TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MHK40LT

Lauri Leemet

ANDMEHÕIVESÜSTEEM PURJELAUALE

Bakalaureusetöö

Autor taotleb
tehnikateaduste bakalaureuse
akadeemilist kraadi

Tallinn
2014

AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.
Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.
Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis nooremteadur Ahti Põlder juhendamisel

22. mai 2014 a.
Töö autor
..... allkiri

Töö vastab bakalaureusetööle esitatavatele nõuetele.
22. mai 2014 .a.
Juhendaja
..... allkiri

Lubatud kaitsmisele.
..... õppekava kaitsmiskomisjoni esimees
“.....”.....201… a.

..... allkiri

TTÜ Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

BAKALAUREUSETÖÖ ÜLESANNE

2014 aasta kevadsemester

Üliõpilane: Lauri Leemet 112346
Õppekava: MAHB02/09
Eriala: Mehhatroonika
Juhendaja: Nooremteadur Ahti Põlder
Konsultant: Nooremteadur Henrik Herranen

BAKALAUREUSETÖÖ TEEMA:
(eesti keeles) Andmehõivesüsteem purjelauale
(inglise keeles) Data acquisition system for windsurfing board

Lõputöös lahendatavad ülesanded ja nende täitmise ajakava:

Nr	**Ülesande kirjeldus**	**Täitmise tähtaeg**
1.	**Elektroonikakomponentide (andurite) valik**	**Mai 2013**
2.	**Elektroonikaskeemi projekteerimine ja trükkplaadi projekteerimine**	**Juuni 2013**
3.	**Vee- ja põrutuskindla korpuse projekteerimine**	**Mai 2014**
4.	**Programmeerimine ja testimine**	**Mai 2014**

Lahendatavad insenertehnilised ja majanduslikud probleemid: Luua seade, mis oleks keerulistes oludes vastupidav, saavutaks vajaliku mõõtmistäpsuse ja resolutsiooni, et luua selle põhjal veelgi paremaid uimi. Kasutades seejuures võimalikult odavaid ja kättesaadavaid materjale.

Täiendavad märkused ja nõuded:

Töö keel: Eesti keel

Kaitsmistaotlus esitada hiljemalt: 22.05.2014 **Töö esitamise tähtaeg** 22.05.2014

Üliõpilane Lauri Leemet /allkiri/ …….. Kuupäev: 24.03.2014

Juhendaja Ahti Põlder /allkiri/ ……... Kuupäev: 24.03.2014

Konfidentsiaalsusnõuded ja muud ettevõttepoolsed tingimused formuleeritakse pöördel

SISUKORD

EESSÕNA

Bakalaureusetöö teema pakkus välja Tallinna Tehnikaülikooli nooremteadur Henrik Herranen. Töö koostamine toimus Henrik Herranen-iga ja Z Fins-i töötajatega konsulteerides.

Avaldan tänu Henrik Herranen-ile ja Z Fins-i meeskonnale meeldiva koostöö eest, Mikk Leini-le elektroonikaalaste konsultatsioonide eest ja Ahti Põlder-ile juhendamise eest.

LÜHENDITE JA TÄHISTE LOETELU

ADC – analoog-digitaal muundur

ESD – (Electrostatic discharge) staatiline elekter, mis kahjustab tundlikke elektroonikakomponente

GPS – (Global Positioning System)- Satelliitsidel põhinev üleilmne positsioneerimissüsteem

Halli andur – magnetvälja muutust mõõtev andur (Halli efektil põhinev)

Herkon – hermeetiline kontakt, mida lülitatakse magnetiga

I2C – andmesideliides tuntud ka kui TWI (Two Wire Interface) [1]

NiMH aku – Nickel-Metal Hybrid tehnoloogial põhinev akumulaator nimipingega 1.25 V

LCD – Liquid Crystal Display, vedelkristallekraan

LED – valgusdiood [1]

LiPo aku – Liitium- polümeer tehnoloogial põhinev aku nimipingega 3,7 V

Optoenkooder – andur, milles kasutatakse nurga mõõtmiseks optopaare ja tühimikega ketast nende vahel

Potentsiomeeter – Kolme kontaktiga muuttakisti [1]

Prescaler – taimerite juures kasutatav arv, millega jagatakse taktsagedust, et saada vajalik mõõtepiirkond.

Pull-down – takisti, millega hoitakse viiku madalal pingenivool vältimaks ekslikke lugemeid ujuvast sisendist

Pull-up – takisti, millega hoitakse viiku kõrgel pingenivool vältimaks ekslikke lugemeid ujuvast sisendist

PWM – Pulsilaiusmodulatsioon

SMD – (Surface Mounted Device), pindmontaaž komponent

SPI – andmesideliides (Serial Peripheral Interface) [1]

Tensoandur – deformatsiooniandur, mille takistus muutub, kui seda füüsiliselt mõjutada

UART – universaalne jadaliides [1]

SISSEJUHATUS

Andmehõivesüsteemi projekteerimise vajadus tulenes Eestis tegutseva purjelaua uimi tootva firma Z Fins [2] soovist luua veelgi kiiremaid ja paremaid uimi. Kui siiani on uimede omaduste kohta tagasisidet saadud sõitjatelt tunnetuse ja GPS süsteemist kiiruse andmeid kogudes, siis nüüd tekkis harrastuspurjelauduril Henrik Herranenil idee mõõta ka purjelaua ründe (kohtumis) nurka ja teisi võimalikke karakteristikuid. Selleks projekteeris bakalaureusetöö raames Hr Rauno Lukka kohtumisnurga anduri [3].

Koostöös Z Fins meeskonnaga said andmehõivesüsteemi ülesanneteks:

- Mõõta purjelaua kohtumisnurka
- Mõõta kiirendust kolmes teljes
- Mõõta laua nurka kasutades güroskoopi
- Siduda andmed GPS-ilt saadud positsiooni ja ajaga
- Salvestada andmed SD mälukaardile
- Antud töö autori poolt lisati XBee raadiomooduli valmidus

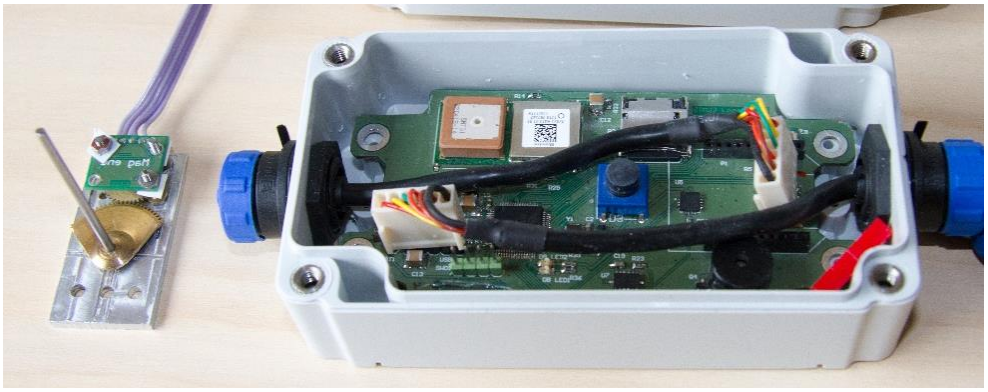
Kuna antud süsteem kinnitatakse purjelauale, siis on sellele esitatavad nõudmised järgmised:

- Korpuse veekindlus IP67 või parem
- Seadme laadimine USB pistikust
- Seadme ja kohtumisnurga anduri elektriline ühendus on veekindel ja lahti võetav
- Aku vastupidavus vähemalt 4 tundi

Pärast nõudmiste väljaselgitamist alustati elektroonika komponentide otsimist. Enamik detaile on valitud Farnelli kataloogist [5]. See on hea koht komponentide ostmiseks kuna tarneaeg on lühike. Nurgaandur koos magnetiga on tellitud näidisena AMS-ist [6]. Magnet on diameetri suunas magnetiseeritud [7]. Tavaliselt on silindrilised magnetid magnetiseeritud telje suunas.

Lisaks anduritele on seadmel ka kõlar ja üks lüliti, et seadet kontrollida ja erinevaid režiime valida. Seadme käivitamiseks on võimalik kasutada ka magnetit, selleks on toiteahelas herkon. Võimalikeks variantideks olid veel seadme käivitamine raputuse peale või muul viisil andurite mõjutamise tulemusena, aga eelistuseks sai see, et süsteem hoiab ise toidet sisse lülitatuna ning kui ta kord välja lülitatakse, siis ükski komponent enam voolu ei tarbi. See on abiks vältimaks aku tühjenemist sõitude vahel ja ka arendustöö käigus. Lihtne oleks aku eemaldada, aga lõpptootes ei ole see võimalik ja alati jääb risk, et korpust koostades ei jää see enam hermeetiline. Lisaks on plaadil ka kolm valgusdioodi, et lihtsustada programmeerimist ja näidata aku laadimise toimimist.

Selleks, et seadme korpuse valmistamiseks liiga palju aega ja vahendeid ei kuluks, otsustati otsida vähemalt IP67 (veekindlusele kuni 1m) (Sele 1) vastav korpus kaubandusvõrgust. Sobivaks osutus Hammondi tootevalikust väliste kinnitusavadega karp [8]. Kuigi Farnelli tootekirjelduses on tegu IP66 karbiga, siis Hammondi kodulehelt võib lugeda vastavuseks IP67 standardit [9]. Üheks valikukriteeriumiks oli korpuse materjal, see ei tohi summutada GPS signaali, seega jäid valikust välja metallist karbid. Lõpptoote karp projekteeritakse kasutades SolidWorks 3D projekteerimistarkvara [10]. Projekteeritud korpus valmistatakse laserpaagutus tehnoloogiat kasutades kahest osast ja liimitakse jäädavalt kokku, et vältida niiskuse korpusesse sattumist läbi liidete.



Sele 0.1. Koostatud andmehõivesüsteem koos anduriga

Programmeerimiskeskkonnaks on valitud Keil uVision [11], tegu on mahupiiranguga versiooniga, mida saab tasuta kasutada. Kasutusel on C keel ja STM32 standard teegid. Programm on jaotatud mitme faili vahel osadeks, et tagada struktuuri ja leida lihtsamalt üles vajalikke funktsioone.

Programmeerimise osa on jagatud etappideks, millest esimene on andmesideliideste konfigureerimine ja testimine. Teiseks osaks on *taskhandler-i* (alamfunktsioonide haldamise funktsiooni) konfigureerimine ja testimine lihtsate funktsioonidega. Kolmandas osas luuakse funktsionaalsus andmete lugemiseks, töötlemiseks ja salvestamiseks. Neljandas osas silutakse programm ja luuakse kalibreerimise võimalused. Antud töös on käsitletud kolme esimest osa.

PÕHIOSA

1 TURUANALÜÜS

Turult võib leida mitmeid tooteid, mis salvestavad GPS andmeid ja ka GPS andmeid koos kiirendusanduriga, enamus neist on mõeldud autosse või on kaasaskantavad. Sellist seadet, mis suudaks salvestada GPS-i, kiirendusanduri, güroskoobi ja välise nurgaanduri andmeid ja oleks veekindel ning piisavalt väike, turul seeriatootmises ei ole. Tellida on võimalik kõike, aga see valmistatakse samamoodi nagu antud bakalaureusetöö teema, üksikkorras vastavalt tellija nõudmistele. Tabelis 1.1 on toodud sarnased tooted:

Tabel 1.1

Toode	Plussid	Miinused
GMOS JI100S [12] Hind: 360 €	Nurga logimine juba sisse ehitatud, samas see andur ei sobi kuna kasutab gravitatsiooni nurga mõõtmiseks	Mootorrattale mõeldud, modifitseeritav, aga pole veekindel ja vajab täiendavaid modifikatsioone
A2 Wireless Data Loggers [13] Hind: 250 €	Logimise osa kasutusvalmis.	Saaks logida nurka läbi potentsiomeetri, pole muid võimalusi
GPS shield [14] Hind: 150 €	Valmis lahendus GPS suhtluseks	Tuleb ise programmeerida, juurde lisada nurga ja kiirenduse mõõtmine

Seega täpselt samade võimalustega toodet hetkel turul ei ole. Et saavutada sama tulemus tuleks koos tööle panna vähemalt kaks seadet ja muuta need veekindlaks. Keeruliseks teeb mitme seadme kasutamise sünkroniseerimise vajadus. Andmehõivesüsteemis on lahendatud see GPS seadme kellaajaga, aga selliste seadmete kasutamine on raskendatud, millel pole kellaaja täpset sünkroniseerimise võimalust.

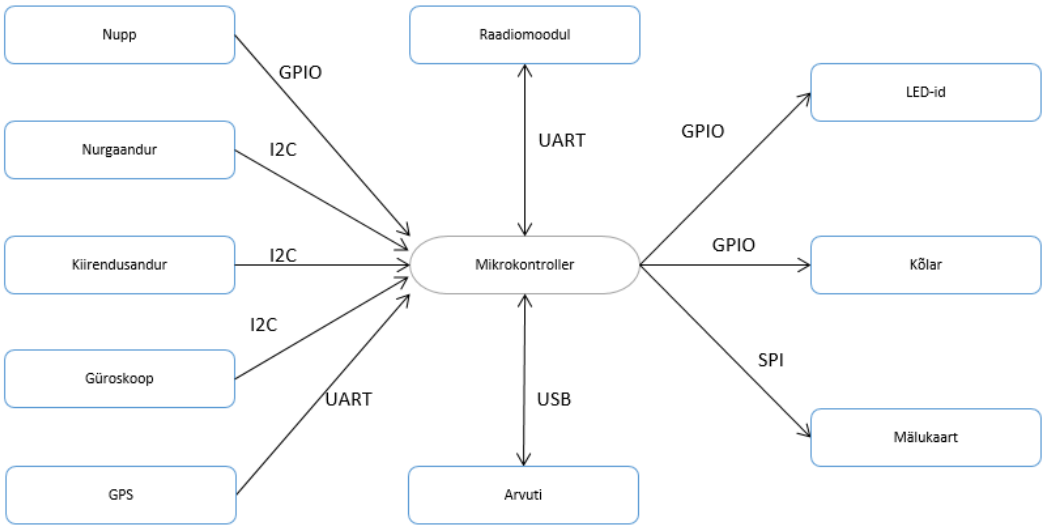
2 ELEKTROONIKA

2.1 Mikrokontroller

Kontrolleriks on valitud STM32L151R6 [15]. STM32 selle pärast, et antud rakenduses, kus tuleb logida nelja andurit korraga ja salvestada andmeid reaalajas, jääb kaheksabitisest mikrokontrollerist väheseks. Antud kontroller tagab piisava arvutusvõimsuse ja kuna seda on TTÜ Robotiklubis kasutatud ka Robotexi jalgpallirobotite [16] peal, siis on selle programmeerimine bakalaureusetöö autorile tuttav. STM32 kontrollerite hulgast valiti L seeria kuna tegu on madala energiatarbega tootega ja 64 viiguga korpuses on piisavalt kommunikatsiooni siinide ühendusi.

STM32 kontrolleriga on hea alustada, kuna ST Microelectronics pakub oma kontrolleritele arendusplaati Discovery [17], milles sisaldub programmaator, üks kallima seeria kontroller ning mitu andurit. Töö autor soetas Discovery F3 komplekti, kus on olemas kiirendusandur koos kompassiga ja güroskoop. See komplekt maksab 12 eurot ja võimaldab kasutada ka silumisvahendeid. Programmeerimiskeskkondi on valikus 4, kõik on saadaval tasuta piirangutega versioonis, osadel on koodi maht piiratud 32 kB-ga.

Antud kontroller võimaldab kasutada mitmeid andmesideliideseid. Nurgaanduri ühendamiseks on kasutuses I2C liides, mis võimaldab andmeid edastada kiirusega 400 kb/s. Nurgaandur toetab kuni 3,4 MHz andmesidet, kontroller seab piiriks 400 kHz. Ülejäänud andmesideliidesed on toodud joonisel Sele 2.1.



Sele 2.1 Andmesideliidesed

2.2 Toiteskeem ja laadimine

Elektriskeem ja trükkplaat on koostatud kasutades programmi Altium Designer [18]. Tegu on professionaalse tarkvaraga mitmekihiliste trükkplaatide projekteerimiseks. Andmehõiveseadme trükkplaat on kahekihiline kuna kahekihilise plaadi projekteerimine on lihtsam, valmistamine odavam ja otsest ruumipuudust ei ole. Trükkplaadi kuju on võetud korpuseks kasutatava karbi mõõtmete järgi. Bakalaureusetöö autor tegi antud kuju joonestamisel vea ja ei arvestanud karbi sees olevaid nurgaraadiusi. Nurkades ühtegi komponenti ei olnud ja viiliga nurkadele raadiusi luues mahtus plaat korpusesse. Plaadil on lühemates külgedes väljalõiked, et mahutada ära karbi külge kinnituvad veekindlad USB pistikud. Üks pistikutest on USB kaabli ja teine nurgaanduri ühendamiseks.

Aku puhul on kaks põhilist valikut, kas NiMH aku, või LiPo aku. Valituks osutus LiPo aku kuna aku kuju sobis ja LiPo laadimine on kiirem. Laadimine vajab küll erilist skeemi, aga selleks on olemas mikroskeem MCP73831 [19], mis keerulisema osa laadimisest ära teeb. Kuna kõik komponendid töötavad 3,3 V pealt, siis 3,7 V LiPo aku on hea valik, kuna siis on pingeregulaatori kasutegur suurem [20]. NiMH aku puhul oleks pidanud kasutama kolme elementi kogupingega $3 \cdot 1{,}25\ \text{V} = 3{,}75\ \text{V}$ [21]. Arvestades aku kuju ja mõõtmeid, on LiPo risttahuka kujulist patja võimalik väiksemasse korpusesse paigutada, kui kolme NiMH aku silindrilist elementi.

Aku mahtuvuse valimiseks leitakse suuremate tarbijate tarbitav võimsus. Suuremad tarbijad on toodud tabelis 2.1:

Tabel 2.1 Ligikaudsed maksimaalsed tarbitavad voolud

GPS	29 - 40mA [22]
Mikrokontroller	7 mA [15]
Nurgaandur	15 mA [23]
Kiirendusandur	110 uA [24]
Güroskoop	6 mA [25]
SD mälukaart	50 - 100 mA
Muud	50 mA

Seega kogu voolutarve on ligikaudu 220 mA

Tarbitava võimsuse valemist $N = U \cdot I$ leiame:

$$N = 3{,}3V \cdot 220 = 726\,mW$$

Pingeregulaatoris läheb kaduma [20] :

$$P_D = (V_{IN} - V_{OUT}) \cdot I_{OUT} + V_{IN} \cdot I_{GND} \quad (2.1)$$

P_D – Võimusus, mis eraldub pingeregulaatorist soojusena

V_{IN} – Sisendpinge

V_{OUT} – väljundpinge

I_{OUT} – väljundvool

I_{GND} – vool GND jalal *(kui* $I_{OUT} \geq 150\,mA, siis\ 3\,mA$ [20])

Valemi 2.1 järgi arvutatakse pingeregulaatorist eralduv võimsus:

$$P_D = (4 - 3{,}3) \cdot 0{,}218 + 4 \cdot 0{,}003 = 0{,}1646\,W$$

Kogu akust tarbitav võimsus on eelmiste summa:

$$N_{kokku} = N + P_D = 0{,}726 + 0{,}1646 = 0{,}8906\,W$$

Leitakse vool aku klemmidel:

$$I_{IN} = \frac{N_{kokku}}{V_{IN}} = \frac{0{,}8906}{4} = 0{,}223\,A$$

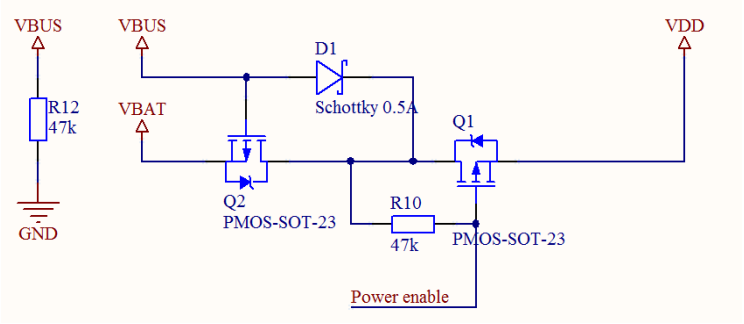
Arvutatakse aku mahtuvus:

$$Mahtuvus = 4 * 0{,}223 = 0{,}892\,Ah$$

Seega piisab 1000 mAh akust. Kui vajatakse pikemat tugiaega, siis saab valida suurema mahtuvusega aku. Näiteks kaheksa tunniseks tööks tuleks leida 2000 mAh aku. Esialgsel hinnangul piisab neljatunnisest tööajast. Kuna arvutustes on kasutatud maksimaalseid voolutarbeid, siis saab oletada, et reaalne tööaeg on pikem kui neli tundi.

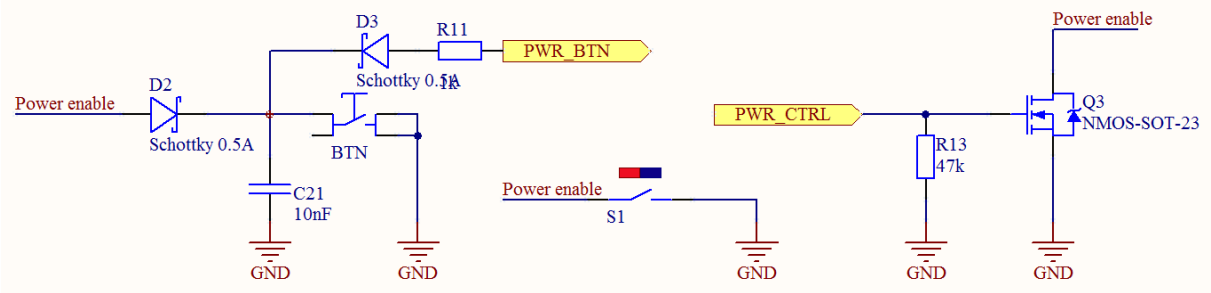
Toiteskeemi koostamisel on eeskuju võetud TTÜ Robotiklubis eelnevalt valmistatud trükkplaatidelt ja skeemis leiduvate komponentide andmelehtetes toodud näiteskeemidest ([15], [20], [26]). Toiteskeemi projekteerimisel on arvestatud seda, et kuna tegu on veekindla

seadmega, siis tuleb andmehõivesüsteemile ainult üks nupp. Lisaks sellele, et antud nupuga saab seadet käivitada saab seda ka teha läbi korpuse magnetiga. Täielikud elektriskeemid on esitatud lisades (LISA 3).



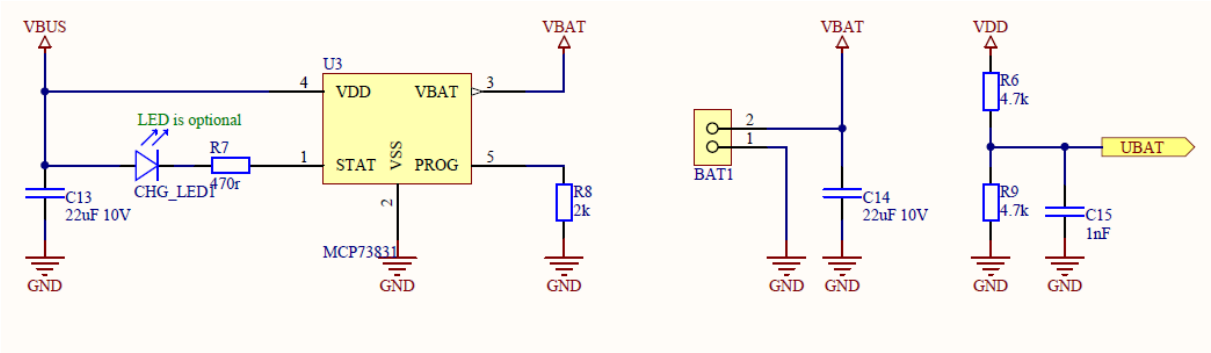
Sele 2.2 Toiteskeem

Skeemil (Sele 2.2) on näha, et USB toitepinge olemasolul ühendatakse aku toiteahelast lahti. Väljatransistoriga Q2 juhitakse aku lülitust. Dioodiga D1 välditakse USB ahelasse akupinge jõudmist. Väljatransistoriga Q1 juhitakse toitepinget. *Power enable* on ühendatud nii lüliti kui ka herkoniga. Antud skeemiosa eeskujuks on TTÜ Robotiklubi universaalse robotipuldi skeem [27].



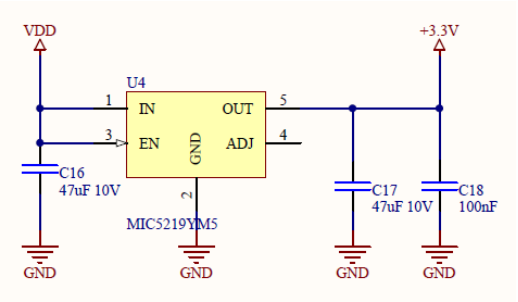
Sele 2.3 Toite lülitamine

Toite lülitamise skeemile (Sele 2.3) on lisatud magnetiga lülitamine. Ka selle skeemi loomisel on eeskuju võetud TTÜ Robotiklubi pultide skeemist [27]. Kui toide on juba mikroprotsessorini jõudnud, siis esimesena asjana tuleb *PWR_CTRL* viik lülitada kõrgele pingenivoole, et läbi mosfeti Q3 hakata mikrokontrolleri toiteahelat suletuna hoidma. Kui seda mitte teha, avaneb ahel magneti eemaldamisel või lüliti vabastamisel. Takistid R10 ja R13 on vastavalt *pull-up* ja *pull-down* takistid, et lülitamata olekus tagada väljatransistorite mittejuhtiv (suletud) olek.

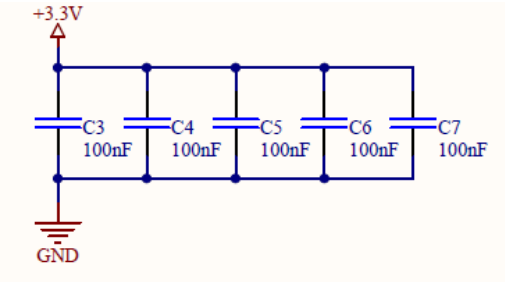


Sele 2.4 Aku laadimine

Kasutusel on LiPo aku, mis vajab laadimiseks eraldi ahelat. Laadimisahelas (Sele 2.4), on mikroskeem U3 [19], mis vastutab laadimisvoolu ja aku täis laadimise eest. Antud mikroskeemi andmelehest on valitud ka tüüpskeem ja takisti R8 väärtus. Kui akut laetakse, siis valgusdiood *CHG_LED1* põleb. Aku pinge mõõtmiseks on skeemiosa, mis kujutab endast pingejagurit takistite R6 ja R9 vahel. Kuna LiPo akud on väga tundlikud üleliigse tühjenemise suhtes, siis mõõdetakse kontrolleriga aku pinget ja katkestatakse toide kui pinge langeb alla kriitilise piiri, mis on antud aku puhul 2.8V [28].



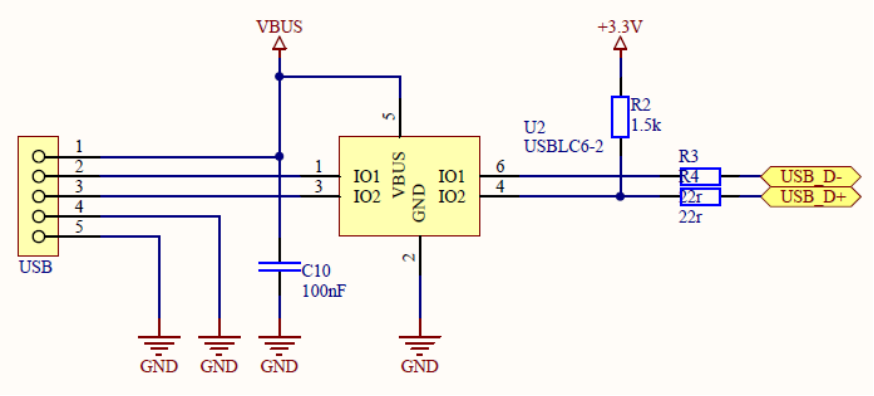
Sele 2.5 Pingeregulaator



Sele 2.6 Toite stabiliseerimise kondensaatorid

Süsteemis on pingeregulaator U4 [20], mille väljundis on 3,3 V. Regulaatori ette ja järele on paigutatud pinge stabiliseerimiseks kondensaatorid (Sele 2.6). Igale mikrokontrolleri toiteviigule on võimalikult lähedale lisatud 100 nF kondensaator, et kõrvaldada häireid

toitepingest, mis tekivad tarbijate lülitusel esinevast pingelangust. Nende kondensaatorite soovituslikud mahtuvused on antud pingeregulaatori andmelehes [20] ja mikrokontrolleri andmelehes [15].USB kaabel ühendatakse kontrolleri külge läbi ESD kaitse U2, mis aitab säästa kontrollerit staatilise elektri eest. USB andmerajad peavad olema ühepikkused ja takistitega R2, R3 ja R4 [29]. USB kaabel on veekindla ühendusega, et seda saaks kasutada märjas keskkonnas. USB ühendus on toiteahela osa kuna sellest laetakse akut, selleks on tehtud ühendused USB pistiku GND ja üldise GND vahel ning 5V toide jõuab aku laadimise skeemi läbi VBUS ühenduse (Sele 2.7).



Sele 2.7 USB ahela kaitse

2.3 Mõõte- ja salvestusseadmed

2.3.1 Nurgaandur

Nurgaandur on antud andmehõivesüsteemi juures üks tähtsamaid. Alguses kaaluti mitmeid erinevaid lahendusi. Üks kõige lihtsamaid lahendusi oleks potentsiomeetriga nurga määramine ja siis ADC-ga potentsiomeetri nurga määramine pingejaguril tekkiva pinge järgi. See idee töösse ei läinud kuna keskkonnaks on soolane merevesi ja potentsiomeeter ise omab liiga suurt takistust hõõrdumise näol. Selleks, et potentsiomeetrit keerata peab olema liiga suur nurgamõõtmise uim. Valituks osutus Halli efektil põhinev magnetenkooder.

Lisaks eelpool mainitud potentsiomeetrile olid valikutes tensoandur, optoenkooder ja halliandur. Tensoandur võiks isegi sobida, aga sai välistatud esialgu liiga kalli hinna ja töö autori kogemuse puudumise tõttu. Optilise enkoodriga on võlli pööramine küll kergem kui potentsiomeetriga, aga probleemiks jääb selle keskkonnast (veest) isoleerimine. Halli anduriga mõõtmisel eelnevaid puuduseid ei ole. Anduri saab valada EPO liimi sisse koos trükkplaadiga

ja magnet ise võib olla vees. Halli andurite põhiline viga on liiga väike resolutsioon. Selleks valisin AS5048B Rotary Sensor 14 bitise resolutsiooniga anduri [30]. AMS pakub I2C ja SPI siinil töötavaid andureid, valisin I2C, kuna antud andmesideliides jäi kontrolleri liideste osas vabaks ja SPI liidest kasutatakse mälukaardi ühendamiseks ning mitut seadet ühele siinile ei ole otstarbekas panna. Kõige parem oleks kasutada CAN, USB või RS422 andmesideliidest kuna diferentsiaalsignaalide häirekindlus on suurem [31].

Lisaks I2C siinile on ühendatud ka PWM väljund, millega saab PWM signaali käidutsüklit mõõtes nurga asendi kätte. Tegu on küll ebatäpsema lahendusega, aga kuna antud anduri I2C väljundi seadistamisega oli probleeme, siis võeti kasutusele PWM väljund. I2C väljund töötas, kuid antud kontrolleriga enam kui 2 sekundit ühendus ei toiminud ja andur vajas toite väljalülitamist, et sellest olekust välja tulla. Bakalaureuse töö autor hindab I2C katkendliku töötamise põhjuseks kas antud anduri viga või puuduva dokumentatsiooni tõttu tehtud programmeerimise viga. PWM väljund töötab laitmatult, seega antud anduriga jääb kasutusse PWM väljund.

2.3.2 GPS- side

Kõigepealt valiti väiksema pindalaga Maestro A2200-A [32] vastuvõtja moodul, aga kuna projekteerimise käigus selgus, et antud seadet enam ei toodeta, siis valiti sama tootja suurema pindalaga mudel Maestro A2035-H. [33] Antud vastuvõtjal ei ole antenn ja elektroonika üksteise kohal, seega eeldatavasti mõjub GPS signaalile vähem häireid ja tulemus on kiirem ja täpsem. Esialgsesse valikusse jäid ka mikrokiibid, mis vajasid lisaks antenni, aga kuna raadiosignaalidega tegelemine vajab kogemust, et toimiv seade projekteerida, siis valiti valmis moodul. GPS moodul ühendatakse kontrolleriga läbi UART liidese.

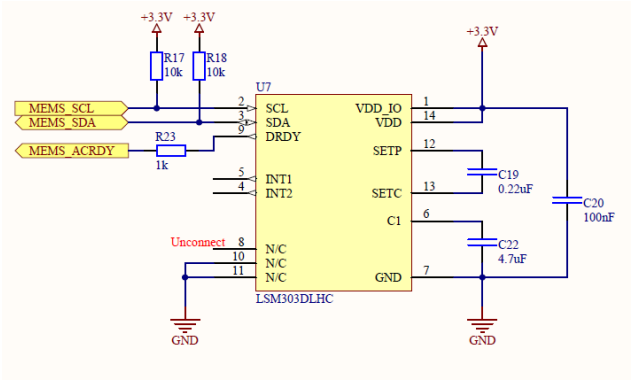
2.3.3 Kiirendusandur ja güroskoop

Kiirendusandur ja magnetomeeter on integreeritud ja töötavad kolmes dimensioonis. Valik tehti arvestades hinda ja kättesaadavust, valituks osutus LSM303DLHC [34] . Antud andur on kontrolleriga ühendatud I2C siiniga ning samale siinile on ühendatud ka güroskoop. Güroskoop L3GD20 [35] on valitud ka Farnelli kataloogist ja on üks odavamaid 3D güroskoope. Olgugi, et tegu on odavama tootega, on antud sensoril väike voolutarve ja suur hulk erifunktsioone, mida antud andmehõivesüsteemi juures vaja ei lähegi. Andurite kalibreerimist antud töös ei käsitleta ja esialgselt ei kasutata ka andurite infot logimiseks. Tegu on lisafunktsionaalsusega,

mida on võimalik projekteerimise käigus lihtsalt lisada ja hiljem programmiliselt kasutusele võtta.

2.3.4 Andurite ühendamine

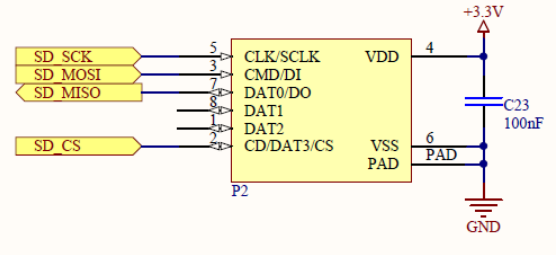
Kui seade on välja valitud, siis tuleb see veel mikrokontrolleriga ära ühendada. See osa on lihtsam, kui seadmel on korralik andmeleht. Kiirendusanduri (Sele 2.8) näitel on tegu I2C andmesiiniga ühendatava seadmega, I2C vajab *pull-up* takisteid (R17, R18) ja kahte signaali: *clock* ja *data*. Lisaks sellele, et andmeside oleks kiirem on antud anduril kasutusel ka *dataready* signaal, et kontrolleri ressurssi paremini ära kasutada, andmeside algatatakse ainult siis, kui andmed on uuenenud. Takisti R23 on skeemis voolu piiramiseks, et vältida anduri või mikrokontrolleri riket juhul kui programmis tehakse viga ja tekib lühis. Kondensaatorite C19, C20 ja C22 väärtused ja ühendusviis on näidatud anduri andmelehes [24].



Sele 2.8 Kiirendusandur

2.3.5 Mälukaart

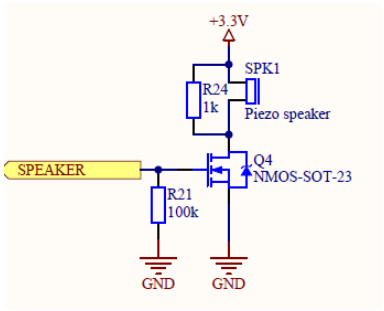
Logitud andmete salvestamiseks kasutatakse süsteemis MicroSD mälukaarti. Kuna antud kontrolleri puhul ei saa kasutada SD kaarti tavapärastes 1-bit ja 4-bit seades. Kasutatakse SPI siini (Sele 2.9) ja kontrolleris peab olema FAT failisüsteemi tugi. SPI siini kiirus paneb salvestuskiirusele ja andmemahule piirangu. Kuna logitav andmehulk piirdub kellaaja ja 3 anduri lugemiga. Mälukaardile on vaja kirjutada andmeid 20x sekundis formaadis: *kell(mmssccc),nurk(nnnn),kiirendus(aaa-aaa-aaa),kompass(kk),güroskoop(ggg-ggg-ggg)*
kus: minut(m), sekund(s) sekundituhandik(c), nurga andur(n) kolm kiirendust(a), kompass(k) ja güroskoobi nurka(g) Kokku umbes 40 tähemärki, ehk 40 baiti, seega 20 · 40=8 kB/s SPI liidese kaudu 100 kB/s kirjutamist on internetis leiduvates materjalides kirjeldatud seega SPI liidese kiirusest piisab.



Sele 2.9 Mälukaart

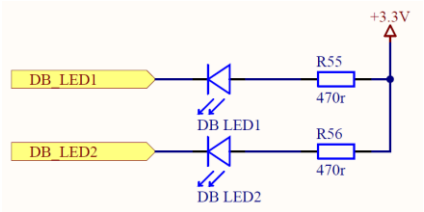
2.4 Väljundid: kõlar ja valgusdioodid

Piesokõlar on valitud võimalikult valju heliväljundiga, et läbi kinnise korpuse oleks võimalik ka seda kuulda. Kõlar hakkab märku andma probleemidest ja toite sisse ja välja lülitamisest. Andmelehes [36] antud helitugevust 85 dB ei saavutata kuna toitepinge on 5 V asemel 3,3 V. Skeemil (Sele 2.10) on kõlari juhtahel. Seda juhitakse väljatransistoriga Q4. Takisti R21 on *pull-down* takisti, et vältida võimalust, kus väljatransistor jääb osaliselt avatuks.



Sele 2.10 Kõlar

Väljundina töötavad ka valgusdioodid, mis on lisatud selleks, et arenduse ajal oleks võimalik visualiseerida programmis toimuvat ja näidata seadme olekut. Skeemil (Sele 2.11) on näha, et valgusdioodid on ühendatud skeemi nii, et neid juhitakse +3,3 V suhtes, mitte maa suhtes. See valik on tehtud sihilik kuna plaadil oli radu lihtsam teha niipidi ja programmiliselt see tähtsust ei oma.



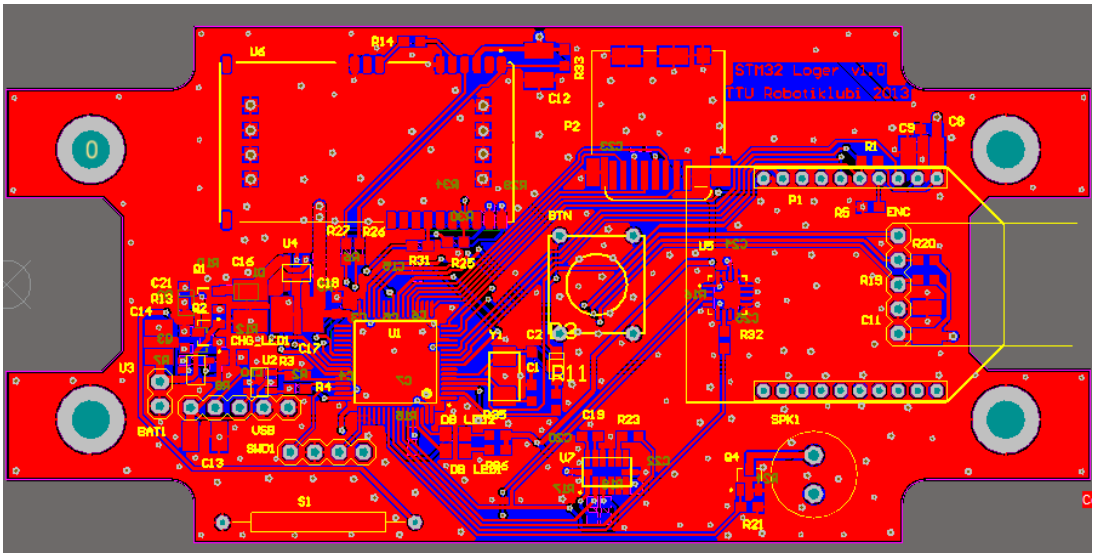
Sele 2.11 Valgusdioodid

2.5 Trükkplaadi projekteerimine

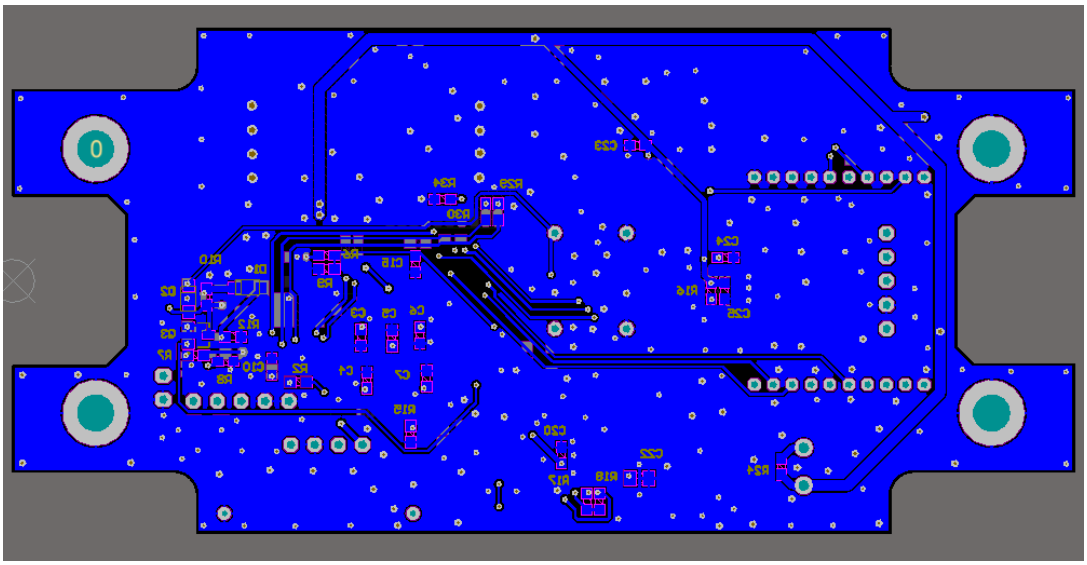
PCB (trükkplaadi) komponentide paigutus ja ühendused on töö autori poolt tehtud. PCB projekteerimisel on arvestatud võimaluste piires suurtest vooludest tulenevaid häireid ja seega on välditud suletud kontuure ning toiteosa on paigutatud võimalikult ühte kohta (Sele 2.13).

Radade laiused on valitud vastavalt seda läbivale voolule. Mikroprotsessori toiteviikudele võimalikult lähedale on paigutatud kondensaatorid, et vältida suuremaid pingekõikumisi, mis tekivad siis, kui mikrokontrollerist lülitada mitu väljundit või andmesidesiini korraga tööle.

Üks viga, mis esile tuli on see, et skeemis on toite lülitamise väljatransistor valepidi ja see tuli joota plaadile selili ja diagonaalis. SMD- komponentide jootmiseks on nende jootesaarekesed tehtud pikemad (Sele 2.12). Siis saab jootekolviga soojendada iga saarekest eraldi.

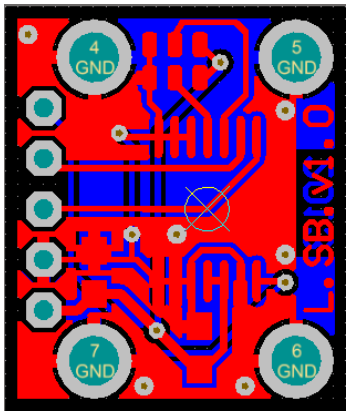


Sele 2.12 Ülemise poole komponentide paigutus



Sele 2.13 Alumise poole komponentide paigutus

Teine, väiksem trükkplaat (Sele 2.14) sisaldab nurgaandurit ja juhtmete ühendust, see kinnitub nurgaanduri külge M2 poltidega. Selle trükkplaadi mõõtmeteks on 16 mm x14 mm. 2x3 saarekest pildi ülemises osas on I2C andmeside aadressi valiku tegemiseks. Vastavalt soovitud aadressile joodetakse saarekesed kokku. Trükkplaadid on tellitud tootjalt Kamitra (QPC Electronics OÜ) [37] . Mõlemad plaadid on 1 mm paksused ja kahepoolsed. Antud trükkplaadi skeem on toodud lisades (LISA 4).



Sele 2.14 Nurgaanduri trükkplaat

3 MEHAANIKA

3.1 Veekindlus

Veekindluse seisukohast tuleb jälgida mitut aspekti. Esiteks peab olema tagatud korpuse enda veekindlus liitest, kust see kahest või enamast elemendist koostatakse. Teiseks tuleb jälgida väliste kommunikatsiooniliideste veepidavust. Ajutise korpuse, Hammondi tootevalikust väliste kinnitusavadega karbi [8], IP67 klass peaks tagama veekindluse kuni 1 m sügavusel. Antud korpust testiti 70 cm sügavuses Biorobootikakeskuse basseinis ja lisaks veel ka 4 kraadises järvevees 2 m sügavusel. Testi tulemuseks olid 3 piiska vett karbi sisemuses kondenseerunud veeauru kuna karp võeti testi soojast autost ja uputati 4 kraadisesse vette. Suured temperatuurivahed tulevad ka purjelauale kinnitatult ette ja kondensaadi kogumiseks tuleb korpusesse paigaldada vett imavat materjali nagu näiteks silikageeli kotike [38].

3.2 Korpuse projekteerimine

Projekteerimise aluseks on valmis oleva trükkplaadi kuju, see peab korpusesse mahtuma ja kahe USB pistiku jaoks kinnituskoht olema. Nupu lahendus peab olema kindaga vajutatav ja veekindel.

Nupu veekindluse saavutamiseks on mitmeid võimalusi, üheks neist oleks osta selline lüliti, mille saab korpusesse installeerida ja see ise on juba veekindel. Tunduvalt kompaktsem lahendus on valmistada painduv korpuse kaas, mida vajutades lülitatakse nuppu, mis on trükkplaadil. See seab piirangud materjalivalikule, see peab olema piisavalt vastupidav painutusele. Olles katsetanud erinevaid materjale, siis tundub, et TTÜ-s kasutatav 3D laserpaagutus meetodil valmistatud polüamiid korpus on piisavalt vastupidav. Selles veendumiseks tuleb teha mitu prototüüpi. Vähendamaks prototüüpide valmistamise vajadust ja korpuse väändumise riski on valitud lahendus, kus korpusesse integreeritud nuppu katab membraan, mis tagab selle nupu veekindluse.

Kõigepealt peab korpuse sisse mahtuma trükkplaat, mida hakkavad kinni hoidma kas kinnituskõrvad või poldid. Selleks, et nupule vajutades lüliti rakenduks on vaja teha korpus piisavalt jäik, et trükkplaat eest ära ei liiguks.

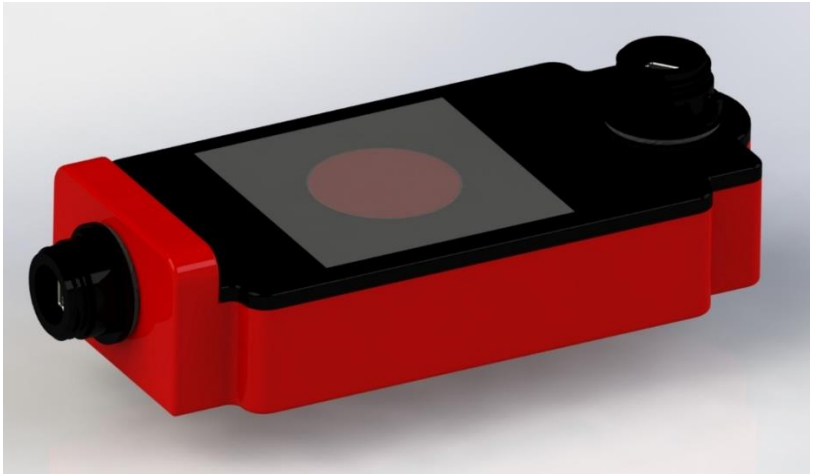
Veekindlad USB pistikud [39] on päris suured, selleks luuakse korpusele otstesse kõrgendused, et oleks võimalik paigaldada trükkplaadi kõrvale pistik. Kuna antud pistikud on odavad ja

kättesaadavad on kasutatud enkoodri pistikuks sama USB pistikut. Valmislahendusel tuleb see pistik kindlasti asendada erineva pistikuga, sest kui enkoodri pistikusse ühendada USB kaabel arvutist, siis mõlemad suure tõenäosusega hävivad kuna antud pistikus ei kasutata USB standardit. Üks pistikutest on USB koos laadimise ja andmesidega, see kasutab USB standardis ettenähtud pingenivoosid ja signaale.

Korpuse alumisel küljel (Sele 3.1) on aas, et seda saaks rihma või takjapaelaga kinnitada purjelaua külge. Lisaks aasale on korpuses ka kruviaugud, millega saab antud korpuse külge kinnitada rakiseid, kui selleks peaks vajadus ilmnema.

Sele 3.1 Korpus

Nuppu katab membraan (Sele 3.2), mis on valmistatud paksemast kilest ja liimitud korpuse külge. Nupp liigub juhtpuksil ja selle liikumisulatuseks on projekteeritud 2mm.



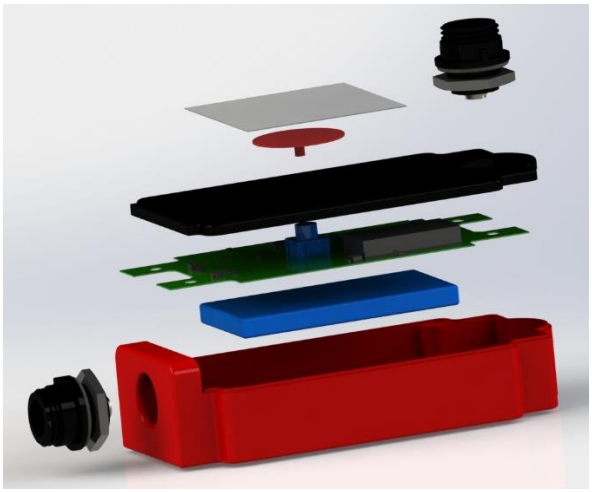
Sele 3.2 Nupu asetus

Korpuse gabariitmõõtmed (LISA 1) tulenevad eelnevalt valmistatud trükkplaadi mõõtmetest ja valitud pistikute kinnitusnõuetest. Kuna antud pistikutele peab kinnitusava olema 17mm ja mutter pistiku kinnitamiseks peab mahtuma korpusesse, siis on korpuse ühte otsa loodud kõrgendus pistiku mahutamiseks.

Korpuse koostamiseks kasutatakse liimi ja kruvisid. Liimiga liimitakse korpuse kaas korpuse külge. Selleks on korpusel rant ja kaanel pesa sellele. Liimi jaoks on projekteeritud pragu nende vahele (Sele 3.3).

Sele 3.3 Serv liimimiseks

Kruvidega kinnitatakse trükkplaat korpuse külge. Koostamise järjekord on järgmine (Sele 3.4): USB pistikud kinnitatakse korpuse ja selle kaane külge. Siis asetatakse aku ja trükkplaat korpusesse ja kinnitatakse. Järgmiseks ühendatakse pistikud ja liimitakse korpus kokku. Seejärel paigaldatakse nupp pessa ja liimitakse membraan selle peale.



Sele 3.4 Lahtivõetud koost

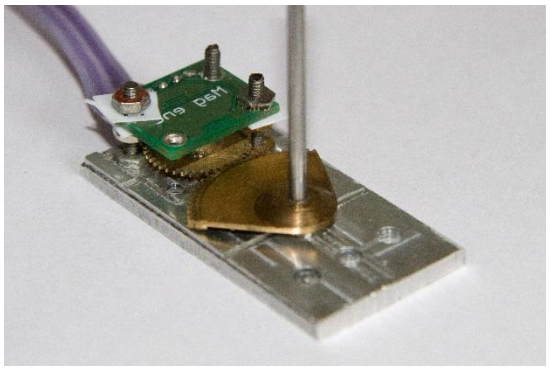
3.3 Korpuse materjal ja valmistamine

Korpuse materjaliks on tootmise tehnoloogiast tulenevalt peenpolüamiid. Materjal on värvuselt valge ja tõmbetugevuseks 45 MPa [40]. Korpus värvitakse pärast laserpaagutuses valmistamist, et sulgeda detaili poorid vähendamaks määrdumist ja suurendamaks veekindlust.

Valmistustehnoloogia valik tuleneb korpuse keerukusest ja sellest, et antud toodet toodetakse ainult üksikkorras. Kui tootmine oleks plaanis suuremates kogustes, siis peaks korpuse valmistamistehnoloogiaks valima pigem survevalu ja ka vastavalt tehnoloogia võimalustele projekteerima korpuse. Antud korpuse disain ei võimalda seda valmistada muul viisil kui 3D printimise teel. Tavapärane 3D printer (printer kus plastikust traati sulatatakse) sobib korpuse valmistamiseks ka, kuid laserpaagutus annab täpsema ja veekindla tulemuse, mida tavapärase printeriga on keerukas saavutada detaili erikuju pärast ning detail vajab pärast printimist järeltöötlust, et eemaldada tugistruktuurid.

3.4 Nurgaanduri mehaanika

Nurgaanduri mehaaniline osa on antud töö autoriga konsulteerides projekteeritud ja valmistatud Hr Henrik Herraneni poolt. Prototüübi korpus on välja freesitud alumiiniumist ja selle sisse on puuritud avad laagrite jaoks. Magnet liimitakse väikese hammasratta peale, mis pärineb vedrukellast ja pöörab laagril. Selle peale kinnitatakse M2 poltidele bakalaureusetöö käigus valmistatud anduri trükkplaat. Prototüüp on nähtav järgneval pildil (Sele 3.5).



Sele 3.5 Nurgaandur

4 PROGRAMMERIMINE

4.1 Programmi ülesehitus

Kõige esimene asi uue trükkplaadi puhul on kontrollida, kas jootmine on olnud edukas ja mikrokontrollerit üldse programmeerida saab. Siis saab juba LED-e vilgutada ja kõlarit proovida. Edasine on funktsioonide kirjutamine kõigile ühendatud seadmetele.

Kõige kiiremini peab töötama nurgaanduri vääruse lugemine ning kiirendusanduri ja güroskoobi väärtuste filtreerimine ja arvutamine. Ülejäänud funktsioonid võivad toimuda aeglasemalt või sootuks taustal. Põhiprogrammi (*main* funktsiooni) kirjutamiseks on võetud eeskuju TTÜ Robotiklubis projekteeritud robotite juhtimispultidest [27]. Nimelt on seal ka vaja täita reaalaja ülesandeid ja kirjutada LCD-le teksti. LCD-le kirjutamine on niivõrd aeglane protsess, et kui seda teha iga kord, kui pultide raadioliidest kasutada, siis raadioside oleks väga aeglane. Sama loogikat on võimalik ära kasutada andmehõivesüsteemi puhul.

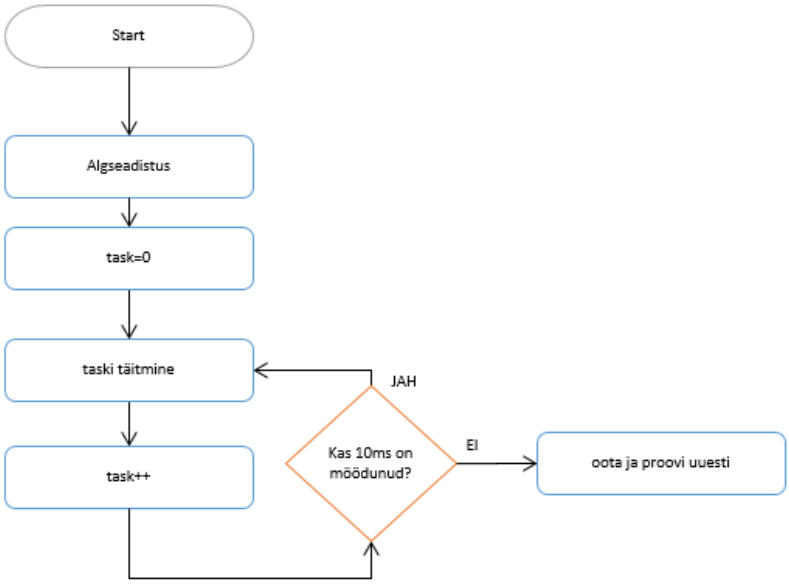
SD mälukaardile kirjutamine, GPS-i lugemine ei ole nii ajakriitilised, kui nurga- ja kiirendusanduri lugemine. GPS positsiooni leidmine võib tunduda kriitiline, aga esialgses variandis loetakse GPS signaalist välja ainult kellaaeg, ja hiljem sünkroniseeritakse teise GPS seadme andmetega andmehõivesüsteemi logi. Nimelt on loodavas süsteemis olev GPS ebatäpsem ja andmeid uuendatakse üks kord sekundis. Ühe sekundi jooksul jõuab laud liikuda 7 m - 8 m, aga sünkroniseerimiseks sellest piisab.

Põhiprogrammis (LISA 2) jagatakse funktsioonid kaheks: reaalaja funktsioonid ja funktsioonid, mis pole ajakriitilised. Reaalajafunktsioonid peavad olema võimalikult lühikesed ja ei tohi sisaldada liialt pikka osa, kus katkestused on keelatud. Neid funktsioone kutsutakse välja kordamööda iga 10 ms tagant. Ülejäänud funktsioonid täidetakse 100 ms tagant või kasutades katkestusi (andmevahetus mälukaardiga ja muud andmesiinid, mis võtavad infot vastu ja kasutavad katkestusi, mis tulenevad välistest muutujatest või taimerist).

Bakalaureusetöös käsitletud failid on üks osa programmist ülejäänud failid sarnanevad ülesehituselt käsitletutele ning ei ole mahukuse tõttu esitatud. Kogu programmist annab ülevaate LISA 2. Järgnevates peatükkides on toodud välja olulisemad funktsioonid.

4.2 Funktsioonid

4.2.1 Peaprogramm main.c



Sele 4.1 Põhiprogramm

Main.c fail (LISA 5) koosneb mitmest osast ja põhineb TTÜ Robotiklubis kasutatavate pultide *main* funktsioonil. Lisatud on vajalikud *Task*-id ja muudetud sobivaks perioodide pikkused. Kõik teekidest kasutatud funktsioonid on konfigureeritud töö autori poolt. Kasutatud on STM32L1xx standard teeki [41], kus on mikrokontrolleri perifeeria seadistamise funktsioonid.

Esimene osa on ülesannete (funktsioonide) loend *(task list*), kus on ära toodud kõik ülesanded, mida tuleb täita. Igal ülesandel on kaks osa: *init* ja *task*. *Init* ehk algseadistamise funktsioon täidetakse esimesena ja üks kord, edasi täidetakse ainult *task* funktsioone. Nagu eelpool mainitud on ülesanded jagatud kaheks: ajakriitilised ehk *realtime task* funktsioonid ja mitte ajakriitilised ehk *background task* funktsioonid. Järgnevalt on toodud väljavõte *main* funktsioonist, kus on näha eelpool mainitud *task list.*

```
static const Task RT_TASK_LIST[] =
{
  { Power_Init,   Power_Task    },
  { LED_Init,     LED_Task      },
  { Button_Init,  Button_Task   },
  { Speaker_Init, Speaker_Task  },
  { Angle_Init,   Angle_Task    },
  { XBEE_Init,     XBEE_Task    },
};
```

```
static const Task BG_TASK_LIST[] =
{
  { Logic_Init,   Logic_Task    }
  { SDCard_Init, SDCard_Task    },
  { GPS_Init,     GPS_Task      },
  { Compass_Init, Compass_Task  }
};
```

Nendesse nimekirjadesse lisatakse kõik täidetavad funktsioonid. Reaalaja funktsioonide ja taustafunktsioonide täitmistiheduse saab määrata eraldi. Esialgselt on määratud Reaalajafunktsioonide ajaks 10 ms ja taustafunktsioonide täitmisajaks 100 ms.

Main funktsioonis tehakse ära enne lõputusse *while* tsüklisse minemata 1 ms katkestuste seadistus, prioriteetide seadmine ja mõlema listi ülesannete algseadistamine. Pärast algseadistuse tegemist hakatakse lõputus *while* tsüklis käitama kõiki *BG_tasklist* –is olnud funktsioone. Väljavõte *main* funktsioonist:

```
int main(void)
{
  uint32_t task;
  uint32_t period;
  /* Seadistatakse SysTick katkestuse 1 ms jaoks */
  RCC_GetClocksFreq(&RCC_Clocks);
  SysTick_Config(RCC_Clocks.HCLK_Frequency / 1000);

  /* Lubatakse SYSCFG kell */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

  /* Konfigureeritakse prioriteetide grupp 2/2 seadistusele */
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

  /* Algseadistatakse reaalaja funktsioonid */
  for (task = 0; task < RT_TASK_COUNT; task++)
  {
    RT_TASK_LIST[task].InitFunction();
  }
  /* Algseadistatakse tausta funktsioonid  */
  for (task = 0; task < BG_TASK_COUNT; task++)
  {
    BG_TASK_LIST[task].InitFunction();
  }
```

Selleks, et süsteem töötaks reaalajas st takt-taktis, on süsteemi lisatud võimalus mõõta ülesandele kulunud aega. Kuna aja mõõtmist ja perioodi lugemist võib segada katkestustest tulenev viide ja katkestustes toimuv tegevus, siis ajakriitiliste operatsioonide korral keelatakse katkestused käsuga *__disable_irq();* Kui katkestused on keelatud kopeeritakse perioodi pikkuse väärtus mällu ja nullitakse *BGPeriodCounter*. Seejärel täidetakse taustafunktsioonide *listist* järgmine funktsioon ja oodatakse järgmist katkestust. Funktsiooni osa on lahendatud järgmiselt:

```
  while (1)
  {
    /* Periood salvestatakse ja nullitakse */
    __disable_irq();
    period = BGPeriodCounter;
    BGPeriodCounter = 0;
    __enable_irq();
    /* Täidetakse taustafunktsioonid */
    for (task = 0; task < BG_TASK_COUNT; task++)
    {
      BG_TASK_LIST[task].TaskFunction(period);
    }

    /* Oodatakse taustafunktsiooni perioodi */
  while (BGPeriodCounter < BG_TASK_PERIOD) {}
  }
  }
```

Reaalajafunktsioonidega tegeleb *SysTick_Handler*, mis kutsub iga 10 ms järel välja järgmise funktsiooni *RealTime Task-ide* nimekirjast. Siin ka suurendatakse *BGPeriodCounter*-it, mida eelnevalt kirjeldatud *main* funktsioonis nulliti:

```
void SysTick_Handler(void)
{
  uint32_t task;

  /* Reaalaja funktsioonid käitatakse iga RT_TASK_PERIOD-i järel */
  if (++RTPeriodCounter >= RT_TASK_PERIOD)
  {
    RTPeriodCounter = 0;
    for (task = 0; task < RT_TASK_COUNT; task++)
    {
      RT_TASK_LIST[task].TaskFunction(RT_TASK_PERIOD);
    }
  }
  /* Taustafunktsioonide perioodi loetakse 1 ms kaupa */
  BGPeriodCounter++;
}
```

4.2.2 Valgusdioodide funktsioonid led.c

Led.c failis (LISA 6) paiknevad LED-ide juhtimiseks vajalikud funktsioonid. Esimese asjana tehakse ära algseadistamine, selleks määratakse ära viikude funktsioonid. Antud juhul kasutatakse viikusid väljundina ja konfigureeritakse ilma *push-pull* režiimita. Kuna LED-i juhtimiseks on vaja antud skeemi puhul viik madalale pingenivoole kommuteerida.

Järgmisena on ära toodud funktsioon, mida täidetakse iga 6 x 10 ms tagant, või olenevalt funktsioonidele määratud väljakutsumise tihedusest ja ka funktsioonide arvust *task_list*-is. Siin

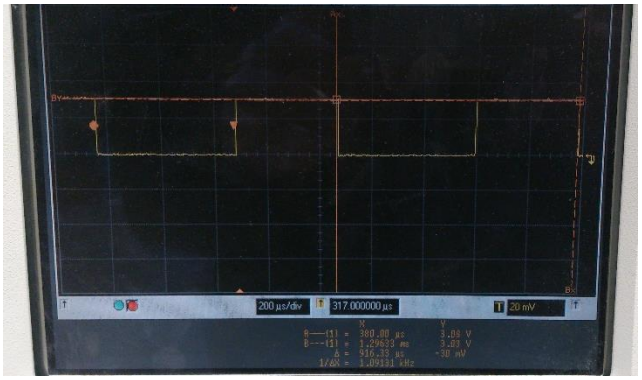
vilgutatakse oleku *LED-i* selleks, et aru saada, kas programm töötab korrektselt. Antud funktsioon on väljatoodud järgnevalt:

```
void LED_Task(uint32_t Period)
{
	if(Set)
	{
		GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_SET);
		Set=0;
	}
	else
	{
		GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_RESET);
		Set=1;
	}
	Cycles=0;
}
```

Lisaks on loodud LED-ide juhtimiseks otsesed funktsioonid `LED_Clear(int i)` ja `LED_Set(int i)` , mis ei sõltu *tasklist*-ist. Algseadistamine peab olema tehtud, kuid nende funktsioonide kasutamisel ei pea ootama *LED_Taski* väljakutsumist.

4.2.3 Nurgaanduri lugemine pwmread.c

Nurgaanduri lugemiseks prooviti kõigepealt kasutada I2C andmesiini. I2C andmeside küll töötas, aga iga paari sekundi tagant vajas taaskäivitamist läbi anduri väljalülitamise. Antud olukord ei sobinud konkreetse ülesande lahendamiseks kuna lugemeid soovitakse teha võimalikult kiiresti. See viga on andurist tulenev ja antud anduri puhul kasutatakse edaspidi lugemiseks anduri PWM väljundit. Andurist väljuv signaal on ilma oluliste häiringuteta PWM signaal perioodiga 916 µs (Sele 4.2). Antud väärtus on mõõdetud ostsilloskoobiga anduri keskmises asendis. Vaja on mõõta PWM pulsilaiust (*duty cycle*), mis muutub proportsioonis magneti nurgaga anduri suhtes.



Sele 4.2 Nurgaanduri PWM väljund

PWM signaali pulsilaiuse mõõtmiseks on STM32 mikrokontrolleril eraldi taimeri funktsioon. Antud funktsiooni kohta on dokumentatsioon kasutamaks seda taimeri esimesel kahel kanalil. Kuna sobiv PWM sisendviik võimaldab kasutada ainult kolmandat kanalit ja selle kanaliga antud funktsionaalsuse kohta dokumentatsioon puudub ning katsetused tulemust ei andnud, siis kasutatakse antud programmis katkestuse peale tööle hakkavat taimerit. Sisendi ja taimeri seadistamine on tehtud funktsioonis *PWMRead_Init()* (LISA 7). Taimeri *prescaler*-iks on valitud 0 kuna antud valik tagab taimeri lugemiks väärtused 0-2000 takti. Kui valida suurem *prescaler*, siis täpsus kannatab kuna vahemik on siis vastav arv kordi väikse.

Funktsioonis *EXTI2_IRQHandler()* täidetakse katkestuse saabudes järgmist ülesannet: Kui tegu on tõusva frondiga, siis nullitakse taimer ja aktiveeritakse. Kui nüüd jõutakse jälle katkestuseni, siis on tegu langeva frondiga ja võetakse lugem, milleks on PWM signaali pulsilaius taimeri taktides. Seejärel pannakse taimer seisma ning kustutatakse mõlemal juhul katkestuse väljakutsumise bitt registrist. Programmikoodi väljavõte (LISA 7):

```
void EXTI2_IRQHandler(void)
{
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_2) == Bit_SET)
    {
        TIM_SetCounter(TIM3, 0);
        TIM_Cmd(TIM3, ENABLE);
    }
    else
    {
        IC2Value = TIM_GetCounter(TIM3);
        TIM_Cmd(TIM3, DISABLE);
    }

    /* Clear the EXTI line 8 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line2);
}
```

5 EELARVE JA OHUTUS

Andmehõivesüsteemi arenduse eelarve on toodud tabelis 5.1. Antud tabel sisaldab ainult kulutusi elektroonikale ja ajutisele korpusele. Kulunud töötundide arv on umbes 300 ja 3D prinditud korpuse hinnaks kujuneb suurusjärk 60€ ja kuna nurgaandur on saadud tootjalt näidisena, siis selle hinda ka tabel ei kajasta. Arendustöö maksumus on hinnanguline.

Tabel 5.1 Eelarve

Akud ja juhtmed	10 €
Elektroonika	62,35 €
Ajutine korpus ja pistikud	56,39 €
Trükkplaadid	22 €
Prinditud korpus	≈60 €
Arendustöö	≈4000 €
KOKKU	≈4210,74 €

Ohutuse seisukohast tuleb silmas pidada asjaolu, et iga kõrvaline objekt purjelaual segab sõitmist ja mõjutab laua tasakaalu. Testisõitjaid informeeritakse riskidest, mis võivad tekkida ja instrueeritakse süsteemi kasutama.

Elektriliselt on seade isoleeritud ja kasutatavad pinged ei kujuta ohtu inimesele. Kõige suurem oht on end vigastada seadme peale kukkudes või astudes. Vigastuste vältimiseks on seadme servad projekteeritud raadiustega ja nurgad ümardatud.

KOKKUVÕTE

Bakalaureusetöö eesmärgiks oli valmistada veekindel andmehõivesüsteem purjelauale kinnitamiseks ja erinevate parameetrite salvestamiseks sõidu ajal. Töö tellijaks on maailmatasemel sõitjatele purjelaua uimi tootev ettevõte, kelle eesmärgiks on luua seadmelt saadud infole tuginedes uusi uimemudeleid. Töö jaguneb kolme ossa: trükkplaadi projekteerimine ja koostamine, mehaanika projekteerimine ja programmi koostamine.

Elektroonika projekteerimisel kasutatakse Altium Designer tarkvara ja luuakse vajalikud ühendused anduritelt andmete kogumiseks ja nende salvestamiseks. Kasutusel on STM32L151R6 mikrokontroller, Halli efektil põhinev nurgaandur, millega mõõdetakse laua ründenurka, GPS vastuvõtja, mille vastuvõetud kellaaja abil sünkroniseeritakse andmehõivesüsteem teiste salvestusseadmetega. Kiirendusanduri ja güroskoobi andmeid kasutatakse jälgimaks laua asendit. Kõik andmed salvestatakse microSD mälukaardile. Süsteemi toiteallikaks on LiPo aku, aku laadimine toimub läbi USB pistiku. Süsteemi saab sisse lülitada kas nupust või magnetiga ja andmehõivesüsteem ei tarbi väljalülitatud olekus voolu. Kui akupinge langeb alla kriitilise piiri lülitatakse seade välja.

Seade on projekteeritud kasutades 3D projekteerimistarkvara SolidWorks. Seadme korpus on projekteeritud kahest osast, kuna antud seadme juures on veekindlus tähtsal kohal, siis seadme korpus koostatakse liimimise teel. Korpuse kaane sisse on projekteeritud nupp, mida katab liimitud membraan. Korpuse küljes on aas ja kruviavad kinnitamiseks. Ühes korpuse otsas on pistik nurgaanduri ühendamiseks ja kaanel USB pistik seadme laadimiseks ja andmete ülekandmiseks. Seadme korpus valmistatakse laserpaagutuse meetodil polüamiidist. Prototüübi valmistamise lihtsustamiseks on seadmele valitud kaubandusvõrgust ajutine IP67 korpus, mis on koostatud kruvidega, et arendusjärgus toodet testides saaks seadme programmi uuendada ja kasutada silumisvahendeid.

Mikrokontrolleri programmeerimiseks kasutatakse Keil uVision tarkavara, mis on tasuta versioonis mahupiiranguga. Programm on kirjutatud C keeles ja programmeerimise lihtsustamiseks kasutatakse kontrolleri tootja poolt loodud standard teeke, mis sisaldavad funktsioone andmesideliideste konfigureerimise hõlbustamiseks. Programm on jaotatud mitme faili vahel osadeks ja alamfunktsioonideks. Programmis kasutatakse kaht erineva sageduse ja prioriteediga funktsioonide kogumit. Funktsioone, mis on ajakriitilised täidetakse 10 ms järel ja ülejäänud funktsioonid töötavad 100 ms perioodiga.

Bakalaureusetöö tulemusena valmis prototüüp, mida hakatakse testima töökeskkonnas. Pärast katsetuste tegemist selguvad arendamise vajadused ja vajalik funktsionaalsus. Põhiline funktsionaalsus nurgaanduri salvestamiseks on tagatud ning elektroonika projekteerimise käigus loodud võimalus ka laiendada seadme otstarvet raadiomoodulit kasutades. Edasiseks tööks on vastavalt katsetel selguvatele nõudmistele vajaliku kalibreerimismeetodi väljatöötamine.

Bakalaureusetöö valmimise käigus valmistasid probleeme kõige rohkem seadmete puudulikud andmelehed, kuid sellele vaatamata on õnnestunud vajalik funktsionaalsus saavutada, kuigi selleks kulus rohkem aega. Autori seisukohast on antud teemavalik olnud silmaringi arendav, kuna kasutusel on mitmeid erinevaid sensoreid ja andmesideliideseid. Bakalaureusetöö tulemusena valminud prototüüp täidab suures osas ülesande nõudmisi.

SUMMARY

The goal of this thesis was to design a data acquisition system for windsurfing board. This project is a part of research for a private company building fins for windsurfing board. The thesis project consists of three main topics: electronic circuit and PCB design, mechanical design and programming.

An electronics design software, Altium Designer is used for designing and creating the necessary connections from sensors to collect data and to save it to the SD card. STM32L151R6 microcontroller is used along with a magnetic rotary position sensor for measuring the angle between board and water. A GPS receiver helps to synchronize measured data with other more precise GPS devices. Accelerometer and gyroscope data is used to monitor the position of the surfboard. All data is stored on the micro SD memory card. The system is powered by a LiPo battery which can be charged using an USB connector, if the voltage drops below critical threshold device will automatically switch itself off.

The device is designed using SolidWorks 3D design software. The appliance is designed in two parts. Seals are used to ensure that the device is waterproof. Housing has a belt loop for mounting. There are two connectors - one for the angle sensor and the other for charging and USB connection. The housing is made of polyamide by 3D printing. A commercially available temporary housing is chosen to simplify prototyping process. In the product development phase the temporary housing is used to be able to update the program and to use debugging tools.

The programming is done in Keil uVision software and the code is written in C programming language and some of the standard libraries are used. The program is divided into a number of sub-functions. And it uses two different sets of functions. The critical (real time) functions are filled every 10 millisecond period, and the rest of the functions are called out every 100 millisecond period.

As a result of thesis project a working prototype was built. After testing the prototype in real life the company will specify the next steps need to be taken in the development of the data acquisition system. There is a need for calibrating software and hardware, which will be engineered later.

KASUTATUD KIRJANDUS

[1] Sell, R., Leini, M., Salong P. Mikrokontrollerid ja praktiline robootika. Tallinn: TTÜ Kirjastus, 2010.

[2] Z Fins koduleheküg [*WWW*] http://zfins.eu/ (20.04.2014)

[3] Kaitstud bakalaureusetööd masinaehituse instituut[*WWW*] http://www.ttu.ee/mehaanikateaduskond/masinaehituse-instituut-2/teadus-ja-arendustegevus-2/kaitstud-bakalaureusetood-2010-2013-2/ (20.04.2014)

[4] XBee raadiomoodul [*WWW*] http://www.digikey.com/product-detail/en/XB24-AWI-001/XB24-AWI-001-ND/935965 (20.04.2014)

[5] Farnelli kataloog [*WWW*] http://ee.farnell.com/ (20.04.2014)

[6] AMS AG koduleheküg [*WWW]* http://ams.com/eng (20.04.2014)

[7] AS5000-MA6H-1 Magnet andmeleht [*WWW*] http://ams.com/eng/Products/Position-Sensors/Magnets/AS5000-MD6H-1/%28oi%29/1 (20.04.2014)

[8] Hammond 1555cf22gy andmeleht [*WWW*] http://ee.farnell.com/hammond/1555cf22gy/box-flanged-abs-ip66/dp/1829799 (20.04.2014)

[9] Hammond koduleht [*WWW*] http://www.hammondmfg.com/1555F_SL.htm (20.04.2014)

[10] SolidWorks koduleht [*WWW*] http://www.solidworks.com/sw/products/3d-cad/packages.htm (20.04.2014)

[11] Keil uVision koduleht [*WWW*] http://www.keil.com/uvision/ (20.04.2014)

[12] SpeedAngle koduleht [*WWW*] http://www.speedangle.com/ (20.04.2014)

[13] a2-wireless-data-loggers andmeleht [*WWW*] http://smtresearch.ca/products/a2-wireless-data-loggers (20.04.2014)

[14] GPSshield andmeleht [*WWW*] http://www.ladyada.net/make/gpsshield/.

[15] STM32L151R6 andmeleht [*WWW*] http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1295/LN962/PF252050?s_searchtype=partnumber (20.04.2014)

[16] Robot Siil koduleht[*WWW*] http://www.robotiklubi.ee/projektid/robotex/2012/voistkonnad/siil (20.04.2014)

[17] STM32F3DISCOVERY andmeleht [*WWW*] http://www.st.com/web/en/catalog/tools/PF254044 (20.04.2014)

[18] Altium koduleht [*WWW*] http://www.altium.com/en/products/altium-designer (20.04.2014)

[19] MCP73831 andmeleht [*WWW*] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en024903 (20.04.2014)

[20] Pingeregulaatori andmeleht [*WWW*] http://www.farnell.com/datasheets/94457.pdf (20.04.2014)

[21] NiMH aku andmeleht [*WWW*] http://data.energizer.com/PDFs/nickelmetalhydride_appman.pdf (20.04.2014)

[22] GPS vastuvõtja andmeleht [*WWW*] http://ee.farnell.com/maestro-wireless-solutions/a2035h/gps-module-w-ant-sirf-iv/dp/2281693 (20.04.2014)

[23] AS5048B andmeleht [*WWW*] http://ams.com/eng/Products/Position-Sensors/Magnetic-Rotary-Position-Sensors/AS5048B (20.04.2014)

[24] Kiirendusanduri andmeleht [*WWW*] http://ee.farnell.com/stmicroelectronics/lsm303dlhc/sensor-3-ch-accel-mag-mod-14lga/dp/2068595?Ntt=2068595 (20.04.2014)

[25] Güroskoobi andmeleht [*WWW*] http://ee.farnell.com/stmicroelectronics/l3gd20/gyroscope-3axis-2000dps-16lga/dp/2295818?Ntt=2295818 (20.04.2014)

[26] Laadimiskiibi andmeleht [WWW] http://ee.farnell.com/microchip/mcp73831t-2ati-ot/batt-charger-li-ion-li-poly-5sot23/dp/1834890?Ntt=1834890 (20.04.2014)

[27] Pult koduleht [*WWW*] http://www.robotiklubi.ee/projektid/pult (15.05.2014)

[28] LiPO andmeleht [*WWW*] http://www.glynstore.com/content/docs/terminals/YT613938%20with%20PCM.PDF (20.04.2014)

[29] USB ESD kaitse andmeleht [*WWW*] http://www.st.com/web/en/resource/technical/document/datasheet/CD00050750.pdf (20.04.2014)

[30] AS5048 andmeleht [*WWW*] http://www.ams.com/eng/Magnetic-Encoders/AS5048 (20.04.2014)

[31] CAN andmeleht [*WWW*] http://www.embedded.com/electronics-blogs/murphy-s-law/4024614/A-short-trip-on-the-CAN-bus (20.04.2014)

[32] A-2200 andmeleht [WWW] http://www.maestro-wireless.com/a-2200-a (20.04.2014)

[33] A2035-h andmeleht [*WWW*] http://www.maestro-wireless.com/a2035-h (20.04.2014)

[34] LSM303DLHC andmeleht [*WWW*] http://www.st.com/web/catalog/sense_power/FM89/SC1449/PF251940 (20.04.2014)

[35] „L3GD20," [*WWW*] http://www.st.com/web/catalog/sense_power/FM89/SC1288/PF252443 (20.04.2014)

[36] Kõlari andmeleht [*WWW*] http://www.farnell.com/datasheets/16378.pdf (20.04.2014)

[37] Kamitra koduleht [*WWW*] http://www.kamitra.ee/ (20.04.2014)

[38] Silikageeli andmeleht [*WWW*] http://www.silicagel.com.au/categories/Silica-Gel-Packets/ (20.04.2014)

[39] Bulgin USB pistik andmeleht [*WWW*] http://www.farnell.com/datasheets/77286.pdf (20.04.2014)

[40] Rapid Prototyping Lab koduleht [WWW] http://innomet.ttu.ee/rapidlab/services.html (20.04.2014)

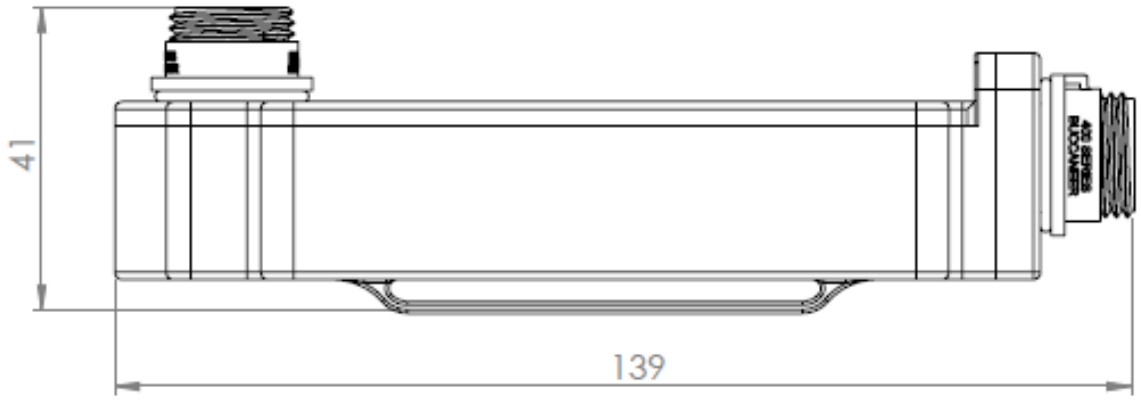
[41] STM32L1XX Library koduleht [*WWW*] http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1743/PF257913 (20.04.2014)

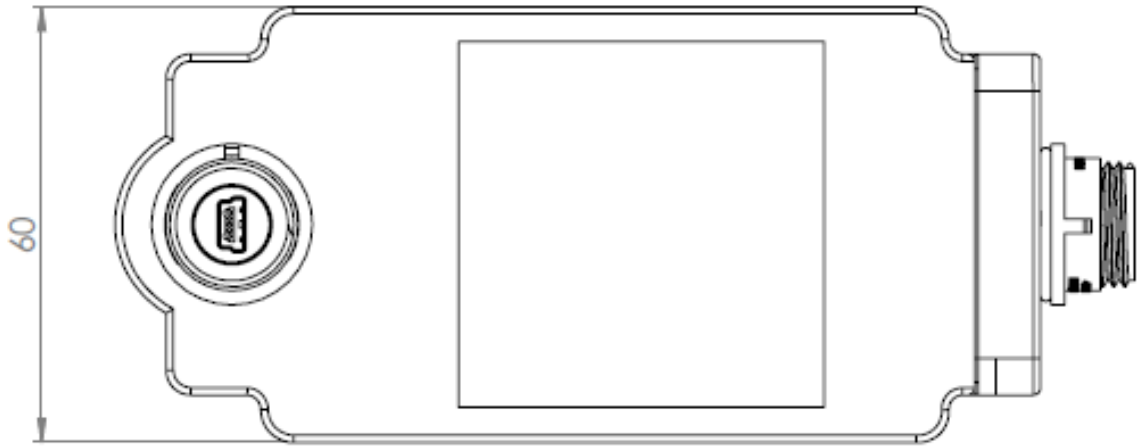
[42] Reed switch andmeleht [*WWW*] http://ee.farnell.com/hamlin/mdsr-4-22-38/switch-reed-spst-no-0-5a-200v-axial/dp/2103636?Ntt=2103636 (20.04.2014)

LISAD

LISA 1

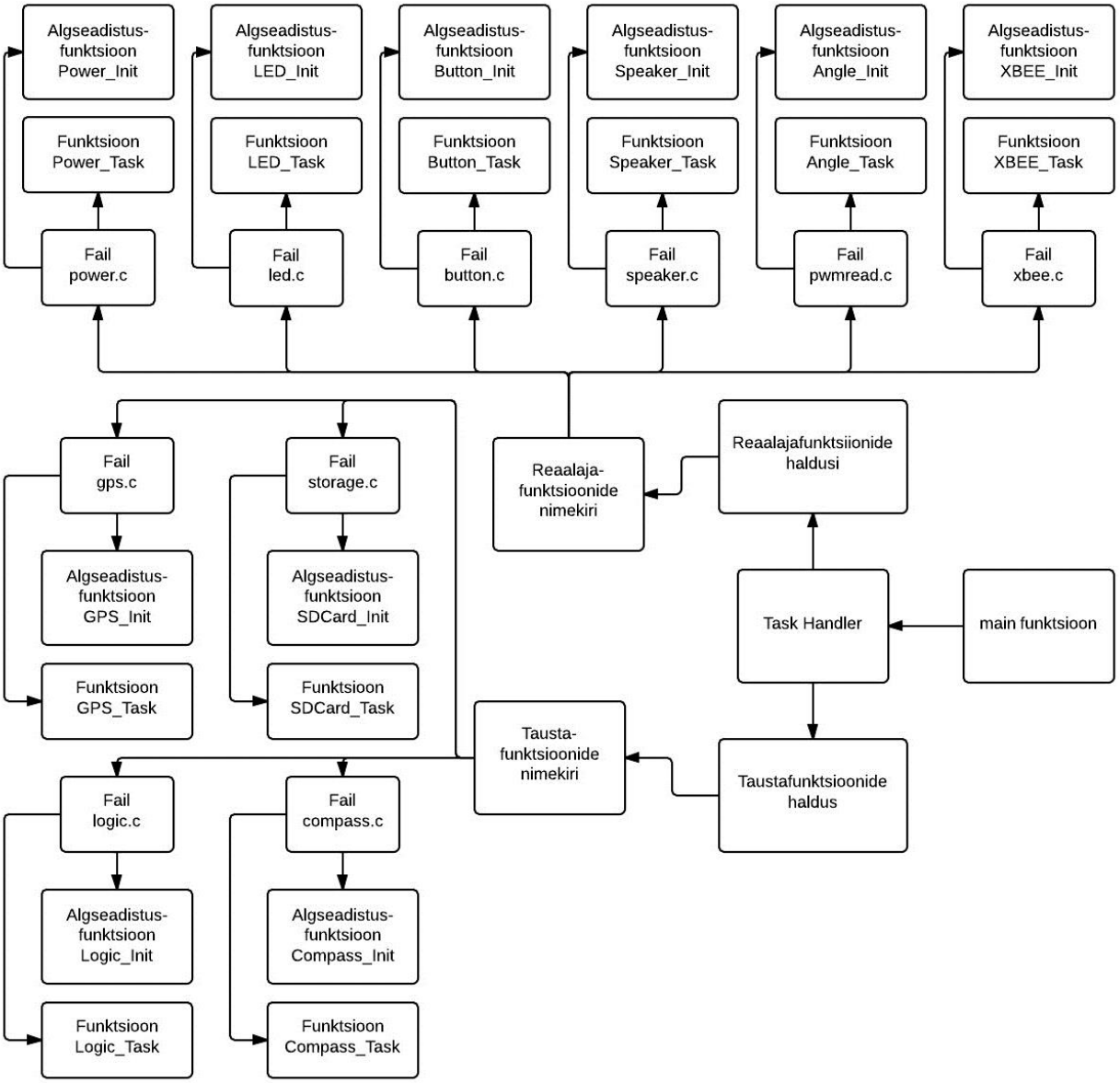
Korpuse gabariitmõõtmed



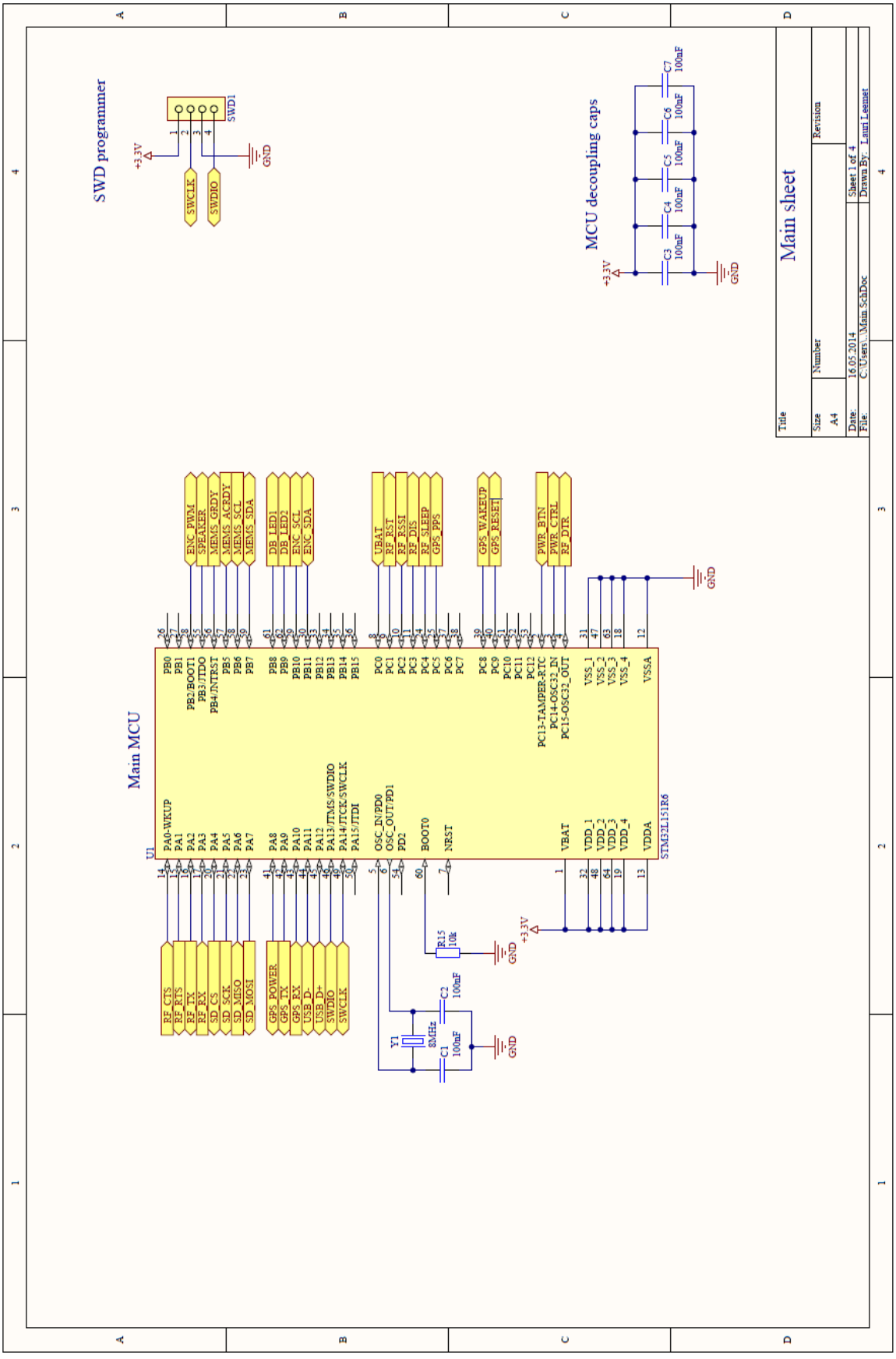


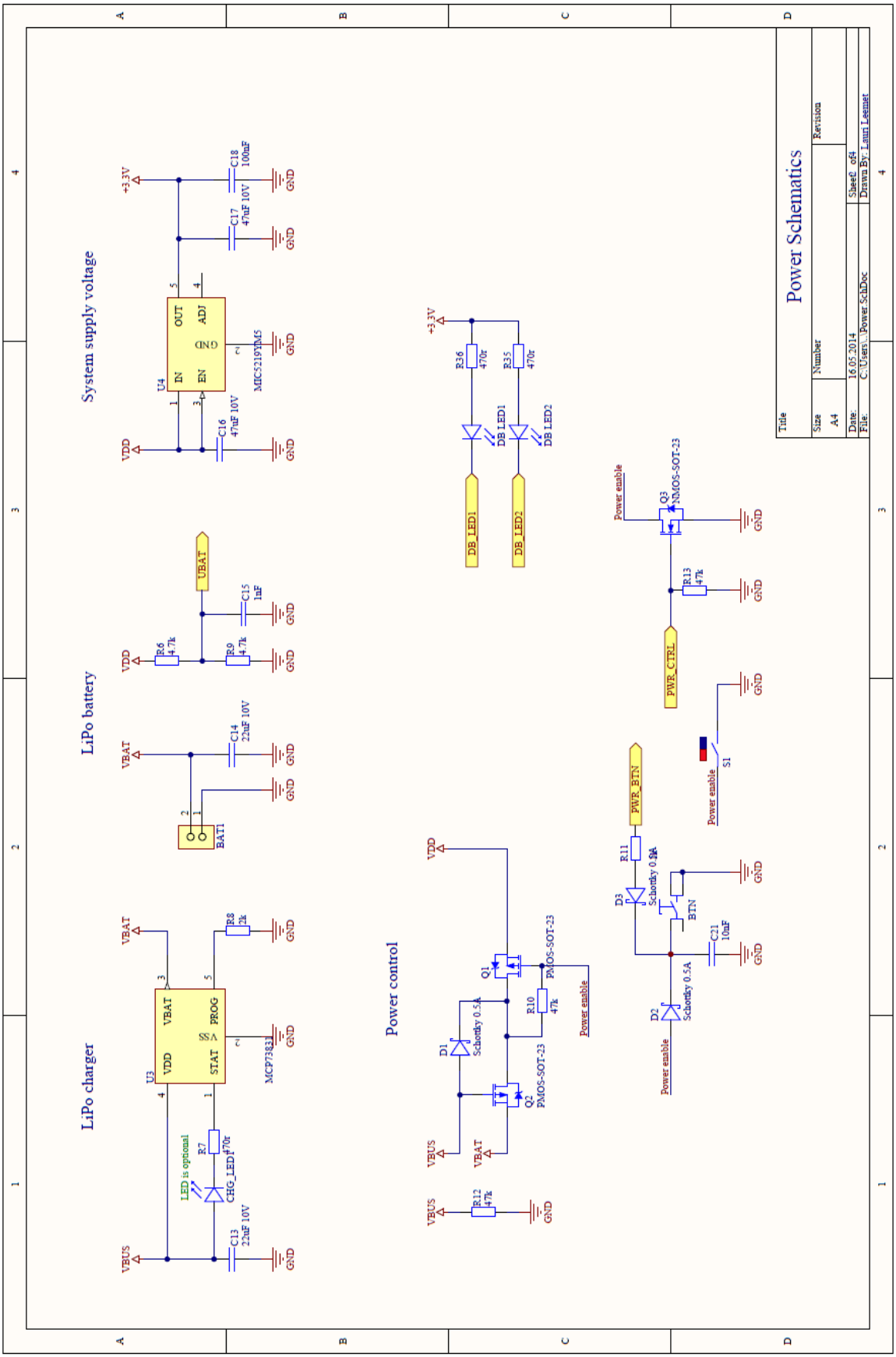
LISA 2

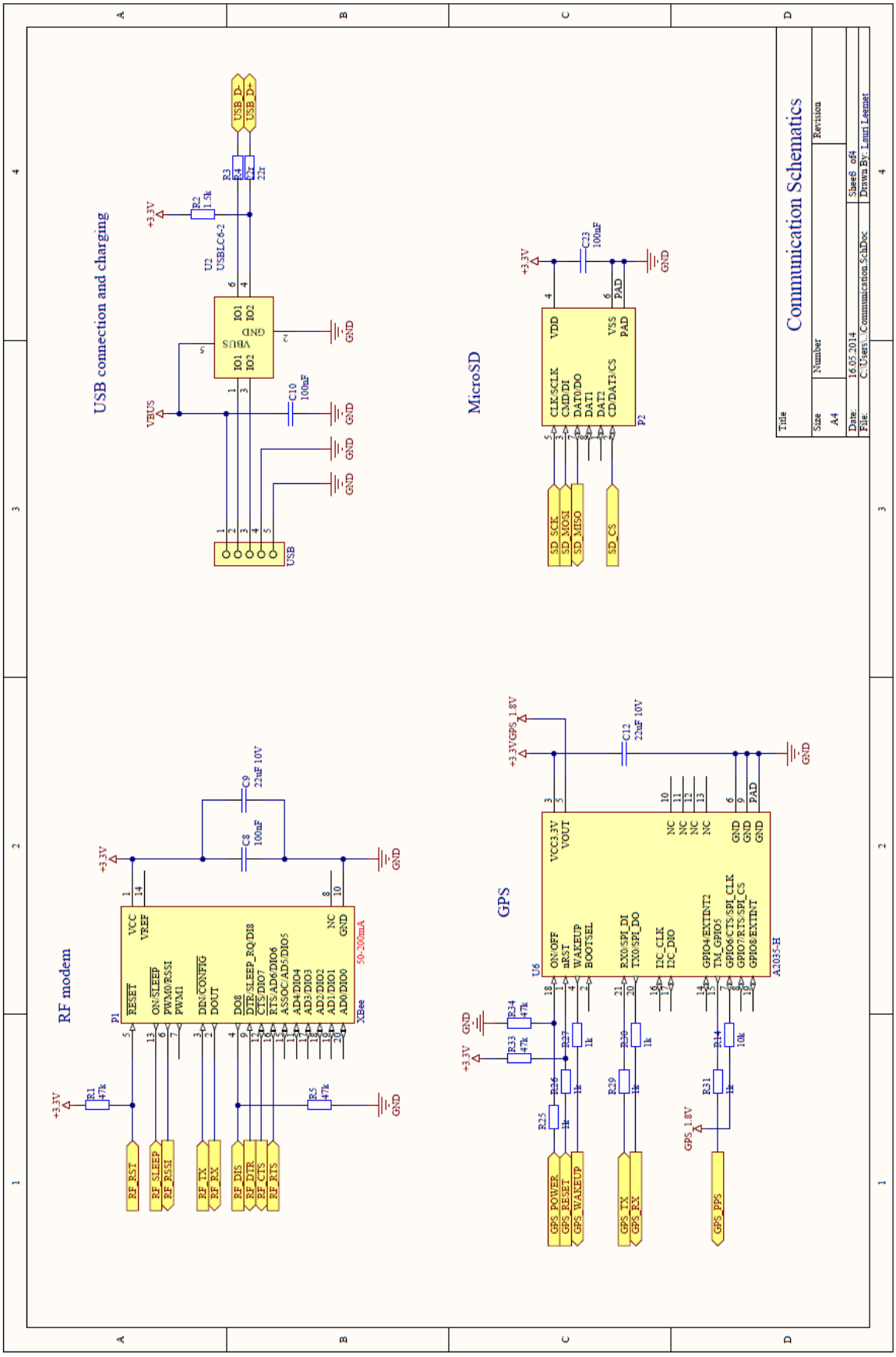
Programmi ülesehitus

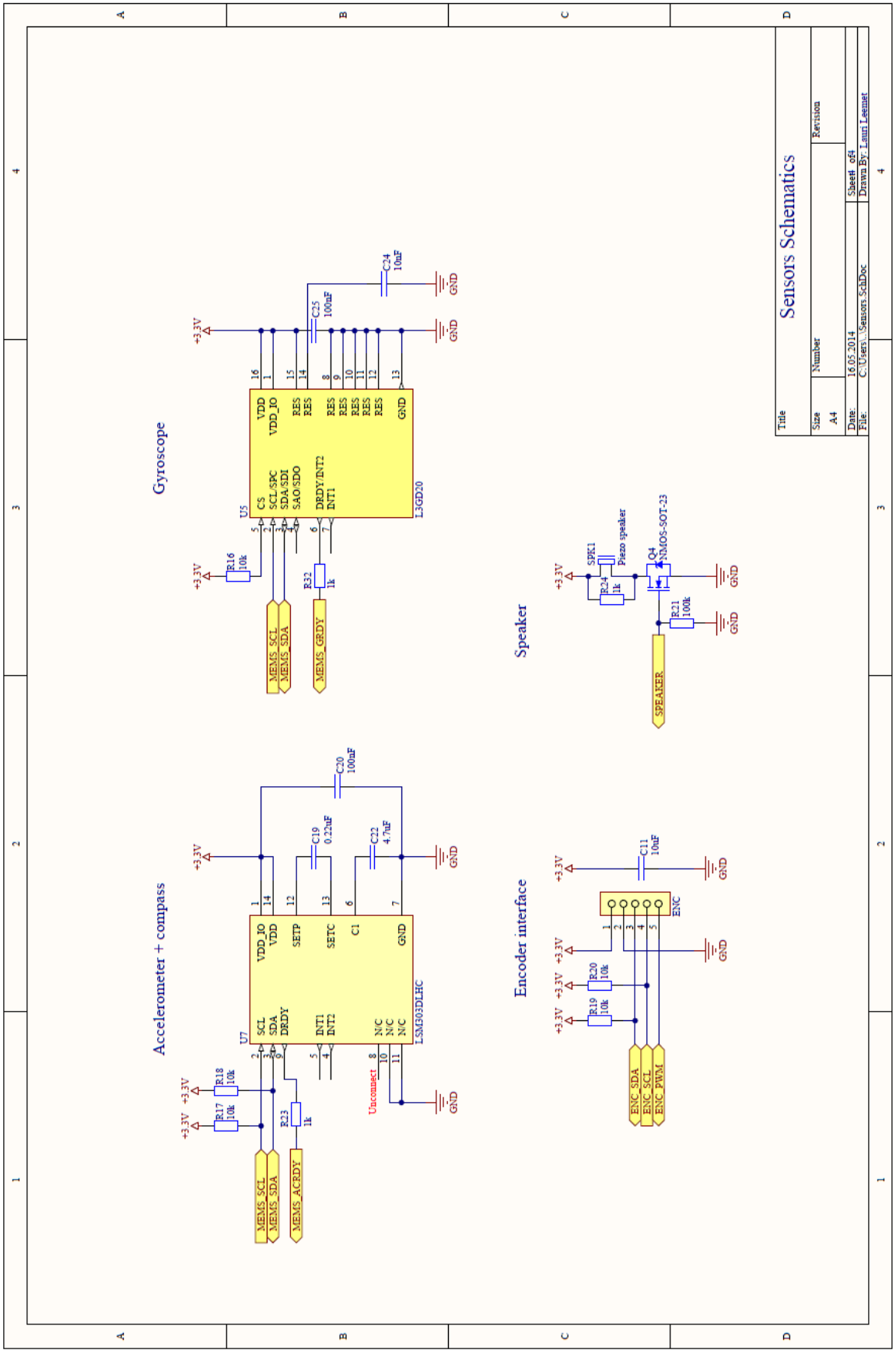


Andmehõivesüsteemi skeem ja komponentide nimekiri




System supply voltage
LiPo battery
LiPo charger
Power control
Power Schematics
Title
Size
A4
Number
Revision
Date: 16.05.2014
File: C:\Users\...\Power.SchDoc
Sheet 6 of 4
Drawn By: Lauri Leemet
MCP73831
MIC5219YM5
PMOS-SOT-23
NMOS-SOT-23
Schottky 0.5A
DB_LED1
DB_LED2
PWR_CTRL
PWR_BTN
Power enable
LED is optional


USB connection and charging
MicroSD
RF modem
GPS
Communication Schematics
Title
Size
A4
Number
Revision
Date: 16.05.2014
File: C:\Users\Communication.SchDoc
Sheet6 of4
Drawn By: Lauri Leemet
USB_D-
USB_D+
USBLC6-2
VBUS
USB
SD_SCK
SD_MOSI
SD_MISO
SD_CS
P2
P1
XBee
U6
A2035-H
RF_RST
RF_SLEEP
RF_RSSI
RF_TX
RF_RX
RF_DTS
RF_DTR
RF_CTS
RF_RTS
GPS_POWER
GPS_RESET
GPS_WAKEUP
GPS_TX
GPS_RX
GPS_PPS
+3.3V
+3.3VGPS_1.8V
GPS_1.8V
GND


Gyroscope
Accelerometer + compass
Speaker
Encoder interface
Sensors Schematics
L3GD20
LSM303DLHC
NMOS-SOT-23
Piezo speaker
MEMS_SCL
MEMS_SDA
MEMS_GRDY
MEMS_ACRDY
SPEAKER
ENC_SDA
ENC_SCL
ENC_PWM
+3.3V
GND

Bill of Materials

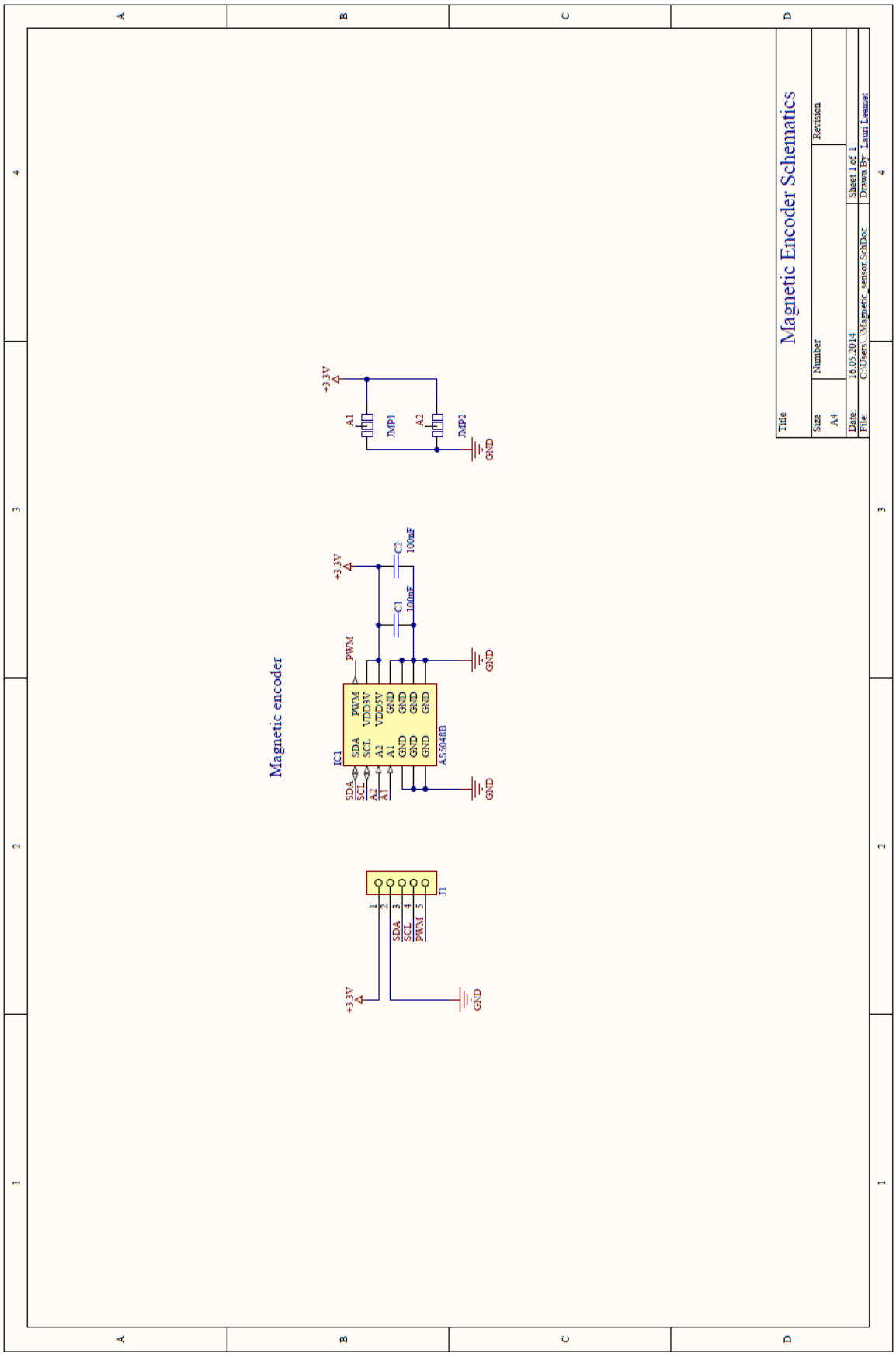
Communication Schematics

Source Data From: Loger.PrjPCB
Project: Loger.PrjPCB
Variant: None

Creation Date: 16.05.2014 15:36:29
Print Date: 41775 41775,65045

Footprint	Comment	LibRef	Designator	Description	Quantity
PINS1x2	Pins1x2	Pins1x2	BAT1		1
3ETL9	3ETL9	3ETL9	BTN		1
0603	100nF	Capacitor_0603	C1, C2, C3, C4, C5, C6, C7, C8, C10, C18, C20, C23, C25	SMD Capacitor	13
1210	22uF 10V	Capacitor_1210	C9, C12, C13, C14	SMD Capacitor	4
1210	10uF	Capacitor_1210	C11	SMD Capacitor	1
0603	1nF	Capacitor_0603	C15	SMD Capacitor	1
1812	47uF 10V	Capacitor_1812	C16, C17	SMD Capacitor	2
0603	0.22uF	Capacitor_0603	C19	SMD Capacitor	1
0603	10nF	Capacitor_0603	C21, C24	SMD Capacitor	2
0805	4.7uF	Capacitor_0805	C22	SMD Capacitor	1
0805	LED_0805	LED_0805	CHG_LED1, DB LED1, DB LED2		3
SOD123F	Diode_SOD123	Diode_SOD123	D1, D2, D3	Schottky diode	3
PINS1x5	Pins1x5	Pins1x5	ENC, USB		2
XBEE_HOLES	XBee	XBee-hole	P1	XBee /XBee-PRO OEM RF Module	1
DM3C	DM3C	microSD_DM3C	P2	microSD DM3C	1
SOT-23	PMOS-SOT-23	PMOS-SOT-23	Q1, Q2	N-MOSFET	2
SOT-23	NMOS-SOT-23	NMOS-SOT-23	Q3, Q4	N-MOSFET	2
0603	47k	Resistor_0603	R1, R5, R10, R12, R13, R33, R34	SMD Resistor	7
0603	1.5k	Resistor_0603	R2	SMD Resistor	1
0603	22r	Resistor_0603	R3, R4	SMD Resistor	2
0603	4.7k	Resistor_0603	R6, R9	SMD Resistor	2
0603	470r	Resistor_0603	R7, R35, R36	SMD Resistor	3
0603	2k	Resistor_0603	R8	SMD Resistor	1
0603	1k	Resistor_0603	R11, R23, R24, R25, R26, R27, R29, R30, R31, R32	SMD Resistor	10
0603	10k	Resistor_0603	R14, R15, R16, R17, R18, R19, R20	SMD Resistor	7
0603	100k	Resistor_0603	R21	SMD Resistor	1
REED_20MM	REED_20MM	REED_20MM	S1		1
QCP-03A	Piezo speaker	QCP-03A	SPK1	9mm piezo buzzer	1
PINS1x4	Pins1x4	Pins1x4	SWD1		1
LQFP64_N	STM32L151R6	STM32F103R6T6	U1	STM32 ARM-based 32-bit MCU with 32 Kbytes Flash, 64-pin LQFP, Industrial Temperature	1
SOT-23-6	USBLC6-2	USBLC6-2	U2		1
SOT-23-5	MCP73831	MCP73831	U3	Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers	1
SOT-23-5	MIC5219YM5	MIC5219YM5	U4		1
LGA-16	L3GD20	L3GD20	U5	Three-axis digital output gyroscope	1
A2035-H	A2035-H	A2035-H	U6	GPS Receiver Module A2035-H	1
LGA14	LSM303DLHC	LSM303DLHC	U7	3D accelerometer and 3D magnetometer module	1
ABM3	ABM3	ABM3	Y1	Ceramic surface mount microprocessor crystal	1
					86

Nurgaaduri skeem



main.c

```
typedef struct
{
  void (* InitFunction)(void);
  void (* TaskFunction)(uint32_t);

} Task;

/* Private define ------------------------------------------------------------*/
#define RT_TASK_PERIOD 10
#define BG_TASK_PERIOD 100

/* Private constants ---------------------------------------------------------*/
static const Task RT_TASK_LIST[] =
{
  { Power_Init,   Power_Task    },
  { LED_Init,     LED_Task      },
  { Button_Init,  Button_Task   },
  { Speaker_Init, Speaker_Task  },
  { Angle_Init,   Angle_Task    },

};

static const Task BG_TASK_LIST[] =
{
  { SDCard_Init, SDCard_Task    },
  { LED_Init,     LED_Task      },
  { GPS_Init,     GPS_Task      },
      { Logic_Init,   Logic_Task    }
  //{ XBEE_Init,     XBEE_Task    },
//  { Compass_Init, Compass_Task  }
};

/* Private macro -------------------------------------------------------------*/
#define RT_TASK_COUNT (sizeof(RT_TASK_LIST) / sizeof(Task))
#define BG_TASK_COUNT (sizeof(BG_TASK_LIST) / sizeof(Task))

/* Private variables ---------------------------------------------------------*/
static RCC_ClocksTypeDef RCC_Clocks;
static volatile uint32_t RTPeriodCounter = 0;
static volatile uint32_t BGPeriodCounter = 0;

/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/
int main(void)
{
  uint32_t task;
  uint32_t period;

  /* Configure SysTick to trigger interrupt every 1 ms */
```

```
  RCC_GetClocksFreq(&RCC_Clocks);
  SysTick_Config(RCC_Clocks.HCLK_Frequency / 1000);
  /* Enable SYSCFG clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE)
  /* Configure the Priority Group to 2 bits */
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
  /* Initialize real-time tasks */
  for (task = 0; task < RT_TASK_COUNT; task++)
  {
    RT_TASK_LIST[task].InitFunction();
  }

  /* Initialize background tasks */
  for (task = 0; task < BG_TASK_COUNT; task++)
  {
    BG_TASK_LIST[task].InitFunction();
  }

  /* Infinite loop */
  while (1)
  {
    /* Store period in temp variable and then reset period */
    __disable_irq();
    period = BGPeriodCounter;
    BGPeriodCounter = 0;
    __enable_irq();

    /* Execute background tasks */
    for (task = 0; task < BG_TASK_COUNT; task++)
    {
      BG_TASK_LIST[task].TaskFunction(period);
    }

    /* Wait for background tasks period time if it's not elapsed yet */
  while (BGPeriodCounter < BG_TASK_PERIOD) {}
  }
}

void SysTick_Handler(void)
{
  uint32_t task;

  /* Execute RT tasks every RT_TASK_PERIOD */
  if (++RTPeriodCounter >= RT_TASK_PERIOD)
  {
    RTPeriodCounter = 0;
    for (task = 0; task < RT_TASK_COUNT; task++)
    {
      RT_TASK_LIST[task].TaskFunction(RT_TASK_PERIOD);
    }
  }

  /* Count background tasks period in milliseconds */
  BGPeriodCounter++;
}
```

led.c

```
static GPIO_InitTypeDef GPIO_InitStructure;
__IO uint8_t Cycles;
__IO uint8_t Set;

void LED_Init(void)
{
  /* LED clock enable */
  RCC_AHBPeriphClockCmd(LED_GPIO_CLK, ENABLE);

  /* LED output pin configuration */
  GPIO_InitStructure.GPIO_Pin = LED_PIN | LED1_PIN;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_40MHz;
  GPIO_Init(LED_GPIO_PORT, &GPIO_InitStructure);
}


void LED_Task(uint32_t Period)
{

      if(Set)
      {
            GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_SET);
            Set=0;
      }
      else
      {
            GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_RESET);
            Set=1;
      }
      Cycles=0;
}

void LED_Clear(int i)
{
      if(i)
  GPIO_WriteBit(LED_GPIO_PORT, LED_PIN, Bit_SET);
      else
      GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_SET);
}

void LED_Set(int i)
{
      if(i)
      GPIO_WriteBit(LED_GPIO_PORT, LED_PIN, Bit_RESET);
      else
      GPIO_WriteBit(LED_GPIO_PORT, LED1_PIN, Bit_RESET);
}
```

pwmread.c

```
volatile uint32_t IC2Value[100] = {0};
volatile uint32_t IC2Counter = 0;
static TIM_ICInitTypeDef       TIM_ICInitStructure;
static EXTI_InitTypeDef        EXTI_InitStructure;
static NVIC_InitTypeDef        NVIC_InitStructure;
static GPIO_InitTypeDef        GPIO_InitStructure;;
static TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
static TIM_OCInitTypeDef       TIM_OCInitStructure;
uint32_t PWMRead_GetWidth(int);

void PWMRead_Init(void)
{

    /* Enable SYSCFG clock */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

    /* TIM3 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    /* GPIOB clock enable */
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_IN;
    GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_40MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /* Connect EXTI2 Line to PB2 pin */
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOB, EXTI_PinSource2);

    /* Configure EXTI0 line */
    EXTI_InitStructure.EXTI_Line = EXTI_Line2;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);

    /* Enable and set EXTI0 Interrupt to the lowest priority */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    /* Time base configuration */
    TIM_TimeBaseStructure.TIM_Period = 0xFFFFFFFF;
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
}

void PWMRead_Task(uint32_t Period)
{
```

```
}
void EXTI2_IRQHandler(void)
{
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_2) == Bit_SET)
    {
        TIM_SetCounter(TIM3, 0);
        TIM_Cmd(TIM3, ENABLE);
    }
    else
    {
        IC2Value[IC2Counter] = TIM_GetCounter(TIM3);
        TIM_Cmd(TIM3, DISABLE);
                        IC2Counter++;
    }

    /* Clear the EXTI line 8 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line2);
}
uint32_t PWMRead_GetWidth(int i)
{
    //value between 0 and 2000
    return IC2Value[i];
        if(i>=99)
                    IC2Counter=0;
}
}

void PWMRead_Task(uint32_t Period)
{

}

void EXTI2_IRQHandler(void)
{
    if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_2) == Bit_SET)
    {
        TIM_SetCounter(TIM3, 0);
        TIM_Cmd(TIM3, ENABLE);
    }
    else
    {
        IC2Value[IC2Counter] = TIM_GetCounter(TIM3);
        TIM_Cmd(TIM3, DISABLE);
                        IC2Counter++;
    }

    /* Clear the EXTI line 8 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line2);
}

uint32_t PWMRead_GetWidth(int i)
{
    //value between 0 and 2000
    return IC2Value[i];
        if(i>=99)
                    IC2Counter=0;
}
```