UNIVERZA V LJUBLJANI
FAKULTETA ZA POMORSTVO IN PROMET

MAGISTRSKO DELO

Peter Valenčič

Video nadzor podvodnega okolja s sprotnimi meritvami temperature in slanosti

Portorož, december 2018

Mentor:

prof. dr. Vlado Malačič, univ. dipl. inž. fiz.

UNIVERZA V LJUBLJANI
FAKULTETA ZA POMORSTVO IN PROMET

MAGISTRSKO DELO

Video nadzor podvodnega okolja s sprotnimi meritvami temperature in slanosti

Mentor: prof. dr. Vlado Malačič, univ. dipl. inž. fiz.

Študent: Peter Valenčič, dipl. inženir pomorstva

Jezikovni pregled: mag. Margit Berlič Ferlinc, prof. ang. in slo.

Vpisna številka: 9150113

Študijski program: Drugostopenjski podpiplomski študij – Pomorsko inženirstvo

Smer študija: Pomorsko inženirstvo

Portorož, december 2018

Univerza v Ljubljani
Fakulteta za pomorstvo in promet

Pot pomorščakov 4
6320 Portorož, Slovenija
telefon: 05 676 71 00,
faks: 05 676 71 30
www.fpp.uni-lj.si
e-mail: dekanat@fpp.uni-lj.si

KZŠZ (2. stopnja)
Portorož, 09.11.2016
POM-PI, 2. stopnja
redni študij

Komisija za študijske zadeve (v nadaljevanju: *KZŠZ*) Fakultete *za pomorstvo in promet* Univerze *v Ljubljani* je, skladno s 64. in 151. členom *Pravil fakultete*, 10. členom *Pravilnika o drugostopenjskem študiju*, sklepom 4.3, sprejetim na 18. redni seji Senata fakultete z dne 22.12.2010 ter sklepom 6.1, sprejetim na 7. redne seje Senata fakultete z dne 19.10.2011, na svoji 1. redni seji z dne 09.11.2016 sprejela naslednji

SKLEP štev. 3.2

Kandidatu **Petru Valenčiču** (1976) se odobri tema magistrskega dela z naslovom **»VIDEO NADZOR PODVODNEGA OKOLJA S SPROTNIMI MERITVAMI TEMPERATURE IN SLANOSTI«** in potrdi se predlaganega mentorja prof. dr. Vlada Malačiča.

O b r a z l o ž i t e v

Kandidat Peter Valenčič je na predpisanem obrazcu predložil prijavo za odobritev teme magistrskega dela na drugostopenjskem podiplomskem študijskem programu Pomorstvo, usmeritev Pomorsko inženirstvo, z naslovom »VIDEO NADZOR PODVODNEGA OKOLJA S SPROTNIMI MERITVAMI TEMPERATURE IN SLANOSTI« in kot mentorja predlagal prof. dr. Vlada Malačiča.

Rok za izdelavo magistrskega dela: **09.11.2019.**

V skladu z 20. členom Pravilnika o prispevkih in vrednotenju stroškov na UL in Cenikom storitev UL FPP se zaračuna strošek zagovora magistrske naloge študentu/ - ki, ki svojega zaključnega dela ne zagovori v roku 24ih mesecev po zaključku zadnjega vpisanega letnika oziroma dodatnega leta.

Na podlagi navedenega je bilo odločiti, kot izhaja iz izreka.

Pravni pouk: Zoper sklep ima kandidat pravico do ugovora v 8 dneh od vročitve na Senat UL FPP. Ugovor je treba vložiti v dveh izvodih. Šteje se, da je ugovor pravočasen, če je oddan na pošto priporočeno zadnji dan roka za ugovor.

prof. dr. Stojan Petelin
predsednik KZŠZ

Predgovor

»Včasih lahko majhen dogodek preobrne naše življenje, če imamo srečo, da se ga zavemo, brez obotavljanja zavrnemo staro življenje in z glavo naprej strmoglavimo v novo. To se je primerilo meni tistega poletnega dne, ko sem pod morjem odprl oči.« (Jacques Yves Cousteau)

Kot otroka me je morje neprestano privlačilo. Zdelo se mi je izredno veliko in postavljal sem si nešteto vprašanj. Fasciniralo me je življenje pod morsko gladino, vonj in šum valov. Kadar sem se premražen iz morja usedel na skale in se na toplih sončnih žarkih posušil in ogrel, sem na svoji koži opazoval bele lise, ki so se v sončnih žarkih lesketale. Z jezikom sem se jih previdno dotaknil in okusil sol. Zakaj je morje slano, je bila zame največja uganka. Drugi pomemben dražljaj v morju je bila temperatura morja. Zelo dobro sem se zavedal, da je morje toplo, kjer je voda nizka in v kolikor sem se z masko potopil globlje, sem na svoji koži začutil mraz. Sklepal sem, da je morje na površini toplejše kot pri dnu zaradi sončnih žarkov, vendar tudi tega pojava nisem povsem dobro razumel. Kasneje sem se v šolskih klopeh marsičesa naučil, vendar sem se k morju nenehno vračal. Bil sem tudi aktiven podvodni lovec in kaj kmalu sem dojel, da se življenje pod morsko gladino različno obnaša in je v veliki meri pogojeno s temperaturo in slanostjo. Po zaključenem visokošolskem študiju ladijskega strojništva sem se odločil nadaljevati študij na podiplomskem študiju pomorskega inženirstva. Zaključno nalogo sem opravil na MBP NIB, kjer sem pod mentorstvom prof. dr. Vlada Malačiča razvil sistem za video nadzor podvodnega okolja s sprotnimi meritvami temperature in slanosti. Postopek izdelave merilne enote in sistema za video nadzor je podrobneje opisan v nadaljevanju naloge.

Zahvala

Zahvaljujem se mentorju, dr. Vladu Malačiču, za izčrpen pregled naloge, za nesebično pripravljenost, za sodelovanje, pomoč in usmerjanje.

Zahvaljujem se bratu Boštjanu Valenčiču, mag. Franciju Henigmanu in osebju Morske biološke postaje Piran, Nacionalnega inštituta za biologijo, za vse napotke in čas, ki so ga posvetili.

Nalogo posvečam hčerkama Klari in Špeli, ki sta mi navdih, da se trudim biti boljši in se trudim doseči zastavljene cilje.

KAZALO

SEZNAM KRATIC

KRATICA	POMEN	PREVOD/RAZLAGA
ADC	angl. Analog to Digital Converter	Analogno digitalni pretvornik
BOOTLOADER	angl. BootLoader	Zagonski program, ki se izvede ob priklopu mikrokrmilnika na napetost
COM	angl. Communication port	Komunikacijska vrata (serijska vrata)
GPIO	angl. General purpose input/output	Večnamensko vhodno/izhodno vodilo
HTML	angl. Hyper Text Markup Language	Jezik za označevanje nadbesedila – je označevalni jezik za izdelavo spletnih strani
HTTP	angl. Hyper Text Transfer Protocol	HTTP je komunikacijski protokol med odjemalcem in strežnikom
I^2C	angl. Inter integrated circuit	Komunikacijski protokol, ki ga je razvila družba Philips
ICSP	angl. In-Circuit Serial Programming	Priključek ali vodilo za programiranje mikrokontrolerja
IDE	angl. Integrated Development Environment	Programska oprema za razvoj aplikacij – razvojno okolje
JSON	angl. Javascript Object Notation	Opis Javascript objekta
JTAG	angl. Joint Test Action Group	JTAG je standard, IEEE 1149, 1. Poimenovana je tudi naprava za programiranje in razhroščevanje
LED	angl. Light Emmitind Diode	LED dioda
PWM	angl. Pulse Width Modulation	Pulzno širinska modulacija

SD	angl. Secure Digital Card	Je spominska kartica majhnega formata, ki so jo za potrebe prenosnih elektronskih naprav razvila podjetja Panasonic, SanDisk in Toshiba
SMD	angl. Surface Mount Device	Tehnologija površinske montaže elektronskih komponent
SRAM	angl. Static Random Access Memory	Statično bralno pisalni pomnilnik
TCP/IP	angl. Transmission Control Protocol	Protokol za krmiljenje prenosa
TTL	angl. Transistor-Transistor logic	Tehnologija digitalnih vezij. Pogosto uporabljamo tudi za označevanje napetostnih nivojev 0-5 V
UART	angl. Universal asynchronous receiver/transmitter	Univerzalni asinhroni sprejemnik/oddajnik
UDP	angl. User Datagram Protocol	Je nepovezovalni protokol za prenašanje paketov. Nepovezovalni pomeni, da odjemalec in strežnik ne vzpostavita povezave, ampak strežnik pošilja pakete odjemalcu in ne preverja, če je odjemalec pakete dobil.
USB	angl. Universal Serial Buss	Univerzalno serijsko vodilo
XML	angl. Extensible Markup Language	Je tričrkovna okrajšava za angleški izraz Extensible Markup Language, razširljivi označevalni jezik, in je jezik, ki ga pogosto srečamo, če brskamo po Internetu.

Povzetek

V magistrskem delu bom predstavil razvoj merilne naprave, s pomočjo katere merimo temperaturo in slanost morske vode ter z njo sočasno zajemamo in predvajamo video signal iz podvodne kamere. Slednji vsebuje tudi podatke o izmerjeni temperaturi in slanosti. Celoten sistem je sestavljen iz treh sklopov, ki so medsebojno povezani v eno napravo. Za merjenje temperature in slanosti morske vode sem uporabil merilni sondi podjetja Sea Bird Scientific. Za njihovo uporabo sem moral izdelati elektronsko vezje, ki skrbi za pretvorbo analognega signala v digitalno obliko. Vezje, ki temelji na uporabi primerjalnika (komparatorja), sem najprej razvil s pomočjo programa, ki omogoča simulacijo delovanja analognih vezij. Po opravljenih testiranjih sem nato s pomočjo programskega orodja Eagle razvil tiskano vezje in ga opremil z elektronskimi komponentami. Tako sem izdelal tiskano vezje, ki sem ga v nadaljevanju povezal na vhodno-izhodno vodilo mikrokontrolerja Arduino. Le-ta s pomočjo programa, ki sem ga razvil, skrbi za merjenje temperature in slanosti morske vode. Izmerjeni podatki se po ethernet povezavi prenašajo v nadzorni računalnik. Slednji vsebuje različne programe, s katerimi obdelujem izmerjene podatke. Izmerjeni podatki se shranjujejo v relacijsko bazo MySQL, v kateri so shranjeni surovi neobdelani podatki meritev ter drugi podatki, ki jih potrebujemo pri nastavljanju merilne naprave. Nadzorni program za pregled in prikaz podatkov je spletna aplikacija, ki je napisana z uporabo odprtokodnih knjižnic v programskem jeziku Java. Za video zajem podvodnega okolja je uporabljena ethernet kamera z visoko ločljivostjo. Ta je vstavljena v vodotesno ohišje domače izdelave. Pridobljenemu video zapisu se v realnem času s pomočjo programa dodajo tudi podatki iz temperaturne sonde in sonde za merjenje prevodnosti – slanosti. Rezultat je video posnetek, ki poleg žive slike vsebuje podatke o temperaturi in slanosti vode. Video podatki se iz računalnika pretakajo na javni strežnik za video predvajanje vsebin (YouTube).

Ključne besede: temperatura, prevodnost, slanost, Arduino, Java, mikrokrmilnik, mikroprocesor, Node.js, ščit.

Summary

In the master's thesis, I will present the development of a measuring device, by means of which we measure the temperature and salinity of seawater, and simultaneously capture and transmit the video signal from the underwater camera. The latter also contains information on the measured temperature and salinity. The entire system consists of three sets, which are interconnected in one device. To measure the temperature and salinity of seawater, I used the measurement probes of Sea Bird Scientific. For their use, I had to create an electronic circuit that takes care of converting an analog signal into a digital format. I developed a circuit based on the use of a comparator using a program that simulates the operation of analog circuits. After the tests, I then developed the printed circuit board with the help of the Eagle software tool and equipped it with electronic components. Therefore, I made a circuit board, which I later connected to the Arduino microcontroller input/output bus. With the help of the program I have developed, I take care of measuring the temperature and salinity of seawater. The measured data is transferred to the control computer via Ethernet connection. The latter contains various programs to process the measured data. Measured data is stored in the MySQL relational database, where raw data of the measurements are stored and other data that are needed when setting the measuring device. A program for viewing and displaying data is a web application that is written using open source libraries in the Java programming language. For video capture of the underwater environment, a high resolution Ethernet camera is used. This is inserted into a waterproof casing of domestic production. In addition to the obtained video, the data from the temperature probe and probe to measure the conductivity – salinity are added to the program in real time. The result is a video clip that, in addition to the live image, contains information about temperature and salinity of the water. Video data are streamed from a computer to a public server for video content playback (YouTube).

Keywords: temperature, conductivity, salinity, arduino, Java, microcontroler, microprocessor, Node.js, shield.

1 Uvod

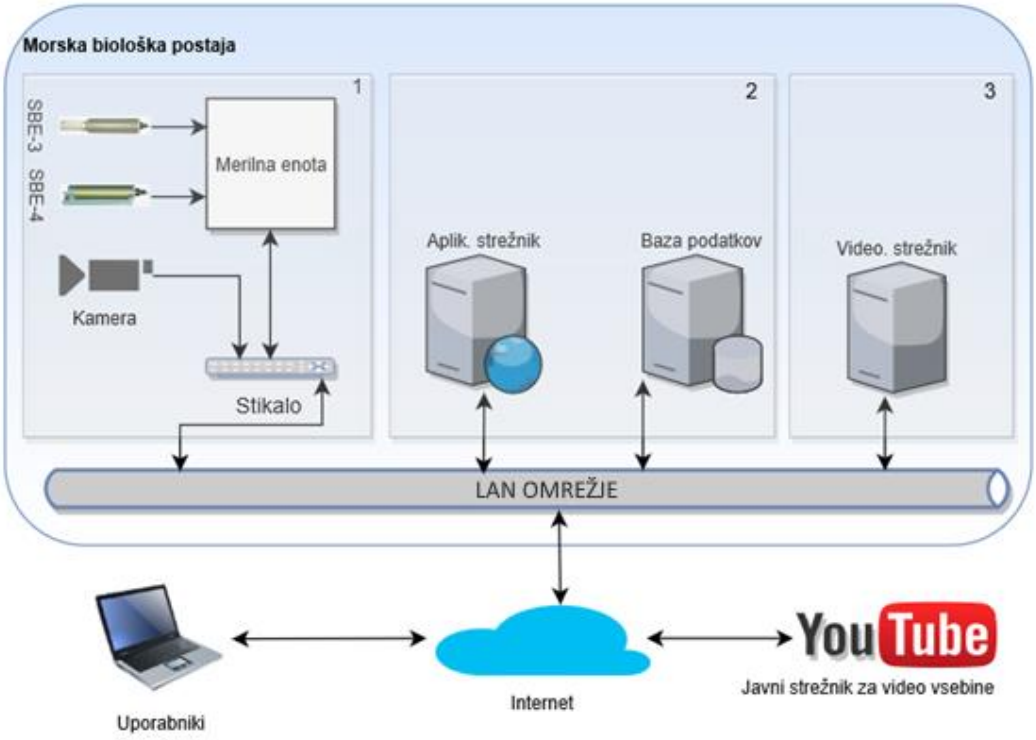
Temperatura vode močno vpliva na življenje v morju. Ta je na površini višja kot v globinah zaradi absorpcije sončne svetlobe. Če pa bi se potopili nekoliko globlje, bi nas kmalu pričelo zebsti. Voda se z globino nekaj sto metrov ohladi do 4 °C, v največjih globinah se približa tudi ledišču. V teh globinah se temperatura z globino skoraj ne spreminja. Spreminja pa se z geografsko širino. Tako imamo v področju tropskih krajev pri gladini temperaturo morja, ki dosega 30 °C, medtem ko na severnem in južne polu voda zmrzuje. Temperatura vpliva tudi na gostoto morske vode, ki pa ima vpliv na vertikalne premike voda in posledično vpliva na kemične in biološke procese v vodnem stolpcu. Poleg tega delno vpliva na koncentracijo plinov v vodi, kot sta kisik in ogljikov dioksid, ki sta močno povezana z biološkimi procesi. Zaradi velike specifične toplote vode pa ima temperatura vode v zemeljskem ekosistemu velik vpliv tudi na temperaturo ozračja. Pomembna lastnost je seveda tudi slanost morske vode. Reke so milijone let izpirale minerale s celine in jih odlagale v morje. S časom je voda izhlapevala, minerali pa so ostali. Zato je morje slano. Slanost večine svetovnih oceanov in morij niha med 33 in 35 ‰, kar pomeni, da je v približno litru morske vode (1 l vode ima maso ≈ 1023 g) raztopljenih od 33 do 35 g soli. Slanost se z izhlapevanjem vode spreminja. Čim večje je izhlapevanje in čim manjši je dotok sladke vode, tem višja je slanost. Zaradi tega je slanost severnih morij manjša, ker je izhlapevanje vode manjše in so tam padavine obilne, medtem ko so morja v subtropskem pasu zelo slana (več kot 40 ‰ soli). Slovensko morje ima približno 37 ‰ soli. Natančno merjenje temperature vode in ostalih meteoroloških podatkov ima zato velik pomen, še posebej v današnjem času, ko je zelo aktualno proučevanje klimatskih sprememb. Svetovno morje je za ljudi neprecenljiv vir dobrin. Zagotavlja nam hrano, iz njega pridobivamo surovine. Iz ležišč na njegovem dnu črpamo nafto, po morju poteka transport, na in v njem se rekreiramo in tja hodimo na oddih. To so neposredne koristi, ki nam jih daje morje. Morda se nekoliko manj zavedamo njegovega posrednega pomena za naš obstoj. S svojo ogromno maso voda oceanov bistveno vpliva na podnebje na Zemlji in igra pomembno vlogo pri kroženju ogljika, ki je v obliki ogljikovega dioksida eden najpomembnejših toplogrednih plinov.

Pisanje zaključne naloge je povezano z razvojem merilne enote, s pomočjo katere lahko merimo temperaturo in slanost morske vode. Odločil sem se, da bom razvil svojo merilno enoto s pomočjo temperaturne sonde in sonde za merjenje prevodnosti (slanosti), katero sem

dobil na izposojo na Morski biološki postaji v Piranu (MBP NIB). Del merilne enote je tudi kamera v vodotesnem ohišju, ki je namenjena zajemu podvodnega dogajanja z možnostjo sočasnega vklopa prikaza podatkov izmerjenih veličin. Sestavni del enote je tudi programska oprema, ki sem jo razvil in je v celoti dostopna na javnem repozitoriju Github (https://github.com/petervalencic). Pri razvoju programske opreme sem uporabljal odprtokodne rešitve, za kar ne potrebujemo dodatnih licenčnih pogodb in tako uporabniku omogočam prosto uporabo celotne programske kode za nadaljnji razvoj. Tako narejena merilna enota je nekakšen prototip, s katerim lahko izredno natančno in hitro beležimo spremembe temperature in prevodnosti (slanosti) morske vode. Enota bo v času testiranja postavljena v akvarijsko posodo, ki se nahaja v avli MBP NIB. Podatki pa bodo prosto dostopni na spletu.

V naslednjih poglavjih bom opisal metodo merjenja temperature in slanosti s pomočjo sond ter mikrokontrolerja in spremljanje podvodnega sveta s pomočjo kamere, ki zajema podvodno dogajanje.

2 Shema merilnega in prikazovalnega sistema razmer v morskem okolju



Vir: Lasten

Slika 1: Blokovni diagram merilnega sistema

Slika 1 prikazuje blokovni diagram merilnega sistema, ki je nameščen oziroma se nahaja v prostorih MBP NIB. Sestavljen je iz treh enot, ki so na sliki označene s številkami 1, 2 in 3. Zaradi lažjega razumevanja bom vsako izmed treh enot podrobneje opisal v nadaljevanju naloge. Posplošen opis diagrama pa lahko opišem v nekaj povedih, in sicer: v prostorih MBP NIB je postavljena akvarijska posoda, v kateri se nahajata merilni sondi SBE-3 (za merjenje temperature) in SBE-4 (za merjenje prevodnosti), ki sta s povezovalnimi kabli priključeni na merilno enoto. Prav tako se v akvarijski posodi nahaja kamera v vodotesnem ohišju. Kamera in merilna enota sta povezani z ethernet stikalom. Stikalo (angl. Switch) je povezano z lokalnim računalniškim omrežjem MBP NIB. To prikazuje enota, označena s številko ena. Enota številka dve vsebuje razvito programsko opremo za prikaz izmerjenih podatkov. Programska oprema za prikaz izmerjenih podatkov je nameščena na odprtokodnem aplikacijskem strežniku Apache Tomcat. Poleg aplikacijskega strežnika potrebujem tudi podatkovno bazo MySQL. V podatkovno bazo se zapisujejo izmerjeni podatki iz merilne enote. Tretja in zadnja enota vsebuje strežnik za obdelavo video zapisa v realnem času, v katerega se s pomočjo lastnega programa vstavljajo izmerjeni podatki temperature in slanosti.

Tako obdelan video zapis je prenešen na javni strežnik za predvajanje video vsebin »YouTube«.

2.1 Osnovno o merjenju temperature in slanosti morja

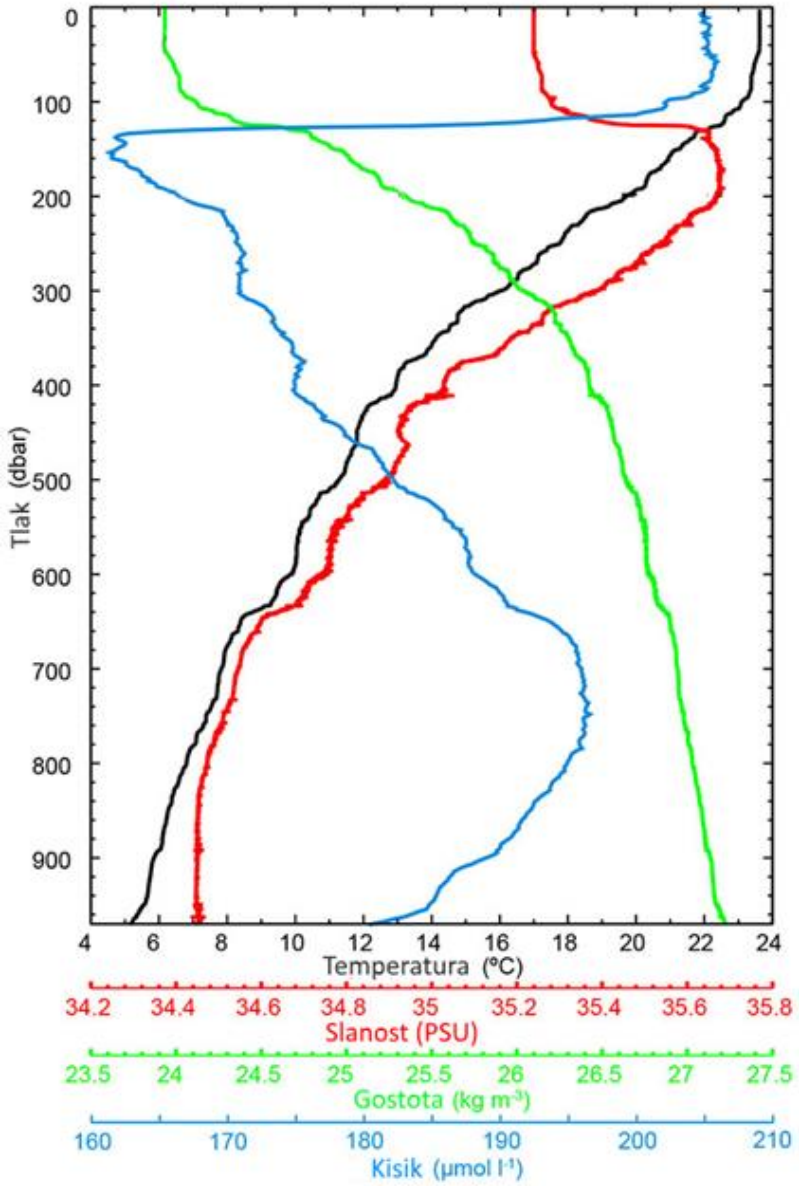
Prva merjenja temperature s pomočjo mehanskega termometra se pojavijo v 16. stoletju. Pri merjenju temperature so bili veliko časa prisotni termomehanski termometri. Kasneje z razvojem elektronskih komponent pa so termomehanske termometre zamenjali električni termometri. Električne termometre lahko z drugo besedo poimenujemo tudi temperaturna tipala. Električni termometer je naprava, ki je sestavljena iz več delov, med katere sodi temperaturni senzor, električno vezje za pretvorbo in prilagajanje signala, električno vezje za procesiranje signala in prikazovalnik izmerjene temperature. Nekateri električni termometri so lahko brez prikazovalnika, saj lahko služijo kot posredovalci informacij v druge enote ali beležijo temperaturo na druge pomnilniške medije (Strnad, 1984).

V primeru merjenja veličin v oceanografiji je eden osnovnih instrumentov t. i. CTD instrument oziroma sistem z več sondami, ki meri vsaj tri osnovne veličine. Te so prevodnost, temperatura ter tlak (globina). CTD instrument se največkrat pojavlja v obliki sonde, kot jo prikazuje Slika 2.

Vir: http://www.seabird.com/sbe911plus-ctd

Slika 2: CTD sistem Sea-Bird Scientific SBE911

CTD sonda se največkrat uporablja za pridobitev vertikalnih profilov. Enega od profilov prikazuje Slika 3. Podatek o prostorski porazdelitvi slanosti in temperature igra ključno vlogo pri gibanju morskih tokov, spremembah temperatur in še mnogih dejavnikov, ki vplivajo na morski ekosistem (Kervina G, 2016 in http://oceanexplorer.noaa.gov/facts/ctd.html, zadnjič obiskano 22. februarja 2018).



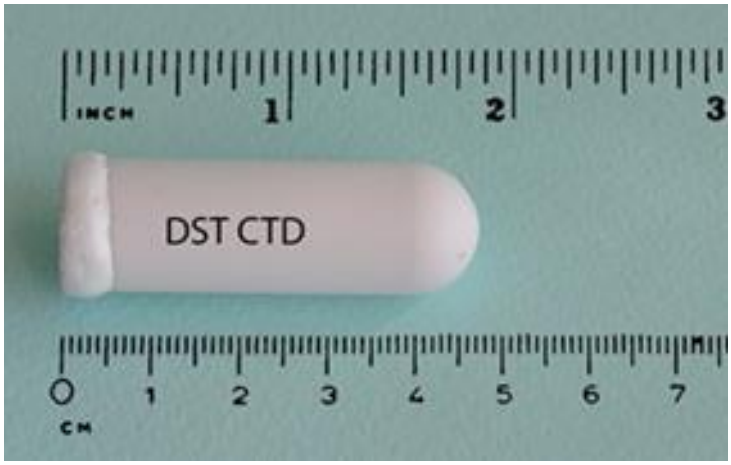
Vir: https://www.deepreef.org/technology/6-ctd.html

Slika 3: Vertikalni profil vzorčenja različnih količin (prevodnost, kisik, temperatura)

Za merjenje vertikalnih profilov se v oceanografiji uporabljajo tudi posebne Argo boje. Gre za posebne vrste boj, ki se avtonomno potapljajo v različne globine v oceanih. Dosežejo lahko različne globine, potopijo pa se lahko tudi do 2 km globoko. Po svetu jih je nekaj manj kot štiri tisoč in prosto plavajo v različnih morjih in oceanih. Opremljene so s CTD sondo za

merjenje prevodnosti, temperature in tlaka, s podobnima merilnikoma temperature in prevodnosti (podjetja Sea-Bird Scientific), ki sem ju uporabil pri nalogi. Novejše boje imajo poleg CTD sonde prisoten še GPS sprejemnik ter Iridium satelitsko telekomunikacijo, s katero posredujejo izmerjene podatke v nadzorni center ter tako bistveno prispevajo k meritvam temperature in slanosti oceanov (Kervina G, 2016 in http://www.argo.ucsd.edu/How_Argo_floats.html, zadnjič obiskano 22. februarja 2018).

Temperaturni senzorji in merilniki prevodnosti pa se v oceanografiji in ostalih vedah pojavljajo tudi v drugačnih oblikah, kot je na primer zajemalnik (angl. Logger), ki beleži prevodnost, temperaturo in tlak v vodi na fiksni lokaciji ali je pričvrščen tudi na živali. Eden takšnih je zajemalnik podjetja Star Oddi (https://www.star-oddi.com/products/ all-products/salinity-logger-meter-probe-ctd), ki je velik manj kot 5 cm. Naprava vsebuje že vgrajeno baterijo, s katero se napaja elektronsko vezje, na katerega so priključena tipala za merjenje in beleženje prevodnosti, temperature in tlaka. Izmerjeni podatki se shranjujejo v interni spomin (RAM). Kasneje jih s pomočjo zunanje naprave brezžično prenesemo v računalnik.

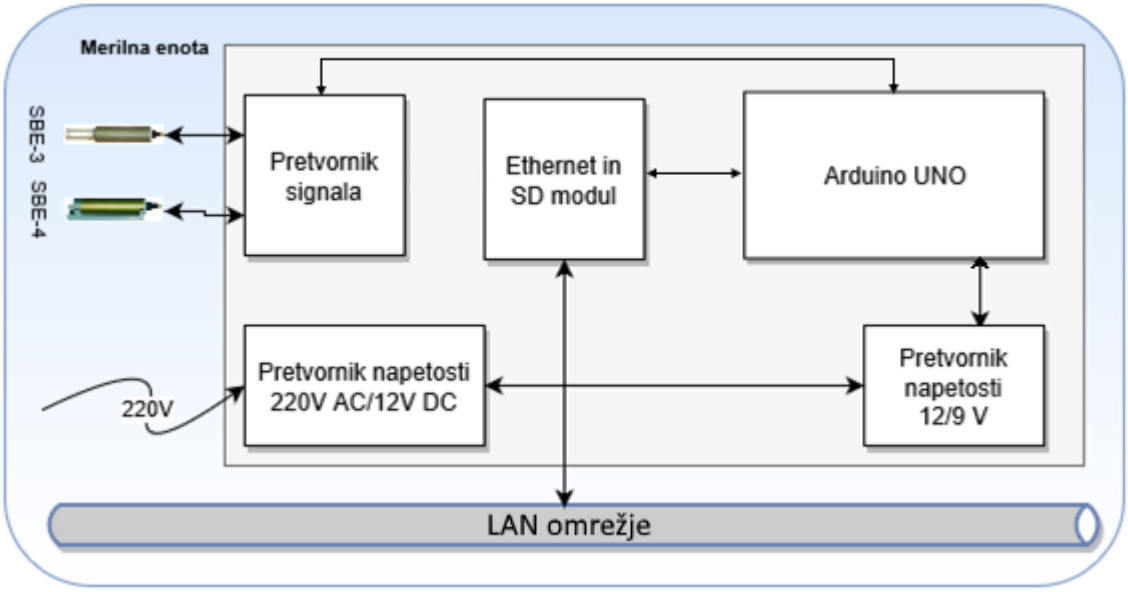


Vir: https://www.star-oddi.com/media/1/__dst-ctd.jpg

Slika 4: CTD zajemalnik podjetja Star-Oddi

3 Metoda merjenja temperature in prevodnosti morske vode s pomočjo sond ter mikrokontrolerja Arduino

Opisal in prikazal bom doma narejeno merilno enoto, s pomočjo katere je možno meriti temperaturo in prevodnost morske vode. Merilna enota uporablja analogni sondi podjetja SeaBird Scientific model SBE-3 in SBE-4. Za priklop analognih sond na mikroprocesorsko vezje sem moral razviti pretvornik, ki pretvori analogni signal določene amplitude in frekvence v digitalno obliko signala. Tako pretvorjeni signal je možno s pomočjo mikroprocesorja tudi prebrati. V merilni enoti sem uporabil razvojno platformo Arduino, model Arduino Uno. Gre za mikroprocesorsko razvojno vezje, ki je med razvijalci zelo priljubljeno zaradi enostavne uporabe. Vmesnik, ki skrbi za pretvorbo signala sond, je z vhodno-izhodnim vodilom povezan z Arduinom. Poleg tega je v enoto vključen tudi ethernet modul in modul za priklop pomnilniške kartice SD. Merilna enota je tako opremljena z ethernet priključkom, kar nam omogoča izmenjavo izmerjenih podatkov po ethernet omrežju. V ohišju merilne enote se poleg elektronskih tiskanih vezij nahaja še napajalnik, na katerega sta priključeni merilni sondi ter pretvornik napetosti za napajanje mikrokrmilnika Arduino. Z manjšimi spremembami in prilagoditvami je možno na obstoječi sistem povezati tudi druge vrste sond in senzorjev (tipal). Slika 5 prikazuje sestavne dele (komponente), ki tvorijo merilno enoto in katere bom podrobneje opisal v nadaljevanju naloge.



Vir: Lasten

Slika 5: Blokovni diagram merilne enote in njenih komponent

3.1 Sondi SBE-3 in SBE-4

Pri izvajanju meritev v oceanografiji in meteorologiji moramo upoštevati tudi posebnosti okolja. Tako ne smemo pozabiti na tlak, ki deluje na merilno sondo, ko je le-ta potopljena v morje. Ker se v oceanografiji meritve izvajajo tudi v globokih morjih (več tisoč metrov globoko), se z globino spreminja tudi tlak, ki deluje na ohišje merilne sonde. Ta se vsakih deset metrov poveča za en bar. Zaradi takšnih surovih pogojev je sonda grajena iz kovinskega ohišja. Za manjše globine je ohišje iz aluminija, medtem ko je za večje globine (>3,400 metrov) ohišje narejeno iz titana. Kovinsko ohišje preprečuje, da bi pri večjem tlaku prišlo do poškodbe (implozije).

Zaradi delovanja galvanskega toka (galvanske korozije) je na sondah prisotna tudi cinkova anoda (Slika 6). Imenujemo jo tudi »žrtvovalna« anoda. Ta je v obliki prstana in je na ohišje sonde privijačena s štirimi vijaki. Po določenem času je potrebno cinkovo anodo zamenjati z novo, saj lahko v nasprotnem primeru galvanski tok poškoduje ohišje sonde. Galvanska korozija nastane zaradi stika dveh različnih kovin v elektrolitu. Čiste kovine so razvrščene po elektrokemijski napetostni vrsti, iz katere lahko ugotovimo, katera ima večjo tendenco do oksidacije oziroma redukcije. To pomeni, da dve kovini v stiku tvorita nekakšen galvanski člen, zaradi česar manj plemenita kovina (z manjšim potencialom) v stiku s plemenitejšo kovino (z večjim potencialom) običajno hitreje korodira, kot če stika ne bi bilo. Pri tem se generira napetost, ki je zadostna za svetenje male žarnice. Omenjeni pogoji za galvanski tok, kjer sta v morski vodi prisotni različni kovini, v navtiki skoraj vedno nastanejo, česar se pogosto niti ne zavedamo (Povhe, 2017).

Vir: Lasten

Slika 6: Cinkova anoda na ohišju sonde

Na skrajnem delu sonde je prisoten priključni konektor (moški tip konektorja), ki vsebuje tri priključne sponke. Ena od treh priključnih sponk (minus napajanja) je nekoliko zadebeljena, kar preprečuje, da bi uporabnik napačno priklopil sondo s povezovalnim kablom. Sonda ima tri priključne sponke, med katerimi se dve uporabljata za napajanje sonde, tretja pa je namenjena izhodnemu signalu, ki ga s pomočjo zunanje enote merimo. Pred priklopom konektorja na sondo je potrebno preveriti, ali so priključne sponke, ki se nahajajo na sondi, čiste. Zaradi prisotnosti vlage je možna površinska oksidacija priključnih sponk. Po navodilih proizvajalca je potrebno oksidirane sponke očistiti s čisto krpo ter alkoholom. Po čiščenju moramo na sponke – kontakte nanesti tanek sloj silikonske masti. Proizvajalec odsvetuje uporabo drugih maziv, ki so na osnovi petroleja, kot je na primer WD-40. Neustrezno mazivo lahko trajno poškoduje konektor in gumijasto tesnilo, ki preprečuje vdor oziroma stik z morsko vodo. Posebno pozornost moramo posvetiti tudi privijanju konektorja na sondo. Pri tem moramo paziti, da ne uporabljamo klešč ali drugih pripomočkov, s katerimi bi lahko poškodovali plastičen navoj in tako omogočili vdor vode v notranjost konektorja. Slika 7 prikazuje primer oksidiranih priključnih sponk (http://www.seabird.com/, zadnjič obiskano 14. septembra 2017).

Vir: http://www.seabird.com/sites/default/files/documents/appnote57Jan14.pdf

Slika 7: Oksidirane priključne sponke

Pomembna lastnost merilnih sond in ostalih senzorjev je odzivni čas. Noben senzor hipoma ne zazna sprememb merjenca. Odzivni čas senzorja nam pove, kako hitro le-ta izmeri ali zabeleži novo vrednost merjene veličine. Pri hipni spremembi vrednosti merjenca ta čas pomeni, koliko časa preteče od spremembe merjene veličine do 63,2 % končne vrednosti nove. Čas vzorčenja pa nam pove, kolikšen je najmanjši čas, ki poteče med dvema

zaporednima meritvama. Želimo si, da bi bil ta čim krajši. Če odzivni čas ni dovolj kratek, nam tudi veliko krajši čas vzorčenja ni v pomoč. Pomemben podatek je tudi stabilnost. Stabilnost je sprememba senzorjevega odziva po enoletnem delovanju. Določena je iz razlike v izmerjenih karakteristikah na začetku uporabe in na koncu, pri predpisanih pogojih delovanja. Pri temperaturni sondi SBE-3 je stabilnost 0,002 °C/leto, medtem ko je pri prevodnostni sondi SBE-4 stabilnost 0,0003 S/m. Območje merjenja nam pove najnižjo in najvišjo vrednost, katero lahko s pomočjo sonde izmerimo. Te podatke pridobimo v tehničnih specifikacijah, ki so za vsako sondo različni (http://lms.fe.uni-lj.si/amon/knjiga/Senzorji_in_aktuatorji.pdf, zadnjič obiskano 26. februarja 2018).

3.1.1 **Temperaturna sonda SBE-3**

Za merjenje temperature morske vode sem uporabil temperaturno sondo proizvajalca Seabird, model SBE-3 (Slika 8). Namenjena je merjenju temperature morja, lahko pa jo uporabljamo tudi v industrijskih procesih, kjer se zahteva visoka natančnost merjenja. Sonda je podolgovate oblike. Na spodnji strani sonde se nahaja tipalo (termistor). Ta je vstavljen v kovinski zaščitni tulec iz nerjaveče kovine. Termistor je dodatno obdan s kovinsko zaščitno kletko. Ta preprečuje, da bi poškodovali zelo občutljivo tipalo. Kovinsko kletko lahko zamenjuje tudi plastičen tulec, ki ima dva priključka za dovod morske vode s pomočjo črpalke. V ohišju sonde se nahaja prilagoditveno vezje, ki vsebuje Wienov mostič (angl. Wien bridge oscilator). Izhod oscilatorja je priključen na priključne sponke sonde. Sonda na izhodu generira sinusni signal. Frekvenca signala je odvisna od izmerjene temperature. Merilno območje temperaturne sonde je med –5 °C in +35 °C, kar pri SBE-3 pomeni izhodni signal frekvence med 2 kHz in 6 kHz ter amplitudo signala 0,5 V. (http://www.seabird.com/sites/default/files/documents/03sbrochureMay15.pdf, zadnjič obiskano 26. februarja 2018).

Vir: http://www.seabird.com/sites/default/files/documents/03sbrochureMay15.pdf

Slika 8: SBE 3 – sonda za merjenje temperature

SBE-3 ima aluminijasto ohišje za merjenje do globine 3400 m, težo na kopnem: 0,6 kg, težo v vodi: 0,3 kg.

Tehnične lastnosti sonde:

- območje merjenja: –5 °C do +35 °C
- točnost: ± 0,001 °C
- stabilnost: 0,002 °C (na leto)
- odzivni čas: 0,58 s
- napaka zaradi lastnega segrevanja: < 0,0001 °C

Električne lastnosti:

- napajanje: enosmerna napetost 11 V do 16 V
- izhodna napetost: sinusni signal amplitude 0,5 V

Izmerjeno temperaturo s pomočjo sonde pridobimo po izrazu (1). Tega navaja proizvajalec sonde v navodilih za uporabo, dostopnih na spletni strani: http://www.seabird.com/sbe3s-temperature-sensor (zadnjič obiskano 27. februarja 2018). Poleg temperaturne sonde prejme uporabnik kalibracijski list, na katerem dobi zapisane koeficiente, ki jih mora pri preračunu temperature upoštevati. Temperaturno sondo mora uporabnik periodično (vsaki dve leti) pošiljati v kalibracijski center proizvajalca, kjer sondo ponovno umerijo (kalibrirajo). S kalibracijskim listom proizvajalec jamči, da je sonda pripravljena za uporabo in da ne odstopa od vrednosti, ki so navedene v tehničnih specifikacijah sonde.

Preračun temperature T v °C iz frekvence f v Hz dobimo z izrazom:

$$T=\frac{1}{\left\{\mathrm{G}+\mathrm{H}\left(\ln\left(\frac{F0}{f}\right)\right)+\mathrm{I}\left(\ln^2\left(\frac{F0}{f}\right)\right)+\mathrm{J}\left(\ln^3\left(\frac{F0}{f}\right)\right)\right\}-273.15}, \quad (1)$$

kjer so koeficienti G= 4,85442486e-003, H= 6,77400494e-004, I= 2,65502731e-005, J= 2,06782794e-006 in F_0=1000,00.

3.1.2 **Sonda za merjenje prevodnosti SBE-4**

Sondo modela Seabird SBE-4 prikazuje Slika 9, namenjena je merjenju prevodnosti vode. Ohišje sonde je podolgovate oblike in izdelano iz aluminija. Po obliki spominja na sondo SBE-3, vendar je nekoliko drugače zasnovano. Senzor za merjenje prevodnosti je vstavljen v stekleno cevko. Ta je zaščitena z aluminijastim profilom v obliki črke U, ki je privijačen na ohišje sonde. Steklena cevka je oblikovana tako, da nanjo lahko priključimo cev za dovod vode v področje senzorja. V ohišju sonde se nahaja prilagoditveno vezje, ki ga sestavlja Wienov oscilator. Izhod oscilatorja je priključen na priključne sponke sonde. Sonda na izhodu generira sinusni signal. Frekvenca signala je odvisna od izmerjene prevodnosti. Merilno območje sonde je med 0,0 S/m do 7,0 S/m, kar se odraža v izhodnem signalu frekvence med 2,5 kHz in 7,5 kHz ter amplitudo signala 0,5 V (http://www.seabird.com/sbe4-conductivity-sensor, zadnjič obiskano 26. februarja 2018).

Vir: http://www.seabird.com/sbe4-conductivity-sensor

Slika 9: SBE 4 – sonda za merjenje prevodnosti vode

Ohišje sonde SBE-4 ima podobne lastnosti kot SBE-3.

Tehnične lastnosti sonde:

- območje merjenja: 0,0 S/m do 0,7 S/m
- točnost: ± 0,0003 S/m
- stabilnost: 0,0003 S/m (na mesec)
- odzivni čas: 0,06 s

Električne lastnosti:

- napajanje: enosmerna napetost 6 V do 24 V
- izhodna napetost: sinusni signal amplitude 0,5 V

Prevodnost morske vode pridobimo iz izmerjene frekvence po izrazu (2). Podrobnosti preračuna prevodnosti iz izmerjene frekvence so objavljene na spletni strani proizvajalca (http://www.seabird.com/ sbe4-conductivity-sensor, zadnjič obiskano 27. februarja 2018). Tudi sondi za merjenje prevodnosti je priložen kalibracijski list, na katerem so zapisani koeficienti. Le-te moramo upoštevati v izrazu za preračun prevodnosti. Poleg koeficientov kalibracijskega centra moramo pri preračunu prevodnosti upoštevati trenutno izmerjeno temperaturo vode ter tlak.

Preračun prevodnosti *S* v S/m iz frekvence *f* v Hz, temperature *t* v °C ter tlaka *p* v dBar.

$$S = \frac{(\mathrm{G}+\mathrm{H}f^2+\mathrm{I}f^3+\mathrm{J}f^4)}{10(1+\mathrm{d}t+\mathrm{e}p)}, \quad (2)$$

kjer so koeficienti G= -1,05697877e+01, H= 1,42707291e+00, I= -4,32023820e-03, J= 4,53455585e-04, d= 3,25e-06 in e= -9,57e-08.

3.1.3 **Izračun slanosti**

Morska voda predstavlja mešanico 96,5 % čiste vode (H_2O) ter 3,5 % drugih snovi, med katerimi so soli, raztopljeni plini, organske substance in neraztopljene nečistoče. Slanost je značilna in pomembna lastnost morske vode. Predstavlja merilo za vsebnost vseh soli v gramih, ki so raztopljene v 1 kg slanice. Navadno je izražena v odstotkih (%) ali promilih (‰), ki ustrezajo utežnemu deležu vseh soli v skupni masi raztopine; slanost 3 % ali 30 ‰ pomeni, da je v 1000 g vodne raztopine (slane vode) raztopljenih 30 g vseh soli.

Slanost merimo tudi s pomočjo električne prevodnosti morske vode in je odvisna od prostorninske koncentracije raztopljenih ionov in njihovega gibanja, ki je pogojeno s temperaturo (*T*) in tlakom (*P*). Izražena je z enoto S/m. Absolutna slanost morske vode (*Sa*) je razmerje med maso vseh raztopljenih snovi v morski vodi in maso morske vode. Absolutne slanosti v praksi ne merimo neposredno. Zato so s slanostjo (*S*) morske vode nekdaj označevali skupno količino vseh trdnih snovi, ki so raztopljene v morski vodi, pri čemer so vsi karbonati pretvorjeni v okside, bromidi in jodidi v kloride, organska snov pa je popolnoma oksidirana. Po letu 1979 izražajo slanost skoraj izključno kot praktično slanost (*Sp*), ki temelji na merjenju električne prevodnosti morske vode. Definiramo jo kot razmerje električne prevodnosti morske vode pri temperaturi 15 °C in tlaku 1 bara glede na prevodnost raztopine KCl, kjer je utežni delež KCl enak 32,4356e10-3 (Unesco Technical Papers in Marine Science, str. 12) pri isti temperaturi in tlaku (K_{15}). V primeru, ko je $K_{15} = 1$, je praktična slanost *Sp* = 35.

Praktično slanost (*Sp*) definiramo z uporabo razmerja K_{15} s sledečim izrazom:

$$S_p = \mathrm{a}_0 + \mathrm{a}_1 K_{15}^{\frac{1}{2}} + \mathrm{a}_2 K_{15} + \mathrm{a}_3 K_{15}^{\frac{3}{2}} + \mathrm{a}_4 K_{15}^2 + \mathrm{a}_5 K_{15}^{\frac{5}{2}} \quad (3)$$

Konstante so:

$a_0 = 0{,}008;\ a_1 = -0{,}1692;\ a_2 = 25{,}3851;\ a_3 = 14{,}0941;\ a_4 = -7{,}0261;\ a_5 = 2{,}7081$

Izraz velja za *Sp* med 2 in 42 in je v merskem sistemu SI brez enote.

Praktično slanost *Sp* za izmerjeno (drugačno od 15 °C) temperaturo, prevodnost in tlak (drugačen od 1 bara) izračunamo po izrazu (4), ki je podan v (Unesco Technical Papers in Marine Science 30, 1978, str. 17–18) in velja za temperaturno območje med –2 °C in 35 °C:

$$S(‰) = a_0 + a_1 R_T^{\frac{1}{2}} + a_2 R_T + a_3 R_T^{\frac{3}{2}} + a_4 R_T^2 + a_5 R_T^{\frac{5}{2}} + \frac{(T-15)}{1+A(T-15)}\left[b_0 + b_1 R_T^{\frac{1}{2}} + b_2 R_T + b_3 R_T^{\frac{3}{2}} + b_4 R_T^2 + b_5 R_T^{\frac{5}{2}}\right] \quad (4)$$

Pri čemer velja:

$\sum_{i=0}^{5} a_i = 35{,}000$, kjer so konstante:

$a_0 = 0{,}008;\ a_1 = -0{,}1692;\ a_2 = 25{,}3851;\ a_3 = 14{,}0941;\ a_4 = -7{,}0261;\ a_5 = 2{,}7081$ (enako kot za izraz ((3)).

$\sum_{i=0}^{5} b_i = 0$, kjer so konstante:

$b_0 = 0{,}0005;\ b_1 = -0{,}0056;\ b_2 = -0{,}0066;\ b_3 = -0{,}0375;\ b_4 = 0{,}0636,\ b_5 = -0{,}0144,$

A = 0,0162, ter razmerje prevodnosti:

$$R_T = \frac{C(S,T,0)}{C(35,T,0)} \quad (5)$$

Pri izračunu slanosti, kjer merimo prevodnost $C(T, S, p)$ in ne razmerja prevodnosti, potrebujemo še podatek o prevodnosti za $C(T$=15 °C, S= 35, p= 0) = 4,2914 S/m (Unesco Technical Papers in Marine Science 30, 1981, str. 27–28, http://salinometry.com/pss-78/, zadnjič obiskano 16. maja 2018, http://www.kpss.si/o-parku/soline-in-solinarstvo/morje_2/slanost, zadnjič obiskano 17. maja 2018, Malačič V., 2015).

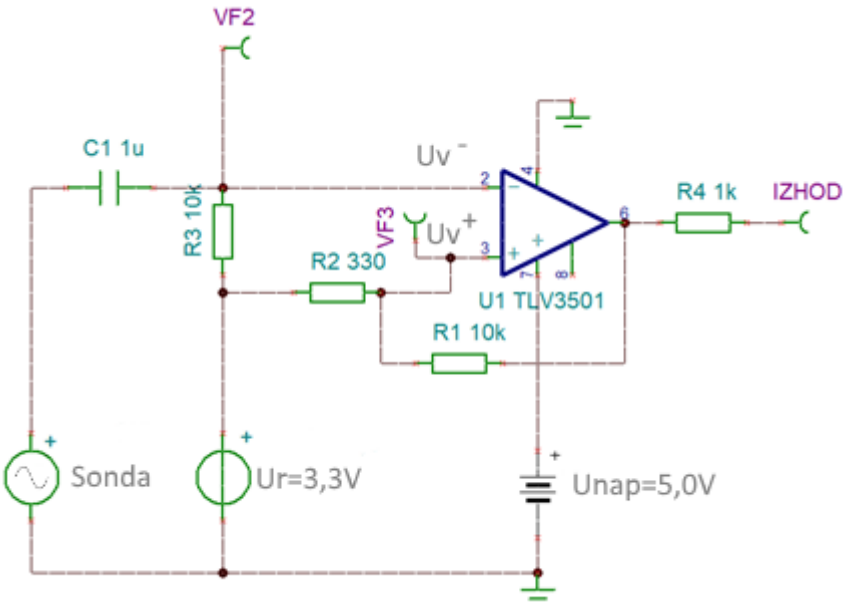
Programska funkcija, ki sem jo uporabil za pretvorbo prevodnosti v slanost, je zapisana v: »Dodatek A-Funkcija za pretvorbo prevodnosti v slanost« na strani 84.

3.2 Pretvornik signala v zaporedje impulzov

Pretvornik signala je električno vezje, ki ima v mojem primeru dva analogna vhoda in dva digitalna izhoda. Z njim želimo pretvoriti sinusni signal, katerega ustvarjata SBE-3 in SBE-4 sondi, v zaporedje enakih pulzov, ki ustrezata standardu TTL in jih je možno izmeriti z uporabo mikrokontrolerja. Pulze z mikrokontrolerjem lahko v eni sekundi preštejem in dobim frekvenco posamezne sonde. Da bi to dosegel, sem moral razviti elektronsko vezje. Pri razvoju prototipa vezja in kasneje tiskanega vezja sem izhajal iz zapiskov o delovanju analognih vezij, dosegljivih na spletnem naslovu (http://fides.fe.uni-lj.si/~arpadb/aev/knjiga-aev.pdf, zadnjič obiskano 27. februarja 2018), med katere spadajo operacijski ojačevalniki in komparatorji (primerjalniki) napetosti. Za razumevanje delovanja pretvornika je potrebno osnovno znanje o delovanju analognih elektronskih komponent in vezij. Podroben opis delovanja komparatorja in bistabilnega komparatorja je v dodatku B »Komparator, delovanje in uporaba« na strani 85.

3.2.1 Pretvorba signala sonde s pomočjo bistabilnega komparatorja

V nalogi sem za pretvorbo signala iz sonde uporabil invertirajoči bistabilni komparator z uporabo referenčne napetosti (*Ur*), ki je prikazana na električni shemi spodaj.



Vir: Lasten

Slika 10: Pretvornik signala

Delovanje vezja je preprosto. Najprej si oglejmo komparator, ki je v električni shemi označen z U1. Ta ima invertirajoči vhod Uv^- (sponka št. 2) ter neinvertirajoči vhod Uv^+ (sponka št. 3). Sponka št. 7 (*Unap* je povezana na enosmerno napetost 5,0 V, medtem ko je sponka št. 4 povezana na ozemljitev (0 V). S tema dvema napetostma določimo našo izhodno napetost na komparatorju (sponka št. 6) in zagotovimo sledeče: dokler bo napetost na sponki št. 2 večja od napetosti na sponki št. 3, bo na izhodu komparatorja napetost. Dokler bo napetost na sponki št. 2. manjša od napetosti na sponki št. 3, bo napetost na izhodu komparatorja 5,0 V. Ker pa je napetost signala iz sonde premajhna (±0,5 V), jo je potrebno ustrezno dvigniti na višjo primerjalno napetost. To storimo tako, da referenčno napetost z uporom R_3 povežemo s signalom sonde. S tem dosežemo seštevanje dveh napetosti, napetost iz sonde (±0,5 V) seštejemo z referenčno napetostjo 3,3 V. Tako dobimo sinusno obliko napetosti U_{vh}, ki niha med 2,8 V in 3,8 V. Zavedati se moramo, da napetost na neinvertirajočem vhodu ni konstantna. Takoj ob vklopu naprave je izhodna napetost na komparatorju U_B = 0 V, zato je napetost na neinvertirajočem vhodu enaka referenčni napetosti 3,3 V. V trenutku, ko napetost na neinvertirajočem vhodu pade pod napetost 3,195 V, se bo na izhodu komparatorja pojavila napetost U_A = 5,0 V. S povratno povezavo z uporom R_1 napetost iz izhoda komparatorja prehaja na neinvertirajoči vhod komparatorja. Tako je nova napetost, ki se pojavi na neinvertirajočem vhodu, velikosti 3,354 V. Ko bo v naslednjem trenutku napetost na invertirajočem vhodu komparatorja presegla vrednost 3,354 V, se bo na izhodu komparatorja pojavila napetost U_B = 0 V.

Tok I skozi upora R_1 in R_2 je določen z

$$I = \frac{Uv^+ - Ur}{R_2} = \frac{Uiz - Uv^+}{R_1} \tag{6}$$

Napetost na vhodu je določena z

$$Uv^+\left(\frac{1}{R_2} + \frac{1}{R_1}\right) = \frac{Uiz}{R_1} + \frac{Ur}{R_2} \rightarrow Uv^+ = \frac{U_R R_1}{(R_1 + R_2)} + \frac{U_{iz} R_2}{(R_1 + R_2)} \tag{7}$$

Dokler je napetost $Uv < Uv^+$, je izhodna napetost $U_{iz} = U_A$.

Preskok iz U_A v nizki nivo U_B se bo zgodil pri $Uv = U_{TA}$

$$U_{TA} = \frac{U_R R_1}{(R_1 + R_2)} + \frac{U_A R_2}{(R_1 + R_2)} \tag{8}$$

Prehod z nizkega nivoja $U_{iz} = U_B$ na visoki nivo pa se zgodi pri

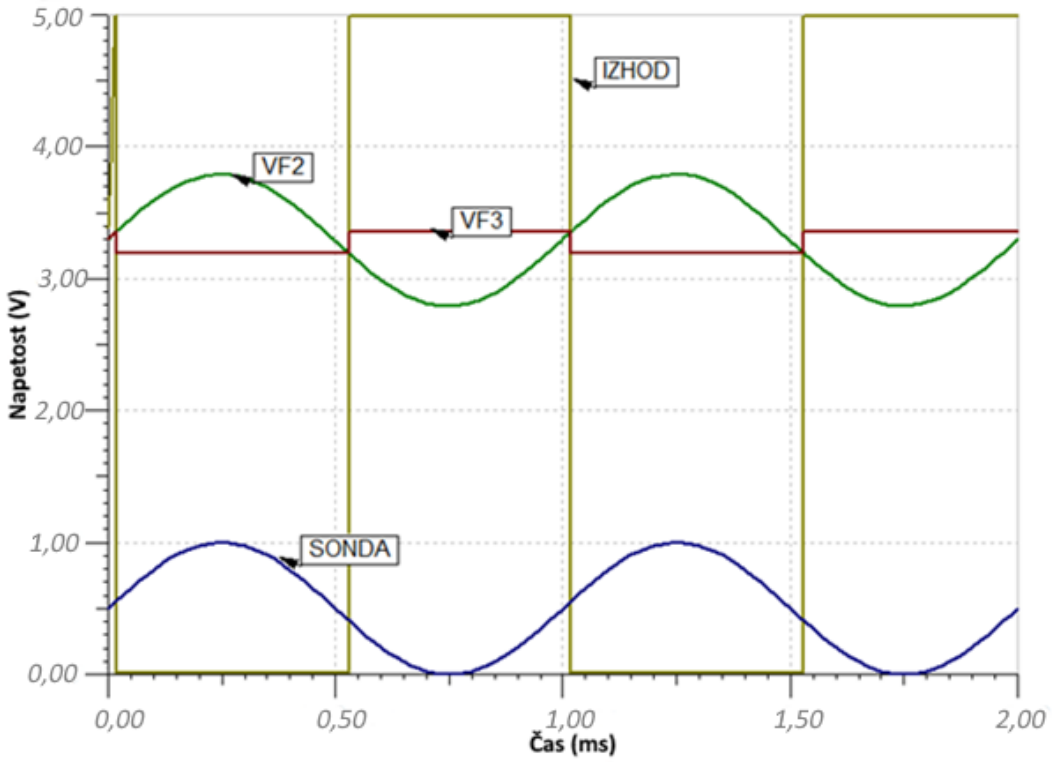
$$U_{TB} = \frac{U_R R_1}{(R_1+R_2)} + \frac{U_B R_2}{(R_1+R_2)} \quad (9)$$

Vrednosti napetosti U_{TA} in U_{TB} sta pri uporih R_1 = 10 kΩ, R_2 = 330 Ω in napetosti U_R = 3,3V in U_A = 5,0 V in U_B = 0 V.

$$U_{TA} = \frac{U_R R_1}{(R_1+R_2)} + \frac{U_A R_2}{(R_1+R_2)} = 3{,}354 \text{ V}$$

$$U_{TB} = \frac{U_R R_1}{(R_1+R_2)} + \frac{U_B R_2}{(R_1+R_2)} = \frac{U_R R_1}{(R_1+R_2)} = 3{,}195 \text{ V}$$

Slika 11 prikazuje izris napetosti v našem vezju, kjer je razvidna napetost iz sonde (modre barve) napetost na invertirajočem vhodu komparatorja VF2 (zelene barve), ki je sešteta z referenčno napetostjo 3,3V, napetost na neinvertirajočem vhodu komparatorja VF3 (rdeče barve) ter izhodni signal, ki je razviden kot pravokotni impulz amplitude 5,0 V in je označen z »IZHOD« (rjave barve). Pravokotni impulz na izhodu komparatorja se pojavi natanko enkrat v periodi. Takšen signal kasneje z uporabo mikrokontrolerja tudi preštejemo (preštejemo število impulzov v eni sekundi) in tako dobimo frekvenco.



Vir: Lasten

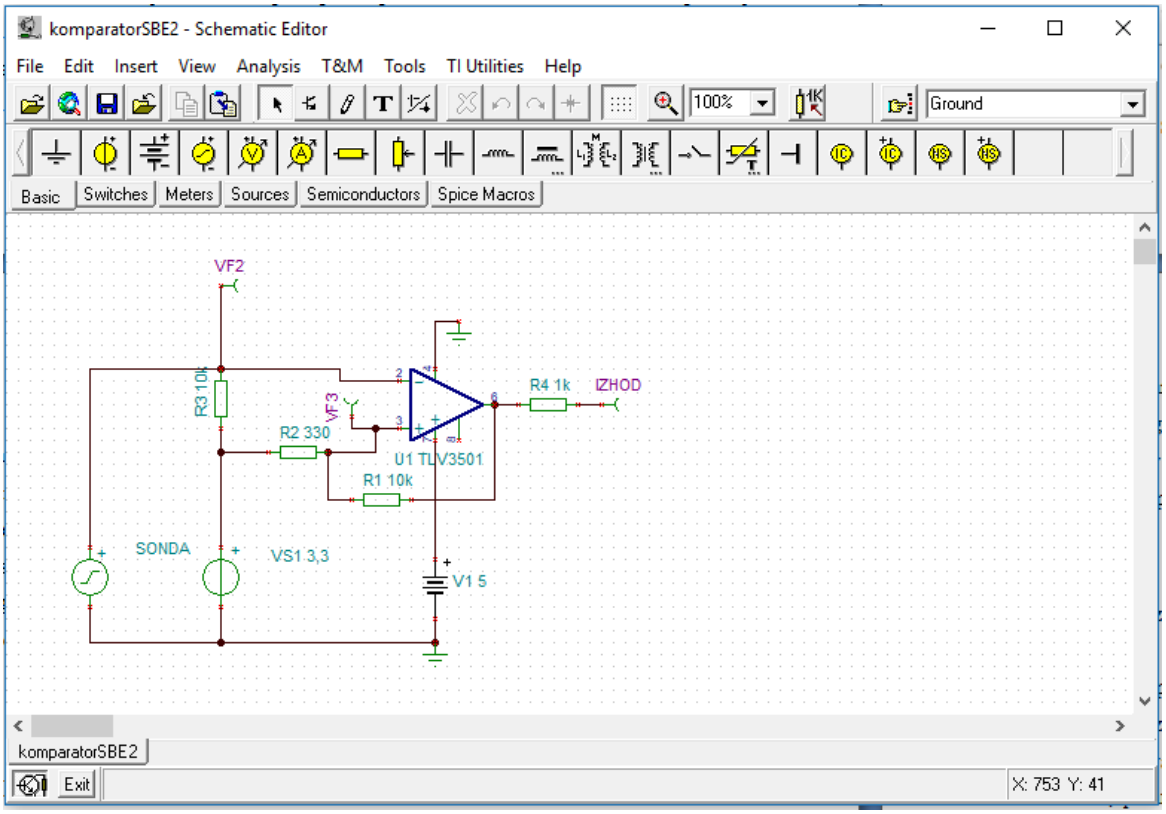
Slika 11: Izris signalov - napetosti v električnem vezju

3.2.2 **Simulacija vezja s pomočjo programske opreme**

Pri razvoju in analiziranju elektronskega vezja pretvornika sem uporabil programsko orodje Tina-ti (Slika 12), ki je namenjeno simulaciji delovanja analognih vezij. Programsko orodje je prosto dostopno na spletnem naslovu (http://www.ti.com/tool/TINA-TI, zadnjič obiskano 27. februarja 2018). Po prenosu programa je potrebno opraviti še njegovo namestitev.

Programsko orodje je razdeljeno na štiri dele, in sicer na menijsko vrstico, kjer so dostopne različne možnosti za upravljanje s programom vrstico z ikonami, katere omogočajo dostop do pogostejših operacij, ki jih program ponuja (shranjevanje, brisanje, preverjanje ...), vrstico z elektronskimi komponentami, ki se dodatno delijo na: osnovne komponente, stikala, merilne instrumente, izvore napajanja in generatorje signala ter na okno za izris elektronskega vezja.

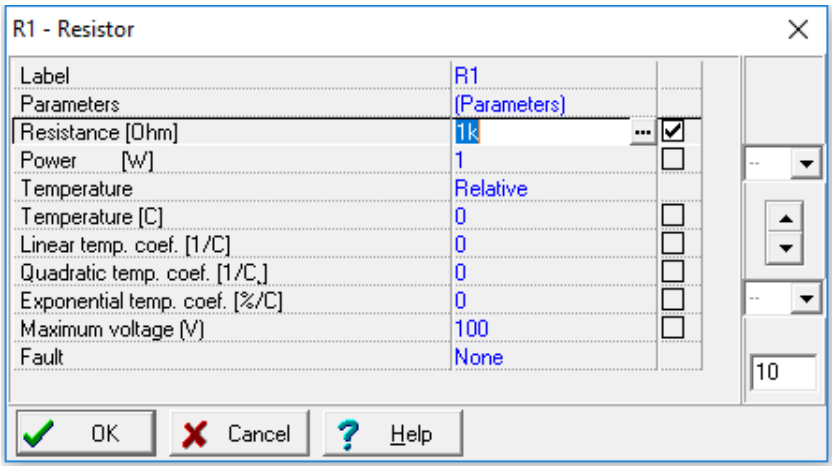
Za postavitev določene komponente je potrebno to komponento izbrati iz orodne vrstice. Takoj za tem se komponenta izriše na risalni površini, kjer jo s pomočjo miške premaknemo na želeno mesto.



Vir: Lasten

Slika 12: Programsko orodje TINA-TI

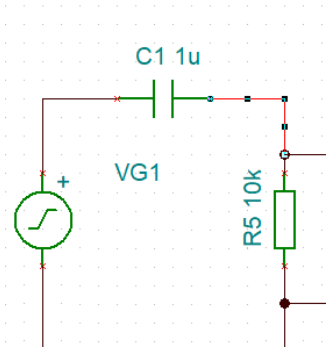
Uporabnik lahko posamezni elektronski komponenti spreminja lastnosti. To naredi tako, da z desnim klikom miške klikne na komponento. Uporabniku se odpre dodatno okno (Slika 13). V oknu so prikazane lastnosti (angl. Properties) izbrane komponente, ki jih je možno spreminjati.



Vir: Lasten

Slika 13: Določitev lastnosti izbrane komponente

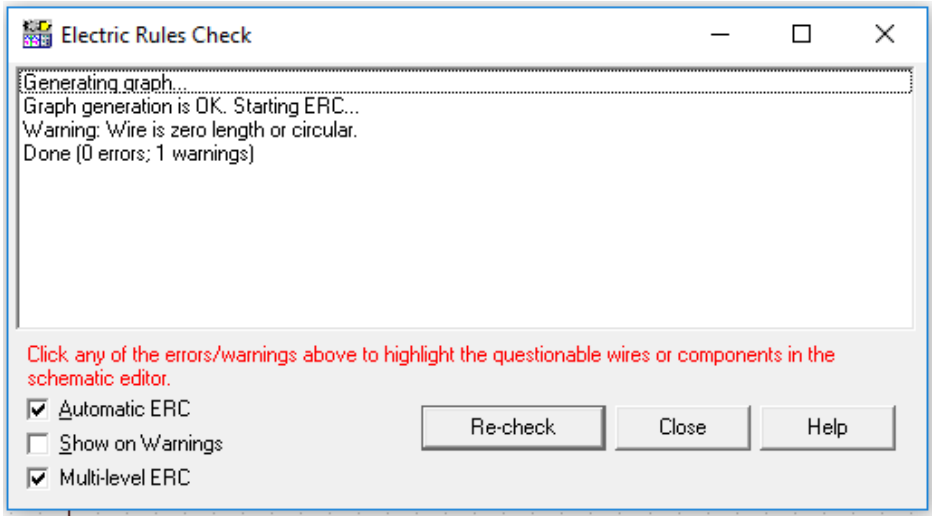
Za izris električne sheme vezja je potrebno elektronske komponente medsebojno povezati (Slika 14). Postopek povezave je enostaven, saj je potrebno označiti element, ki ga želimo povezati z drugim izbranim (označenim) elementom. Pri tem nastane povezava (vez), ki je na shemi razvidna kot črta med komponentami in je ob izboru ali izrisu rdeče obarvana.



Vir: Lasten

Slika 14: Risanje povezovalnih vezi

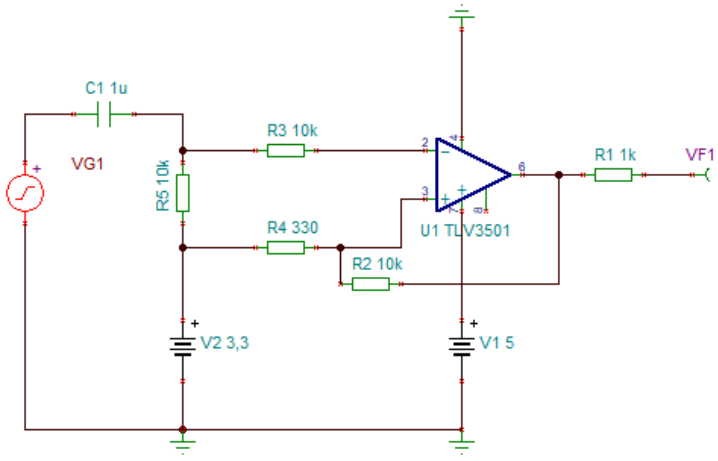
Po uspešno izvedeni povezavi komponent je potrebno električno vezje preveriti. To storimo z uporabo pripomočka ERC (angl. Electric Rules Check). Storitev je dostopna v menijski vrstici »orodja« (angl. Tools). S preverjanjem pravilnosti program ugotovi, ali so povezave med komponentami pravilno vzpostavljene oziroma opozori na druge nepravilnosti, ki jih lahko uporabnik po pomoti naredi ali pozabi. ERC pripomoček je priročen za hitro odpravo napak, ki jih s prostim očesom na zaslonu prezremo.



Vir: Lasten

Slika 15: ERC – preverjanje pravilnosti povezav električnega vezja

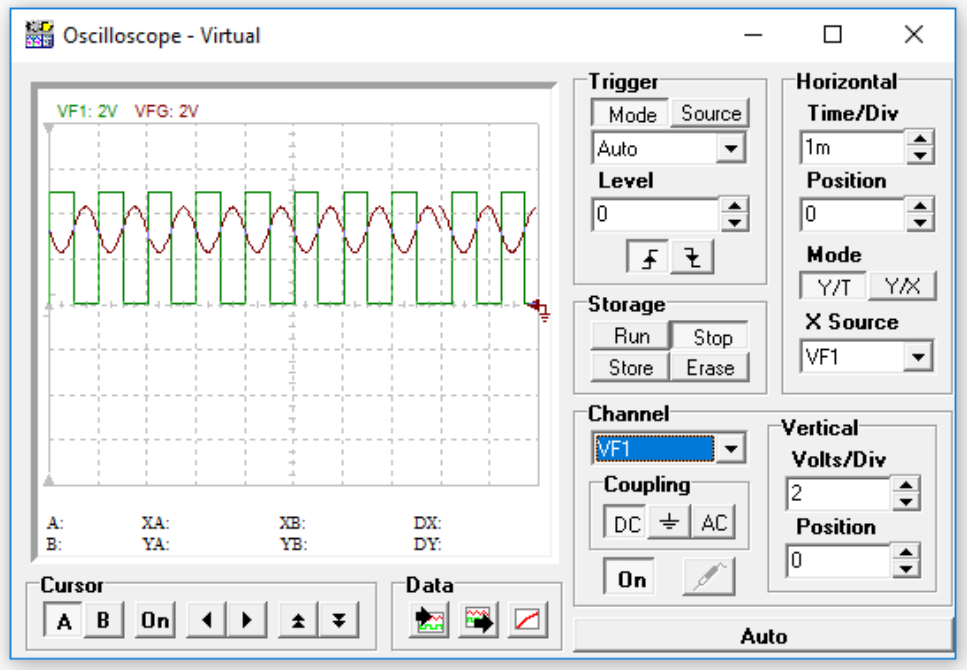
Narisani prototip pretvornika (Slika 16) nato s pomočjo programa tudi izmerimo. Pomagamo si lahko z »virtualnimi« instrumenti, kot so: osciloskop, multimeter, analizator signalov itd. Merilni instrument si izberemo iz menijske vrstice, ki je označena s »T&M«. Za merjenje pravilnosti pretvorbe signala sem tako uporabil dva instrumenta, in sicer osciloskop, ki je namenjen sočasnemu merjenju dveh signalov, ter analizator signalov, s pomočjo katerega lahko sočasno izrišemo več različnih signalov.



Vir: Lasten

Slika 16: Shema pretvornika

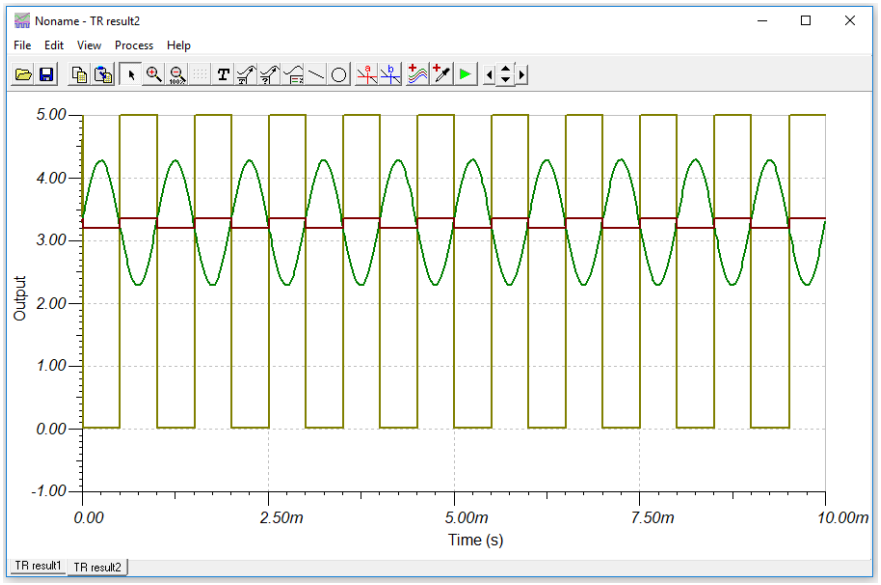
S pomočjo osciloskopa (Slika 17) lahko preverimo potek vhodnega in izhodnega signala. Osciloskop nam ponuja dva kanala. V tem primeru sem izbral signal na izhodu frekvenčnega generatorja (VG1) ter signal na izhodu pretvornika (VF1). Iz prikaza je razvidno, da je vhodni signal sinusne oblike (rdeča črta) in njegova napetost niha med 3,3 V ± 0,5 V. Izhodni signal (zelene barve) je pravokotne oblike amplitude 5,0 V. Razvidno je tudi, da imamo na izhodu pretvornika samo en pravokotni impulz dolžine natanko polovico periode signala na vhodu (VG1). Napetost izhodnega signala odčitamo tako, da preštejemo vertikalne razdelke, ki so na zaslonu prikazane s črtkano črto. Vrednost razdelka določimo z opcijo Volti/razdelek (angl. Volts/Div). V mojem primeru je vertikalni razdelek enak 2,0 V.



Vir: Lasten

Slika 17: Virtualni osciloskop

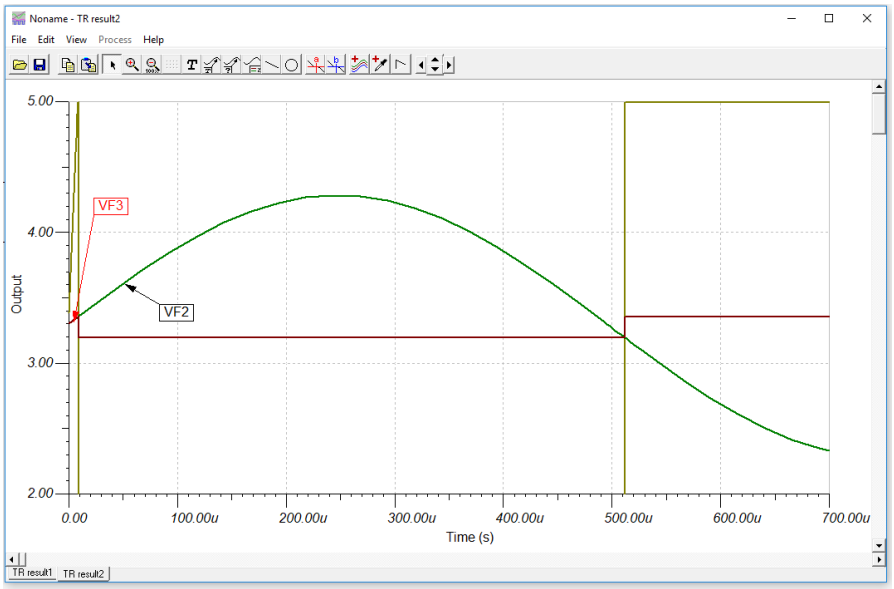
Na podoben način lahko analiziramo potek različnih signalov v vezju v neki časovni enoti. Slika 18 prikazuje potek signalov v vezju. Na sliki je razviden izhodni signal (pravokotne oblike) rumene barve. Sledi signal sinusne oblike zelene barve, ki je prisoten na invertirajočem vhodu komparatorja, ter signal pravokotne oblike, ki se s povratno vezavo na komparatorju spreminja in je priključen na neinvertirajoč vhod komparatorja.



Vir: Lasten

Slika 18: Pregled signalov v vezju

Pri analiziranju signala (Slika 19) lahko določeni del slike tudi povečamo in tako dobimo boljši pregled nad potekom signalov. Ob vklopu vezja lahko tako vidimo, da se napetost na neinvertiranem vhodu komparatorja (VF3) dvigne do 3,35 V, medtem ko se napetost na invertirajočem vhodu (VF2) spreminja s signalom sonde, kateremu je prišteta referenčna napetost 3,3 V. Ko napetost na invertirajočem vhodu pade pod 3,19 V, se na izhodu komparatorja pojavi signal z napetostjo 5,0 V.



Vir: Lasten

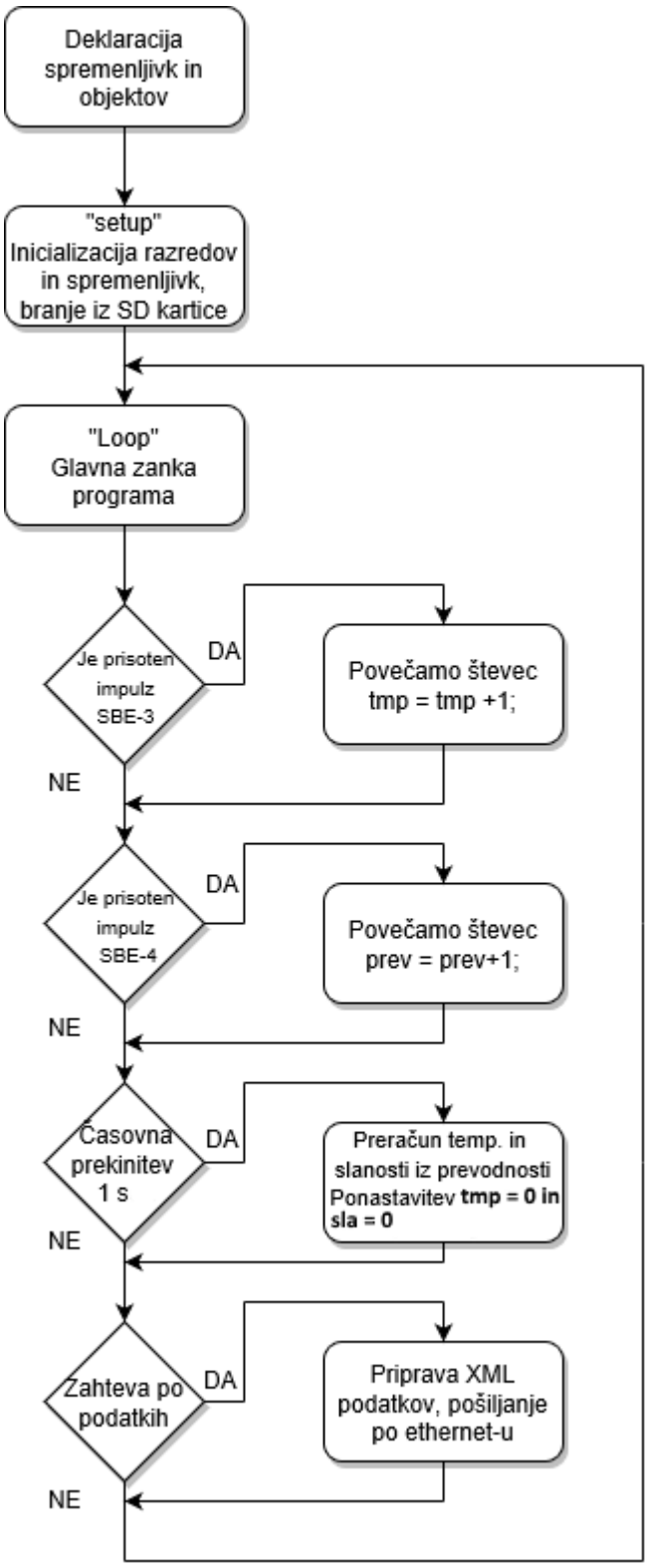
Slika 19: Podrobnejši pregled signalov s povečavo

3.3 Programska oprema merilne enote

Srce merilne enote je program, katerega naloga je meriti frekvenco sond in preračunati temperaturo ter slanost morske vode iz izmerjene prevodnosti. Ker merilna enota vsebuje tudi ethernet povezavo, program vsebuje tudi programsko kodo, ki na zahtevo odjemalca posreduje izmerjene podatke v XML obliki (angl. Extensible Markup Language). Programska koda je napisana v programskem jeziku C. Navzven je merilna enota vidna kot spletna storitev (angl. Web service).

3.3.1 **Idejni načrt programa**

Osnovna ideja pri razvoju programa je štetje impulzov iz pretvornika signala v eni sekundi. Pri tem upoštevamo štetje impulzov iz dveh sond ter sočasno pošiljanje podatkov na zahtevo odjemalca v ethernet omrežje. Ker se program sekvenčno izvaja, moramo biti pozorni na čas, ki ga bo mikroprocesor porabil za posredovanje podatkov po ethernet omrežju in na čas zaznave signala na vhodih mikrokrmilnika. Zato smiselno uporabimo prekinitvene rutine oz. prekinitve (angl. Interrupts). Prekinitev v računalništvu pomeni, da neki dogodek povzroči začasno prekinitev izvajanja programa, ki ga mikroprocesor trenutno izvaja in začne izvajati drugi program, kateremu pravimo prekinitveni ali servisni program. Ta program je običajno kratek. Zahteva po prekinitvi lahko pride iz katere od vhodno-izhodnih naprav (v mojem primeru je to signal iz pretvornika signala merilnih sond). Prekinitve lahko sproži tudi mikroprocesor. Med takšne sodi časovna prekinitev (angl. Timer interrupt), s katero lahko prekinemo izvajanje osnovnega programa točno ob določenem času in izvedemo določeno programsko rutino. Pri pisanju programa sem uporabil vhoda št. 2 in 3 mikrokontrolerja Arduino Uno. Signal na obeh vhodih lahko sproži zunanjo prekinitev (izvajanje servisne rutine). Zunanjo prekinitev sem tako uporabil za štetje impulzov sonde SBE-3 in SBE-4. Ker pa želimo izračunati frekvenco in iz frekvence temperaturo ter prevodnost (slanost), moramo uporabiti časovno prekinitev, ki se izvaja vsako sekundo po predhodni časovni prekinitvi. V časovni prekinitvi program prebere število preštetih impulzov iz dveh globalnih spremenljivk in tako dobimo frekvenco oz. število impulzov v eni sekundi, ki jih sproži posamezna sonda. Sledi ponastavljanje vrednosti števcev impulzov na vrednost 0 in cikel se v zanki ponovi. Programska koda in opis delovanja merilne enote sta opisana v »Dodatek E- Izvorna koda za mikrokrmilnik in opis delovanja« na strani 114. Diagram poteka programa prikazuje Slika 20 (http://www.s-sers.mb.edus.si/gradiva/w3/sistemi/prekinitve.html, zadnjič obiskano 19. marca 2018).



Vir: Lasten

Slika 20: Diagram poteka programa merilne enote

4 Metoda spremljanja podvodnega sveta s kamero

Merilno enoto sestavlja tudi ločena enota, ki je namenjena video zajemu podvodnega okolja. Sestavlja jo kamera v vodotesnem ohišju. Kamera je z mrežnim kablom spojena na LAN preklopnik (angl. Switch). Kamera in programska oprema zajemata podvodno dogajanje in posredujeta video vsebino javnemu strežniku za video predvajanje vsebin (Youtube, Ustream, Amazon video itd.). Video toku (angl. Video stream) s programom za video dekodiranje in enkodiranje dodamo podatke o izmerjeni temperaturi in slanosti.

4.1 Kamera

Za video zajem sem v uporabil video kamero, ki omogoča priklop in prenos podatkov v LAN omrežje. Odločil sem se za kamero z oznako »Aptina AR0130 CMOS Hi3518C CCTV IP camera module«, ki sem jo naročil na spletni strani www.aliexpress.com.

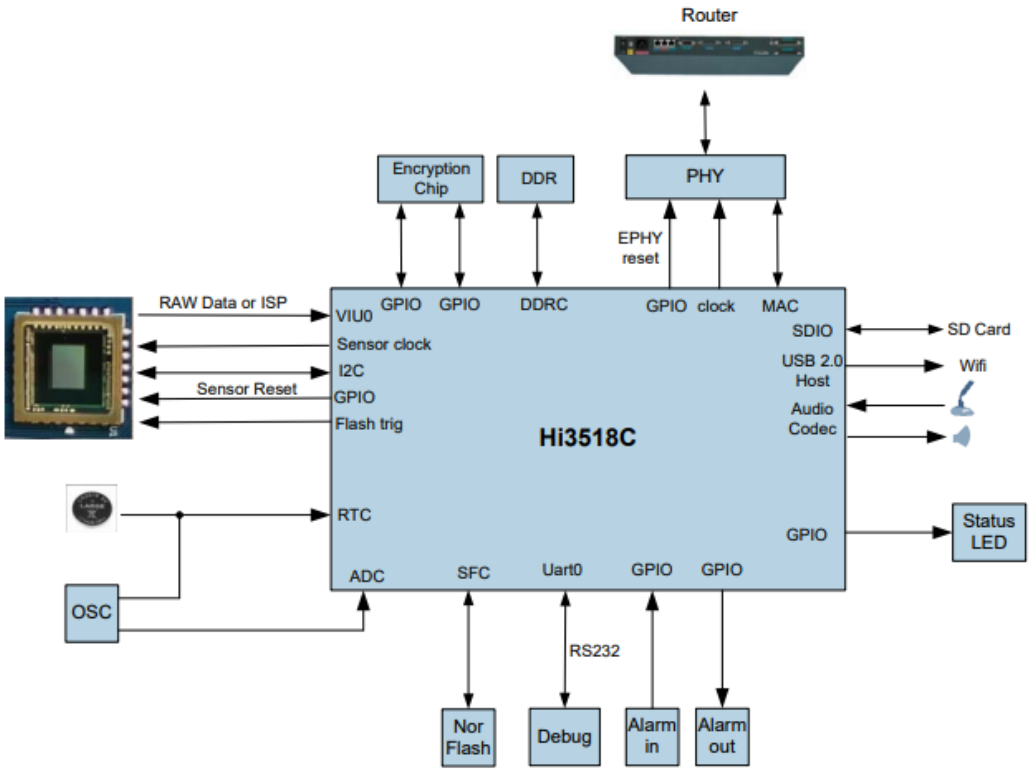
Vir: www.aliexpress.com
Slika 21: Kamera

Slika 21 prikazuje kamero, ki je sestavljena iz dveh delov: objektiva (zgornji del) in mikroračunalnika s CCD tipalom (spodnji del). Objektiv je valjaste oblike, narejen iz plastične mase. Na spodnji strani je oblikovan tako, da ga enostavno nasadimo na priključek, kjer je CCD tipalo. Na srednjem delu objektiva sta prisotna dva zelo majhna elektromotorčka, s katerima lahko upravljamo približevanje/oddaljevanje ter izostritev slike (Slika 22). Objektiv upravljamo s programsko opremo, ki je priložena kameri.

Vir: Lasten

Slika 22: Objektiv kamere

Vezje (srce) kamere je mikroračunalnik, ki temelji na mikroprocesorju Hi3518/Hi3518C, proizvajalca HiSilicon. Slika 23 prikazuje blokovni diagram mikroračunalnika, v katerem zasledimo različne vhodno-izhodne enote.



Vir: https://cdn.hackaday.io/files/19356828127104/Hi3518%20DataSheet.pdf

Slika 23: Blokovni diagram mikroračunalnika

Med najpomembnejše dele spada CCD tipalo, ki skrbi za pretvorbo svetlobe v podatke. Uporabljeno je CCD tipalo proizvajalca »ON Semiconductors«, model Aptina AR0130CS. Tipalo je z ISP vodilom (angl. Image signal procesor) povezano z mikroprocesorjem kamere. Tehnična dokumentacija tipala je dosegljiva na spletnem naslovu

(https://www.onsemi.com/pub/Collateral/AR0130CS-D.PDF, zadnjič obiskano 26. marca 2018).

Ostali vhodno-izhodni priključki so:

- Ethernet priključek, ki omogoča priklop kamere s kablom na ethernet omrežje.
- SD Card priključek, ki omogoča priklop enote za branje in pisanje na SD pomnilniško kartico.
- WiFi priključek, ki omogoča priklop kamere preko brezžičnega omrežja (določeni modeli).
- Audio omogoča priklop mikrofona in dekodiranje zvoka z video signalom.
- Alarm Out priključek omogoča proženje signala, kadar kamera zazna razliko med dvema slikama (opcijo je potrebno programsko aktivirati).
- Debug priključek je namenjen povezavi kamere s serijskimi vrati z računalnikom. Uporablja se za nadgradnjo programske opreme (jedra) kamere (angl. Kernel).
- PTZ izhod za upravljanje objektiva (angl. Pan, Tilt, Zoom). Gre za izhod iz mikroračunalnika, s katerim so povezani elektromotorčki na objektivu. S programsko opremo je možno v realnem času spreminjati parametre objektiva.

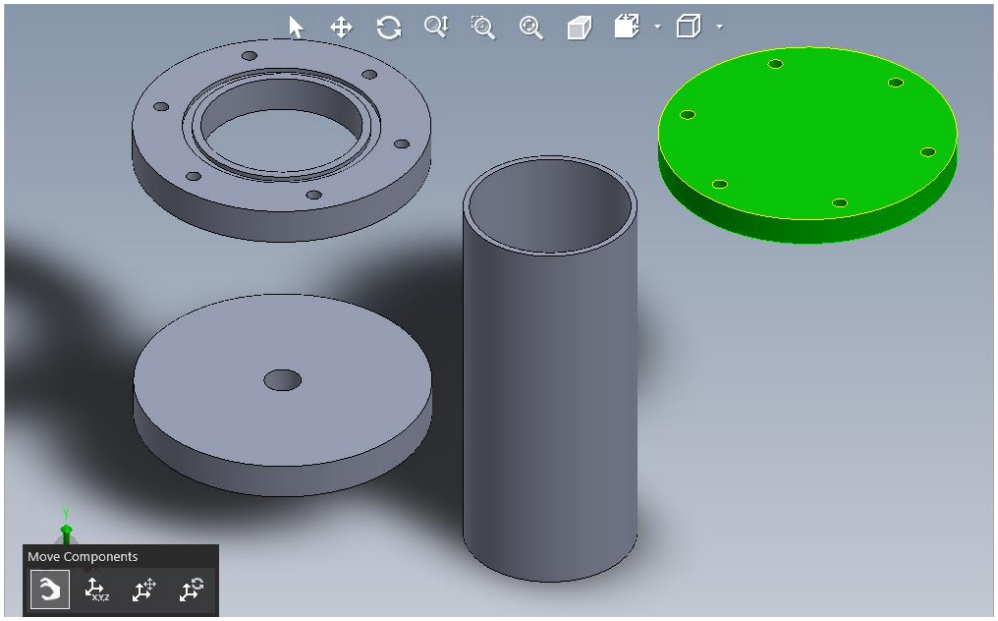
Tehnične lastnosti kamere:

- Zajem v HD ločljivosti 1280 × 1024 pik pri hitrosti 25 slik/sekundo (angl. FPS – frame per second).
- Video format kompresije H.264.
- Dimenzija kamere: 38 × 38 (mm).
- Ethernet povezava: RJ-45 10/100 Mb.
- Procesor: Hi3518C, ki temelji na ARM926 procesorju s taktom 440 MHz in 16 KB predpomnilnika.
- Napajanje: enosmerna napetost 12 V in porabo 110 mA.

Kamero povezuje z merilno enoto stikalo (angl. Switch), ki je kratko opisano v dodatku G »Stikalo za povezavo merilne enote in kamere« na strani 122.

4.2 Vodotesno ohišje

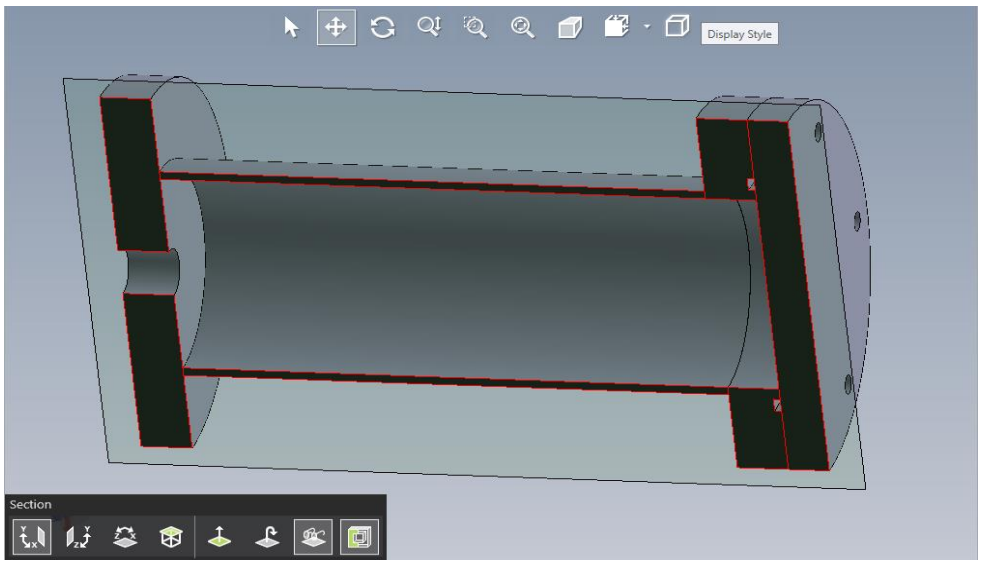
Za snemanje podvodnega okolja je potrebno ustrezno vodotesno ohišje. Tega sem dobil na izposojo na MBP NIB. Ohišje sestavljajo štirje preprosti deli, ki so izdelani iz pleksi stekla in so medsebojno zalepljeni. Prikaz sestavnih delov je prikazan spodaj.



Vir: Lasten

Slika 24: Računalniški prikaz sestavnih delov ohišja

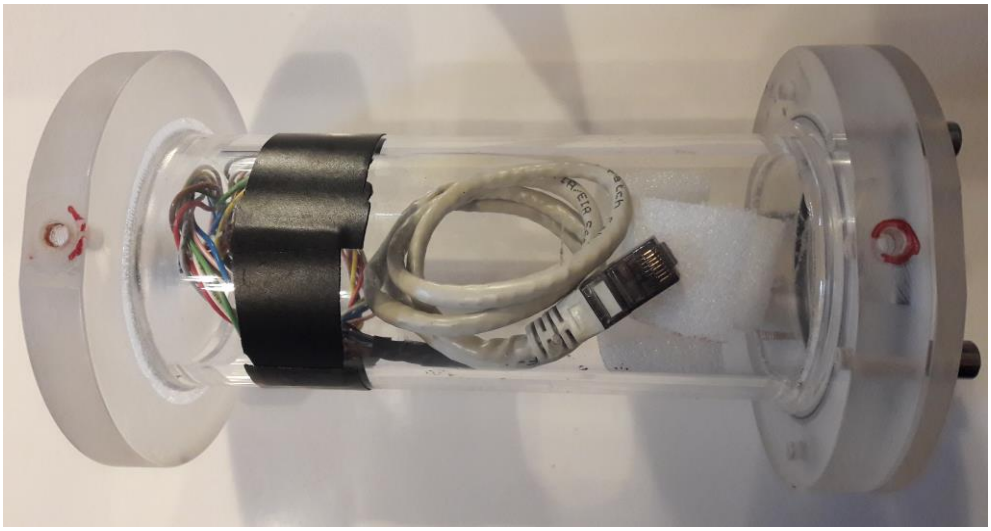
Slika 25 prikazuje ohišje v prerezu. Na levi strani slike je pokrov, ki ima na sredini odprtino, v katero je privijačen vodotesni konektor za povezavo kamere z ethernet omrežjem in napajanjem. Tesnjenje konektorja je izvedeno z gumijastim tesnilom.V sredini je cev, ki tvori ohišje. Cev je dolžine 170 mm in debeline 5 mm. Na desni strani cevi se nahaja prirobnica zunanjega premera 115 mm in notranjega premera 70 mm. Ta ima izdolben utor, v katerem je o-ring tesnilo. Pokrov ohišja je premera 115 mm in debeline 5 mm. V obod pokrova so izvrtane luknje, v katere so vstavljeni inbus vijaki M3 dolžine 20 mm. Vijaki so na nasprotni strani privijačeni v prirobnico.



Vir: Lasten

Slika 25: Prikaz ohišja v prerezu

Končni izgled ohišja z vstavljenim konektorjem in povezovalnim kablom prikazuje Slika 26.

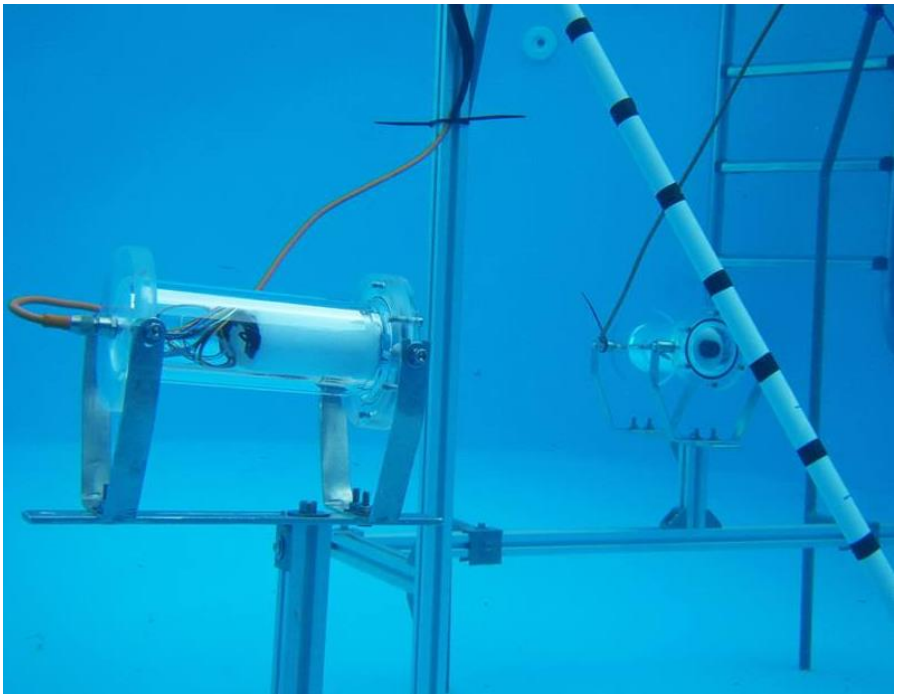


Vir: Lasten

Slika 26: Izdelano ohišje iz pleksi stekla

V tako izdelano ohišje je potrebno vstaviti še kamero. Kamera se priključi z ethernet kablom. V ohišje kamere je poleg kamere vstavljen tudi POE (angl. Power over ethernet) pretvornik, ki omogoča komunikacijo po ethernet kablu na večje razdalje. Kamera je v ohišje pričvrščena s PVC peno. Ta onemogoča premikanje kamere v ohišju. Slika 27 prikazuje uporabo ohišja pri izvajanju poizkusov v bazenu. Vidni so priključni kabel, žice v notranjosti ohišja in PVC

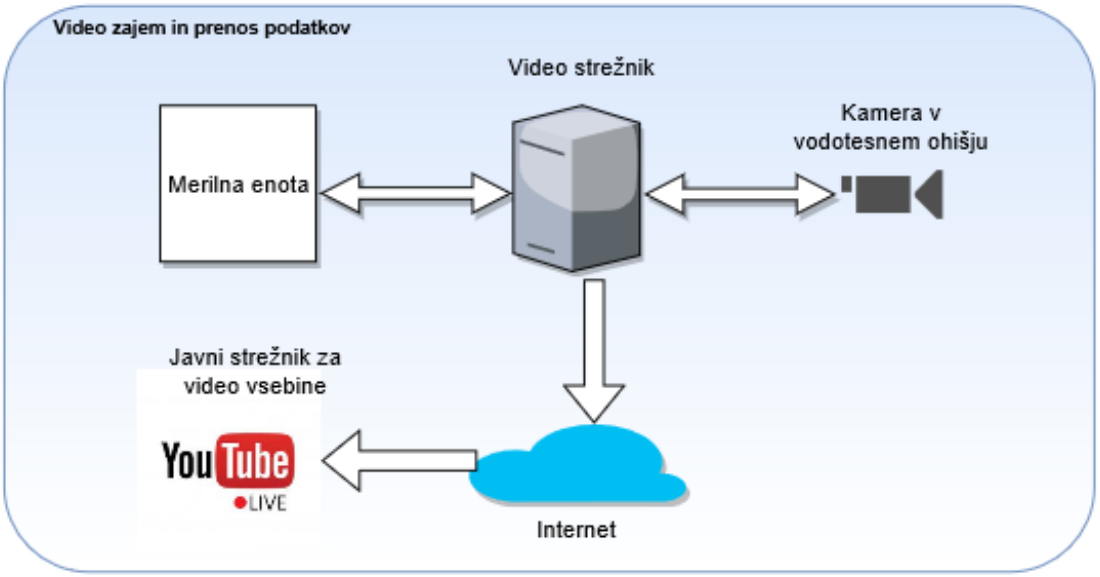
pena, ki obdaja kamero v ohišju. Na sliki skrajno desno je vidno drugo vodotesno ohišje s pokrovom ohišja s šestimi vijaki, vidna sta tudi o-ring tesnilo in objektiv kamere.



Vir: Tihomir Makovec MBP, 2018

Slika 27: Uporaba kamere za raziskovalne namene

5 Metoda dodajanja merljivih količin v sliko, video zajem in posredovanje posnetka v splet



Vir: Lasten

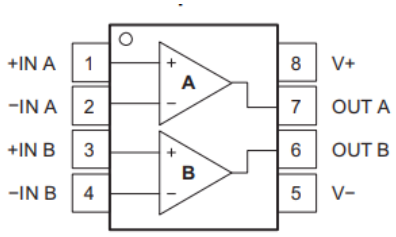
Slika 28: Blokovni diagram video zajema in posredovanja v splet

Pri sestavi merilne enote sem želel poleg merjenja temperature in slanosti zajemati tudi živo sliko podvodnega dogajanja ter jo posredovati v splet v obliki video toka (angl. Video stream). Z odprtokodno programsko opremo in lastnim programom sem uspel združiti podatke iz merilne enote z video zapisom. Enkodirane podatke (video zapis) posredujem na strežnik Youtube Live (Slika 28). Uporabnik si lahko video posnetek ogleda na spletni strani za prikaz podatkov iz merilne enote ali na spletni strani Youtube. V dodatku J »Programska oprema za video zajem in posredovanje žive slike v splet« na strani 136 sem predstavil program FFMPEG, ki je namenjen dekodiranju in enkodiranju ter pošiljanju video posnetka v splet ter FileMover za branje podatkov iz merilne enote in vpisovanje v datoteko, katero ob enkodiranju video posnetka uporablja program FFMPEG. Program za pregled izmerjenih podatkov in video vsebine je podrobneje opisan v poglavju »Spletna programska oprema za prikaz podatkov« na strani 54.

6 Rezultati

6.1 Razvoj prototipa pulznega pretvornika

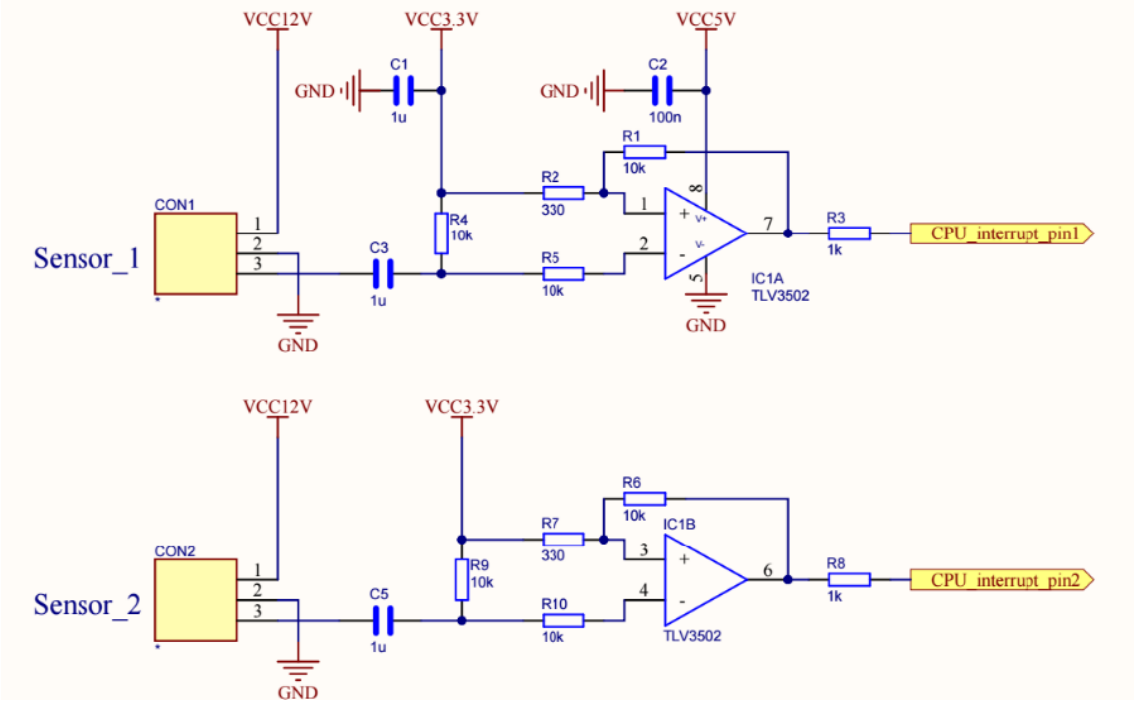
Za razvoj prototipa sem uporabil komparator proizvajalca Texas Instruments model TLV3502. Specifikacija komparatorja je dostopna na spletnem naslovu (http://www.ti.com/lit/ds/symlink/tlv3502.pdf zadnjič obiskano 1. marca 2018). Integrirano vezje vsebuje dva komparatorja (Slika 29). Uporabil sem ju za primerjanje oziroma pretvorbo signala iz sonde SBE-3 in sonde SBE-4.



Vir: http://www.ti.com/lit/ds/symlink/tlv3502.pdf

Slika 29: Komparator TLV3502

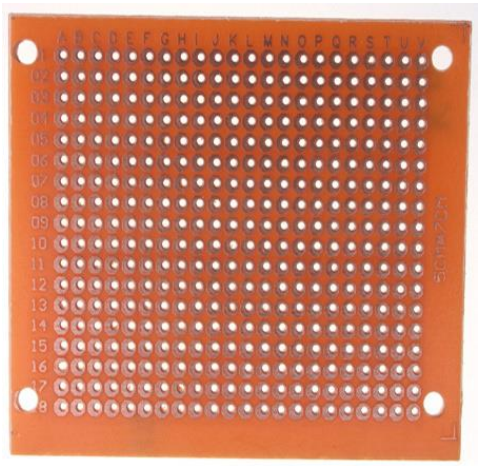
Slika 30 prikazuje električno shemo pretvornika z dvema vhodoma con1 in con2, ki sta povezana z elektronskimi komponentami in s komparatorjem. Shema je izdelana na osnovi električne sheme na strani 22.



Vir: Lasten

Slika 30: Električna shema pretvornika

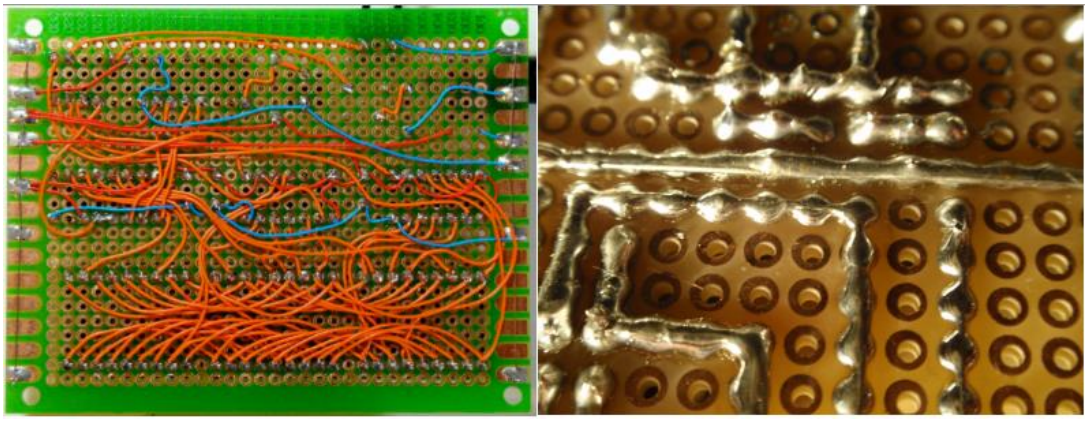
Pri sestavi prototipa sem uporabil posebno ploščico (Slika 31), ki se uporablja pri razvoju prototipnih vezij. Ploščica je narejena iz umetne mase (pertinaks), na katero so naparjeni bakreni kontakti z izvrtano luknjo. Nekatere razvojne ploščice imajo naparjen baker tako s spodnje kot zgornje strani (dvostranske).



Vir: https://electronics.stackexchange.com/questions/55236/how-to-make-traces-on-an-universal-pcb

Slika 31: Razvojna ploščica za izdelavo prototipnih vezij

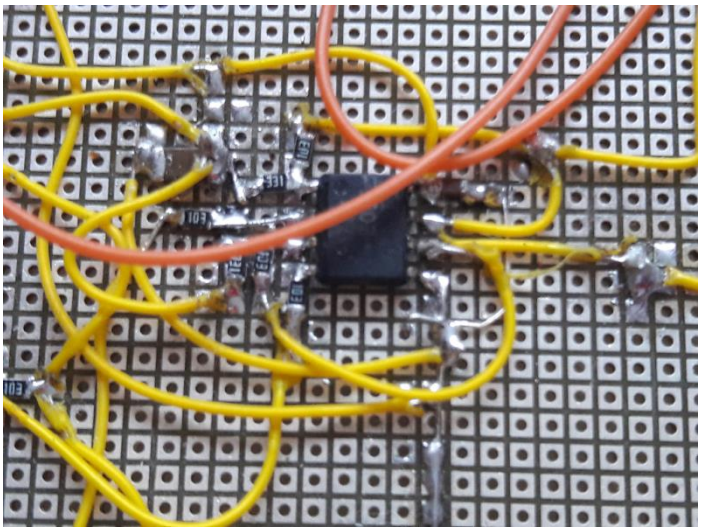
Uporabljamo lahko elemente, ki ustrezajo standardu DIP / DIPP (angl. Dual Inline Package, Dual Inline Pin Package) ali SMD (angl. Surface mount device). Povezovalne vezi med komponentami so lahko narejene z uporabo tanke žice ali cina. Takšen primer povezovanja je prikazan na spodnji sliki.



Vir: Lasten

Slika 32: Povezovalne vezi iz žic (levo) ter povezovalne vezi iz cina (desno)

Prototip pretvornika sem izdelal na ploščici velikosti 50 mm × 45 mm. Uporabil sem SMD elektronske komponente. Slika 33 prikazuje povečani del prototipnega vezja, kjer so razvidni elektronski elementi (upori, kondenzatorji in komparator).

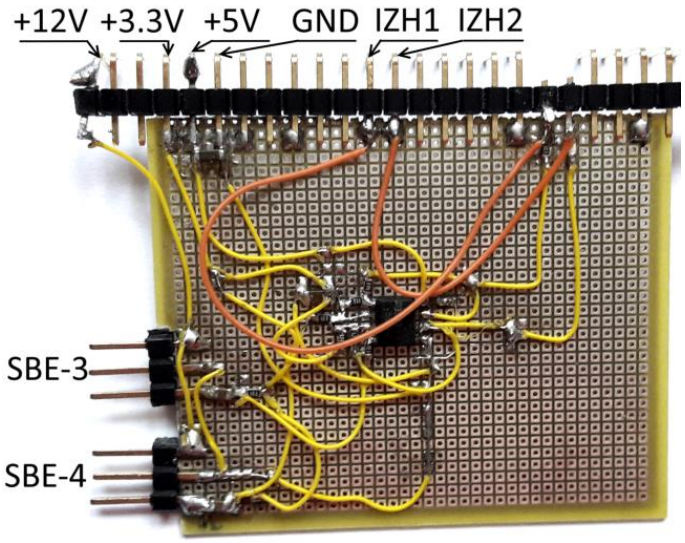


Vir: Lasten

Slika 33: SMD elektronske komponente na prototipni ploščici

Izdelan prototip pretvornika prikazuje Slika 34. Na zgornji strani prototipne ploščice je prisoten konektor (priključna letev), ki je namenjen pričvrstitvi prototipnega vezja na vodilo mikrokrmilnika Arduina, ki ga bom opisal v nadaljevanju, saj je za razumevanje delovanja potrebno poznati tudi osnove mikrokontrolerjev in programiranja. S puščicami sem označil nekatere pomembnejše priključke, kot so:

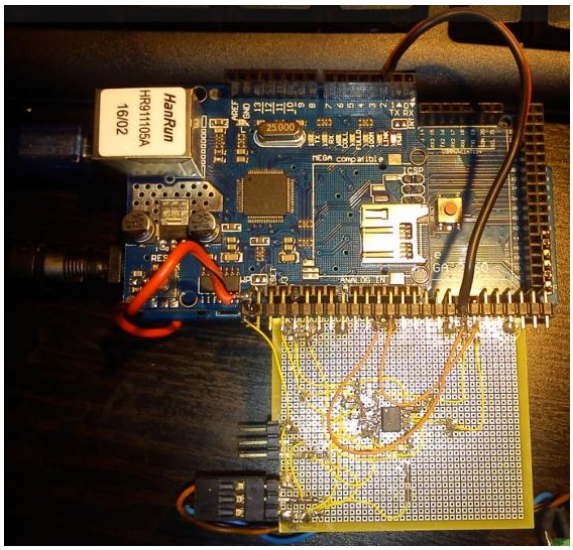
- +12 V je priključek, kamor priklopimo zunanji vir napajanja in je namenjen napajanju merilnih sond. Napetost je lahko med 10 V in 16 V.
- +3,3 V, +5 V in GND so priključki, ki so povezani z vodilom mikrokontrolerja.
- IZH1 in IZH2 sta digitalna izhoda iz pretvornika in sta povezana na digitalna vhoda mikrokrmilnika za beleženje prekinitev.
- SBE-3 in SBE-4 sta vhoda, na katera priključimo sondi SBE-3 in SBE-4. Konektor ima tri sponke: zgornja je namenjena napajanju (+12V), sredinska je ozemljitev (GND), spodnja pa je vhodni signal (signal sonde).



Vir: Lasten

Slika 34: Izdelan prototip pretvornika signala

Slika 35 prikazuje prototip pretvornika, ki je povezan z mikrokrmilnikom Arduino. Tega sem uporabil pri testiranju delovanja prototipa. Naloga mikrokontrolerja je beleženje impulzov, ki prihajajo iz prototipa pretvornika. Ker pa imajo različni modeli Arduino mikrokontrolerjev enak razpored priključkov na vhodno-izhodnem vodilu, sem se odločil, da bom izdelal tiskano vezje ali ščit (angl. Shield), ki bo univerzalen za vse tipe razvojnih platform Arduina. Tega bo možno pričvrstiti na vhodno-izhodno vodilo mikrokontrolerja, sočasno pa bo omogočen priklop še drugega ščita, ki bo skrbel za izmenjavo podatkov po ethernet omrežju.



Vir: Lasten

Slika 35: Testiranje prototipa

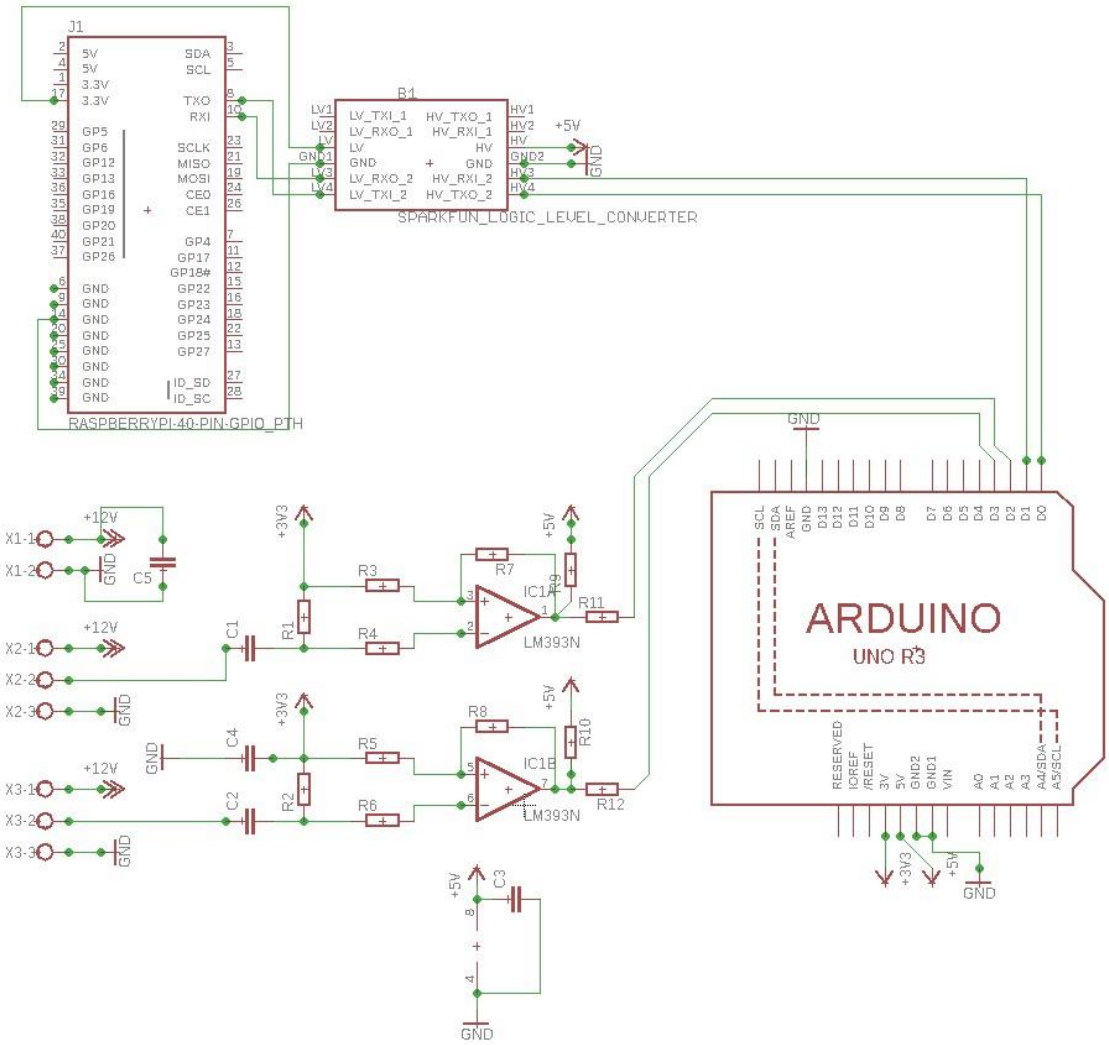
6.2 Izdelava pulznega pretvornika v obliki ščita za platformo Arduino

Po izvedenih testiranjih s prototipnim vezjem sem se odločil za izdelavo tiskanega vezja, ki ga bo možno enostavno uporabiti na različnih platformah Arduina ter bo zaradi same izvedbe zanesljivejše od prototipnega vezja.

Za načrtovanje tiskanega vezja sem uporabil brezplačno različico programskega orodja Eagle, ki sem ga prenesel s spletnega naslova (https://www.autodesk.com/products/eagle/overview , zadnjič obiskano 3. marca 2018). Eagle (angl. Easily Applicable Graphical Layout Editor) je zelo učinkovit in zmogljiv CAD/CAM računalniški program za načrtovanje ploščic tiskanih vezij ter za risanje električnih načrtov. Programski paket vsebuje več različnih opcij za načrtovanje tiskanega vezja.

6.2.1 **Vezalni načrt pulznega pretvornika**

Slika 36 prikazuje vezalni načrt, iz katerega sem kasneje izdelal tiskano vezje. Elektronske komponente, ki so uporabljene v načrtu, sem pridobil iz obstoječih knjižnic programskega paketa Eagle, medtem ko sem 40-polni konektor, napetostni pretvornik in Arduino modul uvozil iz javnega repozitorija Github (https://github.com/sparkfun/SparkFun-Eagle-Libraries, zadnjič obiskano 5. marca 2018), kjer so dodatne knjižnice, ki jih uporabniki sami izdelajo in objavijo za nadaljnjo uporabo v projektih.

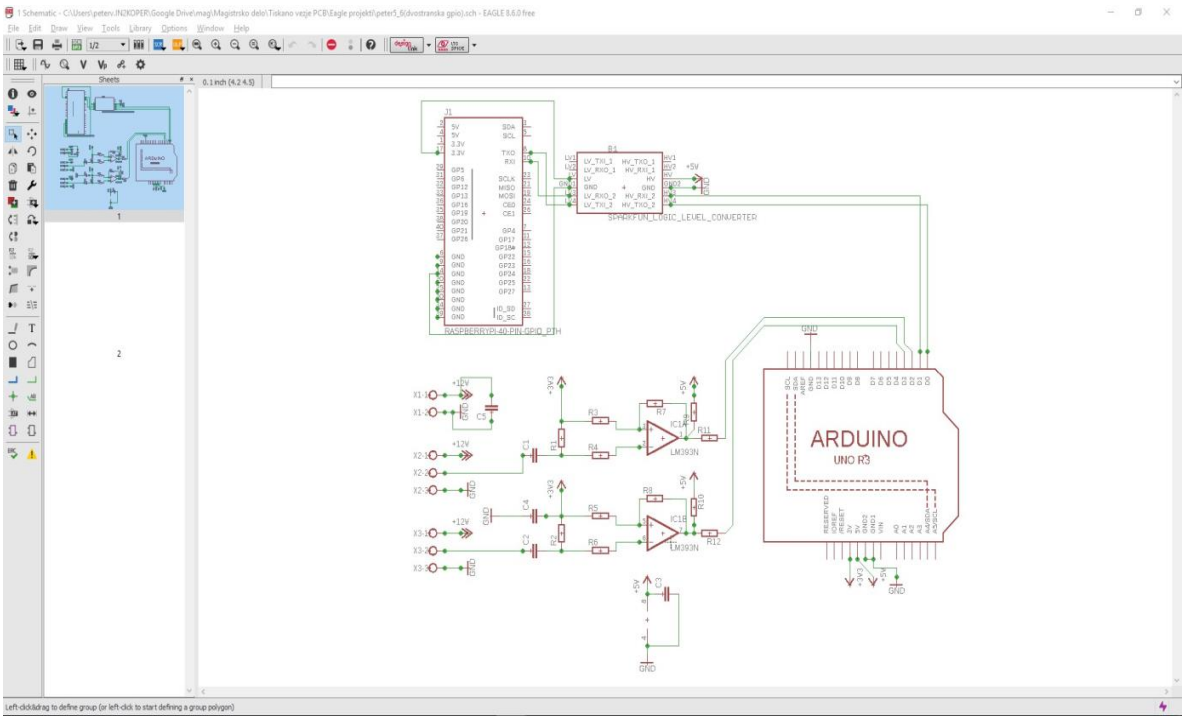


Vir: Lasten

Slika 36: Vezalni načrt

Pri risanju vezalnega načrta sem pričel s priključitvenimi konektorji. Na vezalni shemi so označeni s črko X1 do X3. Konektor X1 je namenjen dovodu napajanja za merilni sondi, medtem ko sta konektorja X2 in X3 namenjena priklopu sond. Za konektorji sledijo komponente pretvornika, ki jih sestavljajo upori, kondenzatorji ter komparator LM393N. Gre za pogosto in zelo razširjeno različico komparatorja. Uporabljen komparator in ostale komponente so v tehnologiji DIP. Ta vsebuje dva komparatorja, ki sta razvidna na vezalni shemi. Izhodi posameznega komparatorja so povezani na vodilo mikrokrmilnika Arduino, ki je v vezalnem načrtu razviden kot »Arduino uno R3«. Povezani so na vhod mikrokrmilnika z oznakama D0 in D1. Na vezalnem načrtu je razviden pretvornik signala sond, ki je povezan na vhoda D3 in D2 mikrokrmilnika Arduino. Ker sem želel pretvornik uporabiti tudi v povezavi z mikro računalnikom RaspberryPi, sem v načrtu predvidel tudi pretvornik

napetostnih nivojev, ki je označen z R1. Pretvornik služi pretvorbi UART signala 5,0 V na strani Arduina na 3,3 V signal, ki ga lahko uporabim na strani mikroračunalnika RaspberryPi ali katerega koli drugega, ki je kompatibilen z RaspberryPi. V primeru, če bi na vodilo mikroračunalnika RaspberryPi pripeljal 5,0 V signal, bi ga lahko trajno poškodoval. Z uporabo napetostnega pretvornika je tako možno vzpostaviti serijsko povezavo med Arduinom in mikroračunalnikom, ki uporablja drugačne napetostne nivoje. Več o tehničnih lastnostih pretvornika je dostopnih na spletnem naslovu (https://www.sparkfun.com/products/12009, zadnjič obiskano 13. junija 2018). Slika 37 prikazuje izris vezalnega načrta v programu Eagle z uporabo modula Schematics.



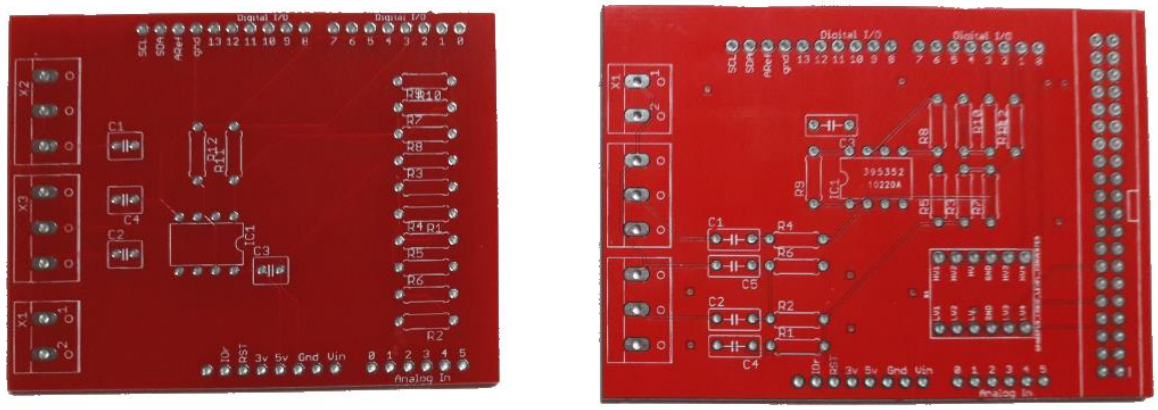
Vir: Lasten

Slika 37: Vezalni načrt v modulu »Eagle Shematic«

Po uspešno izrisani shemi sledi preverjanje povezav (angl. Design Rule Check). S kontrolo preverimo, ali so elementi pravilno povezani. V primeru napak hitreje lociramo napačno povezavo in jo odpravimo. V »Dodatek B- Komparator, delovanje in uporaba«, poglavje B.3, je podrobneje opisano načrtovanje tiskanega vezja.

6.2.2 **Sestavljanje tiskanega vezja pulznega pretvornika**

Izdelal sem dve različici tiskanega vezja. Te prikazuje Slika 38. Na sliki levo je različica tiskanega vezja pulznega pretvornika, ki vsebuje vse potrebne povezave za komunikacijo z mikrokrmilnikom Arduinom. Desna različica, katero sem razvil pozneje, vsebuje še dodatni 40 polni priključek, na katerega so povezana serijska vrata mikrokrmilnika Arduino. Povezavo s 40 polnim priključkom (konektorjem) lahko uporabimo za komunikacijo z mikroračunalnikom tipa Raspberry PI ali drugimi kompatibilnimi različicami (Banana Pi, Orange PI Pc itd.).



Vir: Lasten

Slika 38: Različni izvedbi tiskanine

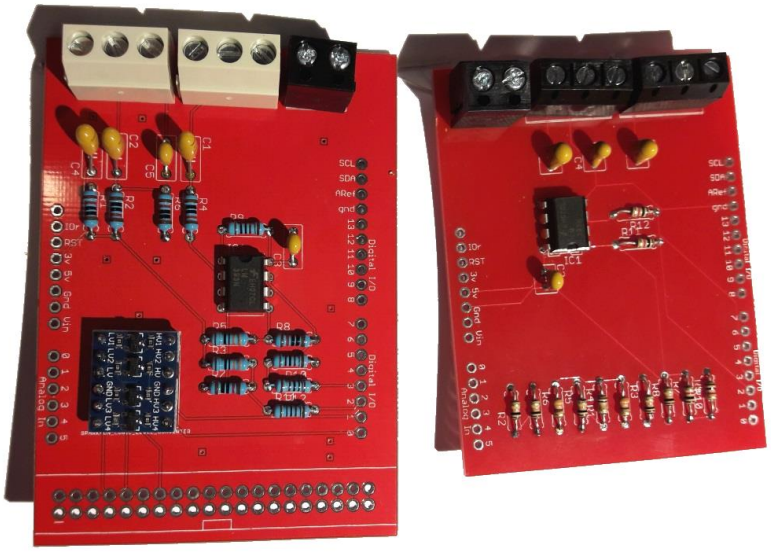
Za sestavo pretvornika sem potreboval elektronske komponente: konektorje, upore, kondenzatorje, komparator in priključne sponke za priklop na vodilo Arduina. Komponente za sestavo sem naročil na spletni strani podjetja Farnell (http://si.farnell.com, zadnjič obiskano 12. marca 2018).

Tabela 1: Seznam uporabljenih komponent

Oznaka	**Vrednost**	**Opis**
X1		konektor z dvema kontaktoma
X2		konektor s tremi kontakti
X3		konektor s tremi kontakti
X4		konektor 40 pinski za priklop na vodilo Raspberry Pi
C1	1 µF	kondenzator
C2	1 µF	kondenzator
C3	100 nF	kondenzator
C4	100 nF	kondenzator
C5	100 nF	kondenzator
R1	10 kΩ	upor

R2	10 kΩ	upor
R3	330 Ω	upor
R4	10 kΩ	upor
R5	330 Ω	upor
R6	10 kΩ	upor
R7	10 kΩ	upor
R8	10 kΩ	upor
R9	1 kΩ	upor
R10	1 kΩ	upor
R11	1 kΩ	upor
R12	1 kΩ	upor
IC1	LM393N	komparator
B1	Logični pretvornik napetosti	

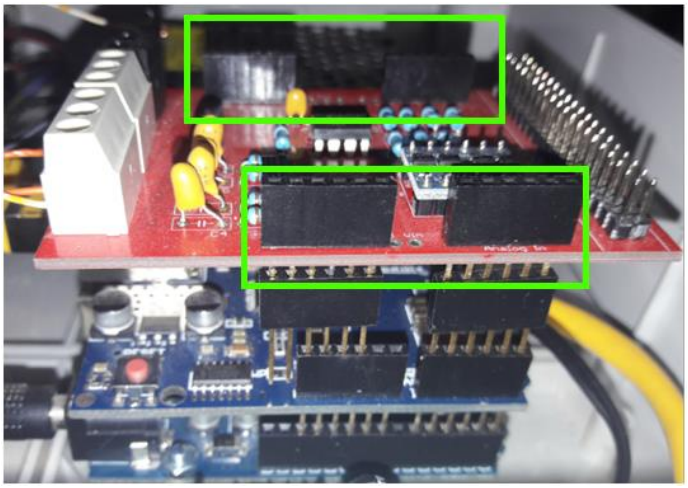
Pri sestavljanju vezja moramo biti pozorni na nekaj osnovnih pravil. V primeru zamenjave polaritete elektrolitskega kondenzatorja lahko le-tega poškodujemo. Enako velja za priključitev integriranega vezja (komparatorja). Napačna postavitev integriranega vezja lahko poruši delovanje vezja oziroma ga lahko trajno poškoduje. Za lažjo postavitev je zato na zgornji strani tiskanine (angl. Top side) prisoten sitotisk z narisanimi elementi in njihovimi označbami. Pri sestavljanju najprej vstavimo najmanjše elemente ter jih na strani lotanja (angl. Solder side) zalotamo. Slika 39 prikazuje sestavljeni tiskani vezji z uporabo DIP elektronskih komponent.



Vir: Lasten

Slika 39: Sestavljeni tiskani vezji pretvornika

Za uporabo pretvornika na mikrokrmilniku Arduino sem na daljši stranici tiskanega vezja zalotal dodaten konektor v obliki glavnika, ki je namenjen priklopu pretvornika na vhodno izhodno vodilo mikrokrmilnika. Slika 40 prikazuje tiskano vezje pretvornika, ki je v obliki ščita (angl. Shield), priklopljeno s pomočjo konektorja v obliki glavnika (označeno z zeleno barvo) na vodilo mikrokrmilnika Arduino.



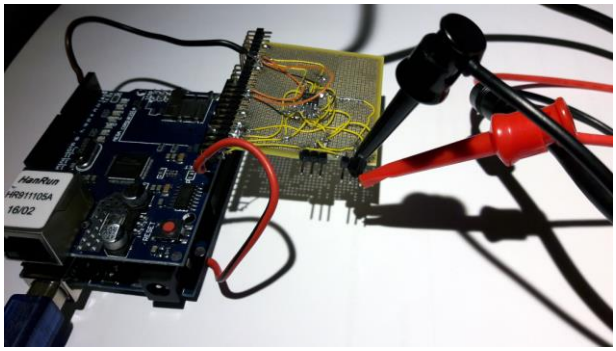
Vir: Lasten

Slika 40: Pretvornik priključen na V/I vodilo mikrokrmilnika Arduino

6.2.3 Testiranje prototipa pulznega pretvornika

Za testiranje prototipnega vezja sem na pulzni pretvornik priklopil funkcijski generator (Slika 41). Na generatorju sem najprej izbral frekvenco nekaj Hz. Pri preverjanju točnosti izmerjenih podatkov nisem zaznal nepravilnosti. Ko pa sem frekvenco povečeval na nekaj kHz, se je na računalniku pojavilo odstopanje izmerjene frekvence od frekvence, ki je bila nastavljena na generatorju. Prvotno sem mislil, da je odstopanje posledica slabe nastavitve časovne prekinitve v programu mikrokrmilnika. Kasneje pa sem s pomočjo uporabnikov spletnega portala Arduina ugotovil, da je lahko napaka posledica oscilatorja, ki določa takt delovanja mikrokrmilnika in vpliva na točnost časovne prekinitve. Večina mikrokrmilnikov ima že vgrajen oscilator, a ta običajno ne omogoča natančnega merjenja časa brez ustreznega kristalnega oscilatorja. Mikrokrmilnik s keramičnim oscilatorjem je napačno izmeril frekvenco, ki se je od izvorne frekvence razlikovala za 5 Hz do 7 Hz. Z drugimi modelom mikrokrmilnika Arduino Uno, ki je imel kvarčni oscilator, so bile meritve točne tudi pri višjih frekvencah (nekaj kHz). Oba mikrokrmilnika imata nazivni takt 16 MHz, vendar so, kot

navaja vir (http://www.token.com.tw/pdf/resonator/crystal_VS_ceramic_resonator.pdf, zadnjič obiskano 23. marca 2018), keramični oscilatorji manj natančni.

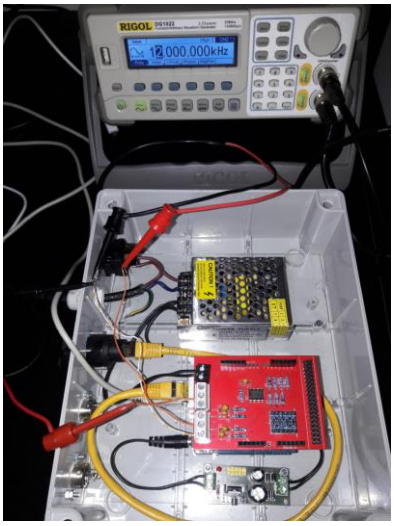


Vir: Lasten

Slika 41: Testiranje prototipa pretvornika

6.2.4 Testiranje pulznega pretvornika v obliki ščita

Po končanem sestavljanju merilne enote sem testiranje merjenja ponovil v enakem zaporedju, kot sem to storil s prototipnim vezjem. Merilno enoto sem tokrat priključil na omrežno napetost 220 V in pričel z izvajanjem meritev. Uporabil sem podoben postopek kot pri testiranju prototipnega pulznega pretvornika, vendar sem tokrat uporabil oba izhoda frekvenčnega generatorja, ki sem ju priključil na vhod za priklop sonde SBE-3 in SBE-4. Slika 42 prikazuje frekvenčni generator, ki je s pomočjo priključnih sponk rdeče in črne barve povezan na vhoda pulznega pretvornika, namenjena sodni SBE-3 in SBE-4.



Vir: Lasten

Slika 42: Testiranje pretvornika merilne enote

Frekvenco pulznega pretvornika sem odčital tako, da sem merilno enoto priklopil na lokalno omrežje. S pomočjo brskalnika na osebnem računalniku sem se povezal na IP naslov merilne enote. Merilna enota odgovori z odgovorom v XML obliki. Slednja vsebuje podatke o izmerjeni frekvenci za sondo SBE-3 in sondo SBE-4. Pri izvajanju meritev sem zasledil določena odstopanja, ki so sprva bila posledica neozemljenih priključkov, kamor sta povezani sondi. Priključke sem ozemljil in na vhoda merilne enote oz. pulznega pretvornika dodal kondenzatorja velikosti 100 nF. Vloga kondenzatorja na vhodu pulznega pretvornika je preprečitev motenj, ki se pojavijo v okolici priključkov sonde ali pulznega pretvornika zaradi zunanjih vplivov elektromagnetnih motenj (https://sl.wikipedia.org/wiki/Kondenzator, zadnjič obiskano 29. avgusta 2018). Pri izvajanju nadaljnjih meritev nisem zaznal nepravilnosti. Stikalni usmernik ni povzročil motenj.

6.3 Merilna enota

Pri razvoju merilne enote sem želel vse komponente, ki sestavljajo merilno enoto, postaviti v plastično ohišje. Uporabil sem električno nadometno dozo dimenzij 240 × 190 × 90 mm. Nadometna doza je izdelana iz umetne mase (PVC), na vrhnji strani ima snemljiv pokrov, ki ga obdajajo štirje vijaki. Pod pokrovom se nahaja gumijasto tesnilo, ki preprečuje vdor prahu, vlage in vode.

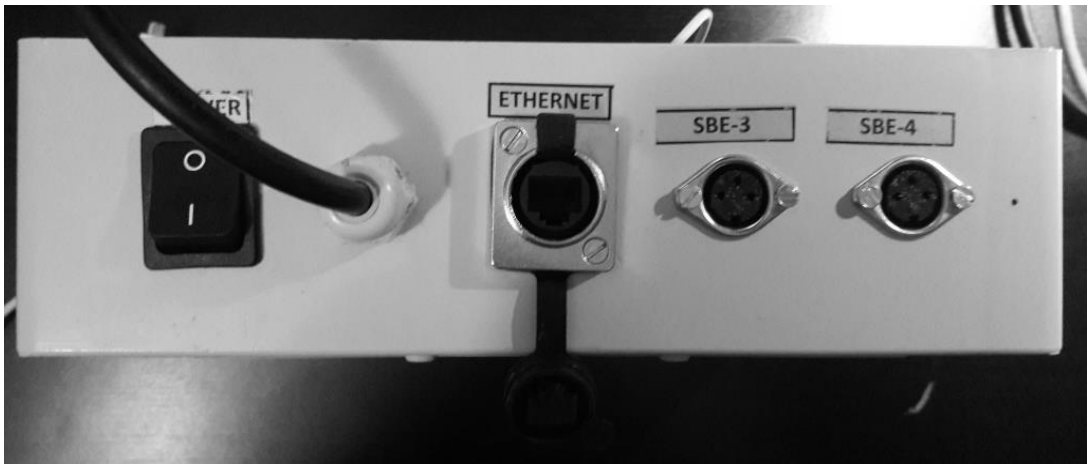
Vir: http://www.elektroklik.si/Stopnje_IP_zascite

Slika 43: Nadometna doza uporabljena kot ohišje

Doza ustreza standardu IP56. Stopnja zaščite IP je običajno izražena kot dvomestno število. Prva številka določa velikost mehanskih delcev, ki še lahko prodrejo v notranjost. Druga številka določa vodotesnost oz. pod kakšnimi pogoji je notranjost še zaščitena pred vdorom vode. Na daljši stranici električne doze sem najprej označil pozicijo za glavno stikalo, uvodnico za napajalni kabel, ethernet priključek in priključka za povezavo s sondama SBE-3 in SBE-4. Pri izdelavi lukenj sem si pomagal s svedri različnih dimenzij ter vbodno žago za rezanje odprtine glavnega stikala.

V izvrtana mesta sem nato vstavil potrebne elemente, ki omogočajo napajanje in povezavo merilne enote z električnim omrežjem, ethernet omrežjem in sondama (Slika 44):

- Dvopolno stikalo za vklop in izklop, ki je pravokotne oblike, je označeno z dvema simboloma '0' izklop ter '1' vklop. Stikalo je dimenzionirano za napetosti do 500 V.
- Uvodnica z napajalnim kablom. Skozi njo napeljemo kabel, ki ga potrebujemo za napajanje merilne enote. Uvodnica preprečuje, da bi v notranjost ohišja, kjer je napeljan napajali kabel, vdrli prah, vlaga ali voda.
- Priključek za ethernet kabel za priklop mikrokrmilnika z ethernet omrežjem. Uporabljeni priključek ustreza standardu IP56.
- Priključka DIN 3 za priklop sond.

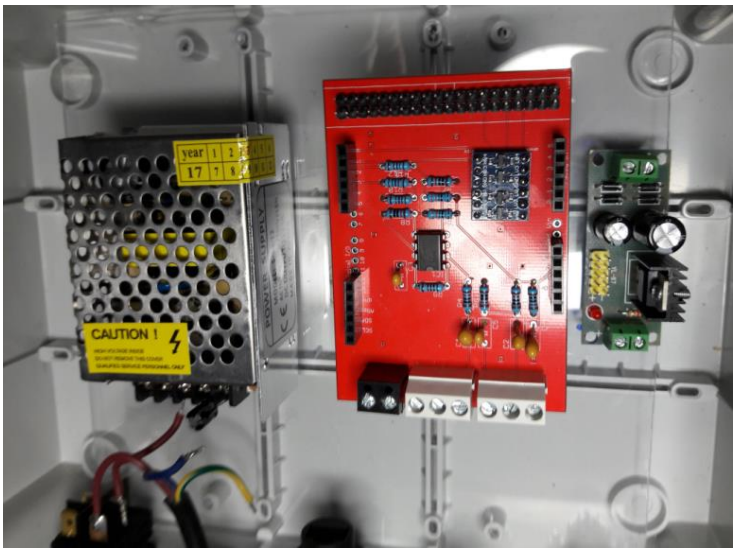


Vir: Lasten

Slika 44: Sprednja stran merilne enote s priključki

V notranjost ohišja sem na dno postavil ploščo iz pleksi stekla za montažo elektronskih komponent z vijaki. Pleksi plošča je privijačena v plastične nosilce, ki so na dnu ohišja. Pri postavitvi komponent sem pretvornik napetost postavil nekoliko dlje od mikrokontrolerja.

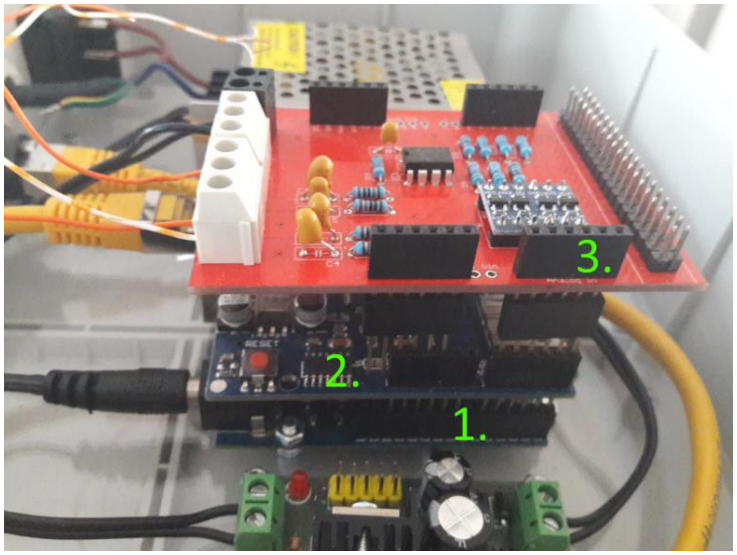
Vrstni red postavitve od leve proti desni je sledeč: napajalnik, mikrokrmilnik s pripadajočimi ščiti ter dodaten stabilizator napetosti za napajanje mikrokrmilnika. Postavitev prikazuje Slika 45.



Vir: Lasten

Slika 45: Postavitev komponent v notranjosti ohišja

Pogled na mikrokontroler iz drugega zornega kota pokaže, da se nad tiskanino mikrokontrolerja na dnu nahaja ščit, ki vsebuje povezavo z ethernet omrežjem in konektorjem za priklop pomnilniške mikro SD kartice. Kot drugi ščit mikrokontrolerja se nad njim nahaja pretvornik signala sond SBE-3 in SBE-4 (Slika 46).



Vir: Lasten

Slika 46: 1.) mikrokontroler, 2.) Ethernet in SD ščit, 3.) Pretvornik signala

Slika 47 prikazuje notranjost merilne enote in njene sklope, ki so označeni s številkami od 1 do 6. Napajalni kabel je s stikalom (št. 1) za vklop in izklop povezan s pretvornikom (napajalnikom) napetosti (št. 2). Pretvornik napetosti služi pretvorbi izmenične napetosti 220 V v enosmerno napetost 12 V. Izhod pretvornika je povezan z dodatnim stabilizatorjem napetosti (št. 3), kateri napetost 12 V zmanjša na 8 V. Ta služi za napajanje mikrokrmilnika in ščitov. Sondi SBE-3 in SBE-4 se napajata z enosmerno napetostjo 12 V. Konektorja bele barve, ki se nahajata na pretvorniku signala (št. 4), sta povezana s tripolnim DIN konektorjem, na katerega prideta priključeni sondi (št. 5). Tripolni DIN konektor ima tri priključne pine. Pri tem se dva uporabljata za napajanje sonde, medtem ko se tretji uporablja za signal iz sonde. Ethernet modul, ki se na sliki nahaja pod pretvornikom napetosti, je s krajšim UTP kablom rumene barve povezan na konektor za priklop na ethernet omrežje (št. 6).

Vir: Lasten

Slika 47: Notranjost merilne enote

6.4 Rezultati meritev pulznega pretvornika

Tabela 2 prikazuje sledeče podatke:

- Temperatura vodne kopeli je podatek iz kalibracijskega lista. To je temperatura vode, ki so jo v laboratoriju uporabili za kalibriranje sonde SBE-3.
- Frekvenca signala sonde nam pove, kolikšna je bila frekvenca signala sonde SBE-3, za podano temperaturo vodne kopeli in je prav tako zapisana na kalibracijskem listu.

- Izračunana temperatura pri kalibraciji je podatek, ki je izračunan po izrazu (1) in je zapisan na kalibracijskem listu.
- Izmerjena frekvenca pulznega pretvornika je pridobljena na sledeči način: na frekvenčnem pretvorniku sem za testiranje štetja impulzov nastavil enako frekvenco izhodnega signala, kot je zapisana na kalibracijskem listu za posamezno temperaturo vodne kopeli. S pomočjo računalnika sem nato odčital izmerjeno frekvenco pulznega pretvornika in jo zapisal v tabelo. Z izrazom (1) sem za izmerjeno frekvenco pulznega pretvornika izračunal temperaturo.
- Podatek $_{\Delta}$T (°C) je razlika med temperaturo, ki je bila izmerjena v laboratoriju, in temperaturo, ki jo izračuna mikrokrmilnik glede na pridobljeno frekvenco iz pulznega pretvornika.

Tabela 2: Primerjava podatkov iz kalibracijskega servisa in pulznega pretvornika

Temperatura vodne kopeli pri kalibraciji (°C)	**Frekvenca signala sonde SBE-3 pri kalibraciji (Hz)**	**Izračunana temperatura z izrazom (1) pri kalibraciji (°C)**	**Izmerjena frekvenca pulznega pretvornika (Hz)**	**Izračunana temperatura po izrazu (1) (°C)**	**$_{\Delta}$T (°C)**
-1,5000	6336,586	-1,5000	6337,00	-1,4971	0,0029
1,0000	6701,419	1,0000	6701,00	1,0039	-0,0039
4,5000	7237,290	4,4999	7237,00	4,4981	0,0018
8,0000	7803,210	8,0000	7803,00	8,0047	-0,0047
11,5000	8399,960	11,5001	8400,00	11,5003	-0,0002
15,0000	9028,308	15,0001	9029,00	15,0039	-0,0038
18,5000	9688,954	18,4999	9689,00	18,5002	-0,0003
22,0000	10382,697	21,9999	10383,00	22,0014	-0,0015
25,5000	11110,235	25,5000	11110,00	25,5036	-0,0036
29,0000	11872,182	29,0001	11873,00	29,0037	-0,0036
32,5000	12669,173	32,5000	12669,00	32,4992	0,0008

6.5 Postavitev razvojnega okolja

Pred razvojem spletne aplikacije je potrebno ustvariti razvojno okolje, ki ga sestavljata aplikacijski strežnik Apache Tomcat in relacijska baza MySQL.

6.5.1 **Kreiranje baze podatkov MySQL in tabel**

Na osebni računalnik sem namestil podatkovno bazo MySQL verzije 5.7. Bazo sem prenesel s spletnega naslova (https://dev.mysql.com/downloads/mysql/, zadnjič obiskano 30. marca 2018). Privzeti IP naslov baznega strežnika je 127.0.0.1 ali »localhost«. Baza podatkov sprejema podatke na privzetih vratih 3306. Za uporabo in upravljanje baze podatkov se s programom »mysql« povežemo na bazni strežnik. To storimo s sledečim ukazom:

```
C:\>mysql –u root –p *****
```

Novo bazo podatkov ustvarimo z ukazom:

```
mysql> CREATE DATABASE IF NOT EXISTS `MBP_AKVARIJ`;
```

Kreirano bazo podatkov izberemo z ukazom:

```
mysql> USE `MBP_AKVARIJ`;
```

Pri načrtovanju aplikacije sem kreiral dve tabeli. Tabela »met_meritve« je namenjena shranjevanju podatkov iz merilne enote. Vsebuje sedem atributov, ki so prikazani v spodnji tabeli.

Tabela 3: Podatkovna struktura tabele MET_MERITVE

Naziv atributa	**Podatkovni tip**	**Opis**
ID	Int(11) Autoincrement	Primarni ključ tabele, ki se avtomatsko povečuje (angl. Autoincrement)
TEMP	Decimal (10,5) default null	Atribut, namenjen shranjevanju temperature v °C na 3 decimalna mesta
SAL	Decimal (10,5) default null	Atribut, namenjen shranjevanju slanosti na 3 decimalna mesta
COND	Decimal (10,5) default null	Atribut, namenjen shranjevanju prevodnosti na 3 decimalna mesta
FREQ_TEMP	Decimal (10,2) default null	Atribut, namenjen shranjevanju frekvence sonde SBE-3
FREQ_COND	Decimal (10,2) defautl null	Atribut, namenjen shranjevanju frekvence sonde SBE-4
DAT_VNO	Datetime	Atribut, namenjen shranjevanju datuma in ure (časovna značka)

Kreiranje tabele »met_meritve« sprožimo s sledečim ukazom:

```
mysql> CREATE TABLE `met_meritve` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `TEMP` decimal(10,5) DEFAULT NULL,
  `SAL` decimal(10,5) DEFAULT NULL,
  `COND` decimal(10,5) DEFAULT NULL,
  `FREQ_COND` decimal(10,2) DEFAULT NULL,
  `FREQ_TEMP` decimal(10,2) DEFAULT NULL,
  `DAT_VNO` datetime DEFAULT NULL,
  PRIMARY KEY (`ID`),
  KEY `datvno` (`DAT_VNO`)
) ENGINE=InnoDB AUTO_INCREMENT=27515 DEFAULT CHARSET=utf8;
```

Tabela »met_nastavitve« shranjuje določene nastavitve, ki jih spletna aplikacija uporablja za komunikacijo z merilno enoto. Vsebuje štiri atribute (Tabela 4).

Tabela 4: Podatkovna struktura tabele MET_NASTAVITVE

Naziv atributa	**Podatkovni tip**	**Opis**
ID	Int(11) Autoincrement	Primarni ključ tabele, ki se avtomatsko povečuje (angl. Autoincrement)
SIFRA	Varchar (255) default null	Atribut, namenjen shranjevanju »šifre« ali »ključa«
VREDNOST	Varchar (1000) default null	Atribut, namenjen shranjevanju vrednosti, ki jo ima neka »šifra« ali »ključ«
OPIS	Varchar (1000) default null	Atribut, namenjen shranjevanju opisa ključa ali šifre. Kratek opis, zakaj se določena šifra v aplikaciji uporablja

Kreiranje tabele »met_nastavitve« sprožimo s sledečim ukazom:

```
mysql> CREATE TABLE `met_nastavitve` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `SIFRA` varchar(255) COLLATE utf8_slovenian_ci DEFAULT NULL,
  `VREDNOST` varchar(1000) COLLATE utf8_slovenian_ci DEFAULT NULL,
  `OPIS` varchar(1000) COLLATE utf8_slovenian_ci DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COLLATE=utf8_slovenian_ci;
```

Po kreiranju tabel lahko njihov obstoj preverimo z ukazom:

```
mysql> show tables;
+-----------------------+
| Tables_in_mbp_akvarij |
+-----------------------+
| met_meritve           |
| met_nastavitve        |
+-----------------------+
```

Pri zagotavljanju varnosti je smiselno ustvariti uporabnika (uporabniško ime in geslo), s katerim se bo aplikacijski strežnik povezoval na bazo podatkov, ter določiti pravice uporabnika, ki jih bo imel nad tabelami v podatkovni bazi. Za dostop do spletne aplikacije sem na bazi podatkov ustvaril uporabnika »mbp_akvarij«, ki ima geslo »1234!TEST«. To storimo z ukazom:

```
CREATE USER  'mbp_akv'@'localhost' IDENTIFIED BY '1234!TEST';
```

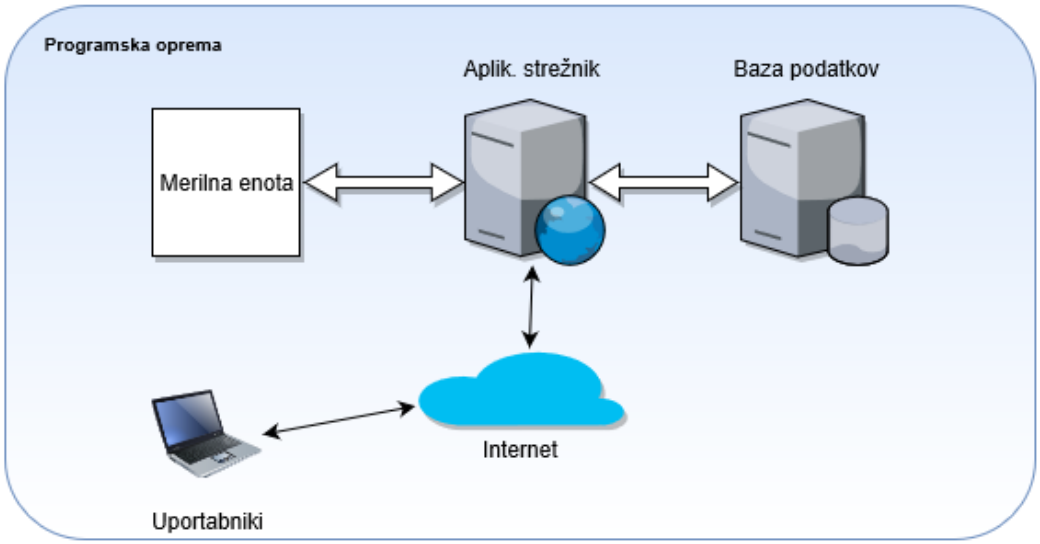
Uporabniku je potrebno določiti še pravice za upravljanje s tabelami v podatkovni bazi »mbp_akvarij«. S spodnjim ukazom določimo uporabniku pravico do branja podatkov, vnosu podatkov in ažuriranju podatkov. Zaradi varnosti sem opcijo brisanja podatkov (angl. Delete) onemogočil.

```
GRANT SELECT,INSERT,UPDATE ON MBP_AKVARIJ.* TO 'mbp_akv'@'localhost';
```

Postopek kreiranja baze podatkov, podatkovnih tabel in pravic je tako zaključen in pripravljen za uporabo. Podrobnejše informacije glede upravljanja z bazo podatkov so dosegljive na spletnem naslovu (https://dev.mysql.com/doc/refman/5.7/en/, zadnjič obiskano 30. marca 2018).

Nastavitev aplikacijskega strežnika Apache Tomcat je podrobneje opisana v dodatku H.7 »Apache Tomcat aplikacijski strežnik« na strani 125.

7 Spletna programska oprema za prikaz podatkov



Vir: Lasten

Slika 48: Arhitektura sistema za prikaz podatkov

Za prikaz podatkov iz merilne enote sem izdelal program, ki omogoča asinhrono komunikacijo z merilno enoto in zapis izmerjenih podatkov v podatkovno bazo ter spletno aplikacijo, ki omogoča uporabniku interaktivni prikaz podatkov za izbrano časovno obdobje. Pri razvoju programa sem se odločil za trinivojsko arhitekturo, ki jo sestavljajo uporabniški nivo (uporabniki), poslovni nivo (aplikacijski strežnik) in nivo dostopa (baza podatkov).

Uporabniški nivo omogoča uporabniku dostop do aplikacije. Ta sloj predstavlja uporabniku prikaz podatkov in upravljanje s podatki. Dve glavni vrsti uporabniškega vmesnika za omenjeni sloj sta tradicionalna aplikacija in spletna aplikacija. Novejše spletne aplikacije vsebujejo enake funkcionalnosti, kot jih uporabljajo tradicionalne aplikacije. To dosežemo z uporabo ustreznega programskega jezika in aplikacijskega strežnika, s katerim lahko generiramo dinamično HTML stran.

Poslovni nivo je sestavljen iz poslovnih pravil in podatkov. Komponente, ki tvorijo ta sloj, obstajajo na aplikacijskem strežniku (strežniškem računalniku). Poenostavljeno zapisano je poslovni nivo aplikacija, ki se izvaja po napisanem algoritmu ter pravilih in se izvaja na aplikacijskem strežniku.

Nivo dostopa vsebuje načine za povezovanje z bazo podatkov in za dogodke, ki so povezani z vstavljanjem, ažuriranjem, brisanjem in branjem (pridobivanjem) podatkov iz baze podatkov. Nivo dostopa komunicira z bazo podatkov z ustreznimi knjižnicami (API-ji). Pri razvoju programa za pregled izmerjenih podatkov sem uporabil orodja, ki so brezplačna in omogočajo platformsko neodvisnost. Izbral sem programski jezik Java in JSP (angl. Java Server Pages). Za shranjevanje podatkov sem izbral podatkovno bazo MySQL, saj to uporabljajo tudi na MBP NIB za shranjevanje drugih meritev. Celotna aplikacija za prikaz podatkov teče na Apache Tomcat strežniku. Programsko kodo sem napisal z uporabo programskega orodja Netbeans. Uporabljeni programski jeziki in knjižnice, ki sem jih uporabil pri razvoju programa, so opisane v »Dodatek H-: Programski in skriptni jeziki, uporabljeni v nalogi« na strani 123 (Mavsar G., 2017).

7.1 Prikaz in zapis podatkov iz merilne enote

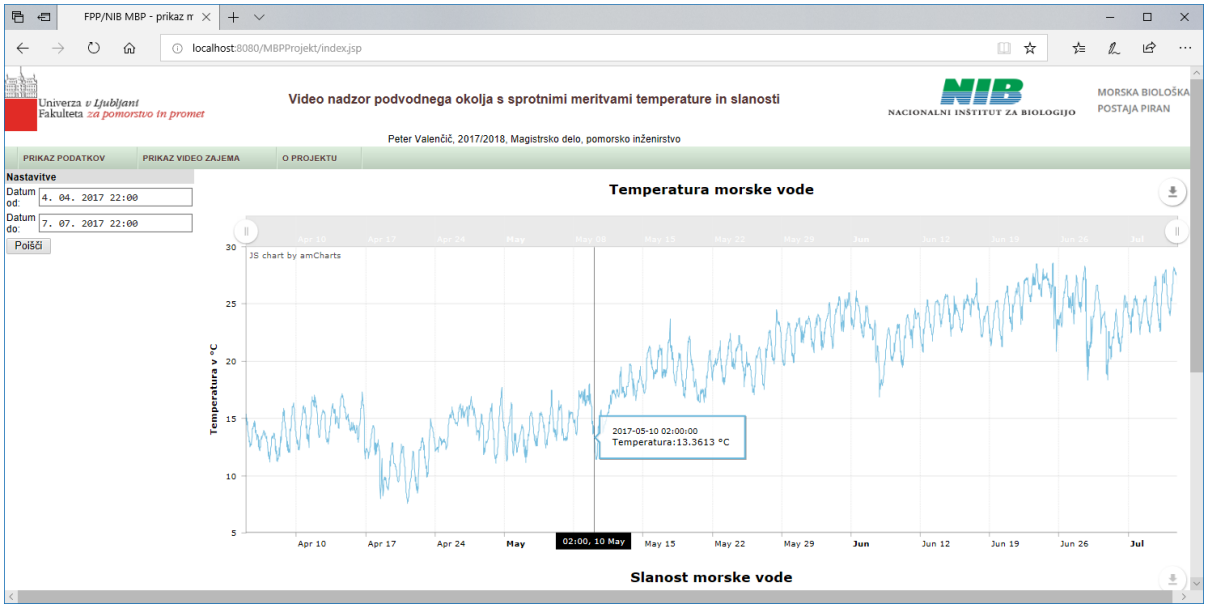
Program za prikaz in zapis podatkov je izdelan v programskem jeziku Java z uporabo spletnih tehnologij Javascript, JSP (Java server pages), servletov in pomožnih knjižnic za izpostavitev podatkov s pomočjo REST servisov. Za delovanje aplikacije sem uporabil aplikacijski strežnik Apache Tomcat 7.0. Lahko pa bi aplikacijo poganjal poljubni aplikacijski strežnik, ki podpira tehnologijo Java EE6.0 (http://www.oracle.com/technetwork /java/javaee/tech/javaee6technologies-1955512.html, zadnjič obiskano 2. aprila 2018).

Programska oprema je zgrajena iz treh podprogramov, ki so:

- Servisni program za komunikacijo z merilno enoto in zapis podatkov v bazo podatkov, opisan v »Dodatek I-: Program za asinhroni prenos podatkov v podatkovno bazo in posredovanje podatkov s pomočjo spletne storitve« na strani 129.
- REST servisi za asinhrono izmenjavo podatkov med spletno aplikacijo (prikazom podatkov) in bazo podatkov, opisan v dodatku I.2 »Posredovanje podatkov s spletno storitvijo (Web Service)« na strani 132.
- Spletna aplikacija za prikaz podatkov, kjer so izrisani podatki za določeno časovno obdobje z uporabo Javascript tehnologije in REST servisa, je opisana v naslednjem razdelku.

7.2 Spletna aplikacija za prikaz podatkov

Aplikacija za prikaz podatkov (Slika 49) je namenjena končnemu uporabniku za pregled, prikaz in izvoz podatkov merilne enote. Razvita je s pomočjo javanske tehnologije za razvoj spletnih strani JSP (Java Server Pages) in servletov (Java Servlets). JSP stran je v osnovi HTML dokument, ki vsebuje Java kodo, uvedeno s posebnimi oznakami. Ker gre za spletno aplikacijo, je tako možno podatke pregledovati na vseh napravah, s katerimi lahko prikazujemo spletne strani (osebni računalnik, tablica, telefon …).



Vir: Lasten

Slika 49: Aplikacija za prikaz podatkov

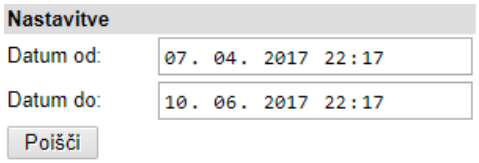
Spletna aplikacija uporabniku omogoča prikaz podatkov, prikaz video zajema in informacijo o projektu. Slednje lahko uporabnik izbere iz menijske vrstice aplikacije (Slika 50).

Vir: Lasten

Slika 50: Menijska vrstica

7.2.1 **Prikaz podatkov**

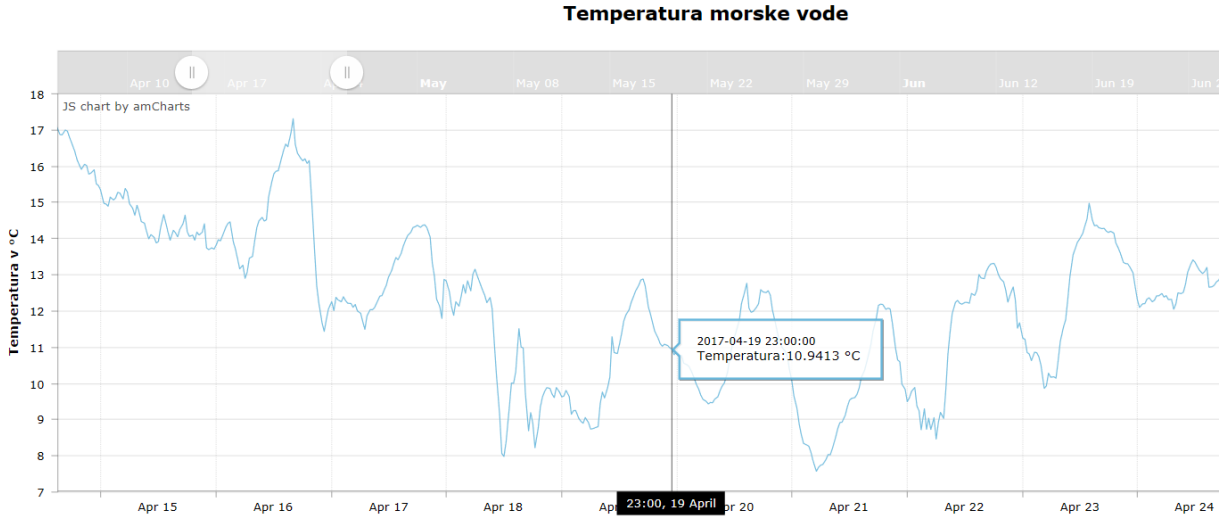
Uporabniku je omogočen pregled podatkov iz merilne enote. Ob prvem klicu spletne strani se uporabniku prikažejo podatki za zadnjih 24 ur od trenutnega datuma. Kasneje pa si lahko uporabnik z nastavitvami (Slika 51) določi datumski interval, za katerega želi izris podatkov. Maska za izbor datumskega intervala vsebuje dve vrednosti, ki sta »Datum od« in »Datum do«. Oba podatka predstavljata datum z uro in minutami (angl. Timestamp).



Vir: Lasten

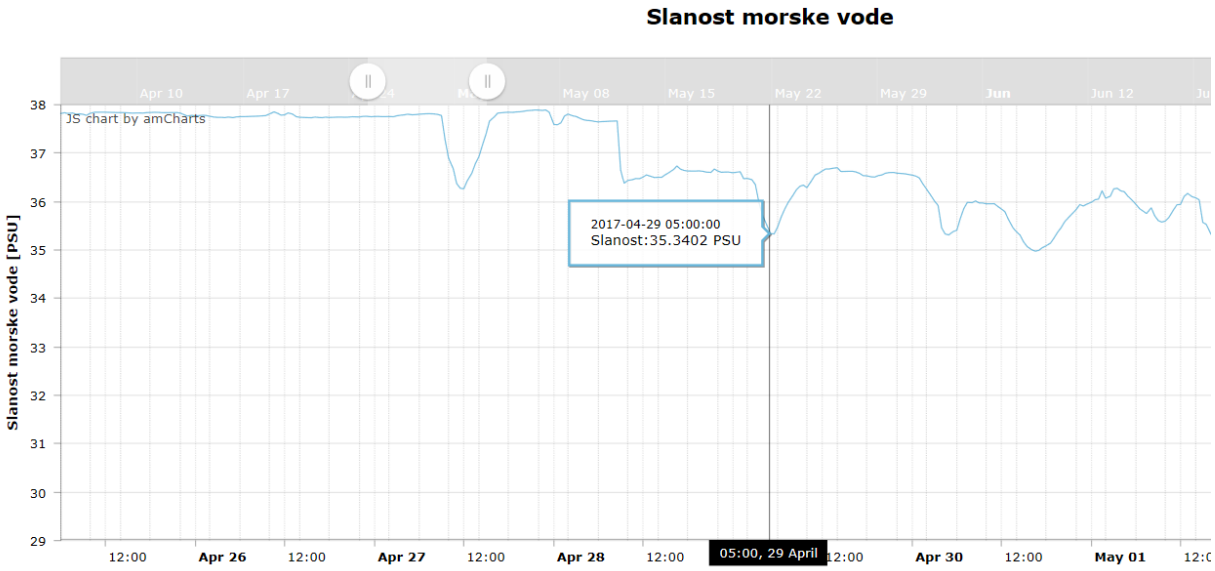
Slika 51: Nastavitve intervala

Vnesene vrednosti potrdimo z gumbom »Poišči«. Gumb »Poišči« omogoči klic spletne storitve, ki vrne podatke o temperaturi in slanosti v JSON obliki. Pridobljeni podatki se nato s komponento za izris grafa »amCharts« (https://www.amcharts.com/, zadnjič obiskano 7. aprila 2018) uporabniku izrišejo na zaslon. Na spletni strani sta dva grafa, ki prikazujeta temperaturo morske vode (Slika 52) in slanost morske vode (Slika 53).



Vir: Lasten

Slika 52: Graf temperature



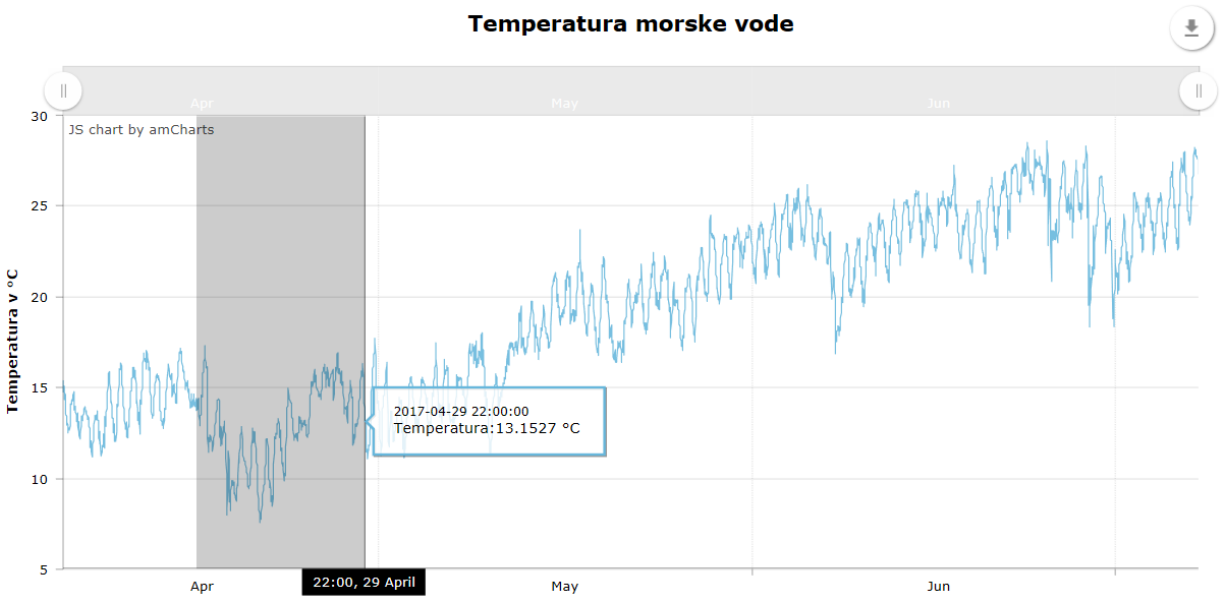
Vir: Lasten

Slika 53: Graf slanosti

Komponenta, ki prikazuje izrisani graf, omogoča tudi interaktivno delo s podatki. Med možnostmi, ki jih komponenta za izris grafa omogoča, so:

- Povečava podatkov z drsnikom.
- Povečava podatkov z miško.
- Grafični izvoz slike v različnih formatih (jpg, pdf, png, svg).
- Izvoz podatkov grafa v različnih formatih (csv, xslx, json).
- Označevanje in risanje na površini grafa.

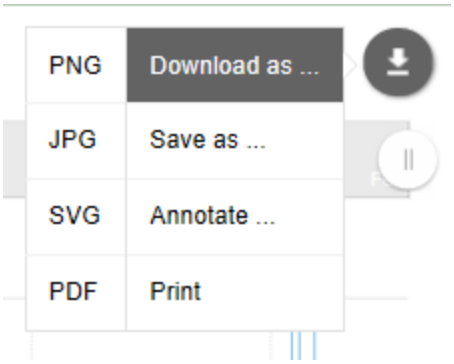
Podatke na grafu je možno povečati z drsnikom, ki je na zgornji strani grafa. Drsnik vsebuje dve mejni vrednosti, ki predstavljata časovni interval prikaza podatkov. Poleg tega je možno podatke na grafu poveča z miško tako, da uporabnik izbere začetek ob pridržanju levega gumba miške in izbere interval na grafu, ki se obarva sivo (Slika 54). Povrnitev grafa v začetno stanje je omogočena tako, da izberemo opcijo »prikaži vse« (angl. Show All), ki je na grafu izrisana v obliki povečevalne lupe.



Vir: Lasten

Slika 54: Povečava s pomočjo miške

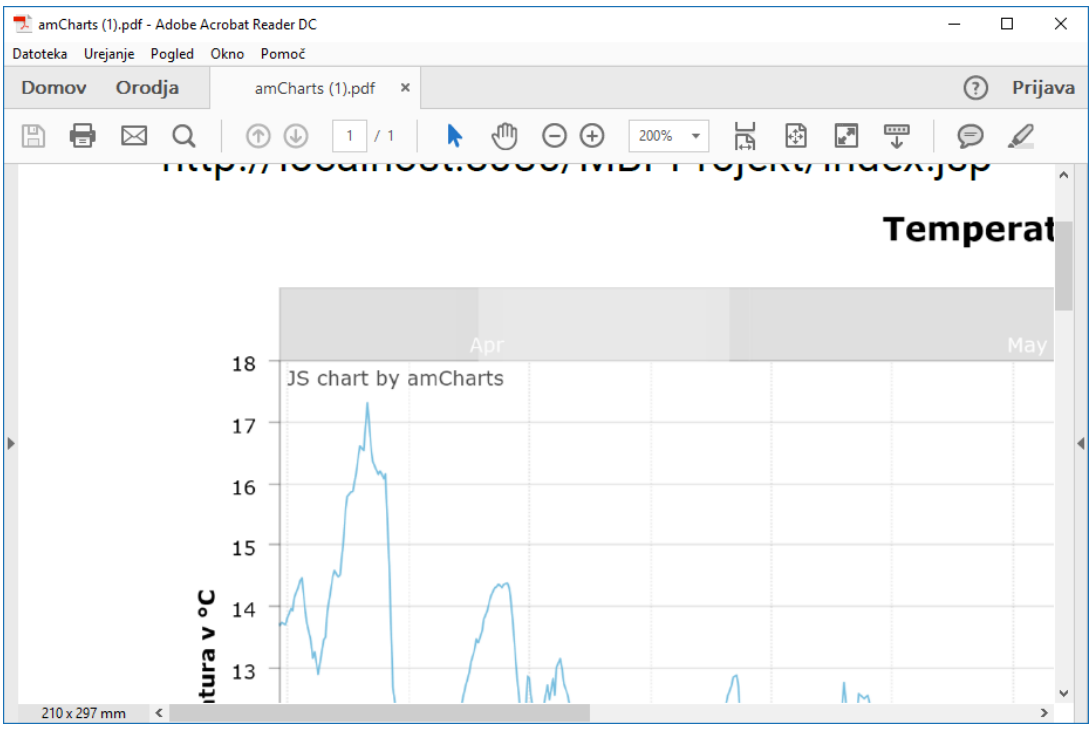
Komponenta za izris grafa omogoča tudi izvoz slike v različne formate datotek. Za izvoz podatkov je potrebno izbrati ikono, ki je prikazana na spodnji sliki in je na grafu desno zgoraj (Slika 55).



Vir: Lasten

Slika 55: Izvoz podatkov

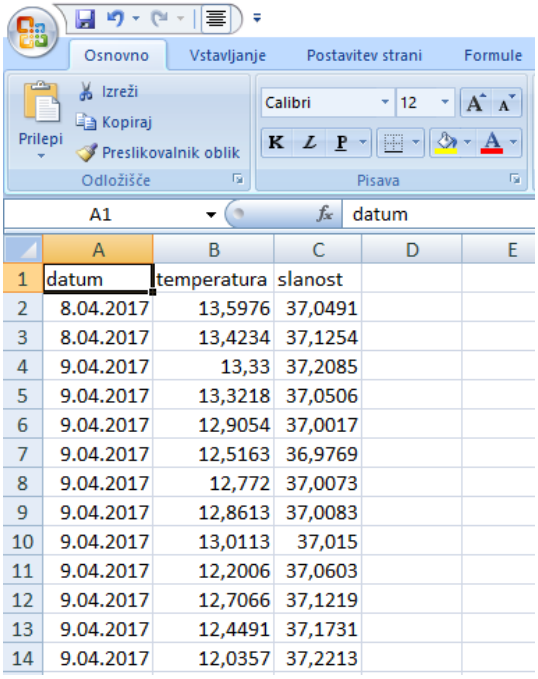
Uporabnik lahko iz seznama izbere željeni format. Razpoložljivi formati datotek so PNG, JPG, SVG ali PDF. Na sliki je prikazan izvoz grafa v PDF datoteko ter prikaz grafa v programu Adobe Acrobat Reader (Slika 56).



Vir: Lasten

Slika 56: Pregled izvožene datoteke v programu Acrobat Reader

Poleg grafičnega izvoza slik je možno izvoziti tudi podatke, ki so prikazani na grafu. Izvoz podatkov je možen v XLSX (Microsoft Excel), CSV (podatki so ločeni z vejico) in JSON (Slika 57).

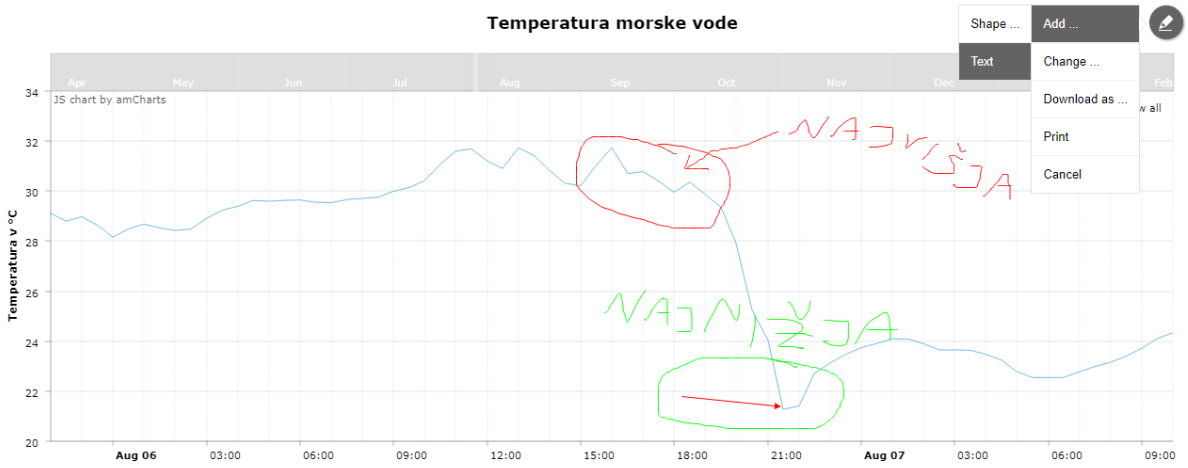


	A	B	C	D	E
1	datum	temperatura	slanost		
2	8.04.2017	13,5976	37,0491		
3	8.04.2017	13,4234	37,1254		
4	9.04.2017	13,33	37,2085		
5	9.04.2017	13,3218	37,0506		
6	9.04.2017	12,9054	37,0017		
7	9.04.2017	12,5163	36,9769		
8	9.04.2017	12,772	37,0073		
9	9.04.2017	12,8613	37,0083		
10	9.04.2017	13,0113	37,015		
11	9.04.2017	12,2006	37,0603		
12	9.04.2017	12,7066	37,1219		
13	9.04.2017	12,4491	37,1731		
14	9.04.2017	12,0357	37,2213		

Vir: Lasten

Slika 57: Pregled izvoženih podatkov v Microsoft Excel-u

Dodatna označitev podatkov na grafu je omogočena na površini grafa (angl. Annotation). Na grafu lahko z različnimi orodji (črta, puščica, prosto risanje) narišemo poljubne označbe, katerim lahko spremenimo barvo in debelino izrisa. Tako popravljen graf lahko ponovno izvozimo v različne grafične formate datotek ali ga natisnemo s tiskalnikom (Slika 58).

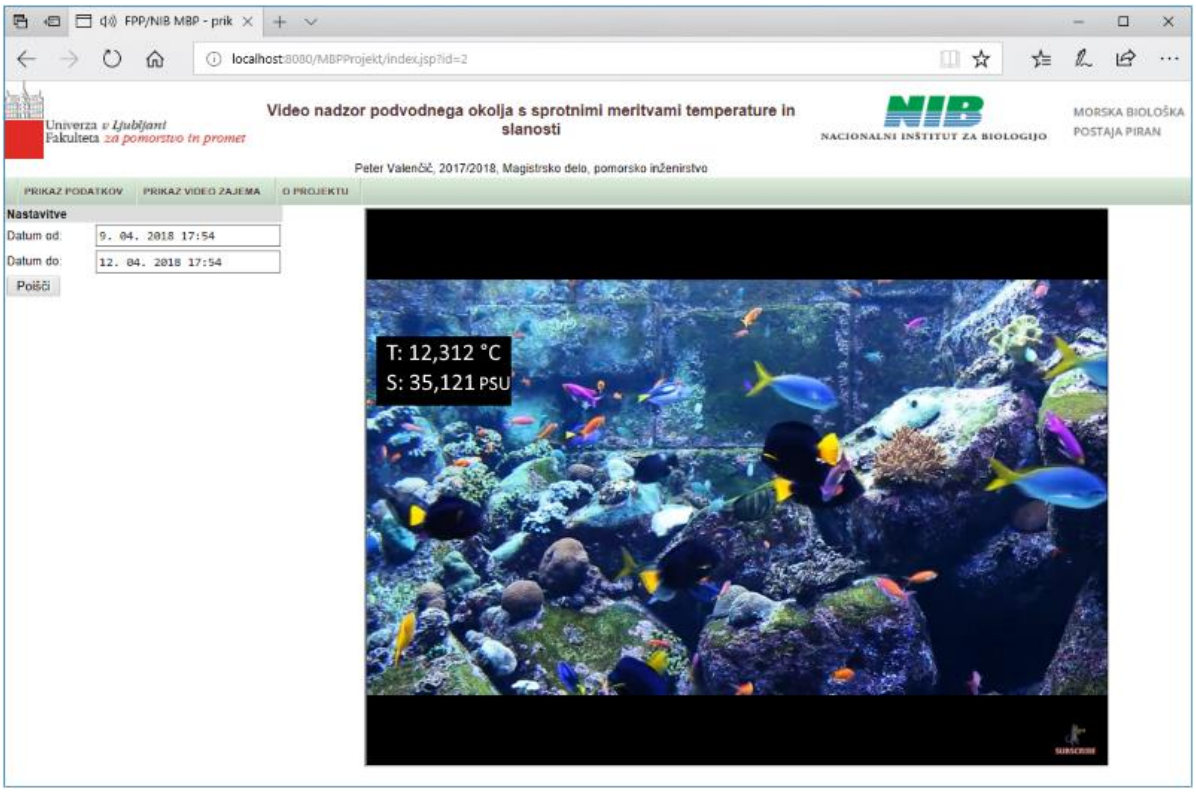


Vir: Lasten

Slika 58: Dodatna označitev grafa

7.3 Prikaz video zajema

Prikaz video zajema omogoča pregled video posnetka v realnem času. Video vsebina se predvaja s spletne strani Youtube (Slika 59), kamor jo dodatna programska oprema za zajem in enkodiranje pošilja v realnem času.

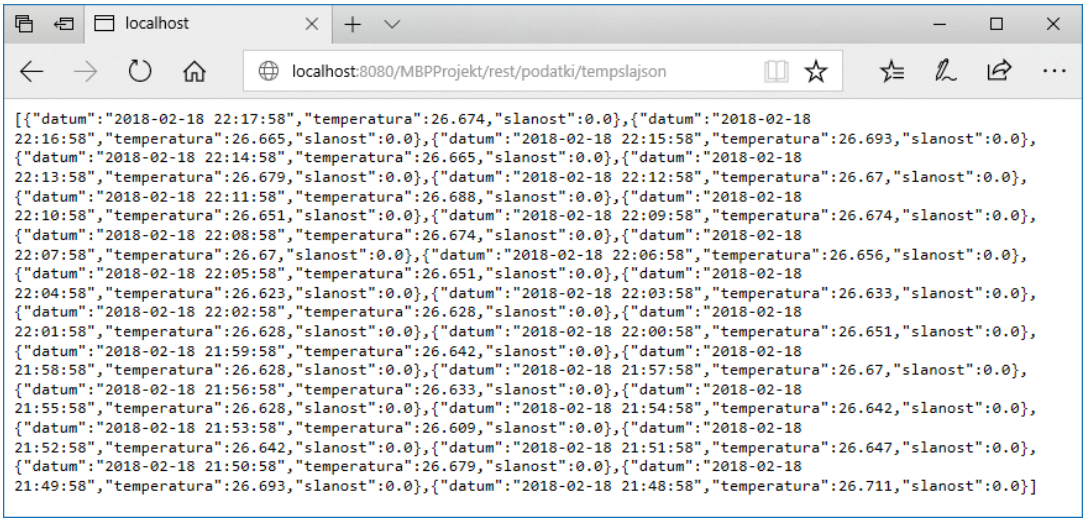


Vir: Lasten

Slika 59: Pregled video zajema s podatki iz merilne enote

7.4 Prikaz podatkov s pomočjo spletne storitve

Za pridobitev podatkov s spletno storitvijo zaženemo aplikacijski strežnik Apache Tomcat. Spletna storitev bo po zagonu aplikacijskega strežnika dosegljiva na URL naslovu: http://localhost:8080/MBPProjekt/rest/podatki/tempslajson. Ker je klic spletne storitve brez parametrov za datumsko omejitev, bo spletna storitev vrnila zadnjih trideset vnesenih zapisov iz baze, ki jih prikazuje Slika 60.



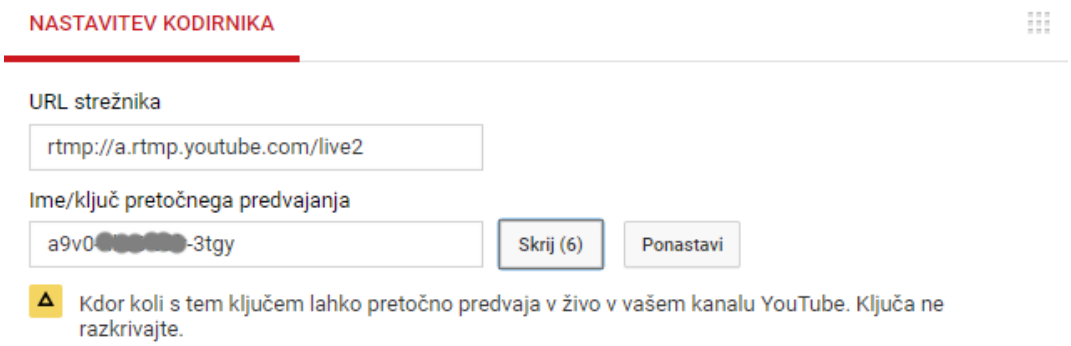
Vir: Lasten

Slika 60: Klic spletne storitve

Spletno storitev je možno uporabiti tudi z drugimi programskimi jeziki in programi. Tako je npr. možno v Excel vgraditi VBA skripto za pridobivanje podatkov o temperaturi in slanosti za določeno časovno obdobje. Razvoj spletne storitve je podrobneje opisan v »I.2 Posredovanje podatkov s spletno storitvijo (Web Service)« na strani 132.

7.5 Združevanje video zapisa in podatkov iz merilne enote ter posredovanje video vsebine na strežnik YouTube

Za predvajanje video vsebine na Youtube Live strežniku potrebujemo uporabniški račun, ki ga uporabnik pridobi z registracijo na spletni strani Youtubea. Po uspešni registraciji je potrebno pridobiti »ključ« (angl. Key), s katerim je možno pretakati podatke iz lokalnega računalnika na oddaljeni strežnik Youtube. Podatke o ključu uporabnik pridobi z uporabniškim menijem za pretočno predvajanje, ki ga ponuja spletna aplikacija Youtubea. Poleg ključa za pretočno predvajanje uporabnik pridobi tudi URL naslov strežnika, ki je v mojem primeru rtmp://a.rtmp.youtube.com/live2 (Slika 61). Ta naslov potrebujemo v programu FFMPEG, da določimo, na kateri naslov naj program pošilja enkodirano sliko (posnetek).



Vir: Lasten

Slika 61: Pridobitev URL naslova strežnika in ključa za pretočno predvajanje

Naslednji korak je nastavitev programa FFMPEG. Program FFMPEG mora delovati neprekinjeno, saj lahko v nasprotnem primeru zaradi napak izgubimo pretok posnetka na strežnik Youtube. Zaradi tega sem ustvaril zagonsko skripto, s katero nastavimo parametre in zaženemo program FFMPEG.

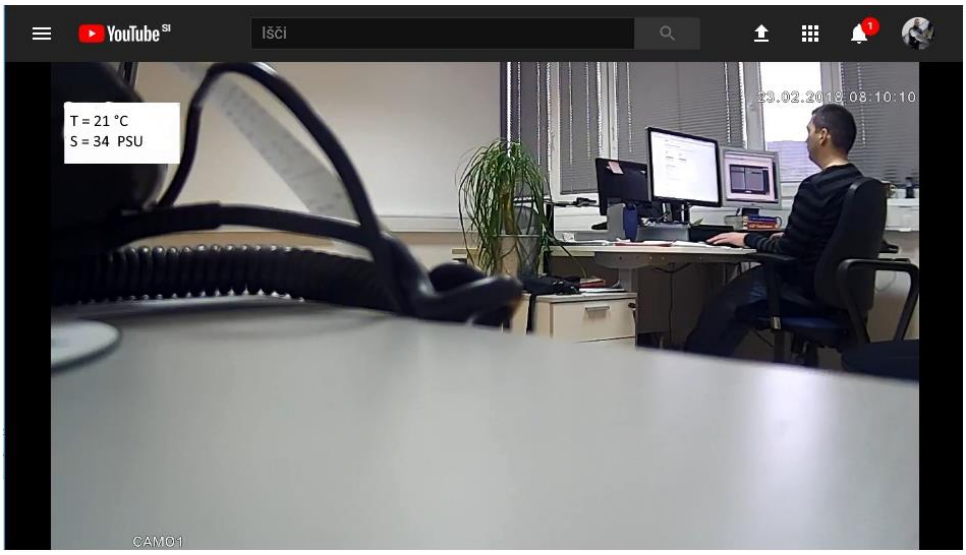
Primer skripte:

```
#!/bin/bash
# Parametri za kodirnik ter naslov strežnika youtube
VBR="1000k" # Bitrate of the output video, bandwidth 1000k = 1Mbit/s
QUAL="ultrafast" # Encoding speed
YOUTUBE_URL="rtmp://a.rtmp.youtube.com/live2" # RTMP youtube URL
THREADS="0" # 0 = autoselect

#Naslov podvodne kamere, ki je na svojem IP naslovu in vratih RTSP
SOURCE="rtsp://172.16.201.88:554/user=admin&password=&channel=1&stream=0.sdp?real_stream"

#Ključ za pretočno predvajanje (vnesemo iz youtube)
KEY="YOUTUBE KLJUČ"

#Uporabljena pisava za izpis podatkov v video posnetku
FONT="/usr/local/share/fonts/OpenSans-Regular.ttf"

#Velikost pisave
FONTSIZE="15"

# Pozicioniranje teksta
x="5"
y="60"

# Ostalo
box="1" # omogoči obrobo
boxcolor="black@0.5"
textfile="ffmpeg.txt"

# Naloži podatke iz tekstovne datoteke v vsakem frame-u
reloadtext="1"
boxborderwidth="5"

#Ta flag dodaj, da se ne izpisujejo smeti v konzolo -loglevel panic \

#Zagon programa FFMPEG
   ffmpeg -f lavfi -i anullsrc \
```

```
-rtsp_transport tcp \
-i "$SOURCE" \
-vcodec libx264 -pix_fmt yuv420p -preset $QUAL -g 20 -b:v $VBR \
-vf "drawtext="fontfile=${FONT}":textfile=${textfile}:x=${x}:y=${y}:reload=${reloadtext}: \
fontcolor=white:fontsize=${FONTSIZE}:box=${box}:boxborderw=${boxborderwidth}:boxcolor=${boxcolor}" \
-threads $THREADS -bufsize 512k \
-f flv "$YOUTUBE_URL/$KEY"
```

Pripravljeno skripto sem shranil z imenom »videopredvajanje.sh«. V terminalu operacijskega sistema Linux sem nato pognal program FileMover.class, ki je pričel komunicirati z merilno enoto in je prebrane podatke zapisoval v tekstovno datoteko z imenom ffmpeg.txt. V drugem terminalskem oknu sem pognal skripto z ukazom »sudo videopredvajanje.sh«, kar je omogočilo zagon programa FFMPEG in pričetek dekodiranja, enkodiranja ter pošiljanja video posnetka na strežnik Youtube.

Združevanje tekstovne datoteke z video posnetkom poteka v programu FFMPEG. Ker pri razvoju programske opreme nisem imel na razpolago akvarija, sem tako zajemal dogajanje v pisarni. Video vsebina se je z izvajanjem programa FFMPEG pričela predvajati na strežniku Youtube, kjer je bilo možno videti enkodirani video posnetek, ki vsebuje podatke iz tekstovne datoteke, v katero se zapisujejo podatki iz merilne enote. V fazi razvoja programske opreme sem uporabljal naključno vnesene podatke za vrednosti temperature in slanosti (Slika 62).



Vir: Lasten

Slika 62: Predvajanje video posneta v živo na spletni strani Youtube Live

Ker pa se program FFMPEG občasno lahko tudi zaustavi ali prekine, je potrebno na strežniku zagotoviti skripto, ki periodično preverja, ali je program FFMPEG aktiven. Za to sem uporabil dodatno skripto, ki sem jo poimenoval »check_ffmpeg.sh«, ki preverja, ali se program FFMPEG izvaja.

```
#!/bin/bash
#
# Preveri če teče ffmpeg in če ne teče reštarta proces
#
script=/path/to/videopredvajanje.sh

if ! pgrep -x "ffmpeg" > /dev/null
then
    /bin/bash $script > /dev/null 2>&1 &
fi
```

Skripta preverja, ali se med programi, ki delujejo, nahaja program FFMPEG. Če pogoj ni izpolnjen, se zažene skripta, ki je določena v spremenljivki »script«. Ta vsebuje lokacijo skripte »videopredvajanje.sh«. S pomočjo ukaza crontab (http://www.adminschoice.com/crontab-quick-reference, zadnjič obiskano 7. julija 2018), ki požene skripto »check_ffmpeg.sh«, se izvaja periodično preverjanje delovanja programa FFMPEG. Preverjanje delovanja programa FFMPEG poteka vsako minuto. To določa pet znakov za množenje, ki sledijo atributu »–e«. Periodično izvajanje skripte z ukazom:

```
chmod +x check_ffmpeg.sh
crontab -e
* * * * * sudo bash /pot_do_skripte/check_ffmpeg.sh
```

8 Razprava z zaključki

Povod za razvoj merilne enote je bila zamisel, da bi razvil široko dostopno aplikacijo meritve temperature in slanosti v vodnem okolju s sprotnim prikazovanjem žive slike tega okolja. Od NIB MBP sem prejel na preizkušnjo sondi SBE-3 in SBE-4, ki sta bili obnovljeni in na novo kalibrirani. Izhodišče naloge je bila izdelava merilne enote z uporabo poceni elektronskih komponent in programske opreme, ki temelji na odprtokodnih rešitvah za prenos, shranjevanje in »online« prikaz izmerjenih podatkov. Posebnost merilne enote je video zajem podvodnega okolja, ki ga sestavlja cenena ethernet kamera. Ta je vstavljena v vodotesno ohišje in skrbi za video zajem podvodnega dogajanja. Tako dobljeni video posnetek lahko uporabimo za nadaljnjo obdelavo ali ga s pomočjo programske opreme FFMPEG posredujemo na javni strežnik za video predvajanje, kateremu dodamo podatke o izmerjeni temperaturi ter slanosti morske vode. Merilna enota je sestavljena iz več delov, ki se jih lahko tudi spremeni ali nadgradi. Ta se deli na tri manjše enote, in sicer:

- Merilna enota, ki jo sestavljata sondi, pretvornik in mikrokontroler s pripadajočo programsko opremo za izvajanje meritev.
- Video zajem podvodnega okolja, ki ga sestavlja vodotesno ohišje, v katerega je vstavljena ethernet kamera, ki je s kablom povezana v LAN omrežje.
- Programska oprema, ki jo sestavlja spletna aplikacija za zajem, shranjevanje in prikaz podatkov, ter programska oprema za prenos video podatkov na splet oz. na strežnik za javno predvajanje video vsebin Youtube.

8.1 Težave pri zasnovi in razvoju merilne enote

Pri razvoju sem se srečal s kopico težav, med katerimi je večjo težavo povzročal mikrokrmilnik cenene izdelave, ki za delovni takt uporablja keramični oscilator. Ker meritve podatkov iz sond SBE-3 in SBE-4 temeljijo na štetju impulzov (merjenju frekvence), je odstopanje delovnega takta mikrokrmilnika vodilo k napačno izmerjeni frekvenci. Izmerjena frekvenca sond SBE-3 in SBE-4 je odstopala tudi za nekaj Hz od realne vrednosti. Posledica je bila napačna vrednost izmerjene temperature in prevodnosti. Po posvetovanju z mag. Francijem Henigmanom in prof. dr. Vladom Malačičem smo na MBP NIB izvedli dodatne meritve na mikrokrmilniku Arduino UNO, ki je za delovni takt uporabljal kvarčni oscilator. Pri tem smo zaznali minimalno odstopanje izmerjene frekvence od realne vrednosti frekvence

sonde SBE-3 in SBE-4, ki je v nekaterih meritvah odstopala za 1 Hz. Po opravljenih meritvah sem se lotil razvoja tiskanega vezja pretvornika, ki se ga kot ščit natakne na vhodno-izhodno vodilo mikrokrmilnika Arduino. Z razvojem tiskanih vezij nisem imel izkušenj, zato sem za razvoj načrta tiskanine uporabljal pomoč in prispevke na internetu. Tiskano vezje sem opremil z elektronskimi komponentami in sem ga priključil na vhodno-izhodno vodilo mikrokrmilnika. Delovanje vezja sem testiral z uporabo dvo-kanalnega frekvenčnega generatorja. Testne meritve sem izvajal s pomočjo podatkov, ki so zapisani na kalibracijskem listu in te primerjal z izračunanimi vrednostmi merilne enote. Rezultate sem spremljal s pomočjo računalnika.

Program za merjenje frekvence in preračun temperature, prevodnosti in slanosti mi je v začetku predstavljal določene težave. Prva različica programa je meritve frekvence opravljala v neskončni zanki. Težava je nastopila, ko sem želel pridobiti podatke iz merilne enote. Takrat se je proces štetja impulzov zaustavil, saj je program izvajal programsko kodo za posredovanje podatkov po ethernet povezavi. Rezultat je bila napačno izmerjena frekvenca sond SBE-3 in SBE-4. Težavo sem odpravil z uporabo časovnih in zunanjih prekinitev. Po testiranju delovanja programa za izračun vrednosti temperature, prevodnosti in slanosti sem pričel s sestavljanjem merilne enote. Komponente merilne enote: napajalnik, mikrokrmilnik, priključne sponke, stikalo in konektorje sem vstavil v plastično ohišje, ki je nadometna doza ustrezne velikosti. Povezovalne kable sond SBE-3 in SBE-4 sem podaljšal s trižilnim kablom. Kabel sem opremil s tripolnim DIN konektorjem. Tako je možno sondi v primeru servisnega posega ali zamenjave priključiti ali izključiti iz merilne enote.

Pri razvoju programske opreme za shranjevanje, posredovanje in prikaz podatkov sem uporabil odprtokodno programsko opremo. Podatki iz merilne enote se shranjujejo v podatkovno bazo MySQL. Za prikaz podatkov sem napisal spletno aplikacijo v programskem jeziku Java.

Največje težave sem imel z video zajemom podvodnega okolja. Zaradi uporabe cenenih kamer so mi le-te med testiranjem odpovedale, zato pričakujem tudi v bodoče podobne težave, v kolikor bo uporabljena kamera kitajskega proizvajalca HaSecurity, model Hi3516c. Smiselno bi bilo uporabiti bolj kvalitetno kamero cenovnega razreda od 50,00 € do 100,00 €.

8.2 Možne izboljšave merilne enote

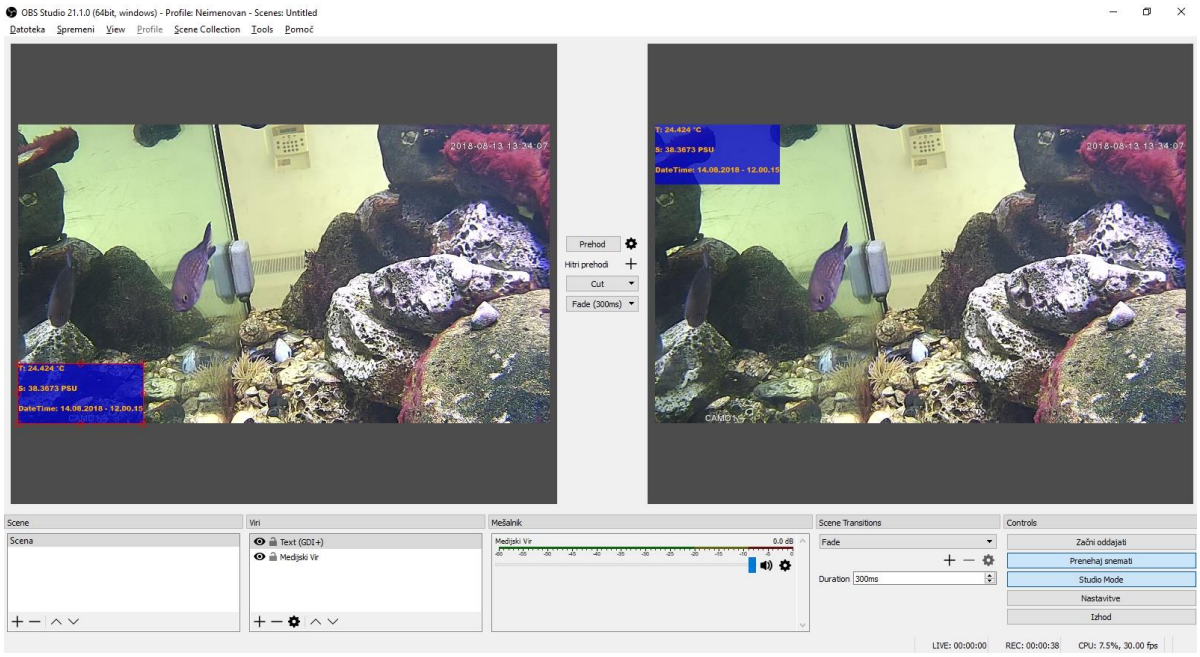
Merilno enoto je možno izboljšati na več načinov. Predvsem bi lahko optimiziral tiskano vezje merilne enote, ki je sedaj sestavljeno iz treh ščitov, ki zavzamejo kar nekaj prostora. Tiskano vezje bi lahko zasnoval tako, da bi bili na tiskanini prisotni: usmernik napetosti merilne enote, usmernik napetosti za sondi SBE-3 in SBE-4, mikrokontroler, integrirano vezje za uporabo ethernet povezave in čitalca SD kartic ter pretvornik. Velikost izdelane tiskanine ne bi presegala 100 mm × 50 mm. Lahko bi izbral manjšo zaščitno škatlo, v katero bi bila vstavljena tiskanina in napajalnik. Merilno enoto bi opremil z LCD zaslonom, kjer bi izpisoval trenutno izmerjene vrednosti.

Programsko opremo za beleženje in prikaz podatkov bi lahko napisal v katerem drugem programskem jeziku. Predvsem je v zadnjem času priljubljen programski jezik Node.js, ki je odprtokodno, večplatformno, JavaScript izvajalno okolje za razvoj različnih orodij in aplikacij. To okolje je v zelo kratkem času doživelo uspeh v svetovnem merilu in je danes ena izmed najbolj znanih platform za pisanje spletnih aplikacij. Izvajalno okolje interpretira z Googlovim V8 JavaScript pogonom, poglavitne temelje Node.js pa postavita dogodkovno orientiran model in asinhrone vhodno/izhodne operacije. Iz javnega repozitorija (https://www.npmjs.com/, zadnjič obiskano 5. septembra 2018) je možno prenesti knjižnice za povezavo z mikrokrmilnikom Arduino in knjižnice za povezavo z bazo podatkov (https://zmaga.com/content.php?id=8673, zadnjič obiskano 5. septembra 2018).

Izboljšati bi bilo možno tudi video zajem podatkov. Uporabili bi lahko kvalitetnejšo kamero, ki je priključena na USB vodilo. Kamera na USB vodilu ima kvalitetnejšo sliko in omogoča prenos video posnetka v več formatih kot kamera z ethernet povezavo. Težava je le ta, da bi v primeru USB kamere moral biti v bližini prisoten osebni računalnik, na katerega bi bila kamera priključena.

Video obdelava poteka z uporabo programa FFMPEG, ki deluje v terminalskem oknu operacijskega sistema Linux. Program FFMPEG je zmogljiv program za dekodiranje in enkodiranje video in avdio zapisov, vendar je za uporabo uporabniku neprijazen. Zajem video podatkov in podatkov iz merilne enote bi lahko bolje opravil s programom OBS studio (https://obsproject.com/download, zadnjič obiskano 11. julija 2018). Program OBS studio (Slika 63) potrebuje za delovanje operacijski sistem z grafičnim vmesnikom. Uporabo

programa OBS studio so že preizkusili na MBP NIB. Ta omogoča video zajem na datotečni sistem ali pretok video podatkov na različne strežnike za javno predvajanje video vsebine (npr. Youtube). Opremljen je z različnimi vtičniki (angl. Pluggins), ki so dostopni na spletni strani (https://obsproject.com/forum/resources/categories/obs-studio-plugins.6/?page=1, zadnjič obiskano 12. junija 2018). Vtičniki omogočajo dodatne funkcionalnosti, ki jih program v osnovi ne ponuja. V bodoče bi lahko napisal vtičnik, ki komunicira z merilno enoto in podatke v tekstovni obliki izpiše v oknu programa OBS.

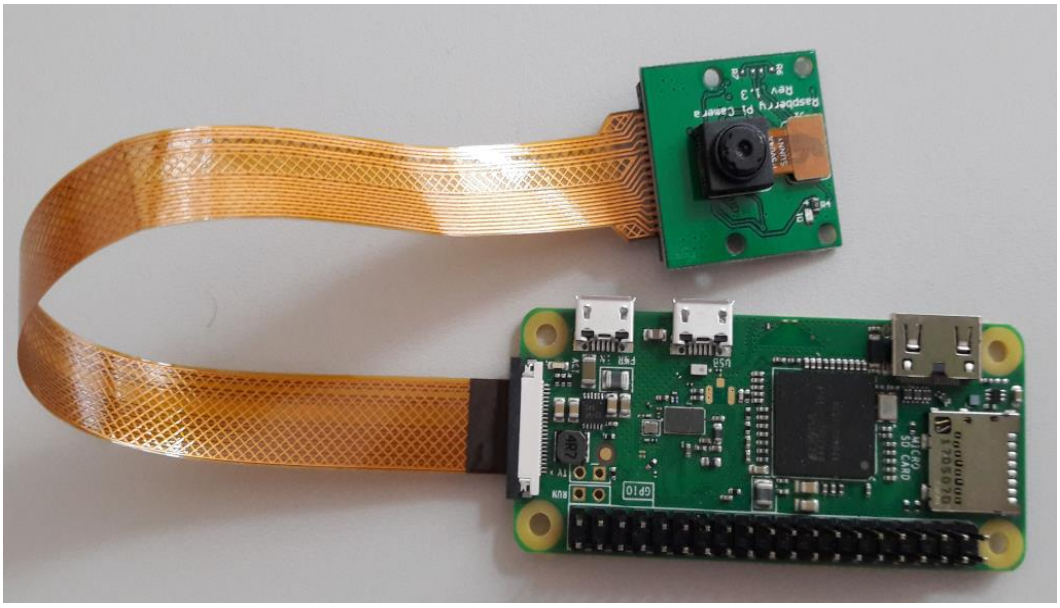


Vir: Lasten

Slika 63: Program OBS studio

Zaradi prisotnosti organizmov v akvarijski posodi je potrebno sondi SBE-3 in SBE-4 dodatno zaščititi. Sondi SBE-3 in SBE-4 imata izpostavljeno tipalo, na katerega se lahko naselijo organizmi, predvsem alge in majhni morski polži. To lahko privede do poškodbe temperaturnega tipala, ki je na sondi SBE-3, oziroma zamašitev kanalčka za pretok vode na sondi SBE-4. Na MBP NIB so mi predlagali uporabo zaščitne mrežice, s katero bi mehansko onemogočili naselitev polžev v območje tipal. Težavo predstavlja tudi obrast tipal, zaradi katere postanejo meritve napačne. Obrast je večja v poletnih mesecih, ko je v avli MBP NIB prisotne več sončne svetlobe. Izboljšavo bi lahko dosegel z uporabo črpalke, s katero bi lahko v območje tipala dovajal kemikalije za čiščenje sonde, vendar je to v akvarijski posodi neizvedljivo.

Možna je tudi povsem drugačna zasnova merilne enote z uporabo mikroračunalnika, kot je na primer Raspberry PI ali Raspberry PI nano. V tem primeru gre za miniaturni osebni računalnik, ki ga poganja operacijski sistem Linux (Slika 64). Datotečni sistem se nahaja na micro SD kartici, ki je velikosti 64 GB ali več. Takšen sistem omogoča poganjanje zahtevnejših aplikacij, ki jih lahko napišemo s poljubnim programskim jezikom (npr. Node.js, C/C++, Python ali Java). Na voljo je zelo veliko prostora za shranjevanje podatkov in za različne baze podatkov. Mikroračunalnik je opremljen z vhodno-izhodnim vodilom, ki podpira tudi protokol SPI, na katerega lahko priključimo zunanjo merilno enoto ali pretvornik. Če bi se odločil za razvoj programa z mikroračunalnikom, bi programsko opremo napisal v trenutno zelo zmogljivem in priljubljenem jeziku Node.js (https://nodejs.org/en/, zadnjič obiskano 13. julija 2018). Mikroračunalnik je poleg vhodno-izhodnega vodila opremljen še z dvema mikro USB priključkoma ter HDMI video izhodom, na katerega lahko priključimo zunanji monitor in ostale periferne enote. Cena takšnega mini računalnika ne presega 20 €.



Vir: Lasten

Slika 64: Raspberry PI nano s kamero za video zajem v HD ločljivosti

8.3 Primerjava merilne enote z drugimi napravami

Med razvojem merilne enote sem s pomočjo spletnega iskalnika Google iskal podobne projekte oz. naprave, ki so namenjene merjenju temperature in ostalih parametrov morske vode. Mojemu projektu se je najbolj približal razvoj termometra za merjenje temperature, ki

ga je v svojem diplomskem delu opisal Kervina Gregor (https://repozitorij.uni-lj.si/IzpisGradiva.php?id=83469&lang=slv&prip=dkum:8713775:d4, zadnjič obiskano 13. julija 2018). V nalogi opisuje različne temperaturne senzorje, ki pa se po delovanju razlikujejo od sonde SBE-3. Senzorji, ki jih opisuje, vsebujejo AD pretvornike in komunicirajo z drugimi komponentami po SPI vodilu. Avtor je v svoji nalogi izbral temperaturni senzor ADT7420 proizvajalca Analog Devices. Senzor ADT7420 se po delovanju razlikuje od sonde SBE-3. Ta ima AD pretvornik s 16-bitno ločljivostjo, medtem ko je sonda SBE-3 analognega tipa. Zaradi tega senzor ADT7420 ne potrebuje pretvornika signala in ga lahko brez posebnih prilagoditvenih vezij priključimo na SPI vodilo mikrokrmilnika. Slabost senzorja ADT7420 je njegova oblika, saj le-ta ni namenjen neposrednemu merjenju temperature vode, zato ga je avtor prilagodil. Senzor je za potrebe preizkusa zatesnil s plastično maso. Tako prilagojen senzor je nato uporabil za izvajanje meritev (Slika 65).

Vir: https://repozitorij.uni-lj.si/IzpisGradiva.php?id=83469&lang=slv&prip=dkum:8713775:d4

Slika 65: Prilagoditev senzorja ADT7420

Zaključim lahko, da je moja merilna enota izdelana z namenom uporabe obstoječih sond SBE-3 in SBE-4. Če bi želel na merilno enoto priključiti druge tipe senzorjev, bi moral spremeniti delovanje programa v mikrokrmilniku tako, da bi ustrezal branju podatkov za določen tip senzorja. Ohranil bi izhodno strukturo podatkov v XML obliki, kjer na zahtevo odjemalca posredujem izmerjeno temperaturo, prevodnost in slanost. Strukturo XML podatkov sem podrobneje opisal v dodatku E »Izvorna koda za mikrokrmilnik in opis delovanja« na strani 114. Tako bi lahko uporabili obstoječo spletno aplikacijo za shranjevanje in prikaz izmerjenih podatkov.

8.4 Stroškovnik

Za razvoj merilne enote sem porabil denarna sredstva, ki so povezana z nabavo elektronskih komponent, kablov, konektorjev, ohišja, kamer in mikrokrmilnika (Tabela 5). Med razvojem merilne enote so mi iz neznanega razloga odpovedale tri kamere, kar nakazuje na slabo kvaliteto cenenih izdelkov. Večino elektronskih komponent sem kupil na spletnih straneh različnih kitajskih prodajalcev. Težava je le čas dostave, saj lahko od naročila preteče več tednov, preden pošiljka prispe v Slovenijo.

Tabela 5: Stroškovnik

Količina	**Artikel**	**Cena**
1	Mikrokrmilnik DcDuino (Arduino UNO)	5,00 €
1	Ethernet ščit W5100	4,76 €
1	PoE adapter za priklop kamere	4,06 €
1	Ploščice za izdelavo prototipih vezij 5 × 7 cm	1,04 €
1	USB vmesnik JTAG razhroščevalnik za AVR procersorje	5,98 €
1	Pretvornik 10 pin na 6 pin za USB JTAG	0,91 €
1	Pretvornik 10 pin na 6 pin za USB JTAG	0,91 €
1	DIN konektorji – ženski	2,27 €
1	DIN konektorji – moški	1,38 €
1	Napajalnik 220 V/12 V	4,14 €
1	Stabilizator napetosti 5 V	1,46 €
1	Napajalnik za kamero	4,29 €
4	Kamera IPC (720P) Motorized Zoom & Auto Focal LEN 1/3"	25,31 €
1	Povezovalni ethernet kabel + napajanje 5 mt	4,15 €
1	RJ45 priključek za ethernet vodotesni	7,28 €
1	Elektronske komponente tiskanega vezja	10,28 €
1	Wire wrap žica za prototipna vezja	4,21 €
1	Wire wrap žica za prototipna vezja	4,21 €
1	Nadometna doza	15,33 €
1	Tiskana vezja 1. prototipa	10,00 €
1	Tiskana vezja 2. prototipa	10,00 €
	SKUPAJ	**202,90 €**

Seznam virov

Amon S. (2013). *Senzorji in aktuatorji.* Ljubljana: Univerza v Ljubljani, Fakulteta za elektrotehniko, 2013.

Banzi, M. 2009. *Getting Started with Arduino, druga izdaja.* ZDA : O'Reilly, 2009.

Basarič N. (1996). *Analogna elektronska vezja.* Ljubljana: Fakulteta za elektrotehniko, 1996.

Bogataj L. (2014). *Planet voda.* Ljubljana: Mladinska knjiga, 2017.

Drnovsek J., Bojkovski J, Pušnik I., Batagelj V., Hudoklin D. (2005). *Merilni sistemi 1 in 2. Študijska skripta.* Ljubljana: Fakulteta za elektrotehniko, Univerza v Ljubljani, 2005.

Ekart, Matej. 2012. *Krmiljenje mikrokrmilniškega modula arduino in njegova uporaba pri komunikaciji z ostalimi napravami.* Maribor : Univerza Maribor, Fakulteta za elektrotehniko in informatiko, 2012.

Hercog D. (2000). *Protokoli in standardi v TK (Telecommunication Protocols and Standards).* Ljubljana: Fakulteta za elektrotehniko, 2000.

Jagrič A. (2013). *Aplikacija za porazdeljeno kodiranje pretočnih video vsebin.* Ljubljana: Fakulteta za računalništvo in informatiko, 2013.

Kervina G. (2016). *Termometer za merjenje temperature vode v oceanografiji.* Ljubljana: Fakulteta za elektrotehniko, 2016.

Kos U. (2016). *Sistem za daljinsko merjenje nivoja vode v zbiralniku.* Ljubljana: Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2016.

Lenhart M. (2013). *Oddaljeno upravljanje in nadzor pametne hiše preko mobilne platforme android in mikrokrmilnika Arduino*. Maribor: Fakulteta za elektrotehniko in računalništvo, 2013.

Malačič V. (2014). *Oceanografija in meteorologija. Študijski program pomorstvo, pomorski sistemi.* Ljubljana: Fakulteta za pomorstvo in promet, 2014.

Mavsar, Gašper. 2017. *Testiranje aplikacijskih strežnikov.* Kranj : Univerza v Mariboru, 2017.

Mesojedec, Fabjan. (2004). *Java2 – temelji programiranja.* Ljubljana : Pasadena, 2004.

Perko A. (2015). *Načrtovanje in izvedba pametne hiše z mikrokrmilnikom Arduino.* Maribor: Fakulteta za elektrotehniko, 2015.

Povhe T. (2017). *Omejitev korozije na plovilih.* chttp://www.povhenavtika.4mg.com/izdelki/clanki/Omejitev%20korozije%20na%20plovilih.pdf : s.n., (3. 3 2017).

Rajnar M. (2014). *Primerjava uporabe SOAP in REST za potrebe povezave mobilnih naprav s spletnimi storitvami.* Ljubljana: Fakulteta za računalništvo in informatiko, 2014.

Repolusk, K. (2010). *Sistem pretvorbe spletnih avdiovizualnih vsebin v DVB transportne tokove v realnem času.* Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 2010.

Richter M. (2005). *Naše morje – Okolja in živi svet Tržaškega zaliva.* s.l.: Znanstveno raziskovalno središče Republike Slovenije, 2005.

Sea-Bird (2017). *http://www.seabird.com/sbe3s-temperature-sensor?qt-product_tabs=1#qt-product_tabs. (*3. 3. 2017).

Strnad J. (1984). *O merjenju temperature in termometrih.* 1984 PRESEK, Letnik 11, št. 1, str. 34–39.

Unesco (1978). *Unesco technical papers in marine science***.** *Ninth report of the joint panel on oceanographic tables and standards.* Paris: Unesco, 1978.

Unesco (1980). *Unesco technical papers in marine science. Tenth report of the joint panel on oceanographic tables and standards.* Canada, Sidney, B.C: Unesco, 1980.

Unesco (1981). *Unesco technical papers in marine science. Background papers and supporting data on the Practical Salinity Scale.* Unesco, 1980.

Vrečar B. (1998). *Računalniško podprto projektiranje tiskanih vezij.* Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, 1998.

Seznam slik

Seznam tabel

Dodatek A- Funkcija za pretvorbo prevodnosti v slanost

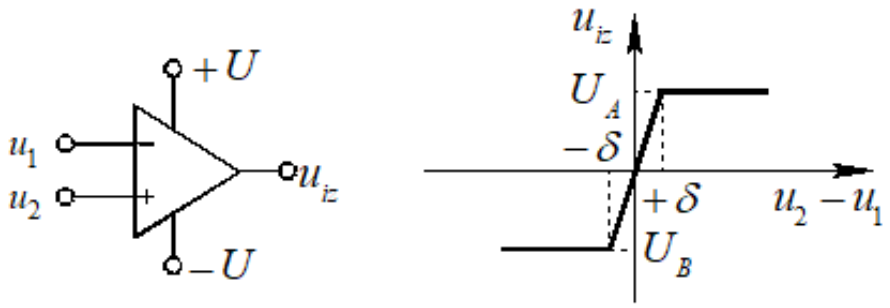
Funkcija calcSlanost prejme tri vhodne atribute: temperaturo *t* v °C, prevodnost *c* v mS/cm ter tlak *p* v mb (mili bar). Izhod funkcije je slanost *S*, ki ne vsebuje enote. Funkcija je prevedena v C/C++ programski jezik iz obstoječe funkcije, dosegljive na naslovu (http://salinometry.com/ctd_calc.js, zadnjič obiskano 17. maja 2018).

```
double calcSlanost(double t, double c, double p)
{
  double a0 = 0.008;
  double a1 = -0.1692;
  double a2 = 25.3851;
  double a3 = 14.0941;
  double a4 = -7.0261;
  double a5 = 2.7081;

  double b0 = 0.0005;
  double b1 = -0.0056;
  double b2 = -0.0066;
  double b3 = -0.0375;
  double b4 = 0.0636;
  double b5 = -0.0144;
  double k = 0.0162;

  double Aa1 = 2.070e-5;
  double Aa2 = -6.370e-10;
  double Aa3 = 3.989e-15;

  double Bb1 = 0.03426;
  double Bb2 = 0.0004464;
  double Bb3 = 0.4215;
  double Bb4 = -0.003107;

 //konstante za izračun rt
  double c0 = 0.6766097;
  double c1 = 0.0200564;
  double c2 = 0.0001104259;
  double c3 = -0.00000069698;
  double c4 = 0.0000000010031;

  double temp = 1.00024*t;

  //razmerje prevodnosti
  double R = c / 42.914;
  // izračun Rt
  double rt = c0+c1*temp+c2*temp*temp+c3*temp*temp*temp+c4*temp*temp*temp*temp;
  double alpha = (Aa1*p+Aa2*p*p+Aa3*p*p*p)/(1+Bb1*temp+Bb2*temp*temp+Bb3*R+Bb4*temp*R);
  double Rt = R/(rt*(1+alpha));
  // izračun slanosti
  double S = a0+a1*pow(Rt,0.5)+a2*Rt+a3*pow(Rt,1.5)+a4*Rt*Rt+a5*pow(Rt,2.5)+ (temp-
15)*(b0+b1*pow(Rt,0.5)+b2*Rt+b3*pow(Rt,1.5)+b4*Rt*Rt+b5*pow(Rt,2.5))/(1+k*(temp-15));
  return S;
}
```

Dodatek B- Komparator, delovanje in uporaba

B.1 Komparator

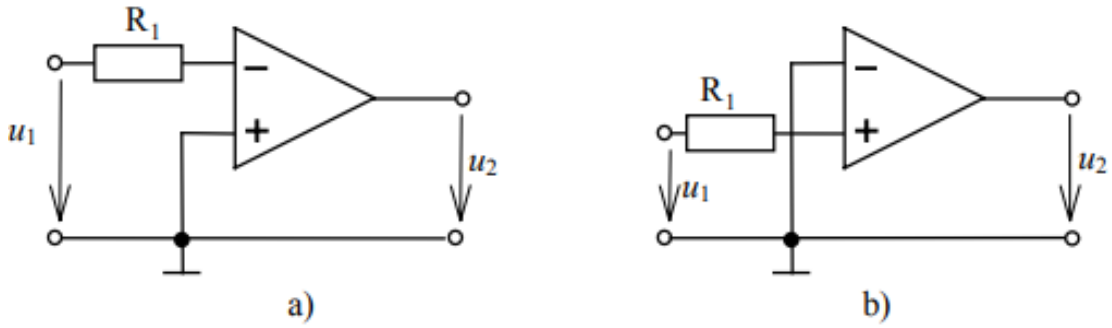
Komparatorji so pomemben člen med analognimi in digitalnimi signali. So nelinearna elektronska vezja, s katerimi med seboj primerjamo dve napetosti. Komparator ima dva vhoda in ga je možno sočasno krmiliti z dvema signaloma (Slika 66). Običajno je na enem izmed vhodov konstantna referenčna napetost, s katero primerjamo drugi signal. (http://fides.fe.uni-lj.si/~arpadb/aev/knjiga-aev.pdf, zadnjič obiskano 12. februarja 2018).



Vir: fides.fe.uni-lj.si/~basaricn/RezultatiAEV/Poglav6C.doc

Slika 66: Shema komparatorja in prenosna funkcija

Razlikujemo med komparatorji s povratno zanko in brez nje. Pri slednjih (Slika 67) je vhodna napetost priključena na enega izmed vhodov, ki jo ojača z odprtozančnim ojačenjem A_d.

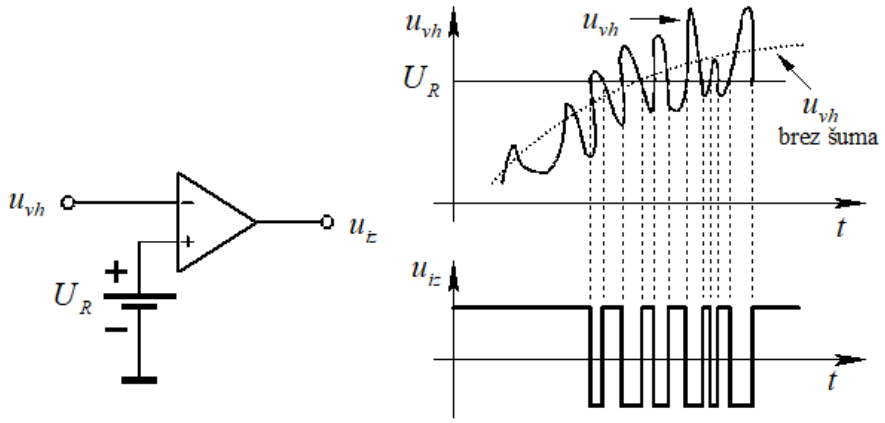


Vir: http://lrtme.fe.uni-lj.si/lrtme/slo/UNIVSS/indu_elek/oper_ojac3del_2007.pdf

Slika 67: Komparator napetosti z referenčno napetostjo enako 0 V

Zaradi velikega ojačenja A_d zadošča že majhna razlika primerjanih napetosti, da bo izhodna napetost komparatorja dosegla eno izmed veljavnih vrednosti. Tako je na primer pri ojačenju A_d = 30000 in napajalni napetosti U_b = 12V potrebna vhodna diferenčna napetost zgolj v velikosti U_d = 0,4 mV, da pride do spremembe izhodne napetosti u_2.

Prikazani primer komparatorja ni primeren za večino praktičnih primerov, saj v praksi pogosto želimo, da ima komparator prenosno karakteristiko s histerezo. Zaradi zelo velikega ojačenja, ki ga imajo komparatorji, lahko v področju preklopa pride do težav, če imamo na vhodu komparatorja signal zelo nizke frekvence, ki vsebuje določen šum (motnjo). V tem primeru lahko pride do dodatnih neželenih preklopov zaradi šuma (motenj), kar prikazuje Slika 68.



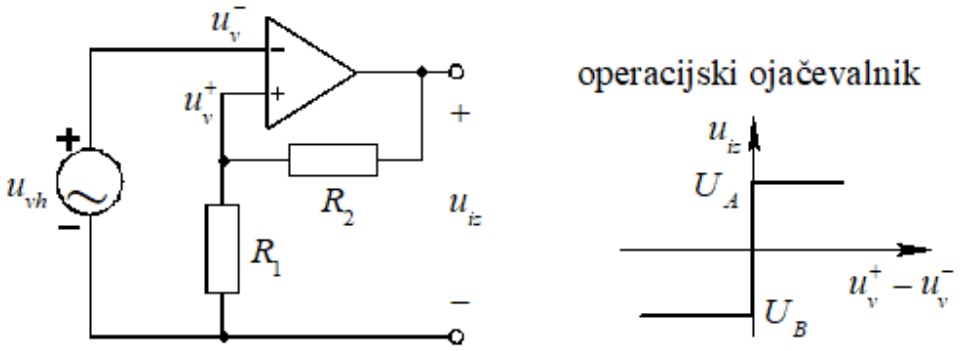
Vir: http://lrtme.fe.uni-lj.si/lrtme/slo/UNIVSS/indu_elek/oper_ojac3del_2007.pdf

Slika 68: Vpliv motilnega šuma na izhodni signal

Tem nevšečnostim se izognemo z uporabo bistabilnega komparatorja s histerezo (angl. Scmitt trigger), katerega sem tudi uporabil pri načrtovanju vezja (http://fides.fe.uni-lj.si/~arpadb/aev/knjiga-aev.pdf, http://fides.fe.uni-lj.si/~basaricn/RezultatiAEV/Poglav6C.doc, zadnjič obiskano 17. februarja 2018).

B.2 Bistabilni komparator

Bistabilni komparator s histerezo uporablja pozitivno povratno vezavo za dosego dveh stabilnih stanj. Invertirajoči bistabilni komparator prikazuje Slika 69.



Vir: http://lrtme.fe.uni-lj.si/lrtme/slo/UNIVSS/indu_elek/oper_ojac3del_2007.pdf

Slika 69: Invertirajoči bistabilni komparator

Vhodni tok operacijskega ojačevalnika je določen kot:

$$I=\frac{U_{iz}}{R_1+R_2}=\frac{Uv^+}{R_1} \tag{10}$$

Če zanemarimo vhodni tok operacijskega ojačevalnika, je:

$$u_v{}^+=\frac{R_1}{R_1+R_2}U_{iz} \tag{11}$$

Napetost $u_v{}^+$ ni konstanta, temveč je odvisna od izhodne napetosti U_{iz}. Za določitev napetostne prenosne karakteristike predpostavimo, da je izhodna napetost operacijskega ojačevalnika $U_{iz} = U_A$. V tem primeru je:

$$u_v{}^+=\frac{R_1}{R_1+R_2}U_{A.} \tag{12}$$

Dokler je $u_{vh} < u_v{}^+$, je U_{iz} enaka konstanti $U_{A.}$Torej je pri višanju u_{vh} mejna vhodna napetost, ko preskoči izhodna napetost na vrednost $u_{iz} = U_{B,}$ enaka:

$$U_{TA}=\frac{R_1}{R_1+R_2}U_{A.} \tag{13}$$

Ko pa je $u_{vh} > U_{TA}$, je $u_v^- > u_v^+$ in je izhod v nizkem stanje $u_{iz} = U_B$, je v tem primeru:

$$u_v^+ = \frac{R_1}{R_1+R_2} U_B \tag{14}$$

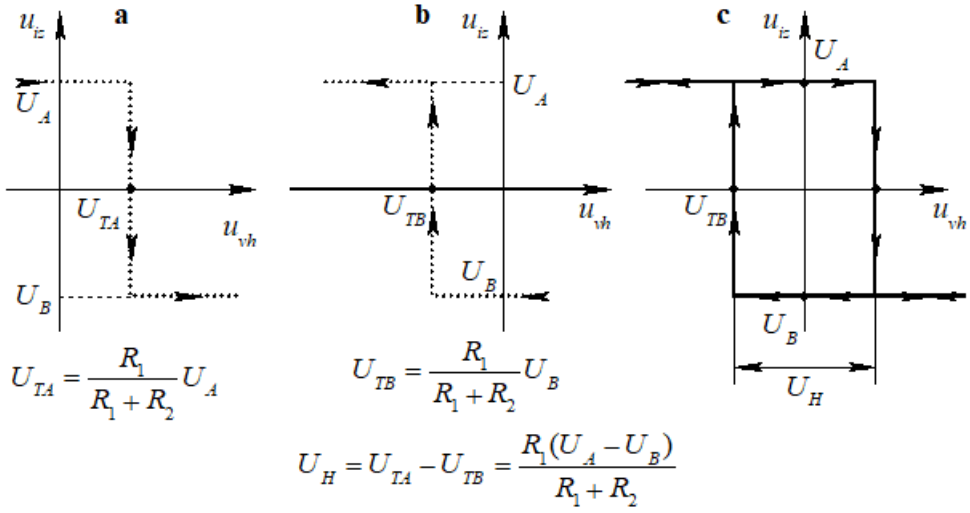
Ker je $U_B < U_A$, je vhodna napetost u_{vh} še vedno večja od u_v^+ in izhod ostane v nizkem stanju, četudi u_{vh} narašča. Predpostavimo, da vhodna napetost upada. Dokler je:

$$u_{vh} > \frac{R_1}{R_1+R_2} U_B \,, \tag{15}$$

je izhod komparatorja v nizkem stanju ($u_{iz} = U_B$). Mejna (pragovna) vhodna napetost, ko pride do preklopa, je:

$$U_{TB} = \frac{R_1}{R_1+R_2} U_B. \tag{16}$$

Razmere prenosnih karakteristik prikazuje (Slika 70) (http://fides.fe.uni-lj.si/~basaricn/RezultatiAEV/Poglav6C.doc, str. 209–210).

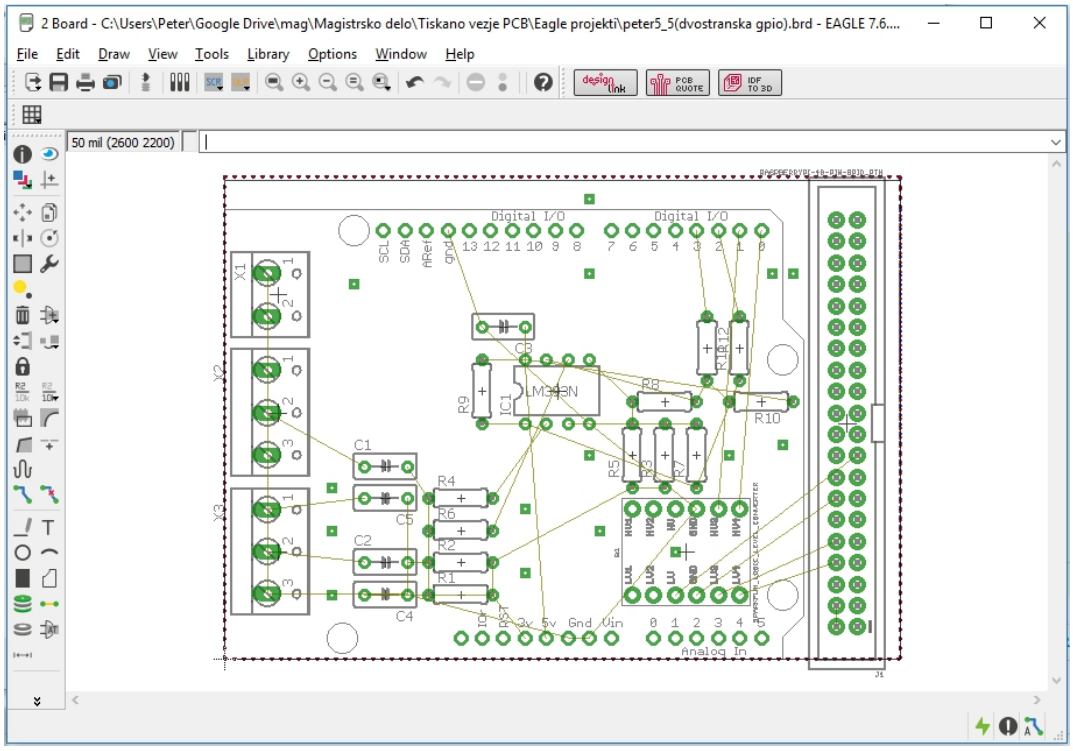


Vir: http://fides.fe.uni-lj.si/~basaricn/RezultatiAEV/Poglav6C.doc

Slika 70: a) napetost u_{vh} narašča, b) napetost u_{vh} upada, c) združena prenosna karakteristika s prikazano histerezo

B.3 Načrtovanje tiskanega vezja pulznega pretvornika

Po opravljenem preverjanju povezav s pomočjo programa Eagle lahko pričnemo z izrisom tiskanega vezja v modulu »Layout«. Uporabniku se bodo na zaslonu izrisale naključno razporejene komponente. Te bodo medsebojno povezane z ravnimi črtami tako, kot jih narekuje vezalna shema. Da bi uspešno izrisali tiskano vezje, moramo elemente ustrezno prerazporediti. To storimo tako, da izbrani element premaknemo na želeno lokacijo na tiskanem vezju. Slika 71 prikazuje izgled tiskanega vezja v programu Eagle layout. Prikazane ravne črte, ki so med elementi, so vezi med elektronskimi komponentami. Na začetku načrtovanja so to ravne linije med priključki elementov.

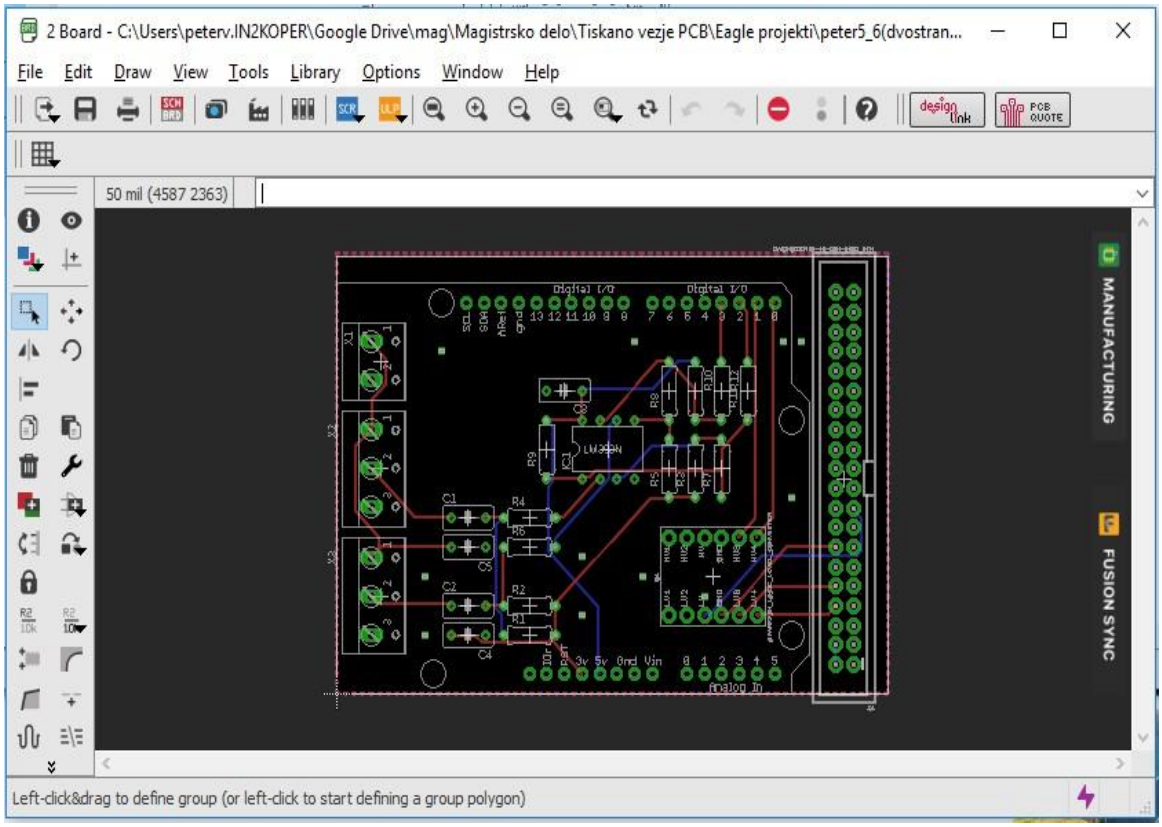


Vir: Lasten

Slika 71: Izris tiskanega vezja

Pri izdelavi vezi med elementi lahko izberemo dve opciji, in sicer: 1.) Vezi med komponentami lahko rišemo ročno, 2.) Uporabimo opcijo avtomatskega povezovanja komponent (angl. Autoroute). Pri avtomatskem povezovanju komponent tudi določimo, koliko slojno (angl. Layer) tiskanino želimo. Pri razvoju sem se odločil za dvoslojno tiskanino, saj je izdelava cenovno ugodna, hkrati pa je povezovanje elementov lažje zaradi uporabe medplastnih povezav (angl. Vias).

Slika 72 prikazuje izdelano tiskano vezje z uporabo avtomatskega povezovanja (angl. Autoroute). Z belo barvo so označene komponente in njihov položaj na tiskanem vezju. Te bodo na tiskano vezje dodane kot sitotisk. Luknje (vstavimo elektronske komponente in medplastne povezave (angl. Vias)) so označene z zeleno barvo. Povezave z rdečo barvo predstavljajo bakrene vezi, ki povezujejo kontakte elementov na zgornji strani tiskanega vezja (angl. Top side), medtem ko povezave modre barve prikazujejo bakrene vezi na spodnji strani tiskanega vezja (angl. Solder side).

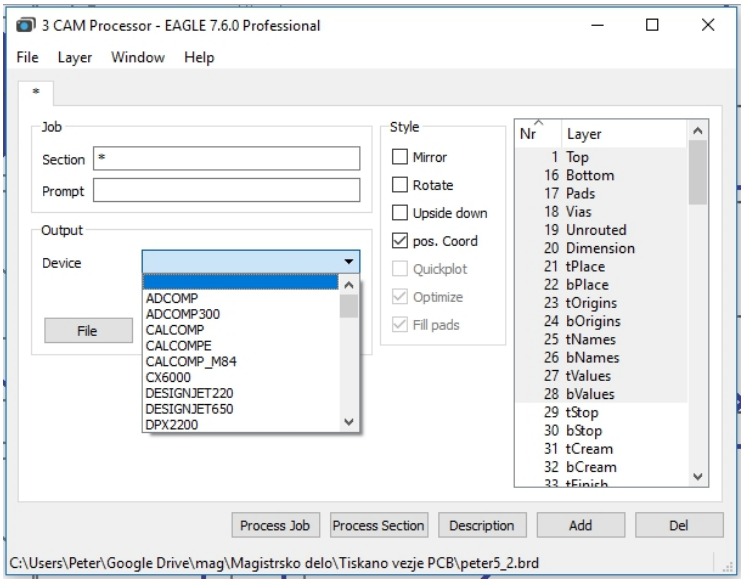


Vir: Lasten

Slika 72: Izdelano tiskano vezje v programu Eagle Layout

B.4 Izdelava gerber datotek

Če želimo izdelati tiskano vezje v fizični obliki, potrebujemo nekaj datotek, od katerih vsaka zase natančno opisuje dimenzije in položaj vezi, otočkov in lukenj na posamezni plasti (angl. Layer) ter zgornjo in spodnjo obliko ploščice tiskanega vezja (Slika 73).



Vir: Lasten

Slika 73: Izdelava gerber datoteke z programom »Eagle«

Takšna datoteka se imenuje gerber datoteka. Običajno narišemo vezje, ki ima vezi vsaj na strani za spajkanje (angl. Solder side). V takšnem primeru moramo izdelati gerber datoteko te plasti, v kateri so zapisane dimenzije in položaj vezi, otočkov in centrov lukenj. Če imamo dvostransko vezje, kot je v mojem primeru vezje pretvornika, moramo izdelati več datotek. Izdelovalci prototipnih vezij morajo seveda vedeti, ali želimo narisano vezje imeti na strani za spajkanje (angl. Solder side) ali na strani za montažo elementov (angl. Component side), saj se stran za spajkanje rezka ali jedka obrnjeno, kot je videti narisana, zato se moramo držati dogovorjenih pravil. Pri bolj razgibanih zunanjih oblikah ploščic potrebujemo še datoteko z narisanimi zunanjimi mejami ploščice in z označenimi izrezi. Za notranje izreze generiramo pot rezkarja po notranji strani narisane neprekinjene linije, za zunanjo obliko pa po zunanji strani te linije. V odvisnosti od tega, kakšnih dimenzij so notranji izrezi, pred generiranjem poti izberemo še ustrezen premer rezkarja, da te oblike lahko proizvajalec tiskanih vezij tudi fizično izdela. Gerber datoteka lege vseh komponent v plasteh opiše v inčih, formata 4.4 (kar pomeni, da bo en inč predstavljen s štirimi decimalnimi mesti, torej bo najmanjši delček

1/10.000 inča) in formata RS-274X. RS-274X je oblika zapisa gerber datoteke. Specifikacijo datoteke si lahko prenesemo s spletnega naslova (http://www.ucamco.com/public/RS-274X_Extended_Gerber_Format_Specification _201201.pdf, zadnjič obiskano 7. marca 2018). Sledi še izvoz NC Drill datoteke, to pa je spisek lukenj, njihovi premeri in položaji na tiskanini. S pomočjo te datoteke generiramo zaporedje menjave orodij (svedrov, rezkarjev), ki se bodo uporabljala pri izdelavi tiskanine. Izhodni format vrtalne datoteke mora biti v inčih, izhodni zapis v ASCII obliki brez kodiranja. Slika 74 prikazuje izdelane gerber datoteke za izdelavo tiskanine pretvornika (http://erid.tsckr.si/12/eagle_navodila/vaja9.html, zadnjič obiskano 9. marca 2018, Mele J., 2007).

Ime	Datum spremembe	Vrsta	Velikost
peter5_2.brd	9. 02. 2017 23:04	Datoteka BRD	89 KB
peter5_2.dri	9. 02. 2017 22:49	Datoteka DRI	1 KB
peter5_2.GBL	9. 02. 2017 22:49	Datoteka GBL	5 KB
peter5_2.GBO	9. 02. 2017 22:49	Datoteka GBO	1 KB
peter5_2.GBP	9. 02. 2017 22:49	Datoteka GBP	1 KB
peter5_2.GBS	9. 02. 2017 22:49	Datoteka GBS	3 KB
peter5_2.GML	9. 02. 2017 22:49	Datoteka GML	1 KB
peter5_2.gpi	9. 02. 2017 22:49	Datoteka GPI	2 KB
peter5_2.GTL	9. 02. 2017 22:49	Datoteka GTL	5 KB
peter5_2.GTO	9. 02. 2017 22:49	Datoteka GTO	91 KB
peter5_2.GTP	9. 02. 2017 22:49	Datoteka GTP	1 KB
peter5_2.GTS	9. 02. 2017 22:49	Datoteka GTS	3 KB
peter5_2.pro	9. 02. 2017 22:44	Datoteka PRO	2 KB
peter5_2	9. 02. 2017 22:49	Besedilni dokument	2 KB

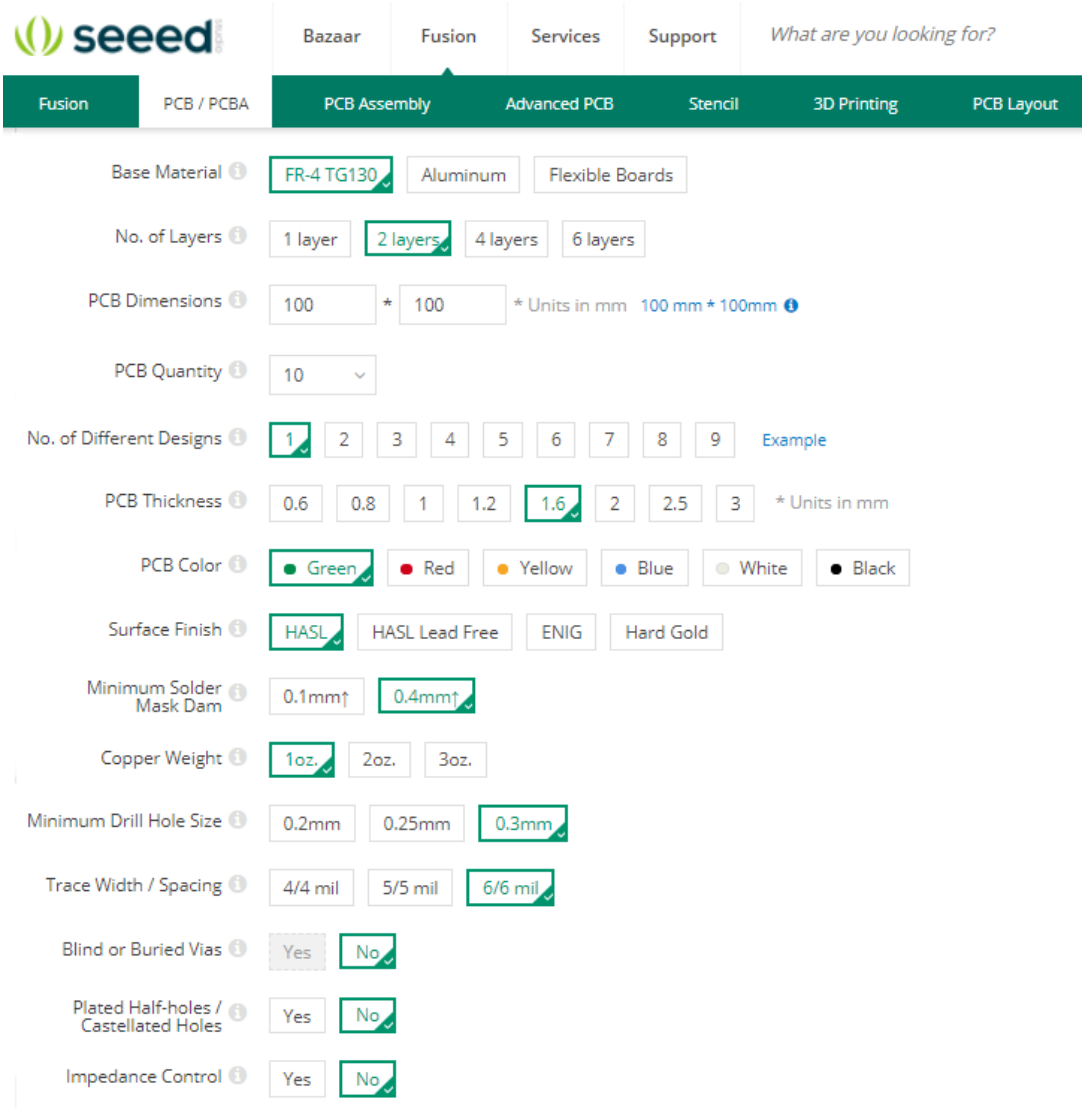
Vir: Lasten

Slika 74: Izhodne gerber datoteke

B.5 Izdelava tiskanega vezja

Za izdelavo tiskanega vezja sem izbral kitajsko podjetje Seeedstudio, ki omogoča izdelavo tiskanih vezij po zelo ugodnih cenah. Naročilo izdelave tiskanega vezja je potekalo na spletni strani (https://www.seeedstudio.com/fusion_pcb.html, zadnjič obiskano 8. marca 2018). Postopek naročanja je preprost. Gerber datoteke, katere sem izdelal s pomočjo programa Eagle, sem moral stisniti s pomočjo programa zip. Nato sem zip datoteko s spletno aplikacijo naložil na strežnik podjetja Seeedstudio. Aplikacija nato preveri pravilnost gerber datotek in uporabniku ponudi dodaten nabor možnosti, ki vplivajo na izgled, kvaliteto in ceno izdelave tiskanega vezja. Končna cena izdelave tiskanega vezja je odvisna tudi od izbranih parametrov,

ki jih spletna stran ponuja. Parametri, s katerimi vplivamo na končno ceno izdelave, so: material, iz katerega bo tiskano vezje izdelano, število plasti tiskanine, dimenzija, količina izdelanih tiskanih vezij, debelina tiskanega vezja, barva (lak), obdelava kontaktov, minimalna razdalja med vezmi tiskanega vezja, vidne ali nevidne medplastne povezave ter impedančna kontrola tiskanega vezja. Možnosti izbire prikazuje Slika 75.

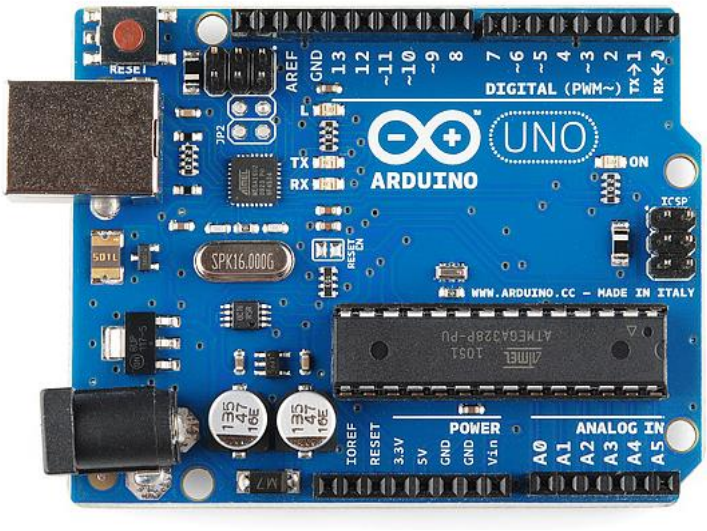


Vir: Lasten

Slika 75: Parametri, ki vplivajo na kvaliteto, izgled in ceno izdelave tiskanine

Dodatek C- Mikrokrmilnik Arduino

Arduino je odprtokodna platforma, ki je zasnovana na enostavni uporabi in programiranju. Namenjena je širši množici uporabnikov, ki s pomočjo enostavnih programskih knjižnic in modulov ustvarjajo različne elektronske izdelke. Predvsem je namenjena ustvarjanju interaktivnih naprav, kot so krmilniki, regulatorji, elektronske ure, preklopniki, alarmne naprave itd. Nastala je kot pomoč neukim s področja elektrotehnike in računalništva oziroma programiranja, da ustvarjajo svoje izdelke s preprostimi programskimi knjižnicami in elektronskimi komponentami. Slika 76 prikazuje najbolj razširjen model družine Arduin, model Arduino Uno (Banzi M., 2009).



Vir: http https://learn.sparkfun.com/tutorials/what-is-an-arduino

Slika 76: Mikrokrmilnik Arduino Uno

Naprave, ki temeljijo na Arduino platformi, so lahko samostojne enote, ki lahko komunicirajo tudi z drugimi napravami in programskimi jeziki. Za programiranje mikrokrmilnika Arduino se uporablja programski jezik, ki izhaja iz implementacije jezika Wiring. V osnovi gre za C/C++ jezik s pripadajočimi knjižnicami, s pomočjo katerih lahko upravljamo proces v mikrokrmilniku (https://www.arduino.cc/en/Main/ OldSoftwareReleases, zadnjič obiskano 18. septembra 2017).

C.1 Izbira ustrezne izvedbe mikrokrmilnika

Celotna strojna in programska oprema je pod odprtokodno licenco, zato na tržišču najdemo veliko različnih izvedb. Skupnost Arduino ponuja več strojnih izvedb, ki se razlikujejo po uporabljenem mikrokrmilniku Atmel, številu serijskih vrat, vhodov A/D, izhodov PWM (angl. Pulse width modulation), pomnilniku in še nekaterih drugih karakteristikah. Za razvoj merilne naprave sem izbral model Arduino UNO. Arduino UNO ustreza vsem kriterijem, ki sem jih potreboval za merjenje frekvence, preračun temperature in slanosti ter komunikacijo s PC računalnikom. Tabela 6 prikazuje nekatere od ključnih tehničnih podatkov različnih izvedb mikrokrmilnika Arduino. Predvsem se modeli razlikujejo po hitrosti delovanja (taktu), ki ga merimo v mega hertzih (MHz), številu vhodno-izhodnih vrat ter spominu, med katere spadajo eeprom, sram in flash.

Tabela 6: Primerjava različnih modelov – izvedb

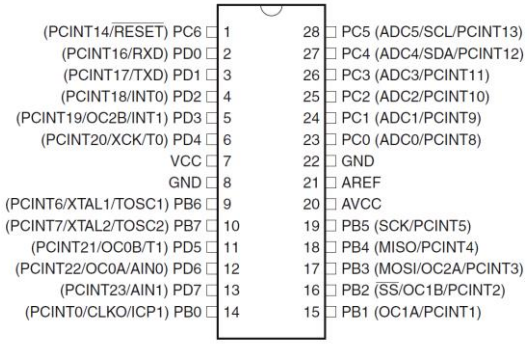
Model	CPE	Frek.	Analogni V/I	Digitalni VI/PWM	EEPROM [KB]	SRAM [KB]	FLASH [KB]
Uno	ATMega328	16 MHz	6/0	14/6	1	2	32
Mega	ATMega2560	16 MHz	16/0	54/15	4	8	256
Gemma	ATinny85	8 MHz	1/0	3/2	0,5	0,5	8
Nano	ATMega328P	16 MHz	8/0	14/6	1	2	32
LillyPad	ATMega328	8 MHz	4/0	9/4	1	2	32

C.2 Mikrokrmilnik Arduino UNO

Model Arduino UNO uporablja mikrokontroler Atmel, model Atmega328 (Slika 77). To je 8-bitni AVR RISC mikrokrmilnik (https://en.wikipedia.org/wiki/Atmel_AVR, zadnjič obiskano 11. marca 2018), ki deluje s taktom 16 MHz. Mikrokrmilnik ima 6 analognih vhodov in 14 digitalnih vhodov/izhodov, od teh jih lahko 6 uporabimo kot PWM izhode. PWM (angl. Pulse Width Modulation) oziroma pulzno-širinska modulacija je modulacijska tehnika, ki se uporablja pretežno za krmiljenje moči na električnih napravah. Analogni vhodi imajo 10-bitno ločljivost, digitalni izhodi PWM pa 8-bitne izhode. Mikrokrmilnik ima tri časovnike ter možnost uporabe internih in zunanjih prekinitev (angl. Interrupts). Dodatno ima mikrokrmilnik tudi programabilen serijski vmesnik UART, serijski vmesnik SPI,

programabilen časovnik Watchdog z ločenim urnim signalom in možnost komunikacije po protokolu I^2C.

Družina mikrokmilnikov AVR je izredno priljubljena zaradi programiranja v programskem jeziku C/C++, kar precej olajša programiranje v primerjavi s programiranjem v zbirniku (angl. Assembler).



Vir: http://electropark.pl/atmega/3065-atmega328p-pu-dip28.html

Slika 77: Priključne sponke mikrokontrolerja ATmega328p (DIP izvedba)

Čip ATmega328P ima že vnaprej naložen zagonski nalagalnik v angl. (Boot loader). Zagonski nalagalnik je program, ki steče ob vsakem vklopu mikrokrmilnika in naloži po potrebi nov program v programski pomnilnik. Tehnične lastnosti mikrokrmilnika Arduino Uno so prikazane v Tabeli 7 (https://store.arduino.cc/arduino-uno-rev3, zadnjič obiskano 11. oktobra 2017 in Kos U., 2016).

Tabela 7: Tehnične lastnosti mikrokrmilnika Arduino UNO

Mikrokrmilnik	ATmega 328
Napajanje	Preko USB priključka ali zunanje vir (5-12) V
Operativna napetost	5 V
Priporočljiva vhodna napetost	7–12 V
Število digitalnih vhodov in izhodov	14
Število analognih vhodod	6
Tok v vhodno izhodnih priključih	40 mA
Tok v 3.3 V priključku	50 mA
Flash spomin	32 KB
Velikost SRAM	2 KB

Velikost EEPROM	1 KB
Hitrost ure (frekvenca)	16 MHz
Dimenzije	68,6 mm × 53,3 mm
Teža	25 g

C.3 Pomnilniški prostor mikrokrmilnika

Pri pisanju programa za mikrokrmilnik moramo biti pozorni na porabo pomnilnika, saj lahko pri prekoračitvi zmogljivosti le-tega pride do napak pri izvajanju programa. Na porabo pomnilnika vpliva celoten program, ki ga sestavljajo uporabljene programske knjižnice in naš program. Poleg tega moramo upoštevati, da delček pomnilnika (približno 0,5 KB) zavzema sistemski nalagalnik (https://store.arduino.cc/arduino-uno-rev3, zadnjič obiskano 11. oktobra 2017 in Banzi M., 2009) .

Velikost spomina je odvisna od vrste mikrokontrolerja, ki ga mikrokrmilnik uporablja. Arduino ima tri vrste pomnilnikov, in sicer:

- **Flash** – 32 KB, spomin, v katerega je naložen program. Gre za trajni zapis tudi v primeru, če pride do izpada napetosti.
- **SRAM (statični RAM)** – 2 KB, spomin, v katerega program shranjuje spremenljivke med izvajanjem. Pomembno je, da v programski kodi tega spomina ne presežemo. Omejen je na 2048 bajtov. Pri izpadu napetosti se podatki v pomnilniku ne shranijo (se izbrišejo).
- **EEPROM** – 1 KB, spomin, v katerega se shranjujejo podatki za dlje časa, tudi če pride do izpada napetosti (https://www.arduino.cc/en/Tutorial/Memory, zadnjič obiskano 7. januarja 2018).

Podatki, shranjeni v flash in EEPROM pomnilniku, so trajni, saj ostanejo tudi, ko zmanjka napajanje. SRAM pomnilnik pa se ob izpadu električne energije izbriše. Zavedati se moramo, da je velikost SRAM pomnilnika omejena, saj ga lahko ob manjši nepazljivosti hitro presežemo. Mikrokontroler ATmega328 ga ima le 2048 bajtov (arduino.cc/en/Main/ ArduinoBoardUno, zadnjič obiskano 7. januarja 2018).

Kot primer si lahko ogledamo izvorno kodo, ki v pomnilniku SRAM zasede 100 bajtov (vsak znak v spremenljivki »podatek« zasede 1 bajt).

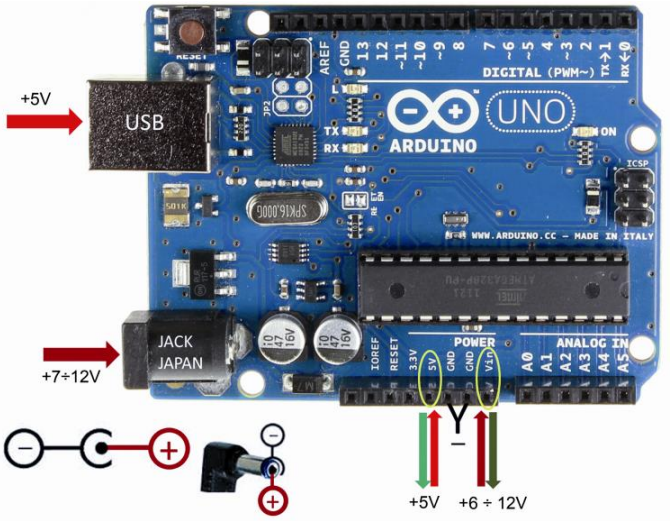
```
char podatek[] = "Primer spremenljivke, katera zasede 100 bytov pomnilnika. Dolga je natanko 100 znakov..............";
```

Če nam zmanjka pomnilnika SRAM, lahko izvajanje programa spodleti na različne načine. Program se bo uspešno naložil na mikrokrmilnik, ampak se morda ne bo zagnal ali pa se bo začel čudno obnašati. Temu se lahko izognemo z različnimi pristopi (arduino.cc/en/Main/ArduinoBoardUno, zadnjič obiskano 7. januarja 2018):

- Pri delu s tabelami uporabljamo podatkovne tipe, ki zasedejo čim manj prostora. Namesto tipa »int«, ki zasede 2 bajta, lahko večkrat uporabimo tip »bajt« (shrani lahko manjše vrednosti), ki zasede 1 bajt.
- Če določenih podatkov ali tekstov ni potrebno spreminjati, jih lahko shranimo na flash pomnilnik, namesto v SRAM.
- Zahtevne preračune izvajamo v napravi, ki je povezana z Arduinom. V svojem primeru bi lahko temperaturo in slanost preračunal v računalniku, ki je povezan z mikrokrmilnikom.

C.4 Napajanje mikrokrmilnika

Arduino lahko napajamo z zunanjim virom ali z USB priključkom, ki je primarno namenjen programiranju in komunikaciji s PC računalnikom. V primeru, da mikrokrmilnik napajamo z zunanjim virom, se le-ta upošteva kot primaren vir. USB priklop ima za napajanje najnižjo prioriteto in se uporablja, ko ni druge možnosti. Slika 78 prikazuje različne možnosti priklopa napajanja mikrokrmilnika Arduino Uno.



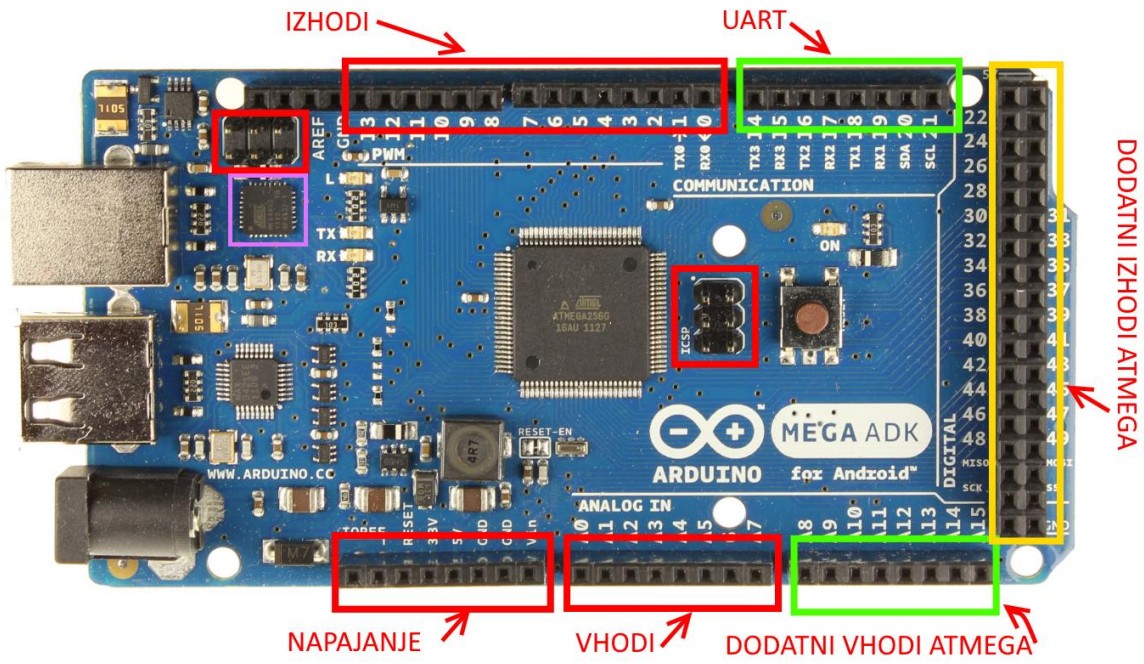
Vir: https://www.open-electronics.org/the-power-of-arduino-this-unknown/

Slika 78: Napajalni priključki mikrokrmilnika Arduino Uno

Glede na tehnične specifikacije lahko mikrokrmilnik deluje pri zunanji napetosti od 6 V do 20 V. Če je napetost manjša od 7 V, lahko 5 V priključek (pin) na ploščici dostavlja manjšo napetost od 5 V, kar lahko povzroči nestabilnost v delovanju. Poleg tega lahko nižja napetost onemogoči delovanje drugih povezanih naprav (ščitov). Pri napajanju več kot 12 V pa se lahko regulator napetosti pregreva in poškoduje mikrokrmilnik. Priporočljiva napetost je tako med 7 in 12 V. Arduino modul vsebuje na USB priključku tudi varovalko, ki varuje pred električnimi sunki ali preveliko napetostjo. Čeprav ima večina računalnikov lastno varovanje, je to dodatna zaščita. V primeru, da skozi USB priključek teče električni tok večji od 500 mA, bo varovalka avtomatsko prekinila povezavo, dokler le-ta ni varna (Elkart M. 2012, zadnjič obiskano 7. januarja 2018).

C.5 Priključne sponke mikrokrmilnika

Mikrokmilnik Arduino vsebuje veliko različnih priključkov. Razpored priključnih sponk (pinov) prikazuje Slika 79. Z rdečo barvo so označeni vhodno-izhodni priključki, ki jih premorejo vsi modeli mikrokrmilnika Arduino, medtem ko so z zeleno in rumeno barvo označeni dodatni vhodno-izhodni priključki, ki jih premore le močnejša različica Arduina, model Atmega2560.



Vir: https://makezine.com/2011/12/01

Slika 79: Priključne sponke mikrokrmilnika Arduino ATMEGA2560

C.6 Digitalni vhodi in izhodi

Priključke, ki so označeni s številkami od 0 do 13, lahko določimo kot digitalne vhode ali izhode. To storimo s pomočjo funkcije v programski kodi. Priključki delujejo pri napetosti 5 V in lahko zavzemajo visoko ali nizko stanje. V programu je visoko stanje določeno s konstanto HIGH, nizko pa s konstanto LOW. Vrednost HIGH ima digitalni vhod takrat, ko napetost preseže 2,4 V, vrednost LOW pa pri napetosti, manjši od 0,8 V (Ekart M., 2012 in https://store.arduino.cc/arduino-uno-rev3, zadnjič obiskano 9. januarja 2018).

Določeni priključki imajo poleg vhodno/izhodnih funkcij tudi druge funkcionalnosti:

- Priključka št. 0 in 1 sta priključka za sprejemanje serijskih podatkov. Na mikrokrmilniku Arduino sta povezana z dodatnim mikrokontrolerjem (ATmega8U2), ki se nahaja na tiskanini (Slika 79, označeno z vijolično barvo). Ta skrbi za USB povezavo med PC računalnikom in mikrokontrolerjem.
- Priključka št. 2 in 3 omogočata zaznavo zunanje prekinitve. Ta dva priključka sem uporabil za zaznavo impulzov iz sond SBE-3 in SBE-4. Uporaba prekinitev je ključnega pomena pri hitrih spremembah.
- Priključki št. 3, 5, 6, 9, 10 in 11 omogočajo poleg digitalnega vhoda in izhoda tudi pulzno širinsko modulacijo (PWM), s katero lahko emuliramo analogni izhod. Več o emulaciji PWM signala si lahko preberete na povezavi (https://www.arduino.cc/en/Tutorial/PWM, zadnjič obiskano 15. junija 2018).
- Priključki z oznakami SS (pin 10), MOSI (pin 11), MISO (pin 12), SCK (pin 13) omogočajo komunikacijo s serijskim sinhronskim vmesnikom (SPI). Gre za standard za sinhrono podatkovno povezavo elektronskih naprav. Protokol za komuniciranje uporablja princip nadrejen–podrejen (angl. master/slave), po katerem nadrejena naprava vzpostavlja stik in vodi komunikacijo s podrejeno napravo. Dandanes je na tržišču vedno več tipal in sond, ki podpirajo komunikacijo po protokolu SPI. Slika 80 prikazuje shematski prikaz povezave dveh enot z vodilom SPI (https://sl.wikipedia.org/wiki/SPI, zadnjič obiskano 9. januarja 2018).

Vir: https://makezine.com/2011/12/01

Slika 80: Nadrejena in podrejena naprava na SPI vodilu

- Priključek 13 je poseben, ker je nanj povezana kontrolna LED dioda. V primeru, da na priključek pošljemo visoko stanje, se bo LED dioda prižgala. S tem priključkom lahko nadziramo izvajanje programa v mikrokrmilniku. Na priključek lahko izmenično pošiljamo visoki ali nizki signal s časovno zakasnitvijo in s tem dobimo utripajočo LED diodo. V primeru, da bi se program zaradi nepravilnega delovanja ustavil, bi LED dioda ostala ugasnjena ali vključena, odvisno od zadnjega stanja signala (Ekart M., 2012 in http://johnny-five.io/examples/led/, zadnjič obiskano 10. januarja 2018).

C.7 Analogni vhodi

Arduino UNO omogoča uporabo šestih analognih vhodov. Analogni vhodi so označeni z oznako od A0 do A5. Na analogne priključke lahko pripeljemo napetost med 0 in 5 V. Mikrokontroler Arduino Uno ima vgrajen 6-kanalni A/D pretvornik. A/D pretvorba je postopek, s katerim analogno veličino (električno napetost) pretvorimo v ustrezno digitalno vrednost (številko). Arduinu UNO vsebuje A/D pretvornik, ki deluje z 10-bitno ločljivostjo, kar pomeni, da bo analogno vrednost razdelil na 1024 števil.

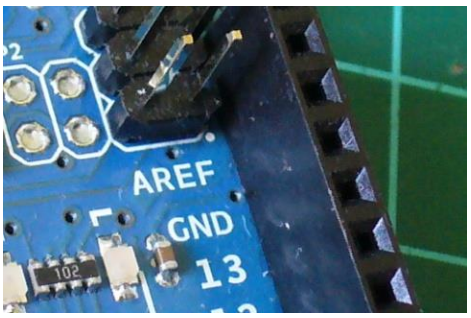
Primer:

Če na analogni vhod priključimo električno napetost 5 V, bo A/D pretvornik to pretvoril v 2^{10}= 1024 števil, oziroma 2^{10}-1 = 1024 – 1 = 1023 korakov. Če torej 5 V razdelimo z maksimalnim številom korakov, dobimo: $\frac{5}{2^{10}-1} = 4{,}8876$ V = 4,89 mV. Iz tega sledi, da bo vsak korak A/D pretvorbe predstavljal 4,89 mV (http://www.st.com/resource/ja /application_note/cd00004444.pdf , zadnjič obiskano 11. januarja 2018).

Čeprav je analogni vhod namenjen branju analogne vrednosti, lahko analogni vhod uporabimo tudi kot digitalni vhod. Vrednost visokega ali nizkega nivoja signala pridobimo s programsko funkcijo, ki vrača konstanto HIGH pri visoki napetosti in LOW pri nizki (Ekart M., 2012 in https://www.arduino.cc /en/Tutorial/ AnalogInputPins, zadnjič obiskano 11. januarja 2018) .

C.8 Ostali priključki

Poleg priključkov z digitalnimi vhodi, izhodi ter analognimi vhodi se na levi strani mikrokrmilnika nahajajo še dodatni priključki, med katerimi sta priključek GND, ki omogoča povezavo skupne ničle (mase), ter priključek AREF (Slika 81). Priključek AREF je namenjen dovodu zunanje napetosti med 0 in 5 V. Ob ustrezni konfiguraciji v kodi je lahko napetost AREF referenca za maksimalno napetost pri branju analognih vrednosti (Ekart M., 2012 in https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/, zadnjič obiskano 21. januarja 2018).

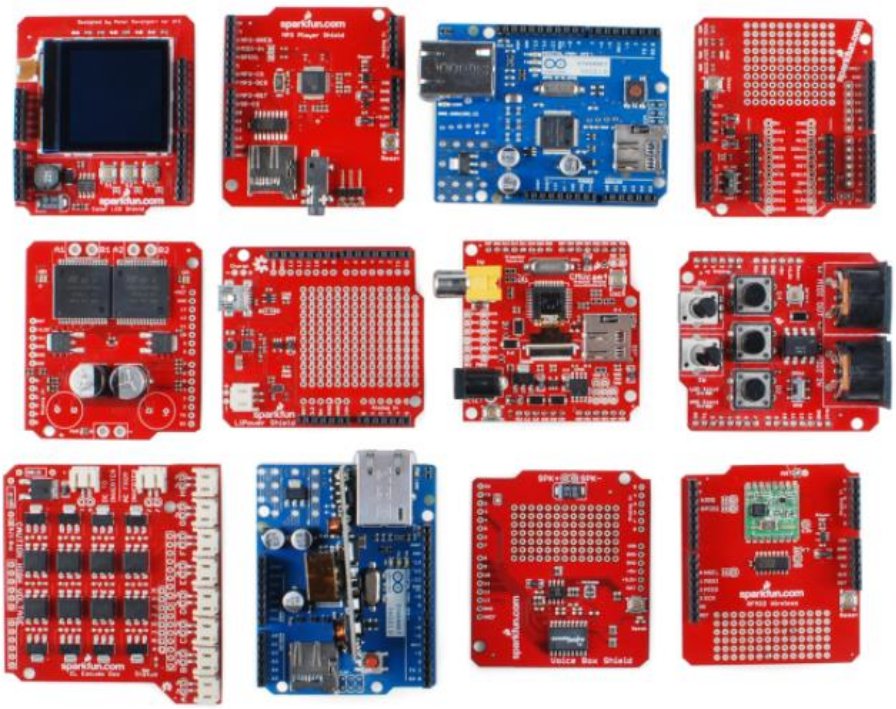


Vir: http://tronixstuff.com/wp-content/uploads/2013/12/Arduino-Uno-AREF.jpg

Slika 81: Priključka AREF in GND

C.9 Razširitveni moduli

V osnovni obliki nam Arduino mikrokrmilniki ponujajo omejeno število digitalnih in analognih priključkov. Kakšno drugo elektronsko komponento težko zasledimo. Nekatere različice Arduina imajo dodatke, kot so USB priklop, ali npr. bluetooth povezavo, vendar to včasih ne zadostuje. Pri razvoju merilne enote sem moral zagotoviti branje in prenos podatkov v mikrokrmilnik, hkrati pa sem potreboval tudi prenos podatkov po ethernet omrežju. Razširitveni moduli ali ščiti (angl. Shield) so tiskana vezja – moduli, ki omogočajo različna povezovanja zunanjih enot z mikrokrmilnikom Arduino (Slika 82). Na tržišču obstaja veliko število različnih modulov, ki pa se med seboj razlikujejo po delovanju in priključkih. To niso priključki, na katere je vmesnik priključen, ampak so tisti, ki jih potrebuje za delovanje. Dodatne funkcionalnosti, ki jih vmesnik ponuja, krmilimo s priključki (pini). Tako vsak vmesnik zasede določeno število priključkov, ki jih potrebuje za komunikacijo. Na tržišču lahko najdemo veliko različnih modulov, med katerimi so najpogostejši moduli za komunikacijo po ethernet omrežju, GPS moduli, GPRS moduli, relejski moduli, moduli za grafični prikaz itd. (Ekart M., 2012, Tomažinčič. T, 2015).



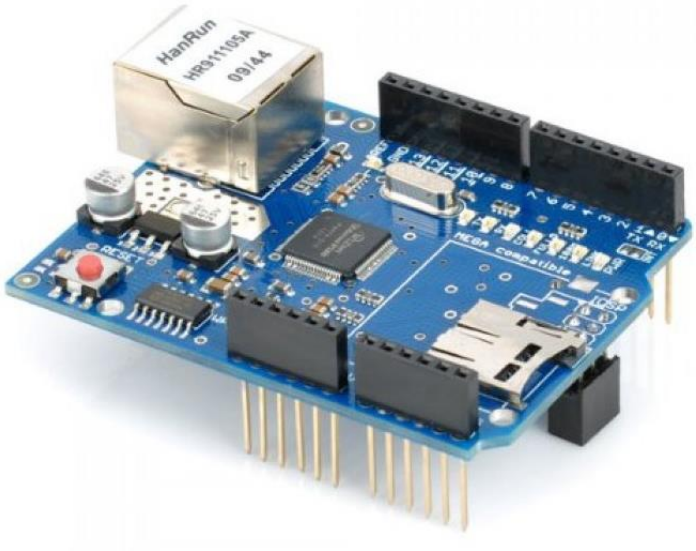
Vir: https://cdn.sparkfun.com/assets/3/d/8/6/1/5144cc2bce395fb170000002.jpg

Slika 82: Različne izvedbe modulov

C.9.1 Ethernet modul HanRun W5100

Ethernet modul (Slika 83) je razširitvena ploščica, ki omogoča povezljivost ter komunikacijo po ethernet omrežju. V merilni enoti sem modul uporabil za izmenjavo podatkov merilne enote z drugim napravami. Modul je zgrajen okrog integriranega vezja proizvajalca Wiznet, model W5100. S čipom je na razpolago robustna implementacija internetnega sklada, ki podpira protokol TCP kot tudi UDP. Ethernet modul vsebuje tudi signalne LED diode, ki nas obveščajo o delovanju. Signalne lučke so PWR, LINK, FULLD, 100M, RX, TX in COLL. Modul vsebuje tudi konektor (»priključek«) za uporabo pomnilniške SD kartice. S pomočjo knjižnic lahko podatke zapisujemo in beremo s pomnilniške kartice. Pri razvoju merilne enote sem na SD kartico zapisal konfiguracijsko datoteko, v kateri so potrebni parametri za vzpostavitev povezave z ethernet omrežjem. V primeru spremembe mrežnih nastavite tako ni potrebno spreminjati glavnega programa na mikrokrmilniku. Zadostuje, da SD kartico odstranimo in konfiguracijsko datoteko popravimo na drugem računalniku. Spremenjeno datoteko nato zapišemo na SD kartico in to ponovno vtaknemo v režo na modulu. Vsakič, ko merilno enoto priključimo na napetost, se vrši ponastavitev mrežnih parametrov. Za priklop modula z ethernet omrežjem je na razpolago standarden konektor za priklop omrežnega kabla

RJ45 (http://www.instructables.com/id/Arduino-Ethernet-Shield-Tutorial/, zadnjič obiskano 2. marca 2018).



Vir: https://www.fabtolab.com/ethernet-shield-hanrun

Slika 83: Ethernet in SD modul

C.10 Programska oprema mikrokrmilnika Arduino

V poglavju bom opisal osnove programskega jezika Arduina ter orodje za razvoj aplikacij, imenovano Arduino IDE. Na spletu obstajajo tudi druge različice razvojnih orodij (Visual Studio Code, NetBeans, Eclipse), ki pa se po izgledu in funkcionalnosti nekoliko razlikujejo.

C.10.1 Programski jezik Arduino

Programski jezik »Arduino« temelji na programskem jeziku C, ki so ga razvili v 70-ih letih, ter jeziku Processing, ki je dostopen na naslovu (https://processing.org/, zadnjič obiskano 12. marca, 2018). Jezik C je bil razvit za programiranje na UNIX operacijskih sistemih in je za začetnike težko berljiv in razumljiv. Zaradi tega so razvijalci Arduina razvili knjižnice, ki vsebujejo nabor enostavnih programskih funkcij, s katerimi upravljamo mikrokrmilnik. Same knjižnice so v osnovi napisane v programskem jeziku C/C++. Knjižnice nam omogočajo pisanje programov na enostavnejši in preglednejši način. Še vedno lahko celotno programsko kodo napišemo tudi v C/C++ jeziku z uporabo standardnih C ukazov. Uporaba knjižnic ima poleg prednosti tudi določene pomanjkljivosti, saj knjižnice vsebujejo določene funkcije, ki jih v svojem programu ne uporabljamo in posledično zasedejo določen del v pomnilniku. Z

uporabo standardnih knjižnic bo programska koda enostavnejša, pregledna in berljiva (Ekart M., 2012 in Evans B., 2011).

C.10.2 Pravilna struktura programa, osnovne funkcije in konstante

V osnovi lahko programski jezik razdelimo na podatkovne strukture, spremenljivke, konstante ter funkcije (metode). Pomembna funkcija je funkcija *setup()*, ki se izvede samo ob zagonu mikrokrmilnika. V omenjeni funkciji poteka inicializacija spremenljivk, struktur, nastavitev strojne opreme, nastavitev vhodov in izhodov ter nastavitev modulov (ščitov). Druga pomembna funkcija je *loop()*, ki se izvede takoj za funkcijo *setup()*. Kot že ime pove, se funkcija izvaja v neskončni zanki. V tej zanki se nahaja glavnina programa, ki deluje v mikrokontrolerju. Programska koda mora biti tako napisana, da znotraj funkcije preverjamo, do katerih sprememb je prišlo na vhodih in izhodih mikrokontrolerja in na te dogodke ustrezno reagiramo. Zavedati se moramo, da je mikrokrmilnik zelo hiter in zato lahko v eni sekundi izvede tudi več tisoč vrstic programske kode. Poleg tega lahko za zaznavo sprememb na vhodih uporabljamo prekinitve, ki jih bom obravnaval v nadaljevanju, kjer bom predstavil delovanje programa za štetje impulzov za določitev frekvence (Banzi M., 2009).

C.10.3 Osnovne knjižnice, funkcije in konstante programskega jezika

Osnovni programski paket vsebuje standardne knjižnice, s pomočjo katerih izvajamo programske operacije oziroma jih uporabljamo v programu. Glede na različne izvedbe mikrokrmilnika Arduino (Uno, Nano, Mega itd.) se le-te ob prevajanju programa prevedejo za ustrezen tip mikrokontrolerja. Zaradi odprtokodne tehnologije obstaja veliko število programskih knjižnic, katere si lahko razvijalec prenese s spleta ter jih uporablja v svojih projektih. Veliko že izdelanih knjižnic olajša programiranje ter zmanjša možnost napak pri programiranju, saj so v večini primerov že uporabljene in preizkušene.

Pred pričetkom programiranja mikrokrmilnika je priporočljivo poznati osnovne knjižnice, funkcije in konstante. Njihov nabor je obsežen, zato opišem le osnovne, ki sem jih uporabil pri programiranju prototipa in končnem izdelku merilne enote. Celoten nabor knjižnic za uporabo z mikrokrmilnikom Arduino je dostopen na spletnem naslovu (https://www.arduino.cc/en/Reference/Libraries, zadnjič obiskano 17. junija 2018).

Osnovne knjižnice:

- **SPI.h**: knjižnica omogoča uporabo SPI vodila, ki ga v mojem primeru uporabljam za komunikacijo z ethernet ščitom in čitalcem SD kartice.
- **SD.h**: knjižnica vsebuje funkcije za branje in zapisovanje podatkov na SD kartico.
- **Ethernet.h**: knjižnica vsebuje funkcije, ki omogočajo uporabo ethernet ščita z mikrokrmilnikom in ethernet omrežjem.

Osnovne konstante:

- **HIGH**: konstanta predstavlja binarno vrednost 1. Vrednost 1 zavzame takrat, ko napetost na vhodu mikrokrmilnika preide na visoki nivo. Visoki nivo je takrat, ko napetost na vhodnem priključku preseže napetost 2 V. Če uporabimo to konstanto kot vrednost izhoda na digitalnem priključku, bo le-ta oddajal maksimalno napetost (npr. 5 V).
- **LOW**: konstanta predstavlja ničelno vrednost napetosti na digitalnem vhodu. Vrednost je enaka nič pri napetosti na vhodu mikrokrmilnika, manjši od 0,8 V.
- **INPUT**: predstavlja način delovanja priključka. S to konstanto določimo, da bo izbrani priključek mikrokontrolerja deloval kot vhod.
- **OUTPUT**: S to konstanto določimo, da bo priključek mikrokontrolerja deloval kot izhod.

Osnovne funkcije:

- **DigitalRead**: funkcija sprejme parameter za številko priključka in prebere vrednost napetosti na priključku. Rezultat funkcije je konstanta LOW ali HIGH glede na jakost napetosti na priključku. Če priključek ni nikamor priključen, potem je vrednost, ki jo funkcija vrne, naključna in se stalno spreminja (https://www.arduino.cc /en/Reference/DigitalRead).
- **DigitalWrite**: funkcija sprejme dva parametra, ki sta številka priključka in digitalna vrednost, ki jo želimo nastaviti na izhodu. Funkciji lahko nastavimo dve vrednosti, in sicer maksimalno napetost (HIGH) in ničelno napetost (LOW) (https://www.arduino. cc/en/Reference/DigitalWrite).
- **AnalogRead**: funkcija sprejme parameter za številko priključka. S funkcijo preberemo analogno vrednost, ki je prisotna na priključku. Vrednost, katero mikrokrmilnik vrne, je 10-bitna A/D pretvorba. Vrednost 0 pomeni 0 V na priključku mikrokrmilnika, vrednost 1023 pa pomeni maksimalno napetost. Ta je lahko 3,3 V ali 5 V (odvisno od nastavitev mikrokrmilnika). Dobljeno vrednost lahko pretvorimo v »napetost« s pomočjo funkcije

map (https://www.arduino.cc/ en/Reference/AnalogRead, https://www.arduino.cc/en/Reference/Map, zadnjič obiskano 17. junija 2018).

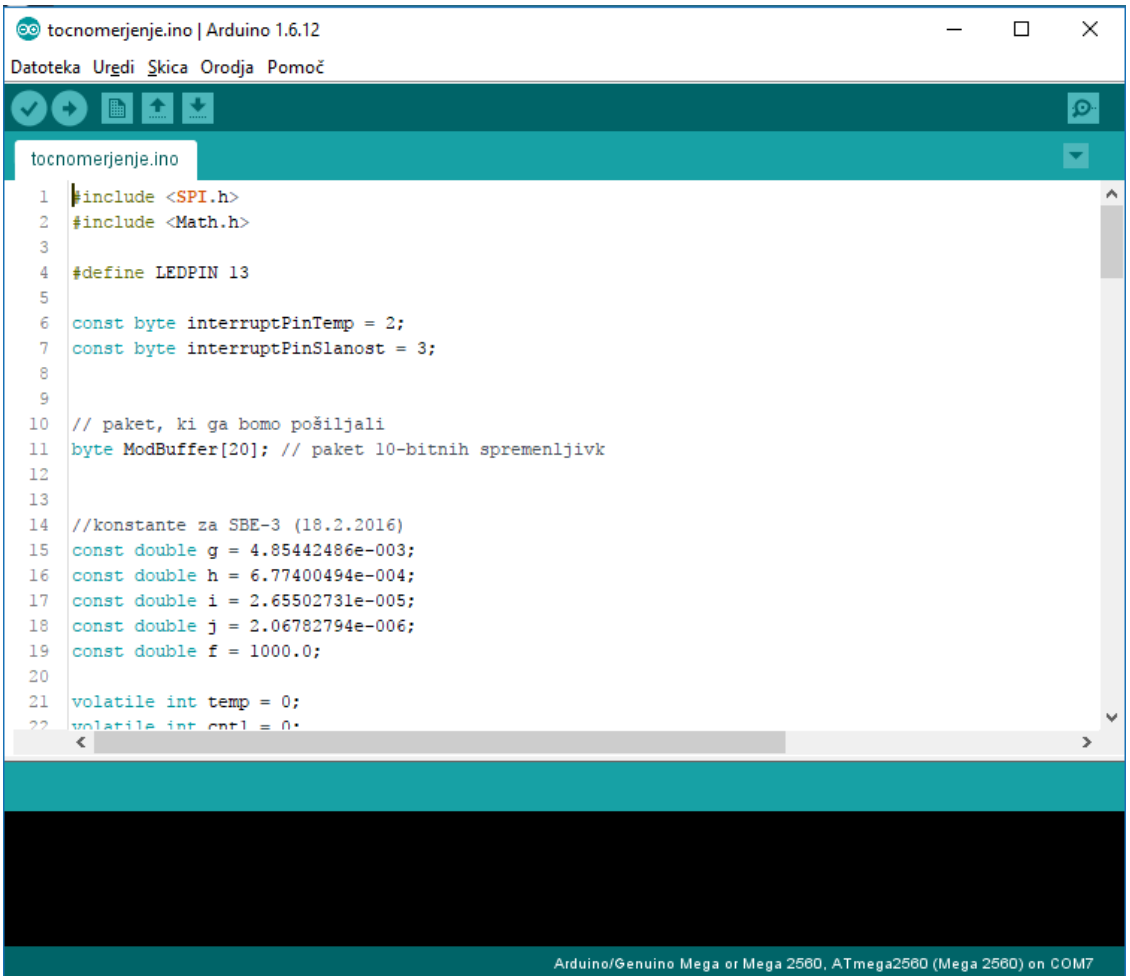
- **AnalogWrite**: funkcija sprejme dva parametra, in sicer priključek in analogno vrednost. Za razliko od analognega vhoda, ki je 10-bitni, je »analogni« izhod le 8-bitin in je v resnici digitalni izhod, ki se s hitrejšim preklapljanjem obnaša kot analogni izhod. To pomeni, da ima parameter lahko vrednosti od 0 do 255 (https://www.arduino.cc/en/Reference/AnalogWrite, https://svet-el.si/nova/215-17-3/, zadnjič obiskano 17. junija 2018).
- **PinMode**: funkcija sprejme dva parametra (številko priključka ter način delovanja), ki je lahko določen kot vhod (INPUT), izhod (OUTPUT) ali kot vhod z notranjim uporom za dvig nivoja (INPUT_PULLUP) (https://www.arduino.cc/ en/Reference/Constants, zadnjič obiskano 17. junija 2018).
- **Delay**: funkcija kot parameter sprejme število milisekund. Funkcija zaustavlja izvajanje glavnega programa (zanke) za podano število milisekund.
  Dodatno gradivo je dostopno na spletni strani (https://www.arduino.cc/, zadnjič obiskano 18. septembra 2017, Lenhart M., 2013 in Ekart M. 2012).

C.10.4 Razvojno okolje Arduino IDE

Za programiranje mikrokontrolerja sem uporabil program Arduino IDE (Slika 84), ki je dostopen na spletnem naslovu (https://www.arduino.cc/en/Main/Software, zadnjič obiskano 2. marca 2018). Arduino IDE je mogoče uporabljati z Windows, Mac OS in Linux operacijskimi sistemi in podpira različne izvedbe mikrokrmilnikov (Uno, Mega2560, Duemilanove, Mega, Diecimila itd.). Programski vmesnik je enostaven za uporabo in ga lahko razdelimo na štiri osnovne dele:

- Menijska vrstica: v menijski vrstici najdemo različne možnosti za upravljanje z datotečnim sistemom, urejevalnikom programske kode, prevajalnikom programske kode, orodji in pomoč.

- Orodna vrstica s pogosto uporabljenimi funkcijami, ki omogoča šest osnovnih funkcij, med katerimi so:
    - preverjanje programske kode,
    - nalaganje programske kode v mikrokrmilnik,
    - nova skica (nov projekt),
    - odpiranje poljubnega projekta iz datotečnega sistema,
    - shranjevanje projekta na datotečni sistem,
    - serijski vmesnik, ki prikaže okno (terminal), v katerem spremljamo delovanje mikrokontrolerja z UART povezavo.
- Urejevalnik programske kode, ki ga uporabljamo za pisanje programa. Ta na zaslonu zavzema največji del.
- Okno za izpis besedila, ki je nekakšen razhroščevalnik (angl. Debugger), s katerim spremljamo potek prevajanja in nalaganja programske kode.



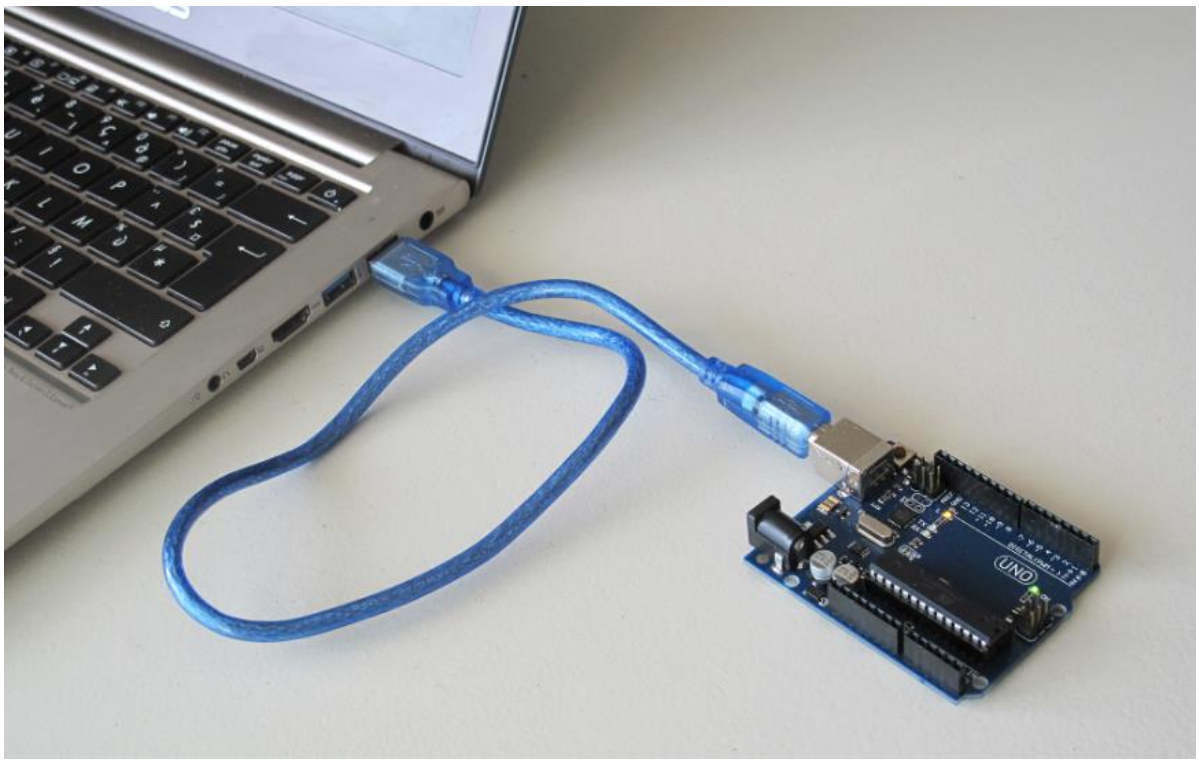
Vir: Lasten

Slika 84: Razvojno okolje Arduino IDE

Pred pričetkom programiranja najprej ustvarimo novi projekt oz. skico (ang. Sketch). Program bo tako na datotečnem sistemu računalnika ustvaril mapo z imenom projekta, vanj pa shranil datoteko s končnico »*.ino«. Datoteka s končnico »*.ino« je tekstovna datoteka, ki vsebuje programsko kodo. Vsak projekt je shranjen v svoji mapi in mora imeti enolično ime. Program podpira tudi datoteke s končnico ».cpp, .c in .h« , ki so standardne datoteke izvorne kode v programskem jeziku C/C++). Pomembno je tudi, da pred pričetkom pisanja programske kode v menijski vrstici (Orodja) izberemo ustrezen model mikrokontrolerja (https://www.arduino.cc/en/Guide/Environment, zadnjič obiskano 25. septembra 2017).

C.10.5 Prenos programa v mikrokrmilnik

Prenos prevedenega programa v mikrokrmilnik poteka z USB/UART pretvornikom (Slika 85). Pred uporabo na osebni računalnik namestimo ustrezne gonilnike, ki prepoznajo mikrokrmilnik kot zunanjo periferno enoto. Mikrokrmilnik je ob priklopu na računalnik razviden kot serijska vrata (COM). Nekatere različice mikrokrmilnika uporabljajo čipovje, katerega gonilniki so že del operacijskega sistema. V cenejših (predvsem kitajskih) različicah mikrokrmilnika pa se za komunikacijo uporablja čip z oznako CH340. Gre za cenejšo izvedbo USB/COM pretvornika, za katerega moramo predhodno namestiti ustrezne gonilnike. Gonilniki za omenjeni pretvornik so dostopni na spletnem naslovu (https://sparks.gogo.co.nz/ch340.html, zadnjič obiskano 13. marca 2018).



Vir: http://help.blynk.cc/how-to-connect-different-hardware-with-blynk/arduino/usb-serial

Slika 85: Povezava mikrokrmilnika z računalnikom

Dodatek D- Napajalnik in merilne enote in povezovalni kabli

Merilna enota potrebuje ustrezen napajalnik oziroma pretvornik napetosti. V merilni enoti uporabljam dva sklopa naprav, ki za delovanje uporabljata različno enosmerno napetost. Sondi SBE-3 in SBE-4 potrebujeta enosmerno napetost 12 V. Mikrokrmilnik napajamo z enosmerno napetostjo 8 V. V merilni enoti sem za pretvorbo napetosti uporabil preklopni napajalnik (angl. Switching power supply), ki izmenično napetost 220V pretvori v enosmerno napetost 12 V/3 A (Slika 86).

Vir: Lasten

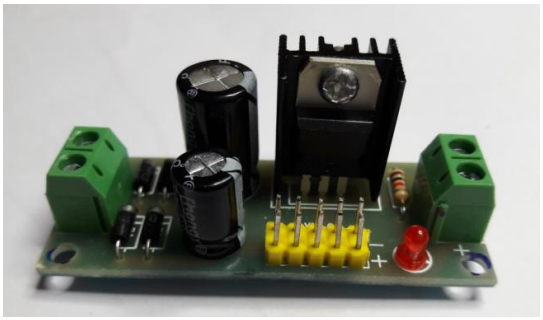
Slika 86: Preklopni pretvornik 220V / 12V

Pretvornik na sprednjem delu vsebuje priključno letev s petimi priključki:

- »L«: (angl. Live) je sponka, na katero priključimo vodnik, ki je pod električno napetostjo 220 V.
- »N« (angl. Neutral) je ničelni vodnik.
- »GND« (angl. Ground) je ozemljitev.
- »V-« je sponka s potencialom 0 V (ozemljitev).
- »V+« je sponka s potencialom +12 V enosmerne napetosti.

Na desni strani od priključne letve se nahaja manjši drsnik, s pomočjo katerega lahko natančno določimo izhodno napetost (fina nastavitev). Napetost nastavimo tako, da pretvornik priključimo na vir napetosti ter na sponki V- in V+ priključimo voltmeter in z drsnikom nastavimo želeno vrednost izhodne napetosti. Skrajno desno je prisotna tudi manjša kontrolna led dioda, ki sveti, kadar je pretvornik priključen na omrežno napetost.

Zaradi možnosti pregrevanja regulatorja napetosti na mikrokrmilniku sem uporabil dodaten pretvornik (stabilizator) napetosti, ki enosmerno napetost 12 V iz primarnega pretvornika zniža na 8 V napetost (Slika 87). Tako znižano napetost nato uporabim za napajanje mikrokrmilnika. Stabilizator napetosti je opremljen z dodatnim hladilnim rebrom, ki je namenjen odvajanju odvečne toplote. V pretvorniku je uporabljen stabilizator napetosti LM7805. Specifikacija stabilizatorja napetosti je dostopna na spletnem naslovu (https://www.sparkfun.com/ datasheets/Components/LM7805.pdf, zadnjič obiskano 14. marca 2018).



Vir: Lasten

Slika 87: Pretvornik napetosti 12V/8V

Povezovalni kabli povezujejo sondi in merilno enoto (Slika 88). Uporabljen je trižilni kabel, dolžine 2 m. Opremljen je z dvema priključkoma. Na strani sonde je prisoten ženski tripolni priključek, ki je izdelan iz gume zaradi boljšega vodotesnega tesnjenja pri priklopu na sondo. Poleg gumijastega priključka je na kablu prisoten še dodaten plastičen pokrovček, s katerim privijemo kabel na sondo. Na drugem koncu kabla je prisoten moški tip konektorja DIN 3 za priklop sonde na merilno enoto.

Vir: Lasten

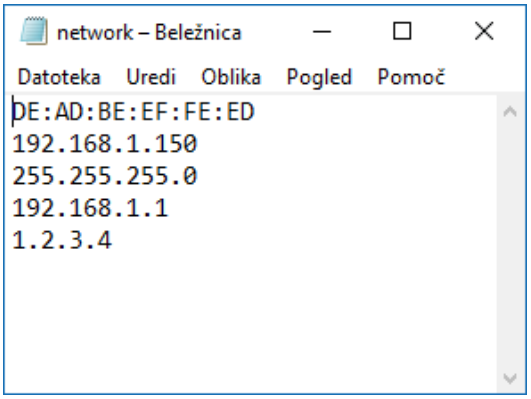
Slika 88: Povezovalni kabel sonde in merilne enote

Dodatek E- Izvorna koda za mikrokrmilnik in opis delovanja

Izvorna koda programa za mikrokontroler ArduinoUno je dostopna v javnem repozitoriju za programsko kodo Github na naslovu (https://github.com/petervalencic/MBP-Seabird-SBE3-SBE4/blob/master/Arduino/maincode.ino, zadnjič obiskano 15. aprila 2018).

Program se prične z navajanjem knjižnic. Knjižnice v program vključimo z ukazom #include <naziv_knjižnice>. Program uporablja tri standardne knjižnice, ki so namenjene uporabi ethernet ščita, komunikacije po SPI protokolu ter branju in pisanju na SD pomnilniško kartico. Sledi navajanje lokalnih spremenljivk, konstant, polj (angl. Array) ter razreda (angl. Class) EthernetServer, ki vzpostavijo komunikacijo ethernet ščita z Arduinom ter z lokalnim omrežjem (https://www.arduino.cc/en/Reference/EthernetServer, zadnjič obiskano 20. marca 2018). Pričetek izvajanja programske kode se prične v funkciji **setup()**. Funkcija se izvede le enkrat, takoj po vklopu mikrokrmilnika. V funkciji poteka ponastavljanje nekaterih pomembnejših nastavitev, kot je hitrost prenosa podatkov po serijskih vratih. Serijska vrata uporabljamo za razhroščevanje programa in za izpis drugih željenih vrednosti, predvsem v fazi razvoja in testiranja programa. Sledi določanje vhodov mikrokontrolerja, na katerih pričakujemo podatke. To storimo z ukazom pinMode. Ker bo program uporabljal zunanjo prekinitev, to določimo z uporabo ukaza attachInterrupt, kjer navedemo, kateri vhod bomo uporabili in katero funkcijo (rutino) bomo ob prekinitvi izvajali. V programu je določeno, da prekinitev na vhodu št. 2 izvede funkcijo beriTemperaturo(), kjer se poveča vrednost števca impulzov temperature, medtem ko prekinitev na vhodu št. 3 izvede funkcijo beriPrevodnost(), kjer se poveča vrednost števca impulzov prevodnosti. Časovno prekinitev (angl. Timer interrupt) določimo z uporabo registrov TCCR1A, TCCR1B, TCNT1, TCCR1B, TIMSK1. Podrobnosti nastavitev posameznega registra so dostopne na naslovu (https://www.robotshop.com/letsmakerobots/arduino-101-timers-and-interrupts, zadnjič obiskano 20. marca 2018). Z vnesenimi vrednostmi registra TCNT1 za takt 16 MHz tako določimo časovno prekinitev, ki se bo izvajala vsako sekundo. Časovna prekinitev se v programski kodi izvaja v prekinitveni rutini ISR(TIME1_OVF_VECT). Po nastaviti časovne prekinitve sledi branje podatkov iz pomnilniške mikroSD kartice, ki se nahaja na ethernet ščitu. Zaradi priklopa merilne enote na ethernet omrežje sem predvidel možnost nastavitve mrežnih parametrov s pomočjo konfiguracijske datoteke. Konfiguracijska datoteka je tekstovna datoteka z nazivom network.txt. Datoteka vsebuje pet vrstic, v katerih se nahajajo

sledeči parametri: MAC naslov, IP številka, maska omrežja, prehod in številka DNS strežnika. Slika 89 prikazuje vsebino nastavitvene datoteke.



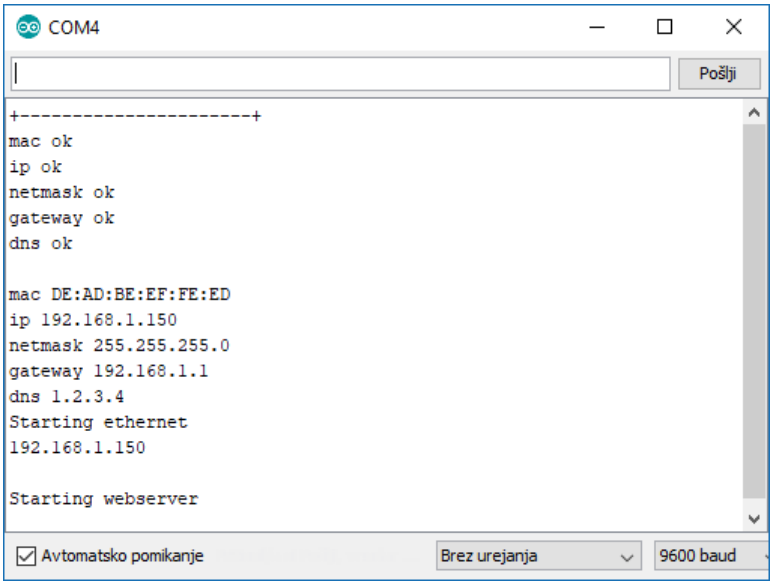
Vir: Lasten

Slika 89: Vsebina datoteke network.txt

- MAC naslov (angl. Media Acess Control) je zaporedje črk in številk, različno za vsako omrežno napravo. Prvi trije bajti naslova MAC predstavljajo identifikacijo izdelovalca strojne opreme. V kolikor je MAC naslov podan s strani proizvajalca, gre za registrirano identifikacijsko številko, ki se nahaja v imeniku, dosegljivem na spletnem naslovu (https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries, zadnjič obiskano 20. junija 2018). Zato se tudi MAC naslov imenuje fizični naslov. Pri večini omrežnih naprav (npr. pri čipih omrežnih kartic) ta naslov določi proizvajalec in ga naknadno ni možno spreminjati. Pri večini ostalih naprav pa imamo možnost, da ta naslov spreminjamo, saj sicer ne bi mogli omrežne naprave preprosto povezati z nekaterimi načini dostopa do interneta (https://sl.wikipedia.org/wiki/Naslov_MAC, zadnjič obiskano 21. marca 2018).
- IP naslov (angl. Internet Protocol). Ta napravam v medmrežju omogoča, da se prepoznajo, komunicirajo in delijo podatke. Najbolj razširjen standard naslovov IP je IPv4 naslov, sestavljen iz štirih s piko ločenih številk.
- Podmaska omrežja (angl. Subnet Mask) je v enakem formatu kot IP številka in jo potrebujemo za delovanje protokola TCP/IP. Protokol TCP/IP na podlagi maske podomrežja določi, ali je gostitelj v lokalnem podomrežju ali v oddaljenem omrežju (https://support.microsoft.com/sl-si/help/164015/understanding-tcp-ip-addressing-and-subnetting-basics, zadnjič obiskano 21. marca 2018).

- Prehod (angl. Gateway) je skupek naslovov strojne in programske opreme, ki je potrebna za komunikacijo dveh tehnološko različnih omrežij in zagotavlja pretvorbo protokolov iz ene omrežne arhitekture v drugo. Naloga prehoda je, da različnim računalniškim sistemom, ki med seboj niso neposredno združljivi, dajejo občutek, kot da komunicirajo z enakim sistemom na drugi strani. V veliko primerih ima naprava IP prehod identičen svojemu, le da ima za svoj naslov gostitelja za število 1 (npr. za naslov IP 192.168.1.150 je prehod navadno 192.168.1.1). IP naslov prehoda dobimo pri upravljalcih omrežja (http://wiki.fmf.uni-lj.si/wiki/Prehod, zadnjič obiskano 21. marca 2018).
- DNS številka strežnika (angl. Domain Name Server) je naslov za dostop do DNS strežnika. DNS strežnik je sistem za domenska imena. Je informacijska storitev, namenjena pretvarjanju domenskih imen strežnikov v internetne naslove (IP številke).

Branje datoteke poteka tako, da se prebere vrstica za vrstico. Ker je v prvi vrstici zapisana MAC številka, bo program za pridobitev MAC številke uporabil funkcijo getMAC(char* macBuf, byte* thisMAC). Ostale štiri vrstice vsebujejo podatke, ki so enakega formata (IP številka, podmaska omrežja, prehod in DNS), zato uporabimo funkcijo getIP(char* ipBuf, byte* thisIP). Pri zapisovanju parametrov v lokalne spremenljivke se sočasno skozi serijska vrata izpisujejo prebrani parametri. Na ta način lahko ugotovimo, ali je program pravilno prebral podatke iz nastavitvene datoteke in ali so parametri pravilni (Slika 90). V primeru, da je v omrežju prisoten DHCP strežnik, bodo temu samodejno dodeljeni IP naslov, podmaska omrežja, prehod in DNS naslov strežnika. Več o delovanju DHCP strežnika je objavljeno na spletnem naslovu (http://www.s-sers.mb.edus.si/gradiva/rac/drugo/omrezja/60_storitve/05_datoteka.html, zadnjič obiskano 21. marca 2108).



Vir: Lasten

Slika 90: Izpis mrežnih nastavitev s serijskimi vrati mikrokrmilnika

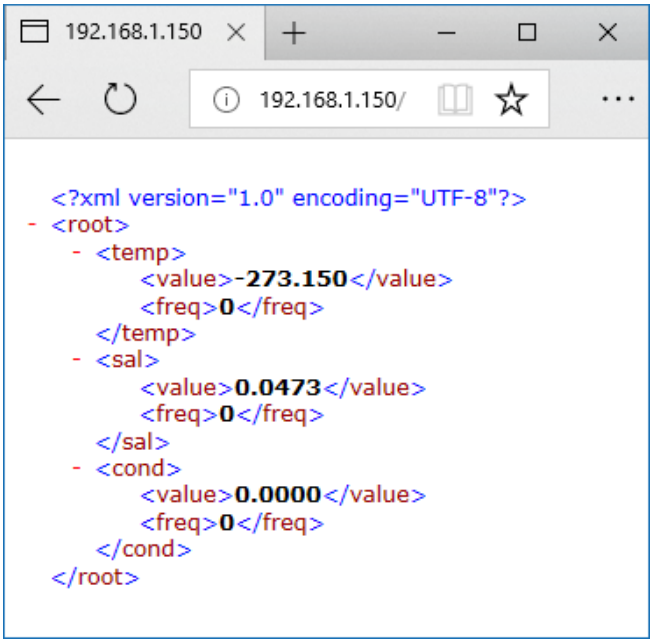
Po branju konfiguracijske datoteke sledi nastavitev ethernet knjižnice z ukazom Ethernet.begin (myMac, myIP, myDNS, myGW, myNM). Takoj za tem se zažene strežnik z ukazom server.begin(). Ta spremlja ukaze in podatke na določeni IP številki in vratih 80 (angl. Ports). Vrata v računalništvu pomenijo vmesnik, skozi katerega pošiljamo in prejemamo podatke in se navezuje na komunikacijo po ethernet omrežju. Vrata 80 so privzeta vrata za komunikacijo po HTTP protokolu. Zadnji ukaz v setup() funkciji je klic funkcije interrupts(). Omenjena funkcija sporoči mikroprocesorju, da prične izvajati in spremljati prekinitve. Po inicializaciji programskih funkcij, razredov in spremenljivk sledi izvajanje programa v neskončni zanki void loop().

Neskončna zanka pomeni, da se bo programska koda znotraj zanke izvajala v nedogled, razen če pride do večje programske napake (hrošča) ali okvare mikrokrmilnika. Če si podrobneje pogledamo vsebino loop() funkcije (https://github.com/petervalencic/MBP-Seabird-SBE3-SBE4/blob/master/Arduino/maincode.ino, zadnjič obiskano 21. junija 2018) opazimo, da se v zanki nahajata dva »if« pogoja. Prvi »if« stavek je v prvi iteraciji zanke izpolnjen. To pomeni, da se bo pobrisala vrednost spremenljivk f_temp in f_prevodnost. V ti dve spremenljivki program ob poteku časovne prekinitve zapiše število preštetih impulzov (frekvenco). S tem pogojem izločimo prvo meritev, saj lahko odjemalec zahteva podatke še pred pretekom prve sekunde. V tem primeru program posreduje napačno število prebranih impulzov (frekvenc) in posledično napačno preračunano temperaturo in prevodnost/slanost. Za preverjanjem pogoja

»bPrvaMeritev« poteka preverjanje prisotnosti zahteve po branju podatkov iz ethernet omrežja. Spremenljivka, ki nam sporoča, ali je uporabnik zahteval podatke, se imenuje »client« in je podatkovnega tipa, boolean. V primeru, da je pogoj izpolnjen, se preveri z »if« stavkom naslednji pogoj, in sicer »client.available()«, to je prisotnost odjemalca. Pogoj se uporabi v »while« zanki in v vsaki iteraciji, program preveri prisotnost odjemalca. Znotraj vsake iteracije se v spremenljivko »char c« vpiše podatkovni tok odjemalca. Ko v zanki program prebere zadnji znak odjemalca, ki je »\n« in označuje konec vrstice, se prične priprava podatkov. Ko je izpolnjen pogoj »(c =='\n' && bTekocaVrstica)«, program izračuna temperaturo ter slanost. Izračun temperature se izvede s funkcijo »double calcTemp(double frekvenca)«, prevodnost s pomočjo funkcije »double calcPrevodnost(double frekvenca, double temperatura)« in slanost s funkcijo »double calcSlanost(double t, double c, double p)«. Pri izračunu prevodnosti bi morali praviloma upoštevati tudi temperaturo in tlak, vendar je vpliv tlaka v globinah, manjših od 10 m, zanemarljiv (International oceanographic tables Unesco, 1978). Pri izračunu slanosti se upošteva izmerjena temperatura, prevodnost morske vode, medtem ko se tlak zanemari. Če je frekvenca > 0, s funkcijami dobimo podatke temperature, prevodnosti in slanosti morske vode. Temperatura je zabeležena v spremenljivki »double local_temp«, prevodnost v »double local_cond«. Slanost se v programu ne shranjuje lokalno, ampak se izračuna na zahtevo odjemalca. Funkcija za preračun slanosti ima tri vhodne atribute, ki so temperatura v °C, prevodnost v mS/cm in tlak v dBar-ih (https://github.com/petervalencic/MBP-Seabird-SBE3-SBE4/blob/master/Arduino/maincode.ino, vrstica 238, zadnjič obiskano 21. junija 2018). Priprava izhodnih podatkov je v XML obliki. To je označevalni jezik, ki ga je možno razširjati z novimi značkami (oznakami) in je tako kot HTML označevalni jezik ali jezik, katerega ključni element so značke. Razvit je bil za opisovanje podatkov. V primeru merilne enote sem za predstavitev podatkov uporabil sledečo obliko XML zapisa:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <temp>
    <value>-273.150</value>
    <freq>0</freq>
  </temp>
  <sal>
    <value>0.0473</value>
  </sal>
  <cond>
    <value>0.0000</value>
    <freq>0</freq>
  </cond>
</root>
```

XML podatki se pričnejo z vrstico, ki označuje XML datoteko »<?XML version='1.0' encoding='UTF-8'?>«. Poleg verzije XML dokumenta je naveden še atribut »encoding«, ki določa kodni nabor, v katerem so zapisani podatki. V mojem primeru sem uporabil kodni nabor UTF-8. Kodni nabor vsebuje vse razpoložljive znake, ki so uporabljeni za prikazovanje podatkov. Ta lastnost ga ločuje od ostalih in ga postavlja med najbolj pogosto uporabljen kodni nabor. XML dokument lahko vsebuje samo en korenski element (značko). V mojem primeru je to element »<root>«, ki se prične v drugi vrstici dokumenta in zaključi v zadnji vrstici z »</root>«. Znotraj korenskega elementa se nahajajo trije elementi: <temp>, <sal> in <cond>. Elementa <temp> in <cond> vsebujeta dodatna dva podelementa, v katerih je zapisana frekvenca <freq> ter izračunana vrednost za temperaturo in slanost, ki se nahaja v elementu <value>. Element <sal> vsebuje le element <value>, saj se slanost preračuna iz izmerjene temperature in prevodnosti. Na takšen način sem omogočil prenos podatkov iz merilne naprave po HTTP protokolu v XML obliki. XML oblika zapisa podatkov je zelo razširjena in jo je možno uporabljati z vsemi sodobnimi programskimi jeziki in orodji, kot so recimo Excel in ostali. Z napravo se lahko povežemo tudi z uporabo brskalnika. Rezultat je XML datoteka, ki je prikazana na spodnji sliki (Slika 91) (http://salinometry.com/pss-78/, zadnjič obiskano 23. aprila 2018).



Vir: Lasten

Slika 91: Izpis podatkov v XML obliki

Po pošiljanju podatkov v ethernet omrežje moramo odprto povezavo zapreti z ukazom »client.stop«. Ker pa se poleg glavne zanke izvajata še dve zunanji prekinitvi in časovna prekinitev v intervalu sekunde, opišem še ostali del programske kode. Časovna prekinitev se izvede ob izpolnjenem pogoju »ISR(TIMER1_OVF_VECT«. V tej prekinitvi so dovoljene le krajše programske operacije, zato se v časovni prekinitvi poveča vrednost registra TCNT1, ki ga uporabljamo za časovno prekinitev ter tako omogočimo ponovno proženje v času ene sekunde. Sledi prepis preštetih impulzov temperature in prevodnosti v spremenljivki »f_temp« in »f_prevodnost«. Takoj za tem se števec impulzov »cnt_temp« in »cnt_prevodnost« postavita na vrednost nič. Dejansko štetje impulzov, to je povečava števca impulzov, se izvede z rutinama »beriTemperatuo()« in »beriPrevodnost()«. Ti dve rutini sta poklicani, kadar pride do zunanje prekinitve. To pomeni, da bo na vhodu mikrokrmilnika, kjer beremo impulze iz SBE-3 in SBE-4, signal v »visokem« stanju (+5 V).

Dodatek F- Funkcijski generator za izvedbo testiranja

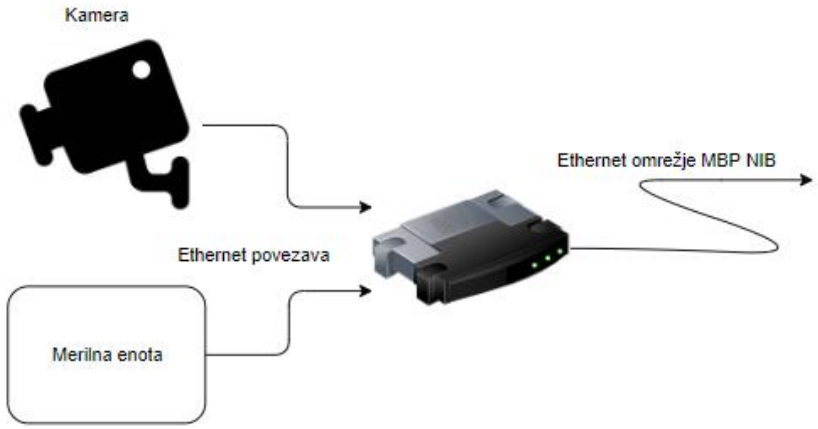
Po razvoju prototipa sem pretvornik testiral. Pri tem sem uporabil funkcijski generator Rigol, model DG1022. Funkcijski generator (Rigol, model DG1022) je naprava, s katero generiramo različne oblike izhodnega signala (Slika 92). Obliko izhodnega signala, frekvenco in amplitudo lahko spreminjamo v realnem času z gumbi na sprednji strani naprave. Naprava omogoča generiranje signala na dveh izhodih (kanalih), zato je bila idealna za testiranje delovanja pretvornika signala. Po opravljenih testiranjih s pomočjo funkcijskega generatorja sem lahko na pretvornik priključil tudi sondi SBE-3 in SBE-4.

Vir: Lasten

Slika 92: Funkcijski generator RIGOL DG1022

Dodatek G- Stikalo za povezavo merilne enote in kamere

Povezovalni člen merilne enote in podvodne kamere ter LAN omrežja je stikalo (angl. Switch) (Slika 93). Stikalo je naprava, ki deluje na sloju podatkovne povezave in vsebuje več priključkov. Uporabljamo ga za ohranitev pasovne širine na omrežju. V projektu sem uporabil stikalo, ki vsebuje pet priključnih vrat in omogoča delovanje s hitrostjo 100/10 Mbps.



Vir: Lasten

Slika 93: Povezava stikala z merilno enoto in kamero

Dodatek H- Programski in skriptni jeziki, uporabljeni v nalogi

H.1 Java programski jezik

Program za asinhrono komunikacijo z merilno enoto in spletna aplikacija sta napisana v programskem jeziku Java. To je objektno orientiran programski jezik, v katerem je osnovni gradnik objekt (angl. Object), ki ga definira razred (ang. Class), sestavljajo pa ga lastnosti (atributi – spremenljivke) in metode (funkcije), ki upravljajo s podatki objekta. Programska koda napisana v Javi, je platformsko neodvisna, kar pomeni, da je prenosljiva med različnimi operacijskimi sistemi. Beseda Java označuje tudi programsko platformo. Javanska platforma obsega programski jezik, izvajalno okolje in programske knjižnice. Java velja za eno najbolj popularnih platform za razvoj spletnih aplikacij, ki temeljijo na arhitekturi odjemalec – strežnik (angl. Client server). Programski paket Java si lahko brezplačno prenesete s spletne strani (https://java.com/en/download/, zadnjič obiskano 29. marca 2018) (Mesojedec U., 2004).

H.2 XPath poizvedovalni jezik

XPath jezik nam omogoča dostop do elementov in atributov v strukturi ali drevesu XML dokumenta. Ime pomeni način iskanja po drevesih, saj se v programu sprehaja po gradnikih dokumenta s pomočjo sintakse, ki ni v XML obliki. Rezultat izraza XPath je lahko nabor vozlišč, entiteta ali pa drugo zaporedje, opisano s sintakso, ki ga podpira podatkovni model. Glavni konstrukt sintakse je XPath izraz, s katerim poiščemo objekt v dokumentu. Nabor možnih rezultatov je lahko: neurejen seznam vozlišč, logična vrednost, zapis s plavajočo vejico ali niz znakov.

Primer dostopa do podatkov merilne enote z uporabo Xpath sintakse:

```
//root/temp/value/text()
//root/temp/freq/text()
//root/cond/value/text()
//root/cond/freq/text()
//root/sal/value/text()
```

H.3 REST spletna storitev

REST predstavlja arhitekturo, v kateri je vsak vir predstavljen kot spletna storitev z enoznačnim naslovom URL. REST sam po sebi ni standardiziran protokol, opredeljuje pa, kako uporabljati obstoječe standarde. Načelo REST je uporaba protokola HTTP, tako kot je modeliran. Za dostopanje in spreminjanje virov se tako uporabljajo standardne metode HTTP: GET, PUT, POST in DELETE. Na enak način delujejo tudi spletni brskalniki, ki z uporabo naslovov URL izvajajo zahteve na strežniku, ko je rezultat datoteka HTML. Zaradi svoje prilagodljivosti je protokol HTTP primeren tudi za izmenjavo podatkov, ki nimajo vizualne komponente. Arhitektura REST tako izkorišča razširjenost protokola HTTP in njegove zmožnosti, zato je odlična za prenos podatkov (Rajnar M., 2014, str. 13).

H.4 JSON format za izmenjavo podatkov

JSON (angl. JavaScript Object Notation) je preprosta oblika za izmenjavo podatkov, ki je neodvisna od programskega jezika. Zaradi besedne zasnove je tak zapis enostaven za branje in pisanje tako ljudem kot tudi računalnikom. Uporablja se predvsem za izmenjavo podatkov med strežnikom in spletno aplikacijo. V primerjavi z XML je JSON manjši in hitrejši (Rajnar M., 2014, str. 35). Več o strukturi JSON zapisa lahko dobite na spletnem naslovu (https://www.json.org/json-sl.html, zadnjič obiskano 3. aprila 2018).

H.5 MySQL podatkovna baza

MySQL je eden izmed najbolj popularnih in uporabljenih odprtokodnih sistemov za upravljanje z bazami. Razvoj podatkovne baze se je pričel leta 1995 na Švedskem v podjetju MySQL AB. Kasneje je podjetje in njegov produkt MySQL prevzelo ameriško podjetje Oracle Corporation. MySQL uporablja model na principu strežnik – odjemalec, kar pomeni, da bazni strežnik streže ostalim odjemalcem. Nudi tudi podporo strukturiranemu programskemu jeziku SQL, ki je standardiziran jezik za delo in upravljanje s podatki. MySQL je relacijska baza podatkov, kar pomeni, da so podatki razporejeni ali zapisani v relacijskih tabelah in ne v eni podatkovni datoteki. Takšen način nam omogoča preglednejšo zbirko in hitrejše poizvedovanje po vsebini podatkovne baze. Do strežnika z MySQL bazo lahko dostopamo z različnimi vmesniki, ki posredujejo SQL ukaze do strežnika in prikažejo rezultat.

Relacijska baza MySQL deluje na različnih platformah in si jo lahko brezplačno prenesemo z uradne spletne strani (https://dev.mysql.com/, zadnjič obiskano 29. marca 2018).

H.6 NetBeans IDE urejevalnik programske kode

NetBeans IDE je odprtokodno integrirano razvojno okolje, ki je bilo postavljeno leta 2000 v podjetju Sun Microsystems. Razvojno okolje je napisano s programskim jezikom Java in deluje na vseh operacijskih sistemih, kjer je nameščen. NetBeans IDE omogoča uporabnikom široko paleto funkcij za izdelavo Java namiznih aplikacij, Jave EE (angl. Enterprise Edition) in spletnih aplikacij ter vizualni razvoj spletnih in mobilnih aplikacij. Orodje vsebuje različne elemente, ki jih potrebujemo za hiter in učinkovit razvoj aplikacij, med katerimi so: prevajalnik programske kode, refaktoriranje (angl. Refactoring) programske kode, razhroščevalnik (angl. Debugger), kontrolo verzij (verzioniranje), testiranje enot, dopolnjevanje programske kode, vključitev različnih podatkovnih baz, aplikacijskih strežnik in spletnih storitev (angl. Webservices). Pri razvoju programa za pregled in poizvedbo podatkov sem se odločil za program NetBeans 8.2 zaradi njegove enostavnosti. Programski paket NetBeans je dosegljiv na spletnem naslovu (https://netbeans.org/, zadnjič obiskano 29. marca 2018).

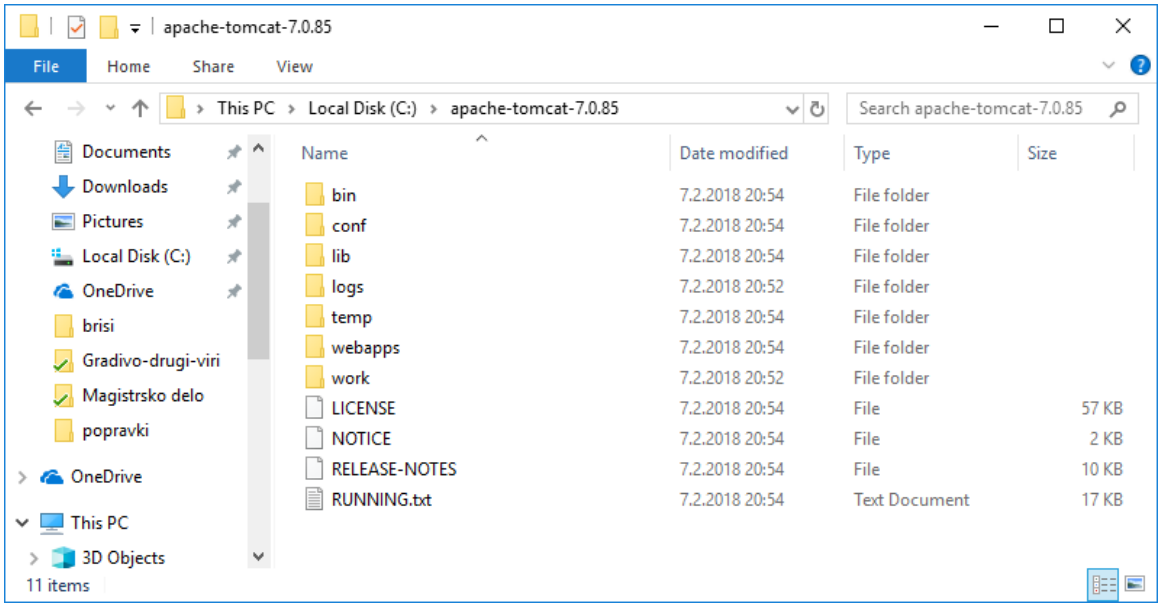
H.7 Apache Tomcat aplikacijski strežnik

Za razvoj in poganjanje spletne aplikacije sem uporabil aplikacijski strežnik Apache Tomcat 7.0. Program je dostopen na spletnem naslovu (https://tomcat.apache.org/download-70.cgi, zadnjič obiskano 30. marca 2018). Za namestitev aplikacijskega strežnika je potrebno preneseno datoteko razširiti v predvideno mapo.

Aplikacijski strežnik vsebuje sledečo strukturo podmap (Slika 94), ki so namenjene različnim opravilom:

- bin mapa vsebuje datoteke za upravljanje oziroma zagon aplikacijskega strežnika,
- conf mapa vsebuje nastavitvene datoteke aplikacijskega strežnika,
- lib mapa vsebuje javanske knjižnice, ki so dosegljive vsem nameščenim aplikacijam na aplikacijskem strežniku (angl. Shared librarys),

- logs mapa vsebuje dnevniške datoteke aplikacijskega strežnika ter dnevniške datoteke nameščenih aplikacij, ki so prisotne v mapi webapps,
- temp mapa vsebuje začasne datoteke, ki jih med delovanjem uporablja aplikacijski strežnik,
- webapps mapa vsebuje spletne aplikacije; v omenjeno mapo uporabnik namesti aplikacijo, ki jo želi izvajati s pomočjo aplikacijskega strežnika,
- work mapa je mapa, v katero aplikacijski strežnik shranjuje prevedene spletne aplikacije, ki so nameščene v mapi webapps.

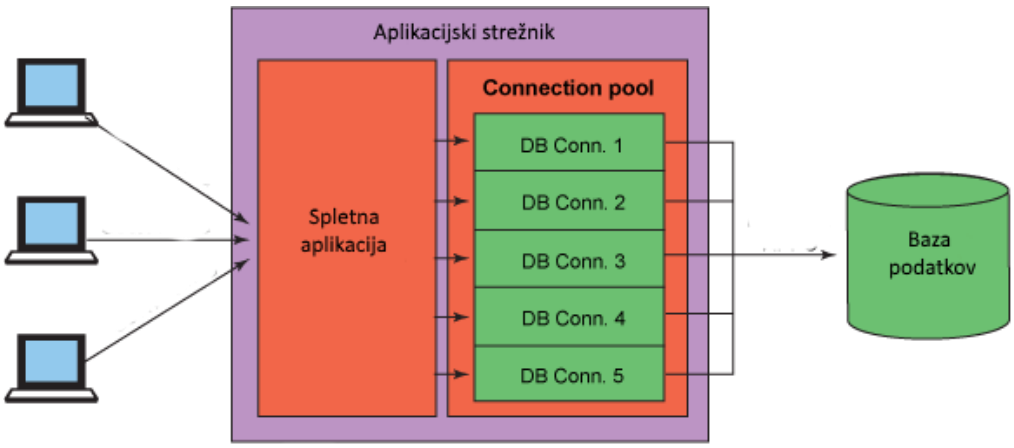


Vir: Lasten

Slika 94: Struktura map aplikacijskega strežnika Tomcat

H.7.1 Nastavitev dostopa do relacijske baze podatkov

Za dostop do baze podatkov je potrebno v aplikacijskem strežniku zagotoviti »sklad povezav« (angl. Connection Pool) (Slika 95). Sklad povezav pomeni, da ob zagonu aplikacijskega strežnika slednji vzpostavi določeno število povezav do željene baze podatkov. Prednost takšnega povezovanja je hitrost, saj bi v nasprotnem primeru spletna aplikacija na zahtevo uporabnika najprej vzpostavila povezavo z baznim strežnikom, kar je časovno potratno. Sklad povezav omogoča aplikaciji, ki se izvaja na aplikacijskem strežniku, povezavo do baze podatkov s prvo prosto povezavo, ki se nahaja v skladu povezav.



Vir: Lasten

Slika 95: Aplikacijski strežnik in sklad povezav

Pred nastavljanjem sklada povezav moramo zagotoviti ustrezen gonilnik, ki omogoča aplikacijskemu strežniku povezavo do baze MySQL. Za to namestimo gonilnik mysql-connector-java-5.1.46.jar, ki je dostopen na naslovu (https://dev.mysql.com/downloads/connector/j/5.1.html, zadnjič obiskano 30. marca 2018). Preneseno datoteko je potrebno umestiti v mapo $CATALINA_HOME/lib (https://tomcat.apache.org/tomcat-8.0-doc/setup.html, zadnjič obiskano 3. marca 2018).

Sklad povezav določimo v datoteki /conf/context.xml. Sintaksa za nastavitev sklada je sledeča:

```
<data-source
name="MySqlDS" location="jdbc/MySqlPooledDS"
class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource"
max-connections="100"
min-connections="5"
username="UPORABNIŠKO IME"
password="GESLO"
url="jdbc:mysql://localhost:3306/?useUnicode=true"/>
```

Element »data-source« ima sledeče atribute, ki določajo lastnosti sklada povezav:

- *Name* je enolično poimenovanje izvora podatkov (angl. Data Source).
- *Location* je poimenovanje povezave, ki jo uporabimo v programski kodi za sklicevanje na podatkovni sklad. V programskem jeziku Java dostopamo do podatkovnega sklada z JNDI

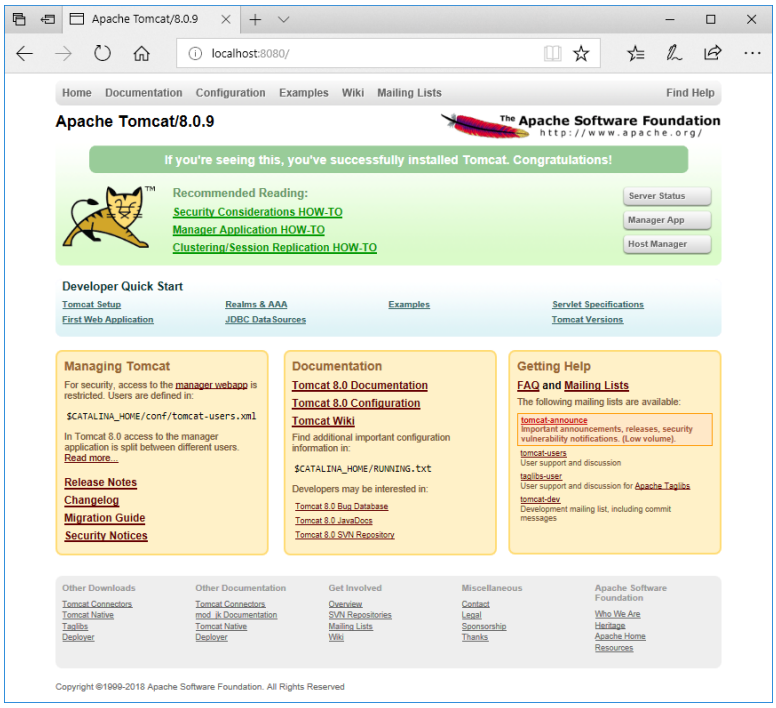
povezavo (angl. Java Naming and Directory ineterface). Več o uporabi JNDI je dostopno na naslovu https://docs.oracle.com /javase/tutorial/jndi/overview/ index.html (zadnjič obiskano 30. marca 2018).

- *Class* določa razred, ki ga sklad povezav uporablja za povezavo do podatkovne baze (angl. JDBC Driver). Istoimenski razred se nahaja v arhivu mysql-connector-java-5.1.46.jar.
- *Max-connections* določa število povezav, ki jih lahko vzpostavi aplikacijski strežnik zaradi zasedenosti drugih povezav.
- *Min-connections* določa minimalno število povezav do baze podatkov, ki jih aplikacijski strežnik vzpostavi ob zagonu.
- *Username* je uporabniško ime, s katerim se vzpostavi povezava do baze.
- *Password* je geslo za prijavo na bazo podatkov.

Po namestitvi sklada povezav aplikacijski strežnik ponovno zaženemo z ukazom:

```
c:\>Tomcat\bin\catalina stop
c:\>Tomcat\bin\catalina run
```

Delovanje aplikacijskega strežnika preverimo na lokalnem naslovu »http://localhost:8080«, ob uspešni namestitvi in zagonu strežnika se na zaslonu prikaže spodnja slika.



Vir: Lasten

Slika 96: Uvodna spletna stran aplikacijskega strežnika Tomcat

Dodatek I- Program za asinhroni prenos podatkov v podatkovno bazo in posredovanje podatkov s pomočjo spletne storitve

I.1 Program za asinhroni prenos podatkov

Komunikacija z merilno enoto poteka s pomočjo razreda »MerilnaNapravaListener«, ki implementira vmesnik »ServletContextListener« (https://docs.oracle.com/javaee/6/api/javax/servlet/ServletContextListener.html, zadnjič obiskano 2. aprila 2018). Omenjeni vmesnik vsebuje dve metodi, ki nam sporočata, kdaj se je začel pričetek izvajanja spletne aplikacije in kdaj se spletna aplikacija zaključi. Razred »MerilnaNapravaListener« tako ob prejemu dogodka »public void contextInitialized(ServletContextEvent servletContextEvent)« požene asinhroni časovnik, ki se izvaja enkrat na minuto. Asinhroni časovnik je razširitev razreda »TimerTask« (https://docs.oracle.com/javase/7/docs/api/java/util/TimerTask.html, zadnjič obiskano 2. aprila 2018).

Primer asinhronega časovnika:

```
package org.mbp.mbpprojekt.servlet.listener;

import java.io.IOException;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.Timer;
import java.util.TimerTask;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.sql.DataSource;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
```

```
public class MerilnaNapravaListener implements ServletContextListener {

    final String DATASOURCE_CONTEXT = "java:comp/env/jdbc/PETERDB";
    String urlNaslov = "http://192.168.1.150";

    public String getUrlNaslov() {
        return urlNaslov;
    }

    public void setUrlNaslov(String urlNaslov) {
        this.urlNaslov = urlNaslov;
    }

    public static final Logger logger = Logger.getLogger(MerilnaNapravaListener.class.getName());

    @Override
    public void contextDestroyed(ServletContextEvent arg) {

        logger.log(Level.INFO, "MerilnaNapravaListener je uničen..");

    }

    private Connection getConnection() throws SQLException, NamingException {
        Connection result = null;
        DataSource datasource = null;

        Context initialContext = new InitialContext();
        datasource = (DataSource) initialContext.lookup(DATASOURCE_CONTEXT);
        if (datasource == null) {
            logger.warning("Failed to lookup datasource: " + DATASOURCE_CONTEXT);
        }

        return datasource.getConnection();
    }

    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {

        logger.log(Level.INFO, "MerilnaNapravaListener je inicializiran..");
        TimerTask vodTimer = new VodTimerTask();
        Timer timer = new Timer();
        timer.schedule(vodTimer, 1000, (60 * 1000));

    }

    class VodTimerTask extends TimerTask {

        DocumentBuilderFactory dbf;
        DocumentBuilder db;
        Document doc;
        XPathFactory xpathFactory;

        Connection con;
        PreparedStatement stmt;

        @Override
        public void run() {

            String temperatura;
            String slanost;
            String prevodnost;
            String frekvencaTemperatura;
            String frekvencaPrevodnost;

            logger.log(Level.INFO, "Klic merilne naprave na IP:{0}", getUrlNaslov());
            try {
```

```
            dbf = DocumentBuilderFactory.newInstance();
            db = dbf.newDocumentBuilder();
            doc = db.parse(new URL(getUrlNaslov()).openStream());
            xpathFactory = XPathFactory.newInstance();
            XPath xpath = xpathFactory.newXPath();

            XPathExpression expr = xpath.compile("/root/temp/value/text()");
            NodeList nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            temperatura = nodes.item(0).getNodeValue().trim();

            expr = xpath.compile("/root/temp/freq/text()");
            nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            frekvencaTemperatura = nodes.item(0).getNodeValue().trim();

            expr = xpath.compile("/root/cond/value/text()");
            nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            prevodnost = nodes.item(0).getNodeValue().trim();

            expr = xpath.compile("/root/cond/freq/text()");
            nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            frekvencaPrevodnost = nodes.item(0).getNodeValue().trim();

            expr = xpath.compile("/root/sal/value/text()");
            nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
            slanost = nodes.item(0).getNodeValue().trim();



            System.out.println("Temperatura: " + temperatura);
            System.out.println("Slanost: " + slanost);
            System.out.println("Frek. temp: " + frekvencaTemperatura);
            System.out.println("Frek. prevodnost: " + frekvencaPrevodnost);
            System.out.println("Prevodnost: " + prevodnost);


            //podatke zapišemo v bazo..
            con = getConnection();
            stmt = con.prepareCall("insert into met_meritve (temp,sal,freq_cond,freq_temp,cond,dat_vno) values
(?,?,?,?,?,?)");
            stmt.setBigDecimal(1, new BigDecimal(temperatura));
            stmt.setBigDecimal(2, new BigDecimal(slanost));
            stmt.setBigDecimal(3, new BigDecimal(frekvencaPrevodnost));
            stmt.setBigDecimal(4, new BigDecimal(frekvencaTemperatura));
            stmt.setBigDecimal(5, new BigDecimal(prevodnost));
            stmt.setTimestamp(6, new Timestamp(System.currentTimeMillis()));
            stmt.execute();

            stmt.close();
            con.commit();
            con.close();

        } catch (ParserConfigurationException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (MalformedURLException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SAXException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (XPathExpressionException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NamingException ex) {
            Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
```

```
            if (stmt != null) {
                try {
                    stmt.close();
                } catch (SQLException ex) {
                    Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            if (con != null) {
                try {
                    con.close();
                } catch (SQLException ex) {
                    Logger.getLogger(MerilnaNapravaListener.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }

    }
  }

}
```

Glavnina programa se nahaja v metodi »public void run()«, v kateri zasledimo pet spremenljivk za shranjevanje slanosti, temperature, frekvence sonde SBE-3, frekvence sonde SBE-4 ter izračunano vrednost za prevodnost. Za parsanje XML podatkov iz merilne enote sem uporabil razred »DocumentBuilder« in metodo »parse«, ki vsebuje URL naslov do merilne enote. V primeru uspešne inicializacije parserja z XPath izrazom tako pridobim posamezne parametre, ki jih zapišem v ustrezne spremenljivke (slanost, temperatura, frekvenca sonde SBE-3, frekvenca sonde SBE-4 ter prevodnosti). Po uspešnem parsanju podatkov sledi zapis podatkov v bazo. Za povezavo do baze je uporabljena prosta povezava iz sklada povezav, ki jo pridobim z lokalno metodo »getConnection()«. V primeru uspešne povezave se za vnos podatkov izvede SQL stavek za vnos podatkov v tabelo met_meritve. Po uspešnem vnosu se transakcija zaključi s klicem metode »con.commit()«. Ta sproži zapis podatkov v tabelo »met_meritve«. Sledi sproščanje uporabljene povezave do baze podatkov in zapiranje odprtih tokov.

I.2 Posredovanje podatkov s spletno storitvijo (Web Service)

Izraz spletna storitev (angl. Web Service) se v današnjem času pogosto uporablja, čeprav nima vedno enakega pomena. Najpogosteje označujemo aplikacije, ki so z omrežjem dostopne drugim aplikacijam (Rajnar M., 2014). Za pridobitev in izris podatkov s komponento za izris grafa temperature in slanosti se v spletni aplikaciji uporablja asinhroni klic do spletne storitve. Spletna storitev za vračanje podatkov je spletna aplikacija, ki temelji na REST (angl. Representational State Transfer) arhitekturi. Rezultat spletne storitve pa so pridobljeni podatki

iz baze podatkov v JSON (angl. JavaScript Object Notation) obliki zapisa. Pri pisanju spletne storitve sem uporabil knjižnici Jersey 1.8 in Gson 2.8.0. Knjižnici sta dostopni v repozitoriju javanskih knjižnic (https://mvnrepository.com/, zadnjič obiskano 3. aprila 2018).

Primer spletne storitve je prikazan tukaj:

```
    @GET
    @Path("/tempslajson")
    @Produces(MediaType.APPLICATION_JSON)

    public String getPodatkiJson(@QueryParam("datumOd") String datumOd, @QueryParam("datumDo") String datumDo)
{
        logger.log(Level.INFO, "Podatki service DatumOd; {0}", datumOd);
        logger.log(Level.INFO, "Podatki service DatumDo: {0}", datumDo);
        Connection con = null;
        Statement st;
        ResultSet rs;
        PreparedStatement pstmt;
        String sql;

        ArrayList<PodatkiJson> podatki = new ArrayList<PodatkiJson>();

        try {
            con = this.getConnection();
            st = con.createStatement();
            if (StringUtils.isNullOrEmpty(datumOd) & StringUtils.isNullOrEmpty(datumDo)) {

                sql = "select temp,sal,dat_vno from met_meritve order by dat_vno desc limit 30";
                pstmt = con.prepareCall(sql);
            } else {
                sql = "SELECT * FROM met_meritve WHERE  dat_vno BETWEEN '" + datumOd + "' AND '" + datumDo + "'
order by dat_vno asc";
                System.out.println(sql);
                pstmt = con.prepareCall(sql);

            }

            rs = pstmt.executeQuery(sql);

            while (rs.next()) {
                PodatkiJson pod = new PodatkiJson();
                pod.setDatum(new Date(rs.getTimestamp("dat_vno").getTime()));
                pod.setTemperatura(rs.getDouble("temp"));
                pod.setSlanost(rs.getDouble("sal"));
                podatki.add(pod);
            }
            rs.close();
            st.close();
           pstmt.close();

        } catch (SQLException ex) {
            Logger.getLogger(PodatkiService.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NamingException ex) {
            Logger.getLogger(PodatkiService.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            if (con != null) {
                try {
                    con.close();
                } catch (SQLException ex) {
                    Logger.getLogger(PodatkiService.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
```

```
        }

        Gson gson = new GsonBuilder().setDateFormat("yyyy-MM-dd HH:mm:ss").create();
        return gson.toJson(podatki);

    }
```

Izvorna koda spletne storitve se prične z beležkami (angl. annotations), ki narekujejo lastnost spletne storitve. Beležka/anotacija »@GET« določa, da bo spletna storitev uporabljala GET metodo za dostop s HTTP protokolom. Beležka/anotacija »@Path("/tempslajson")« določa naslov spletne storitve. Beležka/anotacija »@Produces(MediaType.APPLICATION_JSON)« pa določa izhodni tok spletne storitve v formatu JSON. Spletno storitev opisuje metoda »public String getPodatkiJson(@QueryParam("datumOd") String datumOd, @QueryParam("datumDo") String datumDo)«. Metoda sprejme dva parametra, ki sta »datumOd« in »datumDo«. Parametra sta podatkovnega tipa »String« in predstavljata datum in uro v formatu »yyyy-MM-dd HH:mm:ss«. Z njima določimo pogoj iskanja pri SQL poizvedbi. Ker lahko spletna storitev vrne več zapisov, se bodo podatki zapisovali v sklad, ki ga določa razred »ArrayList<PodatkiJson> podatki = new ArrayList<PodatkiJson>();«. Spremenljivka »podatki«, ki je v osnovi objekt tipa PodatkiJson, je namenjena shranjevanju vrednosti temperature, slanosti in datuma. Za pridobivanje podatkov iz podatkovne baze je uporabljena SQL poizvedba. V kolikor sta parametra spletne storitve »datumOd« in »datumDo« nedoločena (angl. null) bo poizvedba iz tabele »met_meritve« vrnila zadnjih 30 zapisov. To določa spodnja SQL poizvedba:

```
select temp,sal,dat_vno from met_meritve order by dat_vno desc limit 30
```

V primeru, ko uporabnik izbere datumsko obdobje, pa bo opravljena SQL poizvedba, ki v »where« pogoju vsebuje datumsko omejitev, katero jo določata spremenljivki »datumOd« in »datumDo«:

```
select * from met_meritve where  dat_vno between '" + datumOd + "' and '"
+ datumDo + "' order by dat_vno asc
```

Polnjenje sklada podatkov »podatki«, je v zank »while(rs.next())«. Zanka se izvaja toliko časa, dokler obstajajo zapisi, ki jih vrača metoda »rs.next()«. Za vsak vrnjeni zapis iz tabele »met_meritve« je ustvarjen nov objekt »pod«, ki vsebuje tri spremenljivke in pripadajoče metode (angl. Methods):

- datum je podatkovnega tipa »java.util.Date« in predstavlja datum in uro zapisa trenutno prebranega podatka iz tabele met_meritve.
- temperatura je podatkovnega tipa »Double« in predstavlja temperaturo v °C.
- slanost je podatkovnega tipa »Double« in predstavlja slanost PSU.

V vsaki iteraciji zanke se razredu »pod« določijo lastnosti (datum, temperatura in slanost). Na koncu zanke sledi dodajanje objekta »pod« skladu »podatki«. To določa sintaksa »podatki.add(pod)«. Zanka »while« se zaključi, ko metoda »rs.next()« ni več resnična (angl. True) oziroma ko smo pregledali vse zapise, ki jih je vrnila SQL poizvedba. Podatki se v JSON obliko pretvorijo z razredom »gson« (com.google.gson.Gson). Ta z metodo »gson.toJson(podatki)« tvori JSON datoteko in jo s HTTP protokolom pošlje spletni aplikaciji ali uporabniku.

Dodatek J- Programska oprema za video zajem in posredovanje žive slike v splet

J.1 Program FFMPEG

Program FFMPEG je prosto dostopen program za multimedijsko obdelavo podatkov in sodi v GNU (General Public License), kar pomeni, da je prosto dostopen vsakomur, ki ga sme tudi spreminjati. Ime FFMPEG pomeni »Fast forward MPEG«. Prevedeni program z navodili se nahaja na spletnem naslovu (https://www.ffmpeg.org/, zadnjič obiskano 12. aprila, 2018), medtem ko se izvorna koda nahaja v javnem repozitoriju Github na naslovu (https://github.com/FFmpeg/FFmpeg, zadnjič obiskano 12. aprila 2018). Programski paket FFMPEG je napisan v C/C++ programskem jeziku in ga je možno prevesti na skoraj vseh operacijskih sistemih (Windows, Linux, Mac ...). Upravljanje programa se odvija v ukazni vrstici ali v terminalu operacijskega sistema.

Programski paket FFMPEG je sestavljen iz programov:

- *Ffmpeg* je ukazni program za pretvorbo video formata v drugi format. Prav tako pa lahko zajema sliko ali video zapis iz TV kartice ali kamere.
- *Ffserver* je pretočni HTTP in RTSP strežnik za predvajanje v živo.
- *Ffplay* je video predvajalnik, katerega osnova so FFMPEG knjižnice.
- *Ffprobe* je program za prikaz informacij o neki vsebini.
- *Libavcodec* je knjižnica, ki vsebuje vse FFMPEG video in avdiokodirnike in dekodirnike.
- *Libavformat* je knjižnica, ki vsebuje multiplekserje in demultiplekserje za avdio in video vsebine.
- *Libavutil* je knjižnica za pomoč.
- *Libpostproc* je knjižnica, ki vsebuje podrutine za video.
- *Libswscale* je knjižnica za slikovno vzorčenje videa.

Podpira sledeče kodeke (kodirnike in dekodirnike):

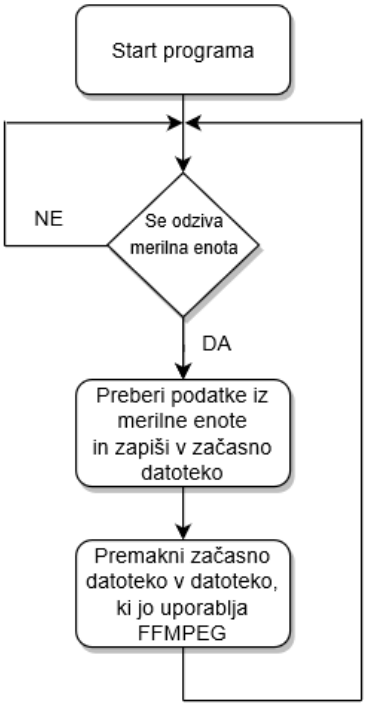
- TU-T videostandard: H.261, H.262 (oziroma MPEG-2 video), H.263, H.263v2 in H.264/MPEG-4 AVC.
- ITU-T vocoder standard: G.711 μ-law, G.711 A-law, G.722.2 (oziroma AMR-WB) in G.726.

- ISO/IEC MPEG videostandard: MPEG-1 Video, MPEG-2 Video (oziroma H.262), MPEG-4 Visual in H.264/MPEG-4 AVC.
- ISO/IEC MPEG avdiostandard: MP2, MP3, AAC in MPEG-4 ALS.
- ISO/IEC/ITU-T JPEG slikovni standard: JPEG in JPEG-LS.
- SMPTE videostandard: VC-1 (oziroma WMV3), VC-3 (oziroma AVID DNxHD) in DPX slike.
- DVD Forum standard: avdiokodeki: MLP in AC-3.
- 3GPP vocoder standard: AMR-NB, AMR-WB (oziroma G.722.2.).
- Windows Media Player videokodeki: Microsoft RLE, Microsoft Video 1, Cinepak, Indeo 2, 3 in 5, Gibajoč JPEG, Microsoft MPEG-4 v1, v2 in v3, WMV1, WMV2 in WMV3.
- Windows Media Player avdiokodeki: WMA, WMA2, WMA Pro in WMA Voice.
- Real Player avdiokodeki: Real Audio 1, 2, 3, 4, 5, 6, 7, 8 in 9.
- QuickTime videokodeki: Cinepak, Gibajoči JPEG in Sorenson 3 kodek.
- Adobe Flash Player video kodeki: Sorenson 3 kodek, VP6 in Flash Screen Video.
- Sony: ATRAC1 in ATRAC3.
- On2: Duck TrueMotion 1, Duck TrueMotion 2, VP3, VP5 in VP6.

Formati zapisa so: ASF, AVI, BFI, IFF, RL2, FLV, MXF, Matroska, Maxis XA, MSN Webcam stream, MPEG transport stream, TXD, OMA in GXF. Protokoli za prenos podatkov so: HTTP, RTP, RTSP, RealMedia RTSP/RDT, TCP, UDP, Gopher, RTMP, RTMPE, RTMPTE, RTMPS in SDP (Repolusk K., str. 38-40 in https://en.wikipedia.org/wiki/FFmpeg, zadnjič obiskano 12. aprila 2018).

J.2 Program FileMover

Program FileMover je samostojni pomožni program, ki prenaša podatke iz merilne enote v tekstovno datoteko. Slednjo uporablja program FFMPEG, iz katere pridobi podatke za prikaz podatkov v enkodiranem video posnetku. Diagram poteka programa FileMover je prikazan na spodnji sliki (Slika 97).



Vir: lasten

Slika 97: Diagram poteka programa »FileMover«

Izvorna koda programa je napisana v programskem jeziku Java:

```
package org.mbp;

import java.io.PrintWriter;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.util.Date;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathFactory;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

public class FileMover {

   private void runIt(String args[]) {

      System.out.println("Začasna datoteka: " + args[0]);
      System.out.println("Datoteka katero prebira FFMPEG: " + args[1]);
      System.out.println("Naslov merilne naprave: " + args[2]);
      System.out.println("====================================");

      Timer timer = new Timer();

      timer.schedule(new TimerTask() {
         public void run() {
            DocumentBuilderFactory dbf;
            DocumentBuilder db;
            Document doc;
            XPathFactory xpathFactory;
            String temperatura;
            String slanost;
            String frekvencaTemperatura;
            String frekvencaSlanost;
            try {
               dbf = DocumentBuilderFactory.newInstance();
               db = dbf.newDocumentBuilder();
               doc = db.parse(new URL(args[1]).openStream());
               xpathFactory = XPathFactory.newInstance();
               XPath xpath = xpathFactory.newXPath();

               XPathExpression expr = xpath.compile("/root/temperature/value/text()");
               NodeList nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
               temperatura = nodes.item(0).getNodeValue().trim();

               expr = xpath.compile("/root/temperature/freq/text()");
               nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
               frekvencaTemperatura = nodes.item(0).getNodeValue().trim();

               expr = xpath.compile("/root/salinity/value/text()");
               nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
               slanost = nodes.item(0).getNodeValue().trim();
               expr = xpath.compile("/root/salinity/freq/text()");
               nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
               frekvencaSlanost = nodes.item(0).getNodeValue().trim();
               PrintWriter writer = new PrintWriter(args[0], "UTF-8");
               writer.println("Temperatura: " + temperatura + " °C");
               writer.println("Slanost: " + slanost + " PSU");
               writer.close();
               Path sourceFile = Paths.get(args[0]);
               Path destinationFile = Paths.get(args[1]);
               Files.move(sourceFile, destinationFile, StandardCopyOption.ATOMIC_MOVE);
               System.out.println("move.." + new Date().toString());
               System.out.println(args[0]);
               System.out.println(args[1]);
               System.out.println("==================");
            } catch (ArrayIndexOutOfBoundsException ex) {
               System.out.println("Preveri število argumentov");
               System.out.println("arg[0] = naziv začasne datoteke npr. c:/video/temppodatki.txt");
               System.out.println("arg[1] = naziv datoteke iz katere bere FFMPEG npr. c:/video/podatki.txt");
               System.out.println("arg[2] = IP naslov merilne enote od koder se pridobi XML response");
               ex.printStackTrace();
               System.exit(1);
            } catch (Exception ex) {
               ex.printStackTrace();
            }
```

```
            }
        }, 0, 500);
    }
    public static void main(String[] args) {
        new FileMover().runIt(args);
    }
}
```

Prevedeni program poženemo v ukazni vrstici. Ta sprejme tri vhodne argumente, ki so:

- Pot in naziv začasne datoteke, v katero program shrani podatke iz merilne enote.
- Pot in naziv datoteke, iz katere program FFMPEG bere podatke.
- IP naslov merilne naprave.

Program se prične v metodi »public static void main(String[] args)«. V metodi inicializiran razred »FileMover« pokliče privatno metodo »runIt(String[] args)«. Metoda »runIt« zažene časovnik, ki se izvaja vsake 0,5 sekunde. Kadar pride do proženja časovnika, se izvede metoda »public void run()«. V metodi je prisotna programska koda, ki se poveže na merilno enoto in iz pridobljenih XML podatkov izlušči izmerjene vrednosti temperature ter slanosti. Pridobljene vrednosti nato program shrani v začasno datoteko in jo po zapisu premakne na lokacijo, kjer podatke bere program FFMPEG. Pri tem uporablja nastavitev »StandardCopyOption.ATOMIC_MOVE«. Ta nastavitev preprečuje, da bi aplikacija zapisala podatke v datoteko, kadar je ta v uporabi. Težava lahko nastane takrat, kadar program FFMPEG želi brati iz datoteke v trenutku, ko v isto datoteko želimo zapisati prebrane podatke. V takšnem primeru se izvajanje programa FFMPEG nepričakovano konča in proces dekodiranja in enkodiranja je prekinjen. Tekstovna datoteka, ki vsebuje podatke, je zelo majhna in vsebuje vsega dve vrstici podatkov, v katerih sta zapisani temperatura in slanost. Podatki v izhodni tabeli so zapisani v formatu UTF-8.

Izjava

Peter Valenčič potrjujem s svojo častjo in s svojim podpisom zagotavljam, da:

- je predloženo magistrsko delo rezultat izključno mojega lastnega raziskovalnega dela;
- sem poskrbel, da so dela in mnenja drugih avtorjev oz. avtoric, ki jih uporabljam v diplomskem delu, navedena oz. citirana v skladu s navodili Fakultete *za pomorstvo in promet* Univerze *v Ljubljani*;
- sem poskrbel, da so vsa dela in mnenja drugih avtorjev oz. avtoric navedena v seznamu virov, ki je sestavni del magistrskega dela in je zapisan v skladu s navodili Fakultete *za pomorstvo in promet* Univerze *v Ljubljani*;
- sem pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti prenesena v magistrsko delo in sem to tudi jasno zapisal v magistrskem delu;
- se zavedam, da je plagiatorstvo kaznivo po zakonu (*Zakon o avtorskih in sorodnih pravicah, Uradni list RS št. 21/95*), prekršek pa podleže tudi ukrepom Fakultete *za pomorstvo in promet* Univerze *v Ljubljani* v skladu z njenimi pravili;
- se zavedam posledic, ki jih dokazano plagiatorstvo lahko predstavlja za predloženo magistrsko delo in za moj status na Fakulteti *za pomorstvo in promet* Univerze *v Ljubljani*;
- je magistrsko delo jezikovno korektno in da je delo jezikovno pregledano.

Portorož, decembra 2018 Peter Valenčič