Manne Kitinmäki

MegaSquirt

Asennus, säätäminen ja ohjelmointi

Opinnäytetyö
Kevät 2010
Tekniikan yksikkö
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖN TIIVISTELMÄ

Koulutusyksikkö: Tekniikan yksikkö
Koulutusohjelma: Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto: Ohjelmistotekniikan suuntautumisvaihtoehto

Tekijä: Manne Kitinmäki

Työn nimi: MegaSquirt

Ohjaaja: Seppo Stenberg

Vuosi: 2010 Sivumäärä: 65 Liitteiden lukumäärä: 1

Tässä työssä esitellään jälkiasennettavan moottorinohjausyksikön, MegaSquirtin asennus Boschin L-Jetronic-yksikön tilalle, sen säätäminen käyttökuntoon sekä oman ohjelmiston tekeminen ohjausyksikön tietojen lukemiseen. Kohdeautona toimii Opel Ascona B vuosimallia 1979, 20E-moottorilla.

Asennuksen tarkoituksena on saavuttaa parempi toimivuus tulevaisuudessa, kun moottoria tullaan virittämään enemmän.

Laitteisto saatiin asennettua onnistuneesti ja toimimaan kohtalaisen hyvin.

Työtä varten tehty oma ohjelma toimii ja osaa lukea joitain tietoja MegaSquirt-yksiköltä, mutta lisäkehitys on tarpeen kaikkien tietojen lukemiseksi. Ohjelmisto on toteutettu C#-kielellä.

Asiasanat: MegaSquirt, polttoaineenruiskutus, polttomoottori, MegaSquirt arvojen luku, C#

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

THESIS ABSTRACT

Faculty: School of Technology
Degree programme: Information Technology
Specialisation: Software Systems

Author/s: Manne Kitinmäki

Title of the thesis: MegaSquirt

Tutor/s: Seppo Stenberg

Year: 2010 Number of pages: 65 Number of appendices: 1

This thesis describes the installation of MegaSquirt, an aftermarket fuel injection system, and creating custom software to communicate with it. The unit replaces the original Bosch L-Jetronic engine control unit. The project car is Opel Ascona model B 1979 with 20E engine.

The purpose of this installation is to prepare for future modifications with the increase of power in mind.

The installation was successful and the system worked rather well.

The custom software is able to read some variables from the MegaSquirt control unit, but further development is necessary to read all of them. The program has been written in C#.

Keywords: MegaSquirt, fuel injection system, internal combustion engine, reading MegaSquirt values, C#

SISÄLLYS

KÄYTETYT TERMIT JA LYHENTEET

Analog to Digital Converter (ADC)

Anturin analoginen arvo muutetaan digitaaliseen muotoon. Esimerkiksi lämpötila-anturista mitataan jännitettä vastuksen yli ja tämä arvo muutetaan 8-bittiseksi eli luvuksi väliltä 0-255.

L-Jetronic (L-Jet) Boschin valmistama ruiskutusjärjestelmä, joka oli käytössä useissa autoissa 70-luvun lopulla ja 80-luvun alussa.

Lambda Tässä lambda tarkoittaa polttoaineen ja ilman suhdetta, joka optimaaliselle palamiselle on 14,7 osaa ilmaa ja yksi osa polttoainetta, jolloin lambda = 1.

MegaSquirt (MS) MegaSquirt (MS) on jälkiasennettava polttoaineen ruiskutusjärjestelmä.

MegaTune Ohjelmisto joka on tarkoitettu MegaSquirtin säätämiseen ja tarkkailuun.

MegaTest C#-ohjelmisto MegaSquirtin tietojen lukemiseen.

Megastimulator Laite joka emuloi auton antureiden lähettämiä arvoja ja mahdollistaa näin laitteen testaamisen ilman autoa.

NTC -vastus NTC-vastus (Negative Temperature Coefficient) on vastus, jonka resistanssi pienenee lämpötilan kasvaessa.

(Bowling & Grippo 2010.) (TM Autosanasto 1998.)

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

1.1 Työn tausta

Työssä haluttiin modernisoida vanhan auton moottorinohjausjärjestelmä, sekä tutkia mahdollisuutta tehdä ohjelmisto, joka osaa kommunikoida uuden ruiskujärjestelmän kanssa sarjaportin välityksellä. Ohjelmisto haluttiin tuottaa C#- kielellä sen selkeän ja ymmärrettävän rakenteen vuoksi.

1.2 Työn tavoite

Työn tarkoituksena on saada aikaan toimiva ja luotettava ruiskutusjärjestelmä, jonka avulla auto voidaan saada parempiin säätöihin mahdollisten tulevien viritysten kanssa, sekä saada aikaan ohjelmisto, joka osaa lukea MegaSquirt- yksikön muistista ruiskutusmäärien laskennassa käytettäviä parametreja. Työssä ei yritetä mahdollistaa kaksisuuntaista kommunikointia, vaan keskitytään pelkkään tietojen lukuun.

1.3 Työn rakenne

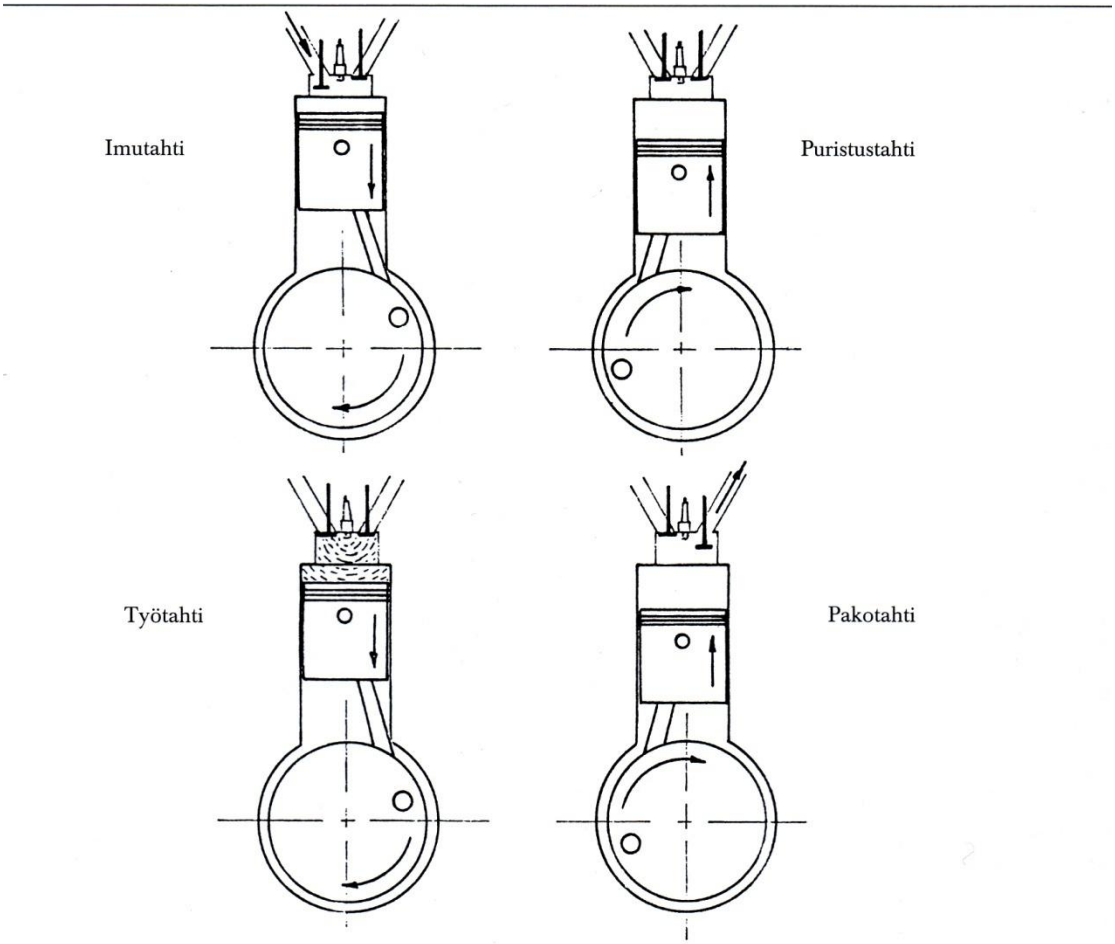
Toinen luku esittelee yleisesti moottorin toiminnan, sekä esittelee tarkemmin polttoaineen ruiskutusjärjestelmien toimintaa. Luku kolme esittelee tässä työssä asennetun MegaSquirt- järjestelmän tiedot ja tarvittavat sovitukset. Luku neljä kertoo lukuohjelmiston tekemisestä ja toimintaperiaatteista. Luku viisi esittelee ohjelmalla saavutetut lopputulokset. Kuudes luku kertoo, missä työn mahdollisista etenemisistä voi lukea lisää.

2 YLEISTIEDOT POLTTOAINEEN RUISKUTUKSESTA

2.1 Ottomoottori

Ottomoottorilla tarkoitetaan yleensä kaksi- tai nelitahtista bensiinikäyttöistä mäntämoottoria (Bauer 2002, 412). Koska kohdeautossa on nelitahtinen moottori, tässä työssä selitetään vain sen toimintaperiaate.

Nelitahtiperiaate tarkoittaa sitä, että neljän täyden moottorikierroksen aikana kukin mäntä käy kaksi kertaa ylhäällä ja kaksi kertaa alhaalla. Tahteja kutsutaan imu-, puristus-, työ- ja pakotahdeiksi. (Juurikkala 1987, 38-41)



KUVA 1. Ottomoottorin toiminta. (Bell, 1998)

Imutahdissa mäntä liikkuu alaspäin ja imee samalla ilman ja polttoaineen seosta sylinteriin avautuneen imuventtiilin kautta (Juurikkala 1987, 38-41). Uudemmissa moottoreissa imutahdissa imetään usein pelkkää ilmaa sylinteriin ja polttoaine ruiskutetaan myöhemmin suoraan sylinteriin.

Puristustahdissa sekä imu- että pakoventtiilit ovat kiinni ja mäntä liikkuu ylöspäin puristaen samalla ilman ja polttoaineen seosta kasaan, jolloin paine kasvaa sylinterissä. Puristustahdin lopussa polttoaine sytytetään sähkökipinän avulla. (Juurikkala 1987, 38-41.)

Työtahdissa palavan polttoaineen tuottamat laajenevat palokaasut työntävät mäntää alaspäin ja voima välittyy männän liikkeestä kiertokankeen joka pyörittää kampiakselia. Kampiakselista voima välittyy voimansiirtoon, eli auton tapauksessa pyörille. (Juurikkala 1987, 38-41)

Pakotahdissa mäntä liikkuu ylöspäin ja palokaasut poistuvat avoimen pakoventtiilin kautta pakosarjaan ja edelleen ulos pakoputkesta. (Bell 1998, 1.10.)

2.2 Polttoaineen tarve ja palaminen

Polttoaine tarvitsee palaakseen happea. Jotta yksi kilo bensiiniä palaisi täydellisesti, se tarvitsee noin 3,4 kiloa happea. Tämä happi saadaan ulkoilmasta, jossa hapen osuus on noin 23 painoprosenttia. Siksi oikea ilman ja bensiinin suhde on 14,7 kilogrammaa ilmaa ja yksi kilogramma polttoainetta kun halutaan parasta tehoa. Tätä suhdetta kuvataan usein ilmakertoimella, eli lambdalla (λ), jonka arvo on 1, kun ilman ja polttoaineen suhde on 14,7:1. Vaikka moottorit aiheuttavat aina päästöjä, tässä tilanteessa niitä syntyy tarpeettoman paljon. Tästä johtuen lambdan arvoa 1 tavoitellaan yleensä vain huipputehoa tarvittaessa ja pienemmissä kuormitustilanteissa seos on laihempi. (Bell 1998, 7.1, 7.2.)

Jos happea on liian vähän, on polttoaineen palaminen epätäydellistä. Palamaton polttoaine ei tuota palokaasuja, eli se ei hyödytä voiman tuottamisessa ja lisää siis turhaan polttoaineen kulutusta. Jos taas happea on liikaa, käy moottori liian laihalla seoksella.

Tarvittavaan seossuhteeseen vaikuttaa myös mm. lämpötila. Pakkassäällä ja moottorin ollessa kylmä saattaa tarvittava polttoaineen ja ilman seossuhde olla käynnistyksessä jopa 1:1. Tasakaasulla pienillä kierroksilla moottorin ollessa lämmin voi seossuhde olla jopa hieman laiha, esimerkiksi 16:1. Tällä haetaan parempaa taloudellisuutta, sillä luonnollisesti laihempi seos tarkoittaa pienempää polttoaineenkulutusta. Seos ei kuitenkaan voi olla kuinka laiha tahansa, koska laiha seos palaa hitaammin. Tästä johtuen lämpö ehtii johtua osiin pidemmän aikaa, ja saattaa esimerkiksi kuumentaa mäntää niin paljon että se sulaa puhki. (Bell 1998, 7.1, 7.2.)

Palamiseen vaikuttavat monet muutkin asiat, esimerkiksi ilman lämpötila ja ilmanpaine, jotka vaikuttavat suoraan ilman tiheyteen sekä se, kuinka hienojakoinen seos on. Mitä pienempinä pisaroina polttoaine on ilman seassa, sitä nopeammin ja tehokkaammin se palaa. (Bell 1998, 7.1, 7.2.)

2.3 Polttoainejärjestelmät yleisesti

Ottomoottoreissa on käytössä kolmenlaisia polttoainejärjestelmiä: nykyään jo vanhanaikainen kaasutin, imusarjaan ruiskuttava ruiskujärjestelmä ja suoraan sylinterin ruiskuttava suorasuihkutus. (Bell 1998, 7.1.)

Kaasutin on vanhin ja yksinkertaisin tapa siirtää polttoainetta palotilaan. Kaasuttimia on monenlaisia, mutta yleisin tyyppi on kohokammiollinen kiinteäkurkkuinen kaasutin. Sen toiminta perustuu uimurikammioon jonka sisällä polttoaine on. Moottorin kierroslukua säädellään kaasuläpällä, jota kaasupoljin liikuttaa. Kaasuläppä säätää kaasuttimen läpi virtaavan ilman määrää. Kurkku on keskeltä supistettu

putki jossa ilma virtaa tietyllä nopeudella, jota säätelee alipaineen mukaan säätyvä läppä. Virtausnopeus on yleensä vakio koko moottorin kierroslukualueella. Uimurikammion sisällä kelluu koho, jota tämä alipaineläppä painaa alas puristaen polttoainetta. Läppä avaa samalla neulaventtiiliä, joka sulkee kurkkuun menevän aukon. Tästä aukosta polttoaine imeytyy kaasuttimen kurkkuun ja sekoittuu virtaavaan ilmaan, päätyen edelleen imuventtiilien kautta sylinteriin. (Juurikkala, Airola, Pohjanpalo & Seppälä 1986)

Kaasutin on vanhanaikainen järjestelmä, jonka käyttö on nykyään hyvin harvinaista, koska ruiskujärjestelmillä päästään huomattavasti parempaan taloudellisuuteen ja moottoritehoon. Kaasuttimen kanssa on huomattavasti vaikeampi saada seos sopivaksi kaikissa olosuhteissa, mistä johtuen seos on monissa tapauksissa liian rikas. Tämä johtaa siihen, että polttoainetta kuluu turhaan. (Bell 1998, 7.1.)

Kaasuttimen on nykyään korvannut lähes täysin jonkinlainen polttoaineen ruiskutuslaitteisto. Järjestelmä voi olla joko mekaaninen tai sähköinen, mutta mekaaniset järjestelmät ovat jäämässä pois useimmista käyttökohteista. Sähköisten järjestelmien etuja: ne eivät tarvitse mekaanista kytkentää moottoriin, laitteita voi sijoitella vapaammin, sekä järjestelmien mahdollistama suuri määrä säätösuureita. (Juurikkala ym., 1986, 178-179) Sähköiset järjestelmät jakautuvat yleensä kolmeen tyyppiin. Ruiskusuuttimia voi olla imusarjassa yksi tai useampi, niin että jokaiselle sylinterille on omansa. Uusimmissa versioissa suuttimet suihkuttavat seoksen suoraan sylinteriin, jopa useassa eri erässä. Tässä työssä keskitytään imusarjaan suihkuttavaan järjestelmään, jossa jokaiselle sylinterille on oma suutin.

Ruiskujärjestelmässä on monia asioita jotka tekevät siitä kaasutinta paremman polttoainejärjestelmän. Järjestelmä pystyy sopeutumaan hyvin kaikkiin tilanteisiin, koska se sisältää useita antureita jotka mittaavat eri ominaisuuksia. Näin järjestelmä pystyy säätämään seoksen oikeaksi kaikkiin tilanteisiin. Tarkkuus tietysti vaihtelee ruiskutyypin mukaan, mutta kaikki ovat huomattavasti kaasutinta parempia. Ruiskujärjestelmä tarkkailee moottorin ja ilman lämpötilaa, moottorin kuormitusta, sisään virtaavan ilman määrää, kaasupolkimen asentoa sekä joissain järjes-

telmissä myös ilmanpainetta ja kullakin hetkellä käytössä olevaa seosta. Näiden arvojen perusteella järjestelmä säätää tarvittavaa polttoainemäärää jopa useita kertoja sekunnissa, sekä säätää ruiskutyypistä riippuen joko suuttimien aukioloaikaa tai ruiskutuspainetta. Koska polttoaineen ja ilman suhde on kaikissa tilanteissa lähes optimaalinen, on polttoainetaloudellisuus luonnollisesti parempi. Ruiskujärjestelmät osaavat myös katkaista polttoaineen syötön kokonaan esimerkiksi alamäessä, kun kaasupoljin päästetään kokonaan ylös. Tällaisessa tilanteessa kaasutin syöttää koko ajan polttoainetta jonkin verran; ruiskuautossa polttoainetta kuluu vasta, kun kaasua taas painetaan tai kun kierrosluku putoaa tietyn rajan alapuolelle. (Tranter 1997, 128-130.)

Nykyään uudemmat järjestelmät eivät sisällä enää edes mekaanista yhteyttä kaasupolkimen ja kaasuläppäkotelon välillä. Näissä järjestelmissä kaasupolkimen asento välitetään sähköisessä muodossa ruiskutusjärjestelmälle, joka sitten säätää kaasun halutulle tasolle. Polttoaine myös suihkutetaan yleensä suoraan sylinteriin useassa erässä, jotta palotapahtuma olisi hallitumpi. (Bell 1998, 7.2, 7.3.)

2.4 Alkuperäinen järjestelmä

Tässä projektissa asennusalustana toimiva auto on Opel Ascona B vuosimallia 1979. Autossa oleva moottori on nelisylinterinen Opelin 20E, jossa 20 tarkoittaa kahden litran iskutilavuutta ja E tulee sanasta Einspritzungen, joka tarkoittaa polttoaineen ruiskutusta. Koska autossa on jo valmiiksi ruiskutuslaitteisto, on toteutus paljon yksinkertaisempi kuin tapauksessa, jossa autossa on alun perin kaasutin. Tämä moottori voi käyttää joko Boschin L- tai LE-Jetronic-ruiskutusta, joista on lisäksi monenlaisia variaatioita. Tässä esitellään vain nimenomaan autossa ollut L-Jetronicin versio. Kirjain L ilmaisee, että toiminta perustuu moottoriin virtaavan ilman määrään (=Luftmengengesteuert). (Juurikkala, Airola, Pohjanpalo, Seppälä, 1986, 183.)

Järjestelmän toiminnalliset osat on helpointa jakaa kolmeen osaan; ilmajärjestelmä, polttoainejärjestelmä ja sähköinen järjestelmä. (Bell 1998, 7.3.)

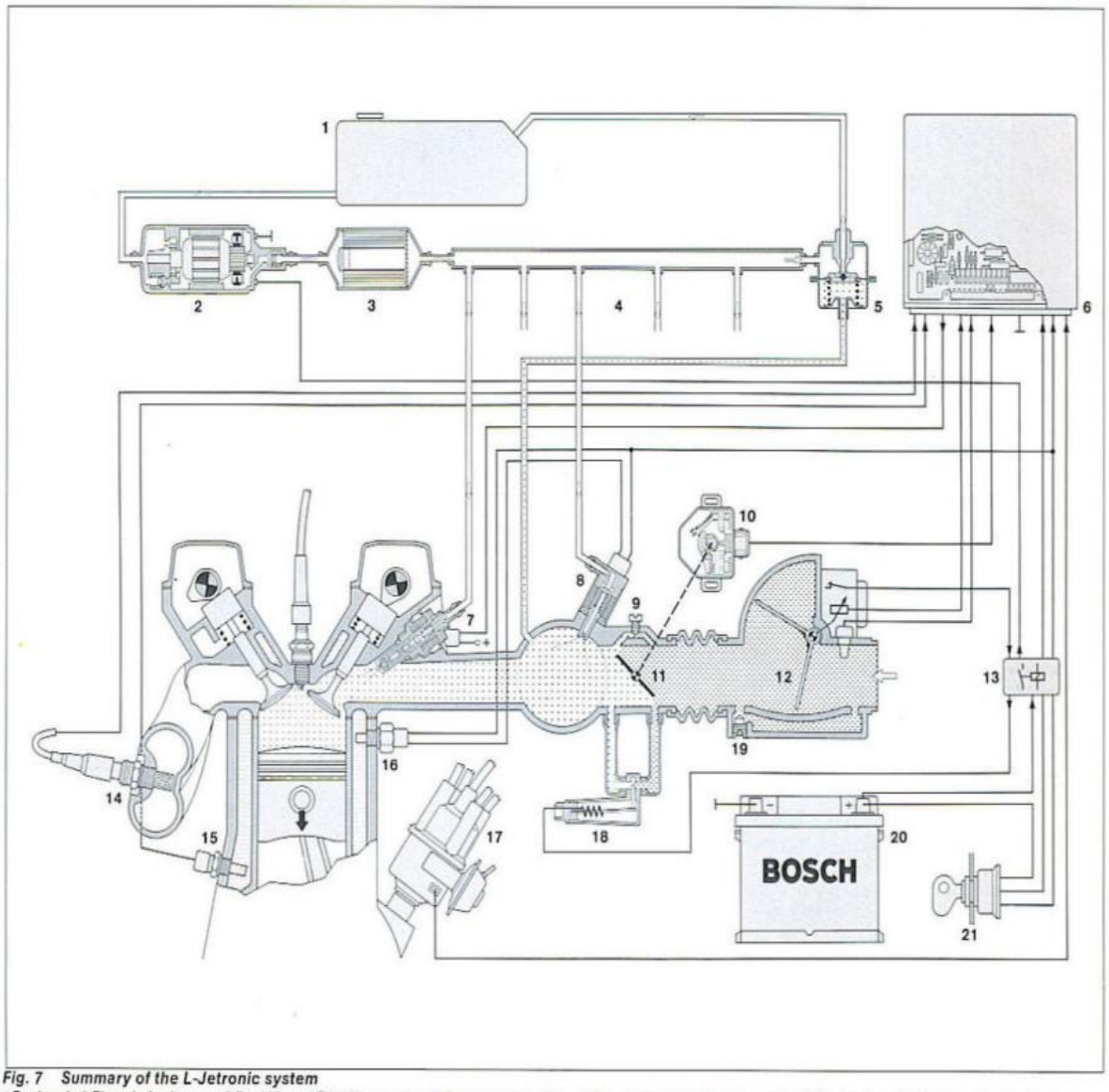


Fig. 7 Summary of the L-Jetronic system
1 Fuel tank, 2 Electric fuel pump, 3 Fuel filter, 4 Distributor pipe, 5 Pressure regulator, 6 Control unit, 7 Injection valve, 8 Start valve, 9 Idle-speed adjusting screw, 10 Throttle-valve switch, 11 Throttle valve, 12 Air-flow sensor, 13 Relay combination, 14 Lambda sensor (only for certain countries), 15 Engine temperature sensor, 16 Thermo-time switch, 17 Ignition distributor, 18 Auxiliary-air device, 19 Idle-mixture adjusting screw, 20 Battery, 21 Ignition-starter switch

KUVA 2. L-Jetronic osakaavio.
(Robert Bosch GmbH 1981.)

2.5 Ilmajärjestelmä

Ilmajärjestelmä käsittää osat, jotka liittyvät ilman saamiseen moottoriin. Osia on kolme: imusarja, kaasuläppäkotelo sekä lisäilmaluisti (tyhjäkäyntiventtiili).

2.5.1 Imusarja

Imusarja on koko järjestelmän runko, johon muut osat kiinnittyvät tavalla tai toisella. Imusarja on yksinkertaistettuna putkisto, jossa ilma ja polttoaine sekoittuvat keskenään ennen sylinteriin päätymistään. (Bell 1998, 2.13-2.16.)

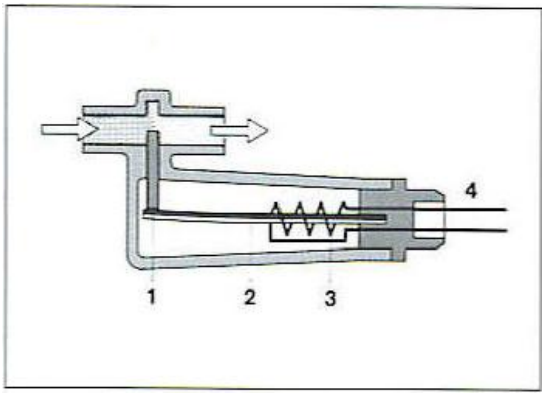
2.5.2 Kaasuläppäkotelo

Moottorin tehoa säädellään kaasuläpän avulla. Kaasuläppäkotelossa kiinni olevan putken sisällä sijaitseva läppä on akseloitu keskeltä putkeen. Kaasupoljinta painettaessa vaijeri vetää läppää enemmän auki, jolloin läpän ohi virtaa enemmän ilmaa imusarjaan. Kaasuläppä ei siis vaikuta suoraan polttoaineen määrään, vaan säätelee moottorin imemän ilman määrää, joka mitataan ilmamäärämittarilla. (Bell 1998, 2.11-2.13.)

2.5.3 Lisäilmaluisti

Lisäilmaluistin tehtävänä on päästää kylmään moottoriin enemmän ilmaa kaasuläpän ohi, samalla kun moottoriin ruiskutetaan enemmän polttoainetta. Lisäilmaluisti reagoi lämpötilaan kahdella tavalla. Sen sisällä on kaksoismetallinen liuska, joka taipuu lämpötilan mukaan ja sulkee hiljalleen ilman virtausaukon moottorin lämmetessä. Osa tästä lämmöstä tulee lisäilmaluistin runkoon suoraan termostaattikotelosta (termostaatti on osa, joka säätelee moottorin jäähdytystä päästämällä kuu-

maa vettä jäähdyttimelle), jossa osa on kiinni. Tämän lisäksi luistille tulee jatkuva sähkövirta, joka lämmittää kaksoismetalliliuskaa. (Probst 1989, 14.)



1. Sulkuläppä
2. Kaksoismetalliliuska
3. Lämmitysvastus
4. Liitin

KUVA 3. Lisäilmaluisti. Valkoiset nuolet kuvaavat ilman kulkua.
(Robert Bosch GmbH 1981.)

2.6 Polttoainejärjestelmä

Polttoainejärjestelmä käsittää osat, jotka liittyvät polttoaineen siirtoon tankista sylinteriin. Näitä osia ovat polttoainepumppu, suodatin, paineensäädin sekä suihkutussuuttimet.

2.6.1 Polttoainepumppu, suodatin ja paineensäädin

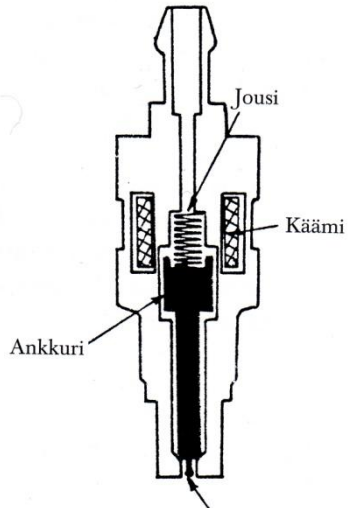
Polttoainepumppu on Boschin valmistama sähköpumppu joka nostaa polttoainelinjan paineen tarpeeksi korkeaksi ruiskutusta varten. Pumppu sijaitsee tässä automallissa tankin ulkopuolella kiinnitettynä auton pohjaan. Pumpun jälkeen letkussa on suodatin, joka kerää polttoaineen seassa olevat epäpuhtaudet.

Polttoainepumppu työntää polttoainetta putkissa eteenpäin aina täydellä paineella, joka on suurempi kuin oikeasti tarvittava paine. Painetta säätelee erillinen säädin, jota ohjaa imusarjan sisällä vallitseva alipaine. Säädin pyrkii pitämään paineen

oikeana kaikissa kuormitustilanteissa. Normaalisti paine tyhjäkäynnillä on noin 2,5 bar (Robert Bosch GmbH 1981, 9). Koska pumpun paine on suurempi kuin tämä tarvittava paine, osa polttoaineesta virtaa paluulinjaa pitkin takaisin polttoainetankkiin.

2.6.2 Suihkutussuuttimet

Imusarjan loppupäässä juuri ennen sylintereitä sijaitsevat ruiskutussuuttimet, joita on yksi sylinteriä kohden, eli yhteensä neljä kappaletta. Suutin on periaatteessa solenoidi, jota ohjataan sähköllä. Kun suuttimen käämiin johdetaan sähköä, syntyvä magneettikenttä nostaa sisällä olevan neulan ylös, jolloin polttoaine pääsee ulos suuttimen päässä olevista pienistä rei´istä sumuna imusarjaan. Kun sähkö katkeaa, palauttaa jousi neulan takaisin alas ja ruiskutusaukko sulkeutuu. L-Jetronic on yksinkertainen järjestelmä, joten siinä kaikki suuttimet avautuvat ja sulkeutuvat samaan aikaan. Suuttimet ovat Boschin valmistamat, osanumeroltaan 0 280 150 205 (Bosch eCat 2010). Tässä järjestelmässä suutinten tuotto on vakio, 170 cc/min (Chichak 2002). Moottoriin menevän polttoaineen määrää annostellaan suuttimien aukioloajalla. Koska kaikki suuttimet avautuvat samaan aikaan, polttoaineen ja ilman seos jää imuputkiin odottamaan imuventtiilien avautumista. (Bell 1998, 7.2 - 7.4.)



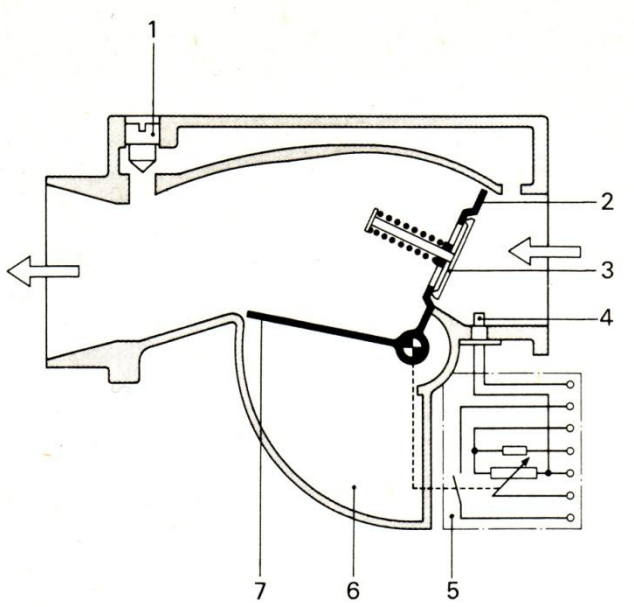
KUVA 4. Suutin
(Bell, 1998)

2.7 Sähköinen järjestelmä

Sähköinen järjestelmä käsittää joukon erilaisia antureita sekä itse laitteiston ohjausyksikön. Antureihin kuuluvat ilmamäärämittari, kaasuläpän asentotunnistin, moottorin lämpötunnistin, imuilman lämpötunnistin, lämpöaikakytkin ja moottorin käyntinopeudentunnistin. (Bell 1998, 7.3.)

2.7.1 Ilmamäärämittari

Ilmajärjestelmän alkupäässä heti ilmasuodattimen jälkeen on ilmamäärämittari, joka nimensä mukaisesti mittaa moottoriin virtaavaa ilmamäärää. Tässä järjestelmässä mittari on läppätyyppinen. Mittarin kotelon sisällä oleva läppä avautuu sitä enemmän, mitä enemmän ilmaa sen ohi virtaa. Mittarin päällä on piirilevy, jonka hiilipinnalla läppään kiinnitetty neula liikkuu. Neulan ja hiilipinnan alun välillä oleva sähköinen vastus muuttuu sen mukaan missä kohdassa neula on, ja tätä resistanssiksi kutsuttua ilmiötä mittaamalla moottorinohjausyksikkö tietää moottoriin virtaavan ilman määrän. (Juurikkala ym. 1986, 184.)



1. Tyhjäkäyntiseoksen säätöruuvi
2. Mittausläppä
3. Suuntaventtiili
4. Ilman lämpötilatunnistin
5. Piirilevy
6. Vaimenninkammio
7. Vaimenninläppä

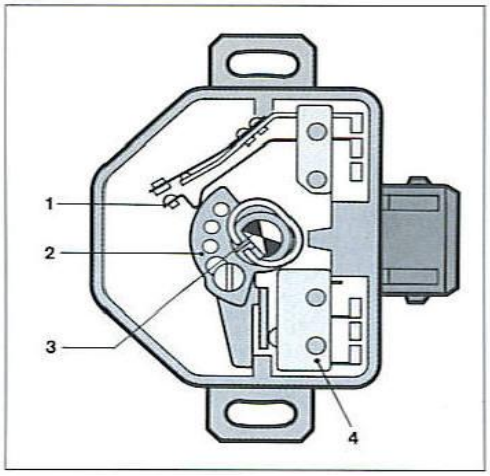
KUVA 5. L-Jetronic ilmamäärämittari.
(Juurikkala ym., 1986, 185)

2.7.2 Kaasuläpän asentotunnistin

Kaasuläpän akseliin kiinnitetty kaasuläpän asentotunnistin on tässä järjestelmässä kolmiasentoinen kytkin. Kytkimessä on kolme liitäntänastaa. Kun kaasua ei paineta, näistä alimman ja keskimmäisen välillä on sähköinen yhteys. Tämä on tyhjäkäyntiasento. Tätä asentoa tarvitaan moottorijarrutuksen yhteydessä tapahtuvaan polttoaineen syötön katkaisuun, täyskaasurikastukseen sekä hyvän tyhjäkäynnin varmistamiseen. (Probst 1989, 15.)

Kaasua painettaessa sisällä olevat metalliliuskat irtoavat toisistaan ja yhteys katkeaa. Tällöin ruiskutusjärjestelmä siirtyy osakaasuasentoon. (Probst 1989, 15.)

Kun kaasu painetaan pohjaan asti, yhdistyvät ylimmäinen ja keskimmäinen metalliliuska. Tällöin liittimen kahden ylimmäisen nastan välillä on sähköyhteys ja järjestelmä siirtyy täyskaasuasentoon. Tässä asennossa järjestelmä pyrkii ottamaan kaiken mahdollisen moottoritehon käyttöön esimerkiksi ohitustilanteita varten. (Probst 1989, 15.)

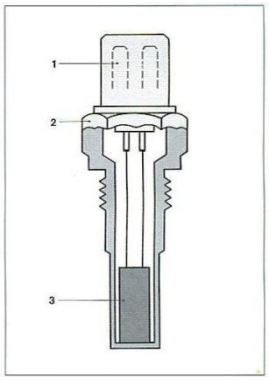


1. Täyskaasunastat
2. Kontaktisilta
3. Kaasuläppäakseli
4. Tyhjäkäyntinasta

KUVA 6. Kaasuläpän asentotunnistin.
(Robert Bosch GmbH 1981, 15)

2.7.3 Moottorinlämpötunnistin

Moottorin lämpötilatunnistin sijaitsee termostaattikotelossa. Anturi on metallinen sylinteri, jonka sisällä on vastus. Vastuksen resistanssi muuttuu lämpötilan mukaan. Tätä vastusta kutsutaan NTC-vastukseksi. Anturi mittaa moottorissa virtaavan jäähdytysnesteen lämpötilaa, joka vastaa hyvin moottorin lämpötilaa. Kaksinapaisen anturin toinen johto on kiinni moottorinohjausyksikön maatasossa ja toisen johdon kautta yksikkö mittaa anturin vastusta. Tämän vastuksen arvojen ohjaamana moottorinohjaus säätää ruiskutusta moottorin lämpötilan mukaan. (Tranter 1999, 22.)



1. Liitin
2. Kuori
3. NTC-vastus

KUVA 7. Lämpöanturit
(Robert Bosch GmbH 1981, 14.)

2.7.4 Imuilman lämpötunnistin

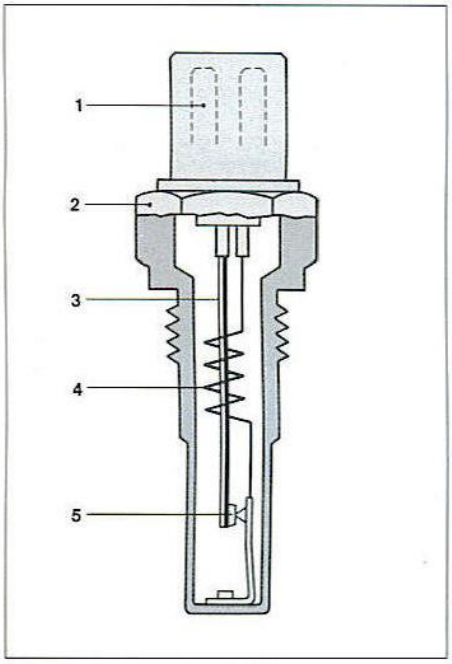
Imuilman lämpötunnistin sijaitsee ilmamäärämittarin kotelossa juuri ennen mittarin läppää. Imuilman lämpötunnistin on samanlainen NTC-vastus kuin moottorinlämpötunnistinkin, mutta sen rakenne on huomattavasti avoimempi, jolloin se reagoi nopeammin ilman lämpötilan muutoksiin. (Tranter 1999, 22.)

2.7.5 Käyntinopeudentunnistin

Käyntinopeudelle ei ole omaa varsinasta tunnistinta, vaan tämä tieto saadaan suoraan virranjakajalta, joka tuottaa sähköimpulssin aina kun moottori on pyörähtänyt täyden kierroksen. (Robert Bosch GmbH 1981, 11)

2.7.6 Lämpöaikakytkin

Lämpöaikakytkin on periaatteessa jäänne vanhemmasta mallista, jossa on erillinen lisäsuutin kylmäkäynnistyksiin, mutta se hoitaa samaa tehtävää myös tässä versiossa ohjaten suuttimien aukioloaikaa pitemmäksi. Toimintaperiaate on hyvin samantyyppinen kuin lisäilmaluistissa. Kytkimen sisällä oleva metalliliuska taipuu lämmetessään joko moottorin lämmöstä tai tietyn ajan kuluttua sisäisellä lämmitysvastuksella ja kytkin menee pois päältä lopettaen lisärikastuksen. Moottorin ollessa valmiiksi lämmin, ei tarvita lisärikastusta, ja tällöin kytkin on valmiiksi pois päältä moottorinlämmön vaikutuksesta. (Probst 1989, 3.18)



1. Liitin
2. Kuori
3. Kaksoismetalliliuska
4. Lämmitysvastus
5. Kytkentänasta

KUVA 8. Lämpöaikakytkin
(Robert Bosch GmbH 1981, 14)

2.7.7 Moottorinohjausyksikkö (ECU)

Ruiskutuslaitteiston tärkein osa on moottorinohjausyksikkö. Sen tehtävänä on antureilta saamiensa tietojen mukaan laskea tarvittava polttoainemäärä, jotta moottori toimisi optimaalisesti. L-Jetronicissa yksikkö on lähes täysin analoginen eikä sisällä varsinaista ohjausohjelmistoa, jonka voisi vaihtaa. Tätä kyseistä moottoria ei siis voi "lastuvirittää" tehon kasvattamiseksi. Moottorinohjausyksikön tehtävä on ohjata suuttimien aukioloaikaa anturitietojen perusteella eri tilanteissa. (Bell 1998, 7.4.) Jos kierrosluku on vaikkapa noin 300 kierrosta minuutissa, imuilman lämpötila -20 ja moottorin lämpötila +30, laskee ohjausyksikkö näistä arvoista, että starttaus on käynnissä, moottori on osittain lämmin ja on kylmä sää. Tällöin tilanne ei ole sama kuin käynnistettäisiin kylmää moottoria samoissa olosuhteissa. ECU laskee sopivan polttoainemäärän näihin olosuhteisiin ja moottori käynnistyy. Moottorin käydessä sen lämpötila nousee hiljalleen, jolloin ECU säätää polttoaineen määrää samalla pienemmäksi jos mikään muu anturiarvo ei muutu. (Bell 1998, 7.4.)

Esimerkiksi, jos moottorin kierrosluku on 2000 kierrosta minuutissa ja kaasuläpän asentoanturin kaksi alimmaista nastaa on kytkettynä, ECU laskee, että on menossa moottorijarrutus ja katkaisee polttoaineen syötön kunnes kierrosluku laskee tarpeeksi alas, eli tässä moottorissa noin 1500 kierrokseen minuutissa.

ECU on siis yksinkertaistettuna koko ruiskujärjestelmän aivot ja ohjaa koko järjestelmän toimintaa.

KUVA 9. ECU.

3 MEGASQUIRTIN KOKOAMINEN, TESTAUS JA ASENNUS

MegaSquirt on jälkiasennettava polttoaineenruiskutus, jonka ovat kehittäneet Bruce Bowling ja Al Grippo. MegaSquirt ei ole varsinainen kaupallinen tuote, vaan sen ideana on itse tekeminen ja oppiminen. Ominaisuuksiltaan MegaSquirt ei ole kaupallisia ratkaisuja huonompi, ja on joissain asioissa jopa parempi. (Bowling & Grippo 2010.)

MegaSquirt on huomattavasti edullisempi kuin kaupalliset järjestelmät. Sen hankintakustannukset ovat noin kymmenesosa normaalista järjestelmästä, mutta takuuta sillä ei luonnollisesti ole, ja tuotetuki toimii pääasiassa Internetissä, keskustelupalstoilla ja alan harrastajien projektisivustoilla. Tässä järjestelmässä MegaSquirt haluttiin saada toimimaan ilman ulkopuolista apua. Varsinaisia tehotavoitteita ei tässä vaiheessa ollut, koska ruiskua oltiin asentamassa virittämättömään moot-

toriin. Ajatuksena on kuitenkin tulevaisuudessa virittää kyseistä moottoria ja tässä tapauksessa alkuperäinen ruiskun ohjainyksikkö vaikeuttaa viritystä, koska siinä ei ole kunnollisia säätöjä. Vakiomoottoriin MegaSquirt on helpompi asentaa kuin viritettyyn, koska tässä asennuksessa mahdolliset ohjainyksikköön liittymättömät ongelmat tulisivat esiin myös vakioruiskulla. Lisäksi säätäminen on huomattavasti helpompaa, jos on olemassa vertailukohta siitä, miten moottorin pitäisi kyseisellä kokoonpanolla toimia.

Mahdollinen viritys voi aiheuttaa monenlaisia ongelmia vakioruiskun kanssa. Aikanaan autossa oli jyrkempikulmainen nokka-akseli, joka tarkoitti sitä että imuventtiilit olivat pitempään auki. Tämä aiheutti ongelmia käynnissä, kun osa sylinterin paineesta karkasi puristustahdin aikana osittain avoimen imuventtiilin kautta imusarjaan ja siitä edelleen ilmamäärämittariin, jonka läppä heilui paineiskujen mukana. Tämä aiheutti sen, että ilmamäärämittarin vastusarvo ei ollut vakaa, jolloin aiheutui monenlaisia käyntihäiriöitä. Kyseinen viritys oli vielä aika lievä, joten enemmän virittäessä ongelmat luultavasti vain pahenisivat.

Toinen suuri ongelma on se, että vanha ruisku ei osaa usein sopeutua muuttuneisiin polttoainetarpeisiin, jos moottoria viritetään. Tätä voi osittain kiertää esimerkiksi nostamalla polttoaineen ruiskutuspainetta, mutta ratkaisu ei ole suotava, koska se lisää kulutusta myös niissä olosuhteissa, joissa lisää polttoainetta ei oikeasti tarvittaisi.

Muun muassa näistä syistä johtuen on tarpeen vaihtaa moottorinohjaus johonkin muuhun, jos moottorista halutaan ottaa ylimääräistä tehoa irti. Pelkästään moottorinohjausta vaihtamalla voi saavuttaa jonkin verran lisätehoa joissain autoissa, mutta tässä tapauksessa sellaista ei varsinaisesti odotettu saatavaksi. Kulutusta sen sijaan oletettiin saatavan alemmaksi, koska MegaSquirt on huomattavasti tarkempi ohjauksessaan eri olosuhteissa kuin alkuperäinen järjestelmä.

Vaikka MegaSquirtia saa hankittua myös valmiina, yleisin tapa on tilata tarvittavat osat ja koota järjestelmä itse. Niin tehtiin myös tässä tapauksessa. Tarvittavat osat

tilattiin Partco Oy -nimisen yrityksen sivujen kautta. Paketissa tuli mukana valmis piirilevy, tarvittavat sähkökomponentit, kotelo sekä kokoamisohjeet. Ruiskun lisäksi tilattiin Megastimulator, joka on erillinen laite MegaSquirtin testaamiseen.

3.1 MegaSquirtin muutokset lyhyesti

MegaSquirt eroaa alkuperäisestä järjestelmästä, joten joitain muutoksia täytyy tehdä, kun MegaSquirt liitetään auton vanhaan järjestelmään. Koska järjestelmästä haluttiin mahdollisimman helposti purettava, tehtiin siihen kokonaan oma johtosarja. Tätä varten hankittiin käytetty johtosarja toisesta ruiskusta. Johtosarjan lisäksi tehtiin sovitelaippa kaasuläpän ja asentoanturin väliin, koska MegaSquirt käyttää potentiometrityyppistä mittausta kaasuläpän asennolle. Alkuperäinen imuilman lämpötilatieto ei jostain syystä toiminut oikein MegaSquirtin kanssa, joten projektiin hankittiin myös uusi imuilman lämpöanturi.

MegaSquirt ei myöskään tarvitse alkuperäistä ilmamäärämittaria, vaan se mittaa imusarjassa olevaa alipainetta. Tästä syystä valmistettiin putki, johon kiinnitettiin vapaavirtaussuodatin sekä imuilman lämpöanturi.

Uusi johtosarja asennettiin autoon vanhan johtosarjan viereen. Näin vaihto alkuperäisen ja uuden ruiskujärjestelmän välillä kestää vain joitain minuutteja. Suuttimien ja antureiden liittimet vaihdetaan uuden ruiskun liittimiin. Tämän jälkeen irrotetaan ilmamäärämittari sekä kaasuläpän asentotunnistin ja asennetaan tilalle korvaavat osat. Tämän jälkeen liitetään vain johtosarja MegaSquirt- ohjausyksikköön ja auto on käyttövalmis.

Itse ruiskun vaatimien osien lisäksi lisättiin autoon laajakaistalambda säätämisen helpottamiseksi. Lambdatunnistin mittaa pakokaasuista polttoaineen ja ilman seossuhdetta ja kertoo onko se optimaalinen. Asennetussa järjestelmässä tieto näkyy sekä numeroarvona, että erivärisillä LED-valoilla ilmaistuna.

3.2 Laitteiston kokoaminen ja ensitestit

MegaSquirtin kokoamisprosessissa ei ollut mitään erityisen haastavaa, koska siihen oli valmiit ohjeet. Komponentit laitettiin paikoilleen ja juotettiin käsin kiinni ja laitteen toimivuutta testattiin aina, kun sitä ohjeissa suositeltiin.

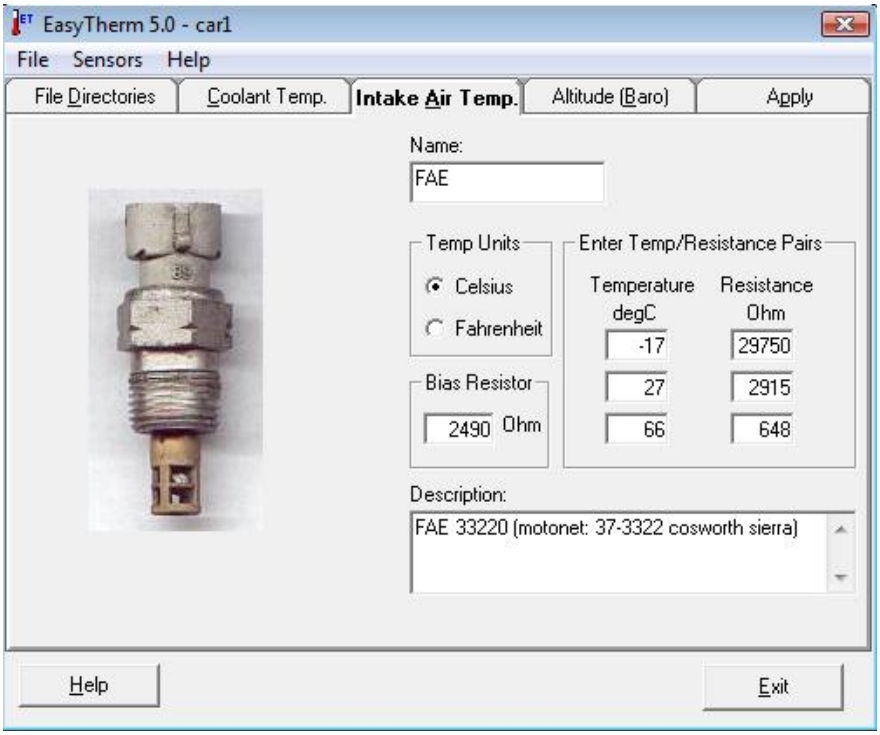
MegaSquirtin testaamista varten koottiin myös Megastimulator, joka on yksinkertainen laite, joka emuloi moottorin lähettämiä signaaleja, joiden arvoja pystyy muuttamaan säätövastusten avulla. (Bowling & Grippo 2010.) Megastimulatoriin ei tullut valmista piirilevyä, vaan se koottiin reikälevylle mukana tulleen kytkentäkaavion avulla.

Kun laitteet oli saatu koottua, ensimmäisenä haettiin uusin extra-koodi. Tässä koodissa on muutoksia ja parannuksia alkuperäiseen versioon nähden, mm. tarkempi polttoainekartta, joka mahdollistaa enemmän säätöarvoja eri tilanteisiin. Vakiokartan koko on 8x8 ja extra-koodissa 12x12.

3.3 Osien ja antureiden sovitus

Koska MegaSquirt tukee hyvin monenlaisia antureita, ei esimerkiksi moottorin lämpöanturin vaihtaminen ollut tarpeen. Moottorissa valmiiksi oleva anturi on Boschin normaali NTC-vastus. Koska MegaSquirtissa on valmis kartta eri anturille, piti kartta skaalata sopimaan vanhaan anturiin. Tässä tarvittiin ohjelmaa nimeltä EasyTherm. Ohjelmalle syötetään vastusarvo kolmessa eri lämpötilassa ja se luo näiden arvojen perusteella uuden taulukon, joka pitää ladata sekä MegaSquirtin sisäiseen muistiin, että käytettävän säätöohjelmiston muistiin. Boschin sivuilta löytyi lämpöanturin osanumero 0 280 130 026. Tällä osanumerolla hakemalla löytyi Bosch Motorsportin Saksan sivuilta anturin datalehti, josta tarvittavat arvot saatiin. Samalla ohjelmalla skaalataan myös imuilman lämpöanturi. Tässä arvot otettiin suoraan Megasquirt Opeliin -sivustolta, koska projektiin tilattiin sama anturi, joka sivuilla mainitaan.

EasyTherm luo lopputuloksena tiedostot thermfactor.inc ja matfactor.inc sekä uuden .s19-tiedoston (nimi riippuu alkuperäisestä). Kaksi ensin mainittua ovat MegaTunen kalibrointitiedostoja, joilla korvataan vastaavat projektin kansiossa. Viimeisin taas on MegaSquirtin sisäinen koodi, joka pitää lähettää laitteeseen, jotta se ja säätöohjelmisto tulkitsevat lämpötilaa oikein.



KUVA 10. EasyTherm ja käytetyt imuilman anturin arvot

EasyTherm lähettää valmiiksi oikeat tiedot MegaSquirt-ohjainyksikölle, jos se on kytkettynä. Kun arvot saatiin lähetettyä ohjainyksikölle, oli sovitus lämpöanturien osalta valmis.

3.3.1 Sovite ja kaasuläppäanturi

Koska MegaSquirt käyttää kaasuläpän asennontunnistukseen potentiometrityyppistä anturia, ei auton alkuperäistä kaasuläpän asentoanturia voinut käyttää, koska se on vain kolmiasentoinen. Tässäkin valittiin helppo lähestymistapa ja otettiin mallia erään harrastajan toteutuksesta. Projektiin tilattiin A Vectra 2.0 (Motonet: 28-1416)-mallin anturi. Tämä anturi ei vaadi erillistä kalibrointia MegaSquirtia var-

ten johtuen MegaSquirtin mittaustavasta. MegaSquirt vertaa anturin arvoja siihen arvoon, jota anturi lähettää virran kytkeytyessä. Anturi on mahdollista kalibroida MegaTunessa, mutta tämä vaikuttaa vain ohjelmanäkymään eikä ruiskun toimintaan.

Vaikka anturi sopii sähköisiltä ominaisuuksiltaan suoraan, jouduttiin fyysisen sopivuuden takia tekemään hieman muutoksia, koska uusi anturi on alkuperäistä matalampi, eikä näin ollen sovi paikoilleen. Tarkoitukseen valmistettiin yksinkertainen 10 mm paksu metallilaippa, johon tehtiin yksi iso reikä, josta kaasuläpän akseli tulee läpi, sekä neljä pienempää reikää. Näistä kahteen tehtiin kierteet, joihin anturi kiinnittyy ja toiset kaksi tehtiin vähän soikeiksi, jotta anturin asentoa voi tarvittaessa säätää. Näiden kahden reiän läpi laippa kiinnittyy kaasuläpän runkoon.

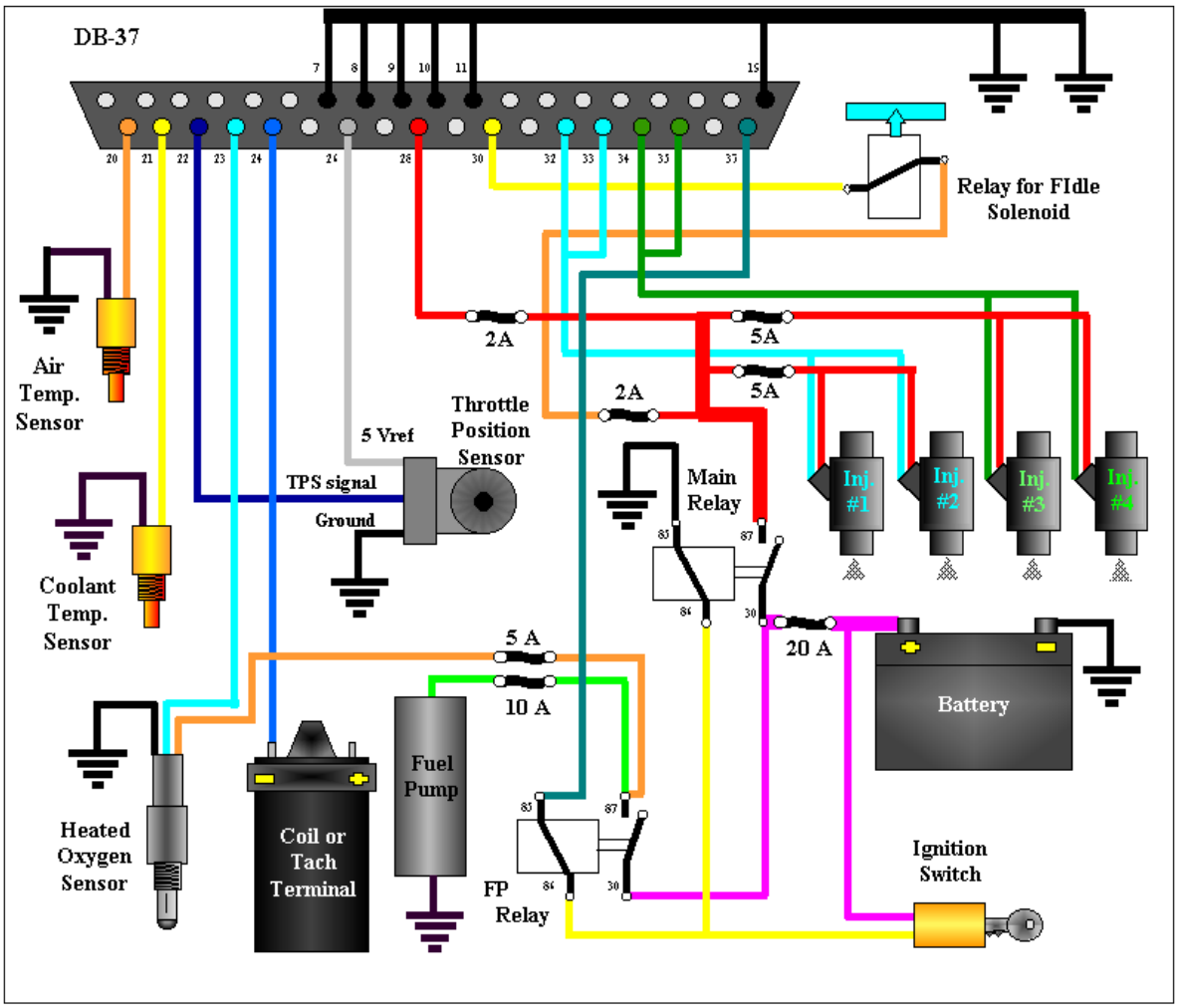
3.3.2 Ilmamäärämittarin ohitus

Koska MegaSquirt laskee tarvittavan polttoaineen eri tavalla, ei alkuperäistä ilmamäärämittaria tarvittu enää. Sen tilalle valmistettiin putki, johon ilmansuodatin kiinnittyy suoraan. Alkuperäinen imuilman lämpöanturi sijaitsee ilmamäärämittarissa ja näin ollen putkeen hitsattiin sopiva mutteri, johon uusi imuilman lämpöanturi kiinnitettiin.

KUVA 11. Ilmamäärämittarin ohitusputki.

3.3.3 Johdotus

Koska auto oli jokapäiväisessä käytössä projektin aikana, oli tarpeen tehdä ruiskun vaihtamisesta mahdollisimman helppoa. Tätä varten päätettiin valmistaa kokonaan oma johtosarja vanhan rinnalle. Projektia varten hankittiin käytetty ruiskun johtosarja, josta saatiin erivärisiä johtoja kytkentöjen helpottamiseksi. Tämän lisäksi tarvittiin liittimet, jotka sopisivat suoraan kiinni antureihin. Tutkimusten jälkeen sopiviksi liittimiksi osoittautuivat AMP Junior Timerit, joita hankittiin tarpeellinen määrä. Näissä liittimissä on lisänä hyvä suojaus kosteutta vastaan, mikä on auto-olosuhteissa erittäin hyvä asia. Muihin kohteisiin sopi tavallinen kaksinapainen naarasliitin, mutta kaasuläppää varten tarvittiin kolminapainen liitin.



KUVA 12. MegaSquirtin ulkoinen johdotus.
(Megamanual 2010)

Lisäilmaluistin lämmitystä ei kytketty MegaSquirtiin, vaan se toimii itsenäisesti aina kun autossa on virta päällä. Luonnollisesti tämä voisi aiheuttaa ongelmia, jos autossa pidetään virtaa päällä pitkään ennen käynnistystä ja ongelma tullaan korjaamaan tulevaisuudessa. Myös käyttöjännitteiden osalta oikaistiin alkuvaiheessa hieman. MegaSquirt ei ohjaa polttoainepumppua, vaan sen ohjaus kiertää alkuperäisen releen kautta aina kun virrat ovat päällä. Myöskään lambdatunnistimen kanssa MegaSquirt ei ole missään yhteydessä toistaiseksi.

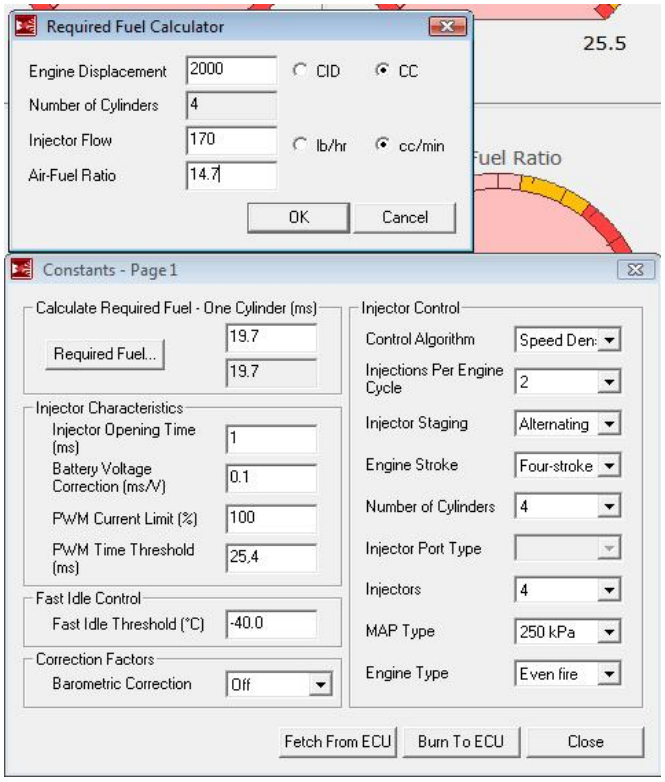
Johtosarjan kytkennässä käytettiin Megamanualissa olevaa kaaviota apuna sen selkeyden vuoksi. Johtosarjan valmistuksen jälkeen se kiinnitettiin nippusiteillä alkuperäisen johtosarjan rinnalle ja ohjausyksikön päähän tuleva DB37-liitin vietiin auton alkuperäisten läpivientien kautta apukuljettajan puolen jalkatilaan, johon MegaSquirt ECU on asennettu.

Johdotuksen lisäksi ohjainyksikkö tarvitsee tiedon imusarjan paineesta. Tämä tulee ohuella letkulla valmiista haarasta imusarjassa.

Näiden sovitusten jälkeen oli tarvittavat muutostyöt tehty ja ruiskun vaihto helppoa. Vaihtaminen ruiskujen välillä onnistuu vaihtamalla suuttimien ja antureiden liittimet eri johtosarjan liittimiin, vaihtamalla ilmamäärämittarin ohittava putki paikoilleen ja lopuksi irrottamalla vakioruiskun ohjainyksikkö liitimestään ja kiinnittämällä MegaSquirtin DB37 -liitin.

3.4 Testaus autossa ja säätäminen

Koska säätäminen on lähes mahdotonta tietämättä mitä moottorissa tapahtuu, lambdatunnistin on käytännössä pakollinen hankinta. Lambdatunnistimia on kahta tyyppiä, kapeakaistainen ja laajakaistainen. Kapeakaistainen ilmaisee, onko seos liian rikas vai liian laiha, kun taas laajakaistainen ilmaisee myös sen, kuinka paljon seos on vääränlainen. Tähän projektiin päätettiin hankkia laajakaista sen huomattavasti tarkemman mittauksen vuoksi. Tarkoitukseen tilattiin suhteellisen edullinen AEM seosmittari, johon kuuluu mukaan Boschin valmistama lambdatunnistin. Tällä yhdistelmällä on helppo seurata pakokaasujen jäännöshappea, koska tapahtumat näkyvät mittarissa koko ajan ja mittari näyttää suoraan seossuhteen. Mittarissa on myös mahdollisuus signaalin läpivientiin MegaSquirt-ohjausyksikölle, joka osaisi säätää itse arvojaan tietojen perusteella, mutta tätä ominaisuutta ei ole toistaiseksi otettu käyttöön. Tätä säätömahdollisuutta kutsutaan takaisinkytkennäksi (Closed loop). (Tranter 1997,139.)



KUVA 13. MegaTune moottorikohtaiset vakiot.

Säätö aloitettiin antamalla MegaTune-ohjelmalle sen tarvitsemat moottorin parametrit Engine Constants -valikosta. Virheilmoitusten välttämiseksi pitää ensimmäisenä säätää kohdalleen ruiskusuutinten asetukset eli Injector Characteristics. Injector Opening Time on aika, joka suuttimella kestää aueta sen jälkeen, kun siihen johdetaan sähkövirta. Tämä arvo on tyypillisesti kaikilla suuttimilla yksi millisekunti. Battery Voltage Correction on korjausarvo akkujännitteen vaikutukselle suuttimien aukeamisnopeuteen. Tässä jätettiin oletusarvo 0.1 ms/V, joka on useimmissa tapauksissa toimiva. PWM-asetukset on tarkoitettu matalaohmisille suuttimille, joita pitää ohjata hieman eri tavalla. PWM tulee sanoista Pulse Width Modulation ja tarkoittaa sitä, että suuttimelle menevää pulssia muokataan suuttimien rikkoutumisen estämiseksi, mikä matalaohmisissa suuttimissa johtuu ylikuumentumisesta. Koska tässä autossa on korkeaohmiset suuttimet, voidaan niitä ohjata suoraan. Siksi PWM-arvot asetetaan arvoihin 100 % ja 25.4 ms. Tämä tarkoittaa sitä että PWM on pois käytöstä ja suuttimille menevä ohjaus on vain päällä tai pois. (Megamanual 2010.)

Injector Control -kentissä määritellään perusarvoja, joiden avulla suuttimia ohjataan. Control Algorithm tarkoittaa arvoja, joiden perusteella MegaSquirt laskee polttoaineen tarvetta. Oletusarvona on Speed Density, joka on yleisin tapa. Se tarkoittaa sitä, että tarvittava polttoaine lasketaan kierrosluvun ja moottorin imusarjan paineen perusteella. Toinen vaihtoehto Alpha-N on tarkoitettu moottoreihin, joissa paineen mittaus ei ole käytännöllistä. Tässä valinnassa tarvittava polttoaine lasketaan kaasun asennon ja kierrosluvun perusteella, joka on luonnollisesti epätarkempaa koska se ei huomioi moottorin kuormaa. Tässä tapauksessa pysyttiin oletusarvossa Speed Density. (Megamanual 2010.)

MegaSquirt tarjoaa mahdollisuuden ohjata suuttimia erillisinä ryhminä. Injections per Engine Cycle määrittää montako kertaa polttoainetta ruiskutetaan yhden moottorin kierroksen aikana ja Injector Staging tarkoittaa tapaa, jolla ruiskutetaan. Tässä arvoiksi valittiin kaksi ruiskutusta per kierros ja Alternating. Tämä tarkoittaa sitä, että vain kaksi suuttimista on kerrallaan auki. Tällä saavutetaan pientä etua kahdessa asiassa: koska vain kaksi suuttimista on kerrallaan auki, on paineenvaihtelu

polttoainelinjassa pienempi ja koska kaksi suuttimista ruiskuttaa polttoaineen myöhemmin, se ei ehdi tiivistyä sumusta nesteeksi yhtä paljon ennen imeytymistään imukanavaan. Luonnollisesti suuttimet pitää tällöin kytkeä vastaamaan moottorin sytytysjärjestystä. Näiden asioiden hyöty on hyvin pieni, käytännössä usein merkityksetön, mutta näin tehtiin koska se on mahdollista. Tällä asetuksella pystytään asettamaan myös esimerkiksi kaikki suuttimet toimimaan yhtäaikaisesti, mutta niin, että kierrosta kohden polttoainetta suihkutetaan useamman kerran. (Himanen 2005, 44.)

Engine Stroke tarkoittaa moottorin tahtisuutta, ja koska kyseessä on nelitahtinen moottori, valittiin Four-stroke. Number of cylinders on sylintereiden lukumäärä, eli neljä. Port Type on uusista koodiversioista poistettu tarpeeton valinta, jolla määriteltiin suuttimien paikka imusarjassa. Injectors 4 tarkoittaa neljää suutinta. MAP type määrittää minkälaista ilmanpaineen anturia käytetään. Käytetyssä sarjassa mukana tullut anturi on MPX4250AP, joka on 250 kPa. Engine Type voi olla Even tai Odd Fire. Tämä tarkoittaa moottorin sytytysvälin tyyppiä. Useimmissa nelisylinterisissä moottoreissa sytytysväli on 180 astetta, mikä tarkoittaa että sytytysväli on tasajakoinen ympyrään nähden, eli tyyppi on Even Fire. (Himanen 2005, 44.)

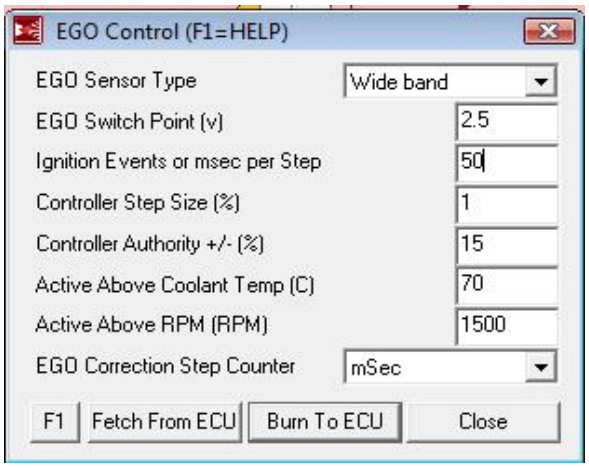
Barometric Correction mahdollistaa ruiskutuksen säätämisen ilmanpaineen mukaan. Ominaisuutta ei tässä otettu käyttöön koska Suomen korkeuseroissa painevaihtelut ovat hyvin merkityksettömiä. Fast Idle Treshold tarkoittaa lisäilmaluistin tai tyhjäkäyntimoottorin ohjauslämpötilaa. Tätä arvoa ei muutettu, koska tässä tapauksessa lisäilmaluistia ei ohjata MegaSquirtin avulla ollenkaan. (Megamanual 2010.)

Tärkein perustieto on Required Fuel, joka on perusarvo polttoaineen määrälle. Tämä arvo on yhden sylinterin tarvitseman polttoaineen ruiskutukseen kuluva aika millisekunteina, kun halutaan suurimmalla teholla paras hyötysuhde. Ohjelma laskee tämän neljän arvon avulla: iskutilavuus, sylintereiden määrä, suutinten tuotto ja ilman ja polttoaineen suhde. Tässä tapauksessa moottori on kaksilitrainen eli iskutilavuus on 2000 cc. Suutinten ilmoitettu tuotto on 170 kuutiosenttimetriä mi-

nuutissa. Sylintereitä on neljä, ja koska polttoaineena on bensiini, on ilman ja polttoaineen suhde 14,7. Näiden arvojen mukaan laskettuna tässä moottorissa Required Fuel on 19.7 ms.

Näiden asetusten jälkeen asetukset siirrettiin MegaSquirt ohjausyksikölle Burn to ECU-toiminnolla. Näillä perusasetuksilla kokeiltaessa moottori lähti käymään ensimmäisellä käynnistyksellä ja kävi jonkinlaista tyhjäkäyntiä. Koska suurin osa asetuksista oli vielä määrittelemättä, ei toiminta ollut tietenkään täydellistä, mutta lyhyen testiajon perusteella moottorin lämmettyä oltiin jo aika lähellä normaalia toimintaa.

Seuraavaksi oli aika tehdä tarkempia säätöjä. Vaikka tässä projektissa ei vielä ole testattu lambda-anturin avulla tapahtuvaa seoksen itsesäätymistä, laitettiin sen asetukset valmiiksi MegaTunen Exhaust Gas Settings-valikosta.

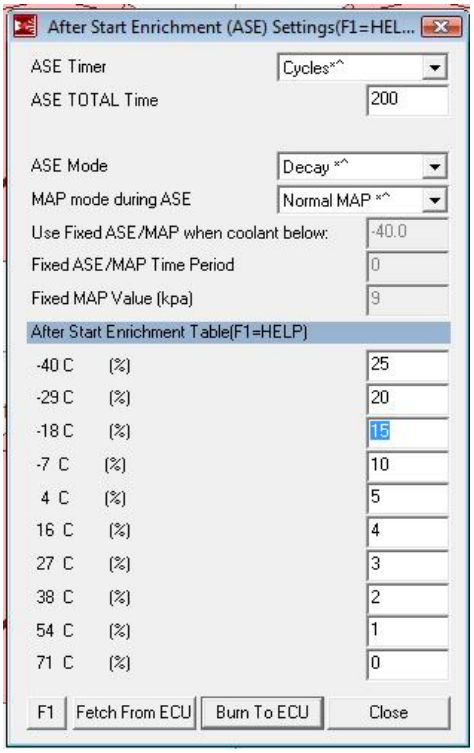


KUVA 14. MegaTunen lambda-asetukset.

Koska projektissa käytettävä lambdatunnistin on laajakaistainen, valittiin tyypiksi Wide band. EGO Switch Point tarkoittaa jännitteen arvoa, jota anturi pyritään saamaan lähettämään. Laajakaistalambdalla lambdakertoimen yksi jännitearvo on 2,5 volttia. Ignition Events or msec per Step tarkoittaa kuinka usein arvoa korjataan. Liian pieni arvo johtaa ylireagointiin, eli seosta yritetään korjata liian pienten vaihteluiden perusteella. Tässä arvo 50 on hyvä aloituskohta. Koska EGO Correction Step Counter on arvossa mSec, nämä asetukset yhdessä säätävät seosta 50 millisekunnin välein. Koska säätö haluttiin rajata pois kylmällä moottorilla sekä tyh-

jäkäynnillä, säädettiin alin lämpötila jossa säätö toimii (Active Above Coolant Temp) arvoon 70 astetta ja kierroslukuraja 1500 kierrokseen minuutissa. Controller Step Size tarkoittaa sitä, montako prosenttia polttoainekartan arvoa säädetään kerrallaan. Tämä asetettiin yhteen prosenttiin, joka on aivan riittävä arvo, koska silloinkin säätö voi muuttua jopa 20 % sekunnin aikana. Controller Authority -arvo tarkoittaa arvoa, jonka MegaSquirt maksimissaan voi lambdan avulla korjata. Tämä asetettiin arvoon 15 %, ettei mahdollinen anturin rikkoutuminen aiheuta liikaa ongelmia, vaan auto on yhä ajettavissa. Lopuksi asetukset siirrettiin MegaSquirt ECU:lle.

Vaikka säätöä tehtiin kesällä, haluttiin kylmempiä säitä varten tehdä jo alustavat säädöt kylmäkäynnistystä varten.



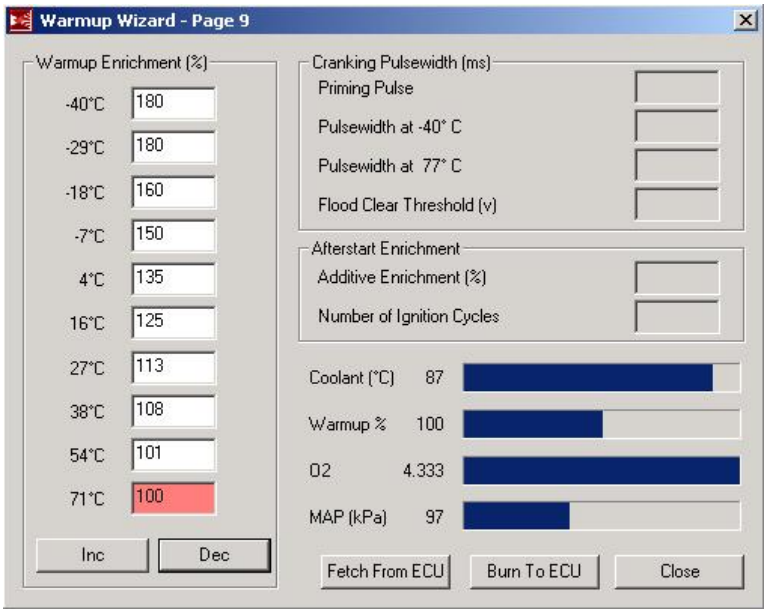
KUVA 15. MegaTunen käynnistysrikastus.

Tässä tarvittiin kahta eri toimintoa, After Start Enrichment ja Warmup Wizard. Ensimmäinen näistä on tarkoitettu estämään joillekin moottoreille tyypillistä huonoa käyntiä tai sammumista heti käynnistyksen jälkeen. Tämä johtuu siitä että käynnistyksessä moottori saa paljon polttoainetta, ja kun se alkaa käydä, polttoaineen

määrä vähenee. Käynnistyksen jälkeisellä rikastuksella moottoriin syötetään hieman ylimääräistä polttoainetta hetken ajan.

ASE Timer tarkoittaa minkä perusteella ylimääräistä polttoainetta syötetään. Tässä valittu Cycles tarkoittaa moottorin kierroksia. Sopiva arvo tähän osoittautui testeissä olevan 200 kierrosta. ASE Mode Decay tarkoittaa että kuvan 15 alareunassa olevaan taulukkoon syötetystä prosenttiluvusta alkava rikastuksen määrä vähenee kierros kierrokselta, kunnes se 200 kierroksen kohdalla on 0. Map mode during ASE jätettiin arvoon Normal, joka tarkoittaa että käytetään tässä syötettyjä arvoja. Arvolla Fixed Map voidaan poistaa ongelmia käynnistyvyydessä erittäin kylmässä ohittamalla tietyksi ajaksi paineanturin tieto kiinteällä arvolla. Alhaalla kuvassa 15 näkyvät arvot eri lämpötiloissa ovat vain arvioita, joita ei ole testattu pakkasasteiden puolella ollenkaan.

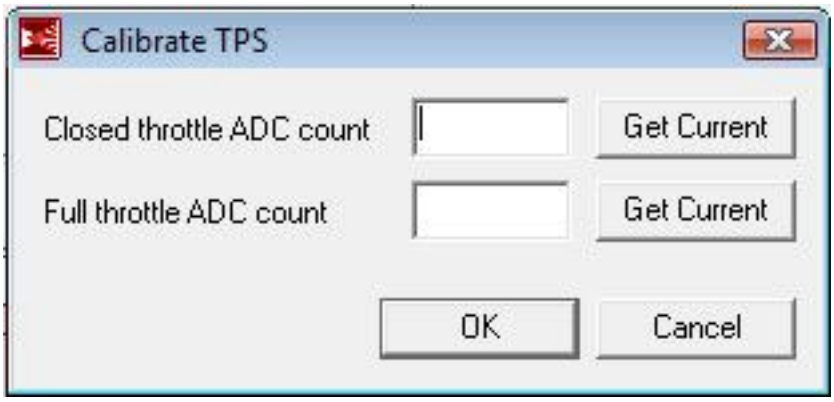
Toinen kylmissä olosuhteissa vaikuttava asia on se, että kylmä moottori tarvitsee enemmän polttoainetta, kunnes se lämpenee normaaliin käyttölämpötilaan. Tämä asetus on erillinen ASE:sta, joka säätää vain heti käynnistyksen jälkeistä tilannetta 200 kierroksen ajan, eli tämän auton tapauksessa noin 25 sekuntia (200 kierroksen ajan ja tyhjäkäynti noin 800 rpm). Warmup Wizard säätää jatkuvaa lisäpolttoaineen tarvetta eri lämpötiloissa.



KUVA 16. MegaTunen kylmäkäyntirikastus.

Tässä ohjelman ikkunassa ei ole muita säätöjä kuin prosentuaaliset rikastukset eri lämpötiloissa. Myös nämä arvot ovat vain jonkinlaisia lähtökohtia, joita ei ole kylmässä testattu.

Tässä kohtaa kalibroitiin vielä kaasuläppäanturi MegaTunen käyttöä varten. Tämä kalibrointi vaikuttaa vain ohjelman käytössä näkyvään kaasun arvoon, eikä MegaSquirtin sisäiseen ohjaukseen. Kalibrointi tehdään yksinkertaisesti painamalla ilman kaasua ylempää Get Current -näppäintä ja kaasu pohjassa alempaa näppäintä. Kun klikataan OK, näyttää MegaTune nyt mittareissaan kaasupolkimen asennon välillä 0-100 %.



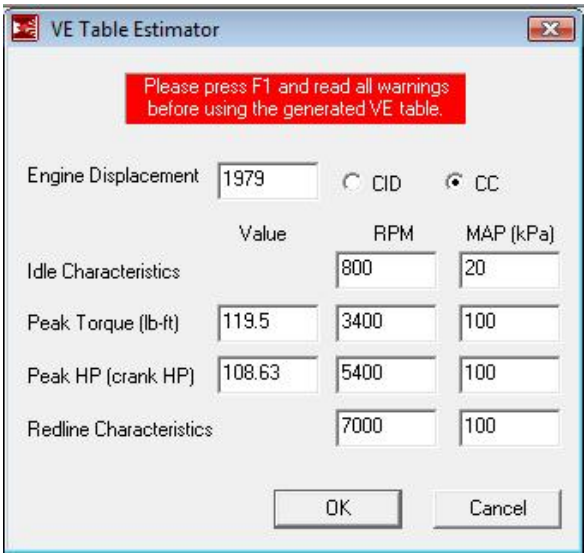
KUVA 17. MegaTunen kaasuläppäanturin kalibrointi.

KUVA 18. MegaTunen päänäkymä moottorin käydessä.

MegaTunen pääikkunassa on kahdeksan mittaria eri tarkoituksiin. Mittarit voidaan valita suhteellisen vapaasti näyttämään haluttuja arvoja. Esimerkiksi tässä Spark Advance on turha, koska sytytyksen ohjaus ei ole käytössä. Tämän mittarin tilalle on mahdollista vaihtaa esimerkiksi kaasun asento. Kun MegaTune on yhteydessä MegaSquirt ohjainyksikköön, näyttävät mittarit reaaliajassa moottorin tapahtumia.

MegaTune sisältää myös monia muita säätöjä, mutta ne kaikki jätettiin tässä vaiheessa oletusarvoihinsa. Suuri osa niistä on myös kokonaan tarpeettomia, koska tässä projektissa ei tehty sytytyksen ohjausta, vaan sen hoitaa auton alkuperäinen järjestelmä. Näiden viimeisten säätöjen jälkeen auton tyhjäkäynti oli asettunut hyväksi ja auto kävi tasaisesti 800 kierroksen paikkeilla. Testiajo osoitti, että myös muut asiat olivat melko hyvin kunnossa ja auto tuntui toimivan jokseenkin yhtä hyvin kuin alkuperäiselläkin ruiskulla. Seuraavaksi siirryttiin polttoainekartan hienosäätöön.

Aluksi luotiin pohjakartta MegaTunesta löytyvällä Generate Table -toiminnolla.

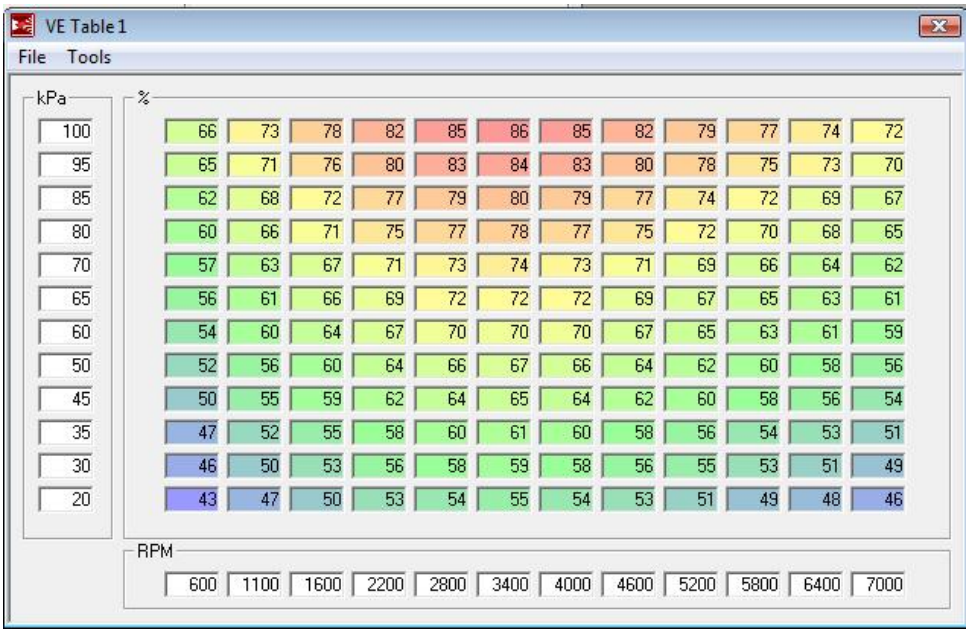


KUVA 19. MegaTune polttoainekartan luonti. (Megamanual 2010)

Tässä tarvittavat tiedot ovat iskutilavuus, tyhjäkäynnin kierrosluku ja imusarjan paine, maksimivääntö, sen kierrosluku ja imusarjan paine, huipputeho, sen kierrosluku sekä imusarjan paine ja lopuksi suurin kierrosluku, jolla kartan halutaan vaikuttavan.

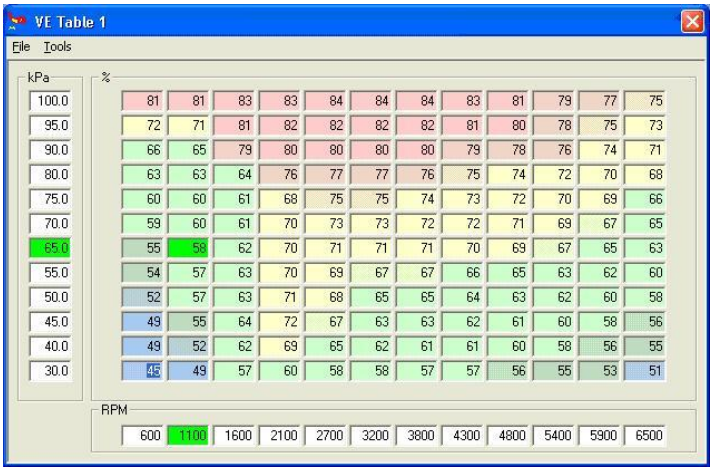
Opelin maahantuontitodistuksesta löytyivät tarkat arvot tähän, eli tehollinen iskutilavuus 1979 kuutiosenttimetriä, teho 81kW@5400rpm ja vääntö 162Nm@3400rpm. Nämä arvot muutettiin sopivaan muotoon eli kilowatit hevosvoimiksi ja Newton-metrit jalkapaunoiksi. (Adam Opel AG, 1984.)

Näiden arvojen perusteella ohjelma luo valmiin kartan, jota käytettiin säätämisen pohjana. Tämän kartan kanssa pitää olla varovainen, koska se ei ole vielä optimaalinen ja sillä ajaminen voi aiheuttaa ongelmia tai jopa moottoririkon.



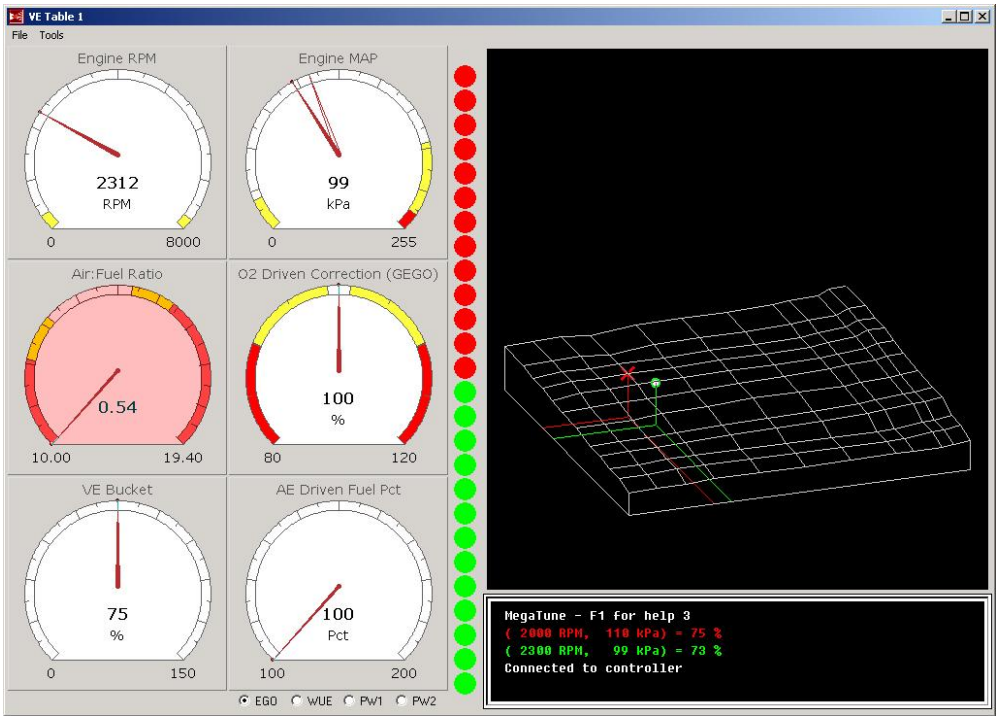
KUVA 20. MegaTune polttoainekartta.

Kuten kuvasta 20 näkyy, polttoainekartta on tässä versiossa 12x12. Tämä johtuu käytössä olevasta extra-koodista, alkuperäisessä versiossa kartta olisi 8x8. Kartta ilmaisee ruiskutettavan määrän tietyillä kierrosluvuilla ja imusarjan paineilla. Kuten tässä näkyy, vapaastihengittävässä moottorissa imusarjan paine on maksimissaan sama kuin vallitseva ilmanpaine (100 kPa). Taulukossa olevat arvot kertovat polttoaineen ruiskutusmäärän maksimiin verrattuna. Koska tämä taulukko ei ole optimaalinen, aloitettiin säätö varovasti läheltä tyhjäkäyntiä ja ilman kuormitusta, eli vaihde vapaalla. Auton käydessä vihreät laatikot ilmaisevat moottorin kierrosluvun ja imusarjan paineen, sekä kulloinkin käytössä olevan VE-arvon.



KUVA 21. MegaTune polttoainekartta auton käydessä.

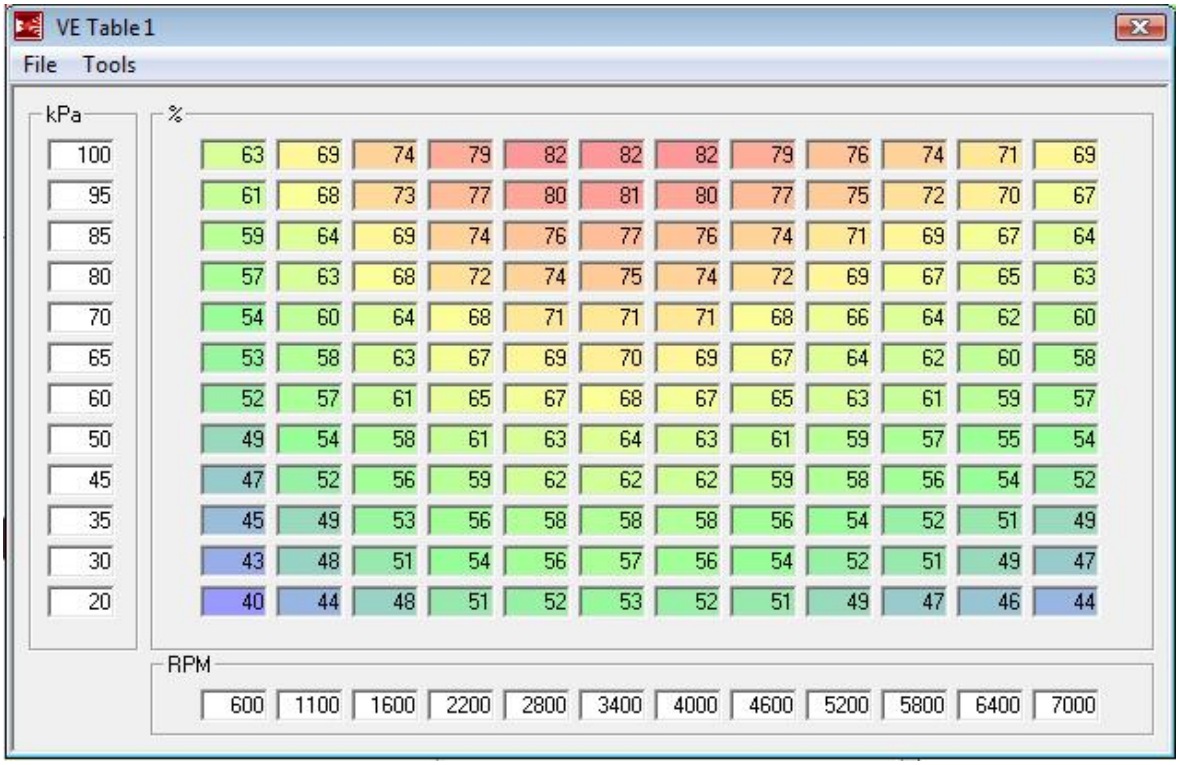
Kun vihreä laatikko on halutussa kohdassa, kuunnellaan moottorin ääntä ja tehdään johtopäätöksiä käynnin tasaisuudesta ja tarkkaillaan lambdamittaria. Jos moottori käy huonosti tai lambda-arvo on selkeästi liian laihalla tai rikkaalla, säädetään ruudussa olevaa lukua sen mukaan. Esimerkiksi jos lambda-arvo on 10,1 tilanteessa jossa moottori käy vapaalla 1100 kierrosta minuutissa, on seos selkeästi liian rikas. Tämä yleensä kuuluu myös käyntiäänessä ja pakokaasu haisee kitkerästi bensiinille. Jos sillä hetkellä käytössä oleva VE-arvo on esimerkiksi 65 %, täytyy arvoa pienentää. Tämä onnistuu suoraan kirjoittamalla ruutuun pienempi arvo, esimerkiksi 55 %, jonka jälkeen valikosta valitaan Burn to ECU. Nyt uusi arvo astuu heti käyttöön, ja lambda-arvon pitäisi muuttua selkeästi suuremmaksi (seos laihenee). Ilman kuormaa haluttu seos on mieluummin laihempi kuin stoikiometrinen arvo eli 14,7. Jos taas moottori on kuormitustilanteessa, eli kiihdytetään vauhtia vaihde päällä, on seoksen oltava rikkaampi.



KUVA 22. MegaTune graafinen polttoainekartta auton käydessä.

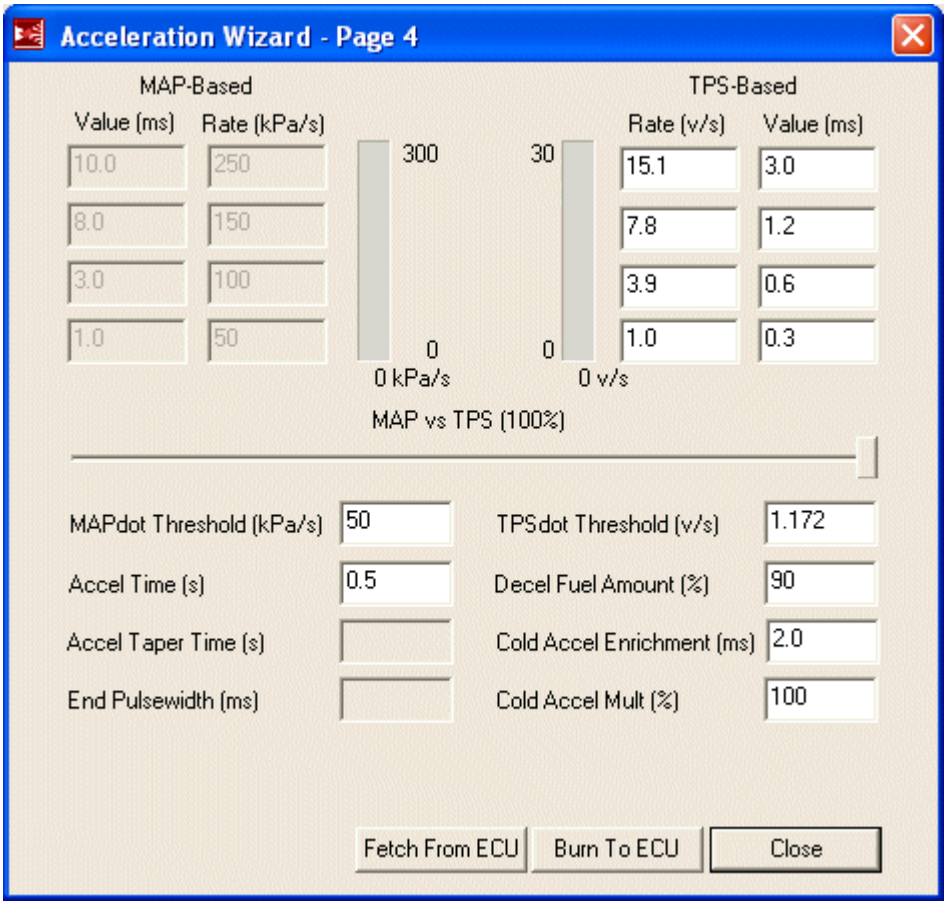
Arvoja on mahdollista säätää myös graafisesti, mutta tätä ominaisuutta ei hyödynnetty. Kuvassa vihreä pallo kuvaa moottorin senhetkistä tilannetta ja punainen rasti säädettävää käyrän kohtaa. Rastin kohdalla shift+nuolinäppäin ylös tai alas muuttaa kyseisen kohdan polttoainemäärää suuremmaksi tai pienemmäksi. Tässä ikkunassa voi myös aktivoida lambdatunnistimen avulla toimivan automaattisäädön.

Arvoja säädettiin ajamalla testiajoa ja paikallaan käyttämällä. Oikeiden arvojen etsintä olisi ollut hyvin vaikeaa ilman laajakaistalambdaa, mutta nyt mittarista saatiin suoraan tietoa mitä moottorissa tapahtuu. Lopputuloksena saatu VE-kartta ei eronnut paljoa MegaTunella luodusta. MegaTunen luoma kartta pyrkii olemaan mieluummin liian rikkaalla kuin laihalla, joten useissa ruuduissa arvoja sai laskea hieman, lisätä ei tarvinnut kuin parissa ruudussa. Useiden säätöajojen jälkeen moottori toimi jokseenkin hyvin, kiihdytystä lukuun ottamatta. Käytännössä kiihdytysten piti olla hyvin hillittyjä, tai auto alkoi selkeästi nykimään polttoaineen puutteesta.



KUVA 23. MegaTune valmis polttoainekartta.

Kun moottorin toiminta vaikutti nyt muuten olevan erittäinkin hyvällä tasolla suhteellisen kokemattoman säätäjän mielestä, voitiin ryhtyä säätämään kiihdytyksessä tarvittavaa lisärikastusta. Tämän rikastuksen tarkoitus on estää nykimistä, kun tarvitaan maksimikiihtyvyyttä. Koska kaasua nopeasti painettaessa kaasuläppä avautuu myös hyvin nopeasti, virtaa moottoriin paljon ilmaa, jonka kanssa tarvitaan myös paljon polttoainetta. Tätä säätöä varten MegaTunessa on toiminto Acceleration Wizard. Tähän liittyviä asetuksia on myös muualla ohjelmassa, mutta niihin ei koskettu.



KUVA 24. MegaTunen kiihdytysrikastus.

Kuvassa 24 keskellä oleva liukusäädin säätää suhdetta, jonka perusteella kiihdytysrikastus toimii imusarjan paineen ja kaasuläpän asentotunnistimen välillä. Tässä pysyttiin kokonaan kaasuläppäanturin ohjauksessa. Ylänurkan TPS-Based-arvot, Rate (v/s) ja Value (ms) ilmaisevat kaasun asennon muutosnopeutta ja lisärikastuksen määrää. Koska kaasuläpän asentoa mitataan suoraan jännitteenä, on muutosnopeus voltteja sekunnissa. Rate 1.0 v/s tarkoittaa kaasun painamista pohjaan viidessä sekunnissa. Lisärikastuksen määrä ilmaisee suuttimien aukioloajan lisäystä. TPSdot Threshold tarkoittaa arvoa, jolla kiihdytysrikastus aloitetaan. Tämä arvo on jännitteen muutosnopeus ja sillä pyritään eliminoimaan tahattomat kiihdytysrikastukset, jos kaasun asento muuttuu hitaasti. Decel Fuel Amount on prosenttiarvo polttoaineen määrän vähennykselle kaasua vähennettäessä. Tässä 90 % tarkoittaa että syöttöä pienennetään 10 %. Cold Accel Enrichment tarkoittaa arvoa, joka kylmissä olosuhteissa (pakkasella) lisätään ruiskutusajan laskentaan.

Tämä aika vähenee lineaarisesti lämpötilan kasvaessa. Cold Accel Mult on toinen tapa toteuttaa sama asia, siinä vain ilmaistaan lisäpolttoaineen tarve prosenteilla ajan sijasta. Muuten toiminta on sama kuin edellisessä kohdassa. MAP-puolen asetukset jätettiin oletusarvoihinsa, koska toistaiseksi haluttiin käyttää vain kaasun asentoon perustuvaa säätöä.

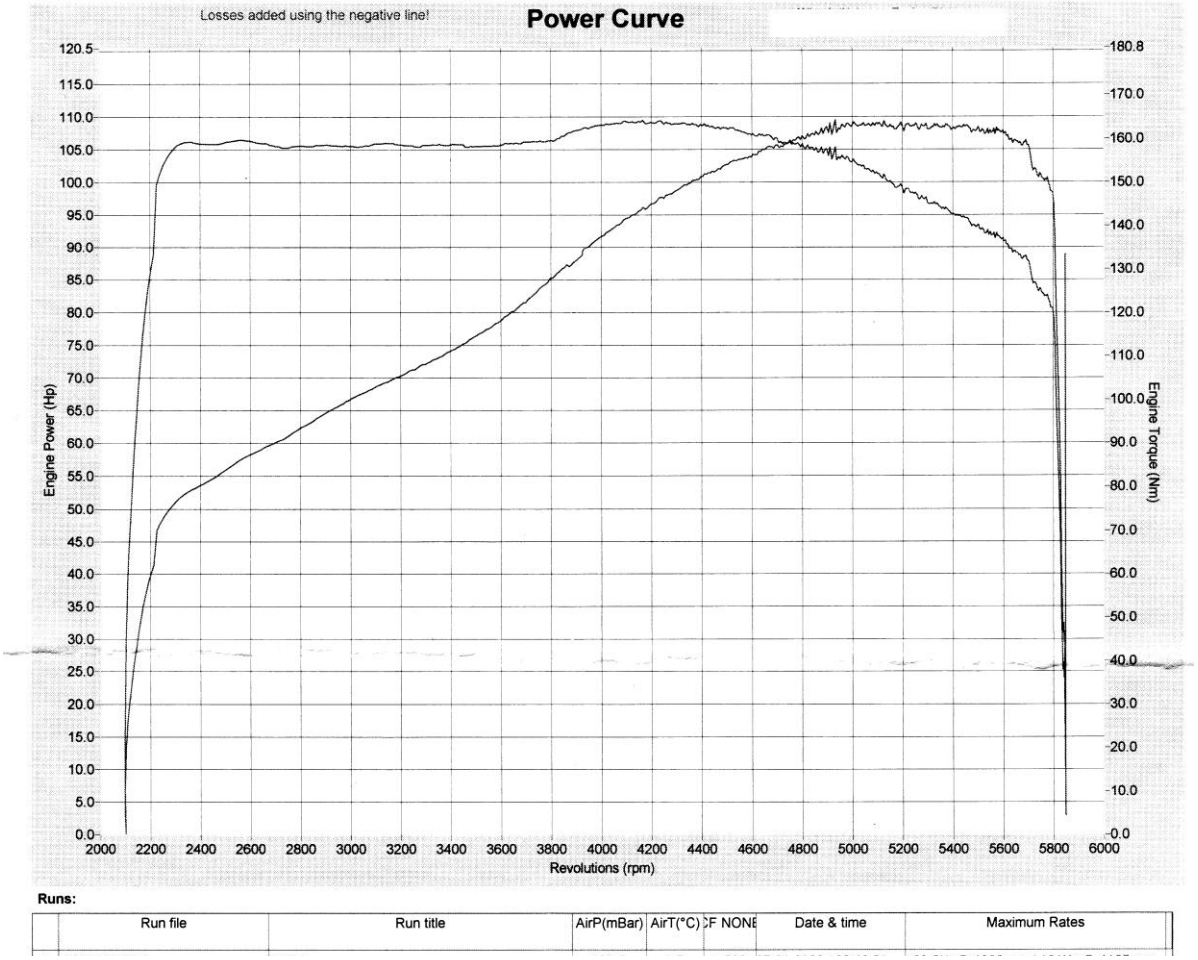
Acceleration Wizardiin syötetyt arvot kopioitiin suoraan toisen projektin sivuilta ja ne toimivat testissä niin hyvin, että niitä ei katsottu tarpeelliseksi muuttaa ainakaan tässä vaiheessa. Auto kiihtyi huomattavasti aikaisempaa paremmin ja nykiminen oli lähes olematonta kaikissa kiihdytystilanteissa.

Kun moottori tuntui toimivan kaikin puolin hyvin, vietiin se tehomittaukseen Autosähkö Matalamäen tehodynamometriin.

Ensin auto tehot mitattiin alkuperäisellä ruiskulla ja tulos oli muutaman kokeilun ja sytytyksen säädön jälkeen 112,7 hevosvoimaa kierroksilla 5259 rpm ja maksimivääntö 169 Nm@4056 rpm. Arvot ylittävät tehtaan lupaamat arvot hieman, mikä osoittaa että moottori on ikäisekseen hyvässä kunnossa.

KUVA 25. Tehodynamometrimittaus alkuperäisellä ruiskulla.

Seuraavaksi laitettiin MegaSquirt käyttökuntoon ja testattiin tehot uudelleen. Tässä vaiheessa ei tehty muutoksia säätöihin, koska säätöjä hiomaan oli tarkoitus kutsua myöhemmin eräs henkilö jolla on kokemusta kymmenien MegaSquirt-ruiskujen säätämisestä. Testissä tulos oli yllättävän hyvä 109,5 hevosvoimaa tehoa ja 164 Newton-metriä vääntöä. Tämä osoitti, että nytkään ei oltu kaukana optimaalisesta toiminnasta. Moottorin tehokäyrässä vakioruiskulla ollut tehon lasku 3000 kierroksen kohdalla oli käytännössä kadonnut, minkä pitäisi tuntua ajossakin.



KUVA 26. Tehodynamometrimittaus MegaSquirtilla.

Autolla ajettiin tällaisenaan parin viikon ajan ja tarkkailtiin kulutusta. Tätä tutkittiin tankkaamalla tankki aivan täyteen ja ajamalla tietyn pituisia matkoja ja seuraamalla paljonko polttoainetta tankista oli hävinnyt. Tämä mittaustapa ei ole kovinkaan tarkka, koska pelkästään tankin täyttöasteessa on jonkin verran eroja. Erot voivat riippua huoltoaseman pihan kaltevuudesta ja muista tankin asentoon vaikuttavista tekijöistä. Tämän lisäksi eroja aiheuttavat erilaiset kuormitustilanteet, vaikka pyrittiin ajamaan samoja reittejä kuin vakioruiskulla aikanaan. Tuloksia analysoidessa oli havaittavissa pientä eroa kulutuksessa MegaSquirtin hyväksi, jolla kulutus oli noin 8,9 litraa sataa kilometriä kohden. Alkuperäisellä ruiskulla kulutus oli aikanaan hieman alle kymmenen litran sadalle kilometrille. Vaikutti siis siltä, että ainakin tällä saralla MegaSquirt olisi vakioruiskua parempi, mutta mittaustarkkuuden huomioon ottaen oikea ero voi olla huomattavasti pienempi.

Valitettavasti tutkimuksia ei päästy jatkamaan pidemmälle, koska jotain hajosi ja MegaSquirt laski jostain syystä että ilman- ja moottorin lämpötilat olivat noin kolmenkymmenen pakkasasteen kohdalla säästä riippumatta. Tämän seurauksena moottori kävi aivan liian rikkaalla ja niin huonosti, että ajaminen oli mahdotonta. Vakioruisku laitettiin käyttökuntoon ennen auton viemistä tulevaan katsastukseen. Katsastuksessa löytyneiden muiden vikojen vuoksi auto otettiin pois tieliikennekäytöstä toistaiseksi ja siirryttiin tutkimaan varsinaista omaa osuutta, eli ohjelman tekemistä MegaSquirtin tietojen lukemiseen.

4 TESTIOHJELMA

Ruiskun toimintaan saattamisen lisäksi työssä tutustuttiin MegaSquirtin tietojen lukemiseen ohjelmallisesti, ajatuksena tuottaa oma ohjelma jolla pääsee käsiksi moottorinohjauksen käyttämiin tietoihin. Ohjelma päätettiin toteuttaa aluksi C#-kielellä, koska tästä oli eniten kokemusta.

4.1 Sarjaportin lukeminen

C# mahdollistaa sarjaportin käsittelyn hyvin helposti valmiilla SerialPort-komponentilla, jota tässäkin käytettiin hyödyksi. Suurin ongelma olikin saada selville missä muodossa MegaSquirt lähettää dataa sarjaporttiin. Vaikka MegaSquirt onkin kilpailevia järjestelmiä avoimempi, ei senkään toiminnasta ollut helppoa löytää tietoa. Pitkän etsinnän jälkeen löytyi Werner Haussmanin Internet-sivusto, jossa asiaa käsitellään.

Tässä lähestymistavaksi otettiin yksinkertaisin mahdollinen, eli MegaSquirtin tukema arvojen pyytäminen. Ohjelma voisi myös lukea dataa reaaliajassa, mutta alkuvaiheessa toiminta haluttiin pitää mahdollisimman yksinkertaisena, jotta saataisiin toimiva ohjelma aikaan.

Haussmanin sivuilta selvisi, että kun sarjaporttiin lähetetään merkki A, vastaa MegaSquirt tällöin siihen lähettämällä yhtenä ketjuna listan muistissaan olevista arvoista. Sen lisäksi selvisi että välissä tarvitaan noin 200 millisekunnin viive. Ilman viivettä MegaSquirt ei ole välttämättä ehtinyt muodostaa lukuketjua valmiiksi, jolloin sarja olisi epätäydellinen. Haussmanin esimerkit olivat Visual Basic -kielelle, joten niistä ei sen suurempaa apua ollut ohjelman toteutukseen. (Haussmann, 2009.)

Koska kyseisen lukusarjan tulkinta olisi voinut tässä kohtaa aiheuttaa ongelmia, päätettiin ensin lähestyä ongelmaa helpommalla tavalla. Selvisi että lähettämällä

sarjaporttiin merkin Q, lähettäisi MegaSquirt takaisin sisäisen koodinsa versiotiedon tekstimuodossa (Haussmann, 2009). Tästä tiedosta lähdettiin liikkeelle ohjelman tekoa varten.

MegaSquirtin säätämisen kautta oli selvillä, että sarjaportti pitäisi käynnistää arvoilla nopeus 9600 baudia sekunnissa, pariteetti None, 8 databittiä, 1 Stop-bitti. Lisäksi portin alustukseen tarvittiin muutama muu arvo, joissa päätettiin käyttää useimmissa ohjeissa löytyviä arvoja.

```
//configuring the serial port
#region SerialPort
serialPort1 = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);
serialPort1.Handshake = Handshake.None;

serialPort1.DataReceived += new SerialDataReceivedEventHandler(serialPort1_DataReceived);
serialPort1.ReadTimeout = 2000;
serialPort1.WriteTimeout = 2000;
//opening the serial port
serialPort1.Open();
serialPort1.DtrEnable = true;
#endregion
```

KUVA 27. Sarjaportin alustus.

Kuten kuvassa 27 näkyy, sarjaportti avataan heti alussa, jolloin yhteys on valmiina koko ajan. Sarjaporttiin lähetys tarvitsee vain yhden koodirivin: serialPort1.Write("Q"); Tämän jälkeen lisättiin aiemmin mainittu 200 ms viive: Thread.Sleep(200);.

Tässä vaiheessa MS on saanut käskyn Q, johon se vastaa lähettämällä versiotietonsa, joka tallentuu sarjaportin puskurimuistiin. Puskurimuistista tämä arvo luetaan seuraavasti: string versio = serialPort1.ReadExisting();. Tässä tapauksessa arvo luetaan siis muuttujaan versio, josta se kutsutaan napin painalluksella esiin.

Ohjemaa ensimmäistä kertaa testatessa todettiin, että jostain syystä serialPort1.ReadExisting(); ei koskaan sisällä mitään tietoa. Asiaa tutkittaessa selvisi, että nämä komennot eivät päde tässä käytettyyn MegaSquirt v2.2 extra - versioon. Oikea komento tässä versiossa on "S". Kun ohjelma muutettiin lähettämään oikea

merkki, ilmestyi näytölle vihdoin odotettu "MS1/Extra format 029y3 ********". Tämä tarkoitti siis sitä, että yksinkertainen yhteys MegaSquirtin ja tietokoneen välillä oli olemassa ja toimi molempiin suuntiin.

4.2 Datan tulkitseminen

Kun yhteys oli saatu toimimaan, seuraavassa vaiheessa ohjelma piti saada pyytämään MegaSquirtilta sen laskentaan käyttämät arvot. Tässä törmättiin samaan ongelmaan kuin aikaisemmin; tarvittava komento oli muuttunut. Haussmanin sivuilta onneksi selvisi, että tarvittava komento voisi löytyä MegaTunen tiedostojen avulla. Sivulla viitattiin tiedostoon msns-extra.ini, jota tutkimalla löytyi kohta [OutputChannels]. Tämän alla oli rivi, jossa luki: ochGetCommand = "R" ; was "a" and before that "A". Näin päästiin eteenpäin asiassa. Kun koodiin muutettiin lähetettäväksi dataksi "R", vastasi MegaSquirt heti pyyntöön lähettämällä muistissaan olevat arvot. Tässä kohtaa lukujono ei vielä ole ihmiselle ymmärrettävässä muodossa, koska MegaSquirt lähettää kaikki arvonsa yhtenä muuttujana, joka pitää purkaa osiin ja skaalata. (Fahlgren, 2009.)

MegaSquirt lähettää datan yhtenä taulukkona, koska datan täytyy liikkua tehokkaasti, ettei se hidasta muuta toimintaa. Taulukossa 1 on täydellinen listaus arvoista, joita MegaSquirt lähettää tällä komennolla.

Kentässä muuttuja on periaatteessa nimi, jolla ohjelmat viittaavat tähän arvoon. Tyyppi kertoo missä muodossa arvo on tallennettu, esimerkiksi U08 on kahdeksanbittinen unsigned int (=etumerkitön kokonaisluku). Indeksi kertoo mistä taulukon kohdasta tämä kyseinen muuttuja pitää lukea. Arvon tunniste kertoo missä muodossa tieto on tallennettu, esimerkiksi "sec" tarkoittaa, että arvo on suoraan sekunteja, kun taas ADC tarkoittaa, että kyseessä on MegaSquirtin sisäinen jännitearvo. Skaala1 ja skaala2 ilmaisevat onko arvoa tarpeen skaalata muunnoksen yhteydessä. Skaalaus on tarpeen joissain tapauksissa siksi, että 0-255 on liian pieni väli ilmaisemaan joitain arvoja, jolloin ne skaalataan sopivalla kertoimella.

TAULUKKO 1. MegaSquirtin lähettämät arvot.(Fahlgren 2010, msns-extra.ini)

Muuttuja		tyyppi	indeksi	arvon tunniste	skaala1	skaala2
secl	= scalar,	U08,	0,	"sec",	1.000,	0.000
squirt	= scalar,	U08,	1,	"bits",	1.000,	0.000
engine	= scalar,	U08,	2,	"bits",	1.000,	0.000
baroADC	= scalar,	U08,	3,	"ADC",	1.000,	0.000
mapADC	= scalar	U08,	4,	"ADC",	1.000,	0.000
matADC	= scalar,	U08,	5,	"ADC",	1.000,	0.000
cltADC	= scalar,	U08,	6,	"ADC",	1.000,	0.000
tpsADC	= scalar,	U08,	7,	"ADC",	1.000,	0.000
batADC	= scalar,	U08,	8,	"ADC",	1.000,	0.000
egoADC	= scalar,	U08,	9,	"ADC",	1.000,	0.000
egoCorrection	= scalar,	U08,	10,	"%",	1.000,	0.000
airCorrection	= scalar,	U08,	11,	"%",	1.000,	0.000
warmupEnrich	= scalar,	U08,	12,	"%",	1.000,	0.000
rpm100	= scalar,	U08,	13,	"r100",	1.000,	0.000
pulseWidth1	=scalar,	U08,	14,	"ms",	0.100,	0.000
accelEnrich	= scalar,	U08,	15,	"ms",	1.000,	0.000
baroCorrection	= scalar,	U08,	16,	"%",	1.000,	0.000
gammaEnrich	= scalar,	U08,	17,	"%",	1.000,	0.000
veCurr1	= scalar,	U08,	18,	"%",	1.000,	0.000
pulseWidth2	= scalar,	U08,	19,	"ms",	0.100,	0.000
veCurr2	= scalar,	U08,	20,	"%",	1.000,	0.000
idleDC	= scalar,	U08,	21,	"%",	1.000,	0.000
iTime	= scalar,	U16,	22,	"s",	1.000,	0.000
advance	= scalar,	U08,	24,	"deg",	1.000,	0.000
afrtarget	= scalar,	U08,	25,	"ADC",	1.000,	0.000
fuelADC	= scalar,	U08,	26,	"ADC",	1.000,	0.000
egtADC	= scalar,	U08,	27,	"ADC",	1.000,	0.000
CltIatAngle	= scalar,	U08,	28,	"deg",	1.000,	0.000
KnockAngle	= scalar,	U08,	29,	"deg",	1.000,	0.000
egoCorrection2	= scalar,	U08,	30,	"%",	1.000,	0.000
porta	= scalar,	U08,	31,	"",	1.000,	0
portb	= scalar,	U08,	32,	"",	1.000,	0
portc	= scalar,	U08,	33,	"",	1.000,	0
portd	= scalar,	U08,	34,	"",	1.000,	0
stackL	= scalar,	U08,	35,	"",	1.000,	0
tpsLast	= scalar,	U08,	36,	"",	1.000,	0
iTimeX	= scalar,	U08,	37,	"s",	1.000,	0.000
bcDC	= scalar,	U08,	38,	"%",	0.3922,	0.000

Ohjelmaan toteutettiin ajastin timer1, joka aiheuttaa tapahtuman timer1.Tick sekunnin välein. Tämä timer1.Tick kutsuu funktiota PollSeconds(). Aina kun laskurin sekunti vaihtuu, PollSeconds() lähettää pyyntökäskyn, odottaa 200 millisekuntia, lukee arvot MegaSquirtilta, tekee tarvittavat laskutoimitukset ja päivittää arvot näytölle.

Ensimmäisenä päätettiin lukea juokseva arvo, joka kertoo sekunteina kauanko MegaSquirt on ollut käynnissä. Tämä arvo alkaa nollasta aina, kun virta on ollut poikki. [OutputChannels] kertoi, että MegaSquirt lähettää 39-merkkisen lukujonon, jossa ensimmäinen luku on haluttu sekuntiarvo, tyypiltään 8-bittinen unsigned int. Koska yhdellä bitillä voi olla kaksi tilaa(1 tai 0) ja bittejä on kahdeksan, erilaisia mahdollisuuksia luvulle on 256 (2^8 = 256). Tietokonemaailmassa lasku ensimmäinen arvo on yleensä 0, eli tässä tapauksessa laskuri alkaa nollasta, ja viimeinen arvo on 255.

Sarjaporttiin vastaanotettu data tallentuu muuttujaan buffer[], joka on taulukkomuotoinen. Koska sekuntilaskuri on ensimmäinen MegaSquirtin lähettämä tieto, sijaitsee se muuttujan indeksissä 0. Sieltä se saadaan esille esimerkiksi näin:

```
int seconds = buffer[0];
lbl_raw_timer.Text = seconds.ToString();
```

Tässä arvo luetaan puskurista muuttujaan seconds, josta se luetaan seuraavalla rivillä ruudulla näkyväksi tekstiksi. Ohjelmaa testattaessa sekunnit etenivät ruudulla juuri kuten kuuluikin, eli aina virran katketessa laskuri aloitti nollasta ja päättyi arvoon 255, jonka jälkeen se aloitti alusta.

4.3 Moottorin lämpötila

Huomattavasti sekunteja haasteellisempaa oli saada eri antureiden lämpötila-arvot näkyviin. Sama ongelma koskee kaasuläpän asentotunnistinta ja paineanturia. Antureiden tiedot tulevat MegaSquirtin lähetyksissä samalla tavalla, eli ADC-

arvona välillä 0-255. ADC tulee sanoista Analog to Digital Converter. Tämä tarkoittaa sitä, että MegaSquirt muuntaa analogisen jännitearvon kyseisen anturin yli digitaaliseen muotoon, 8-bittiseksi muuttujaksi. Koska antureita on erilaisia, ja ne tarvitsevat omanlaisensa skaalauksen, on näitä tietoja mahdoton muuntaa suoraan lämpöasteiksi. Näiden arvojen muuntaminen vaatii ulkoista tietoa MegaTunenkin käyttämistä asetustiedostoista. Esimerkiksi thermfactor.inc pitää sisällään lämpötilat jokaista MegaSquirtin arvoa kohden.

```
; gentherm2.exe v.2.2
;
; GM 121 463 12 sensor data.
; For use with 2490 ohm bias resistor installed at R7.
;
; Known Steinhart-Hart coefficients: A=0.00149032  B=0.00022746  C=1.1639e-007
            ; ADC - Temp (dF)
    DB  210T    ;   0 - sensor failure.
    DB  475T    ;   1 -  435.4
    DB  409T    ;   2 -  369.4
    DB  375T    ;   3 -  334.8
```

KUVA 28. Alkuperäinen thermfactor.inc.

Tiedostoissa olevat rivit ovat muotoa "DB 168T ; 65 - 128.4". Tässä 65 tarkoittaa ADC-arvoa (arvo jonka MegaSquirt palauttaa) ja 128.4 on lämpötila Fahrenheit-asteina.

Tiedoston lukemisen lisäksi päätettiin tehdä algoritmi, joka muuntaa tiedostot sopivampaan muotoon. Tämä palvelee ajatusta siitä, että ohjelma julkaistaan joskus täysin vapaaseen käyttöön, jolloin jokainen voi käyttää omia valmiiksi kalibroituja anturitietojaan.

Tiedostojen käsittelyä varten ohjelmaan tehtiin uusi luokka Filehandler.cs. Tämä luokka avaa tiedostot, kopioi niiden tiedot muuttujiin ja tarvittaessa tallentaa uuden tiedoston. Ensimmäisenä tarkistetaan löytyykö kansiosta jo muokattu versio tiedostosta. Uusi versio on eroteltu vanhasta lisäämällä tiedostonimen loppuun ennen tiedostopäätettä kolme i-kirjainta. Jos tiedostoa ei löydy, avataan alkuperäinen tiedosto. Algoritmi hakee kohdan, jossa kerrotaan mitä rivillä olevat

arvot tarkoittavat ja aloittaa sitten seuraavalta riviltä prosessin, jossa jokaiselta riviltä karsitaan ylimääräiset merkit.

Lopputuloksena tallennetaan uusi tiedosto, jossa jokaisella rivillä on vain ADC-arvo ja lämpötila eroteltuna puolipisteellä. Tämän jälkeen ohjelma sammuttaa itsensä ja se pitää käynnistää uudelleen. Koska muokattu tiedosto tämän jälkeen löytyy, luetaan se rivi kerrallaan taulukkoon coolant_table[] joka sijaitsee Filehandler-luokassa.

```
0;sensor failure,
1;435,4
2;369,4
3;334,8
4;311,7
5;294,7
6;281,2
7;270,1
8;260,7
9;252,6
10;245,4
```

KUVA 29. Muokattu lämpötilakartta thermfactoriii.inc.

Tämän jälkeen on mahdollista muuntaa arvot ihmiselle sopivampaan muotoon. Tämä on hyvin yksinkertainen suoritus. Moottorin lämpötila sijaitsee taulukon buffer[] indeksissä 6.

```
double coolant_raw = buffer[6];
lbl_raw_coolant.Text = coolant_raw.ToString();
double coolant = Find_Coolant(coolant_raw);
coolant = (coolant - 32) / 1.8;
lbl_coolant.Text = coolant.ToString("##");
```

Kun lämpötilan ADC-arvo on luettu sarjaportin puskurista, välitetään se funktiolle Find_Coolant(), joka vertaa arvoa coolant_table[] -taulukon arvoihin. Kun oikea rivi löytyy, funktio palauttaa ADC-arvoa vastaavan lämpötilan. Arvo on Fahrenheit-asteina, joten se pitää vielä muuntaa Celsius-asteiksi ja pyöristää ennen sen kirjoittamista ruudulle.

$$℃ = (℉ - 32) \div 1{,}8 \qquad (1)$$

4.4 Imuilman lämpötila

Imuilman lämpötila sijaitsee buffer[]-taulukon indeksissä 5. Sen käsittely toimii pääasiassa samalla tavalla kuin moottorin lämpötilankin, mutta sillä on oma taulukkonsa tiedostossa matfactor.inc. Tämä tiedosto muunnetaan sopivaan muotoon, jonka jälkeen se luetaan ohjelman sisäiseen taulukkoon. Funktio Find_Mat() etsii ADC-arvolle lämpötilavastineen taulukosta mat_table[].

4.5 Kaasuläpän asento

Kaasuläpän asentoanturin lukeminen tapahtuu samalla tavalla kuin lämpötilatietojenkin. Asentoanturilla on oma tiedostonsa, josta ADC-arvoja vastaavat prosentit löytyvät. Tiedoston nimi on throttlefactor.inc. Tiedoston alku eroaa kuitenkin edellisistä, joten ajan säästämiseksi tehtiin uusi luokka Throttlehandler.cs, johon kopioitiin Filehandler.cs -luokan toiminnot sopiviksi muutettuina.

4.6 Imusarjan paine

Imusarjan painetta luettaessa turvauduttiin samaan keinoon kuin kaasuläpän asennonkin kanssa, ja tehtiin taas uusi luokka, Maphandler.cs, jonka toiminnot muokattiin Throttlehandlerin vastaavista toiminnoista. Painetaulukon tiedosto riippuu käytettävästä anturista, mutta tässä tapauksessa se on nimeltään kpafactor4250.inc.

4.7 Käyttöjännite

Yksi MegaSquirtin lähettämistä tiedoista on käyttöjännite(batADC). Se löytyy buffer[]-taulukon indeksistä 8. Käyttöjännite ei tarvitse erillistä muuntotaulukkoa, vaan

sen muuntamiseksi analogiseen muotoon tapahtuu yksinkertaisella kaavalla, joka löytyy msns-extra.ini-tiedostosta.

$$V = \frac{batADC}{255.0} * 30 \quad (2)$$

Koska tämä luku sisältää suuren määrän desimaaleja, muutettiin se ruudulla näyttämisen yhteydessä kahden desimaalin tarkkuuteen.

```
double bat = bat_raw / 255 * 30;
lbl_bat.Text = bat.ToString();
lbl_raw_bat.Text = bat_raw.ToString();
```

4.8 Moottorin tilat

MegaSquirt lähettää tietoa myös siitä, missä tilassa moottori on (käynnissä/starttaus/kiihdytys jne.). Tämä tieto on yhtenä lukuna, joka kertoo 7 eri asiaa. Tieto on ilmaistu bitteinä, eli onko muuttujan tietty bitti 1 vai 0. Tästä johtuen pitää näiden tietojen lukemiseksi muuntaa saatu arvo binäärimuotoon, mikä onnistuu C#-kielessä valmiilla Convert -funktiolla. Muunnos jättää ylimääräiset nollat pois alusta, joten tehtiin silmukka, joka lisää nollia alkuun kunnes binäärimuotoinen teksti sisältää kahdeksan merkkiä. Tämän jälkeen merkit luetaan merkkitaulukkoon, josta niitä on helpompi lukea. Testeissä huomattiin, että jostain syystä saatu binäärijono oli aina väärinpäin, joten se käännettiin merkkitaulukkoon lukemisen jälkeen komennolla Array.Reverse(). Bittien merkitykset ovat:

0	*running (ready)*	-Onko MegaSquirt käynnissä
1	*cranking (crank)*	-Startataanko moottoria
2	*after start enrichment (ASE)*	-Onko käynnistyksenjälkeinen rikastus päällä
3	*in warmup (warmup)*	-Onko moottori lämmin vai lämpiämässä
4	*acceleration mode (accaen)*	-Kiihdytetäänkö
5	*in deceleration mode (accden)*	-Moottorijarrutetaanko
6		
7	*idle on (tentative)*	-Tyhjäkäyntimoottorin ohjauksen tila

(Haussmann 2010.)

Näiden bittien tilaa ilmaisemaan tehtiin ohjelmaan värilliset ruudut ilmaisemaan bittien päälläoloa vihreällä värillä. Jos MegaSquirt lähettää tähän engine -muuttujaan esimerkiksi arvon 13, on tämä binääriksi muunnettuna 1101. Kun ylimääräiset nollat lisätään, saadaan arvoksi 00001101, joka siis on väärinpäin. Tämä binäärijono siirretään merkkitaulukkoon ja käännetään ympäri. Näin taulukossa binaaritable on binäärijono 10110000. Tämä tarkoittaa siis sitä, että MegaSquirt on käynnissä (merkki 1), moottori on käynnistynyt (ei startata, merkki 2), käynnistyksenjälkeinen rikastus on päällä (merkki 3) ja moottori ei ole saavuttanut normaalia käyttölämpötilaa (warmup päällä, merkki 4). Moottori käy tasakaasulla (ei kiihdytystä eikä moottorijarrutusta, merkit 5 ja 6). Merkki 7 jätettiin huomiotta, koska tyhjäkäyntiä ei ohjata MegaSquirtin avulla.

```
string binary = Convert.ToString(engine, 2);
while (binary.Length < 8)
{ binary = ("0" + binary.ToString()); }
char[] binaaritable = binary.ToCharArray();
Array.Reverse(binaaritable);

if (binaaritable[0] == '1')
{lbl_engbit0.BackColor = System.Drawing.Color.Green;}
else
{lbl_engbit0.BackColor = System.Drawing.Color.Red;}
```

4.9 Muut arvot

Edellä mainittujen lisäksi luettiin useita muita tietoja, joiden lukeminen on lyhyt operaatio, koska niitä ei joko tarvinnut muuttaa eri yksiköiksi, tai muunnos oli lyhyt ja yksinkertainen.

```
double warm_raw = buffer[12];     Lämpötilan mukainen rikastus
double pulse1 = buffer[14];       Suuttimien pulssinleveys
double egocorr = buffer[10];      Lambdakorjauksen määrä
double vecurr1 = buffer[18];      Polttoainekartan arvo
double barocorr = buffer[16];     Ilmanpainekorjaus
double accelenrich = buffer[15];  Kiihdytysrikastus
double egotargetADC = buffer[25]; Tavoiteltava lambda
double egoADC = buffer[9];        Nykyinen lambda
```

Kuten taulukosta 1 voi todeta, arvo pulseWidth1 (ohjelmassa pulse1) ei ole suoraan oikeassa muodossa, koska sen skaala1 on 0,100. Tämä tarkoittaa että

oikea arvo saadaan jakamalla MegaSquirt-arvo kymmenellä. Näin saatu arvo on suuttimille menevä pulssinleveys, joka ilmaisee suuttimien aukioloaikaa millisekunteina.

Warm_raw on lämpötilasta riippuvainen seoksen rikastus, eli ryyppy. Tämän arvon MegaSquirt lähettää suoraan prosentteina, niin että 100 % tarkoittaa normaalitilannetta, eli ei yhtään rikastusta. Egocorr ilmaisee lambdakorjauksen määrää ja barocorr ilmanpainekorjauksen määrää. Nämä arvot toimivat samalla tavalla kuin warm_raw ja ovat suoraan prosentteja.

Accelenrich on kaasun asentoon perustuvan kiihdytysrikastuksen määrä. Tämä arvo on suoraan millisekunteja ja tarkoittaa aikaa, joka lisätään normaaliin suuttimien aukioloaikaan painettaessa kaasua.

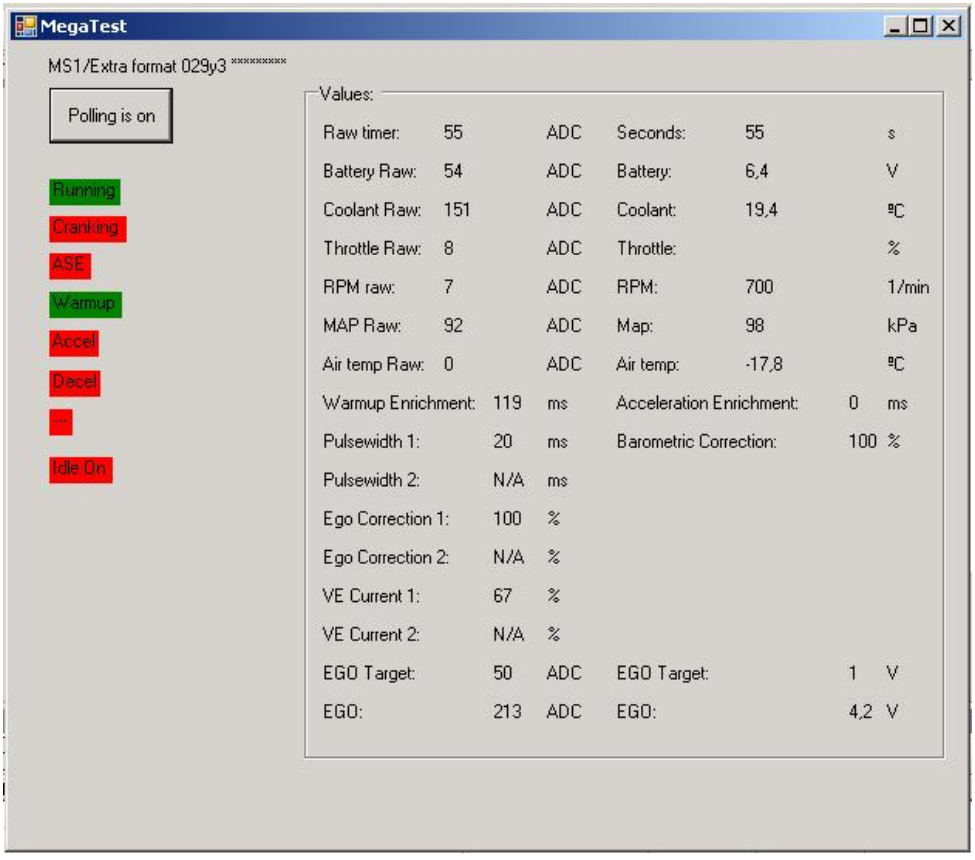
Vecurr1 tarkoittaa kyseisellä hetkellä käytössä olevaa polttoainekartan arvoa, eli suoraan polttoainekartasta kyseisen imusarjan paineen ja kierrosluvun kohdalta löytyvää prosenttilukua.

EgotargetADC on jännitearvo, johon lambda-anturin avulla pyritään seosta korjaamaan ja egoADC on arvo, jossa lambdajännite on kulloinkin on. Nämäkin arvot on ADC-arvoja, jotka eivät siis ole suoraan oikeita, vaan ne pitää muuntaa oikeaan muotoon kaavalla ADC / 255.0 * 5.0 (msns-extra.ini 2010). Näin saadaan volttiarvot lambdajännitteelle ja sen tavoitearvolle.

5 OHJELMA TESTISSÄ

5.1 Toiminta

Ohjelman käyttöliittymä on äärimmäisen yksinkertainen, sillä siinä on vain yksi nappi, jolla käynnistetään tai sammutetaan arvojen mittaus. Kaikki loput toiminnalliset ohjelman osat ovat lukukenttiä, jotka muuttuvat MegaSquirtin antamien arvojen mukaan.



KUVA 30. MegaTest.

Kun ohjelmaan oli saatu lisättyä tarvittava määrä toimintoja, aloitettiin testaaminen. Testissä kannettava tietokone oli sarjakaapelilla kytkettynä MegaSquirt-yksikköön ja sen DB37-liittimessä oli kiinni MegaStimulator. Kun MegaStimulatorin säätövastuksilla muutettiin jotain arvoista, päivittyi se ohjelman ikkunaan. Arvojen todenmu-

kaisuus tarkistettiin vertaamalla niitä MegaTunen näyttämiin arvoihin. Ohjelma toimi odotetusti ja arvot vastasivat MegaTunen lukemia. Ohjelmaa ei testattu autossa, mutta ei ole mitään syytä olettaa, etteikö se toimisi myös silloin, koska Megastimulator emuloi autossa olevien antureiden signaaleja.

5.2 Kehitettävää

Vaikka ohjelma toimikin, on siinä vielä paljon kehitettävää. Koska koodi on tehty kiireen alaisena, se ei ole selkeää eikä johdonmukaista, sisältäen suuren määrän turhia koodirivejä. Tämä johtuu pääasiassa siitä, että monet toiminnot on kopioitu suoraan toisesta vastaavasta toiminnosta muuttaen vain muutamia asioita funktioissa. Koodin siistiminen on siis tarpeen, että se olisi helpommin muidenkin luettavissa. Tämä on tarpeen, koska ohjelma on tarkoitus julkaista paranneltuna lähdekoodeineen muiden asiasta kiinnostuneiden saataville.

Pelkän siistimisen lisäksi myös ohjelman ominaisuudet kaipaavat kehittämistä, koska nyt siinä on vain hyvin alkeellisia lukutoimintoja, eikä sillä pysty kirjoittamaan mitään tietoja MegaSquirtin suuntaan. Kaiken kaikkiaan erilaisia muuttujia joita MegaSquirt voi lähettää ja vastaanottaa on useita satoja(Fahlgren 2010, msns-extra.ini). Lisäksi on syytä korjata ohjelman kaatuminen jos luku on päällä ja sarjaporttiyhteys MegaSquirtiin katkeaa.

5.3 Kehittyneemmät toiminnot testissä

Koska ohjelmaa haluttiin kehittää sisältämään edistyneempiä ominaisuuksia, yritettiin myös lukea arvoja monimutkaisemmista taulukoista. MegaSquirt sisältää kaiken kaikkiaan 13 "sivua", joita se voi pyydettäessä lähettää ohjelmalle. Näiden sivujen lukumääritykset löytyvät msns-extra.ini -tiedostosta. Lähetettäessä sarjaportilla komento "P\001", MegaSquirtin pitäisi siirtyä käyttämään sivua 1. Kun sivu halutaan lukea, lähetetään komento "V". Tässä lukuyrityksessä otettiin testiin usei-

ta eri arvoja, joita tämä sivu sisältää. Lukuyrityksissä ei kuitenkaan missään vaiheessa onnistuttu. Ohjelma kyllä vastaanotti joka kerralla arvoja, jotka tulivat MegaSquirt-ohjausyksiköltä, mutta arvot eivät koskaan olleet odotetunlaisia. Osa arvoista oli edes oikeansuuntaisia, mutta useimmat olivat useita satoja prosentteja vääränkokoisia. Tätä ongelmaa yritetään selvittää tulevaisuudessa, koska se on edellytys kirjoitustoiminnoille, jos halutaan kommunikoida MegaSquirtin kanssa molempiin suuntiin. Lisäksi näiden arvojen lukeminen mahdollistaisi ohjelman toimintojen lisäämisen, sillä hyvin useissa asioissa tarvitaan laskutoimituksia, joihin tarvittavia arvoja ei nykyisellä ohjelmalla päästä lukemaan ollenkaan.

6 PROJEKTIN INTERNETSIVU

Projektille tullaan avaamaan jossain vaiheessa Internet-sivu, jossa kerrotaan mahdollisesta etenemisestä, sekä jossa on tarkoitus julkaista ohjelman koodi avuksi muille, jotka haluavat yrittää oman ohjelman tekoa MegaSquirtin kanssa kommunikointiin. Sivusto tulee avautumaan osoitteessa http://incarnated.pp.fi/carstuff/megasquirt sen jälkeen kun tämä opinnäytetyö on hyväksytty.

7 YHTEENVETO

Tässä työssä haluttiin saavuttaa toimiva moottorikokoonpano MegaSquirt–moottorinohjausjärjestelmän avulla, sekä saada aikaan ohjelmistopohja joka osaa lukea ohjausjärjestelmän sisältä dataa. Ohjelma on tarkoitus julkaista vapaaseen käyttöön ja sitä on tarkoitus kehittää edelleen, jotta sillä saataisiin luettua kaikki mahdolliset arvot.

MegaSquirt asennettiin auton oman järjestelmän rinnalle, jolloin vanha järjestelmä jäi varalle ongelmatilanteita varten. Datan lukemista varten tehtiin yksinkertainen ohjelma, joka osaa lukea niitä tietoja, jotka MegaSquirt pyydettäessä lähettää, mutta reaaliaikainen kaikkien tietojen lukeminen ohjelmasta puuttuu.

Mekaaninen asennus ja säätäminen onnistuivat ilman suurempia ongelmia, kunnes lämpömittaukseen tuli vika. Koska tätä ennen järjestelmä oli kuitenkin saatu toimimaan, ei ongelmaa ole vielä ratkaistu. Ohjelmiston tekemisessä aiheutti suuria vaikeuksia se tosiasia, että vaikka MegaSquirt on suhteellisen avoin laitealusta kilpailijoihinsa nähden, ei sen tietoliikennettä ole dokumentoitu kattavasti. Kunnollisen dokumentoinnin puuttuessa tietoja joutui etsimään useasta paikasta ja pitkään.

Työssä saavutettuihin tuloksiin voidaan olla kaiken kaikkiaan melko tyytyväisiä, sillä auto saatiin toimimaan uudella järjestelmällä melko hyvin ja tuotettu ohjelmistokin täyttää tarkoituksensa, eli tarjota pohja parempien toimintojen kehittämiseen.

LÄHTEET

Adam Opel AG. 1984. Maahantuontitodistus Opel Manta 20E. [Viitattu 12.6.2008]. Saatavissa http://www.netikka.net/santanmanta/pics/muutos1.jpg

Anttila, J. 2006. Megasquirt Opeliin. [Verkkosivu]. [Viitattu 7.6.2009]. Saatavissa http://www.hellfish.org/~juho/mega/

Bauer, H. 2002. Autoteknillinen taskukirja. Suomentanut Heikki Haapaniemi. Jyväskylä: Gummerus.

Bell, A. G. 1998. Nelitahtimoottorin virittäminen. Suomentanut Esko Mauno. Helsinki: Alfamer Oy.

Bosch eCat 2010.[Verkkokatalogi]. [Viitattu 2.3.2010]. Saatavissa http://ecat-online.bosch.de/toc/frame1.html Tarkka polku: Suomi -> Moottorikoodi -> Valmistaja Opel, moottorikoodi 20E -> Ascona B

Bowling, B. & Grippo, A. 2010. Megamanual. [www-dokumentti]. Saatavissa www.megamanual.com

Chichak, M. 2002. Injectors. [Verkkokirja]. Saatavissa http://www.scribd.com/doc/12773746/Injectors

Fahlgren, E. 2008. MegaTune-ohjelmiston ohjeet ja alustustiedostot. [Viitattu 1.2.2010]. Saatavilla http://www.megamanual.com/files/software/

Himanen, O. 2005. MegaSquirt käyttöohje v.1.0. [Verkkokirja]. [Viitattu 15.5.2008]. Saatavilla http://www.finsquirt.net/manuaali/MegaSquirt_ohje_v1.pdf

Juurikkala, J., Airola, L., Pohjanpalo, Y. & Seppälä, P. 1986. Autotekniikan käsikirja: Polttoainelaitteet. Helsinki: Kustannusosakeyhtiö Tammi.

Juurikkala, J. 1987. TaitoTieto: Autokirja. Keuruu: Otava.

Probst, C. O. 1989. Bosch Fuel Injection and engine management. Massachusetts: Robert Bentley, Inc.

Ringwood, P. (daxtojeiro), Murrey, J. (jsmcortina), Calver, K. (muythaibxr) & Haussmann, W. 2008. RS-232 Communication with MS2/Extra. [Verkkosivu]. [Viitattu 20.8.2009]. Saatavilla

http://home.comcast.net/~whaussmann/RS232_MS2E/RS232_MS2_E.htm

Robert Bosch GmbH 1981.Bosch L-Jetronic Technical Manual. . [Verkkojulkaisu]. [Viitattu 1.5.2009]. Saatavissa http://bama.ua.edu/~darren/boschindex.html

TDC engineering / Chichak, M. 2002. Injectors. [Verkkojulkaisu]. [Viitattu 5.3.2010]. Saatavissa http://www.scribd.com/doc/12773746/Injectors

TM 1998. Autosanasto. Keuruu: Otava.

Tranter, A. 1995. Auton sähkövarusteet. Suomentaja Kari Kuurne. 2. painos. Teekkarien Autopalvelu.

LIITTEET

Liite 1. Oman ohjelman osittainen lähdekoodi kommentoituna

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.IO;
using System.Collections;

namespace MegaTest
{
    public partial class Form1 : Form
    {
        byte[] buffer = new byte[256]; //sarjaportin lukupuskuri
        string dataReceived = string.Empty;
        public int polling = 0;
        public string[] test_coolant_table = Filehandler.ReadThermfactor();    //ladataan taulukot muistiin
        public string[] test_mat_table = Filehandler.ReadMatfactor();
        public string[] test_throttle_table = Throttlehandler.ReadThrottlefactor();
        public string[] test_map_table = Maphandler.ReadMapFactor();

        private delegate void SetTextDeleg(string text);

        private static void Main()
        { Application.Run(new Form1()); }

        public Form1()
        {
            InitializeComponent();
            #region GetDataFromFile
            //pyynnöt ladata kalibrointikartat
            Filehandler.LoadThermFactor();
            Throttlehandler.LoadThrottleFactor();
            Maphandler.LoadMapFactor();
            #endregion
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //sarjaportin alustukset
            #region SerialPort
            serialPort1 = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);
            serialPort1.Handshake = Handshake.None;
            serialPort1.DataReceived += new SerialDataReceivedEventHandler(serialPort1_DataReceived);
            serialPort1.ReadTimeout = 2000;
            serialPort1.WriteTimeout = 2000;
            //sarjaportin avaus
            serialPort1.Open();
            serialPort1.DtrEnable = true;
            #endregion
        }


        private void btn_version_Click(object sender, EventArgs e)
        {try
                {
```

```
                #region Askversion
                  //testi onko portti auki
if (!serialPort1.IsOpen)
                  serialPort1.Open();
                  //versiopyyntö, megasquirt näkee S komentona lähettää versiotiedot
                  serialPort1.Write("S");
                  //viive että MS ehtii muodostaa datan
                  Thread.Sleep(200);
                  //luetaan versio bufferista ja näytetään ruudulla
                  string versio = serialPort1.ReadExisting();
                  lbl_version.Text = versio.ToString();
                              #endregion
                 //valitaan onko lukeminen päällä vai ei
                #region polling
                 if (polling == 0)
                 {btn_version.Text = "Polling is on";
                   polling = 1;
                   timer1.Enabled = true;}
                  else
                 {btn_version.Text = "Polling is off";
                   polling = 0;
                   timer1.Enabled = false;}
               #endregion
            }
            catch (Exception ex)
            { MessageBox.Show("error" + ex.Message); }
        }

        private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            string x = serialPort1.ReadLine();
            this.BeginInvoke(new SetTextDeleg(si_DataReceived), new object[] { x });
        }
        private void si_DataReceived(string data)
        {   dataReceived = data.Trim();     }
        private void PollSeconds() //suurin osa toiminnasta
        {
            try
            {
                #region SerialRequestData
                  if (!serialPort1.IsOpen) //jos portti ei auki, avataan
                  serialPort1.Open();
                  serialPort1.Write("R"); //pyyntökäsky lähettää dataa
                 #endregion
                 #region SerialReadReply
                  //odotellaan että MS ehtii mukaan
                  Thread.Sleep(200);
                  //luetaan vastaus bufferiin
                  serialPort1.Read(buffer, 0, (int)buffer.Length);
                 #endregion
                #region TestData
                  #region Readfrombuffer
                  int seconds = buffer[0];    //luetaan sarjaportin datat sopiviin muuttujiin
                  double coolant_raw = buffer[6];
                  double mat_raw = buffer[5];
                  double bat_raw = buffer[8];
                  double throt_raw = buffer[7];
                  double map_raw = buffer[4];
                  double warm_raw = buffer[12];
                  double pulse1 = buffer[14];
                  double pulse2 = buffer[19];
                  double egocorr = buffer[10];
```

```
        double aircorr = buffer[11];
        double vecurr1 = buffer[18];
        double vecurr2 = buffer[20];
        double egocorr2 = buffer[30];
        double barocorr = buffer[16];
        double accelenrich = buffer[15];
        double egotargetADC = buffer[25];
        double egoADC = buffer[9];
        double egoV = egoADC / 255.0 * 5.0; //muunnellaan arvot oikeaan skaalaan
        double egoTargetV = egotargetADC / 255.0 * 5.0;
        int engine = buffer[2];
        pulse1 = pulse1 / 10; //muunnellaan arvot oikeaan skaalaan
        pulse2 = pulse2 / 10;
        #endregion
        #region Seconds
        lbl_raw_timer.Text = seconds.ToString();
        lbl_timer.Text = seconds.ToString();
        #endregion
        #region Coolant
        lbl_raw_coolant.Text = coolant_raw.ToString(); //näytetään raaka adc-arvo
        double coolant = Tablefinder(coolant_raw,"coolant"); //kysytään taulukon vastine arvolle
        coolant = (coolant - 32) / 1.8;   //muunnetaan celsiusasteiksi
        lbl_coolant.Text = coolant.ToString("#.#"); //näytetään pyöristettynä
        #endregion
        #region IAT
        lbl_raw_mat.Text = mat_raw.ToString();
        double mat = Tablefinder(mat_raw,"mat");
        mat = (mat - 32) / 1.8;
        lbl_mat.Text = mat.ToString("#.#");
        #endregion
        #region Voltage
        double bat = bat_raw / 255 * 30; //muunnetaan volteiksi
        lbl_bat.Text = bat.ToString("#.#"); //näyttö ja pyöristys
        lbl_raw_bat.Text = bat_raw.ToString();
        lbl_raw_map.Text = map_raw.ToString();
        #endregion
        #region RPM
        double rpm_raw = buffer[13];
        lbl_rpm_raw.Text = rpm_raw.ToString();
        double rpm = rpm_raw * 100;  //skaalaus
        lbl_rpm.Text = rpm.ToString();
        #endregion
        #region Throttle
        lbl_raw_throt.Text = throt_raw.ToString();
        double throttle = Tablefinder(throt_raw,"throt");
        lbl_throt.Text = throttle.ToString("#.#");
        #endregion
        #region MAP
        lbl_raw_map.Text = map_raw.ToString();
        double map = Tablefinder(map_raw,"map");
        lbl_map.Text = map.ToString("#.#");
        #endregion
    #endregion
    #region Other data
    lbl_warm.Text = warm_raw.ToString();    //näytetään suoraan arvot
    lbl_accel.Text = accelenrich.ToString();
    lbl_baro.Text = barocorr.ToString();
    lbl_ego1.Text = egocorr.ToString();
    lbl_ego2.Text = "N/A";
    lbl_raw_ego.Text = egoADC.ToString();
    lbl_raw_egotarget.Text = egotargetADC.ToString();
    lbl_pulse1.Text = pulse1.ToString();
```

```
            lbl_pulse2.Text = "N/A";
            lbl_ve1.Text = vecurr1.ToString();
            lbl_ve2.Text = "N/A";
            lbl_raw_ego.Text = egoADC.ToString();
            lbl_raw_egotarget.Text = egotargetADC.ToString();
            lbl_egotarget.Text = egoTargetV.ToString();
            lbl_egov.Text = egoV.ToString();
            #region LED  //moottorin tilan ilmaisut
            lbl_test.Text = "";
            lbl_test2.Text = "";
            string binary = Convert.ToString(engine, 2); //muunnetaan binääriksi
            while (binary.Length < 8)
            { binary = ("0" + binary.ToString()); } //lisää nollia kunnes 8
            char[] binaaritable = binary.ToCharArray(); //taulukkoon
            for (int i = 0; i < binaaritable.Length; i++)
            { lbl_test.Text += binaaritable[i].ToString(); }
            Array.Reverse(binaaritable);   //käännetään ympäri koska muuten väärinpäin
            for (int i = 0; i < binaaritable.Length; i++)
            { lbl_test2.Text += binaaritable[i].ToString(); }
            if (binaaritable[0] == '1')
            {lbl_engbit0.BackColor = System.Drawing.Color.Green;} //eli jos bitti 0 = 1, vihreä, jos 0, punainen
            else
            {lbl_engbit0.BackColor = System.Drawing.Color.Red; }
            #region loput bitit
            //tässä muut bitit samalla tavalla
            #endregion
            #endregion
            #endregion
          }
            catch (Exception ex)
            { MessageBox.Show("error" + ex.Message); }
        }

        private double Tablefinder(double raakaarvo,string tyyppi)
         {   //etsii taulukosta haluttuja arvoja
            double arvo = 0;
            int counter = 0;  //arvojen alustukset
            int reverser = 0;
            string tmp = " ";
            String[] Splitti = { "0", "0" };
            string[] hakutable = { "0", "0" };
            if (tyyppi == "coolant")  //jos pyydetty tyyppi coolant, hakutaulukko on test_coolant_table
            {
               reverser = 0;  //0 eli muotoa alkuarvo - vastine, muuten vastine - alkuarvo
               hakutable = (string[])test_coolant_table.Clone();
            }
            #region muut tyypit
            //muut samalla tavalla tyypin mukaan
            #endregion
            while (counter < hakutable.Length - 1 && Splitti[reverser].ToString() != raakaarvo.ToString())
//etsitään kunnes löytyy
            {
               Splitti = hakutable[counter].Split(new Char[] { ';' }, 2);  //halkaistaan rivi kahti kohdassa  ;
               if (Splitti[reverser].ToString() != "0" && Splitti[reverser].ToString() != "255")
               {
                  if (reverser == 0)  //otetaanko alku- vai loppuosa halkaistusta tesktirivistä
                  { tmp = Splitti[1].ToString(); }
                  else
                  { tmp = Splitti[0].ToString(); }
                  arvo = double.Parse(tmp);
               }
               counter = counter + 1;
```

```
    }
    counter = 0;
    return arvo; //heitetään vastine pyytäjälle
}
private void timer1_Tick(object sender, EventArgs e)
{   //kutsutaan lukufunktiota aina kun "kello" naksahtaa
    PollSeconds();
}



private void btn_test_Click(object sender, EventArgs e)
{
    int throt_raw = 122;
    lbl_raw_throt.Text = throt_raw.ToString();
    double throttle = Tablefinder(throt_raw, "throt");
    lbl_throt.Text = throttle.ToString();
    int map_raw = 92;
    lbl_raw_map.Text = map_raw.ToString();
    double map = Tablefinder(map_raw, "map");
    lbl_map.Text = map.ToString("##");
    int mat_raw = 65;
    lbl_raw_mat.Text = mat_raw.ToString();
    double mat = Tablefinder(mat_raw, "mat");
    mat = (mat - 32) / 1.8;
    lbl_mat.Text = mat.ToString("##");
    int coolant_raw = 40;
    lbl_raw_coolant.Text = coolant_raw.ToString();
    double coolant = Tablefinder(coolant_raw, "coolant");
    coolant = (coolant - 32) / 1.8;
    lbl_coolant.Text = coolant.ToString("##");
    double bat_raw = 72;
    double bat = bat_raw / 255 * 30;
    double rpmraw = 33;
    double rpm = rpmraw * 100;
    lbl_rpm_raw.Text = rpmraw.ToString();
    lbl_rpm.Text = rpm.ToString();

    lbl_bat.Text = bat.ToString("##.##");
    lbl_raw_bat.Text = bat_raw.ToString();
    int seconds = 42;
    lbl_raw_timer.Text = seconds.ToString();
    lbl_timer.Text = seconds.ToString();
    double egoADC = 254;
    double ego = egoADC / 255.0 * 5.0;
    double egotADC = 254;
    double egot = egotADC / 255.0 * 5.0;
    lbl_egov.Text = ego.ToString();
    lbl_raw_ego.Text = egoADC.ToString();
    lbl_raw_egotarget.Text = egotADC.ToString();
    lbl_egotarget.Text = egot.ToString();
    lbl_test.Text = "";
    lbl_test2.Text = "";

    int engine = 22;
    string binary = Convert.ToString(engine, 2);


    while(binary.Length < 8)
    { binary = ("0"+binary.ToString()); }

    char[] binaaritable = binary.ToCharArray();
```

```
            for(int i = 0; i < binaaritable.Length;i++)
            {lbl_test.Text += binaaritable[i].ToString();}
            Array.Reverse(binaaritable);
            for (int i = 0; i < binaaritable.Length; i++)
            { lbl_test2.Text += binaaritable[i].ToString(); }

        }       //tällä napilla testataan funktioita yms
    }
    }
```

Filehandler.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.IO;

namespace MegaTest
{
    class Filehandler
    {    //julkisia muuttujia käsittelyn helpottamiseksi
        public static string[] arrLines;
        public static string[] arrLines2;
        public static string[] temp_table;
        public static string[] coolant_table;
        public static string[] mat_table;
        public static int hakurivi = 0;


        public static void LoadThermFactor()
        {

            if(File.Exists("thermfactoriii.inc")) //jos luotu jo uusi
            {LoadFileToArray("thermfactoriii"); //ladataan tiedosto muistiin
            TableConverter("thermfactoriii",1);} //luetaan taulukko talteen
            else if (File.Exists("thermfactor.inc")) //jos vain vanha
            {LoadFileToArray("thermfactor"); //ladataan tiedosto muistiin
             FileConverter("thermfactor");} //muunnetaan tiedosto sopivaan muotoon
            else { MessageBox.Show("ERROR!"); } //ei kumpaakaan

            if (File.Exists("matfactoriii.inc"))
            {LoadFileToArray("matfactoriii");
             TableConverter("matfactoriii",2);}
            else if (File.Exists("matfactor.inc"))
            {LoadFileToArray("matfactor");
             FileConverter("matfactor");}
            else { MessageBox.Show("ERROR!"); }
        }
        public static void LoadFileToArray(string filename)
        {StreamReader reader = new StreamReader(filename+".inc"); //avataan luikja ja tiedosto
```

```
        string strAllFile = reader.ReadToEnd().Replace("\r\n", "\n").Replace("\n\r", "\n"); //Luetaan rivit talteen
        arrLines = strAllFile.Split(new char[] { '\n' });
        reader.Close(); }

        public static void FileConverter(string filenamee)
        {   hakurivi = 0;
            #region Type1Conversion
                #region FindStartPoint
                //etsitään alkurivi
                string haku = "ADC - Temp";
                while (!arrLines[hakurivi].ToString().Contains(haku))
                {hakurivi++;} //seuraava rivi
                hakurivi = hakurivi + 1;   //+1 että luku alkaa vasta seuraavalta riviltä
                #endregion
                #region Preparetable
                int newindexlenght = arrLines.Length - hakurivi; //valmistellaan taulukko valmiiks
                temp_table = new string[newindexlenght];
                int x = 0;
                #endregion
                #region CopyTable
                // create a writer and open the file
                TextWriter tw = new StreamWriter(filenamee+"iii.inc");
                while (x < newindexlenght && hakurivi < 263) //luodaan uusi tiedosto ja kirjoitetaan rivit
                {
                    #region Split
                    //jako kahteen osaan kohdassa -, esim DB       48T              ; 226 -   8.1 --> "DB              48T
                       ; 226 " ja "  8.1"
                    String[] Split = arrLines[hakurivi].Split(new Char[] { '-' }, 2); //pakotus kahteen osaan, että
negatiiviset arvot toimii
                    string[] Split2 = Split[0].Split(new Char[] { ';' }); //jaetaan ensimmäinen osa uudelleen kohdassa ;
                    #endregion
                    #region Trim
                    char[] MyChar = { ' ' };//mitä poistetaan
                    Split2[1] = Split2[1].TrimEnd(MyChar); //poistetaan välilyönnit kaikista
                    Split2[1] = Split2[1].TrimStart(MyChar);
                    Split[1] = Split[1].TrimStart(MyChar);
                    #endregion
                    Split[1] = Split[1].Replace(".", ",");
                    temp_table[x] = Split2[1] + ";" + Split[1];//trimmatut tekstit uuteen taulukkoon ja ; erottamaan
                    tw.WriteLine(temp_table[x].ToString()); // kirjoitetaan rivi tiedostoon
                    hakurivi++;//seuraava rivi
                    x++;
                }
                tw.Close();// suljetaan tiedosto
                hakurivi = 0;
                x = 0;
                MessageBox.Show("File conversion ready, please restart program, shutting down...");
                Environment.Exit(0); //pakotetaan ohjelman sammutus
                #endregion
            #endregion
        }  //endfileconverter

        public static string[] TableConverter(string filenamee,int tabletype)
        {
            hakurivi = 0;
                #region Type0Conversion
                int newindexlenght = arrLines.Length;
                temp_table = new string[newindexlenght];
                coolant_table = new string[newindexlenght];
                mat_table = new string[newindexlenght];
                int x = 0;
                hakurivi = 0;
```

```
            string guu;
            while (hakurivi < arrLines.Length - 1) //-1 ettei mee ohi rajojen
            {   //jako kahtia kohdassa ;
                String[] Split2 = Filehandler.arrLines[hakurivi].Split(new Char[] { ';' }, 2);
                guu = Split2[1].ToString();
                temp_table[x] = guu; //kerätään väliaikaistaulukkoon
                hakurivi = hakurivi + 1;
            }
            if (tabletype == 1)
            { MessageBox.Show("Coolant values imported OK"); }
            if (tabletype == 2)
            { MessageBox.Show("Intake air temp values imported OK"); }
            hakurivi = 0;
            if (tabletype == 1)
            {coolant_table = (string[])temp_table.Clone();
            return coolant_table;} //kloonataan taulukko tyypin mukaan

            else if (tabletype == 2)
            {mat_table = (string[])temp_table.Clone();
             return mat_table;}
            else
            { return null; }
            #endregion
        }//endtableconverter

        public static string[] ReadThermfactor() //luetaan tiedosto muistiin
        {
            if(File.Exists("thermfactoriii.inc"))
            {
            StreamReader reader2 = new StreamReader("thermfactoriii.inc");
            string strAllFile2 = reader2.ReadToEnd().Replace("\r\n", "\n").Replace("\n\r", "\n");
            arrLines2 = strAllFile2.Split(new char[] { '\n' });
            }
            return arrLines2;
        }
        public static string[] ReadMatfactor()
        {
            if (File.Exists("matfactoriii.inc"))
            {
                StreamReader reader2 = new StreamReader("matfactoriii.inc");
                string strAllFile2 = reader2.ReadToEnd().Replace("\r\n", "\n").Replace("\n\r", "\n");
                arrLines2 = strAllFile2.Split(new char[] { '\n' });
            }
            return arrLines2;
        }

    }//endclass
}//endnamespace
```

Maphandler.cs ja Throttlehandler.cs ei esitellä tässä, koska ne toimivat juuri samalla tavalla kuin Filehandler.cs, ero lähinnä siinä minkä nimistä tiedostoa käsitellään.