

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Novak

**Porazdeljeni sistem za pridobivanje,
hranjenje in analizo podatkov o hrupu
v urbanem okolju**

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Blaž Zupan

Ljubljana, 2021

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil L^AT_EX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogu:

Tematika naloge:

Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode uporabiti, morda bo zapisal tudi ključno literaturo.

Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.

Svoji dragi Nanici.

Kazalo

Povzetek

Abstract

1 Uvod	1
2 Tehnologije	5
2.1 Mikrokontrolerji	5
2.2 Merjenje hrupa	7
2.3 Povezljivost	9
2.4 Programska oprema za internet stvari	11
2.5 Strežniški del	11
3 Rešitev	13
3.1 Uporabniške zahteve	13
3.2 Arhitektura	14
3.3 Merilna enota	16
3.4 Zbirna enota	24
3.5 Poraba energije	31
3.6 Strežnik	35
3.7 Uporabniški vmesnik	42
3.8 Podatkovna analitika	47
4 Primera uporabe	51
4.1 Košenje trave	51

4.2 Glasnost v dnevni sobi	53
5 Zaključek	57
Appendices	59
A Dodatek: Uporabniška navodila	61
A.1 Polnjenje meritnih enot	61
A.2 Prižiganje enot v različnih načinih	61
A.3 Registracija meritne enote	62
A.4 Ustvarjanje raziskave	64
A.5 Urejanje raziskave	64
A.6 Prenašanje podatkov na enote	65
A.7 Postavljanje enot na lokacijo zaznavanja	66
A.8 Uravnavanje intervala zaznavanja	67
A.9 Konec zaznavanja	68
Literatura	70

Povzetek

Naslov: Porazdeljeni sistem za pridobivanje, hranjenje in analizo podatkov o hrupu v urbanem okolju

Avtor: Andraž Novak

Hrup je pri načrtovanju mest prihodnosti velik dejavnik, saj se vedno bolj zavedamo, kakšen vpliv ima na zdravje in kvaliteto življenja. Načrtovalci javnih površin tako vedno več odločitev sprejemajo z namenom obvladovanja hrupa. Za sprejemanje takšnih odločitev potrebujejo podatke o trenutnem stanju in o tem kakšen vpliv imajo posamezne odločitve na količino hrupa. Trenutne rešitve za pridobivanje podatkov o hrupu so drage in zahtevajo veliko načrtovanja, zato se jih velikokrat preskoči. V sklopu diplomske naloge smo razvili fleksibilen, cenovno dostopen sistem, ki omogoča hitro in enostavno profiliranje hrupa na poljubno velikih območjih. Razvili smo cenovno dostopne meritve enote, odprtakodno platformo za upravljanje zaznavanja in shranjevanje podatkov, ter gradnik v programu Orange za učinkovito podatkovno analizo.

Ključne besede: hrup, internet stvari, zaznavanje, mesto, senzorji.

Abstract

Title: Urban noise data sensing and collection with internet of things

Author: Andraž Novak

Noise management is an ever-growing part of designing future urban landscapes. We're becoming more aware of the effects that noise pollution can have on health and quality of life. With that, the pressure on city planners to manage noise more efficiently is becoming more prominent. Noise management relies on a vast amount of information on what the current state of noise is and what kind of impact planning decisions will have. Current solutions for researching noise are often too expensive and time-consuming, so this part of the research is often skipped. As a part of this thesis, a cost-effective, scalable, flexible system for urban noise profiling has been implemented. We've developed measuring units, a user interface, a server for data storage, and an Orange widget for data analysis.

Keywords: noise, internet of things, sensing, city, sensors.

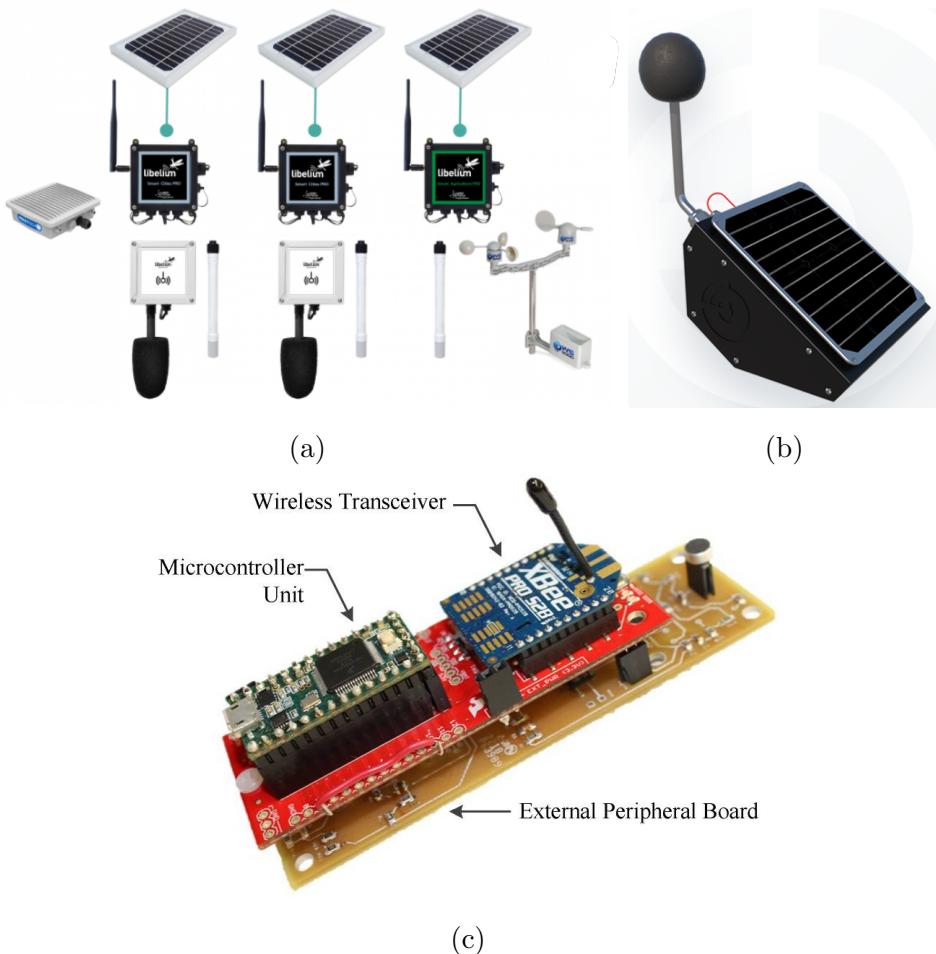
Poglavlje 1

Uvod

Internet stvari je tehnologija s katero lahko upravljamo in nadzorujemo svet okoli nas [8]. Omogoča vse od povezanih pametnih luči, do domačega ogrevanja, do spremeljanja stanja pametnih mest. V domači uporabi se povezane naprave v veliki meri zanašajo na tehnologijo WiFi, ki napravam omogoča prenos podatkov do strežnika. V mestih in širši okolici se uveljavlja uporaba tehnologij LoRa in 5G.

Moderno načrtovanje javnih površin vedno bolj upošteva vpliv hrupa na vedenje in zdravje ljudi. Na oboje lahko vplivamo s premišljeno zasnovo prostorov in izbiro primernih materialov. Pravilne odločitve lahko sprejemamo le s pomočjo primernih podatkov o trenutnem stanju in o uspešnosti takšnih odločitev v ostalih projektih. Vpliv teh odločitev mora biti merljiv. Zato potrebujemo robusten in fleksibilen sistem za merjenje in analizo glasnosti ter tipa hrupa. Tak sistem mora biti cenovno ugoden, prenosen in lahek za uporabo. Upravljanje s potekom raziskav o hrupu in analizi pridobljenih podatkov mora biti intuitivno in prijazno uporabnikom.

S problemom spremeljanja in analize hrupa se ukvarja več podjetij in raziskovalcev (slika 1.1), ki ponujajo različne rešitve. Podjetja Libelium, Cesva in Brüel & Kjær ponujajo zaprtokodne profesionalne rešitve za trajno namestitev. Problem teh rešitev je, da so cenovno nedostopne, in večinoma zahtevajo trajno namestitev na lokaciji. Naša rešitev je prav v teh pogledih



Slika 1.1: Podobne konkurenčne rešitve. Na sliki a je rešitev podjetja Libelium, ki za komunikacijo uporablja protokol LoRaWAN in omogoča napajanje iz sončnih celic. Na sliki b je podobna rešitev podjetja SensorTeam. Na sliki c je rešitev, ki so jo razvili na Ameriškem nacionalnem centru za biotehnološko informatiko. V primerjavi z drugima je veliko bolj cenovno dostopna [14].

boljša. Najbolj podobno rešitev so razvili na Ameriškem nacionalnem centru za biotehnoško informatiko, kjer so z uporabo poceni mikrokontrolerjev in komunikacijskih modulov implementirali meritne enote [14]. Naša rešitev jo v primerjavi z njihovo cenejša in ima boljši način za upravljanje z enotami.



Slika 1.2: Material za štiri zbirne in 20 meritnih enot za potrebe projekta. Zadaj so ohišja za enote, levo spredaj so držala za baterije, na sredi so elektronski deli v antistatičnih vrečkah. Spredaj desno so USB kabli za napajanje in polnjenje.

V sklopu te diplomske naloge smo implementirali nizko cenovni porazdeljen sistem za zaznavanje in analizo hrupa na poljubno velikem območju. Zasnovali in sestavili smo cenovno ugodne meritne in zbirne enote. Implementirali smo sistem za upravljanje s temi enotami in razvili gradnik za podatkovno analizo pridobljenih podatkov v programu Orange. Izvedli smo

tudi nekaj testov za prikaz primerov uporabe.

Poglavlje 2

Tehnologije

Izbor tehnologij temelji na mojem domenskem znanju. Do končne oblike smo ga skrčili s testiranjem in raziskovanjem različnih možnosti v okviru primernih tehnologij. Zastavljeni cilji so nam pri izboru pomagali, saj smo lahko glede na njih oblikovali točne zahteve, ki so jim morale tehnologije zadostiti.

2.1 Mikrokontrolerji

Mikrokontrolerji so ene najmanjših enot računalništva, s katerimi se srečujemo na vsakem koraku našega vsakdanjega življenja. Skoraj vsako električno napravo upravlja vsaj en mikrokontroler. Z napredki v razvoju postaja dodajanje funkcionalnosti mikrokontrolerjem vedno cenejše, zato je na tržišču vedno več enot z zmožnostjo brezžične komunikacije.

Eden takšnih mikrokontrolerjev je ESP32 (tabela 2.1). Nadomestil je mikrokontroler ESP8266, ki je predstavljal prvo nizko cenovno rešitev za povezovanje naprav na internet preko protokola WiFi. Zaradi hitrega procesiranja, veliko perifernih naprav in možnosti nizke porabe energije, je bil perfekten kandidat za naš projekt.

Zaradi nizke cene in veliko funkcionalnosti, je tudi perfekten kandidat za proizvajalce razvojnih ploščic. Te se uporablja za razvoj kode, prototipira-

Ime	ESP32
Proizvajalec	Espressif Systems
Procesor	Tensilica Xtensa LX6 dvojederni + koprocessor
SRAM	512 KiB
Možnosti povezovanja	Wi-Fi 802.11 b/g/n, v4.2 BT + BLE
Največja moč oddajanja	20 dB
Periferne naprave	UART, I2C, I2S, SPI, CAN, ADC, PWM
Napajalna napetost	3.3 V
Poraba električne energije pri 240 MHz	55 mA
Poraba električne energije pri 20 MHz	20 mA
Poraba električne energije v spanju	0.8 mA
Največja poraba	350 mA

Tabela 2.1: Značilnosti mikrokontrolerja ESP32 [7].

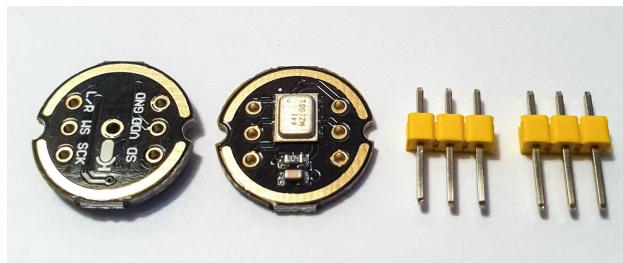


Slika 2.1: Razvojna ploščica TTGO T7 z mikrokontrolerjem ESP32. Uporabili smo jo v merilnih in zbirnih enotah. Izbrali smo jo zaradi majhne porabe energije v načinu spanja in priključka za zunanjo anteno.

nje in kot sestavni del butičnih električnih naprav. Na tržišču je na voljo veliko različnih razvojnih ploščic, ki temeljijo na mikrokontrolerju ESP32. Razlikujejo se po električnih karakteristikah, velikosti vgrajenega spomina, dodatnimi vezji za polnjenje in napajanje iz baterij, vgrajenih senzorjih in prikazovalnikih, priključkih in več. Razvojno ploščico T7 (slika 2.1) podjetja TTGO smo izbrali zaradi priključka za zunanjou anteno, nizke porabe energije in velikim naborom električnih povezav.

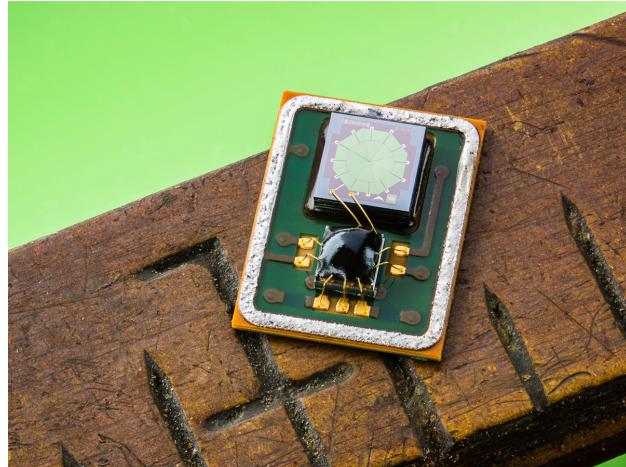
2.2 Merjenje hrupa

Glavni del tega projekta je zaznavanje hrupa. Ta naloga je razdeljena na pridobivanje podatkov in obdelavo teh podatkov. Glavni zahtevi sta konsistencija podatkov med napravami in točnost meritev. Kar se tiče zaznavanja podatkov smo tema dvema zahtevama lahko brez težav ustregli s pravilno programsko kodo [11], pri zaznavanju pa rešitev ni bila trivialna.



Slika 2.2: Mikrofon INMP441. Montiran je na svoji ploščici za lažji dostop do kontaktov. Primeren je zaradi majhne porabe, majhne velikosti, natančnosti meritev in enostavnosti uporabe.

Izbira pravilnega mikrofona je bila tako tu ključnega pomena. Večina mikrofonov deluje po principu zaznavanja sprememb v kapacitivnosti, ki jo povzročajo zvočni valovi. To se običajno dogaja v več komponentah, med katerimi se prenašajo šibki analogni signali. Za točnost takšnih meritev so običajno nujno potrebne kvalitetne komponente in natančna izdelava. Nič od tega pri tem projektu nismo imeli na voljo. Zato smo izbrali mikrofon, ki za to



Slika 2.3: Notranjost mikrofona. Večja struktura je membrana iz silicija, ki zvočne valove pretvori v razlike v kapacitivnosti. Z dvema žicama je povezana s posebno enoto za pretvarjanje analognega signala v digitalnega in pošiljanje preko protokola I2S. Oboje je običajno prekrito s kovinskim ohišjem, ki je bilo za fotografijo odstranjeno.

Ime	INMP441
Proizvajalec	InvenSense
Protokol	24bit I2S
Razpon frekvence vzorčenja	7.8 kHz - 50 kHz
Frekvenčni razpon	60 Hz - 15 kHz
Napajalna napetost	3.3 V
Poraba električne energije v aktivnem stanju	2.2 mA
Poraba električne energije v pripravljenosti	0.8 mA
Poraba električne energije pri spanju	0.0045 mA

Tabela 2.2: Značilnosti mikrofona INMP441. [9]

vse poskrbi z eno komponento in mikrokontrolerju odda le digitalni signal. Tehnologija mikroelektromehanskih sistemov omogoča izdelavo mikrofonov [1] (slika 2.3) z isto tehnologijo kot izdelavo čipov, zato je smiselno na isti komponenti implementirati del, ki fizične spremembe v obliki zvočnih valov spremeni v analogni električni signal in del ki ta analogen signal pretvori v digitalnega.

Mikrofoni, ki temeljijo na tehnologiji MEMS so znani po zelo dobri natančnosti, majhnimi razlikami med komponentami, odlični ceni, majhni porabi energije in majhni velikosti [12]. Za ta projekt smo izbrali mikrofon INMP441 (slika 2.2 in tabela 2.2). Izbrali smo ga, ker je na voljo na svoji razvojni ploščici z dostopom do električnih signalov, dobavljivosti in odlične kompatibilnosti z izbranim mikrokontrolerjem.

2.3 Povezljivost

Pošiljanje podatkov je kompleksen problem, s katerim smo se ukvarjali precej časa. Tehnologija za povezovanje mora ustrezati strogim pogojem glede porabe energije, hitrosti prenosa, načina naslavljanja, dosega signala in cene enote. Preizkusili in raziskali smo precej možnosti (tabela 2.3). In se na koncu odločili za protokol ESP-NOW, ki ga izbrani mikrokontroler podpira brez dodatnih naprav.

Prva in najbolj naivna raziskana možnost je bila protokol WiFi. Ta bi ustrezal našim zahtevam po hitrosti prenosa podatkov, ne bi pa bil primeren s stališča porabe energije, saj povezovanje na omrežje in vzpostavitev povezave s strežnikom trajata predolgo. Pod vprašajem bi bil tudi domet, saj je WiFi primeren za povezavo na nekaj deset metrih. Naslednji protokol, ki smo ga preučili, je bil LoRa. Ta ustreza tako dometu signala, kot tudi porabi energije, ampak je zaradi prepočasnega prenosa podatkov neprimeren za pošiljanje meritev vsako sekundo. Z uporabo te tehnologije bi se cena meritne enote podvojila. Predzadnji preizkušeni protokol je NRF24. Nepriemeren je zaradi omejitve šestih povezav naenkrat, nezanesljivega delovanja



Slika 2.4: Prikaz testiranega dometa v urbanem okolju. Oddaljenost mejnih točk od zbirne točke je 250 m, 160 m in 270 m. Testiranje na prostem je ob optimalnih pogojih pokazalo domet do 1500 m.

	Doseg	Hitrost prenosa	Poraba energije	Cena
ESP-NOW	100 m-1 km	250 kbps	100 mA	4 €
NRF24	100 m-1 km	250 kbps	100 mA	MCU + 4 € za NRF24 modul
LoRa	1 km-10 km	300 bps-19 kbps	30 mA	MCU + 10 € za LoRa modul
WiFi	10 m-100 m	1 mbps-150 mbps	150 mA	4 €

Tabela 2.3: Primerjava različnih tehnologij za povezovanje.

in premajhnega dosega povezave.

Zadnjo možnost smo odkrili, ko smo pregledovali dokumentacijo mikrokontrolerja ESP32 [4]. Proizvajalec teh čipov je namreč omogočil dokaj prosto pošiljanje paketov podatkov med ESP napravami. Protokol ESP-NOW ustrezava vsem našim zahtevam. Hitrost podatkov do 250 kBps, pošiljanje podatkov brez vzpostavitve povezave, kar drastično zmanjša porabo energije, in več kot 1.5 km dosega signala s poceni zunanjim antenom v perfektnih pogojih in do 200 m v urbanem okolju.

2.4 Programska oprema za internet stvari

Izbor programske opreme za internet stvari je bila pri tem projektu zaradi strogih zahtev težka naloga. Zaradi cenovnih omejitev bi morala biti odprtokodna, da bi jo lahko namestili na strežnik v laboratoriju. Zaradi potreb po podatkovni analitiki, bi morala imeti fleksibilen vmesnik. Zaradi strukture projekta in zaznavanja, bi morala podpirati fleksibilno prestavljanje enot z lokacije na lokacijo in razdelitev meritev na študije.

	Oprtakodno	Pregled nad napravami	Fleks. nastavljanje	Vmesnik API
Razviti sistem	Da	Da	Da	Da
Google Cloud	Ne	Da	Ne	Ne
Thinger.io	Da	Da	Da	Da
Kaa IoT Platform	Da	Da	Ne	Ne

Tabela 2.4: Primerjava različnih Programske opreme za internet stvari.

Glede na tabelo 2.4, se raziskane opcije ne ujemajo z našimi zahtevami. To bi ohromilo funkcionalnost celotnega sistema in poslabšalo uporabniško izkušnjo. Odločili smo se, da bo glede na kompleksnost in specifičnost projekta najboljša rešitev, da tak sistem razvijemo sami. Naš sistem tako temelji na odprtakodnih okoljih, je nameščen na strežniku v Laboratoriju za bioinformatiko in podpira naš način nastavljanja in zaznavanja.

2.5 Strežniški del

Zaledni strežnik in uporabniški vmesnik smo zasnovali z uporabo sklada MEAN - MongoDB, Express.js, Angular, Node.js. Uporabili smo ga zaradi enotne kode napisane v jeziku JavaScript. Prav tako se oblika podatkov JSON, uporablja v podatkovni bazi MongoDB, pri procesiranju v okoljih Node.js in Express.js, ter za prikaz podatkov na uporabniškem vmesniku. Ta oblika podatkov je tudi primerna za uporabo z mikrokontrolerji, saj v

primerjavi z obliko XML zahteva občutno manj procesorskega časa in spomina.

Node.js je programsko okolje, namenjeno poganjaju JavaScript kode izven internetnega brskalnika. Skupaj z robustno platformo za upravljanje s paketi in razsiritvami, odlično podporo in učinkovitim delovanjem, je postal priljubljen način za razvoj internetnih aplikacij.

Express.js je okolje za razvoj aplikacij, ki razširja Node.js. Omogoča enostaven in hiter razvoj spletnih aplikacij in vmesnikov API.

MongoDB je NoSQL podatkovna baza, ki ne zahteva strogo določene oblike podatkov, ampak omogoča shranjevanje podatkov po principu dokumentov v obliki JSON. Omogoča indeksiranje dokumentov za hitrejše iskanje, agregacije po podatkih in definiranje delovnih cevi v jeziku JavaScript za napredno delo s podatki.

Poglavlje 3

Rešitev

Razviti sistem sestavlja več povezanih enot, ki smo jih posamično razvijali. Najprej smo določili uporabniške zahteve in iz tega cilje, ki smo jih želeli z razvitim sistemom doseči. Za tem smo se lotili implementacije po komponentah. Najprej smo razvili zaledni del, potem smo razvili merilno in zbirno enoto, za tem uporabniški vmesnik, nazadnje pa gradnik za podatkovno analitiko.

3.1 Uporabniške zahteve

Cilj tega projekta je bil implementacija sistema za podrobno razumevanje hrupa v poljubno velikih delih okolja. Zato smo se odločili za fleksibilen porazdeljen sistem za zaznavanje in analizo hrupa. Določili smo, da bo sistem omogočal prosto razporejanje merilnih enot, ki bodo sinhronizirano, v specifiranih časovnih intervalih, zajemale podatke o hrupu in jih posredovale v podatkovno bazo, kjer bodo potem dostopne za analizo v programu Orange. Za analizo potrebujemo podatke o glasnosti in podatke o tipu hrupa, zato smo določili, da bomo zbirali podatke o jakosti hrupa in rezultate spektralne analize tega hrupa.

Velik poudarek smo pri tem projektu namenili cenovni učinkovitosti. Na tržišču obstaja veliko podobnih rešitev, ki po našem mnenju stanejo preveč.

Zato je bila ena naših zahtev to, da zaledni sistem temelji na odprtokodnih okoljih in je nameščen na strežniku, ki je del laboratorija za bioinformatiko. Prav tako morajo biti meritne enote veliko bolj cenovno dostopne kot podobne enote, ki so trenutno na voljo.

Večina obstoječih meritnih enot poleg velikega finančnega vložka zahteva tudi zunanje napajanje in trajno namestitev. Tako smo zanje določili, da morajo biti dovolj majhne in lahke, da z montiranjem ne bomo imeli prevelikih težav. Hkrati morajo imeti dovolj majhno porabo, da lahko vsaj en teden delujejo z napajanjem iz vgrajenih baterij in tako ne potrebujejo zunanjega napajanja.

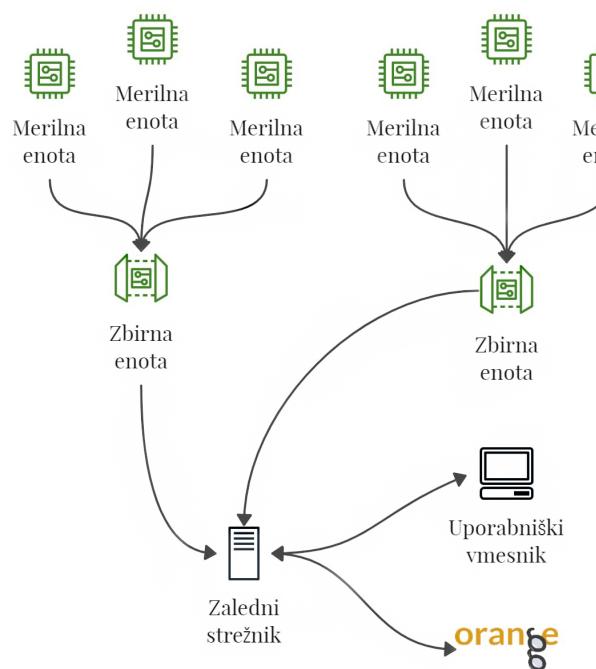
Ker želimo zbrane podatke obdelati in prikazati, smo se odločili, da bomo omogočili dostop do podatkov v programu Orange. Tako smo se odločili implementirati svoj gradnik za program Orange, ki bo podpiral poizvedbe na zalednem delu.

3.2 Arhitektura

Arhitektura razvitega sistema (slika 3.1) temelji na toku podatkov. Zasnovali smo jo z upoštevanjem omejitev izbranih tehnologij in jo prilagodili tako, da kar najbolje izkoristi prednosti vseh delov sistema.

Osrčje celotnega sistema je zaledni strežnik, ki koordinira pot podatkov. Za delo s podatki smo implementirali vmesnik REST API. Tako lahko povezani uporabniki in enote dodajajo, spreminjajo, berejo in brišejo podatke. Na strežniku gostimo tudi podatke za uporabniški vmesnik, ki je implementiran v obliki spletnne strani.

Delovanje meritnih in zbirnih enot temelji na izbiri tehnologije za prenos podatkov o meritvah. Ker meritne enote nimajo neposrednega dostopa do interneta, smo pot podatkov speljali skozi zbirne enote, ki lahko istočasno komunicirajo z zalednim strežnikom in z meritnimi enotami. Zbirne enote tako delujejo kot posredniki, ki sami po sebi ne ustvarjajo novih podatkov, ampak jih samo prenašajo iz enega medija na drugega.



Slika 3.1: Struktura razvitega sistema. Merilne enote so na robu in komunicirajo z zbirnimi enotami. Nanjo pošiljajo zajete podatke o hrupu. Zbirne enote prejete podatke z merilnih enot posredujejo na zaledni del. Tam se shranijo in do njih lahko dostopa gradnik v programu Orange in uporabniški vmesnik.

Uporabniški vmesnik je namenjen upravljanju in pregledu celotnega sistema. Pot podatkov je tu dvosmerna, saj po eni strani z njim uporabnik nastavlja študije in jim dodeljuje fizične enote, po drugi strani pa lahko tam pregleduje osnovne podatke o stanju enot, poteku zaznavanja in povprečno glasnost meritev.

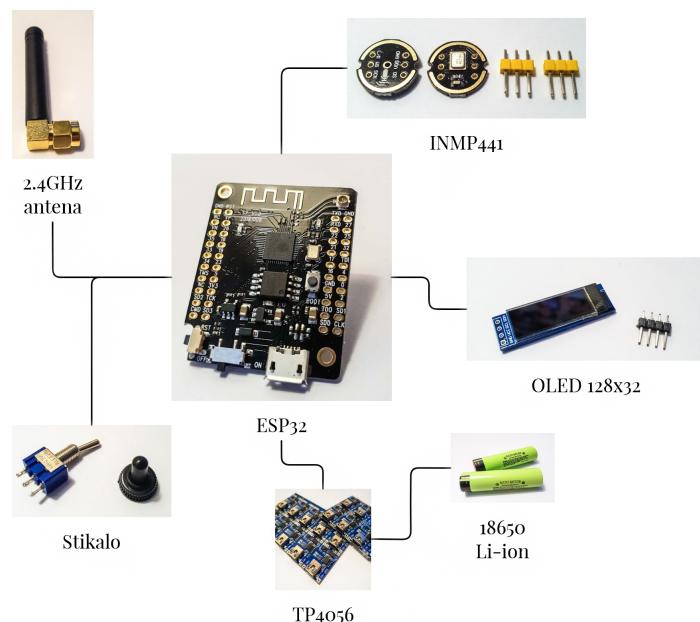
Gradnik v programu Orange je namenjen dostopu do podatkov. S strežnikom komunicira z zahtevki HTTP in prejete podatke spreminja v obliko, ki je primerna za uporabo z drugimi gradniki.

3.3 Merilna enota

Merilna enota je zaključen del sistema, ki je odgovoren za zbiranje in pošiljanje podatkov na zbirno enoto. Zasnova enote temelji na uporabniških zahtevah in omejitvah tehnologije. Odločili smo se, da bomo uporabili prosto dostopne komponente, ki jih lahko z malo dela z žicami povežemo in montiramo v ohišje. To v primerjavi z oblikovanjem svojih tiskanih vezij zmanjša možnosti za napake in omogoča lažja popravila in menjavo delov enote, ko je ta že postavljena.

Glavni del je mikrokontroler ESP32 na razvojni ploščici T7 podjetja TTGO. Za ta projekt je primerna zaradi majhne porabe v načinu spanja in priključka za zunanjo anteno [6]. Nanjo se povezujejo vsi ostali deli merilne enote. Zaznavanje zvoka smo omogočili z MEMS mikrofonom INMP441, ki je s protokolom I2S povezan z mikrokontrolerjem. Ker je ta povezava povsem digitalna, smo eliminirali veliko problemov s kvaliteto žic in spojev. Za pomoč pri nastavljanju enote smo se odločili, da bo vsaka enota opremljena z OLED ekranom SSD1306 z ločljivostjo 32 krat 128 slikovnih točk. Z mikrokontrolerjem komunicira s protokolom I2C.

Za napajanje skrbita dve 18650 Li-ion polnilni bateriji, ki lahko ob polni napolnjenosti in zaznavanju hrupa vsako sekundo napajata enoto približno tri tedne. Za polnjenje in varno uporabo smo izbrali vezje TP4056, ki skrbi, da baterije niso prenapolnjene ali preveč prazne in da skozi baterije ne steče



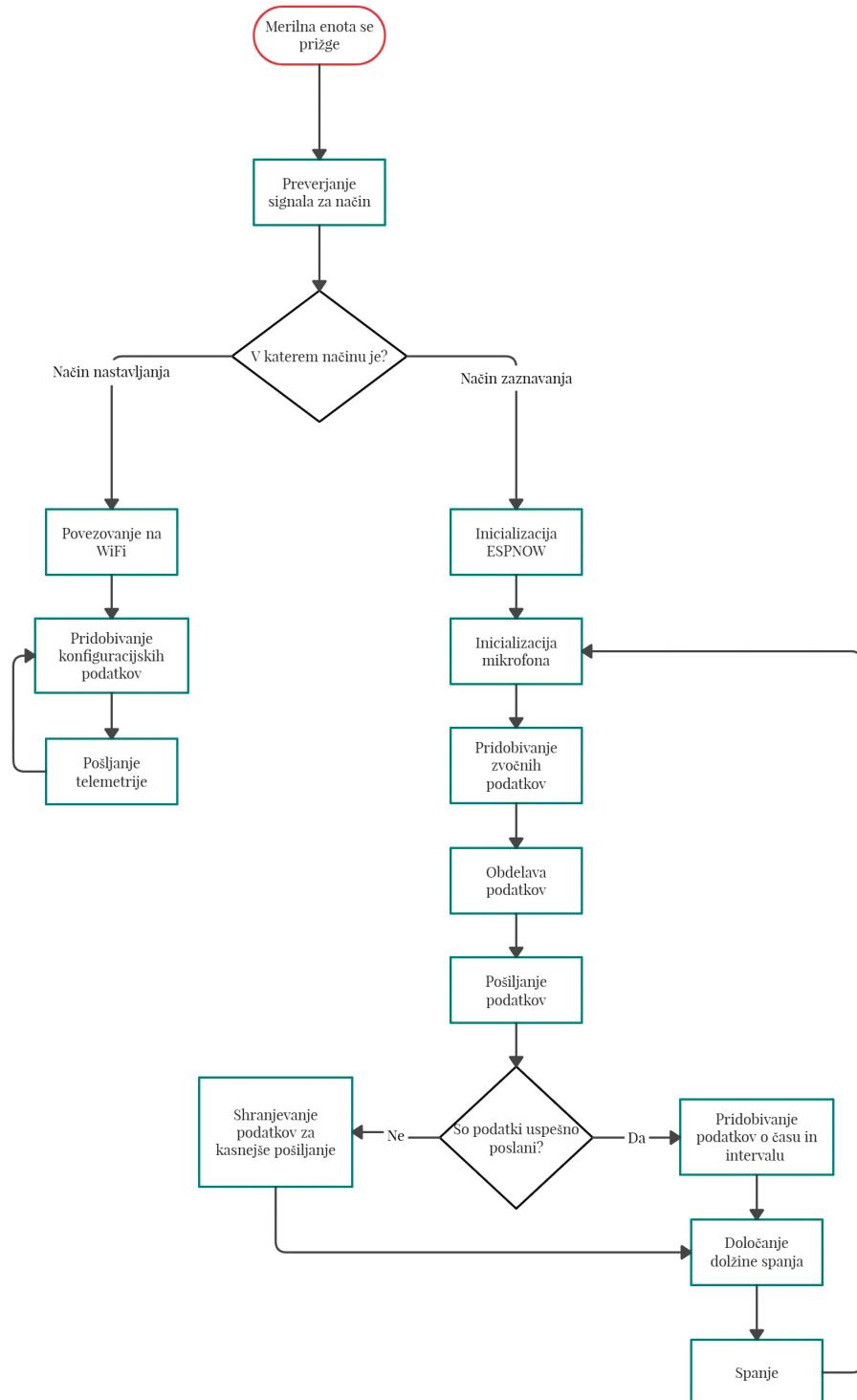
Slika 3.2: Merilna enota z vrisanimi povezavami med sestavnimi deli. Osrčje vsake enote je mikrokontroler ESP32, ki sprejema podatke, jih obdeluje in pošilja na zbirne enote. Podatke zbira s pomočjo mikrofona INMP441. Za lažje nastavljanje se ob zagonu glavni podatki o enoti izpišejo na OLED ekranu. Enoto napajata dve 18650 Li-ion polnilni bateriji, za kateri skrbi vezje TP4056. Za daljši doseg signalov, smo uporabili zunanjou anteno.

previsok tok.

Merilna enota podpira dva načina delovanja - način za nastavljanje in način za zaznavanje. Med njima lahko uporabnik preklaplja s stikalom, ki je montirano na zunanjji strani enote.

Način za nastavljanje (slika 3.3, leva stran) potrebujemo, ker je na vse enote naložena enaka programska koda, kar zmanjša zmedo pri nalaganju kode in olajša nastavljanje. Ko enota začne z delovanjem v načinu za nastavljanje, se poveže na določeno WiFi omrežje, preko katerega komunicira z zalednim delom. Osnovna identifikacija poteka z uporabo MAC naslova, ki je v vsakem mikrokontrolerju unikaten. Ko zaledni del prejme naslov, v bazi podatkov ustvari nov zapis za merilno enoto, kamor ga shrani za prihodnje identifikacije, dodeli identifikacijsko številko za notranjo identifikacijo in enoti naključno dodeli ljudem razumljivo ime. V odgovoru na MAC naslov enoti odgovori z MAC naslovom zbirne enote, če je ta zbirna enota trenutno dodeljena študiji. Enota si naslov zbirne enote shrani v spomin za kasnejšo uporabo, na ekranu pa se izpišejo podatki o imenu, trenutni jakosti zvoka in napetosti na bateriji. Enota potem periodično na zaledni del pošilja podatke o napolnjenosti baterije in preverja, če je ta enota dodeljena drugi zbirni enoti.

Način za zaznavanje (slika 3.3, desna stran) je namenjen zbiranju in pošiljanju podatkov na zbirno enoto. Ko enota začne z delovanjem v načinu zaznavanja, najprej inicializira brezžični vmesnik za komunikacijo s protokolom ESP-NOW. Temu sledi inicializacija mikrofona in zbiranje ter obdelava podatkov. Obdelani podatki se skupaj s časovno oznako zapišejo v sporočilo, ki se doda v vrsto. Sledi pošiljanje, ki smo ga implementirali paketno z naključnim intervalom med paketi sporočil. To pomaga pri zmanjševanju možnosti za trčenje paketov in za zmanjševanje porabe energije. Ko enota pošilja podatke, vzame sporočilo iz vrste sporočil in ga poskusi poslati. Če je pošiljanje uspešno, je sporočilo zbrisano, če pošiljanje ni uspešno, se sporočilo ohrani v vrsti, nakar enota določi nov čas naslednjega pošiljanja. Na koncu določi čas spanja do naslednjega zaznavanja in začne spanje. Poleg tega



Slika 3.3: Merilna enota lahko deluje v dveh načinih. V načinu za nastavljanje enota pridobi podatke o delovanju z zaledja in pošilja svojo telemetrijo. V načinu zaznavanja enota zajema podatke o zvoku, jih obdelava in jih periodično pošilja na zbirno enoto.

enota vsake tri sekunde pošilja podatke o napolnjenosti baterije in zahteva podatke o točnem času in o trenutnem intervalu zaznavanja.

```
void sensing_and_data_preparation(){
    setCpuFrequencyMhz(20);
    // set cpu frequency to 20mhz to lower the consumption

    // get noise data
    sensing_start = get_secs();
    get_samples((int*)&samples_pub);

    setCpuFrequencyMhz(240);
    // set cpu frequency to 240mhz for processing

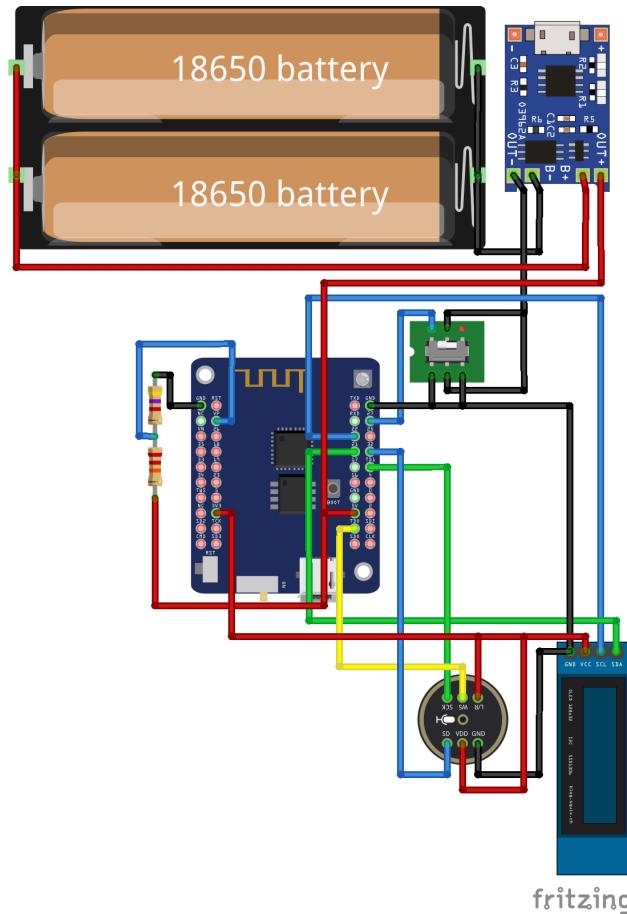
    // process the data
    calculate_fft((int*)&samples_pub, (double*)&fft_downsampled, DOWNSAMPLED__FFT);
    decibels = 0.0;
    decibels = calculate_decibels((int*)&samples_pub, SAMPLES_SIZE);

    setCpuFrequencyMhz(20);
    // set cpu frequency to 20mhz to lower the consumption

    // put data into a sending queue
    add_to_sending_queue((double*) &fft_downsampled, decibels, sensing_start);
}
```

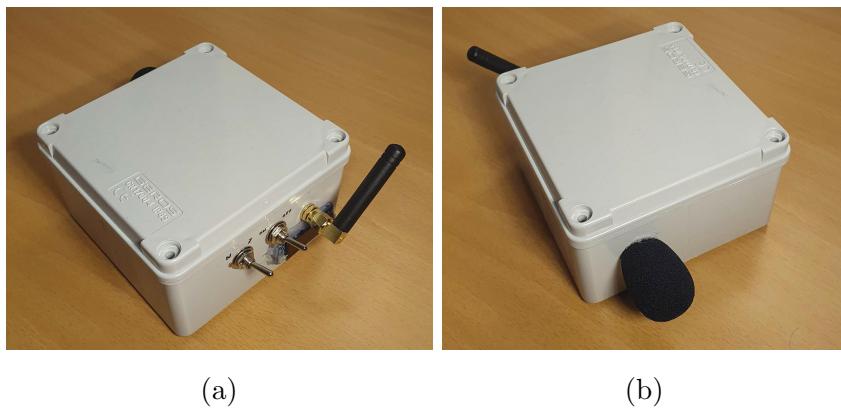
Programska koda [3], ki se izvede ob vsakem zajemanju in obdelavi podatkov. Mikrokontroler se najprej nastavi na najmanjšo frekvenco delovanja, saj med zajemanjem zvočnega signala večino časa ne dela nič računsko zahtevnega in čaka, da se medpomnilnik napolni. Sledi procesiranje podatkov, zato

se nastavi na najhitrejši način delovanja, kjer na podatkih izvede spektralno analizo in izračuna jakost zvoka. Na koncu nastavi frekvenco na najnižjo in podatke prepiše v vrsto sporočil.

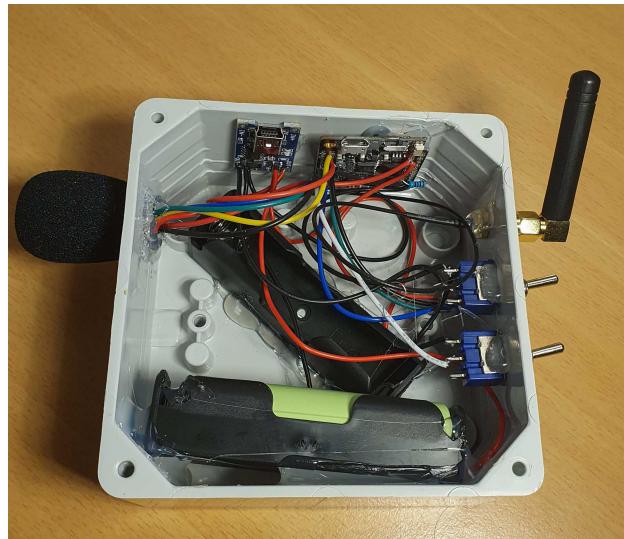




Slika 3.5: Material za meritve. Na sliki sta dve bateriji 18650 z držaloma, upravljalno vezje za baterije TP4056, mikrofon INMP441, stikalo za vklop in nastavljanje načina delovanja, prikazovalnik SSD1306, razvojna ploščica TTGO T7 z mikrokontrolerjem ESP32, antena in kabel za anteno, pena za zaščito mikrofona in ohišje v katerem je vse montirano. Manjkajo žice, ki povezujejo različne dele enote in vroče lepilo, ki smo ga uporabili za pritrjevanje v ohišje.



Slika 3.6: Sestavljeni model meritvene enote iz dveh zornih kotov. Na sliki a se vidijo prikazovalnik, antene in stikala, na sliki b se vidi pokrivalo mikrofona namenjeno dušenju vetra in zaščiti pred dežjem.



Slika 3.7: Notranjost sestavljenega merilnega enot. Vidijo se povezave med mikrofonom, ekranom, polnilnim vezjem, stikali in TTGO T7. Vse povezave so barvno koordinirane s celotnim projektom. Luknje so zatesnjene z vročim lepilom, za preprečevanje vdora vode.

da sta na eni strani vezana skupaj - to predstavlja signal, na eni strani je en povezan na negativni terminal baterije, na drugi strani pa je drugi povezan na pozitivni terminal baterije. Tako ta konfiguracija ustvari napetostni gradient, kjer 6 V na strani, ki je povezana na baterijo ustreza signalu 1,1 V, 0 V na bateriji pa ustreza signalu 0 V. Tako se lahko določi napetost na bateriji z merjenjem signala, ki se vedno primerja z napetostjo 1,1 V.

Komponente merilne enote (slika 3.5) smo povezali z v naprej narezanimi žicami. Pri tem smo upoštevali enotno barvno paletto (slika 3.4) za hitrejše sestavljanje in lažja popravila. Tako sta rdeča in črna uporabljeni za napajanje, zelena je uporabljena za prenos urinega signala, modra je uporabljena za prenos podatkov in rumena je uporabljena za prenos ostalih podatkov - v tem primeru prenaša signal, da poteka prenos podatkov z mikrofona.

Vse skupaj je montirano v plastičnem ohišju (slika 3.6 , in 3.7), ki ima tesnilo proti vdoru vode in prahu. Za mikrofon, ekran, stikalo in anteno smo v ohišje zvrtili luknje, ki smo jih potem zatesnili z vročim lepilom. Mikrofon

smo prekrili s peno, ki zmanjšuje verjetnost vdora vode, napačnih podatkov zaradi vetra in manjšo verjetnost za zamašitev mikrofona s prahom in ostalo umazanijo.

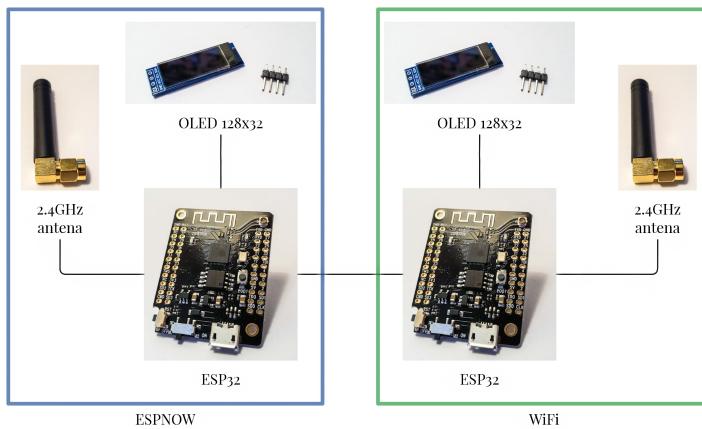
V sklopu izdelave meritnih enot smo optimizirali tudi čas, ki ga namenimo izdelavi vsake enote. Najbolj učinkovito smo čas porabili ko smo se lotili sestavljanja več enot naenkrat. Korake smo izvajali v istem vrstnem redu kot če bi sestavljali eno enoto, le da smo ga ponovili za vsako enoto posebej. Tako smo naenkrat zvrtili vse luknje v vsa ohišja, naenkrat smo spajkali povezave na posamezno komponento naenkrat za vse enote. Tako smo pri izdelavi štirih enot naenkrat v povprečju porabili dobro uro na enoto.

3.4 Zbirna enota

Zbirna enota (slika 3.8) je zaključen del sistema, ki je odgovoren za sprejemanje sporočil z meritnih enot, časovno usklajevanja njihovega delovanja in posredovanje podatkov na zaledni del. Zbirne enote potrebujemo, ker meritne enote v načinu zaznavanja ne morejo komunicirati z zalednim delom, ker nimajo dostopa do interneta. Zato smo razvili enoto, ki sprejme pakete, ki so poslani po protokolu ESP-NOW in jih preko WiFi povezave pošlje na zaledni del.

Osrče te enote sta dva mikrokontrolerja ESP32 na razvojnih ploščah T7. Dva potrebujemo, ker lahko en komunicira naenkrat samo z enim protokolom - torej prvi lahko komunicira samo s protokolom ESP-NOW, drugi pa samo s protokolom WiFi. Podatke si izmenjujeta po serijski UART povezavi s protokolom PJON [15]. Ta skrbi za zanesljivo prenašanje sporočil in naslavljjanje.

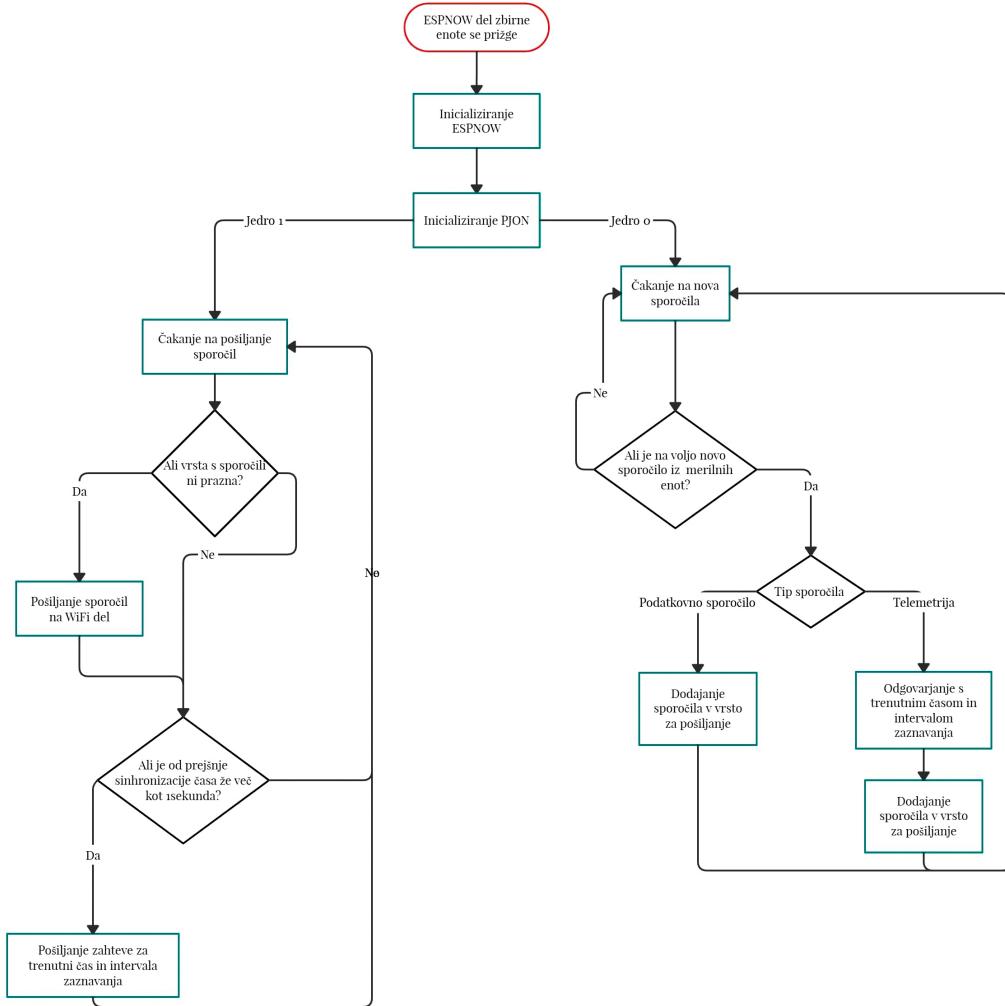
Vsaka zbirna enota je opremljena z dvema OLED prikazovalnikoma, ki uporabniku sporočata uporabne podatke - količina preostalega RAM pomnilnika, povprečen povratni čas do zalednega dela, koliko sporočil iz meritnih enot je bilo sprejetih v zadnji sekundi, moč WiFi signala, MAC naslov te enote, dodeljeno ime enote in število sekund delovanja brez napak.



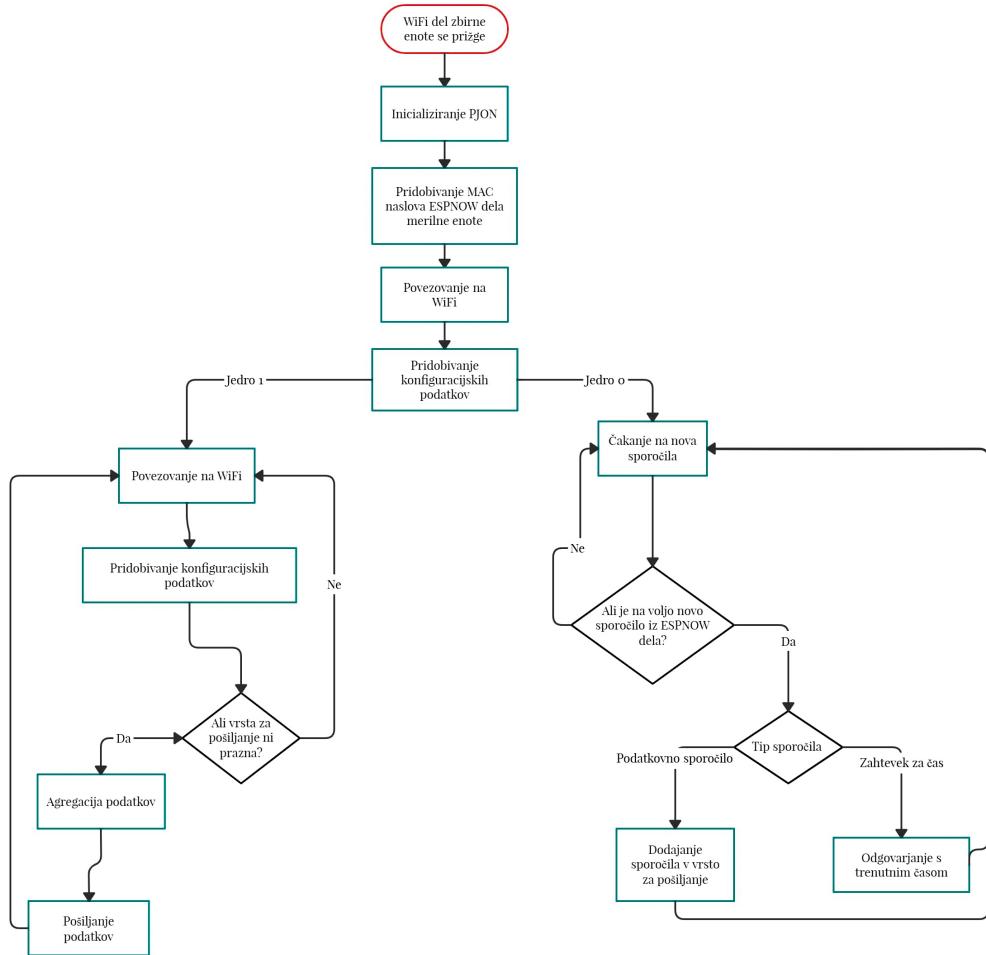
Slika 3.8: Koncept zbirne enote. Razdeljena je na del, ki komunicira s protokolom WiFi, in na del, ki komunicira s protokolom ESP-NOW. Povezana sta z UART serijsko povezavo preko katere komunicirata s protokolom PJON. Takšna zasnova je potrebna, ker ESP32 ne more istočasno komunicirati preko obeh protokolov.

Posebnost ESP32 je, da ima dvojedrni procesor. V primeru merilne enote je to zaradi večje porabe hiba, pri zbirni enoti pa smo to uporabili v našo korist, saj smo lahko pri vsakem mikrokontrolerju eno celo jedro namenili komunikaciji z drugim delom enote, drugo jedro pa komunikaciji s protokolom ESP-NOW ali s protokolom WiFi. Za vsako aktivnost smo napisali svoj proces in ga dodelili svojemu jedru. Pri tem smo si pomagali z operacijskim sistemom FreeRTOS. Za komunikacijo med deloma enote smo definirali posebna sporočila, ki so namenjena prenosu podatkov o meritvah, zahtevah za sinhronizacijo časa, sinhronizaciji časa in pošiljanju telemetričnih podatkov. S tem smo ukrotili asinhrono delovanje mikrokontrolerjev in ustvarili iluzijo, da je zbirna enota ena homogena celota.

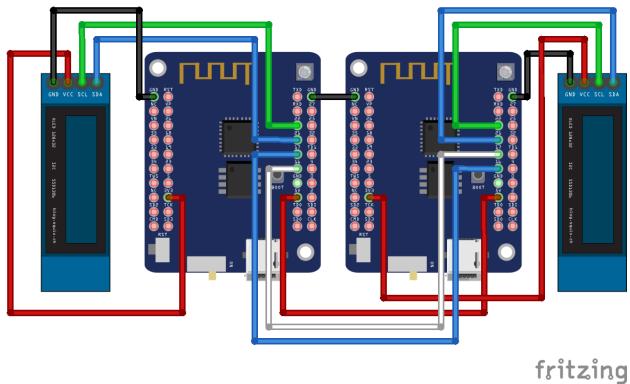
Del zbirne enote, ki komunicira s protokolom ESP-NOW (slika 3.9) z merilnimi enotami je zadolžen za sprejem podatkov, sinhronizacijo časa in sprejem telemetričnih podatkov. Ob zagonu najprej inicializira komunikacijo s protokoloma ESP-NOW in PJON, potem pa vsakemu jedru dodeli svoj proces. Jedro 0 potem čaka na sporočila z merilnih enot. Ko sporočilo



Slika 3.9: Potek delovanja dela zbirne enote, ki komunicira z merilnimi enotami. Ker ima ESP32 dvojedrni procesor, se potek programa razdeli v dva dela, ki se izvajata istočasno. Jedro 0 je zadolženo za brezžično komunikacijo z merilnimi enotami preko protokola ESP-NOW, jedro 1 pa je zadolženo za komunikacijo z drugim delom zbirne enote s protokolom PJON.



Slika 3.10: Potek delovanja dela zbirne enote, ki komunicira z zalednim delom. Ker ima ESP32 dvojedrni procesor, se potek programa razdeli v dva dela, ki se izvajata istočasno. Jedro 0 je zadolženo za komunikacijo z delom zbirne enote, ki iz merilnih enot prejema podatke o hrupu. Jedro 1 je zadolženo za komunikacijo z zalednim delom, kamor posreduje podatke iz merilnih enot.



Slika 3.11: Električne povezave v zbirni enoti. Barve povezav se ujemajo s celim projektom. Rdeča in črna sta namenjeni napajanju, modra je namenjena prenosa podatkov, zelena pa urnemu signalu. Posebnost je UART povezava med enotama, ki je tu prikazana z modro in belo povezavo.



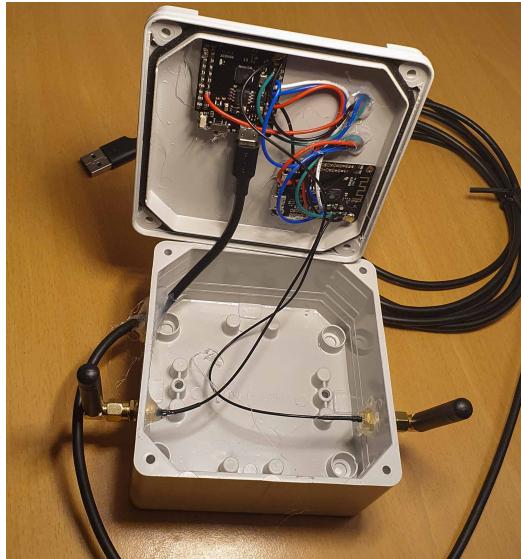
Slika 3.12: Material za zbirno enoto. Uporabili smo dve razvojni ploščici TTGO T7, dva prikazovalnika SSD1306, dve anteni s kabli in ohišje. Manjkajo žice, ki povezujejo sestavne dele, vroče lepilo, s katerim smo zatesnili luknje v ohišju in pritrdili električne dele v ohišje, in napajalni kabel, ki mora biti med delovanjem priklopiljen v USB polnilnik.

prejme, najprej določi tip sporočila. Če sporočilo vsebuje podatke o meritvi, jih prepiše v vrsto za pošiljanje na WiFi del. V nasprotnem primeru gre za sporočilo s telemetričnimi podatki na katere odgovori s trenutnim časom v milisekundah, telemetrične podatke pa prav tako zapiše v vrsto za pošiljanje na WiFi del. Jedro 1 je zadolženo za prenašanje podatkov na WiFi del enote in za sinhronizacijo časa. Proces vsako sekundo s posebnim sporočilom zahteva podatke o trenutnem času in hkrati preverja, če je v vrsti za pošiljanje novo sporočilo. V primeru da je, to sporočilo prenese na drug del enote.



Slika 3.13: Zbirna enota. Na pokrovu sta montirana dva prikazovalnika, pri straneh sta dve anteni in na eni strani je napajalni USB kabel.

Del zbirne enote, ki komunicira s protokolom WiFi (slika 3.10) z zalednim delom, je zadolžen za registracijo enote, sinhronizacijo časa in prenos podatkov o meritvah in o telemetriji na zaledni del. Ob zagonu najprej inicializira komunikacijo s protokoloma WiFi in PJON in pošlje MAC naslov enote na zaledni del za identifikacijo, nato vsakemu jedru dodeli svoj proces. Jedro 0 Čaka na sporočila. Če je sporočilo podatkovno, ga doda v vrsto za pošiljanje na zaledni del, v nasprotnem primeru odgovori s podatki o trenutnem času. Jedro 1 najprej inicializira časovno sinhronizacijo s protokolom NTP, nato z

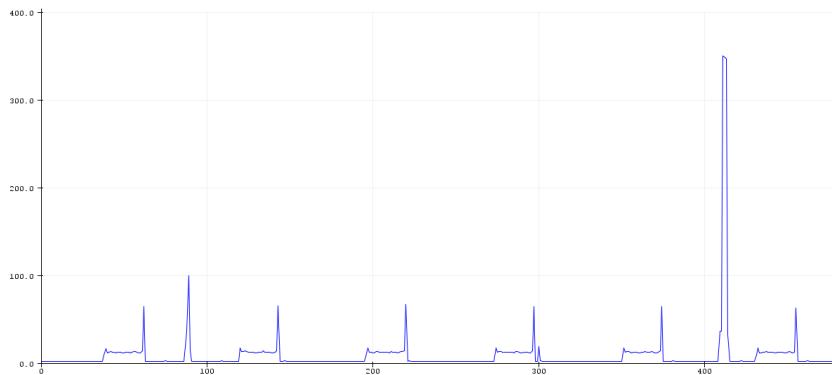


Slika 3.14: Notranjost zbirne enote. Razvidna je uporaba enotne barvne sheme za električne povezave in uporaba vročega lepila za zatesnitev lukenj in pritrjevanje električnih komponent. Napajalni kabel je zaradi potrebe po tesnjenju trajno pritrjen v ohišje.

zalednega dela pridobi konfiguracijske podatke, končno dodana sporočila iz vrste za pošiljanje agregira glede na to ali sporočilo vsebuje podatke o meritvi ali telemetrijo in vsake posebej paketno pošlje na zaledni del preko WiFi povezave.

Enoto smo sestavili iz več manjših komponent, ki smo jih z v naprej narezanimi žicami električno povezali. Celemu projektu smo določili enotno barvno shemo žic in njihovih namenov (slika 3.11). Pri napajanju smo se držali standardne črne in rdeče barve, pri signalih pa smo se načeloma držali principa, da je zelena barva namenjena urinemu signalu, modra pa podatkovnemu signalu. Vse električne komponente (slika 3.12) smo namestili v plastično ABS ohišje z zaščito proti vodi in drugimi vplivi. Vse zvrtane luknje smo zatesnili z vročim lepilom (slika 3.14), da smo preprečili vdor vode. Na zunanjost enote (slika 3.13), smo montirali antene, prikazovalnike in trimetrski USB kabel za napajanje. Ker mora enota preprečevati vdor dežja, je kabel trajno z vročim lepilom pritrjen v ohišje.

3.5 Poraba energije



Slika 3.15: Na grafu porabe se jasno vidijo zaznavanja zvoka vsako sekundo in pošiljanja podatkov. Manjši vrh se ujema s pridobivanjem podatkov o času in intervalu, večji vrh pa s pošiljanjem podatkov. Manjši vrhovi se ujemajo s porabo energije med zbiranjem in obdelavo podatkov.

Ena od glavnih zahtev tega projekta je bila možnost napajanja merilnih enot z baterijami. To je predpogoj za fleksibilnost in prosto postavljanje enot na lokacijo merjenja. Cilj je bil vsaj en teden delovanja z baterijskim napajanjem. Ta cilj smo presegli s teoretičnimi vsaj tremi tedni napajanja iz dveh baterij, če je interval med zaznavanji dolg eno sekundo.

Če želimo enoto napajati en teden z eno 18650 Li-ion polnilno baterijo, ki ima kapaciteto 3000 mAh, mora biti povprečen električni tok manjši od 17 mA. Naš izbran mikrokontroler po specifikacijah proizvajalca pri polni hitrosti obratovanja porabi povprečno 55 mA, med pošiljanjem pa nad 150 mA. Za naše potrebe je to veliko preveč. Zato smo pri implementaciji kode za merilno enoto upoštevali nekaj konceptov in dobrih praks za zmanjševanje porabe.

Prvi koncept, ki smo ga uporabili pri naši implementaciji je spanje [10]. Celotno zajemanje zvoka, procesiranje in pošiljanje traja manj kot eno tretrjino sekunde, tako da lahko ostali dve tretjini mikrokontroler spi. Ta mikrokontroler podpira dve vrsti spanja - globoko spanje in dremež. V načinu

globokega spanja enota porabi le nekaj mikroamperov, ampak se ob spanju delovni pomnilnik izbriše in se enota efektivno gledano ponovno zažene. Čeprav bi bila majhna poraba med spanjem zelo priročna, se mikrokontroler iz globokega spanja zbuja 250 ms pri polni hitrosti delovanja. V načinu dremeža porabi približno en miliamper, ampak se vsebina delovnega pomnilnika ohrani in se izvajanje programa po spanju takoj nadaljuje. Zaradi tega je dremež v tem primeru veliko bolj primeren način spanja. Tako enota približno 250 ms zbira in obdeluje podatke, preostali čas pa spi. To je najbolje razvidno na sliki 3.15. S tem smo porabo zmanjšali na približno 20 mA.

```

while (true) {

    if(is_interval_now()){
        sensing_and_data_preparation();
    }

    // sleep for a random amount of time to prevent signal congestion
    int random_sleep = (int)get_random_sleep_time();
    if (random_sleep > 0) {
        esp_sleep_enable_timer_wakeup(random_sleep);
        esp_light_sleep_start();
    }

    sending_and_telemetry();

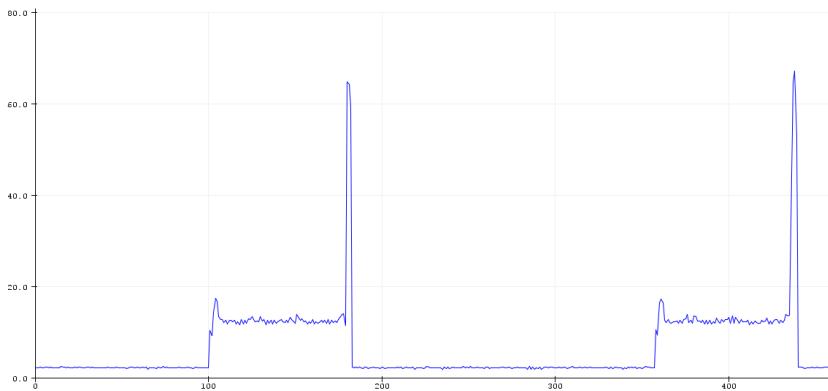
    // calculate time till next second and enter light sleep
    long left = 0;
    left = get_remaining_sleep_time();
    if (left > 0) {
        esp_sleep_enable_timer_wakeup(left);
        esp_light_sleep_start();
    }
}
```

{}

Koda, ki se izvede ob vsakem ciklu zaznavanja. Najprej ugotovi, če je glede na interval zaznavanja pravilen čas za zaznavanje. Če je, enota zajame in obdela podatke. Nato enota pred pošiljanjem naključno dolgo časa spi, da s tem zmanjša možnost trčenja paketov. Za tem pošlje podatke in uskladi informacijo o trenutnem času, na koncu izračuna preostali čas za spanje in ta čas spi.

Drugi koncept, ki smo ga pri implementaciji upoštevali, je dinamično nastavljanje frekvence delovanja mikrokontrolerja. Izbiramo lahko med 240 MHz, 160 MHz, 80 MHz, 40 MHz, 20 MHz in 10 MHz [5].ol Počasnejše delovanje procesorja pomeni daljše izvajanje iste kode. Tega načeloma nočemo, ker je najbolje, da je mikrokontroler čim več časa v načinu spanja. Ker se poraba s frekvenco delovanja ne veča linearно, ampak v korist višjim frekvencam, je počasnejše procesiranje tudi manj učinkovito. Torej lahko frekvenco procesorja zmanjšamo na delih, ki niso računsko zahtevni. Na srečo je zbiranje podatkov iz mikrofona, ki predstavlja najdaljši del aktivnosti, tudi najmanj računsko zahtevno, saj se medpomnilnik, ki je namenjen podatkom iz mikrofona, polni brez posegov procesorja. Zato smo frekvenco med zbiranjem podatkov nastavili na 20 MHz. Načeloma bi lahko frekvenco nastavili na 10 MHz, ampak je ura procesorja vezana na I2S enoto in ker je najmanjša frekvenca I2S signala za pridobivanje podatkov iz mikrofona 20 MHz, je to najmanjša ustrezna frekvenca. Za procesiranje podatkov se frekvenca dvigne na 240 MHz, ker to traja zelo malo časa in je zato najbolj učinkovito. Tako lahko iz grafa porabe skozi čas predpostavimo v katerem delu cikla delovanja je trenutno enota. To se najbolje vidi na sliki 3.16. S tem smo porabo zmanjšali na približno 15 mA.

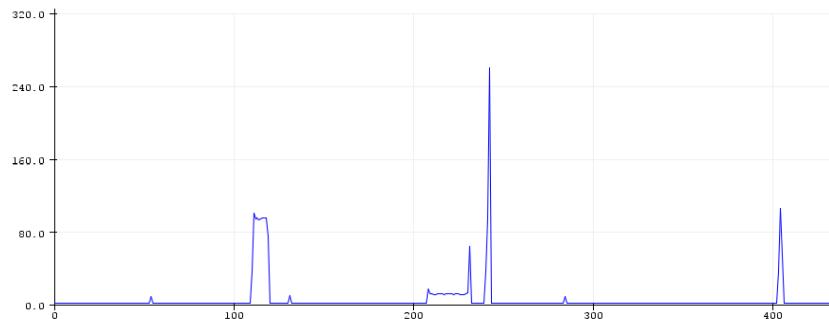
Tretji koncept, ki smo ga upoštevali, je paketno pošiljanje podatkov in optimizacija intervala za sinhronizacijo časa. Največja neučinkovitost pošiljanja podatkov ob vsakem zaznavanju je inicializacija dela mikrokontrolerja, ki je zadolžen za brezzično komunikacijo. Pred vsakim dremežem je treba ta del pravilno onemogočiti in pred ponovnim pošiljanjem nazaj omogočiti. To pri



Slika 3.16: Na grafu porabe so jasno razvidni različni deli običajnega zaznavnega cikla. Enota najprej približno 650 ms spi, kar se ujema z zelo nizko porabo. Sledi pridobivanje podatkov o zvoku, ki traja približno 250 ms in ima zaradi nizke frekvence procesorja majhno porabo. Na koncu sledi obdelava podatkov, ki se zgodi pri največji hitrosti procesorja, in se ujema z vrhom, ki traja le nekaj milisekund.

80 MHz traja 20 ms, pri 240 MHz pa 8 ms in se ne poveča, če mikrokontroler pošlje več podatkov. Tako se veliko bolj splača, da podatke meritev posilja vsakih nekaj sekund. To se najbolje vidi na sliki 3.15, kjer najvišji vrh sovpada s posiljanjem podatkov. Na podoben način smo optimizirali tudi porabo pri časovnem usklajevanju. Enote morajo zaradi natančnosti čim bolj istočasno zaznavati hrup. To časovno usklajenost smo dosegli s periodičnim usklajevanjem podatkov o trenutnem času. Najboljši kompromis med porabo energije in natančnostjo je usklajevanje enkrat na štiri sekunde. Usklajevanje časa se vidi na sliki 3.15 in sovpada z drugim najvišnjim vrhom. Te optimizacije so porabo zmanjšale pod 10 mA.

Uporabniku smo omogočili dodatno možnost optimizacije z uravnavanjem intervala zaznavanja. Z daljšim časom med zaznavanjami je enota večji del časa obratovanja v načinu dremeža, kar zmanjša porabo. Na sliki 3.17 se vidi, kako se poleg enega zaznavanja, enega posiljanja in dveh usklajevanj časa ne dogaja prav dosti. Enota se kljub temu zbudi vsako sekundo, da preveri, če je na vrsti za zaznavanje. To na grafu sovpada z najmanjšimi odstopanjimi od



Slika 3.17: Na grafu porabe se vidi princip delovanja, ko je interval zaznavanja daljši od ene sekunde. Enota se eno sekundo po začetku prejšnjega merjenja zbudi in preveri, ali je trenutno število sekund deljivo z intervalom merjenja. Ker ni, gre nazaj v način spanja. Nekaj milisekund kasneje, se enota spet zbudi in spet preveri, če je čas za pošiljanje ali usklajevanje časa.

porabe med spanjem. S preudarnim nastavljanjem intervala med zaznavanjem lahko tako uporabnik podaljša trajanje baterije, kar je vidno v tabeli 3.1, kjer je prikazana odvisnost porabe od intervala zaznavanja.

Dodatno smo bili pozorni tudi pri izboru razvojne ploščice, saj imajo nekatere neprimerne regulatorje napetosti, ki tudi, ko ne napajajo ničesar, porabijo nekaj toka. Najbolj pogost tak regulator napetosti je AMS1117, ki porabi kar 10 mA. Pri sestavljanju smo z razvojnih ploščic prav tako odstranili led diode.

3.6 Strežnik

Programska oprema, ki se izvaja na strežniku v laboratoriju za bioinformatiko Fakultete za računalništvo in informatiko v Ljubljani, je bila razvita s skladom MEAN. Njena primarna naloga je skrb za shranjevanje, dostop, spreminjanje in osnovno obdelavo podatkov. Zadolžena je tudi za omogočanje dostopa do uporabniškega vmesnika.

Oblika podatkov ustreza delovanju meritnih in zbirnih enot, ter organizaciji študij. Tako ima vsaka enota in študija svoj zapis. Izjema so podatki,

Interval zaznavanja (s)	Električni tok (mA)
1	8.35
2	5.65
3	4.65
4	4.14
5	3.80
6	3.60
7	3.50
8	3.35
9	3.25
10	3.15
1000	2.61

Tabela 3.1: Povprečne porabe ob različnih intervalih merjenja. Tok smo merili na sestavljeni enoti. Uporabili smo čip INA219 za merjenje napetosti. V porabo je vključena poraba mikrokontrolerja, ekrana in mikrofona.

kjer smo več podatkov združili v en zapis. Podatki se v podatkovni bazi hranijo v obliki JSON. V tej obliki so dostopni tudi preko implementiranega vmesnika. Za dostop in delo s podatki smo implementirali vmesnik REST API [16].

Merilna in zbirna enota sta podobni entiteti (slika 3.18, zato sta tu opisani skupaj. Obe enoti imata predpostavljeno polje za ObjectId, ki se uporablja pri identifikaciji znotraj strežniškega dela. Za identifikacijo obe enoti uporabljata MAC naslove in naključno dodeljena imena. Poleg tega v polju CurrentDeployment hranita identifikacijo trenutne študije, v polju LastTelemetry je zapisan datum zadnje prejete telemetrije, v polju LastData je zapis s prejetimi podatki, v polju BatteryVoltage je zadnja meritev napetosti na bateriji in v polju LastMeasurement je datum zadnjega zaznavanja.

Zaradi podobne oblike podatkov in podobnih načinov uporabe teh podatkov, imata enoti enak REST API vmesnik. Tu je opisan vmesnik merilne enote, ampak enako velja za zbirno enoto.

SensorSchema		GatewaySchema	
Name	String	Name	String
MAC	[Number]	MAC	[Number]
CurrentDeployment	ObjectId	CurrentDeployment	ObjectId
CurrentLocation	[Number]	WiFiCredentials	
LastTelemetry	Date	LastTelemetry	Date
LastData	DataSchema	LastData	DataSchema
BatteryVoltage	Number	BatteryVoltage	Number
LatestMeasurement	Date	LatestMeasurement	Date

Slika 3.18: Oblika zapisov za merilno in zbirno enoto. V skoraj vseh pogledih sta enaki. Razlika je le to, da ima merilna enota shranjeno trenutno lokacijo, zbirna pa ime in geslo WiFi omrežja, do katerega lahko dostopa na kraju zaznavanja.

GET /api/sensor

Ta klic vrne osnovne podatke o vseh merilnih enotah.

GET /api/sensor/:sensor_id

Ta klic vrne vse podatke o specificirani merilni enoti. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

POST /api/sensor

S tem klicem registriramo novo enoto. V telesu zahtevka pričakuje MAC naslov enote, vrne pa polni zapis iz podatkovne baze. Uporablja se ga tudi pri vsakem nastavljanju enot, saj je MAC naslov edini identifikacijski podatek, ki se ohrani v enoti po nalaganju nove programske kode.

POST /api/sensor/telemetry/:sensor_id

S tem klicem se v zapis merilne enote zapisejo novi telemetrični podatki. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

PUT /api/sensor/:sensor_id

S tem klicem se posodobi zapis specificirane meritne enote. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

DeploymentSchema	
Name	String
Sescription	String
Sensors	[ObjectId]
Gateways	[ObjectId]
Status	String
MeasurementNum	Number
MeasurementInterval	Number

Slika 3.19: Oblika zapisov v podatkovni bazi za hranjenje podatkov o študijah.

Študije, oziroma deployments (slika 3.19) so zaključene enote zaznavanja podatkov. Predstavljajo eno namestitev meritnih in zbirnih enot na kraj zaznavanja in vse pridobljene podatke. Takšna organizacija sistemu omogoča fleksibilnost. V zapisu študije se predpostavi polje ObjectId za sklicevanje na zapis v okviru strežnika. Poleg tega zapisi hranijo podatke o imenu, opisu, dodeljenih meritnih in zbirnih enotah, trenutnem stanju, številu vseh meritev in trenutnem intervalu zaznavanja.

Ker je študija kompleksnejša entiteta, smo poleg klicev za neposredno delo s podatki implementirali še dodatne klice, ki posodobijo tudi entitete, ki niso del študije.

GET /api/deployment

Ta klic vrne osnovne podatke o vseh študijah.

GET /api/deployment/:deployment_id

Ta klic vrne cel zapis o specificirani študiji. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

```
GET /api/deployment/deploy/:deployment_id  
GET /api/deployment/finish/:deployment_id
```

Prvi klic zažene študijo, drugi klic pa študijo zaključi. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

```
GET /api/deployment/interval/:deployment_id/:interval  
GET /api/deployment/interval/:deployment_id/
```

Prvi klic spremeni trenutno dolžino intervala zaznavanja. Drugi klic pričopi podatke o trenutnem intervalu zaznavanja. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

```
POST /api/deployment/
```

Klic ustvari nov zapis za študijo. Telo klica mora vsebovati vsaj ime študije.

```
PUT /api/deployment/:deployment_id
```

Klic posodobi podatke trenutne študije. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

```
DELETE /api/deployment/:deployment_id
```

Klic izbriše študijo in vse shranjene podatke meritov. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

Pri hranjenju podatkov o meritvah, smo uporabili metodo združevanja podatkov v vedra s specificirano velikostjo. Tako smo združili po 1000 zaporednih meritov iz iste enote pri isti študiji v en zapis. S tem smo prihranili

DataSchema	
Sensor	ObjectId
Deployment	ObjectId
Location	[Number]
Size	Number
First	Date
Last	Date
Measurement	[ObjectId]

Slika 3.20: Oblika zapisov v podatkovni bazi za hranjenje podatkov o podatkih.

MeasurementSchema	
FrequencyRange	Number
FftValues	[Number]
Decibels	Number
MeasuredAt	Date

Slika 3.21: Oblika zapisov v podatkovni bazi za hranjenje podatkov o posameznem zaznavanju.

tako na velikosti podatkovne baze, kot tudi na času iskanja podatkov. Vse to je uporabniku in napravam nevidno. Tako to odraža le oblika podatkov.

Tako je v zapisu s podatki (slika 3.20) shranjena identifikacija meritve enote in študije, lokacija zajetih podatkov, velikost, čas prve meritve, čas zadnje meritve in podatki zajeti z meritvami. V zapisu posamezne meritve (slika 3.21) se hranijo podatki o frekvenčnem razponu podatkov spektralne analize, rezultati spektralne analize, glasnost v decibelih in časovna oznaka meritve zaokrožena na sekundo natančno.

POST /api/sensor/data

Klic v podatkovno bazo zapiše podatke meritov. V telesu zahtevka mora biti seznam meritov, vsak mora vsebovati MAC naslov meritne enote in vsa polja zapisa o meritvah (slika 3.21).

GET /api/data/deployment/:deployment_id

Klic vrne podatke vseh meritov, ki so se zgodile v okviru specificirane študije.

GET /api/data/deployment/average/:deployment_id

Klic vrne največ 1000 podatkovnih točk, kjer vsaka predstavlja povprečje vseh meritov v intervali enakemu eni tisočini dolžine študije.

GET /api/data/deployment/:deployment_id/:last_n

GET /api/data/deployment/:deployment_id/seconds/:seconds

Prvi klic pridobi zadnjih n meritov vsake meritne enote. Drugi klic pridobi vse meritve iz zadnjih n sekund.

The screenshot shows a web application interface for the 'Urban Noise Sensing Platform'. At the top, there is a navigation bar with links for 'Deployments', 'Sensors', and 'Gateways'. Below the navigation bar, the main content area is titled 'Deployments'. It features a section for 'Not yet deployed' which contains a single item: 'Nov deployment za prikaz' (New deployment for display). This item includes a link 'Opis tega deploymonta' (Description of this deployment) and a note 'Number of measurements: 0'. There is also a small '+' icon next to the title. Below this, there is a section for 'Deployed' which contains two items: 'ok' and 'dnevna soba'. Both items have a note 'Number of measurements:' followed by a large number (177690 for 'ok' and 80599 for 'dnevna soba').

Slika 3.22: Pregled trenutnih in preteklih študij. Vidi se minimalističen oblikovalski stil, ki smo ga uporabili na vseh delih vmesnika.

3.7 Uporabniški vmesnik

Ena glavnih zahtev tega projekta je bila fleksibilnost, saj bo raziskovanje potekalo v sklopu manjših študij, ki bodo opravljene na različnih manjših območjih. Na vsakem od teh območij bomo morali postaviti meritne enote na druge položaje. Ker je treba vsako študijo posebej nastaviti, je intuitiven grafični uporabniški vmesnik ključnega pomena za učinkovito delo in hitro vpeljevanje novih sodelavcev. Prav tako mora vmesnik omogočati pregled nad trenutnim stanjem meritnih in zbirnih enot.

Uporabniški vmesnik (slika 3.22) smo razvili na podlagi dveh konceptov - MVC in SPA. Zaledni del tako deluje samo kot vir podatkov, ki deluje po principu REST API. Prikaz pri uporabniku ustvari aplikacija Angular. Za oblikovanje smo uporabili knjižnico Bootstrap.

Primarni namen uporabniškega vmesnika je upravljanje z meritnimi in zbirnimi enotami. Zato smo najprej implementirali dva pogleda (sliki 3.23), ki omogočata pregled nad trenutnim stanjem enot. Običajno nas najbolj



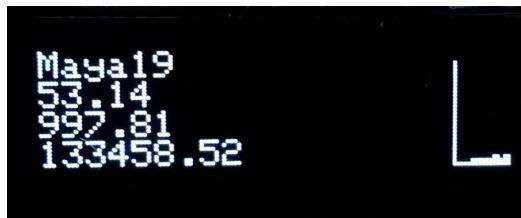
(a)



(b)

Slika 3.23: Pregled meritnih in zbirnih enot v spletnem uporabniškem vmesniku. Zapis meritnih enot (slika a), prikazujejo podatke o trenutni študiji, napetosti na bateriji in trenutno lokacijo, prav tako je pri vsaki enoti izpisano ime za lažjo identifikacijo. Podobno je pri zbirnih enotah (slika b), izpisano ime in trenutna študija.

zanima, kateri študiji je določena enota trenutno dodeljena, zato je pri dodeljenih enotah vedno izpisana ta informacija. Pri merilnih enotah nas poleg tega zanima še točna lokacija in napolnjenost baterije.



Slika 3.24: Primer dodeljenega imena na OLED prikazovalniku ene od merilnih enot.

Običajno za identifikacijo enot uporabljamo MAC naslove in posebne naključne identifikacijske kode, ki jih baza podatkov naključno določi zapisom. Takšen način ljudem ni blizu in je precej neuporaben, ko je enot več, saj si ljudje težko zapomnimo več kot 7 naključnih zaporednih znakov. Zato smo implementirali sistem človeških imen z dodanimi naključnimi številkami. Takšno ime dobi vsaka enota ko se prvič poveže na zaledje in se ”registrira”. Ime posamezne enote je konkatenacija naključnega imena iz korpusa stotih najbolj pogostih angleških imen in naključne številke med 0 in 100. Na sliki 3.24 se vidi primer imena na merilni enoti, na sliki 3.23 pa imena na spletnem uporabniškem vmesniku.

Drugi del upravljanja z enotami je ustvarjanje študij in dodeljevanje enot tem študijam. Implementirali smo sistem, kjer uporabnik lahko prosto postavlja proste enote na zemljevid (slika 3.25), izbira med zbirnimi enotami in nastavlja WiFi ime in geslo, ki bo dostopno na kraju študije.

Za hitro diagnosticiranje težav in sproten pregled nad rezultati študije, smo uporabnikom omogočili pregled nad številom zaznavanj po merilnih enotah, povprečno glasnostjo med zaznavanjem (slika 3.26) in lokacije enot med zaznavanjem.

Sensors

To place a sensor:

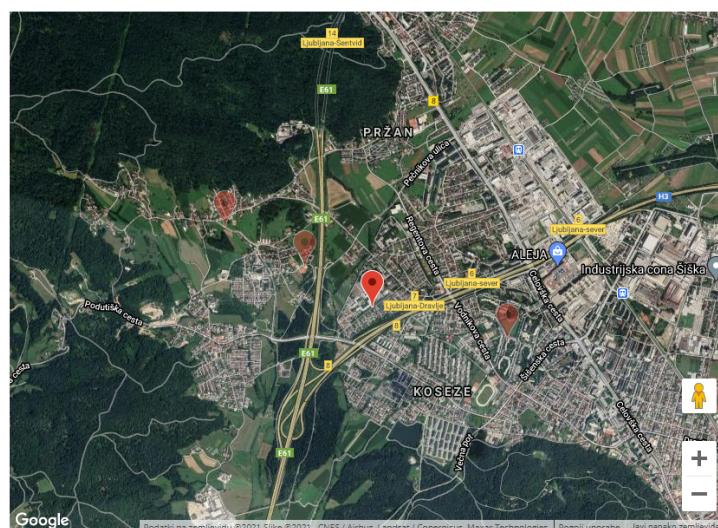
1. Click on the map to place a marker
2. Click on the marker to remove it
3. Click on sensor name to show its position

Free sensors

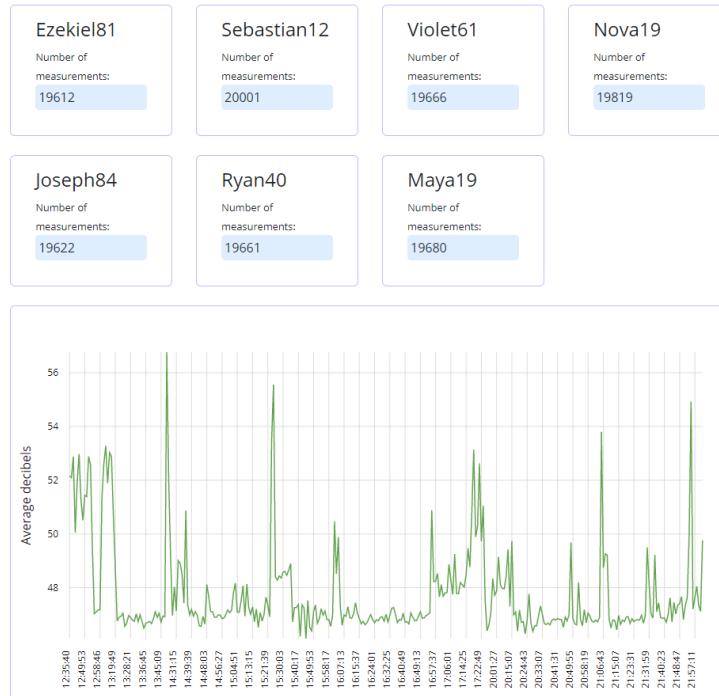
Ezekiel81 Violet61 Maya19 Jeremiah89 Theodore84 Emery20

Placed sensors

Sebastian12 Joseph84 Ryan40 Nova19



Slika 3.25: Postavljanje prostih meritnih enot na zemljevid. Uporabnik s klikom na zemljevid razvršča meritne enote, ki trenutno niso dodeljene nobeni študiji. Lokacije trenutno postavljenih enot lahko uporabnik preveri s klikom na ime postavljene enote, odstrani pa jih lahko s klikom na postavljen zaznamek.



Slika 3.26: Pregled merilnih enot in povprečne glasnosti hrupa med študijo. Med zaznavanjem in po koncu zaznavanja lahko uporabnik pregleda te najbolj osnovne podatke o poteku študije. Graf je omejen na maksimalno 1000 točk. Ko je meritev več, se izračuna povprečje več meritev, tako da rezultat ustreza tej omejitvi.

3.8 Podatkovna analitika

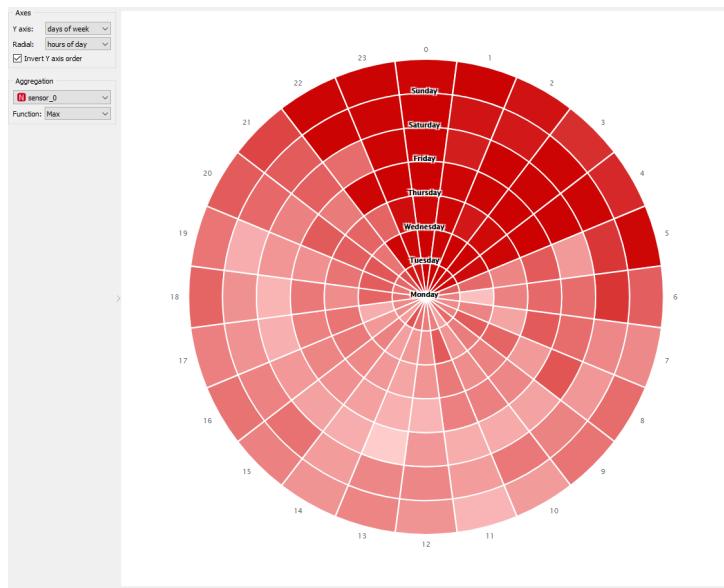
Orange je program za podatkovno rudarjenje, ki ga razvija laboratorij za bioinformatiko na Fakulteti za računalništvo in informatiko v Ljubljani [2]. Posebnost tega programa je, da omogoča vizualno programiranje z uporabo gradnikov, ki jih uporabnik povezuje med sabo (slika 3.27).



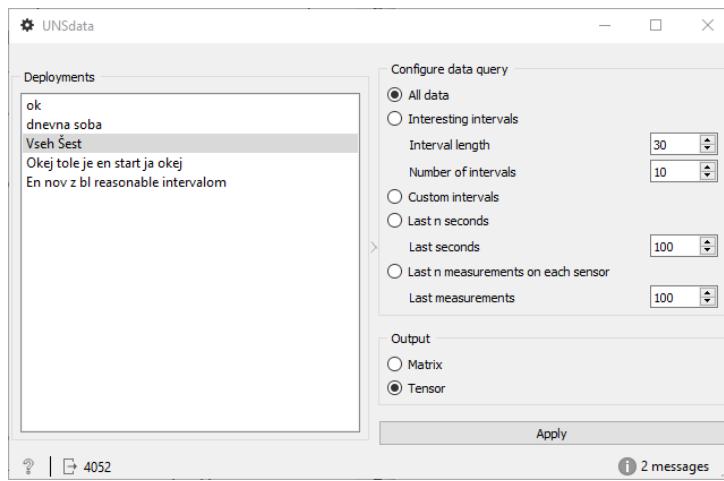
Slika 3.27: Primer delovnega toka v programu Orange na primeru kosilnice. V ozadju se vidi definiran delovni tok, v ospredju pa so odprti prikazi analiziranih podatkov. Prikazana je uporaba časovne rezine in prikaza na zemljevidu, ter graf glasnosti po senzorjih v odvisnosti od časa.

Ena naših zahtev je bila integracija razvitega sistema s programom Orange. Zato smo razvili svoj gradnik, ki omogoča prenos podatkov z zaledja in njihovo uporabo v delovnih tokovih. Ko so podatki naloženi v program, lahko uporabimo vgrajene gradnike za dodatno analizo in prikaz teh podatkov. Ti gradniki nam omogočajo prikaz glasnosti na zemljevidu, pomikanje skozi čas s časovno rezino, ugotavljanje periodičnosti podatkov (slika 3.28), iskanje najbolj zanimivih odsekov podatkov in predvsem pregledno in intuitivno delo z večjo količino podatkov.

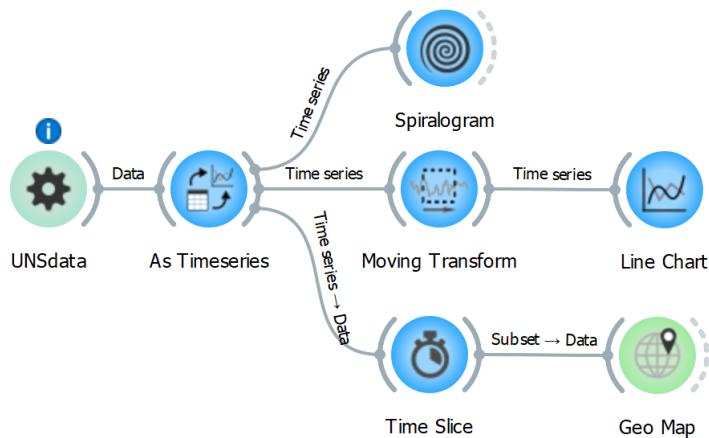
Naš gradnik (slika 3.29) se imenuje UNSdata in je namenjen pridobivanju podatkov z zalednega strežnika. Nanj pošilja zahtevke oblike REST in vrnjene podatke pretvarja v kompatibilno obliko za uporabo z drugimi



Slika 3.28: Primer prikaza podatkov s spiralnim grafom. Tu smo podatke uredili po dnevih v tednu in urah dneva. Prikazani so podatki zaznavanja hrupa v bivalnih prostorih, tako da se jasno vidi kdaj je šla družina spati, kdaj so se zjutraj zbudili in celo daljše spanje v soboto zjutraj.



Slika 3.29: Zaslonski posnetek implementiranega gradnika. Na levi strani se izpišejo imena vseh študij s podatki, do katerih lahko uporabnik dostopa, desno zgoraj uporabnik definira podatkovno poizvedbo, desno spodaj pa izbere obliko izstopnih podatkov in potrdi svojo izbiro.



Slika 3.30: Primer delovnega toka v programu Orange. Podatke, pridobljene s pomočjo UNSdata gradnika, se najprej pretvori v časovno zaporedne podatke, potem se lahko periodičnost podatkov prikaže na spiralnem prikazu. Podatke se lahko zgladi in osnovno obdela s funkcijo "Moving Transform" in se jih prikaže na grafu, ali pa se izbere rezino podatkov, ki so potem prikazani na zemljevidu.

gradniki v programu. Implementiran princip uporabe predpostavlja, da uporabnik najprej s seznama izbere študijo, nato nastavi obliko poizvedbe in na koncu izbere obliko podatkov. Pridobljene podatke lahko potem z drugimi gradniki obdela in prikaže (slika 3.30).

Poglavlje 4

Primera uporabe

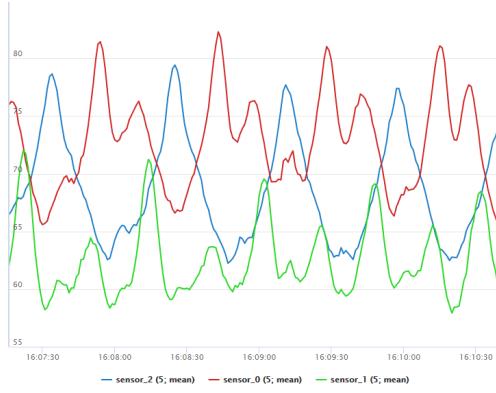
Z dvema primeroma uporabe smo primarno želeli prikazati dva načina delovanja. Primer dnevne sobe je iz stališča zaznavanja in analiziranja zvoka bolj tradicionalen, saj se osredotoča na trende in spremembe skozi daljša časovna obdobja. V nasprotju s tem pa primer kosilnice prikazuje inovativne zmožnosti našega sistema. Zanima nas namreč kako se izvor zvoka premika in spreminja glede na premikanje izvora zvoka.

4.1 Košenje trave

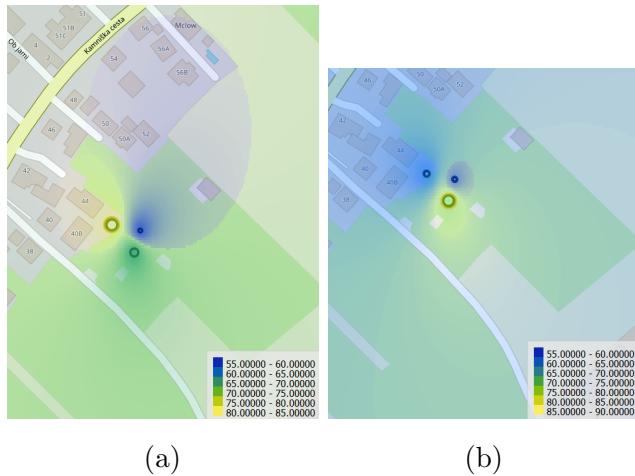
Pri načrtovanju primera uporabe s košenjem trave smo v ospredje postavili prikaz unikatnih zmožnosti našega sistema. Želeli smo prikazati, kako lahko zvoku sledimo z razporeditvijo več meritnih enot na manjše območje. Skupaj z možnostjo točnega nastavljanja lokacije posamezne enote in sinhroniziranega zaznavanja, nam je uspelo prikazati gibanje kosilnice po travniku.

Za zaznavanje smo uporabili tri meritne enote, ki smo jih postavili na rob travnika v obliki trikotnika (sliki 4.2). Zaznavanje je trajalo približno pol ure med košenjem zelenice. Interval med merjenji smo nastavili na eno sekundo.

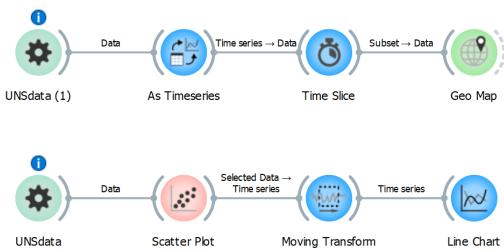
Po končanem zaznavanju smo podatke s pomočjo našega gradnika uvozili v program Orange (slika 4.3 in 3.27), jih obdelali in prikazali na zemljevidu (slika 4.2) in na grafu (slika 4.1). Na zemljevidu smo s pomikanjem skozi čas



Slika 4.1: Na grafu vseh meritev po času in senzorju se jasno vidi, kako se je kosilnica približevala in oddaljevala vsakemu od senzorjev. Ker smo senzorje postavili na rob travnika, kosilnica pa se je v spirali gibala proti sredini, je mogoče opaziti, da so vrhovi po času vedno manjši, ker je razdalja med kosilnico in senzorjem z zmanjševanjem premera kroga vedno večja.



Slika 4.2: Prikaz izmerjene glasnosti v različnih časovnih obdobjih. S prikazom na zemljevidu in izbiro dovolj kratke časovne rezine, lahko prikažemo približno pozicijo izvora zvoka in kako se je ta izvor premikal skozi čas. To je primerno tako za akutne spremembe v poziciji izvora zvoka, kot je prikazano tu, kot tudi za prikaz počasnejših sprememb.



Slika 4.3: Delovni tok v programu Orange. Za prikaz na zemljevidu smo najprej z gradnikom UNSData pridobili podatke v obliki tenzorjev, jih spremenili v časovno vrsto, izbrali časovno rezino za pregled in to prikazali na zemljevidu. Za prikaz podatkov na grafu, smo pridobili podatke z gradnikom UNSData v obliki matrice, v razsevnem diagramu izbrali zanimivo časovno rezino, podatke zgradili in jih prikazali na grafu.

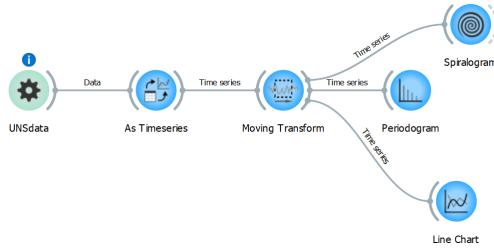
z gradnikom time slice opazovali kako se je kosilnica premikala med tremi merilnimi enotami v krogu, kar se sklada z dejanskim premikanjem. Na grafu smo opazili podobno, le da smo poleg tega opazili tudi sprotno zmanjševanje glasnosti, saj se je kosilnica v spirali premikala proti sredini travnika, kar je vedno dlje od merilnih enot, ki so bile montirane na robu travnika.

S tem primerom smo pokazali natančnost in uporabnost naših meritev ko je v igri več merilnih enot. Razvit sistem je poseben prav iz tega stališča, da so enote časovno usklajene in dovolj cenovno dostopne, da jih lahko montiramo več na manjše območje. Poleg tega je sistem razvit s hitrostjo in fleksibilnostjo v mislih. Postavitev enot in konfiguracija študije nam je vzela le nekaj minut.

4.2 Glasnost v dnevni sobi

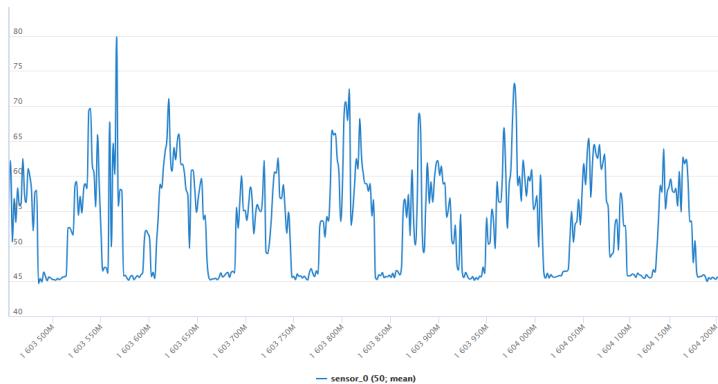
S primerom glasnosti v dnevni sobi smo želeli prikazati sposobnost razvitega sistema za merjenje glasnosti na tradicionalen način, kjer so pomembni trendi skozi čas. Pri merjenju glasnosti je prisotnih manj merilnih enot in merjenje

traja več časa.



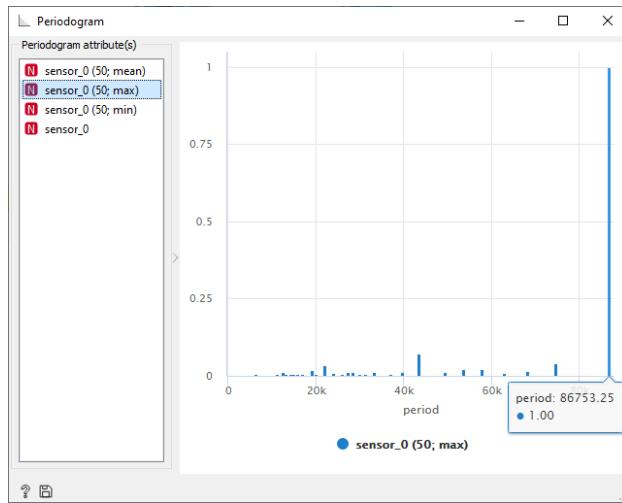
Slika 4.4: Delovni tok v programu Orange. Podatke o glasnosti v dnevni sobi smo z našim gradnikom v celoti prenesli v obliki matrice, te podatke smo potem spremenili v časovno vrsto in izračunali povprečje v intervalih pet minut. Rezultate smo potem prikazali na spiralogramu, periodogramu in grafu.

Merilno enoto smo postavili v dnevno sobo velike družine. Interval zaznavanja smo nastavili na deset sekund in pustili, da zbira podatke 10 dni. Podatke smo potem z uporabo razvitega gradnika za program Orange prenesli in obdelali (slika 4.4).



Slika 4.5: Graf glasnosti v dnevni sobi v odvisnosti od časa. Na njem se jasno vidi glasnost ko je bila dnevna soba prazna in ko je bila družina budna.

Pri tem primeru nas je zanimala periodičnost podatkov, torej kakšen je trend spanja in zbjanja družine, kdaj so otroci v šoli in starši v službi, kdaj se samodejno vklopi pomivalni stroj in podobno. Za to smo uporabili



Slika 4.6: Uporabniški vmesnik gradnika za računanje periodičnosti podatkov. Tu se vidi periodičnost budnosti in spanja družine, ki je ocenjena na vrednost, ki je blizu dolžine enega dneva (86400 s).

spiralogram (slika 3.28) in periodogram (slika 4.6).

Na izbranih prikazih se dobro vidi aktivnost družine. Ker je družina časovno precej neuskajena enota, smo opazili le večje dnevne aktivnosti - budnost in spanje. Pri merjenju glasnosti bolj časovno strukturiranih organizacij - šola, fakulteta, tovarna,... - pa bi pričakovali bolj zanimive podatke, saj se tam osebe ravnajo po določenem urniku.

Poglavlje 5

Zaključek

Cilj tega projekta je bil ustvariti sistem za zaznavanje in analizo hrupa. Ta sistem mora biti enostaven za uporabo, fleksibilen in cenovno učinkovit. Pod fleksibilnost sodi zmožnost prostega postavljanja enot brez zunanjega napajanja in trajne namestitve.

Razvili smo poceni meritne in zbirne enote iz prosto dostopnih komponent, s primernim dosegom signala, baterijskim napajanjem in zmožnostjo dodajanja enot. Za shranjevanje podatkov in upravljanje smo implementirali strežnik. Za upravljanje in nastavljanje smo implementirali intuitiven uporabniški vmesnik. Za analiziranje podatkov v programu Orange smo razvili svoj gradnik. Vsa programska koda je prosto dostopna v repozitoriju git na portalu Github (<https://github.com/andraz213/UrbanNoiseSensing>).

Sestavljanje meritnih enot kljub optimizacijam traja preveč časa. To bi se dalo rešiti z razvojem svojih tiskanih vezij, prav tako bi s tem razširili možnosti za uporabo bolj učinkovitih komponent, kar bi pripomoglo k dolžini delovanja baterije. Za boljšo analizo podatkov bo potrebno razviti specializirane gradnike za program Orange. Predvsem pa projekt potrebuje podporo s strani urbanistov in arhitektov. Da bo sistem zares zaživel potrebuje uporabnike in veliko testiranja.

Appendices

Dodatek A

Dodatek: Uporabniška navodila

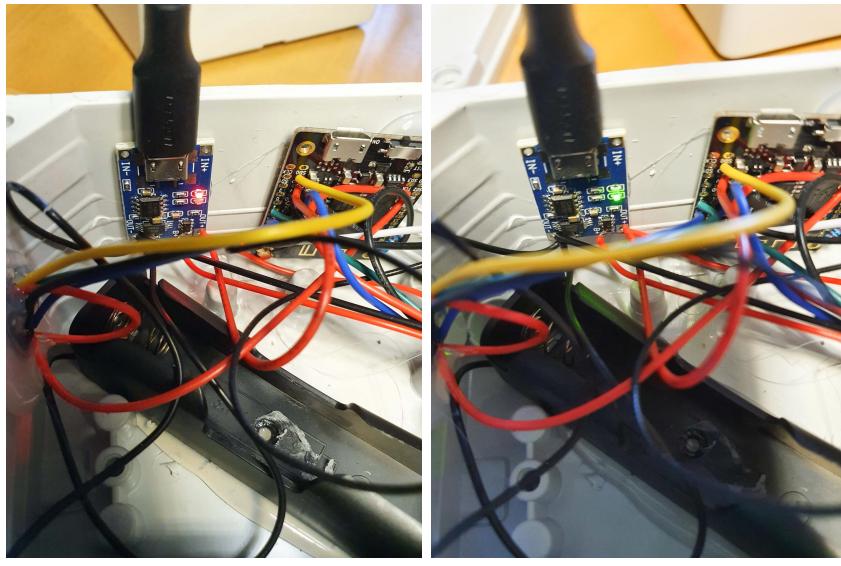
A.1 Polnjenje meritnih enot

Meritne enote napajajo li-ion polnilne baterije. Napoljenost posamezne meritne enote se lahko preveri na strani "Sensors". Baterije so prazne pri napetosti približno 3.0V.

1. Odstranite pokrov enote, ki je pritrjen s 4 vijaki.
2. Z USB kablom povežite polnilno vezje z USB napajalnikom. Polnilno vezje je manjša ploščica z USB priključkom, montirana zraven mikrokontrolerja.
3. Polnite dokler ne sveti zelena luč. Če je naprava med polnjenjem prižgana v načinu nastavljanja, lahko preverite napoljenost na strani "Sensors". Enota je napolnjena, ko napetost preseže 4.05 V.

A.2 Prižiganje enot v različnih načinih

Prva iteracija meritnih enot ima dve stikali, ki sta uporabljeni za prižiganje in izbiro načina delovanja. Naslednje bodo imele le eno stikalo in bo prižiganje in izbira delovanja trivialna operacija.



(a) Med polnjenjem.

(b) Polnjenje je končano.

Slika A.1: Prikaz načina polnjenja meritne enote.

1. Stikalo za napajanje premaknete v pozicijo OFF in s tem ugasnete napravo.
2. Stikalo za način delovanja prestavite v način ki se ujema z želenim načinom delovanja. "N" predstavlja način za nastavljanje, Ž pa način za zaznavanje.
3. Stikalo za napajanje premaknete v pozicijo ON in s tem prižgete napravo.

A.3 Registracija meritne enote

Preden enote uporabite za raziskave, jih je potrebno registrirati v sistem.

1. Poskrbite, da je napravam na voljo WiFi omrežje s povezavo na internet z SSID "UNSwifi" in geslom "uns12wifi34".
2. Prižgite enote v način za nastavljanje.



Slika A.2: Označba stikal za prižiganje enote in izbiro načina delovanja, kjer ON in OFF predstavlja prižgano in ugasnjeno enoto, N in Z pa način za nastavljanje in način za zaznavanje.

3. Počakajte, da se na ekranu izpiše ime enote.



Slika A.3: Ekran meritne enote, ko je enota v načinu nastavljanja in se je naprava uspešno registrirala v sistem. V prvi vrstici je izpisano ime meritne enote, v drugi so trenutni decibeli, v tretji pa najmočnejša frekvenca zvoka.

A.4 Ustvarjanje raziskave

Projekt je zasnovan na principu raziskav, ki so urejene v ”deployments”. Ustvarja in upravlja se jih preko spletnega vmesnika, ki je dostopen na naslovu: ”<http://urbannoisesensing.biolab.si/>”.

1. Registrirajte vse meritne in zbirne enote, ki bodo uporabljeni v tej raziskavi.
2. V spletnem vmesniku v zavihku ”Deployments” pritisnite gumb “+”, vpišite ime raziskave in pritisnite gumb ”Create Deployment”.
3. Nadaljujte z urejanjem raziskave.

A.5 Urejanje raziskave

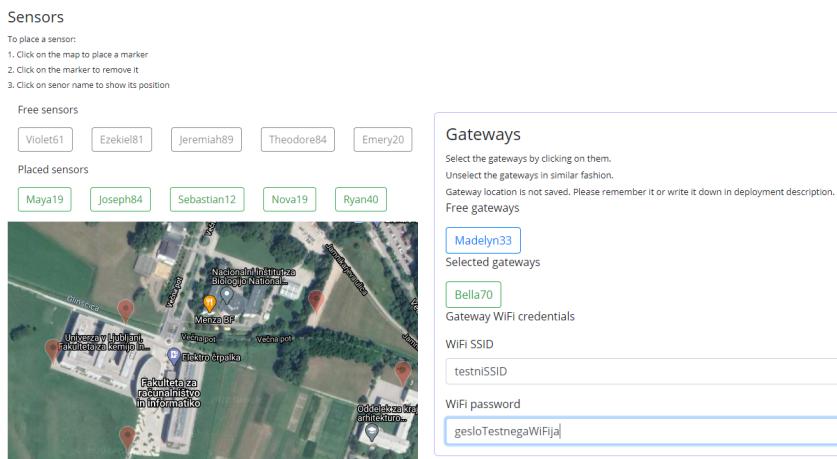
Pred začetkom zaznavanja je treba nastaviti in zagnati raziskavo. Spletni vmesnik je dostopen na naslovu: ”<http://urbannoisesensing.biolab.si/>”.

1. Ustvarite raziskavo, ali pa na strani "Deployments" v razdelku "Not yet deployed" izberite raziskavo, ki jo želite urediti.
2. Raziskavi uredite ime in opis tako, da spremenite besedilo v vnosnih poljih in pritisnete gumb "Save changes".
3. Merilne enote lahko raziskavi dodajate tako, da v razdelku "Sensors" s klikom na zemljevid na ustrezena mesta postavljate zaznamke.
 - S klikom na zaznamek na zemljevidu odstranite enoto iz raziskave.
 - S klikom na ime postavljeni merilni enoti, se zaznamek na zemljevidu obarva.
4. Zbirno enoto lahko dodate v razdelku "Gateways" s klikom na ime izbrane zbirne enote.
5. Zbirni enoti lahko dodate možnost povezovanja na dodatno WiFi dostopno točko tako, da vpišete podatke o tem omrežju v ustreza polja v razdelku "Gateways".
6. Raziskavo zaženete s klikom na gumb "DEPLOY".
7. Raziskavo zbrišete s klikom na gumb "DELETE DEPLOYMENT".

A.6 Prenašanje podatkov na enote

Ko je raziskava zagnana, je treba podatke o nastavivah prenesti na enote.

1. Poskrbite, da je napravam na voljo WiFi omrežje s povezavo na internet z SSID "UNSwifi" in geslom "uns12wifi34".
2. Prižgite merilne enote v načinu za nastavljanje in počakajte, da se enota poveže na WiFi in pridobi podatke o nastavivah.
3. Povežite zbirno enoto na napajanje in počakajte, da se poveže na WiFi omrežje.



(a) Primer postavljenih meritnih enot na zemljevidu (b) Primer izbire zbirne enote in nastavljanja WiFi omrežja.

Slika A.4: Zaslonski posnetki uporabniškega vmesnika med nastavljanjem raziskave.

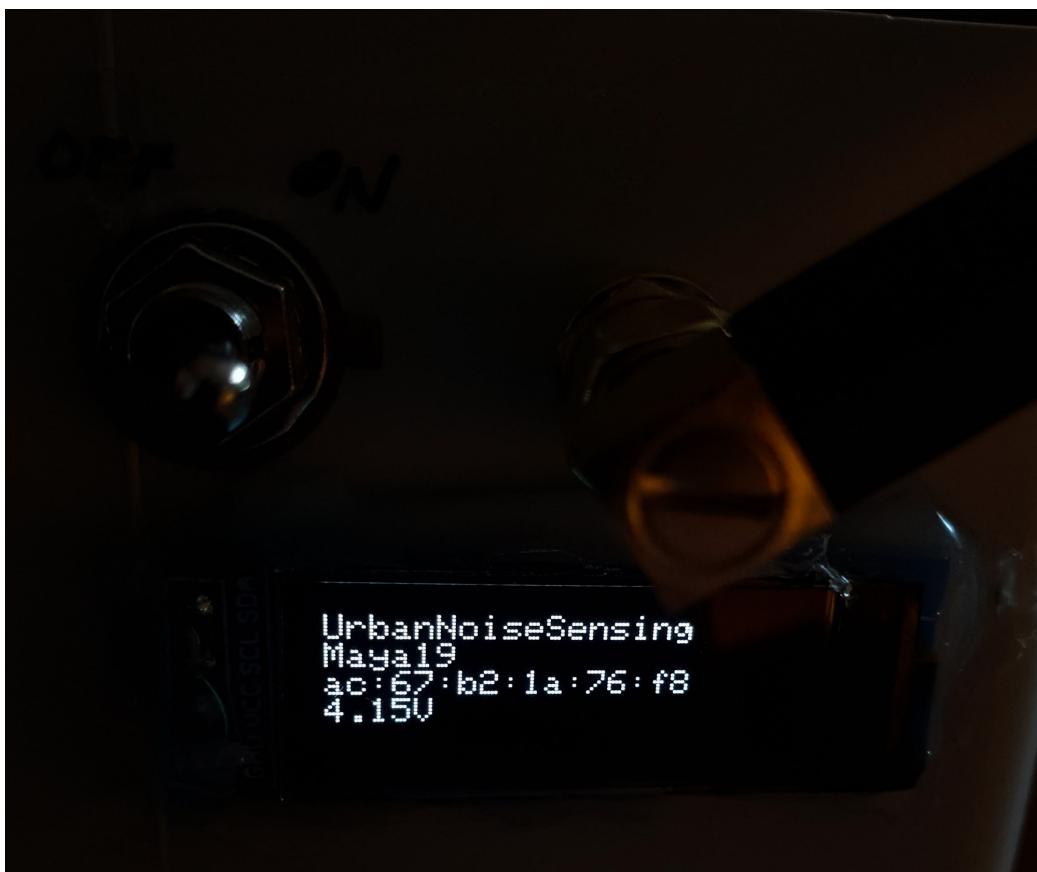
4. Ugasnite enote.

A.7 Postavljanje enot na lokacijo zaznavanja

Ko je raziskava zagnana in so podatki preneseni na enote, lahko postavite enote na lokacijo zaznavanja.

1. Poskrbite, da so pokrovi vseh enot ustrezno pritrjeni in da so baterije meritnih enot ustrezno polne.
2. Enote vsako posebej postavite na specificirane lokacije in jih prižgite v način za zaznavanje. Točne lokacije meritnih enot lahko preverite tako, da v spletnem vmesniku na strani "Deployments" v razdelku "Deployed" izberete trenutno raziskavo in v zavihku "Sensors" s klikom na posamezno ime enote na zemljevidu preverite, kje točno naj bi se posamezna enota nahajala.
3. Postavite in priklopite zbirno enoto v dosegu prej specificiranega WiFi

omrežja.



Slika A.5: Ekran meritne enote, ko se pravilno nastavljena enota prižiga v načinu zaznavanja. V prvi vrstici je napisano ime projekta, v drugi je napisano ime enote, v tretji je napisan MAC naslov zbirne enote, v zadnji pa napetost na bateriji.

A.8 Uravnavanje intervala zaznavanja

Privzeti interval zaznavanja je 1 sekunda. S povečevanjem intervala se podaljša doba delovanja baterij in zmanjša količina podatkov. Spletni vmesnik je dostopen na naslovu: "<http://urbannoisesensing.biolab.si/>".

1. V spletnem vmesniku na strani "Deployments" v razdelku "Deployed" izberite trenutno raziskavo.
2. V razdelku "Sensing interval" v vpisno polje vpišite ustrezno dolžino intervala v sekundah.
3. Pritisnite gumb "Save changes".

A.9 Konec zaznavanja

Po končani raziskavi, je treba zaključiti "deployment". Spletni vmesnik je dostopen na naslovu: <http://urbannoisesensing.biolab.si/>.

1. V spletnem vmesniku na strani "Deployments" v razdelku "Deployed" izberite trenutno raziskavo.
2. Pritisnite gumb "FINISH SENSING". Če ni bilo opravljenih nič uspešnih meritov, je treba raziskavo izbrisati z gumbom "DELETE DEPLOYMENT".

Literatura

- [1] Alfons Dehé, Martin Wurzer, Marc Füldner, and Ulrich Krumbein. The infineon silicon mems microphone. In *AMA conferences*, volume 2013, pages 14–16. Citeseer, 2013.
- [2] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. In *European conference on principles of data mining and knowledge discovery*, pages 537–539. Springer, 2004.
- [3] Esp-idf programming guide. Dosegljivo: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html>. [Dostopano: 10. 1. 2021].
- [4] Esp-now documentation. Dosegljivo: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html. [Dostopano: 11. 7. 2020].
- [5] Esp power management. Dosegljivo: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power_management.html. [Dostopano: 20. 8. 2020].
- [6] Esp32-mini-32-v1.3. Dosegljivo: <https://github.com/LilyGO/ESP32-MINI-32-V1.3>. [Dostopano: 1. 7. 2020].
- [7] Esp32 series datasheet. Dosegljivo: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Dostopano: 12. 1. 2021].

- [8] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Mariamuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.
- [9] Inmp441 datasheet. Dosegljivo: <https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf>. [Dostopano: 15. 7. 2020].
- [10] Insight into esp32 sleep modes their power consumption. Dosegljivo: <https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>. [Dostopano: 3. 7. 2020].
- [11] Ivan Kostoski. Sound level meter with arduino ide, esp32 and i2s mems microphone. Dosegljivo: <https://hackaday.io/project/166867-esp32-i2s-slm>. [Dostopano: 1. 7. 2020].
- [12] Charlie Mydlarz, Samuel Nacach, Agnieszka Roginska, Tae Hong Park, Eric Rosenthal, and Michelle Temple. The implementation of mems microphones for urban sound sensing. In *Audio Engineering Society Convention 137*. Audio Engineering Society, 2014.
- [13] Andrey Ovcharov. How to measure battery level with esp32 microcontroller. Dosegljivo: <https://en.ovcharov.me/2020/02/29/how-to-measure-battery-level-with-esp32-microcontroller/>. [Dostopano: 1. 10. 2020].
- [14] Courtney Peckens, Cédric Porter, and Taylor Rink. Wireless sensor networks for long-term monitoring of urban noise. *Sensors*, 18(9):3161, 2018.
- [15] Pjon documentation. Dosegljivo: <https://www.pjon.org/documentation.php>. [Dostopano: 11. 1. 2021].
- [16] Andrew John Poulter, Steven J Johnston, and Simon J Cox. Using the mean stack to implement a restful service for an internet of things

application. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 280–285. IEEE, 2015.