Određivanje lokacije ronioca pomoću triangulacije signala akustičnih jedinica

Bradarić Lisić, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1521

ODREĐIVANJE LOKACIJE RONIOCA POMOĆU TRIANGULACIJE SIGNALA AKUSTIČNIH JEDINICA

Luka Bradarić Lisić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1521

ODREĐIVANJE LOKACIJE RONIOCA POMOĆU TRIANGULACIJE SIGNALA AKUSTIČNIH JEDINICA

Luka Bradarić Lisić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1521

Pristupnik:	**Luka Bradarić Lisić (0036539200)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Hrvoje Džapo

Zadatak: **Određivanje lokacije ronioca pomoću triangulacije signala akustičnih jedinica**

Opis zadatka:

Istražiti metode i matematičke postupke određivanja lokacije primjenom akustičkih senzora. Proučiti mogućnosti akustičke jedinice Succorfish Delphis v3 za slanje i primanje signala pod vodom. Implementirati prototipni sustav temeljen na mreži čvorova s akustičnim jedinicama za podvodnu lokalizaciju ronioca koji se sastoji od tri stacionarne površinske bove i jedinici na roniocu. Akustična jedinica svakog čvora upravljana je mikrokontrolerom putem serijskog sučelja. Implementirati programsku potporu za mikrokontroler koja će omogućiti generiranje i očitavanje akustičnih signala i lokalizaciju ronioca obradom izmjerenih vremena propagacije među čvorovima. Sustav treba dodatno omogućiti slanje izmjerenih podataka o kretanju ronioca i GPS-lokacije pojedinačnih bova na obalu pomoću LoRa-mreže.

Rok za predaju rada: 14. lipnja 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1521

Određivanje lokacije ronioca pomoću triangulacije signala akustičnih jedinica

Luka Bradarić Lisić

Zagreb, kolovoz 2024.

Zahvaljujem kompaniji H2O-Robotics na svoj pruženoj pomoći i savjetima koji su mi dani pri realizaciji ovog projekta.

SADRŽAJ

1. Uvod

Jedan od najvažnijih izazova u podvodnom ronjenju je praćenje lokacije ronioca. Poseban problem predstavlja prestanak praćenja lokacije, što se može dogoditi u raznim situacijama npr. kod ronjenja u vodama sa slabom vidljivošću, ronjenja na velikim dubinama gdje svjetlost ne doseže, ronjenja u području snažnih struja gdje ronioc može izgubiti kontrolu nad kretanjem itd. Zbog takvih razloga svaki dan ronioce se oprema novim uređajima kako bi se rizik od takvog mogućeg događaja smanjio.

U ovom radu predložena je nadogradnja postojećeg sustava za praćenje lokacije ronioca u cilju olakšanja praćenja njegove lokacije vanjskim promatračima, uz ocjenu dubine ronjenja korištenjem posebno razvijenog skupa elektroničkih uređaja. Takav sustav treba omogućiti da se lokacija ronioca prati i bilježi u bazi podataka u kojoj se svaka ruta ronioca može zabilježiti. Za realizaciju sustava potrebno je prikupiti skup podataka koji omogućuje aproksimaciju lokacije ronioca.

U nastavku rada dan je opis korištenih elektroničkih sklopova, detaljan opis implementirane programske potpore te analiza funkcionalnosti sustava s opisom prednosti i nedostataka koji bi se u budućim nadogradnjama mogli poboljšati. U prvom poglavlju opisuje se opća funkcionalnost sustava. U drugom poglavlju opisuje se sklopovska implementacija sustava i korištene tehnologije. U trećem poglavlju opisuje se programska potpora te metodologije i algoritmi koji su korišteni u tu svrhu. Četvrto poglavlje opisuje testiranje sustava i dobivene rezultate.

2. Opis sustava

Pri određivanju lokacije ronioca teško je izravno dobiti njegovu lokaciju izravnim spajajnjem uređaja za pozicioniranje kao što su standardni GPS i GNSS (engl. *Global Navigation Satellite System*) moduli. Razlog tome su podvodni uvjeti u kojima se ronioc kreće jer GNSS signali ne prodiru pod vodu. Profesionalni ronioci mogu biti na dubinama od 10 do 50 metara, što su uvjeti u kojima je propagacija radio signala gotovo nemoguća bez upotrebe posebnih tehnika pojačavanja signala koje nisu lako dostupne široj okolini.

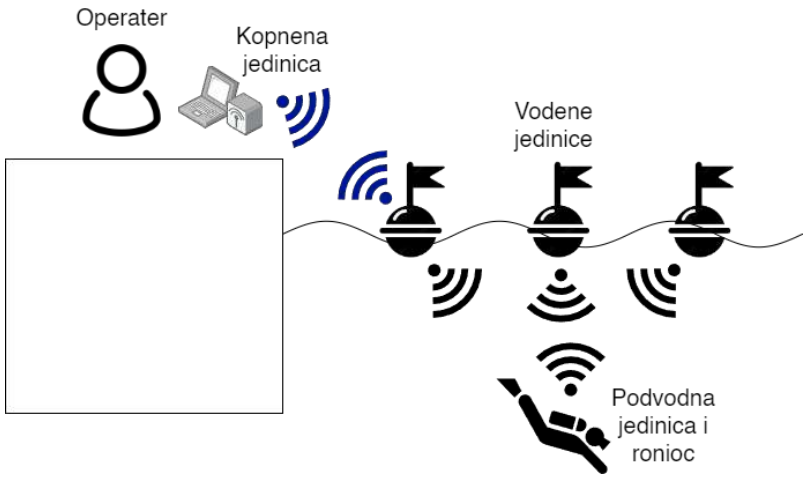
Kako bi se omogućilo lociranje ronica pod vodom, za njegovu lokalizaciju ne koriste se radio signali, već ultrazvučni koji putuju kroz vodu. U tu svrhu koriste se ultrazvučni pretvornici (engl. *ultrasonic transducer*) koji u komunikaciji s akustičnom jedinicom mogu odrediti međusobnu udaljenost i time informaciju o relativnoj poziciji ronioca u odnosu na referentnu točku. Da bi se dobila lokacija, mora se koristiti više akustičnih jedinica iz koje se određuju parovi udaljenosti prema ultrazvučnom pretvorniku na roniocu.

Sustav se sastoji od tri glavne cjeline:

- kopnena jedinica,
- skup vodenih jedinica,
- podvodna jedinica.

Kopnena jedinica je namijenjena za direktnu interakciju osobe koja je zadužena za kopneni nadzor ronioca (operatera) sa skupom vodenih jedinica. Ova jedinica se sastoji od mikrokontrolera i *LoRa* modula, kojemu je glavna zadaća primanje zahtjeva operatera te prikupljanje izmjerenih podataka pozicije i udaljenosti ronioca. Prikupljene informacije se pritom procesiraju na računalu operatera te vizualiziraju.

Vodena jedinica je namijenjena kako bi na zahtjev kopnene jedinice prikupila podatke o apsolutnoj poziciji pomoću GPS modula te kako bi mogla prikupiti podatke o međusobnoj udaljenosti dotične jedinice i ronioca. Tako prikupljene podatke pritom šalje natrag kopnenoj jedinici da daljnju obradu. Kako bi se smanjila mogućnost pogreške pri slanju naredbe s kopna i da bi se unificirao cijeli protokol, samo jedna vodena jedinica je zadužena za direktnu interakciju s kopnenim sustavom. Ostale vodene jedinice služe kako bi prikupile podatke i poslale ih natrag na glavnu vodenu jedinicu.



Slika 2.1: Blok-shema sustava

Podvodna jedinica je napravljena s namjenom da šalje odgovor vodenoj jedinici pri očitavanju udaljenosti. Razlog zašto mora poslati odgovor je kako bi se pomoću *ping* algoritma moglo odrediti vrijeme propagacije zvučnog impulsa kroz vodu. Jedinica se sastoji od akustičkog modula te baterijskog paketa koji su postavljeni u vodonepropusnu cijev. Prikaz tako opisanog sustava se može vidjeti na slici 2.1.

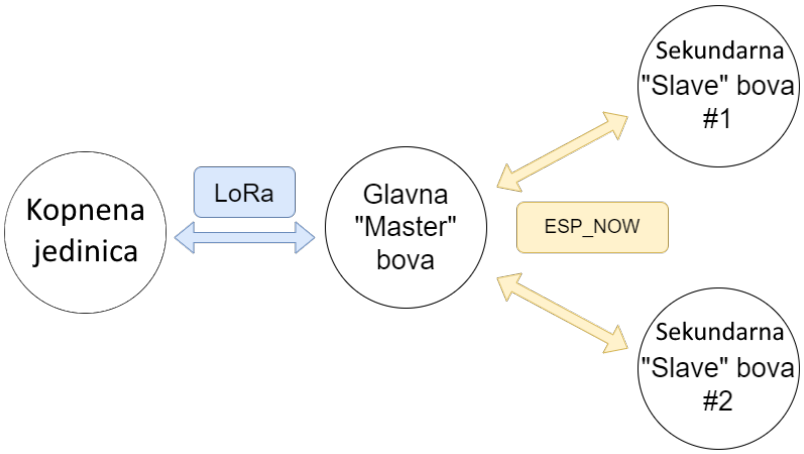
Ideja pristupa je da operater putem *LoRa* mreže komunicira s vodenim jedinicama, koje sekvencijalno dobivaju svoje udaljenosti od ronioca te javljaju očitanja kopnenoj jedinici. Ta se očitanja obrađuju algoritmom za procjenu lokacije ronioca.

Nakon primitka odgovora od vodenih jedinica, kopnena jedinica obrađuje podatke koji su izmjereni. Na zaslonu se prikazuje status vodenih jedinica i procijenjena lokacija ronioca. Sustav također omogućava određivanje lokacije više ronioca sekvencijalno, što se postiže slanjem različitih komandi za određene ronioce. Jedna kopnena jedinica može adresirati do 253 različita ronioca, radi ograničenja adresiranja same akustične jedinice koja se temelji na adresama veličine jednog okteta.

Iako vodene jedinice koriste iste senzore i obavljaju istu funkcionalnost, postoje različitosti pri samoj sekvenci rada. Radi takve arhitekture vodenih jedinica, određena je nomenklatura koja spaja njihovu ulogu i funkcionalnost u samom sustavu:

- Glavna ili *Master* jedinica,
- Sekundarna ili *Slave* jedinica.

Glavna ili *Master* jedinica je zadužena za izravno upravljanje ostatkom vodenih jedinica. Ova jedinica je zadužena da procesira zahtjev kopna na ostale sekundarne (engl. *Slave*) jedinice te da prikupi njihove izmjerene podatke. Način na koji glavna jedinica to radi je pomoću protokola temeljenog na WiFi mreži *ESP-NOW*. Glavna jedinica pri zahtjevu s kopna

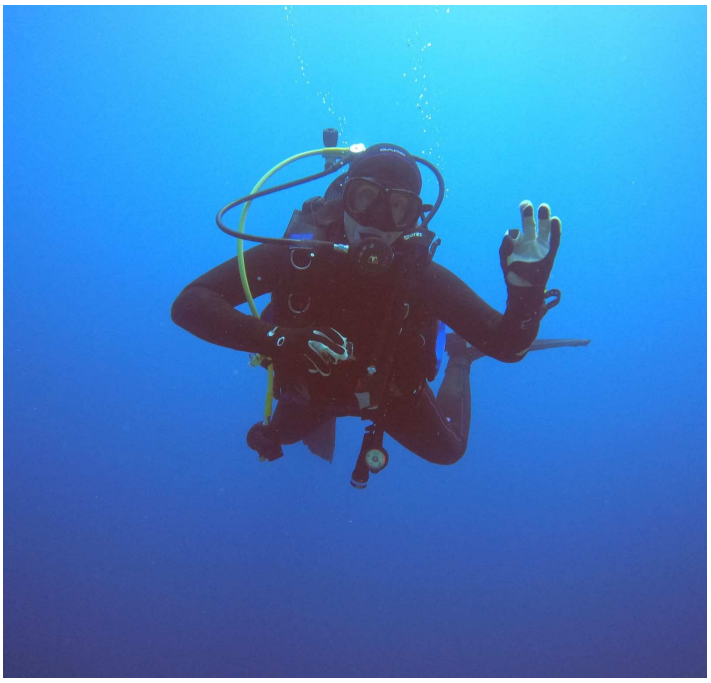


Slika 2.2: Blok shema vodenih jedinica

prvobitno određuje lokaciju pomoću GPS senzora te određuje udaljenost od ronioca pomoću akustičkog senzora. Potom se prosljeđuje zahtjev akustičnoj jedinici, koja pritom vraća status obrade signala koji je dobila od podvodne jedinice. Nakon obrade signala, glavna jedinica šalje sekvencijalno jedan po jedan zahtjev za prikupljanjem podataka o roniocima. Ovakav način omogućava da se ovisno o željenoj preciznosti proširi sustav na više od dvije sekundarne jedinice.

Sekundarna ili *Slave* jedinica je zaslužna za očitanje identičnih podataka kao i glavna jedinica, ali ova jedinica za razliku od glavne ne šalje podatke na kopno, nego natrag na glavnu jedinicu. Takav način omogućava veću modularnost i jednostavnu zamjenu vodenih jedinica pri potencijalnom kvaru ili radi drugih razloga. Jedinica također sadržava sve senzore koji se nalaze i na glavnoj jedinici, ali bez modula za radio komunikaciju s kopnom.

Slika 2.3: Probna podvodna jedinica



Slika 2.4: Primjer ronioca praćenog opisanim sustavom

Kako bi sustav s visokom sigurnošću mogao dobiti lokaciju ronioca i kako ne bi morao koristiti razne metode procjene temeljene na aproksimativnim pristupima, potrebne su minimalno četiri jedinice, točnije, jedna glavna i tri sekundarne. U trenutnom postavu nalaze se tri vodene jedinice, što je manje od četiri minimalno potrebne za točnu trilateraciju pozicije. Radi toga bilo je potrebno implementirati algoritam koji će uzeti određene nekonzistentnosti u informacijama da obavi dobru aproksimaciju lokacije ronioca.

3. Opis sklopovlja

Kako bi prototip sustava bio cjenovno pristupačan i kako bi ga se mogao kasnije lakše nadograđivati, početna verzija sustava napravljena je korištenjem eksperimentalnih pločica. S obzirom da se koriste moduli koji su montirani na razvojnim pločicama, olakšano je pronalaženje problema te je omogućena jednostavna zamjena modula, u odnosu na monolitno rješenje korištenjem vlastito razvijene tiskane pločice.

3.1. Baterijsko napajanje

Sustav uz module koji su potrebni za rad također sadrži i vanjsku bateriju s dva USB-A izlaza od 5 V/10 W. Baterija i glavni mikrokontroler spojeni su preko standardnog kabela USB-A na USB-C, što je odabrano radi lakšeg testiranja i ožičenja. Baterijski paket proizvod je tvrtke *Voltaic*, koja također uz baterijske pakete proizvodi i solarne pločice kompatibilne s njihovim baterijama. Prikaz njihove baterije se može vidjeti na slici 3.1. Solarna ploča korištena u ovom sustavu može puniti bateriju maksimalno s 10 W snage pri idealnim uvjetima, što je teško ostvarivo u većini praktičnih situacija. Ovaj sustav napajanja osigurava da sustav može raditi samostalno tijekom dugih vremenskih perioda radi jako niske razine snage koju sustav troši u radu. Sustav bi s ovakvim sustavom napajanja uz konstantnu potrošnju od 500 mW pri naponskoj razini od 5 V i 20000 mAh baterijskim paketom mogao raditi 200 sati, što je za potrebne funkcionalnosti dovoljno dug vremenski period, pogotovo ako se uzme u obzir vrijeme koliko jedan ronioc može biti maksimalno pod vodom.

Baterijski paket dolazi s većim brojem zaštita i mogućnosti:

- zaštita od kratkog spoja,
- zaštita od prebrzog pražnjenja,
- zaštita od prevelike temperature (maksimalna temperatura je 60°C),
- zaštita od preniske temperature (minimalna temperatura je -15°C).

Vrsta ćelija koju ovaj baterijski paket sadrži je Li-Ion, što je tipičan odabir u srodnim sustavima zbog dobrog omjera cijene i kvalitete. Baterijski paket također ima opciju stalno

upaljenog načina rada (engl. *Always On Mode*). Ova opcija omogućava da se baterijski paket nikad ne ugasi, čak ni ako je izlazna struja minimalna. Radi ove opcije, bilo koji ugradbeni sustav može bez poteškoća prijeći iz hibernacijskog načina rada bez mogućnosti da se baterijski paket trajno ugasi, što je nažalost veliki problem kod većine ostalih proizvođača.

Slika 3.1: Baterijski paket sa solarnom pločom

3.2. Razvojni sustav ESP32-WROOM-32D

ESP32-WROOM-32D je mikrokontroler razvijen od kompanije *Espressif*, a korištena inačica sadrži dvojezgreni procesor s integriranim Wi-Fi i Bluetooth modulima. Ovaj modul omogućuje razne mogućnosti povezivanja, uključujući potporu za 2,4 GHz Wi-Fi i Bluetooth 4.0. Dvojezgreni procesor omogućava obradu više zadataka istovremeno, što značajno povećava učinkovitost i performanse uređaja. Također, *ESP32-WROOM-32D* podržava različite periferije, uključujući ADC (engl. *Analog to Digital Converter*), DAC (engl. *Digital to Analog Converter*), I2C (engl. *Inter-Integrated Circuit*), SPI (engl. *Serial Perephial Interface*), PWM (engl. *Pulse Width Modulation*), CAN/TWAI (engl. *Controler Area Network*, *Two Wire Automotive Interface*) i druge, što ga čini izuzetno fleksibilnim za razne primjene.

Modul *ESP32-WROOM-32D* koristi se u vodenoj i kopnenoj jedinici sustava. Njegov fizički izgled i veličina je prikazana na slici 3.2. Modul dolazi i s ugrađenom antenom i naprednim funkcijama za upravljanje energijom, što ga čini idealnim za prijenosne i energetski učinkovite aplikacije. Sa svojim malim dimenzijama i integriranim komponentama, *ESP32-*

Slika 3.2: *ESP32-WROOM-32D* modul [2]

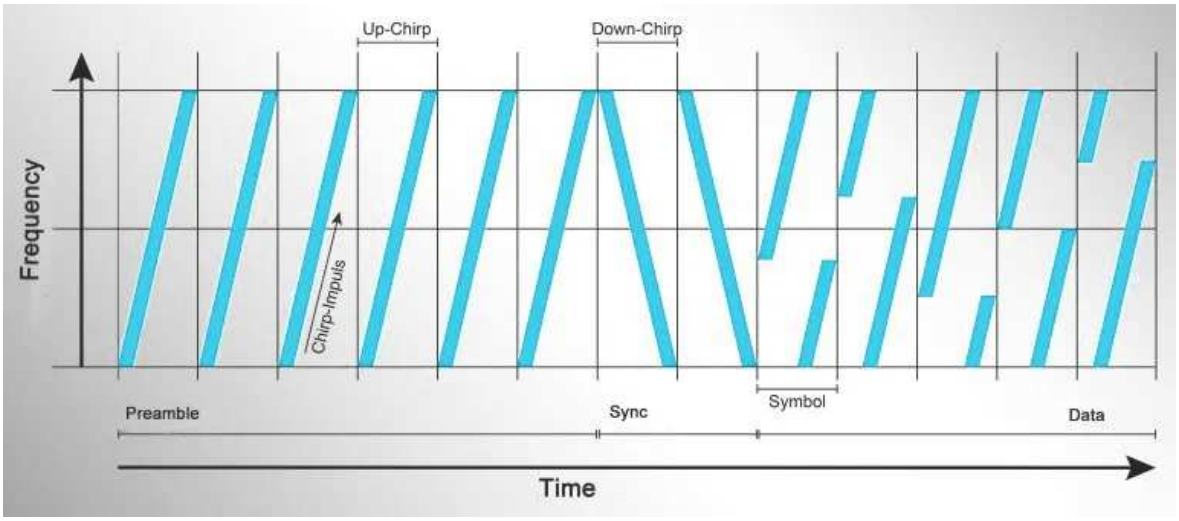
WROOM-32D omogućava jednostavnu integraciju u različite projekte i uređaje, uključujući pametne kućne uređaje, nosive tehnologije, automatizaciju industrije i brojne druge primjene.

3.2.1. Komunikacijski sustav LoRa

Problem često korištenih radiokomunikacijskih protokola u *IoT* sustavima je mogućnost rada na relativno malim udaljenostima i relativno velika potrošnja prilikom prijenosa signala. Primjerice, Wi-Fi ili Bluetooth odlično rade na malim udaljenostima i sveukupni kapacitet kanala, odnosno pojasna širina (engl. *bandwith*), je velik u odnosu na protokole kao što su *ZigBee*, *SigFox*, ali im je maksimalna udaljenost između uređaja bez gubitka informacije relativno ograničena. Također, radi visoke frekvencije rada (2,4 GHz i više), moduli za Bluetooth i Wi-Fi zahtijevaju relativno veliku snagu u razmjerima od 1000 mW u vremenskim intervalima, što povećava potrošnju i nije poželjno pri razvoju ugradbenih računalnih sustava niske potrošnje. Naravno, snaga se kod ovakvih modula može prilagoditi i smanjiti kao kod primjerice *BLE* (engl. *Bluetooth Low Energy*), ali to i dalje ne rješava probleme s velikom udaljenošću. Ovaj problem obično se rješava tako da se moduli aktivno koriste za prijenos signala u što kraćim intervalima.

Iz tog razloga, bilo je potrebno pronaći protokol koji bi mogao dvosmjerno slati informacije na velikim udaljenostima (do nekoliko kilometara) te da nema veliku potrošnju energije kao prethodno navedeni. Iz tog je razloga odabran komunikacijski sustav *LoRa* (engl. *Long Range*).

Pojam *LoRa* ne odnosi se primarno na komunikacijski protokol, već na način modulacije signala. *LoRa* je komunikacijski sustav koji radi pomoću digitalne modulacije signala, ili kako se ponekad može naći u literaturi, radi na primjeru modulacije signala inspiriran



Slika 3.3: CSS modulacija signala u *LoRa* komunikacijskom sustavu [11]

"cvrkutima" (engl. *Chirp Spread Spectrum*, tj. *CSS*). Da bi se *LoRa* mogla smatrati nekom vrstom protokola, potrebno je dodati jednu razinu apstrakcije na sam komunikacijski sustav. Danas postoji nekoliko primjera i metoda koji se koriste kao navedena razina apstrakcije nad *LoRa* sustavom, od kojih je najpoznatiji *LoRaWAN*.

LoRa, kao i svi ostali bežični sustavi, koristi radio valove kako bi se informacija prenijela kroz prostor. Postoji više načina kako interpretirati takav radio val. Postoje standardne modulacije kao što su frekvencijska modulacija (engl. *Frequency Modulation*, tj. *FM*) koja se koristi danas kod standardne radio uređaje u automobilima, radio stanicama, amaterskim radio organizacijama i slično. Uz *FM* radio, postoji i nešto rjeđe korištena amplitudna modulacija (engl. *Amplitude Modulation* tj. *AM*). Te dvije modulacije možemo svrstati pod kategoriju analogne modulacije signala. Digitalna modulacija signala je danas često korištena u modernim radio sustavima, kao što je i sam Wi-Fi.

CSS modulacija, korištena u *LoRa* komunikacijskom sustavu, je jedna od metoda digitalne modulacije koja koristi "cvrkute" kao način prijenosa kvantiziranih informacija (odnosno digitalnih informacija) pomoću radiovalova. *CSS*, za razliku od *FM* i *AM* modulacije, ne modulira cijeli signal koji se kasnije pri demodulaciji pretvara u informaciju, već informaciju kodira u skup simbola koji se moduliraju sinusnim signalom promjenjive frekvencije. Ovaj način modulacije inspiriran je primjerom raspoznavanja cvrkuta ptica: svaki cvrkut preko zvučnih valova (za razliku od radio valova kod *CSS* modulacije) povećava i spušta svoju frekvenciju i tako označava njegov početak i kraj. Na isti način se kod *CSS* modulacije određenim simbolima pridaje određena modulacija jednog cvrkuta. Specifično za *LoRa* komunikacijski sustav, postoji nekoliko vrsta simbola koji se šalju kako bi komunikacija osigurala redundanciju i otpornost na vanjske smetnje. Postoje simboli za sinkronizaciju, za početak prijenosa te simboli koji odgovaraju bitovima 0 i 1 kako bi se digitalna informa-

Slika 3.4: LoRa modul Samtech SX1276
[8]

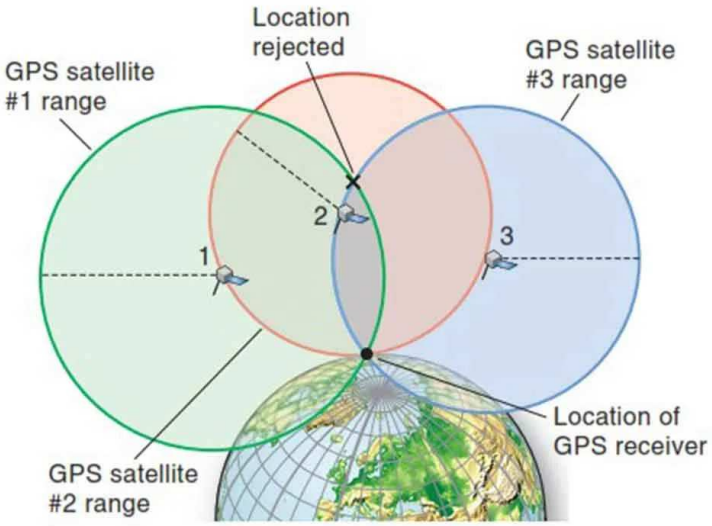
cija mogla kodirati. *LoRa* također uključuje nekoliko podesivih parametara koji utječu na količinu informacija i brzinu njihovog prijenosa kao što su faktor raširenosti cvrkuta (engl. *Spreading Factor*), propusnost (engl. *Bandwith*) i faktor kodiranja (engl. *Coding Factor*). Grafički prikaz promijene frekvencije u vremenu i načina sinkronizacije poruka prikazana je na slici 3.3. S obzirom da svaka geografska regija ima predodređenu dopuštenu frekvenciju, u ovom radu koriste se sljedeće postavke:

- frekvencija: 866 MHz,
- propusnost: 125 kHz,
- faktor kodiranja: 4/5,
- faktor raširenosti: 12.

Faktor kodiranja i faktor propusnosti odabrani su u skladu s najčešće korištenom konfiguracijom u praksi.

U okviru ovog rada koristi se *LoRa* modul SX1276 kompanije *Semtech*, koji je jedan od popularnijih *LoRa* modula u *IoT* projektima radi svoje niske tržišne cijene te odlične gotove javno dostupne programske potpore. Modul SX1276 s mikrokontrolerom komunicira pomoću *SPI* protokola. Fizički izgled modula vidljiv je na slici 3.4. Potrošnja modula pri stalnom slušanju za nove poruke je maksimalno 30 mA, a pri slanju podataka može porasti na 100 mA, što zadovoljava postavljena ograničenja u smislu trajanja baterijskog paketa. U dokumentaciji je navedena također maksimalna teorijska udaljenost između dva *LoRa* modula od 2 km, što je također u skladu s potrebama ovog rada.

U odnosu na protokole Wi-Fi i Bluetooth, *LoRa* komunikacijski sustav ima puno manju propusnost od 22 kbps (ne računajući i faktor kodiranja) u odnosu na propusnost od 600 Mbps kod 2,4 GHz Wi-Fi komunikacije. Ipak, za potrebe ovog rada propusnost komu-



Slika 3.5: Grafički prikaz trilateracije pozicije uz pomoć GPS sustava [10]

nikacijskog sustava *LoRa* je zadovoljavajuća s obzirom da se razmjenjuju samo jednostavne poruke.

3.2.2. GPS sustav za lokalizaciju

Kako bi se mogla odrediti lokacija ronioca, uz same udaljenosti od jedinica, potrebno je odrediti fiksnu lokaciju vodenih jedinica kako bi se relativna očitanja mogla pretvoriti u apsolutnu. U tu svrhu koristi se GPS.

GPS, prethodno poznatiji kao *NavStar*, je tehnologija koja služi za apsolutnu lokalizaciju objekta na Zemlji. Tehnologija je razvijena 1995. godine za potrebe američke vojske. Danas je vrlo raširena u području navigacije, kartografije, pri spašavanju izgubljenih ili ozljeđenih osoba i slično. Zbog porasta popularnosti autonomne vožnje počeo se koristiti kao dodatni senzor za razne lokalizacijske algoritme.

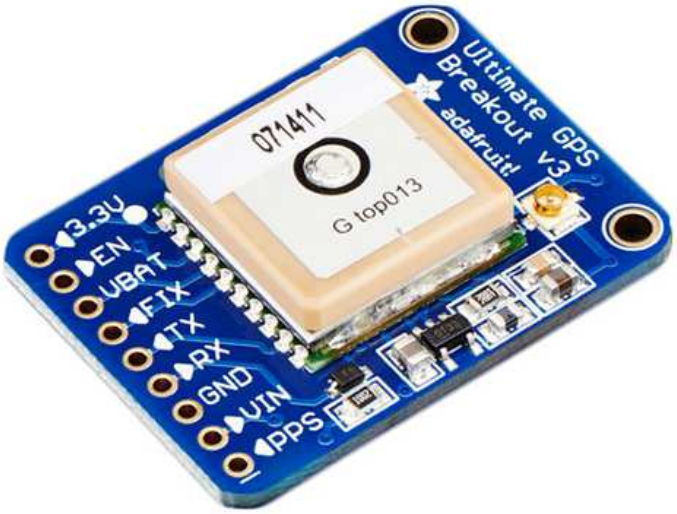
GPS radi na principu trilateracije, čija je vizualna prezentacija prikazana na slici 3.5. Trilateracija je način određivanja lokacije predmeta pomoću parametara udaljenosti tri ili više apsolutno pozicioniranih objekata u prostoru (primjerice GPS sateliti) i nekog predmeta (primjerice lokatora). Za proces trilateracije potrebna su minimalno tri satelita kako bi se mogla odrediti minimalna aproksimacija lokacije na Zemlji. Ako se koriste samo 3 satelita, razlike koje će se pojaviti u nadmorskoj visini mogu biti velike. Upravo iz tog razloga je standard koristiti minimalno četiri satelita kako bi sustav mogao odrediti lokaciju i procjenu nadmorske visine zemaljskog prijemnika. Kako bi sustav mogao koristiti trilateraciju, u orbiti Zemlje danas se nalazi više od 30 satelita koji se aktivno kreću na visini od 20 000 kilometara. Također da bi sustav mogao odrediti lokaciju bilo gdje na Zemlji, potrebno je više satelita zbog svojeg aktivnog kretanja u orbiti. Uz to, kako bi GPS sustav postigao 95%

pokrivenost Zemljine površine, potrebno je u orbiti imati najmanje 24 satelita, dok ostatak satelita služi kao aktivna redundancija. Svaki satelit odašilje ostalim satelitima svoje vlastite podatke. Unutar vlastitih podataka jednog satelita nalaze se:

- pseudoslučajni broj,
- trenutna lokacija i ruta,
- telemetrijski podatci.

Pseudoslučajni se broj koristi kako bi se svaki satelit mogao identificirati te kako bi prijamnik signala mogao raspoznati satelite jedan od drugog. Telemetrijski podatci uključuju trenutno vrijeme, datum, status satelita te lokaciju s koje su uzeti ti podatci. Za razliku od trenutne lokacije i rute, ti podatci se ažuriraju svakih 30 sekundi i služe kao redundantna provjera pri validaciji statusa satelita. Iz tih podataka može se očitati je li satelit u funkcionalnom stanju. Ako nije, može se izbaciti iz razmatranja. Trenutna lokacija i ruta su najvažniji podatci koji se dobivaju od pojedinog satelita. Osim podataka o lokaciji i svojoj ruti satelit javlja i informacije o ostalim satelitima u orbiti tako da prijamnik može lakše dobiti informaciju o sljedećoj poruci novog satelita očekivati. Na taj način može se prije uspostaviti početna referentna točka prijamnika odnosno fiksirati referentnu točku (engl. *GPS fix*).

Nažalost, *GPS fix* se relativno sporo izvodi samo pomoću podataka o trenutačnoj lokaciji i ruti. Ponekad se iz vlastitih mjerenja nad korištenim modulom GPS moglo primijetiti da je vrijeme potrebno da bi se odredila početna fiksna lokacija bilo u vremenskom razmaku između jedne do pet minuta, što nije idealno za brzu inicijalizaciju sustava. Postoje metode koje ubrzavaju samo traženje početne fiksne lokacije, a jedna od njih je tehnologija *A-GPS* (engl. *Assisted Global Positioning System*). *A-GPS* koristi mobilnu mrežu odnosno mobilne podatke kako bi se GPS modul izravno spojio sa specijaliziranim poslužiteljem, koji lokalno drži spremljene podatke o lokaciji i statusu satelita. Pomoću ove tehnologije fiksiranje referentne točke ne ovisi u cijelosti o podacima o trenutačnoj poziciji i statusu od samih satelita, nego uključuje i zemaljsku infrastrukturu i pričuvne servere. Nažalost, kako ovaj sustav nema nikakvu mogućnost povezivanja s poslužiteljima (zbog manjka stabilne Wi-Fi ili mobilne mreže) ne može se koristiti ova metoda. Metoda koja se koristi u radu nije učinkovita kao *A-GPS* tehnologija, ali i dalje ubrzava postupak početne lokalizacije. Korišteni GPS modul dolazi s pričuvnom baterijom, koja osigurava održavanje jednom uspostavljene početne lokacije, čak i nakon gašenja uređaja. Pričuvna baterija u GPS modulu omogućava pohranu kritičnih podataka poput onih tipa *ephemeris* i *almanac* (točnije, podataka o orbiti i statusu satelita) te zadnje poznate lokacije i vremena. Ti podaci omogućuju modulu brzo pronalaženje satelita pri ponovnom pokretanju značajno skraćujući vrijeme potrebno za dobivanje fiksne lokacije. Bez pričuvne baterije modul mora ponovo preuzeti podatke, što može potrajati nekoliko minuta. Time se pričuvna baterija pokazuje ključnom za brži i učinkovitiji rad



Slika 3.6: GPS razvojni sustav *Ultimate GPS V3* kompanije *Adafruit* [1]

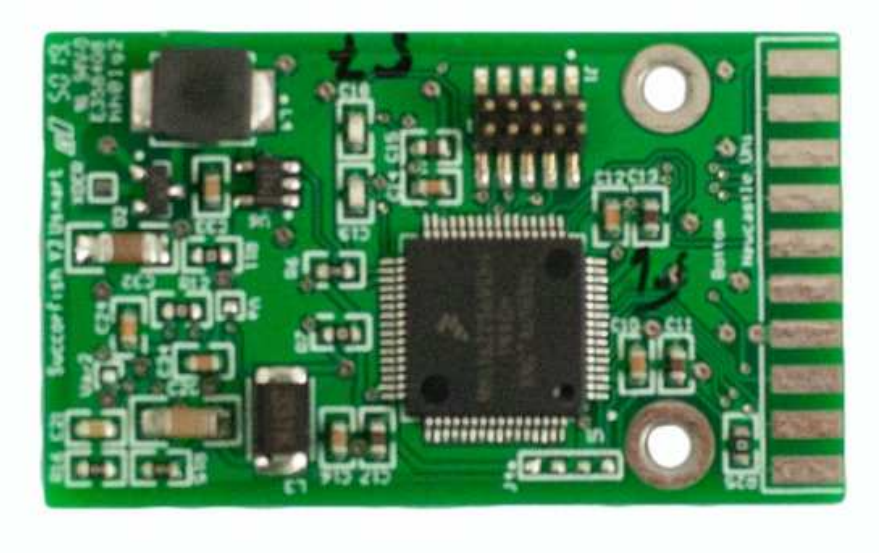
GPS modula.

U ovom sustavu koristi se razvojni sustav s GPS modulom kompanije *Adafruit* pod nazivom *Ultimate GPS V3*. Njegov fizički izgled je prikazan na slici 3.6. Razvojni sustav je napravljen kako bi se lako koristio u procesu razvoja ugradbenih sustava te kako bi se na jednostavan način ostvarila potpora za lokalizaciju.

Razvojni sustav temelji se na GPS modulu PA1616s kompanije *CDtop Technology*. Dolazi s ugrađenom keramičkom antenom malih dimenzija, što je idealno za primjene u kojima je važna kompaktnost sustava. Modul koristi naponsku razinu od 3,3 V. To je primjereno za izrađeni sustav zbog identične naponske razine modul ESP32, koji se koristi u svim jedinicama. Također, kako bi se modul mogao integrirati u sustave koji rade s naponskom razinom od 5 V, na razvojnoj je pločici ugrađen naponski pretvornik koji pretvara ulaz razine 5 V na 3,3 V. Maksimalna frekvencija osvježavanja statusa signala i pozicije koja je podržana na modulu je 10 Hz, što je relativno malo, ali je testiranje pokazalo da ovaj sustav ne treba GPS frekvenciju veću od 2 Hz. Prema dokumentaciji najveća moguća preciznost modula je 1,8 m. Za izradu funkcionalnog prototipa je takva nepreciznost prihvatljiva, ali u sljedećim iteracijama sustava potrebno je naći adekvatniji modul. Također, razvojna pločica dolazi s dodatnim konektorom tipa *Hirose U.FL*, kojim se spaja vanjska antena i dobiva bolja povezanost sa satelitima.

3.2.3. Delphis V3 i Nordic modul

Kako bi se odredila pozicija ronioca u odnosu na vodene jedinice, korišteno je rješenje izrađeno na sveučilištu Newcastleu [4]. Sveučilište je u suradnji sa kompanijom *Succorfish* je



Slika 3.7: Delphis V3 modul kompanije Succorfish [7]

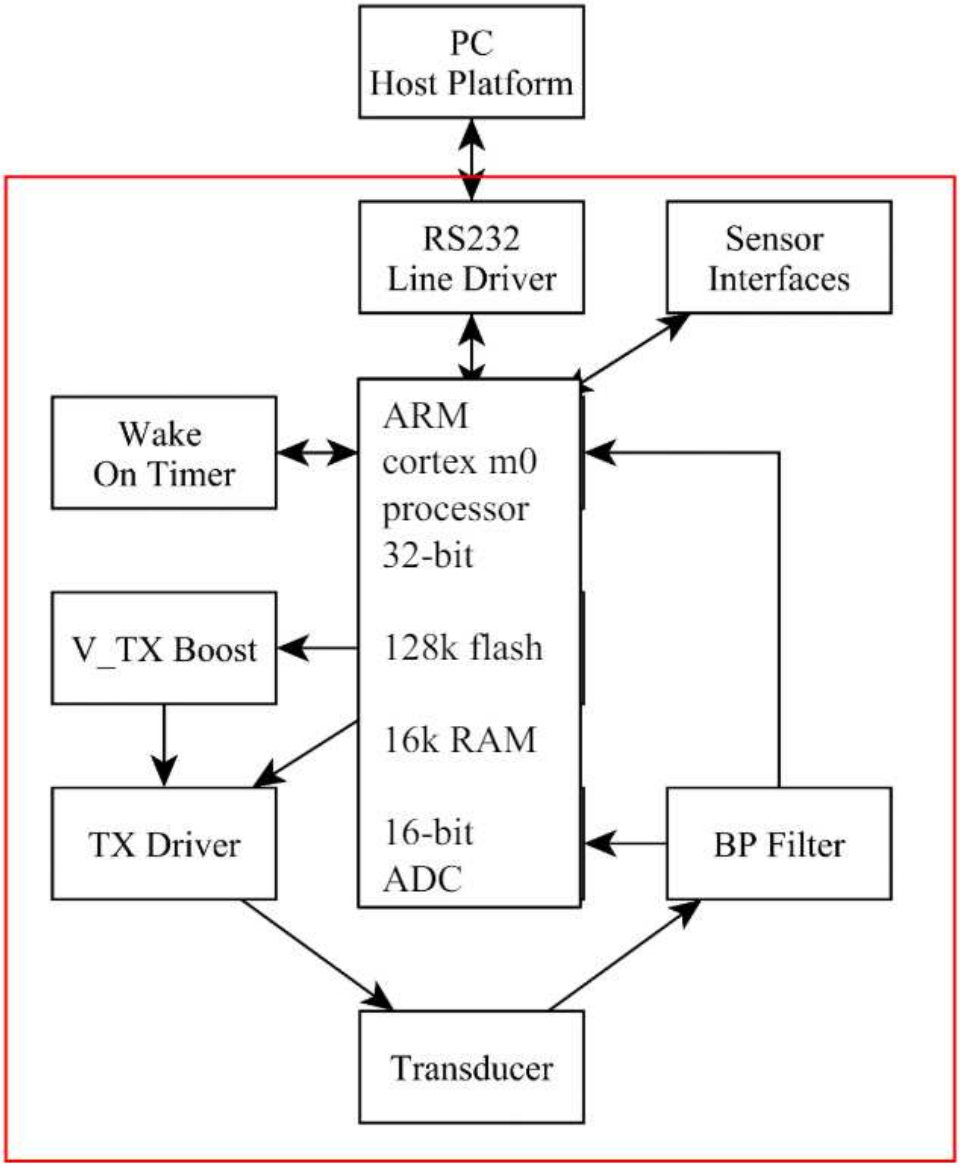
izradilo cijenom pristupačno rješenje, koje postiže odlične performanse pri komunikaciji i lokalizaciji ronioca. Izrađeni su proizvod nazvali Delphis V3. Delphis V3 je uređaj koji može ostvariti podvodnu komunikaciju pomoću ultrazvučnih valova te se se koristi u mnogim područjima industrije, kao što je i sama podvodna autonomna vožnja, profesionalno ronjenje te istraživačko pretraživanje vodenog tla. Uređaj se sastoji od dvije glavne komponente:

- *nanomodem*,
- ultrazvučni pretvornik.

Nanomodem je dodatni uređaj koji služi za napajanje ultrazvučnog pretvornika te za njegovo upravljanje i očitavanje podatka. Fizički izgled *nanomodema* vidljiv je na slici 3.7. *Nanomodem* temelji se na mikrokontroleru ARM arhitekture. Arhitektura je prikazana na slici 3.8. Korišten je mikrokontroler iz serije ARM-Cortex-M0, koja se često koristi u današnjim mikrokontrolerima zbog svoje male potrošnje i dovoljno brzog procesiranja informacija za razne ugradbene sustave. Konkretan model mikrokontrolera modula *nanomodem* dolazi s 128 kB *flash* memorije te 16 kB RAM memorije. *Nanomodem* zahtijeva napajanje od 3 V do 6,5 V, što ga čini idealnim za napajanje preko baterijskog paketa. Prema dokumentaciji proizvođača pri dovođenju napajanja razine 5 V potrošnja je struje sljedeća:

- pri osluškivanju za nove poruke - 2 mA,
- pri primanju poruke - 4 mA,
- pri slanju poruke - 250 mA.

Iz ovih podataka može se vidjeti da je potrošnja struje u stanju mirovanja minimalna, dok u stanju slanja nove poruke razina potrošnje struje može porasti 10 puta. Takvo je ponašanje u sustavu prihvatljivo zbog jako malog broja zahtjeva za slanjem poruka. Od vanjskih



Slika 3.8: Arhitektura *nanomodema Delphis V3* [4]

Slika 3.9: Ultrazvučni pretvornik s dodatnim kućištem kompanije *H2O-Robotics*

periferija *nanomodem* uključuje dodatno sklopovlje, koje je nužno kako bi sustav mogao samostalno raditi. Jedan od najvažnijih elemenata dodatnog sklopovlja je pojasno propusni filtar (engl. *Band Pass filter*), koji služi kako bi se filtrirale frekvencije koje odgovaraju rasponu spojenog ultrazvučnog pretvornika. Frekvencijski raspon pojasno propusnog filtra projektiran je da propušta frekvencije između 24 i 32 kHz. Još jedan od elementa sklopovlja koji su dodani je *RS-232* serijsko sučelje, kojim se mogu razmjenjivati željene informacije između sustava i *nanomodema*. Uz serijsko sučelje implementirana su *SPI* i *I2C* sučelja kako bi se što više povećala kompatibilnost sustava s ostalim porodicama mikrokontrolera. *Nanomodem* može adresirati maksimalno 256 jedinica zbog ograničenja u veličini adrese od jednog bajta. U izrađenom sustavu to ne stvara potencijalni problem zbog malog broja modula koji se koriste. Maksimalna dubina za koju je testiran ovaj modul je do 350 metara, što je dovoljno za potrebe sustava. Maksimalna veličina podataka koja se može poslati u trenutku pomoću modula je 64 bajta, što je idealno za slanje manjih telemetrijskih podataka kao što su izmjereni tlak, status ronioca, njegov srčani ritam i slične informacije.

Ultrazvučni pretvornik također je važna komponenta modula. Njegov izgled se može vidjeti na slici 3.9. Princip rada ultrazvučnog pretvornika oslanja se na mogućnosti propagacije zvuka određenih frekvencija kroz vodu, to jest, oslanja se na propagaciju ultrazvučnih valova. Struktura zvučnog vala stvara se na *nanomodemu* koji potom pretvara digitalni signal u analognu reprezentaciju, koja se na kraju odašilje. Impulsi koji se generiraju su longitudi-

nalno kratki te snažni, što je i glavni razlog zašto jedinica može poslati podatke na velikim udaljenostima. Po dokumentaciji izmjerena maksimalna udaljenost na kojoj je signal i dalje bio uspješno primljen je 2 km, a u čistoj vodi čak i 3,5 km.

Jedna od bitnih značajki zbog koje se upotrebljava ovaj sustav je njegova mogućnost slanja testnih *ping* signala za izračun udaljenost između 2 modula. U slučaju izrađenog sustava to je dosta važno jer se time može izmjeriti udaljenost vodene jedinice od ronioca, što je veliki faktor pri računanju procjene njegove lokacije. Prema dokumentaciji varijanca izmjerenih udaljenosti korištenjem ove metode aproksimirana je na 10 cm, što je za ovaj sustav dovoljno precizno. *Nanomodem* pri slanju *ping* naredbe dobiva povratni odgovor, koji pokazuje broj izmjerenih taktova koji su prošli u vremenskom intervalu između slanja i primanja odgovora. Nakon toga ako se izmjereni taktovi žele pretvoriti u udaljenost u metrima, koristi se jednadžba ispod odlomka.

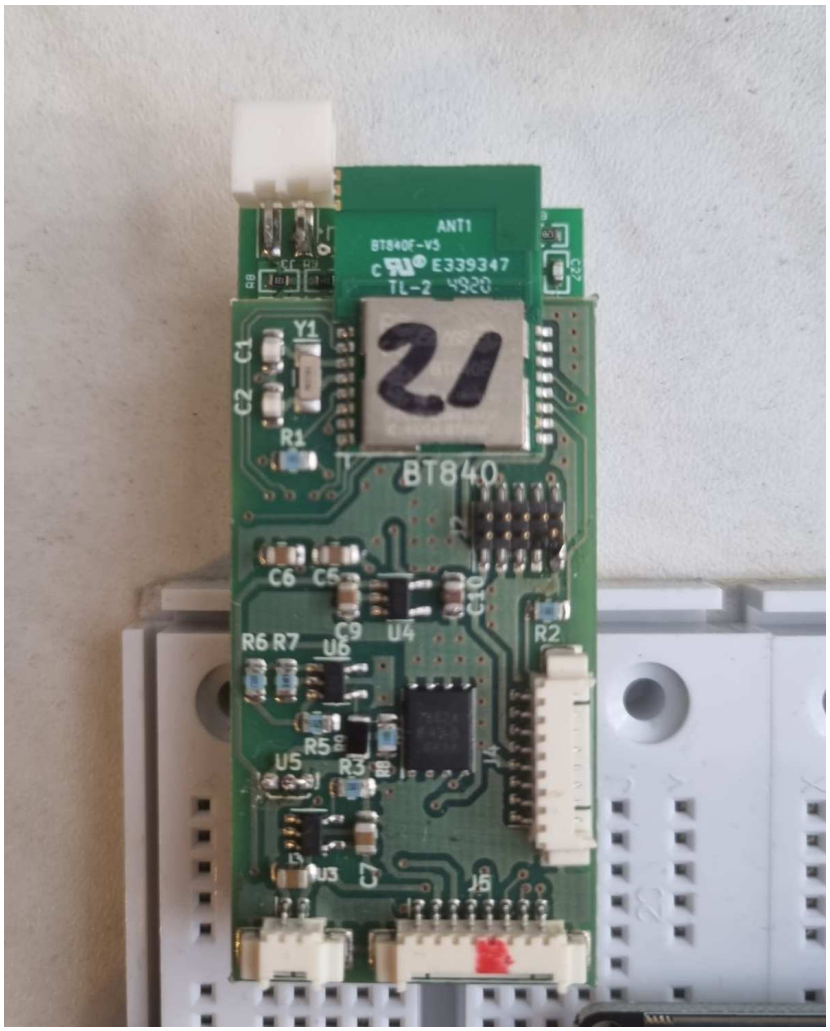
$$d = y \cdot c \cdot 3.125 \cdot 10^{-5}$$

pri čemu je:

- d - udaljenost između modula (m),
- y - broj proteklih taktova izmjerenih između vremena slanja i odgovora (*ticks*),
- c - brzina zvuka (uzimamo konstantu od 1500 m/s).

Konstanta na kraju izraza jednadžbe predstavlja dvostruki broj taktova u sekundi *nanomodema*, zato ga se koristi kako bi se pretvorio broj taktova izmjerenih pri putovanju signala u oba smjera u potrebne sekunde u jednom smjeru.

Kako je sustav Delphis V3 korišten u mnogo pojekata unutar *LABUST*-a (engl. *Laboratory for Underwater Systems and Technologies*) i kompanije *H2O-Robotics*, dodan je modul između Delphis v3 sustava te ostatka sustava. Dodatni je modul temeljen na mikrokontroleru kompanije *Nordic* pod nazivom *nRF52832*. Izgled modula može se vidjeti na slici 3.10. Mikrokontroler je također temeljen na ARM arhitekturi, ali na drugačijoj porodici. Glavna procesorska jedinica je ARM Cortex - M4. Ona sadrži 32-bitne arhitekture te radi na frekvenciji od 64 MHz. Od periferije dolazi s raznim integriranim funkcionalnostima kao što su 12-bitni *ADC*, 32 generička izvoda tipa *GPIO* (engl. *General-Purpose Input/Output*), 3 *SPI* sučelja, serijsko *RS-232* sučelje i još mnogi. Mikrokontroler dolazi također i s Bluetooth LE (engl. *Low Energy*) koji je odličan za razne *IoT* upotrebe. Dodatno je implementiran vlastiti protokol, koji je relativno jednostavan za korištenje te prikazuje više informacija od samostalne upotrebe *nanomodema*. Dodatnim *Nordic* modulom upravlja se serijskim sučeljem *RS-232*. Također je podržana i komunikacija bežičnim serijskim protokolom temeljenog na



Slika 3.10: Posredni modul temeljen na *Nordic* mikrokontroleru

Bluetooth LE modulu. Radi jednostavnosti svaka vodena jedinica ovog sustava koristi žičano povezano *RS-232* serijsko sučelje.

3.3. Raspored modula jedinica

Kao što je i prije bilo objašnjeno, sustav sadržava tri glavne jedinice u kojima svaka ima svoju ulogu:

- kopnena jedinica,
- skup vodenih jedinica,
- podvodne jedinice.

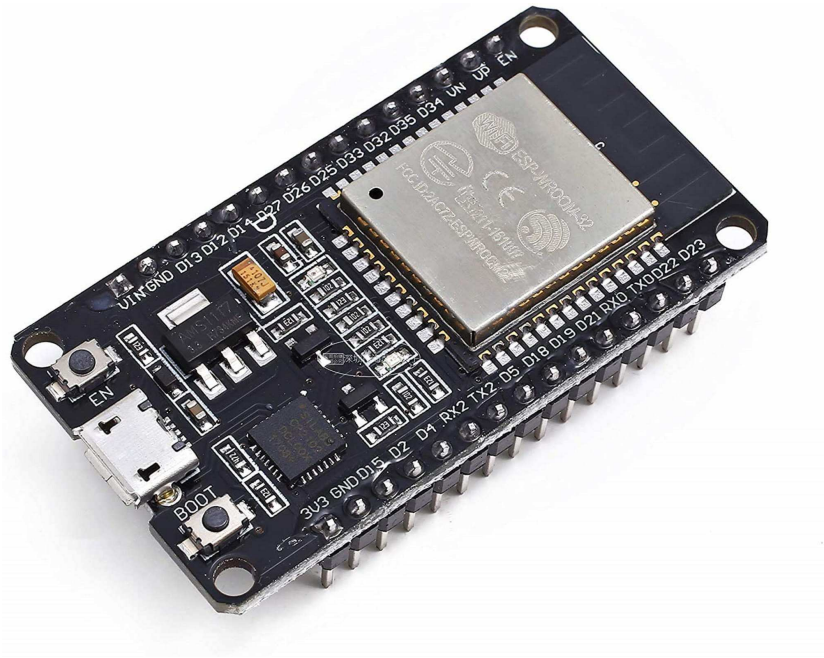
3.3.1. Kopnena jedinica

Kopnena jedinica predstavlja najjednostavniji sustav. On se sastoji od tri komponente:

- mikrokontrolera ESP32 s *LoRa* modulom,
- adaptera USB na *RS-232*,
- osobnog računala za dodatnu vizualizaciju i procesiranje.

Odabranu razvojnu pločicu za ovu jedinicu proizvodi hrvatska kompanija *Soldered* (prije poznata kao *E-radionica*) te se temelji na razvojnoj pločici kompanije *Espressif*. Izgled i veličina razvojne pločice je vidljiva na slici 3.12. Nažalost, kako se pločica više ne proizvodi, količina informacija koja se može pronaći o njoj jako je ograničena. Kako bi razvojna pločica ukomponirala što je više dostupnih značajki, dodan je modul *SX1276 LoRa* zajedno s helikoidalnom antenom malih dimenzija. Moguće je također konektorom *U.FL* spojiti vanjsku antenu koja može povećati domet i snaga signala mreže *LoRa*. Na prototipnoj pločici korišten je modul *ESP32-Wroom-32D*. Uz *LoRa* modul i ESP32 jezgru razvojna pločica sadrži i odvojeno sklopovlje za punjenje Li-Po baterije, što smanjuje cijenu pri razvoju *IoT* rješenja. Razvojna pločica dolazi također i s regulatorom napajanja s 5 V na 3,3 V. Uz to dolazi i s konektorom tipa *USB-microB*, koji izravno može napajati bateriju spojenu na dodatno sklopovlje za punjenje. Razvojna pločica ne dolazi sa ugrađenim pretvornikom serijskog protokola *RS-232* na virtualni serijski uređaj USB. Iz tog razloga je potrebno koristiti vanjski pretvornik za programiranje modula ESP32. Za potrebe rada korišten je pretvornik kompanije *E-radionica* pod nazivom *NOVA programer*, koji se izravno spaja na predviđene ulaze razvojne pločice.

Nažalost, zbog navedenih nedostataka i nedostupnosti detaljnije tehničke dokumentacije raspoložive razvojne pločice, ona nije korištena kod vodenih jedinica. Bila je samo primje-

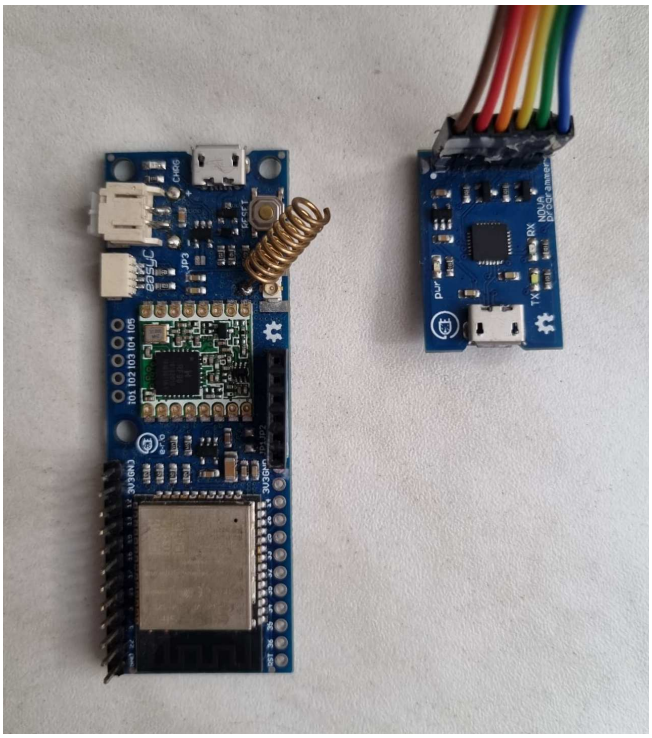


Slika 3.11: Razvojna pločica *ESP32 DevKit V1* kompanije *DOIT*

njiva kod kopnene jedinice zbog nemogućnosti nabave nove i pouzdanije razvojne pločice prilikom izrade rada. Zbog toga druge jedinice koriste razvojnu pločicu *ESP32 DevKit V1* kompanije *DOIT*, čiji je izgled vidljiv na slici 3.11. Novokorištena razvojna pločica ne dolazi s ugrađenim *LoRa* modulom i zbog toga je bilo potrebno koristiti dodatnu razvojnu pločicu *RFM95W* kompanije *Adafruit*.

Neki od problema koji su pri radu s razvojnom pločicom kompanije *E-radionica* su sljedeći:

- ESP32 modul postavljen tako da se njegova 2,4 GHz antena nalazi odmah iznad PCB-a (engl. *Printed Circuit Board*) razvojnog sustava, što jako utječe na kvalitetu komunikacijskog signala i domet,
- regulator napajanja napravljene tako da često izaziva nagle padove napona na ESP32 modulu i ostaloj spojenoj elektronici (engl. *brownout error*) što može potencijalno trajno oštetiti dodatno povezano sklopovlje,
- na pločici ne postoji nikakav indikator uključenja, što predstavlja veliki problem pri korištenju i otkrivanju problema,
- razvojna pločica nema sve moguće izlaze spojene na vanjske izvode (engl. *header pins*), što znači da je potrebno dodatno lemljenje u slučaju korištenja određenog izvoda,
- kako razvojna pločica više nije dostupa u prodaji, dokumentacija za nju ne postoji i nikad nije napravljena pa se samo istraživanje problema svodi na eksperimentalne metode.



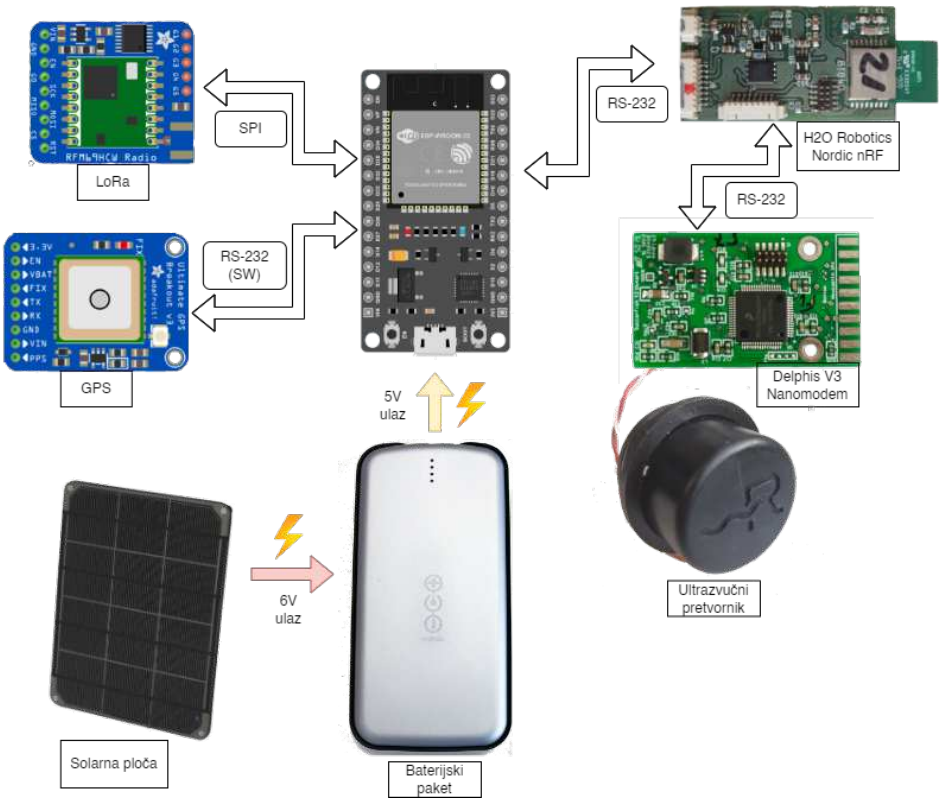
Slika 3.12: Razvojna pločica ESP32 kompanije *E-radionica* (lijevo) zajedno s *NOVA* pretvornikom *USB* (desno)

Iz tih razloga ova se prototipna pločica koristi jedino za kopnenu jedinicu zbog već ugrađene *LoRa* funkcionalnosti.

Kopnena je jedinica serijskim USB pretvornikom spojena na računalo, koje može slati daljnje naredbe vodenim jedinicama u obliku točno određenih znakovnih nizova.

3.3.2. Vodena jedinica

Glavna jezgra vodene jedinice je mikrokontroler koji je kao i kopnena jedinica temeljena na platformi ESP32. Također koristi *ESP32-WROOM-32D* modul zbog svoje široke upotrebe i dobrih performansi. Za razliku od kopnene jedinice kod nje se ne koristi razvojna pločica kompanije *E-radionica*, već razvojna pločica kompanije *DOIT*. Ova razvojna pločica za razliku od kopnene sadržava određene dodatne elemente koji olakšavaju razvoj i omogućuju samostalan rad sustava. Jedna od važnijih promjena je bolje projektirano napajanje te korištenje kvalitetnijeg regulatora napajanja. Pri testiranju ovih pločica pokazalo se da sustav stabilnije radi bez naglih padova napona, što je u vidu stabilnosti rada vrlo važan faktor. Nadalje, ova razvojna pločica dolazi s konektorom *USB-C* umjesto zastarjele *USB-microB* varijante što pojednostavljuje razvoj. Također, dolazi i s pretvornikom *USB* na serijsko sučelje *RS-232* tako da se ne mora koristiti vanjsko kao što je kod kopnene jedinice.



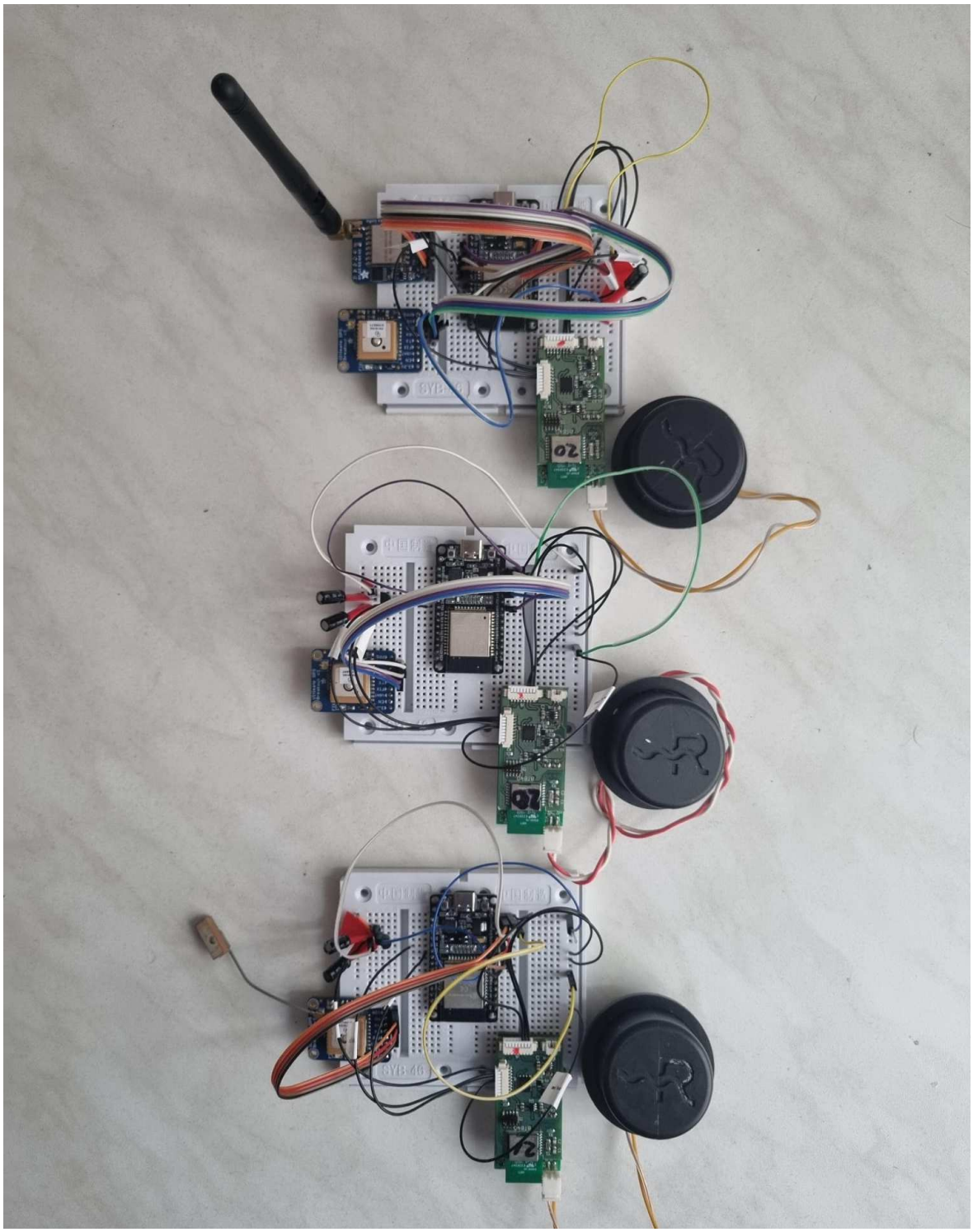
Slika 3.13: Blokovska shema spajanja jedne vodene jedinice (LoRa modul uključen samo na glavnoj vodenoj jedinici)

Osim nove razvojne pločice uključeni su i ostali moduli kako bi se ostvarila cjelovita funkcionalnost sustava. Iz tog je razloga na vodenu jedinicu dodan GPS modul *Ultimate GPS V3* kompanije *Adafruit*. GPS razvojna pločica koja se spaja na mikrokontroler ima već ugrađen regulator 5 V na 3,3 V naponsku razinu, tako da se ne mora opterećivati regulator mikrokontrolera. Podatkovna veza između GPS modula i mikrokontrolera ostvarena je serijskom komunikacijom *RS-232*, pri čemu je korišteno programski impelentirano serijsko sučelje (engl. *software serial*) jer *ESP32 DevKit V1* ne dolazi s dovoljnim brojem fizičkih modula *RS-232*. Također je dodan i sustav *Delphis V3* sastavljen od *nanomodema* i ultrazvučnog pretvornika. Napajanje za *nanomodem* se izravno spaja na izlaz od 5 V iz baterijskog paketa, tako da kao i GPS modul smanje opterećenje regulatora napajanja razvoje pločice. Iz istog razloga *Nordic* pločica za napajanje koristi vanjski regulator od 3,3 V, koji je sastavljen na testni postav. Preopterećenje napajanja može utjecati na rad cijelog sustava u trenucima kada se pokuša koristiti više zasebnih električnih modula. Na primjer kada se koristi *ESP-NOW* komunikacija paralelno sa serijskom. Za te potrebe, korišten je regulator *MCP1700-3302E* koji može pretvarati s naponske razine od 5 V na 3,3 V uz maksimalnu potrošnju od 250 mA, što je primjereno za uvijete rada.

Kako razvojna pločica *ESP32 DevKit V1* kompanije *DOIT* ne sadrži *LoRa* prijemnik,

bilo je potrebno naći alternativno rješenje. Kako bi se maksimalno olakšao razvoj, iskorištena je razvojna pločica kompanije *Adafruit RFM95W*. *LoRa* modul se preko *SPI* sabirnice spaja na glavni mikrokontroler. Ova razvojna pločica sadržava isti *LoRa* modul *SX127x* kao kod kopnene jedinice, ali s dodatnim regulatorom napajanja i pretvornikom logičke razine s 5 V na 3,3 V. Na razvojnu pločicu dodatno je zalemljen *SMA* konektor tipa *SMA* na koji se spaja antena. Ovaj se modul koristi samo na glavnoj vodenoj jedinici jer ona uspostavlja komunikaciju s kopnenom. Zbog jednostavnosti i cijene modul nije uključen u ostale dvije sekundarne jedinice.

Na posljetku je svakoj jedinici dodan i baterijski paket kompanije *Voltaic*. On se izravno spaja preko izlaza tipa *USB-A* na ulaz tipa *USB-C* na razvojnoj pločici, što osigurava manji broj spojeva i bolje raspoređene komponente.



Slika 3.14: Vodene jedinice: prva gledana odozgo je glavna, ostale dvije su sporedne

4. Programska potpora

Cijeli sustav podijeljen je u tri glavna dijela:

- kopneni sustav,
- vodeni sustav,
- podvodni sustav.

Vodeni sustav fokusira se primarno na prikupljanje podataka, dok je kopneni zadužen za izračun i vizualizaciju lokacije ronioca na temelju podataka primljenih od vodenog sustava. U tu svrhu razvijena je dodatna programska potpora za ta dva sustava u okviru ovog rada. S druge strane, podvodni sustav koristi ugrađenu funkcionalnost modula *Nordic* za slanjem odgovora na poslani *ping* zahtjev i ne sadrži razvijenu dodatnu programsku potporu jer korišteni modul kompanije *H2O-Robotics* sadrži svu potrebnu funkcionalnost, pa se podvodni sustav u ovom poglavlju neće detaljnije opisivati.

4.1. Vodene jedinice i kopnena jedinica

Za razvoj programske potpore za ugradbene sustave postoji više različitih arhitektura i razvojnih okolina. Određene razvojne okoline namijenjene su za specifične uloge (primjerice, razvojna okolina s alatima specijaliziranima za analizu zvuka, za pametne kuće i sl.). Takve razvojne okoline služe kako bi se programer više fokusirao na razvoj programske potpore, a manje na implementaciju već ranije implementiranih funkcionalnosti. Kako je ovaj sustav prototip i kako je fokus bio na dokazivanju moguće funkcionalnosti (engl. *proof of concept*), korištena je razvojna okolina *Arduino*, koja drastično pojednostavljuje i ubrzava razvoj prototipnih sustava bez potrebe za poznavanjem detalja niske razine kod ugradbenih računalnih sustava.

4.1.1. PlatformIO i Arduino razvojna okolina

Arduino kao razvojna okolina je jedna od najraširenijih razvojnih okolina za razvoj programske potpore za ugradbene sustave. *Arduino* je dizajniran da bude pristupačan i jednostavan za korištenje, čak i za amatere bez prethodnog iskustva u elektronici ili programiranju.

Razvojna okolina obuhvaća vlastita elektronička rješenja i programsku potporu, pri čemu elektronička rješenja uključuju razne razvojne pločice početno temeljenje na mikrokontrolerima kompanije *Atmel AVR* i kasnije na mikrokontrolerima kompanija kao što su *Espressif*, *Nordic*, *STMicroelectronics* i druge, dok softver uključuje razvojno okruženje i programske biblioteke, koje olakšavaju programiranje. Razvojna okolina *Arduino* raširena je do te mjere da su ga i mnoge druge kompanije počele integrirati kao jednu od mogućnosti pri razvoju vlastitih programa.

Arduino je vrlo korištena i najpopularnija platforma za razvoj ugradbenih računalnih sustava iz nekoliko ključnih razloga. Prvo, njezina jednostavnost i pristupačnost omogućuju početnicima brzu izradu prototipnih rješenja bez potrebe za dubokim tehničkim znanjem o samoj implementaciji određenog dijela protokola, izvedbe prekida i sl. Takav je razvoj idealan za početnike jer ne zahtijeva poznavanje velike količine teorije, ali može dovesti do manjka razumijevanja važnih detalja niske razine. Drugo, velika i aktivna zajednica korisnika pruža mnoštvo resursa uključujući razne pokazne videe, forume i dijeljene projekte, što značajno pojednostavljuje učenje i rješavanje problema. Konačno, široka dostupnost i popularnost razvojne okoline omogućili su da se danas po vrlo niskim cijenama može početi s izradom prototipa, koji postaju sve pristupačniji za hobi korisnike, obrazovne ustanove, kao i profesionalce koji trebaju brze prototipe.

Arduino kompanija imala je značajan utjecaj u cijelom svijetu u raznim domenama. U obrazovanju, originalne razvojne pločice *Arduino* i razvojna okolina postali su standardni dio kurikuluma u školama i fakultetima diljem svijeta omogućujući studentima da steknu praktična iskustva u elektronici i programiranju. U industriji razvojna okolina *Arduino* koristi se kako bi se izradio brzi prototip, čime se smanjuje vrijeme i trošak razvoja novih proizvoda te potiče proučavanje i razmišljanje o idejnom riješenu.

U svojim početcima platforma *Arduino* bila je poznata po svojim razvojnim sustavim temeljenim na *Atmel AVR* arhitekturi, ali u kasnijem razvoju ostalih kompanija, počela se koristiti kao pojam za nove razvojne pločice različitih arhitektura dostupnih na tržištu. U njih je uključen i asortiman pločica kompanije *Espressif*.

Espressif je razvio vlastiti programski okvir *ESP-IDF*, koji se danas često koristi u profesionalnim primjenama kada je potrebno ostvariti punu funkcionalnost i mogućnosti koje nude pristup na niskoj razini. Kako bi se početnicima pružila mogućnost korištenja već napisanih programskih isječaka bez potrebe za njihovim prepisivanjem, *Espressif* je razvio biblioteku *Arduino Core* koja je povezana s razvojnim okvirom *Arduino*. Ova biblioteka omogućuje korištenje razvojne okoline *Arduino* na mikrokontrolerima kompanije *Espressif* bez potrebe za zančajnijim promjenama. Kako bi bilo prezahtjevno ostvariti potpunu kompatibilnost između ova dva razvojna okvira pri svakom novom ažuriranju značajki, biblioteka se obnavlja svakih nekoliko verzija. U trenutku pisanja ovog rada biblioteka je obnovljena

tako da koristi verziju 5.1.4 *ESP-IDF* razvojne okoline, dok je najnovija verzija 5.2.2. Donedavno je razlika u verzijama bila značajno veća, što može uzrokovati probleme u kasnijoj izradi projekta. Kako je već bio razvijen temelj cjelokupnog sustava u razvojnoj okolini *Arduino*, nije bilo potrebe za ažuriranjem razvojne okoline zbog toga što je glavni cilj napraviti funkcionalni prototip sustava.

Organizacija *Arduino* nudi besplatni program za uređivanje i pisanje programa nazvan *Arduno IDE*. Program je posebno prikladan u situacijama kad je potrebno napraviti brz i jednostavan prototip koji može poslužiti kao demonstracijski primjer. Kod velikih projekata poput ovog to nije moguće zbog velikog broja funkcionalnosti koje je potrebno ostvariti. Iz tog razloga mnoge firme i napredniji inženjeri koriste alternativnu opciju *PlatformIO*, koja je jedna od mogućih alternativa.

PlatformIO [5] napredno je razvojno okruženje koje podržava više platformi, među kojima su *Arduino* i *ESP-IDF* razvojna okruženja. Razvojno okruženje *PlatrofmIO* nema svoj specifičan uređivač koda te je integriran u jedan od najpoznatijih uređivača koda kompanije *Microsoft* pod nazivom *Visual Studio Code*. *PlatformIO* pruža profesionalne alate za razvoj softvera, kao što su napredni sustavi za upravljanje bibliotekama, integracija s različitim razvojnim okruženjima i mogućnost jednostavnog prelaska između različitih razvojnih okolina (engl. *multiple enviroment*), korištenih funkcionalnosti ovog sustava. Takva fleksibilnost omogućava profesionalnim programerima da koriste *PlatformIO* za razvoj složenih i profesionalnih projekata. Okruženje je popularno u industriji zbog svojih brojnih mogućnosti, koje omogućuju efikasan razvoj, testiranje i implementaciju programskih rješenja.

4.1.2. Korištene biblioteke

Kako bi se uspostavila komunikacija s dodatnim modulima, bilo je potrebno ugraditi odgovarajuću programsku potporu u projekt. S obzirom na to da je razvojna okolina *Arduino* dosta raširena i popularna u svijetu ugradbenih sustava, postoji veliki broj postojećih biblioteka koje su otvorene za korištenje, što u fazi izrade prototipa jako olakšava implementaciju. Iako je biblioteka *Arduino* navedena kao jedna od prednosti za brzi razvoj, također je i jedna od nedostataka zbog manjka kontrole i povećane razine apstrakcije pri pisanju programa. Zbog strukture biblioteka nije moguće lako promijeniti željene parametre na jednostavan način. Uvijek se može mijenjati izvorni kod biblioteke, ali nema načina da se s korisničke strane jednostavno mijenjaju parametri. Takav način korištenja vanjskih biblioteka bez dubljeg razumijevanja i mogućnosti izmjene ključnih parametara može biti od pomoći početnicima, ali iskusnim programerima stvara posebne probleme kada žele potpuno iskoristiti mogućnosti sustava.

Arduino LoRa

Ova biblioteka koristi se za uspostavljanje komunikacije između mikrokontrolera i modula *LoRa*. Biblioteka također podržava veliki broj modula raznih proizvođača, među kojima je i modul *SX127x* korišten u ovom sustavu. Biblioteka koristi *SPI* komunikaciju za spajanje s modulom. Kako bi se mogli primati i slati podatci, biblioteka implementira prekidni i kontinuirani način rada. Prekidni način rada koristi standardni izvod tipa *GPIO* mikrokontrolera kako bi mogao ući u prekidnu funkciju odmah pri primitku nove poruke. Nakon primanja poruke na odabrani se izvod dovodi impuls, koji dovodi mikrokontroler u prekidni način rada i pritom odrađuje korisničku funkciju. Kontinuirani način aktivno traži od korisnika da provjeru postoji li neobrađena poruka u međuspremniku. U trenutačnoj implementaciji jedinica iskorišten je kontinuirani način rada, što nije idealno rješenje zbog velikog broja pregledavanja spremnika u jedinici vremena, čak i ako je prazan. Ovakav način nepotrebno troši procesorske resurse. Ipak, bilo je prihvatljivo za testiranje funkcionalnosti sustava. Biblioteka omogućava samo prijenos podataka pomoću komunikacijskog sustava *LoRa*, a nigdje ne definira protokol slanja i primanja. Protokol koji je za to napravljen temeljen je na prijenosu znakovnih nizova posloženih tako da prate standardni raspored prikupljenih podatka GPS modula i akustične jedinice.

SafeString

S obzirom da je sustav napravljen za obradu velikog broja podataka, koji dolaze u obliku polja znakova, bilo je potrebno osigurati da manipulacija znakovnim nizovima ne usporava rad sustava. Iz tog je razloga korištena biblioteka *SafeString*. Uz neblokirajuće funkcije koje se nalaze unutar biblioteke također se nalaze i određene funkcije koje su korisne pri spremanju znakovnih nizova u pričuvnu memoriju. Te funkcije rade sigurnosnu provjeru za razne napade temeljene na standardnim znakovnim nizovima (engl. *buffer overflow*) i nude brojne druge mogućnosti. Biblioteka ne pohranjuje znakovne nizove na način kako se pohranjuju u implementaciji jezika *C* i *C++*, nego ih pohranjuje na vlastiti način, koji omogućuje brži dohvat znakova i bolji način njihova spremanja. Biblioteka također sadrži funkcije za njihovu pretvorbu u standardne znakovne nizove jezika *C*.

TinyGPS

TinyGPS jedna je od važnijih korištenih biblioteka jer se njome interpretira komunikacija s GPS modulom. Iako GPS modul šalje podatke određenom fiksnom frekvencijom na serijsko sučelje, ti su podatci formatirani u poseban standard *NMEA* (engl. *National Marine Electronics Association*).

NMEA je protokol koji se koristi za komunikaciju između elektroničkih uređaja u bro-

dovima i za GPS tehnologiju. Standard omogućava različitim uređajima poput GPS prijemnika, sonara, autopilota i računala, razmjenjuju podatka u njima poznatom obliku. *NMEA 0183*, koja je najčešće korištena verzija *NMEA* protokola, specificira posebni format za serijsku komunikaciju, gdje se informacije prenose u obliku rečenica koje počinju znakom *$* i završavaju kontrolnom sumom (engl. *checksum*), koja povećava redundanciju i robustnost podataka. *NMEA* standard koristi kodirane tekstualne poruke, koje uključuju različite vrste podataka, kao što su pozicija u obliku zemljopisne širine i dužine, brzina, smjer kretanja i vrijeme. Primjerice, rečenica koja započinje s *$GPGGA* sadrži podatke o fiksnoj poziciji.

Kako bi sustav bio što jednostavniji i s manje potencijalnih grešaka, korištena je biblioteka *TinyGPS*. Ona pruža objektno orijentirano sučelje koje sadrži metode za nadopunjavanje vlastitog međuspremnika sa znakovima koji pristižu sa serijskog sučelja GPS modula. Ovakvim načinom apstrahiraju se informacije koje pristižu u oblik jednostavnim za čitanje i manipulaciju u kodu.

4.1.3. ESP-NOW komunikacija

Komunikacija s kopnenom jedinicom je ostvarena pomoću *LoRa* komunikacijskog sustava. Za komunikaciju između vodenih jedinica također je mogao biti korišten i sustav *LoRa*, ali bi to značilo da se na sekundarne jedinice također moraju instalirati dodatni moduli te implementirati napredniji protokol, kojim bi se osiguralo da se informacije neće pogrešno interpretirati te da ih neće detektirati pogrešni uređaji. Uvođenjem dodatnih jedinica također se povećava mogućnost kvara. Iz tih je razloga traženo alternativno rješenje koje bi omogućilo komunikaciju među jedinicama korištenjem već postojećih elektroničkih komponenti sustava. Budući da odabrane razvojne pločice na jedinicama sadržavaju antene frekvencije 2,4 GHz, koje se primarno koriste za Wi-Fi i Bluetooth komunikaciju, te su pločice implementirane u sustav. Konkretno, Wi-Fi i Bluetooth protokoli dosta su napredni te zahtijevaju određene procedure uparivanja uređaja kako bi se mogla ostvariti jednostavna komunikacija. Primjerice, kako bi se sustav spojio Wi-Fi mrežu, potrebno je koristiti protokol koji je temeljen na njoj, kao što je *MQTT* (engl. *Message Queuing Telemetry Transport*), koji se često upotrebljava u *IoT* rješenjima zbog svoje robusne i jednostavne arhitekture. Tim se protokolom podatci razmjenjuju kao znakovni nizovi najčešće kodirani u formatu *JSON* (engl. *Javascript Object Notation*). Kako protokol zahtijeva konstantnu povezanost uređaja s instancom tzv. *MQTT Broker*, to dovodi nepotrebnu složenost u sustav. Iz sličnih se razlog ne koristi ni Bluetooth komunikacija.

ESP-NOW je bežični protokol za prijenos podataka razvijen od strane tvrtke *Espressif*. Protokol omogućava brzu i efikasnu komunikaciju između uređaja bez potrebe za uspostavljanjem stalne Wi-Fi veze. *ESP-NOW* koristi jednostavnu arhitekturu sličnu mrežama tipa *peer-to-peer*, gdje uređaji mogu komunicirati izravno jedni s drugima, bez potrebe za izra-

dom dodatnih pristupnih točaka ili mrežnog sustava nalik arhitekturi zvijezde s posebnim mrežnim ulogama. Uspoređujući *ESP-NOW* s Wi-Fi i Bluetooth komunikacijom, postoji nekoliko ključnih razlika. Wi-Fi pruža visoke brzine prijenosa podataka i široku pokrivenost, ali troši više energije i zahtijeva složeniju arhitekturu. S druge strane, Bluetooth je poznat po nižoj potrošnji energije od Wi-Fi komunikacije, ali ima ograničen domet i manju brzinu prijenosa podataka. *ESP-NOW* bolji je od oba protokola u specifičnim scenarijima jer omogućava brzi prijenos podataka uz minimalnu potrošnju energije, pri čemu nudi jednostavnu početnu konfiguraciju i izravnu komunikaciju između uređaja.

Kompanija *Espressif* također je napravila istoimenu biblioteku kao dio svoje standardne razvojne okoline *ESP-IDF*, koja omogućuje korištenje *ESP-NOW* protokola. Osnovne su funkcije slanja i primanja podataka *ESP-NOW* protokola:

- `esp_now_send()`,
- `esp_now_recv()`.

Funkcija `esp_now_send()` omogućava uređaju slanje podataka na određenu *MAC* adresu (engl. *Media Access Control address*) drugog uređaja, dok `esp_now_recv()` omogućava prijem podataka koji su poslani na uređaj. Prilikom slanja podataka korisnik može definirati podatkovni paket i *MAC* adresu odredišnog uređaja, a prijem podataka obrađuje se funkcijom povratnog poziva (engl. *callback*), koja se poziva pri primanju novog paketa.

Zahvaljujući tim značajkama, *ESP-NOW* idealan je za primjene u *IoT* sustavima gdje je potrebna brza i pouzdana komunikacija s niskom potrošnjom energije. Primjeri takvih sustava su pametne kuće, razni ruralni sustavi i sl. Zbog svoje jednostavnosti i učinkovitosti *ESP-NOW* pokazao se kao vrlo koristan protokol za brojne bežične aplikacije kao u ovom sustavu.

4.1.4. Arhitektura programske potpore

Programska potpora glavne vodene jedinice, sekundarne vodene jedinice te dijela kopnene jedinice je pisana unutar *Arduino* razvojne okoline unutar razvojnog sučelja *PlatformIO*. Budući da jedinice koriste dijelove sustava sličnog tipa (kao što su primjerice način procesiranja pristiglih *LoRa* i GPS poruka) sva programska potpora stavljena je unutar istog projekta s mogućnošću odabira jedinice za koju se u određenom trenutku uređuje kod. Za to je korištena mogućnost razvojnog sučelja *PlatformIO* pod nazivom *environments*, kojom se jednostavno mogu odabrati i postaviti određene postavke specifične za određenu arhitekturu, elektronički postav ili neku sličnu konfiguraciju. Sve tri vrste jedinica koriste istu arhitekturu i iste programske biblioteke, ali s različitim bazama programskog koda. U konfiguracijskoj datoteci *PlatformIO* projekta postavljeni su parametri, koji se pri prevođenju izvornog koda postavljaju kao izrazi tipa makro unutar programske potpore. Ovisno o izabranoj opciji iz-

bornika *environments* pri prevođenju programske potpore mogu se odabrati željeni dijelovi koda ili vrijednosti određenih konstanti.

Kako jedinice koriste određene sustave na isti način, za potrebe rada napisano je više vlastitih biblioteka. Napisane su dodatne biblioteke za *LoRa* i *ESP-NOW* komunikaciju.

Vlastite biblioteke

Vlastita *LoRa* biblioteka sadrži metode koje enkapsuliraju pozive iz *Arduino LoRa* biblioteke te time smanjuju redundanciju programske potpore i unificiraju protokol slanja i primanja podatka. Glavne funkcionalnosti ove dodatne biblioteke obrada su znakovnih nizova kako bi se mogli poslati s jedne jedinice na drugu te inicijalizacija modula *LoRa*. Biblioteka sadrži datoteku zaglavlja te datoteku programske implementacije. Unutar datoteke zaglavlja nalaze se parametri koji pomažu pri korištenju biblioteke. Od važnijih parametara potrebno je istaknuti makro naredbe koje definiraju adresu sustava *LoRa* kopnene jedinice (*LORA_CoastAddress*) te glavne vodene jedinice(*LORA_MasterAddress*). One služe za jednostavno definiranje vlastitog protokola slanja podataka. Također, neki od parametara koji su potrebni su oznake izvoda mikrokontrolera *GPIO*, preko kojih je ostvarena komunikacija s modulom *LoRa*.

```
#ifndef LORA_CONFIG_H
#define LORA_CONFIG_H

#include <Arduino.h>
#include <LoRa.h>
#include "SafeStringReader.h"

#define LORA_frequency 866E6         // LoRa Frequency
#define LORA_csPin     5             // LoRa radio chip select
#define LORA_resetPin  4             // LoRa radio reset
#define LORA_irqPin    27            // Hardware interrupt pin (NOT USED)

#define LORA_MasterAddress    0xAA   // address of this device
#define LORA_CoastAddress     0xFF   // destination of board on the coast

// Initialize LORA communication
void LORA_init();

// Send a LORA message with "outgoing" data from "sourceAddress" to "
   destinationAddress"
void LORA_sendMessage(SafeString& outgoing, byte sourceAddress, byte
   destinationAddress);

```

```
22 #endif
```

Odsječak kôda 4.1: Prikaz zaglavlja vlastite *LoRa* biblioteke

Iako komunikacijski sustav *LoRa* ne propisuje specifičan protokol prema kojem su podatci uređeni, postoji nekoliko standardiziranih metoda koje se često koriste kako bi se ostvarila jednostavna komunikacija između nekoliko modula. U ovom sustavu implementiran je modificirani protokol preuzet iz biblioteke *Radiohead* koja je postala standard u mnogim drugim sustavima. Protokol propisuje da se pri slanju podataka moraju također poslati informacije o pošiljatelju i primatelju te dodatni bitovi za validaciju poruke. Za potrebe testiranja nije implementiran nikakav napredan sustav kao što su *CRC* (engl. *Cyclic Redundancy Check*) ili funkcija sažetka (engl. *hash function*), nego su za zaštitnu metodu korištena duljina poruke, čija je veličina postavljena na mjesto zadnjeg okteta. Također je impementiran brojčani identifikator poruke koji se sekvencijalno povećava svakim novim slanjem poruke. On je stavljen na mjesto predzadnji oktet poruke. U budućnosti bi se trebala implementirati jača i konkretnija zaštita poruke pomoću sigurnije funkcije sažetka ili asimetrične enkripcije kao što je *AES* (engl. *Advanced Encryption Standard*), ali za trenutačno testiranje funkcionalnosti sustava to nije potrebno.

```
1  void LORA_sendMessage(SafeString& outgoing, byte sourceAddress, byte
       destinationAddress) {
2    LoRa.beginPacket();                   // start packet
3
4    LoRa.write(destinationAddress);       // add destination address
5    LoRa.write(sourceAddress);            // add sender address
6    LoRa.write(msgCount);                 // add message ID
7    LoRa.write(outgoing.length());        // add payload length
8
9    LoRa.print(outgoing);                 // add payload
10   LoRa.endPacket();                     // finish packet and send it
11
12   msgCount++;                           // increment message ID
13 }
```

Odsječak kôda 4.2: Prikaz formata funkcije slanja *LoRa* poruke s vidljivim formatom slanja

Druga vlastita biblioteka predstavlja dodatnu potporu protokola *ESP-NOW*. Napisana je iz istih razloga kao i *LoRa* biblioteka, da smanji redundanciju u programskoj potpori te da poveća efikasnost pisanja i uređivanja dijelova sustava. Biblioteka sadrži vlastite metode koje olakšavaju slanje i primanje poruka preko protokola *ESP-NOW* zajedno s posebnim izrađenim sustavom za smanjenje greške pri slanju podataka. Biblioteka se također sastoji od datoteke zaglavlja koja definira parametre potrebne za samu inicijalizaciju protokola te za

Tablica 4.1: Prikaz formata *LoRa* poruke preuzet iz *Radiohead* biblioteke

Paketno zaglavlje biblioteke *Radiohead* (4 bajta)	
Bajt 0	DO — odredišni čvor za ovaj paket (255 = Emitiranje prema SVIMA)
Bajt 1	OD — izvorni čvor za ovaj paket (255 = Emitiranje poruke - podrijetlo nepoznato)
Bajt 2	ID - Identifikator — sadrži brojač sekvenci za Pouzdan Datagram Mod - Korisnički definirano za ostale modove
Bajt 3	Bit 7: 1 = ACK Paket, 0 = Normalni Paket Bit 6: 1 = PONOVI, 0 = Prvi pokušaj Bit 5: REZERVIRANO Bit 4: REZERVIRANO Bitovi 3-0: Korisnički definirano

konfiguraciju adresa jedinica. Kako se protokol oslanja na poznavanje *MAC* adresa uređaja kojemu se šalju podatci, potrebno je na neki način saznati tu informaciju prije početka slanja. *ESP-NOW* protokol ne sadržava metodu koja omogućuje dinamičko adresiranje uređaja kao kod protokola IP (engl. *Internet Protocol*) u svojoj implementaciji pomoću protokola *ARP* (engl. *Address Resolution Protocol*). Poznavanjem funkcionalnosti takvog protokola *ARP* može mu se replicirati ponašanje i napraviti svoj vlastiti. Zbog jednostavnosti implementacije uzete su fizičke *MAC* adrese jedinica, koje su definirane u datoteci zaglavlja. Kako bi se unificirao paket koji se šalje preko mreže *ESP-NOW*, napravljene su i vlastite strukture koje se koriste kako bi mogli poslati informacije između jedinica. Prva struktura tipa `request_ping_t` koristi se kako bi glavna vodena jedinica mogla poslati naredbu koja je primljena s kopna sekundarnim vodenim jedinicama. Struktura se sastoji od polja znakova maksimalne duljine 40 znakova. Ovo ograničenje definirano je testiranjem i zaključkom o maksimalnom broju znakova koji se mogu nalaziti u jednoj poruci, a da je ona i dalje validna. Druga struktura tipa `response_ping_t` sadrži polje znakova maksimalne duljine 100 okteta i sadrži još dodatni oktet koji predstavlja identifikator sekundarne jedinice. Identifikator sekundarne jedinice služi kako bi glavna jedinica mogla provjeriti od koje sekundarne jedinice dolaze poruke te shodno tome izvršiti određeni dio programskog bloka.

```
#ifndef ESP_NOW_CONFIG_H
#define ESP_NOW_CONFIG_H

#include <Arduino.h>

#include <esp_now.h>
#include <WiFi.h>

#define SLAVE_1_MAC {0xCC, 0xDB, 0xA7, 0x12, 0x66, 0x48}
#define SLAVE_2_MAC {0xCC, 0xDB, 0xA7, 0x12, 0x92, 0x84}
#define MASTER_MAC  {0x0C, 0xB8, 0x15, 0x75, 0xC1, 0x68}

#define REQUEST_MAX_STR_LENGTH  40
#define RESPONSE_MAX_STR_LENGTH 100

#define REQUEST_RETRY_MAX_NUM 10
#define SLEEP_BETWEEN_TRIES 2000

typedef struct {
    char request[REQUEST_MAX_STR_LENGTH];
} request_ping_t;

typedef struct {
    uint8_t slave_id;
    char response[RESPONSE_MAX_STR_LENGTH];
} response_ping_t;

bool esp_now_setup_master();
bool esp_now_setup_slave();

void esp_now_request_ping(const uint8_t *reciever_mac, request_ping_t
   data);
void esp_now_send_ping_data(response_ping_t data);

void esp_now_custom_send_callback(const uint8_t *mac_addr,
   esp_now_send_status_t status);

#endif
```

Odsječak kôda 4.3: Prikaz zaglavne datoteke vlastite*ESP-NOW*biblioteke

Kako bi sustav očuvao redundanciju, dodan je također odziv na status slanja poruke, koji prenosi informaciju o samoj uspješnosti slanja. Ovaj mehanizam implementiran je tako da se slanje podataka ponavlja s podacima iz prijašnjeg pokušaja slanja ako je broj trenutnih pokušaja manji od maksimalnog. Prijašnji podatci spremaju se u pričuvnu memoriju odmah

pri prvobitnom pokušaju slanja. Ako je slanje bilo neuspješno, podatci iz pričuvne memorije šalju se ponovno. Nakon maksimalnog broja pokušaja slanja na serijski se izlaz šalje poruka da je slanje bilo neuspješno te se prekida ciklus slanja.

Kopnena jedinica

Primarni je zadatak kopnene jedinice dekodiranje poruka s vodenih jedinica, koje su joj poslane preko serijskog sučelja računala, javljanje primljenih podataka vodenih jedinica te njihovo slanje natrag na računalo. Zbog toga je programska arhitektura ove jedinice relativno jednostavna.

Na početku se definiraju globalne varijable za odlazne poruke sustava *LoRa* i varijable za spremanje primljenih podataka od vodenih jedinica. U funkciji `setup`, koja je standardna funkcija razvojne okoline *Arduino*, inicijalizira se serijska komunikacija i provjerava uspješnost njezine inicijalizacije. Ako inicijalizacija ne uspije, ESP32 se ponovno pokreće. Uz serijsku komunikaciju inicijalizira se i *LoRa* komunikacija. U `loop` funkciji program prvo provjerava postoji li nova poruka na serijskom sučelju. Ako je dostupna, njezin sadržaj se dekodira u instancu objekta `SafeStringReader` i prosljeđuje na glavnu jedinicu koristeći funkciju `LORA_sendMessage`, koja je definirana u vlastitoj biblioteci. Također je definirana dodatna funkcija koja se koristi za obradu dolaznih paketa, gdje se prvo provjerava veličina paketa, zatim čitaju osnovne informacije (primatelj, pošiljatelj, ID poruke, duljina poruke) i, konačno, provjerava duljina poruke kako bi se otkrile eventualne greške.

Funkcija za dekodiranje poruka periodički se poziva kako bi se konstantno obrađivali najnoviji podatci, dok se međuspremnik serijskog sučelja postavlja na prazno početno stanje kako bi se osigurala stalna propusnost podataka.

Glavna vodena jedinica

Glavna vodena jedinica ima najkompleksniju zadaću. Cilj glavne jedinice je da pri pristizanju poruke od kopnene jedinice odredi podatke potrebne za izračun udaljenosti ronioca te da proslijedi zahtjev sekundarnim jedinicama. Nakon što i sekundarne jedinice prime podatke udaljenosti od ronioca, prosljeđuju ih natrag na glavnu jedinicu kako bi se svi podatci skupili i poslali natrag na kopnenu jedinicu.

Na početku u funkciji `setup` inicijaliziraju se serijski izlazi *ESP-NOW*, *LoRa* i GPS modul. Od serijskih izlaza definiraju se tri kako bi se ispunile sljedeće funkcionalnosti:

- komunikacija s *Delphis v3 nanomodemom*,
- komunikacija s GPS modulom,
- izlazno sučelje za poruke zapisnika.

Registriraju se *callback* funkcije za *ESP-NOW* komunikaciju kako bi glavna jedinica mogla reagirati na poruke koje pristižu od sekundarnih jedinica. Ovisno o tipu (primjerice poruka o dobivenoj udaljenosti ronioca) poruka se obrađuje i pohranjuje u globalnu varijablu, koja kasnije služi za daljnje slanje na kopnenu jedinicu.

Također se pri samoj inicijalizaciji sustava izrađuje nova dretva kojoj je svrha ciklički pregledavati jesu li došli novi podatci od kopnene jedinice preko komunikacijskog kanala *LoRa*. Ovdje se također obrađuju odgovori i, ako su svi primljeni, šalje se konačna poruka natrag obalnoj stanici.

Funkcija za obradu *LoRa* poruka prvo provjerava veličinu paketa i čita zaglavlje poruke. Ako duljina poruke ne odgovara očekivanoj duljini, izvođenje funkcije se prekida. Također, funkcija provjerava o kojem se uređju radi pomoću dodatnog bajta, koji se nalazi u zaglavlju poruke. Ako je poruka namijenjena glavnoj jedinici, prosljeđuje ju dalje sekundarnim jedinicama i pokreće automat s konačnim brojem stanja mjerenja udaljenosti.

Mjerenje udaljenosti odvija se sekvencijalno, pri čemu se poruka prosljeđuje različitim jedinicama. Prvobitno stanje uzima podatke mjerenja od akustične jedinice, dekodira ih te ih sprema u globalni međuspremnik. Ovaj međuspremnik nastavlja se potom puniti podacima koje glavna jedinica dobije od sekundarnih jedinica prelascima između ostala dva stanja.

Kako bi se potrebni podatci za slanje na sekundarne jedinice mogli upakirati, definira se dodatna funkcija `publishData`, koja sprema podatke u strukturu te ih šalje protokolom *ESP-NOW*. Funkcija u prototipu prima poruku tipa *SafeString* i indeks ciljane sekundarne jedinice. Ovisno o tom indeksu, odabire *MAC* adresu iz predefiniranih makro konstanti `SLAVE_1_MAC` ili `SLAVE_2_MAC`.

Glavna `loop` funkcija provjerava stanje mjerenja. Ako mjerenje nije aktivno, pokreće `GPS_Manager` za ažuriranje GPS podataka. Funkcija `GPS_Manager` obrađuje GPS podatke tako da čita pristigle GPS podatke sa serijskog priključka i ažurira varijable globalnog međuspremnika zemljopisne širine i dužine. Takvim načinom osigurana su najažurnija mjerenja neovisno o trenutku primitka podataka od modula.

Sekundarna vodena jedinica

Funkcionalnost sekundarne vodene jedinice slična je funkcionalnosti glavne jedinice, ali ne uključuje cijelu logiku komunikacije s kopnenom jedinicom. Iz tog su razloga preuzeti samo određeni programski dijelovi.

U `setup` funkciji inicijaliziraju se serijski kanali, *ESP-NOW* i GPS modul te se registriraju *callback* funkcije za komunikaciju protokolom *ESP-NOW*. Kao i kod glavne jedinice uspostavljaju se serijski komunikacijski protokoli, koji se koriste za sljedeće funkcionalnosti:

- komunikacija s *Delphis v3 nanomodemom*,
- komunikacija s GPS modulom,

– izlazno sučelje za poruke zapisnika.

U `loop` funkciji provjerava se stanje mjerenja kao i kod glavne jedinice. Ako se trenutno odvija mjerenje, poziva se `GPS_Manager` funkcija za ažuriranje GPS podataka. Također se poziva i funkcija za analizu podataka mjerenja udaljenosti preko akustične jedinice. Ako je dekodirana poruka ispravnog statusa (tj. ako je akustična jedinica vratila podatke mjerenja ili statusne podatke neuspjeha), šalje se obrađena poruka pomoću protokolom *ESP-NOW*, nakon čega se briše globalni međuspremnik za podatke akustične jedinice i zaustavlja mjerenje.

Funkcija `publishData` šalje podatke udaljenosti ronioca natrag na glavnu jedinicu protokolom *ESP-NOW*. Funkcija za dekodiranje podataka akustične jedinice analizira ulazni niz tražeći standardne ključne riječi poput *ACK*, *LOG* i *RNG*, na temelju kojih određuje vrstu primljene poruke. Ako je poruka tipa *LOG* ili *RNG*, funkcija obrađuje podatke i pohranjuje ih globalnu međuspremnik.

4.2. Kopnena obrada informacija

Informacije dobivene od vodenih jedinica preko *LoRa* mreže se trebaju se na neki način prikazati korisniku sustava. U tu svrhu bilo je potrebno razviti dodatnu programsku potporu za vizualizaciju GPS lokacija vodenih jedinica. Također potrebno je prikazati lokaciju ronioca, a ta se informacija ne primamo izravno od vodenih jedinica, nego se računa pri dolasku novih valjanih podataka.

Zbog toga je sustav podijeljen u dva dijela. Prvi je dio aproksimacija lokacije ronioca što obuhvaća i vizualizaciju podataka, a drugi dio opisuje praćenje stanja vodenih jedinica i komunikacijski sustav općenito.

4.2.1. Algoritam aproksimacije lokacije i dubine

Sa svake vodene jedinice dobivaju se dvije informacije:

– geografska širina i dužina vodene jedinice,

– udaljenost vodene jedinice od ronioca izražena u metrima.

Tim se podacima procjenjuju lokacija i dubina. Zbog rada u 3D prostoru i dodatne preciznosti u idealnom je scenariju potrebno koristiti skup od četiri vodene jedinice. S četiri jedinice moguće je odrediti točno i nadmorsku visinu ronioca (tj. njegovu dubinu). Kako sustav koristi samo 3 vodene jedinice zbog troškova izrade dodatne, algoritam za procjenu prilagođen je tako da može prikazivati i dalje usporedive i iskoristive rezultate. Slično tome, GPS tehnologija može koristiti samo 3 satelita za određivanje lokacija na površini Zemlje s

relativnom aproksimacijom nadmorske visine.

Presjekom triju sfera dobivaju se dvije točke u 3D prostoru, koje mogu biti potencijalne nadmorske visine ronioca. Jedna točka nalazit će se u 3D prostoru s pozitivnom nadmorskom visinom, a druga na nadmorskoj visini koja je negativna. Kako je poznato da je ronioc na nadmorskoj visini koja je niža od horizonta (negativna nadmorska visina), odbacuje se točka s pozitivnom nadmorskom visinom. Također, u slučaju da je procijenjena dubina iznad razine horizonta, obje se točke zanemaruju. Ova metoda nije najpreciznija, ali se njome može odrediti dubina ronioca. Što je ronioc na većoj dubini, to će magnituda greške u određivanju lokacije biti veća. Kako se sustav koristi na relativno malim dubinama (do 100 metara), pogreška nije toliko izražena.

Slanje zahtjeva i njegovo dekodiranje

Kopneni sustav radi kao beskonačna petlja koja svakih 10 sekundi šalje zahtjev kopnenoj jedinici za prikupljanje podataka udaljenosti i lokacije. Poruka koja se šalje je u obliku:

$G,<NUM>,RNG

gdje je *<NUM>* zamjenjiv decimalni identifikacijski broj akustične jedinice ronioca. Ova naredba jedna je od mnogih koje su implementirane na dodatnom modulu akustične jedinice, ali radi jednostavnosti je jedina koja se koristi i za koju je napravljen algoritam procesiranja odgovora.

Nakon slanja poruke zahtjeva potrebno je dekodirati odgovor. Kopnena jedinica šalje podatke koje primi preko *LoRa* mreže serijskim protokolom *RS-232* u obliku znakovnih nizova. Redovi su napravljeni tako da svaki sadrži oznaku tipa poruke:

- *[RNG]*; izmjereni podatci vodene jedinice zajedno s njenom identifikacijskom oznakom,
- *[FAIL]*; obavijest vodene jedinice o pogrešci prilikom procedure mjerenja.

Poruka *[RNG]* daje informacije o udaljenosti određene vodene jedinice od ronioca (podvodne jedinice) zajedno s GPS lokacijom i njezinom preciznošću. Preciznost koja se opisuje standardna je horizontalna preciznost *HDOP* (engl. *Horizontal Dilution Of Precision*). Trenutno se koristi samo za potrebe nadgledanja sustava, ali se može implementirati u budućim iteracijama računanja cjelokupne greške kod procjene lokacije ronioca. Također, ova poruka daje informaciju o ukupnom vremenu trajanju procedure dohvaćanja lokacije i udaljenosti za sve vodene jedinice. Koristi se također samo u svrhe nadgledanja sustava.

Poruka *[FAIL]* sustavu javlja informaciju o neuspjehu pri dobivanju informacije o udaljenosti vodene jedinice od ronioca. Moguće je da akustična jedinica uđe u stanje pogreške koja se mora javiti korisniku. Trenutno se ta poruka koristi samo za potrebe nadgledanja sustava te nisu poduzete nikakve mjere za rješavanje problema ili naknadne intervencije.

Algoritam estimacije pozicije i dubine ronioca

Nakon što su podatci s kopnene jedinice dekodirani, potrebno je odrediti estimaciju lokacije ronioca. Početni je korak estimacije je pretvoriti i translatirati koordinatni sustav. Pretvorba sustava iz geografskih širina i visina u Kartezijeve koordinate s odmakom od apscise i ordinate izvodi se trigonometrijskom translacijom. U pretvorbi se kao jedan od parametara uzima radijus Zemlje kako bi se mogli odrediti potrebni odmaci. Kada se generiraju Kartezijeve koordinate, zbog jednostavnosti sustava koordinate se pomiču tako da referentno ishodište sustava postane prvobitna glavna vodena jedinica.

```
def latlon_to_cartesian(lat, lon, origin_lat, origin_lon):
    lat = np.radians(lat)
    lon = np.radians(lon)
    origin_lat = np.radians(origin_lat)
    origin_lon = np.radians(origin_lon)

    x = EARTH_RADIUS * (lon - origin_lon) * np.cos((lat + origin_lat) / 2)
    y = EARTH_RADIUS * (lat - origin_lat)
    return x, y

def cartesian_to_latlon(x, y, origin_lat, origin_lon):
    origin_lat = np.radians(origin_lat)
    origin_lon = np.radians(origin_lon)

    lat = y / EARTH_RADIUS + origin_lat
    lon = x / (EARTH_RADIUS * np.cos((lat + origin_lat) / 2)) + origin_lon

    lat = np.degrees(lat)
    lon = np.degrees(lon)
    return lat, lon
```

Odsječak kôda 4.4: Funkcije za pretvorbu i translaciju koordinata

Estimacija se izvodi pomoću algoritma minimizacije kvadratne pogreške udaljenosti. Korišten je algoritam *L-BFGS-B* (engl. *Limited-memory Broyden-Fletcher-Goldfarb-Shanno*

with Box constraints) [9] implementiran u biblioteci *SciPy* u sklopu programskog jezika *Python*. Funkcija minimizacije sadrži veliki broj ugrađenih implementacija, ali zbog njezine jednostavnosti i široke upotrebe u industriji je upravo ona odabrana.

L-BFGS-B je algoritam za optimizaciju koji se koristi za minimizaciju funkcija s ograničenjima na varijable. To je varijanta *BFGS* algoritma, koji je popularan u numeričkoj optimizaciji zbog svoje efikasnosti i sposobnosti aproksimacije matrice *Hessian* (matrice druge derivacije funkcije) koristeći informacije o gradijentu. Ključna prednost *L-BFGS-B* algoritma naspram standardnog algoritma *BFGS* je mogućnost postavljanja ograničenja na varijable. Ograničenja koja se mogu postaviti su donja i gornja granica numeričke vrijednosti (engl. *box constraints*). U sustavu je bilo potrebno postaviti ograničenja za aproksimaciju dubine ronioca jer nadmorska visina ronioca ne može biti viša od nadmorske visine vodenih jedinica.

U praksi algoritam koristi ograničenu memoriju, što ga čini prikladnim za probleme s velikim brojem varijabli. Umjesto pohranjivanja cijele matrice *Hessian*, koja može biti vrlo velika, algoritam pohranjuje samo nekoliko vektora, koji predstavljaju aproksimaciju. Ova tehnika smanjuje potrebnu memoriju i računalnu složenost čineći algoritam bržim i efikasnijim za složene probleme. Osnovni koraci uključuju iterativno ažuriranje procjena varijabli kako bi se postigao minimum funkcije koristeći informacije o gradijentu i aproksimiranoj matrici *Hessian*.

```
diver_est = minimize(objective_function, initial_guess, bounds=bounds)
```

Odsječak kôda 4.5: Pozivanje fukncije za minimizaciju biblioteke SciPy

Funkcija koja se minimizira kvadratna je suma pogrešaka Kartezijeve udaljenosti. Naime, kako bi se olakšala implementacija algoritma pretpostavljena je zakrivljenost Zemlje približna nuli. Zbog te pretpostavke može se koristiti Kartezijeva formula, kojom se računa udaljenost između dviju točka koordinatnog sustava.

Formulom se računa Kartezijeva udaljenost između lokacije vodene jedinice i vrijednosti za koju se računa pogreška. Formula je oblika:

$$d_{est}(i) = \sqrt{(x_{vj}(i) - x_t)^2 + (y_{vj}(i) - y_t)^2 + (z_{vj}(i) - z_t)^2}$$

.

pri čemu je:

- $d_{est}(i)$ procijenjena udaljenosti temeljena na trenutnim testnim podacima i vodene jedinice i,
- $x_{vj}(i)$ x-koordinata vodene jedinice i,

– $y_{vj}(i)$ y-koordinata vodene jedinice i,

– $z_{vj}(i)$ z-koordinata vodene jedinice i,

– x_t x-koordinata trenutne testne točke funkcije minimizacije,

– y_t y-koordinata trenutne testne točke funkcije minimizacije,

– z_t z-koordinata trenutne testne točke funkcije minimizacije.

Za svaku vodenu jedinicu računa se udaljenost koja se uspoređuje s izmjerenom udaljenosti dobivenom od akustičnih jedinica. Na kraju se sve dobivene kvadratne pogreške vodenih jedinica zbrajaju te rezultat postaje ukupna pogreška trenutačne iteracije algoritma:

$$E = \sum_{i=1}^{3} (d_{est}(i) - d_{vj}(i))^2$$

.

Varijable iz formule interpretiraju se na sljedeći način:

– $d_{est}(i)$: procijenjena udaljenosti temeljena na trenutnim testnim podacima i vodene jedinice i,

– $d_{vj}(i)$: izmjerena udaljenost između ronioca i vodene jedinice i.

Ti su matematički izrazi u programskoj potpori sustava implementirani u dodatnoj Python funkcija:

```
def objective_function(estimated_position):
    x, y, z = estimated_position
    estimated_distances = [
        np.sqrt((x - buoy1[0])**2 + (y - buoy1[1])**2 + (z - buoy1[2])
    **2),
        np.sqrt((x - buoy2[0])**2 + (y - buoy2[1])**2 + (z - buoy2[2])
    **2),
        np.sqrt((x - buoy3[0])**2 + (y - buoy3[1])**2 + (z - buoy3[2])
    **2)
    ]
    return np.sum((np.array(estimated_distances) - np.array([d1, d2, d3])
    )**2)
```

Odsječak kôda 4.6: Funkcija koja se koristi pri pozivanju algoritma minimizacije udaljenosti (algoritam kvadratne greške kartezijeve udaljenosti)

Nakon procjene lokacije ronioca sustav vraća koordinate u obliku geografske širine i dužine, pri čemu se uklanja translacija u odnosu na početnu vodenu jedinicu. Izračunata

vrijednost pritom se dalje koristi pri vizualizaciji lokacije unutar razvojne okoline *ROS* te programa za vizualizaciju *Foxglove*.

4.2.2. Vizualizacija podataka

Kako bi mogli jednostavno vizualizirati lokacije bova te lokaciju ronioca, korištena je *ROS* razvojna okolina i alat za vizualizaciju *Foxglove*.

ROS operativni sustav

ROS (engl. *Robot Operating System*) [6] skup je programskih biblioteka i alata za pisanje programske potpore usmjerene na područje robotike. *ROS* nije standardni operacijski sustav, već se može definirati kao razvojna okolina napravljena unutar operacijskog sustava *Linux*, koji pruža usluge poput komunikacije između procesa, hardverske apstrakcije, upravljanja uređajima, implementacije često korištenih funkcionalnosti, poruka između procesa i paketa upravljanja. Okolina se koristi za izgradnju raznih robotskih sustava u industriji.

ROS koristi čvorove (engl. *nodes*), koji predstavljaju procese koji se izvode. Čvorovi mogu međusobno komunicirati preko *tema* (engl. *topics*) i *usluga* (engl. *services*). Teme omogućavaju asinkronu komunikaciju postupcima objavljivanja i pretplate na poruke, što je inače poznatije u industriji kao arhitektura tipa *publisher-subscriber*.

Jedna od ključnih značajki razvojne okoline *ROS* njezin je paketni sustav, koji omogućava ponovno korištenje koda i jednostavno dijeljenje programskih komponenti među istraživačima i programerima. Paketi mogu sadržavati biblioteke, alate, skripte te druge resurse potrebne za robotske aplikacije.

ROS ima veliku zajednicu koja kontinuirano doprinosi njegovu razvoju čineći ga jednim od najpopularnijih alata na području robotike. Njegova modularnost i široka potpora za različite robote sustave i senzore čine ga idealnim izborom za istraživače, akademike i industrijske stručnjake.

ROS je trenutačno distrubuiran u dvije verzije: *ROS 1* te *ROS 2*. Zbog jednostavnosti i već poznatog načina rada korištena je prvobitna verzija okoline *ROS*. Razlike u verzijama obuhvaćaju način međusobnog povezivanja čvorova te načina izmjene informacija. *ROS 1* koristi glavni servis, takozvani *ROS Master*, koji služi za obavještavanje čvorova o prisutnosti drugih čvorova u mreži. *ROS 2* ne posjeduje čvor sličan čvoru *ROS Master*, što znatno olakšava nadgledanje velikih mreža čvorova.

Sustav je implementiran tako da se uspostavlja čvor koji se poziva navedenim algoritmima za aproksimaciju lokacije ronioca i za komunikaciju sustava s kopnenom jedinicom. Sustav postavlja podatke na dvije teme:

– *diver_location* -> lokacija ronioca,

– *buyo_gpsJson* -> vlastita karta lokacija vodenih jedinica.

Lokacija ronioca postavlja se na temu kojoj je vrsta poruke *LocationFix*, koja predstavlja lokaciju danu preko geografske širine i dužine, a vlastita karta generirana je pomoću tipa podataka *GeoJSON*, koji se često koristi kako bi se izradile interaktivne karte s poligonima, točkama i drugim geometrijskim likovima. Stvaranje objekata tipa *GeoJSON*, koji se postavljaju na temu za daljnju komunikaciju, izvodi se vanjskim bibliotekama *geojson*.

```
...
buoy1_feature = Feature(geometry=Polygon([buoy1_circle]), properties={
    "name": "Buoy 1",
    "color": "green",
    "style": {
        "stroke": "#00FF00",
        "stroke-width": 2,
        "fill": "#00FF00"
    }
})
...
```

Odsječak kôda 4.7: Prikaz stvaranja *GeoJSON* objekta za vizualizaciju na interaktivnoj karti

Foxglove vizualizacija

Foxglove Studio [3] napredni je alat za vizualizaciju podataka. Dizajniran je prvenstveno za potrebe robotike i autonomnih sustava. Omogućuje korisnicima jednostavno praćenje podataka u stvarnom vremenu, što je ključno za razvoj, testiranje i dijagnostiku robotskih aplikacija. *Foxglove* podržava razne vrste podataka uključujući telemetrijske podatke senzora, video transmisiju, karte i grafikone, čime se omogućuje sveobuhvatnu analiza i pregled složenih sustava.

Jedna je od njegovih ključnih značajki ugrađena potpora za *ROS*, što omogućava jednostavno integriranje s postojećim projektima u okolini. Osim razvojne okoline *ROS* alat *Foxglove* podržava i druge podatkovne oblike, što ga čini fleksibilnim za različite primjene.

Foxglove nudi korisničko sučelje koje je jednostavno za korištenje, s mogućnošću prilagodbe prikaza prema potrebama korisnika. Paneli za vizualizaciju mogu se lako dodavati, uklanjati i namještati, što omogućava prilagodbu okruženja na veliki raspon projekata. Interaktivne značajke poput zumiranja, pomicanja i prilagođavanja vremenskih skala olakšavaju detaljnu analizu podataka.

U sustavu se koriste paneli za mapu i praćenje poruka servisa te modul za praćenje poruka neke teme. Na taj način korisnik može jednostavno pratiti sve promjene koje se događaju pri

Slika 4.1: Izgled prilagođenog sučelja alata *Foxglove Studio*

komunikaciji sa sustavom.

5. Testiranje i rezultati

Kako bi se odredile značajke sustava, provedeno je testiranje na području reakreacijsko-sportskog centra Jarun. Ta je lokacija odabrana zbog odličnih uvjeta za testiranje podvodnih sustava u dostupnim velikim vodenim površinama, pristupačnog ulaska u vodu te mirnoće vode. Prototipi vodenih jedinica dodatno su osigurani od mogućeg prodora vode kako se ne bi oštetili pri korištenju, a podvodna je jedinica postavljena na fiksnu lokaciju.

Prilagođavanje elektroničkih komponenti

Elektronički dijelovi vodenih jedinica osigurani su od mogućeg prodora vode vodootpornim plastičnim kućištem. Kućište je uz samu prototipnu elektroniku sadržavalo i baterijski paket, koji je neposredno prije trajnog zatvaranja bio spojen na sustav i uključen. Kako bi se spojio ultrazvučni pretvornik na mikrokontroler *Nordic*, dodana je također i vodootporna uvodnica. Da bi se ultrazvučni pretvornik spojio preko uvodnice, a da se i dalje može postaviti jednostavno bez naprezanja žica izvan kutije, produžen je kabel te postavljen u vodootporno cijevno kućište. Cijevnim kućištem dubina senzora u vodi može se dodatno podesiti pri korištenju. Kako bi se postigli uniformni rezultati, dubina je za svaki senzor postavljena na 15 centimetara od dna postolja.

Kako bi se osiguralo plutanje vodenih jedinica na površini vode, bilo je potrebno osigurati primjereno postolje. Za postolje je odabran građevinski stiropor zbog svoje odgovarajuće plovnosti i niske cijene. Stiropor je odrezan tako da je za 30% veći od plastične elektroničke kutije. Time se osigurava kutija s elektronikom od preokretanja pod utjecajem jačeg vala ili kojeg drugog vodenog utjecaja. Na podlogu od stiropora također je dodana i rupa za provlačenje akustičng tranpspondera ispod površine vode. Ultrazvučni pretvornik svojim oblikom pridonosi stabilnosti sustava i sprječava moguće prevrtanje. Elektronička kutija i plastična cijev za ultrazvučni pretvornik dodatno su učvršćeni plastičnim vezicama te višenamjenskom ljepljivom trakom, kojima se dodatno osigurava krutost probnog postava.

Za postavljanje podvodne jedinice korištena je sigurnosna ronilačka bova, kojom se upozoravaju plivači na dodatni objekt u vodi. Za fiksiranje podvodne jedinice na određenu loka-

Slika 5.1: Posloženi testni postav vodenih jedinica

ciju dodan je i uteg od 1 kg, koji je imao ulogu sidra. Također je iz jednostavnosti namještanja podvodne jedinice na ovakav postav korištena platnena vreća, koja se izravno vezala špagom na ronilačku sigurnosnu bovu te na uteg.

Provedba testiranja i rezultati

Testiranje je obavljeno na na testnoj lokaciji rekreacijsko-sportskog centra Jarun u blizini Triatlon kluba Zagreb. Na tom se mjestu nalaze dva spojena privremena prilazišta za čamce i plivače. Rubovi prilazišta korišteni su za jednostavnije postavljanje vodene jedinice te lakše namještanje lokacije podvodne jedince. Dvije sekundarne jedinice montirane su na opisan način. Za razliku od sekundarnih jedinica glavna jedinica pričvršćena je na dodatni uteg za održavanje stalne pozicije te je postavljena na središte između dva prilazišta. Podvodna jedinica postavljena je u središte trokuta nastalog između vodenih jedinica kako bi se osigurali jednaki uvjeti po pitanju udaljenosti za sve tri vodene jedinice. Mjerenja su započeta nakon postavljanja vodenih jedinica na njihove konačne lokacije te su spremljena lokalno zajedno s vizualizacijom trenutačne lokacije. Pri završetku snimanja podataka napravljena je analiza rezultata.

Iz očitanih rezultata uočeno je da vodene jedinice nisu bile u mogućnosti dobiti poruku odaziva od podvodne jedinice pri slanju zahtjeva s kopna. Potencijalni je uzrok nenapunje-

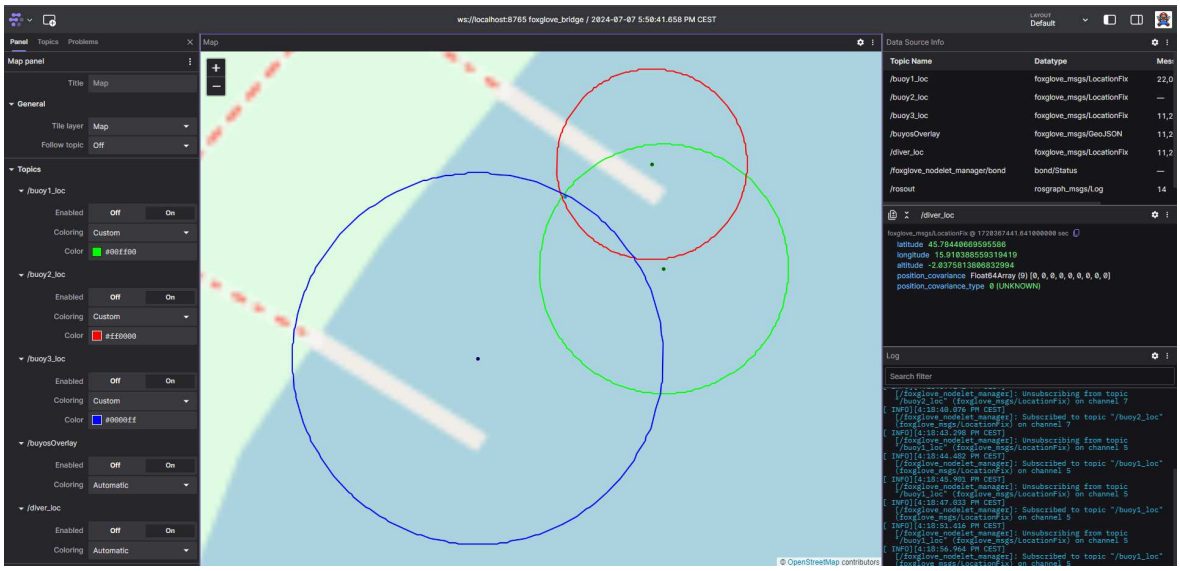
Slika 5.2: Testni postav vodenih jedinica na lokaciji testiranja

Slika 5.3: Vodene jedinice postavljene na svoje lokacije

nost interne baterije podvodne jedinice. Zbog toga podvodna jedinica nije mogla odgovoriti na *ping* signal vodene jedinice. Također se moglo primijetiti da akustična jedinica podvodne jedinice ne smije biti u kontaktu s podlogom ili s nekim drugim objektom pod vodom. Kada je vodena jedinica u početnoj fazi testiranja bila postavljena na samo dno jezera, podatci koji su dolazili na obalu uvijek su pokazivali da se ne može izvršiti *ping* odaziv. Nakon pomicanja podvodne jedinice s dna jezera te promjene baterijskog paketa, podvodna jedinica počela je primati poruke kao što je i predviđeno. Prije navedenih ispravaka nije bilo moguće postići da sve tri vodene jedinice dobiju odaziv na svoj poslani *ping* zahtjev. Iz tog razloga nije se mogla aproksimirati lokacija ronioca.

Primjer podataka koji su dolazili pri ovakvoj pogrešci:

```
[CU] Sending automatically '$G,007,RNG' to Master Bouy...
[RNG] B1,LOG,Timeout,45.784473419189453,15.910343170166014,100
[RNG] B2,LOG,Timeout,45.784446716308595,15.910502433776856,100
[RNG] B3,LOG,Timeout,45.784275054931642,15.910306930541993,100
[RNG] T,9100
[CU] Sending automatically '$G,007,RNG' to Master Bouy...
[RNG] B1,LOG,Timeout,45.784473419189453,15.910343170166014,100
[RNG] B2,LOG,Timeout,45.784446716308595,15.910502433776856,100
[RNG] B3,LOG,Timeout,45.784275054931642,15.910306930541993,100
[RNG] T,9100
[CU] Sending automatically '$G,007,RNG' to Master Bouy...
[RNG] B1,LOG,Timeout,45.784469604492183,15.910343170166014,100
[RNG] B2,LOG,Timeout,45.784446716308595,15.910502433776856,100
[RNG] B3,LOG,Timeout,45.784275054931642,15.910306930541993,100
```



Slika 5.4: Prikaz izmjerene lokacije ronioca te bova u *Foxglove Studio* sučelju

```
[RNG] T,9100
```

Jedan je od problema koji su primijećeni kod testiranja vezan uz nepreciznost GPS sustava na vodenim jedinicama. Za prototip sustava korišten je modul *Adafruit Ultimate GPS V3* s deklariranom preciznošću od 1,8 m. Vodene jedinice međusobno su bile udaljene 20 metara, što je malena udaljenost uzimajući u obzir za svega red veličine manju deklariranu preciznost GPS sustava. Zbog ovako male preciznosti implementiranog GPS sustava koji predaje sustavu lokacije sa greškama velikih razmjera, aproksimacija lokacije ronioca je posljedično time manje precizna. Iz izračunatih podataka može se uočiti da je aproksimirana lokacija ronioca pomaknuta dva do četiri metra ovisno o trenutačnoj GPS lokaciji vodenih jedinica. Kako bi se u budućnosti spriječile ovakve devijacije, potrebno je iskoristiti precizniji GPS sustav kao što je RTK (engl. *Real Time Kinematics*).

Procjena je dubine ronioca dobivena iz podataka nakon dodatnog potvrđivanja s ronilačkim kompjuterom s vlastitim dubinomjerom ispala točna unutar 50 centimetara. To je odličan rezultat, ali zato i nerealističan zbog loše preciznosti korištenog GPS sustava i posljedično time i loše aproksimacije. Ovoliku dobivenu preciznost možemo smatrati čistom slučajnošću. Kako se koristi sustav minimizacije funkcije kvadratne pogreške za određivanje estimacije lokacije i dubine ronioca, treba uzeti u obzir i to da se pogreška obje koordinatne osi lokacije značajno povećava. Upravo iz tog razloga ako se koristi neprecizan GPS modul u kombinaciji s akustičnom jedinicom puno veće preciznosti, očekivano je da će se pokazivati velike greške kod aproksimacije. Također se uzima u obzir da dobivena dubina značajno ovisi o kutu pravca koji prolazi podvodnom i vodenom jedinicom i pravca površine vode. Što je taj kut veći to je aproksimacija lošija jer više ovisi izmjerenoj GPS lokaciji. Ovo pokazuje da je prvi sljedeći korak unaprjeđenja ovog sustava poboljšanje GPS sustava.

6. Zaključak

U ovom radu nadograđen je postojeći sustav za praćenje lokacije ronioca u stvarnom vremenu. Sustav se sastoji od više funkcionalnih jedinica koje primjenom odgovarajućih komunikacijskih protokola i tehnologija lokalizacije omogućuju praćenje ronioca pod vodom. Jedinice koje čine sustav su kopnena jedinica, skup vodenih jedinica i podvodna jedinica. Kopnena jedinica služi za upravljanje sustavom i vizualizaciju podataka, dok se podvodna jedinica montira na samog ronioca. Skup vodenih jedinica predstavlja ključni podsustav za lokalizaciju ronioca i takvih jedinica je potrebno više kako bi se mogla estimirati njegova lokacija. Lokalizacija vodenih jedinica obavlja se putem GPS-a, jer su jedinice postavljene na plutačama do kojih dopire GPS signal. Ispod vode se koristi ultrazvučna komunikacija između više vodenih jedinica i podvodne jedinice kako bi se algoritmima multilateracije mogla procijeniti lokacija ronioca.

U okviru rada provedena je integracija svih sklopovskih komponenti i razvijena dodatna programska potpora kako bi se dobila jedna funkcionalna cjelina koja se može koristiti u stvarnim uvjetima. Za komunikaciju između jedinica korišteni su protokoli LoRa i ESP-NOW. Od sklopovskih modula korišteni su GPS modul Adafruit Ultimate GPS V3, LoRa modul SX1276, akustični podvodni sustav Succorfis Delphis V3 i mikrokontroler ESP32-WROOM-32D. Programska potpora za mikrokontroler razvijena je korištenjem programskih okvira Arduino i PlatformIO, što je omogućilo brz prototipni razvoj i postizanje tražene funkcionalnosti sustava. Kako bi se olakšala upravljivost i poboljšalo korisničko iskustvo, na kopnenoj upravljačkoj jedinici implementirana je vizualizacija lokacije ronioca pomoću razvojne okoline ROS i programa za vizualizaciju Foxglove. Za procjenu lokacije ronioca korišten je L-BFGS-B algoritam, koji je implementiran u programskom jeziku Python koristeći biblioteku SciPy. Nakon uhodavanja u laboratorijskim uvjetima, sve komponente sustava adekvatno su pripremljene za terenska ispitivanja u stvarnim uvjetima.

Ispitivanjem je potvrđena funkcionalnost i ispravnost rada sustava, ali su također uočeni i određeni nedostaci koje je moguće unaprijediti u budućim inačicama sustava. S obzirom da je cilj postići što bolju lokalizaciju ronioca pod vodom, zaključeno je da je glavni izvor razmjerno velike pogreške lokalizacije niska preciznost GPS modula. Za postizanje bolje preciznosti bilo bi preporučljivo koristiti naprednije GNSS sustave, poput RTK (engl. *Real-*

Time Kinematic), jer takvi sustavi omogućuju pozicioniranje s centimetarskom preciznošću. Također bi bilo preporučljivo izraditi vlastite tiskane pločice na kojima bi se na kompaktan način integrirale sve cjeline koje su u ovom prototipnom rješenju bile realizirane modulima, kako bi se postigla veća robusnost i omogućila bolja zaštita od prodora vode. Programsku potporu bilo bi potrebno iz Arduino/PlatformIO okoline prebaciti na drugačiju, koja se uobičajeno koristi u industrijskim i profesionalnim aplikacijama, npr. ESP-IDF programski okvir za ESP32 mikrokontrolere. Razlog je veća robusnost programske potpore s najnovijim ispravcima pogrešaka kod pisanja upravljačkih programa za periferiju mikrokontrolera te više mogućnosti prilikom izrade i konfiguriranje sustava.

Sustav ima veliki tržišni potencijal zbog manjka preciznih i lako upotrebljivih sustava za praćenje lokacije ronioca. Trenutno dostupni sustavi uglavnom se temelje na samostalnom korištenju preciznih GPS sustava, koji se nalaze na površini vode i povezani su preko žice s roniocem. Ovakvo rješenje, iako funkcionalno i jednostavno, teško skalira na veliki broj ronioca na istom području. Rješenje također ograničava ronioca na određenu dubinu. Postoje također sustavi temeljeni na podvodnoj robotici koji koriste podvodno autonomno vozilo, kojim se pomoću senzora poput kamere i senzora udaljenosti prate ronilac i na kopno šalju informacije o njegovoj relativnoj lokaciji. Takav sustav temelji se na računalno zahtjevnom procesiranju vanjske okoline, koje je sklono velikim pogreškama, skupo i često neprikladno u mnogim stvarnim scenarijima korištenja.

LITERATURA

[1] Adafruit ultimate gps - overview. https://learn.adafruit.com/adafruit-ultimate-gps.

[2] ESP32-WROOM-32D Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

[3] Foxglove Studio. https://foxglove.dev/.

[4] Low cost, low power acoustic communication and sensing networks. https://udrc.eng.ed.ac.uk/sites/udrc.eng.ed.ac.uk/files/attachments/UDRC%2025-03-2021.pdf.

[5] PlatformIO: An open-source ecosystem for IoT development. https://platformio.org/.

[6] ROS: Robot Operating System. https://www.ros.org/.

[7] Succorfish delphis v3 - web stranica. https://succorfish.com/products/delphis/.

[8] Sx1276 datasheet. https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf.

[9] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019.

[10] Medium. How gps works. *Aryaman Sharda*, 2018.

[11] Medium. What is lora: The fundamentals. *Vit Prajzler*, 2023.

Određivanje lokacije ronioca pomoću triangulacije signala akustičnih jedinica

Sažetak

U okviru rada nadograđen je postojeći sustav za praćenje lokacije ronioca u stvarnom vremenu. Sustav se sastoji od više funkcionalnih jedinica koje primjenom odgovarajućih komunikacijskih protokola i tehnologija lokalizacije omogućuju praćenje ronioca pod vodom. Za komunikaciju između jedinica koriste se koriste protokoli LoRa i ESP-NOW. Lokalizacija fiksnih plutajućih čvorova obavlja se korištenjem GPS-a, a ronioca ispod vode ultrazvučnim primopredajnicima. Razvijen je kompletan prototipni sustav koji se sastoji od GPS modula Adafruit Ultimate GPS V3, LoRa modula SX1276, akustičnog podvodnog sustava Succorfis Delphis V3 i mikrokontrolera ESP32-WROOM-32D, zajedno s pratećom programskom potporom. Upravljanje sustavom izvedeno je putem kopnene jedinice, na kojoj je implementirana vizualizacija lokacije ronioca pomoću razvojne okoline ROS i programa za vizualizaciju Foxglove. Za procjenu lokacije ronioca koristi se L-BFGS-B algoritam, koji je implementiran u programskom jeziku Python koristeći biblioteku SciPy. Ispitivanje sustava provedeno u realnim okolišnim radnim uvjetima.

Ključne riječi: ESP32-WROOM-32D, LoRa, SX1276, Ultimate GPS V3, ESP-NOW, akustični podvodni sustavi, Delphis V3, PlatformIO, Arduino, SciPy, L-BFGS-B, ROS, Foxglove

Determination of the diver's position by signal triangulation of acoustic units

Abstract

In this thesis an existing system for monitoring the diver's position in a real time was upgraded. The system consists of several functional units that enable underwater monitoring of divers by using suitable communication protocols and localization technologies. LoRa and ESP-NOW protocols are used for communication between the units. The fixed floating nodes are localised using GPS while diver underwater location is monitored using ultrasonic transceivers. A prototype system was developed and it includes the Adafruit Ultimate GPS V3 GPS module, the SX1276 LoRa module, the Succorfis Delphis V3 underwater acoustic system, and the ESP32-WROOM-32D microcontroller. Associated software for all system units was developed. The whole system is controlled via a coastal unit where the visualisation of the diver's position is implemented by using the ROS and the Foxglove visualisation environments. The L-BFGS-B algorithm is used to estimate the diver's position and it is implemented in the Python using the SciPy library. The system was tested under realistic environmental conditions.

Keywords: ESP32-WROOM-32D, LoRa, SX1276, Ultimate GPS V3, ESP-NOW, acoustic underwater systems, Delphis V3, PlatformIO, Arduino, SciPy, L-BFGS-B, ROS, Foxglove