

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Novak

**Porazdeljeni sistem za analizo hrupa  
urbanih okolij**

DIPLOMSKO DELO

UNIVERZITETNI ŠTDIJSKI 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<sup>A</sup>T<sub>E</sub>X.*

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.



*Mentorju prof. dr. Blažu Zupanu se zahvaljujem za vložen trud in odlično akademsko usmerjanje.*

*Hvala družini za vzgojo in podporo v času študija. Hvala Nani Merjasec za nesebično moralno podporo, potrpežljivost in vir motivacije Hvala vsem ostalim, ki so pri pomogli k nastajanju tega diplomskega dela.*



# Kazalo

## Povzetek

## Abstract

<b>1</b>	<b>Uvod</b>	<b>1</b>
<b>2</b>	<b>Tehnologije</b>	<b>5</b>
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 . . . . .	12
<b>3</b>	<b>Rešitev</b>	<b>13</b>
3.1	Uporabniške zahteve . . . . .	13
3.2	Arhitektura . . . . .	14
<b>4</b>	<b>Razvoj merilnih in zbirnih enot</b>	<b>17</b>
4.1	Merilna enota . . . . .	17
4.2	Zbirna enota . . . . .	26
4.3	Poraba energije . . . . .	32
<b>5</b>	<b>Merilni strežnik</b>	<b>39</b>
5.1	Strežnik . . . . .	39
5.2	Uporabniški vmesnik . . . . .	45

5.3	Podatkovna analitika . . . . .	50
<b>6</b>	<b>Primera uporabe</b>	<b>53</b>
6.1	Košenje trave . . . . .	53
6.2	Glasnost v dnevni sobi . . . . .	55
<b>7</b>	<b>Zaključek</b>	<b>59</b>
<b>A</b>	<b>Dodatek: Uporabniška navodila</b>	<b>61</b>
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
<b>Literatura</b>		<b>70</b>

# Povzetek

**Naslov:** Porazdeljeni sistem za analizo hrupa urbanih okolij

**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. Trenutne rešitve za pridobivanje podatkov o hrupu so drage in zahtevajo veliko načrtovanja, zato se jih velikokrat preskoči. V sklopu diplomske naloge smo razvili cenovno dostopen sistem, ki omogoča hitro in enostavno opazovanje hrupa na poljubno velikih območjih s poljubno količino senzorjev. Razvili smo cenovno dostopne meritve, odprtokodno platformo za upravljanje zaznavanja in shranjevanje podatkov, ter spletni in programske vmesnike za učinkovito podatkovno analizo.

**Ključne besede:** hrup, internet stvari, zaznavanje, mreža senzorjev, mikrokontrolerji, porazdeljeni sistem.



# Abstract

**Title:** Distributed system for urban noise analysis

**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. 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 analysis has been implemented. We've developed affordable measuring units, a user interface, an open-source server for data storage, and a widget for efficient data analysis.

**Keywords:** noise, internet of things, sensing, sensor network, microcontrollers, distributed system.



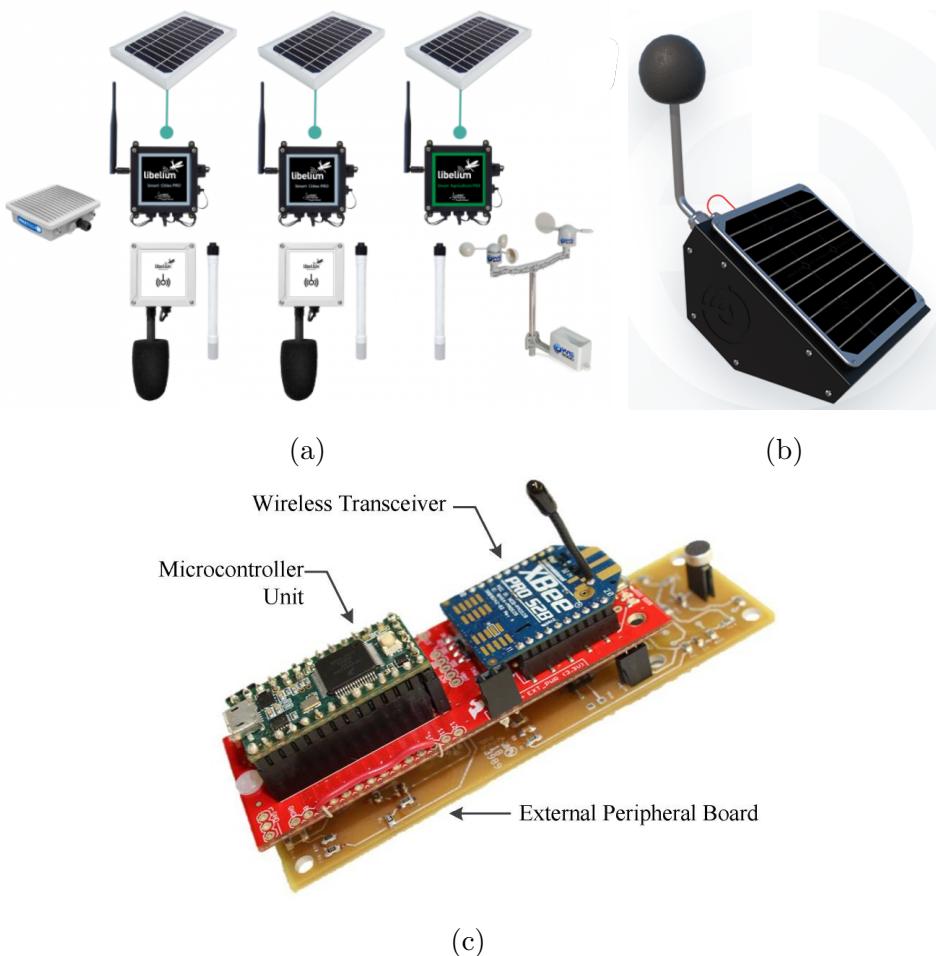
# Poglavlje 1

## Uvod

Internet stvari je tehnologija s katero lahko spremljamo, upravljamo in nadzorujemo svet okoli nas [5]. Omogoča vse od povezanih pametnih luči, upravljanje s hišnim ogrevanjem, do spremljanja 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 [4].

Moderno načrtovanje javnih površin vedno bolj upošteva vpliv hrupa na vedenje in zdravje ljudi [9]. 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 prilagodljiv 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 spremljanja 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. Rešitev, ki smo jo razvili v pričujoči nalogi, je



Slika 1.1: Konkurenčne rešitve. Rešitev podjetja Libelium za komunikacijo uporablja protokol LoRaWAN in omogoča napajanje iz sončnih celic (a). Podobna rešitev podjetja SensorTeam (b). Repitev Ameriškega nacionalnega centra za biotehniko informatiko je primerjavi z drugima je veliko bolj cenovno dostopna [11] (c).

prav v teh pogledih boljša. Najbolj podobno rešitev so razvili na Ameriškem nacionalnem centru za biotehnološko informatiko, kjer so z uporabo poceni mikrokontrolerjev in komunikacijskih modulov implementirali meritne enote [11]. 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 diplomske naloge smo implementirali nizkocenovni porazdeljen sistem za zaznavanje in analizo hrupa na poljubno velikem območju. Ker je moral sistem biti fleksibilen, smo se pri snovanju soočali s problemom porabe energije, dometa signala, velikosti enot in organizacijo projekta. Odločili

smo se za uporabo mikrokontrolerjev, ki preko povezav podobnim WiFi-ju pošiljajo zajete informacije na strežnik. Ker ta povezava ne omogoča neposrednega komuniciranja s strežnikom, smo razvili zbirne enote, ki podatke zberejo in posredujejo na strežnik. Želeli smo, da je projekt popolnoma odprtokoden in da ustrezava vsem našim zahtevam glede organizacije projekta, smo razvili programsko opremo za shranjevanje in pridobivanje podatkov. Dodaten cilj je bil omogočiti dostop do podatkov v programu Orange [2]. Tako smo razvili programski vmesnik, ki to omogoča.

# Poglavlje 2

## Tehnologije

Izbor tehnologij temelji na zahtevi po odzivnosti sestavnih delov sistema, točnosti meritev in energijski porabi. 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 računalniških sistemov, 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 hitre obdelave podatkov, veliko perifernih naprav in možnosti nizke porabe energije, je bil odličen kandidat za naš projekt[8].

Zaradi nizke cene, veliko funkcionalnosti in odlične podporne infrastrukture s strani proizvajalca, je mikrokontroler ESP32 priljubljena izbira proj-

Tabela 2.1: Značilnosti mikrokontrolerja ESP32 <sup>1</sup>


---

Ime	ESP32
Proizvajalec	Espressif Systems
Procesor	Tensilica Xtensa LX6 dvojederni + koprocesor
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 elektriKE pri 240 MHz	55 mA
Poraba elektriKE pri 20 MHz	20 mA
Poraba elektriKE v spanju	0.8 mA
Največja poraba	350 mA

---

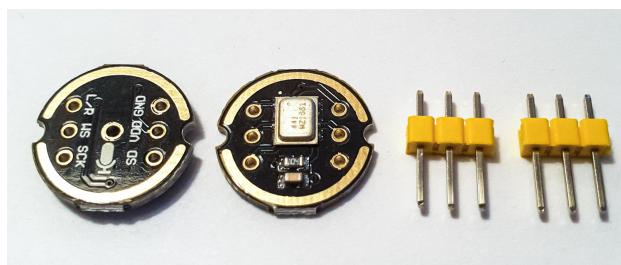


Slika 2.1: Razvojna ploščica TTGO T7 z mikrokontrolerjem ESP32. Uporabili smo jo v merilnih in zbirnih enotah. Izbrali smo jo zaradi majhne porabe energije v načinu spanja in priključka za zunanjo anteno.

zvajalcev razvojnih ploščic. Te se uporablja za razvoj kode, razvoj prototipov 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 zunanjo anteno, nizke porabe energije in velikim naborom električnih povezav.

## 2.2 Merjenje hrupa

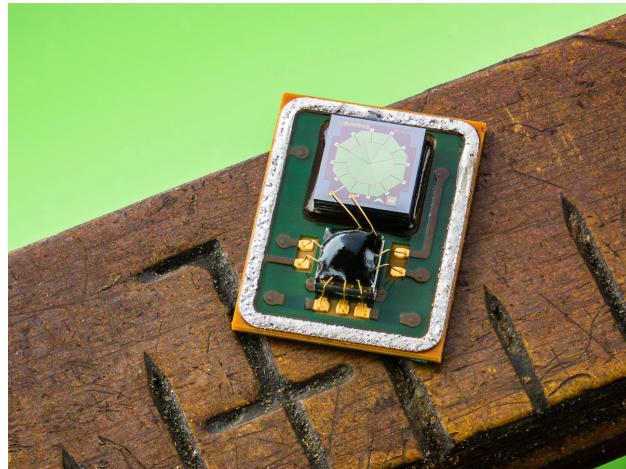
Glavni del pričajočega projekta je zaznavanje hrupa. Ta naloga je razdeljena na pridobivanje podatkov in obdelavo. Glavni zahtevi sta konsistentnost podatkov med napravami in točnost meritev. Kar se tiče zaznavanja podatkov smo tema dvema zahtevama lahko brez težav ustregli s pravilno programsko kodo povzeto po odprtokodnem projektu projektu<sup>2</sup>, 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 pravega 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

<sup>2</sup><https://hackaday.io/project/166867-esp32-i2s-slm>



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.

Tabela 2.2: Značilnosti mikrofona INMP441 <sup>3</sup>

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 stanju pripravljenosti	0.8 mA
Poraba električne energije pri spanju	0.0045 mA

katerimi se prenašajo šibki analogni signali. Za točnost takšnih meritev so običajno nujno potrebne kvalitetne komponente in natančna izdelava. Zato smo izbrali mikrofon, ki za to 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 mikroelektromehanskih sistemov (tehnologija MEMS) so znani po zelo dobri natančnosti, majhnimi razlikami med komponentami, odlični ceni, majhni porabi energije in majhni velikosti [10]. Za naš projekt smo izbrali mikrofon INMP441 (slika 2.2 in tabela 2.2), saj je na voljo na svoji razvojni ploščici z dostopom do električnih signalov, je dobavljen in se enostavno poveže z izbranim mikrokontrolerjem. Glede na specifikacijo je pričakovano največje odstopanje meritev od točne vrednosti 3dB. Ker smo našli primere<sup>4</sup> uporabe enakega algoritma in mikrofona, ki nakazujejo odstopanje v rangu 1dB, smo se odločili, da je ta mikrofon primeren. Kljub temu pa bo v prihodnosti treba opraviti umerjanje našega sistema.

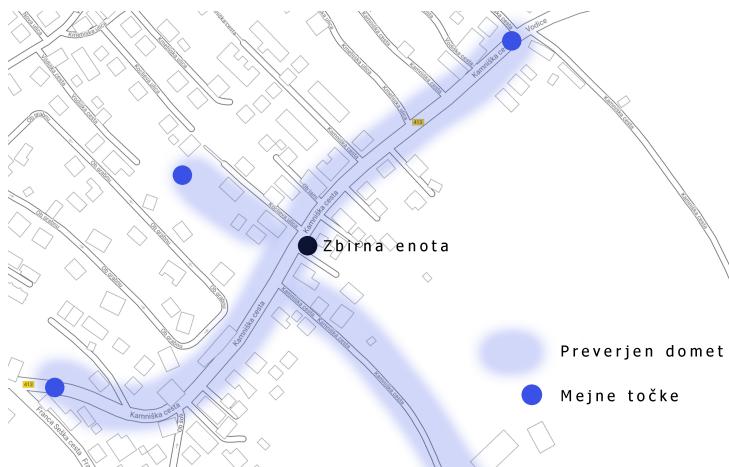
## 2.3 Povezljivost

Tehnologija za povezovanje mora ustrezati strogim pogojem glede porabe energije, hitrosti prenosa, načina naslavljanja, dosega signala in cene enote. Preizkusili in raziskali smo različne možnosti (tabela 2.3) ter 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[13]. Ta bi ustrezal našim zahtevam po hitrosti prenosa podatkov, ne bi pa bil primeren s stališča porabe energije, saj povezovanje na omrežje in vzpostavitev

---

<sup>4</sup><https://hackaday.io/project/162059-street-sense/log/170825-measuring-audible-noise-in-real-time/>



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.

Tabela 2.3: Primerjava različnih tehnologij za povezovanje meritnih in zbirnih enot.

	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 €

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 [14]. Ta ustrezata 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 [3]. Neprimeren je zaradi omejitve šestih povezav naenkrat, nezanesljivega delovanja in premajhnega dosega povezave.

Zadnjo možnost smo odkrili, ko smo pregledovali dokumentacijo mikro-

kontrolerja ESP32<sup>5</sup>. Proizvajalec teh čipov je namreč omogočil dokaj prosto pošiljanje paketov podatkov med ESP napravami. Protokol ESP-NOW ustreza vsem našim zahtevam. Hitrost podatkov do 250 kBps, pošiljanje podatkov brez vzpostavitev povezave, kar bistveno zmanjša porabo energije, in več kot 1.5 km dosega signala s poceni ceneno anteno v idealnih pogojih in doseg do 200 m v urbanem okolju [6].

## 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 mora biti odprtoko-dna, da bi jo lahko namestili na strežnik v laboratoriju. Zaradi potreb po podatkovni analitiki, mora imeti ustrezni vmesnik. Zaradi strukture projekta in zaznavanja, mora podpirati fleksibilno prestavljanje enot iz lokacije na lokacijo in razdelitev meritev na študije.

Tabela 2.4: Primerjava različnih programskih okolij za internet stvari.

	odprtoko-dno	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

Glede na tabelo 2.4 se raziskana že razvita okolja 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 odprtoko-dnih okoljih, ga lahko namestimo na poljuben strežnik in podpira izbrani način nastavljanja in zaznavanja.

---

<sup>5</sup>[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp\\_now.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html)

## 2.5 Strežniški del

Zaledni strežnik in uporabniški vmesnik smo zasnovali z uporabo sklada MEAN, ki vključuje tehnologije 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 obdelavi 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 kode JavaScript izven internetnega brskalnika. Skupaj z robustno platformo za upravljanje s paketi in razširitvami, 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 zapisu 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

Projekt smo pričeli z zbiranjem uporabniških zahtev in postavitev ciljev, ki smo jih želeli z razvitim sistemom doseči. Za tem smo se lotili implementacije po komponentah. Najprej smo razvili zaledni del, potem razvili merilno in zbirno enoto, za tem uporabniški vmesnik, nazadnje pa gradnik za podatkovno analitiko.

### 3.1 Uporabniške zahteve

Cilj projekta je bil implementacija sistema za podrobno razumevanje hrupa v prostorsko poljubno velikih urbanih okoljih. 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 uporabniško določenih časovnih intervalih, zajemale podatke o hrupu in jih posredovale v podatkovno bazo, kjer bodo potem dostopne za analizo v programu za podatkovno analitiko, kot je na primer Orange [2]. 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 so cenovno neustrezne. Zato je

bila ena naših zahtev, da zaledni sistem temelji na odprtakodnih okoljih in je nameščen na poljubnem, projektu dostopnemu strežniku. Prav tako morajo biti merilne enote veliko bolj cenovno dostopne kot podobne enote, ki so trenutno na voljo.

Merilne enote morajo biti dovolj majhne in lahke, da z njihovo namestitvijo 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.

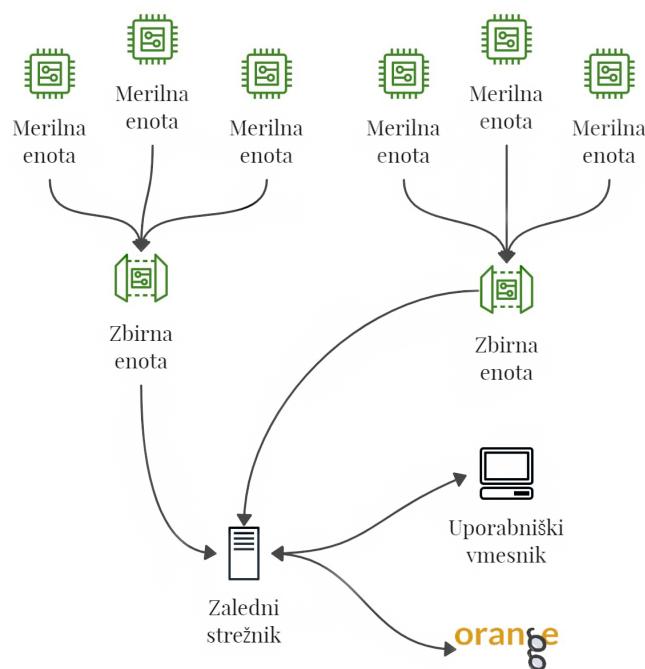
Ker želimo zbrane podatke obdelati in prikazati, smo se odločili, da bomo omogočili dostop do podatkov v programu Orange [2]. Tako smo se odločili implementirati svoj gradnik za ta program, ki bo podpiral poizvedbe na zalednjem delu.

## 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 sistema je zaledni strežnik, ki koordinira pretok 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 merilnih in zbirnih enot temelji na izbiri tehnologije za prenos podatkov o meritvah. Ker merilne enote nimajo neposrednega dostopa do interneta, smo pot podatkov speljali skozi zbirne enote, ki lahko istočasno komunicirajo z zalednim strežnikom in z merilnimi enotami. Zbirne enote delujejo kot posredniki, ki sami po sebi ne ustvarjajo novih podatkov, ampak jih samo prenašajo iz merilnih enot na zaledni strežnik [7].

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



Slika 3.1: Struktura razvitega sistema. Merilne enote na obroblju sistema 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 dostopamo s programskim vmesnikom.

pregleduje osnovne podatke o stanju enot, poteku zaznavanja in povprečno glasnost meritev.

Gradnik v programu Orange [2] je namenjen dostopu do podatkov. S strežnikom komunicira z zahtevki HTTP in prejete podatke spreminja v obliko, ki je primerna za uporabo z drugimi analitičnimi gradniki, ki omogočajo tudi grafične prikaze zajetih meritev.

# Poglavlje 4

## Razvoj merilnih in zbirnih enot

Zbiranje podatkov o hrupu je glavni del razvitega sistema. Za ta namen smo razvili merilne in zbirne enote, ki podatke o hrupu zajamejo, opravijo obdelavo teh podatkov in rezultate pošljejo na strežnik, kjer so na voljo za pregled in obdelavo. Pri razvoju smo veliko truda namenili fleksibilnosti sistema, cenovni učinkovitosti, enostavnosti uporabe in natančnosti zajetih podatkov.

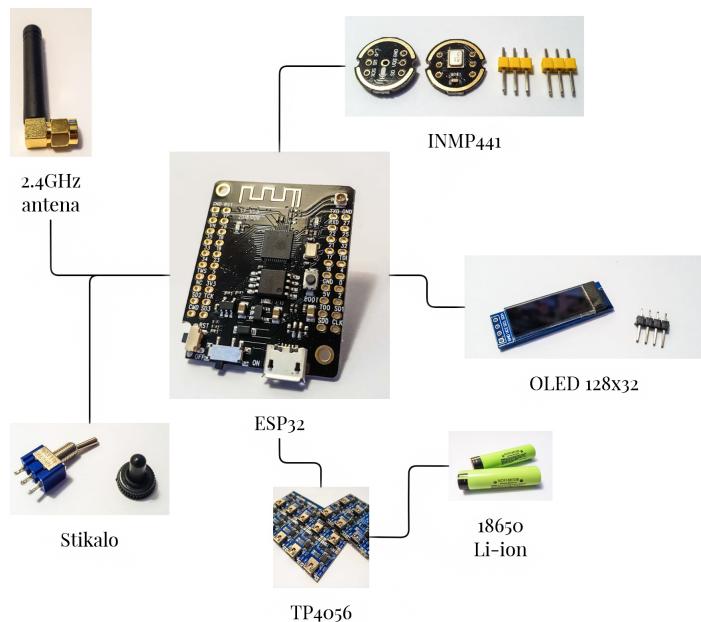
### 4.1 Merilna enota

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 namestimo 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 sestavljena.

Glavni del merilnih enot je mikrokontroler ESP32 na razvojni ploščici T7 podjetja TTGO. Za ta projekt je ta primerna zaradi majhne porabe v načinu spanja in priključka za zunanjo anteno<sup>1</sup>. Nanjo se povezujejo vsi

---

<sup>1</sup><https://github.com/LilyGO/ESP32-MINI-32-V1.3>



Slika 4.1: Merilna enota z nakazanimi povezavami med sestavnimi deli. Osrče 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.

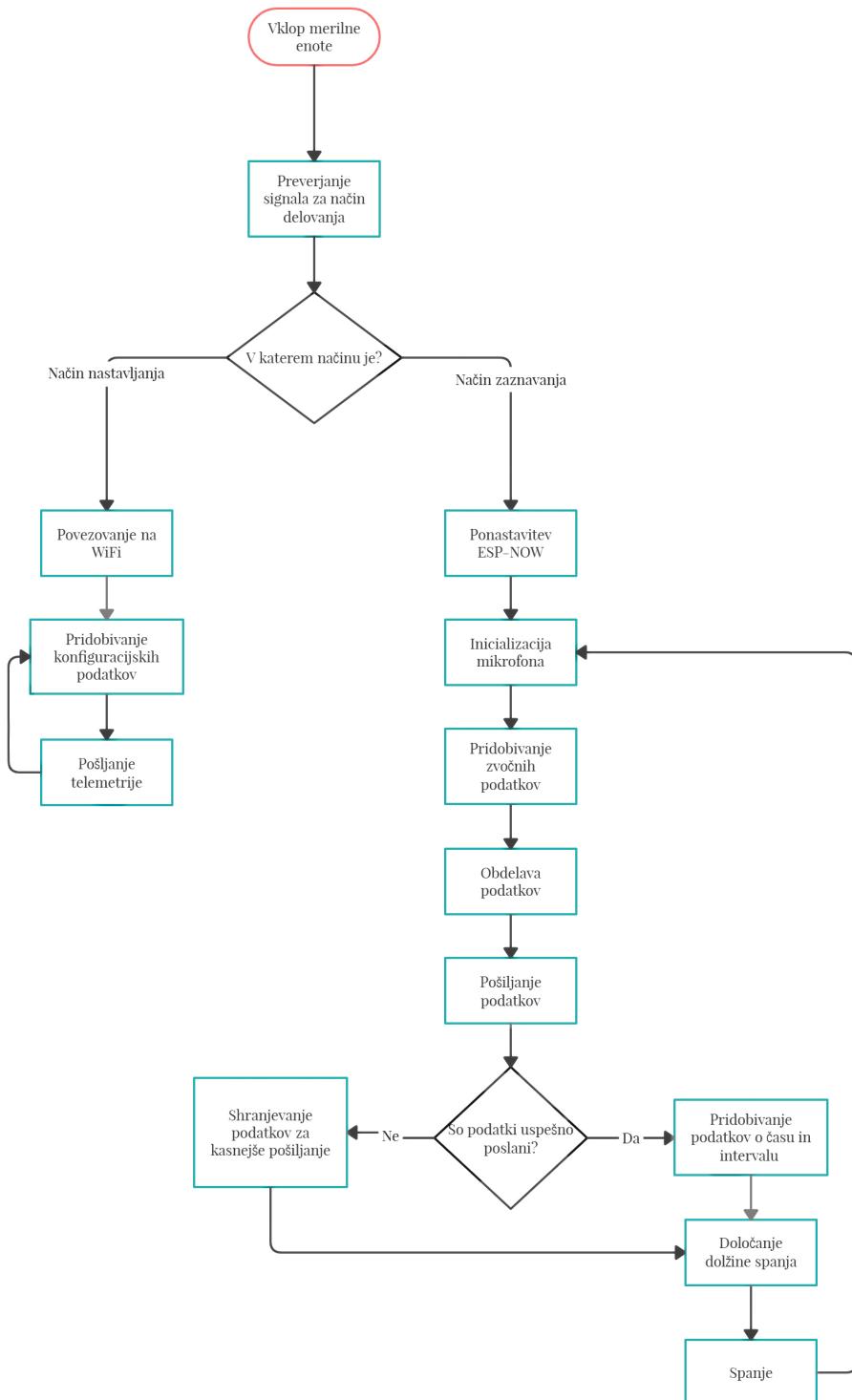
ostali deli meritne enote. Zaznavanje zvoka smo omogočili z MEMS mikrofonom INMP441, ki je s protokolom I2S povezan z mikrokontrolerjem. Ker je ta povezava povsem digitalna, smo na ta račun izločili veliko problemov s kvaliteto žic in spojev. Za pomoč pri nastavljanju enote smo se odločili, da bo vsaka enota opremljena z ekranom OLED SSD1306 z ločljivostjo  $32 \times 128$  slikovnih točk. Z mikrokontrolerjem ekran komunicira s protokolom I2C.

Za napajanje skrbita dve 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 previsok tok.

Meritna enota podpira dva načina delovanja: način za nastavljanje parametrov enote in način za zaznavanje. Med njima lahko uporabnik preklaplja s stikalom, ki je montirano na zunanjih strani enote.

Ko enota začne z delovanjem v načinu za nastavljanje (slika 4.2, levo), 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 meritno enoto, dodeli identifikacijsko številko za notranjo identifikacijo in enoti naključno dodeli berljivo 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 4.2, desno) je namenjen zbiranju in pošiljanju podatkov na zbirno enoto. Ko enota začne z delovanjem v načinu zaznavanja, najprej nastavi brezžični vmesnik za komunikacijo s protokolom ESP-NOW. Temu sledi vključitev mikrofona in zbiranje ter obdelava podatkov. Obdelani podatki se skupaj s časovno oznako zapišejo v sporočilo, ki se doda



Slika 4.2: 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.

v podatkovno 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 podatkovne vrste in ga poskusi poslati. Če je pošiljanje uspešno, je sporočilo zbrisano, če pošiljanje ni uspešno, se sporočilo ohrani v podatkovni vrsti, nakar enota določi nov čas naslednjega pošiljanja. Na koncu enota določi čas spanja do naslednjega zaznavanja in zaspri. Poleg tega enota vsake tri sekunde pošilja podatke o napoljenosti baterije in zahteva podatke o točnem času in o trenutnem intervalu zaznavanja.

Programsko kodo<sup>2</sup>, ki se izvede ob zajemanju in obdelavi podatkov kaže spodnji izpis:

```
\small
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_F
    decibels = 0.0;
    decibels = calculate_decibels((int*)&samples_pub, SAMPLES_SIZE);

    setCpuFrequencyMhz(20);
```

---

<sup>2</sup><https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html>

```
// 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);
}
```

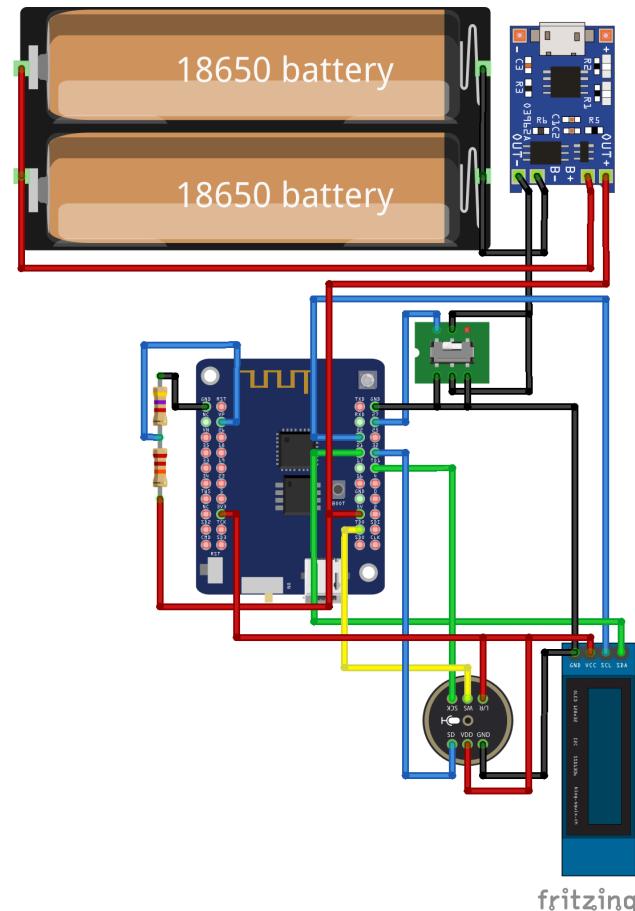
Ob začetku mikrokontroler nastavi uro na najmanjšo frekvenco delovanja, saj med zajemanjem zvočnega signala večino časa ni računsko obremenjen in čaka, da se medpomnilnik napolni. Sledi obdelava podatkov, zato se nastavi na najhitrejši način delovanja, kjer na podatkih izvede spektralno analizo in izračuna jakost zvoka. Na koncu nastavi delovno frekvenco na najnižjo in podatke prepiše v vrsto sporočil.

Za lažji pregled nad stanjem enote smo implementirali merjenje napetosti baterije <sup>3</sup>. Pri implementaciji smo izkoristili sposobnost nastavljanja referenčne napetosti na analogno digitalnem pretvorniku, ki je del mikrokontrolerja in zaporedno vezanih uporov (slika 4.3). Upora sta povezana tako, da sta na eni strani vezana skupaj, 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 v analogno digitalnem pretvorniku vedno primerja z napetostjo 1,1 V.

Komponente meritne enote (slika 4.4) smo povezali z žicami. Pri tem smo upoštevali enotno barvno paleto (slika 4.3) 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 primeru meritne enote za prenos podatkov z mikrofona.

---

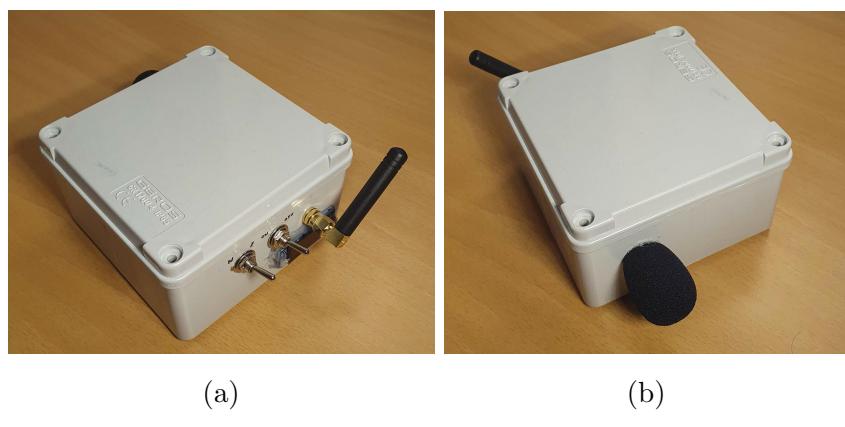
<sup>3</sup><https://en.ovcharov.me/2020/02/29/how-to-measure-battery-level-with-esp32-microcontroller/>



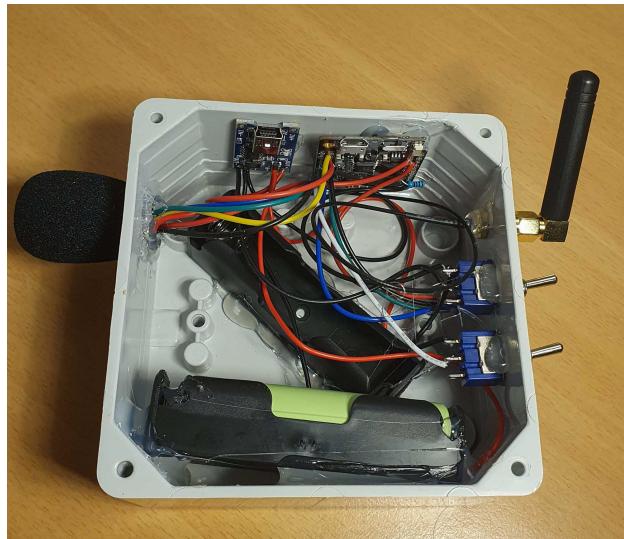
Slika 4.3: Električne povezave v merilnih enotah. Barva povezav na sliki se ujema z uporabljenimi kabli v sestavljenih enotah. Črna in rdeča barva sta vedno uporabljeni za napajanje, zelena je uporabljena za urni signal, modra pa za prenos podatkov. Rumena in bela se uporablja za ostale namene.



Slika 4.4: Material za sestavo meritne enote. Na sliki sta bateriji 18650 z držalom, 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 4.5: Sestavljeni model meritne enote, kjer so razvidni antene in stikala (a), in pokrivalo mikrofona namenjeno dušenju vetra in zaščiti pred dežjem (b).



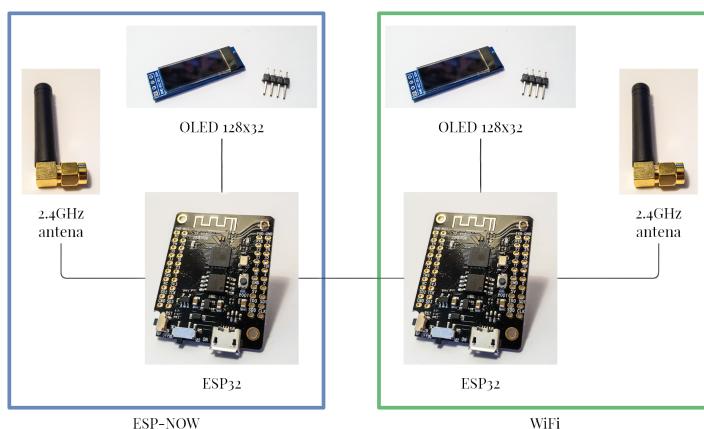
Slika 4.6: Notranjost sestavljenih merilnih enot. Vidne so povezave med mikrofonom, ekranom, polnilnim vezjem, stikali in TTGO T7. Vse povezave so barvno usklajene s projektom. Odprtine so zatesnjene z vročim lepilom za preprečevanje vdora vode.

Elektronsko opremo smo montirali v plastično ohišje (slika 4.5 , in 4.6), ki ima tesnilo proti vdoru vode in prahu. Za mikrofon, ekran, stikalo in anteno smo v ohišje zvrstali odprtine, ki smo jih kasneje zatesnili z vročim lepilom. Mikrofon smo prekrili s peno, ki zmanjšuje verjetnost vdora vode, šuma zaradi vetra in zamašitve mikrofona s prahom.

V sklopu izdelave merilnih enot smo optimizirali tudi čas, ki ga namenimo izdelavi vsake enote. Najbolj učinkovito smo zmanjšali sestavljalni čas ob izdelavi več enot naenkrat. Korake smo izvajali v istem vrstnem redu kot če bi sestavliali eno enoto, le da smo ga ponovili za vsako enoto posebej. Tako smo naenkrat zvrstali 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.

## 4.2 Zbirna enota

Zbirna enota (slika 4.7) je zaključen del sistema, ki je odgovoren za sprejemanje sporočil z merilnih enot, časovno usklajevanje njihovega delovanja in posredovanje podatkov na zaledni del. Zbirne enote potrebujemo, ker merilne enote v načinu zaznavanja ne morejo komunicirati z zalednim delom, saj 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.



Slika 4.7: 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.

Osrčje te enote sta dva mikrokontrolerja ESP32 na razvojnih ploščah T7. Mikrokontroler ESP32 lahko naenkrat komunicira samo z enim brezžičnim protokolom. Mikrokontrolerja si podatke izmenjujeta po serijski UART povezavi s protokolom PJON<sup>4</sup>. Ta skrbi za zanesljivo prenašanje sporočil in naslavljjanje.

Vsaka zbirna enota je opremljena z dvema OLED prikazovalnikoma, ki

---

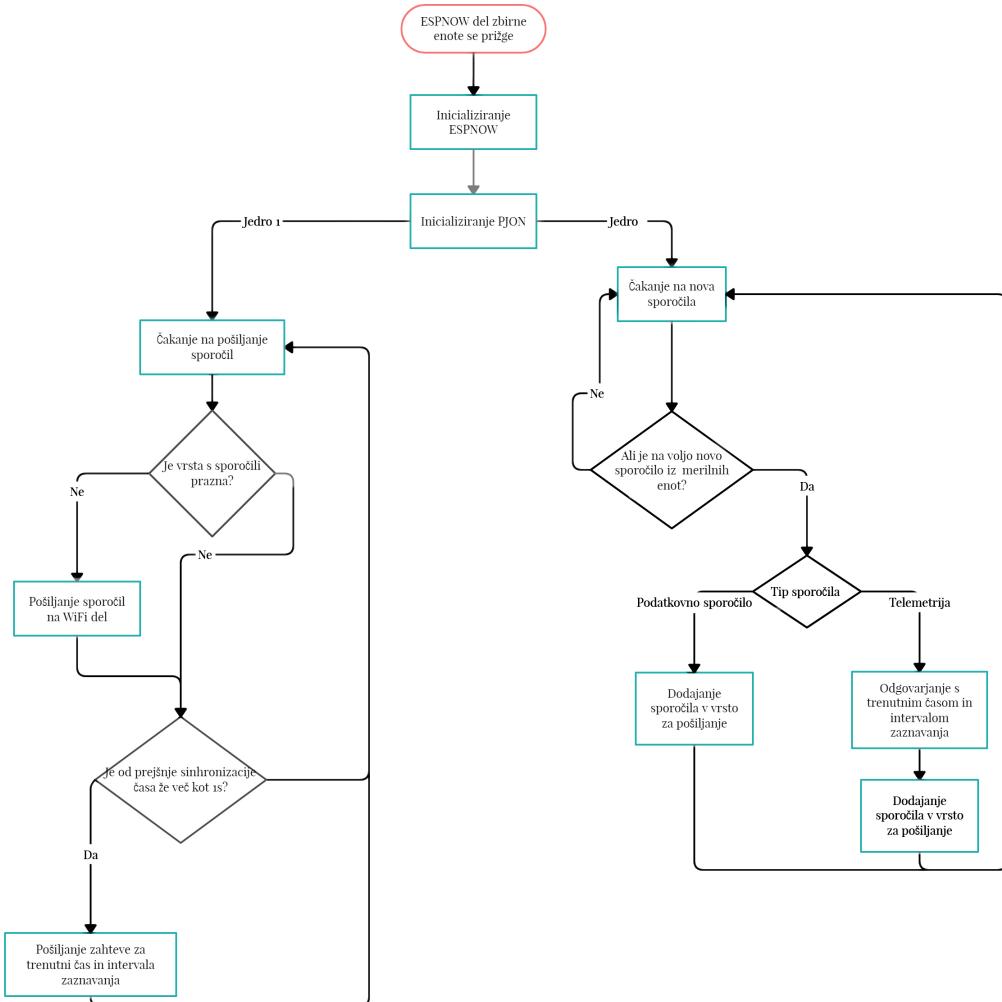
<sup>4</sup><https://www.pjon.org/documentation.php>

prikazujeta količino preostalega pomnilnika RAM, povprečen povratni čas do zalednega dela, število prejetih sporočil iz meritnih enot prejetih v zadnji sekundi, moč WiFi signala, MAC naslov te enote, dodeljeno ime enote in število sekund delovanja brez komunikacijskih napak.

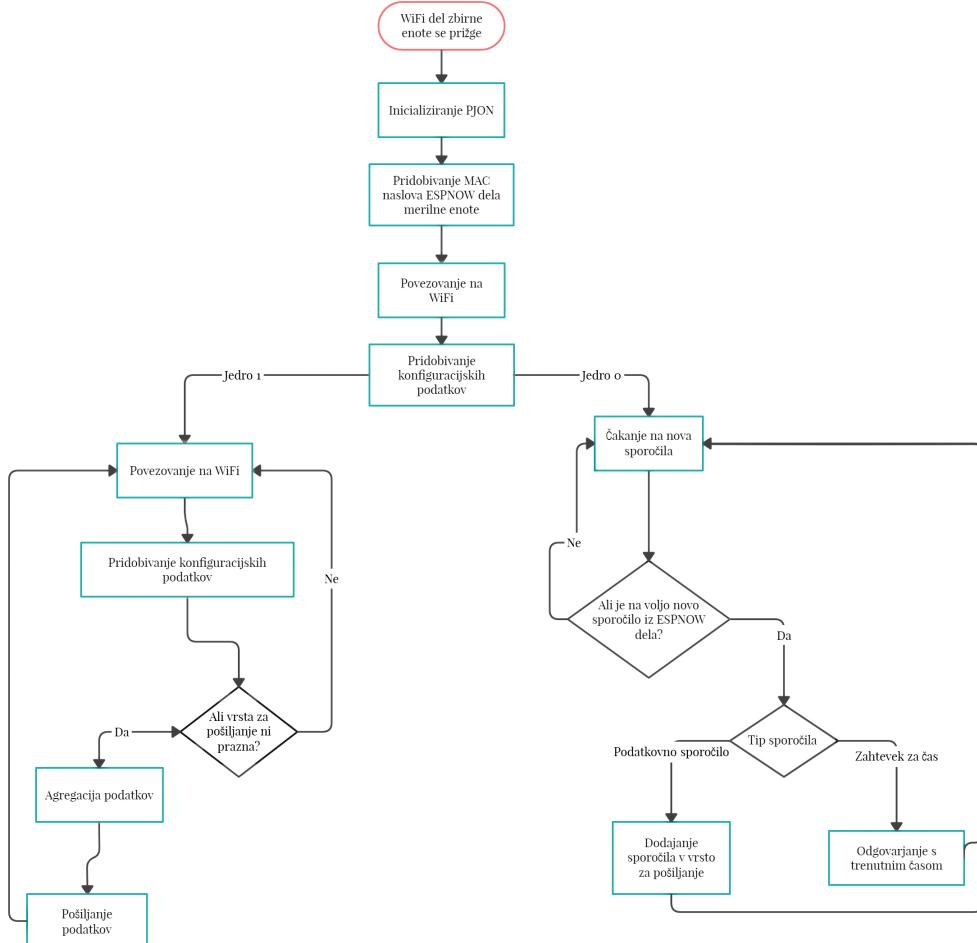
Posebnost ESP32 je, da ima dvojedrni procesor. V primeru meritne enote je to zaradi večje porabe hiba, pri zbirni enoti pa smo to eno jedro namestili komunikaciji z drugim delom enote, drugo pa komunikaciji s protokolom ESP-NOW ali s protokolom WiFi. Za vsako aktivnost smo napisali svoj proces in ga dodelili svojemu jedru. Pri tem smo si pomagali z operacijskim sistemom FreeRTOS. Za komunikacijo med deloma enote smo določili 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 4.8) z meritnimi enotami je zadolžen za sprejem podatkov, sinhronizacijo časa in sprejem telemetričnih podatkov. Ob zagonu najprej ponastavi komunikacijo s protokoloma ESP-NOW in PJON, potem pa vsakemu jedru dodeli svoj proces. Jedro 0 potem čaka na sporočila z meritnih enot. Ko sporočilo 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. Proses 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.

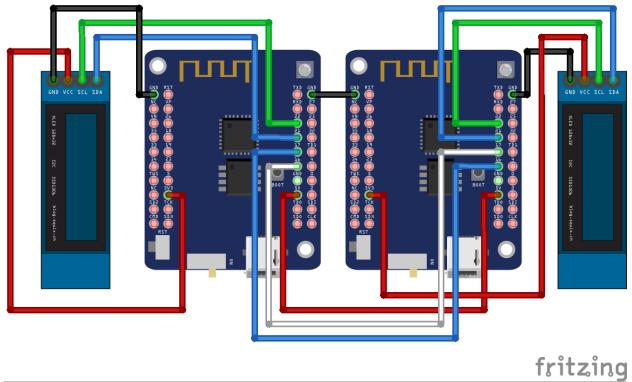
Del zbirne enote, ki komunicira s protokolom WiFi (slika 4.9) z zalednim delom, je zadolžen za registracijo enote, sinhronizacijo časa in prenos podatkov o meritvah in o telemetriji na zaledni del. Ob zagonu najprej ponastavi komunikacijo s protokoloma WiFi in PJON in pošlje MAC naslov enote na



Slika 4.8: 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 4.9: 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.



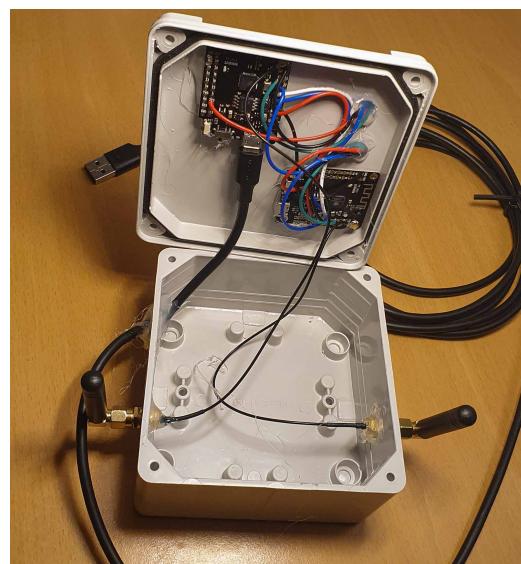
Slika 4.10: Električne povezave v zbirni enoti. Rdeče in črne povezave so namenjene napajanju, modre so namenjene prenosu podatkov, zelene pa ur-nemu signalu. Posebnost je UART povezava med enotama, ki je tu prikazana z modro in belo povezavo.



Slika 4.11: Material za zbirno enoto. Uporabili smo dve razvojni ploščici TTGO T7, dva prikazovalnika SSD1306, dve anteni s kabli in ohišje.



Slika 4.12: Zbirna enota. Na pokrovu sta nameščena dva prikazovalnika, ob straneh pa dve anteni in napajalni USB kabel.



Slika 4.13: Notranjost zbirne enote. Vidna 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 tesnjenu trajno pritrjen v ohišje.

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 zažene časovno sinhronizacijo s protokolom NTP, nato z 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 narazanimi žicami električno povezali. Celemu projektu smo določili enotno barvno shemo žic in njihovih namenov (slika 4.10). 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 4.11) smo namestili v plastično ABS ohišje z zaščito proti vodi in drugimi vplivi. Vse zvrtane luknje smo zatesnili z vročim lepilom (slika 4.13), da smo preprečili vdor vode. Na zunanjost enote (slika 4.12), 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.

## 4.3 Poraba energije

Ena od glavnih zahtev projekta je bila možnost napajanja merilnih enot z baterijami. To je predpogoj za 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 vsaj 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.



Slika 4.14: 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.

Prvi princip, s katerim smo zmanjšali porabo energije, je spanje<sup>5</sup>. Celotno zajemanje zvoka, obdelava meritve in pošiljanje podatkov traja manj kot eno tretjino sekunde, tako da lahko ostali dve tretjini mikrokontroler spi. Naš 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 potem obnaša tako, kot da bi se ponovno zagnala. 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, a se vsebina delovnega pomnilnika ohrani in se izvajanje programa po spanju takoj nadaljuje. Zato radi tega je dremež veliko bolj primeren. Tako enota približno 250 ms zbirata in obdeluje podatke, preostali čas pa spi. To obnašanje je razvidno na sliki 4.14. Z dremežem smo povprečno porabo zmanjšali na približno 20 mA.

Programsko kodo, ki se izvede ob vsakem ciklu zaznavanja prikaže spodnji izpis:

---

<sup>5</sup><https://lastminuteengineers.com/esp32-sleep-modespower-consumption/>

```
\small
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();
    }
}
```

Programska koda 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. Zatem pošlje podatke in uskladi informacijo o trenutnem času, na koncu izračuna preostali čas za spanje in ta čas spi.

Drugi pristop za zmanjševanje energijske porabe, ki smo ga pri implementaciji upoštevali, je dinamično nastavljanje frekvence delovanja mikro-

kontrolerja. Izbiramo lahko med 240 MHz, 160 MHz, 80 MHz, 40 MHz, 20 MHz in 10 MHz<sup>6</sup>. Počasnejše delovanje procesorja pomeni daljše izvajanje iste kode. Tega načeloma nočemo, ker je najbolje, da je mikrokontroler čim več časa v načinu spanja. Frekvenco procesorja zmanjšamo na delih, ki niso računsko zahtevni. Zbiranje podatkov iz mikrofona, ki predstavlja najdaljši del aktivnosti, je 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, saj je najmanjša frekvenca I2S signala za pridobivanje podatkov iz mikrofona 20 MHz, Pri obdelavi podatkov frekvenco dvignemo na 240 MHz. Tako lahko iz grafa porabe skozi čas predpostavimo v katerem delu cikla delovanja je trenutno enota. (slika 4.15). Z upočasnitvijo delovanja procesorja, smo porabo enote za zajemanje podatkov zmanjšali na približno 15 mA.



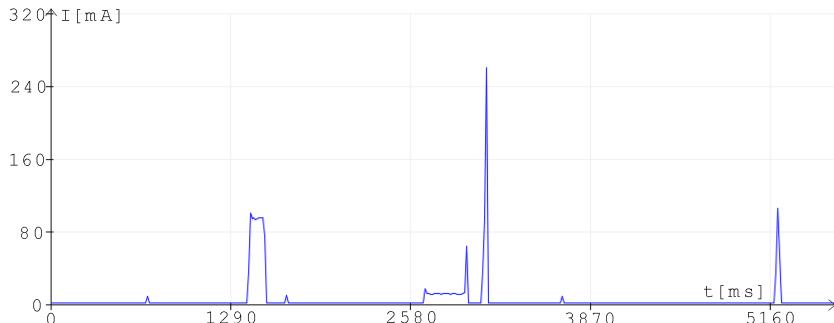
Slika 4.15: Graf porabe z različnimi deli zaznavnega cikla. Enota najprej približno 650 ms spi, kar se ujema z zelo nizko porabo. Sledi pridobivanje podatkov o zvoku, ki traja približno 250 ms in ima zaradi nizke frekvence procesorja majhno porabo. Obdelava podatkov, poteka ob največji hitrosti procesorja in se ujema z vrhom, ki traja le nekaj milisekund.

Tretji pristop za zmanjševanje energijske porabe, je paketno pošiljanje podatkov in optimizacija intervala za sinhronizacijo časa. Največja neučinkovitost

---

<sup>6</sup>[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power\\_management.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power_management.html)

pošiljanja podatkov ob vsakem zaznavanju je inicializacija dela mikrokontrolerja, ki je zadolžen za brezžično komunikacijo. Pred vsakim dremežem je treba ta del pravilno onemogočiti in pred ponovnim pošiljanjem omogočiti. To pri 80 MHz traja 20 ms, pri 240 MHz pa 8 ms in se ne poveča, če mikrokontroler pošlje več podatkov. Tako se bolj splača, da podatke meritev pošiljamo vsakih nekaj sekund. To se najbolje vidi na sliki 4.14, kjer najvišji vrh sovpada s pošiljanjem podatkov. Na podoben način smo optimizirali tudi porabo pri časovnem usklajevanju. Enoto morajo zaradi natančnosti čim bolj istočasno zaznavati hrup. 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 4.14 in sovpada z drugim najvišnjim vrhom. Te optimizacije so porabo zmanjšale pod 10 mA.



Slika 4.16: Na grafu porabe je razviden 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.

Uporabniku smo omogočili dodatno možnost optimizacije z uravnavanjem intervala zaznavanja. Z daljšim časom med zaznavanji je enota večji del časa obratovanja v načinu dremeža, kar zmanjša porabo. Na sliki 4.16 se vidi, kako se poleg enega zaznavanja, enega pošiljanja in dveh usklajevanj časa ne dogaja prav dosti. Enota se kljub temu zbudi vsako sekundo, da preveri, če

Tabela 4.1: 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.

Interval zaznavanja (s)	Povprečni električni tok (mA)	Avtonomija baterije (dni)
1	8.35	29.3
2	5.65	44.25
3	4.65	53.763
4	4.14	60.39
5	3.80	65.79
6	3.60	69.44
7	3.50	71.43
8	3.35	74.63
9	3.25	76.92
10	3.15	79.37
1000	2.61	95.79

je na vrsti za zaznavanje. To na grafu sovpada z najmanjšimi odstopanji od porabe med spanjem. S preudarnim nastavljanjem intervala med zaznavanji lahko tako uporabnik podaljša trajanje baterije(tabela 4.1).

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 med 5 mA in 10 mA. Uporabljen razvojna ploščica uporablja regulator ME6211, ki sam od sebe porabi le nekaj mikroamperov. Pri sestavljanju smo z razvojnih ploščic prav tako odstranili LED diode.



# Poglavlje 5

## Merilni strežnik

Poglavlje opisuje podporno arhitekturo za hranjenje, urejanje in pridobivanje podatkov, ter razvita orodja za upravljanje razvitega sistema. Glavna zahteva, ki smo jo upoštevali pri implementaciji, je da je projekt in vsa uporabljena ogrodja odprtakoden, ter da so vsi deli tega sistema lahko nameščeni na poljubne strežnike.

### 5.1 Strežnik

Programska oprema, ki smo jo namestili na strežnik laboratorija za bioinformatiko ULFRI, je bila razvita s programskim skladom MEAN. Njena primaarna naloga je skrb za shranjevanje, dostop, spreminjanje in osnovno obdelavo podatkov. Zadolžena je tudi za omogočanje dostopa do uporabniškega vmesnika.

Oblika na strežniku zapisanih podatkov ustreza delovanju merilnih in zbirnih enot, ter organizaciji merilnih študij. Tako ima vsaka enota in študija svoj zapis. Izjema so podatki, kjer smo več podatkov združili v en zapis. Podatki se v podatkovni bazi hranijo v obliki zapisa JSON. V tej obliki so dostopni tudi preko implementiranega vmesnika. Za dostop in delo s podatki smo implementirali vmesnik REST API [12].

Zaradi podobne oblike podatkov in podobnih načinov uporabe teh podat-

kov imata enoti enak REST API vmesnik. Spodaj je opisan vmesnik merilne enote, ampak enako velja za zbirno enoto.

## Oblike zapisov v podatkovni bazi

### SensorSchema in GatewaySchema

Merilna in zbirna enota sta podobni entiteti, zato sta tu opisani skupaj. Obe enoti imata v zapisu podatkov predpostavljeno polje za `ObjectId`, ki se uporablja pri identifikaciji znotraj strežniškega dela. Za identifikacijo obe enoti uporabljata naslove MAC 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.

#### Oblika:

**Name** Znakovni niz uporabljen za identifikacijo naprav.

**MAC** Tabela števil uporabljena za interno naslavljjanje enot.

**CurrentDeployment** Trenutna študija kateri je enota dodeljena.

**CurrentLocation** Trenutna lokacija merilne enote (velja le za merilne enote).

**LastTelemetry** Datum in čas zadnje telemetrije.

**LastData** Zapis zadnjih prejetih podatkov z naprave.

**BatteryVoltage** Napetost na bateriji (velja le za merilne enote).

**LatestMeasurement** Datum in čas zadnje prejete meritve.

**WifiCredentials** Podatki o WiFi omrežju (velja le za zbirne enote).

## DeploymentSchema

Študije, oziroma **deployments** so zaključene enote zaznavanja podatkov. Predstavljajo eno namestitev merilnih 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 merilnih in zbirnih enotah, trenutnem stanju, številu vseh meritev in trenutnem intervalu zaznavanja.

### Oblika:

**Name** Ime določeno s strani uporabnika.

**Description** Opis študije.

**Sensors** Seznam merilnih enot dodeljenih tej študiji.

**Gateways** Seznam Zbirnih enot dodeljenih tej študiji.

**Status** Trenutno stanje študije.

**MeasurementNum** Število opravljenih meritev.

**MeasuremetnInterval** Trenutni določen interval zaznavanja.

## DataSchema in MeasurementSchema

Pri hranjenju podatkov o meritvah, smo uporabili metodo združevanja podatkov v vedra s specificirano velikostjo. Tako smo združili po 1000 zaporednih meritev iz iste enote pri isti študiji v en zapis. S tem smo prihranili 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 shranjena identifikacija merilne enote in študije, lokacija zajetih podatkov, velikost, čas prve meritve, čas zadnje meritve in podatki zajeti z meritvami. V zapisu posamezne meritve se hranijo podatki o

frekvenčnem razponu podatkov spektralne analize, rezultati spektralne analize, glasnost v decibelih in časovna oznaka meritve zaokrožena na sekundo natančno.

#### **Oblika DataSchema:**

**Sensor** Referenca na merilno enoto, ki je zajela meritve.

**Deployment** Referenca na študijo katere del so ti podatki.

**Location** Lokacija zajetih podatkov.

**Size** Število meritev v zapisu.

**First** Čas prve meritve.

**Last** Čas zadnje meritve.

**Measurement** Shranjene meritve oblike MeasurementSchema.

#### **Oblika MeasurementSchema:**

**FrequencyRange** Frekvenčni razpon podatkov spektralne analize.

**FftValues** Rezultati spektralne analize.

**Decibels** Izračunana glasnost.

**MeasuredAt** Čas ob opravljeni meritvi.

### **Programski vmesnik strežnika**

**GET /api/sensor**

Vrne osnovne podatke o vseh merilnih enotah.

**GET /api/sensor/:sensor\_id**

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**

Registriranje nove enote. 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**

V zapis merilne enote se zapišejo novi telemetrični podatki. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

**PUT /api/sensor/:sensor\_id**

Posodobi se zapis specificirane merilne enote. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

**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 pri-dobi 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 meritev. Za identifikacijo se uporablja identifikacijska številka, ki je bila zapisu dodeljena s strani podatkovne baze.

```
POST /api/sensor/data
```

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

```
GET /api/data/deployment/:deployment_id
```

Klic vrne podatke vseh meritev, 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 meritev 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 meritev vsake meritne enote. Drugi klic pridobi vse meritve iz zadnjih n sekund.

## 5.2 Uporabniški vmesnik

The screenshot shows a web application interface for the Urban Noise Sensing Platform. At the top, there is a navigation bar with links for 'Urban Noise Sensing Platform', 'Deployments', 'Sensors', and 'Gateways'. Below the navigation bar, the main content area is titled 'Deployments'.

The 'Deployments' section is divided into three categories: 'Not yet deployed', 'Deployed', and 'Finished'.

- Not yet deployed:** Contains one item: 'Nov deployment za prikaz'. It includes a link 'Opis tega deploymenta' and a note 'Number of measurements: 0'.
- Deployed:** Contains two items: 'ok' and 'dnevna soba'. Both items include a note 'Number of measurements: 177690'.
- Finished:** Contains one item: 'dnevna soba'. It includes a note 'Number of measurements: 80599'.

Slika 5.1: Pregled trenutnih in preteklih študij. Vidi se minimalističen oblikovalski stil, ki smo ga uporabili na vseh delih vmesnika.

Ena glavnih zahtev tega projekta je bila prilagodljivost, saj bo raziskovanje z uporabo razvite opreme potekalo v sklopu manjših študij, ki bodo opravljene na različnih urbanih območjih. Na vsakem od teh območij bomo morali postaviti meritne enote v izbrana urbana okolja. 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 5.1) smo razvili na podlagi dveh konceptov **Model-Pogled-Kontroler** (MVC) in aplikacija na eni strani (SPA). Zaledni del tako deluje samo kot vir podatkov, ki deluje po principu REST API. Prikaz pri uporabniku ustvari aplikacija Angular<sup>1</sup>. Za oblikovanje smo uporabili knjižnico Bootstrap<sup>2</sup>.

Primarni namen uporabniškega vmesnika je upravljanje z meritnimi in zbirnimi enotami. Zato smo najprej implementirali dva pogleda (sliki 5.2), ki omogočata pregled nad trenutnim stanjem enot. Običajno nas najbolj zanima, kateri študiji je določena enota trenutno dodeljena, zato je pri dodeljenih enotah vedno izpisana ta informacija. Pri meritnih enotah nas poleg tega zanima še točna lokacija in stanje baterije.

Običajno za identifikacijo enot uporabljam naslove MAC in posebne naključne identifikacijske kode, ki jih baza podatkov naključno določi zapisom. Takšen način uporabnikom ni blizu, saj si težko zapomnimo več kot 7 naključnih zaporednih znakov. Zato smo implementirali sistem naključno izbranih lastnih imen z dodanimi naključnimi številkami. Takšno ime dobi vsaka enota ko se prvič poveže na zaledje. Ime posamezne enote je tako sestavljeno iz naključnega imena iz korpusa stotih najbolj pogostih angleških imen in naključne številke med 0 in 100. Na sliki 5.3 je primer imena na meritni enoti, na sliki 5.2 pa imena na spletnem uporabniškem vmesniku.

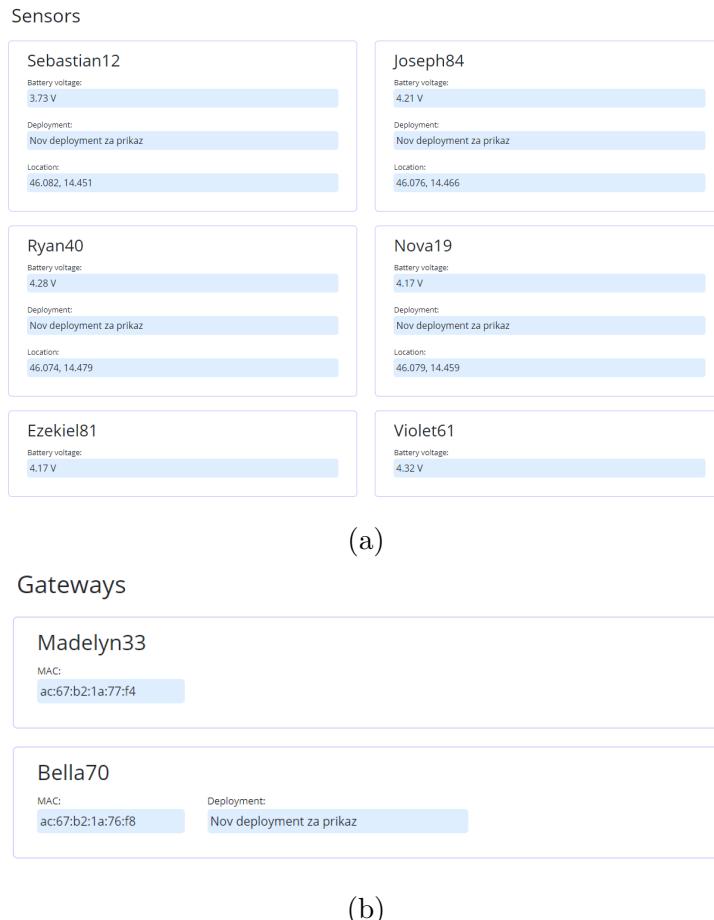
Drugi del upravljanja z enotami je ustvarjanje meritvenih študij in dodeljevanje enot tem študijam. Implementirali smo spletni vmesnik, kjer uporabnik lahko prosto postavlja proste enote na zemljevid (slika 5.4), izbira med zbirnimi enotami in nastavlja WiFi ime in geslo, ki bo dostopno na kraju študije.

Za hitro diagnosticiranje težav in sproten pregled nad rezultati študije

---

<sup>1</sup><https://angular.io/>

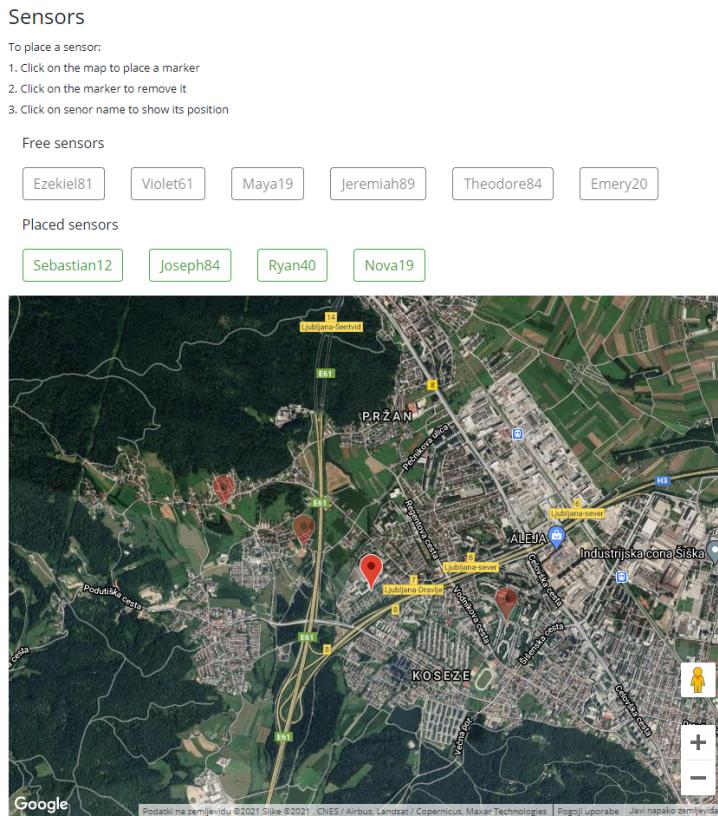
<sup>2</sup><https://getbootstrap.com/>



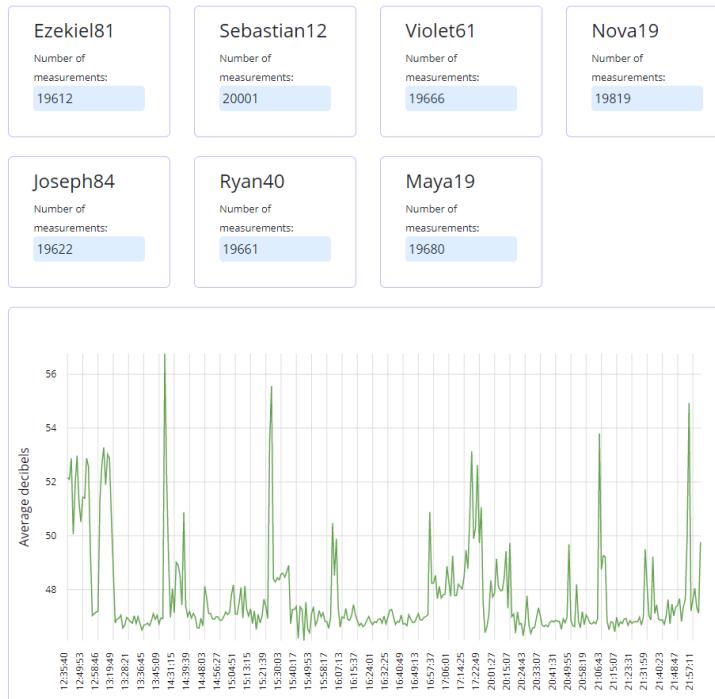
Slika 5.2: Pregled meritnih in zbirnih enot v spletnem uporabniškem vmesniku. Zapis meritnih enot (a), prikazujejo podatke o trenutni študiji, napetosti na bateriji in trenutno lokacijo, prav tako je pri vsaki enoti izpisano ime za lažjo identifikacijo. Podobno je pri zbirnih enotah (b), izpisano ime in trenutna študija.



Slika 5.3: Primer dodeljenega imena na OLED prikazovalniku ene od merilnih enot.



Slika 5.4: Postavljanje prostih merilnih enot na zemljevid. Uporabnik s klikom na zemljevid razvršča merilne 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 naznamek.

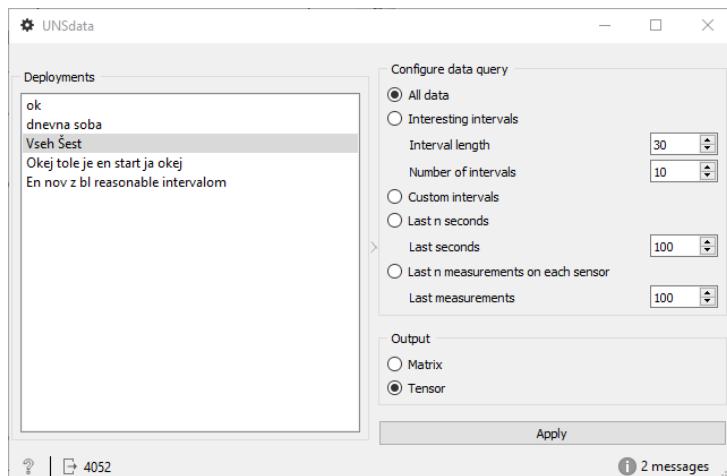


Slika 5.5: Pregled merilnih enot in povprečne glasnosti hrupa med študijo. Med zaznavanjem in po koncu zaznavanja lahko uporabnik pregleda te najbolj osnovne podatke o poteku študije. Graf je omejen na maksimalno 333 točk. Ko je meritev več, se izračuna povprečje več meritev.

smo uporabnikom omogočili pregled nad številom zaznavanj po merilnih enotah, povprečno glasnostjo med zaznavanjem (slika 5.5) in lokacije enot med zaznavanjem.

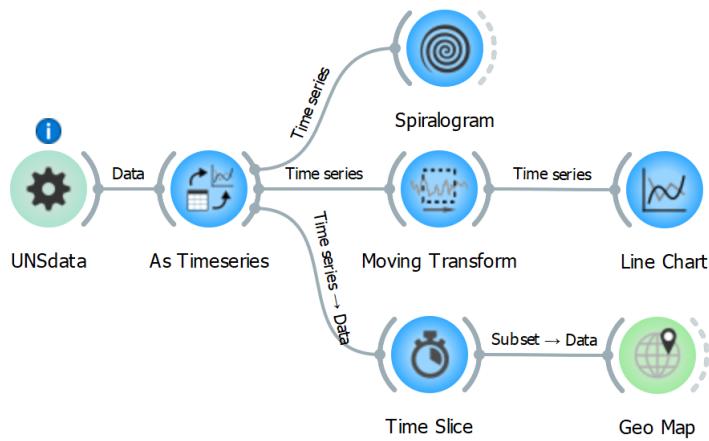
### 5.3 Podatkovna analitika

Razvili smo vmesnik za program za podatkovno rudarjenje Orange [2]. Posebnost tega programa je, da omogoča vizualno programiranje z uporabo gradnikov, ki jih uporabnik povezuje med sabo (slika 5.7). Vmesnik omogoča dostop do podatkov na zalednem strežniku in izbor študije (slika 5.6).



Slika 5.6: Zaslonski posnetek implementiranega gradnika UNSdata za dostop do podatkov na zalednem podatkovnem strežniku. 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 izstropnih podatkov in potrdi svojo izbiro.

Razviti gradnik (slika 5.6) 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 gradniki v programu. Implementiran princip uporabe predpostavlja, da uporabnik najprej s seznama izbere študijo, nato nastavi obliko poizvedbe in na



Slika 5.7: Primer delovnega toka v programu Orange [2]. 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.

koncu izbere obliko podatkov. Pridobljene podatke lahko potem z drugimi gradniki obdela in prikaže (slika 5.7).



# Poglavlje 6

## Primera uporabe

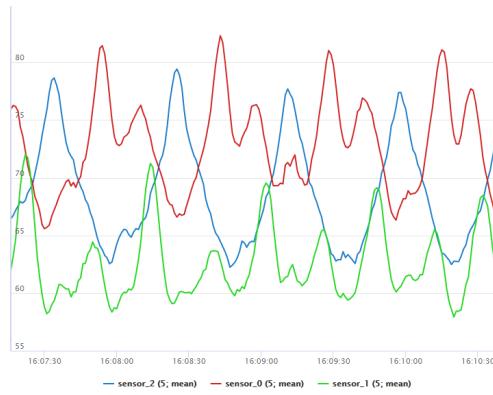
Z dvema primeroma uporabe smo želeli prikazati dva načina delovanja razvitega sistema. Primer dnevne sobe je iz stališča zaznavanja in analiziranja zvoka bolj tradicionalen, saj se osredotoča na trende in spremembe skozi daljša časovna obdobja. V nasprotju s tem pa primer kosilnice prikazuje inovativne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

### 6.1 Košenje trave

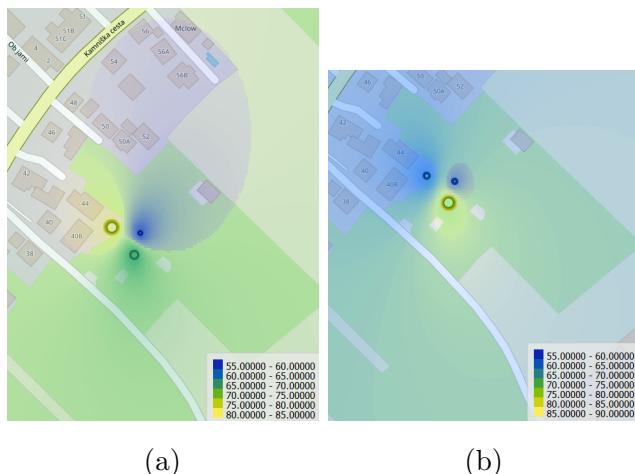
Pri načrtovanju primera uporabe s košenjem trave smo v ospredje postavili prikaz izvirnih zmožnosti našega sistema. Želeli smo prikazati, kako lahko zvoku sledimo z razporeditvijo več meritnih enot na manjše območje. Skupaj z možnostjo točnega nastavljanja lokacije posamezne enote in sinhronizirane zaznavanja 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 6.2). Zaznavanje je trajalo približno pol ure med košenjem zelenice. Interval med merjenji smo nastavili na eno sekundo.

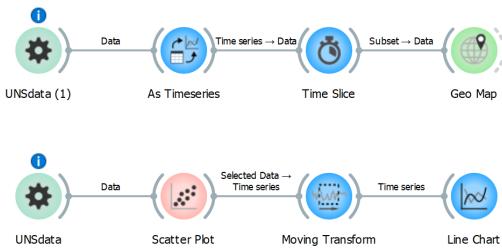
Po končanem zaznavanju smo podatke s pomočjo našega gradnika uvozili v program Orange (slika 6.3), jih obdelali in prikazali na zemljevidu (slika 6.2) in na grafu (slika 6.1). Na zemljevidu smo s pomikanjem skozi čas



Slika 6.1: Na grafu vseh meritev po času in senzorju lahko opazimo približevanje in oddaljevanje kosilnice 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 6.2: 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. Razvidno je premikanje kosilnice od bližine zahodne meritne enote (a) do bližine južne meritne enote (b). To je primerno tako za akutne spremembe v poziciji izvora zvoka, kot je prikazano tu, kot tudi za prikaz počasnejših sprememb.



Slika 6.3: Delovni tok v programu Orange [2]. 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.

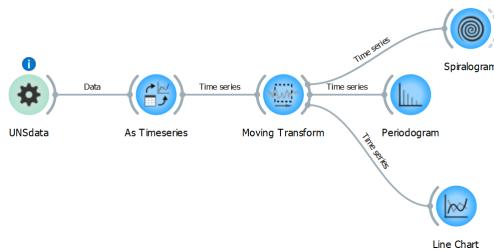
z gradnikom time slice opazovali kako se je kosilnica premikala med tremi merilnimi enotami v krogu, kar se sklada z dejanskim premikanjem. Na grafu smo opazili podobno, le da smo poleg tega opazili tudi sprotno zmanjševanje glasnosti, saj se je kosilnica v spirali premikala proti sredini travnika, kar 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 s stališča, časovne usklajenosti enot in cenovne dostopnosti, da tako lahko enote enostavno postavimo na manjše območje.

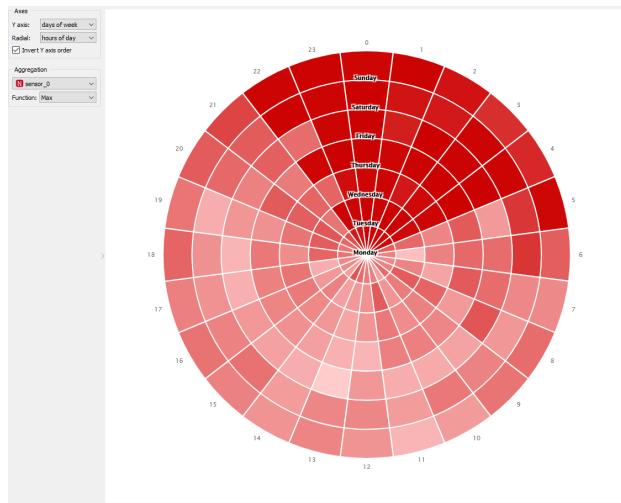
## 6.2 Glasnost v dnevni sobi

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

Merilno enoto smo postavili v dnevno sobo. Interval zaznavanja smo

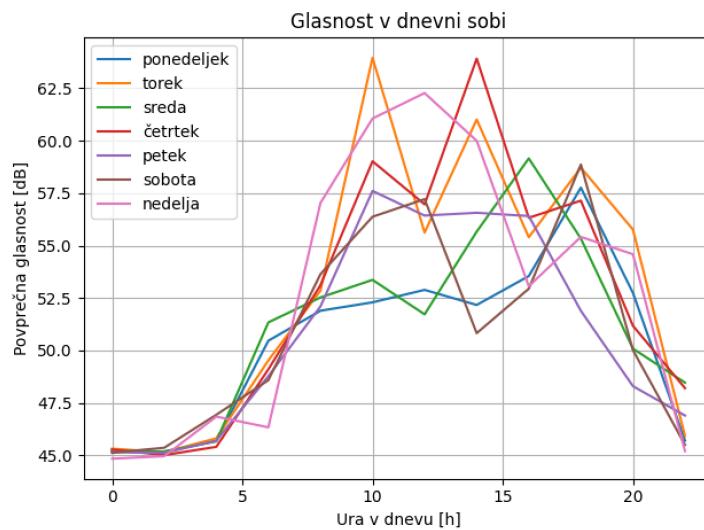


Slika 6.4: Delovni tok v programu Orange. Podatke o glasnosti v dnevni sobi smo z našim gradnikom v celoti prenesli v obliki matrice, te podatke smo potem spremenili v časovno vrsto in izračunali povprečje v intervalih pet minut. Rezultate smo potem prikazali na spiralogramu, periodogramu in grafu.



Slika 6.5: 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 spat, kdaj so se zjutraj zbudili in celo daljše spanje v soboto zjutraj.

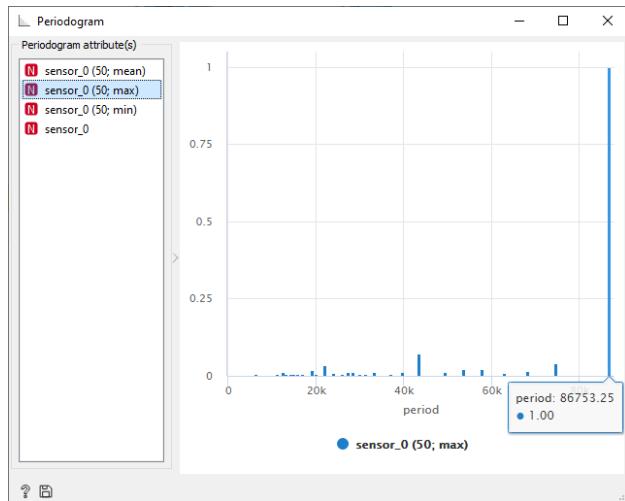
nastavili na deset sekund in pustili, da zbira podatke 10 dni. Podatke smo potem z uporabo razvitega gradnika za program Orange prenesli in obdelali (slika 6.4).



Slika 6.6: Graf glasnosti v dnevni sobi po dnevih.

Pri tem primeru nas je zanimala periodičnost podatkov, torej kakšen je trend spanja in zbujanja družine, kdaj so otroci v šoli in starši v službi, kdaj se samodejno vklopi pomivalni stroj in podobno. Za to smo uporabili spiralogram (slika 6.5) in periodogram (slika 6.7).

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.



Slika 6.7: 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).

# Poglavlje 7

## Zaključek

Cilj projekta je bil razviti sistem za zaznavanje in analizo hrupa. Ta mora biti enostaven za uporabo, prilagodljiv in cenovno učinkovit. Merilne enote lahko postavimo prosto brez zunanjega napajanja in trajne namestitve. Prav tako smo hoteli vpeljati možnost dinamičnega nastavljanja intervalov med zaznavanji, da lahko poleg dolgotrajnih trendov spremljamo tudi akutne spremembe.

Razviti sistem sestavljajo fizične enote za zajem in prenašanje podatkov, zaledni strežnik, uporabniški vmesnik in gradnik za program Orange [2]. Enote, ki jih napajajo polnilne baterije, lahko s poljubnim intervalom zajemajo podatke o hrupu vsaj tri tedne. Zvočne podatke vse enote zajemajo časovno usklajeno v enakomernih časovnih razmakih. Zajete podatke pošiljajo na zbirno enoto, ki je lahko več kot 200m oddaljena od merilnih enot, ki prejete podatke posreduje na zaledni strežnik. Na strežniku so podatki urejeni v študije. Študije nastavi uporabnik. V sklopu nastavljanja študij je uporabniku omogočena izbira merilnih enot, določane točne lokacij teh enot in sprotno nastavljanje intervala zajema podatkov. Tekom študije je uporabniku omogočeno spremjanje poteka zajemanja podatkov na uporabniškem vmesniku. Podatke zaključene študije lahko uporabnik vizualizira in obdeluje v programu Orange.

Razvili smo merilne in zbirne enote iz prosto dostopnih komponent, s

primernim dosegom signala, baterijskim napajanjem in enostavnim nastavljanjem. Za shranjevanje podatkov in upravljanje smo razvili podatkovni strežnik. Za upravljanje in nastavljanje smo implementirali intuitiven uporabniški vmesnik. Za analiziranje podatkov v programu Orange [2] smo razvili gradnik za pridobivanje zajetih podatkov. Vsa programska koda je prosto dostopna na repozitoriju git na portalu Github<sup>1</sup>.

Sestavljanje merilnih 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; študije, kjer smo z njimi pričeli sodelovati in kjer uporabljam razviti sistem za študije v različnih urbanih okoljih so v teku.

---

<sup>1</sup><https://github.com/andraz213/UrbanNoiseSensing>

# Dodatek A

## Dodatek: Uporabniška navodila

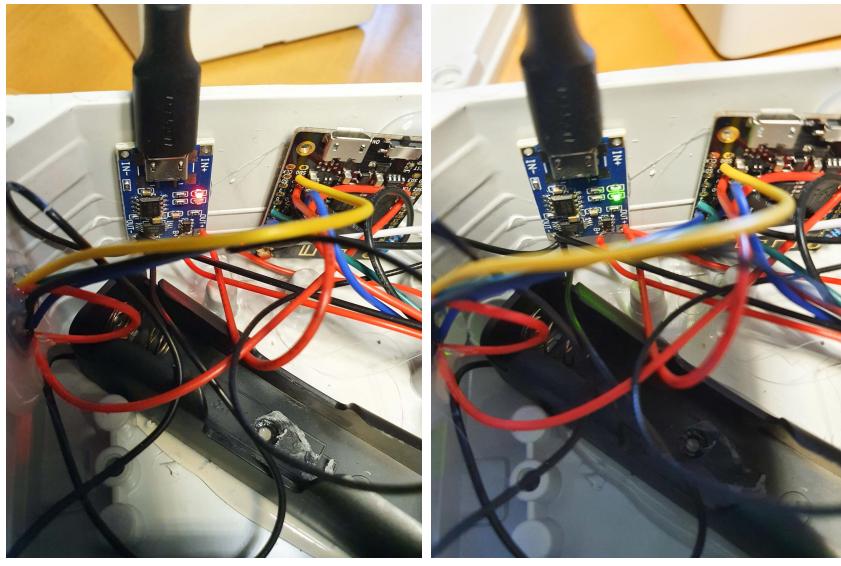
### A.1 Polnjenje meritnih enot

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

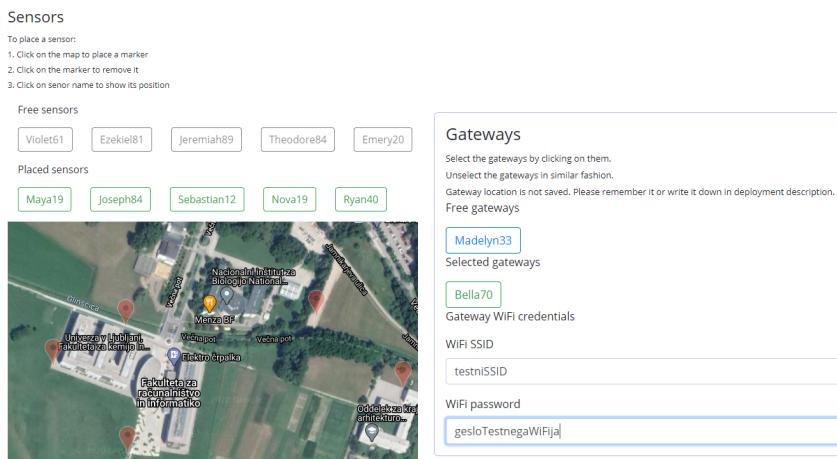
Pred začetkom zaznavanja je treba nastaviti in zagnati raziskavo. Spletni vmesnik je dostopen na naslovu: ”<http://urbannoisesensing.biolab.si/>”.

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 gesлом "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.

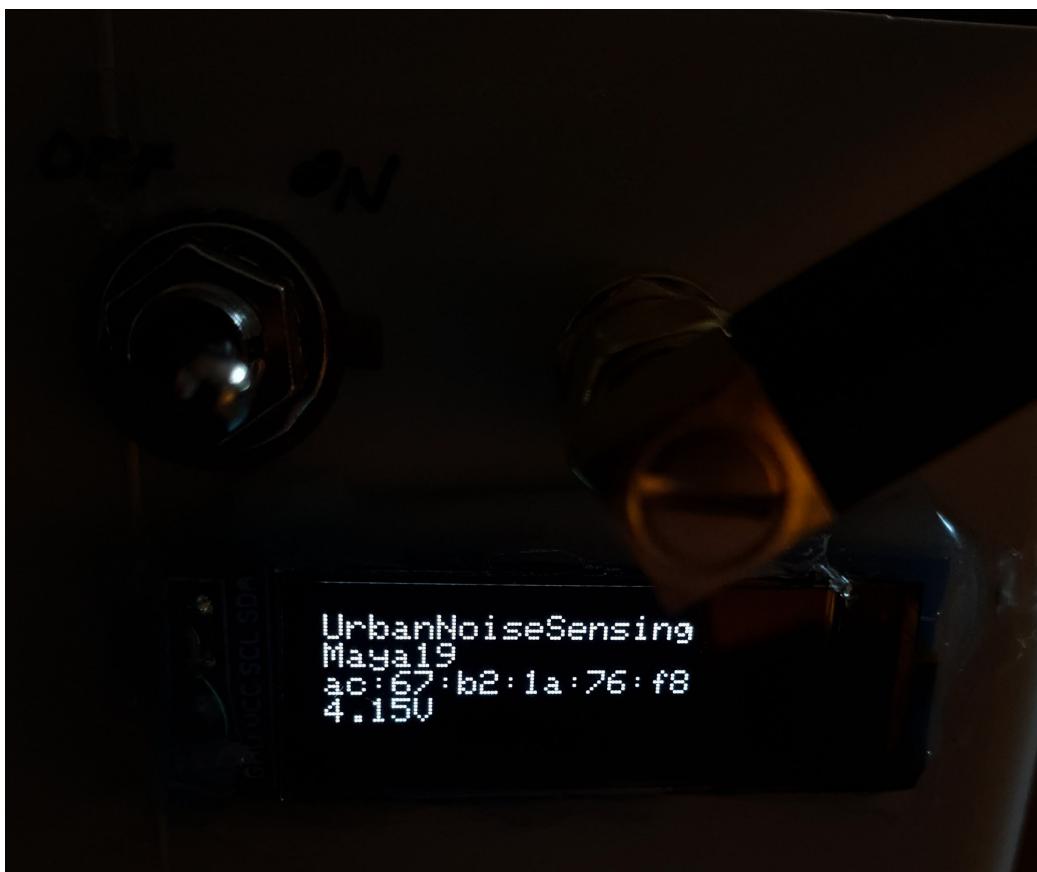
4. Ugasnite enote.

## 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, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.
- [3] A. Di Nisio, T. Di Noia, C. G. C. Carducci, and M. Spadavecchia. Design of a low cost multipurpose wireless sensor network. In *2015 IEEE International Workshop on Measurements Networking (M N)*, pages 1–6, 2015.
- [4] W. Ejaz, A. Anpalagan, M. A. Imran, M. Jo, M. Naeem, S. B. Qaisar, and W. Wang. Internet of Things (IoT) in 5G Wireless Communications. *IEEE Access*, 4:10310–10314, 2016.
- [5] 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.
- [6] T. N. Hoang, S. Van, and B. D. Nguyen. ESP-NOW Based Decentralized Low Cost Voice Communication Systems For Buildings. In *2019 Inter-*

- national Symposium on Electrical and Electronics Engineering (ISEE)*, pages 108–112, 2019.
- [7] K. Khanchuea and R. Siripokarpirom. A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation. In *2019 10th International Conference of Information and Communication Technology for Embedded Systems (ICTES)*, pages 1–6, 2019.
  - [8] A. Maier, A. Sharp, and Y. Vagapov. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In *2017 Internet Technologies and Applications (ITA)*, pages 143–148, 2017.
  - [9] Anne Vernez Moudon. Real Noise from the Urban Environment: How Ambient Community Noise Affects Health and What Can Be Done About It. *American Journal of Preventive Medicine*, 37(2):167–171, 2009.
  - [10] 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.
  - [11] Courtney Peckens, Cédric Porter, and Taylor Rink. Wireless sensor networks for long-term monitoring of urban noise. *Sensors*, 18(9):3161, 2018.
  - [12] 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.
  - [13] George Suciu, Ana Lavinia Petrache, Cristina Badea, Tony Buteau, David Schlachet, Loic Durand, Matthieu Landez, and Ijaz Hussain. Low-

- power IoT devices for measuring environmental values. In *2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging(SIITME)*, pages 234–238. IEEE, 2018.
- [14] Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range IoT technologies: The dawn of LoRa<sup>TM</sup>. In *Future access enablers of ubiquitous and intelligent infrastructures*, pages 51–58. Springer, 2015.