identiteti lastnika (za najbližjo napravo BT se npr. na blagajni zabeleži identiteta uporabnika, ko uporabi nakupovalno kartico ali pa se identiteta znanega uporabnika vnese na podlagi videnega videoposnetka), ga je mogoče spremljati in na podlagi zbranih podatkov ugotavljati njegove navade, interese, interakcije z drugimi ljudmi in napravami ipd.

Nekatere tehnologije vsebujejo mehanizme za preprečevanje tovrstnega problema. Tehnologija BLE za ta namen omogoča uporabo mehanizma zasebnih MAC naslovov, ki se periodično spreminjajo (predstavljeni so v 3.1.6), a je uporaba opcijska. Osnovni gradniki IoT so običajno preproste naprave, z omejeno računsko močjo in majhno količino notranjega pomnilnika. Prav zaradi teh omejitev se pri razvoju pogosto sprejemajo številni kompromisi, ki vplivajo na varnost končnih rešitev.

Poglavje 3

Bluetooth Low Energy

Bluetooth Low Energy (krajše BLE) je vse pogosteje uporabljena brezžična tehnologija, ki jo razvija skupina „Bluetooth Special Interest Group“ (krajše Bluetooth SIG). V primerjavi s klasičnim BT, ki je primarno namenjen za prenos večje količne podatkov po brezžičnem omrežju, se BLE osredotoča na občasne prenose, z nizko porabo energije. Tehnologija BLE je bila prvič definirana v okviru specifikacije „Bluetooth Core Specification 4.0“ (krajše CS 4.0), ki so ji od tedaj sledile še tri različice (CS 4.1, CS 4.2 in CS 5.0).

V okviru diplomskega dela je bila uporabljena četrta različica BLE, ki jo bomo v nadaljevanju tudi obravnavali. Tehnologija BLE je pred različico 5.0 omogočala zgolj komunikacijo v okviru posameznih omrežij (angl. single-hop), medtem ko so to vrzel zapolnjevale alternativne tehnologije, kot so npr.: ZigBee, 6LoWPAN in Z-Wave, ki omogočajo komunikacijo po več omrežjih (angl. multi-hop) [15]. V nadaljevanju bomo predstavili sklad protokola BLE in opisali značilnosti posameznih plasti.

3.1 Sklad BLE

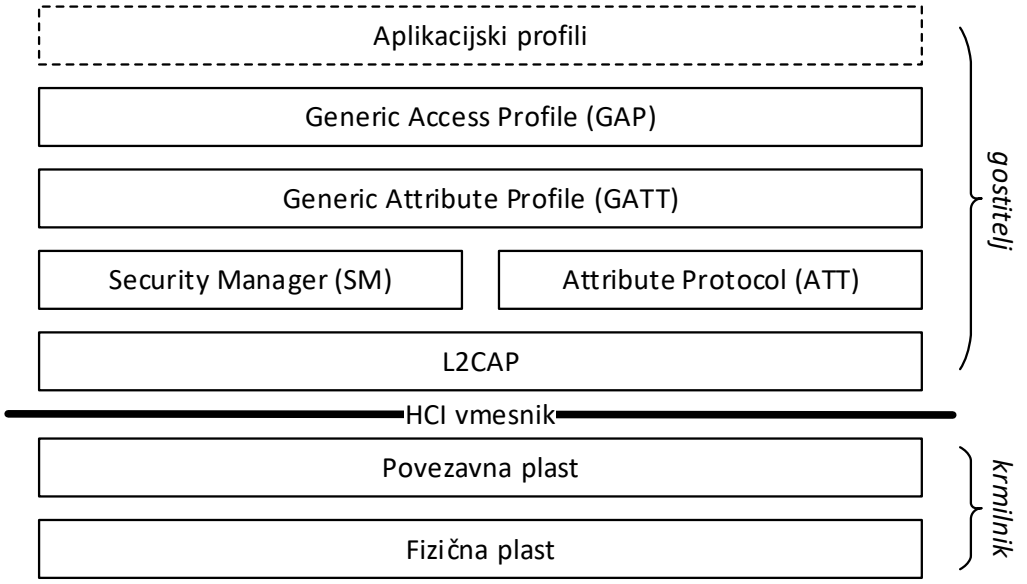
Sklad protokola BLE je sestavljen iz krmilnika (angl. the Controller) in gostitelja (angl. the Host), ki med seboj komunicirata s pomočjo standardiziranega vmesnika HCI (angl. Host Controller Interface).

Krmilnik zajema fizično in povezavno plast in je običajno implementiran kot del sistema na čipu (angl. System on Chip, krajše SoC), z vgrajeno radijsko napravo (angl. radio).

Gostitelj je običajno implementiran kot del operacijskega sistema in vsebuje zgornje ravni sklada: L2CAP (angl. Logical Link Control and Adaptation Protocol), SM (angl. Security Manager), ATT (angl. Attribute Protocol), GATT (angl. Generic Attribute Profile) in GAP (angl. Generic Access Profile), za delovanje pa uporablja centralno procesno enoto (krajše CPE).

Nad gostiteljem je mogoče uporabiti tudi številne druge profile in storitve, ki niso del Bluetooth CS in jih običajno definirajo specifikacije GATT (ne smemo jih enačiti z zgoraj omenjenim profilom).

Čeprav si krmilnik BLE številne lastnosti deli s krmilnikom klasičnega BT, sta med seboj nekompatibilna. Ker je implementacija krmilnika BLE v primerjavi s klasičnim BT relativno preprosta (in posledično cenejša), se v praksi pojavlja vse več krmilnikov, ki podpirajo oba protokola.



Slika 3.1: Sklad protokola BLE.

3.1.1 Fizična plast

Fizična plast predstavlja najnižjo raven sklada, ki se ukvarja s prenosom (pošiljanjem in sprejemanjem) podatkov po brezžičnem mediju (zraku). BLE deluje v industrijskem, znanstvenem in medicinskem (angl. Industrial Scientific Medical, krajše ISM) radiofrekvenčnem pasu, pri frekvenci 2.4 GHz.

Za komunikacijo je predvidenih 40 radiofrekvenčnih (krajše RF) kanalov, z medsebojnim razmikom 2 MHz. Poznamo dve vrsti kanalov, oglaševalske kanale (angl. advertising channels) in podatkovne kanale (angl. data channels). Frekvence številnih kanalov lahko izračunamo z uporabo formule:

$$f(k) = 2,402 + k \cdot 2MHz, k = 0, ..., 39 \tag{3.1}$$

Oglaševalski kanali se tipično uporabljajo za iskanje naprav, vzpostavljanje povezav in deljenje manjših količin podatkov z okolico, medtem ko se podatkovni kanali uporabljajo za dvosmerno komunikacijo med povezanimi napravami. Za oglaševanje so definirani trije kanali (k = 0, 12 in 39) z ustreznimi frekvencami. Te so določene tako, da se čim manj prekrivajo s pogosto uporabljenimi kanali brezžičnih lokalnih omrežij (predvsem kanali 1, 6 in 11, ki so definirani v okviru skupine IEEE 802.11). Za prenašanje podatkov po podatkovnih kanalih se uporablja mehanizem AFH (angl. Adaptive Frequency Hopping), ki omogoča hitrejšo in zanesljivejšo komunikacijo. Za prenos podatkov v posameznih časovnih intervalih (povezavni dogodki, opisani v 3.1.2) uporablja enega od 37 podatkovnih kanalov.

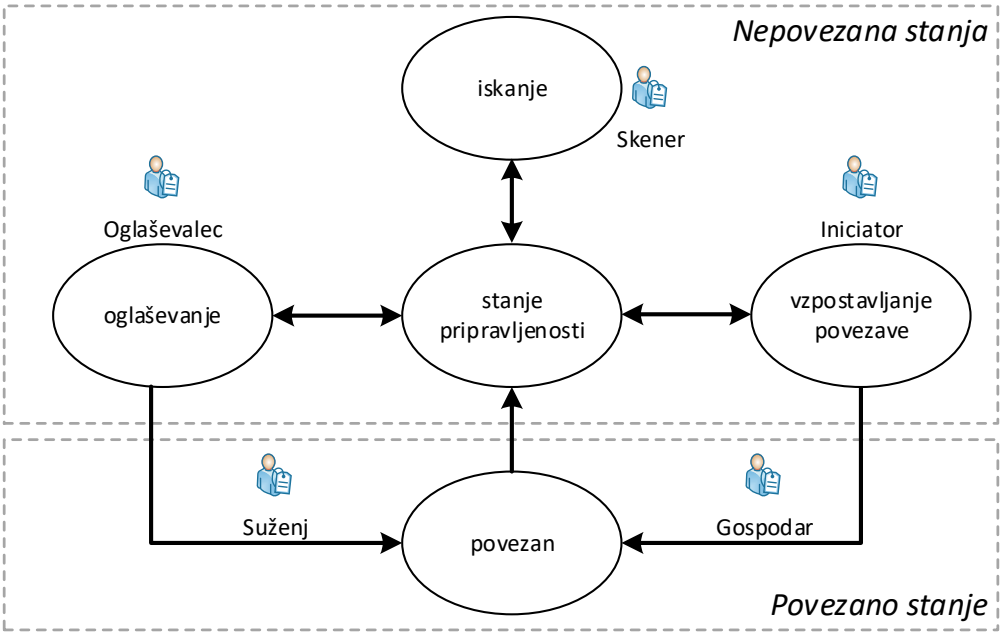
Radijska naprava BLE je lahko radijski oddajnik, radijski sprejemnik ali radijski oddajnik in sprejemnik. Specifikacija določa tudi dovoljeno izhodno moč (angl. output power) radijskega oddajnika, ki je lahko v območju od 0,01 mW (–20 dBm) do 10 mW (+10 dBm) [18]. Glede na različne potrebe se lahko izhodna moč dinamično spreminja, kar občutno vpliva na končni domet, energijsko porabo in motnje z drugo opremo. Za prenos se uporablja modulacija GFSK, ki omogoča hitrost prenosa do 1 Mb/s, domet pa se tipično giblje od 30 do 100 metrov (odvisno od številnih dejavnikov, kot sta npr.

okolica in izhodna moč oddajnika).

3.1.2 Povezavna plast

Povezavna plast je odgovorna za upravljanje povezav (angl. links), izbiro frekvenc prenosa, podporo različnih topologij in načinov izmenjave podatkov. Definira tudi številne vloge vpletenih deležnikov, format paketa, možna stanja povezav in prehode med njimi. S pomočjo vmesnika HCI omogoča svoje storitve višji plasti L2CAP.

Delovanje povezavne plasti lahko predstavimo s preprostim končnim avtomatom stanj (prikazan na sliki 3.2), ki ga sestavljajo: stanje pripravljenosti (privzeto stanje), stanje oglaševanja (pošiljanje oglaševalskih paketov), stanje iskanja (poslušanje oglaševalskih kanalov), stanje vzpostavitve (vzpostavitev povezave) in stanje povezave.

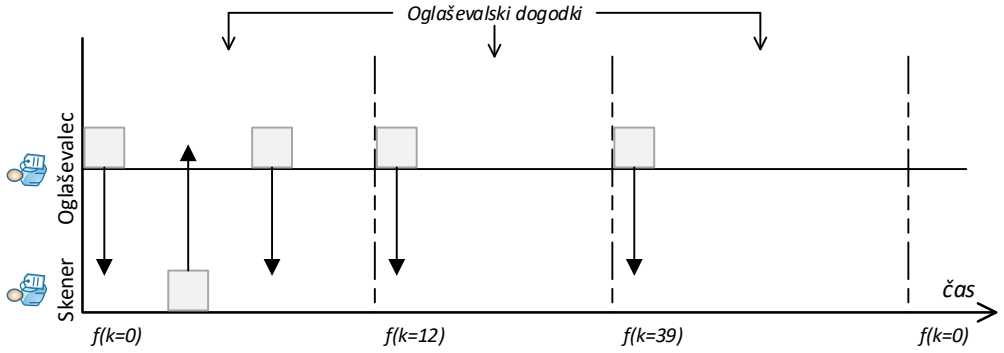


Slika 3.2: Prikaz avtomata stanj povezavne plasti s predvidenimi vlogami.

Oglaševalec (angl. advertiser) je v kontekstu povezavne plasti naprava, ki podatke po oglaševalskih kanalih deli z okolico. To počne v posebnih časovnih intervalih, ki jim rečemo tudi „oglaševalski dogodki“ (angl. advertising events). Vsak naslednji dogodek za prenos paketov uporablja naslednji oglaševalski kanal (npr. k = 39, 0, 12, 39 itn.). Naprava, ki v danem trenutku posluša oglaševalske pakete, se imenuje skener (scanner).

Čas med dvema oglaševalskima dogodkoma t_{adv} je odvisen od predvidenega intervala oglaševanja $t_{advInterval}$ (od 20 ms do 10.24 sec) in naključnega časovnega zamika $t_{advDelay}$(od 0 ms do 10 ms). Določa ga naslednja formula:

$$t_{adv} = t_{advInterval} + t_{advDelay} \tag{3.2}$$



Slika 3.3: Oglaševanje in izmenjava oglaševalskih paketov.

Za dvosmerno komunikacijo morata biti napravi med seboj povezani. To storita tako, da oglaševalec začne pošiljati oglaševalske pakete, s katerimi sporoča, da se je z njim mogoče povezati. Druga naprava (iniciator) posluša oglaševalske kanale. Ko prejme paket ustrezne naprave (oglaševalca), lahko nanj odgovori z zahtevo za odprtje povezave „Connection Request“. Ko je povezava vzpostavljena, se podatki prenašajo po podatkovnih kanalih. Paketi za identifikacijo povezave (seje) uporabljajo isti, naključno ustvarjen 32-bitni dostopni naslov (angl. access address).

Specifikacija definira vlogo gospodarja (angl. master) in sužnja (angl. slave), ki ju prevzameta iniciator in oglaševalec po uspešni vzpostavitvi povezave. Gospodar je lahko hkrati povezan z več sužnji, medtem ko ima suženj natanko enega gospodarja. Tako kot pri klasičnem BT, tovrstnim omrežjem, ki tvorijo zvezdno topologijo, pravimo „piko omrežja“. Drugo vrsto topologije pa predstavljajo naprave, ki med seboj niso povezane in si podatke izmenjujejo po oglaševalskih kanalih.

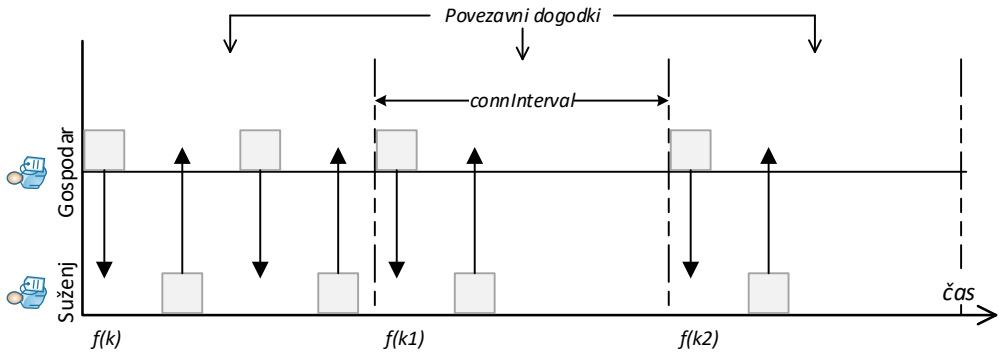
Sužnji so pogosto preproste senzorske naprave, ki delujejo na baterije in občasno komunicirajo z okolico. Privzeto delujejo v načinu spanja (angl. sleep mode) in s tem prihranijo veliko energije. Občasno se zbudijo in čakajo na morebitne pakete gospodarja. Ta določa, kdaj morajo biti sužnji v stanju pripravljenosti (poslušati), in upravlja dostop skupnega medija. Pri tem uporablja mehanizem TDMA (angl. Time Division Multiple Access).

Sužnjem sporoča tudi informacije glede algoritma AFH, skupaj z mapo kanalov, in nadzora povezave (angl. connection supervision). Parametri za upravljanje povezave se prenesejo v okviru zahtevka „Connection Request“, ki jih je iz različnih razlogov mogoče tudi spremeniti (npr. nova mapa kanalov zaradi spremenjenega vzorca motenj).

Ko se dve napravi povežeta, se fizični kanal razdeli v posamezne časovne enote, ki jim rečemo tudi „povezavni dogodki“ (angl. connection events). Vsak povezavni dogodek se začne s paketom gospodarja. Paketi, ki so del istega povezavnega dogodka, se prenesejo po istem podatkovnem kanalu določene frekvence. Suženj mora na paket gospodarja odgovoriti, medtem ko gospodarju ni treba odgovoriti na paket sužnja. Podatkovni paket vsebuje tudi bit „More Data“ (krajše MD), ki sporoča ali bo pošiljatelj še kaj poslal. Povezavni dogodek se lahko predčasno zaključi v primeru, ko ena od naprav prejme dva zaporedna paketa z okvarjenimi podatki ali če pride do okvare dostopnega naslova. Za odkrivanje napak se uporablja 24-bitna CRC koda, ki je sestavni del podatkovnega paketa. Primer izmenjave podatkovnih paketov prikazuje slika 3.4.

Za nov povezavni dogodek se uporabi nov podatkovni kanal, ki ga določa algoritem AFH (3.1.2). Čas med dvema zaporednima povezavnima dogodkoma je določen s parametrom „connInterval“ (število med 6 in 3200) in traja od 7,5 ms do 4 s (= 1,25 ms · connInterval). Naslednji pomemben parameter je „connSlaveLatency“, ki se uporablja predvsem z namenom varčevanja energije. Predstavlja število zaporednih povezavnih dogodkov (od 0 do 499), ki jih suženj lahko presliši (nanje ne odgovori) in ne presega nastavljenega parametra connSupervisionTimeout.

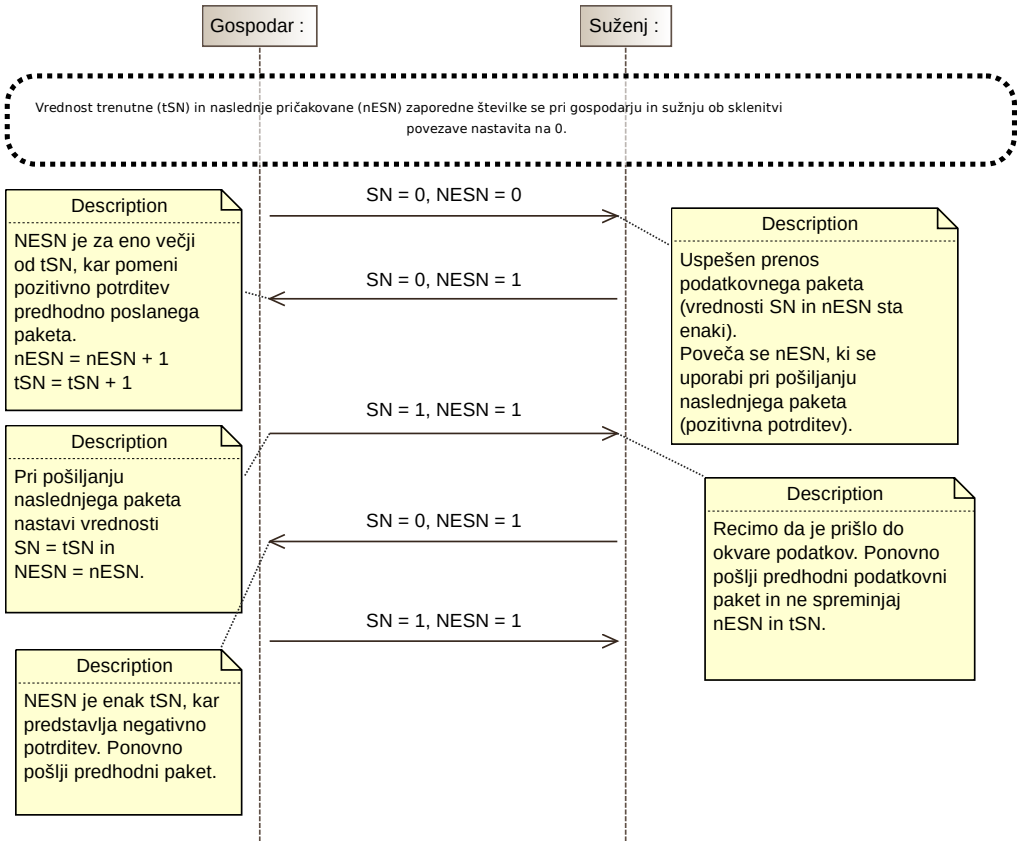
Parameter connSupervisionTimeout se uporablja za prekinitev neaktivne povezave. To se zgodi, če napravi v določenem času, ki ga definira parameter connSupervisionTimeout (od 100 ms do 32 s), ne prejmeta nobenega paketa. Na ta način je mogoče zaznati izgubo signala, kar je lahko posledica motenj ali prekratkega dosega.



Slika 3.4: Povezavni dogodki, ki se uporabljajo za prenos podatkov, ko je povezava vzpostavljena.

Pri povezavah se za kontrolo pretoka uporablja mehanizem „prenehaj in počakaj“ (angl. stop and wait), ki omogoča zaznavo izgubljenih paketov in zmožnost odprave napak. Ta temelji na uporabi posebnih polj podatkovnih paketov: SN (zaporedno število, angl. sequence number) in NESN (naslednje pričakovano zaporedno število, angl. next expected sequence number). SN enolično določa posamezni podatkovni paket, medtem ko NESN predstavlja pričakovano zaporedno število naslednjega (še ne prejetega) paketa povezane

naprave. Potrjevanje poteka tako, da vsaka naprava beleži zaporedno število zadnjega, uspešno prenesenega paketa. Pred prenosom naslednjega podatkovnega paketa naprava ustrezno nastavi polje NESN in tako potrdi predhodno prejet paket. Če pride do napake pri prenosu, se na ta način od naprave zahteva ponovno pošiljanje. Primer tovrstnega pretoka je predstavljen na sliki 3.5.



Slika 3.5: Primer kontrole pretoka povezavne plasti („prenehaj in počakaj“).

Algoritem AFH

Pri prenosu povezavnih dogodkov se z namenom zmanjšanja motenj med brezžičnimi omrežji uporabljajo različni podatkovni kanali. Te določa preprost algoitem AFH, ki izmed 37 podatkovnih kanalov na podlagi inkrementalne vrednosti „hopIncrement“ izbere naslednjega. Vrednost parametra „hopIncrement“ ob vzpostavitvi povezave določi gospodar.

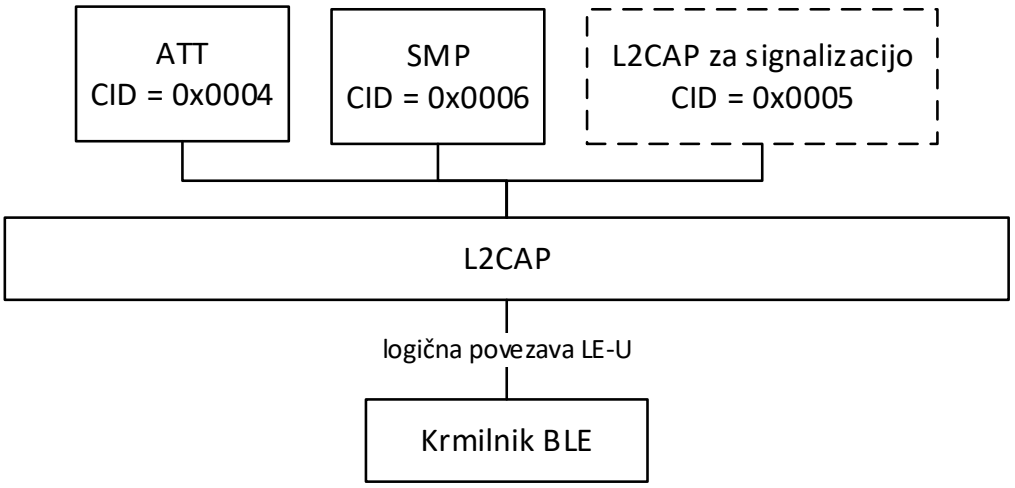
$$f_{n+1} = (f_n + hopIncrement) \mod 37 \tag{3.3}$$

$$hopIncrement \in [5, 16]$$

3.1.3 HCI in L2CAP

HCI omogoča komunikacijo med višjimi in nižjimi protokoli sklada s pomočjo standardiziranega komunikacijskega vmesnika. Specifikacija zajema definicije ukazov HCI in dogodkov, s pomočjo katerih poteka asinhrona komunikacija med krmilnikom in gostiteljem, kar zajema npr. upravljanje s stanjem krmilnika BLE, kontrolo nadzora krmilnika in gostitelja, upravljanje povezav, upravljanje z iskanjem naprav, upravljanje z avtentikacijo in šifriranjem idr. Temelji na specifikaciji predhodne različice BT (BR/EDR) in uporablja veliko obstoječih funkcionalnosti.

L2CAP služi kot vmesnik za komunikacijo med protokoli višjih plasti in nižjega dela sklada. Osnovni nalogi sta multipleksiranje in demultipleksiranje podatkov zgornjih treh plasti: ATT, SMP in L2CAP za signalizacijo, po deljeni logični povezavi „LE-U“ (angl. LE-U Logical Link), ki jo prikazuje slika 3.6. Podatki se prenašajo po logičnih kanalih v okviru posameznih paketov SDU (angl. Service Data Unit). Pri komunikaciji med dvema entitetama L2CAP (gospodarja in sužnja) se podatki prenašajo v obliki paketov PDU (angl. Protocol Data Unit), ki enkapsulirajo posamezne pakete SDU in so lahko različnih vrst (namenjeni različnim logičnim kanalom) in velikosti.



Slika 3.6: Multipleksiranje kanalov po logični povezavi LE-U.

Glavne značilnosti protokola L2CAP so torej:

- specifikacija logičnih kanalov s pripadajočimi identifikatorji CID (angl. Channel Identifiers),
- fragmentacija in defragmentacija podatkov ter
- multipleksiranje in demultipleksiranje različnih kanalov po skupni logični povezavi LE-U.

3.1.4 ATT

Protokol ATT zajema komunikacijo naprav (ki so v vlogah strežnika in odjemalca) s pomočjo logičnega kanala L2CAP in omogoča svoje storitve (za odkrivanje, branje in zapisovanje atributov) višjim plastem (GATT in GAP). Za predstavitev podatkov se uporablja podatkovna struktura, ki ji pravimo atribut. Ta poleg podatkov vsebuje tudi informacije o vrsti atributa, referenci (angl. handle) in dovoljenjih. Vlogi strežnika in odjemalca sta neodvisni od vlog gospodarja in sužnja povezavne plasti. Odjemalec je naprava, ki dostopa do atributov strežnika, lahko pa se tudi naroči na obvestila o spremembah

podatkov posameznih atributov. Posebna vrsta obvestil, ki od odjemalca zahteva potrditev, je t. i. „indikacija“ (angl. indication).

Za komunikacijo se uporabljajo sporočila ATT PDU, ki vsebujejo: informacije o izbrani metodi, morebitne parametre in opcijski podpis pristnosti (angl. authentication signature). Poznamo šest različnih metod, ki so:

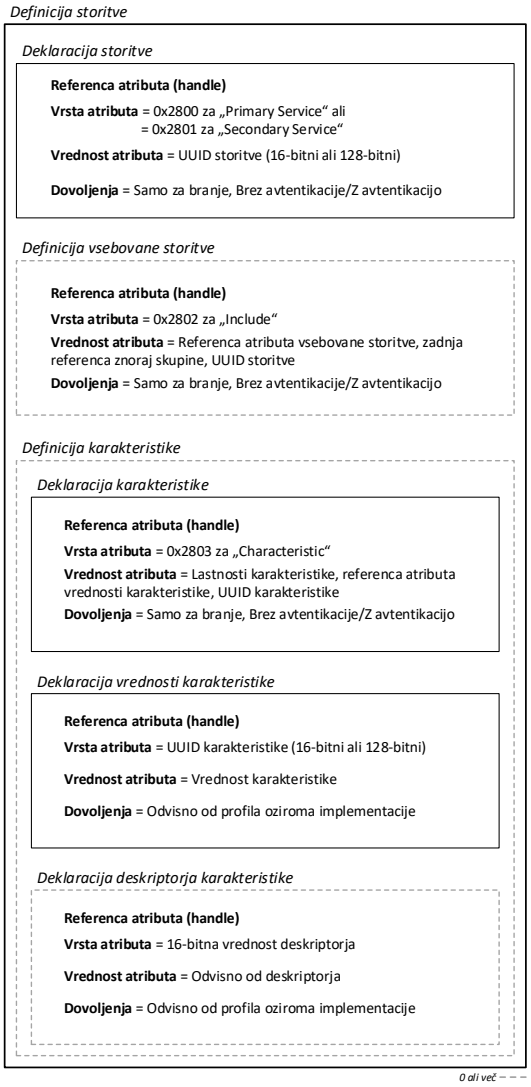
1. zahtevek in odgovor,
2. odgovor,
3. ukaz,
4. obvestilo,
5. indikacija oziroma „obvestilo in potrditev“ in
6. potrditev.

Posamezne metode se izvajajo atomarno, v sklopu transakcij in delujejo na način „prenehaj in počakaj“.

3.1.5 GATT

Profil GATT definira ogrodje storitev in karakteristik (hierarhično podatkovno strukturo). Temelji na funkcionalnosti ATT in omogoča odkrivanje storitev in izmenjavo karakteristik med odjemalcem in strežnikom. Storitve in karakteristike so sestavljene iz logično urejenih atributov, ki jih prikazuje slika 3.7. Po specifikaciji morajo vse naprave BLE obvezno podpirati profila GATT in GAP, ki predstavljata osnovo za modularno gradnjo ostalih profilov. Profil je sestavljen iz ene ali več storitev in opisuje določen primer uporabe, vloge in splošna vedenja. Storitev je podatkovna struktura, ki predstavlja določeno funkcionalnost in je sestavljena iz množice karakteristik, lahko pa se tudi sklicuje na ostale storitve. Karakteristika je najpreprostejša podatkovna struktura profila GATT in vsebuje vrednost ter opcijsko enega ali več deskriptorjev, ki opisujejo in omogočajo konfiguriranje vrednosti. Pri naročanju na obvestila o spremembi vrednosti karakteristike se na primer uporablja deskriptor „Client Characteristic Configuration“, ki ga je treba ustrezno nastaviti. Storitev in karakteristika vsebujeta tudi atribut za deklaracijo (Primary Service, Secondary Service, Include, Characteristic

Declaration), ki se uporablja pri odkrivanju storitev. Za naročanje na obvestila o spremembi vrednosti karakteristike se na primer uporablja deskriptor „Client Characteristic Configuration“, ki ga je potrebno ustrezno nastaviti.



Slika 3.7: Zgradba GATT storitve, ki jo sestavljajo posamezni atributi.

3.1.6 Varnost

BLE za varnost komunikacije uporablja storitve različnih plasti, ki omogočajo zaščito povezave med napravama, zaščito dostopa do podatkov in integriteto posameznih sporočil. Pomembno vlogo pri tem igra plast SM (upravitelj varnosti), ki definira postopke za seznanjanje naprav, overjanje in šifriranje.

Na ravni povezavne plasti sta podprta overjanje in šifriranje podatkov. Za overjanje se uporablja algoritem za blokovno šifriranje AES-CMAC (angl. Cipher-based Message Authentication Code, RFC-4493, [29]), ki omogoča izračun kode za preverjanje pristnosti sporočil (angl. Message Authentication Code, krajše koda MAC). Z uporabo števca podatkovnih paketov (za zaščito pred napadi s ponavljanjem) in kode MAC se lahko ustvari edinstven podpis, ki ga vsebuje polje MIC.

$$MAC = AES\text{-}CMAC(kljuc_{128\text{-}bit}, sporocilo, dolzina\ sporocila) \tag{3.4}$$

Za šifriranje podatkov (podatkovnih paketov povezavne plasti) se uporablja simetrični šifrirni postopek 128-bitni AES, ki ga definira FIPS-1971 [27]. Specifikacija omogoča, da šifriranje izvaja bodisi krmilnik bodisi gostitelj. Predviden je tudi prenos overjenih podatkov po nezaščiteni povezavi, in sicer na ravni plasti ATT. Sporočilo ATT PDU vsebuje opcijski 96-bitni podpis pristnosti, ki se uporablja za overjanje operacijske kode in podatkov atributnih parametrov (glede na uporabljeno metodo).

$$sifropis_{128\text{-}bit} = e(kljuc_{128\text{-}bit}, cistopis_{128\text{-}bit}) \tag{3.5}$$

BLE omogoča tudi zaščito pred sledenjem naprav na podlagi javnega MAC naslova (varovanje zasebnosti). Za ta namen se uporabljajo zasebni naslovi, ki se sčasoma spreminjajo (priporočljiva vrednost intervala je 15 minut). Generiranje zasebnega naslova temelji na šifriranju javnega naslova. Razpoznajo ga lahko samo seznanjene naprave, z uporabo posebnega ključa IRK (angl. Identity Resolving Key).

Zgoraj opisani varnostni mehanizmi temeljijo na uporabi kriptografskih ključev. Generiranje kriptografskih ključev se izvaja na ravni gostitelja, kar omogoča preprosto (programsko) nadgradnjo algoritmov brez potrebe po spreminjanju krmilnika. Za zaščito komunikacije si napravi izmenjata množico kriptografskih ključev. Ta postopek imenujemo seznanjanje (angl. pairing). Specifikacija definira dve varnostni shemi, ki se uporabljata za zaščito povezave pri seznanjanju naprav. To sta „Secure Simple Pairing“ (krajše SSP) in „Secure Connections“ (krajše SC, podprto od različice BLE 4.2 naprej). Seznanjanje je sestavljeno iz treh faz, za katere skrbi SMP (angl. Security Manager Protocol).

Prva faza zajema izmenjavo vhodno-izhodnih zmožnosti (brez vnosa, izbira da/ne, tipkovnica, brez izhoda, zaslon) in varnostnih zahtev (npr. zaščita MiTM), na podlagi katerih se izbere ustrezna metoda za naslednjo fazo.

V drugi fazi se glede na izbrano metodo in shemo izmenjajo ustrezni ključi. Če se uporablja SSP, se na podlagi začasnega ključa TK (angl. Temporary Key) zgenerira ključ STK (angl. Short-Term Key), ki se uporablja za zaščito prenosa v tretji fazi. Za dogovor glede skupnega TK se uporablja izbrana metoda iz prve faze (npr. pri metodi „Just Works“ je vrednost TK enaka 0, pri „Passkey entry“ pa je vrednost TK predstavljena kot šestmestno število), ki vpliva na stopnjo zaščite pred aktivnimi napadi (npr. napadom s posrednikom). Če se uporablja SC, se že v tej fazi zgenerira ključ LTK (angl. Long Term Key). To se zgodi z uporabo protokola ECDH za izmenjavo ključev, kjer si napravi najprej zgenerirata zasebni in javni ključ, nato si izmenjata javna ključa ter izračunata skupen Diffie-Hellmanov ključ (krajše DHkey). Nato se, glede na izbrano metodo izvede avtentikacija in v primeru uspeha, izračuna ključ LTK.

V tretji fazi se po zaščiteni povezavi prenesejo tudi nekateri drugi podatki, kot so npr. informacije za šifriranje povezave (ključ LTK skupaj s številoma EDIV in Rand, samo če se uporablja SSP), informacije o napravi (naslov naprave, ključ IRK) in informacije za overjanje podatkov (ključ CSRK).

Glavna ranljivost SSP je, da nobena od metod seznanjanja ne omogoča

zaščite proti pasivnemu prisluškovanju. Napadalec lahko na podlagi zajetih sporočil seznanjanja ugotovi ključe za šifriranje, overjanje in razpoznavo zasebnih naslovov (LTK, CSRK in IRK).

3.1.7 GAP in aplikacijski profili

Profil GAP je na najvišji ravni sklada (ki je še del jedra) in določa skupne funkcionalnosti naprav BLE. Definira posamezne vloge (oddajnik, opazovalec, periferna in centralna vloga), načine delovanja in splošne procedure, ki so povezane z odkrivanjem naprav v bližini, povezovanjem naprav in varnostjo. Vsebuje tudi priporočila za uporabo skupne terminologije na ravni uporabniškega vmesnika, kar med drugim zajema navodila za uporabo, dokumentacijo, oglaševanje, grafične uporabniške vmesnike, pisanje razumljive programske kode ipd.

Naprava v vlogi oddajnika (angl. broadcaster) periodično pošilja (oglašuje) podatke po oglaševalskih kanalih in ne omogoča povezovanja. Vlogo opazovalca (angl. observer) ima naprava, ki podatke v okolici zgolj opazuje na oglaševalskih kanalih in se ne povezuje z drugimi napravami. Naprava v centralni vlogi (central) začenja z vzpostavljanjem povezav in je lahko povezana z več perifernimi (angl. peripheral) napravami. Posamezne vloge so odvisne od podpore krmilnika (npr. naprava, ki ima samo radijski oddajnik ali sprejemnik, ne more podpirati centralne in periferne vloge). Naprava lahko hkrati deluje v več vlogah, če to omogoča povezavna plast.

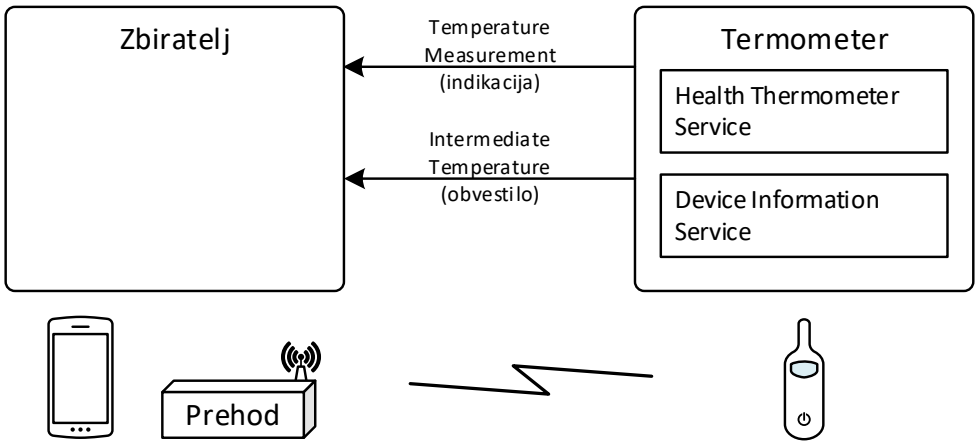
GAP in GATT predstavljata osnovno za gradnjo ostalih aplikacijskih profilov. Omogočata modularno zasnovo novih profilov in storitev, kjer se lahko skupne funkcionalnosti pogosto ponovno uporabijo. Za zbirko splošno sprejetih profilov in storitev (v okviru GATT specifikacij) skrbi organizacija Bluetooth SIG. Specifikacije GATT so bistvenega pomena za interoperabilnost med napravami različnih proizvajalcev in so produkt sodelovanja več deležnikov. Kljub temu se v praksi pogosto srečamo z zasebnimi (angl. proprietary) profili in storitvami, ki jih za podporo specifičnih funkcionalnosti običajno razvijejo posamezne organizacije. Prednost uporabe splošno sprejetih stori-

tev in karakteristik se poleg interoperabilnosti kaže tudi v manjših podatkovnih paketih. Za vrsto atributa se uporablja zgolj 16-bitna vrednost namesto 128-bitne, kar posledično vpliva na čas prenosa in manjšo porabo energije.

Vse specifikacije aplikacijskih profilov GATT so dostopne na [26]. Med njimi so tudi spodaj našteti profili, ki so zanimivi predvsem za področje zdravstva, saj omogočajo zajem podatkov in meritev za naslednje naprave:

- merilec krvnega tlaka - Blood Pressure Profile (krajše BLP),
- sistem za neprekinjeno merjenje glukoze - Continuous Glucose Monitoring Profile (krajše CGMP),
- glukometer - Glucose Profile (krajše GLP),
- termometer - Health Thermometer Profile (krajše HTP),
- inzulinska črpalka - Insulin Delivery Profile (IDP),
- oksimeter - Pulse Oximeter Profile (krajše PLXP) in
- tehtnica - Weight Scale Profile (krajše WSP).

V nadaljevanju je za primer termometra na kratko predstavljen profil HTP, ki omogoča interakcijo naprave BLE z merilnikom telesne temperature, ki podpira storitev „Health Thermometer Service“ (krajše HTS). Specifikacija določa vlogo termometra (angl. thermometer) in zbiratelja (angl. collector). Termometer deluje kot strežnik in predstavlja periferno napravo, medtem ko zbiratelj deluje kot odjemalec in predstavlja centralno napravo. Termometer mora obvezno podpirati naslednji storitvi: „Health Thermometer Service“ in „Device Information Service“.



Slika 3.8: Komunikacija med zbirateljem in termometrom, ki jo definira profil HTP.

Storitev HTS omogoča indikacije o opravljenih meritvah, branje mesta meritve (opcijsko), obveščanje o trenutnem rezultatu merjenja (opcijsko) in možnost upravljanja z intervalom merjenja (opcijsko). Storitev DI pa omogoča branje različnih informacij naprave (kot so npr.: naziv proizvajalca, številka modela, serijska številka, različica strojne in programske opreme). Za vrednost posamezne karakteristike je definirana podatkovna struktura, ki je sestavljena iz enega ali več polj.

Za meritev telesne temperature se uporablja karakteristika „Temperature Measurement“, ki jo predstavlja tabela 3.1.

Polje	**Zahteve polja**	**Format**	**Opis**
zastavice	obvezno	8-bitov	$bit_0 = 1$? C2:C1 $bit_1 = 1$? C3:/ $bit_2 = 1$? C4:/ bit_{3-7}: rezervirani za prihodnjo uporabo
vrednost temperature [°C]	C1	32-bitov (float)	/
vrednost temperature [°F]	C2	32-bitov (float)	/
časovni žig	C3	tabela 3.2	/
mesto meritve	C4	tabela 3.3	/

Tabela 3.1: Polja karakteristike „Temperature Measurement“.

Tabela 3.2: Polja karakteristike „Date Time“.

Polje	**Zahteve polja**	**Format**	**Opis**
leto	obvezno	16-bitov (uint16)	0 če ni znano 1582..9999
mesec	obvezno	8-bitov (uint8)	0 če ni znano jan..dec = 1..12
dan	obvezno	8-bitov (uint8)	0 če ni znano 1..31
ure	obvezno	8-bitov (uint8)	0..23
minute	obvezno	8-bitov (uint8)	0..59
sekunde	obvezno	8-bitov (uint8)	0..59

Tabela 3.3: Polja karakteristike „Temperature Type“.

Polje	**Zahteve polja**	**Format**	**Opis**
mesto meritve	obvezno	8-bitov	1: pazduha 2: telo (splošno) 3: uho 4: prst 5: gastrointestinalni trakt 6: usta 7: danka 8: prst na nogi 9: ušesni boben 0 in 10..255: rezervirano za prihodno uporabo

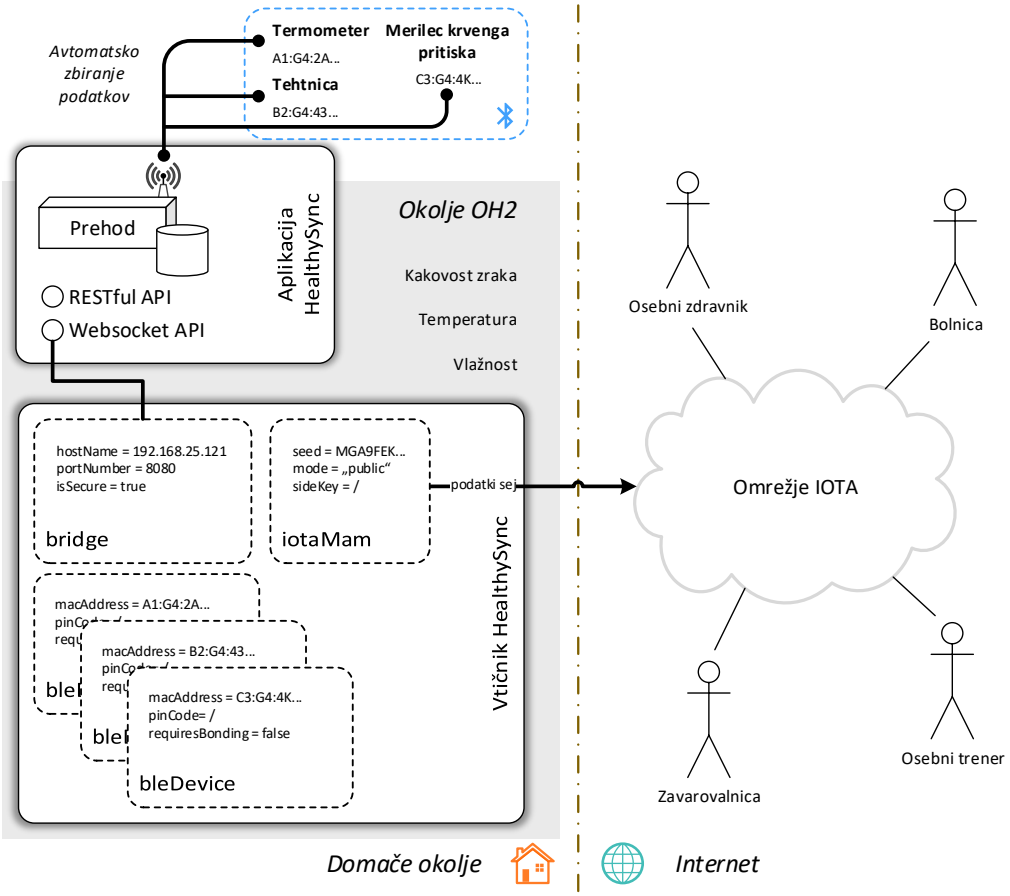
Poglavje 4

Predstavitev programske rešitve HealthySync

V tem poglavju bomo predstavili namen in zasnovo sistema HealthySync, posamezne vloge in implementirane funkcionalnosti, uporabljene tehnologije ter programsko in strojno opremo.

V okviru diplomskega dela smo razvili prilagodljiv sistem za zajem strukturiranih podatkov naprav BLE. Sistem je namenjen uporabi vseh merilnih naprav (ne le zdravstvenih), ki sledijo specifikaciji GATT in uporabljajo standardne, uveljavljene postopke. Biti mora preprost za uporabo (delovati po principu: „priključi in uporabljaj“ (angl.: „plug and play“)), konfigurabilen in za uporabnika čim manj opazen (invaziven). Od uporabnika se pričakuje zgolj, da sistem na začetku ustrezno pripravi: ga vklopi, poveže z omrežjem, doda želene merilne naprave, jim nastavi konfiguracijo za pripravo sej ter doda aktivne naročnine za zajem meritev.

Sistem omogoča shranjevanje podatkov v lokalno podatkovno bazo. Pri tem povezljivost z ostalimi napravami (v času opravljanja meritev) ni obvezna. V nasprotnem primeru (če shranjevanje ni omogočeno) se podatki sej zgolj prenesejo med morebitne poslušalce (odjemalce Websocket API-ja), in se pri tem lahko izgubijo (če ni nobenega poslušalca).



Slika 4.1: Prikaz delovanja sistema HealthySync.

4.1 Arhitektura sistema

Sistem HealthySync je sestavljen iz naslednjih komponent:

- aplikacije Android Things in
- vtičnika OH2.

Za delovanje celotnega sistema so potrebni:

- sistemski modul (prehod), ki podpira tehnologijo BLE in Android Things,
- nameščen operacijski sistem Android Things in
- okolje OH2 (računalniški sistem, na kateremu teče instanca OH2).

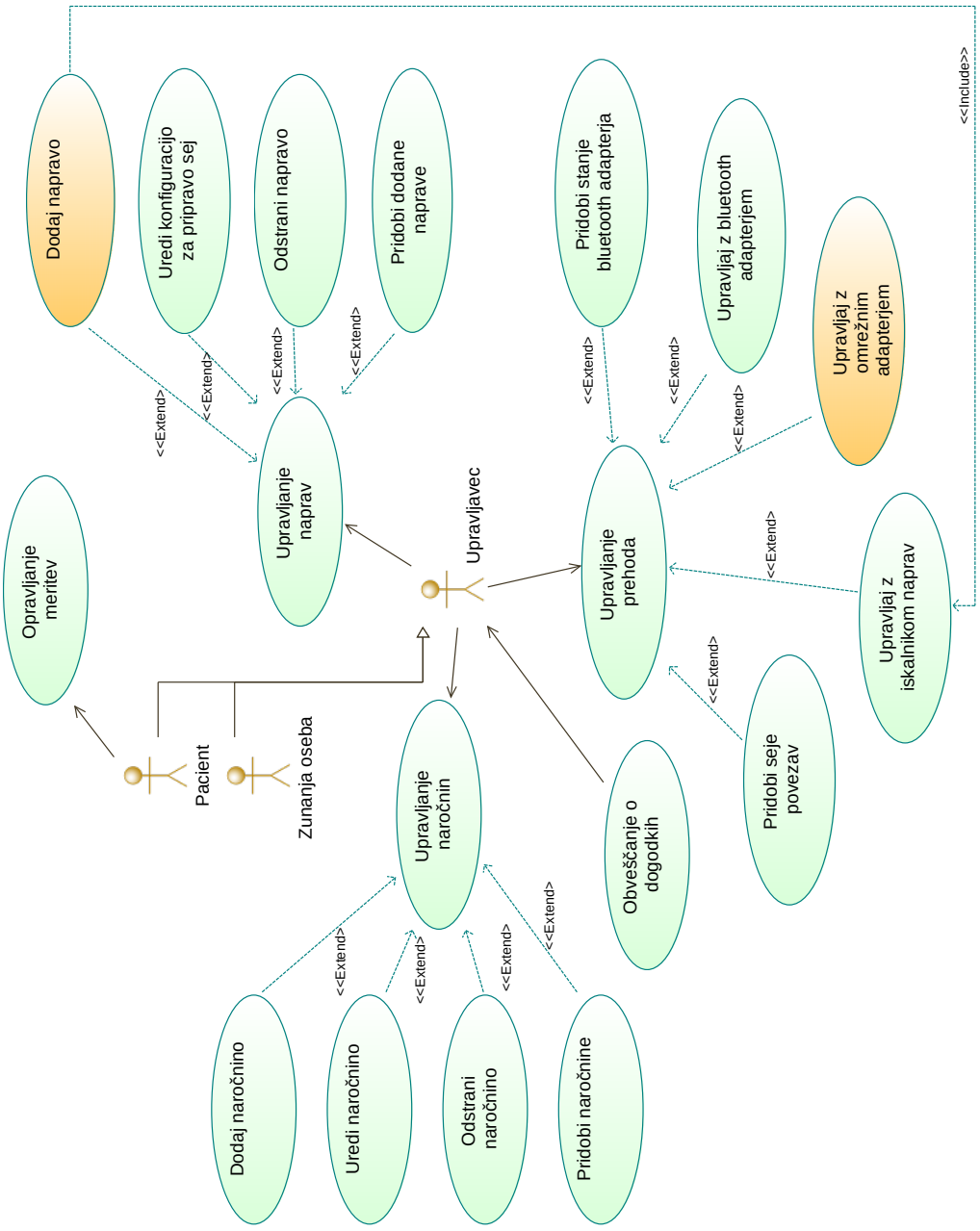
Uporaba vtičnika in sistema OH2 (predstavljena v poglavju 4.3.1) je sicer opcijska, vendar uporabniku omogoča dostop do več kot 200 različnih tehnologij in sistemov [23]. Pametni dom na področju IoT predstavlja pomemben vir podatkov notranjega (domačega) okolja, ki skupaj z zunanjim, komplementarno dopolnjuje zajete podatke naprav BLE, kar predstavlja osnovo za aplikacije na področju APH.

Iz diagrama primerov uporabe na sliki 4.2 so razvidni funkcionalnosti in ključni akterji aplikacije HealthySync.

Pacient je oseba, ki opravlja meritve in ima fizičen dostop do prehoda, vendar ni nujno, da ga zna tudi upravljati. Pri namestitvi/integraciji ji lahko pomaga tehnično podkovana oseba (v nadaljevanju upravljavec, npr. računalniški tehnik, prijatelj, sorodnik, znanec ipd.), saj proces ni zapleten. V nadaljevanju bomo predpostavili, da je pacient tudi upravljavec prehoda, saj želimo, da ima kot lastnik/končni uporabnik popoln nadzor nad sistemom.

Zunanja oseba je avtoriziran uporabnik, ki je izven lokalnega omrežja in mu je omogočeno omejeno oddaljeno upravljanje prehoda.

Upravljavec je oseba, ki ima fizičen dostop do prehoda in okolja OH2. Tipično je to uporabnik oziroma pacient. Prehod lahko upravlja s pomočjo ustreznega programskega vmesnika (v nadaljevanju API-ja). Ta v osnovi ni zaščiten in je dostopen vsem znotraj lokalnega omrežja. Če želi omogočiti dostop osebam izven lokalnega omrežja ali dostop omejiti, mora za to poskrbeti sam (ustrezno konfigurirati usmerjevalnik, uporabiti posredovalni strežnik ipd.). Upravlja lahko tudi vtičnik OH2 in omogoči shranjevanje podatkov v porazdeljeno omrežje IOTA.



Slika 4.2: Diagram primerov uporabe aplikacije HealthySync.

4.2 Aplikacija HealthySync

Razvoj aplikacije smo začeli s popisom zahtev in omejitev, na podlagi katerih smo določili glavne funkcionalnosti sistema. Nato smo definirali posamezne zgodbe in jih v obliki opravil zavedli v sistem Bitbucket. Za verzioniranje izvorne kode smo uporabili sistem Git. Primer zgodovine sprememb, iz katerih je razviden potek razvoja in sosledje opravil (kronološko urejen dnevnik sprememb), prikazuje slika 4.3. Za razvoj aplikacije smo uporabljali integrirano razvojno okolje Android Studio. Ustvarili smo projekt za platformo Android Things, izbrali SDK API s stopnjo 27 in pripravili osnovno strukturo projekta, kot ga prikazuje slika 4.4. Ker ne gre za tipično vrsto aplikacije in jo, z izjemo omrežnega adapterja, ne upravljamo s pomočjo grafičnega vmesnika, smo ustvarili zgolj glavno aktivnost. Ta predstavlja vstopno točko sistema in se izvede ob zagonu naprave/prehoda. Prehod je mogoče upravljati s pomočjo REST in Websocket API-ja.

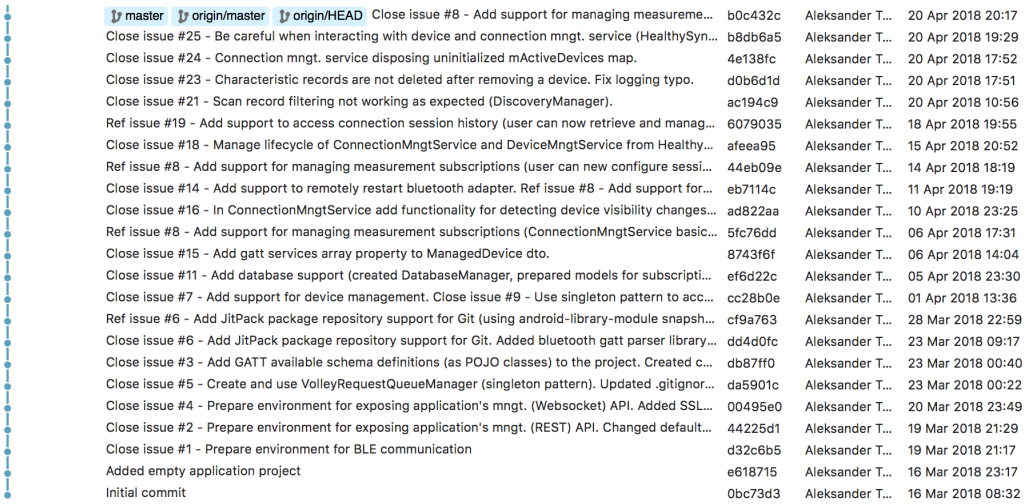
Upravljavec lahko:

- upravlja prehod,
- upravlja naprave in
- upravlja naročne.

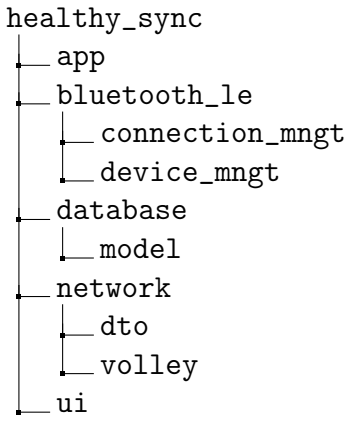
Upravljanje prehoda je primer uporabe, ki zajema upravljanje BT adapterja (vklop, izklop, pregled trenutnega stanja), upravljanje omrežnega adapterja (potreben fizičen dostop), upravljanje iskalnika naprav BLE in pridobivanje povezavnih sej.

Upravljanje naprav je primer uporabe, ki zajema dodajanje novih naprav BLE (včasih potreben fizičen dostop), urejanje konfiguracij za pripravo sej ter pridobivanje in odstranjevanje obstoječih naprav BLE.

Upravljanje naročnin je primer uporabe, ki zajema dodajanje, urejanje, pridobivanje in odstranjevanje naročnin (predstavljene v 4.2.2).



Slika 4.3: Zgodovina sprememb repozitorija aplikacije HealthySync, prikazana v Git odjemalcu SourceTree.



Slika 4.4: Struktura aplikacije HealthySync

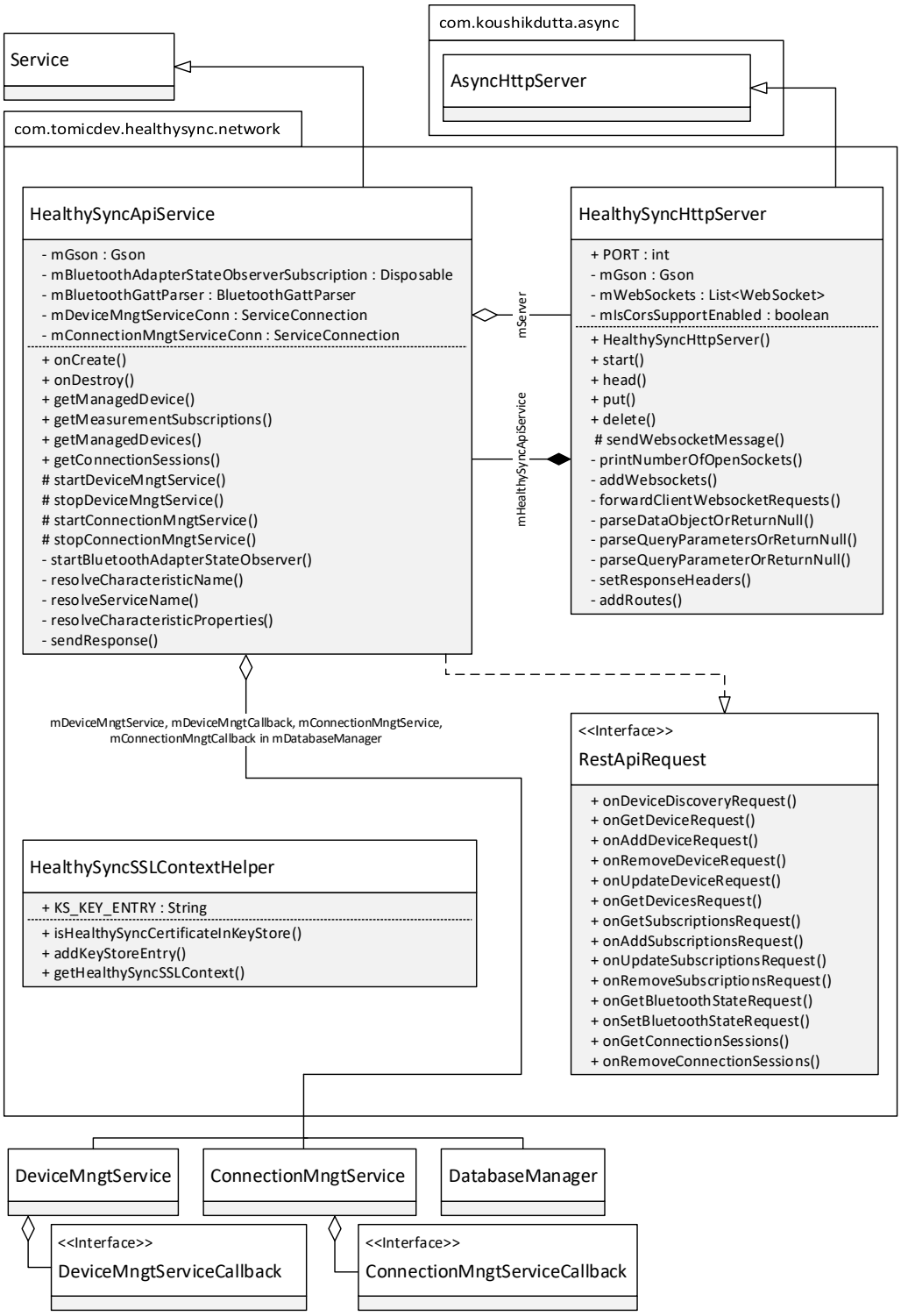
Aplikacija je sestavljena iz naslednjih ključnih kompontent:

- storitve za upravljanje prehoda, HealthySyncApiService,
- storitve za upravljanje povezav, ConnectionMngtService in
- storitve za upravljanje naprav, DeviceMngtService.

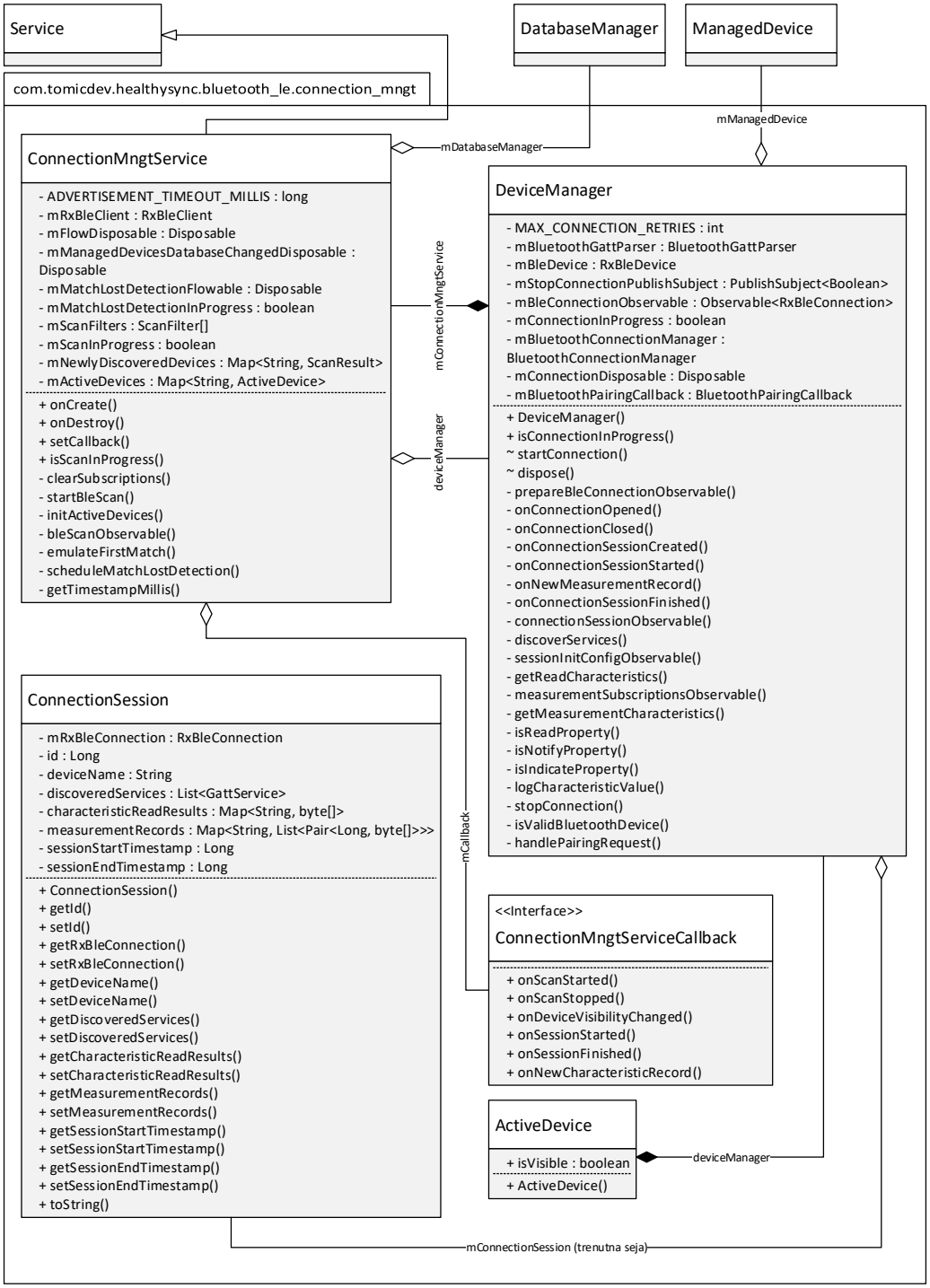
HealthySyncApiService je storitev in glavni razred paketa *network* (predstavljen na sliki 4.5), ki omogoča upravljanje prehoda po računalniškem omrežju. Za to uporablja asinhroni spletni strežnik (knjižnice *Android Async*), ki podpira komunikacijska protokola HTTP in Websocket. Za varno komunikacijo je mogoča uporaba kriptografskega protokola TLS/SSL. Prav tako upravlja storitvi DeviceMngtService in ConnectionMngtService.

DeviceMngtService je storitev in glavni razred paketa *device_ mngt* (predstavljen na sliki 4.7), ki omogoča iskanje, dodajanje in brisanje naprav iz sistema. Sestavljata jo iskalnik (*discoveryManager*) in seznanitelj naprav (*pairingManager*), ki ju storitev HealthySyncApiService uporablja za izvajanje želenih, asinhronih operacij. Za obveščanje o napredku in rezultatu (iskanje se je začelo, iskanje se je končalo, najdena je bila nova naprava, dodajanje naprave končano, odstranjevanje naprave končano) posameznih operacij se uporablja metoda povratnih klicev. Za to mora poslušalec (storitev HealthySyncApiService) pred tem ustrezno nastaviti povratni klic.

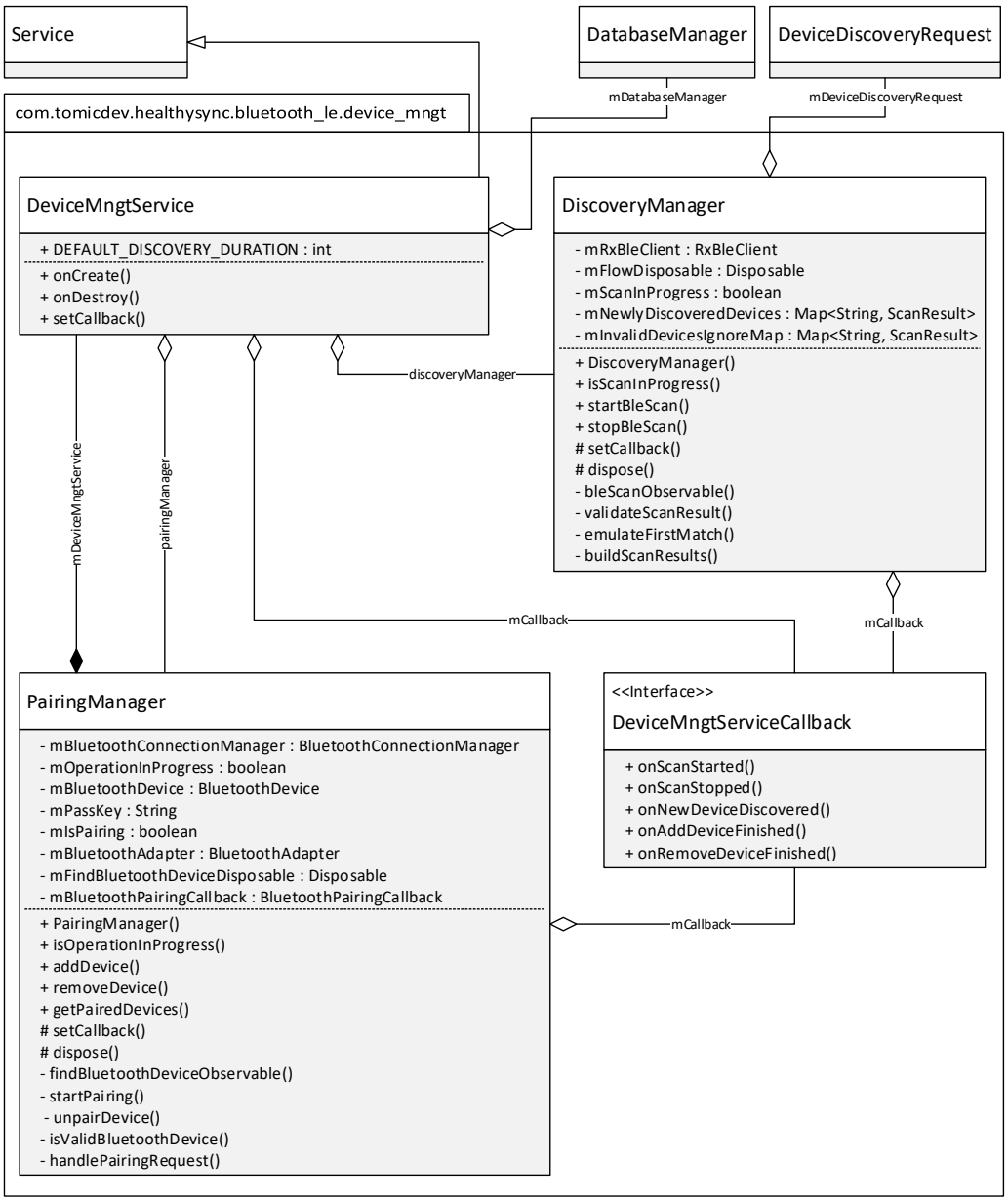
ConnectionMngtService je storitev in glavni razred paketa *connection_ mngt* (predstavljen na sliki 4.6), ki skrbi za samodejno povezovanje aktivnih naprav, pripravo povezavnih sej in zajem meritev. Vsebuje mehanizem za zaznavanje vidnosti trenutno aktivnih naprav. Za samodejno povezovanje aktivnih naprav uporablja posamezne instance razreda *DeviceManager*, ki zajema celoten proces upravljanja povezav (npr. inicializacijo novo dodane naprave, zajem kontekstualnih podatkov seje, zajem meritev, obvladovanje napak ipd.). Tudi ConnectionMngtService za obveščanje o dogodkih (iskanje se je začelo, iskanje se je končalo, vidnost naprave se je spremenila, nova seja se je začela, seja se je končala, nova meritev je na voljo) uporablja metodo povratnih klicev. Za to mora poslušalec (storitev HealthySyncApiService) pred tem ustrezno nastaviti povratni klic.



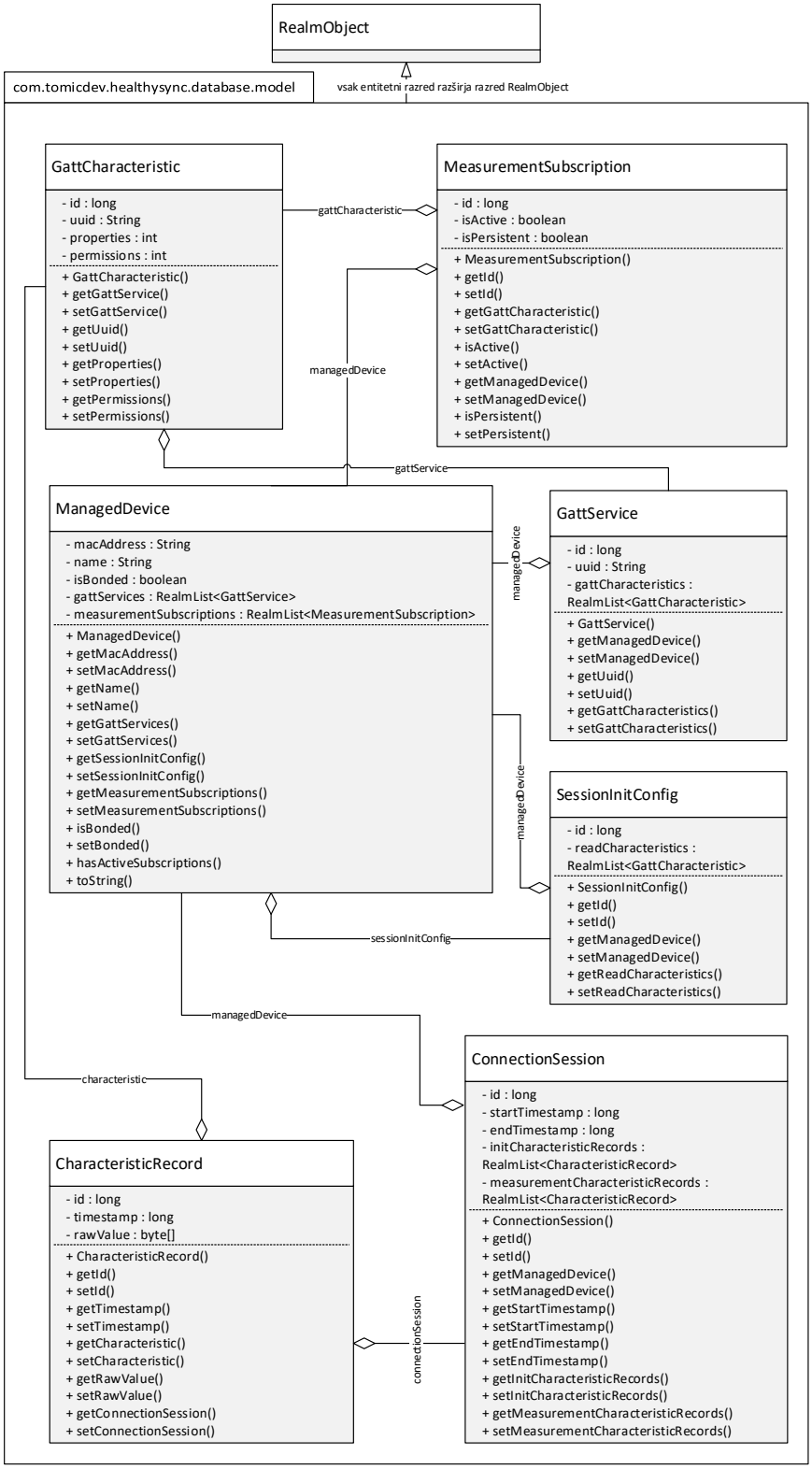
Slika 4.5: Razredni diagram paketa *network*.



Slika 4.6: Razredni diagram paketa *connection_ mngt*.



Slika 4.7: Razredni diagram paketa *device_ mngt*.



Slika 4.8: Razredni diagram paketa *model*.

4.2.1 Uporabljene tehnologije in orodja

V nadaljevanju bomo na kratko predstavili ključne tehnologije in orodja, ki smo jih uporabljali pri razvoju aplikacije HealthySync. Za programski jezik smo izbrali *Javo*, saj zaradi dobre podpore razvojnega okolja in novih programskih konstruktov, ki jih prinaša osma različica (Lambda izrazi, Stream API, ipd.), omogoča hiter razvoj (pisanje krajše, preglednejše programske kode). Pri razvoju podatkovne baze in upravljanju naprav BLE smo uporabljali pristope reaktivnega programiranja. Pomagali smo si z reaktivnimi razširitvami (angl. reactive extensions) knjižnice *RxJava*. Za shranjevanje podatkov smo uporabili persistentno rešitev *Realm Database*, ki predstavlja alternativo klasičnim relacijskim podatkovnim bazam (kot je npr.: SQLite) in sorodnim rešitvam (kot je npr.: Core Data). Za upravljanje naprav BLE smo uporabili knjižnico *RxAndroidBle*, ki vsebuje konstrukte reaktivnega programiranja (observables) in omogoča lažje delo z asinhronimi dogodki. Za razpoznavanje prebranih podatkov in meritev smo uporabili knjižnico *bluetooth-gatt-parser*. Pri implementaciji storitve HealthySyncApiService smo uporabili knjižnico *Android Async* za spletni strežnik in knjižnico *Gson* za serializacijo/deserializacijo podatkov. Pri implementaciji glavne aktivnosti smo uporabili knjižnico *Butter Knife* za asociiranje grafičnih gradnikov s programsko kodo in knjižnico *Volley* za izvajanje zahtev HTTP. Pri razvoju aplikacije smo uporabljali razvojno okolje *Android Studio* in računalniški sistem *Raspberry Pi 3* z operacijskim sistemom *Android Things*. Pri delu s podatkovno bazo smo za vizualizacijo podatkov uporabljali razvojno orodje *Realm Studio*, ki omogoča odpiranje in urejanje posameznih datotek Realm. Za razvoj REST API-ja smo uporabljali orodje *Swagger Editor*, ki omogoča oblikovanje in dokumentiranje API-jev z uporabo specifikacije Open API. Za upravljanje prehoda smo uporabljali spletni brskalnik *Google Chrome* (RESTful API) in vtičnik *Smart Websocket Client* (Websocket API). Za verzioniranje izvorne kode smo uporabljali sistem *Git* in aplikacijo *SourceTree*, za hranjenje repozitorija projekta in beleženje opravil pa spletno storitev *Bitbucket*.

4.2.2 Delovanje

Za nemoteno delovanje sistema in samodejno zbiranje meritev je treba prehod na začetku ustrezno pripraviti. Upravljavec mora v sistem dodati želene naprave BLE in ustrezno nastaviti konfiguracije sej in naročnin.

Dodajanje naprave zajema:

- uporabo iskalnika naprav za poizvedovanje osnovnih informacij, kot so: naziv naprave, fizični naslov (MAC) in primarne storitve (implicitno so razvidne tudi njihove obvezne karakteristike)
- dodajanje naprave, kjer je treba podati fizični naslov naprave, podatek, ali je potrebna sistemska seznanitev (polje boolean) ter morebitno geslo[1] - in
- izvedbo testne meritve, potrebne za inicializacijo osnovnih podatkov naprave.

Po uspešno dodanih napravah mora upravljavec nastaviti konfiguracijo sej in naročnin. Konfiguracija seje ni obvezna in je specifična za posamezno napravo. Uporablja se za pridobivanje kontekstualnih podatkov na začetku seje, ki niso del meritev (npr.: ura naprave, vrsta meritve, stanje baterije ipd.). Naprava ima lahko več naročnin, ki se uporabljajo za pridobivanje meritev (npr. meritev krvnega sladkorja, kontekst meritve krvnega sladkorja, meritev krvnega tlaka ipd.). Naročnina je lahko bodisi aktivna bodisi neaktivna, persistentna (kar pomeni, da se podatki meritev trajno hranijo v datoteki Realm in so na voljo za dostop do izbrisa) ali nepersistentna.

[1] Uporabljena različica sistema Android Things Developer Preview 7, ki je bila v času implementacije zadnja na voljo, ne podpira naprednih načinov seznanitve.

Naročnina je v podatkovni bazi predstavljena z naslednjimi polji:

- id zapisa,
- metapodatki meritve (UUID karakteristike, ki podpira obvestila (angl. notifications) ali indikacije (angl. indications)),
- status aktivnosti,
- status persistentnosti in
- fizični naslov naprave.

Če naprava nima nobene aktivne naročnine, se sistem z njo ne bo povezoval, ko bo ta na voljo (razen če še ni inicializirana).

HealthySync API je namenjen upravljanju in opazovanju sistema, z uporabo Websocket in HTTP protokola. Izvedli smo ga s spletnim strežnikom, ki podpira RESTful in Websocket API. V nadaljevanju bomo predstavili namen posameznega API-ja in skozi primere demonstrirali uporabo Websocket API-ja.

REST API je namenjen predvsem tistim, ki želijo s prehodom komunicirati zgolj občasno (npr. prenašati podatke, dodajati naprave, upravljati naročnine ipd.). Podpira vse zahtevke, ki so na voljo upravljavcu, in predstavlja osnovo za razvoj Websocket API-ja. Razvit je bil s pomočjo orodja *Swagger Editor* in specifikacije *Open API, različice 2.0*.

Websocket API je namenjen vsem, ki želijo biti v realnem času obveščeni o dogodkih sistema (v nadaljevanju opisani kot vrste sporočil, ki jih sproža sistem).

Za prenos podatkov po Websocket povezavi smo definirali razred „Message“, ki ga sestavljajo naslednja polja:

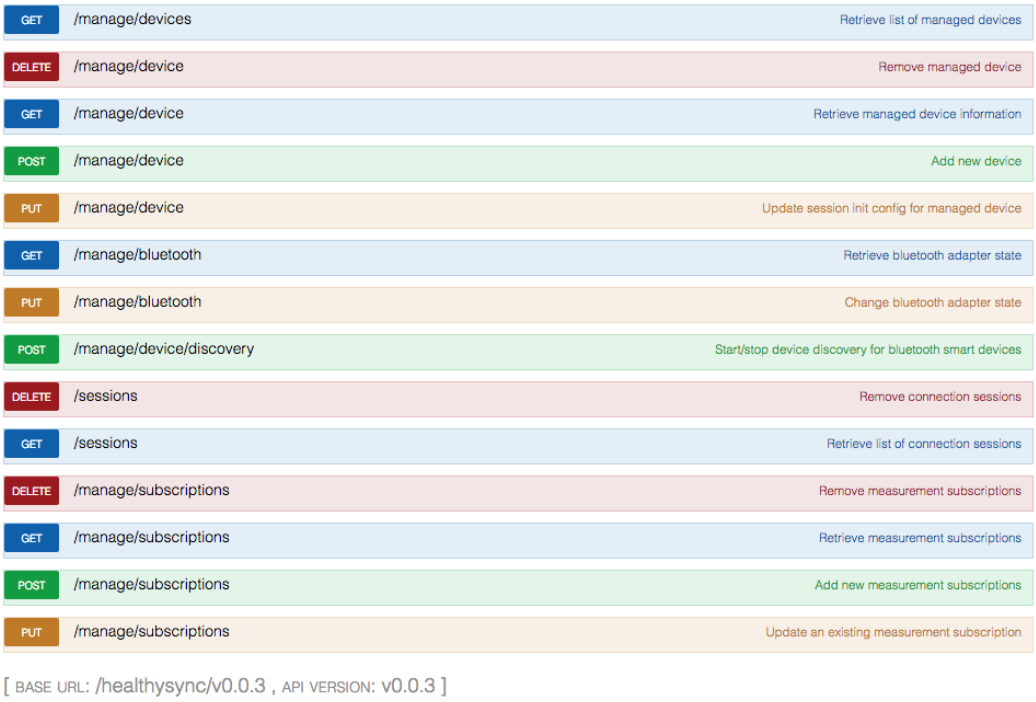
- uuid : niz znakov (obvezno) - unikatni identifikator sporočila,
- type : niz znakov (obvezno) - vrsta sporočila (npr.: discoveryStarted, discoveryFinished, newDeviceFound, ipd.),
- timestamp : veliko število (obvezno) - časovni žig sporočila,
- replyRef : niz znakov (obvezno, kadar je sporočilo odgovor na zahtevo) - unikatni identifikator sporočila in
- data : objekt (opcijsko) - odvisno od vrste spročila

Obstoječe HTTP zahtevke je možno predstaviti kot objekte razreda Message in pri tem uporabiti enake podatkovne konstrukte. Tako lahko za obravnavo Websocket in HTTP zahtevkov uporabimo skupno implementacijo in jo za potrebe sporočanja dogodkov sistema ustrezno razširimo.

V sklopu Websocket API-ja smo definirali naslednje vrste sporočil:

- Sporočila, ki jih sproža sistem (in podpira le Websocket API):
  - discoveryStarted (storitev DeviceMngtService sporoči, da se je iskanje novih naprav v okolici začelo),
  - discoveryFinished (storitev DeviceMngtService sporoči, da se je iskanje novih naprav v okolici končalo),
  - newDeviceFound (storitev DeviceMngtService sporoči, da je bila najdena nova naprava),
  - deviceAdded (storitev DeviceMngtService sporoči, da je bila naprava v sistem uspešno dodana),
  - deviceRemoved (storitev DeviceMngtService sporoči, da je bila naprava iz sistema uspešno odstranjena),
  - scanStarted (storitev ConnectionMngtService sporoči, da se je zaznavanje vidnosti trenutno aktivnih naprav začelo),
  - scanFinished (storitev ConnectionMngtService sporoči, da se je zaznavanje vidnosti trenutno aktivnih naprav končalo),
  - deviceVisibilityChanged (storitev ConnectionMngtService sporoči, da se je vidnost naprave spremenila),
  - connectionSessionStarted (storitev ConnectionMngtService sporoči, da se je začela nova povezavna seja),
  - connectionSessionFinished (storitev ConnectionMngtService sporoči, da se je povezavna seja končala) in
  - measurementRecordReceived (storitev ConnectionMngtService sporoči, da je bila zajeta nova meritev).

- Sporočila, ki jih sproža upravljavec (podprte tudi z zahtevki REST, ki so prikazani na sliki 4.9):
  - deviceDiscovery (uporablja se za upravljanje iskalnika naprav),
  - getDevice (uporablja se za pridobivanje podatkov o napravi),
  - addDevice (uporablja se za dodajanje naprave v sistem),
  - removeDevice (uporablja se za odstranjevanje naprave iz sistema),
  - updateDevice (uporablja se za nastavljanje konfiguracije seje),
  - getDevices (uporablja se za pridobivanje seznama dodanih naprav),
  - getSubscriptions (uporablja se za pridobivanje naročnin),
  - addSubscriptions (uporablja se za dodajanje novih naročnin),
  - updateSubscriptions (uporablja se za spreminjanje statusa aktivnosti in persistentnosti obstoječih naročnin),
  - removeSubscriptions (uporablja se za odstranjevanje naročnin iz sistema),
  - getBluetoothState (uporablja se za pridobivanje trenutnega stanja Bluetooth adapterja),
  - setBluetoothState (uporablja se za vklop/izklop Bluetooth adapterja),
  - getConnectionSessions (uporablja se za pridobivanje povezavnih sej) in
  - removeConnectionSessions (uporablja se za brisanje povezavnih sej).



Slika 4.9: Zaslonska slika, ki prikazuje RESTful API sistema (različica 0.0.3) v orodju Swagger UI.

Z uporabo Websocket API-ja bomo na konkretnem primeru demonstrirali pripravo prehoda za zajem meritev telesne temperature. Imamo pametni digitalni termometer, Philips DL8740, ki ga želimo dodati v sistem HealthySync, mu nastaviti ustrezno konfiguracijo za pripravo sej in zanj ustvariti aktivno naročnino za zajem meritev. Pri tem bomo uporabili naslednja sporočila: deviceDiscovery, addDevice, [testna meritev za inicializacijo naprave], getDevice, updateDevice in addSubscriptions.

Zahtevek, ki ga upravljalec pošlje za začetek iskanja.

```
{
  "uuid": "895667f7-0987-464a-a479-265a8df998f7",
  "type": "deviceDiscovery",
  "timestamp": 1522440674117,
  "data": {
    "duration": 30,
    "action": "start",
    "filters": {
      "devices": [
        {
          "name": "Philips ear thermometer"
        }
      ],
      "services": [
        {
          "uuid": "00001809-0000-1000-8000-00805f9b34fb"
        }
      ]
    }
  }
}
```

Odgovori strežnika po poslanem zahtevku.

takojšnja potrditev zahtevka z ustreznim statusom

```
{
  "data": {
    "statusCode": 202
  },
  "replyRef": "895667f7-0987-464a-a479-265a8df998f7",
  "timestamp": 1524210933581,
  "type": "ack",
  "uuid": "e8dba480-e1b6-440b-9f6f-cd74887621d4"
}
```

obvestilo o najdeni napravi

```
{
  "data": {
    "macAddress": "1C:87:74:08:77:E0",
    "name": "Philips ear thermometer",
    "services": [
      "00001809-0000-1000-8000-00805f9b34fb"
    ]
  },
  "timestamp": 1524210981328,
  "type": "newDeviceFound",
  "uuid": "b6cf5fa7-7738-4f48-82b2-8850e32086f8"
}
```

obvestilo o končanem iskanju


```
{
  "data": {
    "success": true,
    "results": [
      {
        "macAddress": "1C:87:74:08:77:E0",
        "name": "Philips ear thermometer",
        "services": [
          "00001809-0000-1000-8000-00805f9b34fb"
        ]
      }
    ]
  },
  "timestamp": 1524210963571,
  "type": "discoveryFinished",
  "uuid": "b9e78d1c-4548-4e12-ba86-af95f0627cb4"
}
```


Ko smo našli željeno napravo in imamo na voljo potrebne podatke, lahko slednjo dodamo v sistem. V kolikor je za delovanje potrebno seznanjanje, mora biti naprava med tem prižgana in v ustreznem načinu.

Zahtevek za dodajanje naprave.


```
{
  "uuid": "895667f7-0987-464a-a479-265a8df998f7",
  "type": "addDevice",
  "timestamp": 1522440674117,
  "data": {
    "macAddress": "1C:87:74:08:77:E0",
    "requiresBonding": true,
    "passCode": null
  }
}
```


Odgovori strežnika po poslanem zahtevku.

obvestilo o uspešno dodani napravi


```
{
  "data": {
    "success": true
  },
  "timestamp": 1524215766451,
  "type": "deviceAdded",
  "uuid": "d427ec04-7e93-4658-8569-51468204e074"
}
```

obvestilo o začetku iskanja


```
{
  "data": {
    "success": true,
    "activeDevices": [{
      "macAddress": "1C:87:74:08:77:E0"
    }]
  },
  "timestamp": 1524215776463,
  "type": "scanStarted",
  "uuid": "b3c1a82d-7a75-40f8-abbd-61b89c8ee7c9"
}
```


obvestilo o novi povezavni seji (upravljanje testne meritve)


```
{
  "data": {
    "id": 1,
    "initCharacteristicRecords": [],
    "macAddress": "1C:87:74:08:77:E0",
    "startTimestamp": 1524215849257
  },
  "timestamp": 1524215849313,
  "type": "connectionSessionStarted",
  "uuid": "9a71d6fa-997e-42d9-9353-229a9c03056c"
}
```


obvestilo o spremembi vidnosti naprave


```
{
  "data": {
    "macAddress": "1C:87:74:08:77:E0",
    "isVisible": false
  },
  "timestamp": 1524215858004,
  "type": "deviceVisibilityChanged",
  "uuid": "a9824a85-d579-4134-aa81-dd16eb95ae52"
}
```


obvestilo o končani povezavni seji


```
{
  "data": {
    "success": true,
    "connectionSession": {
      "endTimestamp": 1524215883999,
      "id": 1,
      "initCharacteristicRecords": [],
      "macAddress": "1C:87:74:08:77:E0",
      "measurementCharacteristicRecords": [],
      "startTimestamp": 1524215849257
    }
  },
  "timestamp": 1524215884031,
```

```
  "type": "connectionSessionFinished",
  "uuid": "589f1891-0fb8-4211-89fe-f7454adc717b"
}
```

obvestilo o koncu iskanja naprav

```
{
  "data": {
    "success": true
  },
  "timestamp": 1524215893941,
  "type": "scanFinished",
  "uuid": "1fda71a9-0a68-4565-8a87-df48e92a90f4"
}
```

Sedaj je napravi potrebno le še nastaviti ustrezno konfiguracijo seje in ji dodati aktivno naročnino za zajem meritev. V kolikor še ne poznamo karakteristik naprave, jih lahko pridobimo z uporabo naslednjega zahtevka.

Upravljalec pošlje zahtevo za pridobitev podatkov naprave.

```
{
    "uuid": "895667f7-0987-464a-a479-265a8df998f7",
    "type": "getDevice",
    "timestamp": 1522440674117,
    "data": {
      "macAddress": "1C:87:74:08:77:E0"
    }
}
```

Za konfiguracijo seje je potrebno našteti karakteristike, ki jih želimo prebrati na začetku povezave. V našem primeru sta to čas naprave (angl.: Current Time) in različica strojne programske opreme (angl.: Firmware Revision String).

Upravljalec pošlje zahtevo za nastavljanje konfiguracije seje.

```
{
    "uuid": "895667f7-0987-464a-a479-265a8df998f7",
    "type": "updateDevice",
    "timestamp": 1522440674117,
    "data": {
      "macAddress": "1C:87:74:08:77:E0",
      "sessionInitConfig": {
```

```
        "readCharacteristics": [
          {
            "name": "Current Time",
            "uuid": "00002a2b-0000-1000-8000-00805f9b34fb"
          },
          {
            "name": "Firmware Revision String",
            "uuid": "00002a26-0000-1000-8000-00805f9b34fb"
          }
        ]
      }
    }
}
```


Odgovor strežnika po poslanem zahtevku.


```
{
  "data": {
    "statusCode": 202
  },
  "replyRef": "895667f7-0987-464a-a479-265a8df998f7",
  "timestamp": 1524219454232,
  "type": "ack",
  "uuid": "3a9721bd-3836-4fcb-afbe-a7bf915969a6"
}
```


Upravljalec pošlje zahtevo za dodajanje naročnine.


```
{
    "uuid": "895667f7-0987-464a-a479-265a8df998f7",
    "type": "addSubscriptions",
    "timestamp": 1522440674117,
    "data": [
      {
        "managedDevice": {
          "macAddress": "1C:87:74:08:77:E0"
        },
        "isPersistent": true,
        "isActive": true,
        "measurement": {
          "name": "Temperature Measurement",
          "uuid": "00002a1c-0000-1000-8000-00805f9b34fb"
        }
      }
    ]
}
```

Odgovori strežnika po poslanem zahtevku.

obvestilo o uspešni posodobitvi naprave


```
{
  "data": {
    "statusCode": 202
  },
  "replyRef": "895667f7-0987-464a-a479-265a8df998f7",
  "timestamp": 1524221303658,
  "type": "ack",
  "uuid": "d9eea3eb-87bc-474b-aec9-90f83c0a30b7"
}
```


obvestilo o začetku iskanja aktivnih naprav


```
{
  "data": {
    "success": true,
    "activeDevices": [{
      "macAddress": "1C:87:74:08:77:E0"
    }]
  },
  "timestamp": 1524221313701,
  "type": "scanStarted",
  "uuid": "65ae7aad-bc0d-43d3-8521-29c12e4cc19f"
}
```


S tem smo uspešno pripravili prehod za zajem meritev telesne temperature. Na podoben način lahko integriramo tudi druge naprave BLE. Ko je prehod pripravljen za zajem podatkov in meritev, ga lahko integriramo v okolje avtomatiziranega doma OH2.

4.3 Vtičnik HealthySync za OH2

Aplikacija HealthySync se v osnovi uporablja za preprosto beleženje meritev znotraj domačega okolja. Ker jo je mogoče upravljati le s pomočjo Websocket ali HTTP API-ja (saj ni nujno, da prehod za upravljanje podpira grafični vmesnik), smo kot primer odjemalca (angl. client) razvili vtičnik HealthySync z omejeno funkcionalnostjo. Ta omogoča osnoven prikaz zajetih meritev in možnost shranjevanja podatkov v decentralizirano podatkovno omrežje IOTA. Ta nam omogoča, da podatke delimo z ostalimi zunanjimi osebami (kot so npr. zdravnik, medicinska sestra, skrbnik, osebni trener ipd.), z uporabo komunikacijskega protokola MAM (angl. Masked Authenticated Messaging).

4.3.1 Openhab 2

OH2 je odprtokodna programska rešitev, ki se uporablja za avtomatizacijo pametnega doma in temelji na ogrodju Eclipse SmartHome [22]. Sestavljena je iz velikega števila dodatkov (vtičnikov), ki omogočajo integracijo različnih sistemov (za avtomatizacijo), naprav in tehnologij. Ogrodje OH2 vsebuje številne mehanizme za upravljanje sistema (kot so pravila in sprožilci, konfiguracija posameznih elementov, možnosti shranjevanja podatkov ipd.). V okviru OH2 poznamo naslednje osnovne koncepte: *stvar* (angl. thing), *kanal* (angl. channel), *element* (angl. item), *povezava* (angl. link) in *sveženj* (angl. binding).

Vsa komunikacija poteka po asinhronem dogodkovnem vodilu (angl. event bus), ki omogoča spremljanje (uporaba obvestil) in upravljanje (uporaba ukazov) trenutnega stanja sistema.

Stvar predstavlja posamezno zunanjo entiteto, ki jo je mogoče fizično dodati v sistem. To je lahko fizičen (npr. senzor/aktuator) ali virtualen objekt (npr. storitev, ki omogoča dostop do podatkov vremena), ki predstavlja obvladljiv vir informacij in funkcionalnosti.

Element je virtualna predstavitev posamezne funkcionalnosti (stvari), ki

se uporablja pri uporabniških vmesnikih in mehanizmih za avtomatizacijo. Elementi hranijo trenutno stanje in lahko sprejemajo različne ukaze (npr. element Color omogoča spreminjanje vrednosti barve z uporabo naslednjih ukazov: OnOff, IncreaseDecrease, Percent in HSB).

Stvari svoje funkcionalnosti izpostavljajo po posameznih kanalih (komunikacijskih vmesnikov), ki omogočajo komunikacijo med elementi in sistemom. Posamezna vez (asociacija) med kanalom in elementom se imenuje povezava in omogoča upravljanje elementa s pomočjo kanala.

Sveženj (poznamo ga tudi kot dodatek ali vtičnik) je zbirka programskih paketov, ki vsebujejo logiko za integracijo stvari v okolje OH2.

4.3.2 Predstavitev vtičnika

Vtičnik HealthySync je sestavljen iz naslednih stvari: *bridge*, *bleDevice* in *iotaMam*.

Stvar bridge predstavlja virtualen prehod in omogoča, da se s pomočjo Websocket API-ja povežemo s fizičnim prehodom (kjer se izvaja aplikacija HealthySync) in upravljamo iskalnik naprav in stanje adapterja BT. Za to so predvideni naslednji kanali: *websocketConnectionState*, *deviceDiscoveryState* in *bluetoothAdapterState*. Konfiguracijo sestavljajo naslednji parametri: *hostName* (ime gostitelja), *portNumber* (številka vrat) in *isSecure* (pove, ali je povezava zaščitena), ki jih uporablja odjemalec Websocket pri vzpostavljanju povezave s prehodom.

Stvar bleDevice predstavlja virtualno napravo BLE in se uporablja za dodajanje naprave v sistem, odstranjevanje naprave iz sistema in prikaz nekaterih statističnih podatkov (število aktivnih povezav, število meritev, čas zadnje meritve in delež sinhroniziranih sej). Za to se uporabljajo naslednji kanali: *bleDeviceManagedState*, *activeSubscriptions*, *measurements*, *latestSession* in *syncedSessionsProgress*. Konfiguracijo sestavljajo naslednji parametri: *macAddress* (MAC naslov naprave), *pinCode* (geslo za seznanjanje) in *requiresBonding* (pove, ali je seznanjanje potrebno), ki omogočajo dodajanje naprave v sistem.

Stvar iotaMam predstavlja storitev, ki omogoča shranjevanje podatkov posameznih sej v decentralizirano podatkovno omrežje IOTA. Pri tem se uporablja okolje *Node.js*, v katerem se izvaja storitev *mam-service.js*. Ta komunicira z okoljem OH2 (vtičnikom HealthySync) s pomočjo REST API-ja in tehnologije *SSE* (angl. *Server-Sent Events* [13]). Za posodobitev trenutnega stanja storitve (mam-service.js) in podatkov o trenutno sinhroniziranih sejah se uporabljata kanala *iotaMamClientState* in *syncedSessions*. Konfiguracijo sestavljajo naslednji parametri: *mode* (način komunikacije MAM), *seed* (seme), *sideKey* (stranski ključ, ki se uporablja za šifriranje komunikacije pri omejenem načinu) in *macAddresses* (MAC naslovi naprav BLE, za katere želimo podatke sinhronizirati).

Omrežje IOTA in tehnologija MAM

IOTA je tehnologija porazdeljene evidence zapisov (angl. distributed ledger technology, krajše DLT) [9], ki omogoča soglasno in porazdeljeno hranjenje podatkov v obliki posameznih transakcij. Za to skrbi decentralizirano podatkovno omrežje IOTA, v katerem lahko sodeluje vsak, ki se drži določenih pravil (permissionless). Sestavljeno je iz množice medsebojno povezavnih naprav (vozlišč). IOTA je bila ustvarjena s ciljem, da se s pomočjo finančnih spodbud (angl. financial incentive) na področju IoT doseže večja interoperabilnost med napravami različnih proizvajalcev in omogoči M2M ekonomija (angl. machine-to-machine economy) [28]. Kot osnovno denarno sredstvo se uporablja kriptovaluta IOTA, posamezne transakcije pa sestavljajo usmerjen acikličen graf (angl. directed acyclic graph, krajše DAG) (poznan tudi kot *Tangle*). V grafu se vsaka transakcija (z izjemo prve) sklicuje na dve obstoječi transakciji.

Za zaščito pred dvojnim zapravljanjem (angl. double spending) se trenutno uporablja koordinator (posebno vozlišče, ki ga upravlja IOTA Foundation [14]). Velja pravilo, da je transakcija potrjena, ko jo posredno ali neposredno potrjuje transakcija koordinatorja (poznana tudi kot *mejnik* oziroma *milestone*). Transakcije se lahko uporabljajo bodisi za prenos denarnih sredstev

bodisi za prenos podatkov v obliki sporočil. Prenos v IOTA je *sveženj* (angl. *bundle*), ki ga sestavlja množica vhodov in izhodov. Ti se izvedejo atomarno, kar pomeni, da se vse transakcije istega svežnja v omrežje bodisi sprejemejo bodisi zavrnejo. Vsak, ki želi dodati novo transakcijo v Tangle, mora pred tem potrditi dve obstoječi transakciji in izračunati *enkratno kriptografsko število* (angl. *nonce*[2]), ki služi kot mehanizem za zaščito pred smetenjem (angl. spam protection). Zaradi tega pristopa je uporaba transakcij v omrežju IOTA brezplačna in posledično zanimiva za številna področja, saj omogoča tako imenovane *mikrotransakcije*. V tabeli 4.1 je predstavljena zgradba transakcije, ki je v zakodirani obliki velika 2673 trajtov (angl. trytes[3]).

Zaradi brezplačnih transakcij in preproste uporabe smo za deljenje podatkov uporabili protokol MAM. V nadaljevanju bomo na kratko predstavili osnovne konstrukte protokola MAM in delovanje implementirane rešitve za shranjevanje podatkov v Tangle.

Protokol MAM omogoča šifrirano overjeno sporočanje podatkov po komunikacijskih kanalih. Pri tem sodelujeta *sporočevalec* (angl. *publisher*) in *poslušalec* (angl. *subscriber*). Sporočevalec ustvarja nova spročila in jih po kanalih deli z ostalimi poslušalci. Ta podatkovni tok je predstavljen kot *veriga sporočil* [3] (angl. *message chain*).

Za komunikacijo so predvideni naslednji načini:

- javni način (angl. public) – sporočila lahko vidi vsak,
- zasebni način (angl. private) – sporočila lahko vidi samo sporočevalec in
- omejeni način (angl. restricted) – sporočila lahko vidijo zgolj izbrane osebe (poslušalci).

[2]Za računanje enkratnega kriptografskega števila se uporablja računsko zahtevna operacija PoW (angl. Proof of Work).

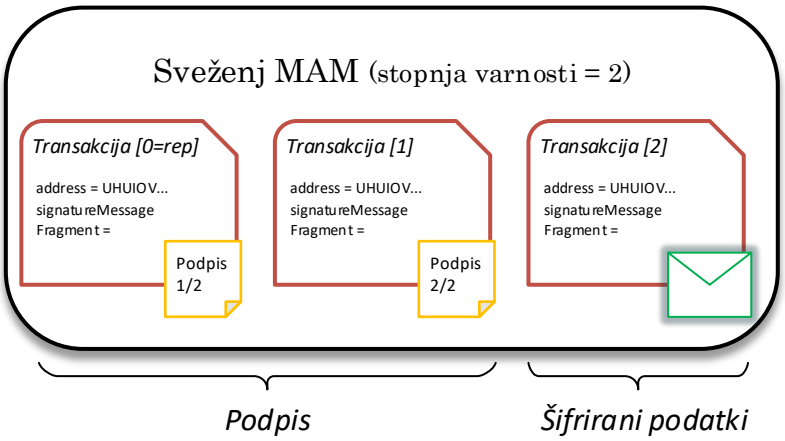
[3]IOTA za predstavitev podatkov uporablja trojiški sistem, kjer je osnovna enota informacije $trit \in \{-1, 0, 1\}$. Posamezne trojčke tritov sestavlja $trajt \in \{9, A, B...M, N, O, P...X, Y, Z\}$, s pripadajočimi desetiškimi vrednostmi $d \in \{0, 1, 2...13, -13, -12, -11... - 3, -2, -1\}$.

Polje	**Podatkovni tip**	**Opis**	**Dolžina**
signatureMessageFragment	niz znakov	Uporablja se za podpisovanje vhoda. Same devetice, če podpis ni potreben in je polje prazno. Za shranjevanje vrednosti spročila.	2187
address	niz znakov	Naslov sprejemnika, če gre za izhod. Naslov pošiljatelja, če gre za vhod.	81
value	celo število	Vrednost transakcije.	27
obsoleteTag	celo število	Uporabniška oznaka (kmalu bo odstranjena).	27
timestamp	celo število	Časovna značka (neobvezna).	9
currentIndex	celo število	Indeks znotraj svežnja.	9
lastIndex	celo število	Zadni indeks svežnja.	9
bundle	niz znakov	Zgoščena vrednost svežnja, ki se uporablja za grupiranje transakcij.	81
branchTransaction	niz znakov	Transakcija, ki se potrjuje.	81
trunkTransaction	niz znakov	Transakcija, ki se potrjuje.	81
attachmentTag	niz znakov	Uporabniška oznaka.	27
attachmentTimestamp	celo število	Časovna značka po POW.	9
attachment TimestampLowerBound	celo število	Spodnja meja časovne značke.	9
attachment TimestampUpperBound	celo število	Zgornja meja časovne značke.	9
nonce	niz znakov	Enkratno kriptografsko število, ki je rezultat POW. Nujno potrebno za dodajanje transakcije v Tangle.	27

Tabela 4.1: Zgradba transakcije, kjer je dolžina predstavljena s številom trajtov.

Ker podatke prenašamo po javnem omrežju IOTA, moramo komunikacijo zaščititi pred nepooblaščenim dostopom in zagotoviti pristnost sporočil. Za to se uporabljata simetrična kriptografija (šifriranje vsebine) in Merklova shema podpisovanja (angl. Merkle signature scheme).

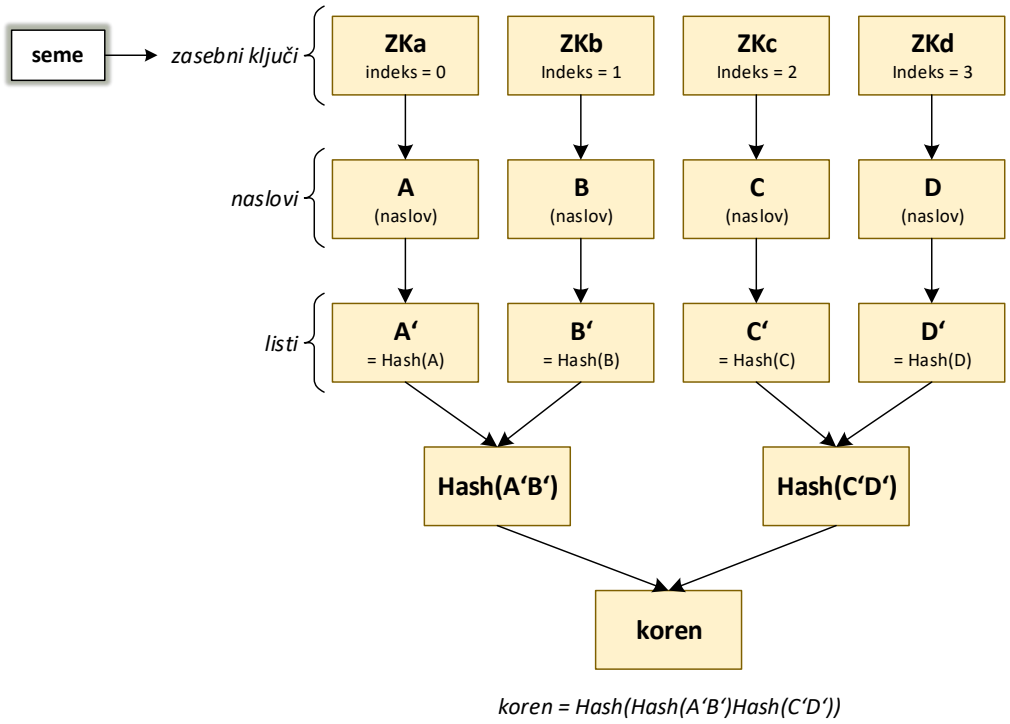
Sporočilo je zajeto v okviru *svežnja* (angl. *MAM bundle*), ki vsebuje šifrirane podatke in podpis (potreben za validacijo pošiljatelja). Za naslov transakcij istega svežnja se glede na način komunikacije uporablja različna vrednost (prikazano v tabeli 4.2).



Slika 4.10: Primer strukture svežnja MAM.

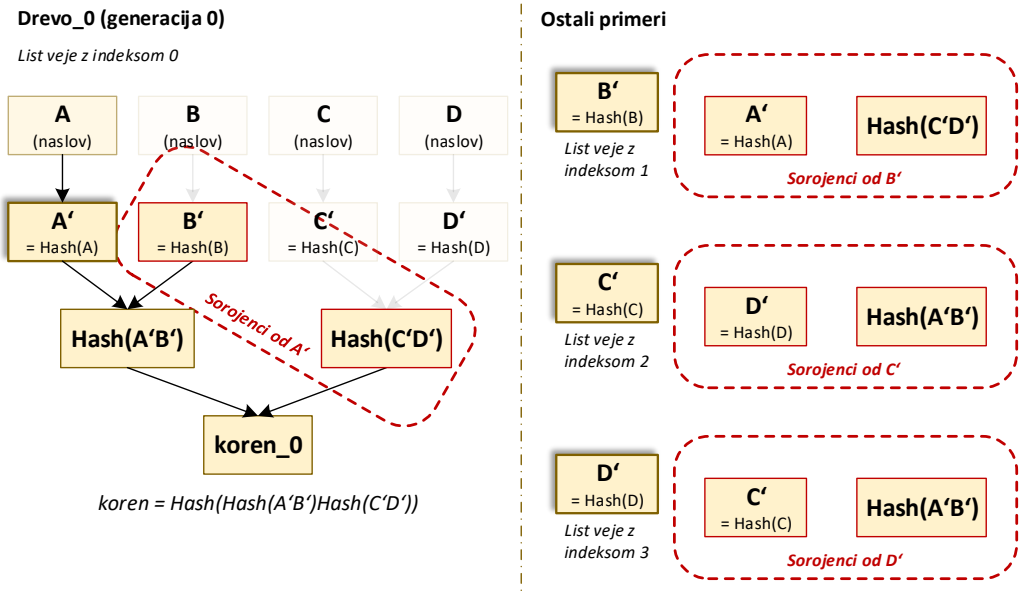
Sporočevalec je entiteta, ki pri generiranju posameznih kanalov uporablja isto *seme*. Na podlagi semena (ki določa identiteto sporočevalca), stopnje varnosti in odmika (branch_index, krajše bi) se generirajo javno-zasebni ključi. Vsaka generacija kanala je predstavljena z Merklovim drevesom, ki ga določata začetni odmik in velikost (število listov). List drevesa A' je zgoščena vrednost javnega ključa A. Primer drevesa z odmikom 0 in velikostjo 4 je predstavljen na sliki 4.11. Pri ustvarjanju nove generacije drevesa je treba zgraditi tudi naslednjo drevo s korenom nextRoot. Vrednost nextRoot se pri kanalu uporabi kot referenca (naslov) za naslednji kanal.

Naslednji pomemben podatek so sorojenci (angl. siblings) lista, saj omogočajo izračun primerjalne vrednosti korena, potrebne za preverjanje pristnosti sporočil. Primer sorojencev prikazuje slika 4.12.



Slika 4.11: Primer Merkle drevesa (0 – generacije z velikostjo 4), ki se uporablja za overjeno sporočanje podatkov. Za generiranje ključev se uporabljajo seme, trenutni indeks in stopnja varnosti (1 – nizka, 2 – srednja in 3 – visoka).

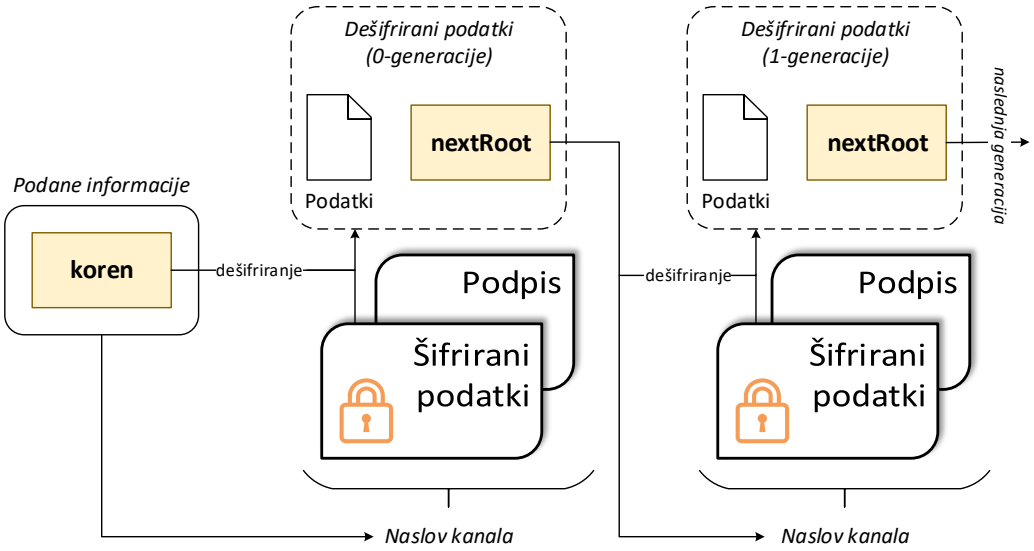
Ker se količina podatkov skozi čas povečuje, se občasno naredijo *vmesni posnetki* (angl. *snapshots*), ki zajemajo zgolj pare posameznih naslovov in pripadajočih denarnih sredstev (ostali podatki se zavržejo). Za trajno hranjenje podatkov skrbijo posebna, *trajnostna vozlišča* (angl. *permanodes*).



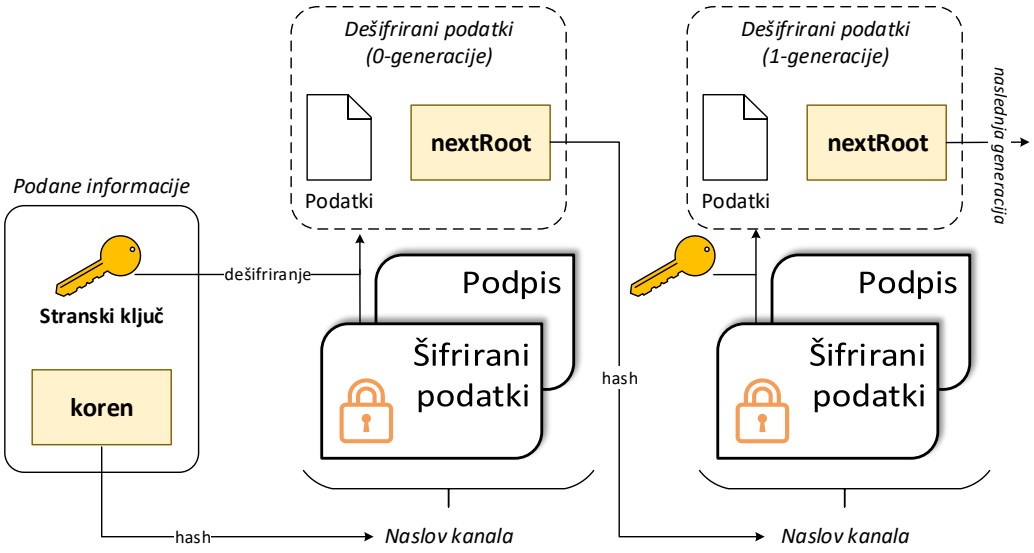
Slika 4.12: Primer sorojencev za Merklovo drevo (0 – generacije z velikostjo 4).

Način komunikacije	Naslov kanala	Ključ za šifriranje
javni	koren	koren
zasebni	Hash(koren)	koren
omejeni	Hash(koren)	stranski ključ

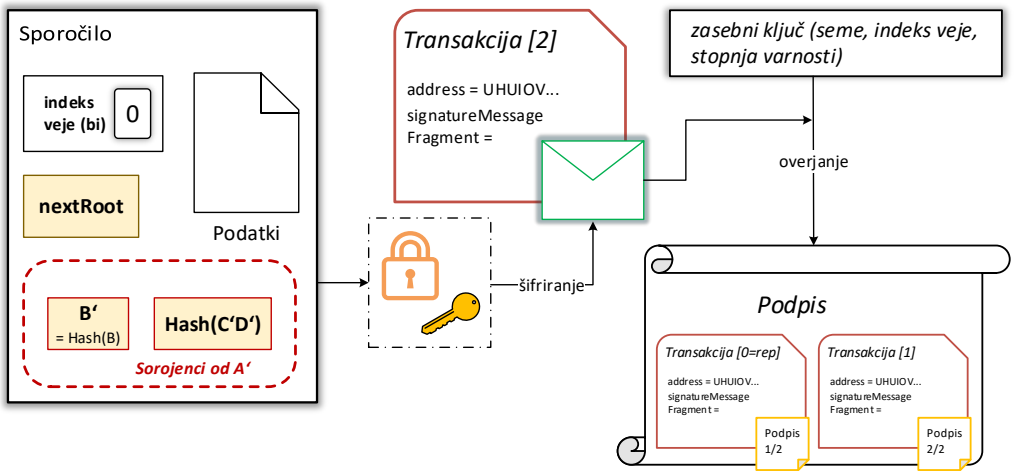
Tabela 4.2: Uporaba naslova in ključa za šifriranje podatkov sporočila pri različnih načinih komunikacije MAM.



Slika 4.13: Javni način komunikacije, kjer se za naslavljanje kanala in dešifriranje podatkov uporablja koren drevesa.



Slika 4.14: Zasebni način komunikacije, kjer se za naslavljanje kanala uporablja zgoščena vrednost korena, za dešifriranje podatkov pa stranski ključ.



Slika 4.15: Prikaz celotnega poteka pošiljanja, ki zajema generacijo trenutnega in naslednjega drevesa, pripravo podatkov (ki so zakodirani v trajtih) ter šifriranje in overjanje sporočila.

Za implementacijo glavnih funkcionalnosti storitve mam-service.js smo uporabili knjižnice *mam.client.js* (za interakcijo z omrežjem IOTA), *axios* (za pošiljanje HTTP zahtevkov), *rxjs* (za lažje delo z asinhronimi dogodki), *eventsource* (za spremljanje dogodkov v okolju OH2) in *data-store.js* (za hranjenje podatkov v datoteki JSON). Storitev mam-service.js s pomočjo SSE čaka na morebitne podatke, ki jih je treba poslati v omrežje IOTA. Ko si uporabnik izmeri meritev, se po končani seji podatki pošljejo v okolje OH2. Od tod se (če je mamClient ustrezno nastavljen) po dogodkovnem vodilu o tem obvesti storitev mam-service.js. Ta podatke ustrezno pripravi in sproži operacijo za dodajanje sporočila v kanal. To traja nekaj časa, saj se za vsako transakcijo svežnja izvaja računanje enkratnega kriptografskega števila (z uporabo algoritma PoW). Po končani operaciji se v datoteko *mam-service-store.json* doda nova sinhronizirana seja z naslednjimi podatki: id, MAC naslov naprave in koren oziroma naslov kanala. O tem se obvesti tudi vtičnik HealthySync, po kanalu syncedSessions, z uporabo ustreznega zahtevka HTTP.

Poglavje 5

Zaključek

Diplomsko delo smo začeli z uvodnim poglavjem, v katerem smo predstavili trenutno finančno stanje in napovedi javnega zdravstva, v katerem se v prihodnje obetajo korenite spremembe. Pomembno vlogo pri tem ima področje IoT, ki omogoča optimizacijo številnih delovnih procesov. Na področju zdravstva se veliko pozornosti namenja razvoju osebnih (personaliziranih) zdravstvenih rešitev, ki temeljijo na kontinuiranem zajemanju zdravstveno relevantnih podatkov. Te v veliki meri ustvarja pacient s pomočjo nosljivih naprav in okoljskih senzorjev. Predstavili smo sklad protokola BLE, ki ga za komunikacijo pogosto uporabljajo zdravstvene merilne naprave, in opisali glavne funkcionalnosti posameznih plasti/profilov.

Praktični del diplomskega dela zajema razvoj sistema HealthySync, ki omogoča integracijo naprav BLE v okolje pametnega doma OH2 in deljenje zajetih povezavnih sej z izbranimi zunanjimi entitetami po porazdeljenem omrežju IOTA. Predstavili smo posamezne gradnike sistema (aplikacijo AndroidThings in vtičnik OH2), uporabljena orodja in ključne tehnologije. Na primeru termometra smo opisali pripravo prehoda za samodejno zajemanje meritev telesne temperature.

Pri razvoju aplikacije smo naleteli na nemalo težav, saj smo uporabljali razvojno različico sistema Android Things Developer Preview 7. Prva težava, ki smo jo uspešno rešili, je bila odkrivanje naprav. Zanje smo implementirali

mehanizem za filtriranje naprav, saj iskanje z uporabo parametra ScanFilter[] ni vračalo rezultatov. Slabost te rešitve je, da jo izvaja CPE (gostitelj), zato je posledično energijsko bolj potratna. Naslednja omejitev je seznanjanje naprave, ki trenutno podpira le način "Just Works". Pri nekaterih napravah smo opazili, da se pri vzpostavljanju povezave občasno zahteva ponovna seznanitev, tudi če smo z njimi že seznanjeni. V ta namen smo pred vzpostavljanjem povezave aktivirali poseben mehanizem za samodejno potrjevanje seznanitve, ki ga po končani seji deaktiviramo (v okviru razreda DeviceManager). Za boljši nadzor posameznih povezav smo implementirali mehanizem za samodejno povezovanje in odkrivanje aktivnih naprav (v okviru razreda ConnectionMngtService). Zadnji pomemben mehanizem predstavlja priprava povezavne seje (v okviru razreda DeviceManager). Opazili smo, da se povezava pogosto ne vzpostavi uspešno. To smo delno rešili z ustreznim mehanizmom, ki v primeru napake ponovno poskusi vzpostaviti povezavo (omejeno z največjim številom ponovitev). Rešitev se izkaže za dokaj učinkovito, saj se povezava v primeru napake pogosto uspešno vzpostavi že ob naslednji ponovitvi. Opazili smo tudi, da iskanje aktivnih naprav včasih preneha delovati, pri čemer je treba ponastaviti adapter BT. To je najverjetneje težava trenutne različice operacijskega sistema, saj sicer zanesljivo deluje na običajnem operacijskem sistemu Android.

Kot primer odjemalca Websocket smo razvili vtičnik OH2. Na začetku smo imeli težave s pripravo razvojnega okolja Eclipse (različica openHAB Development), zato smo se odločili za uporabo virtualnega okolja VirtualBox, v katerem smo poganjali operacijski sistem Ubuntu 18.04 LTS in razvijali vtičnik OH2. Glavni cilj je bil razviti vtičnik OH2, ki bo omogočal shranjevanje podatkov v porazdeljeno omrežje IOTA, kar nam je tudi uspelo. V prihodnje bi bilo treba funkcionalnosti za shranjevanje podatkov v omrežje IOTA in funkcionalnosti za upravljanje prehoda ločiti v dva samostojna svežnja. Prav tako ostaja veliko možnosti za nadaljnje izboljšave (npr. dodati podporo opomnika za opravljanje meritev, omogočiti bolj intuitivno dodajanje in konfiguracijo posameznih stvari ipd.).

Menimo, da je bil cilj diplomskega dela dosežen, saj smo z uporabo pridobljenega znanja razvili delujoč sistem za zajem podatkov in meritev naprav BLE, predstavili inovativen način deljenja podatkov z zunanjimi osebami (z uporabo tehnologije IOTA in protokola MAM) in končnemu uporabniku omogočili popoln nadzor sistema in podatkov.

Literatura

[1] Requirements for support of ubiquitous sensor network (usn) applications and services in the ngn environment. Recommendation, The International Telecommunication Union, Geneva, CH, January 2010.

[2] IERC Cluster SRIA 2014. Ierc-european research cluster on the internet of things. Dosegljivo: `http://www.internet-of-things-research.eu/about_iot.htm`. [Dostopano 30. 12. 2018].

[3] ABmushi. Iota: Mam eloquently explained. Dosegljivo: `https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413`. [Dostopano 13. 01. 2019].

[4] Oliver Amft. How wearable computing is shaping digital health. *IEEE Pervasive Computing*, 17(1):92–98, jan 2018.

[5] Krishnaprasad Thirunarayan Amit Sheth, Utkarshani Jaimini and Tanvi Banerjee. Augmented personalized health. Dosegljivo: `http://wiki.knoesis.org/index.php/Augmented_Personalized_Health:_How_Smart_Data_with_IoTs_and_AI_is_about_to_Change_Healthcare`. [Dostopano 14. 01. 2019].

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, oct 2010.

[7] Debasis Bandyopadhyay and Jaydip Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69, apr 2011.

[8] Luca Catarinucci, Danilo de Donno, Luca Mainetti, Luca Palano, Luigi Patrono, Maria Laura Stefanizzi, and Luciano Tarricone. An IoT-aware architecture for smart healthcare systems. *IEEE Internet of Things Journal*, 2(6):515–526, dec 2015.

[9] IOTA community. Iota documentation. Dosegljivo: `https://github.com/iotaledger/docs`. [Dostopano 12. 01. 2019].

[10] The Advisory Board Company. Cms: Us health care spending to reach nearly 20 Dosegljivo: `https://www.advisory.com/daily-briefing/2017/02/16/spending-growth`. [Dostopano 24. 7. 2018].

[11] Eurostat. Eurostat - data explorer. Dosegljivo: `http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=hlth_sha11_hf&lang=en`. [Dostopano 28. 12. 2018].

[12] Christian Flügel and Volker Gehrmann. Scientific workshop 4: Intelligent objects for the internet of things: Internet of things – application of sensor networks in logistics. In *Communications in Computer and Information Science*, pages 16–26. Springer Berlin Heidelberg, 2009.

[13] Eclipse Foundation. Eclipse smarthome - rest api. Dosegljivo: `https://www.eclipse.org/smarthome/documentation/features/rest.html`. [Dostopano 11. 01. 2019].

[14] IOTA Foundation. The iota foundation. Dosegljivo: `https://www.iota.org/the-foundation/the-iota-foundation`. [Dostopano 12. 01. 2019].

[15] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[16] Eric D. Green, , and Mark S. Guyer. Charting a course for genomic medicine from base pairs to bedside. *Nature*, 470(7333):204–213, feb 2011.

[17] World Bank Group. World bank open data. Dosegljivo: `https://data.worldbank.org`. [Dostopano 28. 12. 2018].

[18] Naresh Gupta. *Inside Bluetooth low energy.* Artech House, London, 2016.

[19] i SCOOP. Internet of things (iot) in healthcare: benefits, use cases and evolutions. Dosegljivo: `https://www.i-scoop.eu/internet-of-things-guide/internet-things-healthcare/`. [Dostopano 29. 12. 2018].

[20] Kno.e.sis. khealth. Dosegljivo: `http://knoesis.org/?q=projects/khealth`. [Dostopano 14. 01. 2019].

[21] Somayya Madakam, R. Ramaswamy, and Siddharth Tripathi. Internet of things (IoT): A literature review. *Journal of Computer and Communications*, 03(05):164–173, 2015.

[22] openHAB Community and the openHAB Foundation e.V. Github - openhab/openhab-docs. Dosegljivo: `https://github.com/openhab/openhab-docs`. [Dostopano 10. 01. 2019].

[23] openHAB Community and the openHAB Foundation e.V. openhub. Dosegljivo: `https://www.openhab.org`. [Dostopano 14. 8. 2018].

[24] Amit Sheth, Utkarshani Jaimini, and Hong Yung Yip. How will the internet of things enable augmented personalized health? *IEEE Intelligent Systems*, 33(1):89–97, 2018.

[25] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164, jan 2015.

[26] Bluetooth SIG. Gatt specifications | bluetooth technology website. Dosegljivo: `https://www.bluetooth.com/specifications/gatt`. [Dostopano 09. 01. 2019].

[27] Bluetooth SIG. Security, bluetooth low energy. Dosegljivo: https://www.bluetooth.com/~/media/files/specification/bluetooth-low-energy-security.ashx. [Dostopano 07. 01. 2019].

[28] David Sønstebø. Iota first chapter synopsis. Dosegljivo: https://blog.iota.org/iota-first-chapter-synopsis-506fdf874437. [Dostopano 12. 01. 2019].

[29] Wikipedia. Cbc-mac - wikipedia. Dosegljivo: https://en.wikipedia.org/wiki/CBC-MAC. [Dostopano 07. 01. 2019].

[30] Wikipedia. Digital health. Dosegljivo: https://en.wikipedia.org/wiki/Digital_health. [Dostopano 14. 01. 2019].

[31] Wikipedia. Wireless sensor network - wikipedia. Dosegljivo: https://en.wikipedia.org/wiki/Wireless_sensor_network. [Dostopano 30. 12. 2018].

[32] Wikipedia. Wireless sensor network - wikipedia. Dosegljivo: https://en.wikipedia.org/wiki/Zigbee. [Dostopano 01. 01. 2019].

[33] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, nov 2014.