Loppuraportti

Radio-ohjattavan pienoismallin mekatroniikan ja ohjelmiston kehitys

10.9.2012 - 14.12.2012

Valvoja: Mika Matilainen

Valter Sandström
Otso Saarentaus

Muutoshistoria

Versionumero	Pvm	Selitys	Tekijä(t)
0.1	10.12.2012	Loppuraportti-pohja	Otso Saarentaus
0.2	12.12.2012	Projektin kulku lisätty	Otso & Valter
0.3	17.12.2012	Viimeistely	Otso & Valter

Alkuperäinen Tehtävänanto

Projektin taustana on Aalto Yliopiston, Insinööritieteden korkeakoulun – koneenrakennustekniikan laitoksen kurssi: Kon-16.4081 Ajoneuvojen tuotekehitys. Kurssilla vuonna 2011 aloitettiin projekti jossa oli tarkoitus rakentaa radio-ohjattavan pienoismalli ajodynamiikan opetuskäyttöön. Projekti jäi kuitenkin kesken ja nyt siihen on tekeillä jatkoprojekti mekaniikan ja mekatroniikan kannalta. Kon-16.4081 kurssin projektiryhmässä on yksi henkilö tekemässä mekatroniikkaa. Nyt tarkoituksena on että tässä sivuprojektissa tehdään tietyt osajärjestelmät Kon-16.4081 kurssin RC-auto projektiin, osajärjestelmät ovat käyttöliittymä sekä ajoneuvonhallintajärjestelmä.

Tehtävänä on toteuttaa ja suunnitella pc-pohjainen käyttöliittymä ja parantaa mekatroniikkaa, jossa päätehtävänä on toteuttaa pienoismallista puuttuvat abs ja esc-järjestelmät.

Projektin yleiset tavoitteet:

- ABS jarrutekniikka toteutettuna mikrokontrolleriin
- Ajoneuvonhallintajärjestelmän (ESC) toteuttaminen mikrokontrolleriin.
- Koko PC käyttöliittymän toteuttaminen, johon kuuluu m.m.:
  - Realaikainen datan näyttäminen loogisesti
  - Ajoneuvon ohjaus
  - Ajodatan keruu ja hallinta

Contents

1 JOHDANTO

1.1 Tausta

Projektin taustana on Aalto Yliopiston, Insinööritieteden korkeakoulun – koneenrakennustekniikan laitoksen kurssi: Kon-16.4081 Ajoneuvojen tuotekehitys. Kurssilla vuonna 2011 aloitettiin projekti jossa oli tarkoitus rakentaa radio-ohjattavan pienoismalli ajodynamiikan opetuskäyttöön. Projekti jäi kuitenkin kesken ja nyt siihen tehtiin jatkoprojekti mekaniikan ja mekatroniikan kannalta. Kon-16.4081 kurssin projektiryhmässä oli yksi henkilö tekemässä mekatroniikkaa. Tämän sivuprojektin tarkoituksena oli tehdä tietyt osajärjestelmät Kon-16.4081 projektiin, osajärjestelmät olivat käyttöliittymä sekä ajoneuvonhallintajärjestelmä. Näin muodostui ohjelmisto ja mekatroniikkaan kolmen hengen ryhmä.

1.2 Tavoitteet

Tavoitteena oli kehittää kurssin Kon-16.4081 radio-ohjattavan pienoismallin mekatroniikkaa ja käyttöliittymää. Tehtäviin kuuluui toteuttaa ja suunnitella pc-pohjainen käyttöliittymä ja parantaa mekatroniikkaa, jossa päätehtävänä oli toteuttaa pienoismallista puuttuvat abs ja esc-järjestelmät.

Projektin yleiset tavoitteet:

- ABS jarrutekniikka toteutettuna mikrokontrolleriin
- Ajoneuvonhallintajärjestelmän (ESC) toteuttaminen mikrokontrolleriin.
- Koko PC käyttöliittymän toteuttaminen, johon kuuluu:
  - Realaikainen datan näyttäminen
  - Ajoneuvon ohjaus
  - Ajodatan keruu ja hallinta
  - Ajoneuvon ohjaus komentotiedostolla (tallennettujen ohjausliikkeiden avulla)

1.3 Riskit

Meidän arviot riskeistä oli

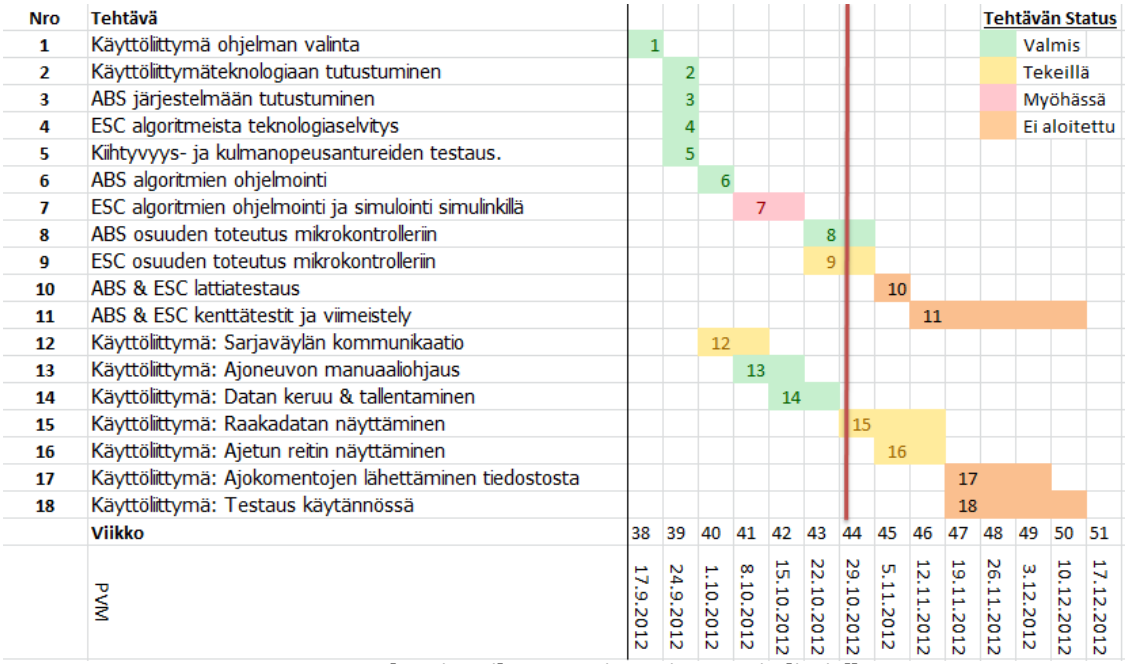
- budjetin ylittyminen
- aikataulun venyminen
- mikrokontrollerin teho abs & esc järjestelmässä
- mikrokontrollerin käyttö samanaikaisesti esc ja abs hallintaan
- radiokommunikoinnin viiveet (radiokommunikointia hallitaan käyttöliittymällä)
- käyttöliittymä-ohjelman rajoitteet (Matlab)
- Pienoismallin mekaniikan kehityksen viivästyminen

2 PROJEKTIN KULKU

2.1 Aikataulu ja Kulku

2.1.1 1 Periodi

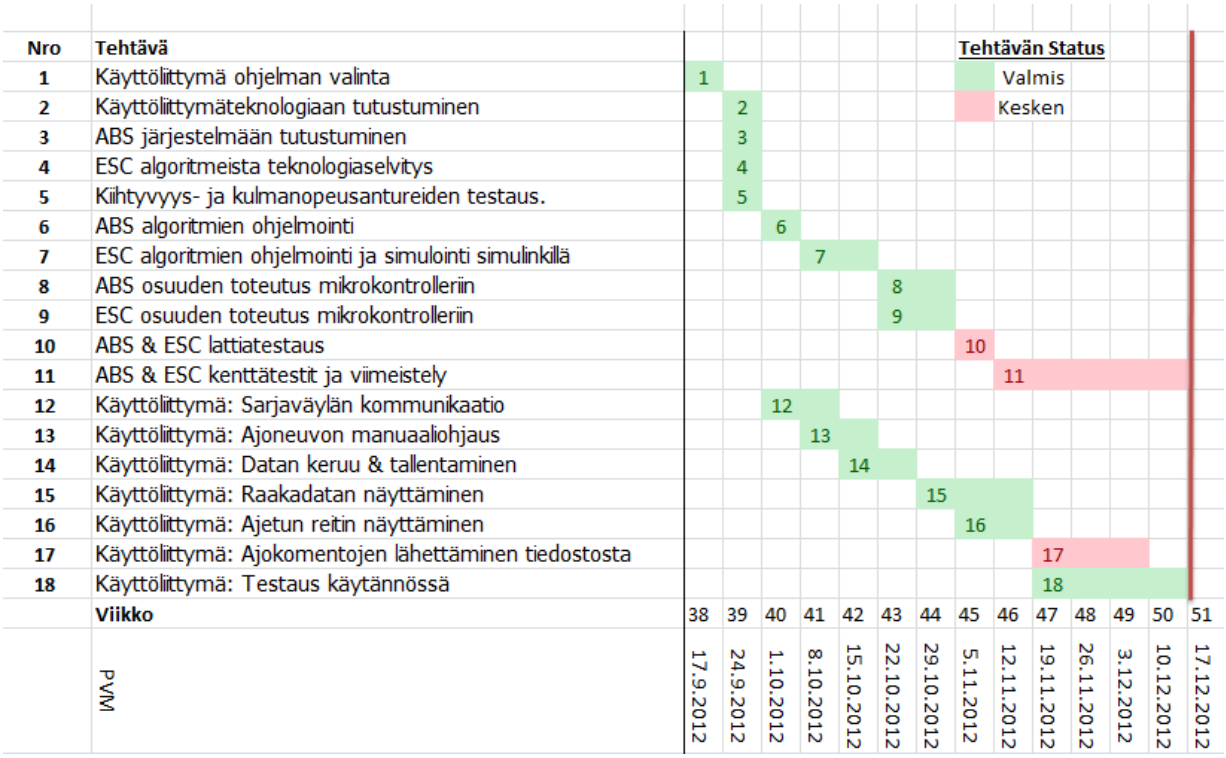
Ensimäisen periodin aikana projekti eteni kohtuullisen hyvin suunnitelman mukaan. Tehtävien tilanne periodin lopussa on näkyvissä kuvassa 1. Kuvasta voidaan todeta tehtävän numero 7, eli ESC algoritmin ohjelmointi ja simulointi simulinkillä, olevan myöhässä. Myöhästymisen syynä oli ABS algoritmin ennustettua vaikeampi toteutus, ja tästä johtuen ESC algoritmi, joka ei ollut yhtä tärkeä toteuttaa, jätettiin myöhemmin toteutettavaksi. Myös numero 12, käyttöliittymän sarjaväylän kommunikaatio, oli tässä vaiheessa hieman aikataulua jäljessä. Syynä oli eteenkin vaikeuksia auton lähettämän datan lukemisessa.



Kuva 1: Tehtävien tilanne ensimmäisen periodin jälkeen

2.1.2 2 Periodi

Toisen periodin lopputilanne on esitetty kuvassa 2. Auton myöhästymisen, sekä muun projektin ajankäytön takia tehtävät numero 10 ja 11 jäivät kesken. Niissä ESC:n prioriteetti oli alhaisempi kuin ABS:än, jonka takia ABS testaukset saatiin tehtyä ajoissa jopa kenttätasolla. Mutta ESC:n testaaminen olisi ollut erittäin vaikeaa koska autoa ei pystytty ajamaan. Tehtävää numero 17 ei toteutettu, koska projekti oli työmäärältään jo venynyt reilusti yli mitä oli suunniteltu. Koimme tärkeämmäksi toteuttaa sulavasti toimivan käyttöliittymän jolla autoa pystytään ajamaan ja testaamaan.



Kuva 2: Tehtävien tilanne toisen periodin jälkeen

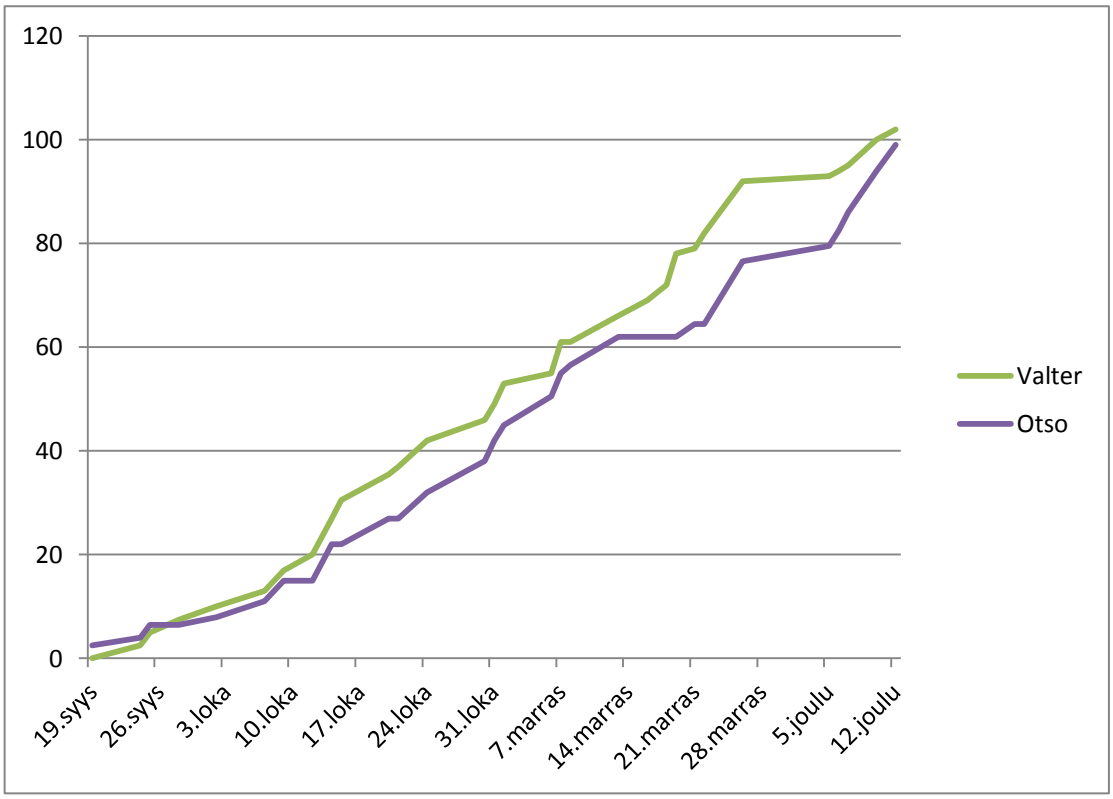
2.2 Kuormitus

Tehtäväkohtainen kuormitusjakauma on esitettu taulukossa 1. Alun perin laskettiin että kuormitus olisi Valterille 70 tuntia ja Otso:lle 71 tuntia. Lopulta varsinaista projektityötä Valter teki 102 tuntia ja Otso 99 tuntia. Arvioon ei kuulu kurssin muita työtunteja, kuten luennot ja esitykset, esityksiin valmistautuminen ja loppuraportin kirjoittaminen.

Taulukko 1. Työmäärä tehtävien mukaan.

Nro	Tehtävä/ Tavoite	Riippuu tehtävästä	Vastuu	Arvioitu työmäärä (h)	Käytetty työmäärä (h)
1	Käyttöliittymä ohjelman valinta, mennäänkö vielä LabView:llä vai otetaanko joku muu?		Valter	1	**2**
2	Käyttöliittymäteknologiaan tutustuminen	1	Valter	2	**5**
3	ABS järjestelmään tutustuminen		Otso	1	**6**
4	ESC algoritmeista teknologiaselvitys		Otso	4	**4**
5	Kiihtyvyys- ja kulmanopeusantureiden testaus.		Valter & Otso	4	**7**
6	ABS algoritmien ohjelmointi	3	Otso	4	**20**
7	ESC algoritmien ohjelmointi	4	Otso	18	**10**
8	ABS osuuden toteutus mikrokontrolleriin ja oheisdatan integraatio	3 & 6	Otso	4	**6**
9	ESC osuuden toteutus mikrokontrolleriin ja oheisdatan integraatio	3 & 7	Otso	10	**4**
10	ABS & ESC lattiatestaus	8 & 9	Otso	8	**16**
11	ABS & ESC kenttätestit ja viimeistely	10	Otso	18	**26**
12	Käyttöliittymä: Sarjaväylän kommunikaatio	2	Valter	6	**21**
13	Käyttöliittymä: Ajoneuvon manuaaliohjaus	12	Valter	4	**7**
14	Käyttöliittymä: Datan keruu & tallentaminen	13	Valter	8	**14**
15	Käyttöliittymä: Raakadatan näyttäminen	13	Valter	16	**14**
16	Käyttöliittymä: Ajetun reitin näyttäminen (hyödyntäen kiihdytysdataa ja/tai renkaiden odometriaa)	15	Valter	10	**11**
17	Käyttöliittymä: Ajokomentojen lähettäminen tiedostosta	13	Valter	14	**1**
18	Käyttöliittymä: Testaus käytännössä	17	Valter	8	**20**

Kumulatiivinen työkuormitus on esitetty kuvassa 3, jossa y akselilla on tuntimäärä ja x-akselilla aika. Kuvaajasta nähdään että 28.11 – 5.12 on ollut hyvin hiljainen jakso projektin suhteen, tässä odottelimme pienoismallin mekaniikan valmistumista. Sitten 5.12 kun pääsimme testaamaan pienoismallia käytännössä, tuli Otso:lle kiire saada ABS ym. mekatroniikkaan liittyvät asiat (mm. kommunikointi) toimimaan oikeassa ajossa.



Kuva 3: Kumulatiivinen työmäärä

2.3 Muutokset

- Käyttöliittymä
  - Todettiin että tarvitaan parametri jolla voidaan säätää maksimi kiihtyvyyttä
  - Käyttöliittymään lisättiin myös uusi välilehti parametrien säätöä varten.
  - Itse parametrien tallennus ja lataus (toi paljon lisätöitä).

- ABS & ESC toteutus
  - ABS toteutus todettiin vaativan simuloinnin, sen sijaan päätettiin että ESC:tä ei simuloida.
  - Lisätöitä tuli myös mm. mikrokontrollerin kommunikoinnin ja parametrien päivityksen kanssa
  - ESC huomattavasti yksinkertaisempi kuin alun perin oletettiin. Ohjelmointityö ei vaatinut niin paljon aikaa ESC:n tekemisessä.
  - Yleistä mekatroniikan korjausta pienoismalliin liittyen. Testausvaiheessa selvisi puutteita elektroniikassa, joita jouduimme korjaamaan yhdessä Ajoneuvotekniikan tuotekehitys kurssin opiskelijan kanssa.

3 TOTEUTUS

3.1 Vastuualueet

Projektin vastuualueet jaettiin seuraavanlaisesti:

Valter Sandström

- Käyttöliittymä
  - Graafinen toteutus
  - Taustakoodi

Otso Saarentaus

- Jarrukontrollerin ohjelmointi
  - ABS ohjelmointi
  - ESC ohjelmointi
- Järjestelmän kommunikointi tallennettavien parametrien osalta

Ajoneuvotekniikan tuotekehityksen kurssin oppilaat

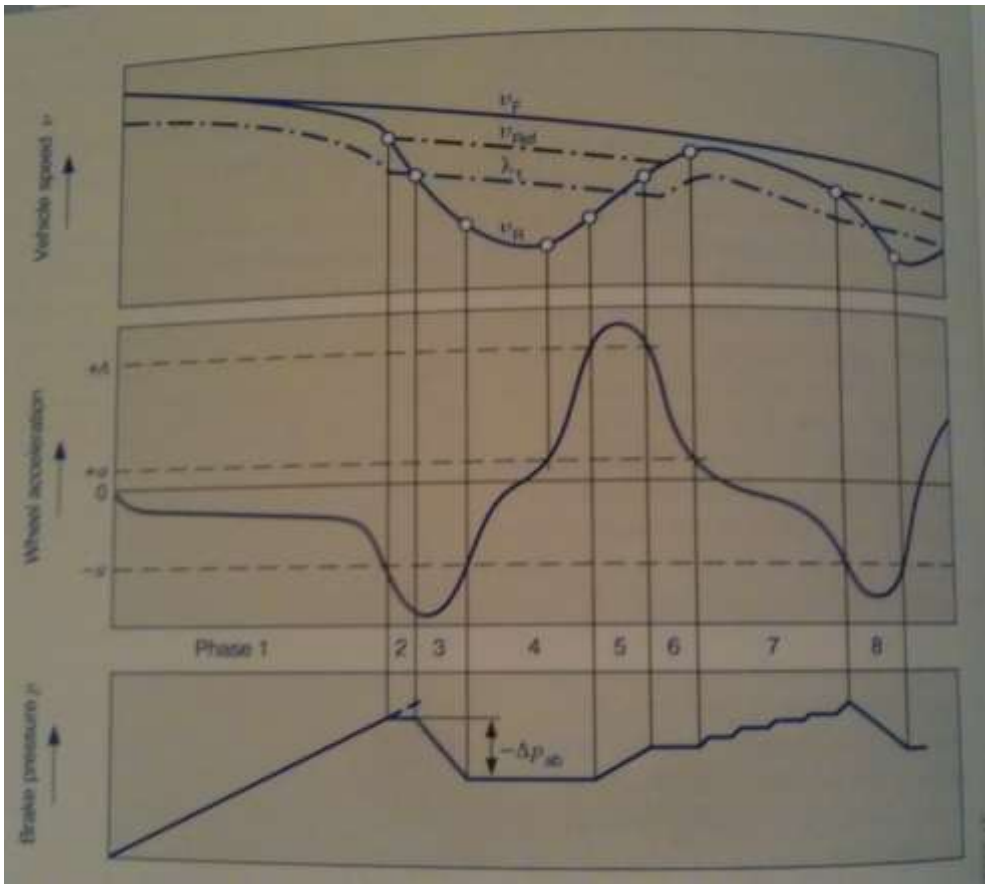
- Mekaniikka
- Sähkö
- Elektroniikka
- Ohjelmointi pää-kontrollerin ja pulssianturi mikrokontrolleri:n osalta

3.2 Mikrokontrolleriohjelmointi

3.2.1 Lukkiutumattomat jarrut

3.2.1.1 Säätöalgoritmi

Lukkiutumattomat jarrut (ABS) pyrkivät ajoneuvon estämään pyörän lukkiutumisen jarrutustilanteessa. Algoritmin herätteenä toimii kuskin antama jarrutusvoima ja säädettävänä arvona on pyörän kulmakiihtyvyys. Mitään muita arvoja ei käytetty simuloinnissa taikka toteutuksessa. Todellisuudessa täydellinen ABS algoritmi sisältää myös ajoneuvon nopeuden estimoinnin. Tätä tarvitaan algrotimin ensimmäisessä vaiheessa. Se laskee kuvassa 4 esitetyn alimman kuvaajan vaiheen kaksi pituuden. Koska nopeuden tarkka estimointi on hankalaa ja vaihe kaksi vaikuttaa jarrutusmatkaan hyvin vähän, niin päätimme jättää sen pois algoritmista. Vaihe kaksi on käytössä vain jarrutustapahtuman alkaessa. [1]



Kuva 4. ABS-säätöalgoritmin toiminta, ote kirjasta Safety,Comfort and Convenience Systems, Robert Bosch GmbH

Varsinainen algortimi joka estää pyörän lukkiutumisen koko jarrutusmatkan ajalla toimii vaiheiden 3 ja 7 välillä. Taulukossa 2 on selitetty miten algoritmi väleillä 3 - 7 toimii. Se on myös sama algoritmi jolla teimme ensimmäiset testit. Keskeisintä ABS järjestelmän kalibroinnissa on sopivien +A, +a ja -a arvojen löytäminen. [1]

Taulukko 2. ABS algoritmin toiminta, lähde: [1]

Vaihe	Tehtävä
3	Kun vaiheessa 2 on käynnistetty säätöalgoritmi kulmakiihtyvyyden laskiessa tarpeeksi pieneksi (kuva 4, keskimmäinen kuvaaja). Niin vaiheessa 3 lähdetään tiputtamaan jarrutusvoimaa. Voimaa tiputetaan niin kauan, kunnes kulmakiihtyvyys nousee samalle tasolle joka aloitti abs-säädön.
4	Jarrutusvoimaa pidetään tasaisena kunnes kulmakiihtyvyys on noussut määritellylle ylätasolle. Pyörät eivät siis pyöri täysin vapaasti, mutta eivät myöskään jarruta täydellä teholla.
5	Kun kulmakiihtyvyys kiihtyy taas liian nopeasti, aletaan jarrutusvoimaa nostamaan, tämä saa aikaan kulmakiihtyvyyden laskemisen ja pyörä alkaa taas hidastumaan. Seuraavaan vaiheeseen siirrytään kun kulmakiihtyvyys alittaa ylärajan (katso kuva 4.

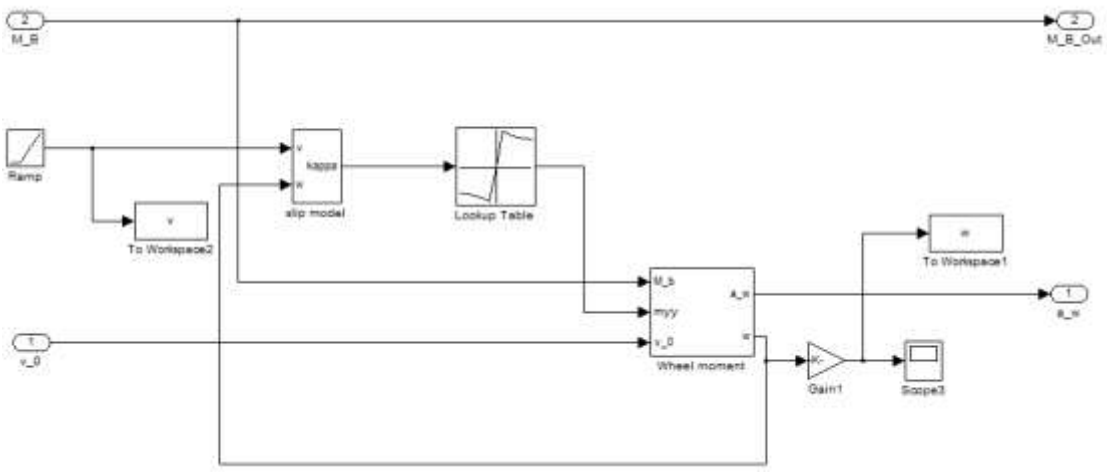
	keskimmäinen kuvaaja)
6	Jarrutusvoiman nosto aiheutti kulmakiihtyvyyden laskemisen. Tässä vaiheessa pidetään sama jarrutusvoima, kunnes se on laskenut alle määritellyn keskitason. Kaikki 3 määriteltyä tasoa kalibroidaan erikseen kokeellisesti.
7	Kun kulmakiihtyvyys on alle keskitason, käynnistyy 7 vaihe. Tällä tasolla jarrutusvoimaa nostetaan portaittain, kunnes saavutaan samaan tilanteeseen mistä algoritmi alkoi. Eli kulmakiihtyvyys on laskenut alle alatason. Tällöin algoritmi jatkaa taas vaiheesta 3.

3.2.1.2 Simulointi

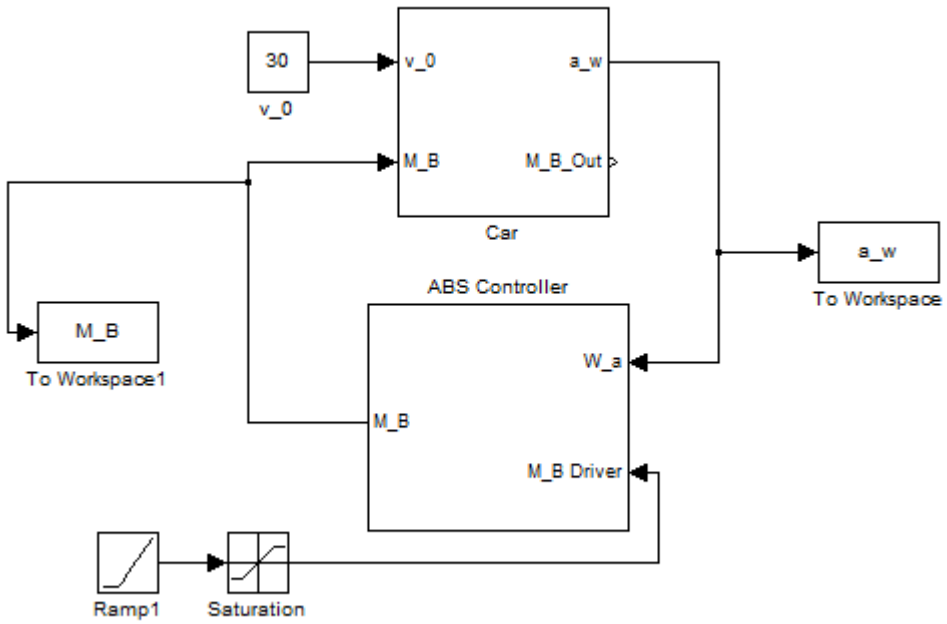
Simuloinnissa käytettiin Matlab Simulink ohjelmistoa, ajoneuvon malli oli yksinkertaistettu yhden pyörän malliksi. Jota voidaan kuvata kaavalla:

$$M_B - r_k * F_x = J_w \ddot{\psi}. \text{ [2]} \quad (1)$$

Jossa M_B on jarruttava momentti, r_k on pyörän dynaaminen vierintäsäde, F_x on pitkittäisvoima, J_w on pyörän hitausmomentti ja $\ddot{\psi}$ on pyörän kulmakiihtyvyys. Pyörän simulointimalli on esitettynä kuvassa 5, koko simulointimalli on kuvassa 6. Simuloinnin tuloksista keskustellaan luvussa 4.



Kuva 5. Yhden pyöränmalli.



Kuva 6. ABS-järjestelmän simulointi.

3.2.1.3 Ensimmäinen toteutus

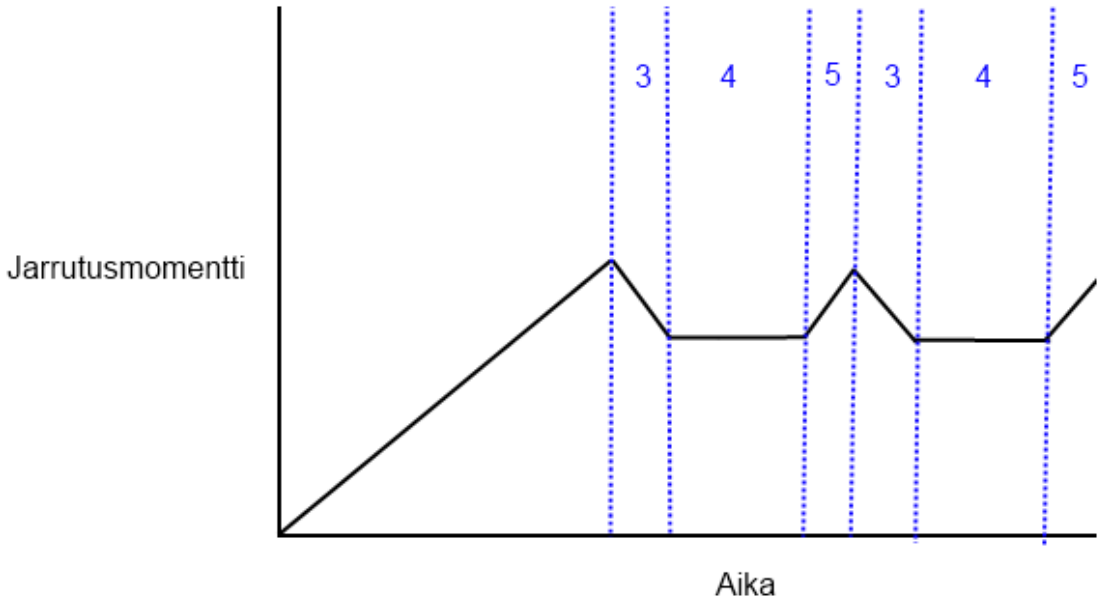
ABS toteutettiin Radio-ohjattavan pienoismallin jarrukontrolleriin. Jarrusäätimenä toimii Arduinon Mega 2560 R3 mikrokontrolleri, elektroniikkaa havainnollistava kaavio on esitetty liitteessä 3. Mikrokontrolleri ohjaa suoraan neljää jarruservoa, servonohjaukseen tarvittava koodi on toteutettu ajoneuvotekniikan tuotekehitysryhmän puolesta.

ABS-algoritmi perustuu luvussa 3.2.1.1 logiikkaan, eli vaiheesta 7 hypätään suoraan vaiheeseen 3. Mikrokontrollerin säätösilmukka pyötii 100Hz:in taajuudella, joten täydellinen säätökierros voidaan toteuttaa 100 millisekunnin aikana. Ensimmäisen vaiheen ABS-ohjelmakoodi on liitteessä 4. ABS:än kalibrointiin tarvittavat parametrit ovat liitteessä 5.

Pyörien pyörimisnopeudet saadaan toiselta mikrokontrollerilta (Teensy), joten niiden laskemiseen ei mene laskentatehoa. Teensy hoitaa myös pyörimisnopeuksien alipäästösuodatuksen. Kuitenkin kokeissa havaitaan että ABS-algoritmi on liian raskas neljää pyörää säädettäessä. Tämän takia varsinaista ajoa varten tehtiin yksinkertaistettu algoritmi.

3.2.1.4 Korjattu toteutus

Mikrokontrollerin rajoitetun laskentatehon takia päädyimme tiputtamaan pois vaiheet 6 ja 7, siten että vaihe viisi nostaa jarrutusvoimaa kunnes alin kulmakiihtyvyysraja (-a) alitetaan. Jolloin algoritmi siirtyy suoraan vaiheeseen 3. Kuva 7 esittää jarrutustapahtumaa yksinkertaistetulla algoritmilla.



Kuva 7. Karsittu ABS-säätö.

3.2.2 Ajoneuvonhallintajärjestelmä

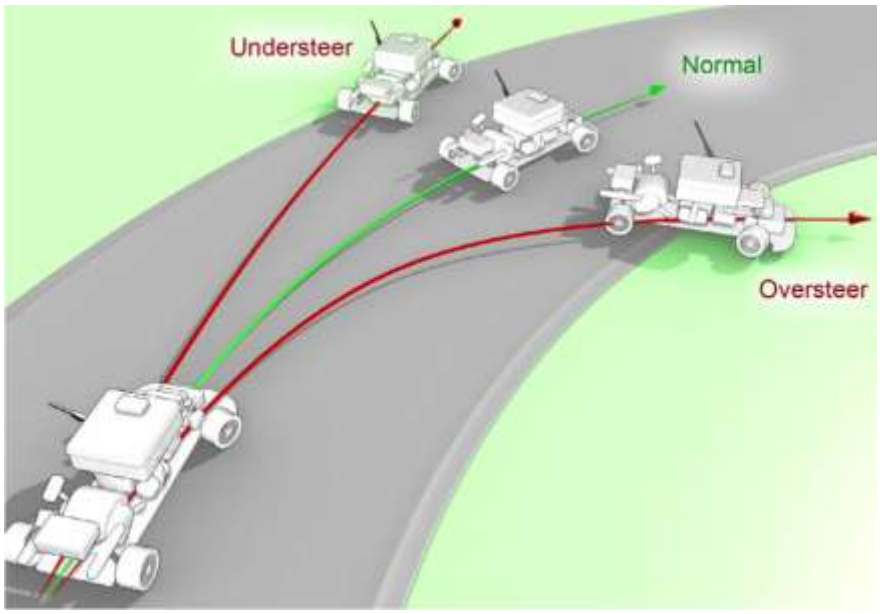
3.2.2.1 Säätöalgoritmi

Ajoneuvohallintajärjestelmän (ESC) tehtävänä on kääntää autoa kuljettajan haluamaan suuntaan. Ajoneuvonhallintajärjestelmä tarvitsee useita antureita:

- Ratin kääntökulma (Kuljettajan haluama ajosuunta)
- Pyörien kulmanopeudet (Ajoneuvon estimoitu etenemisnopeus)
- Kiihtyvyysanturi (Ajoneuvon estimoitu sivuttaisnopeus)
- Gyro (Ajoneuvon pyörimisnopeus)

Suurin haaste ESC säädössä on todellisen etenemisnopeuden ja sivuttaisnopeuden laskenta. Siihen löytyy patentoituja ja autonvalmistajien salassapidettyjä keinoja, lähteestä [3] löytyy osa teoriaa Fordin käyttämään menetelmään.

Itse ESC-algoritmi on yksinkertainen, se jakaantuu kahteen vaiheeseen, kulmanopeuteen perustuvaan säätöön ja sortokulmaan perustuvaan säätöön. Sortokulma on sivuttaisnopeus jaettuna etenemisnopeus (V_y /V_x). Sitä käytetään kun auto on niin pahassa sivuluisussa että kulmanopeus ei anna luotettavaa arvoa, tämä säätö kytkeytyy yleensä päälle jos sortokulma on yli 10 astetta. [4] Normaalissa tilassa ESC säätää kulmanopeuden avulla, jolloin ajoneuvon ali- tai yliohjautumista pyritään korjaamaan. Ali- ja yliohjautuminen on esitetty kuvassa 8.



Kuva 8. Ajoneuvon aliohjautuminen (understeer) ja yliohjautuminen (oversteer).

Säätö toimii aina seuraavanlaisesti:

1) Määritellään referenssi kulmanopeus (tai sortokulma jos todellinen sortokulma > 10 astetta), tämä onnistuu ajoneuvon dynamiikkaan perustuvan kaavan avulla. Kaavan ratkaisemiseen tarvitaan painopiste, painospiteen etäisyys etu ja takaakseleista, etu ja taka-akselin kaartojäykkyydet sekä ajonopeus.

2) Päätetään jarrutettava pyörä (lähteestä 5):

```
if(abs(ω)) ≥ (abs(ωref)·(1/S))  // Yliohjaava
        if(ω) > 0
                Jarrutetaan vasenta etupyörää
        else
                Jarrutetaan vasenta takapyörää
elseif(abs(ω)) < (abs(ωref)·S) // Aliohjaava
        if(ω)>0
                Jarrutetaan oikeaa etupyörää
        else
                 Jarrutetaan oikeaa takapyörää
```

3) aloitetaan säätö alusta

Toisen vaiheen S parametri on herkkyyskerroin, jolla säädetään kuinka herkästi ESC:n tulisi puuttua ajoon. w on kulmanopeus, mutta mikäli sortokulma on kasvanut liian suureksi, ei kulmanopeuden avulla voida säätää. Silloin käytetään samaa säätöä, mutta vertaillaan laskettua kulmanopeutta referenssikulmanopeuteen.

3.2.2.2 Toteutus

Toteutus perustuu suoraan edellä olevaan kappaleeseen. Toteutuksessa ajoneuvon etenemisnopeus lasketaan pyörien pyörimisnopeusantureiden avulla niin, että huomioidaan jos jotakin pyörää on jarrutettu. Ja otetaan vastakkainen diagonaali (esim. etu oikea ja vasen taka), mikäli kumpaakaan näistä ei olla jarrutettu juuri edellisellä silmukan suorituskerralla, otetaan näistä nopeamman pyörän nopeus ja käytetään sitä laskemaan ajoneuvon nopeus. Mikäli kaikkia pyöriä on jarruteltu, lasketaan nopeus kaikkien pyörien keskiarvosta.

Kulmanopeus luetaan suoraan gyrosta, mutta ajoneuvon sivusuuntainen nopeus integroidaan kiihtyvyysanturin y-akselista. Integrointi aloitetaan heti kun kuljettaja kääntää rattia, ja lopetetaan kun ratin kääntö loppuu. Tällä tavoin pyritään helpottamaan kontrollerin kuormitusta. Ratin kääntökulma saadaan potentiometristä joka on kiinnitetty ohjausservoon, pääkontrolleri lähettää ohjauskulman jarrukontrollerille, joten se helpottaa hieman jarrukontrollerin kuormitusta.

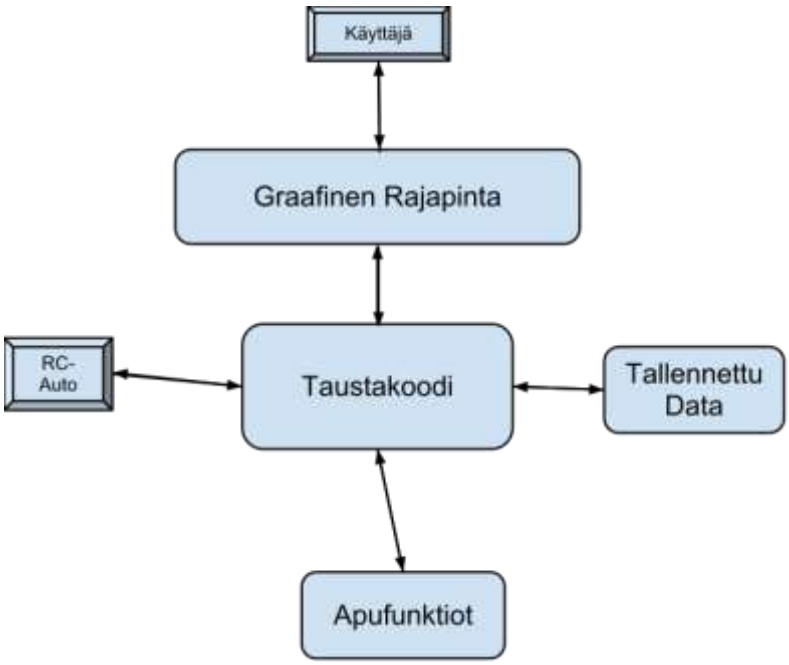
Valitettavasti emme koskaan ehtineet kokeilla esc säätöä todellisuudessa. ESC:hen liittyvät parametrit on taulukossa, liitteessä 5.

3.3 Käyttöliittymäohjelmointi

Tavoitteena oli rakentaa käyttöliittymä jolla ajoneuvoa voitaisiin ohjata reaaliajassa tietokoneen kautta. Tämän lisäksi käyttöliittymä tulisi näyttää ajodataa reaaliajassa sekä käyttäjän halutessa tallentaa tämän data ajon jälkeen. Lopuksi ajoneuvoa tulisi myös olla mahdollista ohjata valmiin komentotiedoston kautta.

Ensinmäinen haaste oli valita ohjelma jonka avulla käyttöliittymää rakennettaisiin. Ensinmäinen vaihtoehto oli LabView, jota oltiin käytetty alkuperäisen auton ohjaukseen. Muita hyötyjä LabView ohjelmassa oli tämän lisäksi että sillä on helppo rakentaa selkeitä ja tyylikkäitä rajapintoja datan näyttämiseen. Viime vuoden projektin aikana oli kuitenkin todettu sen taustakoodin tekemisen olevan hieman hankalaa, tämän lisäksi realiaikainen ohjaus ei ikinä saatu toimimaan kovin hyvin. LabViewin vaihtoehtona oli MATLAB, jonka etuihin kuului että se ohjelmana oli ennestään tuttu käyttöliittymän tekijälle. Käyttöliittymän tekeminen MATLAB:in avulla oli kuitenkin ennestään tuntematonta. Valinta päättyi lopulta MATLAB:iin, koska todettiin sen todennäköisesti toimivan LabViewtä vikkelämmin. Tämän lisäksi autoa simuloitaisiin tulevaisuudessa todennäköisesti MATLAB:issa jolloin opiskelijoille olisi mukavampaa kun sekä ajot että simuloinnit olisi mahdollista suorittaa samassa ohjelmassa.

Käyttöliittymän graafinen rajapinta rakennettiin matlabin GUIDE työkalun avulla. Kuvassa 9 käyttöliittymän ohjelmistorakenne on havainnollistettu.



Kuva 9: Käyttöliittymäohjelman eri osat sekä niiden riippuvuudet.

Ohjelman toiminta on varsin yksinkertainen. Käyttäjä on vuorovaikutuksessa ohjelman kanssa graafisen rajapinnan kautta. Rajapinta kommunikoi taustakoodin kanssa aina käyttäjän antaessa jonkin komennon, tai kun kuuvaajat tulisi päivittää uusimmalla ajodatalla. Taustakoodi hoitaa kaikki ohjelman varsinaiset toiminnot. Kommunikointi auton kanssa, ajodatan näyttäminen ja tallentaminen, sekä auton parametrien lataus autolle tai autosta. Ohjelmaa ajetaan kahdessa säikeissä jotta käyttöliittymä pysyisi myös käyttäjän käytettävissä ajon aikana. Apufunktiot ovat ryhmä pieniä funktioita jolla ohjelmakoodin toistuvuutta on vähennetty. Kaikki data tallennetaan MATLAB:in omina .mat-tiedostoina.

3.4 Haasteet

3.4.1 Jarruohjain

Suurimmat haasteet jarrujen ohjelminnissa oli että ABS simulointi oli oletettua vaikeampaa. Ongelmia tuli mm. yhden pyörän mallin epälineaarisuudesta. Lopulta simulointi saatiin onnistumaan, mutta kuitenkin ESC:n teko myöhästy koska ABS:ää pidettiin tärkempänä. ABS:än kanssa ongelmat jatkuvat koska täydellinen ABS ei toimi jarruohjaimella, epäilemme että servojen ohjaus ei toimi kunnolla kun ohjelma käy liian raskaaksi. Tähän ongelmaan ei löydetty täydellistä ratkaisua, vaan teimme yksinkertaistetun ABS-algoritmin jota mikrokontrolleri jaksoi pyörittää.

ESC:n testaus osoittautui haasteelliseksi, koska pienoismallin mekaniikka valmistui myöhässä ja muutenkin meidän aikataulu ei riittänyt ESC:n testaamiseen. Lisäksi haasteita toi se, että ei ollut kunnolla tilaa testata ESC:tä, olosuhteet Suomessa oli jo varsin lumiset ja autolla ei voitu ajaa ulkona, koska elektroniikka ei ollut tarpeeksi hyvin suojattua.

Parametrien haku osoittautui hankalaksi. Lähinnä koska pääkontrollerin sarajportin puskuri täyttyi liian helposti. Ongelma ratkaistiin osittain, lähettämällä parametrit sykleissä, jolloin puskuri ei ehtinyt täyttyä, vaan pääkontrolleri ehti lähettää parametrit eteenpäin PC:lle.

3.4.2 Käyttöliittymä

Käyttöliittymän toteutuksen suurimmat haasteet osoittautui olevan kommunikointi auton kanssa sekä reaaliajassa tapahtuvan ajon aikana ilmestyvät käyttöliittymän jäätymiset. Auton kanssa kommunikointi saatiin lopulta toimimaan hyvin ajon aikana, mutta lopullisessa ohjelman versiossa ilmastyy vielä hieman vaikeuksia kun auton parametreja halutaan tuoda autosta. Parametrien tuomiseen menee joskus useampikin yritys. Ajon aikana tapahtuvat käyttöliittymän jäätymiset ilmestyivät kun ohjelman ajo sykli asetettiin liian lyhyeeksi. Toisaalta, pitkä sykli aiheutti viiveitä ajossa. Lopulta tämäkin ongelma ratkaistiin, muun muassa tekemällä kommunikaatiosta asynkonisen.

Toinen haaste, joka seurasi kommunikointivaikeuksista, oli yleinen ajan puute. Tästä johtuen moni asia ei ehditty toteuttaa ja testata kunnolla. Tästä syystä esimerkiksi auton ajaminen komentotiedostosta päätettiin jättää kokonaan tekemättä. Virheitä löytyy kuitenkin myös toteutetuista osista: Loppuraportin kirjoitushetkellä huomattiin että auton paikannus kiihtyvyysanturista on laskettu väärin. Paikka lasketaan ainoastaan kiihtyvyydestä, eikä aijempaa nopeutta oteta huomioon. Tällöin esimerkiksi auto pysähtyy reittikartassa heti mikäli se ei kiihdytä mihinkään suuntaan.

3.4.3 Yleiset haasteet

Yleisenä haasteena koko projektissa oli kommunikointi kolmannen osapuolen kanssa, tämä aiheutti hitautta sekä käyttöliittymä että jarru-ohjelmointi puolella. Mikäli testauksissa ilmeni ongelmia elektroniikassa, jouduimme odottamaan että elektroniikkavastaava korjasi piirilevyä. Toinen haaste oli tila, testaukseen ei löytynyt kunnollista tilaa jossa pienoismallilla pystyi ajamaan.

4 TULOKSET

4.1 Radio-ohjattava pienoismalli

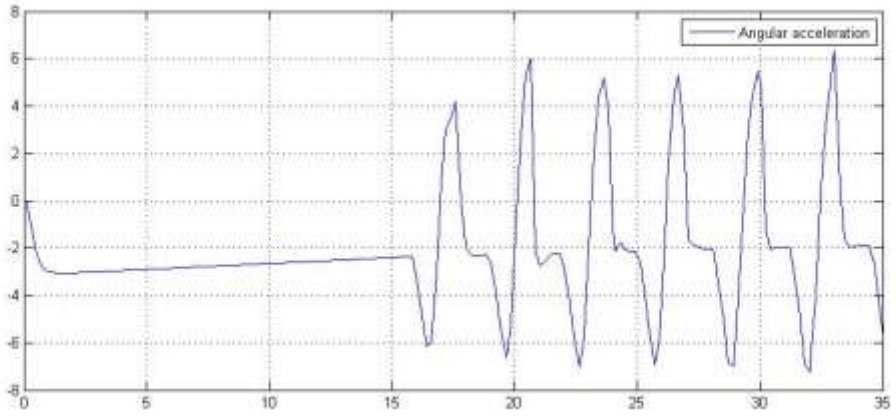
Pienoismalli oli vielä projektiin mukaan mentäessä (1 periodi) aikataulussa. Mutta 2. periodin alussa mekaniikkapuoli oli pahasti myöhässä. Kuitenkin pienoismalli saatiin ajokuntoon projektin lopussa. Ainoat varsinaiset ongelmat mitä vielä jäi olivat kytkimen luistaminen, liian pieni kulmavaihde (vaihde kesti kuitenkin hyvin meidän testit pienillä ajonopeuksilla) sekä akkusäädinpiirin rajoitin. Näistä ongelmista huolimatta autolla pystyttiin ajamaan sisätiloissa täydellisesti kokoonpantuna 12.12. Kuva 10 esittää valmista pienoismallia.

Kuva 10. Radio-ohjattava pienoismalli valmiina.

4.2 Lukkiutumattomat jarrut

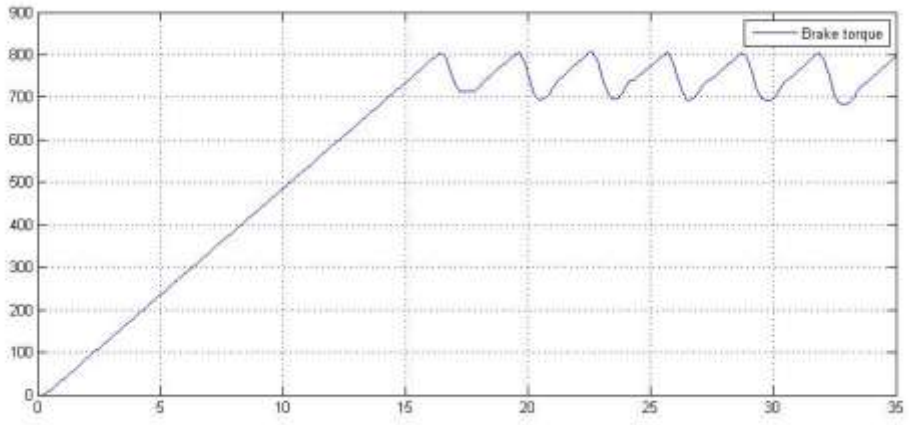
4.2.1 Simulointi

Simuloinnilla onnistuimme säätämään sopivat alkuarvot ABS-algoritmille. Hankalaa simuloinnista teki se, että piti löytää sopiva kulmakerroin pyörän vauhdin vähentymiselle. Kuvassa 11 on esitettynä pyörän kulmakiihtyvyys. Kiihtyvyyden nousu äkillisesti minimiarvostaan johtuu siitä että pyörän dynamiikka ei sisältänyt pyörän realistista hidastuvuutta jarrumomentin funktiona, jolloin vaihe 4 jossa jarrutusvoimaa pidettiin vakiona nosti kulmakiihtyvyyden äkillisesti.



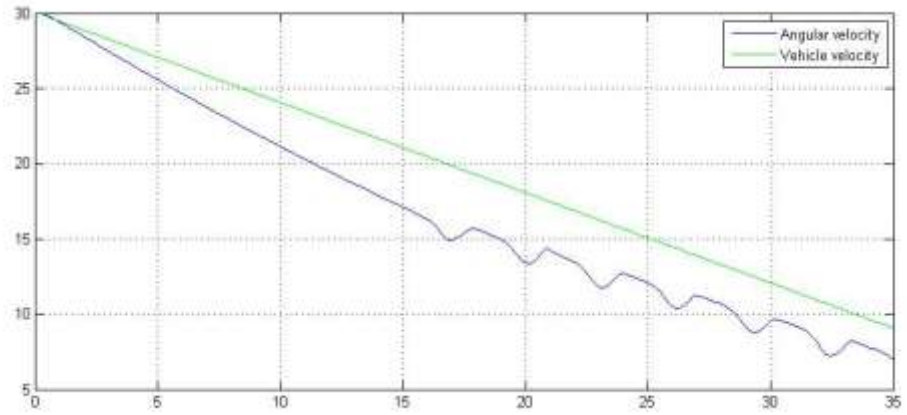
Kuva 11. Simuloitu kulmakiihtyvyys.

Kuvassa 12 on esitetty samasta simuloinnista jarrutusmomentti. Kuvaajasta nähdään selvästi 3, 4 ja 5 vaihe. Vaihe 6 jää simuloinnissa erittäin lyhyeksi ja 7 ei nosta jarrutusmomentti portaittain, vaan lyhyen vakio-jarrutusmomenttiajan takia se nousee vain loivemmin. Kuitenkin näitä vaiheita vastaava kulmakiihtyvyys käyttäytyy oikein kuvaajassa joka on esitetty kuvassa 13.



Kuva 12. Jarrutusmomentti

Skaalattu ajoneuvon nopeus ja pyörän kulmanopeus näkyvät hyvin kuvassa 13, tässä nähdään juuri oikea käyttäytyminen. Eli pyörän kulmanopeuden tulee pysytellä alle ajoneuvon nopeuden ja saada äkillisiä hidastuvuuksia jolloin abs vapauttaa pyörän. Simulointi ei ole tehty loppuun asti, koska mentäessä pieniin nopeuksiin järjestelmä menee epästabiiliksi kitkakertoimen toiminnan takia. Vastaavaa käyttäytymistä kun kuvassa, pyritään tavoittelemaan lattiatesteissä.



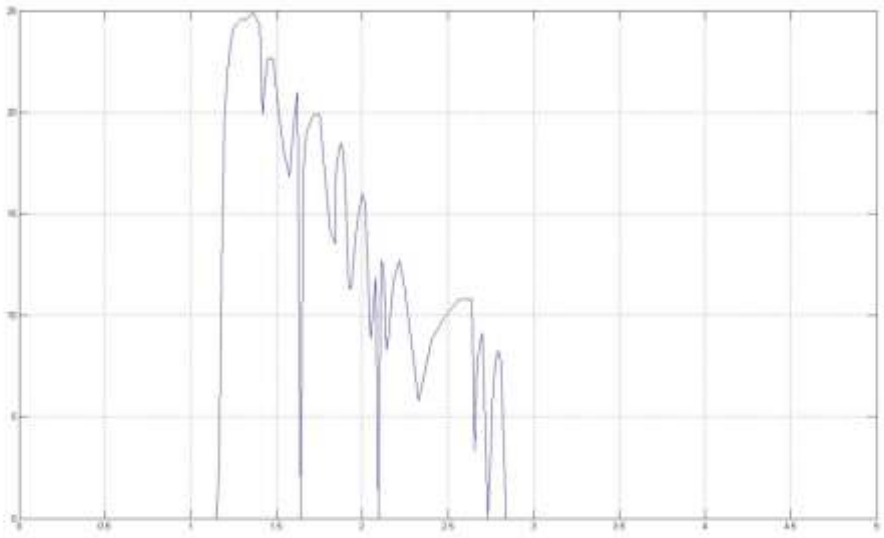
Kuva 13. Ajonopeus ja pyörän kulmakiihtyvyys.

4.2.2 Yhden pyörän testi

Täydellistä ABS-algoritmia kokeiltiin ensiksi yhdellä pyörällä, testaukseen tarvittiin pienoismallin lisäksi akkuporakone. Akkuporakone kiinnitettiin pyörään jolloin sillä voitiin pyörittää pyörää. Akkuporakoneen lukko toimi säädettävänä momentinrajoittimena, sillä voitiin säätää sitä, kuinka helposti pyörä menee lukkoon. Kuvassa 14 näkyy käytetty testipenkki. Kokeen tulos on esitetty kuvassa 15, kuvaajassa on pyörän etenemisnopeus (km/h).

Kuva 14. ABS testipenkki.

.

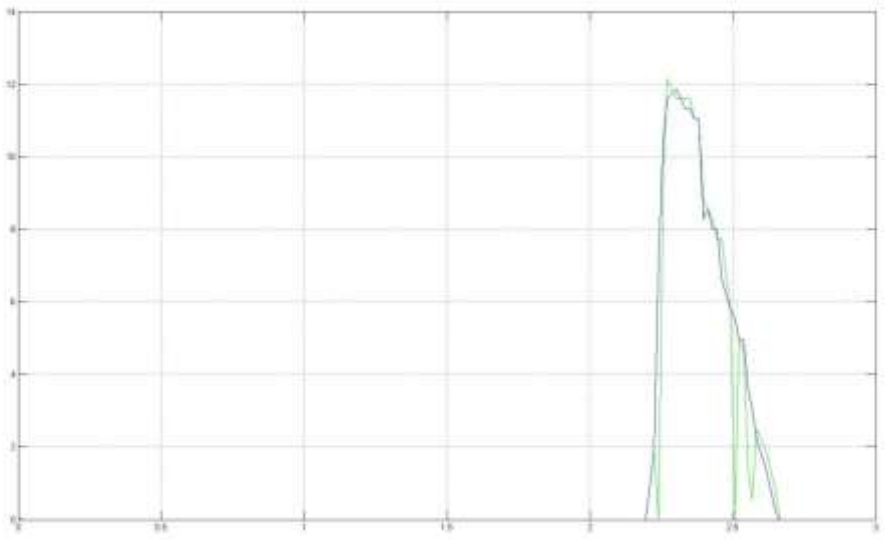


Kuva 15. Yhden pyörän abs-koe.

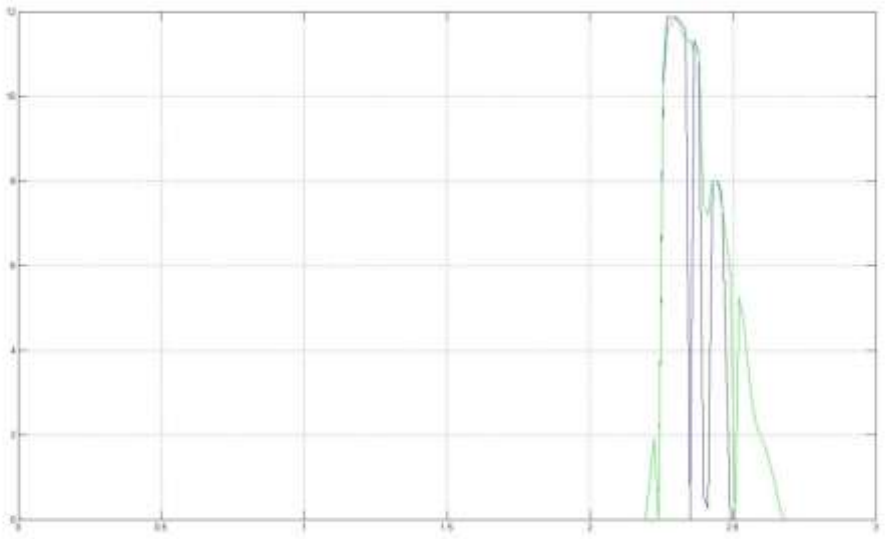
Kuvaajasta nähdään että abs algoritmi toimii varsin hyvin, vain kahteen otteeseen pyörä on lukittautumassa kokonaan. Säädössä on kuitenkin yksi ongelma, servo toimii liian hitaasti ja on vaikea saada säätö vain pienelle servon alueelle. Eli joskus algoritmi vapauttaa pyörää liikaa. Tätä ongelmaa emme saaneet korjattua yhden pyörän kokeessa.

4.2.3 Ajokoe

Ajokokeessa testattiin vain yksinkertaista ABS-algoritmia, koska yhden pyörän testissä selvisi että mikrokontrolleri ei jaksa pyörittää täydellistä ABS-algoritmia. Ongelmasta on kerrottu enemmän kappaleessa 3.4. Ajokokeessa ongelmana oli saada pienoismallille tarpeeksi vauhtia, pienten kulmakiihtyvyyksien takia ABS lukittaa pöyrät pienillä ajonopeuksilla. Tämänkaltainen käyttäytyminen onkin juuri oikein, koska pienillä ajonopeuksilla on tehokkainta lukittaa pyörät jarrutuksen aikana. Saimme ajoneuvolle riittävän vauhdin, jolloin huomattiin silmämääräisestikkin että ABS toimi. Kuvissa 16 ja 17 on esitetty kokeesta saatu jokaisen pyörän nopeus. Ensin autolle annetaan työntö, joka aiheuttaa nopeuden äkillisen nousun.



kuva 16. Etupyörien nopeudet (sininen = vasen, vihr. = oikea).



kuva 17. Takapyörien nopeudet (sininen = vasen, vihr. = oikea).

Kuvaajista nähdään että ABS vapauttaa jokaisen pyörän kun ne ovat lähellä lukkiutua, etenkin etupyörissä ABS näyttää toimivan erityisen hyvin. Takapyörät näyttävät lukkiutuvan helpommin, tämä johtuu siitä että niihin ohjautuu liikaa jarrutustehoa. Pienoismalli on nimittäin erittäin etupainoinen, jolloin etujarrut rasittuu enemmän. Kokeessa jarrut oli kalibroitu jarruttamaan yhtä paljon edessä ja takana. Tästä huolimatta ABS näyttää käyttäytyvän oikein.

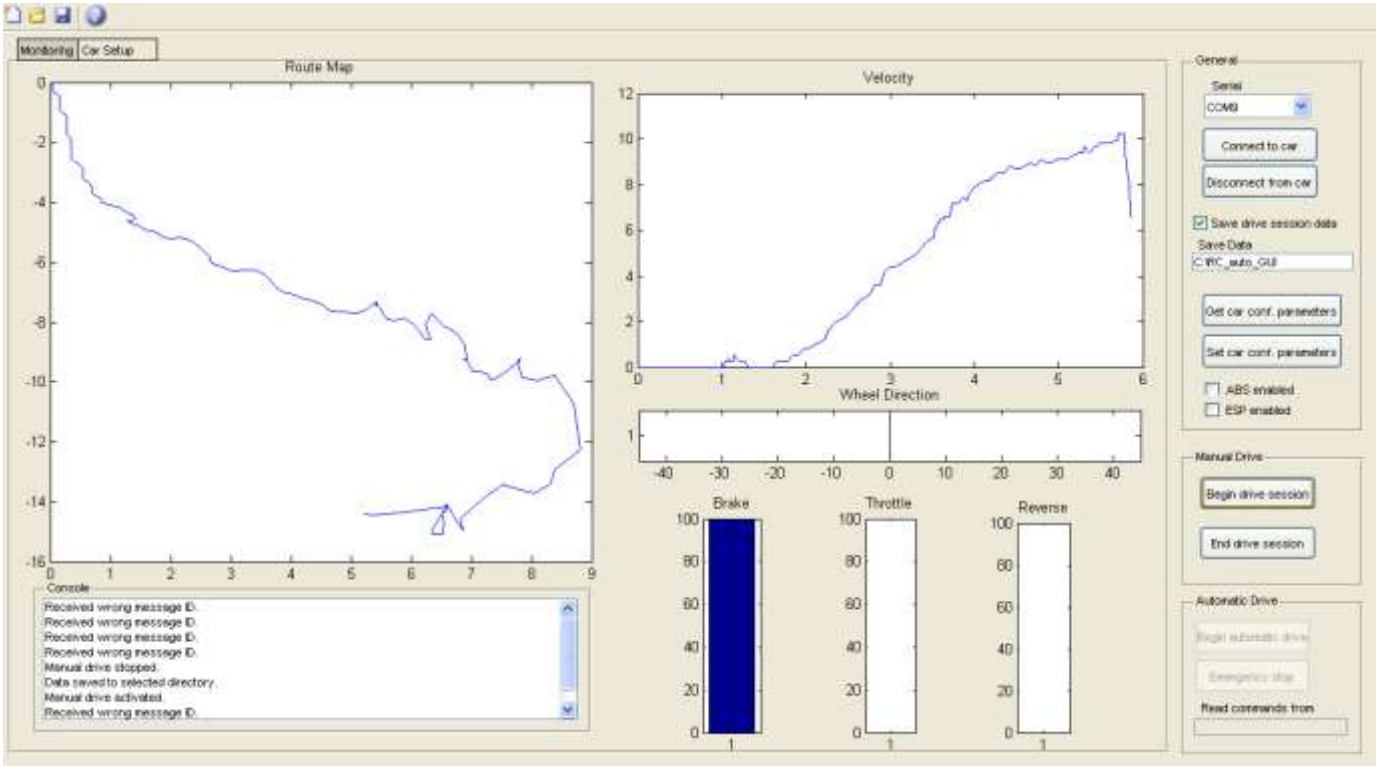
4.3 Käyttöliittymä

4.3.1 Käyttöliittymän esittely

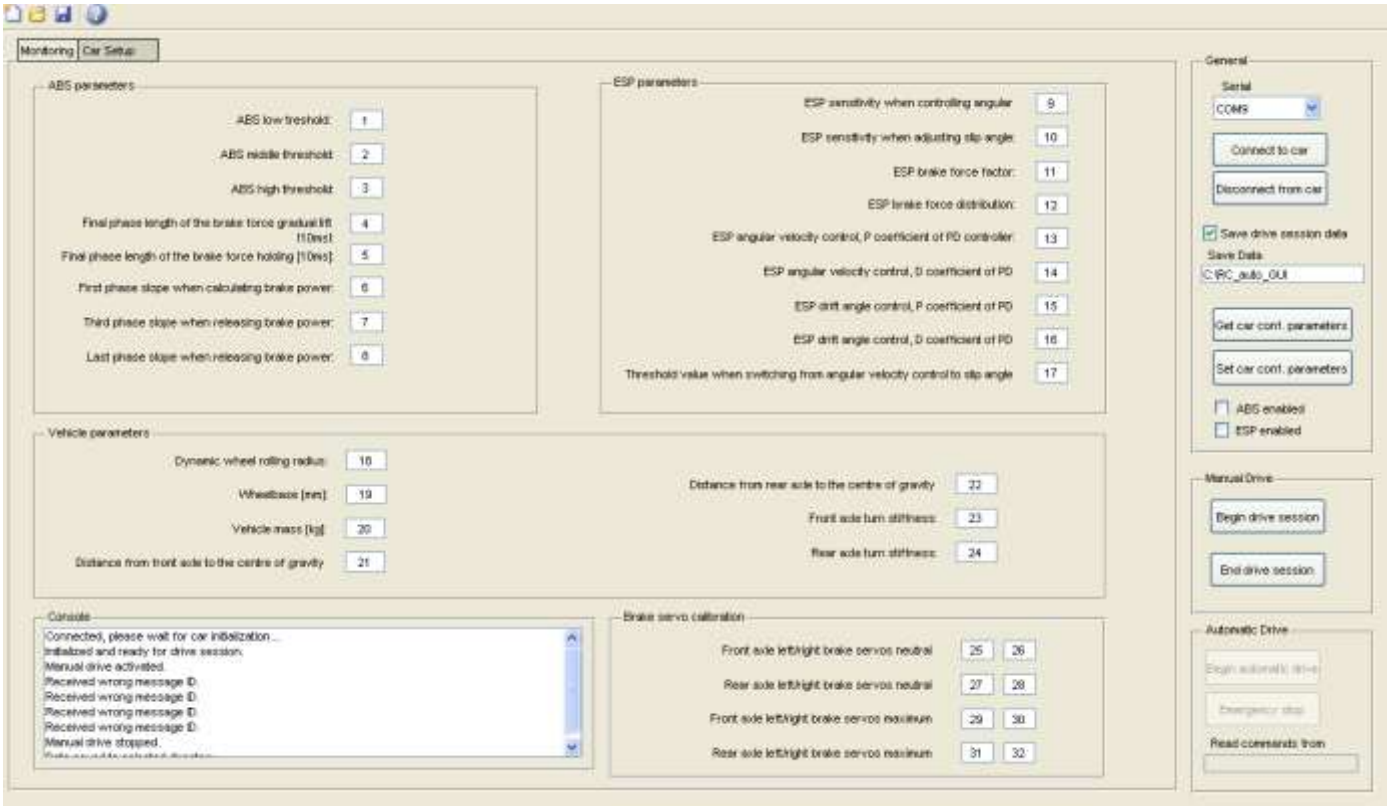
Käyttöliittymän lopullinen versio on esitetty kuvissa 18 ja 19. Kuvassa 18 oleva käyttöliittymän seurantanäkymä on kuvankaappaus lyhyeestä ajosta. Valitettavasti tila jossa autoa ajettiin oli hyvin rajoitettu, joten mitään pidempää ajojaksoa ei pystytty toteuttamaan. Tästä kärsi eteenkin vasemmassa yläkulmassa ajetun reitin karttaa näyttävä Route Map, joka sisältää paljon häiriöitä tässä pienessä skaalassa. Auton pyörien enkoodedereista laskettu nopeus on sen sijaan suhteellisen tarkkaa. Nopeus on nähtävissä Route Map:in oikealla puolella olevassa Velocity kuvaajassa. Nopeuden alapuolella oleva horisontaalinen Wheel Direction kuvaaja näyttää auton etupyörien suunnan. Kolme alhaisinta palkkia ovat vasemmalta oikealle Brake, Throttle sekä Reverse, eli jarru, kaasu ja peruutus. Käyttöliittymän vasemmassa alareunassa oleva Console antaa käyttäjälle kaikenlaisia hyödyllisiä vinkkejä, kuten että autoon on yhdistetty onnistuneesti tai mikis ajojaksoa ei voitu aloittaa. Kuvaajista on ajan puutteen syystä jäänyt yksiöt pois. Toisaalta yksiöt on määritelty ainoastaan nopeuden y-akselilla (km/h), sekä reittikartan akselit (m). Muut kuvaajat ja akselit kuvaavat ainoastaan suhteellista muutosta.

Oikeassa laidassa oleva General osiolla hallitaan yhteydet autoon sekä auton datan tallennus. Manual Drive osiossa käynnistetään ja lopetetaan manuaaliajo. Automatic Drive osiossa käynnistetään automaattinen ajo, eli ajokomentojen lukeminen tiedostosta. Painekkeet ovat poistettu käytöstä koska toimintoa ei toteutettu.

Kuva 19 on kuvankaappaus auton parametrien muokkausnäkymästä. Kuvassa näkyvät parametrit eivät ole auton oikeat, vaan ne ovat testiparametreja jolla parametrien vienti autoon ja tuonti autosta on testattu. Parametrit on jaettu seuraaviin kategorioihin: ABS parameters, ESP parameters, Vehicle parameters, sekä Brake servo calibration. Vasemmassa alanurkassa näkyvä Console osio sekä oikealla reunalla sijaitsevat osiot General, Manual Drive, sekä Automatic Drive, ovat samat kuin seurantanäkymässä.



Kuva 18: Käyttöliittymän seurantanäkymä.



Kuva 19: Käyttöliittymän auton parametrien muokkausnäkymä.

4.3.2 Käyttöliittymän käyttö

Käyttöliittymän ajoa varten on ensin asennettava kirjasto nimeltään Psychtoolbox 3. Kirjaston asennusohjeet, sekä muita käyttöliittymän vaatimuksia löytyy liitteestä 2. Käyttöliittymän käyttö on varsin suoraviivaista. Käyttöliittymä käynnistetään ajamalla RCCarGUI.m tiedosto, jolloin kuvaa 18 vastaava aloitusnäkymä tulee näkyviin. Ennen autoon yhdistämistä on ensin valittava oikea serial portti johon Xbee radio on kiinnitetty. Kun oikea serial portti on valittu autoon voidaan yhdistää painamalla *Connect to car* paineketta. Kun Console kertoo auton olevan valmis ajojakso voidaan aloittaa painamalla *Begin drive session*. Autoa ohjataan seuraavien ohjeiden mukaan näppäimistöä käyttäen:

Ohjaimet

- Nuolinäppäimistöt: Kaasu, pakki, vasemmalle ja oikealle käännös
- Välilyönti: Täysjarrutus
- V, B, N: Heikko, keskivahva ja vahva jarrutus
- C, Kytkin
- T, Torvi

Ajojakso lopetetaan painamalla *End drive session* paineketta. Yksinkertaista ajodataa on näkyvissä seurantanäkymässä, täydelliseen ajodatan pääsee käsiksi avaamalla työhakemiston Drive Sessions kansio, ja avaamalla ajonjaksoa vastaava .mat-tiedosto.

Auton parametreja muokataan Car Setup välilehdessä. Auton tämänhetkisiä parametreja voi ladata autosta *Get car conf. parameters* painekkeella, ja uusia lähetetään autolle painamalla S*et car conf. parameters.* Liitteestä 5 löytyy lista parametrien raja-arvoista. Vasemmasta ylänurkasta löytyy joukko painekkeita joilla voi tallentaa tietokoneelle nykyiset parametrit tai ladata vanhoja parametreja.

Lopuksi muutama hyödyllinen vihje käyttöliittymän käyttäjälle:

- Autoa ajatessa on oltava varovainen kaasun ja peruutuksen kanssa. Käyttöliittymässä on jonkinverran älyä, esimerkiksi jarrukomento poistaa automaattisesti kaasukomennon, mutta mikään ei estä esimerkiksi käyttäjän peruuttamasta autoa täydessä liikkeessä eteenpäin, joka tietenkin saisi tuhoisaa jälkeä moottoreissa sekä voimansiirroissa.
- Auton parametreia evät ole suojattu mitenkään väärältä syötöltä.
- *Get car conf. parameters* toiminto on hieman epävakaa, ja tästä syystä useampikin yritys saattaa olla tarpeen ennen kuin parametrien siirto onnistuu.

5 YHTEENVETO JA JOHTOPÄÄTÖKSET

Projekti oli osakokonaisuus radio-ohjattavan pienoismallin rakentamisprojektia. Projektin tavoitteisiin kuului suunnitella ja toteuttaa radio-ohjattavan pienoismallin käyttöliittymä, sekä simuloida ja toteuttaa lukkiutumattomat jarrut ja ajoneuvonvakausjärjestelmä pienoismallille. Näistä tavoitteista tärkeimmät olivat käyttöliittymä ja lukkiutumattomien jarrujen ohjelmointi ja toteutus mikrokontrolleriin. Vastuualueisiin kuului myös koko jarrukontrollerin sulautettu ohjelmointi, mukaan lukien kommunikointi, anturit ja parametrit.

Aikaa projektille oli varattu 70 tuntia käyttöliittymälle ja jarru-osuudelle. Mutta lopulta aikaa kului noin 100 tuntia per osuus. Vaikka kaikkiin tavoitteisiin ei edes päästy. Tavoitteista käyttöliittymän mänööveri ohjaus komentotiedostosta jäi toteuttamatta ja sulautetun ohjelmoinnin puolelta ajoneuvonvakautusjärjestelmän testaus ja viimeistely jäi puuttumaan. Kuitenkin saimme aikaan nähden toimivan ja käyttötarkoitukseen sopivan käyttöliittymän. Sekä sulautetun ohjelmiston puolella saimme lukkiutumattomat jarrut testattua ja toimimaan. Tärkeänä projektissa oli saada käyttöliittymä ja sulautettu ohjelmisto hyväksi ja toimivaksi ajoa varten. Kommunikoinnissa ei ollut viiveitä ja ajo pienoismallilla oli varsin helppoa projektin loppupuolella.

Aikataulutuksesta opimme että aina tulisi arvioida aika joka menee kolmannen osapuolen kanssa kommunikoimiseen ja asioiden hoitamiseen. Nyt kun projekti oli vain yksi osa suurempaa projektia, jouduimme olemaan myös paljon mukana auton mekaanisen osien testauksessa. Toki samalla päästiin kokeilemaan käyttöliittymän ja jarrujen toimintaa. Voimme myös todeta että projektia suunniteltaessa on vaikeaa arvioida projektiin kuluvaa aikaa jos asiasta ei tiedä tarpeeksi, esimerkiksi nyt Matlab ohjelmiston sarjaporttikommunikaatio osoittauti huomattavasti hankalammaksi kuin aluksi ajateltiin, aiheuttaen mm. käyttöliittymän jumiutumista. Myös ABS puolella oli hankaluuksia, selvisi että ABS-algoritmi onkin oletettua monimutkaisempi ja vaati enemmän aikaa.

Projektin alussa laaditut tehtävät pysyivät samoina koko projektin elinkaaren ajan, ainoastaan pieniä muutoksia toivottiin käyttöliittymään. Eli perinteiseen IT-projektiin nähden toivotut muutokset kesken kaiken olivat erittäin vähäisiä, jonka perusteella voidaan todeta että suunnitellut tehtävät olivat hyvin määritelty heti alussa.

LÄHTEET

1. Robert Bosch GmbH, *Safety,Comfort and Convenience Systems*

2. Ari Tuononen & Tapio Koisaari, *Ajoneuvojen dynamiikka*

3. Hongtei Eric Tseng, Behrouz Ashrafi, Dinu Madau, Todd Allen Brown & Darrel Recker, *The Development of Vehicle Stability Control at Ford*

4. Song Chuanxue,Xuan Shengyi, Li Jianhua & Jin Liqiang, *Research on Vehicle Electronic Stability Control Method*

5. Diomidis I. Katzourakis, Ioannis Papaefstathiou & Michail G. Lagoudakis, *An Open-Source Scaled Automobile Platform for Fault-Tolerant Electronic Stability Control*

LIITTEET

Liite 1: Sarjaväyläprotokolla

MSG_ID	Lähettäjä	Vastaanottaja	Tavu 1	Tavu 2	Tavu 3	Tavu 4	Tavu 5	Tavu 6	Tavu 7	Tavu 8	Tavu 9	Tavu 10
0x10	Teensy	Jarrukontrolleri	Vasemman etupyörän nopeus	Oikean etupyörän nopeus	Vasemman takapyörän nopeus	Oikean takapyörän nopeus						
0x20	Jarrukontrolleri	Pääkontrolleri	Vasemman etupyörän nopeus	Oikean etupyörän nopeus	Vasemman takapyörän nopeus	Oikean takapyörän nopeus	x-kiihtyvyys high byte	x-kiihtyvyys low byte	y-kiihtyvyys high byte	y-kiihtyvyys low byte	z-kiihtyvyys high byte	z-kiihtyvyys low byte
0x30	Pääkontrolleri	Tietokone	Vasemman etupyörän nopeus	Oikean etupyörän nopeus	Vasemman takapyörän nopeus	Oikean takapyörän nopeus	x-kiihtyvyys high byte	x-kiihtyvyys low byte	y-kiihtyvyys high byte	y-kiihtyvyys low byte	z-kiihtyvyys high byte	z-kiihtyvyys low byte
0x31	Pääkontrolleri	Jarrukontrolleri	Tavoitu jarrutusvoima	Etupyörien suunta								
0x41	Tietokone	Pääkontrolleri	Throttle ("kaasupolkimen asento")	Tavoitu ohjaussuunta	Tavoitu jarrutusvoima	Kytkimen asento	Ajosuunta					
0x42	Tietokone	Pääkontrolleri	Parametri numero "key"	Uusi arvo, tavu 1	Uusi arvo, tavu 2							
0x32	PC->Pääkontrolleri	Jarrukontrolleri	Parametri numero "key"	Uusi arvo, tavu 1 "value"	Uusi arvo, tavu 2 "value"							
0x43 (Pyyntö että auto lähettää parametrinsa)	Tietokone	Pääkontrolleri	255 (PC kysyy parametreja)									
0x33	Pääkontrolleri	Jarrukontrolleri	255 (PC kysyy parametreja)									
0x21	Jarrukontrolleri	Pääkontrolleri	Parametri Id	Parametri tavu 1	Parametri tavu 2							
0x34	Pääkontrolleri	Tietokone	Parametri Id	Parametri tavu 1	Parametri tavu 2							
0x44 (Pistä töötti päälle)	Tietokone	Pääkontrolleri										

MSG_ID	Lähettäjä	Vastaanottaja	Tavu 11	Tavu 12	Tavu 13	Tavu 14	Tavu 15	Tavu 16	Tavu 17	Tavu 18	Tavu 19
0x10	Teensy	Jarrukontrolleri									
0x20	Jarrukontrolleri	Pääkontrolleri	x-gyro high byte	x-gyro low byte	y-gyro high byte	y-gyro low yte	z-gyro high byte	z-gyro low byte			
0x30	Pääkontrolleri	Tietokone	x-gyro high byte	x-gyro low byte	y-gyro high byte	y-gyro low yte	z-gyro high byte	z-gyro low byte	Etupyörien suunta	Moottorialkun jännite	Kontrolleralkun jännite
0x31	Pääkontrolleri	Jarrukontrolleri									
0x41	Tietokone	Pääkontrolleri									
0x42	Tietokone	Pääkontrolleri									
0x32	PC->Pääkontrolleri	Jarrukontrolleri									
0x43 (Pyyntö että auto lähettää parametrinsa)	Tietokone	Pääkontrolleri									
0x33	Pääkontrolleri	Jarrukontrolleri									
0x21	Jarrukontrolleri	Pääkontrolleri									
0x34	Pääkontrolleri	Tietokone									
0x44 (Pistä töötti päälle)	Tietokone	Pääkontrolleri									

Liite 2: Käyttöliittymän asennus

Vaatimukset:

- Käyttöliittymä on onnistuneesti käytetty Windowsilla MATLAB 2010 ja 2012 versioiden kanssa.
- Psychtoolbox 3 kirjaston toimintoja vaaditaan ainoastaan ajon aikana, eli käyttöliittymää voidaan avata ilman tätä kirjastoa.

Psychtoolbox 3 asennus

Hyvät asennusohjeet löytyvät Psychtoolboxin kotisivuilta.

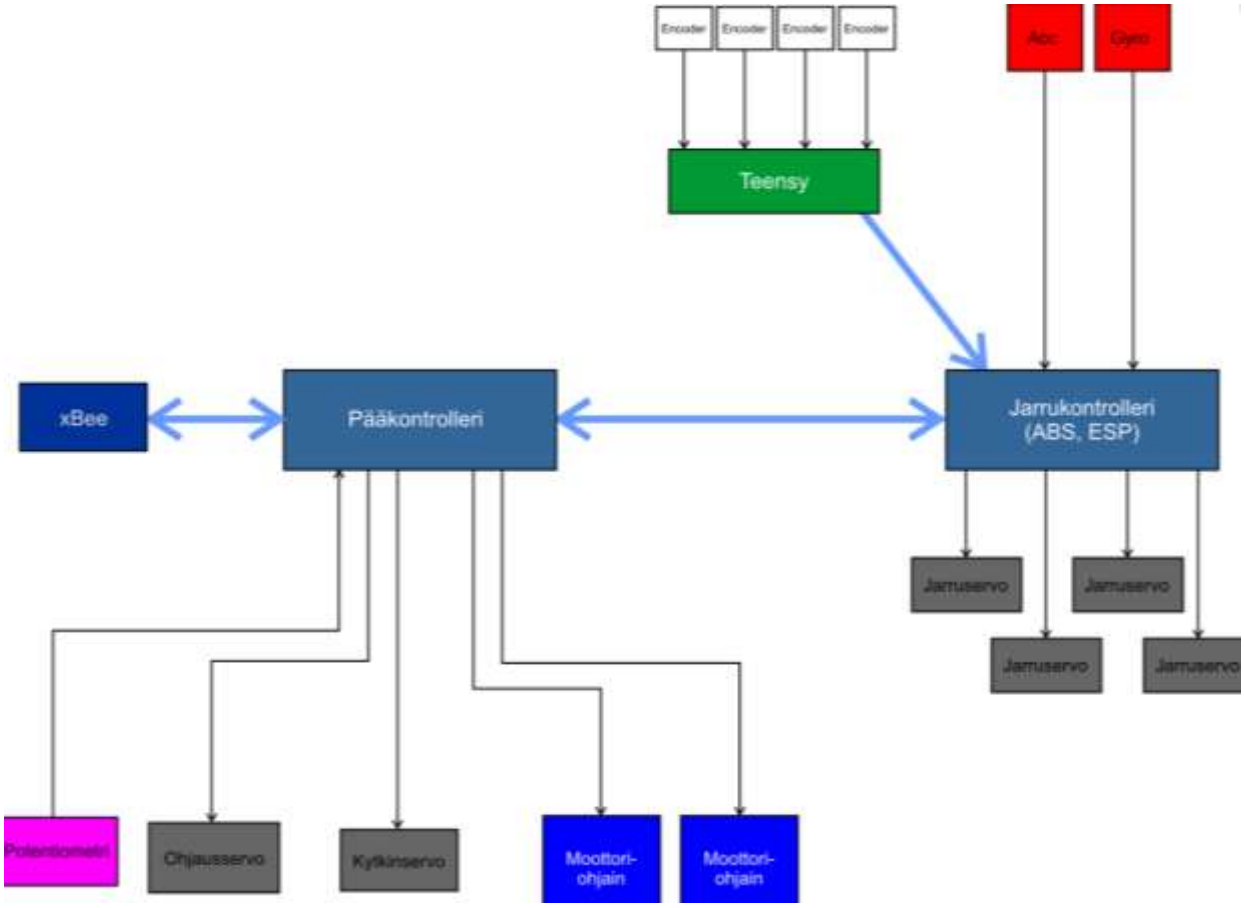
http://psychtoolbox.org/PsychtoolboxDownload

Kotisivuilta löytyy myös kaikenlaista muuta hyödyllistä tietoa kirjastosta.

http://psychtoolbox.org/

Asennuksen jälkeen käyttöliittymän voi käynnistää ajamalla tiedoston RCCarGUI.m

Liite 3: Radio-ohjattavan pienoismallin elektroniikka



Liite 4: ABS-algoritmi

```
int ret = (zero_brake + ((long(brake_pwr) * brake_scale) / 255)) <= max_brake ?
zero_brake + ((long(brake_pwr) * brake_scale) / 255) : max_brake;

  if(abs_on && brake_pwr > 0) // PHASE 0 START
  {
    if(a < (-1 * ((int)abs_start_threshold * 50 / 255)) && *phase == 0)
    {
      *max_brake_pwr = ret;
      *phase = 1;
    }
    else if(*phase == 1) // PHASE 3 (drop braking force)
    {
      (*counter)++;

      *abs_brake_pwr = *max_brake_pwr - (*counter * phase1_gain) > zero_brake ?
*max_brake_pwr - (*counter * phase1_gain) : zero_brake;
      ret = *abs_brake_pwr;
      if(a > (-1 * ((int)abs_start_threshold * 50 / 255)))
      {
        *counter = 0;
        *phase = 2;
      }
    }
    else if(*phase == 2) // PHASE 4 (hold current braking force)
    {

      ret = *abs_brake_pwr;
      (*counter)++;

      if((abs(a) > ((int)abs_high_threshold * 50 / 255)))
      {
        *counter = 0;
        *phase = 3;
      }
      if((*counter) > 2)
      {
        *counter = 0;
        *phase = 3;
      }
    }
    else if(*phase == 3) // PHASE 5 (lift braking force)
    {
      (*counter)++;

      *abs_brake_pwr = *abs_brake_pwr + (*counter * phase3_gain) < max_brake ?
*abs_brake_pwr + (*counter * phase3_gain) : max_brake;
      ret = *abs_brake_pwr;

      // simplified
      if(a < (-1 * ((int)abs_start_threshold * 50 / 255)))
      {
        *max_brake_pwr = *abs_brake_pwr;
        *counter = 0;
        *phase = 1;
      }
      if((*counter) > 2)
      {
        *max_brake_pwr = *abs_brake_pwr;
        *counter = 0;
        *phase = 1;
      }

      // advanced
      /*
      if(a < ((int)abs_high_threshold * 50 / 255))
      {
        *counter = 0;
        *phase = 4;
      }
      */
    }
    /*else if(*phase == 4) // PHASE 6 (advanced)
```

```
    {
      ret = *abs_brake_pwr;
      if(a < ((int)abs_middle_threshold * 50 / 255))
      {
        *counter = 0;
        *phase = 5;
      }
    }
    else if(*phase == 5) // PHASE 7 (advanced)
    {
            (*counter)++;

      if(*counter < phase5_gain_timer_threshold)
      {
        *abs_brake_pwr = *abs_brake_pwr + (*counter * phase5_gain) < max_brake ?
*abs_brake_pwr + (*counter * phase5_gain) : max_brake;
      }
      else if (*counter >= phase5_gain_timer_threshold && *counter <
phase5_hold_timer_threshold)
      {
        // hold
      }
      else
      {
        *counter = 0;
      }

      ret = *abs_brake_pwr;

      if(a < (-1 * ((int)abs_start_threshold * 50 / 255)))
      {
        *max_brake_pwr = *abs_brake_pwr;
        *counter = 0;
        *phase = 1;
      }
    }*/
  }
  else
  {
    *phase = 0;
  }
```

Liite 5: ABS & ESC parametrit

Mikrokontrolleri	Parametri	Id	Säätöarvot
Jarrukontrolleri	etuakselin vasemman jarruservon nolla asento	10	500 - 2500 (2 bytes)
Jarrukontrolleri	etuakselin oikean jarruservon nolla asento	11	500 - 2500 (2 bytes)
Jarrukontrolleri	takaakselin vasemman jarruservon nolla asento	12	500 - 2500 (2 bytes)
Jarrukontrolleri	takaakselin oikean jarruservon nolla asento	13	500 - 2500 (2 bytes)
Jarrukontrolleri	etuakselin vasemman jarruservon max asento	14	500 - 2500 (2 bytes)
Jarrukontrolleri	etuakselin oikean jarruservon max asento	15	500 - 2500 (2bytes)
Jarrukontrolleri	takaakselin vasemman jarruservon max asento	16	500 - 2500 (2 bytes)
Jarrukontrolleri	takaakselin oikean jarruservon max asento	17	500 - 2500 (2 bytes)
Jarrukontrolleri	abs on/off	20	1/0
Jarrukontrolleri	abs low threshold	21	0 - 255
Jarrukontrolleri	abs middle threshold	22	0 - 255
Jarrukontrolleri	abs high threshold	23	0 - 255
Jarrukontrolleri	viimeisen vaiheen pituus jarruvoiman vaiheittaisessa nostossa (1 = 10ms, 2 = 20ms ..)	24	0 - 255
Jarrukontrolleri	viimeisen vaiheen pituus jarruvoiman vakioarvon pitämisessä (1 = 10ms, 2 = 20ms ..)	25	0 - 255
Jarrukontrolleri	Ensimmäisen vaiheen kulmakerroin jarruvoimaa laskettaessa	26	0 - 255
Jarrukontrolleri	Kolmannen vaiheen kulmakerroin jarruvoima nostettaessa	27	0 - 255
Jarrukontrolleri	Viimeisen vaiheen kulmakerroin jarruvoimaa nostettaessa (liittyy parametriin 24)	28	0 - 255
Jarrukontrolleri	ESP on/off	40	1/0
Jarrukontrolleri	ESP herkkyys kulmanopeussäädössä	41	0 - 255
Jarrukontrolleri	ESP herkkyys sortokulmasäädössä	42	0 - 255
Jarrukontrolleri	ESP jarruvoiman kerroin (muut paitsi säädettävä pyörä) (brake_pwr / esp_brake_div) * esp_brake_multi	43	0 - 255
Jarrukontrolleri	ESP jarruvoiman jako (muut paitsi säädettävä pyörä) (brake_pwr / esp_brake_div) * esp_brake_multi	44	0 - 255
Jarrukontrolleri	ESP kulmanopeussäätö, PD säätimen vahvistus P	45	0 - 255
Jarrukontrolleri	ESP kulmanopeussäätö, PD säätimen derivaatta D	46	0 - 255
Jarrukontrolleri	ESP sortokulmasäätö, PD säätimen vahvistus P	47	0 - 255
Jarrukontrolleri	ESP sortokulmasäätö, PD säätimen derivaatta D	48	0 - 255
Jarrukontrolleri	ESP sortokulmasäädön ja kulmanopeussäädön raja-arvo	49	0 - 255
Jarrukontrolleri	pyörän dynaaminen vierintäsäde	50	0 - 255
Jarrukontrolleri	akseliväli (m)	51	value(uint16)/100000,max 65535 (0,65m)
Jarrukontrolleri	ajoneuvon massa (kg)	52	value(uint16)/1000, max 65535 (65,535kg)
Jarrukontrolleri	etäisyys etu-akseliin painopisteestä (l1)	53	value(uint16)/100000,max 65535 (0,65m)
Jarrukontrolleri	etäisyys taka-akseliin painopisteestä (l2)	54	value(uint16)/100000,max 65535 (0,65m)
Jarrukontrolleri	Etu-akselin kaartojäykkyys C1	55	uint16, max 65535
Jarrukontrolleri	Taka-akselin kaartojäykkyys C2	56	uint16, max 65535