Tallinna Ülikool

Digitehnoloogiate instituut

AUTONOOMSE LAEVA LOOMINE ROBOTEX VÕISTLUSE TARBEKS

Bakalaureusetöö

Autor: Joosep Jõeleht

Juhendaja: Jaagup Kippar

Autor: ....., ... „ 2018
Juhendaja: ....., ... „ 2018
Instituudi direktor: ....., ... „ 2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

3. mai 2018 Joosep Jõeleht

Sisukord

Sissejuhatus

Isesõitvad sõidukid on viimase mitme aasta jooksul arenenud väga jõudsalt, eriti autode maailmas. Lennukitel ja laevadel on autopiloodid olemas olnud juba aastakümneid, kuid üheski valdkonnas ei ole veel jõutud täisautonoomsete sõidukiteni. Antud bakalaureusetöö eesmärgina arendab autor laeva, mis oleks täisautonoomne, ning suudaks läbida Robotex võistluse raja. Autor valis just sellise teema, kuna teda huvitab robootika, automatiseerimine ja võimalus aidata kaasa tulevikku kujundava tehnoloogia arendamisele.

Bakalaureuse töö esimene peatükk annab ülevaate autonoomsete sõidukite liigitamisest ja autonoomsete sõidukite ajaloost. Teises peatükkis kirjeldab autor laeva planeerimisetappe ja kolmandas peatükkis on väljatoodud laeva ehituse protsess. Neljas peatükk annab ülevaate kogu laeva kontrollivast elektroonikast ja koodist. Viimases peatükis on välja toodud testimistulemused ja võistluse tulemused.

1 Masinate juhtimise tüübid

Tänapäeval on tehnoloogia kõrgelt arenenud. Maad ümbritsevad satelliidid, mis pakuvad GPS (globaalne positsioneerimissüsteem) tuge. Kaamerad edastavad kõrge resolutsiooniga pilti, et eristada liikumistrajektooril olevaid takistusi. Arvutid on piisavalt võimsad, et töödelda kogu sensoritest saadud informatsioon ning selle abil reguleerida masina liikumissuunda. Tehnoloogia arenguga on välja kujunenud neli suuremat gruppi, kuhu saab liigitada tänapäeval tuntud liikurmasinaid: autod, lennukid, laevad jms.

1.1 Mehitatud

Bakalaureuse töö kirjutamise hetkel on kõige laialdasemalt kasutatavad süsteemid mehitatud, mida juhib inimene olles samal ajal masinas. Näiteks laev, kus kapten keerab tüüri ja selle tulemusel muutub laeva kurss. Mehitatud masinas on juhile lisa-informatsiooniks mitmesugused sensorid, näiteks sonar, GPS või elektrooniline kaardi kuvamise- ja infosüsteem (edaspidi ECDIS), mis võeti kasutusle asendamaks vanemaid paberkaarte.

1.2 Kaugjuhitav

Kaugjuhitavate masinate juht kasutab sõiduki juhtimiseks juhtmevaba kontrollerit. Juhtmevabadel kontrolleritel on mitmeid erinevaid meetodeid ühendumaks kontrollitava seadmega, näiteks:

- raadiolaine, mis on mõeldud kontrollimaks väikeses raadiuses seadmeid;
- satelliitvõrgustik, võimaldab satelliitide õige asetuse korral kontrollida seadmeid igal pool maailmas.

Parim näide kaugjuhitavatest masinatest on viimaste aastate jooksul laialdaselt levinud droonid. Sõjanduses on kaugjuhitavad droonid olnud väga suure tähtsusega, et hoida ära inimeste elude ohtu seadmist, samuti on droonid leidnud kasutusala ka huvi ja meelelahutusena tsiviilmaailmas.

1.3 Pool-autonoomne

Pool-autonoomsed (ing. k *semi-autonomous*) masinad on suutelised toimetama ilma inimsisendita vaid piiratud olukordades. Pool-autonoomsed süsteemid kasutavad selleks sensorite sisendeid ja arvuteid, mis mõjutavad masina juhtimissüsteemi. Hea näide selle kohta on aastal 2003 Jaapanis välja lastud Toyota Prius. Selle sisseehitatud tahavaatekaamera, roolivõim ja spetsiaalne tarkvara koostöös pargivad auto külgboksi ilma liialdase inimsisendita (Time magazine, kuupäev puudub). Antud süsteem on pool-autonoomne kuna inimene peab siiski andma autole teada, et ta soovib parkida või, et on ohutu lahkuda parkimiskohast.

1.4 Autonoomne

Autonoomsed masinad täidavad ülesandeid, kasutades selleks masinale kättesaadavate sensorite infot. Autonoomne laev näiteks kasutaks:

- GPS-i, jälgimaks asukohta maailmas;
- ilmaennustusi, trajektoori parandamiseks etteantud lubatud muutuste piires;
- radareid, mis jälgivad ja ennetavad liikumistrajektooril olevaid sõidukeid või takistusi.

2 Autonoomsete masinate ajalugu

Autonoomseid masinaid on üritatud luua juba mitmekümneid aastaid. Kõige varasem näide on lennukite autopiloot, mille esimene prototüüp avalikustati 18. juunil 1914 (Scheck, 2017). Erinevalt tänapäevastest autonoomsetest masinatest ei olnud Lawrence Sperry loodud lennuki autopiloot juhitud arvutite poolt, vaid hoopiski mehaaniliste jõudude poolt. Antud autopiloot ei olnud tegelikkuses midagi rohkemat kui mehaaniline stabilisaator, mis hoidis lennukit lendamas otse, sellegipoolest pani see alguse isejuhtivate masinate loomisele.

Ühed esimestest automaatse pööramisvõimega masinatest olid purjekad, mille tuulelipu külge oli seotud nöör, mis ühendus paadi tüüri külge, hoides nii laeva muutuvate tuultega õigel kursil liikumas. (Weber, 2014)

2.1 Targad kiirteed

1939. aasta New Yorki maailma messi raames tutvustati esmakordselt ideed tarkadest kiirteedest. Näituse nimi oli Futurama ja selle teemaks oli võimalik maailma olukord 20 aasta pärast (1959–1960). Futurama oli suureskaalaline mudel, mis kujutab peaaegu kõiki tüüpi maastikke Ameerikas ja illustreerib, kuidas kiirtee süsteemi saab laotada üle terve riigi. Kordagi sirgelt kursilt kõrvale kõikumata jälgib kiirtee disain alati nelja põhimõtet: turvalisus, mugavus, kiirus ja ökonoomsus. (Geddes, 1940)

Targa kiirtee toimimise printsiibiks oli idee, et kasutajad sõidavad kiirteele ning seejärel võtab autojuhtimise üle arvuti, mis jälgib maas olevat magneetilist rada, lubades juhil lõõgastuda (Weber, 2014). Targal kiirteel olnuksid autod jagatud gruppidesse, mille potentsiaalsed kasupunktid oleksid järgnevad (Platoon (automobile), kuupäev puudub):

- grupisiseselt oleks autod väga lähestikku, millega tõuseb nende ökonoomsus;
- korraga mahuks ühe teelõigu peale rohkem autosid;
- vähem kokkupõrkeid;
- vähem ummikuid.

2.2 MUNIN projekt

Maritime Unmanned Navigation through Intelligence in Networks (edaspidi MUNIN) on koostööprojekt, mille kaasrahastajaks on Euroopa Komisjoni seitsmes raamprogramm. MUNIN-i eesmärgiks on arendada ja kinnitada kontsept autonoomsele laevale, mida defineeritakse kui laev, mida primaarselt suunab automaatne pardal asuv süsteem, kuid juhitakse kaugjuhtimisega operaatori poolt juhtimiskeskusest (MUNIN, MUNIN, kuupäev puudub). MUNIN on võtnud projekti aluseks Waterbone^{TP} visiooni autonoomsest laevast, mis iseloomustab antud sõidukit, kui „Järgmise generatsiooni modulaarsed juhtimissüsteemid ja sidetehnoloogia, mis võimaldavad monitooringut ja juhtimisfunktsionaalsust nii pardalt kui ka kaugjuhtimiskeskusest. Antud süsteemid hõlmavad endas ka kõrgelt arenenud otsuste tegemise süsteeme, mis annavadlaevale võimekuse olla kaugjuhitult pool-autonoomne või toimida täis-autonoomselt.“ (Cunningham, 2014)

2.2.1 MUNIN projekti vajadus

MUNINi projektis välja töötatud mehitamata ja autonoomse laeva mõiste aitab kaasa jätkusuutliku veetranspordi kõigile aspektidele, võimaldades (MUNIN, MUNIN’s Rationale, kuupäev puudub):

- ökoloogiliselt jätkusuutlikumat transporti:
    - meeskonna kulud on väiksemad;
    - laeva liikumiskiirus on madalam;
    - laev toimib tõhusamalt;
- sotsiaalselt aksepteeritavamat transporti:
    - suurendades märkimisväärselt meretranspordi erialade sotsiaalset ühilduvust ja atraktiivsust;
    - tõstes meretranspordi ohutust.

2.3 Rolls-Royce

25. jaanuaril 2018 avas Rolls-Royce maailma tasemel autonoomsete laevade uurimis- ja arengukeskuse Turku, Soomes. Rolls-Royce on bakalaureuse töö

kirjutamise hetkel olnud üks ambitsioonikamatest pool- ja täisautonoomsete laevade arenduse edasiviiatest. Rolls-Royce merenduse osakonna president Mikael Mäkinen on öelnud: „Autonoomne kaubandus on merendussektori tulevik. Samamoodi kui nutitelefon, on intelligentne laev revolutsioon, mis muudab laeva disainimise ja opereerimise maastiku täielikult.“

Rolls-Royce on teinud avalikkusele nähtavaks mitmeid videosid, mis kujutavad väga futuristlikku juhtimiskeskust, millest on võimalik:

- saada ülevaadet laeva olukorrast kasutades selleks laeval olevaid droone;
- saata välja meeskondasid, kes parandavad laevaga tekkinud probleeme;
- kontrollida mootori töötamisvõimet;
- jooksutada diagnostikat ja vastavalt sellele otsustada, kas laev suudab oma sihtkohta välja sõita antud seisus, või peaks see seisma jääma, ning ootama tehnilist tuge.

Kõik need ambitsoonikad plaanid on Rolls-Royce-l kavas toimima saada aastaks 2035, mil peaks esimene mehitamata täisautonoomne laev olema võimeline vedama veoseid üle ookeanide.

2.4 Robotex

Robotex on planeedi suurim robootikasündmus, mis toimub koostöös Tallinna Tehnikaülikooli ja Tartu Ülikooliga. Tegemist on tehnoloogiaalase kogupereüritusega, kus on tegevusi igas vanuses võistlejale, pealtvaatajale kui ka lihtsalt huvilisele. Kolme päeva jooksul saavad Robotexi külastajad osa rahvusvahelistest robootikavõistlustest, tehnoloogianäitustest, konverentsidest ja töötubadest. (Robotex, kuupäev puudub)

Robotexil toimuvad erinevad robootika võistlused nagu näiteks Tallinki veeralli, kus osales ka autori loodud laev, joonejälgiate kiirusvõistlus, laburindi läbimise võistlus ja palju teisi. Samuti on esindatud ka erinevad robootikaga tegelevad firmad, kes tutvustavad enda arendatud lahendusi ja tooteid.

2.4.1 Tallinki veeralli

Võistluse reeglid[1] olid väga piiravad võrreldes autori esimese ideega. Järgnevana toob autor välja kõige tähtsamad reeglid, antud töö raames.

1. Väljak
    1. Raja trajektoor on kõver ja kinnine;
    2. Raja laius on vähemalt 60 cm;
    3. Rajal võib olla ristteid, mis tuleb läbida otse sirgjooneliselt;
    4. Väljakul võib olla lihtsamaid takistusi nagu poid, seinad ja rajapoolitajad, mis on paigaldatud nii, et seinaääri mööda liikuv robot ei oleks võimeline rada läbima;
2. Robot
    1. Robot peab olema autonoomne;
    2. Robot peab ujuma või heljuma;
    3. Roboti maksimaalsed mõõdud on 35 x 20 x 35 cm (pikkus x laius x kõrgus) ja mass kuni 2 kg;
    4. Robotil peab olema start- ja stoppnupp või pult;
3. Võistlemine
    1. Rajal võistleb korraga kuni 3 robotit;
    2. Raja läbimiseks on aega 5 minutit;
    3. Robot võib sõitma hakata mitte varem kui 5 sekundit pärast stardisignaali.

Reeglitest kõige olulisemateks võib lugeda punkte 2.3 ja 2.4 ja samuti 1.4. Punkt 2.4 on eriti oluline turvalisuse eesmärgil, juhuks kui mõni võistlusmasin peaks hakkama kahjustama teisi võistlejaid või pealtvaatajaid. Punkt 2.3 sai olema laeva disaini juures kõige suuremaks piirajaks.

[1] https://goo.gl/85VWXu

3 Ettevalmistus

Bakalaureusetöö tulemusena valminud laev ehitati koostöös Jaan-Martin Kuusmanniga (edaspidi laeva kaasautor). Kaasautori sisendiks laeva ehitamisel oli teadmised ehitusmaterjalide valikust ja eelnev kogemus laevandusega. Bakalaureusetöö autor aitas laeva kere loomisel, disainis 3D mudelid ja kirjutas laeva toimimiseks vajalikud Python-i ja Arduino koodid.

Töö esimese etapina koostas autor laeva ehitamiseks tarvilike materjalide loendi. Loend jaotati kolmeks grupiks: keretööd, elektroonika ja mootori juhtimiseks tarvilikud süsteemid.

3.1 Keretööd

Autori ja kaasautori arutelu tulemusena jõuti järeldusele, et laeva kere peaks olema loodud vahtpolüstüroolist, mis kaetakse kahe kihi klaaskiud riidega ja on vahtpolüstürooli külge kinnitatud epoksiid liimiga. Vahtpolüstürool sai algselt valitud lihtsa töötlus protsessi tõttu. See pidi toimima vormina, mille ümber luuakse klaaskiud riidest kere. Kiiresti sai selgeks, et klaaskiud riie üksinda ei ole piisavalt tugev, et hoida kinni mootoreid, võlle ja veekindlat karpi, millesse oli paigutatud ülejäänud elektroonika.

Erinevate lahenduste kaalutlemisel otsustas autor kasutada Tallinna Ülikooli Digitehnoloogia Instituudi 3D printerit. Printeri abil sai autori loodud disainidest välja printida vajalikud tükid.

3.2 Elektroonika

Arutluse tulemusel otsustasid laeva autorid, et laeval peaks olema kasutusel kuus hc-sr04 ultraheli sensorit, kuid arenduse käigus lisati juurde veel kaks. Ultraheli sensoreid otsustati kasutada, kuna võistluse reeglistikus ei olnud mainitud, millisest materjalist on võistluse väljak ehitatud. Üritades vältida ebasoodsat olukorda, kus väljak on tehtud läbipaistvast materjalist ja infrapuna sensor ei oleks suuteline saama korrektset lugemit, võeti kasutusse ultraheli sensorid.

Lisaks ultraheli senoritele oli plaanis laeval kasutada veel kompassi ja kiirendussensorit, mille ülessandeks oleks olnud kaardistada rada ja sisse viia ennetavaid parandusi kursile, et läbida rada edukamalt, kuid antud funktsionaalsuse välja arendamiseks ei jäänud aega. Samuti oli alguses plaanis lisada laevale kaks servomootorit, üks mõlema mootori kohta, mis oleksid kontrollinud tüüre.

Võistluse reeglistikuga kooskõlas olemiseks lisas autor laevale ka kolm nuppu. Esimene on kohene seiskamise lüliti, teine nupp mootorite initsialiseerimiseks ja kolmas on start nupp mille tööle panemise hetkest hakkab süsteem loendama alla kümmet sekundit, mis oli võistluse üks nõuetest.

3.3 Liikumissüsteemid

Vastavalt saadavusele otsustasid laeva autor ja kaasautor kasutada laeva kaasautori personaalseid mootoreid ja akut. Laeva vooluallikaks sai valitud Turnigy 2200mAh 3S 40C[2] aku. Laeva mootoriteks oli kasutusel kaks Turnigy Aerodrive DST-1200[3]. DST-1200 mootorid on harjavabad alalismootorid (ingl k. *Brushless DC motor*). Nende juhtimiseks kasutatakse *Electronic speed controller*'id (edaspidi ESC). ESC ühendatakse kahe juhtmega aku külge ja üks signaali juhe ühendatakse mikrokontrolleri külge. Väljundina on ESC-l kolm juhet mis ühendatakse mootoriga.

Autoril oli plaanis kasutada tagurpidi liikumiseks tarkvaraga kontrollitavat ESC-d. See tähendab, et ESC-le oleks olnud võimalik öelda, mis pidi mootor töötama peab. Välismaalt telliti ära vastava spetsifikatsiooniga ESC-d, kuid need ei jõudnud õigeks ajaks kohale. Tagavara plaanina võeti kasutusele laeva kaasautori HobbyKing 20A[4] ESC-d. Tagurpidi liikumise võimekuse saavutamiseks lisati laevale juurde kolmas mootor, suunaga laeva nina poole.

3.4 3D mudelid

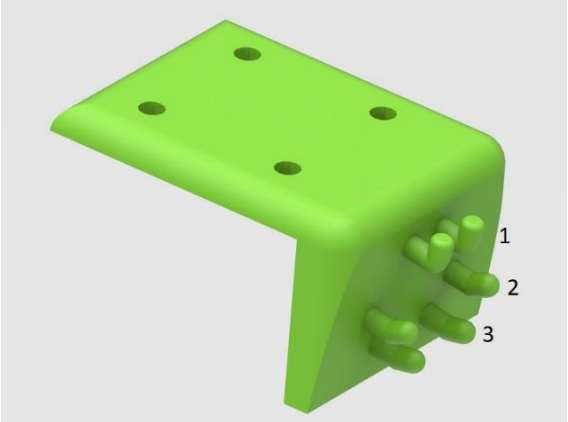
Projekti jooksul lõi autor kaks 3D mudelit, mis prinditi kasutades Tallinna Ülikooli Digitehnoloogiate instituudi 3D printerit. 3D prinditud juppide kasuks otsustas

[2] https://goo.gl/SEqMPF
[3] https://goo.gl/JbwJXF
[4] https://goo.gl/Mw7xn8

autor, kuna nende modifitseerimine toimus kiiresti, kui välja jätta printimisaeg ja autor oli juba varasemalt kokku puutunud 3D modeleerimis tarkvaraga Solid Edge ST10.

3.4.1 Mootorihoidja

Mootorihoidja mudeli disainimisel pidi autor arvestama põhiliselt kahe faktoriga. Esiteks mootorid mida laeval kasutati pidid kõik olema 80 kraadise nurga all, et mootorivõllid tuleksid kerest õige asukohapealt välja. Teiseks arendustegevuse lihtsustamiseks lõi autor kinnitused sellisena, et mootoreid ei peaks kinni kruvima ja neid oleks lihtne eemaldada. Selleks sai loodud mootori kinnitusplaadis olevate aukude asetusega kolm jalgade paari, välja toodud illustratsioonilIllustratsioon 1. Kinnitustoru suunamuutus võimaldas kinnitada mootori ilma liigset jõudu kasutamata stabiilsesse asendisse. Hilisemas testimisfaasis tuli välja, et 3D prinditud jupp ei olnud siiski piisavalt tugev, ning kui mootor töötas üle 30 protsendilise võimsusega, hakkas vibratsioon jalgu katki tegema.



Illustratsioon 1 – 1. 90-kraadise nurga all olevad jalad, 2. 150-kraadise nurga all olevad jalad, 3. 160-kraadise nurga all olevad jalad

3.4.2 Veetakisti

Teine autori disainitud ja 3D prinditud element oli veetakisti, kasutatud asukohad on nähtavad Illustratsioon 4 punktidena 5 ja täiskujul Illustratsioon 2. Veetakistid disainiti 6mm sisemise diameetriga võllikaitse ja 4mm välisdiameetriga võlli jaoks. Tükil on kuus 0.7mm sügavust kolmnurkset sisselõiget, mis aitavad lisaks hoida läbi lekkinud vett kinni. Kuna takisti tehti täpselt võlli mõõtu ja 3D printer ei olnud

printimisel päris täpne pidi autor võlli jõuga läbi suruma. Selle tulemusel leidis autor, et takisti on hakanud toimima ka võlli stabilisaatorina. Laeval kasutati kokku kuute veetakistit, kaks iga võllikaitsme mõlemas otsas. Illustratsioon 4 on välja toodud neist kolm, kuna need olid vee takistamisel rohkem tähtsad.

Illustratsioon 2 - veetakisti

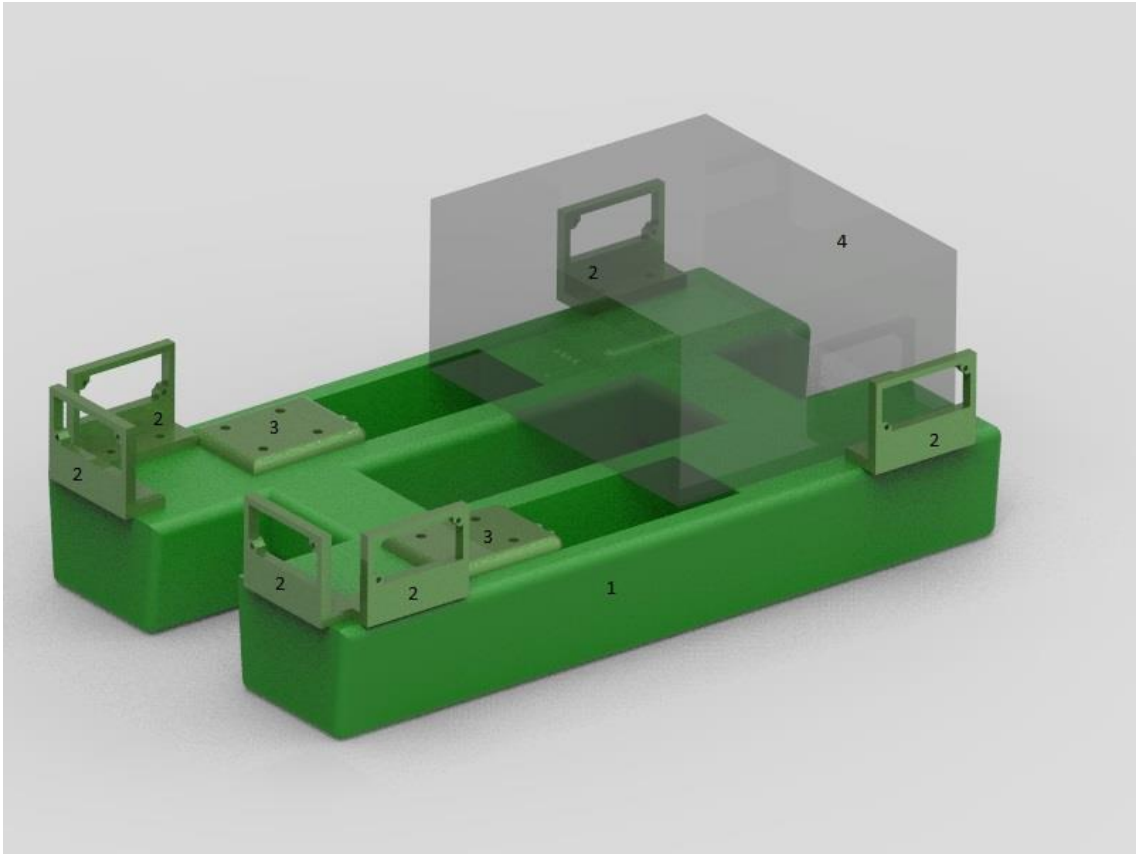
3.4.3 Muud 3D prinditud tükid

Laeva ehitamisel kasutas autor veel kahte 3D prinditud tükki. Esimeseks oli GrabCAD 3D mudelite repositooriumist leitud HC-SR04 sensorihoidja (Rose, 2016), nähtav illustratsioonidel Illustratsioon 3 ja Illustratsioon 4 tükina number 2. Teiseks oli laeva propeller, mille lõi laeva kaasautor Jaan-Martin Kuusmann.

4 Kere ehitamine

4.1 Versioon 1 (Katamaraan)

Alloleval Illustratsioon 3 on näha, milline nägi välja laeva esimene iteratsioon 3D mudelina. Iteratsioon sai inspiratsiooni katamaraanist. Laev disainiti sellisena, et hoida võimalikult palju kaalu kokku. Katamaraani tagaosas oli karp, kus sees oli pingemuundur ning Raspberry Pi ja Arduino Mega mikrokontrollerid. Katamaraani esiosas asus laeva aku ja mootori kinnitused. Laeval kasutatavad kuus sensorit, on kahe kaupa paaris. Paarid on üle kesktelje peegeldatud laeva äärtesse. Laeva ninas olevad sensorid on suunatud otse ette ja kontrollivad, kas laeva liikumissuunas asub takistusi. Mõlemal küljel asus kaks sensorit, mille sisendite pealt kontrolliti, kas laev liigub sirgelt või on ta viltuselt rajal.



Illustratsioon 3 - 1. epoksiid ja klaaskiud kere, 2. ultraheli sensorite hoidjad, 3. Mootorihoidjad, 4. Elektroonika hoiustamise karp

Esimene iteratsioon laeva kerest andis autorile hea ülevaate sellest, mida peab parandama. Kõige suuremaks murekohaks oli ujuvus. Ilma akuta oli veepiir laevatekkist umbes 5mm. Sellise äärise puhul oleks kiirenduse korral vesi üle esiosa tulnud. Peale esimest veetesti otsustas autor välja arvutada, mis on selle laeva maksimaalne kandevõime. Autori kartusi kinnitades oli tulemuseks ligikaudu 1,75 kg koos laeva kerega. Kuigi Robotex võistlusel, oli maksimaalseks kaaluks 2 kg, oli autori laeva kaal juba kõrgem, kui laeva kandevõime.

Autor üritas leida võimalike meetodeid, kuidas vähendada laeva kaalu. Üheks variandiks oleks olnud luua laeva kere puhtalt klaaskiud riidest, kuid autor leidis, et see oleks olnud saadaolevate vahenditega liiga raske. Teiseks variandiks oleks olnud osta juba valmistehtud plastikust kere. Kuid see ei sobinud kuna projekti alguses sai ostetud epoksiidi ja klaaskiud riiet piisavalt suurtes kogustes, et luua teine iteratsioon laevast ilma keskmise tühja alata.

Väiksemad vead, mida autor avastas esimeses variandis:

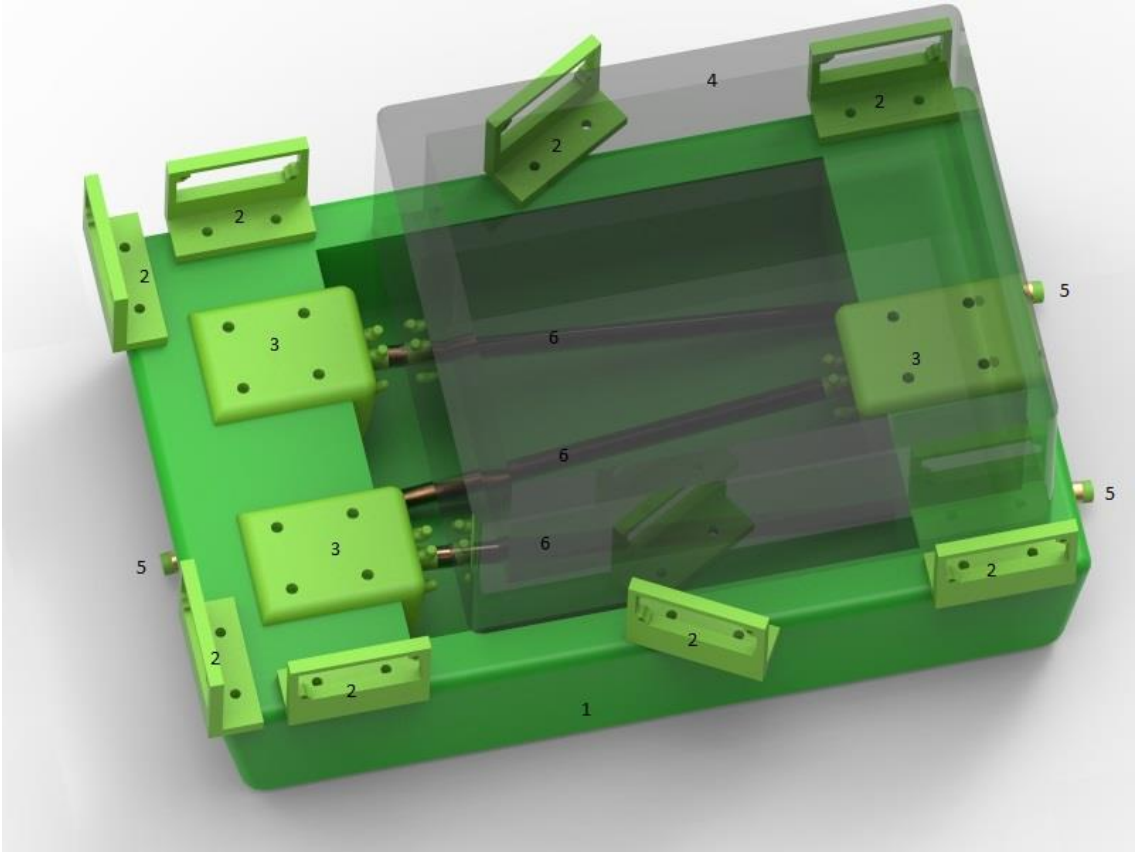
1. Laeva raskuskese oli liiga taga, ning tagumine osa vajust vaikselt vee alla;
2. Mootorite võlle oli väga raske saada sirgelt masinasse, kuna autor pidi epoksiidist pimesi läbi puurima;
3. Ultraheli sensoritel tekkis pime nurk, kus ükski sensor ei saanud korrektset sisendit hetke suuna kohta;

4.2 Versioon 2

Illustratsioonil 4 on näha laeva kere teist iteratsiooni. Eelmise peatükki lõpus olnud loetelu esimese punkti parandamiseks asendati tagaosas asuv eletroonikat hoidev karp pikemaga, mis võimaldas paigutada kogu elektroonika ühtlasemalt. Ultraheli sensorite pimeda nurga probleemi parandamiseks lisas autor laevale juurde kaks sensorit. Need paigutati mõlemale küljele 30 kraadise nurga alla, suunaga sõidusuunas, eelnevas iteratsioonis olnud külgmiste sensorite vahele.

Kere põhiosa koosnes teises iteratsioonis kolmest tükist. Kaks äärmist millesse oli uuristatud mootori võlli jaoks kanal ja üks keskmine mis ühendas neid omavahel. Kõik kolm tükki tehti alguses vahtpolüstüroolist valmis, ning seejärel, kui võllid olid

sisse asetatud, kaeti klaaskiud riidega ja liimiti epoksiidiga vahtpolüstürooli külge. Pärast epoksiidi kivistumist, puuriti mootorikinnituste ja sensorihoidjate kruvikohad ja seejärel eemaldati võimalikult palju sisust.



Illustratsioon 4 – 1. epoksiid ja klaaskiud kere, 2. ultraheli sensorite hoidjad, 3. Mootorihoidjad, 4. Elektroonika hoiustamise karp, 5. veetakistid, 6. mootorivõllid

Sellel hetkel, kui uus kere hakkas valmis saama mõistis autor, et varasemalt tellitud kahesuunalised ESC-d ei jõua siiski õigeks ajaks kohale. Seejärel otsustas autor, et laevale tuleb ka lisada tagurpidi liikumiseks eraldi mootor. Tagupidi käik andis laevale ka kõvasti manööverdamisvõimet juurde, kuna enam ei olnud laeva pöördepunkt taga keskel. Ühe lisamootoriga oli võimalik panna laev pöörlema ümber oma kesktelje, mis võimaldas laeval navigeerida paremini võistlusrajal.

5 Sensorid, kontrollerid ja kood

5.1 Sensorid

Laeval kasutatud HC-SR04 sensorid toimivad lihtsal *trigger – echo* põhimõttel. See tähendab, et kui *trigger* jalale antakse 10 mikrosekundit 5V signaali, siis hakkab sensor saatma välja signaali 20 mikrosekundiks. Viimaks pärast kogu signaali välja saatmist avatakse mooduli kuulari pool. Kuular ootab kuniks varem väljasaadetud ultraheli signaal peegeldub tagasi ja tagastab kogu tsükli pikkuse mikrosekundites. Mikrosekundite põhjal saab välja arvutada, kui kaugel oli peegeldunud objekt. Selleks tuleb leitud aeg jagada kahega ja siis veel 29.1-ga, et saada kaugus sentimeetrites või 74-ga, et saada väärtus tollides. (Ultrasonic Ranging Module HC - SR04, kuupäev puudub)

Sensorid olid ühendatud laeval oleva Arduino Mega mikrokontolleri külge. Kõigi kaheksa sensori toide oli kokku ühendatud ning seejärel ühe juhtmega ühendatud Arduino 5V väljundisse. Samamoodi tehti ka maanduse ehk GND jalgadega. Sensorite „trig“ ehk *trigger* jalad olid kõik eraldi ühendatud paarisarvulistesse päistesse vahemikus 38 kuni 53 ja „echo“ jalad samas vahemikus paaritutesse päistesse nii, et ühe sensori jalad on kõrvuti.

5.2 Mikrokontrollerid

Põhikontrollerina kasutas autor Raspberry Pi-d. Raspberry Pi sai valituks, kuna sellel on sisseehitatud WiFi võimekus ja Python programmeerimiskeel on autorile sobivam. Raspberry Pi jooksutab Linux operatsioonisüsteemi, mis võimaldab kogu süsteemi jooksutada eraldiseisvalt. Süsteemi eraldiseisvus ja WiFi võimekus andis autorile testimisel palju vabamad käed. Raspberry Pi-s seati püsti kaugtöölaua ühendus, ning läbi telefoni loodud mobiilse leviala oli võimalik kontrollida basseinis olevat laeva vajaduse korral manuaalselt, rüperaaliga basseini kõrvalt.

Teise kontrollerina oli kasutusel Arduino Mega, mille ülesandeks oli suhelda sensoritega. Arduino sai valitud, kuna algselt oli plaanis lisaks ultraheli sensoritele kasutada veel kompassi ja veekiiruse mõõtjat. See oleks tähendanud seda, et kokku

oleks kasutusse läinud 15 päist plaadil. Laeva teises iteratsioonis lisati juurde veel 2 sensorit ehk lisaks oleks tulnud võtta veel kasutusele 4 päist. Viimaks oleks olnud tarvis ühendada Raspberry Pi külge kolm mootorit ja kaks nuppu. Raspberry Pi plaadil on saadaval kasutuseks 26 *general-purpose input/output* (edaspidi GPIO) päist. Laeval oleks kasutusel olnud 24 päist ja see tähendaks, et Raspberry Pi oleks pidanud toimima peaagu maksimaalse jõu peal koguaeg. Lisaks oleks olnud tarvis muundada sensoritest tulev informatsioon 5V pinge pealt Raspberry Pi-le sobivaks 3.3V pingeks.

5.3 Arduino kood

Autori loodud Arduino kood täitis kolme ülessannet. Esiteks leiti iga sensori kaugus lähimast takistusest, lisa Lisa 1 funktsioon nimega „getSensorReading“. Teiseks vormindati tulemused *JavaScript Object Notation* (edaspidi JSON) kujul. Kolmandaks kirjutati järjestikliidesesse, mis loodi Raspberry Pi ja Arduino vahele, just loodud JSON.

Järjestikliidese ühenduse loomine Arduino poolel toimub klassi „Serial“ funktsiooni „begin“ välja kutsumisega. Ühenduse loomisel kasutatakse boodikiirust (ingl k. *baud rate*) 115 200. JSON objekti koostamist ja järjestikliidesesse kirjutamist on võimalik näha lisas Lisa 1 funktsioonis „loop“, mis on Arduino süsteemides sisseehitatud funktsioon.

5.4 Raspberry Pi kood

Laeva testimise ajal puutus Raspberry Pi kokku veega, ning sai kahjustada. Kahjustuste tulemusel pidid laeva autorid vahetult enne võistlust välja vahetama Raspberry Pi. Autori õnneks ei olnud kontrolleris olev mälukaart veel viga saanud, ning võistluse ajal toimis kõik nagu tarvis. Kui autor mõned kuud hiljem bakalaureuse tööd kirjutama hakkas, oli mälukaart lõpetanud töötamise. Sellest tulenevalt, ei ole bakalaureuse tööga kaasas lõpliku toimivat koodi. Alljärgnevad koodinäited on autoripoolt taasloodud bakalaureuse töö kirjutamise ajal. Kood on kirjutatud kasutades Python programmeerimiskeelt.

5.4.1 Objektorienteeritud programmeerimine

Raspberry Pi koodi kirjutamisel jälgiti võimalikult palju objektorienteeritud programmeerimise (edaspidi OOP) põhimõtteid. OOP on tarkvaraarenduse ja programmeerimise viis, kus tarkvara vaadatakse kui klasside kogumit. Klass (class) kirjeldab mingi asja (objekti) abstraktset (üldist) iseloomu. (Petuhhov) Klass sisaldab endas iseloomustavaid omadusi ja meetodeid. Programmi jooksutamisel luuakse klassi põhiselt objektid. Objekti loomisel määratletakse sellele klassi põhised omadused.

Näitena võib tuua klassi „motor“, millel on omadused nimega: „pin number“, „direction“, „speed“ ja „name“. Programmi käivitamisel luuakse „motor“ klassist objekt, ning sellele antakse vastavalt väärtused: 1, „forward“, „100“ ja „back left“. Klassil on ka võimalik kirjeldada talle kuuluvaid meetodeid ehk funktsioone. Eelnevas näites sobiks klassile „motor“ kaks meetodit: „move“ ja „changeDirection“. Klassipõhised meetodid annavad võimekuse kontrollida kõiki objekte eraldi, ilma mõjutamata teisi sama klassiga objekte.

5.4.2 Sensorite klass

Esimeseks klassiks loodi sensorite klass. See klass hõlmas endas järjestikliidesest tuleva informatsiooni töötlemist, ülejäänud süsteemiga toimimiseks sobivaks muutmist, ning soovitud külje kohase informatsiooni tagastamist. Sensorite klass omas ühte omadust milleks oli pythoni *dictionary,* kus hoiustati sensoritest saadud informatsiooni. Klassi taasloodud struktuur on nähtav lisas Lisa 2.

5.4.3 Mootorite klass

Teise klassina loodi mootori kontrollerite klass. See klass oli palju keerukam kui eelnev, sellest tulenevalt ei ole autor üritanud ka taasluua klassi. Mootorite klassi omadusteks olid:

- suund;
- mootori kiirus;
- GPIO päise number;

- Raspberry Pi põhine pythoni klass pigpio, mis võimaldab ligipääsu GPIO päistele;
- mootori kiiruse koefitsent;

Lisaks oli klassis ära defineeritud ka mootorite algseadistamise funktsioon, mida jooksutati koheselt, kui klassi instants loodi. Mootorite algseadistamise funktsioon kirjutati vastavalt ESC juhistele. Samuti omas klass meetodeid mootori kiiruse muutmiseks ja hädaolukorra puhul mootorite peatamise funktsiooni.

5.4.4 Kontrolleri klass

Kolmandaks klassiks oli süsteemi kontroller. Kuna kontrolleri klassi loomine oli terve arendustegevuse juures kõige rohkem aegavõttev tegevus otsustas autor seda täielikult uuesti mitte luua. Kontrolleri klassi ülesanneteks oli:

- suhelda järjestikliidesega;
- salvestada ja pärida informatsiooni sensorite klassilt;
- kalkuleerida laeva liikumissuund;
- määrata laeva liikumissuuna põhiselt mootorite jõudlust;

Laeva liikumissuuna parandamiseks kasutas autor tarkvaralist proportsionaal-integraal-diferentsiaalregulaatorit (edaspidi PID-regulaator). PID-regulaator koosneb kolmest elemendist. P element, mis on proportsionaalne hetke veaga *„t“*. I element, mis on proportsionaalne veaga kuni hetkeni *„t“*, mida on võimalik nimetada ka „mineviku“ vea koguveaks. Viimaks on D element, mis on proportsionaalne tuletatud veaga hetkel *„t“*, mida on võimalik nimetada „tuleviku“ vea ennustuseks. PID-regulaatorit võib mõista, kui regulaatorit, mis võtab arvesse eelneva, hetke ja tuleviku seisundid. (Araki, kuupäev puudub)

Kokkuvõte

Käesoleva bakalaureuse töö eesmärgiks oli luua täisautonoomne laev. Eesmärgi saavutamiseks koostas autor nimekirja vajaminevatest materjalidest, valmistas 3D mudeleid ja koos kaasautoriga lõi laeva kere. Lisaks kirjutas autor Arduino Mega ja Raspberry Pi mikrokontrolleritel toimivad koodid. Pärast laeva valmimist võeti osa Robotex 2017 Tallink Veerallist, kus jäädi 27 võistlejast 15. kohale[5].

Töö on jaotatud kaheks suureks osaks. Esimeseks osaks on lühike ülevaade liikurmasinate klassifikatsioonidest ja autonoomsete masinate ajaloost. Teise osana annab autor ülevaate laeva loomise protsessist. Samuti toob autor välja erinevaid takistusi, mis ilmnesid laeva ehituse käigus.

Autori arvates sai töö eesmärk täidetud. Kuid kindlasti oleks tarvis veel PID-regulaatorit seadistada kuna võistluse ajal pööras laev liiga jõudsasti.

[5] https://goo.gl/46Xtw4

Kasutatud kirjandus

Araki, M. (kuupäev puudub). *PID CONTROL*. Kasutamise kuupäev: 28. 04 2018. a., allikas http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf

Cunningham, D. (2014). *Waterborne TP SRA: The Autonomous Ship [Powerpoint esitlus]*. Kasutamise kuupäev: 16. 04 2018. a., allikas MUNIN: http://www.unmanned-ship.org/munin/wp-content/uploads/2014/09/MUNIN-WS@SMM-140909-2-Waterbourne-TP-OJR.pdf

Geddes, N. B. (1940). *Magic motorways*. Kasutamise kuupäev: 15. 04 2018. a., allikas archive.org: https://archive.org/details/magicmotorways00geddrich

MUNIN. (kuupäev puudub). Kasutamise kuupäev: 15. 04 2018. a., allikas MUNIN: http://www.unmanned-ship.org/munin/

MUNIN. (kuupäev puudub). *MUNIN's Rationale*. Kasutamise kuupäev: 16. 04 2018. a., allikas MUNIN: http://www.unmanned-ship.org/munin/about/munins-rational/

Petuhhov, I. (kuupäev puudub). *Objekt-orienteeritud programmeerimine (OOP).* Kasutamise kuupäev: 25. 04 2018. a., allikas http://www.cs.tlu.ee/~inga/alg_andm/OOP.pdf

Platoon (automobile). (kuupäev puudub). Kasutamise kuupäev: 15. 04 2018. a., allikas Wikipedia: https://en.wikipedia.org/wiki/Platoon_(automobile)

Robotex. (kuupäev puudub). *Robotex*. Kasutamise kuupäev: 17. 04 2018. a., allikas Robotex: https://robotex.ee/

Rolls-Royce. (2016). Autonomous ships The next step. 2. Kasutamise kuupäev: 16. 04 2018. a., allikas http://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/rr-ship-intel-aawa-8pg.pdf

Rose, M. (18. 02 2016. a.). *Ultrasonic Sensor Holder*. Kasutamise kuupäev: 21. 04 2018. a., allikas GrabCad: https://grabcad.com/library/ultrasonic-sensor-holder-2

Sanjeev, A. (kuupäev puudub). *Maker.pro*. Kasutamise kuupäev: 26. 04 2018. a., allikas https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino

Scheck, W. (15. 11 2017. a.). *Lawrence Sperry: Genius on autopilot*. Kasutamise kuupäev: 15. 04 2018. a., allikas History Net: http://www.historynet.com/lawrence-sperry-autopilot-inventor-and-aviation-innovator.htm

Time magazine. (kuupäev puudub). *Hybrid Car - Best Inventions of 2003 - TIME*. Kasutamise kuupäev: 15. 04 2018. a., allikas TIME: http://content.time.com/time/specials/packages/article/0,28804,1935038_1935083_1935719,00.html

Ultrasonic Ranging Module HC - SR04. (kuupäev puudub). Kasutamise kuupäev: 22. 04 2018. a., allikas Elec Freaks: http://www.micropik.com/PDF/HCSR04.pdf

Weber, M. (08. 05 2014. a.). *Where to? A History of Autonomous Vehicles*. Kasutamise kuupäev: 15. 04 2018. a., allikas Computer History Museum: http://www.computerhistory.org/atchm/where-to-a-history-of-autonomous-vehicles/

Summary

Title: „Creating an Autonomous Boat for the Robotex Competition“

The goal of this thesis was to create a fully autonomous boat. To achive this goal the author created a bill of materials, designed 3D models and with the co-author built the shell fo the boat. In addition the author wrote the code that was run on the Arduino Mega and the Raspberry Pi microcomputers. After the boat was built and tested, the authors of the boat took part in the Robotex 2017 Tallink Waterrally competition where they placed 15th out of 27 competitors[6].

The thesis is divided into two major parts. The first is a short overview of machine classification and the history of autonomous vehicles. The second part describes the build process of the ship.

The author belives that the goal was met. Even though the PID-regulator still needs tuning.

[6] https://goo.gl/46Xtw4

LISAD

Lisa 1. Arduino täiskood

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 250;   //the value is a number of
milliseconds
// defines pins numbers
const int sensorNumber = 8;
const int trigPin[sensorNumber] = {38, 40, 42, 44, 46, 48, 50, 52};
const int echoPin[sensorNumber] = {39, 41, 43, 45, 47, 49, 51, 53};
// defines variables
long duration;
int distance;
void setup() {
  for (int i = 0; i < sensorNumber; i++ ){
    pinMode(trigPin[i], OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin[i], INPUT); // Sets the echoPin as an Input
  }
  Serial.begin(115200); // Starts the serial communication
  startMillis = millis();  //initial start time
}
void loop() {
  currentMillis = millis();  //get the current "time"
  if (currentMillis - startMillis >= period){ //test whether the
period has elapsed{
    String jsonString = "{";
    for (int i = 0; i < sensorNumber; i++ ){
      jsonString += "\"";
      jsonString += i;
      jsonString += "\": \"";
      jsonString += getSensorReading( trigPin[i], echoPin[i] );
      jsonString += "\",";
    }
    jsonString += "}";
    Serial.println(jsonString);
    startMillis = currentMillis;   //IMPORTANT to save the start
time of the current LED brightness
  }
}
float getSensorReading(int trigPin, int echoPin){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in
microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance in cm
  return (duration/2) / 29.1;
}
```

Lisa 2. Raspberry Pi sensorite klass

```
class Sensor(object):

    def __init__(self, filename):
        self.sensorDict = []

    def is_json(self, myjson):
        try:
            json_object = json.loads(myjson)
        except ValueError, e:
            return False
        return json_object

    def set_json(self, myjson):
        jsonData = self.is_json(myjson)
        if jsonData:
            self.sensorDict = jsonData
        else:
            self.sensorDict = False

    def get_left_side(self):
        return {
            "front": self.sensorDict['4'],
            "back": self.sensorDict['5'],
            "middle": self.sensorDict['6']}

    def get_right_side(self):
        return {
            "front": self.sensorDict['1'],
            "back": self.sensorDict['0'],
            "middle": self.sensorDict['7']}

    def get_front_side(self):
        return {
            "left": self.sensorDict['3'],
            "right": self.sensorDict['2']}

    def __str__(self):
        line = ""
        if self.sensorDict:
            for key, value in self.sensorDict.items():
                line += key + ": " + value + ", "
        return line
```

Lisa 3. Raspberry Pi mootorite klass

```
class Motors(object):
    def __init__(self, position, pin, speed=1000):
        self.position = position
        self.speed = speed
        self.pin = pin
        self.pi = pigpio.pi()
        self.coef = 1.0
        self.arm()

    def arm(self):
        self.speed = 1000
        self.pi.set_servo_pulsewidth(self.pin, self.speed)
        print "Arming: %s , %d, %d" % (self.position, self.pin,
self.speed)
        time.sleep(1)

    def set_speed(self):
        self.speed = self.speed * self.coef
        if self.position == "Reverse":
            if self.speed > 1300:
                self.speed = 1300
        else:
            if self.speed > 1200:
                self.speed = 1200
            elif self.speed < 1000:
                self.speed = 1000
        self.pi.set_servo_pulsewidth(self.pin, self.speed)

    def emergency(self):
        self.pi.set_servo_pulsewidth(self.pin, 0)

    def __str__(self):
        return "Motor= %s Pin= %d Speed=%d" % (
            self.position, self.pin, self.speed)
```