

Ajoneuvon OBD- ja paikkatietodatan kerääminen ja visualisointi

Miikkael Puikkonen

Opinnäytetyö

Elokuu 2020

Tekniikan ja liikenteen ala

Insinööri (AMK), Tietotekniikan koulutusohjelma

Tietoverkkotekniikka

Tekijä(t) Puikkonen, Miikkael	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Elokuu 2020
	Sivumäärä 65	Julkaisun kieli Suomi
		Verkojulkaisulupa myönnetty: x
Työn nimi Ajoneuvon OBD- ja paikkatietodatan kerääminen ja visualisointi		
Tutkinto-ohjelma Tietoverkkotekniikka		
Työn ohjaaja(t) Rantonen Mika, Kotikoski Sampo		
Toimeksiantaja(t) JAMK, IT-Instituutti		
<p>Tiivistelmä</p> <p>Markkinoilta löytyy paljon laitteita auton ajotietokoneen antaman datan lukemista varten, mutta nämä ammattikäytössä käytettävät laitteet ovat kuitenkin usein hyvin kalliita ja niistä puuttuu yleensä paikannusominaisuus.</p> <p>Tämän pohjalta alettiin suunnittelemaan ja rakentamaan laitetta, joka olisi edullinen sekä pienikokoinen ja jolla olisi mahdollista seurata ajoneuvon tilaa reaaliajassa keräämällä auton OBDII-järjestelmästä ja GPS-paikantimesta saatua dataa. Saatua dataa kerättiin talteen tietokantaan ja muokattiin selkeään ja helppolukuiseen muotoon.</p> <p>Laitteen pohjaksi valittiin Raspberry Pi-pienoistietokone, joka pienen kokonsa ja alhaisen hintansa puolesta sopi käyttötarkoitukseen hyvin. Siihen sitten yhdistettiin datan keräämistä varten OBDII-adapteri ja GPS-paikannin, joiden keräämä data lähetettiin JAMKin virtuaalipalvelimella sijaitsevaan tietokantaan. Tietokantaan kerättyä dataa pystyttiin sitten visualisoimaan Grafana-ohjelman avulla siisteiksi graafeiksi ja karttanäkymiksi.</p> <p>Valmistunut laite toimii toivotusti ja lukee autosta OBDII- ja paikkadataa ja lähettää ne sen jälkeen palvelimelle tietokantaan säilöttäväksi. Kerättyä dataa oli mahdollista tarkastella Grafanan käyttöliittymän kautta halutulta aikaväliltä.</p> <p>Laitteessa oli toimivuuden ja helppokäyttöisyyden kannalta asioita, joita voitaisiin parantaa, jos laitteen kehitystä jatkettaisiin pidemmälle. Myös tietoturvan osalta löytyy asioita, joita olisi voitu tehdä toisin. Kokonaisuutena laite kuitenkin teki tehtävänsä halutulla tavalla ja oli siinä mielessä onnistunut kokonaisuus.</p>		
Avainsanat (asiasanat) OBD, GPS, InfluxDB, Grafana, Raspberry Pi, IoT		
Muut tiedot (Salassa pidettävät liitteet)		

Author(s) Puikkonen, Miikkael	Type of publication Bachelor's thesis	Date August 2020
		Language of publication: Finnish
	Number of pages 65	Permission for web publication: x
Title of publication Collection and visualization of vehicle OBD and location data		
Degree programme Data network Technology		
Supervisor(s) Rantonen Mika, Kotikoski Sampo		
Assigned by JAMK, IT-Institute		
<p>Abstract</p> <p>There are many devices on the market for reading the data provided by the car's on-board computer, but these devices used by professionals are often very expensive and lack positioning features.</p> <p>On this basis, work began on designing and constructing a device that would be inexpensive and compact and it would be able to monitor the statistics of the vehicle in real time by collecting data from the car's OBDII system and GPS tracker. The resulting data was then to be collected into a database and translated into a clear and easy-to-read format.</p> <p>The Raspberry Pi minicomputer was chosen as the base of the device, which, due to its small size and low price, was well suited for the purpose. It was then connected to an OBDII adapter and a GPS tracker for data collection, and the collected data was sent to a database located on JAMK's virtual server. The collected data could then be visualized using Grafana into neat graphs and map views.</p> <p>The finished device works as desired and reads OBDII and location data from the car and then sends them to the server to be stored in the database. The collected data can then be viewed via the Grafana user interface for the desired timeframe.</p> <p>In terms of functionality and ease of use, the device had some things that could be improved on in the future, if the development of the device were to be continued. There are also some things about the device's security that could have been improved on. As a whole, however, the device serves its intended purpose and was a success overall.</p>		
Keywords/tags (subjects) OBD, GPS, InfluxDB, Grafana, Raspberry Pi, IoT		
Miscellaneous (Confidential information)		

Sisältö

1	Johdanto	6
1.1	Työn tavoitteet	6
1.2	Toimeksiantajan esittely.....	6
1.3	Tutkimusmenetelmät	6
2	Teoria.....	7
2.1	Raspberry Pi.....	7
2.2	Linux	10
2.2.1	Linuxin historia	10
2.2.2	Raspberry Pi OS	10
2.2.3	Ubuntu	11
2.3	OBD-järjestelmä	11
2.3.1	OBD järjestelmän historiaa.....	11
2.3.2	OBDII	12
2.3.3	PID-koodit	13
2.3.4	OBDII vikakoodit	14
2.4	GPS.....	16
2.4.1	GPS:n historiaa	16
2.4.2	GPS-satelliitit	16
2.4.3	GPS-järjestelmä	16
3	Käytetyt ohjelmistot	17
3.1	InfluxDB	17
3.2	Grafana	18
3.3	Nginx.....	19
3.4	Python-OBD	20
3.5	gpsd	21
4	Työn suunnittelu	21
5	Toteutus.....	25
5.1	Raspberry Pi:n käyttöönotto	25
5.2	Raspberry Pi:n ohjelmien asennus ja konfigurointi	27

	2
5.3	Ubuntun ohjelmien asennus ja konfigurointi.....29
5.3.1	Vaadittavat alkuvalmistelut ja ohjelmien asennus.....29
5.3.2	Nginx konfigurointi32
5.3.3	InfluxDB konfigurointi.....35
5.3.4	Grafanan konfigurointi37
5.3.5	Trackmap liitännäisen lisääminen Grafanaan40
5.4	GPS-datan lähettäminen tietokantaan.....42
5.5	Bluetooth yhteyden muodostaminen OBDII-lukijaan45
5.6	OBDII-datan lukeminen ja lähettäminen tietokantaan.....47
5.7	Järjestelmän toiminnan testaus48
6	Pohdinta.....49
6.1	Kohdatut ongelmat ja haasteet.....50
6.2	Mahdollisia parannus ehdotuksia51
Lähteet52
Liitteet54
Liite 1.	gpsd-influx-python.py54
Liite 2.	obd-influx.py62
 Kuviot	
Kuvio 1	OBDII-liitin.....13
Kuvio 2	InfluxDB komentolinja18
Kuvio 3	Grafanan käyttöliittymä.....19
Kuvio 4	Python-OBd debug-tila20
Kuvio 5	Raspberry Pi Model 3 B+22
Kuvio 6	OBDII Bluetooth-adapteri23
Kuvio 7	GPS-paikannin24
Kuvio 8	Suunniteltu loppukokoonpano25
Kuvio 9	Welcome to Raspberry Pi26
Kuvio 10	Interface-asetukset27

Kuvio 11 gpsd asetukset	28
Kuvio 12 Windows 10 VPN asetukset	30
Kuvio 13 Nginx konfiguraatio	33
Kuvio 14 nginx konfiguraation testaus	34
Kuvio 15 Grafana ulkoverkosta	35
Kuvio 16 InfluxDB terminaali	35
Kuvio 17 SHOW USERS	36
Kuvio 18 InfluxDB konfiguraatio	37
Kuvio 19 Uuden datalähteen lisääminen	38
Kuvio 20 InfluxDB datalähde	38
Kuvio 21 gpsd datalähteen asetukset	39
Kuvio 22 Moottorin kierrosluku paneeli	40
Kuvio 23 TrackMap-paneeli	41
Kuvio 24 Valmiit Grafana paneelit	41
Kuvio 25 gpsd-influx-python.py konfiguraatio	42
Kuvio 26 GPS-datan kysely tietokannasta	43
Kuvio 27 gpsd-python.service konfiguraatio	44
Kuvio 28 OBDII-portin sijainti	45
Kuvio 29 Bluetooth manager	46
Kuvio 30 Bluetooth-laitteen parittaminen	46
Kuvio 31 Serial port connected to /dev/refcomm0	47
Kuvio 32 Raspberry Pi:n käyttämä virtalähde	48
Kuvio 33 Testiajon tuloksia	49

Käytetyt termit ja lyhenteet

CAN	Controller Area Network
Debian	Linux-pohjainen käyttöjärjestelmä
DTC	Diagnostic Trouble Code

ELM327	ELM Electronics:in valmistama mikrokontrolleri
GB	Gigatavu, 1 000 000 000 tavua
GHz	Gigahertsi, 1 000 000 000 hertsiä
GNU	GNU's Not Unix
GPIO	General-purpose input/output
GPS	Global Positioning System
HDMI	High Definition Multimedia Interface
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
JAMK	Jyväskylän Ammattikorkeakoulu
LTS	Long Term Support
MB	Megatavu, 1 000 000 tavua
MHz	Megahertsi, 1 000 000 hertsiä
NAVSTAR	Navigation Satellite Timing & Ranging
OBD	On-board Diagnostics
PID	Parameter ID

Python	Ohjelmointikieli
RAM	Random Access Memory
SD	Secure Digital
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
USB	Universal Serial Bus
V	Voltti
VNC	Virtual Network Computing
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

1 Johdanto

1.1 Työn tavoitteet

Autot ja niiden järjestelmät kehittyvät jatkuvasti monipuolisemmiksi ja monimutkaisemmiksi. Tämän takia auton ongelmien selvittäminenkin ilman oikeanlaisia työkaluja vaikeutuu. Onneksi kaikissa autoissa on nykyään standardoitu OBD-järjestelmä, joka helpottaa auton vikatilojen selvittämistä auton antamien vikakoodien avulla. Ammattikäyttöön tarkoitetut työkalut ovat kuitenkin usein liian kalliita normaalin autoilijan tarvitsemaan käyttöön. Nykyään markkinoilla on kuitenkin laitteita, jotka ovat edullisempia ja ne on tarkoitettu perusautoilijan käyttöön.

Tästä syntyi idea laitteelle, joka olisi edullinen sekä pienikokoinen. Tavoitteena oli myös, että tähän laitteeseen pystyisi liittämään ominaisuuksia, joita ei yleensä näistä edullisemmista laitteista löydy, kuten GPS-paikannus. Lisäksi tavoitteena oli, että laitteesta saatava data voitaisiin lähettää reaaliaikaisena verkon yli palvelimelle, jossa siitä voidaan tehdä selkeää ja helppolukuista.

1.2 Toimeksiantajan esittely

Työn toimeksiantajana toimi Jyväskylän Ammattikorkeakoulu (JAMK). Jyväskylän ammattikorkeakoulu on vetovoimainen ja kansainvälinen korkeakoulu, jossa opiskelee noin 8500 opiskelijaa yli 70 maasta. JAMK:in tutkintotarjontaan kuuluu yli 30 tutkintoa 8 eri alalla. Vaihtoehtoina ovat muun muassa agrologin, insinöörin, musiikkipedagogin, restonomin ja tradenomin koulutukset. JAMK:lla on useita toimipisteitä ympäri Jyväskylää, sekä yksi toimipiste Saarijärven Tarvaalassa.

1.3 Tutkimusmenetelmät

Työtä suunnitellessa pohdittiin, onko vastaavanlaista laitetta jo ennestään saatavilla ja olisiko mahdollista hyödyntää niissä käytettyjä ideoita ja teknologioita. Heräsi myös kysymys, millainen laitteisto ja konfiguraatio vaadittaisiin suunnitellun laitteen

luomiseen ja mitkä saatavilla olevista ratkaisuista sopisivat parhaiten juuri tähän käyttötarkoitukseen. Tutkimuksessa mietittiin myös, että olisiko näitä jo olemassa olevia ideoita ja teknologioita mahdollista hyödyntää suoraan laitteen suunnittelussa vai olisiko niitä mukailtava omiin käyttötarpeisiin sopivaksi. Mikäli muokkaus olisi tarpeen, mistä lähteistä olisi järkevintä lähteä etsimään ratkaisua kohdattaviin ongelmiin ja haasteisiin.

Koska laitteen sijoituspaikan on tarkoitus olla auton ohjaamon läheisyydessä, työn suunnittelussa oli oleellista, että laitteen tuli olla kompakti ja myös edullinen. Tästä syystä käytettävät komponentit eivät saaneet olla liian kalliita, eivätkä painavia tai suurikokoisia. Tästä syystä työn pohjaksi päädyttiin valitsemaan Raspberry Pi.

Työtä suunnitellessa hyödynnettiin jo aiemmin kertynyttä osaamista ja kokemusta eri käyttöjärjestelmistä, verkkoteknologioista ja koodaamisesta. Työn tietolähteinä toimivat työssä käytettävien tuotteiden valmistajien tarjoamat ohjeet, sekä useiden eri harrastajayhteisöjen luomat projektit ja opetusmateriaalit.

2 Teoria

2.1 Raspberry Pi

Raspberry Pi on edullinen, luottokortin kokoinen, Linux-pohjainen tietokone. Sen kehitti hyväntekeväisyysjärjestö Raspberry Pi Foundation, jonka tarkoituksena on kannustaa lapsia ja nuoria oppimaan tietotekniikkaa ja siihen liittyviä aiheita. Raspberry Pi:n pienen koon takia se on helppo kuljettaa minne vain, minkä takia siitä on tullut hyvin suosittu muun muassa erilaisissa IoT-projekteissa. (What is a Raspberry Pi? n.d.)

Raspberry Pi sai alkunsa vuonna 2006 kun Eben Upton ja hänen kollegansa Rob Mullins, Jack Lang ja Alan Mycroft alkoivat huolestua nuorten laskevista ohjelmointi taidoista. He havaitsivat useita ongelmia, jotka vaikuttivat tähän taitotason laskuun:

Koulujen opetusuunnitelmat keskittyivät pääasiassa Word- ja Excel-ohjelmien käyttämiseen, sekä verkkosivujen tekemiseen. Tähän vaikutti myös kotitietokoneiden sekä pelikonsolien suosion kasvaminen. Nämä korvasivat edellisen sukupolven käyttämät Amigat, BBC mikrot, Spectrum ZX:n ja Commodore 64:n, joilla he olivat oppineet ohjelmoimaan.

He kokivat, että he eivät pysty vaikuttamaan paljoa puutteelliseen koulutukseen, mutta he pystyisivät vaikuttamaan tietokoneiden hinnan noususta johtuviin ongelmiin. He alkoivat kehittää alustaa, joka muistuttaisi vanhoja tietokoneita ja jotka käynnistyisivät suoraan ohjelmointiympäristöön. Vuosina 2006 ja 2008, he suunnittelivat useita prototyyppejä, joista syntyi lopulta nykyinen Raspberry Pi.

Vuoteen 2008 mennessä mobiililaitteille suunnitellut prosessorit alkoivat olemaan edullisempia ja tehokkaampia. Tämä mahdollisti erilaiset multimedian käyttötarkoitukset ja teki laitteesta ideaalisemman lapsille, jotka eivät olleet kiinnostuneita pelkästä ohjelmoinnista. Eben, Rob, Jack ja Alan alkoivat tehdä yhteistyötä Pete Lomasin ja David Brabenin kanssa ja muodostivat Raspberry Pi Foundationin. (RPI General History 2015.)

Työn tekohetkellä saatavilla olevat Raspberry Pi:n mallit ovat:

Model A+

Edullinen malli, jossa on 512MB muistia (RAM), ennen vuotta 2016 valmistetuissa malleissa on 265MB muistia. Siinä on yksi USB-portti, 40GPIO pinniä ja se ei sisällä Ethernet-porttia.

Model B+

Lopullinen versio alkuperäisestä Raspberry Pi:stä. Siinä on 512MB muistia, 4 USB-porttia, 40 GPIO pinniä ja Ethernet-portti. Siinä on myös 2 USB-porttia.

Raspberry Pi 2 Model B

Helmikuussa 2015 julkaistu toisen generaation malli. Hyvin samanlainen Model B+:n

kanssa. Käyttää 900MHz neliydin prosessoria ja siinä on 1GB muistia.

Raspberry Pi 3 Model B

Julkaistiin 2016 helmikuussa. Se käyttää 1.2GHz 64-bittistä neliydin suoritinta. Siinä on 1GB muistia, integroitu WLAN ja Bluetooth 4.2. Siinä on 4 USB-porttia.

Raspberry Pi 3 Model B+

Julkaistiin Maaliskuussa 2018. Se käyttää 1.4GHz 64-bittistä neliydin suoritinta. Siinä on 1GB muistia, gigabit Ethernet, integroitu WLAN ja Bluetooth 4.2. Siinä on 4 USB-porttia.

Raspberry Pi 4 Model B

Julkaistiin Kesäkuussa 2019. Se käyttää 1.5GHz 64-bittistä neliydin suoritinta. Siitä on valittavissa kolme eri määrällä muistia varustettua versiota: 1GB, 2GB ja 4GB. Siinä on gigabit Ethernet, integroitu WLAN ja Bluetooth 5. Siinä on 4 USB-porttia.

Raspberry Pi Zero ja Raspberry Pi Zero W/WH

Raspberry Zero on puolet pienempi versio verrattuna Model A+:aan. Siinä on 1GHz yksiytiminen suoritin, 512MB muistia, mini-HDMI-portti, 1 mikro USB on-the-go portti ja mahdollisuus liittää kamera. Zero W:ssä on lisäksi integroitu WLAN ja Bluetooth 4.1. WH mallissa on lisäksi valmiiksi kiinnitetty header liitäntä.

Raspberry Pi Compute module 3

Teollisuuden käyttöön tarkoitettu malli, joka pohjautuu Raspberry Pi 3:een. Siinä on 1.2GHz neliydin suoritin, 1GB muistia ja valinnainen 4GB flash muistia. (FAQs n.d.)

2.2 Linux

2.2.1 Linuxin historia

Linux on maailman tunnetuin ja käytetyin avoimen lähdekoodin käyttöjärjestelmä. Sitä käytetään lukuisissa eri laitteissa, kuten puhelimissa, tietokoneissa, kameroissa ja autoissa. Suurin osa ihmisistä viittaa nimellä Linux kokonaiseen käyttöjärjestelmään, mutta todellisuudessa se tarkoittaa vain Linux-kerneliä. Tämän takia jotkut ihmiset käyttävätkin nimeä GNU/Linux, koska monet sen sisältämät tärkeät työkalut ovat GNU-komponentteja. Kaikki Linux käyttöjärjestelmät eivät kuitenkaan käytä GNU-komponentteja, esimerkiksi Android, joka käyttää Linux-kerneliä, mutta vain hyvin vähän GNU työkaluja.

Linuxin kehitti vuonna 1991 suomalainen Linus Torvalds, entinen Helsingin yliopiston oppilas. Hän kehitti Linuxin ilmaisena vastineena Minix:lle, Unix kloonille, joka oli pääasiassa akateemisessa käytössä. Hänen tarkoituksensa oli alunperin käyttää nimeä Freax, mutta päätyi käyttämään nimeä Linux, kun hänen käyttämänsä palvelimen ylläpitäjä nimesin Torvalds'in käyttämän hakemiston Linux:iksi. Sana Linux tulee Torvalds'in etunimen ja Unix'in yhdistelmästä. (What is Linux? n.d.)

2.2.2 Raspberry Pi OS

Raspberry Pi OS (entiseltä nimeltään Raspbian) on Debian-pohjainen käyttöjärjestelmä Raspberry Pi-tietokoneille. Se on vuodesta 2015 lähtien ollut Raspberry Pi Foundationin virallisesti tukema käyttöjärjestelmä ja on optimoitu käytettäväksi Raspberry Pi laitteissa. Se sisältää useita valmiiksi asennettuja ohjelmia kuten Python, Scratch, Sonic Pi ja Java. Siitä on saatavilla 32- ja 64-bittinen versio. Työssä käytetyssä Raspberry Pi:ssä on asennettuna 32-bittinen versio. (Raspberry Pi OS (previously called Raspbian) 2020.)

Nimenvaihdos Raspbianista Raspberry Pi OS:ään tapahtui toukuussa 2020 Raspberry Pi 4 julkaisun yhteydessä. Syynä muutokseen oli käyttöjärjestelmän uuden 64 bittisen version julkaisu, joka ei 32-bittisen version tavoin pohjaudu alkuperäiseen Raspbian käyttöjärjestelmään. Raspbian projektin luoja Peter Green ei halunnut, että uusi 64-

bittinen käyttöjärjestelmä, joka ei käytä ollenkaan ohjelmistoa vanhasta Raspbianista, käyttäisi Raspbianin nimeä. Tämän seurauksena Raspberry Pi Foundation päätyi käyttämään samaa nimeä molemmille versiolleen. Green myös sanoi, että hän aikoo kuitenkin jatkaa alkuperäistä Raspbian projektiaan, jota tullaan myös todennäköisesti käyttämään tulevien 32-bittisten Raspberry Pi OS versioiden pohjana. (Piltch 2020.)

2.2.3 Ubuntu

Ubuntu on yksi suosituimmista avoimen lähdekoodin Linux-käyttöjärjestelmistä, joka rakentuu Debian-projektin tekemälle työlle. Se sisältää useita hyödyllisiä ohjelmia kuten selaimen, toimisto-ohjelmistoja, kuvankäsittely ja viestintäsovelluksia. Ubuntuun on mahdollista asentaa lisää ohjelmia Ubuntun sovelluskaupan kautta.

Ubuntun versionumerointi perustuu julkaisun päivämäärän kuukauteen ja vuoteen. Ubuntuun julkaistaan kuuden kuukauden välein ohjelmistopaketteja ja joka toinen vuosi siitä julkaistaan pitkäaikaisversio, jota tuetaan viiden vuoden ajan. Työn teko- hetkellä uusin pitkäaikaisversio on 20.04 LTS, jota käytetään myös työssä käytettävässä palvelimessa.

Ubuntun nimi perustuu eteläafrikkalaiseen ideologiaan, joka pohjaa ihmisten keskinäisiin suhteisiin ja kuuliaisuuteen. Itse sana tulee zulu- ja xhosa-kielistä ja sen suomennot tarkoittaa suurin piirtein ”inhimillisyyttä muita kohtaan”. (Esittely n.d.)

2.3 OBD-järjestelmä

2.3.1 OBD-järjestelmän historiaa

OBD on järjestelmä, jonka avulla voidaan selvittää auton vikatilanteita, sekä lukea auton antureita. Se löytyy nykyään lähes jokaisesta henkilöautosta, pakettiautosta ja lava-autosta. OBD:n ensimmäinen versio otettiin käyttöön 80-luvun alkupuolella, tarkoituksena helpottaa autojen päästöjen valvontaa. Tässä vaiheessa OBD ei kuitenkaan ollut standardoitu, joten jokaisella autovalmistajalla oli käytössään omat mitauslaitteet. 90-luvun puolivälissä Kaliforniassa otettiin käyttöön OBDII-standardi,

joka oli vaatimuksena kaikissa Kaliforniassa myydyissä uusissa henkilöautoissa vuodesta 1996 lähtien. Myöhemmin samana vuonna vaatimus otettiin käyttöön ympäri Yhdysvaltoja. Vuonna 2001 standardi otettiin käyttöön EU:ssa kaikissa bensiinikäyttöisissä autoissa ja 2004 kaikissa dieselkäyttöisissä autoissa. (On-Board Diagnostics (OBD) Background History n.d.)

2.3.2 OBDII

OBDII on paranneltu ja standardoitu versio ensimmäisen sukupolven OBD-järjestelmästä. Standardi määrittelee käytettävän liitännän, liitännän johdotuksen, signaalien protokollan ja viestiformaatin. Liitännässä on paikka virtajohdolle, joka mahdollistaa laitteen käytön ilman erillistä virtalähdettä, vaan laite saa virtansa suoraan auton akusta. Standardin ansiosta yhdellä laitteella voidaan lukea tietoa mistä tahansa automerkistä. Eri autonvalmistajat käyttävät eri OBDII-protokollia. Autossa käytetty protokolla on mahdollista tunnistaa OBD-liitännässä käytettävien pinnien perusteella. Alla on lueteltuna käytössä olevat protokollat:

SAE J1850 PWM - käytettävät pinnit: 2, 4, 5, 10 ja 16

SAE J1850 VPW – käytettävät pinnit: 2, 4, 5 ja 16

ISO 9141 – käytettävät pinnit: 4, 5, 7 ja 16

ISO 14230 (KWP2000) (Keyword Protocol 2000) – käytettävät pinnit 4, 5, 7 ja 16

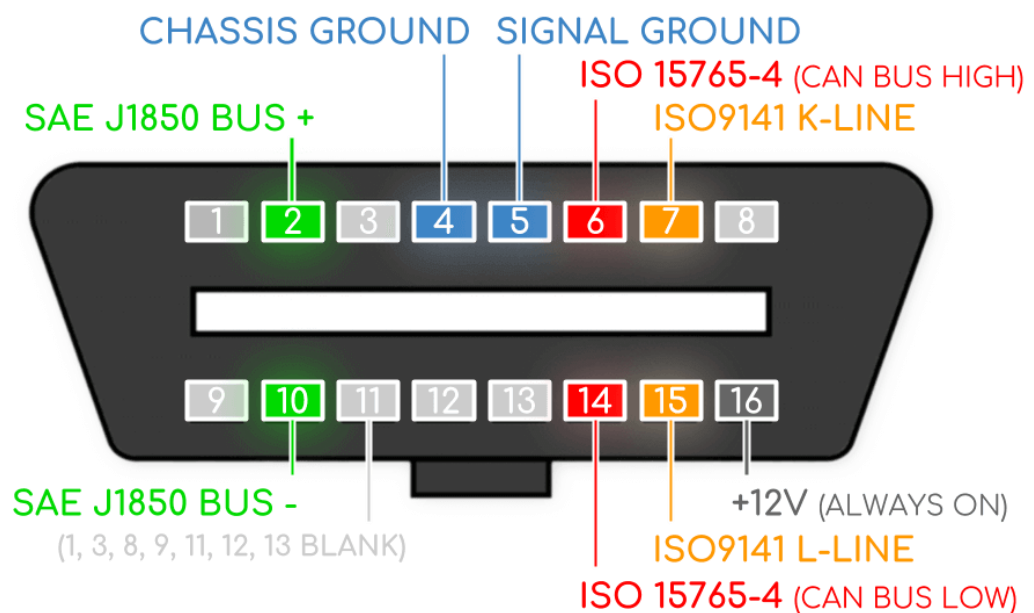
CAN – käytettävät pinnit 4, 5, 6, 14 ja 16

(On-Board Diagnostics (OBD) Background History n.d.)

Alkuperäisessä OBD-järjestelmässä liitäntä saattoi sijaita esimerkiksi auton konepellin tai kojelaudan alla. Uudemman OBD-II liitännän tulee löytyä kaikista autoista kahden

jalan eli 61 cm päästä ohjauspyörästä, ellei autovalmistaja ole erikseen tehnyt pyyntöä poikkeavaan sijaintiin liittyen.

Alla olevassa Kuvio 1 on standardin mukainen 16-pinninen J1962- liitäntä. Kaikki protokollat käyttävät samaa liitäntää, mutta käyttävät eri pinnejä, lukuun ottamatta pinnejä 4 ja 16, jotka ovat maadoitus- ja virtapinnit. (OBD2 Explained - A Simple Intro (2020) 2020.)



Kuvio 1 OBDII-liitin

2.3.3 PID-koodit

PID:t ovat koodeja, joilla voidaan tiedustella dataa OBDII järjestelmältä. OBDII-viestin alussa määritellään käytetty moodi (mode). Se määrittää millaista tietoa ohjausyksiköltä halutaan vastaanottaa. OBDII-standardissa on määriteltynä 10 moodia, mutta ajoneuvonvalmistajien ei kuitenkaan ole pakko tukea niitä kaikkia. Alla on listattuna kaikki moodit ja niiden merkitykset.

Moodi 1: Pyydetään reaaliaikaista tietoa.

Moodi 2: Pyydetään freeze frame tietoa.

Moodi 3: Pyydetään kaikki tallennetut vikakoodit.

Moodi 4: Poistetaan kaikki tallennetut vikakoodit.

Moodi 5: Pyydetään dataa happianturilta, ei CAN-väylälle.

Moodi 6: Pyydetään testituloksia antureilta, joita ei valvota jatkuvasti. Happianturin tulokset haetaan Moodi 6:lla CAN-väyläisissä ajoneuvoissa.

Moodi 7: Pyydetään testiajon aikana tullee vikakoodit

Moodi 8: Ohjelmoidaan tai säädetään auton laitteistoa.

Moodi 9: Pyydetään ajoneuvon tietoja.

Moodi 10: Pyydetään vikakoodit, jotka poistuvat vain vian korjaamalla.

Tietoa kyseltäessä moodi on ensimmäinen datatavu lähetetyssä viestissä. Toinen datatavu viestissä on PID-tavu. Se on numero, millä viitataan haluttuun tietoon. OBDII-standardi tukee valmiiksi useita PID-koodeja, mutta ajoneuvonvalmistajien ei tarvitse kuitenkaan tukea niitä kaikkia. Valmistajilla voi olla myös omia PID-koodeja.

(OBD-II PIDs 2010.)

2.3.4 OBDII vikakoodit

OBD mahdollistaa monien ajoneuvon järjestelmien monitoroinnin. Niihin sisältyvät moottorin hallintamoduulit, runko ja ajotietokoneen ohjaamat järjestelmät. OBD-järjestelmän antamalla koodeilla voidaan määrittää välittömästi missä osassa vika sijaitsee ja minkä tyyppinen vika on. OBD-järjestelmän antama vikakoodi (DTC) koostuu viidestä merkistä, jotka voidaan jaotella tiettyihin ryhmiin.

Ensimmäinen merkki (Kirjain):

Kaikki OBDII vikakoodit alkavat kirjaimella, joka määrittää missä osassa autoa vika sijaitsee.

P (Powertrain) - Sisältää moottorin, vaihteiston ja niihin liittyvät varusteet.

U (Network) - Ajotietokoneen hallitsemat järjestelmät.

B (Body) - Auton runko, pääasiassa matkustamon puolella sijaitsevat osat.

C (Chassis) - Auton runko, sisältää esim. ojauksen, jousituksen ja jarrut.

Toinen merkki (Numero):

Ensimmäistä kirjainta seuraa yleensä numero. Tämä numero voi olla 0 tai 1.

0 – Standardoitu virhekoodi.

1 – Autonvalmistajan oma virhekoodi.

Kolmas merkki (Numero):

Kolmas numero kertoo missä auton järjestelmässä vika sijaitsee. Näitä järjestelmiä on yhteensä kahdeksan.

0 - Polttoaineen ja ilman mittaus sekä ylimääräiset päästöjen hallintalaitteet.

1 - Polttoaineen ja ilman mittaus.

2 - Polttoaineen ja ilman mittaus (injektoripiiri).

3 - Sytytysjärjestelmät tai sytytysvirheet.

4 - Lisäpäästöjen hallinta.

5 - Nopeuden ja tyhjäkäynnin hallintajärjestelmät.

6 - Tietokone ja lähtöpiiri

7 - Vaihteisto

Neljäs ja viides merkki (Numero):

Koodin viimeinen osa koostuu kahdesta numerosta, jotka kertovat tarkan kuvauksen ongelmasta. Numero voi olla mikä tahansa 0 ja 99 välillä.

Nämä yllä luetellut merkit yhdistämällä saadaan kokonainen vikakoodi. Esimerkiksi koodi **P0195** tarkoittaa moottorin öljyn lämpötilasensorin viallista toimintaa. (OBD2 Codes and Meanings 2020.)

2.4 GPS

2.4.1 GPS:n historiaa

GPS on Yhdysvaltojen kehittämä ja rahoittama paikannusjärjestelmä, joka perustuu maata kiertäviin satelliitteihin. Se on viralliselta nimeltään Navstar GPS. Sen kehitys alkoi 1970-luvun alussa ja ensimmäinen NAVSTAR-satelliitti laukaistiin vuonna 1978. Järjestelmä valmistui lopulta vuonna 1993 käytössään 24 satelliittia. Nykyään maata kiertää 31 GPS-satelliittia. (Mai 2017.)

2.4.2 GPS-satelliitit

Satelliitit ovat noin 5 metrin pituisia aurinkokennot avattuina ja painavat noin tonnin verran. Ne kiertävät maata noin 20000 km korkeudessa maan pinnasta tai 25000 km päässä maan keskipisteestä. Jokainen satelliitti kiertää maan kaksi kertaa vuorokaudessa. Satelliittien kiertoradat ovat väliltä 60 astetta pohjoista- ja 60 astetta eteläistä-leveyspiiriä. Tämä tarkoittaa sitä, että satelliitti signaalin saaminen on mahdollista missä tahansa paikassa maailmassa. Maan napoja kohti mennessä satelliitit eivät ole enää suoraan yläpuolella, mikä hieman heikentää paikannustarkkuutta. Satelliittien yksi suurimmista eduista vanhoihin navigointijärjestelmiin verrattuna on, että se toimii kaikissa sääolosuhteissa. (Space Segment 2020.)

2.4.3 GPS-järjestelmä

GPS-järjestelmällä on lukuisia käyttötarkoituksia kuten esimerkiksi: lentäminen, merenkäynti, pelastusoperaatiot, rautatiet, ajoneuvot ja matkapuhelimet, sekä monia muita käyttökohteita, jotka vaativat tarkkaa paikallistamista. Paikallistamisen ehtona on se, että taivaalla on samanaikaisesti vähintään 4 satelliittia, joiden avulla voidaan laskea paikannettavan kohteen sijainti. Paikannus onnistuu myös 3 satelliitin avulla, jos tiedetään jo ennestään kohteen korkeus. Mittauksessa tarkan kellonajan tietäminen on oleellista tarkan tuloksen saamiseksi. Tämän takia satelliiteissa käytetään erittäin tarkkoja atomikelloja. Pienikin virhe ajan mittaamisessa voi aiheuttaa satojen metrien vääristymän sijainnin laskemisessa.

GPS-paikannuksen tarkkuus on yleensä noin 3–10 metriä. Tarkkuuteen vaikuttavat monet tekijät kuten satelliittien sijainti, maastonmuodot, vastaanotin sekä mahdolliset ympäröivät rakennukset. Tarkin paikannus saadaan aukealla alueella, josta on suora näköyhteys taivaalle. Kun kohteen sijainti on paikallistettu, voidaan sen avulla laskea muitakin tietoja kuten: nopeus, suunta, etäisyys määränpäähän ja auringon nousu- ja laskuajat. (What is GPS? n.d.)

3 Käytetyt ohjelmistot

3.1 InfluxDB

InfluxDB on Influxdatan kehittämä avoimen lähdekoodin aikasarja tietokanta, jonka avulla on mahdollista käsitellä paljon dataa nopeasti. Sen käyttötarkoituksena on pystyä varastoimaan suuria määriä aikasarja dataa erilaisia käyttötarkoituksia varten, kuten sovellusten toiminnan monitorointi, IoT sensoridata ja reaaliaikainen analyysi. InfluxDB on kirjoitettu kokonaan GO-kielellä. (InfluxDB 1.8 documentation n.d.)

Aikasarjatietokannat kuten InfluxDB ovat ideaalisia aikaleimatun datan käsittelyyn ja niitä käytettiin alunperin pääasiassa finanssialan datan hallintaan ja analysointiin osakemarkkinoilla. Nykyään käyttötarkoituksia on monia ja erityisesti IoT parissa käyttötarkoitukset ovat lukemattomat. Lähes kaikkeen on nykyään mahdollista asentaa jonkunlainen sensori, autoihin, kaduille, tehtaisiin, vaatteisiin, puhelimiin, kodinkoneisiin ja jopa ihmiskehoihin. Tästä johtuen tarve uusille ja paremmille tavoille säilöä tätä dataa on suuri ja aikasarjatietokannat ovat tällaiselle datalle ideaalisia. (Time series database (TSDB) explained 2020.)

InfluxDB:tä käytetään komentolinjan kautta ja vaikka se toiminnaltaan eroaa SQL-tietokannoista, sen kyselyt muistuttavat SQL-kyselyjä, mikä tekee siitä helpon käyttää käyttäjille, joilla on kokemusta SQL-tietokannoista. (Ks. Kuvio 2).

```

Connected to http://localhost:8086 version 1.8.0
InfluxDB shell version: 1.8.0
> auth
username: admin
password:
> USE OBDII
Using database OBDII
> SELECT "RPM" FROM "OBDII" WHERE "RPM" > 1000
name: OBDII
time                RPM
----                -
1593970800853262853 1063
1593970802305015728 1067
1593970803757067751 1044
1593970805179064967 1048

```

Kuvio 2 InfluxDB komentolinja

3.2 Grafana

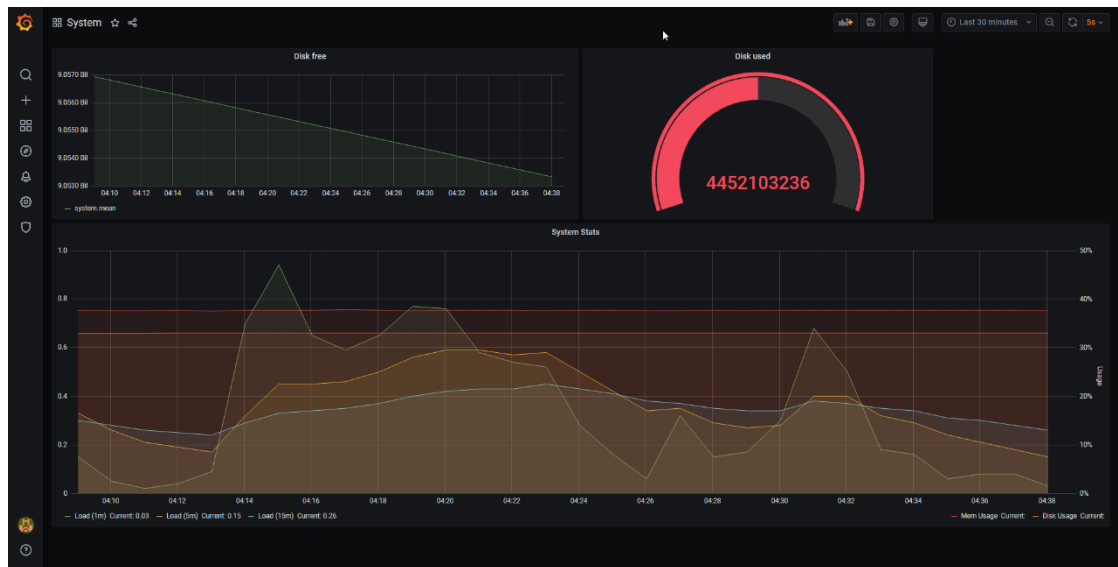
Grafana on avoimen lähdekoodin analyysi- ja monitorointiohjelma, jonka avulla voidaan luoda aikasarjadatasta erilaisia graafisia paneeleja. Grafanaa käyttää useat suuret yritykset mm. Stack Overflow, eBay, PayPal, Uber ja Digital Ocean. Se tukee yli 30 eri tietokantaa kuten MySQL, PostgreSQL, Graphite, Elasticsearch, OpenTSDB, Prometheus sekä InfluxDB, joka on olennaisena osana tässäkin projektissa.

(Kili 2018.)

Grafanalla voidaan visualisoida tietokannasta noudettua dataa reaaliajassa käyttäjälle helpompilukuisiksi monilla erilaisilla tavoilla, kuten graafeilla, taulukoilla, kartoilla ja mittareilla. Tämän lisäksi Grafanaan on saatavilla lukuisia ladattavia yhteisön tekemiä lisäosia, paneeleja ja sovelluksia.

Grafanan käyttöliittymä koostuu hallintapaneelistä, joka puolestaan koostuu pienemmistä paneeleista, joissa visualisoidaan noudettua dataa. Paneeleja on mahdollista muokata halutun näköiseksi, värisiksi ja kokoisiksi. (Ks. Kuvio 3). Paneelien muokkaaminen on tehty helpoksi ja niiden muokkaaminen onnistuu raahaamalla niitä haluttuun kohtaan ja venyttämällä paneeleja oikean kokoisiksi. Jokainen paneeli voidaan

muokata erikseen halutun malliseksi ja niihin on myös mahdollista hakea dataa useasta eri datalähteestä. Paneeleja on tarvittaessa mahdollista tehdä myös itse.



Kuvio 3 Grafanan käyttöliittymä

Paneeleille on myös mahdollista asettaa hälytyksiä. Tietyn raja-arvon alittuessa tai ylittyessä voidaan lähettää automaattisesti huomautus viesti halutulla tavalla. Viesti voidaan toimittaa sähköpostilla tai erilaisiin palveluihin, kuten Discord, Slack ja Microsoft Teams. (Alert notifications n.d.)

3.3 Nginx

Nginx, lausutaan engine x, on Igor Sysoevin kehittämä maksuton avoimen lähdekoodin, HTTP-palvelin, välitys- ja käänteisvälityspalvelin sekä sähköpostipalvelinohjelma. Se on ollu jo pidemmän aikaa käytössä raskaasti kuormitettujen venäläisten verkkosivustojen käytössä, kuten Yandex, Mail.ru, VK ja Rambler. Netcraft:in mukaan se oli käytössä 25.43 % vilkkaimmista verkkosivuista kesäkuussa 2020. Sen ominaisuuksiin kuuluu muun muassa verkkosivujen tarjonta, palvelimien kuormituksen tasapainoit-

taminen ja vikasietoisuus, sekä käyttäjien uudelleenohjaus sähköpostipalvelimille ulkoisen autentikointipalvelimen kautta.

(nginx n.d.)

3.4 Python-OBd

Python-OBd on Python-kirjasto auton OBdII-datan käsittelyyn. Sen avulla on mahdollista lukea autosta reaaliaikaista sensoridataa ja diagnosoida moottorin antamia koodoja. Se on suunniteltu toimimaan Raspberry Pi:n kanssa. Python-OBd mahdollistaa auton vikakoodien kyselyn PID-koodien avulla. Näiden koodien avulla on mahdollista tehdä kyselyitä auton OBdII-järjestelmälle ja saada koodia vastaavan sensorin arvoja. Python-OBd sisältää valmiiksi suurimman osan yleisistä PID-koodeista, mutta mahdollistaa myös kustomoitujen kyselyjen luomisen. Python-OBd sisältää myös sisäänrakennetun debug-tilan, jonka avulla voidaan selvittää mahdollisia ongelmia Python-OBd:n toiminnassa. Kuvio 4 näkyy debug-tilan antama tuloste onnistuneesta ajoneuvoon yhdistämisestä. (Getting started n.d.)

Successful Connection

```
[obd] ===== python-OBd (v0.4.0) =====
[obd] Explicit port defined
[obd] Opening serial port '/dev/pts/2'
[obd] Serial port successfully opened on /dev/pts/2
[obd] write: 'ATZ\r\n'
[obd] wait: 1 seconds
[obd] read: 'ATZ\rELM327 v2.1\r'
[obd] write: 'ATE0\r\n'
[obd] read: 'ATE0\rOK\r'
[obd] write: 'ATH1\r\n'
[obd] read: 'OK\r'
[obd] write: 'ATL0\r\n'
[obd] read: 'OK\r'
[obd] write: 'ATSPA8\r\n'
[obd] read: 'OK\r'
[obd] write: '0100\r\n'
[obd] read: '7E8 06 41 00 FF FF FF FC\r'
[obd] write: 'ATDPN\r\n'
[obd] read: 'A8\r'
[obd] Connection successful
[obd] querying for supported PIDs (commands)...
[obd] Sending command: 0100: Supported PIDs [01-20]
[obd] write: '0100\r\n'
[obd] read: '7E8 06 41 00 FF FF FF FC\r'
[obd] Sending command: 0120: Supported PIDs [21-40]
[obd] write: '0120\r\n'
[obd] read: '7E8 06 41 20 FF FF FF FC\r'
[obd] Sending command: 0140: Supported PIDs [41-60]
[obd] write: '0140\r\n'
[obd] read: '7E8 06 41 40 FF FF FF FE FB\r'
[obd] finished querying with 93 commands supported
[obd] =====
```

Kuvio 4 Python-OBd debug-tila

3.5 gpsd

gpsd on palvelu, jonka avulla voidaan monitoroida yhtä tai useampaa tietokoneen UBS- tai Serial-porttiin kiinnitettyä GPS-laitetta. Se mahdollistaa kaiken datan tiedustelun TCP-portin 2947 kautta. Sen avulla useammat GPS-sovellukset voivat jakaa datan keskenään, ilman kilpailua tai datan menetyistä. Se muotoilee GPS-laitteiden antaman datan helpommin luettavaan ja käsiteltävään muotoon. Gpsd on käytössä useissa eri kohteissa, kuten karttapalvelussa Android puhelimissa, droneissa, robottisukellusveneissä ja kuskittomissa autoissa. Se on myös yleinen miehitetyissä lentokaluksissa, merinavigointi systeemeissä sekä sotilaallisissa ajoneuvoissa. Sovelluksia, jotka käyttävät gpsd:tä ovat muun muassa: Kismet, GpsDrive, gpeGPS, roadmap, roadnav, navit, viking, tangogps, foxtrot, obdgpslogger, geohist, LiveGPS, geoclue, qlandkartegt, gpredict, OpenCPN, gpsd-navigator, gpsd-ais-viewer, ja firefox/mozilla. Tämän lisäksi gpsd on ollut käytössä Android-käyttöjärjestelmän versiosta 4 lähtien, mutta mahdollisesti jo aiemmissa versioissa. (About gpsd 2020.)

4 Työn suunnittelu

Työn suunnittelu alkoi työssä käytettävien laitteiden ja ohjelmistojen valinnalla. Viimeistellyn laitteen oli tarkoitus olla kohtalaisen pienikokoinen ja edullinen, joten päädyttiin käyttämään Raspberry Pi:tä projektin pohjana. Malliksi valittiin Raspberry Pi 3 Model B+, joka vaikutti riittävältä projektin vaatimukseen. (Ks. Kuvio 5).



Kuvio 5 Raspberry Pi Model 3 B+

Auton tietojen keräämiseen tarvittiin laite, jolla pystyy lukemaan auton antamia OBDII-viestejä. Tähän tarkoitukseen oli saatavilla lukuisia erilaisia OBDII-adaptoreita. Adaptoreita oli mahdollista saada langallisena, USB-porttiin yhdistettävänä, sekä langattomina Bluetooth:lla tai WiFi:llä varustettuna. Valinta jäi lopuksi langallisen ja langattoman Bluetooth-adapterin välille. Langallinen olisi todennäköisesti helppokäyttöisempi ja luotettavampi, mutta veisi ylimääräistä tilaa. Langaton taas mahdollistaisi laitteen testauksen, ilman, että pitäisi testatessa olla autossa istumassa. Loppujen lopuksi päädyttiin Partco.fi-verkkokaupasta löytyvään OBD2 bluetooth elm327-adaptoriin. (Ks. Kuvio 6).



Kuvio 6 OBDII Bluetooth-adapteri

Lisäksi laitteeseen tarvittiin keino seurata sen sijaintia, joten tarvittiin myös jonkinlainen GPS-vastaanotin. Vastaanottimia oli pääasiassa kahden tyyliä: USB-malleja ja GPS-moduuleja. Moduulin asentaminen olisi ollut sen vaatiman johdottamisen takia huomattavasti monimutkaisempaa, joten päädyttiin valitsemaan USB:llä toimiva Globalsat BU-353S4 - SiRF IV -GPS-vastaanotin. (Ks. Kuvio 7).



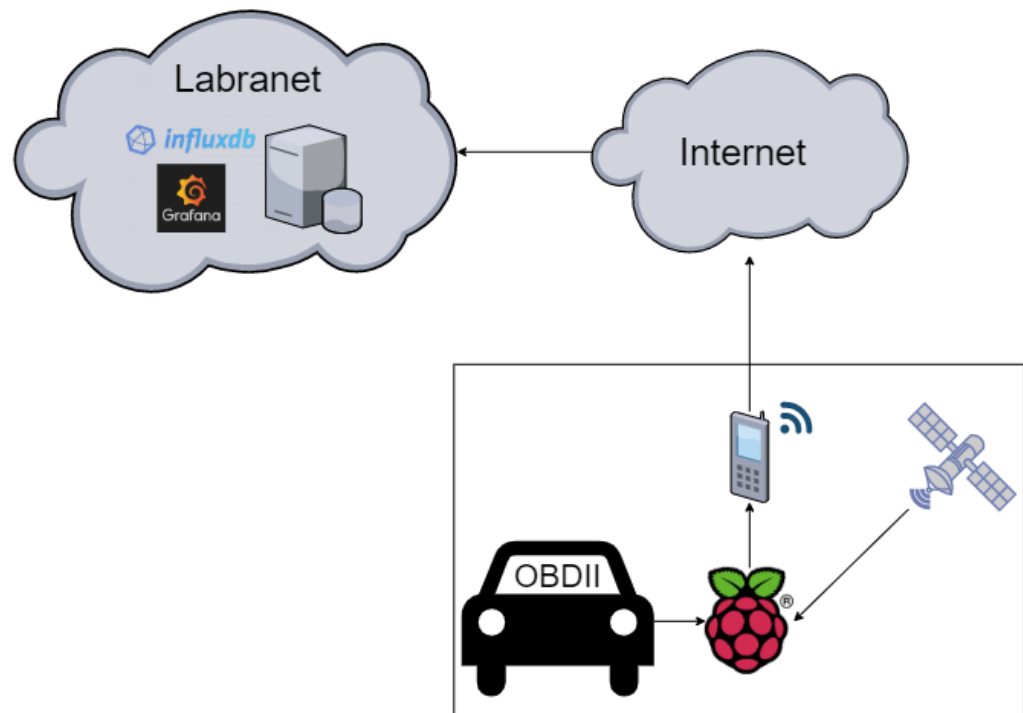
Kuvio 7 GPS-paikannin

Laitteiden keräämän datan talteen keräämistä varten tarvittiin jonkunlainen tietokanta ja koska haluttiin, että dataa olisi mahdollista käsitellä reaaliajassa, oli aikasarjatietokanta tähän käyttötarkoitukseen hyvä valinta. Tietokannaksi päädyttiin valitsemaan InfluxDB sen helppokäyttöisyyden ja selkeän dokumentaation takia. Tämän lisäksi sen käyttämiseen löytyy runsaasti yhteisön tekemiä ohjeita.

Datan visualisointia varten valittiin Grafana sen siistin ulkoasun ja helppokäyttöisyyden takia. Grafana toimii myös erittäin hyvin InfluxDB:n kanssa, mikä tekee siitä hyvän vaihtoehdon valitun tietokannan rinnalle.

Tietokantaa varten tarvittiin palvelin, johon olisi mahdollista päästä myös ulkoverkosta käsin. Tätä varten päädyttiin luomaan virtuaalikone JAMK:in Labranettiin ja saatiin samalla palvelimelle myös julkinen IP-osoite.

Suunnitelmana oli myös, että datan lähettäminen palvelimelle tapahtuisi hyödyntäen 4G-yhteyttä. Kuvio 8 nähdään suunnitelma mahdollisesta lopullisesta kokoonpanosta.



Kuvio 8 Suunniteltu loppukokoonpano

5 Toteutus

Työn toteutus aloitettiin vaadittavien laitteiden asennuksella ja konfiguroinnilla. Näihin laitteisiin kuuluu datan keräämiseen tarkoitettu Raspberry Pi ja datan käsitte-
lyyn tarkoitettu virtuaalikone Labranetissä.

5.1 Raspberry Pi:n käyttöönotto

Raspberry Pi:n käyttöönotto aloitettiin asentamalla SD-kortille Raspbian-käyttöjär-
jestelmä. Tämä onnistuu helposti seuraamalla Raspberry Pi:n verkkosivulta löytyviä

ohjeita ja käyttämällä Raspberry Pi Imager ohjelmaa. Tässä tapauksessa ei kuitenkaan ollut käytettävissä SD-kortin lukijaa, joten asennus tapahtui käyttämällä Android-tablettia, johon oli asennettuna Pi SD Card Imager-ohjelma.

Tämän jälkeen kytkettiin Raspberry Pi monitoriin HDMI-kaapelilla kuvan saamiseksi. Kun Laite käynnistetään ensimmäistä kertaa, ilmestyy ruudulle käynnistytyn jälkeen "Welcome to Raspberry Pi"-niminen ohjelma, joka auttaa käyttäjää laitteen perusasetusten määrittelyssä, Kuvio 9 mukaisesti.



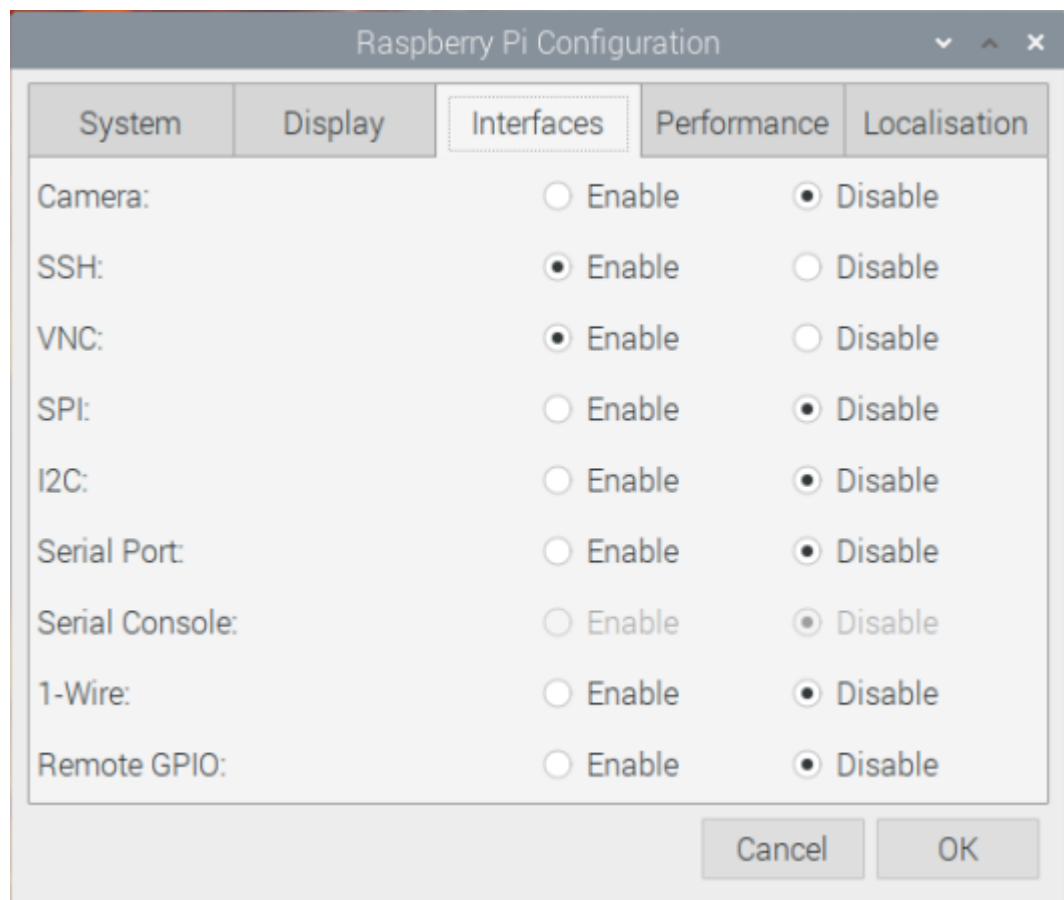
Kuvio 9 Welcome to Raspberry Pi

Painamalla "Next"-painiketta, aloitetaan asetusten määrittely. Ensimmäiseksi käyttäjää pyydetään ilmoittamaan maa, kieli ja aikavyöhyke. Tämän jälkeen pyydetään vaihtamaan järjestelmän salasana, mikä on vahvasti suositeltavaa turvallisuussyistä. Sitten pyydetään valitsemaan käytettävä langaton verkko, jos laite ei ole jo yhdistettynä verkkoon langallisesti. Verkkoon yhdistämisen jälkeen ohjelma tarkistaa saatavilla olevat päivitykset ja asentaa ne. Tämän jälkeen pyydetään uudelleenkäynnistämään Raspberry Pi päivitysten loppuun asentamiseksi, mikäli sille on tarvetta. Päivitysten asentaminen onnistuu myös seuraavilla komennoilla.

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

Koska laitetta halutaan käyttää monesti etäyhteydellä, on hyvä tehdä Raspberry Pi:n asetuksiin joitakin muutoksia. Asetuksista otetaan käyttöön SSH, jonka avulla voidaan ottaa yhteys Raspberry Pi:n terminaaliin, sekä VNC, mikä mahdollistaa etätyöpöytäyhteyden. (ks. Kuvio 10).



Kuvio 10 Interface-asetukset

5.2 Raspberry Pi:n ohjelmien asennus ja konfigurointi

Seuraavaksi voidaan aloittaa tarvittavien ohjelmien asennus Raspberry Pi:lle. Ensimmäiseksi asennetaan OBDII-datan lukemista varten python-OBDD. Asennus tapahtuu seuraavalla komennolla.

```
$ pip install obd
```

Koska OBDII-adapteri toimii bluetoothilla, asennetaan myös bluetooth yhteyteen vaadittavat paketit seuraavalla komennolla.

```
$ sudo apt-get install bluetooth bluez blueman
```

Seuraavaksi asennetaan GPS-datan lukemista varten gpsd. Asentaminen tapahtuu komennolla.

```
$ sudo apt-get install gpsd
```

gpsd:n konfigurointi

Asennuksen jälkeen täytyy tarkistaa, että gpsd lukee dataa oikeasta laitteesta muokkaamalla gpsd:n asetukset tiedostoa. Tiedostoa päästään muokkaamaan komennolla:

```
$ sudo nano /etc/default/gpsd
```

Avatusta tiedostosta löytyy rivi "DEVICES", joka muutetaan käyttämään käytettyä GPS-laitetta. Kuvio 11 nähdään muokatun tiedoston sisältö.

```
# Default settings for the gpsd init script and the hotplug wrapper.

# Start the gpsd daemon automatically at boot time
START_DAEMON="true"

# Use USB hotplugging to add new USB devices automatically to the daemon
USB_AUTO="false"

# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB0"

# Other options you want to pass to gpsd
GPSD_OPTIONS=""
```

Kuvio 11 gpsd asetukset

GPS-laitteen käyttämä polku on mahdollista selvittää kiinnittämällä laite USB-porttiin ja ajamalla seuraavan komennon.

```
$ dmesg | grep -i usb
```

Tämä komento näyttää liitettyjen USB-laitteiden järjestelmälle lähettämät viestit, joiden avulla voidaan selvittää viimeiseksi liitetty laite, joka tässä tapauksessa on juuri liitetty GPS-laite.

Tiedoston muokkauksen jälkeen voidaan käynnistää gpsd-palvelu ja määrittää, että se käynnistyy automaattisesti tietokoneen uudelleenkäynnistyksessä seuraavilla komenoilla.

```
$ sudo systemctl enable gpsd
```

```
$ sudo systemctl start gpsd
```

Nyt voidaan testata GPS:n toimintaa komennolla:

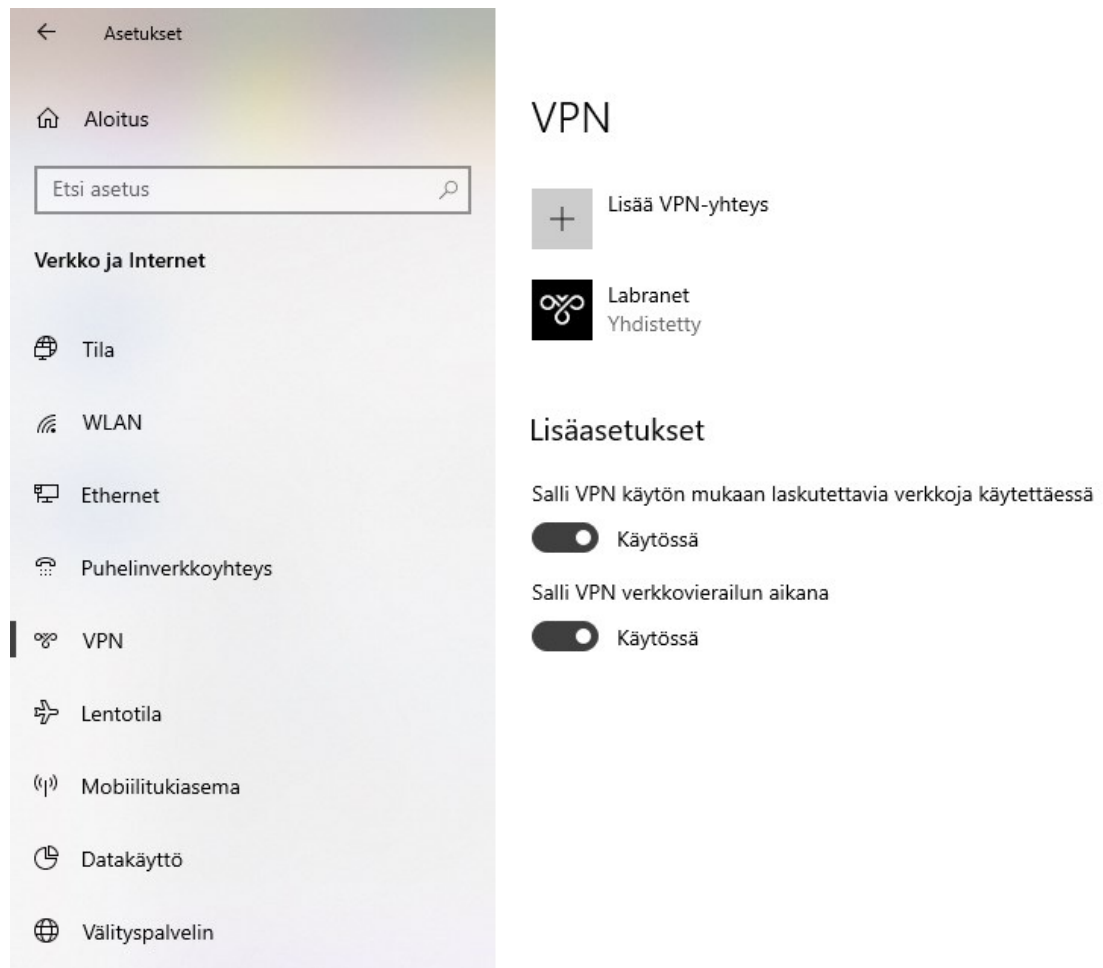
```
$ gpspipe -w -n 5
```

Jos GPS toimii normaalisti, komennon antaman tulosteen "TVP"-rivillä näkyy sen antamia koordinaatteja.

5.3 Ubuntun ohjelmien asennus ja konfigurointi

5.3.1 Vaadittavat alkuvalmistelut ja ohjelmien asennus

JAMK:in Labranetissä sijaitsevalle virtuaalikoneelle voidaan ottaa yhteys SSH:lla, mutta ensin on muodostettava verkkoon VPN-yhteys annettuja käyttäjätunnuksia hyödyntäen. Windows 10-tietokoneella tämä tapahtuu "Asetukset"-valikon "VPN"-osiosta. (Ks. Kuvio 12).



Kuvio 12 Windows 10 VPN asetukset

Virtuaalikoneelle asennettu Ubuntu tulee pääosin valmiiksi JAMK:in helpdeskin konfiguroimana. Siihen on kuitenkin hyvä tehdä pieniä muutoksia ennen tarvittavien ohjelmien asennusta. Ensiksi luodaan uusi käyttäjä sudo-oikeuksilla. Käyttäjän nimeksi valitaan tässä tapauksessa “obd”. Tämä tapahtuu komennolla:

```
$ adduser obd
```

Tämän jälkeen pyydetään valitsemaan salasana sekä ilmoittamaan valinnaisia tietoja käyttäjästä. Seuraavaksi lisätään luotu käyttäjä sudo-ryhmään komennolla:

```
$ usermod -aG sudo obd
```

Nyt luodulla käyttäjällä on sudo-oikeudet. Nyt voidaan vaihtaa juuri luodulle käyttäjälle komennolla:

```
$ su obd
```

Tämän jälkeen varmistetaan, että laitteen ohjelmistot ovat ajan tasalla komennoilla:

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

Seuraavaksi asennetaan tarvittavat ohjelmat Nginx, InfluxDB ja Grafana. Nginx:n asennus tapahtuu komennolla:

```
$ sudo apt-get install nginx
```

InfluxDB:n asennus onnistuu helposti influxdatan verkkosivuilta löytyviä ohjeita seuraten. (Install InfluxDB OSS n.d.) Ensin lisätään InfluxDatan repository komennoilla:

```
$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

```
source /etc/lsb-release
```

```
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

Tämän jälkeen ladataan InfluxDB ja käynnistetään palvelu, sekä asetetaan se käynnistymään automaattisesti tietokoneen käynnistyessä komennoilla:

```
$ sudo apt-get update && sudo apt-get install influxdb
```

```
$ sudo systemctl unmask influxdb.service
```

```
$ sudo systemctl start influxdb
```

```
$ sudo systemctl enable influxdb.service
```

(Install InfluxDB OSS n.d.)

Grafanan asennus aloitetaan lisäämällä Grafanan repository komennoilla:

```
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

```
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee  
/etc/apt/sources.list.d/grafana.list
```

Sitten asennetaan Grafana komennoilla, sekä käynnistetään palvelu ja asetetaan se käynnistymään automaattisesti tietokoneen käynnistyessä komennoilla:

```
$ sudo apt update && sudo apt install -y Grafana
```

```
$ sudo systemctl unmask grafana-server.service
```

```
$ sudo systemctl start grafana-server
```

```
$ sudo systemctl enable grafana-server.service
```

5.3.2 Nginx kofigurointi

Labranetin palomuurin määrittelystä johtuen, on kaikki ulkoverkosta saapuva liikenne estetty lukuunottamatta portteja 443 ja 80. Tämän seurauksena ei ole mahdollista käyttää InfluxDB:n ja Grafanan normaalisti käyttämiä portteja 8086 ja 3000. Ratkaisuna tähän lähetetään InfluxDB:lle ja Grananalle tarkoitettu liikenne normaali porttien sijaan porttiin 80 ja ohjataan se sitten kulkemaan Nginx:n käänteis-välityspalvelimen kautta oikeille porteille.

Aloitetaan poistamalla käytöstä Nginx:n oletuksena asentama konfiguraatiotiedosto. Tämä tapahtuu komennolla:

```
$ unlink /etc/nginx/sites-enabled/default
```

Tämän jälkeen siirrytään äsken mainittuun hakemistoon ja luodaan käytöstä poistetun tiedoston tilalle uusi tiedosto käänteis-välityspalvelinta varten komendoilla:

```
$ cd /etc/nginx/sites-available
```

```
$ nano reverse-proxy.conf
```

Tiedostoon lisättiin Kuvio 13 näkyvä konfiguraatio, joka ohjaa ulkoverkosta porttiin 80 saapuvan liikenteen oikeille porteille ja mahdollistaa Grafanan käyttämisen ulkoverkosta selaimella osoitteessa <http://obd-tracker.labranet.jamk.fi/grafana>.

```
server {  
    listen 80;  
  
    server_name obd-tracker.labranet.jamk.fi;  
  
    location /grafana/ {  
        proxy_pass http://localhost:3000;  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection 'upgrade';  
        proxy_set_header Host $host;  
        proxy_cache_bypass $http_upgrade;  
    }  
  
    location / {  
        proxy_pass http://localhost:8086;  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection 'upgrade';  
        proxy_set_header Host $host;  
        proxy_cache_bypass $http_upgrade;  
    }  
}
```

Kuvio 13 Nginx konfiguraatio

Jotta uusi konfiguraatio saadaan otettua käyttöön kopioidaan se symbolista linkkiä käyttäen hakemistoon `/etc/nginx/sites-enabled`.

```
$ ln -s /etc/nginx/sites-available/reverse-proxy.conf /etc/nginx/sites-enabled/reverse-proxy.conf
```

Konfiguraation toimivuutta voidaan testata komennolla:

```
$ sudo nginx -t
```

Mikäli virheitä ei ole antaa nginx siitä Kuvio 14 mukaisen ilmoituksen.

```
obd@obd-tracker:~$ sudo nginx -t
[sudo] password for obd:
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
obd@obd-tracker:~$
```

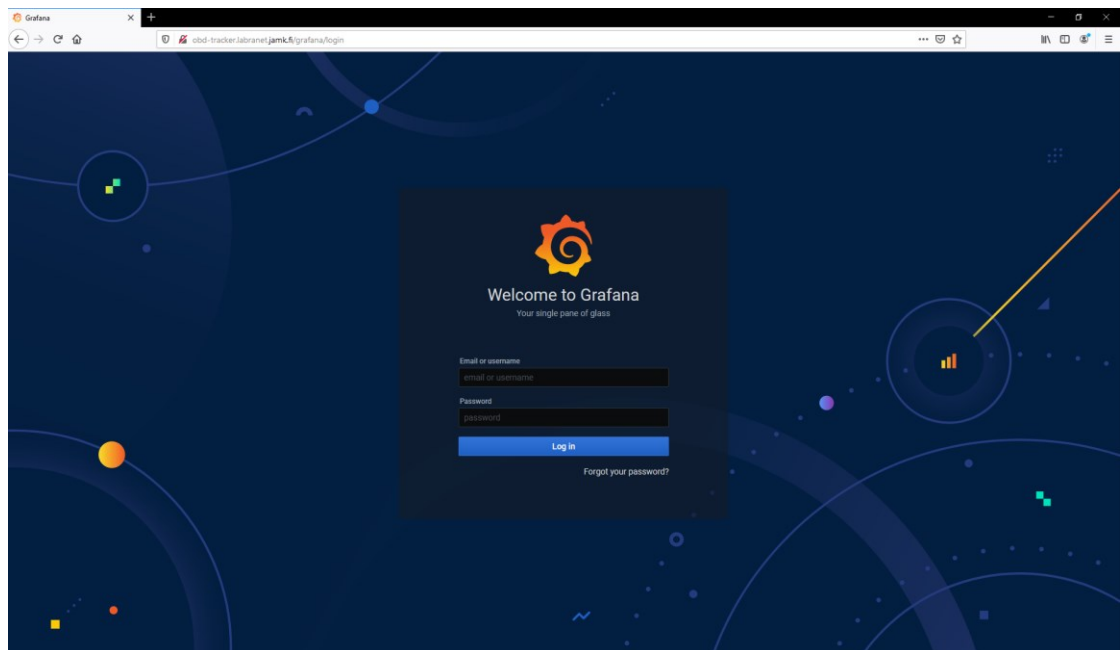
Kuvio 14 nginx konfiguraation testaus

Käynnistetään nginx-palvelu ja asetetaan se käynnistymään automaattisesti.

```
$ sudo systemctl start nginx
```

```
$ sudo systemctl enable nginx
```

Nyt voidaan testata Nginx:n toimivuutta kirjoittamalla osoitekenttään <http://obd-tracker.labranet.jamk.fi/grafana>. Tämä avaa selaimessa Grafanan sisäänkirjautumisivun. (Ks. Kuvio 15).



Kuvio 15 Grafana ulkoverkosta

5.3.3 InfluxDB konfigurointi

Aloitetaan luomalla InfluxDB:een uusi käyttäjä käyttäen InfluxDB:n terminaalia. Terminaali avataan komennolla:

```
$ influx
```

Komento avaa Kuvio 16 mukaisen näkymän.

```
obd@obd-tracker:~$ influx
Connected to http://localhost:8086 version 1.8.0
InfluxDB shell version: 1.8.0
> █
```

Kuvio 16 InfluxDB terminaali

Seuraavaksi halutaan luoda admin-käyttäjä, jolla on oikeudet kirjoittaa tietokantoihin ja muokata niitä. Käyttäjän luominen ja sille admin oikeuksien antaminen tapahtuu komennolla:

```
> CREATE USER admin WITH PASSWORD 'salasana' WITH ALL PRIVILEGES
```

SHOW USERS-komennolla voidaan listata olemassa olevat käyttäjät ja tarkastella niiden oikeuksia. (Ks. Kuvio 17).

```
> SHOW USERS
user  admin
-----
admin true
>
```

Kuvio 17 SHOW USERS

Tämän jälkeen luodaan tietokannat OBDII ja GPS dataa varten. Tietokannan luominen tapahtuu komennoilla:

```
> CREATE DATABASE obd
```

```
> CREATE DATABASE gpsd
```

SHOW DATABASES-komennolla voi listata olemassa olevat tietokannat.

Kun tietokannat ja käyttäjä on luotu, tehdään muutamia muutoksia InfluxDB:n konfiguraatio-tiedostoon. Avataan tiedosto komennolla:

```
$ sudo nano /etc/influxdb/influxdb.conf
```

Konfiguraatio-tiedostossa voidaan ottaa käyttöön ja muokata haluttuja asetuksia poistamalla asetuksen edestä kommentointi-merkki. Tässä tapauksessa otetaan käyttöön HTTP endpoint, jotta yhdistäminen tietokantaan HTTP:tä käyttämällä onnistuu.

Sen lisäksi otetaan käyttöön autentikointi, joka pakottaa autentikoinnin InfluxDB:een muutoksien tekemiseksi. Muokattu konfiguraatio näkyy Kuvio 18.

```
[http]
# Determines whether HTTP endpoint is enabled.
enabled = true

# Determines whether the Flux query endpoint is enabled.
# flux-enabled = false

# Determines whether the Flux query logging is enabled.
# flux-log-enabled = false

# The bind address used by the HTTP service.
# bind-address = ":8086"

# Determines whether user authentication is enabled over HTTP/HTTPS.
auth-enabled = true
```

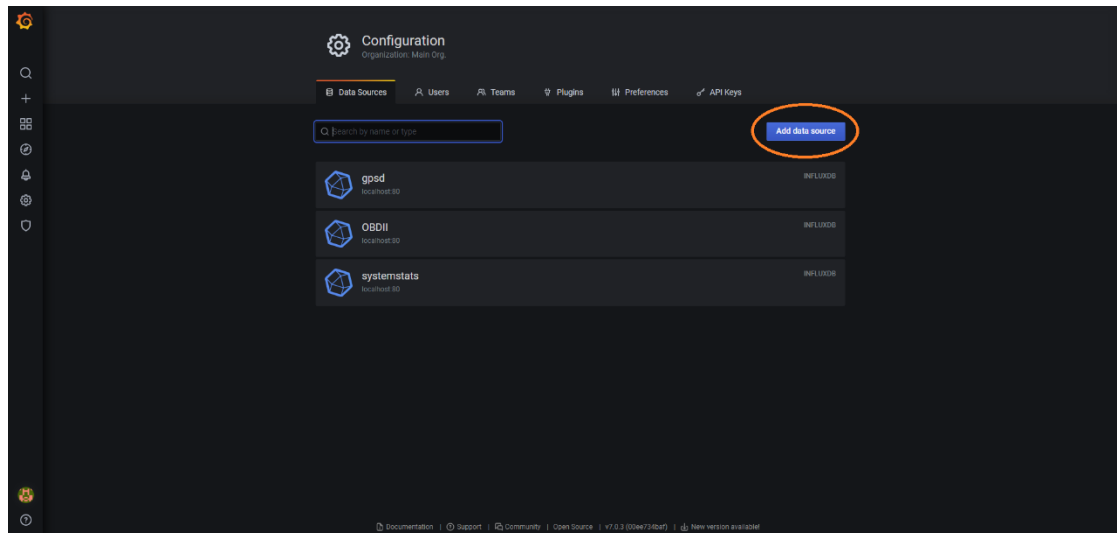
Kuvio 18 InfluxDB konfiguraatio

5.3.4 Grafanan konfigurointi

Aloitetaan avaamalla selaimessa Grafanan sisäänkirjautumissivu aiemman Kuvio 15 mukaisesti. Käyttäjätunnus ja salasana ovat oletuksena “admin” ja “admin”. Sisäänkirjautumisen jälkeen pyydetään vaihtamaan salasana.

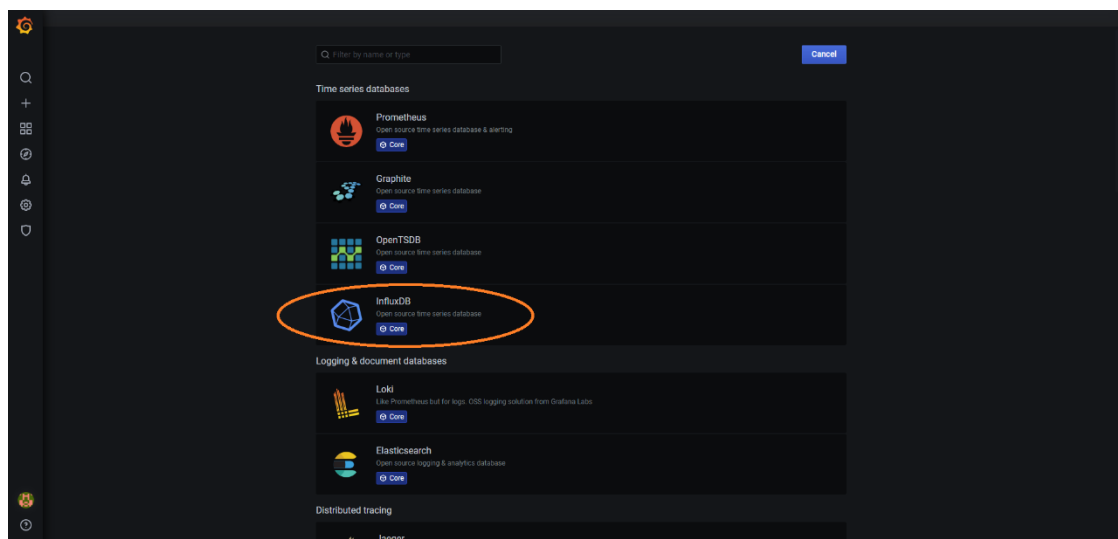
Salasanan vaihdoksen jälkeen lisätään datalähteitä (datasources). Uuden datalähteen lisääminen onnistuu vasemmasta laidasta löytyvästä Configuration-valikosta.

Lisätään uusi datalähde Kuvio 19 rengastetusta painikkeesta.



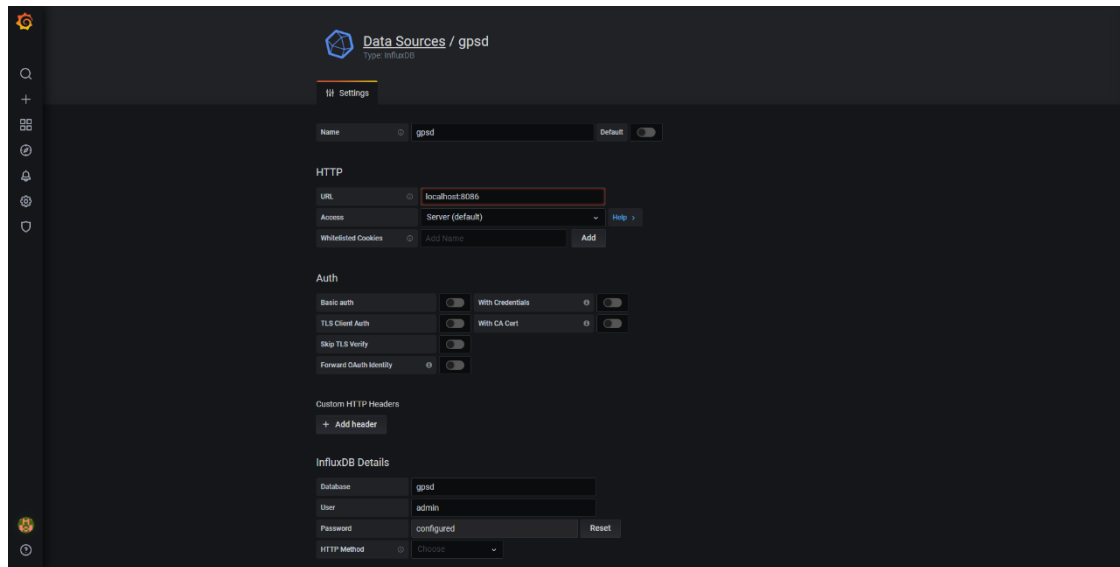
Kuvio 19 Uuden datalähteen lisääminen

Avautuvalta sivulta valitaan käytettävä tietokanta, tässä tapauksessa InfluxDB.
(Ks. Kuvio 20).



Kuvio 20 InfluxDB datalähde

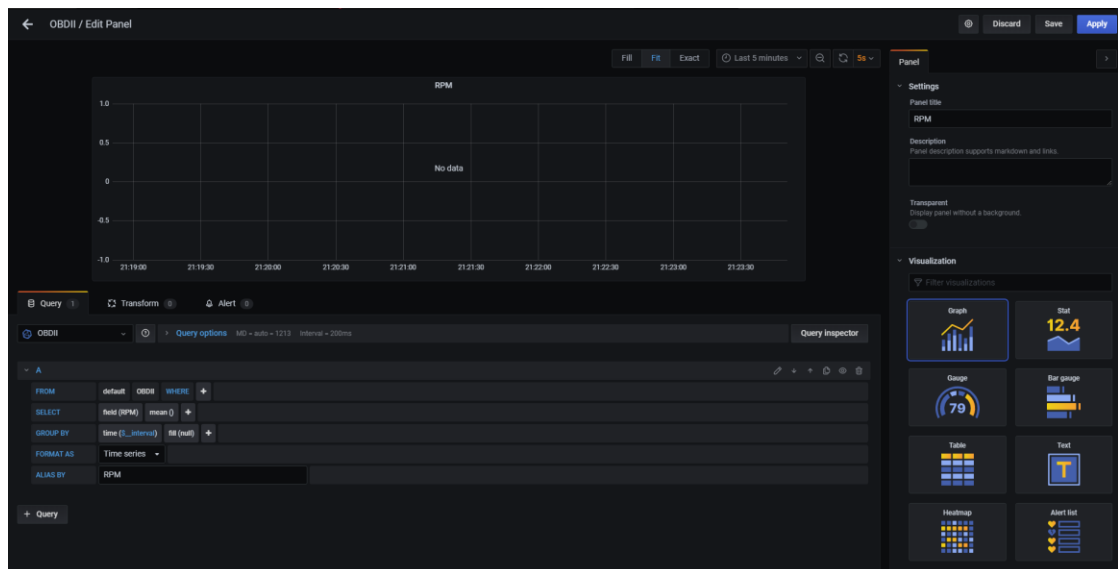
Seuraavaksi määritellään datalähteen asetukset, kuten nimi, käytettävä tietokanta, sekä autentikointiin käytettävät tunnukset Kuvio 21 mukaisesti. Valitaan käytettäväksi tietokannaksi gpsd.



Kuvio 21 gpsd datalähteen asetukset

Lopuksi tallennetaan tehdyt muutokset painamalla “Save & Test”-näppäintä sivun alalaidasta. Jos kaikki toimii odotetusti, antaa Grafana ilmoituksen “Data source is working”. Tämän jälkeen tehdään myös samaa prosessia noudattaen datalähde obd-tietokannalle.

Paneelien luominen tapahtuu oikeassa laidassa olevaa plus-merkistä ja avautuvalla sivulla “Add new panel”-nappia painamalla. Tämä avaa paneelin asetukset-sivun, josta paneelia pääsee muokkaamaan halutunlaiseksi. Oikeasta laidasta löytyy useita vaihtoehtoja paneelin ulkoasulle ja alapuolella määritellään käytetty datalähde, sekä tietokantaan tehtävä kysely. Kuvio 22 nähtävä paneeli on määritelty näyttämään OBDII-adapterin antamaa auton moottorin kierroslukua graafina.



Kuvio 22 Moottorin kierrosluku paneeli

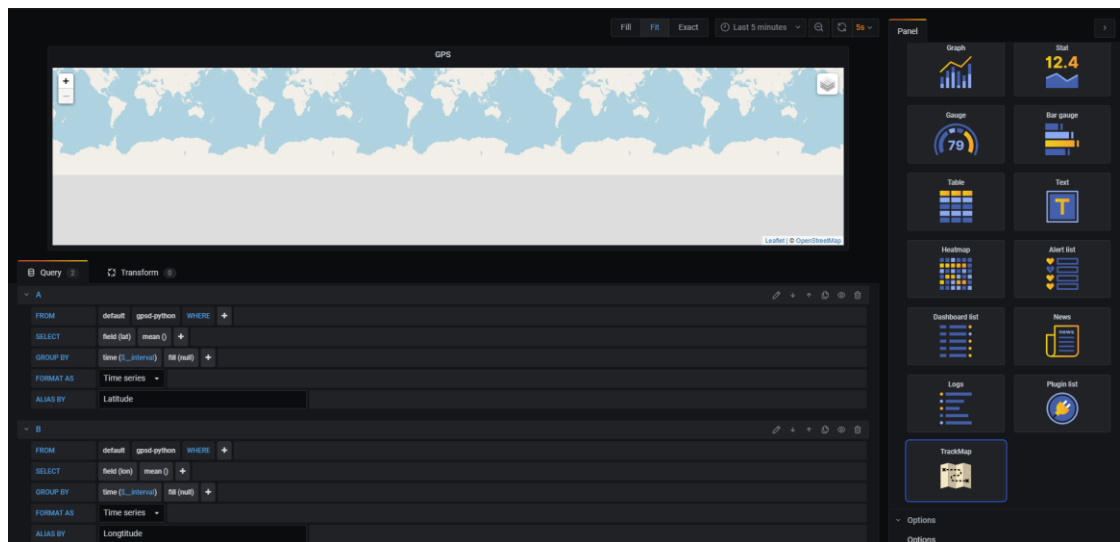
5.3.5 Trackmap liitännäisen lisääminen Grafanaan

Työtä suunnitellessa haluttiin, että kerättävä GPS-data olisi mahdollista esittää kartalla. Tähän tarkoitukseen Grafanan verkkosivuilta löytyvä TrackMap-liitännäinen oli täydellinen ratkaisu. TrackMap mahdollistaa leveys- ja pituus-koordinaattien piirtämisen karttataustalle yhtenäisenä viivana. Kartta taustoiksi on valittavissa kolme vaihtoehtoa: OpenStreetMap, joka näyttää tiet ja niiden nimet. OpenTopo-Map, joka näyttää kaupunkien nimet ja maaston korkeuserot. Satellite imagery, joka on sateelliitti näkymä maasta. Parhaiten työn tarkoituksiin sopi OpenStreetMap, joka muistuttaa perinteistä autonavigaattoria.

Liitännäisen asentaminen tapahtuu Grafanan terminaalin kautta komennolla:

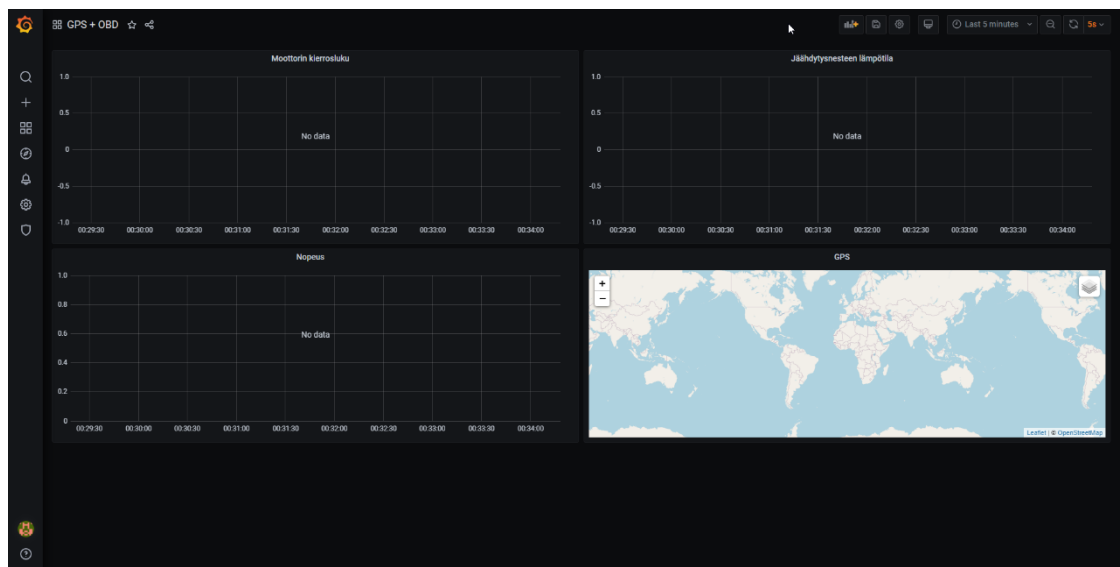
```
$ grafana-cli plugins install proOps-trackmap-panel
```

Asennuksen jälkeen liitännäisen pystyy lisäämään normaalisti paneeliksi muiden paneelien lailla. Paneeli tarvitsee toimiakseen GPS:n leveys ja pituus arvot, jotka saadaan tekemällä samalle paneelille kaksi eri tietokanta kyselyä Kuvio 23 mukaisesti.



Kuvio 23 TrackMap-paneeli

Helposti muokattavien paaneelien avulla voidaan luoda haluttuja tietoja näyttävä kokonaisuus, josta voi nähdä kaiken tarpeellisen kerralla. Kuvio 24 nähdään lopputulos, josta voidaan nähdä karttanäkymä auton sijainnista sekä autosta saatua OBDII-dataa graafina.



Kuvio 24 Valmiit Grafana paneelit

5.4 GPS-datan lähettäminen tietokantaan

Datan lähettämiseen käytetään mzac:in luomaa "gpsd-influx.py" Python-skriptiä, jota hieman muokataan omiin tarpeisiin sopivammaksi. (Ks. Liite 1.)

Luodaan python-tiedosto skriptiä varten komennolla:

```
$ sudo nano gpsd-influx-python.py
```

Tämän jälkeen kopioidaan python koodi tiedostoon ja tehdään samalla tarvittavat muutokset. Muutoksia täytyy tehdä kohtaan jossa määritellään muuttujat tietokantaan yhdistämistä varten. Influx_host riville syötetään tietokannan käyttämä julkinen IP-osoite, influx_port rivillä määritellään käytettävä portti, influx_user ja influx_pass riveillä määritetään aiemmin luodun InfluxDB-käyttäjän tunnus ja salasana, influx_db rivillä määritellään käytettävä tietokanta. Viimeiseksi voidaan määritellä haluttu päivitysväli update_interval rivillä. Tehdyt muutokset näkyvät Kuvio 25.

```
# Your InfluxDB Settings
influx_host = "195.148.26.236"
influx_port = 80
influx_user = "admin"
influx_pass = "salasana"
influx_db = "gpsd"

# Number of seconds between updates
update_interval = 5
```

Kuvio 25 gpsd-influx-python.py konfiguraatio

Tallennetaan tiedosto ja testataan sen toimivuutta ajamalla skripti käyttäen -d debug-optiota:

```
$ python gpsd-influx-python.py -d
```

Jos skripti toimii oikein, pitäisi sen antaa GPS:n sijainnin arvoja. Varmistetaan myös, että data lähetetään onnistuneesti tekemällä kysely gpsd-tietokantaan, mikä antaa viimeisimmän sinne kirjoitetun arvon. (Ks. Kuvio 26).

```
pi@raspberrypi:~ $ influx -host 195.148.26.236 -port 80
Connected to http://195.148.26.236 version 1.8.0
InfluxDB shell version: 1.8.1
> auth
username: admin
password:
> use gpsd
Using database gpsd
> SELECT last("lat") FROM "gpsd-python"
name: gpsd-python
time                last
----
1596041307605431906 62.242098985
> █
```

Kuvio 26 GPS-datan kysely tietokannasta

Kun skripti toimii normaalisti, voidaan tehdä siitä palvelu, joka aloittaa datan lähettämisen Raspberry Pi:n käynnistyessä, sekä käynnistyy uudelleen jos prosessi jostain syystä katkeaa, käyttämällä mzac:in Github-sivulta löytyvää konfiguraatiota.

Luodaan uusi service-tiedosto komennolla:

```
$ sudo nano /etc/systemd/system/gpsd-python.service
```

Kopioidaan tiedostoon Kuvio 27 näkyvä konfiguraatio.

```
[Unit]
Description=GPSD to Influx
After=syslog.target

[Service]
ExecStart=/opt/gpsd-influx/gpsd-influx.sh
KillMode=process
Restart=on-failure
User=root

[Install]
WantedBy=multi-user.target
```

Kuvio 27 gpsd-python.service konfiguraatio

Tallennetaan tiedosto ja otetaan palvelu käyttöön seuraavilla komennoilla:

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl enable gpsd-python.service
```

```
$ sudo systemctl start gpsd-python.service
```

Nyt palvelun pitäisi lähettää jatkuvasti GPS-dataa gpsd-tietokantaan.

5.5 Bluetooth yhteyden muodostaminen OBDII-lukijaan

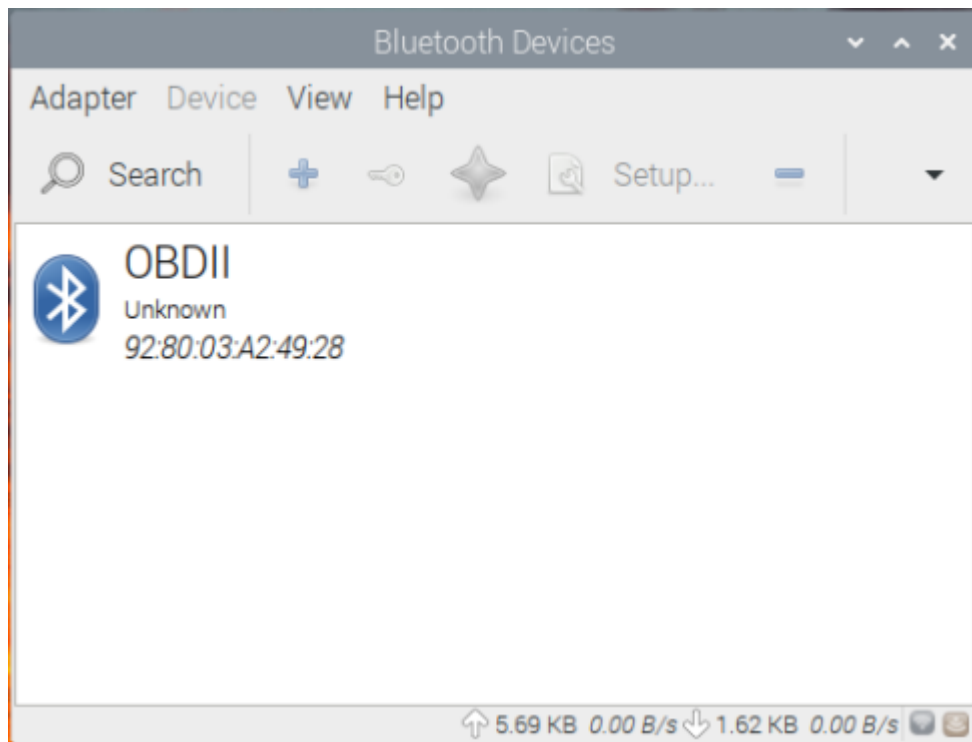
Yhteyden muodostamista varten on aluksi kiinnitettävä OBDII-adapteri autosta löytyvään OBDII-porttiin. Portti sijaitsee yleensä ohjauspyörän alapuolella tai keskikonsolissa. Tässä työssä käytetty auto on vuoden 2002 Citroen C3, jonka portti sijaitsee ohjauspyörän alapuolella olevan paneelin takana. (Ks. Kuvio 28).



Kuvio 28 OBDII-portin sijainti

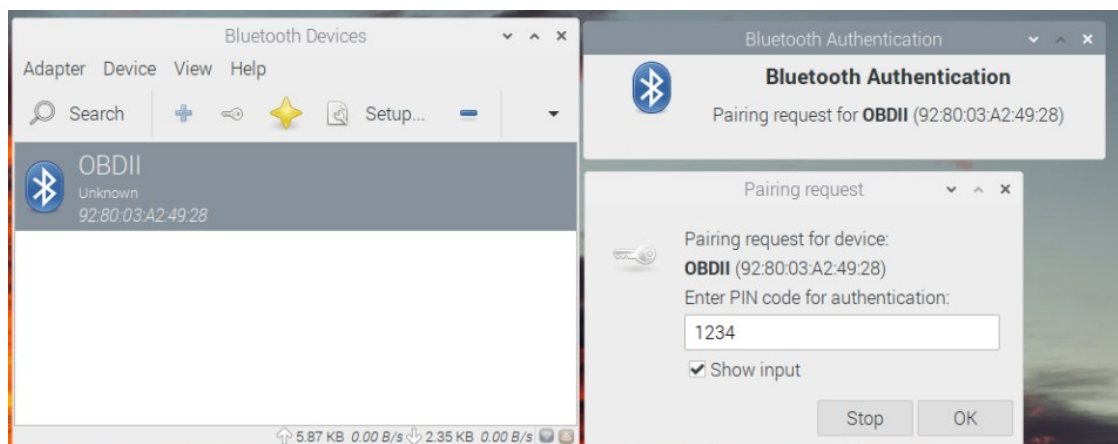
Kun lukija kiinnitetään porttiin, alkaa se vilkuttaa punaista valoa, mikä merkitsee, että adapteri saa virtaa ja on valmis käytettäväksi.

Adapteriin yhdistäminen tapahtuu Raspberry Pi:n Bluetooth manager ohjelmalla. Search-näppäintä painamalla aloitetaan lähellä olevien Bluetooth laitteiden etsintä. Laitteen löytämiseksi tulee sen olla noin 10 metrin etäisyydellä. Löydetyt laitteet tulevat näkyviin managerin laitelistaan Kuvio 29 mukaisesti.



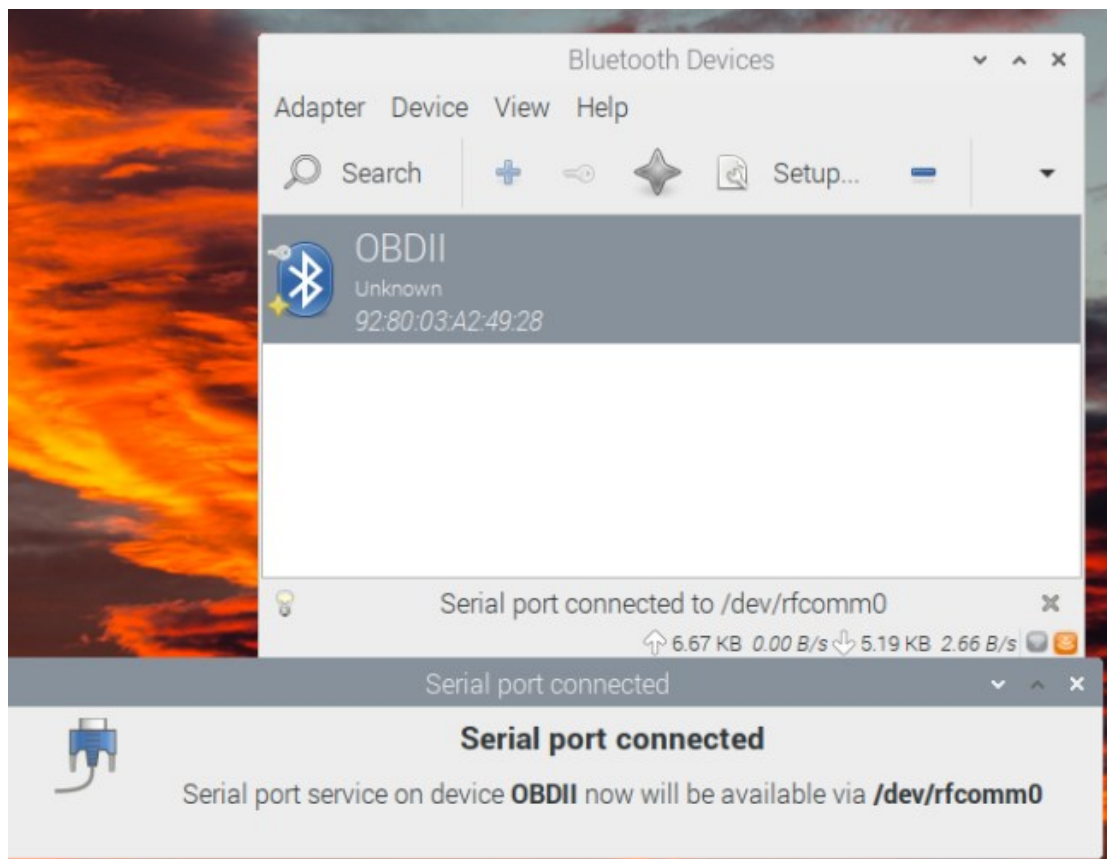
Kuvio 29 Bluetooth manager

Laitteen parittaminen voidaan tehdä painamalla löydettyä laitetta hiiren oikealla näppäimellä ja valitsemalla "Pair". Tämä avaa ruudun jossa pyydetään parittamiseen vaadittavaa salasanaa. Salasana on yleensä OBDII-adapttereissa vakiona "1234". (Ks. Kuvio 30). Sen jälkeen valitaan samasta valikosta Trust, joka merkitsee laitteen luotetuksi.



Kuvio 30 Bluetooth-laitteen parittaminen

OBDII-datan lähettämistä varten yhdistetään paritettu laite virtuaaliseen serial-porttiin painamalla paritettua laitetta hiiren oikealla näppäimellä ja valitsemalla "Connect to: Serial port". Laitteen pitäisi nyt olla yhdistettynä serial-portin kautta `/dev/rfcomm0`-tiedostoon Kuvio 31 mukaisesti.



Kuvio 31 Serial port connected to `/dev/rfcomm0`

5.6 OBDII-datan lukeminen ja lähettäminen tietokantaan

OBDII-datan lukemista varten löytyi valmiiksi muutamia erilaisia skriptejä, mutta ne eivät testatessa toimineet tai sopineet haluttuun käyttötarkoitukseen. Tämän seurauksena kirjoitettiin pythonilla skripti Python-OB:n sivuilta löytyviä ohjeita ja esimerkkejä seuraten. Python osaamisen vähäisyyden vuoksi tämä osoittautui melko haastavaksi ja aikaa vieväksi, mutta lopulta syntyi toimiva koodi, mikä suoritettaessa kerää OBDII-adapterilta haluttua dataa ja lähettää sen tietokantaan. (Ks. Liite 2.)

5.7 Järjestelmän toiminnan testaus

OBDII-datan lukemiseksi on auton virran oltava päällä tai auton on oltava käynnissä. Auton 12V liitännän kautta saatava virta vaihtelee esimerkiksi auton moottorin kierosnopeuden mukaan ja saattaa katketa kokonaan esimerkiksi auton sammuesssa, on se epävarma vaihtoehto Raspberry Pi:n virtalähteeksi. Tämän vuoksi hankittiin erillinen 10000mAh virtapankki, joka toimii Raspberry Pi:n virtalähteenä. Alla olevassa Kuvio 32 on käytetty Cellularline FreePower Slim 10000-virtapankki.

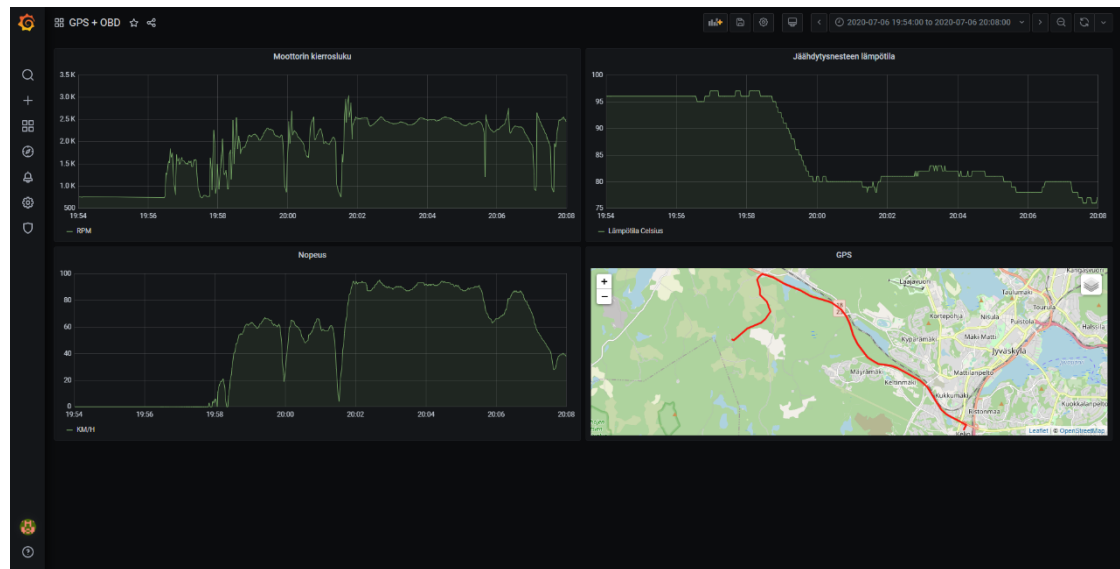


Kuvio 32 Raspberry Pi:n käyttämä virtalähde

Datan lähettämiseksi palvelimelle auton kyydistä, yhdistetään Raspberry Pi ajon ajaksi matkapuhelimesta jaettuun 4G-verkkoon. GPS-datan lähettäminen alkaa automaattisesti Raspberry Pi:n käynnistyessä. Tämän jälkeen otetaan Bluetooth yhteys OBDII-adapteriin ja käynnistetään tehty obd-influx-skripti, joka alkaa lähettää dataa tietokantaan. Tämä tapahtuu komennolla:

\$ python obd-influx.py

Testiajoksi tehtiin automatka Ruokkeelta Keljonkeskuksen S-Marketille. Testiajon jälkeen oli mahdollista tarkastella matkan aikana kerättyjä tietoja Grafanasta. Alla olevassa Kuvio 33 voidaan nähdä ajettu reitti kartalla, sekä auton nopeus, moottorin kierroslukumäärä ja jäähdytysnesteen lämpötila.



Kuvio 33 Testiajon tuloksia

6 Pohdinta

Työn tavoitteena oli rakentaa laite joka mahdollistaisi auton käyttäytymisen ja sijainnin seuraamisen reaaliajassa. Työtä aloitettaessa ei aiheesta ollut aikaisempaa kokemusta, pientä Python ja Linux osaamista lukuunottamatta, joten suunnitteluun kului runsaasti aikaa. Matkalla tuli vastaan useita ongelmia, jotka vaativat paljon aikaa ja kärsivällisyyttä. Suurin osa kohdatuista ongelmista saatiin ratkaistua, mutta vastaan tuli myös ongelmia, joihin ei saatu ratkaisua, mutta löydettiin kuitenkin kiertotie. Työn lopputulos toimii halutulla tavalla, mutta siitä löytyy kuitenkin asiota, jotka olisi parempi tehdä eri tavalla kuin alun perin suunniteltu.

Opinnäytetyötä tehdessä jäi itselle käteen paljon uutta tietoa asioista, joista en ennen ollut edes kuullut, minkä lisäksi se auttoi myös vahvistamaan osaamista jo aiemmin opituista asioista. Jos lähtisin tekemään työtä uudestaan, on työssä joitakin asioita joihin tekisin muutoksia, mutta osaa niistä ei voi muuttaa kuin ajan kanssa lisää oppimalla.

6.1 Kohdatut ongelmat ja haasteet

Työtä tehdessä tuli vastaan useita aikaa vieviä ongelmia, joista suurin oli varmaankin Bluetooth yhteys OBDII-adapterin ja Raspberry Pi:n välillä. Työtä tehdessä luin mahdollisuudesta yhdistää Raspberry Pi sen käynnistyessä automaattisesti OBDII-adapteriin. Tämän toteutus olisi tapahtunut koodilla, joka ajettaisiin laitteen käynnistyessä ja yhdistäisi sen automaattisesti käyttäen komentolinjaa. Tämä idea täytyi kuitenkin loppujen lopuksi jättää tekemättä, koska useiden päivien yrittämisen ja tutkimisen jälkeen ei Bluetoothin yhdistäminen komentolinjaa käyttäen onnistunut. Ongelmaa tutkiessani huomasin, että en suinkaan ollut ainoa joka tästä ongelmasta kärsi, vaan se vaikutti olevan melko yleinen. Ongelman ratkaisuun löytyi paljon ehdotuksia, joista osa toimi joillakin käyttäjistä, mutta enimmäkseen tie näytti umpikujalta. Tämän seuraksena täytyi ottaa kiertotie ja yhdistää Bluetooth joka kerta ennen käyttöä manuaalisesti työpöytäohjelman kautta. Tämä ei kuitenkaan ollut yhtä hyvä tapa tehdä sitä, mutta oli se toimiva vaihtoehto. Loppujen lopuksi epäselväksi jäi onko ongelman alkuperä itse Raspberry Pi:ssä vai siinä olevassa Bluetooth-ohjelmistossa.

Toinen haaste työssä muodostui OBDII-datan lukemisesta ja lähettämisestä. Internetistä löytyi jonkin verran valmiita ratkaisuja datan keräämiseen, mutta ne vain suureksi osaksi keräsivät dataa paikallisesti, mikä ei työn tarkoitukseen ollut sopiva ratkaisu. Tämän takia täytyi miettiä vaihtoehtona oman koodin kirjoittamista haluttua tarkoitusta varten, mikä vähän koodaus osaamisen takia osoittautui melko haastavaksi ja aikaa vieväksi. Esimerkkejä seuraamalla sekä yrityksen ja erehdyksen kautta syntyi lopulta jotakin toimivaa. Lopputulos oli varmaankin melko yksinkertaista ja kankean näköistä, mutta se toimi, mikä oli pääasia. Enemmällä kokemuksella olisi koodista saanut varmasti järkevämmän näköistä. Mielestäni isoin

ongelma tehdyssä koodissa on se, että jos autosta halutaan ulos jotain tiettyä dataa, on sitä varten lisättävä koodiin vaadittavat rivit. Tämän lisäksi jos auto ei tue kaikkia koodissa olevia kyselyitä, mikä on melko yleistä vanhemmissa autoissa, ei koodikaan silloin toimi.

6.2 Mahdollisia parannus ehdotuksia

Työtä tehdessä tuli mieleen joitakin asioita joita olisi voinut tehdä toisin tai olisivat voineet tehdä laitteesta paremman. Ensimmäisenä tulee mieleen OBDII-adapterin vaihtaminen Bluetoothista langalliseksi. Ensinnäkin tämä olisi vähentänyt työssä esiintyneiden ongelmien määrää ja niiden viemää aikaa. Toiseksi, vaikka langaton yhteys säästää tilaa, on se myös tietoturvan kannalta huonompi vaihtoehto. Kun adapteri on autossa kiinni, voi kuka tahansa käyttää sitä halutessaan, olettaen, että he tietävät laitteen oletus salasanan 1234.

Laitetta varten olisi myös voinut hankkia erillisen 4G-mokkulan. Tämä poistaisi vaatimuksen käyttää puhelinta tukiasemana, mikä tekisi käytöstä helpompaa ja eikä eri langattomien verkkojen välillä hyppely olisi tarpeellista.

Viimeiseksi, vaikka ei välttämättä tarpeen, oltaisiin laitteessa voitu käyttää GPS-moduulia USB version sijaan. Tämä vaatisi enemmän työtä, mutta säästäisi jonkin verran tilaa. Tämän työn tapauksessa en pidä tätä muutosta tarpeellisena, mutta jos laitetta lähtisi kehittämään eteenpäin kokonaisemmaksi paketiksi, olisi se varmaankin oikea vaihtoehto.

Lähteet

About gpsd 2020. Gitlab sivusto. Viitattu 25.07.2020.

<https://gpsd.gitlab.io/gpsd/index.html>

Alert notifications N.d. Grafana verkkosivusto. Viitattu 13.07.2020.

<https://grafana.com/docs/grafana/latest/alerting/notifications/>

Esittely N.d. Ubuntu Suomi verkkosivusto. Viitattu 14.07.2020.

<https://www.ubuntu-fi.org/esittely/>

FAQs N.d. Raspberry Pi:n verkkosivusto. Viitattu 10.05.2020.

<https://www.raspberrypi.org/documentation/faqs/>

Getting started N.d. python-OBd verkkosivusto. Viitattu 25.07.2020.

<https://python-obd.readthedocs.io/en/latest/>

Install InfluxDB OSS N.d. InfluxDatan verkkosivusto. Viitattu 13.07.2020.

<https://docs.influxdata.com/influxdb/v1.8/introduction/install/>

InfluxDB 1.8 documentation N.d. InfluxDatan verkkosivusto. Viitattu 13.07.2020.

<https://docs.influxdata.com/influxdb/v1.8/>

Kili, A. 2018. Grafana – An Open Source Software for Analytics and Monitoring.

TecMint verkkosivusto. Viitattu 13.07.2020.

<https://www.tecmint.com/install-grafana-analytics-in-centos-ubuntu-debian/>

Mai, T. 2017. Global Positioning System History. NASA:n verkkosivustolla. Viitattu 15.05.2020.

https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

mzac 2020. gpsd-influx.py-skripti. Github verkkosivusto. Julkaistu 11.05.2020. Viitattu 16.07.2020.

<https://github.com/mzac/gpsd-influx/blob/master/gpsd-influx.py>

nginx N.d. nginx verkkosivusto. Viitattu 25.07.2020.

<https://nginx.org/en/>

OBd2 Codes and Meanings 2020. Bads verkkosivusto. Viitattu 17.08.2020

<https://bads.lt/en/obd2-codes-and-meanings-2/>

OBd2 Explained - A Simple Intro (2020) 2020. CSS Electronics sivustolla. Viitattu 13.05.2020.

<https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en>

On-Board Diagnostics (OBD) Background History N.d. OBD Innovations sivustolla. Viitattu 13.05.2020.

<https://www.obdinnovations.com/on-board-diagnostics-obd-background-history/>

Piltch, A. 2020. Raspberry Pi OS: Why It's No Longer Called 'Raspbian'. Tom's Hardware verkkosivusto. Viitattu 14.07.2020.

<https://www.tomshardware.com/news/raspberry-pi-os-no-longer-raspbian>

Raspberry Pi OS (previously called Raspbian) 2020. Raspberry Pi:n verkkosivusto. Viitattu 14.07.2020.

<https://www.raspberrypi.org/downloads/raspberry-pi-os/>

RPi General History 2015. Embedded Linux Wikin verkkosivustolla. Viitattu 10.05.2020.

https://elinux.org/RPi_General_History

Space Segment 2020. GPS: The Global Positioning System verkkosivusto. Viitattu 15.05.2020.

<https://www.gps.gov/systems/gps/space/>

Time series database (TSDB) explained 2020. InfluxDatan verkkosivusto. Viitattu 13.07.2020.

<https://www.influxdata.com/time-series-database/>

OBD-II PIDs 2010. OBD-II Resource verkkosivusto. Viitattu 17.08.2020

<http://obdcon.sourceforge.net/2010/06/obd-ii-pids/>

What is a Raspberry Pi? N.d. Raspberry Pi:n verkkosivusto. Viitattu 10.05.2020.

<https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>

What is GPS? N.d. Garmin verkkosivusto. Viitattu 15.05.2020.

<https://www.garmin.com/en-US/AboutGPS/>

What is Linux? N.d. Opensource verkkosivusto. Viitattu 14.07.2020.

<https://opensource.com/resources/linux>

Liitteet

Liite 1. gpsd-influx-python.py

```
#!/usr/bin/python
```

```
from gps import *
```

```
from influxdb import InfluxDBClient
```

```
from time import *
```

```
import getopt
```

```
import os
```

```
import socket
```

```
import sys
```

```
import threading
```

```
import time
```

```
# Your InfluxDB Settings
```

```
influx_host = 'influx.lab.local'
```

```
influx_port = 8086
```

```
influx_user = None
```

```
influx_pass = None
```

```
influx_db = 'gpsd'
```

```
# Number of seconds between updates
```

```
update_interval = 10
```

```
# -----
```

```
# Do not change anything below this line
```

```
hostname = socket.gethostname()
```

```
# -----
```

```
# Command Line Options
```

```
options, remainder = getopt.gnu_getopt(
```

```
    sys.argv[1:], 'd', ['debug'])
```

```
debug = None
```

```
for opt, arg in options:
```

```
    if opt in ('-d', '--debug'):
```

```
        debug = True
```

```
# -----
```

```
# GPS Thread
```

```
class GpsPoller(threading.Thread):
```

```
    def __init__(self):
```

```
        threading.Thread.__init__(self)
```

```
        global gpsd
```

```
        gpsd = gps(mode=WATCH_ENABLE|WATCH_NEWSTYLE)
```

```
        self.current_value = None
```

```
        self.running = True
```

```
    def run(self):
```

```
global gpsd
```

```
while gpsp.running:
```

```
    gpsd.next()
```

```
# -----
```

```
# GPS Loop
```

```
if __name__ == '__main__':
```

```
    # Create the thread
```

```
    gpsp = GpsPoller()
```

```
    try:
```

```
        # Start up the thread
```

```
        gpsp.start()
```

```
        # Sleep for 5 seconds to allow the gps to pick up the position
```

```
        time.sleep(5)
```

```
    # Start the loop
```

```
while True:
```

```
    gpsd_alt = gpsd.fix.altitude
```

```
    gpsd_climb = gpsd.fix.climb
```

```
    gpsd_epc = gpsd.fix.epc
```

```
    gpsd_eps = gpsd.fix.eps
```

```
    gpsd_ept = gpsd.fix.ept
```

```
    gpsd_epv = gpsd.fix.epv
```

```
    gpsd_epx = gpsd.fix.epx
```

```
    gpsd_epy = gpsd.fix.epy
```

```
    gpsd_lat = gpsd.fix.latitude
```

```
    gpsd_lon = gpsd.fix.longitude
```

```
    gpsd_mode = gpsd.fix.mode
```

```
    gpsd_speed = gpsd.fix.speed
```

```
    gpsd_track = gpsd.fix.track
```

```
    # Make sure we have a lat, lon and alt
```

```
    if None not in (gpsd_lat, gpsd_lon, gpsd_alt,):
```

```
if debug == True:
```

```
    print "gpsd-python,host=",hostname,",tpv=alt value=",gpsd_alt
```

```
    print "gpsd-python,host=",hostname,",tpv=climb value=",gpsd_climb
```

```
    print "gpsd-python,host=",hostname,",tpv=epc value=",gpsd_epc
```

```
    print "gpsd-python,host=",hostname,",tpv=eps value=",gpsd_eps
```

```
    print "gpsd-python,host=",hostname,",tpv=ept value=",gpsd_ept
```

```
    print "gpsd-python,host=",hostname,",tpv=epv value=",gpsd_epv
```

```
    print "gpsd-python,host=",hostname,",tpv=epx value=",gpsd_epx
```

```
    print "gpsd-python,host=",hostname,",tpv=epy value=",gpsd_epy
```

```
    print "gpsd-python,host=",hostname,",tpv=lat value=",gpsd_lat
```

```
    print "gpsd-python,host=",hostname,",tpv=lon value=",gpsd_lon
```

```
    print "gpsd-python,host=",hostname,",tpv=mode value=",gpsd_mode
```

```
    print "gpsd-python,host=",hostname,",tpv=speed value=",gpsd_speed
```

```
    print "gpsd-python,host=",hostname,",tpv=track value=",gpsd_track
```

```
influx_json_body = [
```

```
{
```

```
"measurement": "gpsd-python",
```

```
"tags": {
```

```
    "host": hostname
```

```
},
```

```
"fields": {
```

```
    "alt": gpsd_alt,
```

```
    "climb": gpsd_climb,
```

```
    "epc": gpsd_epc,
```

```
    "eps": gpsd_eps,
```

```
    "ept": gpsd_ept,
```

```
    "epv": gpsd_epv,
```

```
    "epx": gpsd_epx,
```

```
    "epy": gpsd_epy,
```

```
    "lat": gpsd_lat,
```

```
    "lon": gpsd_lon,
```

```
    "mode": gpsd_mode,
```

```
    "speed": gpsd_speed,
```

```
        "track": gpsd_track

    }

}

]

influx_client = InfluxDBClient(influx_host, influx_port, influx_user, influx_pass, influx_db)

influx_client.write_points(influx_json_body)

time.sleep(update_interval)

except (KeyboardInterrupt, SystemExit): #when you press ctrl+c

    print "\nKilling Thread..."

    gpsp.running = False

    gpsp.join() # wait for the thread to finish what it's doing

    print "Done.\nExiting."
```


Liite 2. obd-influx.py

```
import sys
```

```
import obd
```

```
import datetime
```

```
from influxdb import InfluxDBClient
```

```
from influxdb.line_protocol import _get_unicode, quote_ident, _is_float, text_type
```

```
# Influxdb tietojen määrittely
```

```
host = "IP-osoite"
```

```
port = 80
```

```
user = "Käyttäjätunnus"
```

```
password = "Salasana"
```

```
dbname = "Tietokanta"
```

```
# Tietojen lähettäminen Influxiin
```

```
client = InfluxDBClient(host, port, user, password, dbname)
```

```
# Autoon yhdistäminen
```

```
connection = obd.OBD("/dev/rfcomm0")
```

```
measurement = "OBDII"
```

```
# While-loop tietojen jatkuvaan kyselyyn
```

```
while True:
```

```
    coolant_temp = obd.commands.COOLANT_TEMP
```

```
    rpms = obd.commands.RPM
```

```
    speed = obd.commands.SPEED
```

```
    cmd1 = connection.query(rpms)
```

```
    cmd2 = connection.query(speed)
```

```
cmd4 = connection.query(coolant_temp)
```

```
#JSON tiedosto, johon kerätään autolta saadut arvot
```

```
data = [
```

```
{
```

```
    "measurement": measurement,
```

```
    "fields": {
```

```
        "RPM": cmd1.value.magnitude,
```

```
        "SPEED": cmd2.value.magnitude,
```

```
        "COOLANT TEMP": cmd4.value.magnitude
```

```
    }
```

```
}
```

```
]
```

```
time.sleep(0,5)
```

```
#Lähetetään data influxiin
```

```
client.write_points(data)
```