Sami Kulovuori

Euro 6 -päästöluokan konttivetorekkojen todellisen käyttöympäristön pakokaasupäästöt

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ajoneuvotekniikka

Insinöörityö

30.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Sami Kulovuori Euro 6 -päästöluokan konttivetorekkojen todellisen käyttöympäristön pakokaasupäästöt 88 sivua + 8 liitettä 30.05.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Ajoneuvotekniikka
Ammatillinen pääaine	Ajoneuvosuunnittelu
Ohjaajat	Projekti-insinööri Aleksi Malinen

Tässä työssä selvitettiin Euro 6 -päästöluokan täyttävien nestemäisen maakaasu- ja dieselkonttivetorekkojen pakokaasupäästöjen eroja todellisessa ajoympäristössä. Työ tehtiin yhteistyössä Containership Oyj:n, Iveco Finland Oy:n, Gasum Oy:n sekä Liikenteen turvallisuusvirasto Trafin kanssa, jotka rahoittivat tutkimushanketta. Tutkivana osapuolena tutkimushankkeessa oli Metropolia Ammattikorkeakoulu. Tutkimushankkeeseen liittyvät mittaukset toteutettiin syksyllä 2017.

Mittaukset suoritettiin neljänä päivänä ja ne toteutettiin jahtausmittausperiaatteella. Mittauksissa mitattiin pakokaasupäästöistä CO, CO_2, NO_X, O_3, BC sekä hiukkaspäästöt. Hiukkaspäästöt mitattiin sekä tiiveysydinlaskurilla (TSI Inc, CPC), optisella hiukkaslaskurilla (TSI Inc, OPS) sekä sähköisellä aerosolispektrometrillä (TSI Inc, EEPS). NO_X-päästöt mitattiin Horiba APNA-360CE -mittalaitteella, sekä mitattaviin ajoneuvoihin asennetuilla Proventia Procare NO_X -mittalaitteistolla. CO- ja O_3-päästöt mitattiin Environnement S.A:n valmistamilla kaasuanalysaattoreilla CO12M ja O342M. CO_2-päästöt mitattiin Li-Cor 840A -mittalaitteella. Musta hiili (BC) mitattiin Magee Scientific Aethalometer AE33 -mittalaitteella.

Mitattavien ajoneuvojen pakokaasun jälkikäsittelylaiteistoissa oli huomattavia eroja, johtuen täysin erityyppisistä moottoreista ja niiden tavasta tuottaa pakokaasupäästöjä. Hyvinkin yksinkertaisella ja oikein mitoitetulla järjestelmällä saadaan pudotettua pakokaasupäästöjä tehokkaasti varsinkin maakaasumoottorista.

Mittauksissa havaittiin, että Euro 6 tyyppihyväksytty nestemäistämaakaasua polttoaineena käyttävä rekka on pakokaasupäästöiltään puhtaampi kuin vastaava dieselrekka.

Avainsanat	BC, CO_2, Diesel, Euro 6, Maakaasu, NO_X, PM, Päästöt, Raskas kaulusto

Author	Sami Kulovuori
Title	Real-world Exhaust Emissions of Euro 6- approved Heavy-duty Trucks.
Number of Pages	88 pages + 8 appendices
Date	30 May 2018
Degree	Bachelor of Engineering
Degree Programme	Automotive Engineering
Professional Major	Vehicle Engineering
Instructors	Aleksi Malinen, Project Engineer

The objective of this Bachelor's thesis was to find out the difference between Euro 6 emission class exhaust emissions of liquid natural gas and diesel trucks in the actual driving environment. The work was carried out in co-operation with Containership Plc, Iveco Finland Oy, Gasum Oy and Transport Safety Agency Trafi who funded the project. The project measurements were carried out in the autumn of 2017 by Metropolia University of Applied Sciences.

The measurements were carried out by a chasing measurement principle during four days. The measured components were CO, CO_2, NO_X, O_3, BC and particulate matter. Particulate emissions were measured with the ultrafine condensation particle counter (TSI Inc, CPC), the optical particle counter (TSI Inc, OPS) and the engine exhaust particle sizer (TSI Inc, EEPS). NO_X -emissions were measured with the Horiba APNA-360CE and Proventia Procare NO_X -system installed on the measured vehicles. CO and O_3 -emissions were measured with Environnement S.A analyzers CO12M and 0342M. CO_2 emissions were measured with the Li-Cor 840A measuring instrument. Black carbon (BC) was measured using the aethalometer AE33 from Magee Scientific.

There were considerable differences in the exhaust after treatment system between the vehicles. This was due completely different types of engines and their way of producing exhaust emissions. With a very simple and properly dimensioned system, exhaust emissions can be effectively reduced, especially from liquid natural gas engine.

It was found that Euro 6 approved liquid natural gas truck is cleaner on all exhaust emissions than the corresponding diesel truck.

Keywords	BC, CO_2, Diesel, Emissions, Euro 6, Heavy-Duty Trucks, Natural Gas, NO_X, PM

Sisällys

Lyhenteet

BC	Musta hiili
C	Hiili
C_4H_{10}	Butaani
C_8H_{18}	Oktaani
C_8H_8	Bentseeni
CO	Hiilimonoksidi
CO_2	Hiilidioksidi
CPC	Ultrafine Condensation Particle Counter
DOC	Diesel Oxidation Catalyst
DPF	Diesel particle filter
EEPS	Exhaust emission particle sizer
EF_X	Päästökerroin x
EGR	Pakokaasun takaisinkierrätys
ER_X	Päästökertoimen suhdeluku
H_2O	Vesi
HC	Hiilivedyt
k	Savutusarvo
M	Moolimassa

N_2	Typpi
NH_3	Ammoniakki
NO	Typpimonoksidi
NO_2	Typpidioksidi
NO_X	Typenoksidit
O_2	Happi
OBD	On board Diagnostics
OPS	Optical particle sizer
PM	Hiukkasmateria
ppb	Parts per billion, suhdeyksikkö miljardisosa jostakin
ppm	Parts per million, suhdeyksikkö miiljoonasosa jostakin
ppt	Parts per trillion suhdeyksikkö biljoonasosa jostakin
SCR	Selective Catalytic Reduction
SO_2	Rikkidioksidi
UV	Ultravioletti
WHSC	World Harmonized Stationary Cycle
WHTC	World Harmonized Transient Cycle

Alkulause

Haluan kiittää tutkimushankkeen mahdollistamisesta Containership Oyj:tä, Iveco Finland Oy:tä, Gasum Oy:tä sekä Liikenteen turvalisuusvirasto Trafia sekä Metropolia Ammattikorkeakoulua. Ilman heidän panostaan tutkimushanke olisi jäänyt toteuttamatta. Haluan erityisesti kiittää Containership Oyj:n mittauksissa yhdistelmän ajamisesta vastannutta Jyrki Jussilaa. Kiitos myös mittauksissa mukana olleille opiskelija-assistentti Teemu Keinäselle sekä projekti-insinööri Aleksi Maliselle, jota kiitän myös työn ohjaamisesta.

1 Johdanto

Tämän työn tarkoituksena on selvittää, Euro 6 -päästöluokan täyttävien maakaasu- sekä dieselkonttivetorekkojen todellisten pakokaasupäästöjen eroja. Työ on tehty yhteistyössä Containership Oyj:n, Iveco Finland Oy:n, Gasum Oy:n sekä Liikenteen turvallisuusvirasto Trafin kanssa, jotka ovat olleet rahoittamassa työtä. Tutkivana osapuolena työssä on ollut Metropolia Ammattikorkeakoulu.

Työn painotus on ollut tutkia typpioksidipäästöjen (NO_X) eroja polttoaineiden kesken. Tämä työ osoittaa eri polttoaineita käyttävien rekkojen pakokaasupäästöjen eroja, ajoneuvojen todellisessa käyttöympäristössä.

Työssä tehdyt mittaukset on suoritettu kokeellisella jahtausmittaustekniikalla Nuuskija-tutkimusajoneuvoa käyttäen todellisissa ajotilanteissa, mistä Metropolia Ammattikorkeakoululla on aikaisempia kokemuksia useiden eri hankkeiden yhteydestä. Nuuskija-ajoneuvo on laboratorio, joka mahdollistaa erilaisten ilmansaasteiden mittaamisen mobiilisti. Mittauksissa mitattavat ajoneuvot oli varustettu myös typpioksidin mittauslaitteistolla.

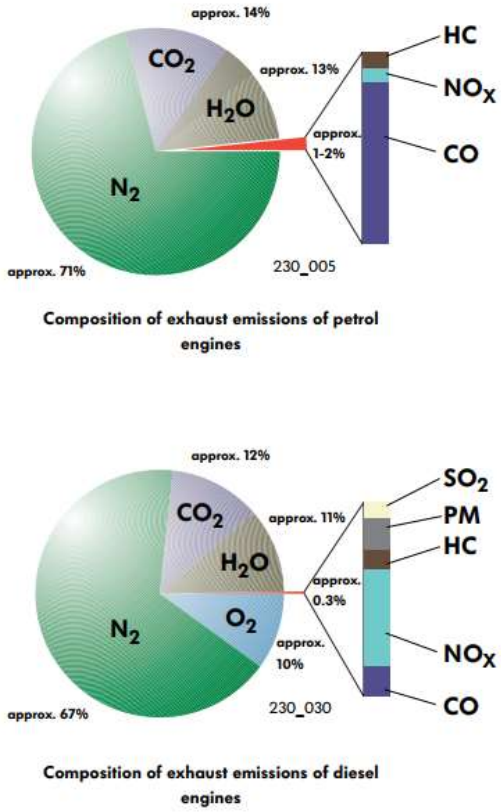
Työssä kerrotaan aluksi yleisesti pakokaasupäästöistä, mitattavia autoja koskevasta lainsäädännöstä sekä ajoneuvoissa olleista pakokaasujen puhdistuslaitteista. Työssä kuvataan myös työhön liittyvien mittausten toteutus, mittauksissa käytetty laitteisto ja niiden toimintaperiaatteet. Lopuksi esitellään työhön liittyvien mittausten tulokset sekä päästökertoimien laskenta.

2 Pakokaasupäästöt

Tässä osiossa tarkastellaan yleisesti pakokaasupäästöjä, lainsäädäntöä sekä pakokaasujen jälkikäsittelyä.

2.1 Yleistä pakokaasupäästöistä

Pakokaasupäästöt koostuvat hiukkaspäästöistä sekä erikaasuista sekä niiden yhdisteistä. Suurin osa pakokaasusta on typpeä (N_2); muita isoja komponentteja pakokaasussa ovat hiilidioksidi (CO_2), happi (O_2), vesi (H_2O). Loput pakokaasun komponentit koostuvat typenoksideista (NO_X), hiilimonoksidista (CO), rikkidioksidista (SO_2), hiilivedyistä (HC) ja hiukkasista (PM). (Kuva 1.) Uusissa Euro 6 -päästöluokan luokan ajoneuvoissa pakokaasun mukana voi kulkeutua myös pieniä määriä ammoniakkia (NH_3). [1, s. 6.]



Kuva 1. Pakokaasujen koostumus [1, s. 6].

Typpi (N_2) on palamaton, väritön ja hajuton kaasu. Sitä on hengitysilmassa n. 78 %. Suurin osa typestä (N_2) pysyy muuntumattomana polttomoottorissa, mutta pienestä osasta muodostuu typenoksideja (NO_X) palotapahtuman aikana, kun typpi reagoin hapen kanssa. [1, s. 7.]

Happi (O_2) on väritön, hajuton ja mauton kaasu. Sitä on hengitysilmassa n. 21 %. Polttomoottorista ei käytännössä tule ulos yhtään happea, koska se reagoi palotapahtuman aikana muiden kaasujen ja aineiden kanssa. Dieselmoottorissa kaikki happi ei ehdi reagoimaan palotapahtumassa, joten pakokaasun mukana kulkeutuu jäännöshappea. [1, s. 7.]

Hiilidioksidi (CO_2) on väritön, palamaton kaasu. Sitä syntyy palotapahtumassa, kun polttoaineessa oleva hiili (C) reagoi hapen (O_2) kanssa. Hiilidioksidi (CO_2) voidaan luokitella kasvihuoneilmiötä nopeuttavaksi kaasuksi, joka pienentää ilmakehän otsonikerrosta ja helpottaa auringon UV-säteilyn läpäisyä maahan. [1, s. 8.]

Vesi (H_2O) on harmiton sivutuote pakokaasussa, joka syntyy moottorin läpi kulkeutuvan ilmankosteuden johdosta. Vettä on pakokaasussa melkein yhtä paljon kuin hiilidioksidia. [1, s. 8.]

Hiilimonoksidi (CO) on väritön, hajuton erittäin myrkyllinen räjähdysherkkä kaasu. Sitä syntyy epäpuhtaan palamisen johdosta. Luonnossa hiilimonoksidi (CO) hapettuu nopeasti ja muuttuu hiilidioksidiksi (CO_2). [1, s. 8.]

Typenoksidit (NO_X) on yleinen nimitys typen (N_2) ja hapen (O_2) muodostamille yhdisteille palotapahtumassa; yhdisteitä ovat mm. typpidioksidi (NO_2), typpimonoksidi (NO), dityppioksidi (N_2O) jne.. Typenoksideja muodostuu korkeassa lämpötilassa ja paineessa palotapahtuman aikana. Useat typenoksidi yhdisteet ovat terveydelle haitallisia. [1, s. 8.]

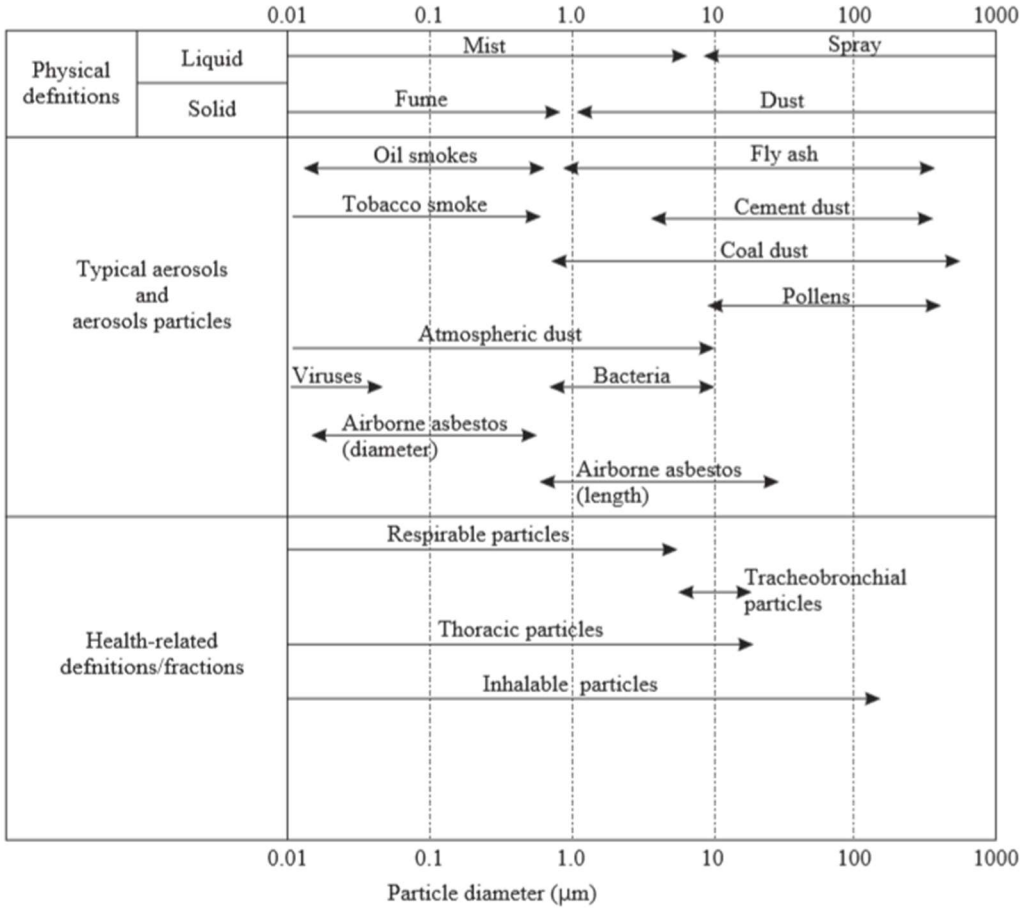
Rikkidioksidi (SO_2) on väritön, palamaton ja pistävän hajuinen kaasu. Rikkidioksidi aiheuttaa hengitystiesairauksia, mutta sitä on todella vähän pakokaasussa. Rikkidioksidi päästöjä voidaan vähentää käyttämällä vähemmän rikkiä polttoaineissa. [1, s. 9.]

Hiilivedyt (HC) ovat polttoaineen palamattomia komponentteja, joita syntyy epäpuhtaan palamisen johdosta. Hiilivetyjä on monessa muodossa: mm. bentseeni (C_6H_6), oktaani

(C_8H_{18}), butaani (C_4H_{10}). Jokaisella hiilivedyllä on erilainen vaikutus ihmiseen. Jotkut hiilivedyt, esimerkiksi bentseeni, on luokiteltu karsinogeeneiksi, jotka altistavat syövälle. [1, s. 9.]

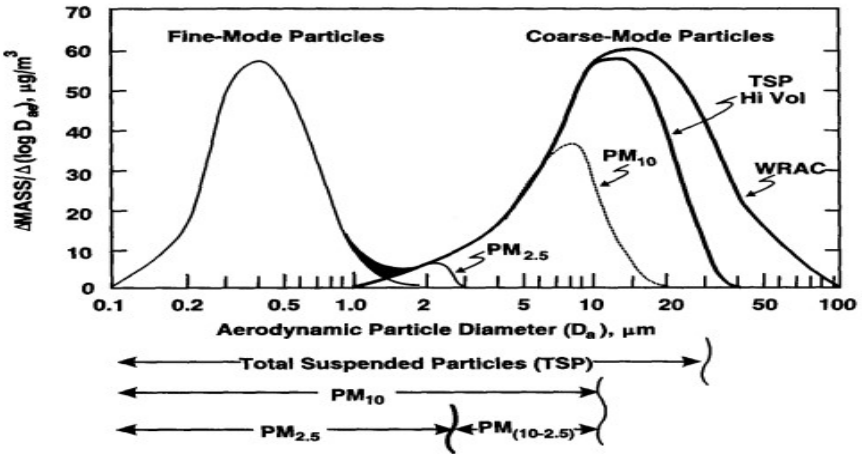
Musta hiili (BC) on epäorgaanista hiiltä, ja se on pienhiukkasten yksi komponentti, jonka aerodynaaminen halkaisija on ≤ 2,5 µm. Mustaa hiiltä syntyy hiilivetyjen (HC) epätäydellisen palamisen yhteydessä. Musta hiili on yksi ilmastoa lämmittävistä aineista, koska sillä on kyky absorboida itseensä auringonvaloa ja näin varata itseensä lämpöä. Musta hiilen terveysvaikutukset ihmisille ovat samat kuin pien hiukkasilla. Noin 20 % mustasta hiilestä tulee biopolttoaineiden poltosta, 40 % fossiilisista polttoaineista ja 40 % vapaasta biomassan poltosta. [2]

Hiukkaset (PM) (kuva 2) koostuvat useista komponenteista, ja ne ovat olomuodoltaan kiinteitä tai nestemäisiä. Hiukkaset ovat luokiteltu pien (< 2,5 µm) ja mekaanisiin (> 2,5 µm) hiukkasiin. Hiukkasilla on erilaisia terveysvaikutuksia riippuen hiukkasten koosta. Mekaaniset hiukkaset eivät tunkeudu kovinkaan syvälle hengitystiessä, vaan ne jäävät pääosin keuhkoputkiin, mistä ne poistuvat noin vuorokauden kuluttua. Pienhiukkaset pääsevät syvemmälle keuhkoputkiin ja keuhkorakkuloihin. Ne voivat viipyä keuhkoissa vuosien ajan. Kaikista hienoimmat (< 1,0 µm) kokoluokan hiukkaset kulkeutuvat keuhkoista verenkiertoon asti. Erikokoiset hiukkaset pääsevät eri kohtiin elimistössä, joten niiden aiheuttamat vaikutukset poikkeavat toisistaan. Riippuen altistuksen määrästä ja hiukkasten koosta, ne voivat aiheuttaa mm. astmaa, keuhkoahtaumatautia, verisuonten kalkkeutumista, pahentaa sepelvaltimotautia aivoverenkiertosairauksia. Kaikista pienempien hiukkasten pitkäaikaisesta altistumisesta ei vielä ole tarpeeksi tutkimustietoa. [3]

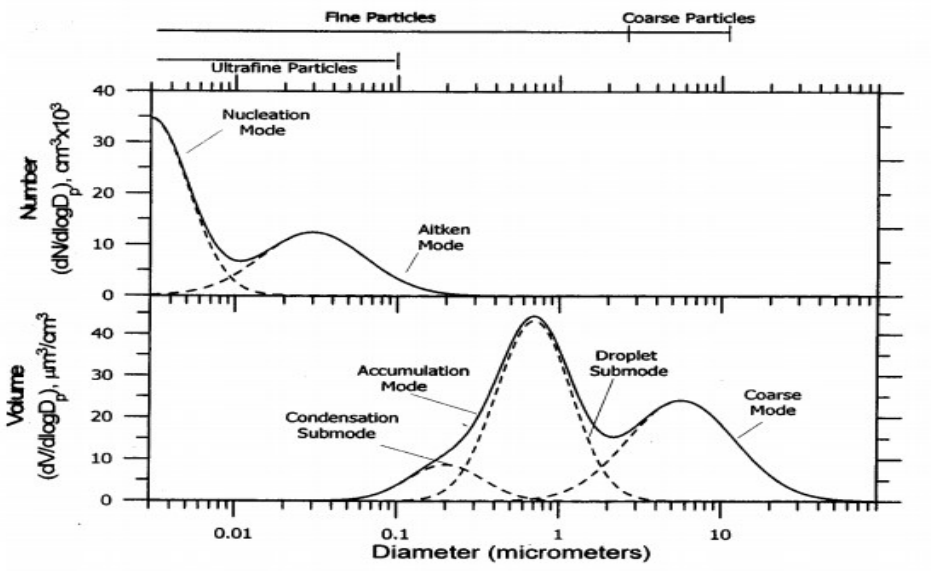


Kuva 2. Ilmassa olevien aineiden tyyppi sekä kokojakauma mikrometreinä [4].

Kuvissa 3 ja 4 on esitetty eri moodit, joissa hiukkasia ilmenee. Tyypillisesti pakokaasupäästöt voidaan jaotella kahteen eri moodiin, nukleaatio- ja akkumulaatiomoodiin. Nukleaatiomoodin hiukkasten koko jakauma on 5…50 nm, ja tällä alueella esiintyy suurin osa hiukkasten lukumäärästä, jopa 90 % kokonaislukumäärästä. Hiukkasmassan suhteen nukleaatiomoodi käsittää noin 20 % kokonaishiukkasmassasta. Akkumulaatiomoodissa sijaitsee suurin osa hiukkasmassasta. Kokoluokka hiukkasilla on 300 nm:n molemmin puolin. Karkeissa (coarse) eli > 2,5 µm:n kokoluokan hiukkasissa massapitoisuus on 5...20 %, ja hiukkaset ovat epätäydellisen palamisen yhteydessä syntyneitä nokihiukkasia. [4, s. 15–16.]



Kuva 3. Hiukkasmoodit [5, s. 2].



Kuva 4. Hiukkasmoodit [6, s. 4].

2.2 Eurooppalainen raskaiden ajoneuvojen pakokaasulainsäädäntö

Eurooppalainen pakokaasulainsäädäntö käsitetään yleisesti termein Euro 1–6. Lainsäädäntö koskee kaikkia raskaita ajoneuvoja, joiden suurin sallittu enimmäismassa on > 3500 kg ja jotka on varustettu puristussytytteisellä tai kipinäsytytteisellä maa-, nestekaasu- tai bensiinimoottorilla. Asetukset on otettu alun perin käyttöön direktiivillä 88/77/ETY[2871] [7].

Euro 1 -standardi otettiin käyttöön vuonna 1992 (direktiivi 91/44/EC) [8], jota seurasi Euro 2 -standardi vuonna 1996 (direktiivi 94/12/EC) [9]. Asetukset kosivat kuorma-auto- ja kaupunkiliikennettä. Kaupunkibussien standardit olivat vapaaehtoisia. Euro 3 -standardi otettiin käyttöön vuonna 2000 (direktiivi 98/68/EC) [10] ja Euro 4 -standardi vuonna 2005 (direktiivi 2002/80/EC) [10]. Euro 5 -standardi vuonna 2008 (direktiivi 715/2007/EC)[12] Euro 6 -standardi tuli voimaan vuonna 2013 [12].

Pakokaasulainsäädännön vahvistaa Euroopan unionin komissio. Pakokaasupäästöjä valvotaan unionin alueella valvontaan erilaisilla testausmenetelmillä, joita ovat mm. sarjatuotannon testaus, tyyppihyväksyntätestaus ja kenttävalvonta.

Sarjatuotannon testaus suoritetaan pääsääntöisesti ajoneuvo- tai moottorivalmistajan toimesta. Viranomaiset voivat suorittaa pistokokeita osalle sarjatuotannon ajoneuvoista.

Ajoneuvo- ja moottorityypit pitää tyyppihyväksyttää ennen niiden markkinointia. Tyyppihyväksyntää varten ajoneuvolle tai moottorille suoritetaan pakokaasutestit. Ajoneuvon tai moottorin tulee läpäistä ennalta määrätty testisykli. Testissä mitataan hiilivedyt, hiilimonoksidi, typenoksidit, hiukkaspäästö sekä savutusarvo (diesel). [13]

2.2.1 Euro 6 -standardin mukainen testi raskaalle kalustolle

Euro 6 -standardin mukainen testi raskaalle kalustolle koostuu useammasta osasta, vakiokuorma testistä, transienttitestistä, Off-Cycle-testistä, käytönmukaisesta testistä sekä päästöjen kestävyystestistä. [13]

2.2.2 Vakiokuormatesti

Vakiokuormatesti kulkee nimellä WHSC (World Harmonized Stationary Cycle) ja koskee dieselmoottorilla varustettuja raskaita ajoneuvoja. Testi suoritetaan dynamometrillä, jossa ajoneuvoa ajetaan eri nopeuksilla ja kuormilla tietyn aikaa. Moottori on lämmin ennen testiä. Testisykli on esitetty taulukossa 1. Taulukossa 1 näkyvä painokerroin on vain referenssinä. Vastus dynamometrille tulee laskea aina ajoneuvokohtaisesti. [13]

Taulukko 1. WHSC-sykli [13].

Table 1
World Harmonized Stationary Cycle (WHSC)

Mode	Speed	Load	Weighting Factor	Mode Length†
-	%	%	-	s
0	Motoring	-	0.24	-
1	0	0	0.17/2	210
2	55	100	0.02	50
3	55	25	0.10	250
4	55	70	0.03	75
5	35	100	0.02	50
6	25	25	0.08	200
7	45	70	0.03	75
8	45	25	0.06	150
9	55	50	0.05	125
10	75	100	0.02	50
11	35	50	0.08	200
12	35	25	0.10	250
13	0	0	0.17/2	210
Total			1	1895

† Including 20 s ramp

Taulukossa 2 on esitetty WHSC:n raskaita ajoneuvoja koskevat päästörajat. Huomattavaa on, että testisyklit vaihtelevat euroluokituksien mukaan.

Taulukko 2. WHSC:n päästörajat [13].

Table 1
EU Emission Standards for Heavy-Duty Diesel Engines: Steady-State Testing

Stage	Date	Test	CO	HC	NOx	PM	PN	Smoke
			g/kWh				1/kWh	1/m
Euro I	1992, ≤ 85 kW	ECE R-49	4.5	1.1	8.0	0.612		
	1992, > 85 kW		4.5	1.1	8.0	0.36		
Euro II	1996.10		4.0	1.1	7.0	0.25		
	1998.10		4.0	1.1	7.0	0.15		
Euro III	*1999.10 EEV only*	ESC & ELR	*1.5*	*0.25*	*2.0*	*0.02*		*0.15*
	2000.10		2.1	0.66	5.0	0.10[a]		0.8
Euro IV	2005.10		1.5	0.46	3.5	0.02		0.5
Euro V	2008.10		1.5	0.46	2.0	0.02		0.5
Euro VI	2013.01	WHSC	1.5	0.13	0.40	0.01	8.0×10^{11}	

a - PM = 0.13 g/kWh for engines < 0.75 dm^3 swept volume per cylinder and a rated power speed > 3000 min^{-1}

2.2.3 Transienttitesti

Transientitesti kulkee nimellä WHTC (World Harmonized Transient Cycle) ja koskee puristus- tai kipinäsytytteisellä moottorilla varustettuja raskaita ajoneuvoja. Testi suoritetaan dynamometrillä kaksi kertaa sekä kylmällä että lämpimällä moottorilla. Testin on tarkoitus simuloida ns. normaalia ajoa liikenteessä. [13]

Kuvassa 5 on esitetty testisykli.

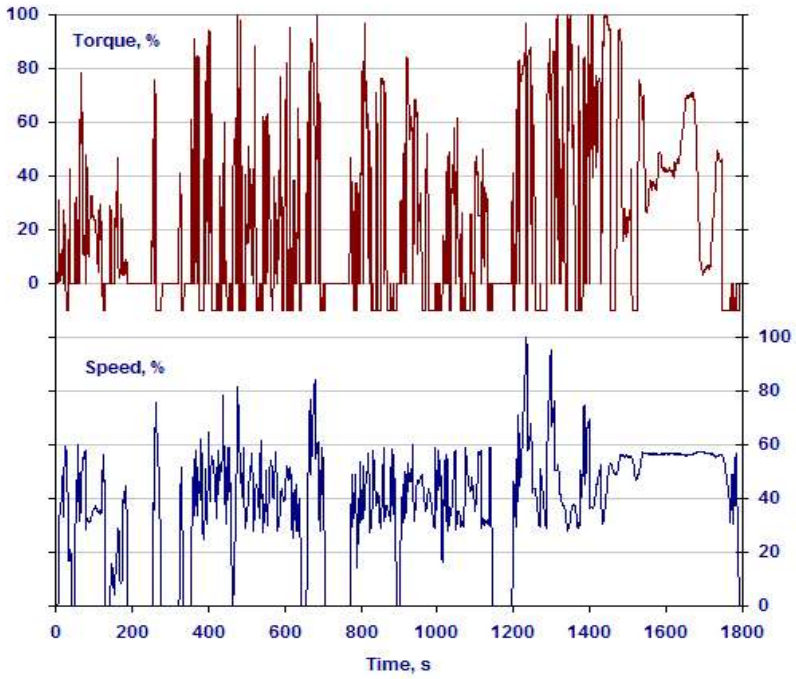


Figure 1. World Harmonized Transient Cycle (WHTC)

Note: Negative torque values are arbitrary representation of closed rack motoring

Kuva 5. WHTC-sykli [13].

Taulukossa 3 on esitetty päästörajat transienttitestille. Testisykli muuttuu euroluokituksen mukaan. Huomattavaa on myös, ettei kipinäsytytteisille moottoreille ole vielä määrätty hiukkasmäärää koskevaa päästörajaa.

Taulukko 3. WHTC:n päästörajat [13].

Table 2
EU Emission Standards for Heavy-Duty Diesel and Gas Engines: Transient Testing

Stage	Date	Test	CO	NMHC	CH_4[a]	NOx	PM[b]	PN[e]
			g/kWh					*1/kWh*
Euro III	*1999.10 EEV only*	ETC	*3.0*	*0.40*	*0.65*	*2.0*	*0.02*	
	2000.10		5.45	0.78	1.6	5.0	0.16[c]	
Euro IV	2005.10		4.0	0.55	1.1	3.5	0.03	
Euro V	2008.10		4.0	0.55	1.1	2.0	0.03	
Euro VI	2013.01	WHTC	4.0	0.16[d]	0.5	0.46	0.01	6.0×10^{11}

a - for gas engines only (Euro III-V: NG only; Euro VI: NG + LPG)
b - not applicable for gas fueled engines at the Euro III-IV stages
c - PM = 0.21 g/kWh for engines < 0.75 dm^3 swept volume per cylinder and a rated power speed > 3000 min^{-1}
d - THC for diesel engines
e - for diesel engines; PN limit for positive ignition engines TBD

2.2.4 Off-Cycle-testi

Euro 6 -standardi toi OCE (Off-cycle emissions), testausvaatimukset. OCE-mittaukset, suoritetaan tyyppihyväksyntätestauksen aikana. Testit noudattavat NTE (not-to-exeed) -rajalähestymistapaa. Moottorikartalle määritellään ohjausalue (kaksi määritelmää, yksi moottoreille, joiden nimellisnopeus on < 3000 rpm, ja toinen moottoreille, joiden nimellisnopeus on ≥ 3000 rpm). Ohjausalueelle tehdään taulukko, minkä jälkeen testissä valitaan kolme satunnaista osaa taulukosta, joihin suoritetaan viisi mittausta per osa. [13]

2.2.5 Käytönmukainen testi

Käytönmukainen (In-Use) testi suoritetaan kenttäolosuhteissa käyttäen PEMS (Portable Emmission Measurement System) -laitteistoa. Testi suoritetaan nopeuksissa kaupunki 0–50 km/h, maaseutu 50–75 km/h sekä maantie > 75 km/h. Tarkat prosenttiosuudet näistä olosuhteista riippuvat ajoneuvoluokasta. Ensimmäinen käytönmukainen testi tulee tehdä tyyppihyväksyntätestauksessa. [13]

2.2.6 Päästöjen kestävyys

Valmistajan on osoitettava, että moottorit läpäisevät päästörajat käyttöikänsä ajan [13]. Päästörajat määräytyvät ajoneuvoluokittain, ja ne on esitetty taulukossa 4.

Taulukko 4. Päästöjen kestävyys [13].

Table 3
Emission Durability Periods

Vehicle Category†	Period*	
	Euro IV-V	Euro VI
N1 and M2	100 000 km / 5 years	160 000 km / 5 years
N2 N3 ≤ 16 ton M3 Class I, Class II, Class A, and Class B ≤ 7.5 ton	200 000 km / 6 years	300 000 km / 6 years
N3 > 16 ton M3 Class III, and Class B > 7.5 ton	500 000 km / 7 years	700 000 km / 7 years

† Mass designations (in metric tons) are "maximum technically permissible mass"
* km or year period, whichever is the sooner

2.2.7 Kenttävalvonta

Kenttävalvontaa suorittaa valtiollinen taho. Suomessa päästöjen valvontaa suoritetaan ajoneuvokatsastuksen yhteydessä. Ajoneuvon päästöt tarkistetaan ja arvostellaan yhtenä kokonaisuutena. Tämä tarkoittaa sitä, että OBD- (On Board Diagnostics) ja päästömittaus on suoritettava samassa yhteydessä.

Ottomoottorilla varustetulle ajoneuvolle, joka on käyttöön otettu vuonna 1978 tai sen jälkeen, tulee suorittaa pakokaasupäästöjen mittaus. Mitattavat suureet, hiilimonoksidi (CO), hiilidioksidi (CO_2), hiilivedyt (HC), happi (O_2), tulee mitata joutokäynnillä sekä vastaavalla pyörintänopeudella. Kolmitiekatalysaattorilla varustettu ajoneuvo mitataan myös korotetulla pyörintänopeudella ja siitä mitataan myös lambda-arvo. Ajoneuvoille, jotka on käyttöön otettu 1.1.2001 ja varustettu OBD-järjestelmällä, tulee tehdä OBD-järjestelmän tarkastus sekä korotetun pyörintänopeuden päästömittaus. [14, s. 2–6.]

Taulukossa 5 on esitetty pakokaasumittauksen raja-arvot.

Taulukko 5. Päästörajat ottomoottorille Suomessa [14, s. 2].

ajoneuvon käyttöönottoajankohta tai moottorityyppi	joutokäynnillä		väh. 2000 rpm pyörintänopeudella		
	CO (%)	HC (ppm)	CO (%)	HC (ppm)	lambda
ennen 1.10.1986	4,5	1000	-	-	-
1.10.1986 tai sen jälkeen	3,5	600	-	-	-
varustettu kolmitoimikatalysaattorilla	0,5	100	0,3	100	1±0,03
varustettu OBD-järjestelmällä	-	-	0,2	100	1±0,03

Dieselmoottorilla varustetun ajoneuvon pakokaasumittaus poikkeaa ottomoottorimittauksessa. Dieselmoottorisesta ajoneuvosta mitataan ajoneuvon savutusarvo (k).

Ajoneuvon pakokaasupäästöt tarkastetaan mittaamalla kuormittamattoman moottorin savutus vapaassa kiihdytyksessä joutokäynniltä ruiskutuksen katkaisun pyörimisnopeuteen asti. Savutusarvon mittaus tapahtuu valonläpäisyyn perustuvalla opasiteettimenetelmällä.

Savutusmittauksessa saatujen absorptio- eli k-kertoimien sallitut arvot ovat seuraavat:

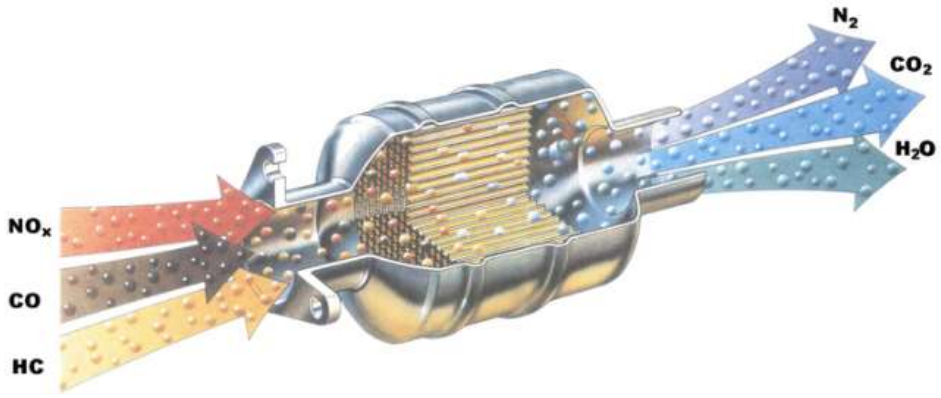
- vapaasti hengittävillä dieselmoottoreilla
  $k \leq 2{,}5\ m^{-1}$
- turboahtimella varustetuilla dieselmoottoreilla
  $k \leq 3{,}0\ m^{-1}$
- Euro 4 -päästöluokan hyväksytyssä dieselmoottorissa
  $k \leq 1{,}5\ m^{-1}$
- Euro 5/6 -päästöluokkaan hyväksytyssä dieselmoottorissa
  $k \leq 0{,}5 m^{-1}$.

Euro 5/6 -savutusarvoa ($k \leq 0{,}5\ m^{-1}$) käytetään, jos valmistaja ei ole ilmoittanut tyyppikilvessä savutusarvoa. OBD-järjestelmä tulee tarkistaa ajoneuvoista, jotka on otettu käyttöön 1.1.2007 tai sen jälkeen. Lisätietoa pakokaasumittauksista Liikenteen turvallisuusviraston ohjeista pakokaasumittauksiin otto- ja dieselmoottorisille ajoneuvoille. [15, s. 3–6.]

2.3 Pakokaasujen jälkikäsittely

Pakokaasupäästöjen hillitsemiseksi on kehitetty erilaisia jälkikäsittelylaitteita, mm. kolmitiekatalysaattori, hapetuskatalysaattori, SCR-katalysaattori.

Kolmitiekatalysaattori (kuva 6) on nimensä mukaisesti kolmitoiminen katalysaattori. Sen tehtävä on muuntaa kolmea saastekomponenttia vähemmän haitalliseksi. Katalysaattori hapettaa hiilimonoksidia (CO) ja muuntaa sitä hiilidioksidiksi (CO_2). Samalla se muuntaa palamattomia hiilivetyjä (HC) vedeksi (H_2O) ja pelkistää typenoksideja (NO_X) typeksi (N_2). Katalysaattori toimii parhaiten, kun moottorin seossuhde on lähellä lambda 1:tä (14,7:1), tällöin moottorissa syntyvien hapetettavien ja pelkistettävien aineiden määrät ovat sellaisia, että katalysaattori pystyy käsittelemään ne tehokkaasti. Katalysaattori tarvitsee reaktioiden syntymiseksi riittävän lämpötilan vähintään 250 °C. Paras puhdistusteho saavutetaan, kun katalysaattorin lämpötila on 400–800 °C. Kyseistä katalysaattoria ei voida käyttää laihaseosmoottoreilla, vaan se toimii parhaiten stökiömetrisellä seossuhteella toimivalla moottorilla. [16, s. 1–16.]



Kuva 6. Kolmitiekatalysaattori [17, s. 11].

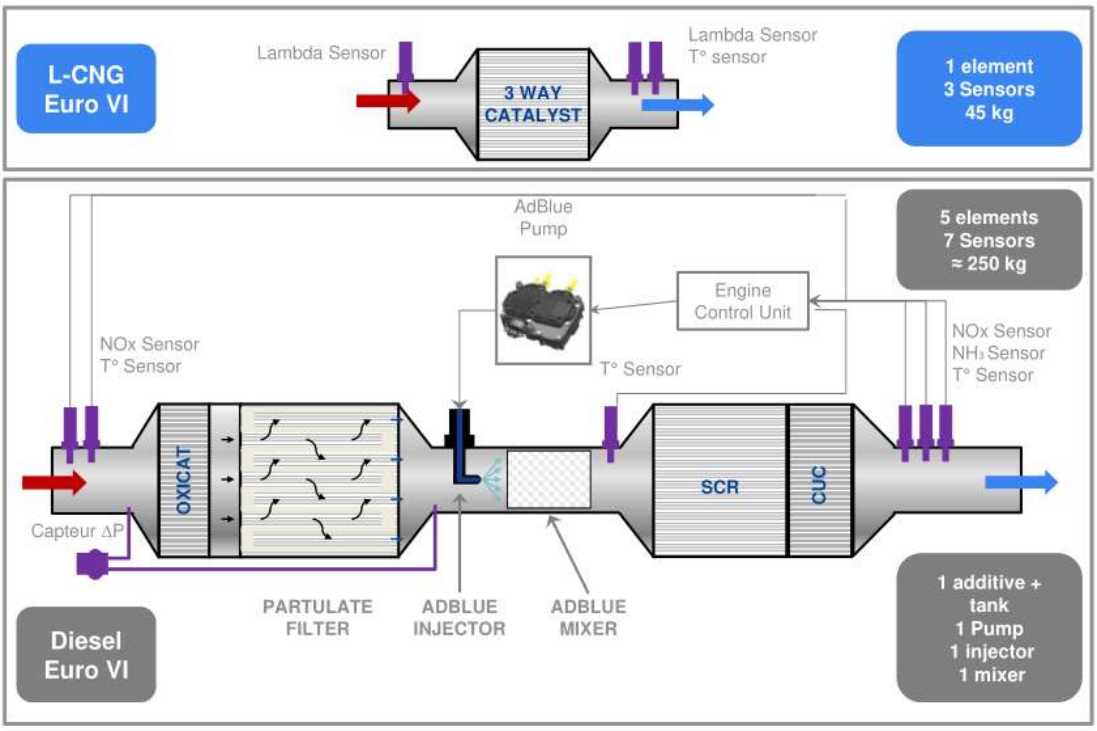
Dieselhapetuskatalysaattori (DOC), on osa dieselmoottorin pakokaasun jälkikäsittelylaitteistoa. Sen tarkoitus tuottaa vähemmän haitallisia yhdisteitä hapettamalla dieselmoottorissa syntyviä hiilimonoksidi (CO), hiilivety (HC) ja Typpioksidi (NO_X) -päästöjä. Joissain tapauksissa typenoksidien (NO_X) hapetusta voidaan pitää ei toivottuna, mutta se on tärkeä osa nykyisissä dieselmoottorien päästöjenhallintajärjestelmissä. Järjestelmän synnyttämää typpidioksidia (NO_2) voidaan tehokkaasti käyttää hiukkassuodattimen (DPF) regenerointi vaiheessa tai parantamaan SCR-katalysaattorin toimintaa. Hapetuskatalysaattori on usein suunniteltu korkean typpidioksidipitoisuuden (NO_2) saavuttamiseksi, jotta se tukisi paremmin DPF/SCR -järjestelmää. Toisaalta typpidioksidi (NO_2) pitoisuudet saattavat nousta hapetuskatalysaattorin käytöstä johtuen. [16, s. 1–16.]

Dieselmoottorissa syntyy huomattavasti vähemmän epätäydellisen palamisesta johtuvia komponentteja hiilimonoksidia (CO) sekä hiilivetyjä (HC), mutta pakokaasu sisältää verrattain suuria määriä typenoksideja (NO_X) jotka muodostuvat korkeiden palolämpötilojen johdosta. Kemiallisten reaktioiden vuoksi hapetuskatalysaattorilla varustettu dieselmoottori muodostaa enemmän hiukkaspäästöjä, minkä takia dieselmoottori on yleensä varustettu hiukkasuodattimella. [16, s. 1–16.]

Hiukkassuodatin (DPF) on dieselajoneuvoissa käytettävä pakokaasuja puhdistava suodatin. Suodatin vähentää huomattavasti dieselmoottorin aiheuttamia hiukkaspäästöjä. Hiukkassuodattimia on sekä aktiivisia että passiivisia malleja. Aktiivinen suodatin puhdistaa itse itsensä siinä olevan katalyytin tai polttoainepolttimen avulla. Aktiivinen suodatin on ohjelmoitu puhdistamaan itsensä, kun suodattimeen varastoitu hiukkasmassa ylittää sille määrätyn raja-arvon. Varastoidut hiukkaset poltetaan joko lämmön tai typpidioksidin (NO_2) ja matalan lämmön avulla. Passiivinen suodatin on nimensä mukaisesti passiivinen ja se on aika-ajoin vaihdettava uuteen, kun suodatin on varastoitu täyteen. [16, s. 1–16.]

SCR-katalysaassori (Selective Catalytic Reduction) on tekniikka, jossa typenoksidipäästöjä (NO_X) vähennetään katalysaattorilla. Puhdistusmenetelmä perustuu siihen, että pakoputkeen suihkutetaan ureasta ja vedestä tehtyä liuosta, joka hajoaa lämpötilan johdosta ammoniakiksi ja hiilidioksidiksi. Ammoniakki reagoi pakokaasun typenoksidien kanssa katalysaattorissa jossa, nämä komponentit pelkistyvät typeksi (N_2) ja vesihöyryksi. SCR-katalysaattorin käyttö on yleistynyt nykyisten tiukkojen päästörajojen johdosta. [16, s. 1–16.]

Kuvassa 7 on esitetty Ivecon EURO 6 -päästöluokan täyttävä pakokaasunkäsittelylaitteisto LNG/CNG-moottorin ja dieselmoottorin osalta. Kuten kuvasta huomataan, maakaasua käyttävän ajoneuvon pakokaasunpuhdistusjärjestelmä sisältää vain kolmitiekatalysaattorin, kaksi happianturia sekä lämpötila-anturin. Vastaavasti dieseliä käyttävän ajoneuvon puhdistusjärjestelmä koostuu kolmesta lämpötila-anturista, kahdesta typpioksidianturista, ammoniakkianturista sekä hiukkassuodattimen tilaa valvovasta anturista. Järjestelmässä on hapettava katalysaattori, hiukkassuodatin, SCR-järjestelmä (katalysaattori ja urearuiskutusjärjestelmä) sekä keräävä NO_X-katalysaattori.



Kuva 7. Pakokaasun jälkikäsittelylaitteisto. [18, s. 12].

Pakokaasunkierrätysventtiili (EGR), nimensä mukaisesti kierrättää moottorissa syntynyttä pakokaasua takaisin moottoriin. Pakokaasun takaisinkierrätyksellä pyritään hallitsemaan palotapahtuman lämpötilaa, mikä johtaa pienempiin typenoksidipäästöihin (NO_x) ja lisää moottorinhyötysuhdetta. [16, s. 1–16.]

3 Tutkimushanke

Tässä osiossa kuvataan tutkimushankkeen tavoitteet, mitattavat ajoneuvot, mittausjärjestelyt, mittauslaitteet, mittaustulokset ja päästöarvojen laskenta, arvioidaan hankkeen onnistuminen ja esitetään johtopäätökset mittauksista.

3.1 Tutkimushankkeen tavoitteet

Tutkimushankkeen tavoitteena oli selvittää reittiajossa olevan, uuden Euro 6 -luokituksen omaavan LNG/CNG-rekan pakokaasupäästöt verrattuna vastaavan Euro 6 -luokituksen omaavaan dieselrekkaan.

3.2 Mitattavat ajoneuvot

Tässä osuudessa esitellään hankkeessa mitattavat ajoneuvot.

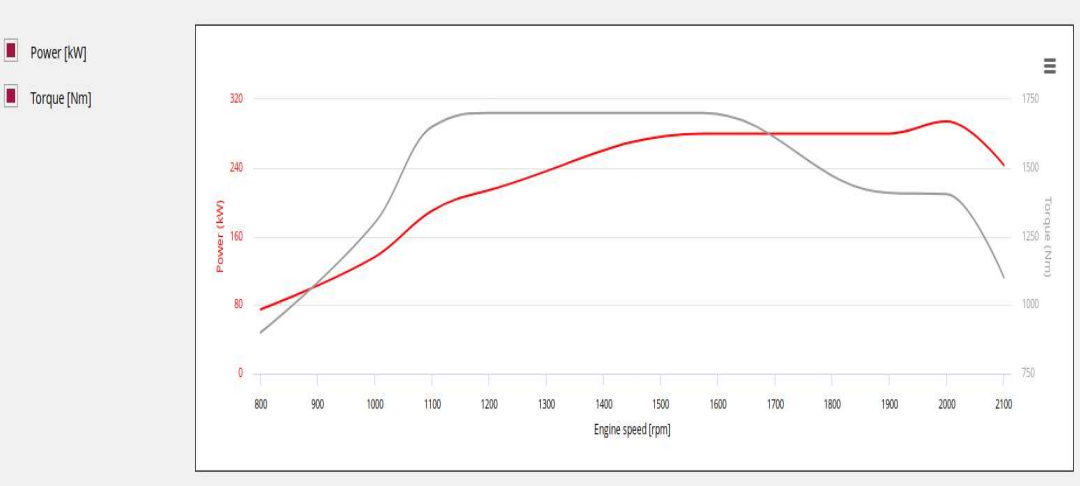
3.2.1 Iveco Stralis NP

Iveco Stralis NP on puhtaasti maakaasulla toimiva rekka. Ajoneuvoa on saatavissa, kolmella eri moottorivaihtoehdolla (taulukko 6.), joista kaksi tehokkainta versiota on saatavissa myös nesteytetyllä maakaasulla toimivana. Iveco ilmoittaa, että kyseisen ajoneuvon typpioksidipäästöt vähenevät 60 %, metaanipäästöt 80 % ja hiukkaspäästöt 99 % Euro 6 -raja-arvoista. Moottorinvalmistaja ilmoittaa polttoainekustannussäästöksi jopa 30 % vastaavaan dieselmoottoriin. [21, s. 1–34,22.]

Taulukko 6. Stralis NP -moottorit [21, s. 11].

FPT ENGINE	MAXIMUM POWER			MAXIMUM TORQUE	
	Rating		@ rpm	Rating [Nm]	@ rpm
	[hp]	[kW]			
CURSOR 8	270	200	2,000	1,100	1,100 ÷ 1,735
	300	221	2,000	1,200	1,200 ÷ 1,760
	330	243	2,000	1,300	1,200 ÷ 1,785
CURSOR 9	400	294	2,000	1,700	1,200 ÷ 1,575
CURSOR 13	460	338	1,900	2,000	1,100 ÷ 1,600

Mittauksissa käytetty ajoneuvo oli varustettu Cursor 9 (kuva 9) -moottorilla, joka toimii sekä nesteytetyllä sekä paineistetulla maakaasulla. Mittauksissa käytettiin nesteytettyä maakaasua. Ajoneuvo oli varustettu automatisoidulla vaihteistolla. Pakokaasun jälkikäsittely laitteisto koostuu vain kolmitiekatalysaattorista, koska moottori toimii stökiömetrisellä seossuhteella, minkä johdosta muille puhdistuslaitteille ei ole tarvetta.



Kuva 8. Cursor 9, vääntö-tehokäyrä [22].

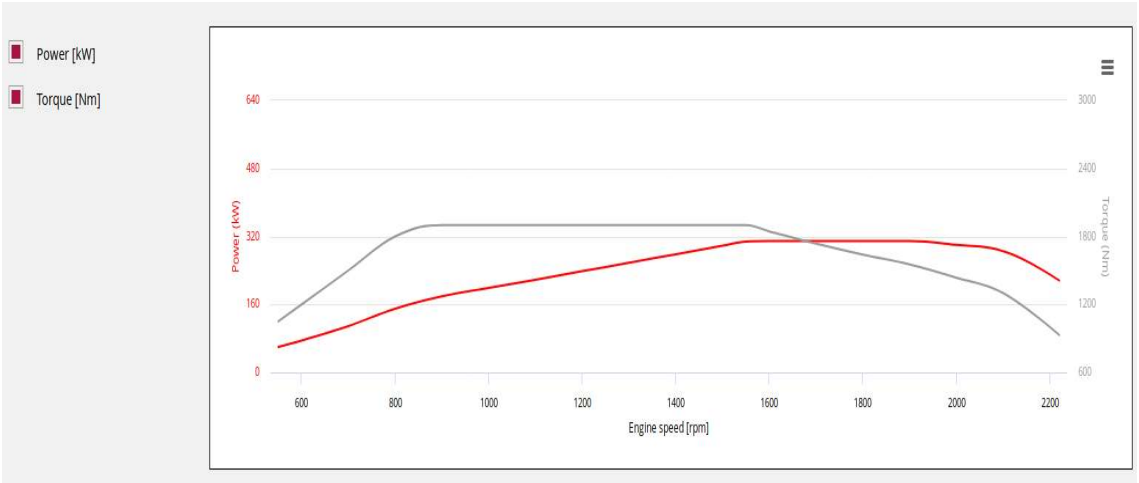
3.2.2 Iveco Stralis XP

Iveco Stralis XP on dieselvastine Iveco Stralis NP:lle. Dieselversion sanotaan olevan CO_2-päästöiltään 11 % pienempi kuin edeltäjänsä sekä 5,6 % edullisempi kokonaisomistuskustannuksiltaan. [19, s 4.] Ajoneuvo on saatavissa kolmella eri moottorivaihtoehdolla (taulukko 7).

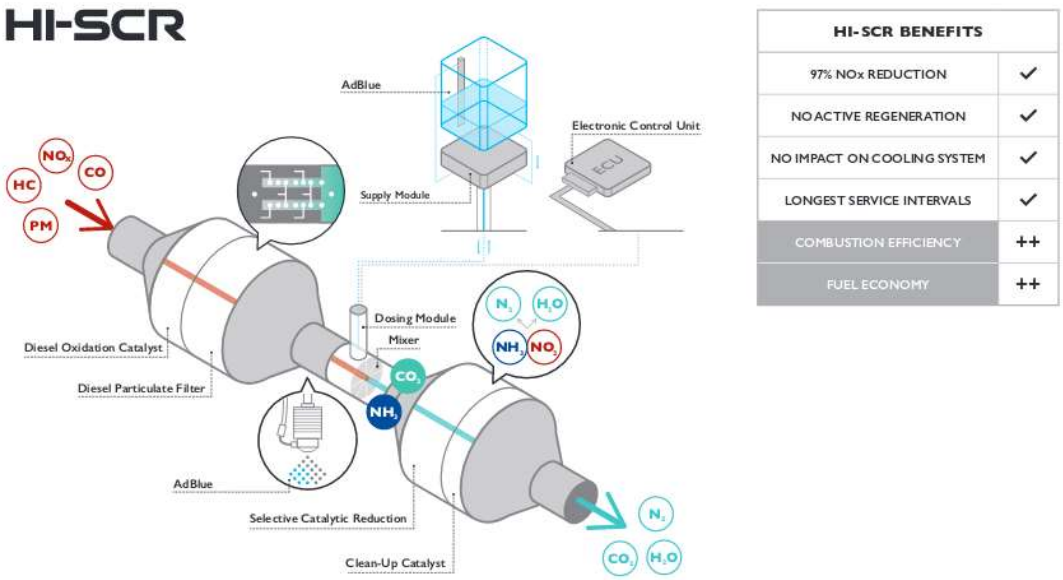
Taulukko 7. Stralis XP, moottorit [19, s. 12]

	NEW STRALIS EURO VI/STEP C			
	DISPLACEMENT	TURBO	POWER	TORQUE
			hp @ rpm	Nm @ rpm
CURSOR 9	8.7 litres	VVG	310 @ 1,675 – 2,200	1,300 @ 1,100 – 1,675
			330 @ 1,655 – 2,200	1,400 @ 1,100 – 1,655
			360 @ 1,530 – 2,200	1,650 @ 1,200 – 1,530
		eVGT	400 @ 1,655 – 2,200	1,700 @ 1,200 – 1,655
CURSOR 11	11.1 litres	eVGT	420 @ 1,475 – 1,900	**2,000 @ 870 – 1,475**
			460 @ 1,500 – 1,900	2,150 @ 925 – 1,500
			480 @ 1,465 – 1,900	**2,300 @ 970 – 1,465**
CURSOR 13	12.9 litres	eVGT	**510 @ 1,560 – 1,900**	2,300 @ 900 – 1,560
			570 @ 1,605 – 1,900	2,500 @ 1,000 – 1,605

Mittauksissa käytetty Iveco Stralis XP oli varustettu 420 -hevosvoimaisella Cursor 11 -moottorilla (kuva 9) sekä automatisoidulla vaihteistolla. Ajoneuvo oli varustettu Ivecon HI-SCR-pakokaasunjälkikäsittelylaitteistolla (kuva 10).



Kuva 9. Cursor 11, teho-vääntökäyrä [23].



Kuva 10. HI-SCR-järjestelmä [19, s. 10].

Kumpikin mittauksissa käytetty ajoneuvo oli mittausten ajan lainassa Iveco Finland Oy:lta. Iveco Finland Oy asensi myös mitattaviin ajoneuvoihin Proventian typpioksidimittalaitteiston, jonka Liikenteen turvallisuusvirasto Trafi hankki tutkimushanketta varten.

Mittauksessa käytetty kärry sekä hyötykuorma tulivat Containership Oyj:ltä. Polttoaineet maakaasun osalta toimitti Gasum Oy.

Mitattavat ajoneuvot vetivät hyötykuormaa, joka oli sijoitettu 45 jalan merikontteihin. Kärry oli koko mittausten ajan sama. Kuorma vaihtui mittausten aikana kolme kertaa. Alla on eroteltu kuormien massat, joita mittauksessa käytettiin.

Päivät 1 ja 2 ovat maakaasurekan vetämät kuormat. Päivät 3 ja 4 ovat dieselrekan vetämät kuormat. Kuormat olivat seuraavat:

- päivä 1 CNEU455***-* 10 480 kg
- päivä 2 KLCU450***-* 11 201 kg
- päivä 3 KLCU450***-* 13 116 kg
- päivä 4 KLCU450***-* 13 116 kg.

3.3 Mittausjärjestelyt

Mittaukset järjestettiin neljänä päivänä, joista kummallekin mitattavalle ajoneuvolle oli varattu kaksi mittauspäivää. Mittaus reitiksi valittiin Helsinki–Paimio–Helsinki. Reitti kulki Seututie 110:tä pitkin, koska mittaus tapahtumat sijoittuivat välille 06.00–16.00, ja ennakkoon selvitetty liikennemäärä kyseisellä tiellä oli huomattavasti pienempi kuin vieressä kulkevalla Valtatie 1:llä. Mittauksen kannalta mitatut päästöt oli helpompi kohdentaa mitattavaan ajoneuvoon käyttämällä vähemmän liikennöityä väylää.

Reitti jaoteltiin neljään eri osaan, joista kolme oli puhtaasti maantiellä ajoa ja yksi hieman kontrolloidumpi mittaus useammalla toistolla. Reitistä jätettiin pois siirtymä lähtöpaikalta (Vuosaaren satama) tielle 110 sekä Salon taajama-alue. Kyseiset osuudet pudotettiin pois, koska taustapäästöt olivat muusta liikenteestä johtuen suuret eikä näin ollen olisi voitu yksilöidä mitattavan ajoneuvon päästöjä. Mittaukset toteutettiin jahtausmittausperiaatteella, josta Metropolia Ammattikorkeakoululla on aikaisempia kokemuksia useiden eri tutkimushankkeiden yhteydestä. Reitit on esitelty liitteessä 5.

4 Mittalaitteet

Tässä osiossa käsitellään mittauksissa käytettyjä mittalaitteita, joita oli sekä mittausajoneuvossa että mitattavissa ajoneuvoissa.

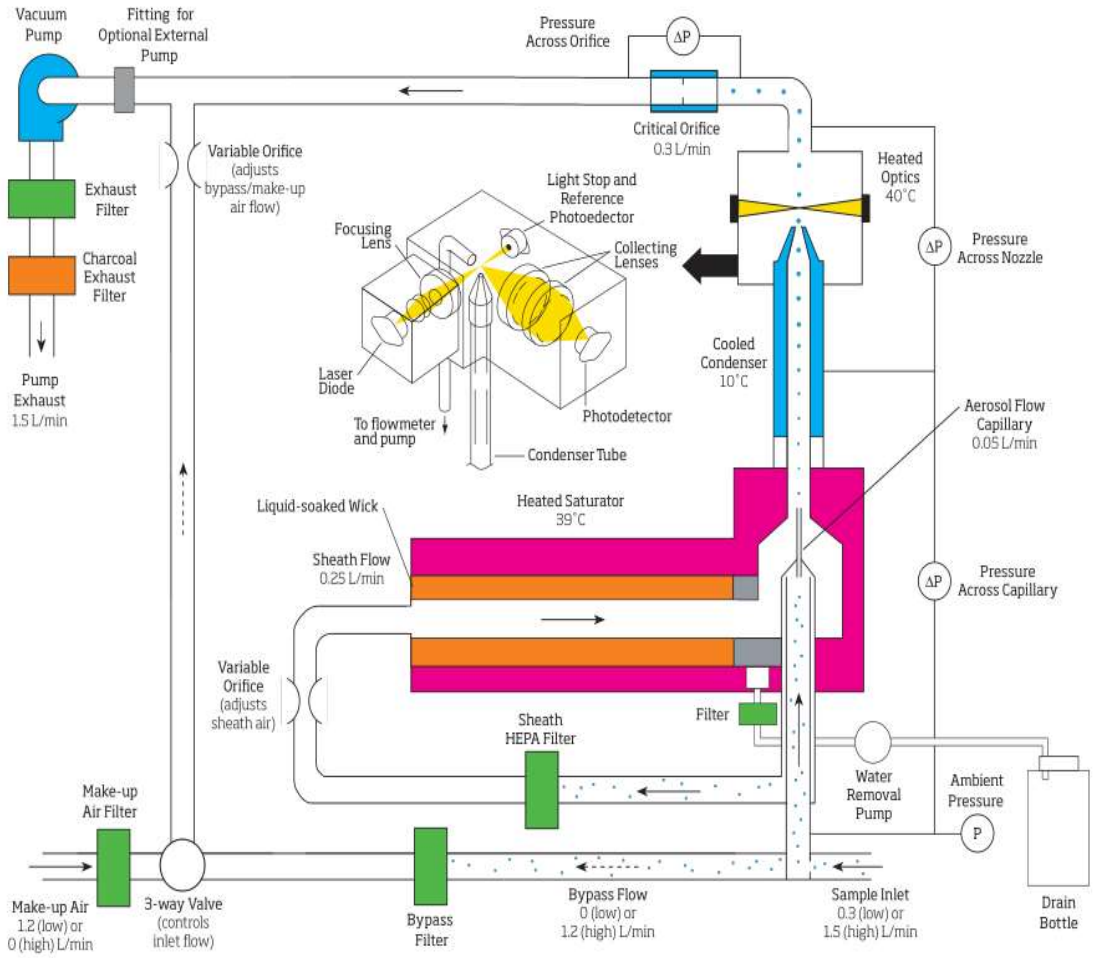
4.1.1 Tiivistymisydinlaskuri

Todella pienten alle yhden mikrometrin kokoisten aerosolihiukkasten havaitseminen on haastavaa. Optiset mittalaitteet jotka perustuvat näkyvän valon optiikkaan eivät pysty havaitsemaan alle puolen aallonpituuden kokoisia hiukkasia. Lyhentämällä valon aallonpituutta esimerkiksi käyttämällä valonlähteenä ultraviolettivaloa voidaan hiukkasten havaitsemista parantaa. Haittana pienemmästä aallonpituudesta voi sen suurempienerginen säteily aiheuttaa ei-toivottuja muutoksia mitattavissa hiukkasissa.

Tiivistymisydinlaskuri pyrkii kasvattamaan aallonpituutta pienempien hiukkasten kokoa tiivistymisen avulla optisesti havaittavaan kokoon. Operaatio tarvitsee suurta tiivistyvän aineen ylikyllästystilaa. Tämä saavutetaan ohjaamalla näyteaerosoli aluksi lämpimän,

kyllästystilassa olevan aineen kammion läpi viileämpään kammioon. Kylmässä tämä ylikyllästänyt höyry tiivistyy pienten aerosolihiukkasten ympärille, mikä mahdollistaa pienhiukkasten havaitsemisen optisesti. Haittapuolena tiivistymisydinlaskureissa on, että pienhiukkaset kasvavat lähes samankokoisiksi (leikkausrajaa suuremmat), jolloin koko- ja koostumustieto menetetään. Tiivistymisydinlaskureissa yleisesti käytetään tiivistyvänä aineen alhaisen tasapainohöyrypaineen omaavia alkoholeja, esimerkiksi n-butanolia. [24]

Mittauksissa käytetty laite oli TSI Inc:n valmistama, Ultrafine Condensation Particle Counter 3776 (kuva 18); laitteeseen viitataan jatkossa nimellä CPC. CPC pystyy havaitsemaan jopa 2,5 nm:n kokoiset hiukkaset 1 Hz:n taajuudella. Hiukkasten leikkausraja oli 2,5 nm [24]. Aerosolinäytteenottomäärä oli 1,5 l/min, ja tiivistyvänä aineena käytettiin n-butanolia.



Kuva 11. CPC [24, s. 2].

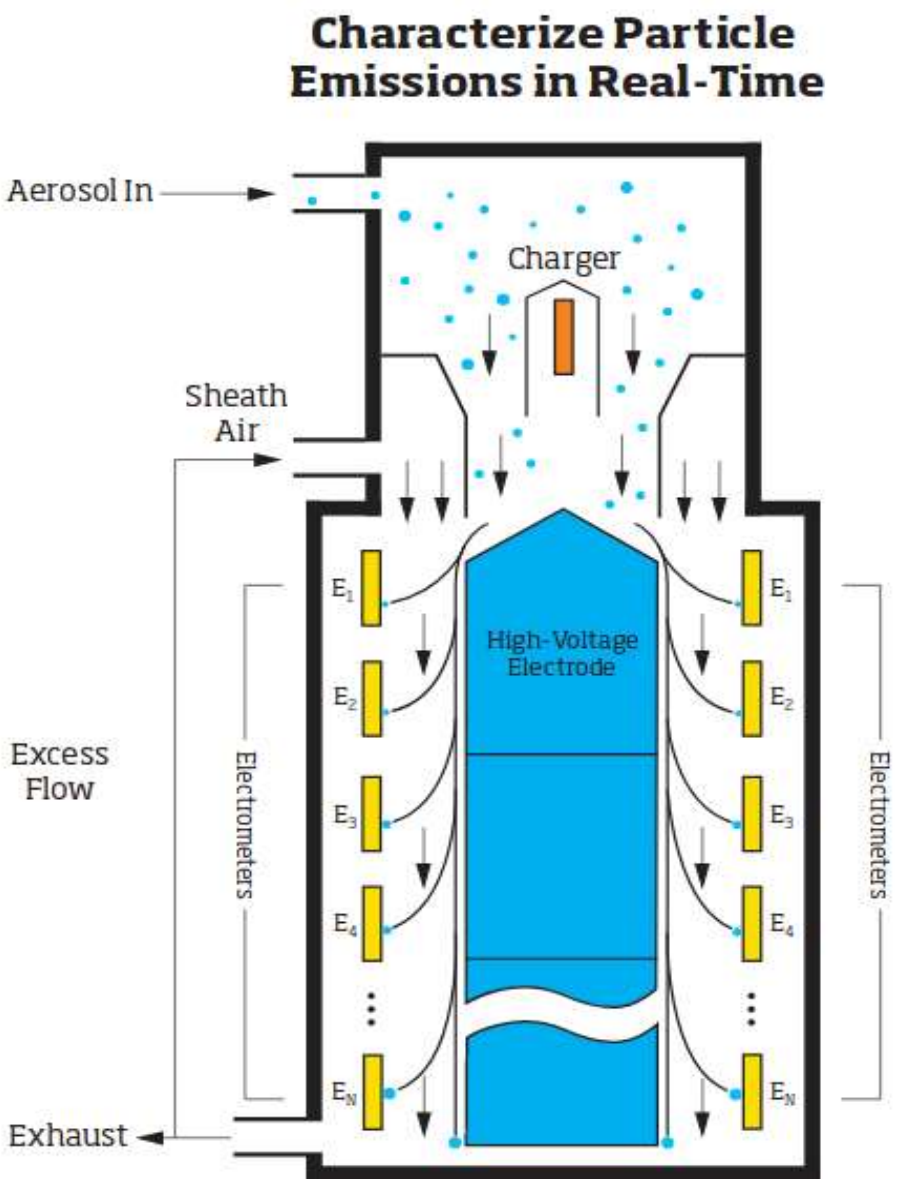
4.1.2 Sähköinen aerosolispektrometri

Sähköisen aerosolispektrometrin tarkoitus on mitata määrä- ja kokojakaumaa aerosolista käyttäen sähkövarausta ja puolijohdeilmaisimia. Tekniikka yhdistää diffuusio- ja kehävarauksen, mikä mahdollistaa hiukkasten havainnoin 10 nm:stä 10 µm:iin. Tämän tekniikan avulla voidaan saavuttaa jopa 1 Hz:n resoluutio, vaikkakin havainnointialuetta saatetaan joutua pienentämään. Laitetyypille luontaista on monivaraus, jonka johdosta tarkka aerosolin sähkövarauksen karakterisointi on tärkeää. [25]

Mittauksessa käytetty sähköinen aerosolispektrometri oli TSI Inc:n valmistama The Engine Exhaust Particle Sizer Model 3090; laitteeseen viitataan jatkossa nimellä EEPS. Laite pystyy havaitsemaan hiukkasten lukumäärän sekä hiukkasten koon 5,6 nm:sta 560 nm:iin. Laiteessa on yhteensä 32 kanavaa hiukkasten jakauman määrittämiseen. Laiteen näytteenotto on varustettu syklonilla, jonka katkaisuraja on 1 µm. Aerosolinäytteenottomäärä on 10 l/min. Laite pystyy näyttämään 10 hiukkasten kokojakaumaa sekunnissa. [25]

Laitteen toiminta

Laite imee aerosolinäytteen näytelinjaan, jonka jälkeen hiukkaset varataan positiivisiksi unipolaarisella koronavaraajalla. Hiukkaset siirtyvät varaajalta mittauskammioon jossa ne ohjataan lähelle kammion keskellä olevaa korkeajännite -elektrodia, joka on ympäröity suodatetulla ilmalla. Hiukkaset kulkeutuvat elektrodin myötäisesti alaspäin. Korkeajännite-elektrodiin syötetään positiivinen jännite, jolloin elektrodi muodostaa sähkökentän, joka aiheuttaa positiivisesti varattujen hiukkasten sinkoutumisen kohti ulkokehää kunkin hiukkasen sähköisen liikkuvuusominaisuuden mukaan. Varatut hiukkaset iskeytyvät mittauskammion kehällä oleviin elektrometreihin, joihin ne purkavat varauksensa. Suuren liikkuvuuden omaavat hiukkaset iskeytyvät mittauskammion yläosissa oleviin elektrometreihin ja pienen liikkuvuuden omaavat hiukkaset alempana kammiossa oleviin elektrometreihin. Varauksen purkaneet hiukkaset ohjataan laitteesta ulos. [25; 26, s. 19–22.]



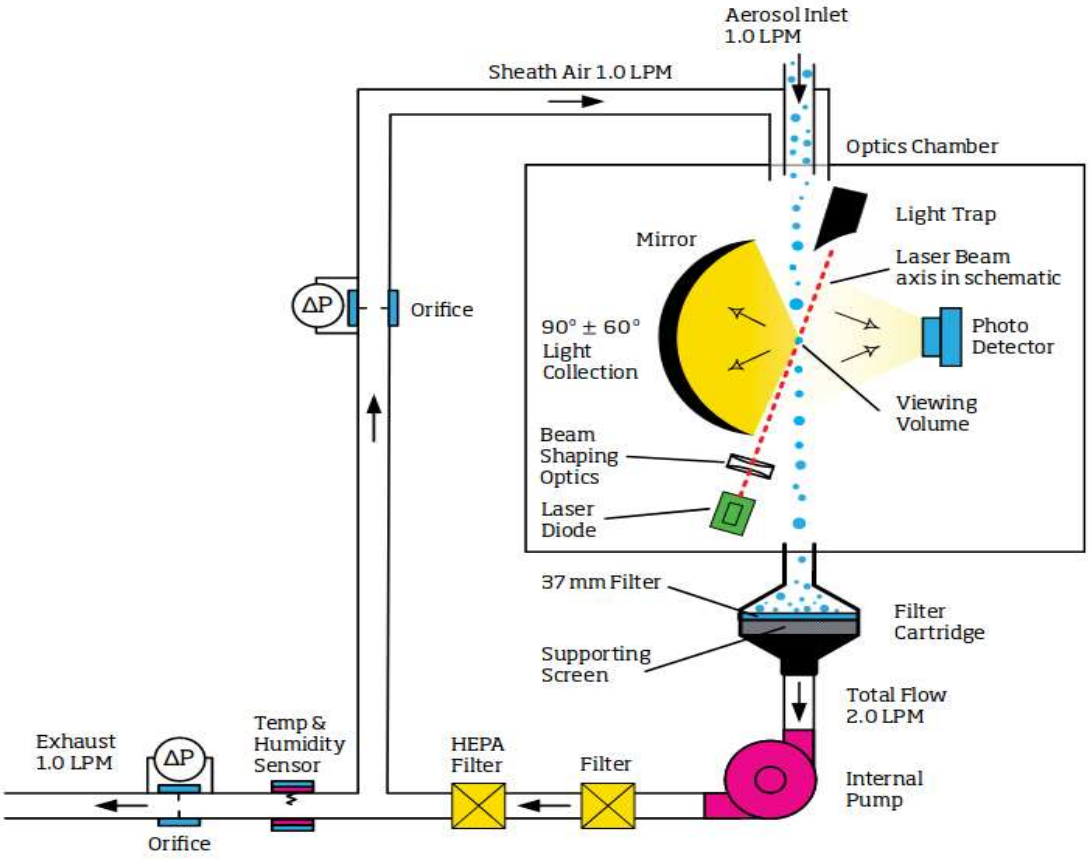
Kuva 12. EEPS [25, s. 5.]

4.1.3 Optinen hiukkaslaskuri

Optinen hiukkaslaskuri perustuu optiseen mittaukseen. Aerosoli syötetään mittauskammioon, jossa optiikka määrittää hiukkasten koon käyttämällä hyväkseen kammiossa olevan valon aallonpituutta, joka vaihtelee valokeilaan syötettävien hiukkasten mukaan niiden osuessa valokeilaan. Laitteessa on suodatin, johon optisesta kammiosta tuleva aerosoli kerätään. Tämä mahdollistaa gravitometrisen analyysin, jonka tuloksena saadaan

myös laitteesta hiukkasmassa. Vaihdettava suodatin mahdollistaa myös hiukkasnäytteen jatkokäsittelyn. [27]

Mittauksessa käytetty laite oli TSI Inc:n valmistama Optical Particle Sizer model 3330; laitteeseen viitataan jatkossa nimellä OPS. OPS pystyy määrittämään hiukkasten koon 0,3 µm:sta 10 µm:n ja hiukkasmassan 0,001 µg/m³:sta 275 000 µg/m³:iin. Laite on varustettu 16 kanavalla, joiden katkaisurajat ovat määriteltävissä. Laite ottaa aerosolinäytettä 1 l/min +-5 %:n tarkkuudella. Laite pystyy mittaamaan jopa 1 Hz:n taajuudella. [27]



Kuva 13. OPS [27].

4.1.4 Typpioksidianalysaattori

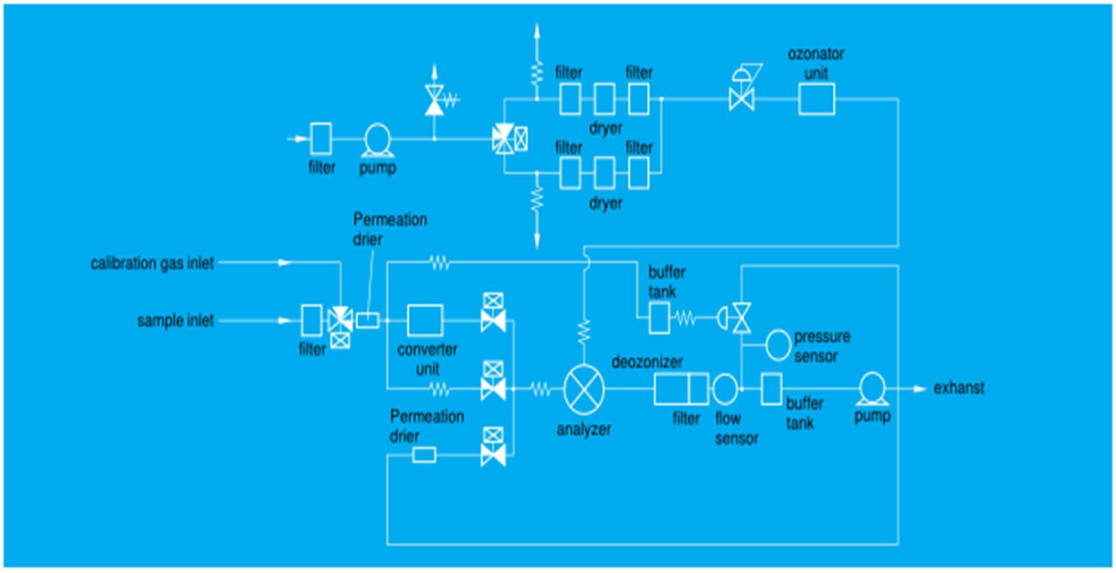
Mittauksessa käytetty typpioksidianalysaattori (NO_X) Horiba APNA-360CE toimii kemiluminenssiperiaatteella, jossa aerosolin typpimonoksidi (NO) yhdistetään otsoniin (O_3). Reaktio muodostaa typpidioksidia (NO_2) sekä happea (O_2). Osa reaktiossa muodostuneista typpidioksidista (NO_2) menevät virittäytyneet tilaan ($NO_2◊$). [28, s. 6.]

$$(NO + O_3 \longrightarrow NO_2◊ + O_2).$$
$$(NO_2◊ \longrightarrow NO_2\ hv)$$

Kun virittäytyneet molekyylit palaavat takaisin stabiiliin tilaansa, kemiluminessi tapahtuu alueella 600 nm – 3000 nm. Kemiluminenssissa vapautunut valon intensiteetti on verrannollinen typpimonoksidin (NO) määrään. Laitteessa oleva hapenpoistomuunnin muuttaa muodostetun typpidioksidin (NO_2) takaisin typpimonoksidiksi (NO), joka mitataan. [28, s. 6.]

Typpidioksidi (NO_2) -pitoisuus saadaan ratkaistua laskemalla erotus mitatun aerosolin typpioksidien (NO_X) pitoisuudesta sen kulkiessa sellaisenaan hapenpoistomuuntimen läpi ja mitatun aerosolin typpimonoksidi (NO), kun se ei kulje hapenpoistomuuntimen läpi. [28, s. 6.]

Laite pystyy mittaamaan typenoksideja (NO_X) välillä 0,5 ppb – 10000 ppb. Laitteen käyttämä aerosolivirtaus on 0,8 l/min. Laitteen vasteaika on 90 Hz. [28, s. 6.]



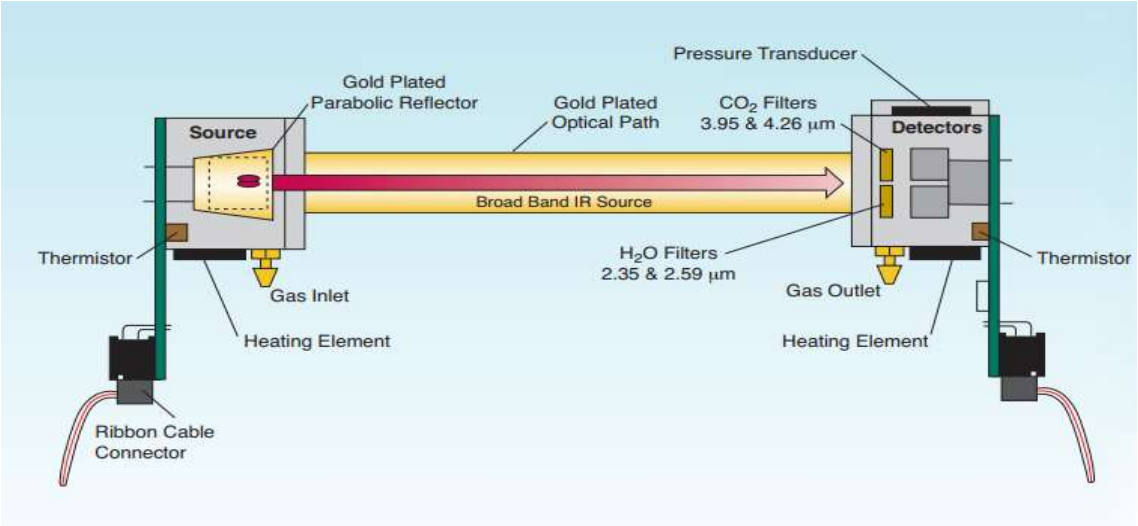
Kuva 14. NO_X-analysaattori [28, s. 6].

4.1.5 Hiilidioksidianalysaattori

Mittauksessa käytetty hiilidioksidianalysaattori (CO_2) LI-COR, 840 A CO2 / H20 analyzer on absoluuttinen ei dispersiiviseen infrapuna-absorbointiin (NDIR) perustuva yksi kanavainen ja kahteen aallonpituuteen perustuva kaasuanalysaattori. Hiilidioksidi (CO_2) - ja vesi (H_2O) -mittaus perustuu niiden ominaisuuteen absorboida infrapunasäteilyä itseensä, kun ne kulkevat mittauskammion läpi. Aineiden pitoisuusmittaus perustuu infrapunan säteilyyn erotukseen referenssi- ja näytesignaalin välillä. Hiilidioksidi (CO_2) näytekanava käyttää 4,26 µm:n suodinta, joka vastaa sen absorbointitaajuutta. Referenssikanava käyttää 3,95 µm:n suodinta, joka ei ole kaasun absorbointialueella. Veden (H_2O) näytekanava käyttää 2,595 µm:n suodinta, joka vastaa sen absorbointitaajuutta, ja referenssikanava käyttää 2,35 µm:n suodinta, joka ei ole sen absorbointialueella. [29, s. 65–70.]

Laite käyttää signaaliprosessointia määrittämään lämpötila- ja painekorjatun hiilidioksidipitoisuuden (CO_2) arvon näytekammiosta käyttämällä ratiometristä laskentaa. Hiilidioksidi (CO_2) -signaali tai kolmannen asteen polynomisignaali vedestä (H_2O) syötetään hyperbeliin, joka linearisoi signaalin mooleiksi ilmassa antaen tuloksen muodossa µmol CO_2 per mooli ilmaa (ppm) ja mmol H_2O per mooli ilmaa (mmol/mol). [29, s. 65–70.]

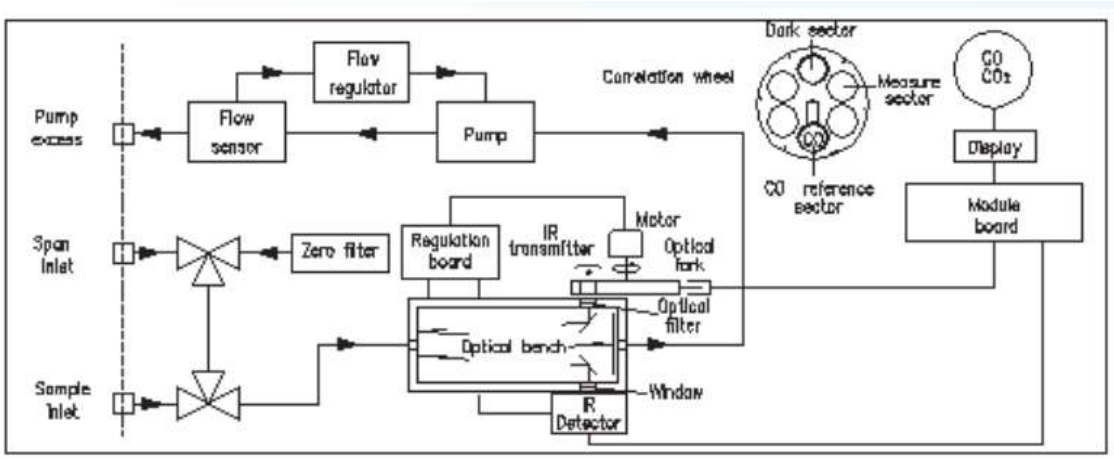
Laite pystyy mittamaan hiilidioksidipitoisuuksia (CO_2) 0–20 000 ppm ja vesipitoisuuksia (H_2O) 0–60 ppt:n tarkkuudella. Mittatarkkuus hiilidioksidilla on < 1 % lukemasta ja vedellä < 1,5 % lukemasta. Laite ottaa aerosolinäytettä 1 l/min. [29, s. 65–70.]



Kuva 15. CO_2-analysaattori [29, s. 67].

4.1.6 Hiilimonoksidianalysaattori

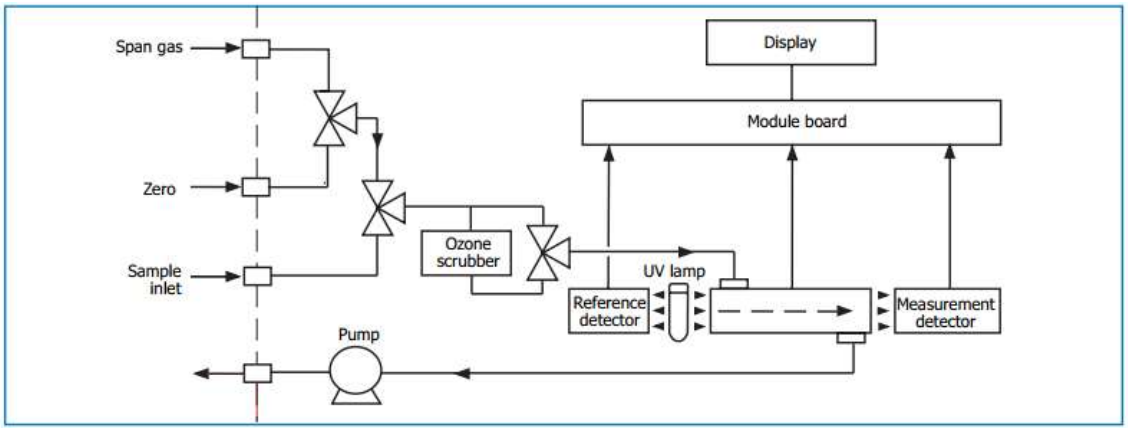
Mittauksessa käytetty hiilimonoksidianalysaattori (CO) Environnement S.A, CO12M analyzer on ei-dispersiiviseen infrapuna-absorbointiin (NDIR) perustuva mittalaite. Hiilimonoksidipitoisuus määritetään mittaamalla infrapunavalon määrää, jota näytekaasu absorboi itseensä sen kulkiessa monisolukorrelaatiorenkaan läpi, jossa toinen puoli toimii referenssinä (referenssisäde) ja toinen mittaavana (mittaussäde). Korrelaatiorengas pyörii ja infrapunavalo kulkeutuu joko referenssi- tai mittaussolun läpi, minkä jälkeen se kulkeutuu optisen suotimen läpi osuen itse ilmaisimeen. Jos näyte on sisältänyt hiilimonoksidia, referenssisädettä ei vähennetä siitä, koska se on jo vähennetty hiilimonoksidin referenssisolussa. Laite pystyy mittamaan hiilimonoksidipitoisuuksia 40 ppb – 200 ppm. Laiteen kohina on 20 ppb ja vasteaika pienimmillään 30 Hz. Laite ottaa aerosolinäytettä 1 l/min. [30]



Kuva 16. CO-analysaattori [30].

4.1.7 Otsonianalysaattori

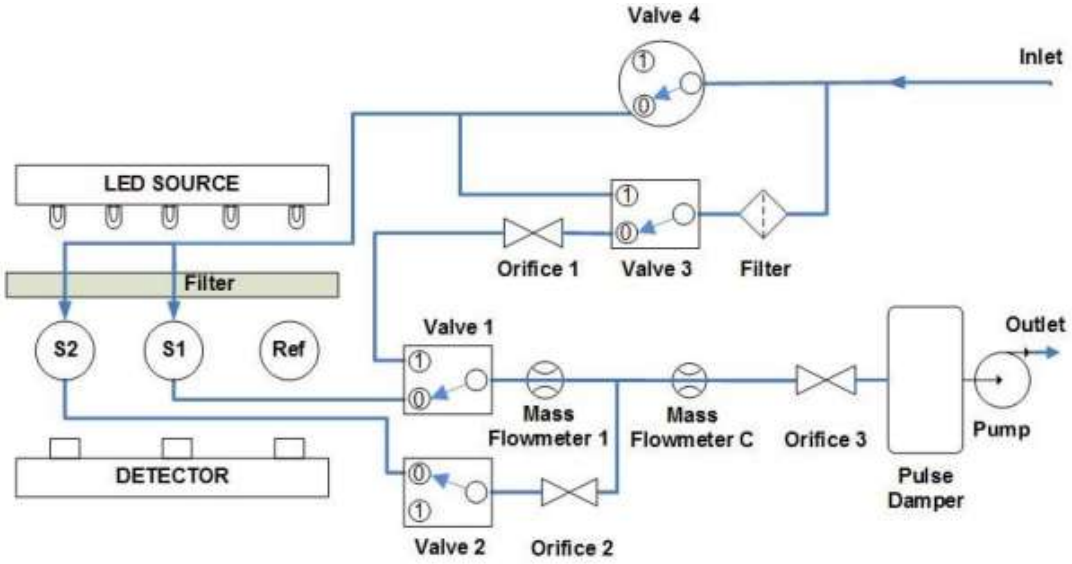
Mittauksessa käytetty otsonianalysaattori (O_3) Environnement S.A:n UV absorpation Ozone anylzer – Model O342M perustuu ultraviolettivalon mittaamiseen, jossa ultraviolettivalo absorboituu otsoni (O_3) -molekyyleihin. Otsoni (O_3) -pitoisuus määritetään vähentämällä ultraviolettivalolla absorboitu näyte otsonista (O_3) suodatettuun näytteeseen joka mitataan, kun näytekaasu on kulkeutunut katalysaattorin lävitse. Laite pystyy mittaamaan 0,4 ppb – 10 ppm:n otsonipitoisuuksia (O_3). Laitteen kohina on 0,2 ppb ja vasteaika pienimmillään 20 Hz. Laite ottaa aerosolinäytettä 1 l/min. [31]



Kuva 17. O_3-analysaattori [31].

4.1.8 Etalometri

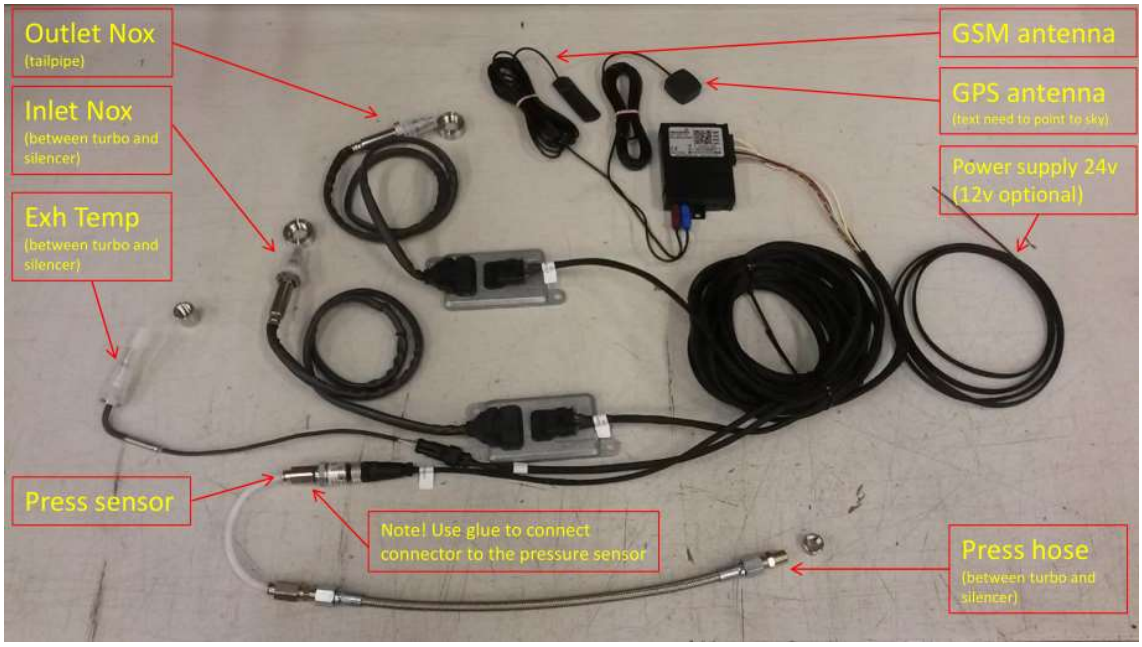
Mittauksessa käytetty etalometrin (Magee Scientific, Aethlometer Model 33) toiminta perustuu aerosolipartikkeleiden keräämiseen pisteiksi suodinpaperille. Laite analysoi aerosolin mittaamalla valon läpäisyä suodinpaperilta, johon on kerrytetty näytettä ja vertaamalla sitä suodinpaperilla olevaan kohtaa, jossa ei ole näytettä. Kyseinen analyysi tehdään eri valon aallonpituuksilla infrapunasta ultraviolettiin. Laite mittaa samanaikaisesti kahdesta näytepisteestä, joissa kummassakin on partikkelikertymä samasta tuloilmavirrasta. Nämä tulokset yhdistetään matemaattisesti epälineaarisuuden poistamiseksi sekä partikkelien valoabsorption kompensaation että mustan hiilen massakonsentraation saamiseksi. Laite mittaa valoa seuraavilla aallonpituuksilla: 370, 470, 520, 590, 660, 880 ja 960 nm. Laite ottaa näytettä 5 l/min. Laite antaa mustan hiilen tuloksen muodossa ng/m^3. [32, s. 21.]



Kuva 18. Etalometri [32, s. 30].

4.1.9 Proventia Procare Drive

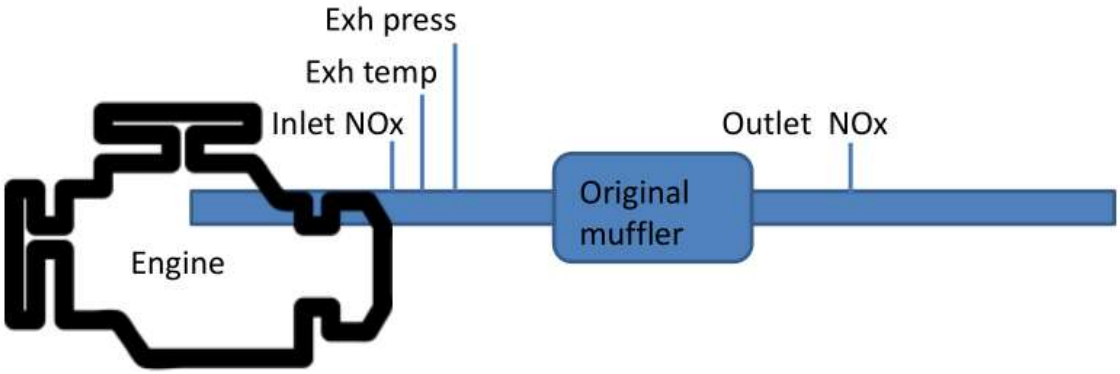
Mittauksissa mitattaviin ajoneuvoihin asennettiin Proventia Oy:n valmistamat Procare Drive -NO_X-monitorointijärjestelmä (kuva 19). Laitteisto asennettiin mitattaviin ajoneuvoihin, jotta typpioksidipäästö (NO_X) saatiin linkitettyä Nuuskijan mittamaan typpioksidipäästöön (NO_X). Järjestelmä mahdollistaa reaaliaikaisen typenoksidipäästöjen (NO_X) pakokaasun lämpötilan sekä vastapaineen seurannan. Laite kostuu GPS- ja GSM (3g) -antenneista, ohjainyksiköstä, yhdestä lämpötila-anturista ja kahdesta typpioksidianturista (NO_X) sekä vastapaineanturista. Laitteistoa voi myös laajentaa mittamaan PM10-hiukkaspäästöjä lisäantureilla. [33]



Kuva 19. Proventian NO_X-mittalaitteisto [33, s. 5].

Laitteisto käynnistyy, kun se saa syöttövirran ajoneuvolta, jolloin se kytkee jokaisen ohjainyksikköön kytketyn laitteen. Anturit lähettävät tietonsa CAN-väylää pitkin ohjainlaitteelle, joka purkaa tiedon ja lähettää sen Proventian palvelimelle käyttäen laitteessa olevaa 3g-yhteyttä. Tietojen lukuun tarvitaan lisenssi, jotta anturitietoja voidaan lukea järjestelmästä. Pakoputkistoon asennetut anturit mittaavat vasta, kun lämpötila-anturin ilmoittama lämpötila on yli 200 °C. Tällä varmistetaan antureiden toiminta, koska antureissa oleva keraaminen mittapää rikkoutuu helposti, anturin ollessa kuuma tai jos mittapäässä on kondensoitunutta vettä ennen anturin lämpenemistä oikeaan lämpötilaan. [33]

Lämpötila sekä toinen typpioksidianturi (NO_X) ja vastapaineanturi asennettiin ennen pakokaasun puhdistuslaitteita ja toinen typpioksidianturi (NO_X) niiden jälkeen (kuva 20). Laskemalla antureiden erotus voidaan todeta, kuinka toimiva pakokaasunpuhdistus laitteisto kussakin tapauksessa on. Proventian internetpohjainen sovellus päivittyy noin 1 minuutin intervalleissa. Riippuen mitattavan ajoneuvon tilasta sovellukseen tulee joko GPS:n mukana tulevat tiedot (paikka, nopeus, suunta, korkeus) tai GPS-tieto sekä anturitiedot, jotka ovat lämpötilariippuvaisia. Sovelluksesta saadaan tulostettua kaikki data 1 Hz:n resoluutiolla. Typpioksidianturit, jotka tulivat laitteiston mukana, olivat Continentalin valmistamia UniNox 5wk9 -sensoreita. Sensorit on kalibroitu niiden valmistajan toimesta, eikä laitteiston toimittaja tee erillistä kalibrointia sensoreille. [33]



Kuva 20. Proventian NO_X-laitteisto [33, s. 6].

5 Tulokset

Tässä osiossa käsitellään tutkimushankkeessa mitattuja tuloksia, jotka on jaoteltu päästökohtaisesti. Tulosten esityksen yhteydessä kerrotaan, miten tuloksia on käsitelty.

5.1 Maantieosuus

Esitettävät tulokset ovat yhdistelmä mittausjärjestelyissä määritetyistä osista siten kuin ne on ajettu (Turuntie ⟶ Kasvihuoneilmiö ⟶ Kasvihuoneilmiö ⟶ Salo ⟶ Paimio ⟶ Salo ⟶ Kasvihuoneilmiö ⟶ Turuntie). Kaikissa esitettävissä tuloksissa on käytetty ehtolausetta [Nopeus > 6,95 m/s (25,02 km/h)].

5.2 Toisto-osuus

Toisto-osuudella suoritetut mittaukset toistettiin useampaan kertaan. Ensimmäisenä mittauspäivänä mittauksia tehtiin viisi kappaletta, joista yksi oli taustan mittaus. Muina mittauspäivinä mittauksia tehtiin kahdeksan kappaletta, joista yksi oli taustan mittaus. Kaikissa esitettävissä tuloksissa on käytetty ehtolausetta [Nopeus > 6,95 m/s (25,02 km/h)].

Esitettävät tulokset on valittu ensin pilkkomalla jokainen toisto omakseen suunnan mukaan (itä, länsi). Tämän jälkeen on laskettu jokaiselle toistolle typpioksidin (NO_X) keskiarvo Nuuskijan mittaamasta typpioksidipitoisuudesta (NO_X). Ensimmäisen päivän osalta keskiarvoltaan korkein tulos on pudotettu pois. Muilta mittauspäiviltä keskiarvoiltaan kaksi korkeinta ja kaksi matalinta tulosta on pudotettu pois. Kaikkien muiden mitattujen päästöjen osalta käytetään näitä samoja toistoja. Tuloksien esityksen yhteydessä kerrotaan, miten tuloksia on käsitelty.

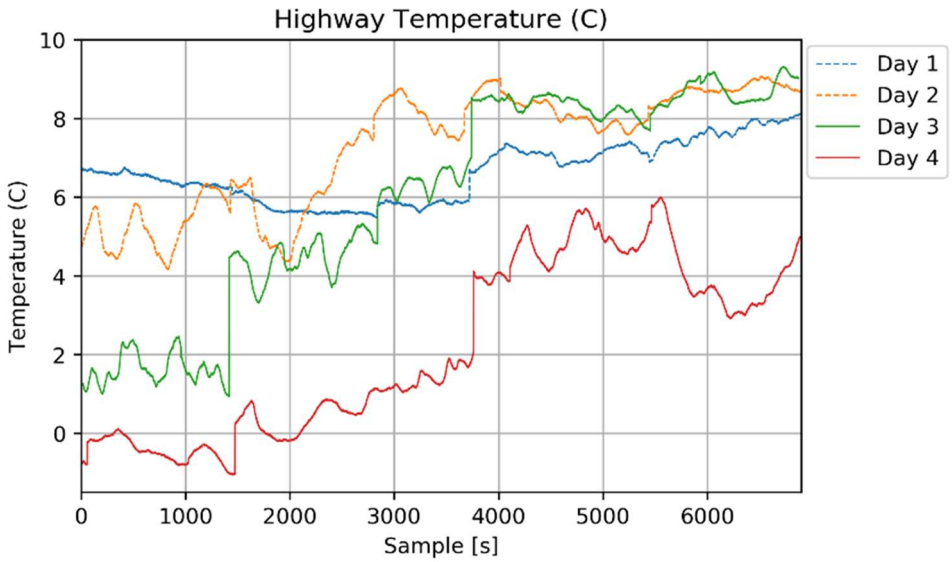
Liitteessä 4 on esitetty tulosten valintaan johtaneet keskiarvot.

5.3 Lämpötila ja ilmankosteus

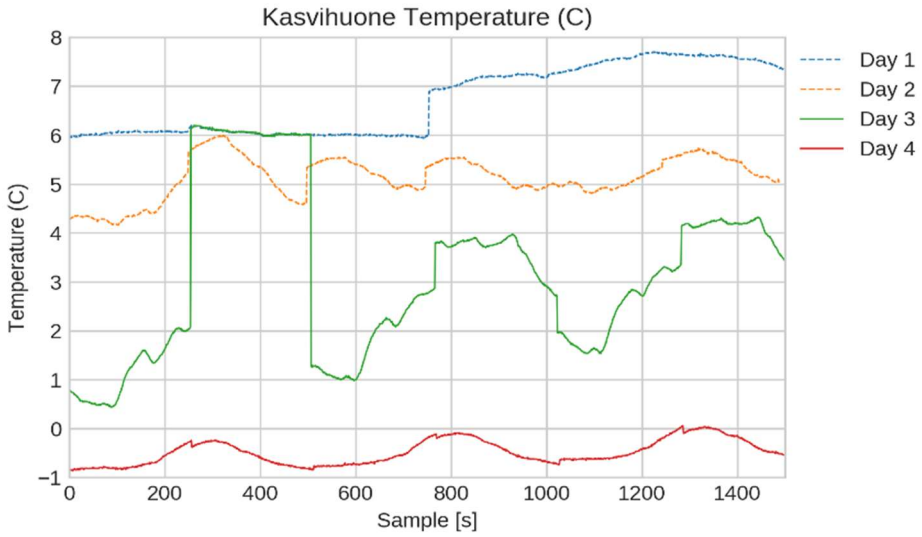
Vallitsevalla lämpötilalla ja ilmankosteudella on vaikutus pakokaasupäästöjen muodostumiseen. Asiaa ovat monet eri tahot tutkineet ja todenneet, että näillä muuttujilla on merkitystä pakokaasun komponenttien muodostumisessa. [32] Tässä työssä esitetyissä tuloksissa ei ole otettu huomioon näitä muuttujia.

Alla esitetyt kuvaajat 1–3 näyttävät maantie, ja toisto-osuuden lämpötilan ja ilmankosteuden vaihtelua. Lämpötila sekä ilmankosteus on mitattu Nuuskijassa olevalla sääasemalla.

Kuvaajista 1 ja 2 huomataan, että lämpötila on noussut sekä ilmankosteus laskenut päivän mittaan, poikkeuksena päivä 4, jolloin ilmankosteus on noussut päivän lopuksi.

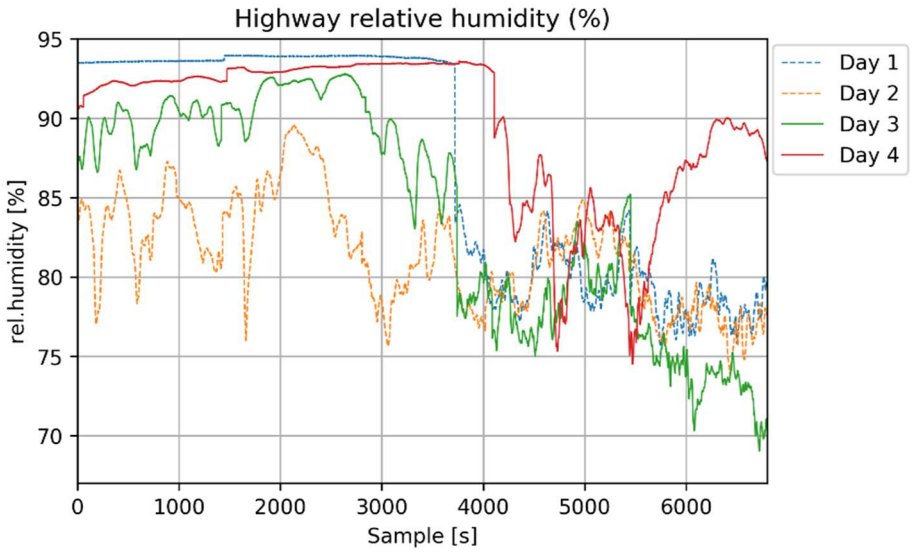


Kuvaaja 1. Lämpötila, maantieosuus.

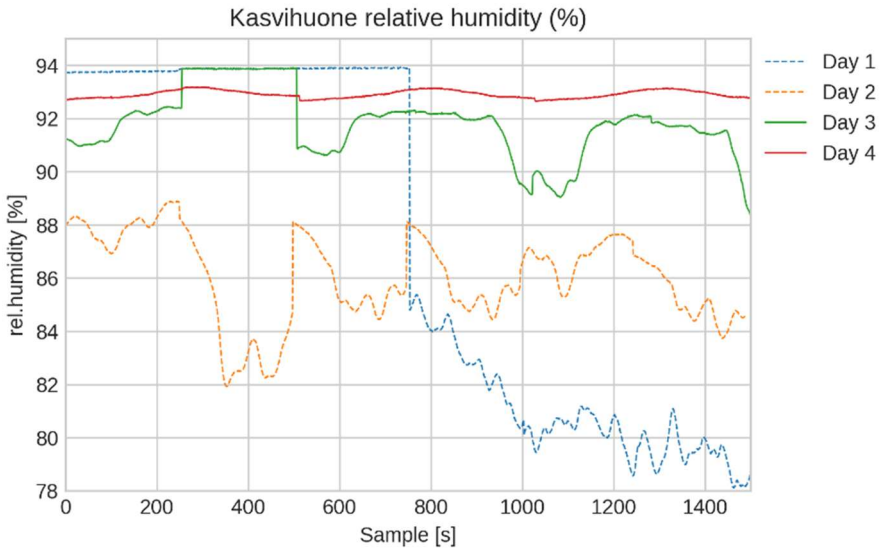


Kuvaaja 2. Lämpötila, toisto-osuus.

Kuvaajista 3 ja 4 nähdään, että lämpötila ja ilmankosteus on toisto-osuudella pysynyt päiväkohtaisesti melko samana, poikkeuksena päivä 1, jolloin ilmankosteus laskenut loppua kohti. Toisto-osuudella lämpötilaa ja ilmankosteutta verrattaessa voidaan todeta, että päivät 1 ja 2 ovat olleet lämpimämpiä ja kuivempia kuin päivät 3 ja 4.



Kuvaaja 3. Suhteellinen kosteus, maantieosuus.



Kuvaaja 4. Suhteellinen kosteus, toisto-osuus.

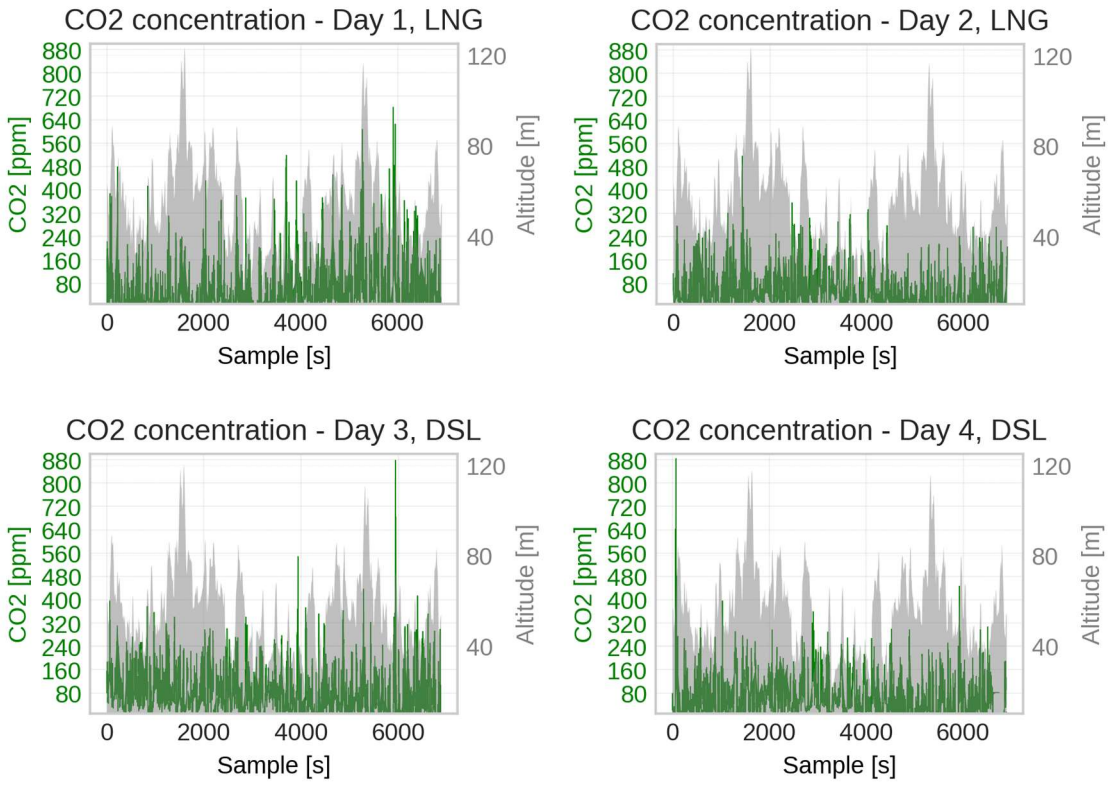
5.4 Hiilidioksidipäästöt

Tässä osuudessa käsitellään mitattuja hiilidioksidipäästöjä maantie- ja toisto-osuudella.

5.4.1 Maantieosuus

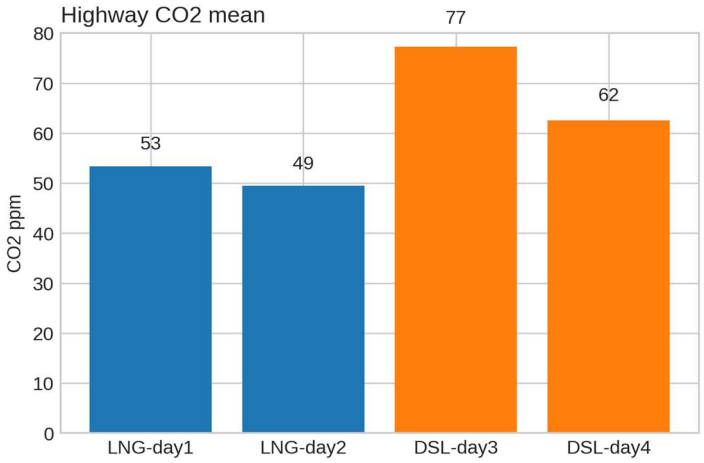
Kuvaaja 5 esittää maantieosuudella mitattua korjattua hiilidioksidipitoisuutta (CO_2). Kuvaajan arvot on saatu vähentämällä mitatusta pitoisuudesta taustapitoisuudenkeskiarvo. Kuvaajaan on myös lisätty reitin korkeusero.

Tarkasteltaessa esimerkiksi ajanhetkeä n. 6000 sekuntia, huomataan jokaisessa päivässä piikki tieprofiilin noustessa. Kyseisessä kohtaa reittiä oli tietyö, ja kaikkina muina päivinä paitsi päivänä nro 2 siinä jouduttiin reilusti hidastamaan nopeutta. Kuten kuvaajista voidaan päätellä, hiilidioksidipäästö (CO_2) on ajoneuvosta riippumatta hyvin samankaltainen.



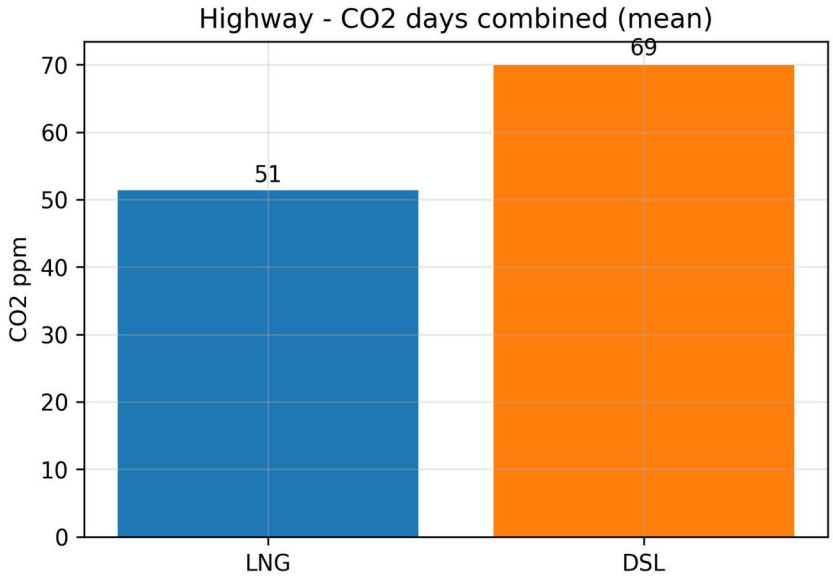
Kuvaaja 5. CO_2-päästöt maantieosuudella.

Kuvaajassa 6 on esitetty päiväkohtainen hiilidioksidipäästön (CO_2) keskiarvo koko mittausajanjaksolta. Kuvaajasta 6 huomataan, että päiväkohtaista varianssia on jonkin verran. Kuvaajasta nähdään, että maakaasua polttoaineena käyttävä rekka päästää vähemmän hiilidioksidia (CO_2) kuin vastaava dieseliä polttoaineena käyttävä rekka.



Kuvaaja 6. CO_2:n keskiarvo, maantieosuus.

Kuvaajassa 7 on esitetty ajoneuvokohtainen hiilidioksidipäästön (CO_2) keskiarvo koko mittausajanjaksolta. Kuvaajasta huomataan, että hiilidioksidipäästö (CO_2) on hieman korkeampi dieselajoneuvolla.

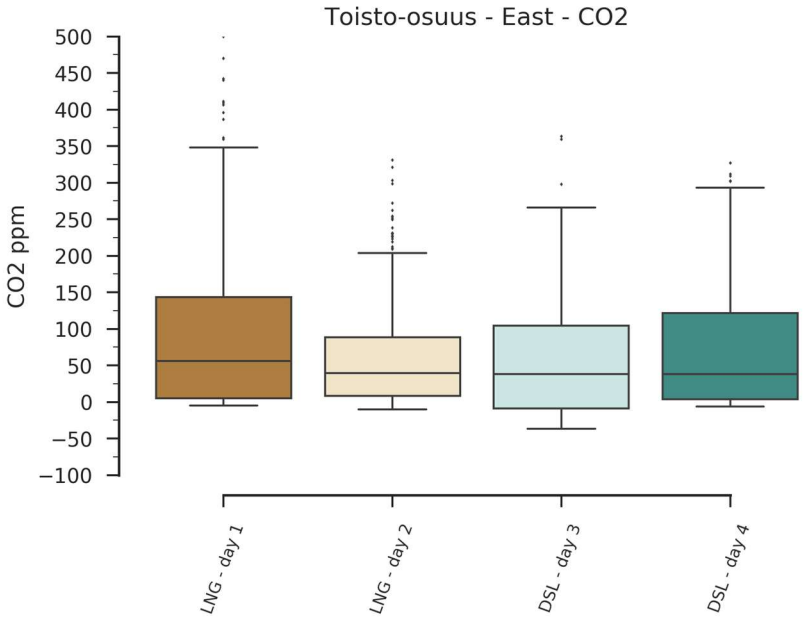


Kuvaaja 7. CO_2:n keskiarvo maantieosuudella polttoainekohtaisesti.

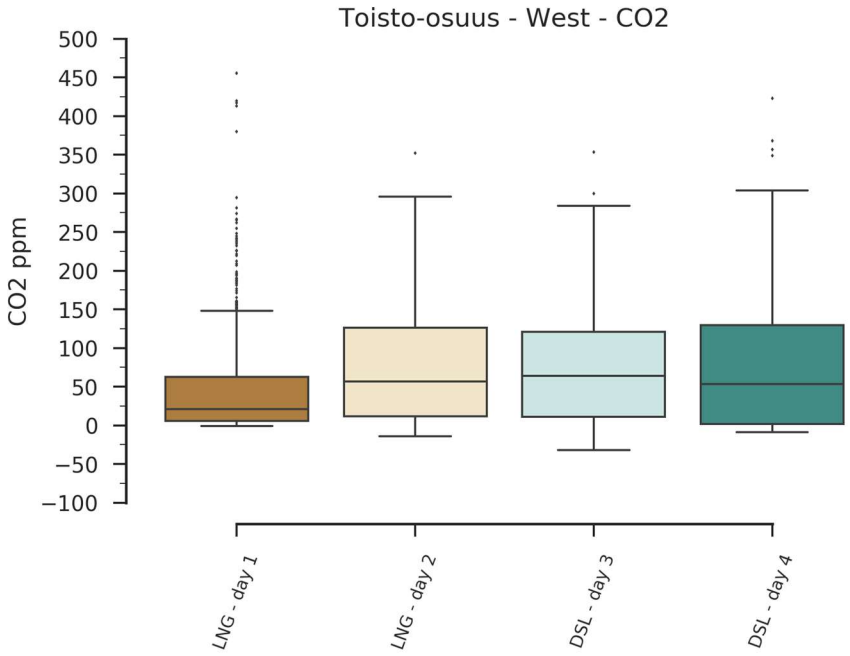
5.4.2 Toisto-osuus

Esitettävät tulokset ovat määritetty toistojen typpioksidipitoisuuksien (NO_X) perusteella; tarkempi määrittely on esitetty luvussa 5.2.

Kuvaajissa 8 ja 9 on esitetty korjattu hiilidioksidipitoisuus (mitattu CO_2 – tausta CO_2) päiväkohtaisesti idän ja lännen suuntiin. Kuvaajista nähdään, että päiväkohtaista hajontaa on jonkin verran. Huomioon otettavaa on, että olosuhteet ovat päiväkohtaisesti erilaiset ja toistoissa on eroja johtuen muusta liikenteestä.

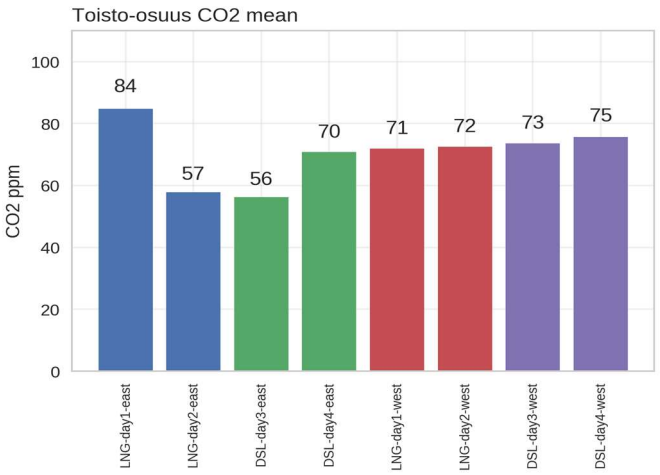


Kuvaaja 8. CO_2-pitoisuus toisto-osuudella idän suuntaan.



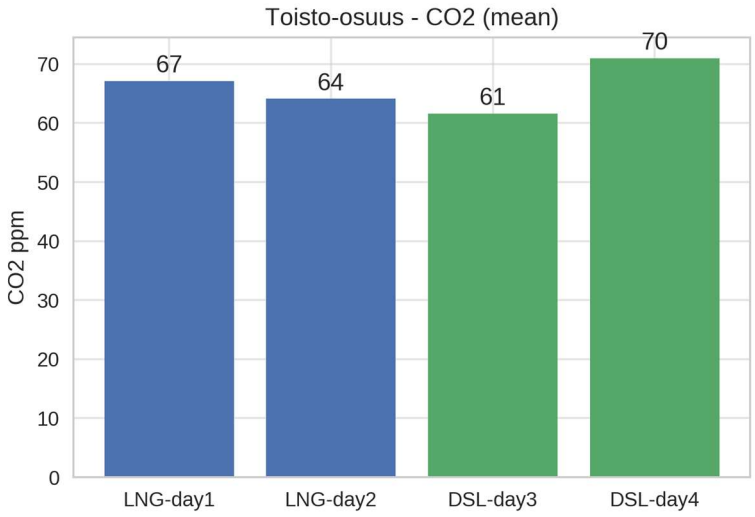
Kuvaaja 9. CO_2-pitoisuus toisto-osuudella lännen suuntaan.

Kuvaajassa 10, on esitetty päivä- ja suuntakohtaiset korjatut hiilidioksidipäästöjen (CO_2) keskiarvot. Jokaisessa pylväässä on kolmen toiston pitoisuudet yhdistettynä ja siitä otettu keskiarvo.



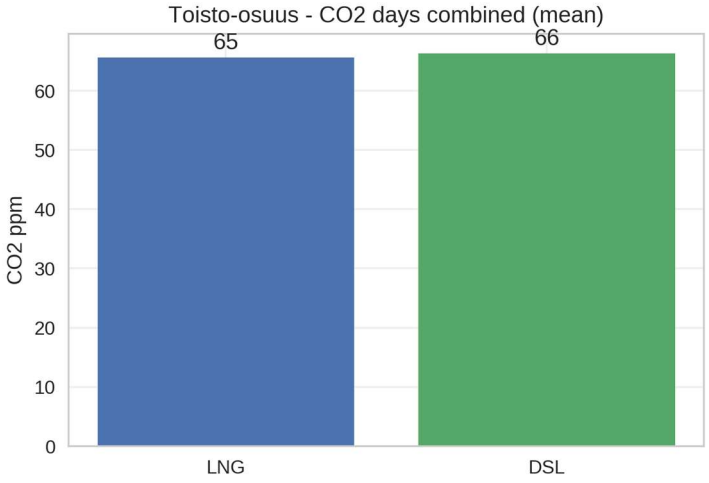
Kuvaaja 10. CO_2:n suuntakohtainen keskiarvo toisto-osuudella.

Kuvaajassa 11 on esitetty samaiset arvot päiväkohtaisesti. Kuvaajasta voidaan huomata, että hiilidioksidipitoisuuksissa (CO_2) ei ole merkittävää eroa ajoneuvojen kesken.



Kuvaaja 11. CO_2:n päiväkohtaiset keskiarvot toisto-osuudella.

Kuvaajassa 12 on esitetty ajoneuvokohtaiset keskiarvot koko mittausten ajata. Voidaan todeta, että hiilidioksidipäästöt (CO_2) eivät riipu polttoaineesta, kun tarkastellaan koko mittausajan jaksoa.



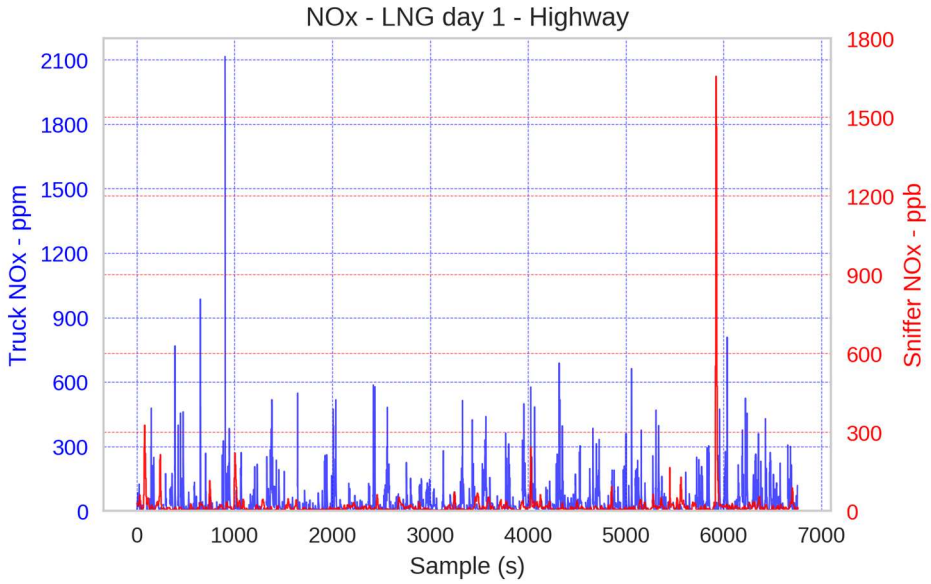
Kuvaaja 12. CO_2:n polttoainekohtainen keskiarvo toisto-osuudella.

5.5 Typpioksidipäästöt

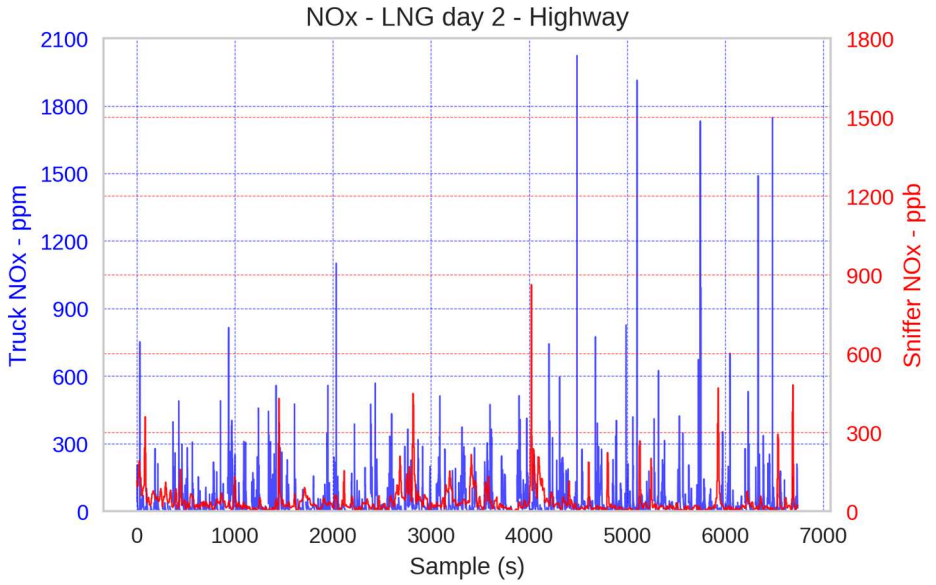
Tässä osiossa käsitellään mitattuja typenoksidipäästöjä maantie- ja toisto-osuudella.

5.5.1 Maantieosuus

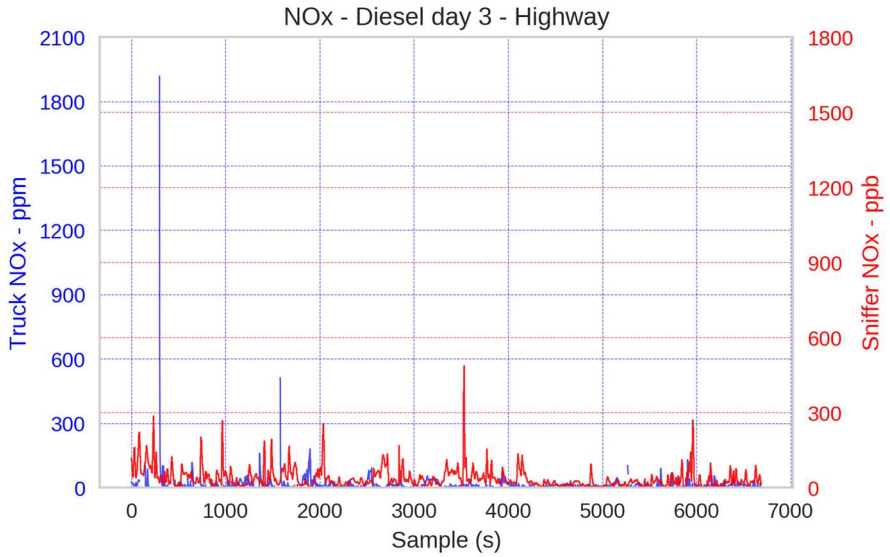
Alla olevissa kuvaajissa 13,14,15 ja 16 on esitetty Nuuskijalla mitattu typpioksidipitoisuus (NO_X) sekä Proventian laitteiston mittaama pitoisuus kunkin rekan ulostulosta. Aika-akselille on tehty korjaus, jotta Nuuskijalla mitattu pitoisuus on saatu korjattua siten, että Nuuskijan mittalinjan sisään tulo vastaisi mitattavan ajoneuvon pakoputken suuta. Aika-akselit on kummassakin tapauksessa aluksi määritetty samaan GPS-pisteeseen, minkä jälkeen on otettu huomioon mittausajoneuvon etäisyys mitattavan ajoneuvon pakoputkesta. Tämän jälkeen on määritetty siirtymä Nuuskijan mittalinjan alusta mittalaitteelle ja siihen on vielä lisätty typpioksidianalysaattorin viive.



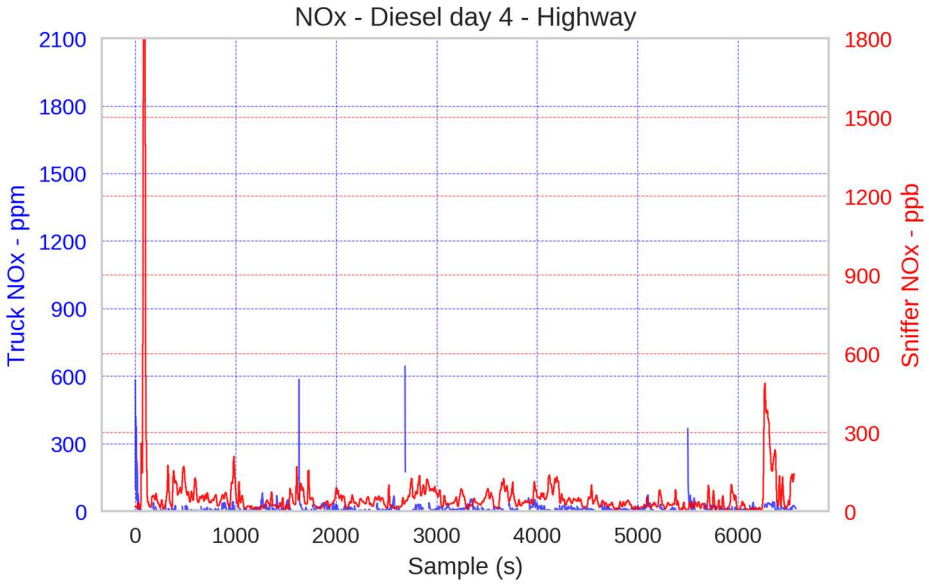
Kuvaaja 13. NO_X-pitoisuus, päivä 1, maantieosuus.



Kuvaaja 14. NO_X-pitoisuus, päivä 2, maantieosuus.



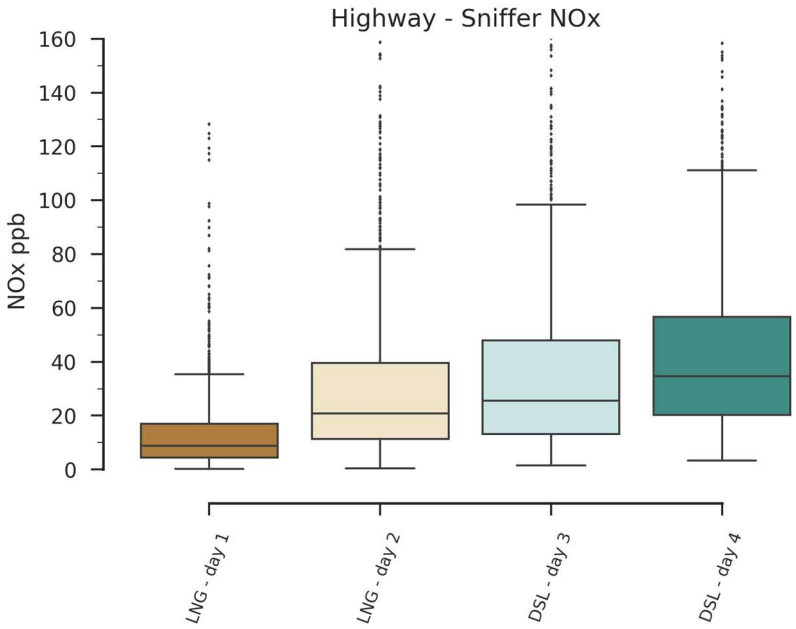
Kuvaaja 15. NO_X-pitoisuus, päivä 3, maantieosuus.



Kuvaaja 16. NO_X-pitoisuus, päivä 4, maantieosuus.

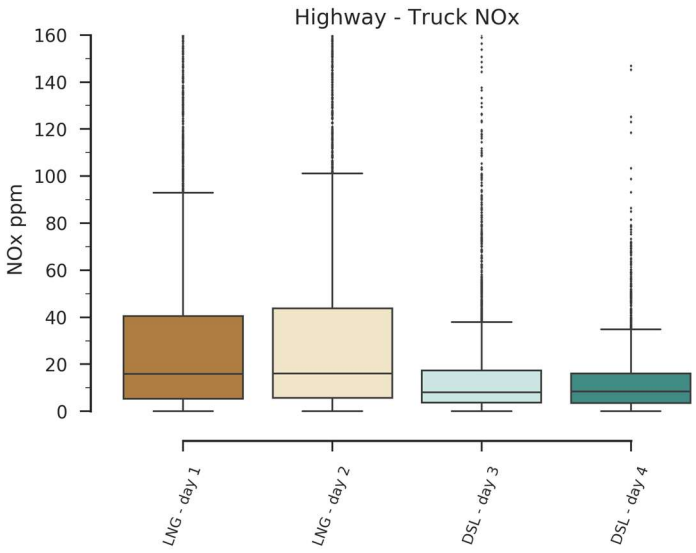
Kuvaaja 17 esittää Nuuskijalla mitattua typpioksidipäästön (NO_X) hajontaa päiväkohtaisesti. Kuvista voidaan nähdä kunkin päivän pitoisuuden mediaani, minimi sekä maksimi. Kuvaaja 18 esittää samat tiedot kuin kuvaaja 17; arvot on otettu vain rekoissa olleista mittalaitteista.

Kuvaajasta 17 voidaan tulkita Nuuskijan laitteiston osalta, että tuloksissa on päiväkohtaisesti hajontaan. Pitoisuudet ovat lineaarisesti nousevia, joka luultavasti johtuu mittauspäivien erilaisista olosuhteista, vedettävistä kuormista sekä muusta liikenteestä. Kuvaajasta nähdään kuitenkin maakaasurekan olevan pienempi päästöisempi.



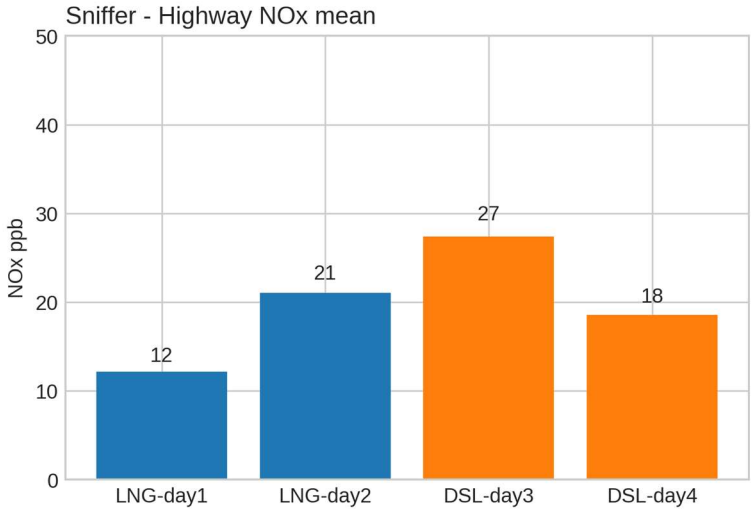
Kuvaaja 17. NO_X-pitoisuus Nuuskijan laitteelta.

Kuvaajasta 18 voidaan tulkita rekan, että maakaasun rekan mittauspäivät ovat olleet hyvin samankaltaisia. Päivänä 2 vaihteluväli on suurempi, mikä voidaan myös todeta päiväkohtaisista viivadiagrammeista. Hajontaan voi vaikuttaa kuljettajan ajotyyli tai vaihtunut kuorma. Dieselrekan mittauspäivät ovat samantyyppisiä, eikä mittauspäivinä kuorma tai mikään muukaan muuttunut. Kuvaajasta 18 nähdään myös, että päästöjen vaihteluväli on suurempi maakaasurekan osalta.

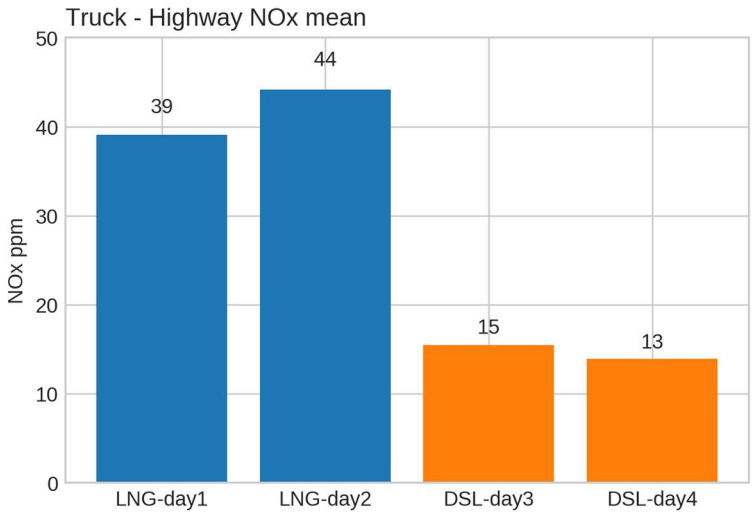


Kuvaaja 18. NO_X-pitoisuus rekkojen laitteista.

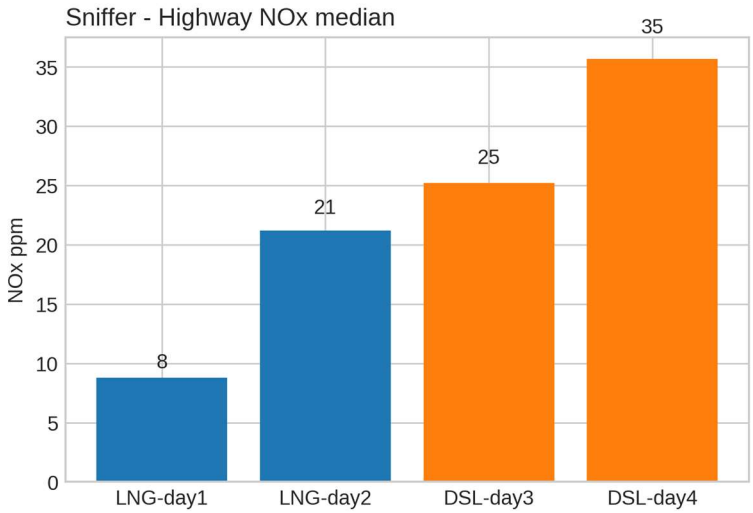
Kuvaajat 19 ja 20 esittävät typpioksidipäästöjen (NO_X) keskiarvoa ja kuvaajat ja 21 ja 22 mediaania koko reitiltä päiväkohtaisesti Nuuskijan laiteelta sekä rekoissa olleista antureista.



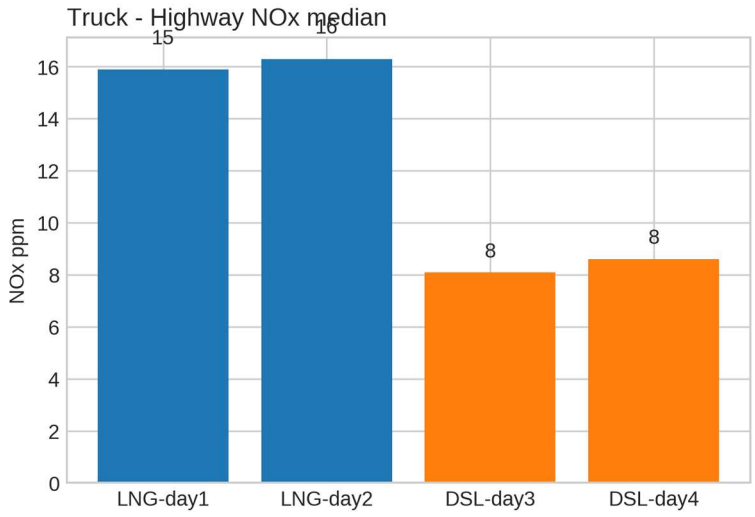
Kuvaaja 19. NO_X-keskiarvo Nuuskijasta maantieosuudella.



Kuvaaja 20. NO_X-keskiarvo rekasta maantieosuudella.

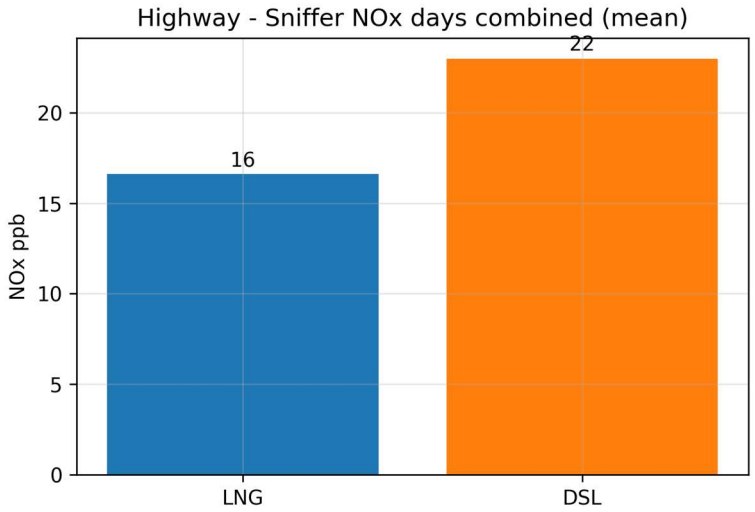


Kuvaaja 21. NO_X-mediaani Nuuskijasta maantieosuudella.

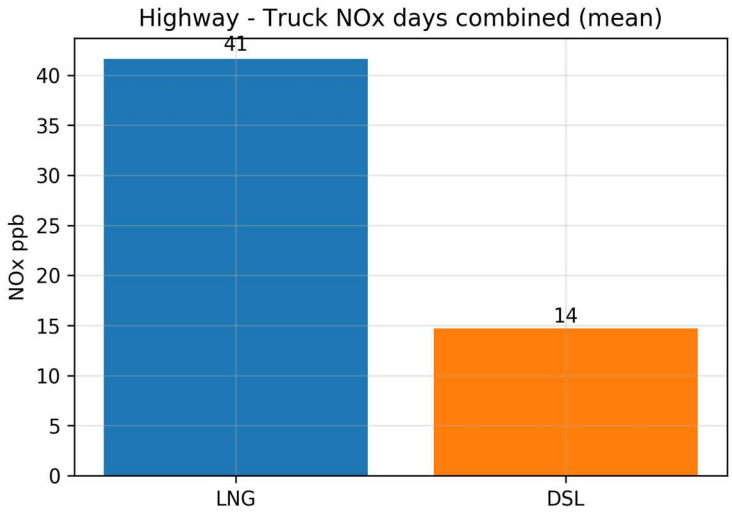


Kuvaaja 22. NO_X-mediaani rekasta maantieosuudella.

Kuvaajissa 23 ja 24 on esitetty ajoneuvokohtainen päästö koko mittausajanjaksolta. Kuvaajista voidaan nähdä, että tulokset ovat hyvin ristiriitaiset. Proventian NO_X-anturit antavat kuvan, että LNG-rekka päästäisi suhteessa enemmän typpioksideja kuin dieselrekka. Toisaalta Nuuskijan mittaamat typenoksidipäästöt ovat päinvastaiset.



Kuvaaja 23. NO_X:n polttoainekohtainen keskiarvo Nuuskijasta maantieosuudella.

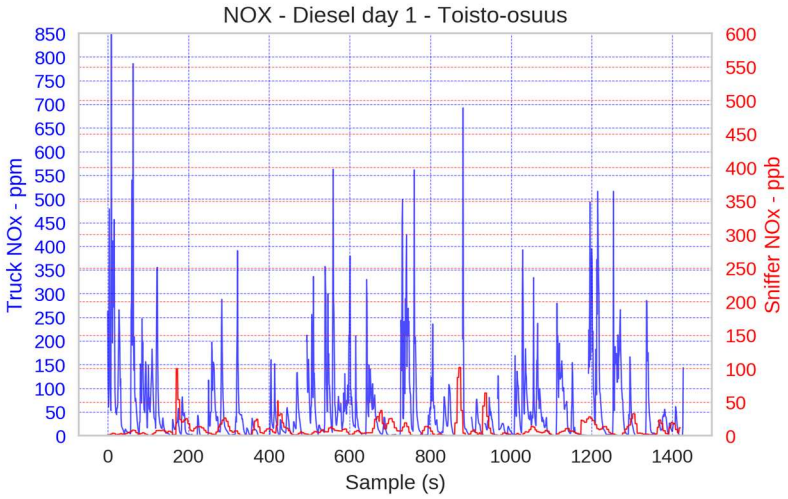


Kuvaaja 24. NO_X:n polttoainekohtainen keskiarvo rekasta maantieosuudella.

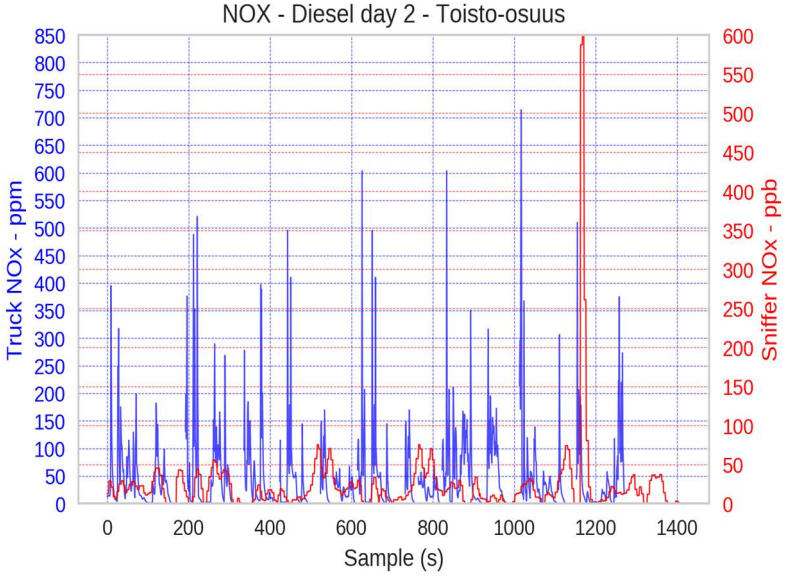
5.5.2 Toisto-osuus

Liitteessä 6 on esitetty mitattu typpioksidipitoisuus (NO_X) suunta- ja päiväkohtaisesti Nuuskijalla sekä Proventian laitteistolla mitattuna.

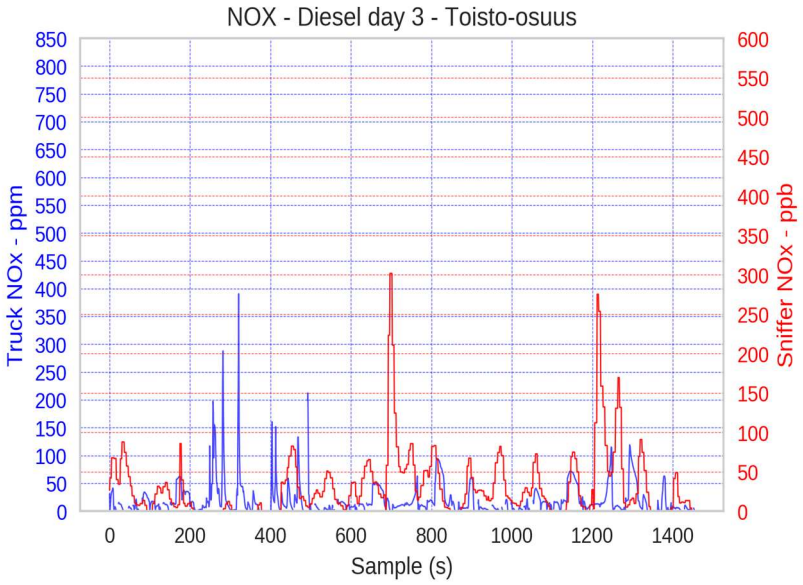
Kuvaajissa 25, 26, 27 ja 28 on esitetty jahdatun ajoneuvon mitattu päästö verrattuna jahtaavan ajoneuvon mitattuun päästöön. Aikasarjassa on otettu huomioon päästön siirtymä lähteestä ahdattavasta jahtaavaan ajoneuvoon. Nuuskijan osalta typpioksidipäästöstä (NO_X) on vähennetty mitattu kyseisen komponentin taustapitoisuus.



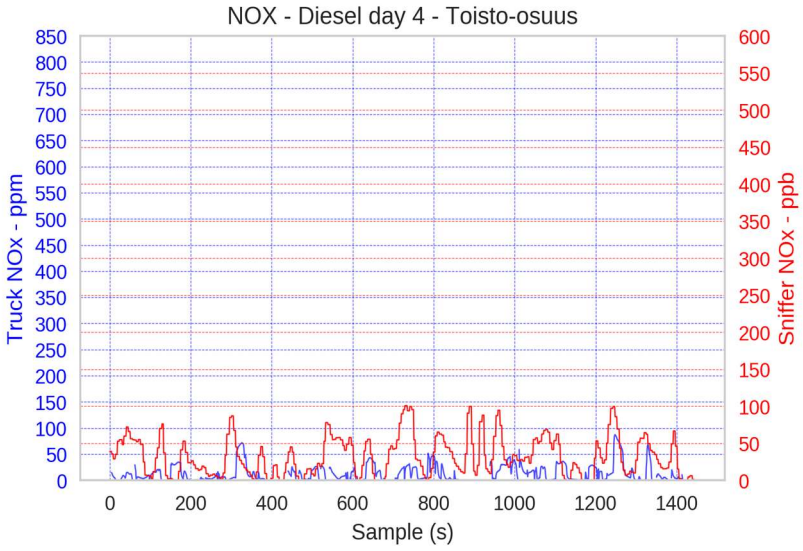
Kuvaaja 25. NO_X-pitoisuus toisto-osuudella, päivä 1.



Kuvaaja 26. NO_X-pitoisuus toisto-osuudella, päivä 2.

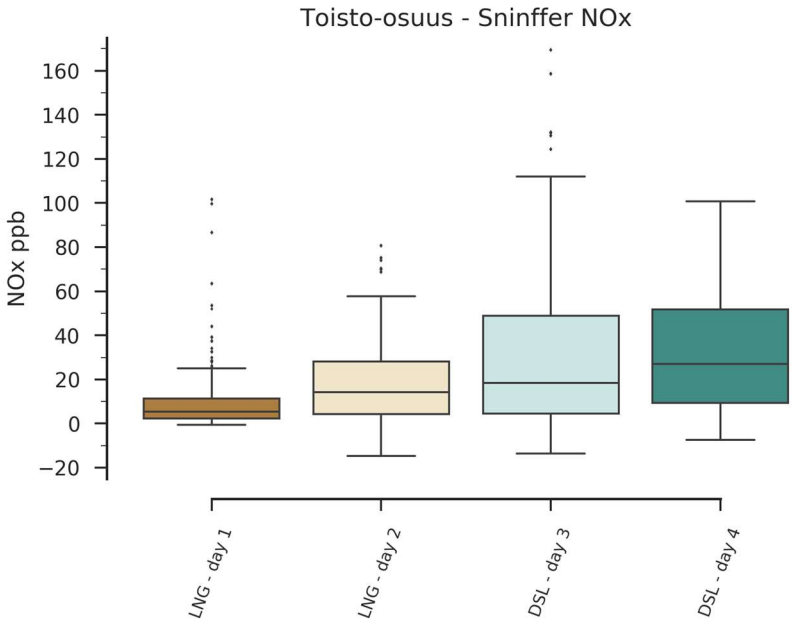


Kuvaaja 27. NO_X-pitoisuus toisto-osuudella, päivä 3.



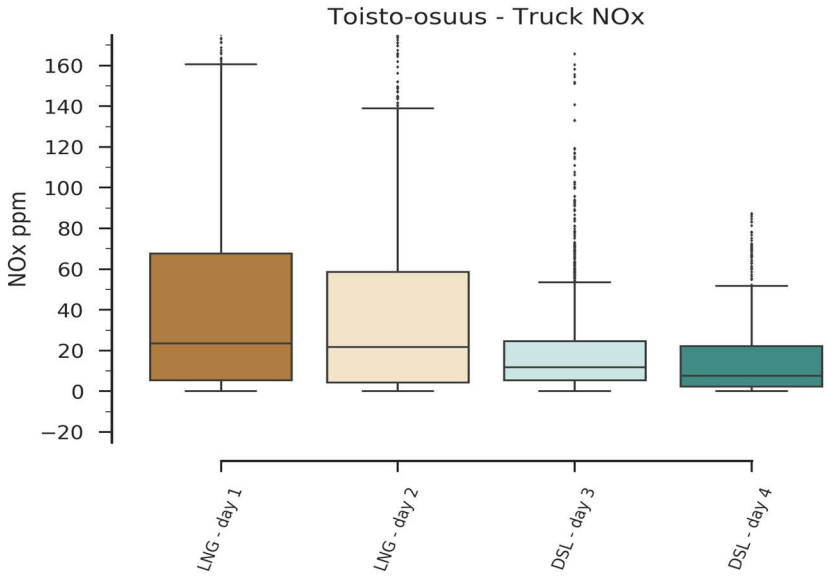
Kuvaaja 28. NO_X-pitoisuus toisto-osuudella, päivä 4.

Kuvaaja 29 esittää Nuuskijalla mitattua typpioksidipitoisuuden (NO_X) hajontaa päiväkohtaisesti. Kuvaajista voidaan nähdä kunkin päivän pitoisuuden mediaani, minimi sekä maksimi. Kuvaajista nähdään hyvin päiväkohtainen poikkeuma.



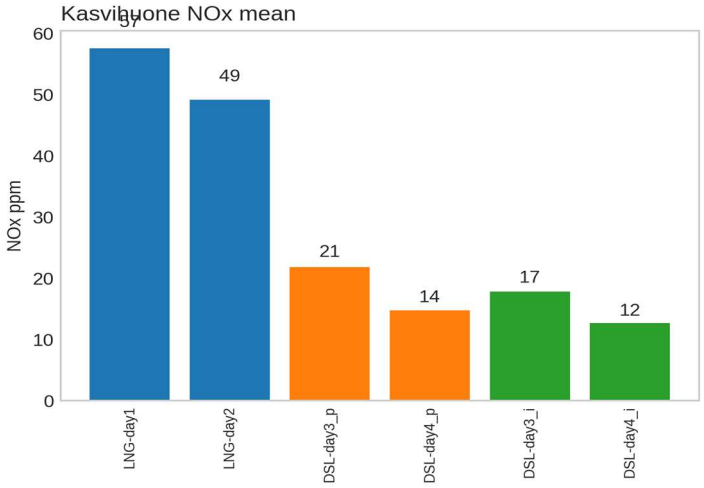
Kuvaaja 29. NO_X-pitoisuus toisto-osuus Nuuskijasta.

Kuvaaja 30 esittää rekoista mitattua typpioksidi (NO_X) -pitoisuuden hajontaa päiväkohtaisesti. Kuvaajista voidaan nähdä kunkin päivän pitoisuuden mediaani, minimi sekä maksimi. Kuvaajista nähdään, että päiväkohtaista varianssia on jonkin verran. Ajoneuvo kohtaiset pitoisuudet ovat hyvin samanlaisia.



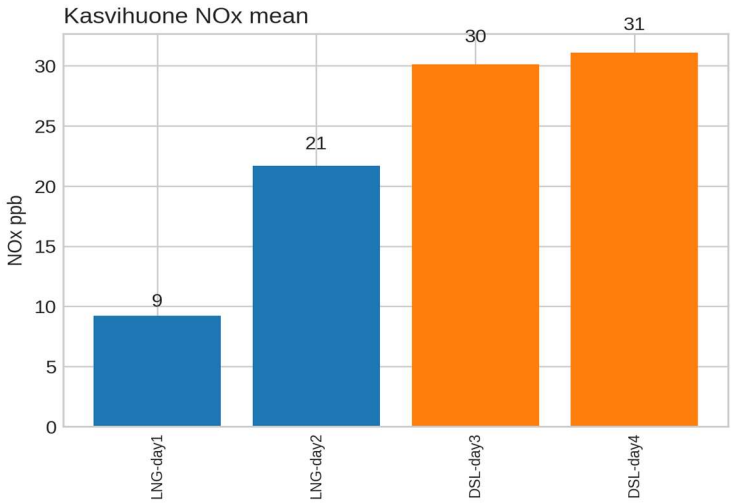
Kuvaaja 30. NO_X-pitoisuus toisto-osuus rekanlaitteistosta.

Kuvaajassa 31 esitetään päiväkohtainen keskiarvo typpioksidipäästölle rekanlaitteistosta. Dieselin osalta tuloksissa on esitetty myös ajoneuvon omasta järjestelmästä saatu typpioksidipitoisuus (NO_X) (vihreä). Kuten kuvaajasta voidaan nähdä, trendi on sama kuin maantieosuudella, jossa typpioksidipitoisuus (NO_X) kyseisillä laitteilla näyttää, että maakaasurekka olisi epäpuhtaampi.



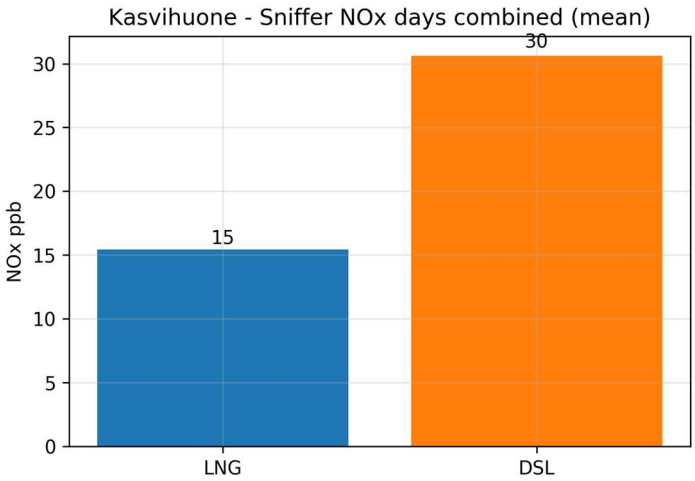
Kuvaaja 31. NO_X-keskiarvo toisto-osuudella rekkojen laitteistosta.

Kuvaajassa 32 on esitetty Nuuskijan mittaama korjattu typpioksidipäästön keskiarvo toisto-osuudella päiväkohtaisesti. Kuten maantieosuudella nähdään, että toisto-osuudella pitoisuus maakaasun osalta on pienempi kuin dieselin. Mittaustulos on ristiriidassa Proventian laitteiston kanssa (kuvaaja 33), mikä todettiin jo maantieosuudella esitetyissä tuloksissa.

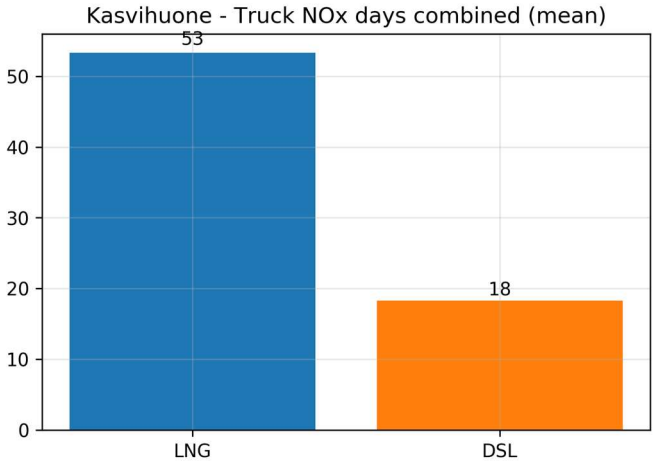


Kuvaaja 32. NO_X-keskiarvo toisto-osuudella Nuuskijasta.

Kuvaajat 33 ja 34 esittävät ajoneuvokohtaiset typpioksidipäästöt (NO_X) koko mittausajanjaksolta. Kuten kuvaajista näemme, tulokset ovat ristiriidassa toistensa kanssa.



Kuvaaja 33. NO_X:n toisto-osuuden polttoainekohtainen keskiarvo Nuuskijasta.



Kuvaaja 34. NO_X:n toisto-osuuden polttoainekohtainen keskiarvo rekoista.

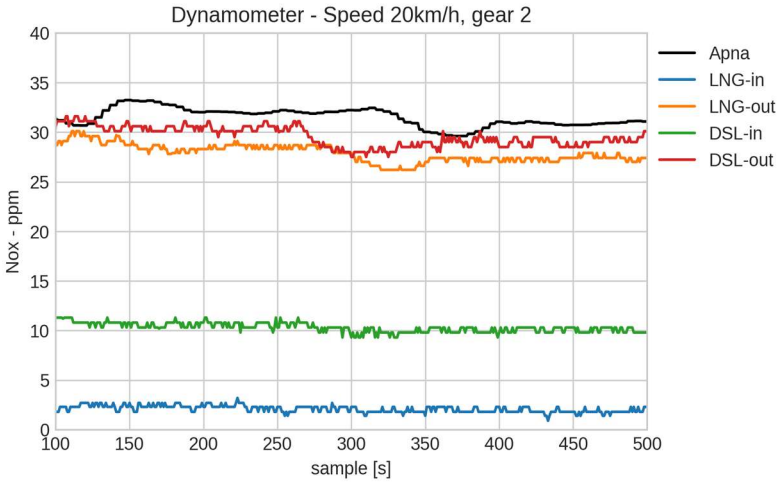
5.6 Laitevertailu

Laite vertailu suoritettiin Metropolia Ammattikorkeakoulun laboratoriotiloissa tammikuussa 2018. Mittaus suoritettiin yhtenä päivänä. Mittauksessa käytettiin samoja typpioksidi (NO_X) -laitteistoja kuin jahtausmittauksessa. Mittauksessa käytettiin Euro 5 -päästöluokan täyttävää dieselmoottorista ajoneuvoa (Toyota Corolla 1.6).

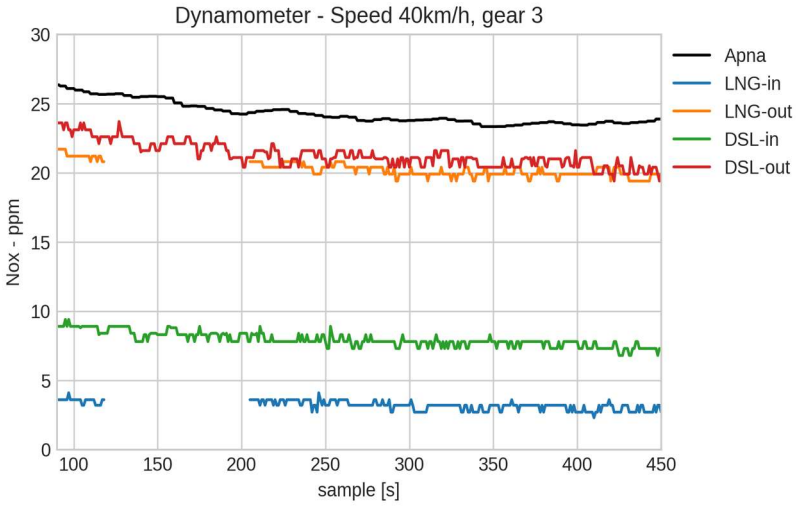
Mittalaiteet asennettiin pakoputken ulostuloon rinnakkain. Proventian laitteet mittasivat raakaa pakokaasupäästöä ja Nuuskijan laite mittasi päästöä kahden ejektorin takaa, joiden yhteinen laimennussuhde oli 1:64.

Ajoneuvoa ajettiin kolmea eri nopeutta 20, 40 ja 70 km/h. Kutakin nopeutta ajettiin n. 5 minuutin ajan. Mittauksen tarkoituksena oli selvittää, kuinka hyvin Proventian laitteet vertautuvat referenssilaitteeseen, joka tässä tapauksessa oli Nuuskijassa oleva Horiba Apna360CE.

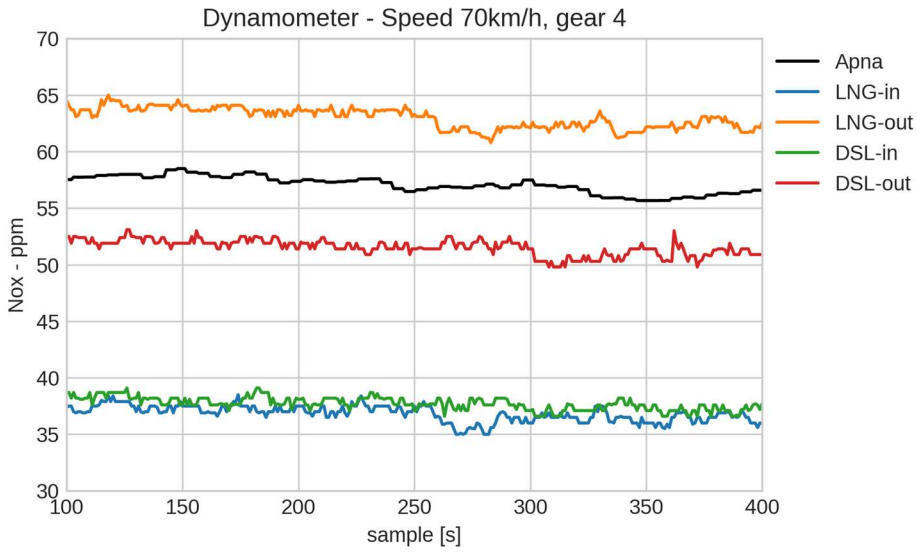
Kuvaajissa 35, 36 ja 37 on esitetty kunkin anturin mittaamat pitoisuudet eri nopeuksilla.



Kuvaaja 35. NO_X-pitoisuus 20 km/h.

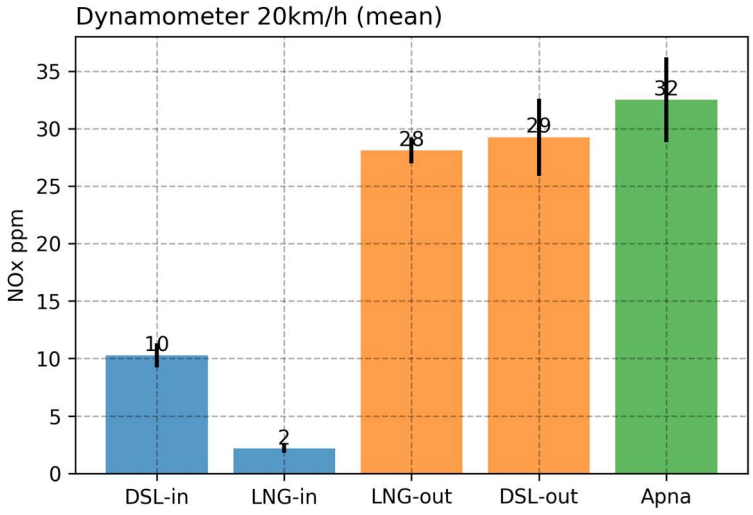


Kuvaaja 36. NO_X-pitoisuus 40 km/h.

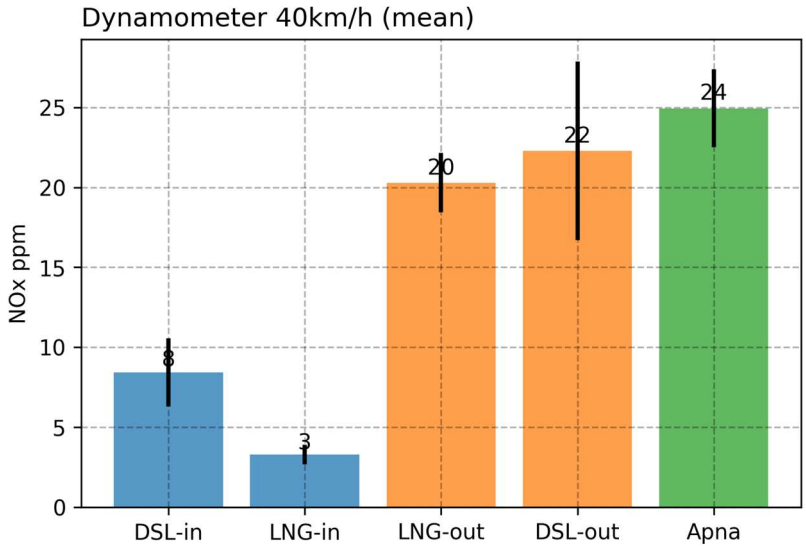


Kuvaaja 37. NO_X-pitoisuus 70 km/h.

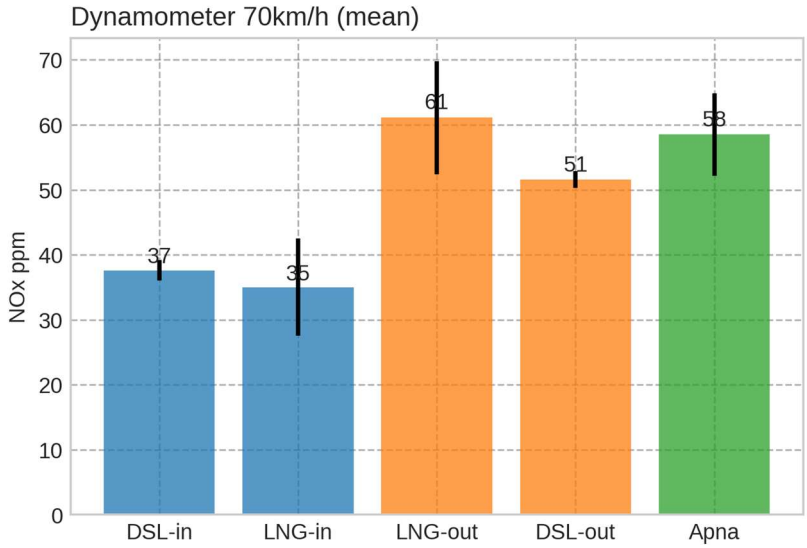
Kuvaajissa 38, 39 ja 40 on esitetty keskiarvot ja keskihajonnat typpioksidipäästöistä eri nopeuksilla.



Kuvaaja 38. NO_X-keskiarvo 20 km/h.



Kuvaaja 39. NO_X-keskiarvo 40 km/h.



Kuvaaja 40. NO_X-keskiarvo 70 km/h.

Taulukossa 8 näemme antureiden suhteen referenssilaitteeseen. Taulukosta huomataan, että prosentuaalinen ero on suuri niillä antureilla, jotka ovat olleet ajoneuvossa ennen puhdistuslaitteita.

Taulukko 8. NO_X-antureiden suhde referenssilaitteeseen.

NOx keskiarvo (ppm)					
Nopeus km/h	**NoxInLNG**	**NoxInDSL**	**NoxOutLNG**	**NoxOutDSL**	**Apna**
70	35,02	37,62	61,11	51,58	58,52
40	3,29	8,42	20,29	22,28	24,94
20	2,16	10,27	28,1	29,25	32,5
Suhde (anturi vs apna)					
Nopeus km/h	**NoxInLNG**	**NoxInDSL**	**NoxOutLNG**	**NoxOutDSL**	**Apna**
70	0,5984278879	0,6428571429	1,0442583732	0,8814080656	1
40	0,13191659984	0,3376102646	0,8135525261	0,8933440257	1
20	0,06646153846	0,316	0,8646153846	0,9	1
Suhde %					
Nopeus km/h	**NoxInLNG**	**NoxInDSL**	**NoxOutLNG**	**NoxOutDSL**	**Apna**
70	59,8427887902	64,285714286	104,42583732	88,140806562	100
40	13,191659984	33,761026464	81,355252606	89,334402566	100
20	6,64615384615	31,6	86,461538462	90	100
% erotus					
Nopeus km/h	**NoxInLNG**	**NoxInDSL**	**NoxOutLNG**	**NoxOutDSL**	**Apna**
70	-40,1572112098	-35,71428571	4,4258373206	-11,85919344	0
40	-86,808340016	-66,23897354	-18,64474739	-10,66559743	0
20	-93,3538461538	-68,4	-13,53846154	-10	0
Keskiarvo erotus, kaikki nopeudet					
	NoxInLNG	**NoxInDSL**	**NoxOutLNG**	**NoxOutDSL**	**Apna**
	-73,44	-56,78	-9,25	-10,84	0

Proventian laitteistossa huomattiin mittauksissa, että anturit tarvitsevat yli 200 °C:n lämpötilan, ennen kuin ne aloittavat mittaamisen. Heti kun lämpötila laskee alle tietyn pisteen, mittaus loppuu ja mitään tietoa ei anturilta lähde eteenpäin. Tämän takia data mittauspäivinäkin on paikoittain katkonaista, koska antureiden toiminta on sidottu lämpötilaan. Tällä ratkaisulla tietysti säästetään antureiden käyttöikää.

Kyseisistä antureista ei löytynyt teknisiä tietoja, mutta vastaavan anturin tiedoista selvisi, että anturi mittaa välillä 0–500 ppm +-10 %:n tarkkuudella; tämän arvon jälkeen mittavirheestä ei ole tietoa. Antureita ei ole millään tavalla kalibroitu mittausta varten.

Proventia ei kalibroi antureita mitenkään (ei mahdollista kalibroida), joten anturit ovat valmistajan kalibroimia. Proventialta saadun tiedon mukaan anturit ovat ainakin ristisensitiivisiä ammoniakille (NH_3) jota muodostuu mm. urearuiskutuksen yhteydessä. Anturit myös tunnistavat paremmin typpimonoksidin (NO) kuin typpidioksidin (NO_2). Tämä yhtälö myös vääristää tuloksia, koska pakokaasunpuhdistuksessa yhdisteitä hapetetaan jolloin anturi ei edes näe välttämättä kaikkea päästöä. Anturit saattavat Proventian mukaan nähdä myös muitakin typenkomponentteja jollain kertoimella josta heillä ei ole tietoa. Anturien lämpötilariippuvuus on myös yksi anturien heikkouksista. Anturit toimivat

parhaiten, kun ne ovat pitkään oikeassa lämpötilassa ja ovat mitanneet tasaisesti pidemmän aikaan jolloin anturien toiminta ei vääristy ja aiheuta häiriöitä mittaustulokseen.

Antureiden todellisen virheen selvittämiseksi tulisi tehdä tarkempia mittauksia, jotta virhe saataisiin selville. Dynamometritestin ja mittaustulosten perusteella voidaan todeta, että antureiden antamaan tulokseen pitää suhtautua varauksella.

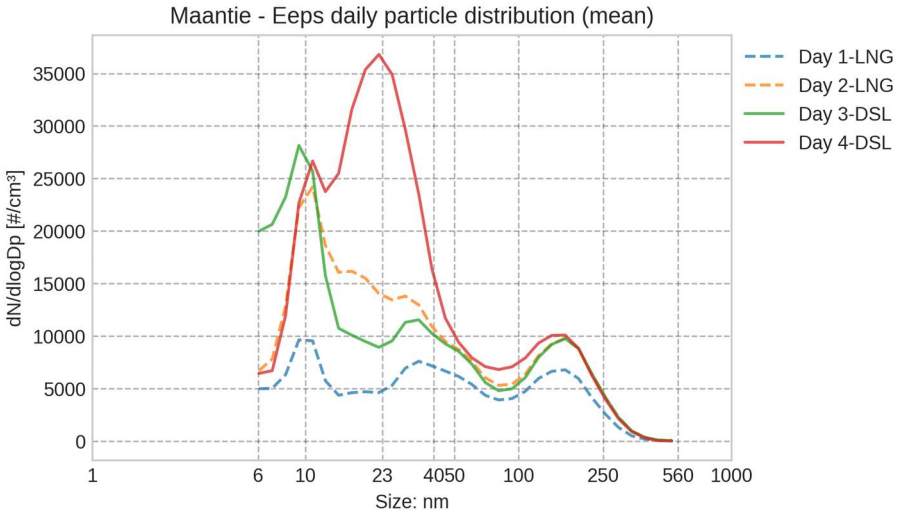
5.7 Hiukkaspäästöt

Tässä osioissa esitetään useamman hiukkasanalysaattorin tuloksia maantie- sekä toisto-osuudella.

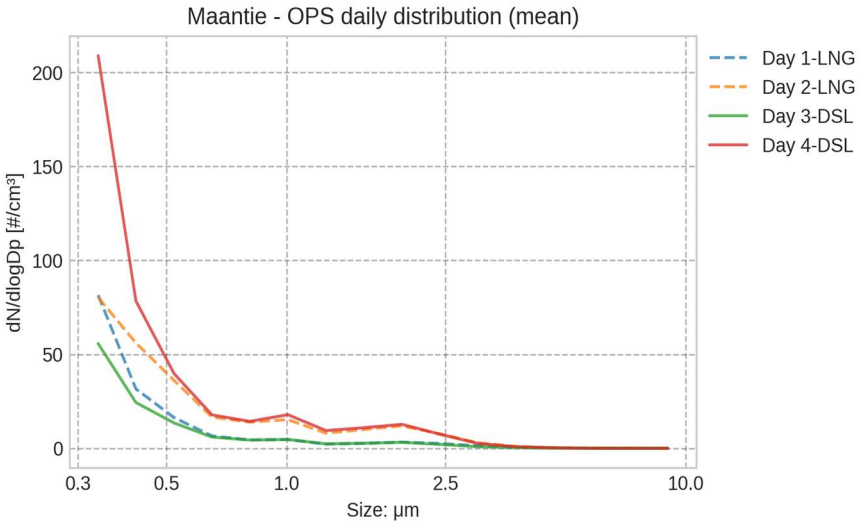
5.7.1 Maantieosuus

Liitteessä 7 on esitetty hiukkasten kokojakauman keskiarvo kanavakohtaisesti kahdella mittalaitteella, EEPS ja OPS. Kuvaajiin on lisätty normaalijakaumaa esittävä käyrä.

Kuvaajat 41 ja 42 esittävät päiväkohtaista mitattua hiukkasten kokojakauman keskiarvoa maantieosuudella logaritmisella asteikolla EEPS:llä ja OPS:lla mitattuna. Kuvaajista voidaan huomata, että kyseisellä mittausosuudella dieselajoneuvo päästää enemmän hiukkasia koko mittausalueella.

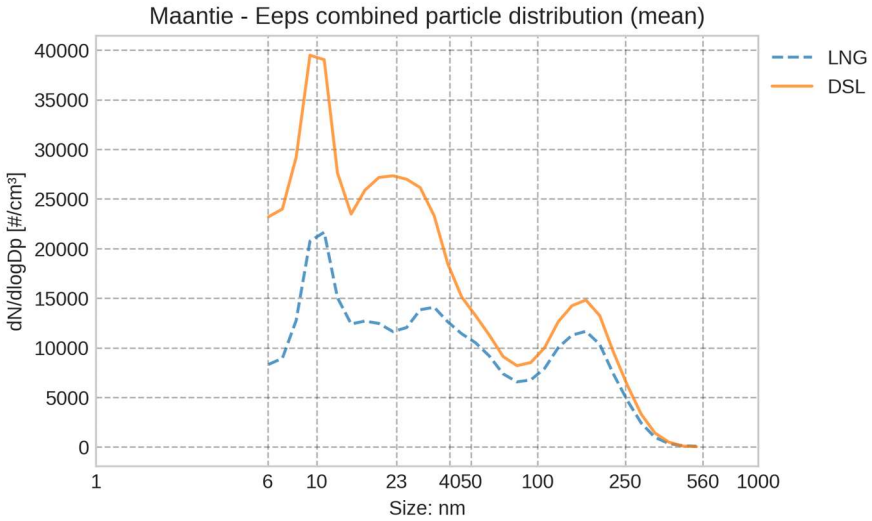


Kuvaaja 41. EEPS:n päiväkohtainen keskiarvo maantieosuudella.

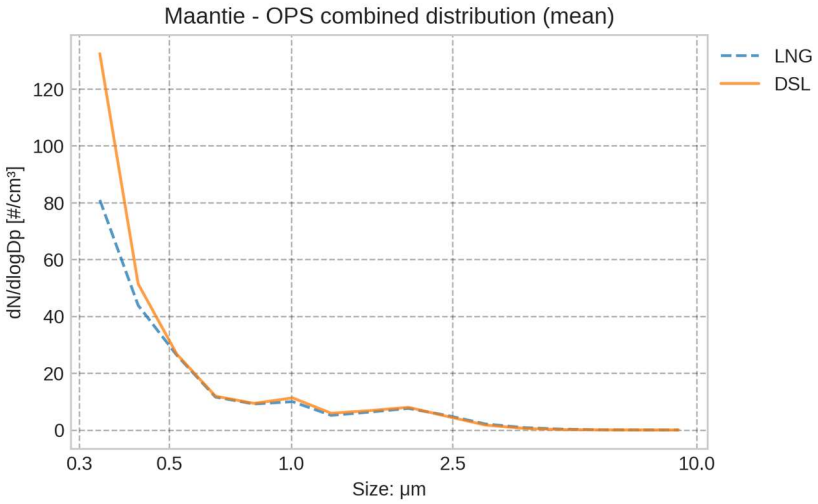


Kuvaaja 42. OPS:n päiväkohtainen keskiarvo maantieosuudella.

Kuvaajat 43 ja 44 esittävät polttoainekohtaista mitattua hiukkasten kokojakauman keskiarvoa maantieosuudella logaritmisella asteikolla EEPS:llä ja OPS:lla mitattuna. Kuvaajista voidaan huomata, että kyseisellä mittausosuudella dieselrekka päästää enemmän hiukkasia koko mittausalueella.

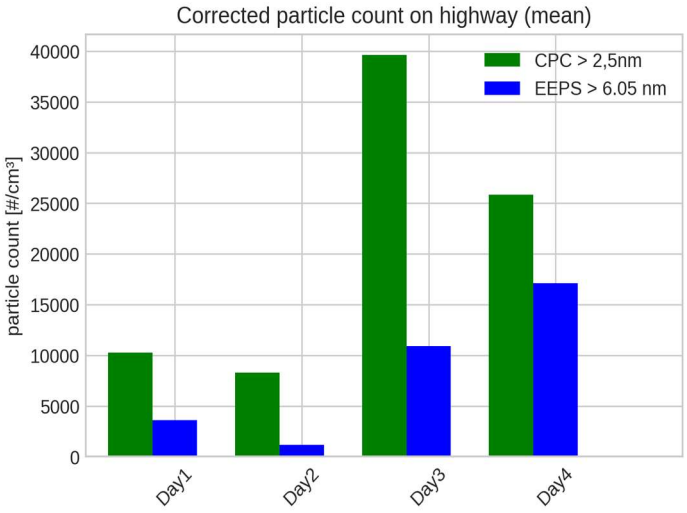


Kuvaaja 43. EEPS:n polttoainekohtainen keskiarvo maantieosuudella.

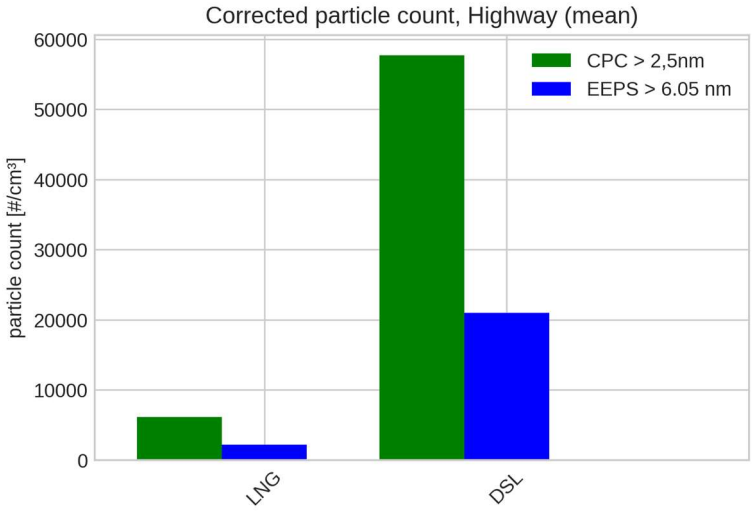


Kuvaaja 44. OPS:n polttoainekohtainen keskiarvo maantieosuudella.

Kuvaajissa 45 ja 46, esitetään hiukkasten kokonaiskonsentraationkeskiarvo maantieosuudella CPC:llä ja EEPS:llä mitattuna. Kuvaajista huomataan, että hiukkaspäästöt ovat suuremmat dieselajoneuvolla kuin maakaasuajoneuvolla.

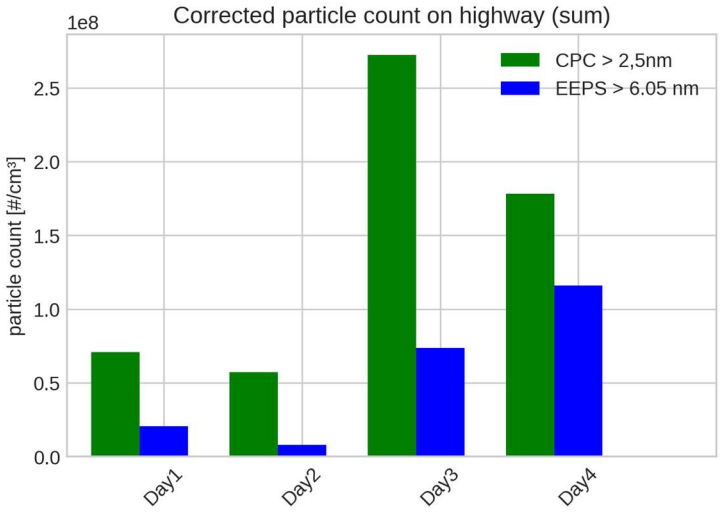


Kuvaaja 45. Päiväkohtainen hiukkaskonsentraatiokeskiarvo maantieosuudella.

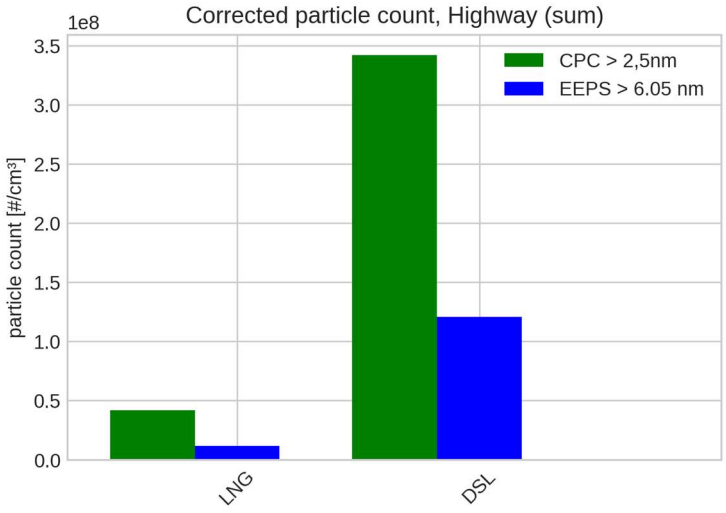


Kuvaaja 46. Polttoainekohtainen hiukkaskonsentraatiokeskiarvo maantieosuudella.

Kuvaajissa 47 ja 48 esitetään hiukkasten kokonaiskonsentraatiosumma maantieosuudella CPC:llä ja EEPS:llä mitattuna. Kuvaajista huomataan, että hiukkaspäästöt ovat suuremmat dieselajoneuvolla kuin maakaasuajoneuvolla.



Kuvaaja 47. Päiväkohtainen hiukkaskonsentraatiosumma maantieosuudella.

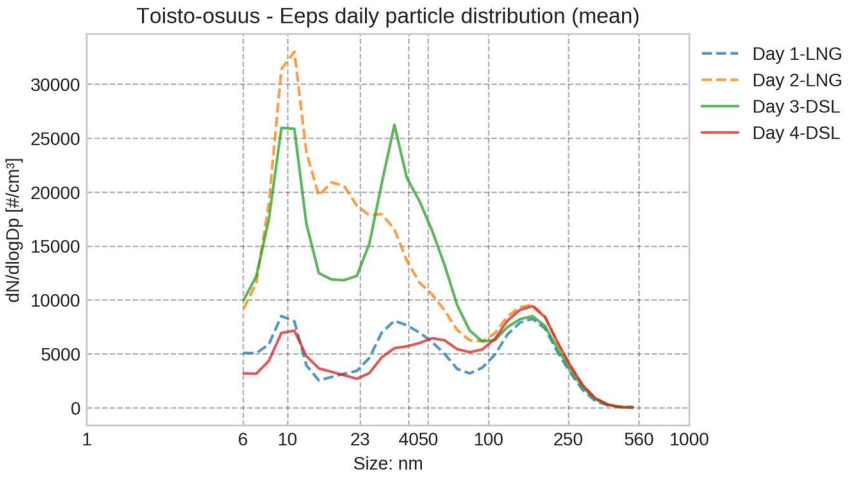


Kuvaaja 48. Polttoainekohtainen hiukkaskonsentraatiosumma maantieosuudella.

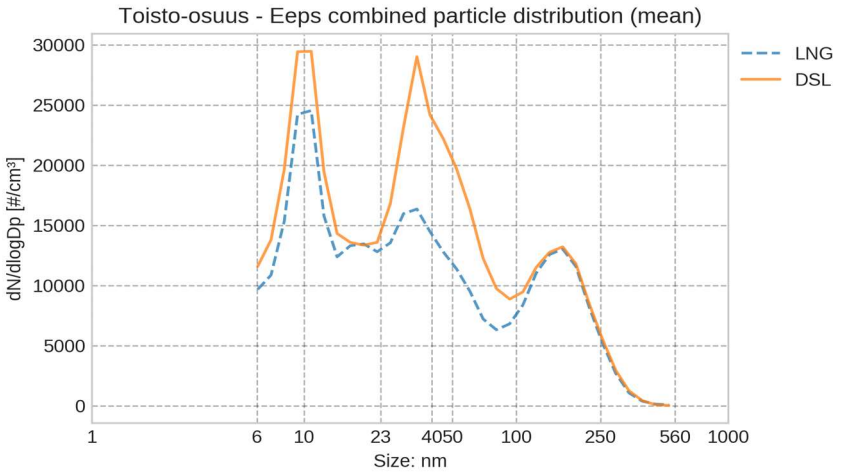
5.7.2 Toisto-osuus

Liitteessä 7 on esitetty hiukkasten kokojakauman keskiarvo kanavakohtaisesti kahdella mittalaitteella EEPS ja OPS. Kuvaajiin on lisätty normaalijakaumaa esittävä käyrä.

Kuvaajat 49 ja 50 esittävät päivä- ja polttoainekohtaista hiukkasten kokojakauman keskiarvoa toisto-osuudella logaritmisella asteikolla EEPS:llä mitattuna. Tarkastelemalla kuvaajia huomataan, että hiukkasten kokojakauma tällä mittausosuudella on hyvin samankaltainen hiukkastenkoon ollessa suurempi kuin n. 130 nm. Erot ovat suurempia, kun hiukkastenkoko on pienempi kuin 100 nm.

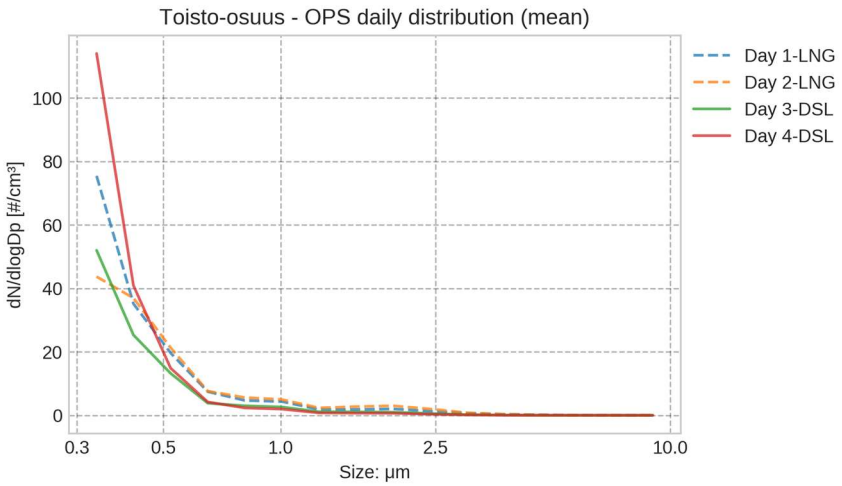


Kuvaaja 49. EEPS:n päiväkohtainen keskiarvo toisto-osuudella.

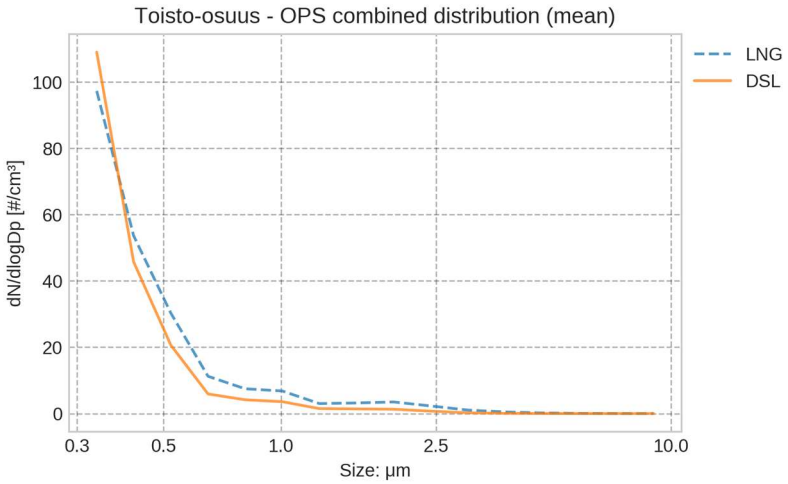


Kuvaaja 50. EEPS:n polttoainekohtainen keskiarvo toisto-osuudella.

Kuvaajat 51 ja 52 esittävät päivä- ja polttoainekohtaista hiukkasten kokojakauman keskiarvoa toisto-osuudella logaritmisella asteikolla OPS:lla mitattuna. Tarkastelemalla kuvaajia huomataan, että hiukkasten kokojakauma tällä mittausosuudella on hyvin samankaltainen.

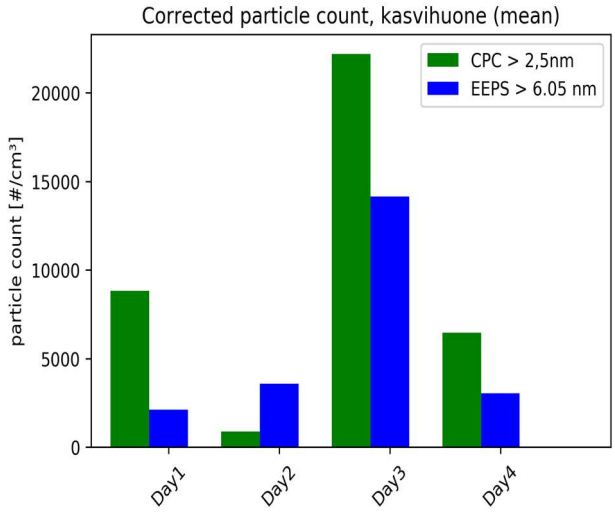


Kuvaaja 51. OPS:n päiväkohtainen keskiarvo toisto-osuudella.

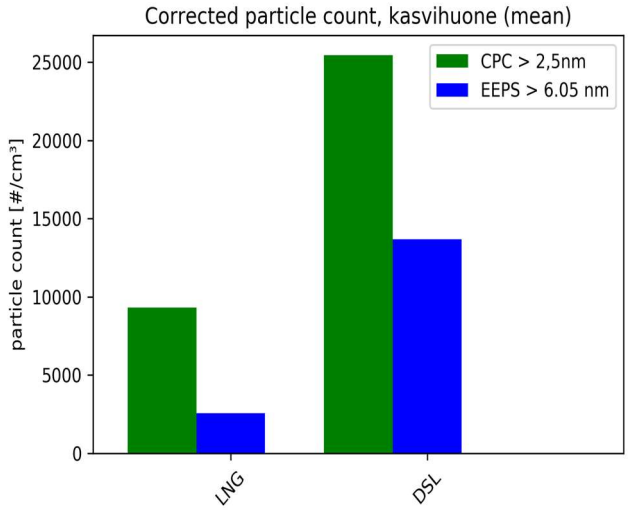


Kuvaaja 52. OPS:n polttoainekohtainen keskiarvo toisto-osuudella.

Kuvaajissa 53 ja 54 esitetään hiukkasten kokonaiskonsentraationkeskiarvo toisto-osuudella CPC:llä ja EEPS:llä mitattuna. Kuvaajista huomataan, että hiukkasten kokonaiskonsentraatio on suurempi dieselajoneuvolla.

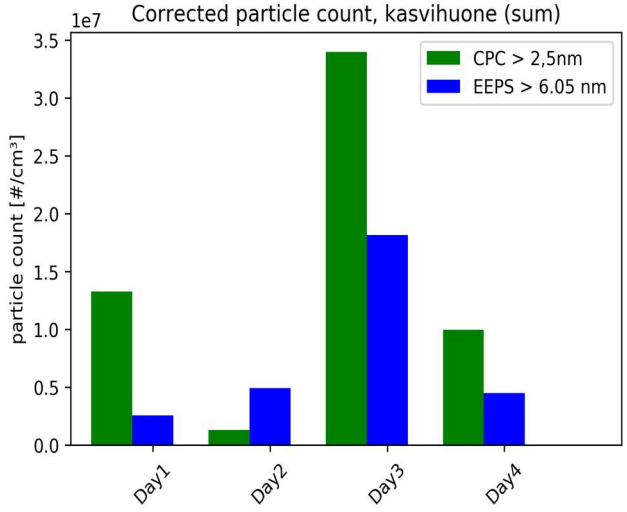


Kuvaaja 53. Päiväkohtainen hiukkaskonsentraatiokeskiarvo toisto-osuudella.

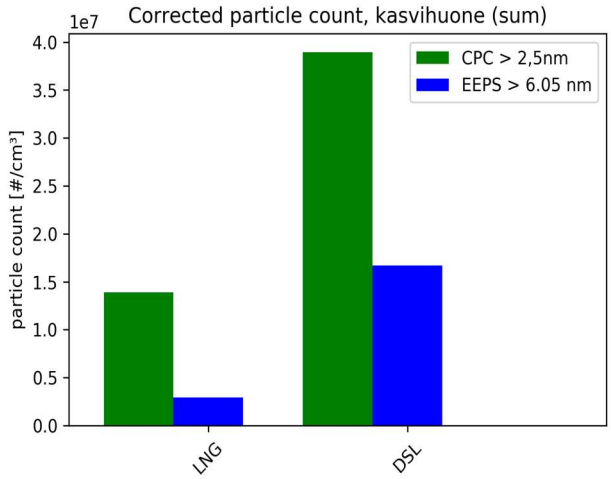


Kuvaaja 54. Polttoainekohtainen hiukkaskonsentraatiokeskiarvo toisto-osuudella.

Kuvaajissa 55 ja 56 esitetään hiukkasten kokonaiskonsentraationsumma toisto-osuudella CPC:llä ja EEPS:llä mitattuna. Tulokset indikoivat samaa kuin kuvaajat 55 ja 56.



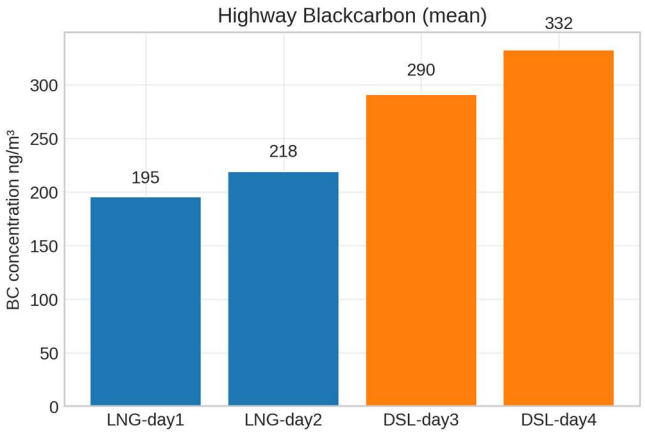
Kuvaaja 55. Päiväkohtainen hiukkaskonsentraatiosumma toisto-osuudella.



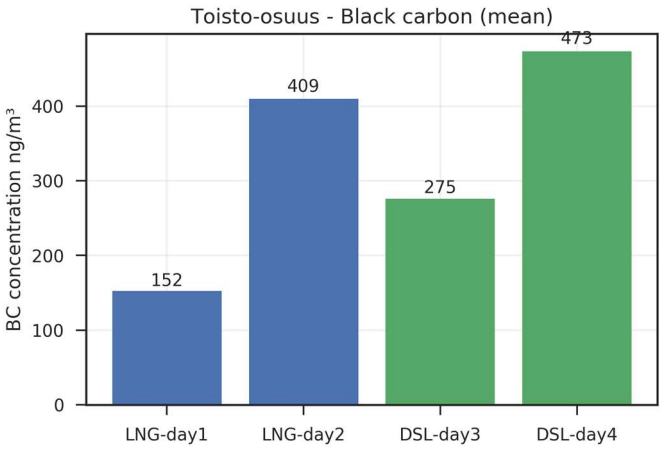
Kuvaaja 56. Polttoainekohtainen hiukkaskonsentraatiosumma toisto-osuudella.

5.8 Musta hiili

Kuvaajissa 57 ja 58 on esitetty maantie- ja toisto-osuuden mustan hiilen konsentraatiokeskiarvot päiväkohtaisesti. Kuvaajista voidaan nähdä, että pitoisuus on suurempi dieselajoneuvolla. Kuvaajat antavat saman kuvan kuin hiukkaspäästöt.

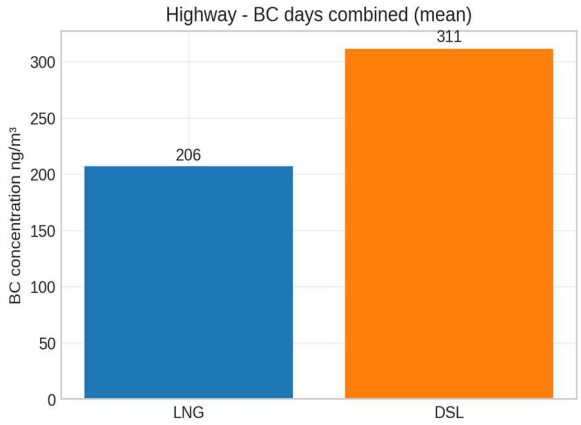


Kuvaaja 57. Mustan hiilen päiväkohtainen keskiarvo maantieosuudella.

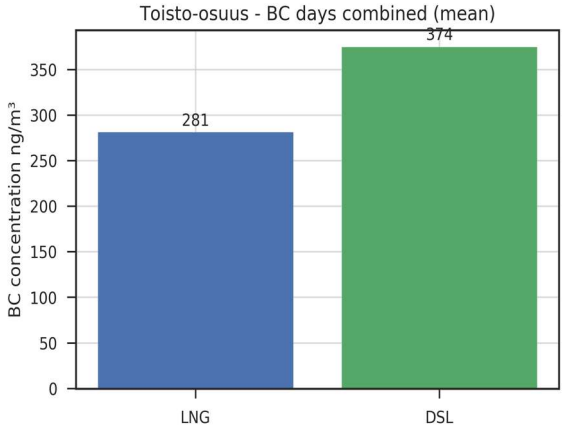


Kuvaaja 58. Mustan hiilen päiväkohtainen keskiarvo toisto-osuudella.

Kuvaajissa 59 ja 60 on esitetty mustan hiilen konsentraationkeskiarvo maantie- ja toisto-osuudelta polttoainekohtaisesti. Kuvaajista huomataan, että maakaasu on päästöltään hieman suurempi.



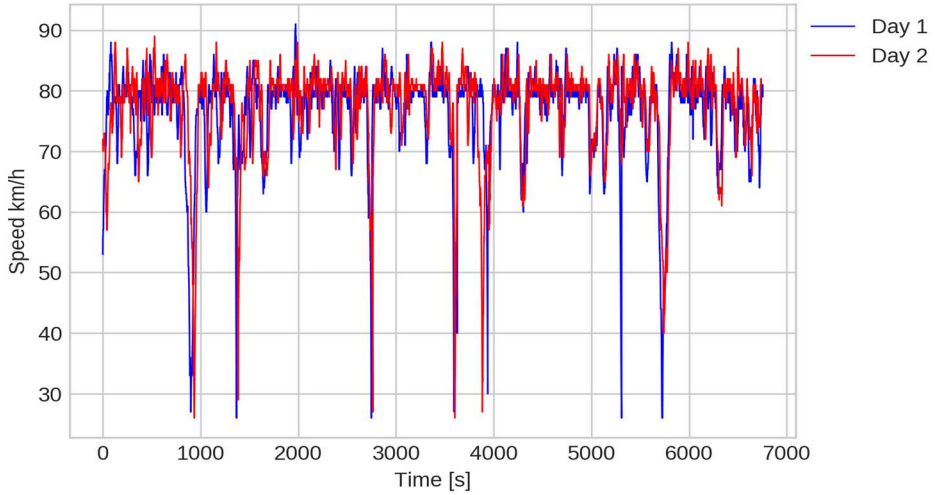
Kuvaaja 59. Mustan hiilen, polttoainekohtainen keskiarvo maantieosuudella.



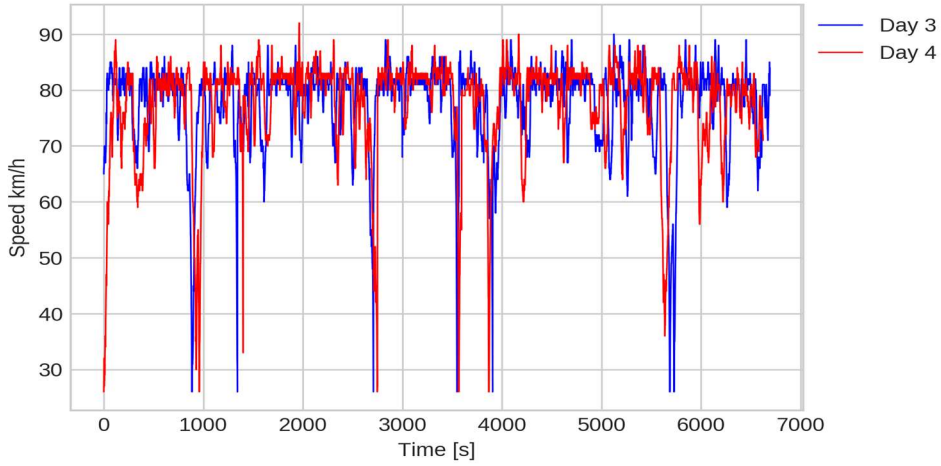
Kuvaaja 60. Mustan hiilen, polttoainekohtainen keskiarvo toisto-osuudella.

5.9 Kuljettajan ajotavan arviointi

Kuljettajan ajotavan arviointi suoritettiin vertailemalla päivä- ja osuuskohtaisia eroja sekä ajoneuvoista tallennettuja parametrejä. Kuvaajissa 61 ja 62 on esitetty ajoneuvon nopeus päiväkohtaisesti maantieosuudella.

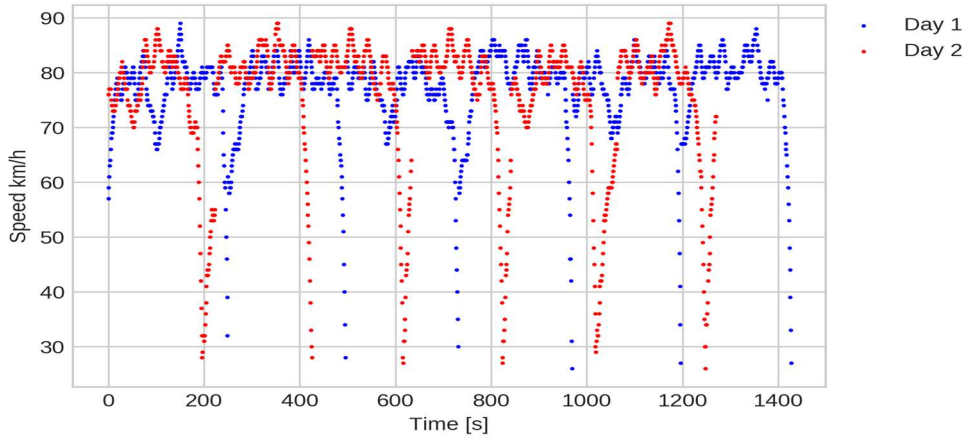


Kuvaaja 61. Nopeus maantiellä, päivät 1 ja 2.

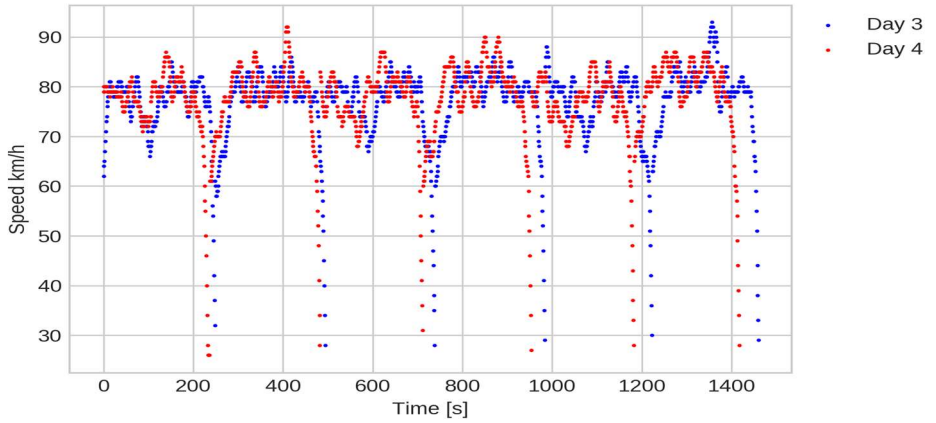


Kuvaaja 62. Nopeus maantiellä, päivät 3 ja 4.

Kuvaajissa 63 ja 64 on esitetty ajoneuvon nopeus päiväkohtaisesti toisto-osuudella.



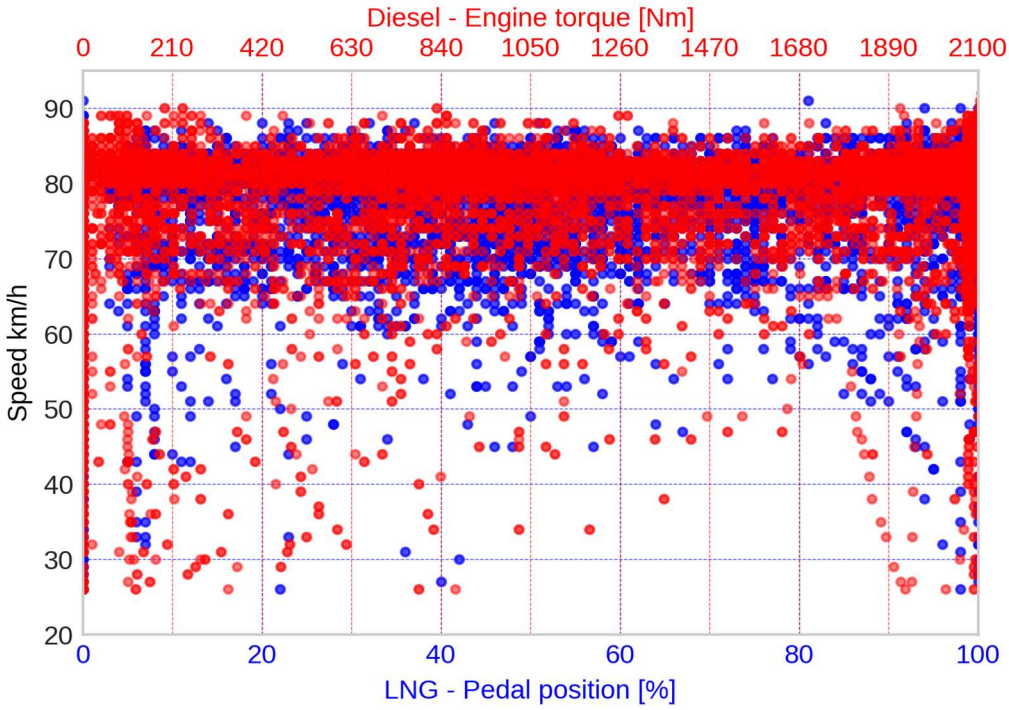
Kuvaaja 63. Nopeus toisto-osuudella, päivät 1 ja 2.



Kuvaaja 64. Nopeus toisto-osuudella, päivät 3 ja 4.

Kuvaajista 61 – 64 voidaan todeta, että kuljettaja on ajanut hyvin samankaltaisesti jokaisena mittauspäivänä.

Kuvaajassa 65 on esitetty kuljettajan kaasunkäyttö suhteutettuna nopeuteen. Punainen moottorin vääntö kuvaa dieselmittauspäivien kaasunkäyttöä ja sininen maakaasuajoneuvon kaasuläpän asentoa. Kuvaajasta voidaan päätellä, että maakaasuautoa joutuu ajamaan hieman eri tavalla kuin dieselajoneuvoa johtuen moottorin luonteesta.



Kuvaaja 65. Kaasunkäytön suhde nopeuteen.

6 Päästökertoimen laskenta

Päästösuhde ER_X, määritetään laskettavan aineen x mitatusta päästöstä, josta on vähennetty laskettavan aineen tausta pitoisuus. Tämä jaetaan mitatusta hiilidioksidipäästöstä, josta on vähennetty hiilidioksidipäästön taustapitoisuus kaavalla (1), jossa x on kaasujen tai hiukkaspäästön määrä tai hiukkasten massa konsentraatio. X_{bg} ja $CO_{2,bg}$ ovat taustapitoisuuksia. [33, appendix 1, s. 3.]

$$ER_x = \frac{X - X_{bg}}{CO_2 - CO_{2_{bg}}} = \frac{\Delta X}{\Delta CO_2} \qquad (1)$$

Päästökerroin EF_X (g/kg polttoainetta) on aineen X pitoisuus määrä per poltettua polttoainekiloa kohti. EF_X määrittää seuraavasti kaavoissa 2a, 2b, 2c. Kaavoissa M_X ja Mco_2 ovat kaasujen moolimassoja. $EFco_2$ on päästökerroin hiilidioksidille, joka vaihtuu polttoaineen mukaan. Käytämme laskuissa seuraavia arvoja, dieselille 3160 g/kg ja maakaasulle 2540 g/kg per poltettu polttoaine kilo. [34, appendix 1, s. 3.]

Kaasuille

$$EF_x = ER_x \cdot \frac{M_x}{M_{CO_2}} \cdot EF_{CO_2} \qquad (2.a)$$

Hiukkasmäärälle

$$EF_{Ntot} = ER_{Ntot} \cdot \frac{RT}{p \cdot M_{CO_2}} \cdot EF_{CO_2} \qquad (2.b)]$$

Hiukkasmassalle

$$EF_{PM} = ER_{PM1} \cdot EF_{CO_2} \qquad (2.c)$$

6.1 Moolimassan määritys typenoksidille (M_{NOX})

Koska typenoksidi on yhdiste, tulee moolimassa määrittää mitatuista komponenteista NO ja NO_2.

Laskemme kertoimet komponenteille NO, NO_2 ja NO_X kaavoilla 3.a, 3.b ja 3.c. Kaavoissa M, on komponenttien moolimassa. Kaavalla 4.a ja 4.b saamme selville komponenttien moolimassa kertoimen. Kaavalla 5 saamme laskettua yhdisteelle moolimassan, jonka voimme sijoittaa kaavaan 2.a. Koska NO ja NO_2 päästöt on mitattu ppb-muodossa, arvot on muutettu ppm:ksi jakamalla mitattu arvo 1000:lla.

$$NO_{ratio} = \frac{NO - NO_{bg}}{(NO + NO_2) - \left(NO_{bg} + NO_{2_{bg}}\right)} \cdot M_{NO} = \frac{\Delta NO}{\Delta NO_x} \cdot M_{NO} \quad (3.a)$$

$$NO_{2rati} = \frac{NO_2 - NO_{2_{bg}}}{(NO + NO_2) - \left(NO_{bg} + NO_{2_{bg}}\right)} \cdot M_{NO2} = \frac{\Delta NO_2}{\Delta NO_x} \cdot M_{NO_2} \quad (3.b)$$

$$NOx_{ratio} = NO_{ratio} + NO_{2ratio} \quad (3.c)$$

$$M_{NOratio} = \frac{NO_{ratio}}{NO_{xratio}} \quad (4.a)$$

$$M_{NO2ratio} = \frac{NO_{2ratio}}{NO_{xratio}} \quad (4.b)$$

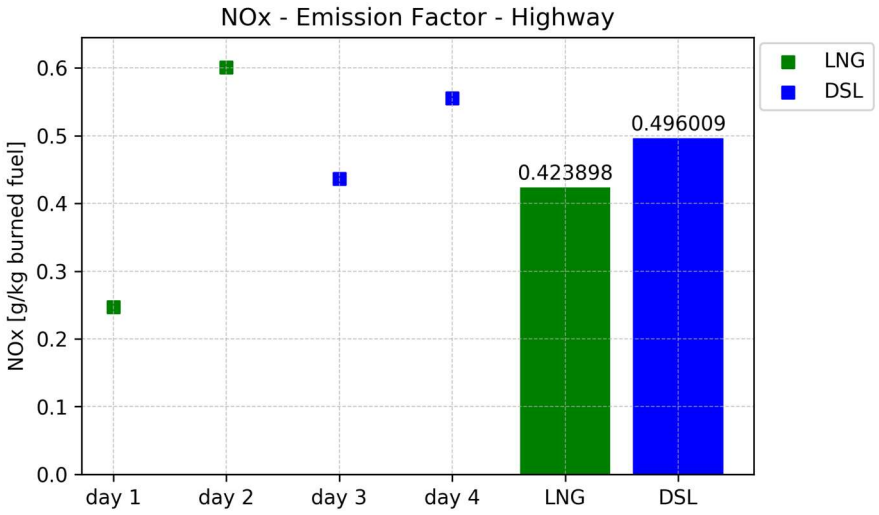
$$M_{NOx} = (M_{NOrat} \cdot M_{NO}) + \left(M_{NO_2 ratio} \cdot M_{NO_2}\right) \quad (5)$$

6.2 Typenoksidin päästökerroin (EF_{NOX})

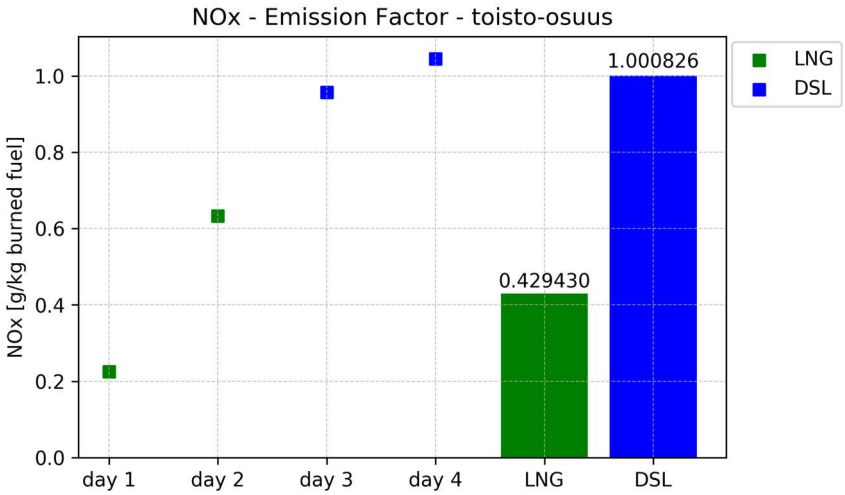
Laskenta on tehty kaavalla 2.a, apuna käyttäen kaavoja 1, 3.a, 3.b, 3.c, 4.a, 4.b ja 5.

Kuvaajissa 66 ja 67 on esitetty typenoksidin päästökerroin (g/kg poltettu polttoainekilo) maantieosuudelta sekä kasvihuoneilmiöllä suoritetuilla toisto-osuudella. Kuvaajista huomataan, että typenoksidin päästökerroin maantieosuudella on n. 15 % ja toisto-osuudella n. 57 % pienemmät maakaasurekalla verrattuna dieselajoneuvoon (LNG/DSL). Tuloksien yhteen lasketusta keskiarvosta voidaan todeta LNG-rekan olevan typpioksidin päästökertoimeltaan n. 57 % puhtaampi kuin dieselrekka (LNG/DSL).

Typpioksidin päästökertoimien laskenta on esitetty liitteessä 1.



Kuvaaja 66. NO_X-päästökerroin maantieosuudella.



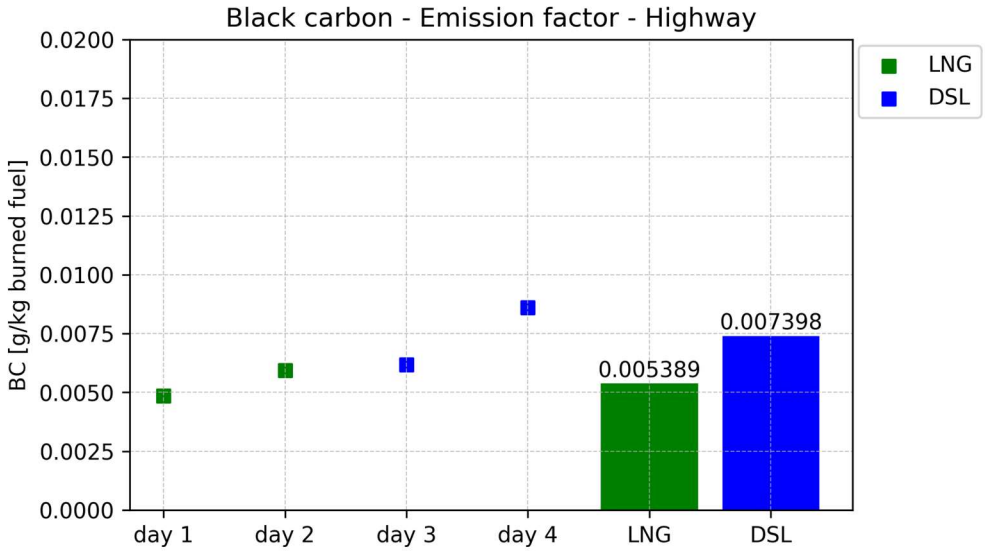
Kuvaaja 67. NO_X-päästökerroin toisto-osuudella.

6.3 Mustan hiilen päästökerroin (EF_{BC})

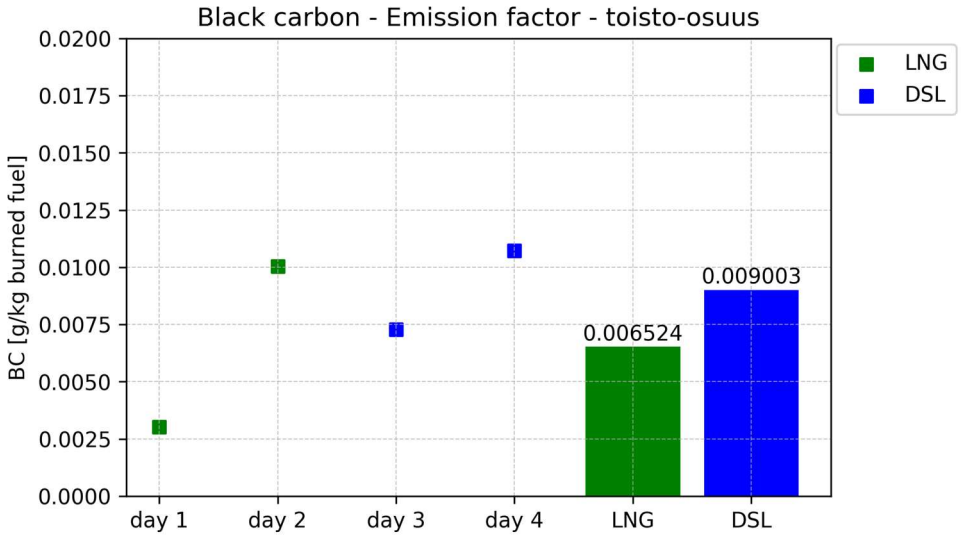
Laskenta on tehty kaavalla 2.c käyttäen apuna kaavaa 1. ER_{BC}:n laskennassa mitattu päästö ng/m^3 on muutettu muotoon g/m^3.

Kuvaajissa 68 ja 69 on esitetty mustan hiilen päästökerroin (g/kg poltettu polttoainekilo) maantieosuudella sekä Kasvihuoneilmiöllä suoritetulla toisto-osuudella. Kuvaajista huomataan, että mustan hiilen päästökerroin maantieosuudella on n. 27 % ja toisto-osuudella n. 28 % pienemmät maakaasurekalla verrattuna dieselrekkaan (LNG/DSL). Tuloksien yhteen lasketusta keskiarvosta voidaan todeta maakaasurekan olevan n. 27 % mustan hiilen päästökertoimeltaan puhtaampi kuin dieselrekka (LNG/DSL).

Mustan hiilen päästökertoimien laskenta on esitetty liitteessä 2.



Kuvaaja 68. Mustan hiilen päästökerroin maantieosuudella.



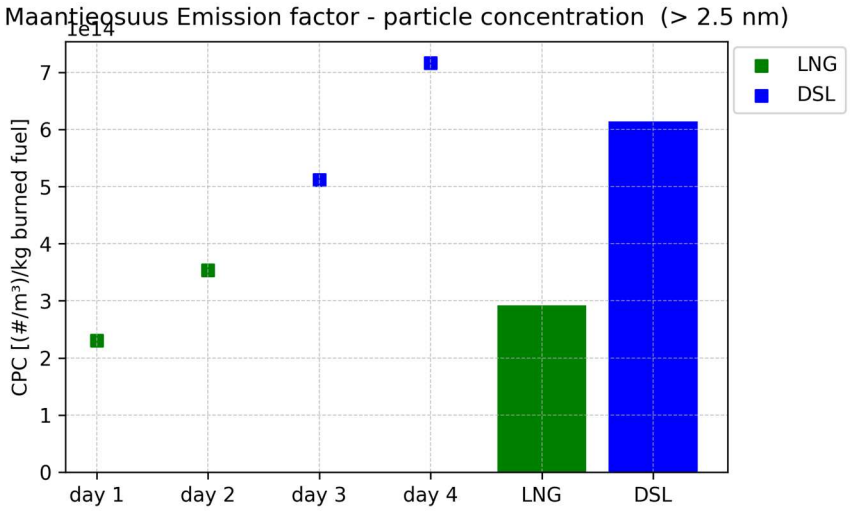
Kuvaaja 69. Mustan hiilen päästökerroin toisto-osuudella.

6.4 Hiukkaskonsentraation päästökerroin (EF_{Ntot})

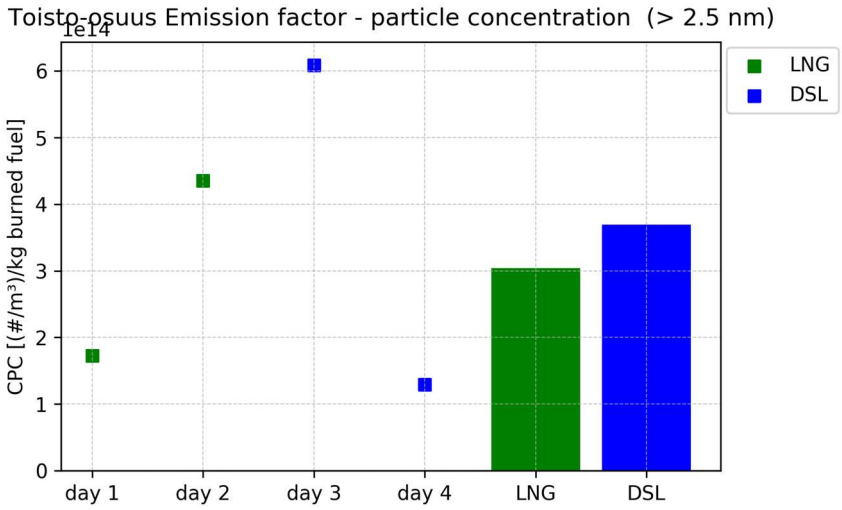
Laskenta on tehty kaavalla 2.c, tulokset esitetään laitekohtaisesti (CPC, EEPS, OPS). Hiukkasten katkaisurajat ovat seuraavat CPC >2,5 nm, EEPS > 5,06 nm ja OPS > 307 nm. Koska laitteet antavat mitatun tuloksen muodossa #/cm^3, ER_{Ntot} laskennassa mitattu CO_2 päästö on muodossa g/m^3, on se muutettu muotoon g/cm^3. Kaavassa 2.c muuttuja R on moolinen kaasuvakio 8,3145 J/mol/K, T on valitseva lämpötila kelvineinä ja p on vallitseva ilmanpaine 101300 Pa.

Liitteessä 3 on esitetty hiukkaskonsentraation laskenta.

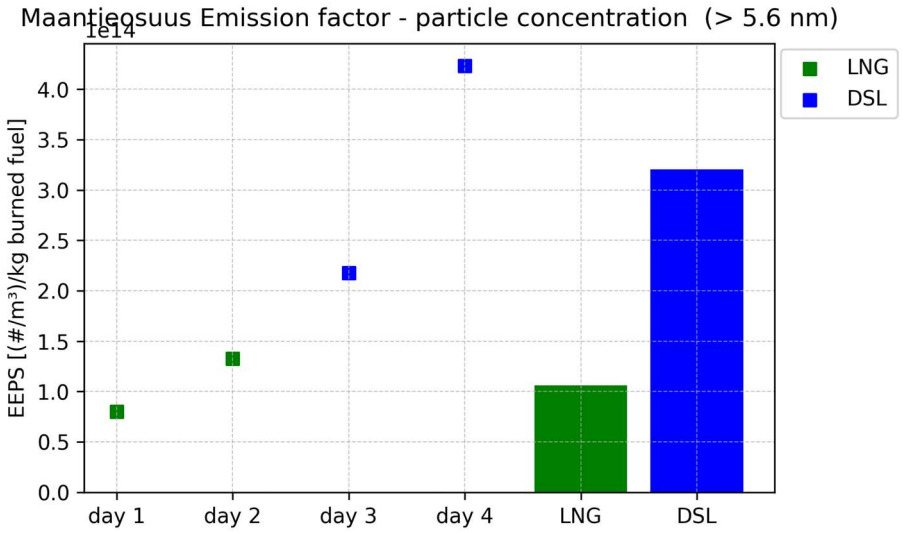
Kuvaajissa 70—75 esitetään hiukkaskonsentraation päästökerroin laitekohtaisesti.



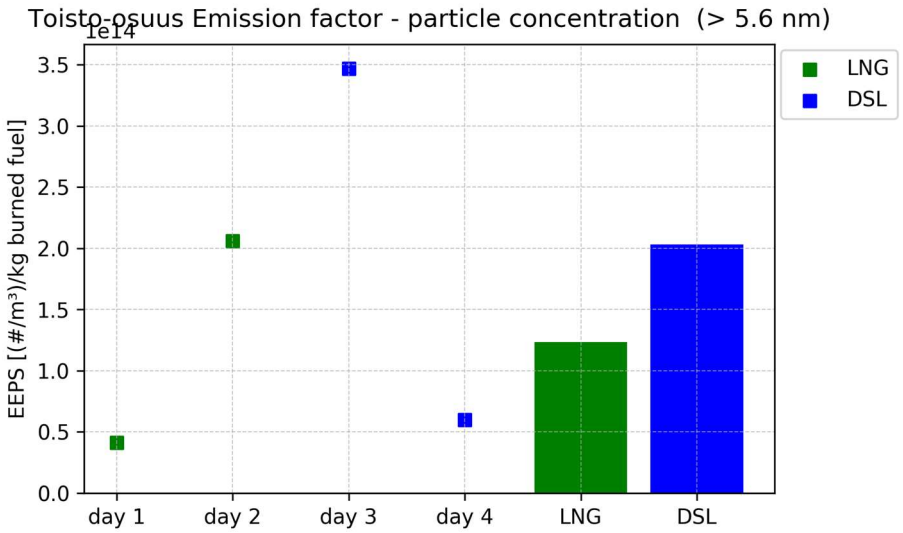
Kuvaaja 70. CPC, päästökerroin maantieosuudella.



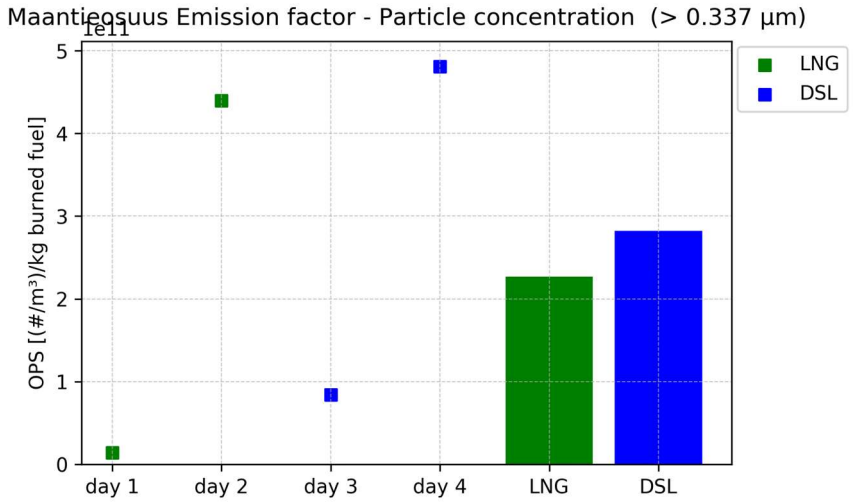
Kuvaaja 71. CPC, päästökerroin toisto-osuudella.



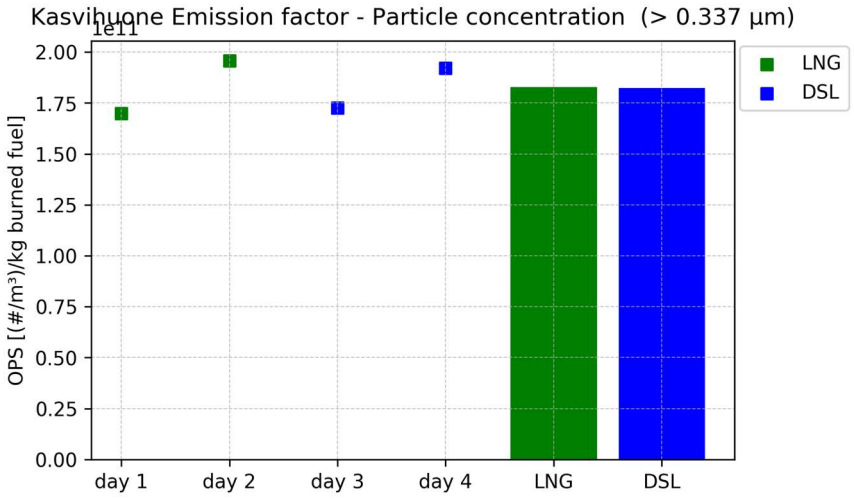
Kuvaaja 72. EEPS, päästökerroin maantieosuudella.



Kuvaaja 73. EEPS, päästökerroin toisto-osuudella.

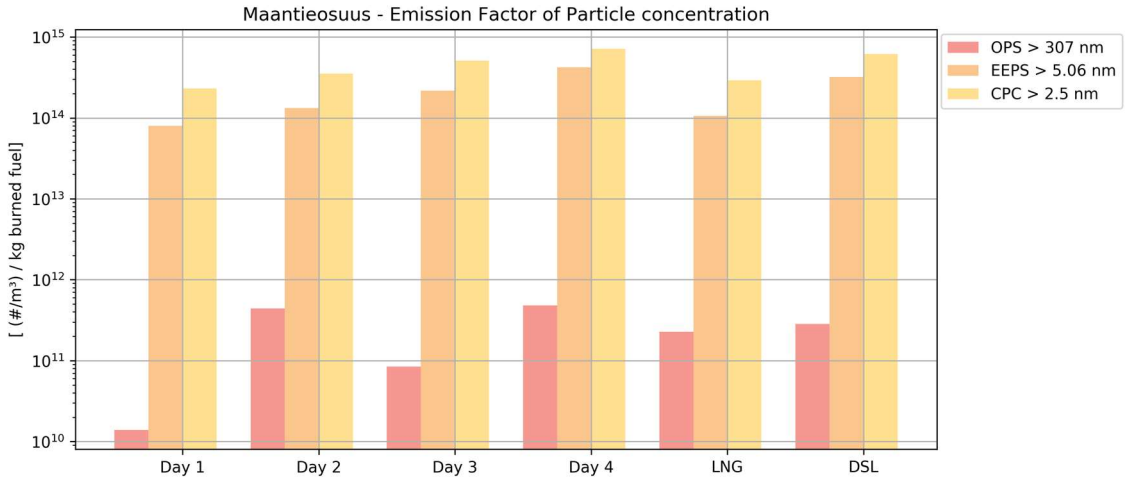


Kuvaaja 74. OPS, päästökerroin maantieosuudella.

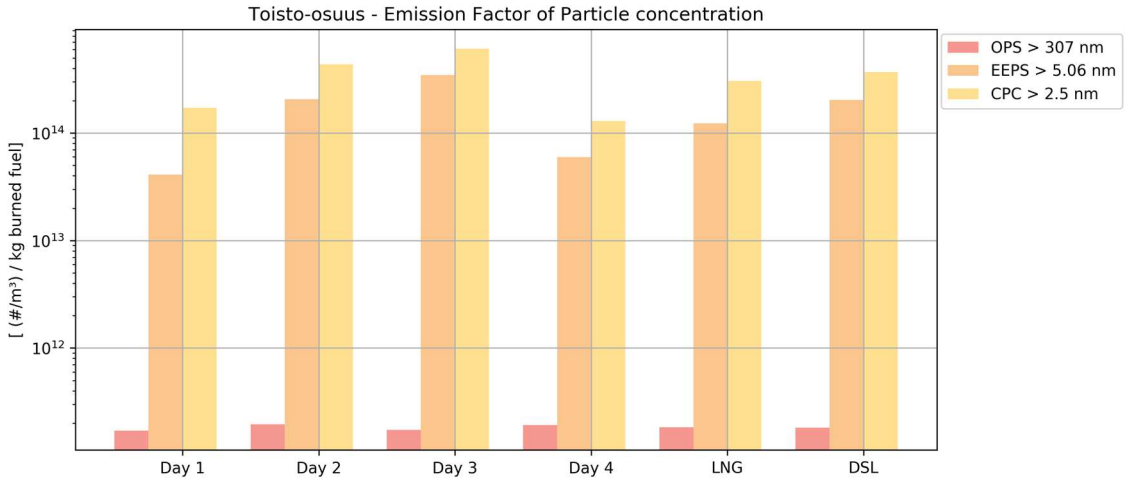


Kuvaaja 75. OPS, päästökerroin toisto-osuudella.

Kuvaajissa 76 ja 77 on esitetty päästökerroin hiukkaskonsentraatiolle päivä-, laite- sekä polttoainekohtaisesti.



Kuvaaja 76. Hiukkaskonsentraation päästökerroin maantieosuudella.



Kuvaaja 77. Hiukkaskonsentraation päästökerroin toisto-osuudelta.

Taulukossa 9 on esitetty kaikkien päästökomponenttien päästökertoimet. Taulukosta näemme, että päiväkohtainen varianssi on suuri varsinkin hiukkasten kohdalla. Polttoainekohtainen varianssi on huomattavasti pienempi ja havainnollistaa paremmin ajoneuvojen päästökertoimien eroja. Kuten taulukosta 9 voidaan nähdä, hiukkasten osuus maakaasulla on suurempi maantieosuudella kuin toisto-osuudella. Toisaalta kokonaiskuva indikoi kuitenkin sitä, että maakaasulla on näistä kahdesta vaihtoehdosta vähemmän ympäristöä kuormittava vaikutus.

Taulukko 9. Päästökertoimet.

	EF Ntot (CPC) #/m³ / kg fuel	EF Ntot (EEPS) #/m³ / kg fuel	EF Ntot (OPS) #/m³ / kg fuel	EF NOx g / kg fuel	EF BC g / kg fuel
	X10^14	X10^14	X10^11		
Maantieosuus					
LNG – Day 1	2.30	0.80	0.14	0.25	0.0048
LNG – Day 2	3.53	1.33	4.40	0.60	0.0059
DSL – Day 3	5.11	2.18	0.84	0.44	0.0062
DSL – Day 4	7.16	4.24	4.81	0.56	0.0086
LNG	2.92	1.06	2.27	0.42	0.0054
DSL	6.14	3.21	2.82	0.50	0.0074
Toisto-osuus					
LNG – Day 1	1.72	0.41	1.70	0.23	0.0030
LNG – Day 2	4.35	2.06	1.96	0.63	0.0100
DSL – Day 3	6.09	3.47	1.73	0.96	0.0073
DSL – Day 4	1.29	0.60	1.92	1.04	0.0107
LNG	3.04	1.24	1.83	0.43	0.0065
DSL	3.69	2.03	1.82	1.00	0.0090

7 Yhteenveto

Työn tarkoituksena oli selvittää Euro 6 -päästöluokan täyttävien maakaasu- sekä dieselkonttivetorekkojen todellisten pakokaasupäästöjen eroja. Työ tehtiin yhteistyössä Containership Oyj:n, Iveco Finland Oy:n, Gasum Oy:n sekä Liikenteen turvallisuusvirasto Trafin kanssa. Tutkivana osapuolena työssä oli Metropolia Ammattikorkeakoulu.

Työn painotus oli tutkia typpioksidipäästöjen (NO_X) eroja polttoaineiden kesken ja osoittaa eri polttoaineita käyttävien rekkojen pakokaasupäästöjen eroja, ajoneuvojen todellisessa käyttöympäristössä.

Työssä tehdyt mittaukset suoritettiin kokeellisella jahtausmittaustekniikalla Nuuskija-tutkimusajoneuvolla. Mittauksia varten mitattaviin ajoneuvoihin asennettiin Proventian Procare Drive -NO_X-monitorointijärjestelmä, jolla saatiin typpioksidipäästöille (NO_X) linkki rekkojen ja Nuuskijan välille.

Mittaustulokset vastasivat alkuodotuksia. Hiilidioksidi- (CO_2) ja typpioksidipäästöt (NO_X) olivat maakaasurekalla pienemmät kuin dieselrekalla joka näkyy kaikissa Nuuskijalla mitatuissa tuloksissa.

Hiukkaspäästöt olivat myös maakaasurekalla pienemmät kuin dieselrekalla, tämä nähdään hyvin hiukkasten kokonaiskonsentraatiossa. Hiukkasten kokojakauma kertoo myös, että dieselrekka muodostaa enemmän pienhiukkasia kuin vastaava maakaasurekka. Erot kuitenkin tasoittuvat hiukkastenkoon kasvaessa. Tämä tasoittuminen nähdään myös mustan hiilen konsentraation kohdalla, kun partikkelien aerodynaaminen halkaisija on ≤ 2.5 µm. Kokojakauman tasoittumisesta huolimatta maakaasurekka on tässäkin tapauksessa pienempi päästöisempi kuin vastaava dieselrekka.

Kaikkien Nuuskijalla mitattujen päästöjen osalta maakaasurekka oli puhtaampi, kuin vastaava dieselrekka joka nähdään myös lasketuissa päästökertoimissa.

On hyvä ottaa huomioon, että maakaasurekan ja dieselrekan pakokaasunjälkikäsittelyssä on huomattava ero. Maakaasurekan pakokaasunpuhdistus on todella yksinkertainen ja dieselrekan pakokaasupuhdistus on monimutkainen, huomattavasti kalliimpi sekä vikaherkempi.

Tuloksia tarkastellessa on otettava huomioon, että jahtausmittauksissa olosuhteet ovat jokseenkin merkittävä muuttuja tuloksien kannalta, mikä voidaan huomata päiväkohtaisissa tuloksissa.

Mittaustavasta voidaan kuitenkin sanoa, että tulokset ovat johdonmukaiset, kun olosuhteet ovat samankaltaiset. Olosuhteiden vaikutusta tuloksiin voidaan pienentää huomattavasti lisäämällä toistoja, jolloin keskimääräinen virhe mittaustapahtumissa pienenee ja trendin löytäminen tuloksista on helpompaa.

Proventian laitteet, jotka olivat rekoissa kiinni, eivät välttämättä anna kovinkaan hyvää kuvaa typpioksidipäästöistä (NO_X), mikä näkyy varsinkin maakaasurekan tuloksissa.

Laitteistoon liittyy paljon teknisiä ongelmia mm: laitetta ei voi kalibroida, anturit ovat ristisensitiivisiä ainakin ammoniakille, anturit tunnistavat paremmin typpimonoksidin (NO) kuin typpidioksidin (NO_2) ja anturit saattavat nähdä muitakin typenkomponentteja tuntemattomalla kertoimella. Laitevalmistajalla ei ole mahdollisuutta kalibroida antureita millään tavoin.

Typpioksidi (NO_X) pitoisuuksien ollessa todella korkeat, laitteiden mittatarkkuus heikkenee ja näin ollen vääristää tulosta. Toisaalta Proventian anturit antoivat dieselin osalta hyvin samanlaisia pitoisuuksia kuin rekan oma NO_X-anturi, joten sinänsä anturit toimivat dieselissä kuin niiden pitää.

Proventia epäili, että maakaasurekassa ollut laitteisto olisi kontaminoitunut jotenkin joka selittäisi ristiriitaiset tulokset. Toisaalta jos laitteisto kontaminoituu heti ensi käyttämällä, voidaan todeta, että laitteiston anturit eivät ole kovinkaan hyviä.

Proventia kommentoi, että laitteisto on tarkoitettu dieselpäästöjen mittaamiseen sekä laitteistoa ei ole tarkoitettu tarkkaan tutkimuskäyttöön. Toisaalta näitä väitteitä ei ilmene missään muodossa heidän markkinointimateriaalissaan. Maakaasunosalta Proventia kommentoi, että oletettavasti jokin säätelemätön päästö olisi heidän mukaansa vaikuttanut anturin näyttämään ja näin vääristänyt tuloksia.

Dynamometritesti osoitti, että anturikohtaiset erot olivat suuret, kun kaikilla antureilla mitattiin täysin samaa pakokaasupäästöä. Tämä testi tukee myös kuvaa anturien epätasalaadusta mitata typpioksidipäästöjä (NO_X). Järjestelmän suurin heikkous on sen käyttämät anturit jotka toimivat kohtalaisesti dieselajoneuvoilla. Anturikohtaisten erojen ja antureista selvinneiden heikkouksien valossa järjestelmän tuloksia ei voida pitää kovinkaan luotettavana. Järjestelmä antaa kyllä suuntaa-antavan tuloksen joka saattaa olla riittävä, kun ei tarvita tarkkaa tulosta typpioksidipäästöistä (NO_X).

Mittauspäivien määrän olisi hankkeen kannalta ollut hyvä olla suurempi, jotta tulosten hajontaa olisi saatu pienemmäksi. Toisaalta kun tarkoitus oli selvittää tosielämässä tulevia pakokaasupäästöjä, väistämättäkin on selvää, ettei ympäristön aiheuttamia muuttujia voida täysin eliminoida tai kontrolloida mittausten yhteydessä.

Tarkempien erojen selvittämiseksi tulisi tehdä mittauksia myös kontrolloidusti dynamoterillä. PEMS-mittaukset antaisivat myös oman lisäaspektinsa mittauksiin. Näiden mittausten toteuttaminen vaatisi huomattavasti isompaa rahallista ja ajallista panostusta hankkeeseen.

Lähteet

1. Self-Study programme 230, Motor Vehicle Exhaust Emissions. Verkkoaineisto. Volkswagen. <http://www.volkspage.net/technik/ssp/ssp/SSP_230.pdf>. Luettu 7.2.2018.

2. Anenberg, Susan C. & Schwartz, Joel. 2012. Global Air Quality and Health Co-benefits of Mitigating Near-Term Climate Change through Methane and Black Carbon Emission Controls. Verkkoaineisto. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3385429/#>. Luettu 22.2.2018.

3. Colbeck, Ian & Lazaridis, Mihalis. 2013. Aerosol Science: Technology and Applications. John Wiley & Sons, Incorporated.

4. Salonen, Raimo O. & Pennanen, Arto. 2006. Pienhiukkasten vaikutus terveyteen. Helsinki: Tekes.

5. Particulate Matter Air Quality guidelines – Secound Edition. 2000. Verkkoaineisto The World Health organization. <http://www.euro.who.int/__data/assets/pdf_file/0019/123085/AQG2ndEd_7_3Particulate-matter.pdf>. Luettu 8.2.2018.

6. Lecture notes. Verkkoaineisto. Georgia Institute of Technology School of Earth and Atmospheric Sciences. <http://www.aerosols.eas.gatech.edu/EAS%20Graduate%20Lab/Class%20Notes%20Aerosols%20and%20Size%20Distrn.pdf>. Luettu 8.2.2018.

7. Council Directive 88/77/EEC. 1988. The European Parliament and the Council of the European Union.

8. Council Directive 91/441/EEC. 1991. The European Parliament and the Council of the European Union.

9. Council Directive 94/12/EC. 1994. The European Parliament and the Council of the European Union.

10. Commission Directive 98/69/EC. 1998. The European Parliament and the Council of the European Union.

11. Commission Directive 2002/80/EC. 2002. The European Parliament and the Council of the European Union.

12. Council Regulation No.715/2007/EC. 2007. The European Parliament and the Council of the European Union.

13. Emissions standards, EU: Heavy Duty Truck and Bus Engines. Verkkoaineisto. Dieselnet.com. <https://www.dieselnet.com/standards/eu/hd.php>. Luettu 9.2.2018.

14. Ohje Ottomoottorikäyttöisten ajoneuvojen pakokaasupäästöjen tarkastus katsastuksessa. 2016. Verkkoaineisto. Liikenteenturvallisuus virasto Trafi. <https://www.trafi.fi/filebank/a/1467194589/ddfc82da9a3f43f0b95381a3d6ca2f7e/21984-Pakokaasupaastojen_tarkastus.pdf>. Luettu 3.2.2018.

15. Ohje Dieselkäyttöisten autojen pako-kaasupäästöjen tarkastus katsastuksessa. 2017. Verkkoaineisto. Liikenteenturvallisuus virasto Trafi. <https://www.trafi.fi/filebank/a/1487308858/d74c4c6af2c6c5b7248675c49127d894/24217-Dieselpaastojen_tarkastus_2017.pdf>. Luettu 3.2.2018.

16. A Comparison of Emissions Reduction Technologies. 2015. Verkkoaineisto. REM Technology Inc. <https://www.spartancontrols.com/~/media/resources/rem%20technology%20inc/papers/rem%20emission%20reduction%20technology%20comparison%20june2015.pdf?la=en>. Luettu 21.2.2018.

17. Stralis XP brochure. Verkkoaineisto. Iveco. <https://www.iveco.com/uk/collections/catalogues/Documents/new-stralis/new-iveco-stralis-brochure.pdf>. Luettu 16.2.2018.

18. LNG Vehicles. Verkkoaineisto Iveco. <http://www.onthemosway.eu/wp-content/uploads/2015/06/TrainMoss-II-Madrid-26-11-2015.pdf>. Luettu 16.2.2018.

19. Stralis NP brochure. Verkkoaineisto. Iveco. <https://www.iveco.com/Common/Documents/Brochures/new_stralisNP.pdf>. Luettu 16.2.2018.

20. Cursor 9 NG. Verkkoaineisto FTP industrial. <http://www.fptindustrial.com/global/en/engines/on-road/trucks/cursor9-ng>. Luettu 16.2.2018.

21. Cursor 13. Verkkoaineisto. FTP industrial. <http://www.fptindustrial.com/global/en/engines/on-road/trucks/cursor13>. Luettu 16.2.2018.

22. Ultrafine condensation particle sizer brochure. Verkkoaineisto. TSI Inc. <http://tsi.com/uploadedFiles/Product_Information/Literature/Spec_Sheets/3776_2980345.pdf>. Luettu 20.2.2018.

23. Engine exhaust particle sizer brochure. Verkkoaineisto. TSI Inc. <http://www.tsi.com/uploadedFiles/_Site_Root/Products/Literature/Spec_Sheets/3090_2980244A.pdf>. Luettu 15.2.2018

24. Kalliokoski, Mikael. 2011. Diffuusiovarautumiseen perustuvan pienhiukkas-suodattimen kehitys. Diplomityö. Tampereen teknillinen yliopisto. TUT DPub -tietokanta.

25. Optical particle sizer brochure. Verkkoaineisto. TSI Inc. <http://www.tsi.com/uploadedFiles/_Site_Root/Products/Literature/Spec_Sheets/3330_5001323_Web.pdf>. Luettu 12.2.2018.

26. AP-360 Series. Käyttöohjekirja. Horiba.

27. LI 840 A, instruction manual. Verkkoaineisto. LI-COR biosciences. <https://www.licor.com/documents/y10gor2jal2p3t8ev4hm>. Luettu 12.2.2018.

28. IR Gas Filter Correlation Carbon Monoxide Analyzer bro-chure. Verkkoaineisto. Environnement S.A. <http://www.environnement.it/public/articoli/47/Files/CO12M_2014.pdf>. Luettu 12.2.2018.

29. UVAbsorption Ozone Analyzer - Model O342M. Verkkoaineisto. Environnement S.A. <http://www.environnement-sa.fr/wp-content/uploads/2012/09/3V_AQMS_2012_EN_s.pdf>. Luettu 12.2.2018.

30. Aethlometer AE33 user manual. Verkkoaineisto. Magee Scientific. <http://www.mageesci.com/EACworkshop2016/MANUALS/AE33/AE33_Users-Manual_Rev154.pdf>. Luettu 12.2.2018.

31. Procare drive install instructions. Asennusopas. Proventia.

32. Chang, Y.; Mendrea, B.; Sterniak, J. & Stanislav, V.B. 2016. Effect of Ambient Temperature and Humidity on Combustion and Emissions of a SparkAssisted Compression Ignition Engine. Verkkoaineisto. <http://gasturbinespower.asmedigitalcollection.asme.org/article.aspx?articleid=2569876> Luettu 22.2.2018.

33. Pirjola, Liisa. 2015. Physical and Chemical Characterization of Real-World Particle Number and Mass Emissions from City Buses in Finland. Verkkoaineisto. <http://pubs.acs.org/doi/abs/10.1021/acs.est.5b04105>. Luettu 24.2.2018.

Typenoksidien päästökerroinlaskut

Typenoksidit maantieosuus

In [2]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [3]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [4]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [5]:
```
#Sniffer
#Turuntie start - Kasvihuoneilmiö
tt_kh_west_snf_1 = df1.between_time('08:01:22','08:25:43')
tt_kh_west_snf_2 = df3.between_time('08:38:49','09:03:14')
tt_kh_west_snf_3 = df5.between_time('08:19:11','08:43:22')
tt_kh_west_snf_4 = df7.between_time('08:13:31','08:53:38')
#Kasvihuoneilmiö - Turuntie start
kh_tt_east_snf_1 = df1.between_time('12:11:46','12:35:56')
kh_tt_east_snf_2 = df3.between_time('13:03:00','13:26:37')
kh_tt_east_snf_3 = df5.between_time('12:48:23','13:15:03')
kh_tt_east_snf_4 = df7.between_time('12:53:35','13:17:18')
#Sniffer
#Kasvihuoneilmiö west turning point - Salo
khwt_salo_west_snf_1 = df1.between_time('09:13:23','09:36:40')
khwt_salo_west_snf_2 = df3.between_time('10:24:57','10:47:54')
khwt_salo_west_snf_3 = df5.between_time('10:09:44','10:32:34')
khwt_salo_west_snf_4 = df7.between_time('10:15:02','10:37:39')
#Salo - Kasvihuoneilmiö west turning point
khwt_salo_east_snf_1 = df1.between_time('11:17:43','11:40:51')
khwt_salo_east_snf_2 = df3.between_time('12:35:49','12:58:38')
khwt_salo_east_snf_3 = df5.between_time('12:21:14','12:44:00')
khwt_salo_east_snf_4 = df7.between_time('12:26:37','12:49:12')
#Sniffer
#Salo - Paimio
salo_pai_west_snf_1 = df1.between_time('09:47:39','10:02:21')
salo_pai_west_snf_2 = df3.between_time('10:58:38','11:13:03')
salo_pai_west_snf_3 = df5.between_time('10:42:45','10:57:08')
salo_pai_west_snf_4 = df7.between_time('10:47:20','11:01:30')
#Paimio - Salo
salo_pai_east_snf_1 = df1.between_time('11:01:51','11:07:32')
salo_pai_east_snf_2 = df3.between_time('12:19:52','12:25:29')
salo_pai_east_snf_3 = df5.between_time('12:05:02','12:10:49')
salo_pai_east_snf_4 = df7.between_time('12:11:07','12:16:48')
#Sniffer
#Highway background [Paimio - Salo]
hw_bg_snf_1 = df1.between_time('10:52:18','10:55:59')
hw_bg_snf_2 = df3.between_time('12:09:42','12:13:45')
hw_bg_snf_3 = df5.between_time('11:56:07','11:59:30')
hw_bg_snf_4 = df7.between_time('12:00:28','12:04:45')
```

In [6]:
```
#Sniffer
#All highway parts combined in driven order
hw_snf_1 = tt_kh_west_snf_1.append(khwt_salo_west_snf_1,ignore_index=True).append(salo_pai_west_snf_1,ignore_index=True).append(salo_pai
_east_snf_1,ignore_index=True).append(khwt_salo_east_snf_1,ignore_index=True).append(kh_tt_east_snf_1,ignore_index=True)
hw_snf_2 = tt_kh_west_snf_2.append(khwt_salo_west_snf_2,ignore_index=True).append(salo_pai_west_snf_2,ignore_index=True).append(salo_pai
_east_snf_2,ignore_index=True).append(khwt_salo_east_snf_2,ignore_index=True).append(kh_tt_east_snf_2,ignore_index=True)
hw_snf_3 = tt_kh_west_snf_3.append(khwt_salo_west_snf_3,ignore_index=True).append(salo_pai_west_snf_3,ignore_index=True).append(salo_pai
_east_snf_3,ignore_index=True).append(khwt_salo_east_snf_3,ignore_index=True).append(kh_tt_east_snf_3,ignore_index=True)
hw_snf_4 = tt_kh_west_snf_4.append(khwt_salo_west_snf_4,ignore_index=True).append(salo_pai_west_snf_4,ignore_index=True).append(salo_pai
_east_snf_4,ignore_index=True).append(khwt_salo_east_snf_4,ignore_index=True).append(kh_tt_east_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
hw_snf_1 = hw_snf_1[(hw_snf_1.Speed > 6.95)]
hw_snf_2 = hw_snf_2[(hw_snf_2.Speed > 6.95)]
hw_snf_3 = hw_snf_3[(hw_snf_3.Speed > 6.95)]
hw_snf_4 = hw_snf_4[(hw_snf_4.Speed > 6.95)]
```

In [7]:
```
#CO2
#Background for CO2
CO2_bg_1 = hw_bg_snf_1['CO2_ppm'].mean()
CO2_bg_2 = hw_bg_snf_2['CO2_ppm'].mean()
CO2_bg_3 = hw_bg_snf_3['CO2_ppm'].mean()
CO2_bg_4 = hw_bg_snf_4['CO2_ppm'].mean()

hw_snf_1['Ticks'] = range(0,len(hw_snf_1.index.values))
hw_snf_2['Ticks'] = range(0,len(hw_snf_2.index.values))
hw_snf_3['Ticks'] = range(0,len(hw_snf_3.index.values))
hw_snf_4['Ticks'] = range(0,len(hw_snf_4.index.values))
```

In [35]:
```
#Calculate corrected CO2 to dataframes
hw_snf_1['CO2_cor'] = hw_snf_1['CO2_ppm'] - CO2_bg_1
hw_snf_2['CO2_cor'] = hw_snf_2['CO2_ppm'] - CO2_bg_2
hw_snf_3['CO2_cor'] = hw_snf_3['CO2_ppm'] - CO2_bg_3
hw_snf_4['CO2_cor'] = hw_snf_4['CO2_ppm'] - CO2_bg_4
```

In [36]:
```
#NOX
#Background for NOx
NOx_bg_1 = hw_bg_snf_1['NOx'].mean()
NOx_bg_2 = hw_bg_snf_2['NOx'].mean()
NOx_bg_3 = hw_bg_snf_3['NOx'].mean()
NOx_bg_4 = hw_bg_snf_4['NOx'].mean()
#Calculate corrected NOx to dataframes
hw_snf_1['NOx_cor'] = hw_snf_1['NOx'] - NOx_bg_1
hw_snf_2['NOx_cor'] = hw_snf_2['NOx'] - NOx_bg_2
hw_snf_3['NOx_cor'] = hw_snf_3['NOx'] - NOx_bg_3
hw_snf_4['NOx_cor'] = hw_snf_4['NOx'] - NOx_bg_4
```

In [37]:
```
#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol
no_mol = 30.0061
no2_mol = 46.0055
co2_mol = 44.0095
```

In [42]:
```
#Molar mass calculation for NOx
#Remove mean() to get 1hz resolution

#Calculate corrected NO2 and NO concentration
day_1_no2 = (hw_snf_1['NO2'].mean()-hw_bg_snf_1['NO2'].mean())
day_2_no2 = (hw_snf_2['NO2'].mean()-hw_bg_snf_2['NO2'].mean())
day_3_no2 = (hw_snf_3['NO2'].mean()-hw_bg_snf_3['NO2'].mean())
day_4_no2 = (hw_snf_4['NO2'].mean()-hw_bg_snf_4['NO2'].mean())
#
day_1_no = (hw_snf_1['NO'].mean()-hw_bg_snf_1['NO'].mean())
day_2_no = (hw_snf_2['NO'].mean()-hw_bg_snf_2['NO'].mean())
day_3_no = (hw_snf_3['NO'].mean()-hw_bg_snf_3['NO'].mean())
day_4_no = (hw_snf_4['NO'].mean()-hw_bg_snf_4['NO'].mean())

#Calculate total ratio for NO+NO2
day1_tot_ratio = (day_1_no2+day_1_no)
day2_tot_ratio = (day_2_no2+day_2_no)
day3_tot_ratio = (day_3_no2+day_3_no)
day4_tot_ratio = (day_4_no2+day_4_no)

#Calculate NO2 daily ratio
day1_no2_ratio = (day_1_no2 / day1_tot_ratio)
day2_no2_ratio = (day_2_no2 / day2_tot_ratio)
day3_no2_ratio = (day_3_no2 / day3_tot_ratio)
day4_no2_ratio = (day_4_no2 / day4_tot_ratio)
#Calculate NO daily ratio
day1_no_ratio = (day_1_no / day1_tot_ratio)
day2_no_ratio = (day_2_no / day2_tot_ratio)
day3_no_ratio = (day_3_no / day3_tot_ratio)
day4_no_ratio = (day_4_no / day4_tot_ratio)

#Calculate molar concentration for each compound
#NO2
day_1_no2_mc = (day1_no2_ratio/no2_mol)
day_2_no2_mc = (day2_no2_ratio/no2_mol)
day_3_no2_mc = (day3_no2_ratio/no2_mol)
day_4_no2_mc = (day4_no2_ratio/no2_mol)
#NO
day_1_no_mc = (day1_no_ratio/no_mol)
day_2_no_mc = (day2_no_ratio/no_mol)
day_3_no_mc = (day3_no_ratio/no_mol)
day_4_no_mc = (day4_no_ratio/no_mol)

#Calculate total molar concentration
day_1_nox_mc = (day_1_no2_mc+day_1_no_mc)
day_2_nox_mc = (day_2_no2_mc+day_2_no_mc)
day_3_nox_mc = (day_3_no2_mc+day_3_no_mc)
day_4_nox_mc = (day_4_no2_mc+day_4_no_mc)

#Calculate ratio for compounds from total concentration
#NO2
day1_no2_t_ratio = (day_1_no2_mc / day_1_nox_mc)
day2_no2_t_ratio = (day_2_no2_mc / day_2_nox_mc)
day3_no2_t_ratio = (day_3_no2_mc / day_3_nox_mc)
day4_no2_t_ratio = (day_4_no2_mc / day_4_nox_mc)
#NO
day1_no_t_ratio = (day_1_no_mc / day_1_nox_mc)
day2_no_t_ratio = (day_2_no_mc / day_2_nox_mc)
day3_no_t_ratio = (day_3_no_mc / day_3_nox_mc)
day4_no_t_ratio = (day_4_no_mc / day_4_nox_mc)

#Calculate NOx molar mass
day1_nox_mol = ((day1_no2_t_ratio * no2_mol) + (day1_no_t_ratio*no_mol))
day2_nox_mol = ((day2_no2_t_ratio * no2_mol) + (day2_no_t_ratio*no_mol))
day3_nox_mol = ((day3_no2_t_ratio * no2_mol) + (day3_no_t_ratio*no_mol))
day4_nox_mol = ((day4_no2_t_ratio * no2_mol) + (day4_no_t_ratio*no_mol))


print('1:',day1_nox_mol, 'g/mol')
print('2:',day2_nox_mol, 'g/mol')
print('3:',day3_nox_mol, 'g/mol')
print('4:',day4_nox_mol, 'g/mol')
```

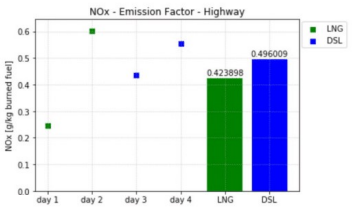
```
1: 31.351092872383425 g/mol
2: 30.54699145792 41 g/mol
3: 33.69247891467855 g/mol
4: 31.158155264517124 g/mol
```

In [39]:
```
#ERnox
ER_nox_1 = (hw_snf_1['NOx_cor'].mean()/1000)/hw_snf_1['CO2_cor'].mean()
ER_nox_2 = (hw_snf_2['NOx_cor'].mean()/1000)/hw_snf_2['CO2_cor'].mean()
ER_nox_3 = (hw_snf_3['NOx_cor'].mean()/1000)/hw_snf_3['CO2_cor'].mean()
ER_nox_4 = (hw_snf_4['NOx_cor'].mean()/1000)/hw_snf_4['CO2_cor'].mean()
```

In [40]:
```
#EFnox
#Calculate emission factor for NOx
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel

EF_nox_1 = (ER_nox_1 * (day1_nox_mol/co2_mol) * efng)
EF_nox_2 = (ER_nox_2 * (day2_nox_mol/co2_mol) * efng)
EF_nox_3 = (ER_nox_3 * (day3_nox_mol/co2_mol) * efdiesel)
EF_nox_4 = (ER_nox_4 * (day4_nox_mol/co2_mol) * efdiesel)

print(EF_nox_1, 'g/kg burned fuel')
print(EF_nox_2, 'g/kg burned fuel')
print(EF_nox_3, 'g/kg burned fuel')
print(EF_nox_4, 'g/kg burned fuel')

EF_nox_lng = (EF_nox_1+EF_nox_2)/2
EF_nox_dsl = (EF_nox_3+EF_nox_4)/2

print('lng: ',EF_nox_lng, 'g/kg burned fuel')
print('diesel: ',EF_nox_dsl, 'g/kg burned fuel')
```

```
0.24712979319015724 g/kg burned fuel
0.60066651415608l7 g/kg burned fuel
0.43651231900605636 g/kg burned fuel
0.5555062029449606 g/kg burned fuel
lng:  0.42389815367311g5 g/kg burned fuel
diesel:  0.4960093009755085 g/kg burned fuel
```

In [41]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_nox_1,EF_nox_2]
lng_c = EF_nox_lng
dsl = [EF_nox_3,EF_nox_4]
dsl_c = EF_nox_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('NOx [g/kg burned fuel]')
ax.set_title('NOx - Emission Factor - Highway')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%f' % float(height),
                ha='center', va='bottom')

autolabel(rects3)
autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)

plt.savefig('/home/sami/Documents/emissionfactor/hw_NOx_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Typenoksidit toisto-osuus

In [37]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [38]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [39]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [40]:
```
#Kasvihuoneilmiö
############################################################
#Sniffer day 1
#West
west1_snf_1 = df1.between_time('08:41:12','08:45:21') #1 to turku
west2_snf_1 = df1.between_time('08:52:18','08:56:28') #2 to turku
west3_snf_1 = df1.between_time('09:02:56','09:07:07') #3 to turku
west4_snf_1 = df1.between_time('11:47:39','11:51:48') #4 to turku
west5_snf_1 = df1.between_time('11:59:46','12:03:27') #5 to turku (background)
#Sinnefer day 1
#East
east1_snf_1 = df1.between_time('08:46:52','08:51:00') #1 to Helsinki
east2_snf_1 = df1.between_time('08:57:23','09:01:34') #2 to Helsinki
east3_snf_1 = df1.between_time('11:42:03','11:46:12') #3 to Helsinki
east4_snf_1 = df1.between_time('11:52:46','11:56:53') #4 to Helsinki
east5_snf_1 = df1.between_time('12:04:08','12:07:47') #5 to Helsinki (background)
############################################################
#Sniffer day 2
#West
west1_snf_2 = df3.between_time('09:17:12','09:21:21') #1 to turku
west2_snf_2 = df3.between_time('09:27:31','09:31:41') #2 to turku
west3_snf_2 = df3.between_time('09:37:48','09:41:56') #3 to turku
west4_snf_2 = df3.between_time('09:48:01','09:52:07') #4 to turku
west5_snf_2 = df3.between_time('09:58:11','10:02:21') #5 to turku
west6_snf_2 = df3.between_time('10:08:23','10:12:32') #6 to turku
west7_snf_2 = df3.between_time('10:18:46','10:22:54') #7 to turku
west8_snf_2 = df3.between_time('09:07:34','09:11:15') #8 to turku (background)
#Sinnefer day 2
#East
east1_snf_2 = df3.between_time('09:22:11','09:26:18') #1 to Helsinki
east2_snf_2 = df3.between_time('09:32:33','09:36:38') #2 to Helsinki
east3_snf_2 = df3.between_time('09:42:45','09:46:53') #3 to Helsinki
east4_snf_2 = df3.between_time('09:52:57','09:57:03') #4 to Helsinki
east5_snf_2 = df3.between_time('10:03:10','10:07:16') #5 to Helsinki
east6_snf_2 = df3.between_time('10:13:23','10:17:30') #6 to Helsinki
east7_snf_2 = df3.between_time('12:58:49','13:02:44') #7 to Helsinki
east8_snf_2 = df3.between_time('09:12:06','09:15:45') #8 to Helsinki (background)
############################################################
#Sniffer day 3
#West
west1_snf_3 = df5.between_time('08:59:37','09:03:45') #1 to turku
west2_snf_3 = df5.between_time('09:09:59','09:14:07') #2 to turku
west3_snf_3 = df5.between_time('09:20:42','09:24:49') #3 to turku
west4_snf_3 = df5.between_time('09:31:05','09:35:14') #4 to turku
west5_snf_3 = df5.between_time('09:44:48','09:48:57') #5 to turku
west6_snf_3 = df5.between_time('09:55:14','09:59:22') #6 to turku
west7_snf_3 = df5.between_time('10:05:26','10:09:34') #7 to turku
west8_snf_3 = df5.between_time('08:49:36','08:53:39') #8 to turku (background)
#Sinnefer day 3
#East
east1_snf_3 = df5.between_time('09:04:38','09:08:44')  #1 to Helsinki
east2_snf_3 = df5.between_time('09:15:10','09:19:17')  #2 to Helsinki
east3_snf_3 = df5.between_time('09:25:48','09:29:53')  #3 to Helsinki
east4_snf_3 = df5.between_time('09:39:29','09:43:36')  #4 to Helsinki
east5_snf_3 = df5.between_time('09:49:58','09:54:04')  #5 to Helsinki
east6_snf_3 = df5.between_time('10:00:10','10:04:14')  #6 to Helsinki
east7_snf_3 = df5.between_time('12:44:12','12:48:07')  #7 to Helsinki
east8_snf_3 = df5.between_time('08:54:24','08:58:05')  #8 to Helsinki (background)
############################################################
#Sniffer day 4
#West
west1_snf_4 = df7.between_time('09:08:36','09:12:42') #1 to turku
west2_snf_4 = df7.between_time('09:19:07','09:23:22') #2 to turku
west3_snf_4 = df7.between_time('09:29:30','09:33:38') #3 to turku
west4_snf_4 = df7.between_time('09:39:49','09:43:56') #4 to turku
west5_snf_4 = df7.between_time('09:49:57','09:54:03') #5 to turku
west6_snf_4 = df7.between_time('10:00:33','10:04:43') #6 to turku
west7_snf_4 = df7.between_time('10:10:48','10:14:51') #7 to turku
west8_snf_4 = df7.between_time('08:53:56','08:57:39') #8 to turku (background)
#Sinnefer day 4
#East
east1_snf_4 = df7.between_time('09:13:49','09:17:57') #1 to Helsinki
east2_snf_4 = df7.between_time('09:24:12','09:28:18') #2 to Helsinki
east3_snf_4 = df7.between_time('09:34:36','09:38:42') #3 to Helsinki
east4_snf_4 = df7.between_time('09:44:45','09:48:47') #4 to Helsinki
east5_snf_4 = df7.between_time('09:55:19','09:59:23') #5 to Helsinki
east6_snf_4 = df7.between_time('10:05:33','10:09:37') #6 to Helsinki
east7_snf_4 = df7.between_time('12:49:24','12:53:19') #7 to Helsinki
east8_snf_4 = df7.between_time('08:58:23','09:02:31') #8 to Helsinki (background)
############################################################
```

In [41]:
```
#Sniffer
#East
kh_snf_1_east = east2_snf_1.append(east3_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2_east = east1_snf_2.append(east3_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3_east = east2_snf_3.append(east5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4_east = east2_snf_4.append(east3_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)
#West
kh_snf_1_west = west1_snf_1.append(west3_snf_1,ignore_index=True).append(west4_snf_1,ignore_index=True)
kh_snf_2_west = west1_snf_2.append(west2_snf_2,ignore_index=True).append(west4_snf_2,ignore_index=True)
kh_snf_3_west = west2_snf_3.append(west4_snf_3,ignore_index=True).append(west5_snf_3,ignore_index=True)
kh_snf_4_west = west4_snf_4.append(west5_snf_4,ignore_index=True).append(west6_snf_4,ignore_index=True)

kh_snf_1 = west1_snf_1.append(east2_snf_1,ignore_index=True).append(west3_snf_1,ignore_index=True).append(east3_snf_1,ignore_index=True)
.append(west4_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2 = west1_snf_2.append(east1_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True)
.append(west4_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3 = west2_snf_3.append(east2_snf_1,ignore_index=True).append(west4_snf_3,ignore_index=True).append(east5_snf_3,ignore_index=True)
.append(west5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4 = west4_snf_4.append(east2_snf_4,ignore_index=True).append(west5_snf_4,ignore_index=True).append(east3_snf_4,ignore_index=True)
.append(west6_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)

kh_bg_snf_1 = west5_snf_1.append(east5_snf_1,ignore_index=True)
kh_bg_snf_2 = west8_snf_2.append(east8_snf_2,ignore_index=True)
kh_bg_snf_3 = west8_snf_3.append(east8_snf_3,ignore_index=True)
kh_bg_snf_4 = west8_snf_4.append(east8_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
kh_snf_1 = kh_snf_1[(kh_snf_1.Speed > 6.95)]
kh_snf_2 = kh_snf_2[(kh_snf_2.Speed > 6.95)]
kh_snf_3 = kh_snf_3[(kh_snf_3.Speed > 6.95)]
kh_snf_4 = kh_snf_4[(kh_snf_4.Speed > 6.95)]

kh_snf_1_east['Ticks'] = range(0,len(kh_snf_1_east.index.values))
kh_snf_2_east['Ticks'] = range(0,len(kh_snf_2_east.index.values))
kh_snf_3_east['Ticks'] = range(0,len(kh_snf_3_east.index.values))
kh_snf_4_east['Ticks'] = range(0,len(kh_snf_4_east.index.values))

kh_snf_1_west['Ticks'] = range(0,len(kh_snf_1_west.index.values))
kh_snf_2_west['Ticks'] = range(0,len(kh_snf_2_west.index.values))
kh_snf_3_west['Ticks'] = range(0,len(kh_snf_3_west.index.values))
kh_snf_4_west['Ticks'] = range(0,len(kh_snf_4_west.index.values))

kh_snf_1['Ticks'] = range(0,len(kh_snf_1.index.values))
kh_snf_2['Ticks'] = range(0,len(kh_snf_2.index.values))
kh_snf_3['Ticks'] = range(0,len(kh_snf_3.index.values))
kh_snf_4['Ticks'] = range(0,len(kh_snf_4.index.values))

hw_bg_snf_2 = df3.between_time('12:09:42','12:13:00')
hw_bg_snf_2 = hw_bg_snf_2[(hw_bg_snf_2.Speed >= 1)]
hw_bg_snf_2['Ticks'] = range(0,len(hw_bg_snf_2.index.values))
```

In [42]:
```
#CO2
#Calculate total mean for backgound & print values
day_1_co2_tot_bg = ((east5_snf_1['CO2_ppm'].reset_index(drop=True) + west5_snf_1['CO2_ppm'].reset_index(drop=True)) / 2)
day_2_co2_tot_bg = hw_bg_snf_2['CO2_ppm'].mean()
day_3_co2_tot_bg = ((east8_snf_3['CO2_ppm'].reset_index(drop=True) + west8_snf_3['CO2_ppm'].reset_index(drop=True)) / 2)
day_4_co2_tot_bg = ((east8_snf_4['CO2_ppm'].reset_index(drop=True) + west8_snf_4['CO2_ppm'].reset_index(drop=True)) / 2)

#Calculate corrected CO2 to dataframes
#East
kh_snf_1_east['CO2_cor'] = kh_snf_1_east['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_east['CO2_cor'] = kh_snf_2_east['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_east['CO2_cor'] = kh_snf_3_east['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_east['CO2_cor'] = kh_snf_4_east['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
#West
kh_snf_1_west['CO2_cor'] = kh_snf_1_west['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_west['CO2_cor'] = kh_snf_2_west['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_west['CO2_cor'] = kh_snf_3_west['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_west['CO2_cor'] = kh_snf_4_west['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()

#Day
kh_snf_1['CO2_cor'] = kh_snf_1['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2['CO2_cor'] = kh_snf_2['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3['CO2_cor'] = kh_snf_3['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4['CO2_cor'] = kh_snf_4['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
```

In [44]:
```
#NOX
```

In [98]:
```
#Calculate total mean for backgound & print values
#day_1_nox_tot_bg = ((east5_snf_1['NOx'].reset_index(drop=True) + west5_snf_1['NOx'].reset_index(drop=True)) / 2)
#day_2_nox_tot_bg = hw_bg_snf_2['NOx']
#day_3_nox_tot_bg = ((east8_snf_3['NOx'].reset_index(drop=True) + west8_snf_3['NOx'].reset_index(drop=True)) / 2)
#day_4_nox_tot_bg = ((east8_snf_4['NOx'].reset_index(drop=True) + west8_snf_4['NOx'].reset_index(drop=True)) / 2)

hw_bg_snf_2 = hw_bg_snf_2[(hw_bg_snf_2['NOx'] < 30)]
kh_bg_snf_1 = kh_bg_snf_1[(kh_bg_snf_1['NOx'] < 30)]
kh_bg_snf_3 = kh_bg_snf_3[(kh_bg_snf_3['NOx'] < 30)]
kh_bg_snf_4 = kh_bg_snf_4[(kh_bg_snf_4['NOx'] < 30)]

kh_snf_1 = kh_snf_1[(kh_snf_1['NOx'] < 100)]
kh_snf_2 = kh_snf_2[(kh_snf_2['NOx'] < 100)]
kh_snf_3 = kh_snf_3[(kh_snf_3['NOx'] < 100)]
kh_snf_4 = kh_snf_4[(kh_snf_4['NOx'] < 100)]
```

In [99]:
```
kh_snf_1['NOx_cor'] = kh_snf_1['NOx'].reset_index(drop=True) - kh_bg_snf_1['NOx'].mean()
kh_snf_2['NOx_cor'] = kh_snf_2['NOx'].reset_index(drop=True) - hw_bg_snf_2['NOx'].mean()
kh_snf_3['NOx_cor'] = kh_snf_3['NOx'].reset_index(drop=True) - kh_bg_snf_3['NOx'].mean()
kh_snf_4['NOx_cor'] = kh_snf_4['NOx'].reset_index(drop=True) - kh_bg_snf_4['NOx'].mean()
```

In [100]:
```
#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol
no_mol = 30.0061
no2_mol = 46.0055
co2_mol = 44.0095
```

In [101]:
```
#Molar mass calculation for NOx
#Remove mean() to get 1hz resolution

#Calculate corrected NO2 and NO concentration
day_1_no2 = (kh_snf_1['NO2'].mean()-kh_bg_snf_1['NO2'].mean())
day_2_no2 = (kh_snf_2['NO2'].mean()-hw_bg_snf_2['NO2'].mean())
day_3_no2 = (kh_snf_3['NO2'].mean()-kh_bg_snf_3['NO2'].mean())
day_4_no2 = (kh_snf_4['NO2'].mean()-kh_bg_snf_4['NO2'].mean())
#
day_1_no = (kh_snf_1['NO'].mean()-kh_bg_snf_1['NO'].mean())
day_2_no = (kh_snf_2['NO'].mean()-hw_bg_snf_2['NO'].mean())
day_3_no = (kh_snf_3['NO'].mean()-kh_bg_snf_3['NO'].mean())
day_4_no = (kh_snf_4['NO'].mean()-kh_bg_snf_4['NO'].mean())

#Calculate total ratio for NO+NO2
day1_tot_ratio = (day_1_no2+day_1_no)
day2_tot_ratio = (day_2_no2+day_2_no)
day3_tot_ratio = (day_3_no2+day_3_no)
day4_tot_ratio = (day_4_no2+day_4_no)

#Calculate NO2 daily ratio
day1_no2_ratio = (day_1_no2 / day1_tot_ratio)
day2_no2_ratio = (day_2_no2 / day2_tot_ratio)
day3_no2_ratio = (day_3_no2 / day3_tot_ratio)
day4_no2_ratio = (day_4_no2 / day4_tot_ratio)
#Calculate NO daily ratio
day1_no_ratio = (day_1_no / day1_tot_ratio)
day2_no_ratio = (day_2_no / day2_tot_ratio)
day3_no_ratio = (day_3_no / day3_tot_ratio)
day4_no_ratio = (day_4_no / day4_tot_ratio)

#Calculate molar concentration for each compound
#NO2
day_1_no2_mc = (day1_no2_ratio/no2_mol)
day_2_no2_mc = (day2_no2_ratio/no2_mol)
day_3_no2_mc = (day3_no2_ratio/no2_mol)
day_4_no2_mc = (day4_no2_ratio/no2_mol)
#NO
day_1_no_mc = (day1_no_ratio/no_mol)
day_2_no_mc = (day2_no_ratio/no_mol)
day_3_no_mc = (day3_no_ratio/no_mol)
day_4_no_mc = (day4_no_ratio/no_mol)

#Calculate total molar concentration
day_1_nox_mc = (day_1_no2_mc+day_1_no_mc)
day_2_nox_mc = (day_2_no2_mc+day_2_no_mc)
day_3_nox_mc = (day_3_no2_mc+day_3_no_mc)
day_4_nox_mc = (day_4_no2_mc+day_4_no_mc)

#Calculate ratio for compounds from total concentration
#NO2
day1_no2_t_ratio = (day_1_no2_mc / day_1_nox_mc)
day2_no2_t_ratio = (day_2_no2_mc / day_2_nox_mc)
day3_no2_t_ratio = (day_3_no2_mc / day_3_nox_mc)
day4_no2_t_ratio = (day_4_no2_mc / day_4_nox_mc)
#NO
day1_no_t_ratio = (day_1_no_mc / day_1_nox_mc)
day2_no_t_ratio = (day_2_no_mc / day_2_nox_mc)
day3_no_t_ratio = (day_3_no_mc / day_3_nox_mc)
day4_no_t_ratio = (day_4_no_mc / day_4_nox_mc)

#Calculate NOx molar mass
day1_nox_mol = ((day1_no2_t_ratio * no2_mol) + (day1_no_t_ratio*no_mol))
day2_nox_mol = ((day2_no2_t_ratio * no2_mol) + (day2_no_t_ratio*no_mol))
day3_nox_mol = ((day3_no2_t_ratio * no2_mol) + (day3_no_t_ratio*no_mol))
day4_nox_mol = ((day4_no2_t_ratio * no2_mol) + (day4_no_t_ratio*no_mol))


print(day1_nox_mol, 'g/mol')
print(day2_nox_mol, 'g/mol')
print(day3_nox_mol, 'g/mol')
print(day4_nox_mol, 'g/mol')
```

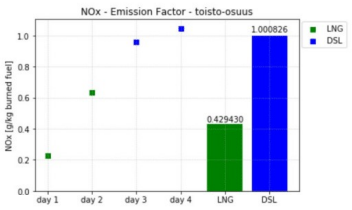
```
30.510556419073506 g/mol
32.521603908726036 g/mol
31.680419246984023 g/mol
34.23676352509603 g/mol
```

In [102]:
```
#ERnox
ER_nox_1 = (kh_snf_1['NOx_cor'].mean()/1000)/kh_snf_1['CO2_cor'].mean()
ER_nox_2 = (kh_snf_2['NOx_cor'].mean()/1000)/kh_snf_2['CO2_cor'].mean()
ER_nox_3 = (kh_snf_3['NOx_cor'].mean()/1000)/kh_snf_3['CO2_cor'].mean()
ER_nox_4 = (kh_snf_4['NOx_cor'].mean()/1000)/kh_snf_4['CO2_cor'].mean()
```

In [103]:
```
#EFnox
#Calculate emission factor for NOx
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel

EF_nox_1 = (ER_nox_1 * (day1_nox_mol/co2_mol) * efng)
EF_nox_2 = (ER_nox_2 * (day2_nox_mol/co2_mol) * efng)
EF_nox_3 = (ER_nox_3 * (day3_nox_mol/co2_mol) * efdiesel)
EF_nox_4 = (ER_nox_4 * (day4_nox_mol/co2_mol) * efdiesel)

print(EF_nox_1, 'g/kg burned fuel')
print(EF_nox_2, 'g/kg burned fuel')
print(EF_nox_3, 'g/kg burned fuel')
print(EF_nox_4, 'g/kg burned fuel')

EF_nox_lng = (EF_nox_1+EF_nox_2)/2
EF_nox_dsl = (EF_nox_3+EF_nox_4)/2

print('lng: ',EF_nox_lng, 'g/kg burned fuel')
print('diesel: ',EF_nox_dsl, 'g/kg burned fuel')
```

```
0.22544222878762812 g/kg burned fuel
0.6334168865959383 g/kg burned fuel
0.9571326725612154 g/kg burned fuel
1.0445190060150010 g/kg burned fuel
lng:  0.4294295576917832 g/kg burned fuel
diesel:  1.0008262392881486 g/kg burned fuel
```

In [104]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_nox_1,EF_nox_2]
lng_c = EF_nox_lng
dsl = [EF_nox_3,EF_nox_4]
dsl_c = EF_nox_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('NOx [g/kg burned fuel]')
ax.set_title('NOx - Emission Factor - toisto-osuus')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')


def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%f' % float(height),
                ha='center', va='bottom')

autolabel(rects3)
autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)

plt.savefig('/home/sami/Documents/emissionfactor/kh_NOx_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Mustan hiilen päästökerroinlaskut

Musta hiili maantieosuus

In [4]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [5]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [6]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [7]:
```
#Sniffer
#Turuntie start - Kasvihuoneilmiö
tt_kh_west_snf_1 = df1.between_time('08:01:22','08:25:43')
tt_kh_west_snf_2 = df3.between_time('08:38:49','09:03:14')
tt_kh_west_snf_3 = df5.between_time('08:19:11','08:43:22')
tt_kh_west_snf_4 = df7.between_time('08:13:31','08:53:38')
#Kasvihuoneilmiö - Turuntie start
kh_tt_east_snf_1 = df1.between_time('12:11:46','12:35:56')
kh_tt_east_snf_2 = df3.between_time('13:03:00','13:26:37')
kh_tt_east_snf_3 = df5.between_time('12:48:23','13:15:03')
kh_tt_east_snf_4 = df7.between_time('12:53:35','13:17:18')
#Sniffer
#Kasvihuoneilmiö west turning point - Salo
khwt_salo_west_snf_1 = df1.between_time('09:13:23','09:36:40')
khwt_salo_west_snf_2 = df3.between_time('10:24:57','10:47:54')
khwt_salo_west_snf_3 = df5.between_time('10:09:44','10:32:34')
khwt_salo_west_snf_4 = df7.between_time('10:15:02','10:37:39')
#Salo - Kasvihuoneilmiö west turning point
khwt_salo_east_snf_1 = df1.between_time('11:17:43','11:40:51')
khwt_salo_east_snf_2 = df3.between_time('12:35:49','12:58:38')
khwt_salo_east_snf_3 = df5.between_time('12:21:14','12:44:00')
khwt_salo_east_snf_4 = df7.between_time('12:26:37','12:49:12')
#Sniffer
#Salo - Paimio
salo_pai_west_snf_1 = df1.between_time('09:47:39','10:02:21')
salo_pai_west_snf_2 = df3.between_time('10:58:38','11:13:03')
salo_pai_west_snf_3 = df5.between_time('10:42:45','10:57:08')
salo_pai_west_snf_4 = df7.between_time('10:47:20','11:01:30')
#Paimio - Salo
salo_pai_east_snf_1 = df1.between_time('11:01:51','11:07:32')
salo_pai_east_snf_2 = df3.between_time('12:19:52','12:25:29')
salo_pai_east_snf_3 = df5.between_time('12:05:02','12:10:49')
salo_pai_east_snf_4 = df7.between_time('12:11:07','12:16:48')
#Sniffer
#Highway background [Paimio - Salo]
hw_bg_snf_1 = df1.between_time('10:52:18','10:55:59')
hw_bg_snf_2 = df3.between_time('12:09:42','12:13:45')
hw_bg_snf_3 = df5.between_time('11:56:07','11:59:30')
hw_bg_snf_4 = df7.between_time('12:00:28','12:04:45')
```

In [8]:
```
#Sniffer
#All highway parts combined in driven order
hw_snf_1 = tt_kh_west_snf_1.append(khwt_salo_west_snf_1,ignore_index=True).append(salo_pai_west_snf_1,ignore_index=True).append(salo_pai
_east_snf_1,ignore_index=True).append(khwt_salo_east_snf_1,ignore_index=True).append(kh_tt_east_snf_1,ignore_index=True)
hw_snf_2 = tt_kh_west_snf_2.append(khwt_salo_west_snf_2,ignore_index=True).append(salo_pai_west_snf_2,ignore_index=True).append(salo_pai
_east_snf_2,ignore_index=True).append(khwt_salo_east_snf_2,ignore_index=True).append(kh_tt_east_snf_2,ignore_index=True)
hw_snf_3 = tt_kh_west_snf_3.append(khwt_salo_west_snf_3,ignore_index=True).append(salo_pai_west_snf_3,ignore_index=True).append(salo_pai
_east_snf_3,ignore_index=True).append(khwt_salo_east_snf_3,ignore_index=True).append(kh_tt_east_snf_3,ignore_index=True)
hw_snf_4 = tt_kh_west_snf_4.append(khwt_salo_west_snf_4,ignore_index=True).append(salo_pai_west_snf_4,ignore_index=True).append(salo_pai
_east_snf_4,ignore_index=True).append(khwt_salo_east_snf_4,ignore_index=True).append(kh_tt_east_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
hw_snf_1 = hw_snf_1[(hw_snf_1.Speed > 6.95)]
hw_snf_2 = hw_snf_2[(hw_snf_2.Speed > 6.95)]
hw_snf_3 = hw_snf_3[(hw_snf_3.Speed > 6.95)]
hw_snf_4 = hw_snf_4[(hw_snf_4.Speed > 6.95)]
```

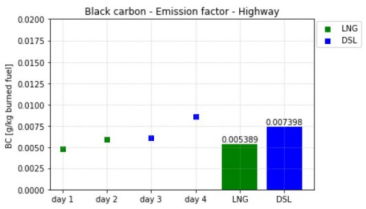
In [27]:
```
#CO2
#Background for CO2
CO2_bg_1 = hw_bg_snf_1['CO2_ppm'].mean()
CO2_bg_2 = hw_bg_snf_2['CO2_ppm'].mean()
CO2_bg_3 = hw_bg_snf_3['CO2_ppm'].mean()
CO2_bg_4 = hw_bg_snf_4['CO2_ppm'].mean()

hw_snf_1['Ticks'] = range(0,len(hw_snf_1.index.values))
hw_snf_2['Ticks'] = range(0,len(hw_snf_2.index.values))
hw_snf_3['Ticks'] = range(0,len(hw_snf_3.index.values))
hw_snf_4['Ticks'] = range(0,len(hw_snf_4.index.values))
```

In [31]:
```
#Calculate corrected CO2 to dataframes
hw_snf_1['CO2_cor'] = hw_snf_1['CO2_ppm'] - CO2_bg_1
hw_snf_2['CO2_cor'] = hw_snf_2['CO2_ppm'] - CO2_bg_2
hw_snf_3['CO2_cor'] = hw_snf_3['CO2_ppm'] - CO2_bg_3
hw_snf_4['CO2_cor'] = hw_snf_4['CO2_ppm'] - CO2_bg_4
#BC
hw_snf_2 = hw_snf_2[(hw_snf_2['BC6'] < 6000)]
hw_snf_1['BC_cor'] = hw_snf_1['BC6'] - hw_bg_snf_1['BC6'].mean()
hw_snf_2['BC_cor'] = hw_snf_2['BC6'] - hw_bg_snf_2['BC6'].mean()
hw_snf_3['BC_cor'] = hw_snf_3['BC6'] - hw_bg_snf_3['BC6'].mean()
hw_snf_4['BC_cor'] = hw_snf_4['BC6'] - hw_bg_snf_4['BC6'].mean()
```

In [32]:
```
#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + hw_snf_1['OutTemp'].mean()
T2 = K + hw_snf_2['OutTemp'].mean()
T3 = K + hw_snf_3['OutTemp'].mean()
T4 = K + hw_snf_4['OutTemp'].mean()

#Print Delta CO2
print('Delta CO2')
print('1 CO2:',hw_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',hw_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta BC
print('Delta BC')
print('1 CO2:',hw_snf_1['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('2 CO2:',hw_snf_2['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('3 CO2:',hw_snf_3['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('3 CO2:',hw_snf_4['BC_cor'].mean()*10**-9 ,'ng/m^3')


#Emission ratio for BC.
ER_BC_1 = ((hw_snf_1['BC_cor'].mean()*10**-9)/hw_snf_1['CO2_cor'].mean())
ER_BC_2 = ((hw_snf_2['BC_cor'].mean()*10**-9)/hw_snf_2['CO2_cor'].mean())
ER_BC_3 = ((hw_snf_3['BC_cor'].mean()*10**-9)/hw_snf_3['CO2_cor'].mean())
ER_BC_4 = ((hw_snf_4['BC_cor'].mean()*10**-9)/hw_snf_4['CO2_cor'].mean())


#print emission ratio
print('BC Emission ratio: [delta_ERbc/delta_CO2]')
print('Day1', ER_BC_1)
print('Day2', ER_BC_2)
print('Day3', ER_BC_3)
print('Day4', ER_BC_4)

#Emission factor

#Calculate emission factor for NOx
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel


EF_BC_1 = (ER_BC_1 * (r*T1)/(p*co2_mol) * efng) *10**6
EF_BC_2 = (ER_BC_2 * (r*T2)/(p*co2_mol) * efng) *10**6
EF_BC_3 = (ER_BC_3 * (r*T3)/(p*co2_mol) * efdiesel) *10**6
EF_BC_4 = (ER_BC_4 * (r*T4)/(p*co2_mol) * efdiesel) *10**6

#Calculate BC emission factor (mean) for each type of fuel
EF_BC_lng = ((EF_BC_1+EF_BC_2)/2)
EF_BC_dsl = ((EF_BC_3+EF_BC_4)/2)



#print BC emission factor
print('BC Emission Factor: [ER_bc*R*T/p*CO2mol*efCO2*10^6]')
print('Day1', EF_BC_1, 'g/kg burned fuel')
print('Day2', EF_BC_2, 'g/kg burned fuel')
print('Day3', EF_BC_3, 'g/kg burned fuel')
print('Day4', EF_BC_4, 'g/kg burned fuel')

#print BC emission factor for lng & dsl
print('BC Emission Factor for each fuel:')
print('natural gas:', EF_BC_lng, 'g/kg burned fuel')
print('diesel:', EF_BC_dsl, 'g/kg burned fuel')
```

```
Delta CO2
1 CO2: 53.33440309655569 ppm
2 CO2: 48.97967564830652 ppm
3 CO2: 77.32543134518379 ppm
3 CO2: 62.578998660720345 ppm
Delta BC
1 CO2: 1.9505327944400 64e-07 ng/m^3
2 CO2: 2.1867017802937 07e-07 ng/m^3
3 CO2: 2.9064933679653665e-07 ng/m^3
3 CO2: 3.3223603495195654c-07 ng/m^3
BC Emission ratio: [delta_ERbc/delta_CO2]
Day1 3.6571756337252953e-09
Day2 4.4645084953095495e-09
Day3 3.758780672028411e-09
Day4 5.309066013555356e-09
BC Emission Factor: [ER_bc*R*T/p*CO2mol*efCO2*10^6]
Day1 0.004847075780994811 g/kg burned fuel
Day2 0.005930538250247749 g/kg burned fuel
Day3 0.006182952529651086 g/kg burned fuel
Day4 0.008613604938717127 g/kg burned fuel
BC Emission Factor for each fuel:
natural gas: 0.005388807015621279 g/kg burned fuel
diesel: 0.0073982787341841 07 g/kg burned fuel
```

In [33]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_BC_1,EF_BC_2]
lng_c = EF_BC_lng
dsl = [EF_BC_3,EF_BC_4]
dsl_c = EF_BC_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('BC [g/kg burned fuel]')
ax.set_title('Black carbon - Emission factor - Highway')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%f' % float(height),
                ha='center', va='bottom')

autolabel(rects3)
autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
plt.ylim(0,0.02)

plt.savefig('/home/sami/Documents/emissionfactor/hw_BC_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Musta hiili toisto-osuus

In [11]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [12]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [13]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [14]:

```
#Kasvihuoneilmiö
############################################################
#Sniffer day 1
#West
west1_snf_1 = df1.between_time('08:41:12','08:45:21') #1 to turku
west2_snf_1 = df1.between_time('08:52:18','08:56:28') #2 to turku
west3_snf_1 = df1.between_time('09:02:56','09:07:07') #3 to turku
west4_snf_1 = df1.between_time('11:47:39','11:51:48') #4 to turku
west5_snf_1 = df1.between_time('11:59:46','12:03:27') #5 to turku (background)
#Sinnefer day 1
#East
east1_snf_1 = df1.between_time('08:46:52','08:51:00') #1 to Helsinki
east2_snf_1 = df1.between_time('08:57:23','09:01:34') #2 to Helsinki
east3_snf_1 = df1.between_time('11:42:03','11:46:12') #3 to Helsinki
east4_snf_1 = df1.between_time('11:52:46','11:56:53') #4 to Helsinki
east5_snf_1 = df1.between_time('12:04:08','12:07:47') #5 to Helsinki (background)
############################################################
#Sniffer day 2
#West
west1_snf_2 = df3.between_time('09:17:12','09:21:21') #1 to turku
west2_snf_2 = df3.between_time('09:27:31','09:31:41') #2 to turku
west3_snf_2 = df3.between_time('09:37:48','09:41:56') #3 to turku
west4_snf_2 = df3.between_time('09:48:01','09:52:07') #4 to turku
west5_snf_2 = df3.between_time('09:58:11','10:02:21') #5 to turku
west6_snf_2 = df3.between_time('10:08:23','10:12:32') #6 to turku
west7_snf_2 = df3.between_time('10:18:46','10:22:54') #7 to turku
west8_snf_2 = df3.between_time('09:07:34','09:11:15') #8 to turku (background)
#Sinnefer day 2
#East
east1_snf_2 = df3.between_time('09:22:11','09:26:18') #1 to Helsinki
east2_snf_2 = df3.between_time('09:32:33','09:36:38') #2 to Helsinki
east3_snf_2 = df3.between_time('09:42:45','09:46:53') #3 to Helsinki
east4_snf_2 = df3.between_time('09:52:57','09:57:03') #4 to Helsinki
east5_snf_2 = df3.between_time('10:03:10','10:07:16') #5 to Helsinki
east6_snf_2 = df3.between_time('10:13:23','10:17:30') #6 to Helsinki
east7_snf_2 = df3.between_time('12:58:49','13:02:44') #7 to Helsinki
east8_snf_2 = df3.between_time('09:12:06','09:15:45') #8 to Helsinki (background)
############################################################
#Sniffer day 3
#West
west1_snf_3 = df5.between_time('08:59:37','09:03:45') #1 to turku
west2_snf_3 = df5.between_time('09:09:59','09:14:07') #2 to turku
west3_snf_3 = df5.between_time('09:20:42','09:24:49') #3 to turku
west4_snf_3 = df5.between_time('09:31:05','09:35:14') #4 to turku
west5_snf_3 = df5.between_time('09:44:48','09:48:57') #5 to turku
west6_snf_3 = df5.between_time('09:55:14','09:59:22') #6 to turku
west7_snf_3 = df5.between_time('10:05:26','10:09:34') #7 to turku
west8_snf_3 = df5.between_time('08:49:36','08:53:39') #8 to turku (background)
#Sinnefer day 3
#East
east1_snf_3 = df5.between_time('09:04:38','09:08:44')  #1 to Helsinki
east2_snf_3 = df5.between_time('09:15:10','09:19:17')  #2 to Helsinki
east3_snf_3 = df5.between_time('09:25:48','09:29:53')  #3 to Helsinki
east4_snf_3 = df5.between_time('09:39:29','09:43:36')  #4 to Helsinki
east5_snf_3 = df5.between_time('09:49:58','09:54:04')  #5 to Helsinki
east6_snf_3 = df5.between_time('10:00:10','10:04:14')  #6 to Helsinki
east7_snf_3 = df5.between_time('12:44:12','12:48:07')  #7 to Helsinki
east8_snf_3 = df5.between_time('08:54:24','08:58:05')  #8 to Helsinki (background)
############################################################
#Sniffer day 4
#West
west1_snf_4 = df7.between_time('09:08:36','09:12:42') #1 to turku
west2_snf_4 = df7.between_time('09:19:07','09:23:22') #2 to turku
west3_snf_4 = df7.between_time('09:29:30','09:33:38') #3 to turku
west4_snf_4 = df7.between_time('09:39:49','09:43:56') #4 to turku
west5_snf_4 = df7.between_time('09:49:57','09:54:03') #5 to turku
west6_snf_4 = df7.between_time('10:00:33','10:04:43') #6 to turku
west7_snf_4 = df7.between_time('10:10:48','10:14:51') #7 to turku
west8_snf_4 = df7.between_time('08:53:56','08:57:39') #8 to turku (background)
#Sinnefer day 4
#East
east1_snf_4 = df7.between_time('09:13:49','09:17:57') #1 to Helsinki
east2_snf_4 = df7.between_time('09:24:12','09:28:18') #2 to Helsinki
east3_snf_4 = df7.between_time('09:34:36','09:38:42') #3 to Helsinki
east4_snf_4 = df7.between_time('09:44:45','09:48:47') #4 to Helsinki
east5_snf_4 = df7.between_time('09:55:19','09:59:23') #5 to Helsinki
east6_snf_4 = df7.between_time('10:05:33','10:09:37') #6 to Helsinki
east7_snf_4 = df7.between_time('12:49:24','12:53:19') #7 to Helsinki
east8_snf_4 = df7.between_time('08:58:23','09:02:31') #8 to Helsinki (background)
############################################################
```

In [15]:
```
#Sniffer
#East
kh_snf_1_east = east2_snf_1.append(east3_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2_east = east1_snf_2.append(east3_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3_east = east2_snf_3.append(east5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4_east = east2_snf_4.append(east3_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)
#West
kh_snf_1_west = west1_snf_1.append(west3_snf_1,ignore_index=True).append(west4_snf_1,ignore_index=True)
kh_snf_2_west = west1_snf_2.append(west2_snf_2,ignore_index=True).append(west4_snf_2,ignore_index=True)
kh_snf_3_west = west2_snf_3.append(west4_snf_3,ignore_index=True).append(west5_snf_3,ignore_index=True)
kh_snf_4_west = west4_snf_4.append(west5_snf_4,ignore_index=True).append(west6_snf_4,ignore_index=True)

kh_snf_1 = west1_snf_1.append(east2_snf_1,ignore_index=True).append(west3_snf_1,ignore_index=True).append(east3_snf_1,ignore_index=True)
.append(west4_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2 = west1_snf_2.append(east1_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True)
.append(west4_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3 = west2_snf_3.append(east2_snf_1,ignore_index=True).append(west4_snf_3,ignore_index=True).append(east5_snf_3,ignore_index=True)
.append(west5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4 = west4_snf_4.append(east2_snf_4,ignore_index=True).append(west5_snf_4,ignore_index=True).append(east3_snf_4,ignore_index=True)
.append(west6_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)

kh_bg_snf_1 = west5_snf_1.append(east5_snf_1,ignore_index=True)
kh_bg_snf_2 = west8_snf_2.append(east8_snf_2,ignore_index=True)
kh_bg_snf_3 = west8_snf_3.append(east8_snf_3,ignore_index=True)
kh_bg_snf_4 = west8_snf_4.append(east8_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
kh_snf_1 = kh_snf_1[(kh_snf_1.Speed > 6.95)]
kh_snf_2 = kh_snf_2[(kh_snf_2.Speed > 6.95)]
kh_snf_3 = kh_snf_3[(kh_snf_3.Speed > 6.95)]
kh_snf_4 = kh_snf_4[(kh_snf_4.Speed > 6.95)]

kh_snf_1_east['Ticks'] = range(0,len(kh_snf_1_east.index.values))
kh_snf_2_east['Ticks'] = range(0,len(kh_snf_2_east.index.values))
kh_snf_3_east['Ticks'] = range(0,len(kh_snf_3_east.index.values))
kh_snf_4_east['Ticks'] = range(0,len(kh_snf_4_east.index.values))

kh_snf_1_west['Ticks'] = range(0,len(kh_snf_1_west.index.values))
kh_snf_2_west['Ticks'] = range(0,len(kh_snf_2_west.index.values))
kh_snf_3_west['Ticks'] = range(0,len(kh_snf_3_west.index.values))
kh_snf_4_west['Ticks'] = range(0,len(kh_snf_4_west.index.values))

kh_snf_1['Ticks'] = range(0,len(kh_snf_1.index.values))
kh_snf_2['Ticks'] = range(0,len(kh_snf_2.index.values))
kh_snf_3['Ticks'] = range(0,len(kh_snf_3.index.values))
kh_snf_4['Ticks'] = range(0,len(kh_snf_4.index.values))

hw_bg_snf_2 = df3.between_time('12:09:42','12:13:45')
hw_bg_snf_2 = hw_bg_snf_2[(hw_bg_snf_2.Speed >= 1)]
hw_bg_snf_2['Ticks'] = range(0,len(hw_bg_snf_2.index.values))
```

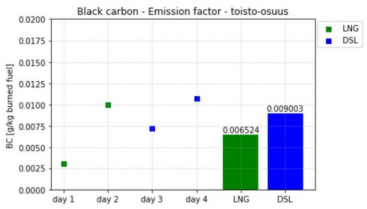
In [16]:
```
#CO2
#Calculate total mean for backgound & print values
day_1_co2_tot_bg = ((east5_snf_1['CO2_ppm'].reset_index(drop=True) + west5_snf_1['CO2_ppm'].reset_index(drop=True)) / 2)
day_2_co2_tot_bg = hw_bg_snf_2['CO2_ppm'].mean()
day_3_co2_tot_bg = ((east8_snf_3['CO2_ppm'].reset_index(drop=True) + west8_snf_3['CO2_ppm'].reset_index(drop=True)) / 2)
day_4_co2_tot_bg = ((east8_snf_4['CO2_ppm'].reset_index(drop=True) + west8_snf_4['CO2_ppm'].reset_index(drop=True)) / 2)

#Calculate corrected CO2 to dataframes
#East
kh_snf_1_east['CO2_cor'] = kh_snf_1_east['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_east['CO2_cor'] = kh_snf_2_east['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_east['CO2_cor'] = kh_snf_3_east['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_east['CO2_cor'] = kh_snf_4_east['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
#West
kh_snf_1_west['CO2_cor'] = kh_snf_1_west['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_west['CO2_cor'] = kh_snf_2_west['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_west['CO2_cor'] = kh_snf_3_west['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_west['CO2_cor'] = kh_snf_4_west['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()

#Day
kh_snf_1['CO2_cor'] = kh_snf_1['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2['CO2_cor'] = kh_snf_2['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3['CO2_cor'] = kh_snf_3['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4['CO2_cor'] = kh_snf_4['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
```

In [22]:
```
#BC
kh_snf_2 = kh_snf_2[(kh_snf_2['BC6'] < 6000)]
kh_snf_1['BC_cor'] = kh_snf_1['BC6'] - kh_bg_snf_1['BC6'].mean()
kh_snf_2['BC_cor'] = kh_snf_2['BC6'] - kh_bg_snf_2['BC6'].mean()
kh_snf_3['BC_cor'] = kh_snf_3['BC6'] - kh_bg_snf_3['BC6'].mean()
kh_snf_4['BC_cor'] = kh_snf_4['BC6'] - kh_bg_snf_4['BC6'].mean()
```

In [23]:
```
#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + kh_snf_1['OutTemp'].mean()
T2 = K + kh_snf_2['OutTemp'].mean()
T3 = K + kh_snf_3['OutTemp'].mean()
T4 = K + kh_snf_4['OutTemp'].mean()

#Print Delta CO2
print('Delta CO2')
print('1 CO2:',kh_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',kh_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta BC
print('Delta BC')
print('1 CO2:',kh_snf_1['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('2 CO2:',kh_snf_2['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('3 CO2:',kh_snf_3['BC_cor'].mean()*10**-9 ,'ng/m^3')
print('3 CO2:',kh_snf_4['BC_cor'].mean()*10**-9 ,'ng/m^3')


#Emission ratio for BC.
ER_BC_1 = ((kh_snf_1['BC_cor'].mean()*10**-9)/kh_snf_1['CO2_cor'].mean())
ER_BC_2 = ((kh_snf_2['BC_cor'].mean()*10**-9)/kh_snf_2['CO2_cor'].mean())
ER_BC_3 = ((kh_snf_3['BC_cor'].mean()*10**-9)/kh_snf_3['CO2_cor'].mean())
ER_BC_4 = ((kh_snf_4['BC_cor'].mean()*10**-9)/kh_snf_4['CO2_cor'].mean())


#print emission ratio
print('BC Emission ratio: [delta_ERbc/delta_CO2]')
print('Day1', ER_BC_1)
print('Day2', ER_BC_2)
print('Day3', ER_BC_3)
print('Day4', ER_BC_4)

#Emission factor

#Calculate emission factor for NOx
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel


EF_BC_1 = (ER_BC_1 * (r*T1)/(p*co2_mol) * efng) *10**6
EF_BC_2 = (ER_BC_2 * (r*T2)/(p*co2_mol) * efng) *10**6
EF_BC_3 = (ER_BC_3 * (r*T3)/(p*co2_mol) * efdiesel) *10**6
EF_BC_4 = (ER_BC_4 * (r*T4)/(p*co2_mol) * efdiesel) *10**6

#Calculate BC emission factor (mean) for each type of fuel
EF_BC_lng = ((EF_BC_1+EF_BC_2)/2)
EF_BC_dsl = ((EF_BC_3+EF_BC_4)/2)



#print BC emission factor
print('BC Emission Factor: [ER_bc*R*T/p*CO2mol*efCO2*10^6]')
print('Day1', EF_BC_1, 'g/kg burned fuel')
print('Day2', EF_BC_2, 'g/kg burned fuel')
print('Day3', EF_BC_3, 'g/kg burned fuel')
print('Day4', EF_BC_4, 'g/kg burned fuel')

#print BC emission factor for lng & dsl
print('BC Emission Factor for each fuel:')
print('natural gas:', EF_BC_lng, 'g/kg burned fuel')
print('diesel:', EF_BC_dsl, 'g/kg burned fuel')
```

```
Delta CO2
1 CO2: 67.10498418671361 ppm
2 CO2: 65.18065982599457 ppm
3 CO2: 61.63141667021775 ppm
3 CO2: 70.97332567271671 ppm
Delta BC
1 CO2: 1.526491283337504e-07 ng/m^3
2 CO2: 4.960509874460842e-07 ng/m^3
3 CO2: 2.757739942648739e-07 ng/m^3
3 CO2: 4.734495640271029e-07 ng/m^3
BC Emission ratio: [delta_ERbc/delta_CO2]
Day1 2.274780780947923e-09
Day2 7.610401440708568e-09
Day3 4.474568477640343e-09
Day4 6.670809906955009e-09
BC Emission Factor: [ER_bc*R*T/p*CO2mol*efCO2*10^6]
Day1 0.003015787404997772 g/kg burned fuel
Day2 0.010032232008612522 g/kg burned fuel
Day3 0.007286378597238736 g/kg burned fuel
Day4 0.010719383558265638 g/kg burned fuel
BC Emission Factor for each fuel:
natural gas: 0.006524009706805147 g/kg burned fuel
diesel: 0.009002881077752187 g/kg burned fuel
```

In [24]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_BC_1,EF_BC_2]
lng_c = EF_BC_lng
dsl = [EF_BC_3,EF_BC_4]
dsl_c = EF_BC_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('BC [g/kg burned fuel]')
ax.set_title('Black carbon - Emission factor - toisto-osuus')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%f' % float(height),
                ha='center', va='bottom')

autolabel(rects3)
autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
plt.ylim(0,0.02)

plt.savefig('/home/sami/Documents/emissionfactor/kh_BC_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Hiukkaskonsentraation päästökerroinlaskut

Toisto-osuus

In [2]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [3]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [4]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [5]:
```
#Kasvihuoneilmiö
############################################################
#Sniffer day 1
#West
west1_snf_1 = df1.between_time('08:41:12','08:45:21') #1 to turku
west2_snf_1 = df1.between_time('08:52:18','08:56:28') #2 to turku
west3_snf_1 = df1.between_time('09:02:56','09:07:07') #3 to turku
west4_snf_1 = df1.between_time('11:47:39','11:51:48') #4 to turku
west5_snf_1 = df1.between_time('11:59:46','12:03:27') #5 to turku (background)
#Sinnefer day 1
#East
east1_snf_1 = df1.between_time('08:46:52','08:51:00') #1 to Helsinki
east2_snf_1 = df1.between_time('08:57:23','09:01:34') #2 to Helsinki
east3_snf_1 = df1.between_time('11:42:03','11:46:12') #3 to Helsinki
east4_snf_1 = df1.between_time('11:52:46','11:56:53') #4 to Helsinki
east5_snf_1 = df1.between_time('12:04:08','12:07:47') #5 to Helsinki (background)
############################################################
#Sniffer day 2
#West
west1_snf_2 = df3.between_time('09:17:12','09:21:21') #1 to turku
west2_snf_2 = df3.between_time('09:27:31','09:31:41') #2 to turku
west3_snf_2 = df3.between_time('09:37:48','09:41:56') #3 to turku
west4_snf_2 = df3.between_time('09:48:01','09:52:07') #4 to turku
west5_snf_2 = df3.between_time('09:58:11','10:02:21') #5 to turku
west6_snf_2 = df3.between_time('10:08:23','10:12:32') #6 to turku
west7_snf_2 = df3.between_time('10:18:46','10:22:54') #7 to turku
west8_snf_2 = df3.between_time('09:07:34','09:11:15') #8 to turku (background)
#Sinnefer day 2
#East
east1_snf_2 = df3.between_time('09:22:11','09:26:18') #1 to Helsinki
east2_snf_2 = df3.between_time('09:32:33','09:36:38') #2 to Helsinki
east3_snf_2 = df3.between_time('09:42:45','09:46:53') #3 to Helsinki
east4_snf_2 = df3.between_time('09:52:57','09:57:03') #4 to Helsinki
east5_snf_2 = df3.between_time('10:03:10','10:07:16') #5 to Helsinki
east6_snf_2 = df3.between_time('10:13:23','10:17:30') #6 to Helsinki
east7_snf_2 = df3.between_time('12:58:49','13:02:44') #7 to Helsinki
east8_snf_2 = df3.between_time('09:12:06','09:15:45') #8 to Helsinki (background)
############################################################
#Sniffer day 3
#West
west1_snf_3 = df5.between_time('08:59:37','09:03:45') #1 to turku
west2_snf_3 = df5.between_time('09:09:59','09:14:07') #2 to turku
west3_snf_3 = df5.between_time('09:20:42','09:24:49') #3 to turku
west4_snf_3 = df5.between_time('09:31:05','09:35:14') #4 to turku
west5_snf_3 = df5.between_time('09:44:48','09:48:57') #5 to turku
west6_snf_3 = df5.between_time('09:55:14','09:59:22') #6 to turku
west7_snf_3 = df5.between_time('10:05:26','10:09:34') #7 to turku
west8_snf_3 = df5.between_time('08:49:36','08:53:39') #8 to turku (background)
#Sinnefer day 3
#East
east1_snf_3 = df5.between_time('09:04:38','09:08:44')  #1 to Helsinki
east2_snf_3 = df5.between_time('09:15:10','09:19:17')  #2 to Helsinki
east3_snf_3 = df5.between_time('09:25:48','09:29:53')  #3 to Helsinki
east4_snf_3 = df5.between_time('09:39:29','09:43:36')  #4 to Helsinki
east5_snf_3 = df5.between_time('09:49:58','09:54:04')  #5 to Helsinki
east6_snf_3 = df5.between_time('10:00:10','10:04:14')  #6 to Helsinki
east7_snf_3 = df5.between_time('12:44:12','12:48:07')  #7 to Helsinki
east8_snf_3 = df5.between_time('08:54:24','08:58:05')  #8 to Helsinki (background)
############################################################
#Sniffer day 4
#West
west1_snf_4 = df7.between_time('09:08:36','09:12:42') #1 to turku
west2_snf_4 = df7.between_time('09:19:07','09:23:22') #2 to turku
west3_snf_4 = df7.between_time('09:29:30','09:33:38') #3 to turku
west4_snf_4 = df7.between_time('09:39:49','09:43:56') #4 to turku
west5_snf_4 = df7.between_time('09:49:57','09:54:03') #5 to turku
west6_snf_4 = df7.between_time('10:00:33','10:04:43') #6 to turku
west7_snf_4 = df7.between_time('10:10:48','10:14:51') #7 to turku
west8_snf_4 = df7.between_time('08:53:56','08:57:39') #8 to turku (background)
#Sinnefer day 4
#East
east1_snf_4 = df7.between_time('09:13:49','09:17:57') #1 to Helsinki
east2_snf_4 = df7.between_time('09:24:12','09:28:18') #2 to Helsinki
east3_snf_4 = df7.between_time('09:34:36','09:38:42') #3 to Helsinki
east4_snf_4 = df7.between_time('09:44:45','09:48:47') #4 to Helsinki
east5_snf_4 = df7.between_time('09:55:19','09:59:23') #5 to Helsinki
east6_snf_4 = df7.between_time('10:05:33','10:09:37') #6 to Helsinki
east7_snf_4 = df7.between_time('12:49:24','12:53:19') #7 to Helsinki
east8_snf_4 = df7.between_time('08:58:23','09:02:31') #8 to Helsinki (background)
############################################################
```

In [6]:
```
#Sniffer
#East
kh_snf_1_east = east2_snf_1.append(east3_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2_east = east1_snf_2.append(east3_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3_east = east2_snf_3.append(east5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4_east = east2_snf_4.append(east3_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)
#West
kh_snf_1_west = west1_snf_1.append(west3_snf_1,ignore_index=True).append(west4_snf_1,ignore_index=True)
kh_snf_2_west = west1_snf_2.append(west2_snf_2,ignore_index=True).append(west4_snf_2,ignore_index=True)
kh_snf_3_west = west2_snf_3.append(west4_snf_3,ignore_index=True).append(west5_snf_3,ignore_index=True)
kh_snf_4_west = west4_snf_4.append(west5_snf_4,ignore_index=True).append(west6_snf_4,ignore_index=True)

kh_snf_1 = west1_snf_1.append(east2_snf_1,ignore_index=True).append(west3_snf_1,ignore_index=True).append(east3_snf_1,ignore_index=True)
.append(west4_snf_1,ignore_index=True).append(east4_snf_1,ignore_index=True)
kh_snf_2 = west1_snf_2.append(east1_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True).append(east3_snf_2,ignore_index=True)
.append(west4_snf_2,ignore_index=True).append(east4_snf_2,ignore_index=True)
kh_snf_3 = west2_snf_3.append(east2_snf_1,ignore_index=True).append(west4_snf_3,ignore_index=True).append(east5_snf_3,ignore_index=True)
.append(west5_snf_3,ignore_index=True).append(east6_snf_3,ignore_index=True)
kh_snf_4 = west4_snf_4.append(east2_snf_4,ignore_index=True).append(west5_snf_4,ignore_index=True).append(east3_snf_4,ignore_index=True)
.append(west6_snf_4,ignore_index=True).append(east4_snf_4,ignore_index=True)

kh_bg_snf_1 = west5_snf_1.append(east5_snf_1,ignore_index=True)
kh_bg_snf_2 = west8_snf_2.append(east8_snf_2,ignore_index=True)
kh_bg_snf_3 = west8_snf_3.append(east8_snf_3,ignore_index=True)
kh_bg_snf_4 = west8_snf_4.append(east8_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
kh_snf_1 = kh_snf_1[(kh_snf_1.Speed > 6.95)]
kh_snf_2 = kh_snf_2[(kh_snf_2.Speed > 6.95)]
kh_snf_3 = kh_snf_3[(kh_snf_3.Speed > 6.95)]
kh_snf_4 = kh_snf_4[(kh_snf_4.Speed > 6.95)]

kh_snf_1_east['Ticks'] = range(0,len(kh_snf_1_east.index.values))
kh_snf_2_east['Ticks'] = range(0,len(kh_snf_2_east.index.values))
kh_snf_3_east['Ticks'] = range(0,len(kh_snf_3_east.index.values))
kh_snf_4_east['Ticks'] = range(0,len(kh_snf_4_east.index.values))

kh_snf_1_west['Ticks'] = range(0,len(kh_snf_1_west.index.values))
kh_snf_2_west['Ticks'] = range(0,len(kh_snf_2_west.index.values))
kh_snf_3_west['Ticks'] = range(0,len(kh_snf_3_west.index.values))
kh_snf_4_west['Ticks'] = range(0,len(kh_snf_4_west.index.values))

kh_snf_1['Ticks'] = range(0,len(kh_snf_1.index.values))
kh_snf_2['Ticks'] = range(0,len(kh_snf_2.index.values))
kh_snf_3['Ticks'] = range(0,len(kh_snf_3.index.values))
kh_snf_4['Ticks'] = range(0,len(kh_snf_4.index.values))

hw_bg_snf_2 = df3.between_time('12:09:42','12:13:45')
hw_bg_snf_2 = hw_bg_snf_2[(hw_bg_snf_2.Speed >= 1)]
hw_bg_snf_2['Ticks'] = range(0,len(hw_bg_snf_2.index.values))
```

In [7]:
```
#CO2
#Calculate total mean for backgound & print values
day_1_co2_tot_bg = ((east5_snf_1['CO2_ppm'].reset_index(drop=True) + west5_snf_1['CO2_ppm'].reset_index(drop=True)) / 2)
day_2_co2_tot_bg = hw_bg_snf_2['CO2_ppm'].mean()
day_3_co2_tot_bg = ((east8_snf_3['CO2_ppm'].reset_index(drop=True) + west8_snf_3['CO2_ppm'].reset_index(drop=True)) / 2)
day_4_co2_tot_bg = ((east8_snf_4['CO2_ppm'].reset_index(drop=True) + west8_snf_4['CO2_ppm'].reset_index(drop=True)) / 2)

#Calculate corrected CO2 to dataframes
#East
kh_snf_1_east['CO2_cor'] = kh_snf_1_east['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_east['CO2_cor'] = kh_snf_2_east['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_east['CO2_cor'] = kh_snf_3_east['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_east['CO2_cor'] = kh_snf_4_east['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
#West
kh_snf_1_west['CO2_cor'] = kh_snf_1_west['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2_west['CO2_cor'] = kh_snf_2_west['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3_west['CO2_cor'] = kh_snf_3_west['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4_west['CO2_cor'] = kh_snf_4_west['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()

#Day
kh_snf_1['CO2_cor'] = kh_snf_1['CO2_ppm'].reset_index(drop=True) - day_1_co2_tot_bg.mean()
kh_snf_2['CO2_cor'] = kh_snf_2['CO2_ppm'].reset_index(drop=True) - day_2_co2_tot_bg.mean()
kh_snf_3['CO2_cor'] = kh_snf_3['CO2_ppm'].reset_index(drop=True) - day_3_co2_tot_bg.mean()
kh_snf_4['CO2_cor'] = kh_snf_4['CO2_ppm'].reset_index(drop=True) - day_4_co2_tot_bg.mean()
```

In [56]:
```
#Format cpc data to correct type
kh_snf_1['cpcConcentration_x/cm3']=kh_snf_1['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_snf_2['cpcConcentration_x/cm3']=kh_snf_2['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_snf_3['cpcConcentration_x/cm3']=kh_snf_3['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_snf_4['cpcConcentration_x/cm3']=kh_snf_4['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')

kh_bg_snf_1['cpcConcentration_x/cm3']=kh_bg_snf_1['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_bg_snf_2['cpcConcentration_x/cm3']=kh_bg_snf_2['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_bg_snf_3['cpcConcentration_x/cm3']=kh_bg_snf_3['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
kh_bg_snf_4['cpcConcentration_x/cm3']=kh_bg_snf_4['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')

kh_bg_snf_1 = kh_bg_snf_1[(kh_bg_snf_1['cpcConcentration_x/cm3'] < 20000)]
kh_bg_snf_2 = kh_bg_snf_2[(kh_bg_snf_2['cpcConcentration_x/cm3'] < 20000)]
kh_bg_snf_3 = kh_bg_snf_3[(kh_bg_snf_3['cpcConcentration_x/cm3'] < 20000)]
kh_bg_snf_4 = kh_bg_snf_4[(kh_bg_snf_4['cpcConcentration_x/cm3'] < 20000)]

kh_snf_1['CPC_cor'] = kh_snf_1['cpcConcentration_x/cm3'] - kh_bg_snf_1['cpcConcentration_x/cm3'].mean()
kh_snf_2['CPC_cor'] = kh_snf_2['cpcConcentration_x/cm3'] - kh_bg_snf_2['cpcConcentration_x/cm3'].mean()
kh_snf_3['CPC_cor'] = kh_snf_3['cpcConcentration_x/cm3'] - kh_bg_snf_3['cpcConcentration_x/cm3'].mean()
kh_snf_4['CPC_cor'] = kh_snf_4['cpcConcentration_x/cm3'] - kh_bg_snf_4['cpcConcentration_x/cm3'].mean()
```

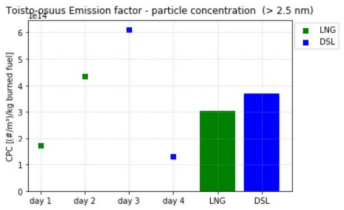
In [72]:

```
#CPC

#atm pressure = 101300 (N/M²)
#CO2 molar mass = 44.0095 g/mol
#r=molar gas factor = 8.3145 J/mol/K
co2_mol = 44.0095
p = 101300
r = 8.3145

#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + kh_snf_1['OutTemp'].mean()
T2 = K + kh_snf_2['OutTemp'].mean()
T3 = K + kh_snf_3['OutTemp'].mean()
T4 = K + kh_snf_4['OutTemp'].mean()

#Emission ratio for cpc. (CO2 ppm to ppm/cm³)
ER_cpc_1 = ((kh_snf_1['CPC_cor'].mean()*10**6)/kh_snf_1['CO2_cor'].mean())
ER_cpc_2 = ((kh_snf_2['CPC_cor'].mean()*10**6)/kh_snf_2['CO2_cor'].mean())
ER_cpc_3 = ((kh_snf_3['CPC_cor'].mean()*10**6)/kh_snf_3['CO2_cor'].mean())
ER_cpc_4 = ((kh_snf_4['CPC_cor'].mean()*10**6)/kh_snf_4['CO2_cor'].mean())


#Print Delta CO2
print('Delta CO2')
print('1 CO2:',kh_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',kh_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta CPC
print('Delta CPC')
print('1 CPC:',kh_snf_1['CPC_cor'].mean(),'#/cm^3')
print('2 CPC:',kh_snf_2['CPC_cor'].mean(),'#/cm^3')
print('3 CPC:',kh_snf_3['CPC_cor'].mean(),'#/cm^3')
print('4 CPC:',kh_snf_4['CPC_cor'].mean(),'#/cm^3')
print('1 CPC:',kh_snf_1['CPC_cor'].mean()*10**6 ,'#/m^3')
print('2 CPC:',kh_snf_2['CPC_cor'].mean()*10**6 ,'#/m^3')
print('3 CPC:',kh_snf_3['CPC_cor'].mean()*10**6 ,'#/m^3')
print('4 CPC:',kh_snf_4['CPC_cor'].mean()*10**6 ,'#/m^3')

#print emission ratio
print('CPC Emission ratio:')
print('Day1', ER_cpc_1)
print('Day2', ER_cpc_2)
print('Day3', ER_cpc_3)
print('Day4', ER_cpc_4)

#Emission factor

#Calculate emission factor for CPC
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel

#C30*(($B$4*F23)/($B$2*$B$3)*$E$3)*10^6

EF_cpc_1 = (ER_cpc_1 * (r*T1)/(p*co2_mol) * efng)*10**6
EF_cpc_2 = (ER_cpc_2 * (r*T2)/(p*co2_mol) * efng)*10**6
EF_cpc_3 = (ER_cpc_3 * (r*T3)/(p*co2_mol) * efdiesel)*10**6
EF_cpc_4 = (ER_cpc_4 * (r*T4)/(p*co2_mol) * efdiesel)*10**6


#Calculate cpc emission factor (mean) for each type of fuel
EF_cpc_lng = ((EF_cpc_1+EF_cpc_2)/2)
EF_cpc_dsl = ((EF_cpc_3+EF_cpc_4)/2)

#print cpc emission factor
print('CPC Emission Factor:')
print('Day1', EF_cpc_1, '#/kg burned fuel')
print('Day2', EF_cpc_2, '#/kg burned fuel')
print('Day3', EF_cpc_3, '#/kg burned fuel')
print('Day4', EF_cpc_4, '#/kg burned fuel')

#print cpc emission factor for lng & dsl
print('CPC Emission Factor for each fuel:')
print('natural gas:', EF_cpc_lng, '#/kg burned fuel')
print('diesel:', EF_cpc_dsl, '#/kg burned fuel')
```

```
Delta CO2
1 CO2: 67.10498418671361 ppm
2 CO2: 64.12349886621317 ppm
3 CO2: 61.63141667021775 ppm
3 CO2: 70.97332567271671 ppm
Delta CPC
1 CPC: 8728.880091673425 #/cm^3
2 CPC: 21181.754979865775 #/cm^3
3 CPC: 23050.911278031785 #/cm^3
4 CPC: 5717.72971722025 #/cm^3
1 CPC: 8728880091.673426 #/m^3
2 CPC: 21181754979.865776 #/m^3
3 CPC: 23050911278.031784 #/m^3
4 CPC: 5717729717.22025 #/m^3
CPC Emission ratio:
Day1 130077969.57951884
Day2 330327498.5674011
Day3 374012354.7925958
Day4 80561671.06479889
CPC Emission Factor:
Day1 172450684308193.56 #/kg burned fuel
Day2 435458626442626.7 #/kg burned fuel
Day3 609040990361769.6 #/kg burned fuel
Day4 129455263196459.45 #/kg burned fuel
CPC Emission Factor for each fuel:
natural gas: 303954655375410.1 #/kg burned fuel
diesel: 369248126779114.56 #/kg burned fuel
```

In [96]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_cpc_1,EF_cpc_2]
lng_c = EF_cpc_lng
dsl = [EF_cpc_3,EF_cpc_4]
dsl_c = EF_cpc_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('CPC [(#/m³)/kg burned fuel]')
ax.set_title('Toisto-osuus Emission factor - particle concentration  (> 2.5 nm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/kh_CPC_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



In [78]:

```
#Eeps
kh_snf_1['Eeps_cor'] = kh_snf_1['Eeps_Total_Conc_x/cm3']-kh_bg_snf_1['Eeps_Total_Conc_x/cm3'].mean()
kh_snf_2['Eeps_cor'] = kh_snf_2['eepsConcentration_x/cm3']-kh_bg_snf_2['eepsConcentration_x/cm3'].mean()
kh_snf_3['Eeps_cor'] = kh_snf_3['eepsConcentration_x/cm3']-kh_bg_snf_3['eepsConcentration_x/cm3'].mean()
kh_snf_4['Eeps_cor'] = kh_snf_4['eepsConcentration_x/cm3']-kh_bg_snf_4['eepsConcentration_x/cm3'].mean()
```

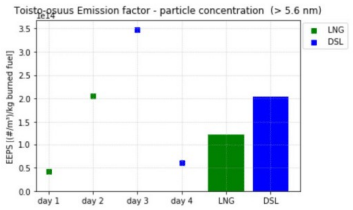
In [79]:

```
#EEPS

#atm pressure = 101300 (N/M²)
#CO2 molar mass = 44.0095 g/mol
#r=molar gas factor = 8.3145 J/mol/K
co2_mol = 44.0095
p = 101300
r = 8.3145

#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + kh_snf_1['OutTemp'].mean()
T2 = K + kh_snf_2['OutTemp'].mean()
T3 = K + kh_snf_3['OutTemp'].mean()
T4 = K + kh_snf_4['OutTemp'].mean()

#Emission ratio for cpc. (CO2 ppm to ppm/cm³)
ER_eeps_1 = ((kh_snf_1['Eeps_cor'].mean()*10**6)/kh_snf_1['CO2_cor'].mean())
ER_eeps_2 = ((kh_snf_2['Eeps_cor'].mean()*10**6)/kh_snf_2['CO2_cor'].mean())
ER_eeps_3 = ((kh_snf_3['Eeps_cor'].mean()*10**6)/kh_snf_3['CO2_cor'].mean())
ER_eeps_4 = ((kh_snf_4['Eeps_cor'].mean()*10**6)/kh_snf_4['CO2_cor'].mean())


#Print Delta CO2
print('Delta CO2')
print('1 CO2:',kh_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',kh_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',kh_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta CPC
print('Delta Eeps')
print('1 CPC:',kh_snf_1['Eeps_cor'].mean(),'#/cm^3')
print('2 CPC:',kh_snf_2['Eeps_cor'].mean(),'#/cm^3')
print('3 CPC:',kh_snf_3['Eeps_cor'].mean(),'#/cm^3')
print('4 CPC:',kh_snf_4['Eeps_cor'].mean(),'#/cm^3')
print('1 CPC:',kh_snf_1['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('2 CPC:',kh_snf_2['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('3 CPC:',kh_snf_3['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('4 CPC:',kh_snf_4['Eeps_cor'].mean()*10**6 ,'#/m^3')

#print emission ratio
print('CPC Emission ratio:')
print('Day1', ER_eeps_1)
print('Day2', ER_eeps_2)
print('Day3', ER_eeps_3)
print('Day4', ER_eeps_4)

#Emission factor

#Calculate emission factor for CPC
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel



EF_eeps_1 = (ER_eeps_1 * (r*T1)/(p*co2_mol) * efng)*10**6
EF_eeps_2 = (ER_eeps_2 * (r*T2)/(p*co2_mol) * efng)*10**6
EF_eeps_3 = (ER_eeps_3 * (r*T3)/(p*co2_mol) * efdiesel)*10**6
EF_eeps_4 = (ER_eeps_4 * (r*T4)/(p*co2_mol) * efdiesel)*10**6


#Calculate cpc emission factor (mean) for each type of fuel
EF_eeps_lng = ((EF_eeps_1+EF_eeps_2)/2)
EF_eeps_dsl = ((EF_eeps_3+EF_eeps_4)/2)

#print cpc emission factor
print('CPC Emission Factor:')
print('Day1', EF_eeps_1, '#/kg burned fuel')
print('Day2', EF_eeps_2, '#/kg burned fuel')
print('Day3', EF_eeps_3, '#/kg burned fuel')
print('Day4', EF_eeps_4, '#/kg burned fuel')

#print cpc emission factor for lng & dsl
print('CPC Emission Factor for each fuel:')
print('natural gas:', EF_eeps_lng, '#/kg burned fuel')
print('diesel:', EF_eeps_dsl, '#/kg burned fuel')
```

```
Delta CO2
1 CO2: 67.10498418671361 ppm
2 CO2: 64.12349886621317 ppm
3 CO2: 61.63141667021775 ppm
3 CO2: 70.97332567271671 ppm
Delta Eeps
1 CPC: 2084.8003740148715 #/cm^3
2 CPC: 10011.98523925385 #/cm^3
3 CPC: 13123.871945959987 #/cm^3
4 CPC: 2645.4423728813554 #/cm^3
1 CPC: 2084800374.0148716 #/m^3
2 CPC: 10011985239.25385 #/m^3
3 CPC: 13123871945.959988 #/m^3
4 CPC: 2645442372.8813553 #/m^3
CPC Emission ratio:
Day1 31067742.57205845
Day2 156135978.4833761
Day3 212941266.8896488
Day4 37273755.28491411
CPC Emission Factor:
Day1 41188015801454.03 #/kg burned fuel
Day2 205828334073051.47 #/kg burned fuel
Day3 346753144417594.6 #/kg burned fuel
Day4 59895527698870.77 #/kg burned fuel
CPC Emission Factor for each fuel:
natural gas: 123508174937252.75 #/kg burned fuel
diesel: 203324336058232.7 #/kg burned fuel
```

In [90]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_eeps_1,EF_eeps_2]
lng_c = EF_eeps_lng
dsl = [EF_eeps_3,EF_eeps_4]
dsl_c = EF_eeps_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('EEPS [(#/m³)/kg burned fuel]')
ax.set_title('Toisto-osuus Emission factor - particle concentration  (> 5.6 nm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/kh_eeps_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



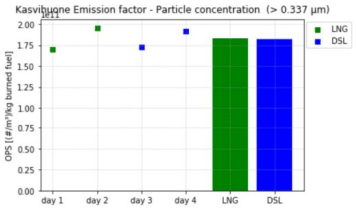
In [82]:

```
#OPS
kh_snf_1['Ops_cor'] = (kh_snf_1['Total_Conc.x/cm3']-kh_bg_snf_1['Total_Conc.x/cm3'].mean())
kh_snf_2['Ops_cor'] = (kh_snf_2['Total_Conc.x/cm3']-kh_bg_snf_2['Total_Conc.x/cm3'].mean())
kh_snf_3['Ops_cor'] = (kh_snf_3['Total_Conc.x/cm3']-kh_bg_snf_3['Total_Conc.x/cm3'].mean())
kh_snf_4['Ops_cor'] = (kh_snf_4['Total_Conc.x/cm3']-kh_bg_snf_4['Total_Conc.x/cm3'].mean())
```

```
Delta CO2
1 CO2: 67.10498418671361 ppm
2 CO2: 64.12349886621317 ppm
3 CO2: 61.63141667021775 ppm
3 CO2: 70.97332567271671 ppm
Delta OPS
1 OPS: 8.602968078259174 #/cm^3
2 OPS: 9.5213852119313 96 #/cm^3
3 OPS: 6.5326404503357045 #/cm^3
4 OPS: 8.480213175320298 #/cm^3
1 OPS: 8602968.078259174 #/m^3
2 OPS: 9521385.211931396 #/m^3
3 OPS: 6532640.450335705 #/m^3
4 OPS: 8480213.175320297 #/m^3
CPC Emission ratio:
Day1 128201.62589297668
Day2 148485.1166932851
Day3 105995.29920415545
Day4 119484.51189148979
CPC Emission Factor:
Day1 169963124317.92368 #/kg burned fuel
Day2 195742483574.18478 #/kg burned fuel
Day3 172602538856.7429 #/kg burned fuel
Day4 192000721067.11722 #/kg burned fuel
CPC Emission Factor for each fuel:
natural gas: 182852803946.05423 #/kg burned fuel
diesel: 182301629961.93005 #/kg burned fuel
```

In [91]:

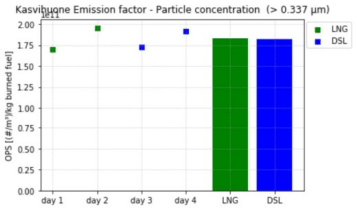
```
#ops

fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_ops_1,EF_ops_2]
lng_c = EF_ops_lng
dsl = [EF_ops_3,EF_ops_4]
dsl_c = EF_ops_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('OPS [(#/m³)/kg burned fuel]')
ax.set_title('Kasvihuone Emission factor - Particle concentration  (> 0.337 µm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/kh_OPS_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



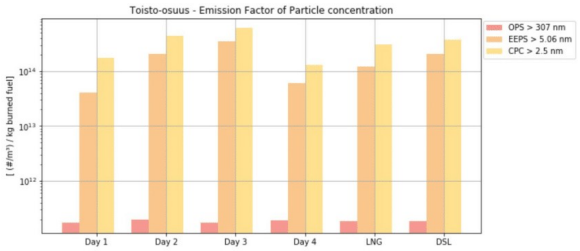
```
Delta CO2
1 CO2: 67.10498418671361 ppm
2 CO2: 64.12349886621317 ppm
3 CO2: 61.63141667021775 ppm
3 CO2: 70.97332567271671 ppm
Delta OPS
1 OPS: 8.602968078259174 #/cm^3
2 OPS: 9.5213852119313 96 #/cm^3
3 OPS: 6.5326404503357045 #/cm^3
4 OPS: 8.480213175320298 #/cm^3
1 OPS: 8602968.078259174 #/m^3
2 OPS: 9521385.211931396 #/m^3
3 OPS: 6532640.450335705 #/m^3
4 OPS: 8480213.175320297 #/m^3
CPC Emission ratio:
Day1 128201.62589297668
Day2 148485.1166932851
Day3 105995.29920415545
Day4 119484.51189148979
CPC Emission Factor:
Day1 169963124317.92368 #/kg burned fuel
Day2 195742483574.18478 #/kg burned fuel
Day3 172602538856.7429 #/kg burned fuel
Day4 192000721067.11722 #/kg burned fuel
CPC Emission Factor for each fuel:
natural gas: 182852803946.05423 #/kg burned fuel
diesel: 182301629961.93005 #/kg burned fuel
```

In [91]:

```
#ops

fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_ops_1,EF_ops_2]
lng_c = EF_ops_lng
dsl = [EF_ops_3,EF_ops_4]
dsl_c = EF_ops_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('OPS [(#/m³)/kg burned fuel]')
ax.set_title('Kasvihuone Emission factor - Particle concentration  (> 0.337 µm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/kh_OPS_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



In [97]:
```
jep = pd.DataFrame(columns = ['EF_ops', 'EF_eeps', 'EF_cpc', 'day', 'fuel'])
jep['EF_ops'] = [EF_ops_1,EF_ops_2,EF_ops_3,EF_ops_4,EF_ops_lng,EF_ops_dsl]
jep['EF_eeps'] = [EF_eeps_1,EF_eeps_2,EF_eeps_3,EF_eeps_4,EF_eeps_lng,EF_eeps_dsl]
jep['EF_cpc'] = [EF_cpc_1,EF_cpc_2,EF_cpc_3,EF_cpc_4,EF_cpc_lng,EF_cpc_dsl]
jep['day'] = ['Day 1','Day 2','Day 3','Day 4','LNG','DSL']

#jep = pd.DataFrame(raw_data, columns = ['EF_ops', 'EF_eeps', 'EF_cpc', 'day'])



# Setting the positions and width for the bars
pos = list(range(len(jep['EF_cpc'])))
width = 0.25

# Plotting the bars
fig, ax = plt.subplots(figsize=(10,5))

# Create a bar with pre_score data,
# in position pos,
plt.bar(pos,
        #using df['pre_score'] data,
        jep['EF_ops'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#EE3224',
        # with label the first value in first_name
        label=jep['day'][0])

# Create a bar with mid_score data,
# in position pos + some width buffer,
plt.bar([p + width for p in pos],
        #using df['mid_score'] data,
        jep['EF_eeps'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#F78F1E',
        # with label the second value in first_name
        label=jep['day'][1])

# Create a bar with post_score data,
# in position pos + some width buffer,
plt.bar([p + width*2 for p in pos],
        #using df['post_score'] data,
        jep['EF_cpc'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#FFC222',
        # with label the third value in first_name
        label=jep['day'][2])

# Create a bar with post_score data,
# in position pos + some width buffer,


# Set the y axis label
ax.set_ylabel('[ (#/m³) / kg burned fuel]')

# Set the chart's title
ax.set_title('Toisto-osuus - Emission Factor of Particle concentration')

# Set the position of the x ticks
ax.set_xticks([p + 1.5 * width for p in pos])

# Set the labels for the x ticks
ax.set_xticklabels(jep['day'])

# Setting the x-axis and y-axis limits
#plt.xlim(min(pos)-width, max(pos)+width*4)
#plt.ylim([0, max(jep['pre_score'] + df['mid_score'] + df['post_score'])] )

# Adding the legend and showing the plot
plt.legend(['OPS > 307 nm', 'EEPS > 5.06 nm', 'CPC > 2.5 nm'],bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
plt.grid()
plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/kh_ntot_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Maantie-osuus

In [10]:
```
# -*- coding: utf-8 -*-
#supress warnings message
import warnings; warnings.simplefilter("ignore")
#Import some libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import seaborn as sns
import bokeh
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
%matplotlib inline
```

In [11]:
```
# Read data and create dataframe
#Day one LNG
df1 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_nuuskija.csv', header=0, sep=";", index_col=None)
df2 = pd.read_csv('/home/sami/Containerships/Containerships/20171013/Muokatut/20171013_LNG_iveco.csv', header=0, sep=";", index_col=None
)
#Day two LNG
df3 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_Nuuskija.csv', header=0, sep=";", index_col=None)
df4 = pd.read_csv('/home/sami/Containerships/Containerships/20171017/Muokatut/20171017_LNG_Iveco.csv', header=0, sep=";", index_col=None
)
#Day three Diesel
df5 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Nuuskija.csv', header=0, sep=";", index_col=None)
df6 = pd.read_csv('/home/sami/Containerships/Containerships/20171019/Muokatut/20171019_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
#Day four Diesel
df7 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Nuuskija.csv', header=0, sep=";", index_col=None)
df8 = pd.read_csv('/home/sami/Containerships/Containerships/20171020/Muokatut/20171020_Diesel_Iveco.csv', header=0, sep=";", index_col=N
one)
```

In [12]:
```
#Set index to all dataframes [Using Datetime]
df1 = df1.set_index(pd.DatetimeIndex(df1['Datetime']))
df2 = df2.set_index(pd.DatetimeIndex(df2['Datetime']))
df3 = df3.set_index(pd.DatetimeIndex(df3['Datetime']))
df4 = df4.set_index(pd.DatetimeIndex(df4['Datetime']))
df5 = df5.set_index(pd.DatetimeIndex(df5['Datetime']))
df6 = df6.set_index(pd.DatetimeIndex(df6['Datetime']))
df7 = df7.set_index(pd.DatetimeIndex(df7['Datetime']))
df8 = df8.set_index(pd.DatetimeIndex(df8['Datetime']))
```

In [13]:
```
#Sniffer
#Turuntie start - Kasvihuoneilmiö
tt_kh_west_snf_1 = df1.between_time('08:01:22','08:25:43')
tt_kh_west_snf_2 = df3.between_time('08:38:49','09:03:14')
tt_kh_west_snf_3 = df5.between_time('08:19:11','08:43:22')
tt_kh_west_snf_4 = df7.between_time('08:13:31','08:53:38')
#Kasvihuoneilmiö - Turuntie start
kh_tt_east_snf_1 = df1.between_time('12:11:46','12:35:56')
kh_tt_east_snf_2 = df3.between_time('13:03:00','13:26:37')
kh_tt_east_snf_3 = df5.between_time('12:48:23','13:15:03')
kh_tt_east_snf_4 = df7.between_time('12:53:35','13:17:18')
#Sniffer
#Kasvihuoneilmiö west turning point - Salo
khwt_salo_west_snf_1 = df1.between_time('09:13:23','09:36:40')
khwt_salo_west_snf_2 = df3.between_time('10:24:57','10:47:54')
khwt_salo_west_snf_3 = df5.between_time('10:09:44','10:32:34')
khwt_salo_west_snf_4 = df7.between_time('10:15:02','10:37:39')
#Salo - Kasvihuoneilmiö west turning point
khwt_salo_east_snf_1 = df1.between_time('11:17:43','11:40:51')
khwt_salo_east_snf_2 = df3.between_time('12:35:49','12:58:38')
khwt_salo_east_snf_3 = df5.between_time('12:21:14','12:44:00')
khwt_salo_east_snf_4 = df7.between_time('12:26:37','12:49:12')
#Sniffer
#Salo - Paimio
salo_pai_west_snf_1 = df1.between_time('09:47:39','10:02:21')
salo_pai_west_snf_2 = df3.between_time('10:58:38','11:13:03')
salo_pai_west_snf_3 = df5.between_time('10:42:45','10:57:08')
salo_pai_west_snf_4 = df7.between_time('10:47:20','11:01:30')
#Paimio - Salo
salo_pai_east_snf_1 = df1.between_time('11:01:51','11:07:32')
salo_pai_east_snf_2 = df3.between_time('12:19:52','12:25:29')
salo_pai_east_snf_3 = df5.between_time('12:05:02','12:10:49')
salo_pai_east_snf_4 = df7.between_time('12:11:07','12:16:48')
#Sniffer
#Highway background [Paimio - Salo]
hw_bg_snf_1 = df1.between_time('10:52:18','10:55:59')
hw_bg_snf_2 = df3.between_time('12:09:42','12:13:45')
hw_bg_snf_3 = df5.between_time('11:56:07','11:59:30')
hw_bg_snf_4 = df7.between_time('12:00:28','12:04:45')
```

In [14]:
```
#Sniffer
#All highway parts combined in driven order
hw_snf_1 = tt_kh_west_snf_1.append(khwt_salo_west_snf_1,ignore_index=True).append(salo_pai_west_snf_1,ignore_index=True).append(salo_pai
_east_snf_1,ignore_index=True).append(khwt_salo_east_snf_1,ignore_index=True).append(kh_tt_east_snf_1,ignore_index=True)
hw_snf_2 = tt_kh_west_snf_2.append(khwt_salo_west_snf_2,ignore_index=True).append(salo_pai_west_snf_2,ignore_index=True).append(salo_pai
_east_snf_2,ignore_index=True).append(khwt_salo_east_snf_2,ignore_index=True).append(kh_tt_east_snf_2,ignore_index=True)
hw_snf_3 = tt_kh_west_snf_3.append(khwt_salo_west_snf_3,ignore_index=True).append(salo_pai_west_snf_3,ignore_index=True).append(salo_pai
_east_snf_3,ignore_index=True).append(khwt_salo_east_snf_3,ignore_index=True).append(kh_tt_east_snf_3,ignore_index=True)
hw_snf_4 = tt_kh_west_snf_4.append(khwt_salo_west_snf_4,ignore_index=True).append(salo_pai_west_snf_4,ignore_index=True).append(salo_pai
_east_snf_4,ignore_index=True).append(khwt_salo_east_snf_4,ignore_index=True).append(kh_tt_east_snf_4,ignore_index=True)

#Drop all values from dataframe when speed is less than 6.95m/s (25.02 km/h)
#Sniffer
hw_snf_1 = hw_snf_1[(hw_snf_1.Speed > 6.95)]
hw_snf_2 = hw_snf_2[(hw_snf_2.Speed > 6.95)]
hw_snf_3 = hw_snf_3[(hw_snf_3.Speed > 6.95)]
hw_snf_4 = hw_snf_4[(hw_snf_4.Speed > 6.95)]
```

In [16]:
```
#CO2
#Background for CO2
CO2_bg_1 = hw_bg_snf_1['CO2_ppm'].mean()
CO2_bg_2 = hw_bg_snf_2['CO2_ppm'].mean()
CO2_bg_3 = hw_bg_snf_3['CO2_ppm'].mean()
CO2_bg_4 = hw_bg_snf_4['CO2_ppm'].mean()

hw_snf_1['Ticks'] = range(0,len(hw_snf_1.index.values))
hw_snf_2['Ticks'] = range(0,len(hw_snf_2.index.values))
hw_snf_3['Ticks'] = range(0,len(hw_snf_3.index.values))
hw_snf_4['Ticks'] = range(0,len(hw_snf_4.index.values))

#Calculate corrected CO2 to dataframes
hw_snf_1['CO2_cor'] = hw_snf_1['CO2_ppm'] - CO2_bg_1
hw_snf_2['CO2_cor'] = hw_snf_2['CO2_ppm'] - CO2_bg_2
hw_snf_3['CO2_cor'] = hw_snf_3['CO2_ppm'] - CO2_bg_3
hw_snf_4['CO2_cor'] = hw_snf_4['CO2_ppm'] - CO2_bg_4
```

In [45]:
```
#Format cpc data to correct type
hw_snf_1['cpcConcentration_x/cm3']=hw_snf_1['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_snf_2['cpcConcentration_x/cm3']=hw_snf_2['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_snf_3['cpcConcentration_x/cm3']=hw_snf_3['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_snf_4['cpcConcentration_x/cm3']=hw_snf_4['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')

hw_bg_snf_1['cpcConcentration_x/cm3']=hw_bg_snf_1['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_bg_snf_2['cpcConcentration_x/cm3']=hw_bg_snf_2['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_bg_snf_3['cpcConcentration_x/cm3']=hw_bg_snf_3['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')
hw_bg_snf_4['cpcConcentration_x/cm3']=hw_bg_snf_4['cpcConcentration_x/cm3'].apply(pd.to_numeric, errors='coerse')

hw_bg_snf_1 = hw_bg_snf_1[(hw_bg_snf_1['cpcConcentration_x/cm3'] < 22000)]
hw_bg_snf_2 = hw_bg_snf_2[(hw_bg_snf_2['cpcConcentration_x/cm3'] < 22000)]
hw_bg_snf_3 = hw_bg_snf_3[(hw_bg_snf_3['cpcConcentration_x/cm3'] < 22000)]
hw_bg_snf_4 = hw_bg_snf_4[(hw_bg_snf_4['cpcConcentration_x/cm3'] < 22000)]
hw_snf_3 = hw_snf_3[(hw_snf_3['cpcConcentration_x/cm3'] < 900000)]

hw_snf_1['CPC_cor'] = hw_snf_1['cpcConcentration_x/cm3'] - hw_bg_snf_1['cpcConcentration_x/cm3'].mean()
hw_snf_2['CPC_cor'] = hw_snf_2['cpcConcentration_x/cm3'] - hw_bg_snf_2['cpcConcentration_x/cm3'].mean()
hw_snf_3['CPC_cor'] = hw_snf_3['cpcConcentration_x/cm3'] - hw_bg_snf_3['cpcConcentration_x/cm3'].mean()
hw_snf_4['CPC_cor'] = hw_snf_4['cpcConcentration_x/cm3'] - hw_bg_snf_4['cpcConcentration_x/cm3'].mean()
```

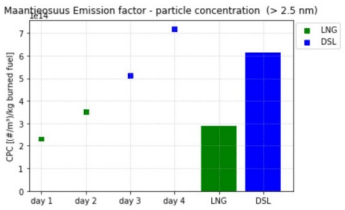
In [51]:

```
#CPC

#atm pressure = 101300 (N/M²)
#CO2 molar mass = 44.0095 g/mol
#r=molar gas factor = 8.3145 J/mol/K
co2_mol = 44.0095
p = 101300
r = 8.3145

#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + hw_snf_1['OutTemp'].mean()
T2 = K + hw_snf_2['OutTemp'].mean()
T3 = K + hw_snf_3['OutTemp'].mean()
T4 = K + hw_snf_4['OutTemp'].mean()

#Emission ratio for cpc. (CO2 ppm to ppm/cm³)
ER_cpc_1 = ((hw_snf_1['CPC_cor'].mean()*10**6)/hw_snf_1['CO2_cor'].mean())
ER_cpc_2 = ((hw_snf_2['CPC_cor'].mean()*10**6)/hw_snf_2['CO2_cor'].mean())
ER_cpc_3 = ((hw_snf_3['CPC_cor'].mean()*10**6)/hw_snf_3['CO2_cor'].mean())
ER_cpc_4 = ((hw_snf_4['CPC_cor'].mean()*10**6)/hw_snf_4['CO2_cor'].mean())


#Print Delta CO2
print('Delta CO2')
print('1 CO2:',hw_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',hw_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta CPC
print('Delta CPC')
print('1 CPC:',hw_snf_1['CPC_cor'].mean(),'#/cm^3')
print('2 CPC:',hw_snf_2['CPC_cor'].mean(),'#/cm^3')
print('3 CPC:',hw_snf_3['CPC_cor'].mean(),'#/cm^3')
print('4 CPC:',hw_snf_4['CPC_cor'].mean(),'#/cm^3')
print('1 CPC:',hw_snf_1['CPC_cor'].mean()*10**6 ,'#/m^3')
print('2 CPC:',hw_snf_2['CPC_cor'].mean()*10**6 ,'#/m^3')
print('3 CPC:',hw_snf_3['CPC_cor'].mean()*10**6 ,'#/m^3')
print('4 CPC:',hw_snf_4['CPC_cor'].mean()*10**6 ,'#/m^3')

#print emission ratio
print('CPC Emission ratio:')
print('Day1', ER_cpc_1)
print('Day2', ER_cpc_2)
print('Day3', ER_cpc_3)
print('Day4', ER_cpc_4)

#Emission factor

#Calculate emission factor for CPC
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel

#C30*(($B$4*F23)/($B$2*$B$3)*$E$3)*10^6

EF_cpc_1 = (ER_cpc_1 * (r*T1)/(p*co2_mol) * efng)*10**6
EF_cpc_2 = (ER_cpc_2 * (r*T2)/(p*co2_mol) * efng)*10**6
EF_cpc_3 = (ER_cpc_3 * (r*T3)/(p*co2_mol) * efdiesel)*10**6
EF_cpc_4 = (ER_cpc_4 * (r*T4)/(p*co2_mol) * efdiesel)*10**6


#Calculate cpc emission factor (mean) for each type of fuel
EF_cpc_lng = ((EF_cpc_1+EF_cpc_2)/2)
EF_cpc_dsl = ((EF_cpc_3+EF_cpc_4)/2)

#print cpc emission factor
print('CPC Emission Factor:')
print('Day1', EF_cpc_1, '#/kg burned fuel')
print('Day2', EF_cpc_2, '#/kg burned fuel')
print('Day3', EF_cpc_3, '#/kg burned fuel')
print('Day4', EF_cpc_4, '#/kg burned fuel')

#print cpc emission factor for lng & dsl
print('CPC Emission Factor for each fuel:')
print('natural gas:', EF_cpc_lng, '#/kg burned fuel')
print('diesel:', EF_cpc_dsl, '#/kg burned fuel')
```

```
Delta CO2
1 CO2: 53.33440309655569 ppm
2 CO2: 49.49403664079436 ppm
3 CO2: 77.37066886286259 ppm
3 CO2: 62.578998660720345 ppm
Delta CPC
1 CPC: 9270.390242039273 #/cm^3
2 CPC: 13158.889014547383 #/cm^3
3 CPC: 24057.849740798243 #/cm^3
4 CPC: 27627.306633140724 #/cm^3
1 CPC: 9270390242.039274 #/m^3
2 CPC: 13158889014.547384 #/m^3
3 CPC: 24057849740.798244 #/m^3
4 CPC: 27627306633.140724 #/m^3
CPC Emission ratio:
Day1 173816330.61977497
Day2 265868171.35261628
Day3 310942765.44823635
Day4 441478886.278215
CPC Emission Factor:
Day1 230369282437306.53 #/kg burned fuel
Day2 353171642078949.25 #/kg burned fuel
Day3 511481783441903.8 #/kg burned fuel
Day4 716270000311933.2 #/kg burned fuel
CPC Emission Factor for each fuel:
natural gas: 291770462258127.9 #/kg burned fuel
diesel: 613875891876918.5 #/kg burned fuel
```

In [67]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_cpc_1,EF_cpc_2]
lng_c = EF_cpc_lng
dsl = [EF_cpc_3,EF_cpc_4]
dsl_c = EF_cpc_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('CPC [(#/m³)/kg burned fuel]')
ax.set_title('Maantieosuus Emission factor - particle concentration  (> 2.5 nm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/hw_CPC_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



In [53]:

```
#Eeps
hw_snf_1['Eeps_cor'] = hw_snf_1['Eeps_Total_Conc_x/cm3']-hw_bg_snf_1['Eeps_Total_Conc_x/cm3'].mean()
hw_snf_2['Eeps_cor'] = hw_snf_2['eepsConcentration_x/cm3']-hw_bg_snf_2['eepsConcentration_x/cm3'].mean()
hw_snf_3['Eeps_cor'] = hw_snf_3['eepsConcentration_x/cm3']-hw_bg_snf_3['eepsConcentration_x/cm3'].mean()
hw_snf_4['Eeps_cor'] = hw_snf_4['eepsConcentration_x/cm3']-hw_bg_snf_4['eepsConcentration_x/cm3'].mean()
```

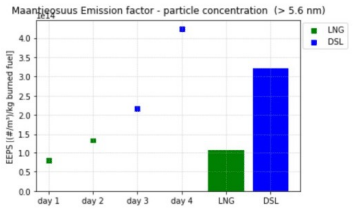
In [65]:

```
#EEPS

#atm pressure = 101300 (N/M²)
#CO2 molar mass = 44.0095 g/mol
#r=molar gas factor = 8.3145 J/mol/K
co2_mol = 44.0095
p = 101300
r = 8.3145

#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + hw_snf_1['OutTemp'].mean()
T2 = K + hw_snf_2['OutTemp'].mean()
T3 = K + hw_snf_3['OutTemp'].mean()
T4 = K + hw_snf_4['OutTemp'].mean()

#Emission ratio for cpc. (CO2 ppm to ppm/cm³)
ER_eeps_1 = ((hw_snf_1['Eeps_cor'].mean()*10**6)/hw_snf_1['CO2_cor'].mean())
ER_eeps_2 = ((hw_snf_2['Eeps_cor'].mean()*10**6)/hw_snf_2['CO2_cor'].mean())
ER_eeps_3 = ((hw_snf_3['Eeps_cor'].mean()*10**6)/hw_snf_3['CO2_cor'].mean())
ER_eeps_4 = ((hw_snf_4['Eeps_cor'].mean()*10**6)/hw_snf_4['CO2_cor'].mean())


#Print Delta CO2
print('Delta CO2')
print('1 CO2:',hw_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',hw_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta CPC
print('Delta Eeps')
print('1 eeps:',hw_snf_1['Eeps_cor'].mean(),'#/cm^3')
print('2 eeps:',hw_snf_2['Eeps_cor'].mean(),'#/cm^3')
print('3 eeps:',hw_snf_3['Eeps_cor'].mean(),'#/cm^3')
print('4 eeps:',hw_snf_4['Eeps_cor'].mean(),'#/cm^3')
print('1 eeps:',hw_snf_1['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('2 eeps:',hw_snf_2['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('3 eeps:',hw_snf_3['Eeps_cor'].mean()*10**6 ,'#/m^3')
print('4 eeps:',hw_snf_4['Eeps_cor'].mean()*10**6 ,'#/m^3')

#print emission ratio
print('CPC Emission ratio:')
print('Day1', ER_eeps_1)
print('Day2', ER_eeps_2)
print('Day3', ER_eeps_3)
print('Day4', ER_eeps_4)

#Emission factor

#Calculate emission factor for CPC
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel



EF_eeps_1 = (ER_eeps_1 * (r*T1)/(p*co2_mol) * efng)*10**6
EF_eeps_2 = (ER_eeps_2 * (r*T2)/(p*co2_mol) * efng)*10**6
EF_eeps_3 = (ER_eeps_3 * (r*T3)/(p*co2_mol) * efdiesel)*10**6
EF_eeps_4 = (ER_eeps_4 * (r*T4)/(p*co2_mol) * efdiesel)*10**6


#Calculate cpc emission factor (mean) for each type of fuel
EF_eeps_lng = ((EF_eeps_1+EF_eeps_2)/2)
EF_eeps_dsl = ((EF_eeps_3+EF_eeps_4)/2)

#print cpc emission factor
print('eeps Emission Factor:')
print('Day1', EF_eeps_1, '#/kg burned fuel')
print('Day2', EF_eeps_2, '#/kg burned fuel')
print('Day3', EF_eeps_3, '#/kg burned fuel')
print('Day4', EF_eeps_4, '#/kg burned fuel')

#print cpc emission factor for lng & dsl
print('eeps Emission Factor for each fuel:')
print('natural gas:', EF_eeps_lng, '#/kg burned fuel')
print('diesel:', EF_eeps_dsl, '#/kg burned fuel')
```

```
Delta CO2
1 CO2: 53.33440309655569 ppm
2 CO2: 49.49403664079436 ppm
3 CO2: 77.37066886286259 ppm
3 CO2: 62.578998660720345 ppm
Delta Eeps
1 eeps: 3217.873712675773 #/cm^3
2 eeps: 4939.9231944613475 #/cm^3
3 eeps: 10237.179050813334 #/cm^3
4 eeps: 16335.910500664597 #/cm^3
1 eeps: 3217873712.6757727 #/m^3
2 eeps: 4939923194.461348 #/m^3
3 eeps: 10237179050.813335 #/m^3
4 eeps: 16335910500.664597 #/m^3
CPC Emission ratio:
Day1 60333921.93121932
Day2 99808452.28513297
Day3 132313436.1027492
Day4 261044613.21331975
eeps Emission Factor:
Day1 79964191237748.84 #/kg burned fuel
Day2 132582681136916.02 #/kg burned fuel
Day3 217647489478017.06 #/kg burned fuel
Day4 423527446043934.5 #/kg burned fuel
eeps Emission Factor for each fuel:
natural gas: 1062734361873332.44 #/kg burned fuel
diesel: 320587467760975.75 #/kg burned fuel
```

In [66]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_eeps_1,EF_eeps_2]
lng_c = EF_eeps_lng
dsl = [EF_eeps_3,EF_eeps_4]
dsl_c = EF_eeps_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('EEPS [(#/m³)/kg burned fuel]')
ax.set_title('Maantieosuus Emission factor - particle concentration  (> 5.6 nm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/hw_eeps_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



In [57]:

```
#OPS
hw_snf_1['Ops_cor'] = (hw_snf_1['Total_Conc.x/cm3']-hw_bg_snf_1['Total_Conc.x/cm3'].mean())
hw_snf_2['Ops_cor'] = (hw_snf_2['Total_Conc.x/cm3']-hw_bg_snf_2['Total_Conc.x/cm3'].mean())
hw_snf_3['Ops_cor'] = (hw_snf_3['Total_Conc.x/cm3']-hw_bg_snf_3['Total_Conc.x/cm3'].mean())
hw_snf_4['Ops_cor'] = (hw_snf_4['Total_Conc.x/cm3']-hw_bg_snf_4['Total_Conc.x/cm3'].mean())
```

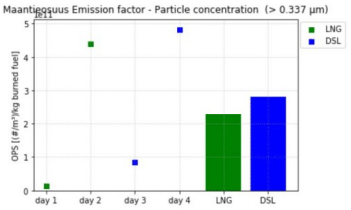
In [58]:

```
#OPS

#atm pressure = 101300 (N/M²)
#CO2 molar mass = 44.0095 g/mol
#r=molar gas factor = 8.3145 J/mol/K
co2_mol = 44.0095
p = 101300
r = 8.3145

#Molar mass
#NO molar mass = 30.0061 g/mol
#NO2 molar mass = 46.0055 g/mol
#CO2 molar mass = 44.0095 g/mol

co2_mol = 44.0095 #g/mol
p = 101300 #N/m² (Pa)
r = 8.3145 #J/mol/K
K = 273.15 # Kelvin


#Temp Celsius to Kelvin
T1 = K + hw_snf_1['OutTemp'].mean()
T2 = K + hw_snf_2['OutTemp'].mean()
T3 = K + hw_snf_3['OutTemp'].mean()
T4 = K + hw_snf_4['OutTemp'].mean()

#Emission ratio for ops. (CO2 ppm to ppm/cm³)
ER_ops_1 = ((hw_snf_1['Ops_cor'].mean()*10**6)/hw_snf_1['CO2_cor'].mean())
ER_ops_2 = ((hw_snf_2['Ops_cor'].mean()*10**6)/hw_snf_2['CO2_cor'].mean())
ER_ops_3 = ((hw_snf_3['Ops_cor'].mean()*10**6)/hw_snf_3['CO2_cor'].mean())
ER_ops_4 = ((hw_snf_4['Ops_cor'].mean()*10**6)/hw_snf_4['CO2_cor'].mean())


#Print Delta CO2
print('Delta CO2')
print('1 CO2:',hw_snf_1['CO2_cor'].mean() ,'ppm')
print('2 CO2:',hw_snf_2['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_3['CO2_cor'].mean() ,'ppm')
print('3 CO2:',hw_snf_4['CO2_cor'].mean() ,'ppm')
#Print Delta ops
print('Delta OPS')
print('1 OPS:',hw_snf_1['Ops_cor'].mean(),'#/cm^3')
print('2 OPS:',hw_snf_2['Ops_cor'].mean(),'#/cm^3')
print('3 OPS:',hw_snf_3['Ops_cor'].mean(),'#/cm^3')
print('4 OPS:',hw_snf_4['Ops_cor'].mean(),'#/cm^3')
print('1 OPS:',hw_snf_1['Ops_cor'].mean()*10**6 ,'#/m^3')
print('2 OPS:',hw_snf_2['Ops_cor'].mean()*10**6 ,'#/m^3')
print('3 OPS:',hw_snf_3['Ops_cor'].mean()*10**6 ,'#/m^3')
print('4 OPS:',hw_snf_4['Ops_cor'].mean()*10**6 ,'#/m^3')

#print emission ratio
print('CPC Emission ratio:')
print('Day1', ER_ops_1)
print('Day2', ER_ops_2)
print('Day3', ER_ops_3)
print('Day4', ER_ops_4)

#Emission factor

#Calculate emission factor for ops
efdiesel = 3160 #g/kg fuel
efng = 2540 #g/kg fuel



EF_ops_1 = (ER_ops_1 * (r*T1)/(p*co2_mol) * efng)*10**6
EF_ops_2 = (ER_ops_2 * (r*T2)/(p*co2_mol) * efng)*10**6
EF_ops_3 = (ER_ops_3 * (r*T3)/(p*co2_mol) * efdiesel)*10**6
EF_ops_4 = (ER_ops_4 * (r*T4)/(p*co2_mol) * efdiesel)*10**6


#Calculate ops emission factor (mean) for each type of fuel
EF_ops_lng = ((EF_ops_1+EF_ops_2)/2)
EF_ops_dsl = ((EF_ops_3+EF_ops_4)/2)

#print ops emission factor
print('OPS Emission Factor:')
print('Day1', EF_ops_1, '#/kg burned fuel')
print('Day2', EF_ops_2, '#/kg burned fuel')
print('Day3', EF_ops_3, '#/kg burned fuel')
print('Day4', EF_ops_4, '#/kg burned fuel')

#print ops emission factor for lng & dsl
print('OPS Emission Factor for each fuel:')
print('natural gas:', EF_ops_lng, '#/kg burned fuel')
print('diesel:', EF_ops_dsl, '#/kg burned fuel')
```

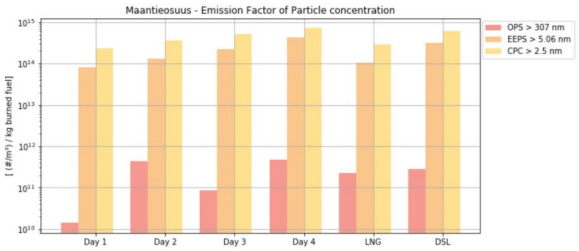
```
Delta CO2
1 CO2: 53.33440309655569 ppm
2 CO2: 49.49403664079436 ppm
3 CO2: 77.37066886286259 ppm
3 CO2: 62.578998660720345 ppm
Delta OPS
1 OPS: 0.5578240141407085 #/cm^3
2 OPS: 16.380369558778398 #/cm^3
3 OPS: 3.9621747475689 31 #/cm^3
4 OPS: 18.536551514420257 #/cm^3
1 OPS: 557824.0141407085 #/m^3
2 OPS: 16380369.558778398 #/m^3
3 OPS: 3962174.747568931 #/m^3
4 OPS: 18536551.514420256 #/m^3
CPC Emission ratio:
Day1 10458.990478075351
Day2 330956.4276937
Day3 51210.29462200693
Day4 296210.420606415
OPS Emission Factor:
Day1 13861931861.416943 #/kg burned fuel
Day2 439633012219.9748 #/kg burned fuel
Day3 84237794650.37883 #/kg burned fuel
Day4 480581619313.15955 #/kg burned fuel
OPS Emission Factor for each fuel:
natural gas: 226747472040.69586 #/kg burned fuel
diesel: 282409706981.76917 #/kg burned fuel
```

In [59]:

```
#ops

fig = plt.figure()
ax = fig.add_subplot(111)

lng = [EF_ops_1,EF_ops_2]
lng_c = EF_ops_lng
dsl = [EF_ops_3,EF_ops_4]
dsl_c = EF_ops_dsl
bars = ('day 1','day 2')
bars1= ('day 3','day 4')
bars2 = ('LNG')
bars3 = ('DSL')


#plt.ylim(-0,0.04)


# Create horizontal bars
rects1 = ax.scatter(bars,lng, marker='s',color='g', label = 'LNG')
rects2 = ax.scatter(bars1,dsl, marker='s',color='b', label = 'DSL')
rects3 = ax.bar(bars2,lng_c,color='g')
rects4 = ax.bar(bars3,dsl_c,color='b')

#ax.set_ylim(0,50)
ax.set_ylabel('OPS [(#/m³)/kg burned fuel]')
ax.set_title('Maantieosuus Emission factor - Particle concentration  (> 0.337 µm)', loc='center',va='bottom')
plt.grid(alpha=0.7, linewidth=0.5, linestyle='--')



def autolabel(rects):
#     """
#     Attach a text label above each bar displaying its height
#     """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.01*height,
                '%d' % float(height),
                ha='center', va='bottom')

#autolabel(rects3)
#autolabel(rects4)
plt.legend(bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
#plt.yscale('log')

plt.savefig('/home/sami/Documents/emissionfactor/hw_OPS_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



In [68]:
```
jep = pd.DataFrame(columns = ['EF_ops', 'EF_eeps', 'EF_cpc', 'day', 'fuel'])
jep['EF_ops'] = [EF_ops_1,EF_ops_2,EF_ops_3,EF_ops_4,EF_ops_lng,EF_ops_dsl]
jep['EF_eeps'] = [EF_eeps_1,EF_eeps_2,EF_eeps_3,EF_eeps_4,EF_eeps_lng,EF_eeps_dsl]
jep['EF_cpc'] = [EF_cpc_1,EF_cpc_2,EF_cpc_3,EF_cpc_4,EF_cpc_lng,EF_cpc_dsl]
jep['day'] = ['Day 1','Day 2','Day 3','Day 4','LNG','DSL']

#jep = pd.DataFrame(raw_data, columns = ['EF_ops', 'EF_eeps', 'EF_cpc', 'day'])



# Setting the positions and width for the bars
pos = list(range(len(jep['EF_cpc'])))
width = 0.25

# Plotting the bars
fig, ax = plt.subplots(figsize=(10,5))

# Create a bar with pre_score data,
# in position pos,
plt.bar(pos,
        #using df['pre_score'] data,
        jep['EF_ops'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#EE3224',
        # with label the first value in first_name
        label=jep['day'][0])

# Create a bar with mid_score data,
# in position pos + some width buffer,
plt.bar([p + width for p in pos],
        #using df['mid_score'] data,
        jep['EF_eeps'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#F78F1E',
        # with label the second value in first_name
        label=jep['day'][1])

# Create a bar with post_score data,
# in position pos + some width buffer,
plt.bar([p + width*2 for p in pos],
        #using df['post_score'] data,
        jep['EF_cpc'],
        # of width
        width,
        # with alpha 0.5
        alpha=0.5,
        # with color
        color='#FFC222',
        # with label the third value in first_name
        label=jep['day'][2])

# Create a bar with post_score data,
# in position pos + some width buffer,


# Set the y axis label
ax.set_ylabel('[ (#/m³) / kg burned fuel]')

# Set the chart's title
ax.set_title('Maantieosuus - Emission Factor of Particle concentration')

# Set the position of the x ticks
ax.set_xticks([p + 1.5 * width for p in pos])

# Set the labels for the x ticks
ax.set_xticklabels(jep['day'])

# Setting the x-axis and y-axis limits
#plt.xlim(min(pos)-width, max(pos)+width*4)
#plt.ylim([0, max(jep['pre_score'] + df['mid_score'] + df['post_score'])] )

# Adding the legend and showing the plot
plt.legend(['OPS > 307 nm', 'EEPS > 5.06 nm', 'CPC > 2.5 nm'],bbox_to_anchor=(1, 1), loc=2, borderaxespad=0.3)
plt.grid()
plt.yscale('log')

plt.savefig('/home/sami/Documents/emissiontactor/hw_ntot_EF.png',dpi=300,facecolor='w', transparent=True,bbox_inches='tight')
```



Toisto-osuuden toistojen valinta

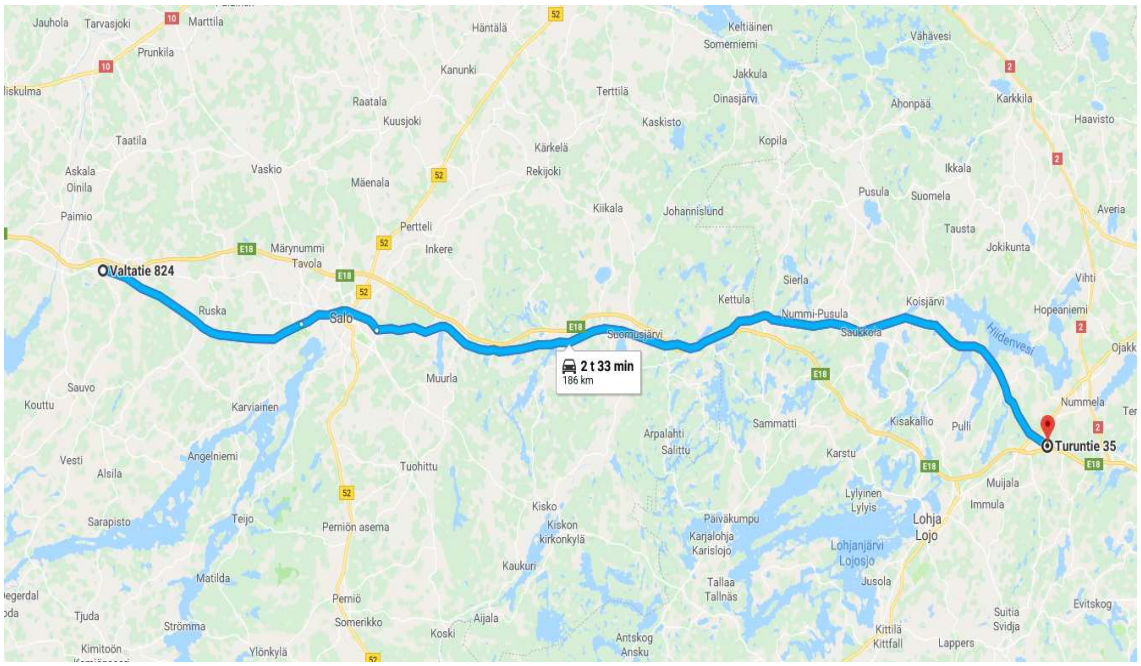
Taulukot toisto-osuuden Nox-keskiarvoista. Punaisella merkityt ovat jätetty pois tuloksista, sinisellä merkityt on valittu tuloksiin ja vihreällä merkityt ovat mitattuja taustapitoisuuksia. Keltaisella merkitty on osuudella mitattu taustapitoisuus, mutta taustanmittausongelmien johdosta käytetään samaa taustapitoisuutta, joka on mitattu maantieosuudelta.

Kasvihuoneilmiö

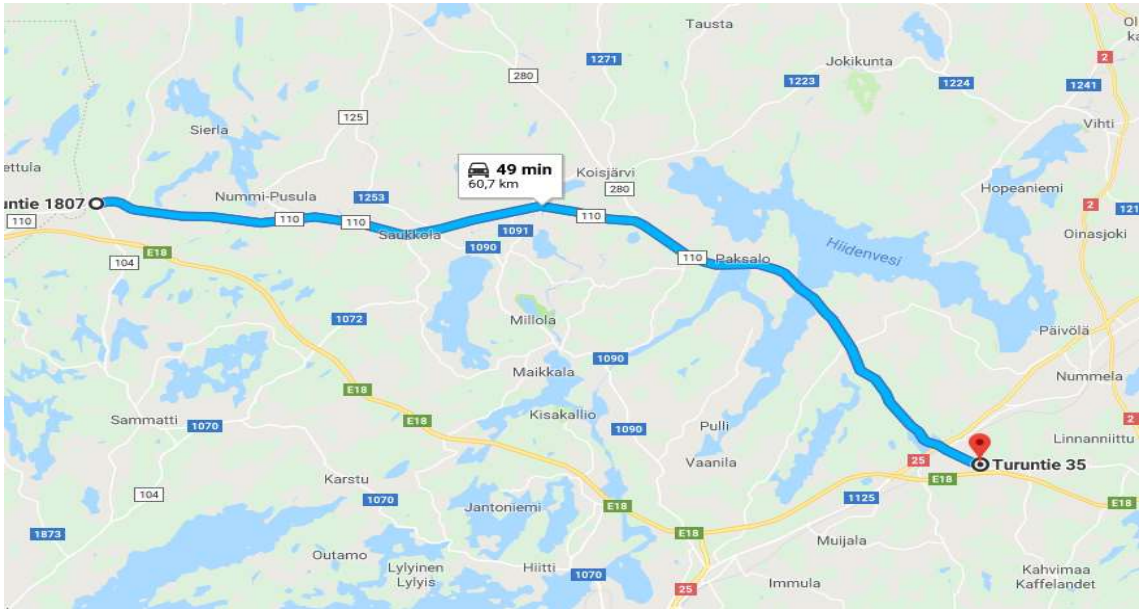
	Sniffer – Nox ppb							
Lap	**Day 1 - East**	**Day 1 - West**	**Day 2 - East**	**Day 2 - West**	**Day 3 - East**	**Day 3 - West**	**Day 4 - East**	**Day 4 - West**
1	23.6289157	6.6844	38.8983871	33.6672	69.9112903	198.291165	23.8568	130.454762
2	13.5313492	36.0533865	79.8495935	32.2358566	52.4468504	47.4211765	31.409375	21.4784091
3	12.7592	9.1734127	41.9120482	43.7176707	40.3346304	54.3996078	54.1844961	30.6790698
4	12.0805668	8.0332	61.5562753	35.3246964	82.7823529	45.6692308	43.110757	37.9233463
5	1.338 18182	0.96531532	33.6704453	30.7621514	65.2910156	39.9546154	50.1614173	36.9326848
6			89.9826613	28.3988	60.0460317	32.7064394	78.1063492	41.7123552
7			18.8773663	37.1096386	32.6220339	32.1741313	16.506383	48.2176
8			40.6377273	36.6054054	17.8261261	13.2146341	8.9456	10.0088889
			20.5765	20.5765				

Truck – Nox ppm							
Day 1 – East	**Day 1 - West**	**Day 2 - East**	**Day 2 - West**	**Day 3 - East**	**Day 3 - West**	**Day 4 - East**	**Day 4 - West**
50.4047826	75.5872807	106.095872	65.8945607	23.7887324	13.8311688	12.5539906	19.1285
55.0595652	73.226	58.8938865	73.1734783	19.8709091	16.7414747	14.7516854	8.30266667
67.295045	73.465873	51.2666667	92.2230769	17.1591549	15.1069307	15.6864706	10.9482456
70.808547	59.3927711	51.6261468	92.8008368	16.1564356	14.013615	18.9876777	9.66604651
		35.7668831	106.527397	19.720362	18.9415179	25.1658228	11.6109589
		51.5171806	62.521875	22.8474747	12.8528571	29.7881988	15.78159
		51.6703297	72.8878924	13.2978142	11.6665094	9.82171429	14.6663677

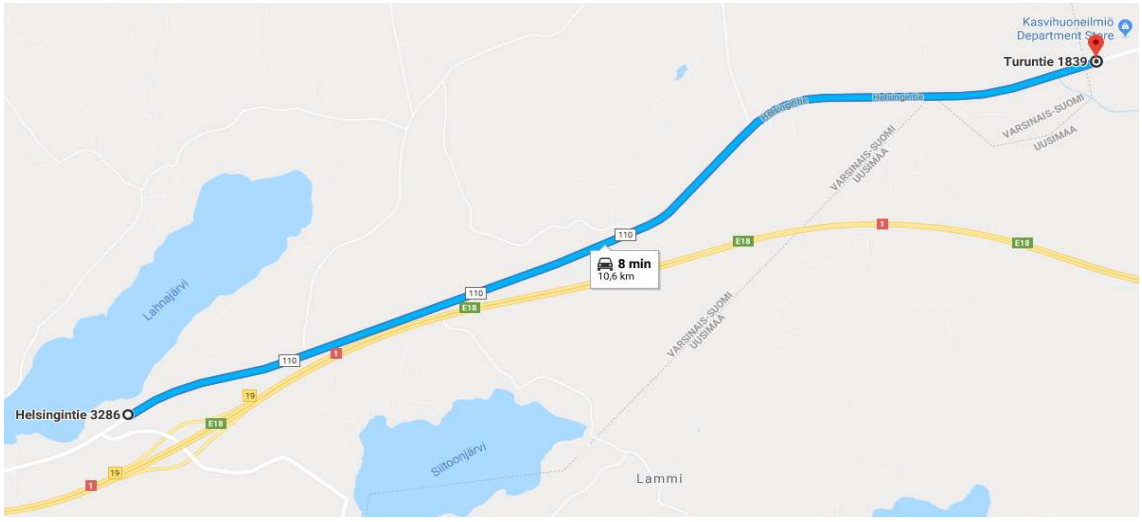
Mittausreitin jako



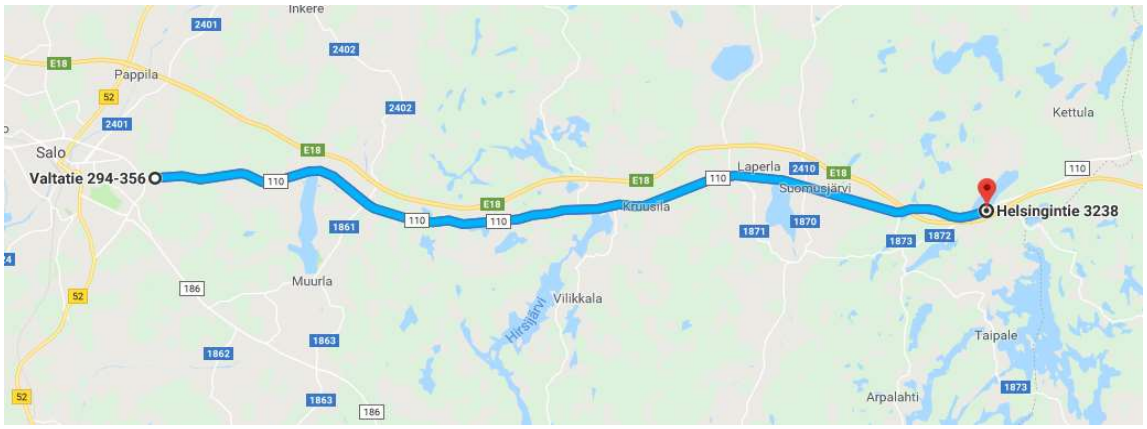
Koko reitti



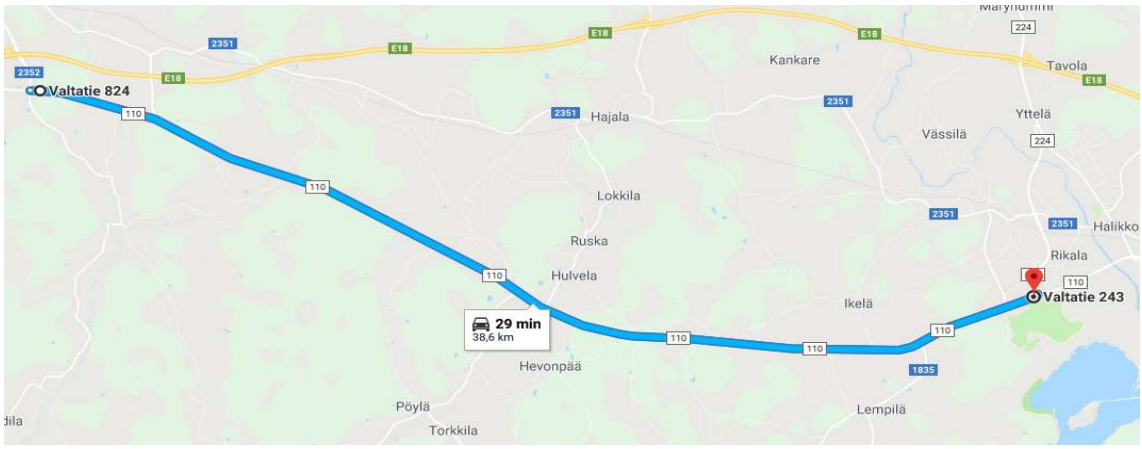
Turuntie – kahvio Kasvihuoneilmiö – Turuntie



Toisto-osuustoistot:

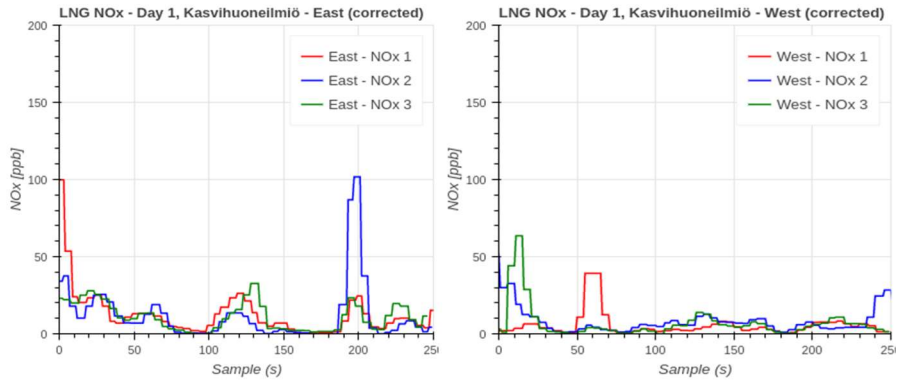


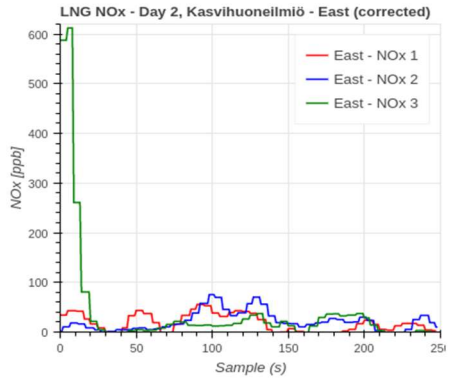
Kahvio Kasvihuoneilmiö–Salo–Kasvihuoneilmiö

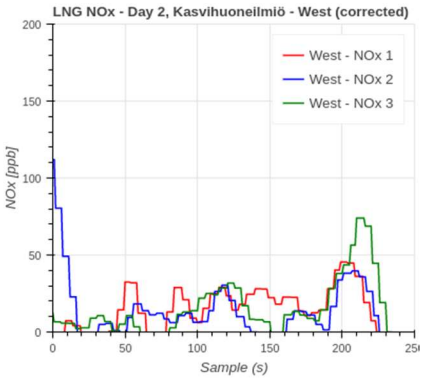


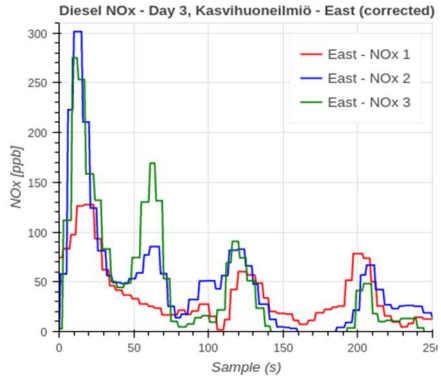
Salo–Paimio–Salo

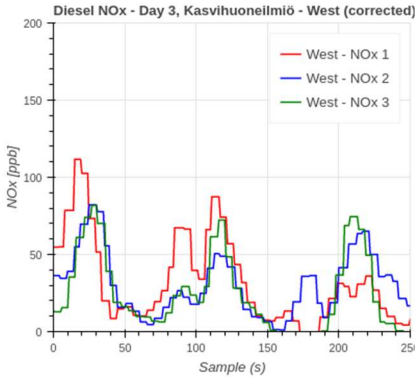
Toisto-osuuden typenokisidipäästöt eriteltyinä

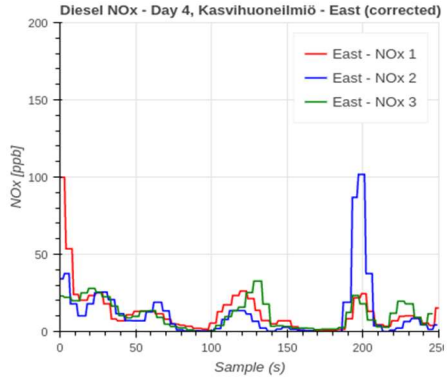


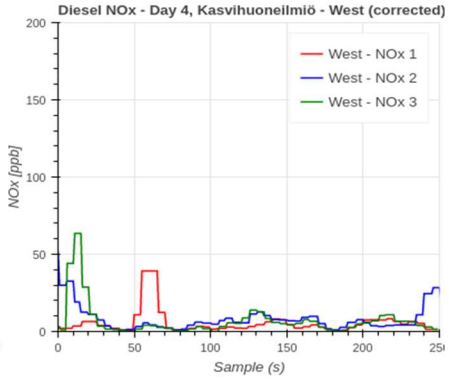


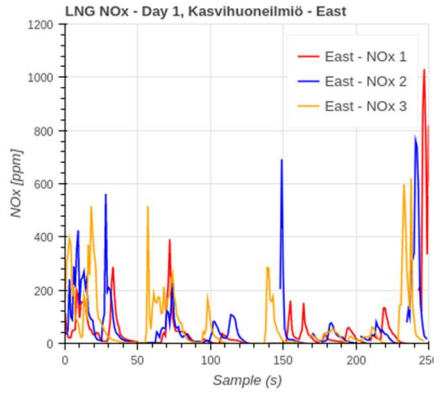


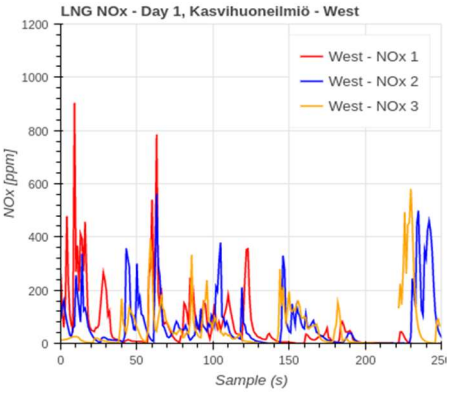

Diesel NOx - Day 3, Kasvihuoneilmiö - East (corrected)
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppb]
Sample (s)

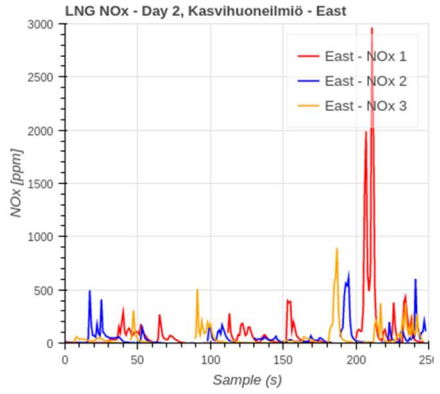

Diesel NOx - Day 3, Kasvihuoneilmiö - West (corrected)
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppb]
Sample (s)

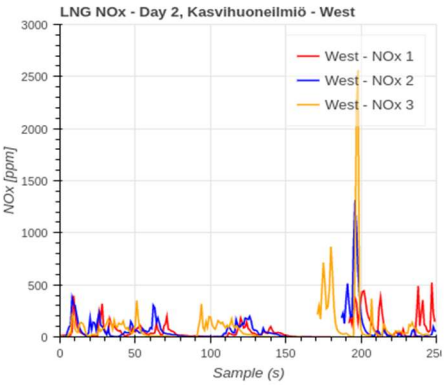

Diesel NOx - Day 4, Kasvihuoneilmiö - East (corrected)
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppb]
Sample (s)

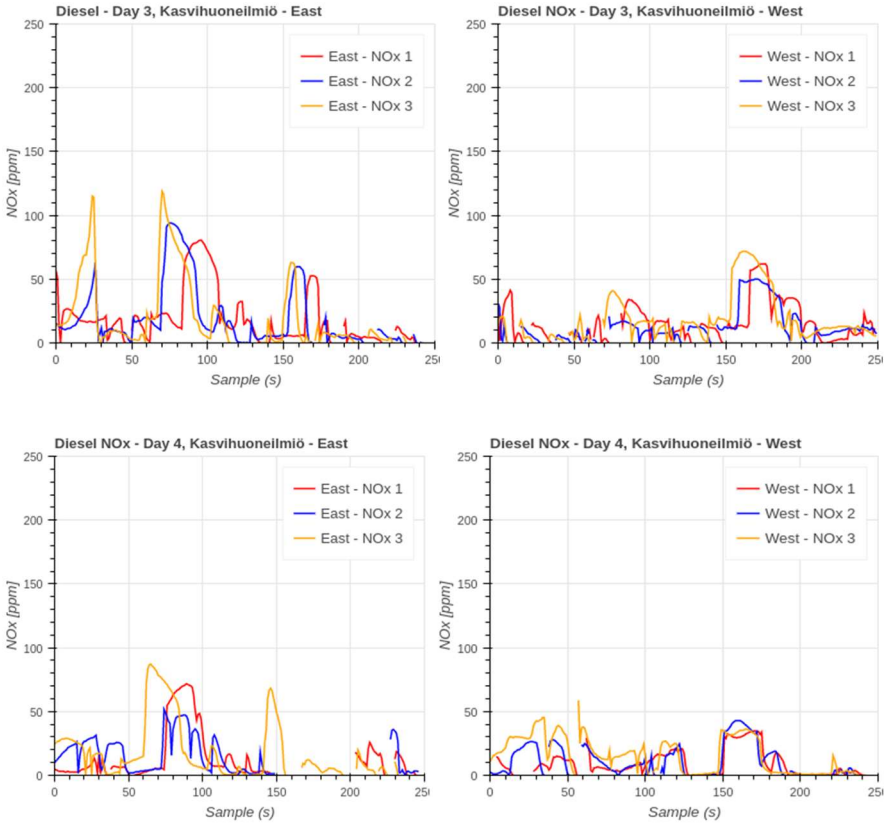

Diesel NOx - Day 4, Kasvihuoneilmiö - West (corrected)
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppb]
Sample (s)


LNG NOx - Day 1, Kasvihuoneilmiö - East
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppm]
Sample (s)

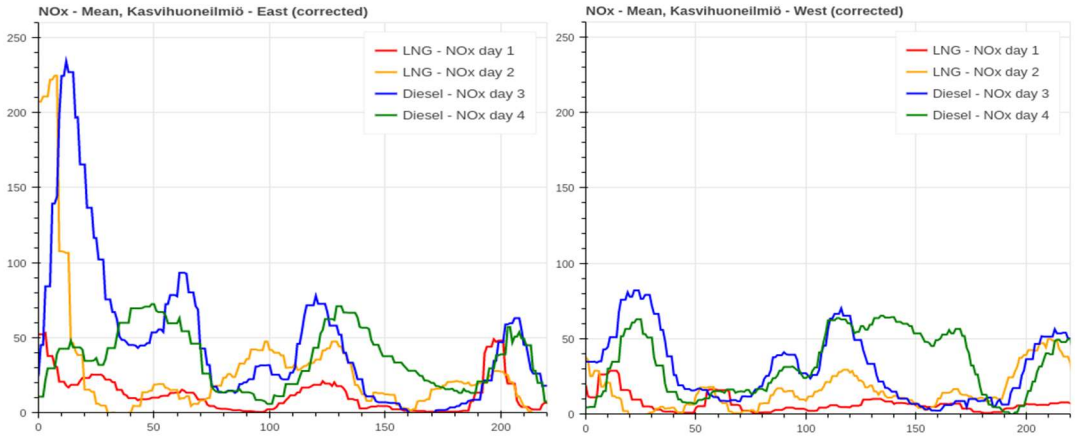

LNG NOx - Day 1, Kasvihuoneilmiö - West
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppm]
Sample (s)

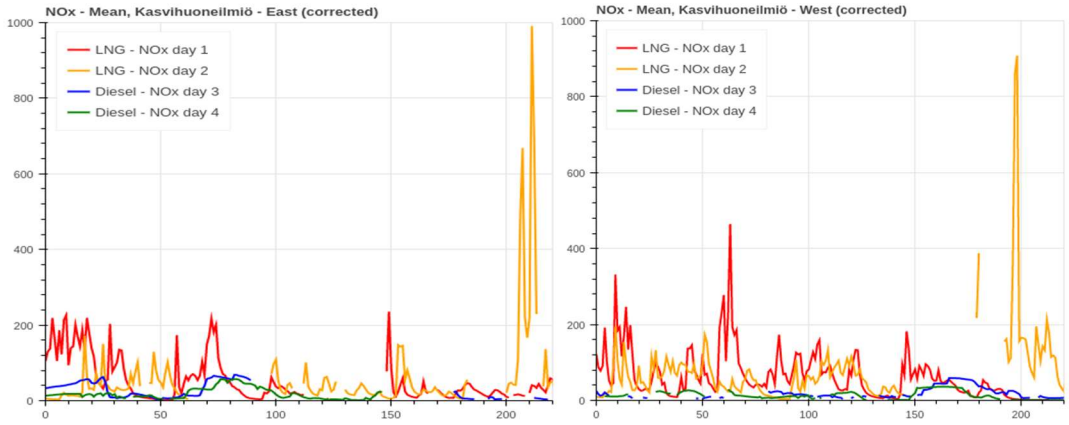

LNG NOx - Day 2, Kasvihuoneilmiö - East
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppm]
Sample (s)


LNG NOx - Day 2, Kasvihuoneilmiö - West
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppm]
Sample (s)


Diesel - Day 3, Kasvihuoneilmiö - East
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppm]
Sample (s)
Diesel NOx - Day 3, Kasvihuoneilmiö - West
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppm]
Sample (s)
Diesel NOx - Day 4, Kasvihuoneilmiö - East
East - NOx 1
East - NOx 2
East - NOx 3
NOx [ppm]
Sample (s)
Diesel NOx - Day 4, Kasvihuoneilmiö - West
West - NOx 1
West - NOx 2
West - NOx 3
NOx [ppm]
Sample (s)

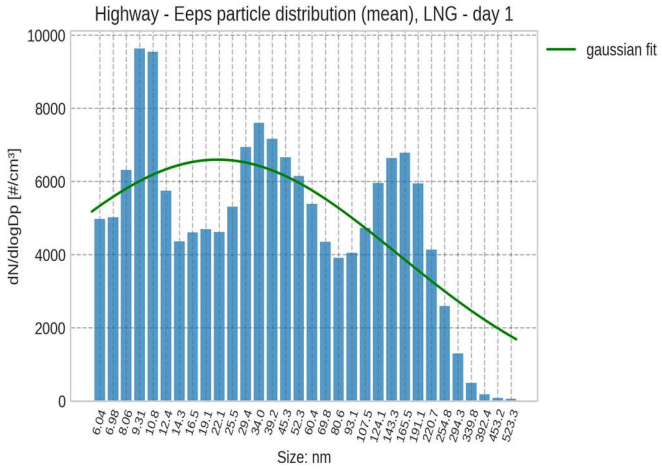
Toisto-osuuden typpioksidipäästöt päiväkohtaisesti eriteltynä.

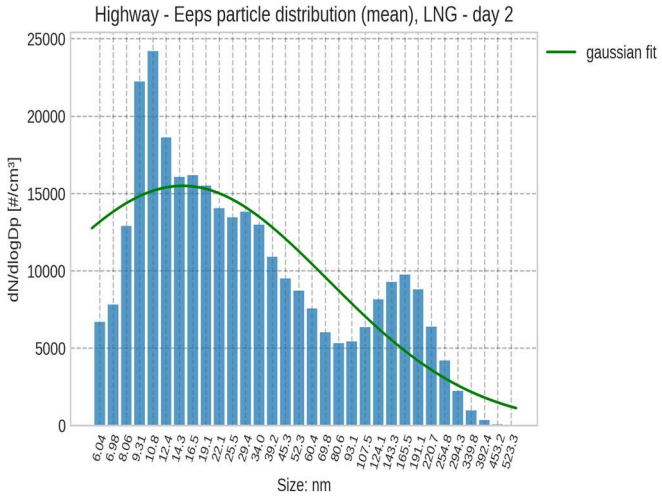


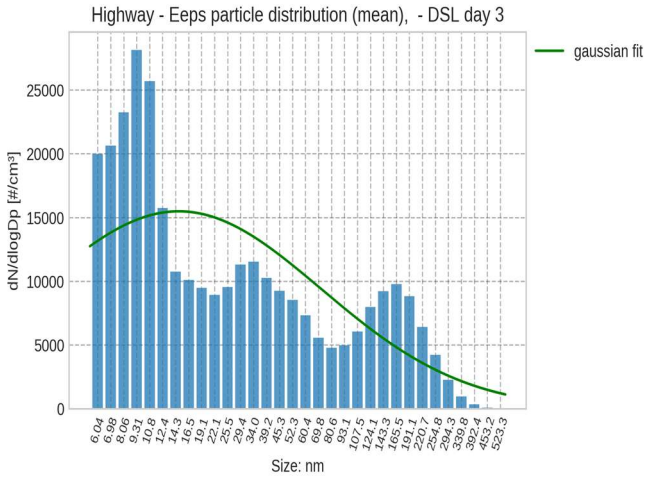


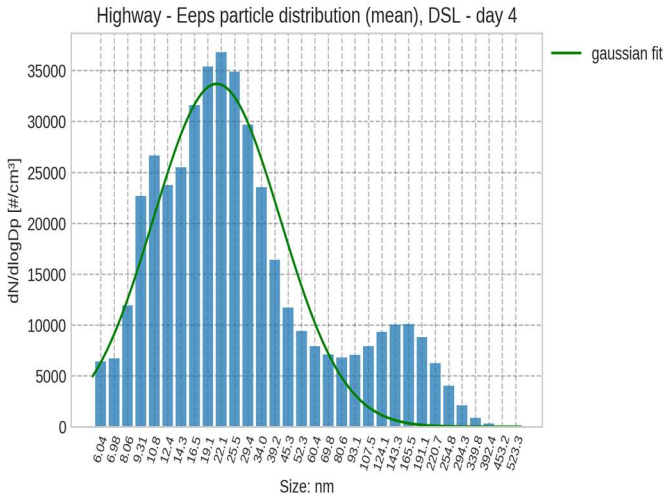
Hiukkaskonsentraatiokeskiarvot maantieosuudella

EEPS maantie.

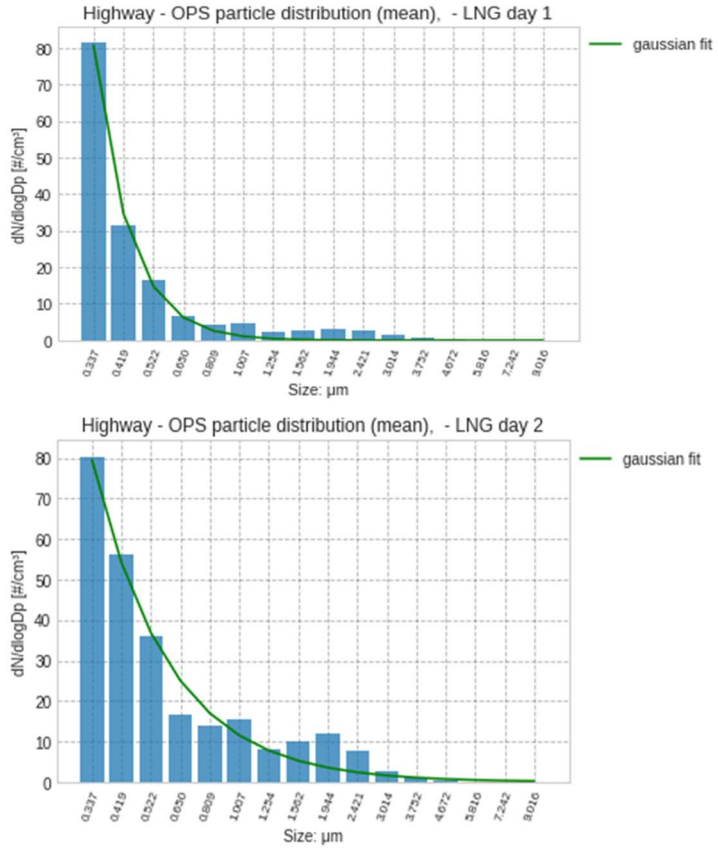


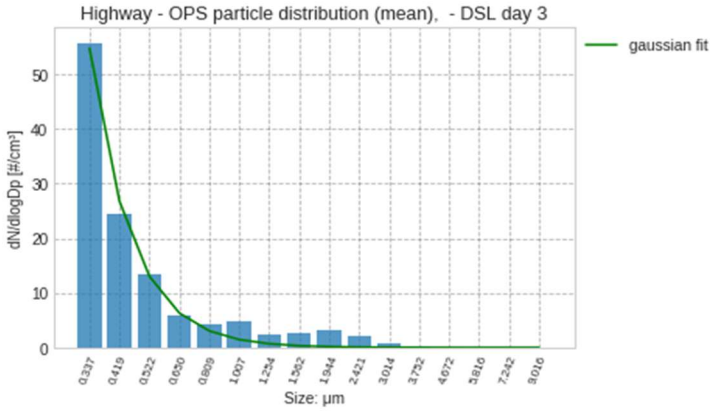


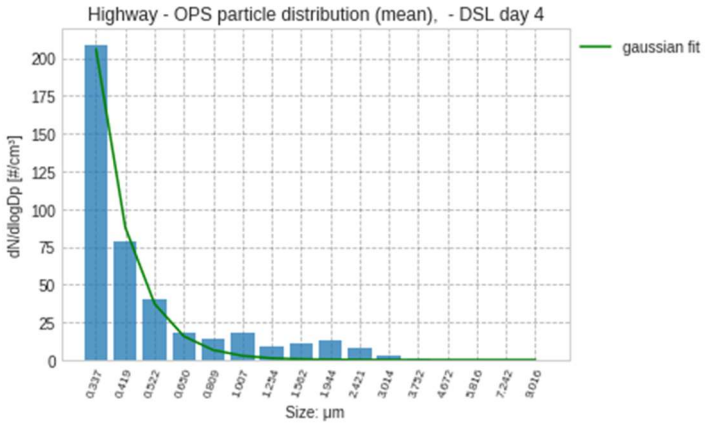

Highway - Eeps particle distribution (mean), - DSL day 3
gaussian fit
dN/dlogDp [#/cm³]
0
5000
10000
15000
20000
25000
6.04
6.98
8.06
9.31
10.8
12.4
14.3
16.5
19.1
22.1
25.5
29.4
34.0
39.2
45.3
52.3
60.4
69.8
80.6
93.1
107.5
124.1
143.3
165.5
191.1
220.7
254.8
294.3
339.8
392.4
453.2
523.3
Size: nm


Highway - Eeps particle distribution (mean), DSL - day 4
gaussian fit
dN/dlogDp [#/cm³]
0
5000
10000
15000
20000
25000
30000
35000
6.04
6.98
8.06
9.31
10.8
12.4
14.3
16.5
19.1
22.1
25.5
29.4
34.0
39.2
45.3
52.3
60.4
69.8
80.6
93.1
107.5
124.1
143.3
165.5
191.1
220.7
254.8
294.3
339.8
392.4
453.2
523.3
Size: nm

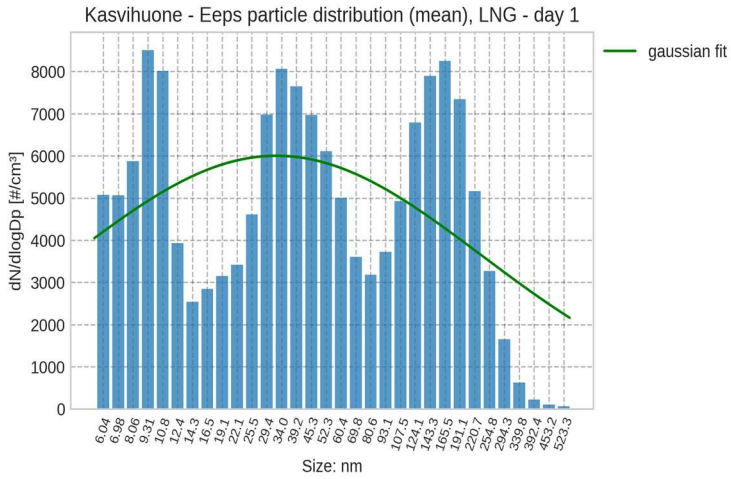
OPS maantie.

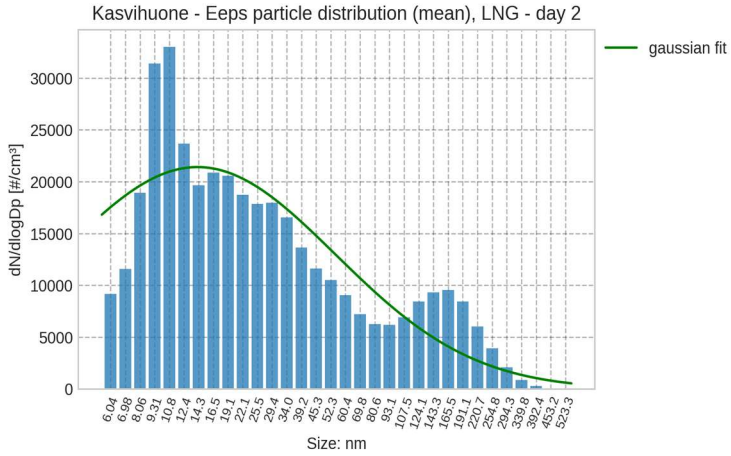

Highway - OPS particle distribution (mean), - LNG day 1
gaussian fit
dN/dlogDp [#/cm³]
Size: µm
Highway - OPS particle distribution (mean), - LNG day 2
gaussian fit
dN/dlogDp [#/cm³]
Size: µm

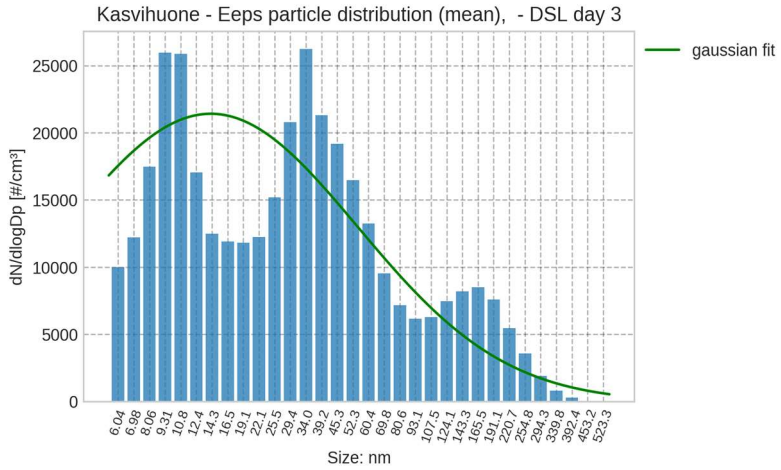

Highway - OPS particle distribution (mean), - DSL day 3
gaussian fit
dN/dlogDp [#/cm³]
0
10
20
30
40
50
0.337
0.419
0.522
0.650
0.809
1.007
1.254
1.562
1.944
2.421
3.014
3.752
4.672
5.816
7.242
9.016
Size: µm

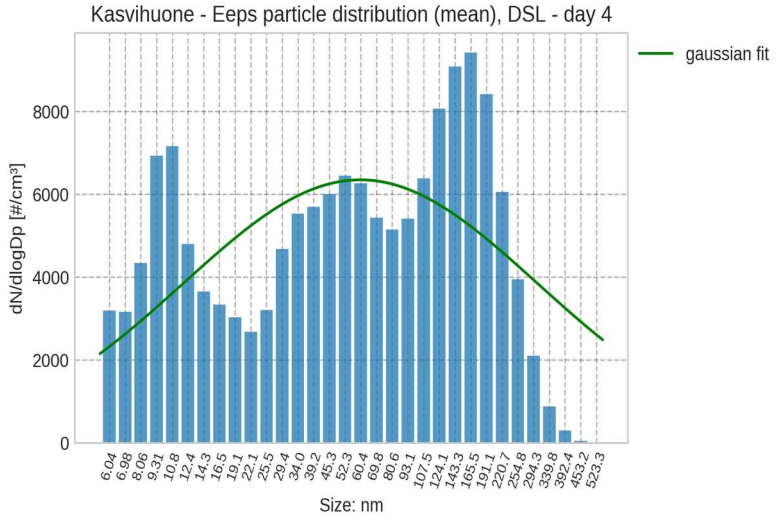

Highway - OPS particle distribution (mean), - DSL day 4
gaussian fit
dN/dlogDp [#/cm³]
0
25
50
75
100
125
150
175
200
0.337
0.419
0.522
0.650
0.809
1.007
1.254
1.562
1.944
2.421
3.014
3.752
4.672
5.816
7.242
9.016
Size: µm

EEPS toisto-osuus

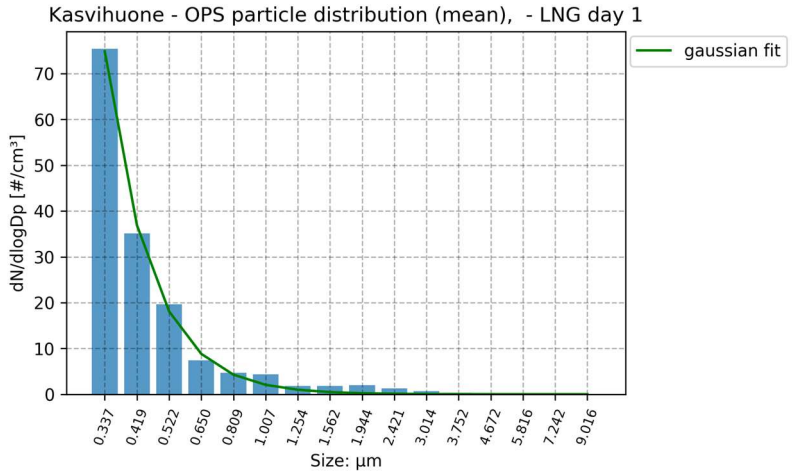


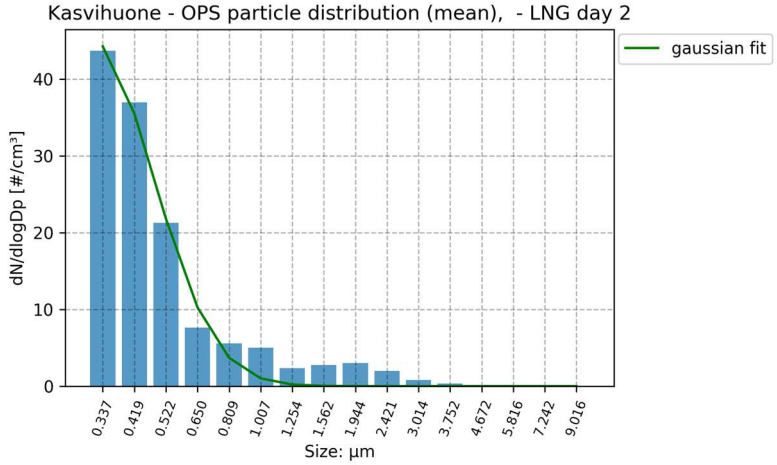


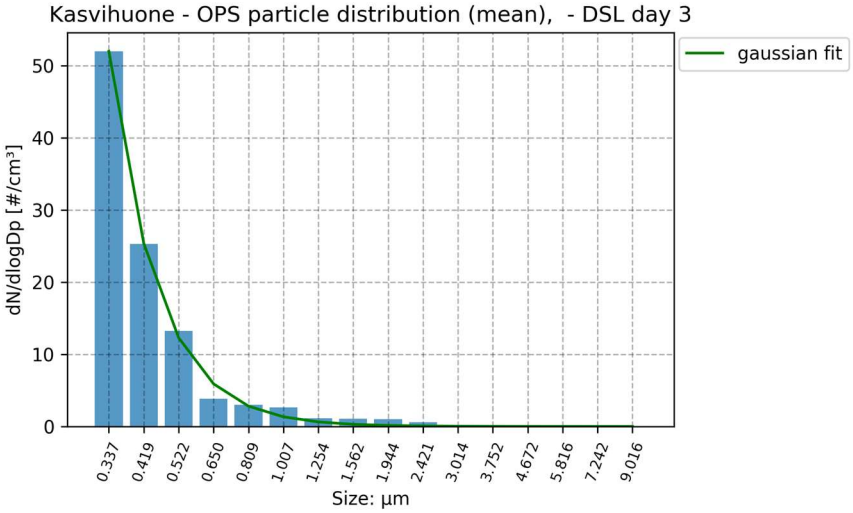

Kasvihuone - Eeps particle distribution (mean), - DSL day 3
gaussian fit
dN/dlogDp [#/cm³]
25000
20000
15000
10000
5000
0
6.04 6.98 8.06 9.31 10.8 12.4 14.3 16.5 19.1 22.1 25.5 29.4 34.0 39.2 45.3 52.3 60.4 69.8 80.6 93.1 107.5 124.1 143.3 165.5 191.1 220.7 254.8 294.3 339.8 392.4 453.2 523.3
Size: nm

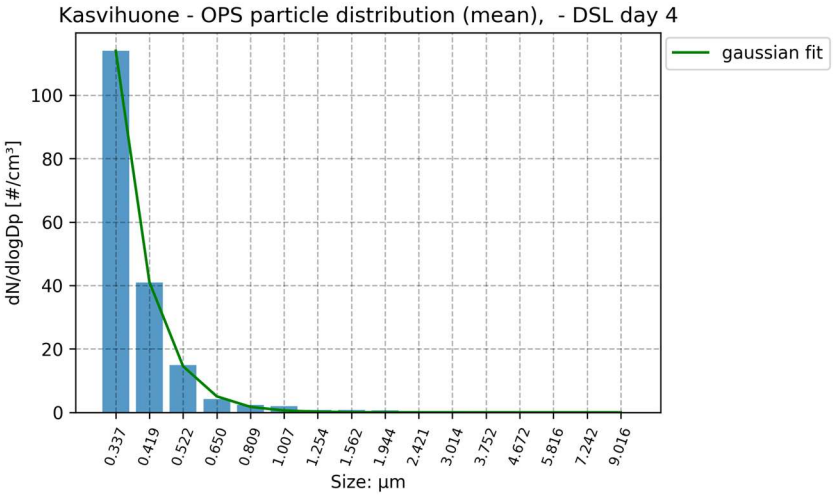

Kasvihuone - Eeps particle distribution (mean), DSL - day 4
gaussian fit
dN/dlogDp [#/cm³]
8000
6000
4000
2000
0
6.04 6.98 8.06 9.31 10.8 12.4 14.3 16.5 19.1 22.1 25.5 29.4 34.0 39.2 45.3 52.3 60.4 69.8 80.6 93.1 107.5 124.1 143.3 165.5 191.1 220.7 254.8 294.3 339.8 392.4 453.2 523.3
Size: nm

OPS toisto-osuus






Kasvihuone - OPS particle distribution (mean), - DSL day 3
gaussian fit
dN/dlogDp [#/cm³]
0
10
20
30
40
50
0.337
0.419
0.522
0.650
0.809
1.007
1.254
1.562
1.944
2.421
3.014
3.752
4.672
5.816
7.242
9.016
Size: µm


Kasvihuone - OPS particle distribution (mean), - DSL day 4
gaussian fit
dN/dlogDp [#/cm³]
0
20
40
60
80
100
0.337
0.419
0.522
0.650
0.809
1.007
1.254
1.562
1.944
2.421
3.014
3.752
4.672
5.816
7.242
9.016
Size: µm