Kalle Kantola

AUTON MITTARISTOSOVELLUS WEB –KEHITTÄJÄN TYÖKALUILLA

Opinnäytetyö
CENTRIA AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Syyskuu 2013

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Ylivieskan yksikkö	**Aika** Syyskuu 2013	**Tekijä/tekijät** Kalle Kantola
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn nimi AUTON MITTARISTOSOVELLUS WEB -KEHITTÄJÄN TYÖKALUILLA		
Työn ohjaaja FM Joni Jämsä		**Sivumäärä** 33
Työelämäohjaaja FM Joni Jämsä		

Opinnäytetyön tilaajana toimi Centria ammattikorkeakoulu ja se on kehitetty D2I –ohjelmaa varten. D2I-ohjelma käynnistyi huhtikuussa 2012. Sen tavoitteena on oppia hyödyntämään ja jalostamaan yhä kasvavaa tietomassaa uusilla, moderneilla tavoilla.

Opinnäytetyöni tavoitteena oli korvata autossa jo olemassa oleva mittaristo digitaalisella näytöllä. Tavoitteena on luoda teknologiademonstraatio ja pohtia mittariston digitalisoinnin tuomia mahdollisuuksia ja rajoituksia käyttäen web –kehittäjän työkaluja.

Web –kehittäjän työkaluiksi voidaan kuvailla HTML –sivunkuvauskielen, JavaScriptin ja CSS:n käyttöä. Mittariston näyttönä toimi seitsemän tuumainen Android –taulutietokone. Auton mittariston tiedot saadaan taulutietokoneelle langattomasti Bluetoothin ja auton OBD –väylään tulevan lukijan avulla. Toiminnallisuus, jota ei voitu toteuttaa JavaScript:llä, tehtiin käyttäen Phonegap –työkalua, joka mahdollistaa natiivin alustan toiminnallisuuden ja kirjastojen käytön JavaScriptillä.

Opinnäytetyöhöni ei kuulu mittariston käyttöliittymän suunnittelu, eikä käytettävyyden arviointi, vaan lähestyin opinnäytetyötäni teknisestä näkökulmasta. Sovelluksen käyttöliittymän on suunnitellut mediatekniikan opiskelija Johanna Hartikka.

Opinnäytetyön tuloksena syntyi järjestelmä, joka visualisoi auton mittariston digitaalisella näytöllä.

Asiasanat
Android, Bluetooth, CSS, HTML, Java, JavaScript, Phonegap

ABSTRACT

CENTRIA UNIVERSITY OF APPLIED SCIENCES	Date September 2013	Author Kalle Kantola
Degree programme Media Technology		
Name of thesis DASHBOARD APPLICATION FOR VEHICLES USING WEB DEVELOPMENT TOOLS		
Instructor M.Sc. Joni Jämsä		**Pages** 33
Supervisor M.Sc. Joni Jämsä		

This thesis was commissioned by Centria University of Applied Sciences and it was developed for the D2I program. D2I program started in April 2012. The aim of the program is to develop intelligent tools and methods for managing, refining and utilizing diverse data.

The goal of this thesis was to replace the existing meters in the vehicle dashboard with a digital display. The aim was to create a technology demonstration and to contemplate the (possibilities that is) opportunities brought by digitalization using web development tools.

The Web development tools include HTML, JavaScript and CSS. The display was chosen to be a seven inch Android tablet device. The meter information is brought to the device by A Bluetooth enabled OBD adapter. The functionality which was not possible to implement with JavaScript was developed using the Phonegap toolset, which provides the possibility of using native development libraries with JavaScript.

Designing the user interface and the assessment of the usability were not the aim of this thesis and thus, the thesis was made purely from the technical point of view. The user interface for the application was designed by Johanna Hartikka.

As a result of this thesis a system which visualizes the car meters on a digital display was created.

Key words
Android, Bluetooth, CSS, HTML, Java, JavaScript, Phonegap

KÄSITTEIDEN MÄÄRITTELY

ADT – Android Development Tools. Työkalut, joilla voidaan kehittää Android –sovelluksia.

Apache 2 – Ohjelmistolisenssi, joka sallii lähdekoodin käytön vapaasti, myös suljetun lähdekoodin projekteihin, kunhan säilyttää huomautuksen tekijänoikeuksista.

CSS – Cascading Style Sheets. Tyylisivukieli Www –dokumenttien tyyliohjeistukseen.

DLC – Data Link Connector. Tiedonsiirtoliitin, jonka avulla ajoneuvosta luetaan OBD:n mukaisesti tietoja.

EOBD – European On-board Diagnostics. Eurooppalainen standardi OBD2:sta.

HTML – Hypertext Markup Language. Merkintäkieli Www –dokumenttien ohjelmointiin.

ISM –taajuusalue – Industrial, Scientific and Medical. Radiotaajuuskaista, jonka käyttö ei vaadi erillistä lupaa.

JavaScript – Selaimessa suoritettava skriptikieli dynaamisten Www –dokumenttien ohjelmointiin.

JavaVM – Virtuaalikone, joka suorittaa java –tavukoodia.

JSON – JavaScript Object Notation. Yksinkertainen tiedonsiirtoformaatti.

LGPL – GNU Lesser General Public License. Vapaan lähdekoodin lisenssi, joka mahdollista kaupallisten, suljetun lähdekoodin omaavan tuotteen tekemisen, kunhan LGPL –lisensoituun komponenttiin ei tehdä muutoksia.

OBD – On-board Diagnostics. Ajoneuvon diagnostiikkajärjestelmä.

OBD PID – On-board Diagnostics parameter ID. Koodeja, joilla OBD –järjestelmästä voidaan lukea tietoja.

OBD2 – On-board Diagnostics 2. Uudempaan standardiin perustuva järjestelmä. Katso OBD.

Datex2 – Standardi, joka määrittää tiedonsiirtoformaatin iikennehallintakeskuksille.

TAULUKOT

1 JOHDANTO

Tässä opinnäytetyössä käyn läpi ne käytännöt, välineet ja keinot, joilla toteutin auton mittaristosovelluksen web- kehittäjän työkaluilla. Opinnäytetyön tilaajana toimi Centria ammattikorkeakoulu Oy ja sovellus kehitettiin Data to Intelligence ohjelman tarpeisiin.

Sovelluksen käyttöliittymän ja grafiikat suunnitteli Johanna Hartikka, eikä opinnäytetyöni laajuus kata käyttöliittymään tai grafiikkaan liittyviä seikkoja. Käsittelen kuitenkin joitain käyttöliittymään liittyviä asioita, jotka ovat sovelluskehityksen ja jatkokehitysmahdollisuuksien kannalta tärkeitä.

Ensimmäisessä luvussa käyn läpi käytetyn teknologian. Kehitin sovelluksen Android –mobiilialustalle käyttäen HTML –sivunkuvauskieltä ja siihen liittyviä kehitystyökaluja, kuten JavaScriptiä ja CSS:ää. Toiminnallisuudet, joita ei HTML:n avulla voinut tehdä, toteutin Phonegap –kirjastoilla ja Java –liitännäisillä. Mittariston tiedot saadaan auton OBD2 –väylästä erityisen adapterin avulla. Android –laite ja adapteri viestivät Bluetooth –teknologian avulla.

Toisessa luvussa käyn läpi laitteiston jota käytin kehitystyössäni. Laitteistoon kuului Android –taulutietokone, aiemmin mainitsemani OBD2 –lukija sekä OBD2 –väylän simulaattori. Lisäksi kerron ohjelmistoista, joita käytin kehitystyössä.

Kolmannessa luvussa kerron varsinaisesta sovelluskehityksestä ja selvitän tarkemmin järjestelmän arkkitehtuurin. Käyn läpi pintapuolisesti tarvittavan Java –koodin, joka tarvitaan HTML –sovelluksen luomiseksi. Lisäksi käyn läpi kuinka Phonegapia ja HTML –kieltä käyttäen luodaan liitännäisiä, joiden avulla voidaan käyttää Androidin Java –kirjastoista löytyviä toiminnallisuuksia. Tämä toiminnallisuus on erittäin tärkeää sovelluksen kannalta.

Neljännessä luvussa pohdin sovelluksen jatkokehitysmahdollisuuksia ja viidennessä luvussa summaan opinnäytetyöni ja pohdin, miksi web –kehittäjän työkaluilla halutaan kehittää sovelluksia. Toimikoon tämä opinnäytetyö

dokumentaationa kehittämälleni sovellukselle ja ohjeena heille, jotka haluavat sovelluksiaan web –kehittäjän työkaluilla tehdä.

2 TEKNOLOGIA

2.1 Android

Android on mobiililaitteille kehitetty vapaan lähdekoodin käyttöjärjestelmä ja ohjelmistopino, jonka kehittämisestä vastaa Open Handset Alliance (Open Handset Alliance 2013a). Androidia jaetaan avoimen lähdekoodin lisensseillä Apache 2 ja LGPL (Open Handset Alliance 2013d). Androidin markkinaosuus kaikista älypuhelinten käyttöjärjestelmistä oli 70% heinäkuussa 2013 (Lunden 2013), joten on siis ymmärrettävää, miksi sovelluskehittäjät ovat edelleen kiinnostuneita alustasta.

Jokainen Android sovellus ajetaan omassa Dalvik -virtuaalikoneessaan. Dalvik perustuu Java VM –virtuaalikoneeseen, mutta se on optimoitu mobiililaitteita varten. Jokainen sovellusten luoma virtuaalikone suoritetaan omassa prosessissaan. Android –sovellukset käännetään tulkatusta koodista, joten ne ovat vähemmän alttiita kaatamaan järjestelmän tai ajamaan sen toimimattomaan tilaan. (Darcey & Conder 2012, 29 – 31.)

Kun sovellus asennetaan, käyttöjärjestelmä luo sovellusta varten edelleen oman käyttäjätilin. Jokaisella sovelluksella on täten omat tiedostot tiedostojärjestelmässä, käyttäjätunnus ja tietoturvallinen toimintaympäristö. (Darcey & Conder 2012, 31.)

Jotta sovellus voisi päästä järjestelmän resursseihin käsiksi, täytyy erikseen pyytää oikeus jokaista resurssia varten. Näitä resursseja ovat muunmuassa toiminnallisuus tehdä puheluita, verkkoyhteydet tai käyttäjän paikkatiedot ja yhteystiedot. (Darcey & Conder 2012, 31.)

Tavallisesti Android sovellukset ohjelmoidaan Java –kielellä Windows, Linux tai Mac -alustalla. Open Handset Alliancen tarjoamilla Android –kehitystyökaluilla Java –koodin voi kääntää Android –paketiksi, joka voidaan asentaa Android –

käyttöjärjestelmällä varustettuun päätelaitteeseen, tavallisesti matkapuhelimeen tai taulutietokoneeseen. (Open Handset Alliance 2013b).

2.2 HTML

HTML on sivunkuvauskieli, jonka kehitys alkoi 80-luvulla fyysikko Tim-Berners Leen johdolla. Alkujaan tarkoituksena oli, että HTML:ää käytettäisiin sisäisten dokumenttien jakoon yrityksen sisäisessä verkossa, ja sitä kuvattaisiin yksinkertaisilla HTML tageilla, jotka edelleen tulkattaisiin visuaalisesti esitettäväksi selainsovelluksella. HTML kasvatti suosiotaan hyvin nopeasti ja vaikka HTML standardi on laajentunut yksinkertaisesta tekstin esittämisestä huomattavasti, se on säilyttänyt yksinkertaisen rakenteensa. HTML:n kehitys on määritelty eri versionumeroin ja tällä hetkellä käytössä on HTML 4.0 ja osa HTML5 –määrittelystä. Nykyään HTML –standardia kehittää W3C ja WHATWG.

Termiä ”HTML5” – käytetään usein iskusanan tavoin ja puhuttaessa yleisesti webin uusista suuntauksista ja tekniikoista (Korpela 2011, 4). Keskeneräiseksi standardiksi HTML5 on saanut erityisen paljon huomioita ja termin suurpiirteinen käyttö on herättänyt paljon hämmennystä. HTML5 on yksinkertaisuudessaan vain HTML –standardin seuraava kehitysaskel, joka ei syrjäytä vanhoja piirteitä, mutta määrittää uusia toiminnallisuuksia mahdollistaakseen edelleen yksinkertaisemman kehityksen kuluttajien suosiossa oleville monipuolisille web- sovelluksille (Facebook, Youtube jne.). Näitä uusia ominaisuuksia ovat muunmuassa videoiden esitys, paikkatieto ja uudentyyppiset lomakkeet. (Korpela 2011, 13.)

HTML:n tarjotessa uusia mahdollisuuksia monipuolisille sovelluksille, tarvitaan kuitenkin edelleen JavaScript –kieltä toteuttamaan toiminnallisuutta, esimerkiksi resurssien hankinnassa ja tyylitietojen reaaliaikaisessa päivityksessä, kuten myöhemmin tässä opinnäytetyössä todetaan (Korpela 2011, 14). Toiminnallisuutta voi rakentaa myös palvelimen ohjelmoinnilla, johon voi käyttää esimerkiksi PHP:tä tai Pythonia. HTML:n visuaalisuutta voi kehittää hyvinkin laajasti CSS –tyylimäärittelyillä.

2.3 Phonegap

Phonegap on vapaan lähdekoodin työkalu, jonka avulla mobiilialustoille voi luoda sovelluksia käyttäen HTML:ää, CSS:ää ja JavaScriptiä. Ohjelmallisesti Phonegapin avulla tehty sovellus toteutuu siten, että Androidin grafiikkaelementtien sijaan sovellus käsketään piirtämään WebView näkymä, jolla on samat ominaisuudet, kuin tietokoneen tai puhelimen WWW- selaimella. WebView näkymä käsketään hakemaan .html tiedosto, joka piirtyy sovellukseen kuten se selaimessakin piirtyisi. (Phonegap.com 2013a.)

Tämä ei kuitenkaan ole Phonegapin tärkein ominaisuus, sillä saman toiminnallisuuden pystyy toteuttamaan myös ilman Phonegapia Androidista valmiiksi löytyvillä työkaluilla. Phonegapin tärkein ominaisuus on mahdollisuus tehdä liitännäisiä, joiden avulla JavaScript –koodilla voi kutsua Androidin natiivia ohjelmakirjastoja. Phonegapin ollessa tarjolla iOS:lle, Androidille, Windowsille ja Windows Phonelle sekä monelle muulle käyttöjärjestelmälle, on mahdollista kehittää samalla HTML5 –koodilla useille alustoille. Liitännäisiä käytettäessä alustan oma koodi kuitenkin on kirjoitettava uudestaan, mutta työläs käyttöliittymäohjelmointi toimii pienillä muutoksilla suosituimmilla alustoilla. (Phonegap.com 2013a.)

2.4 Bluetooth

Langaton Bluetooth –teknologia on lyhyen matkan radioteknologia, joka on suunniteltu toteuttamaan langattomat yhteydet henkilökohtaisten kannettavien laitteiden välillä. Teknologian kehitys alkoi 90- luvun puolivälissä Ericsson Mobilen toimesta ja nykyään kehityksestä vastaa Bluetooth Special Interest Group (SIG). (Gehrmann & Smeets 2004, 3.) Tärkeimpiä Bluetoothin ominaisuuksia ovat lujatekoisuus, pienitehoisuus ja edullinen hinta (Bluetooth.com 2013a).

Bluetooth –teknologia välittää dataa lyhyillä etäisyyksillä käyttäen radiolähetystä. Bluetooth toimii vapaalla ISM –taajuusalueella (Industrial, Scientific, Medical), joka on alunperin tarkoitettu teolliseen, tieteelliseen ja lääketieteelliseen käyttöön.

Taajuus on 2,400 – 2,485 GHz ja käytössä on hajaspektritekniikalla ja taajuushyppelyllä varustettu kaksisuuntainen signaali. (Bluetooth.com 2013b.)

Bluetoothin toimintaetäisyys riippuu sovelluksesta ja minkä luokan radiota käytetään. Kolmannen luokan radioille luvataan metrin etäisyys, toisen luokan radioille luvataan kymmenen metrin ja ensimmäisen luokan radioille sadan metrin etäisyys. Luokan kaksi radiot ovat yleisimmin käytössä ja niiden lähetysteho on 2,5mW. Bluetooth on suunniteltu toimiman hyvin vähäisellä energiankulutuksella.

Bluetooth –standardi käsittelee yhteydet eri laitteiden välillä profiilien avulla. Profiili määrittää yhtenäiset säännöt laitteiden välillä. Bluetooth –profiileja voi rakentaa lisää vanhojen päälle, joka mahdollistaa jo olemassaolevien protokollien ja käytäntöjen uudelleenkäyttämisen Bluetooth –ympäristössä. Kaikkien profiilien pohjalla toimii generic access profile (GAP). GAP:n päällä toimii muunmuassa audio/video jakeluprofiili, PAN (personal area network) –profiili ja sarjaporttiprofiili. Sarjaporttiprofiili emuloi RS232 –standardin mukaista tietoliikenneporttia, joka käytännössä mahdollistaa minkä tahansa informaation siirtämisen laitteiden välillä (Persson 2004, 18). Tässä opinnäytetyössä käytetään Bluetooth –sarjaporttiprofiilia langattomaan viestintään.

2.5 Auton diagnostiikka

Uusissa henkilö-/pakettiautoissa käytetyt diagnostiikkajärjestelmät ovat OBD2 –standardin mukaisia. OBD –standardi kehitettiin alunperin Yhdysvaltojen Kalifornian ARB:ssä (Air Resources Board) seuraamaan joitakin päästöjenohjauskomponentteja (California Environmental Protection Agency 2009). Tätä alkuperäistä versiota kutsuttiin OBD1:ksi, ja se vaadittiin Yhdysvalloissa 1991 valmistuneisiin ja siitä uudempiin ajoneuvoihin. Standardi ei kuitenkaan ollut kovinkaan laaja, sillä OBD1 oli rajoittunut tarkkailemaan vain tiettyjä päästöihin liittyviä komponentteja. OBD2 kehitettiin tämän heikkouden korjaamiseksi (California Environmental Protection Agency 2009). Yhdysvalloissa OBD2 vaadittiin vuonna 1995 valmistuneisiin ja sitä uudempiin autoihin. Euroopassa standardin nimi on EOBD, ja se on tullut voimaan vuonna 2001

bensiinikäyttöisille ajoneuvoille ja vuonna 2003 dieselkäyttöisille ajoneuvoille (OBD.fi 2007). OBD –laitteiden maahantuojat kertovat EOBD- ja OBD2 –laitteiden olevan yhteensopivia keskenään (Elekma 2013) (Gendan Automotive Products 2014).

OBD2 –standardissa on määritelty tiedot, mitä diagnostiikkaportista on saatava. Suurin osa tiedoista on vikakoodeja, mutta saatavilla on myös joitain anturiarvoja. Kaikkien tietojen antaminen vapaaseen käyttöön ei kuitenkaan ole pakollista, eivätkä autonvalmistajat niitä yleensä olekaan antaneet. (OBD.fi 2007).

Data Link Connector (DLC) on tiedonsiirtoliitin, jolla autosta voi lukea OBD2:n mukaisesti tietoja. Tiedonsiirtoliitin tulisi löytyä 30cm:n etäisyydeltä ohjauspyörästä. DLC:hen voi liittyä siihen suunnitellulla liittimellä, joka liitetään päätelaitteeseen RS232:n, USB:n tai Bluetooth:n avulla. Luettava tieto on RS232 –standardin mukaista. (Elekma 2013).

3 LAITTEISTO

3.1 Taulutietokone

Vaatimusmäärittelyn mukaisesti täytyi valita Android 4.x käyttöjärjestelmällä varustettu taulutietokone. Taulutietokoneessa tuli myös olla Bluetooth -radio, jotta yhteyden luominen auton diagnostiikan kanssa olisi mahdollista. Taulutietokoneen tuli olla myös kooltaan mahdollisimman lähellä autossa olevaa mittaristoa. Markkinoilla ei kuitenkaan ole helposti saatavilla erityisen leveää, mutta matalaa näyttöä jonka voisi sovittaa mittariston tilalle. Totesimme seitsemän tuuman (7”) taulutietokoneen olevan sopivin vaihtoehto.

Projektipäällikön kanssa päädyimme seitsemän tuuman Samsung Galaxy Tab 2:een. Taulutietokoneen oleellisimpiin ominaisuuksiin kuuluu kaksiytiminen 1,0GHz prosessori, 1024x600 pikselin resoluutio, 4,000mAh akku, GPS ja Bluetooth v3.0. (Samsung 2012).

Valitsemamme taulutietokone ei ominaisuuksiltaan ole parasta mitä markkinoilla on tarjolla, eikä täten yliarvoi loppukäyttäjän laitteen suorituskykyä. Laitteen suorituskyky on kuitenkin olennaisessa osassa tässä sovelluksessa, sillä mittariston reaaliaikainen grafiikka toteutetaan HTML –tekniikoita käyttäen, joka voi tehottomammalla laitteella tuottaa suorituskykyongelmia.

3.2 OBD –lukija

Auton OBD- väylään tuleva sovitin oli Iso-Britannialaisen KBM Systems Ltd:n valmistama OBDKey –laite, joka oli opinnäytetyön tilaajalla jo valmiiksi. OBDKey tukee ELM –pohjaisista laitteista tuttuja komentoja, sekä kaikkia OBD2 viestintäprotokollia. OBDKey:n virta saadaan auton DLC –liittimestä, joten erillistä virtalähdettä ei tarvita. OBDKey –laitteita on tarjolla RS232, Bluetooth ja USB –liitettävyydellä. (KBM Systems Ltd. 2013). Tässä opinnäytetyössä käytimme Bluetooth –sovitinta. OBDKey:stä ei suuresti löytynyt mainittavaa dokumentaatiota, joten pääasiassa päädyin käyttämään toisen laitevalmistajan

vastaavaa dokumentaatiota.

KUVIO 1. OBDKey OBD –adapteri

ELM327 on Elm Electronicsin valmistama mikropiiri, jonka avulla on mahdollista toteuttaa ratkaisuja OBD –sovittimen luomiseksi. ELM327 ei siis ole loppukäyttäjälle valmis tuote. ELM327:aan perustuvia valmiita laitteita on markkinoilla kuitenkin paljon eri valmistajien toteuttamina. Yhteisen mikropiirin ansiosta kaikki laitteet kommunikoivat samalla tavalla auton diagnostiikan kanssa. Myös OBDKey perustuu samaan mikropiiriin.

Alustavan tutkimuksen Bluetooth OBDKey –laitteen avulla aloitin parittamalla Windows 8 PC:n ja OBDKey:n. Parituksen ollessa valmis, Windows loi automaattisesti virtuaalisen sarjaportin. Liityin tähän sarjaporttiin hTerm asiakasohjelman kanssa. ELM327:n dokumentoinnin mukaan laite on valmis käytettäväksi, kun se ilmoittaa valmiutensa tulostamalla rivit ”ELM327 v.x.x” ja ”>”. Jos näitä rivejä ei näy, kehotetaan käyttäjää antamaan laitteelle komento ”ATZ”, joka alustaa laitteen. Huomasin, ettei tuota versionumeroa näkynyt koskaan laitteeseen liityttäessä ja uskonkin sen tulostuvan heti, kun laitteeseen laite kytketään käyttöjännitteeseen. Tällöin on kuitenkin mahdotonta olla

kuuntelemassa Bluetoothin avulla, mitä laite yrittää kommunikoida. Se ei kuitenkaan tuota ongelmaa, sillä tämän sovelluksen ei tarvitse tallettaa laitteelle mitään pysyviä tietoja, joten laitteen resetointi aina ennen varsinaisen kommunikoinnin aloitusta ei ole ongelma, joten ”ATZ” –komento voidaan syöttää toiminnan vakauttamiseksi.

”ATZ” –komento ei aina kuitenkaan palauttanut vastaukseksi laitteen versionumeroa, joka tässä tapauksessa on ”OBDKey v1.30” eikä ”ELM327 v.x.x”. Laite kuitenkin resetoitui oikealla tavalla aina, jos resetointikomennon syötti muutaman kerran, sekunti edellisen jälkeen.

Alustuskomennon jälkeen käytin komentoa ”AT SP 0”, joka aiheuttaa laitteen automaattisesti huomaamaan autossa käytettävän viestintäprotokollan. Kun laite on löytänyt protokollan, vastaa se tulostamalla ”OK” –viestin terminaaliin.

Näiden asetusten jälkeen laitteelle voi syöttää PID –koodeja. PID –koodeja on useita kymmeniä, ja jokainen koodi vastaa jotain tiettyä ajoneuvoon liittyvää arvoa (SAE 1979.) Esimerkiksi standardin mukainen koodi ”01 0C”, vastaa auton kierroslukua. Kun koodi syötetään, kysyy adapteri auton diagnostiikasta sen hetkisen arvon ja tulostaa sen edelleen sarjaporttiin (Elm Electronics.)

Kaikki PID- koodit eivät toimi jokaisessa ajoneuvossa. Suurin osa koodeista ei myöskään ole oleellisia auton kuljettajalle näytettävän informaation kannalta. Koodit, joita päädyin työssäni käyttämään, pyytävät tiedot auton nopeudesta, auton kierrosluvusta, moottoriveden lämpötilasta ja polttonesteen määrästä. Auton kuljettajan kannalta muita oleellisia tietoja voisi olla esimerkiksi vilkkujen indikaattorit ja pitkien/lyhyiden valojen indikaattori. Näitä tietoja OBD2 –väylästä ei kuitenkaan saa luettua.

TAULUKKO 1. Käytetyt PID- koodit

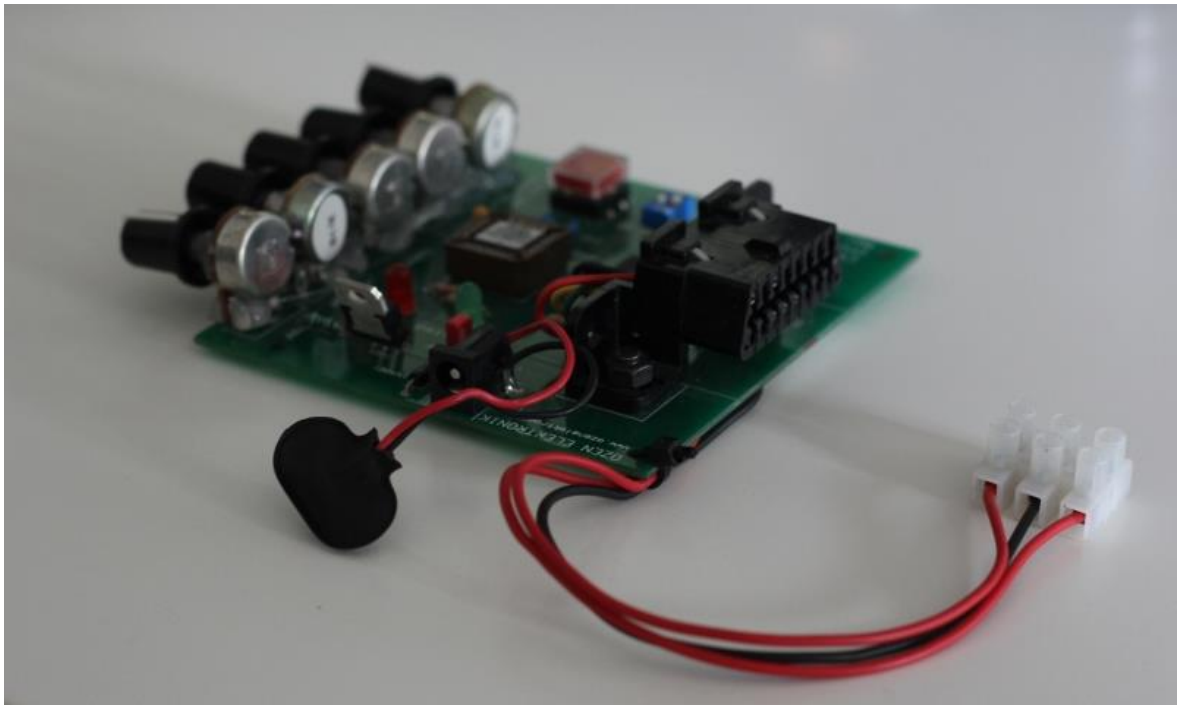
Tieto	**Koodi**
Nopeus	010D
Kierrosluku	010C
Polttonesteen määrä	012F

Jäähdytysnesteen lämpötila	0105

3.3 Diagnostiikkaväylän simulointi

Kehitystyön kannalta olisi erittäin epäkäytännölistä kehittää sovellusta autossa, jossa varsinaiseen diagnostiikkaan pääsee käsiksi. Jotta tältä vältyttäisiin, markkinoilla on tarjolla OBD2 –väylän simulaattoreita.

Centrialta löytyi valmiiksi Turkkilaisen Özen Elektronikin mOByDic –simulaattori. Simulaattorissa on DLC –paikka OBD –lukijalle, potentiometrit jäähdytysnesteen lämpötilan, kierrosluvun, nopeuden, jäännöshappimäärän ja virtaavan ilman massan muuttamiselle. Simulaattori sisälsi myös joitain arvoja, joita ei voinut muuttaa. Esimerkiksi polttonesteen määrää ei voinut muuttaa. (Özen Elektronik).



KUVIO 2. Özen Elektronik:n mOByDic –simulaattori

3.4 Työkalut

Android –ohjelmointi toteutettiin Open Handset Alliancen tarjoamalla ADT Bundle työkalupaketilla. Pakettiin kuuluu Eclipse IDE ADT pluginilla, Android SDK Tools,

Android Platform –tools ja Android –emulaattori sekä uusin käyttöjärjestelmälevykuva. (Open Handset Alliance 2013c).

ADT Bundlen lisäksi täytyy asentaa Java Development Kit ja asettaa Java työkalut järjestelmämuuttujiin. Tämän jälkeen Android –kehitys voi alkaa.

Sovelluksen HTML-, CSS- ja JavaScript- koodin ohjelmointiin käytin Sublime Text 3 tekstinkäsittelyohjelmistoa. Terminaaliohjelmana OBD2 –tietojen keräämiseen testin aikana käytin hTerm ohjelmistoa. Sovelluskehitystä nopeuttaakseni testasin osan HTML –koodista www –selaimella sen sijaan, että koko projekti olisi käännetty ja asennettu taulutietokoneelle.

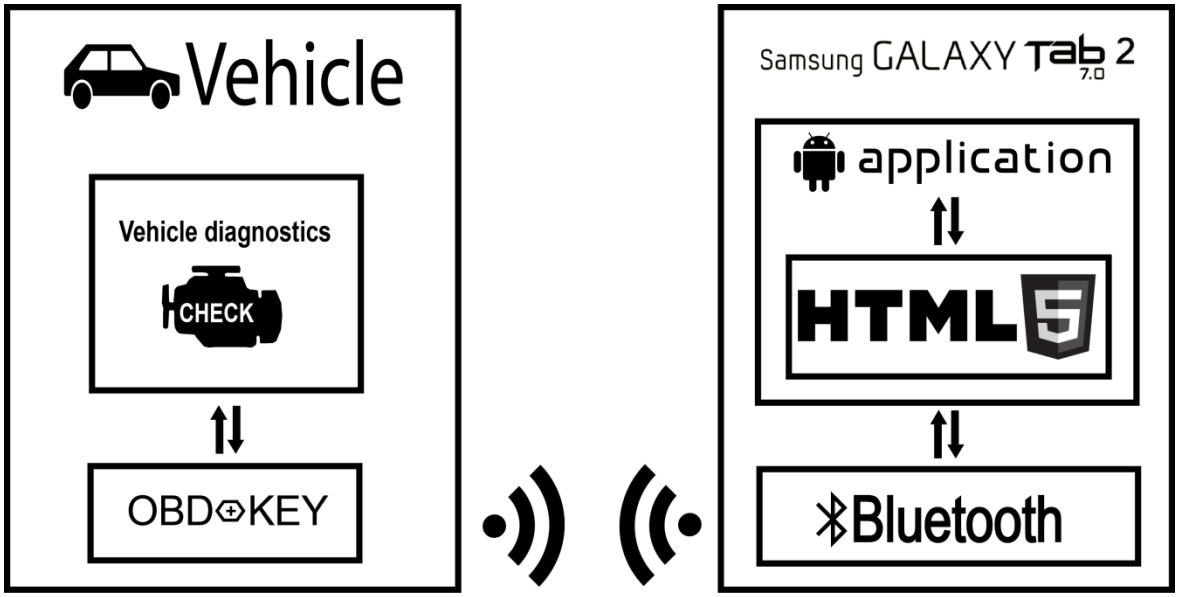
4 SOVELLUSKEHITYS

4.1 Arkkitehtuuri

Sovelluksen arkkitehtuuri koostuu natiivista Android Java –koodista ja HTML5 -koodista. Sovellus alkaa D2IDashboard.java -luokasta, joka luo WebViewin, asettaa sen aktiiviseksi näkymäksi ja lataa siihen html tiedoston.

Seuraavaksi HTML –koodista luodaan Bluetooth –sarjaporttiyhteys OBDKey:hin. HTML:ssä itsessään ei ole mahdollisuuksia kutsua laitteen Bluetooth –rajapintaa, joten käytin Phonegap ohjelmointirajapintaa luodakseni liitännäisen, jolla Bluetoothia voisi käyttää JavaScriptin avulla. Tällä tavalla kaikki toiminnallisuus on näennäisesti HTML:n puolella. Perehdyn tähän tarkemmin myöhemmässä osassa tätä opinnäytetyötä.

Kun yhteys on muodostettu, sovellus alkaa kysellä arvoja Bluetoothin yli. Arvot saatuaan se päivittää ruudun grafiikkaa vastaamaan diagnostiikasta saatuja arvoja.



KUVIO 3. Järjestelmäkaavio

4.2 Android

D2IDashboard.java –luokka perii DroidGap –luokan, joka on osa PhoneGapista saatavia työkaluja. DroidGap –luokka periytyy Androidin Activity –luokasta. Activity on sovelluksen komponentti, joka näyttää sovelluksen käyttöliittymän, jonka kanssa käyttäjä voi tehdä toimintoja. (Open Handset Alliance 2013a).

D2IDashboard.java –luokassa myös luodaan webView –niminen muuttuja, joka on tyyppiä CordovaWebView. CordovaWebView -luokka periytyy Androidin omasta WebView –luokasta. WebView on näkymä, joka kykenee näyttämään nettisivuja. Androidin WebView käyttää WebKit –selainmoottoria, joka on perustana muunmuassa Applen Safari –selaimelle (Open Handset Alliance 2013e). Tästä syystä kehitettäessä tälle alustalle, voidaan käyttää samoja periaatteita kuin Applen Safari –selaimen kanssa. CordovaWebView –luokka lisää tähän toiminnallisuuteen ominaisuudet, joiden ansiosta voidaan kutsua Java –koodilla tehtyjä liitännäisiä HTML –koodista JavaScriptin avulla. WebView –muuttujaan haetaan sovellukselle tarkotetuista resursseista HTML –dokumentti ja piirretään ruutuun. Tämän jälkeen käyttöliittymään liittyvä logiikka toteutetaan täysin HTML –koodin avulla ja Bluetooth –toiminnallisuus toteutetaan liitännäisten avulla.

BluetoothConnect.java –luokka on liitännäinen ja luokassa olevia metodeja voi kutsua JavaScriptistä. Ensimmäisenä JavaScript –koodista kutsutaan checkBluetooth –nimistä metodia. Metodi tarkistaa onko laitteessa Bluetooth radiota ja onko se päällä. Jos radio löytyi ja se on päällä, lähetetään Java liitännäisestä merkkijono ”Ok” takaisin JavaScript –koodiin. Tämän jälkeen JavaScript –koodista käsketään aloittamaan lähellä olevien Bluetooth –laitteiden tiedustelu. Heti tiedustelukäskyn antamisen jälkeen käynnistetään ajastin, joka jatkuvasti tarkastaa Java –liitännäiseltä, onko laitteiden tiedusteluprosessi päättynyt. Kun prosessi on päättynyt, JavaScript –koodi kutsuu Java –liitännäiseltä listaa löytyneistä Bluetooth laitteista. Saatu lista esitetään käyttäjälle, ja käyttäjän valitsema laite (jonka siis tässä vaiheessa tulisi olla OBDKey –laite) viedään parametrina connectionStart metodiin, joka luo yhteyden laitteeseen, aloittaa tietojen kyselyn ja vastaanottamisen.

Datan vastaanottaminen OBDKey –laitteelta aloitetaan heti yhteyden luomisen jälkeen. Java –liitännäisessä luodaan uusi säie, jossa käsitellään käskyjen lähettäminen OBDKeylle ja toinen säie, jossa käsitellään vastaukset käskyihin.

Käskyjen lähetyssäikeessä lähetetään ensiksi käsky ”ATZ” laitteen asetusten alustamiseksi alkuperäisiksi. Sen jälkeen lähetetään käsky ”AT SP 0”, jotta laite asettaisi itse viestintäprotokollan OBDKeyn ja auton diagnostiikan kanssa. Tämän jälkeen aloitetaan silmukka, jossa aktiivisesti kysytään diagnostiikalta eri arvoja. Aina, kun silmukka suoritetaan, pysäytetään säikeen suorittaminen 0,2s ajaksi, kysytään laitteelta auton kierroslukua, pysäytetään säikeen suorittaminen 0,2s ajaksi ja kysytään auton nopeustietoa. Joka viidennellä silmukan suorituskerralla kysytään nopeuden ja kierrosluvun lisäksi yksi vähemmän tärkeä tieto. Vähemmän tärkeitä tietoja ovat moottoriveden lämpötila ja polttonesteen määrä. Nämä arvot eivät muutu niin nopeasti, että niitä täytyisi päivittää jatkuvasti. Vähemmän tärkeä tieto myös muuttuu joka kerta, eli esimerkiksi polttonesteen määrä päivittyy seuraavan kerran vasta, kun kaikki muut arvot on kysytty

Ennen käskyn antamista, tai ehkä pikemminkin käskyn antamisen jälkeen, täytyy odottaa 0,2s, koska tällöin laite ehtii vastaamaan kyselyyn. Vähemmän tärkeiden tietojen kysely harvemmin tehdään siksi, että auton kierroslukumäärä ja nopeustieto päivittyisi mahdollisimman reaaliajassa, eivätkä muut mittaritiedot hidastaisi niiden päivitystä. Käyttäjälle ne ovat näkyvimmät muutokset mittaritiedoissa ajon aikana.

Laitteelta tulevat viestit käsitellään myös omassa säikeessään. Käytännössä silmukan sisällä odotetaan Bluetooth -sarjamuotoista dataa. Tavut muodostetaan viesteiksi. Kun laitteelta tulee mittariarvo, on siinä varsinaisen arvon lisäksi tunnistetieto siitä, mitä tieto on. Tunnistetiedon perusteella viesti muutetaan oikealla kaavalla merkitykselliseksi tiedoksi. Kun viesti on saatu merkitykselliseksi, lähetetään se edelleen JSON –muotoiseen muuttujaan. Auton tiedot saadaan JavaScript koodiin Java –liitännäisen avulla. Tietojen kysely ja vastaanotto jatkuu koko ohjelman suorituksen ajan taustalla.

Muu sovelluskehitykselle olennainen toiminnallisuus on toteutettu HTML –koodilla. Tässä vaiheessa on hyvä muistuttaa, että tällä hetkellä Bluetooth –toiminnallisuutta on mahdoton toteuttaa HTML:n avulla ilman Android Java –liitännäisiä, eikä HTML5 standardissa ole minkäänlaista mainintaa Bluetoothista, joten tälläistä toiminnallisuutta tuskin lähiaikoina on tulossa.

4.4 HTML5

HTML –koodi koostuu index.html tiedostosta, kahdesta CSS –tyylitiedostosta ja useasta JavaScript tiedostosta. Lisäksi mukana on Phonegapiin liittyvä JavaScript –tiedosto. Mittariston käyttöliittymään ja ulkoasuun liittyvä koodi on toteutettu index.html tiedostossa, joka toimii aloituskohtana sovelluksen HTML –osioon.

JavaScriptin ollessa hyvin kyvykäs ohjelmointikieli se antaa vapaat kädet käytettävälle ohjelmointitavalle. Päädyin käyttämään toiminnallisuuksia siten, että jokaiselle ”luokalle” luotiin oma JavaScript tiedosto, johon edelleen luotiin tiedostonimellä olevan muuttuja, johon lisättiin ”luokan” muuttujat ja funktiot. Esimerkiksi Tervehdys.js näyttäisi kuvion 4 mukaiselta ja kuviossa näkyvää funkiota kutsuttaisiin linkityksen jälkeen seuraavasti; Tervehdys.tervehdi(). Tässä dokumentissa ei kuitenkaan ole tarkoitus keskittyä ohjelmointityyleihin tai oppeihin tämän enempää.

```
1  var Tervehdys = {
2      teksti: "Tervehdys kaikille!",
3      tervehdi: function(){
4          alert(teksti);
5      }
6  }
```

KUVIO 4. Luokkamainen muuttuja JavaScriptissä

Sovelluksen käynnistyessä, ensimmäisenä ladataan käyttöliittymään ja ulkoasuun liittyvät elementit, sekä linkitetään JavaScript tiedostot niihin. Sen jälkeen näytetään latausanimaatio kutsumalla Animate.showLoading() funkiota, joka asettaa pyörivän .GIF kuvan ruudulle, jotta käyttäjä tietäisi, että taustalla tehdään jotain. Tämän jälkeen odotetaan Phonegapilta ilmoitusta siitä, että sovelluksen

lataus on valmis ja liitännäiset ovat valmiita käytettäväksi. Tällöin järjestelmä kutsuu onDeviceReady funktiota. Laitteen ollessa valmis, kutsutaan NativePlugin.startConnect() –funkiota, joka aloittaa pluginien avulla Bluetooth –yhteyden muodostamisen OBDKey:n ja puhelimen välillä.

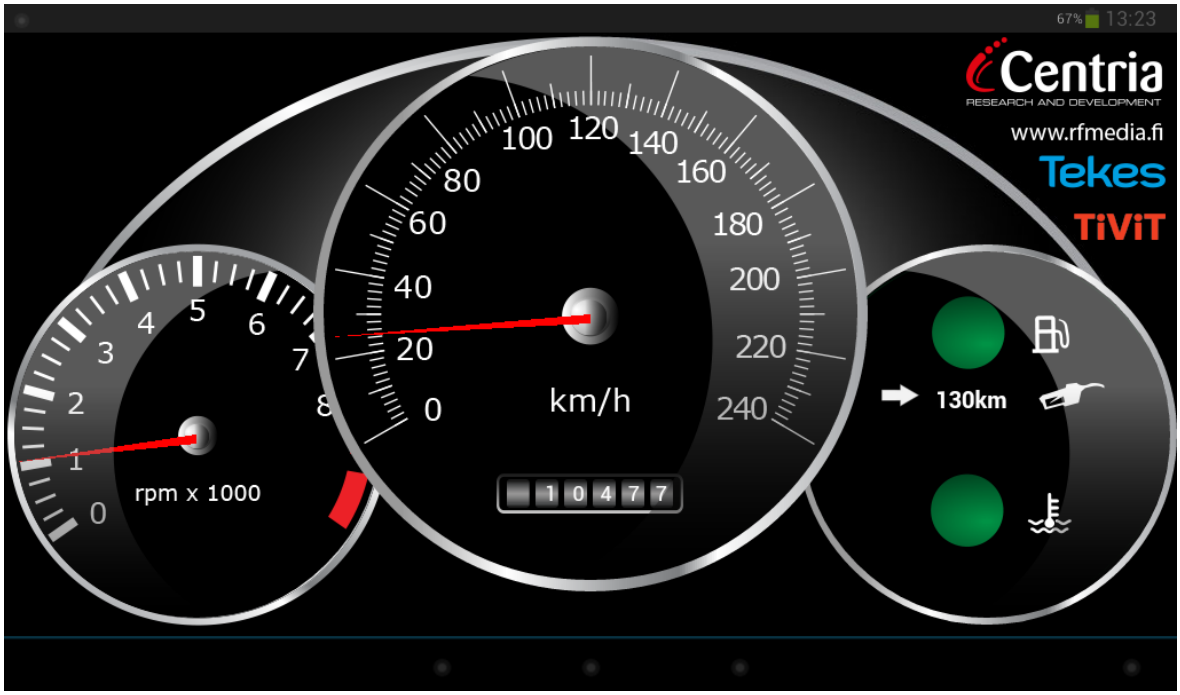
Yhteyden muodostamisen aikana käyttäjän tulee valita OBDKey –laite listasta, jossa on lähialueen kaikki Bluetooth laitteet. Koska HTML:ssä ei ole mahdollisuutta luoda dialogia usealla painonapilla, täytyi toiminnallisuus toteuttaa div –elementin sisälle ja käyttää CSS tyylejä, jotta luotu grafiikkaelementti olisi vastaavan näköinen, kuin esimerkiksi alert –dialogi. Dialogi luodaan dynaamisesti saatujen Bluetooth laitteiden perusteella. Kun dialogissa olevaa Bluetooth –laitteen nimeä painetaan, aloitetaan yhteydenotto siihen pluginin kautta.

Kun yhteys OBDKey:n kanssa on luotu, ryhdytään näytön grafiikkaa päivittämään vastaamaan auton tietoja. Grafiikka päivitetään JavaScript interval –ajastimen avulla 15 kertaa sekunnissa. Samalla ajastimella haetaan myös auton tiedot liitännäisen avulla. Näytöllä näkyvät viisarit ovat HTML div –elementtejä. Div –elementin kääntymiskulmaa voidaan muokata CSS:n ja JavaScriptin avulla reaaliajassa ajastimen avulla, esimerkiksi document.getElementById('viisari').style.WebkitTransform = 'rotate(' + KULMA + 'deg)';. Tämä muuttaa viisari ID:llä varustetun HTML elementin tyylitietoa siten, että se kääntyy annettuun asteeseen. Auton diagnostiikasta saatu arvo muutetaan asteiksi. Tämän lisäksi saadut arvot lineaarisesti interpoloidaan, jotta visuaalinen ulosanti olisi kuljettajalle miellyttävä.

Jos auto on pysähdyksissä, kuljettajalle näytetään moottoriveden lämpötilasta ja polttonesteen määrästä tarkat mittarilukemat. Ajon aikana kuljettajan ei tarvitse tietää tarkkaa tietoa, joten sovellus näyttää kuljettajalle väri-indikaattorit arvon mukaisesti. Jos polttonesteen määrä on pienempi kuin 10% tankista, näytetään punainen indikaattori. Jos polttonesteen määrä on pienempi kuin 30%, näytetään keltainen indikaattori, muutoin näytetään vihreä indikaattori. Jos polttonesteen määrä on alle 5%, näytetään indikaattorin lisäksi varoitusmerkki. Moottoriveden lämpötilasta näytetään punainen indikaattori, jos lämpötila nousee päälle 115 Celsius asteen, keltainen jos päälle 100 Celsius asteen ja muutoin näytetään

vihreä indikaattori. Kun indikaattorin kuvaa painetaan, saa käyttäjä näkyviin tarkemmat tiedot, vaikka auto olisi liikkeessä. Kuvioissa 5 ja 6 kuvakaappaukset sovelluksesta mainituista tilanteista.

KUVIO 5. Mittaristosovellus, auto pysähtyneenä



KUVIO 6. Mittaristosovellus, auto liikkeessä

4.5 Phonegap

Liitännäisten kehittäminen Phonegap HTML –sovellukseen koostuu JavaScriptistä ja valitun alustan natiivilla ohjelmointikielellä tehdystä koodista. Android –alustalle kehitetään liitännäiset täten Java –kielellä (Phonegap.com 2013b). Phonegap on kokoajan kehittyvä työkalu ja tämän opinnäytetyön sovellukset on kehitetty Phonegapin versiolla 2.8.0. Phonegapista löytyy jo versio 3.0.0, eivätkä tässä opinnäytetyössä käytetyt menetelmät välttämättä enää toimi uusilla versioilla.

```
cordova.exec(
    function(success) {
        alert(success)
    },
    function(error) {},
    "Esimerkki",
    "tervehdi",
    ["Terveisiä", "JavaScriptistä"]
);
```

KUVIO 7. Phonegap –liitännäisen JavaScript esimerkki

Liitännäisiä kutsutaan JavaScript koodista Phonegapin työkaluissa mukana tulevalla cordova.exec -funktiolla. Funktio ottaa viisi argumenttia. Kuvion 7 mukaisesti, ensimmäinen parametri on funktio, joka suoritetaan kun cordova.exec on suoriutunut ilman virheitä. Funktioon voidaan lisätä argumentti (”success”), jonka arvo saadaan Java –koodista. Toinen cordova.exec funktion parametri on funktio, joka suoritetaan virheen sattuessa. Kolmas parametri on ”palvelu”, joka on käytännössä Java –luokka. Neljäs parametri on ”toiminto”, joka halutaan palvelussa suorittaa. Viides parametri on Java –koodiin vietävät argumentit.

```
<plugins>
    <plugin name="Esimerkki" value="fi.nimiavaruus.sovellus.Esimerkki"/>
</plugins>
```

KUVIO 8. Liitännäisten listaus

Android Java –kehityksessä täytyy sovelluksen työtilasta löytyvään res/xml/config.xml –tiedostoon määrittää liitännäiset kuvion 8 mukaisesti. Name –

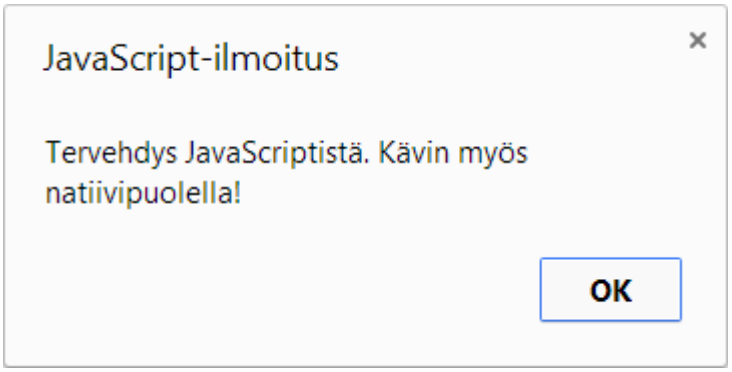
argumentin tulee olla Java –luokan nimi ja value –argumentin tulee olla luokan koko nimi, johon sisältyy sovelluksen nimiavaruus.

```
public class Esimerkki extends CordovaPlugin {

    @Override
    public boolean execute(String action, JSONArray args, CallbackContext callbackContext) throws JSONException {
        if (action.equals("tervehdi")) {
            String viesti = args.getString(0) + " " + args.getString(1) + ". Kävin myös natiivipuolella!";
            callbackContext.success(viesti);
            return true;
        }
        return false;
    }
}
```

KUVIO 9. Phonegap –liitännäisen Java esimerkki

Kun varsinainen liitännäinen luodaan tulee sen periä CordovaPlugin –luokka. Luodussa luokassa execute –metodi korvataan ja siinä käsitellään JavaScript –koodista saadut argumentit, käsitellään ne ja lähetetään viesti onnistumisesta (callbackContext.success) tai epäonnistumisesta. Kuvioiden 7, 8 ja 9 mukainen liitännäinen luo kuvion 10 mukaisen alert –dialogin.



KUVIO 10. Esimerkin tuotos

Tämän sovelluksen liitännäiset on luotu näitä periaatteita käyttäen. Vaikka esimerkki oli hyvin yksinkertainen, samalla tekniikalla toteutin Bluetooth –toiminnalisuuden ilman kompromisseja HTML –sovellukseen.

4.6 Testaus

Sovellusta testattiin Özen Electronic:n OBD –simulaattorin lisäksi Volvon, Volkswagenin ja Jeepin valmistamilla autoilla. Taulukossa 1 on esillä tarkemmat tiedot autoista ja testien tulokset.

TAULUKKO 2. Testatut autot ja tulokset

Merkki	**Malli**	**Vm.**	**Isku tilavuus**	**Polttoneste**	**Nopeus mittari**	**Kierroslukumittari**	**Moottoriveden lämpötila**	**Polttonesteen määrä**
Volvo	V70	2007	2,0l	Diesel	**x**	**x**	**x**	**x**
Jeep	Grand Cherokee	1998	5,9l	Bensiini	**x**	**x**	**x**	**x**
Volkswagen	Transporter	2010	2,5l	Diesel				
Volkswagen	Passat	2009	2,0l	Diesel	**x**	**x**	**x**	

Kuten taulukosta 1 käy ilmi, toiminnassa oli ongelmia Volkswagenin autoilla. OBD2 –protokollan implementaatiossa on selvästi valmistajakohtaisia eroja, joka selviää jo neljällä eri autolla tehdyllä testillä. Tämän opinnäytetyön puitteissa en ole perehtynyt valmistajakohtaisiin eroavaisuuksiin.

5 JATKOKEHITYS

5.1 Tietoa taustajärjestelmistä

Sovellusta voisi kehittää käyttämään hyväksi jo olemassa olevia tietolähteitä. Suomalainen Infotripla Oy ylläpitää Oulun kaupungin alueella liikennetiedotetietokantaa, joka on eurooppalaisen Datex2 –standardin mukainen. Datex2 –standardi käsittää erittäin laajan kirjon erilaisia liikennetilanteita onnettomuuksista tietöihin ja ajokaistalla havaittuhin eläimiin. Tilanteista annetaan myös yleensä koordinaatit, joten tarvittavista tilanteista voitaisiin antaa kuljettajalle varoitus hänen ollessaan tarpeeksi lähellä. Sovelluksen navigaattori voisi huomioida tilanteen ja laskea kuljettajalle uuden vaihtoehtoisen reitin tilanteen niin vaatiessa.

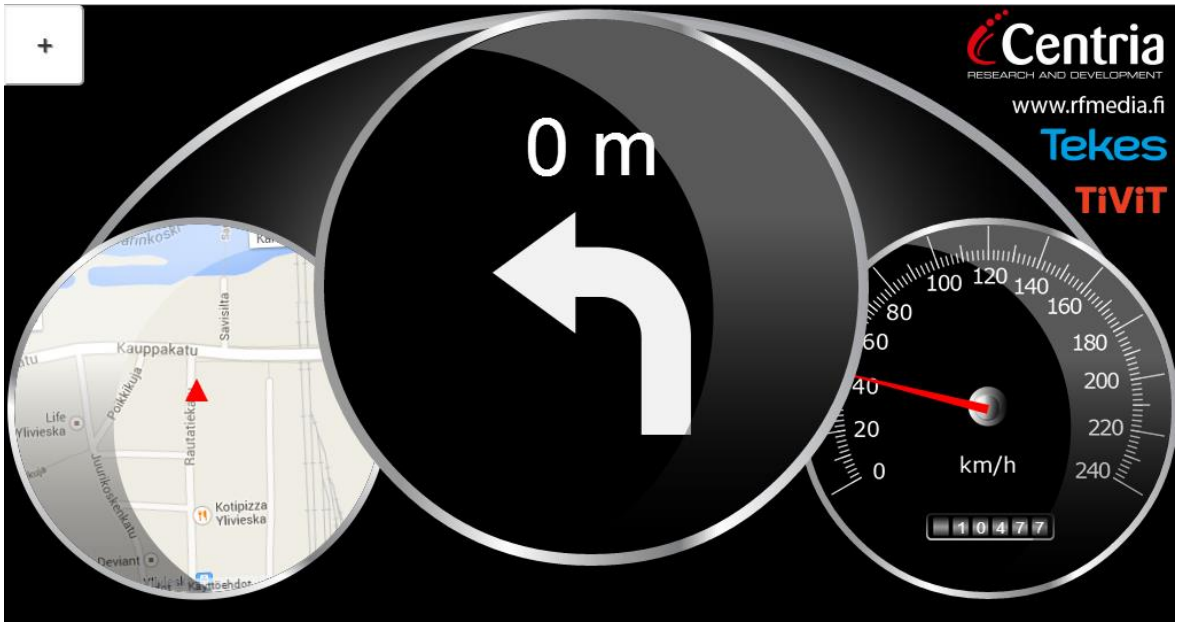
Datex2 –tietolähteiden lisäksi tietolähteenä voi toimia julkiset sääpalvelut, joiden avulla käyttäjää varotetaan reitillä olevasta heikosta kelistä.

Tässä luvussa pohdin sovelluksen jatkokehittelyn mahdollisuuksia. Näitä toiminnallisuuksia ei kuitenkaan ole vaatimusmäärittelyssä listattu, eikä sovellukseen toteutettu. Sovelluksen tämänhetkinen toiminnallisuus kattaa vain auton mittariston perusominaisuudet.

5.2 Adaptiivinen käyttöliittymä

Sovelluksen käyttöliittymän kannalta olennainen vaatimusmääritelmä oli adaptiivisuus. Adaptiivinen käyttöliittymä mukautuu käyttäjän tarpeiden ja tilanteen muuttuessa tarjoamaan käyttäjälle sillä hetkellä tärkeimmät tiedot ja toiminnallisuudet. Adaptiivisuus tuo oman mielenkiintoisen lisänsä jatkokehitysmahdollisuuksia ajatellen.

Tämän opinnäytetyön ohessa yhdessä Johanna Hartikan kanssa kehitimme sovelluksen, jonka tehtävänä on demonstroida adaptiivisen käyttöliittymän mahdollisuuksia auton mittaristossa. Pääpiirteittäin käyttöliittymä tässä ja demo –sovelluksessa on sama, Hartikan kehittämä.



KUVIO 11. Navigaattori mittaristossa

Kuviossa 11 näkyy kuvakaappaus demo –sovelluksesta. Kuvitteellisessa tilanteessa kuski on asettanut navigaatiokohteen. Kun reitillä tulee käännös, mittariston käyttöliittymä mukautuisi tilanteeseen siten, että esille tulee kartta ja etäisyys käännökseen. Muutoin normaalissa ajotilanteessa navigaattorin näkyvyys mittaristossa olisi hyvin pieni, eikä se veisi kuljettajan huomiota.

KUVIO 12. Tilannevaroitus mittaristossa

Kuviossa 12 mittaristo saa taustajärjestelmästä tiedon liian lyhyestä turvavälistä edessäolevaan ajoneuvoon. Mittaristo mukautuu tilanteeseen näyttämällä varoitusmerkin. Tilanteen taustajärjestelmä voi olla auton oma sensori, tai jollain muulla, autosta riippumattomalla tekniikalla toteutettu järjestelmä. Taustajärjestelmän avulla liikenteen turvallisuutta voidaan parantaa myös muilla ilmoituksilla, esimerkiksi nopeusrajoituksen muutoksen ilmoittamisella, nopeusrajoituksen ylittämisen ilmoituksella, varoittamalla havaituista eläimistä alueella tai varoittamalla liukkaasta tiestä.

6 JOHTOPÄÄTÖKSET

Opinnäytetyön tuloksena syntyi auton mittaristosovellus Android –mobiilikäyttöjärjestelmälle, joka käyttää OBDKey –lukijaa lukeakseen auton mittaristoarvoja Bluetoothin yli ja näyttää ne käyttäjälle. Sovellus on toteutettu niiltä osin HTML:llä, JavaScriptillä ja CSS:llä, kun se on ollut mahdollista. Phonegap –rajapintaa käytettiin laajentamaan web- sovelluksen toiminnallisuutta, jotta Bluetooth –toiminnallisuus saatiin toteutettua.

6.1 HTML mobiilisovelluskehityksessä

Web –sovellus ja HTML tuo kehitystyölle omat rajoitteensa, esimerkiksi mainitun Bluetooth –toiminnallisuuden puuttuessa. Tämän lisäksi on syytä huomioida, että web –sovellus ajetaan selainmoottorissa ja sovellusten koodi tulkataan suorituksen aikana, verrattaen siihen, että koodi ensiksi käännettäisiin ja suoritettaisiin myöhemmin. Tästä syystä web –sovellusten toiminnallisuus voi olla hidasta, etenkin graafisesti vaativissa sovelluksissa. Tässä sovelluksessa ruutua päivitetään jatkuvasti, 15 kertaa sekunnissa. Elokuvien ruudunpäivitys on 24 kertaa sekunnissa. 15 oli tarpeeksi korkea lukema, ettei se vaikuttanut negatiivisesti käyttökokemukseen. Galaxy Tab 2:llä 15krt/s oli korkein mahdollinen ruudunpäivitys, joka toimi jättämättä väliin jatkuvasti ruudun piirtoja. Jos nostin lukeman esimerkiksi 30 krt/s, ilmoitti Androidin kehitystyökalut jatkuvasti väliin jätetyistä ruudun piirroista ja ruudunpäivitys oli epätasaista ja nykivää.

Mobiililaitteiden järjestelmäpiirit kuitenkin kehittyvät kokoajan huimaa vauhtia, ja tulevaisuudessa uskon myös web –sovellusten kykenevän graafisesti hyvinkin vaativiin suorituksiin.

Syy, miksi sovelluksia halutaan kehittää web –sovelluksiksi on se, että lähes samalla koodilla ohjelmisto voidaan julkaista hyvin monelle eri käyttöjärjestelmälle. Jos saman sovelluksen haluaisi tehdä usealle eri järjestelmälle sen omilla työkaluilla, tarkoittaa se lähes aina uuden ohjelmointikielen, ohjelmointirajapintojen ja järjestelmään liittyvien työkalujen opettelua. HTML:n avulla yksinkertaiset

sovellukset voidaan toteuttaa täysin samalla koodilla. Toiminnallisuus jota ei HTML:stä löydy, joudutaan kuitenkin toteuttamaan liitännäisten avulla.

6.2 Oman oppimisen arviointi

Työskennellessäni opinnäytetyön parissa oma osaamiseni kehittyi huomattavasti. Tietoni HTML:n, CSS:n ja JavaScriptin osalta syventyi huomattavasti ja onnistuin löytämään työkaluille uusia käyttötarkoituksia. Myös Android ja Java osaamiseni kehittyi työskennellessäni Androidin Bluetooth –kirjastojen parissa. Samalla opin myös käytännön tasolla, kuinka Bluetooth toimii ja kuinka sitä voidaan hyödyntää sovelluskehityksessä. Web- teknologioiden liittäminen Android –sovelluskehitykseen oli alunperin minulle vieras aihe, josta olin vain lukenut eri artikkeleista.

Auton tietojen lukeminen diagnostiikkaväylästä oli myös uusi haaste. Oppimisen myötä tuli tutuksi myös teknisten dokumenttien lukeminen ja edelleen niiden käytännön hyödyntäminen ohjelmointityössä.

Mediateknkiikan koulutusohjelman ansiosta minulla oli selkeä näkemys siitä, mitä vaiheita työni vaatii ja kuinka lähteä selvittämään vaiheisiin liittyvät onglemat. Kaikkien näiden uusien ja tuttujen teknologioiden yhteentuominen oli hyvä mahdollisuus kokea, mitä reaalimaailman sovelluskehitys parhaimmillaan on.

LÄHTEET

Bluetooth.com. 2013a. A Look at the Basics of Bluetooth Wireless Technology . Www –dokumentti. Saatavissa: http://www.bluetooth.com/Pages/Basics.aspx. Luettu 16.9.2013.

Bluetooth.com. 2013b. Bluetooth Fast Facts. Www –dokumentti. Saatavissa: http://www.bluetooth.com/Pages/Fast-Facts.aspx. Luettu 16.9.2013.

California Environmental Protection Agency. 2009. On-Board Diagnostic II (OBD II) Systems - Fact Sheet / FAQs. Www –dokumentti. Saatavissa: http://www.arb.ca.gov/msprog/obdprog/obdfaq.htm. Luettu 16.9.2013.

Darcey, L., Conder, S. 2012. Android Wireless Application Development. Crawfordsville: RR Donnelley.

Elekma. 2013. Mikä on OBD? Www –dokumentti. Saatavissa: http://www.elekma.com/mika_on_obd. Luettu 16.9.2013.

Elm Electronics. ELM327 datalehti. Kanada. Saatavissa: http://elmelectronics.com/DSheets/ELM327DS.pdf. Luettu 14.11.2013.

Gendan Automotive Products. 2014. What is EOBD, EOBD2 and OBDII? Www –dokumentti. Saatavissa: http://www.gendan.co.uk/article_5.html. Luettu 12.1.2014.

Gehrmann, C., Persson, J. & Smeets, B. 2004. Bluetooth Security. Lontoo: Artech House.

Jordan, L., Greyling, P. 2011. Practical Android Projects. Apress.

KBM Systems Ltd. 2013. What Is OBD?. Www –dokumentti. Saatavissa: http://www.obdkey.com/obd.asp. Luettu 16.9.2013.

Korpela, J. 2011. HTML5 Uudet Ominaisuudet. Porvoo: Bookwell Oy.

Lunden, I. 2013. Android, Led By Samsung, Continues To Storm The Smartphone Market, Pushing A Global 70% Market Share. Www –dokumentti. Saatavissa: http://techcrunch.com/2013/07/01/android-led-by-samsung-continues-to-storm-the-smartphone-market-pushing-a-global-70-market-share/. Luettu 16.9.2013.

OBD.fi. 2007. EOBD, OBD2 ja valmistajakohtaiset protokollat. Www –dokumentti. Saatavissa: http://www.obd.fi/index.php?option=com_content&task=view&id=32&Itemid=2. Luettu 16.9.2013.

Open Handset Alliance. 2013a. Android Frequently Asked Questions, 2007. Www –dokumentti. Saatavissa: http://www.openhandsetalliance.com/android_faq.html. Luettu 16.9.2013.

Open Handset Alliance. 2013b. Android Application Fundamentals. Www –dokumentti. Saatavissa: http://developer.android.com/guide/components/fundamentals.html. Luettu 16.9.2013.

Open Handset Alliance. 2013c. Get the Android SDK. Www –dokumentti. Saatavissa: http://developer.android.com/sdk/index.html. Luettu 16.9.2013.

Phonegap.com. 2013a. Phonegap Explained Visually. 2013. Www –dokumentti. Saatavissa: http://phonegap.com/2012/05/02/phonegap-explained-visually. Luettu 16.9.2013.

Phonegap.com. 2013b. Phonegap Plugin Development Guide. Saatavissa: http://docs.phonegap.com/en/3.0.0/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide. Luettu 16.9.2013.

SAE 1979. Diagnostic Test Modes. 2007. Yhdysvallat: Society of Automotive Engineers SAE.

Samsung. 2012. Samsung Galaxy Tab 2 7.0 Spec Sheets.

Özen Elektronik. mOByDic oe91c1610 -datalehti. Luettavissa: http://www.ozenelektronik.com/downs/pdf/oe91c1610.pdf. Luettu 14.11.2013.