



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

PANU AHO 266300  
SULAUTETUT JÄRJESTELMÄT PLA-32310

Oppimispäiväkirja

Tarkastaja: Mika Saari

## SISÄLLYSLUETTELO

1.	JOHDANTO .....	1
2.	KÄYTTÖJÄRJESTELMÄT .....	2
3.	OHJELMOINTI .....	13
4.	ARDUINO .....	18
5.	RASPBERRY PI .....	24
6.	IOT .....	30
7.	RAKENNUSPROJEKTI .....	33
	LÄHTEET .....	46

### LIITE 1: TUNTIKIRJANPITO

.

# 1. JOHDANTO

Oppimispäiväkirja on tehty syksyllä 2017 TTY:n Porin laitoksen kurssille Sulautetut Järjestelmät (PLA-32310). Päiväkirjassa on tarkoitus käsitellä kurssin eri aihealueet sekä millaisia lähestymistapoja olen henkilökohtaisesti käyttänyt sisäistääkseni kurssilla käsiteltävät asiat. Haasteita kurssin suoritukseen tuo suoritustapani, joka on etäpainotteinen itsenäinen työskentely, mutta en usko tämän kuitenkaan kasvavan ylitsepääsemättömäksi ongelmaksi.

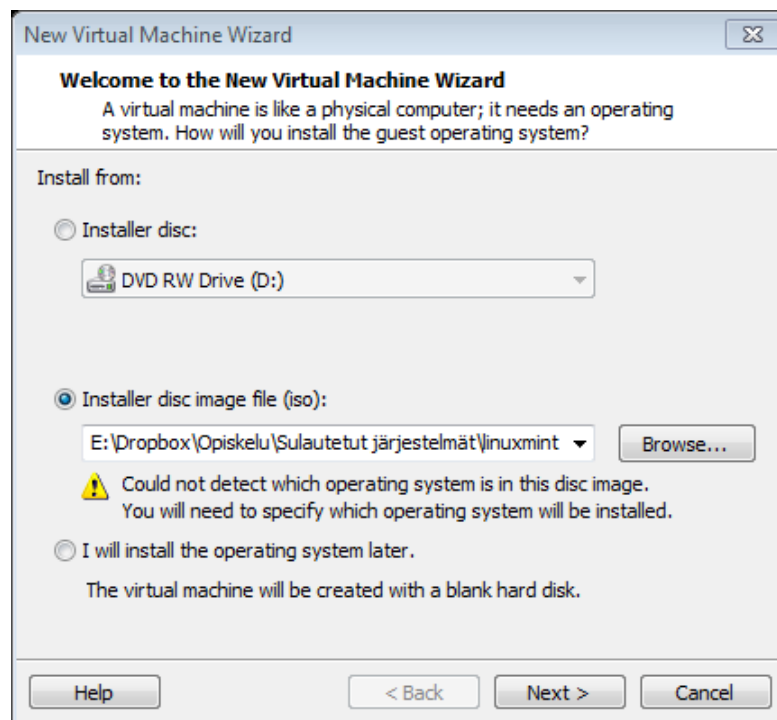
Oppimispäiväkirja on nimensä mukaisesti muodoltaan päiväkirjamainen, tarkoittaen sitä, että kerran kirjoitettuja merkintöjä ei jälkikäteen ole juurikaan palattu editoimaan. Tekstin koherenttius kärsinee tästä hieman, mutta etuna saavutetaan paras mahdollinen kuvaus nimenomaan oppimisprosessista, mihin sudenkuoppiin on pudottu ja miten niistä on (toivon mukaan) myös kavuttu ylös.

Teksti tulee päivittymään ja elämään kurssin aikana. Liitteessä 1 esitetään viiteellinen tuntikirjanpito.

## 2. KÄYTTÖJÄRJESTELMÄT

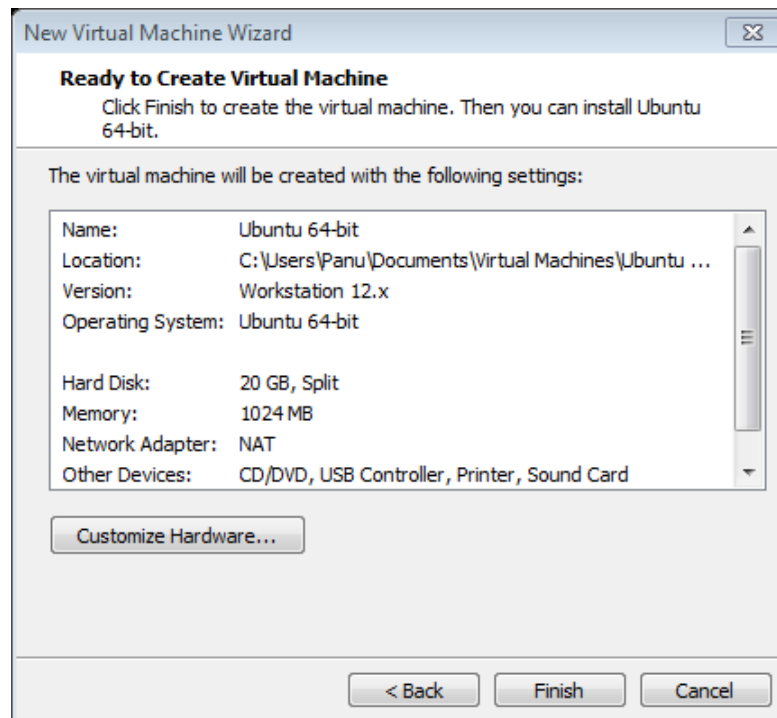
9.9.2017 – 10.9.2017

Tutustuttiin virtuaalikoneiden mahdollisuuksiin sekä Linux-käyttöjärjestelmän asennukseen sekä peruskäyttöön vMWare-virtualisointiympäristössä. Linux-distrokseksi valittiin Mint, jonka iso-levykuva ladattiin [www.linuxmint.com](http://www.linuxmint.com) sivustolta. VMwaresta asennettiin 'Workstation 12 Player'-versio, joka on ilmainen ei-kaupallisessa käytössä. Uusi virtuaalikone otettiin käyttöön 'New Virtual Machine Wizard'-velhotoiminnolla. Käyttöjärjestelmän kohdalle asetettiin em. ladattu ISO-image. Vaihtoehtona olisi ollut myös uuden virtuaalikoneen alustaminen tyhjällä kovalevyllä, mikäli oltaisiin haluttua asentaa käyttöjärjestelmä myöhemmin manuaalisesti.



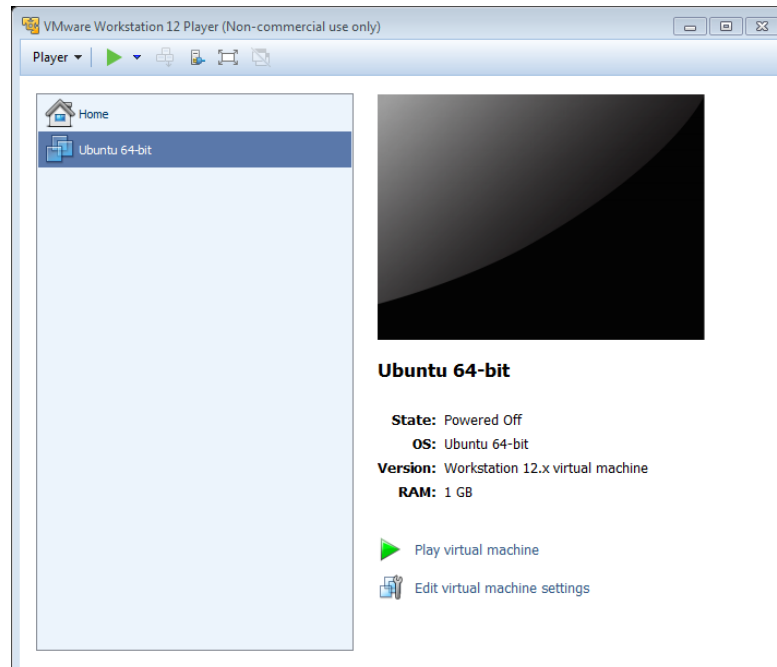
**Kuva 1. Virtuaalikoneen luontityökalu**

Seuraavana piti valita asennettavan Linuxin tyyppi. Listalla ei ollut valittavana Mintiä, mutta Mintin dokumentaatiosta löytyi tieto että se perustuu Ubuntu versioon 16.04. Niinpä tässä kohdassa valittiin Linux-distrokseksi Ubuntu 64-bit. Seuraavaksi oli paljon valintaikkunoita liittyen tallennuslokaatioihin sekä virtualisoitavan koneen kokoonpanoon, jota olisi myös halutessaan päässyt kustomoimaan. Nyt edettiin kuitenkin hyväksyen velhon ehdottamat oletusasetukset.



**Kuva 2. Uuden virtuaalikoneen kokoonpano**

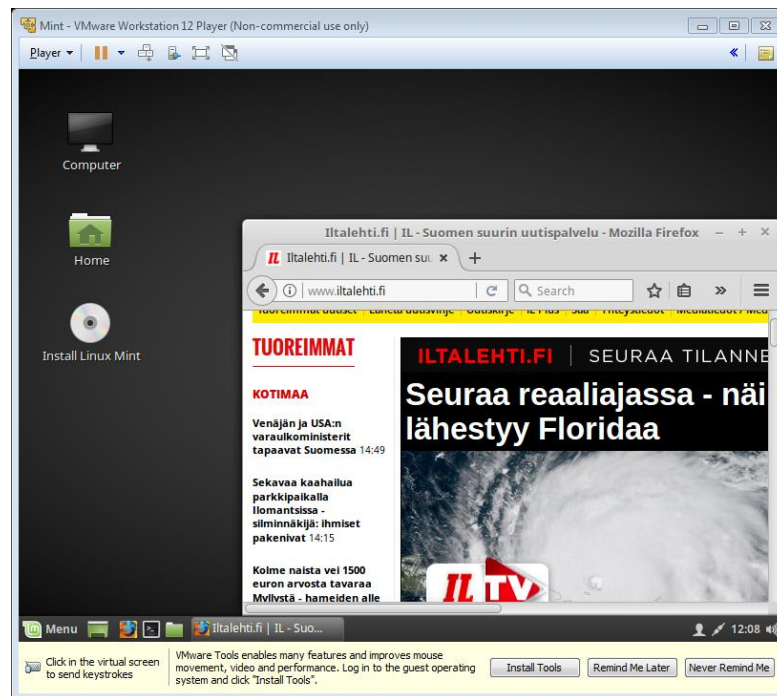
Näin oli saatu virtuaalikone luotua ja oli aika pistää kone käyntiin Play virtual machine-painikkeesta.



**Kuva 3. Juuri luotu virtuaalikone vMwaren home-näkymässä**

Ensimmäisellä käynnistysyrityksellä saatiin pitkä virheilmoitus, jossa mm. moitittiin käyttäjää siitä, että host-järjestelmä olisi pitänyt käynnistää uudelleen vmWaren asennuksen jälkeen (Asennuksen aikana tällaista vaatimusta ei tosin esitetty). Kun

uudelleenkäynnistyskään ei auttanut, ruvettiin tutkimaan tarkemmin virheilmoituksen anatomiaa. Pikaisen selvittelyn ja google-hakujen perusteella ilmeni, että ongelma koski Intelin Vt-x virtualisointitekniologiaa, joka syystä tai toisesta ei ollut isäntäkoneen Core i7-prosessorissa käytettävissä. Sen sijaan että oltaisi lähdetty säätämään hostin BIOS-asetuksia, ongelma ratkaistiin lataamalla ja asentamalla Mintistä 32-bittinen versio, joka pystyy toimimaan ilman Vt-x:ää. Tämän jälkeen virtuaalikone buuttasi ja verkkoyhteydetkin näyttivät saman tien pelaavan hienosti.



**Kuva 4. Mint pyörii komeasti virtuaalikoneessa.**

Tämän jälkeen oli aika hieman harjoitella Linuxin peruskäyttöä. Linuxissa ohjelmien asentaminen käy näppärästi suoraan komentoriviltä. Harjoitustehtävänä oli laatia Hello World- sovelluksia muutamilla eri ohjelmointikielillä, joten ensimmäinen mitä tarvittiin oli tietysti kunnollinen tekstieditori. Kun ensin oli saatu käyttöön suomalainen näppäimistökonfiguraatio (setxkbmap fi), virtuaaliympäristöön asennettiin notepadqq-ohjelmisto suorittamalla ohjelmiston kotisivuilta [1] saatujen ohjeiden mukaiset komennot:

```
sudo add-apt-repository ppa:notepadqq-team/notepadqq
sudo apt-get update
sudo apt-get install notepadqq
```

Kuvassa 5 on lisätty ja päivitetty tarvittavat repositoriot ja ollaan viimeistelemässä asennusta.

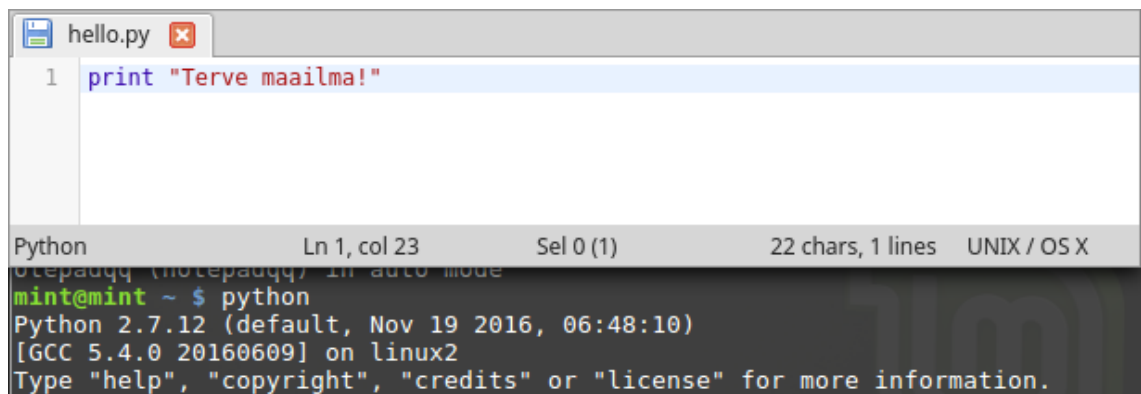
```

mint@mint ~ $ sudo apt-get install notepadqq
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  notepadqq-common
The following NEW packages will be installed:
  notepadqq notepadqq-common
0 upgraded, 2 newly installed, 0 to remove and 257 not upgraded.
Need to get 1,072 kB of archives.
After this operation, 5,676 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

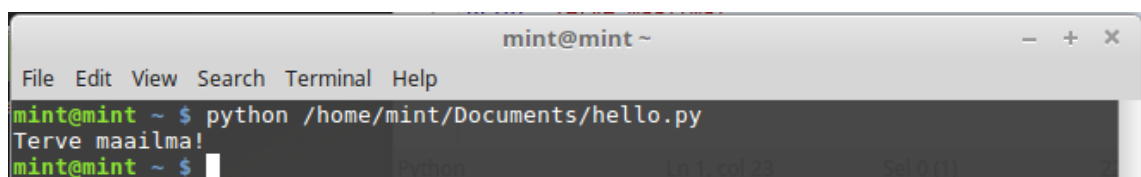
**Kuva 5. Tekstieditorin asennusta terminaalin kautta**

Nyt voitiin laatia juuri asennetulla tekstieditorilla yksinkertainen python-sovellus, joka tulostaa ruudulle tekstin ”Terve maailma!”. Koodi tallennettiin nimellä ’hello.py’ polkuun /home/mint/Documents. Terminaalikomennolla python saatiin selville, että järjestelmässä oli valmiina asennettuna Python version 2.7.12 eli sen asennuksesta ei tässä kohtaa tarvitsisi huolehtia.



**Kuva 6. Hello World-koodi tekstieditorissa sekä Pythonin versiotiedot Terminalissa**

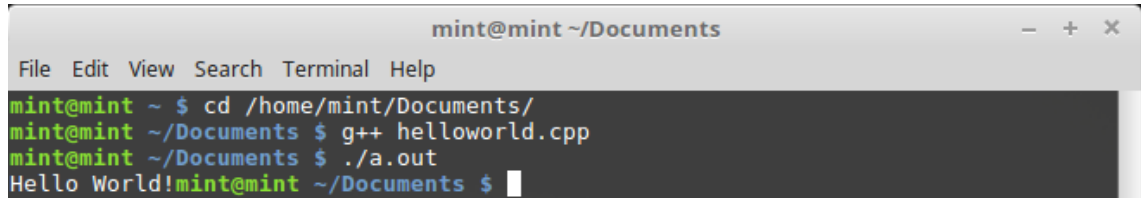
Koodi ajettiin komentoriviltä windows-ympäristöstäkin tutun oloisella python-syntaksilla:



**Kuva 7. Ensimmäinen python-sovellus Linux-ympäristössä näyttää toimivan!**

Yksi asia missä menttiin aluksi hieman vipuun oli virtuaalikoneen sammuttaminen. Jos sammutuksen tekee ’normaalilla’ tavalla käyttöjärjestelmän valikkojen kautta, on mitä ilmeisemmin seurauksena guest-järjestelmän kovalevyn tyhjentäminen. Parempi tapa on käyttää vmwaren suspend-toimintoa, jolloin seuraavalla käynnäytöksellä voidaan palata samaan tilaan kuin mihin viimeksi jäätin.

Seuraavaksi oli vuorossa C++ testisovelluksen laatiminen jota varten asennettiin g++ työkalupaketti komennolla `sudo apt install g++`. HelloWorld-testikoodi tallennettiin jälleen Documents-hakemistoon nimellä `helloworld.cpp`. Tämän jälkeen koodi voitiin kääntää ja suorittaa Terminalin kautta.



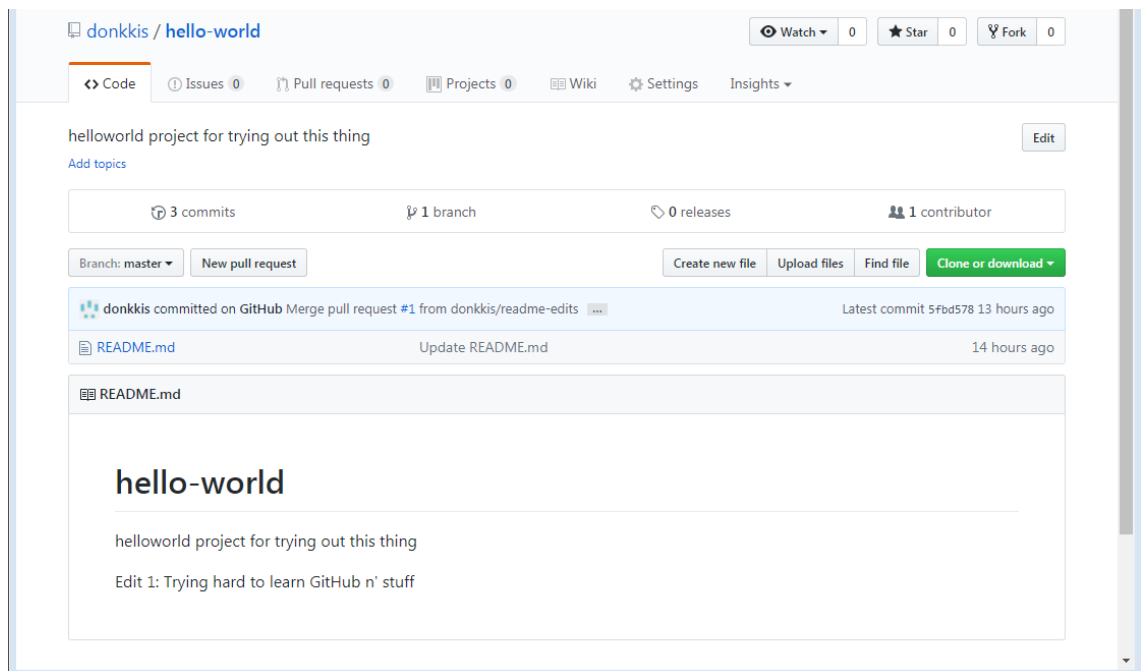
```

mint@mint ~/Documents
File Edit View Search Terminal Help
mint@mint ~ $ cd /home/mint/Documents/
mint@mint ~/Documents $ g++ helloworld.cpp
mint@mint ~/Documents $ ./a.out
Hello World!mint@mint ~/Documents $

```

**Kuva 8. C++ lähdekoodin kääntäminen g++:lla sekä ajaminen**

Myös GitHub oli uusi tuttavuus, joten siihen perehtyminen aloitettiin luomalla tunnukset [3] sekä harjoittelemalla GitHubin guides-sivulta löytyvä Hello World-tutoriaali, jossa tutustuttiin versionhallinnan perustoiminallisuuksiin selainkäyttöliittymän kautta [2]. Ja eipä aikaakaan, kun oli luotu ensimmäinen repositorio sekä harjoiteltu hieman branchien, committien sekä mergejen tekemistä.



**Kuva 9. GitHub-harjoittelua**

Seuraavaksi harjoiteltiin GitHub:ssa toimimista Terminalin kautta:

```
//Ensimmäisellä käyttökerralla pitää konfiguroida user-tiedot
git config --global user.email " " // sähköposti...
git config --global user.name "donkkis"
```

```
//sitten itse asiaan
git clone https://github.com/donkkis/hello-world
```

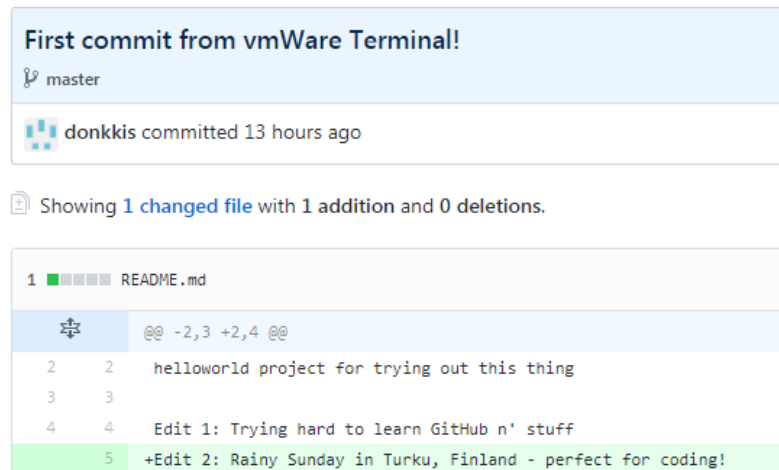
```
//Käytiin tekemässä muutoksia repon, esim. README-tiedoston sisältöön...
```



```
git commit -am "First commit from vmWare Terminal!"
git push -u origin master
```

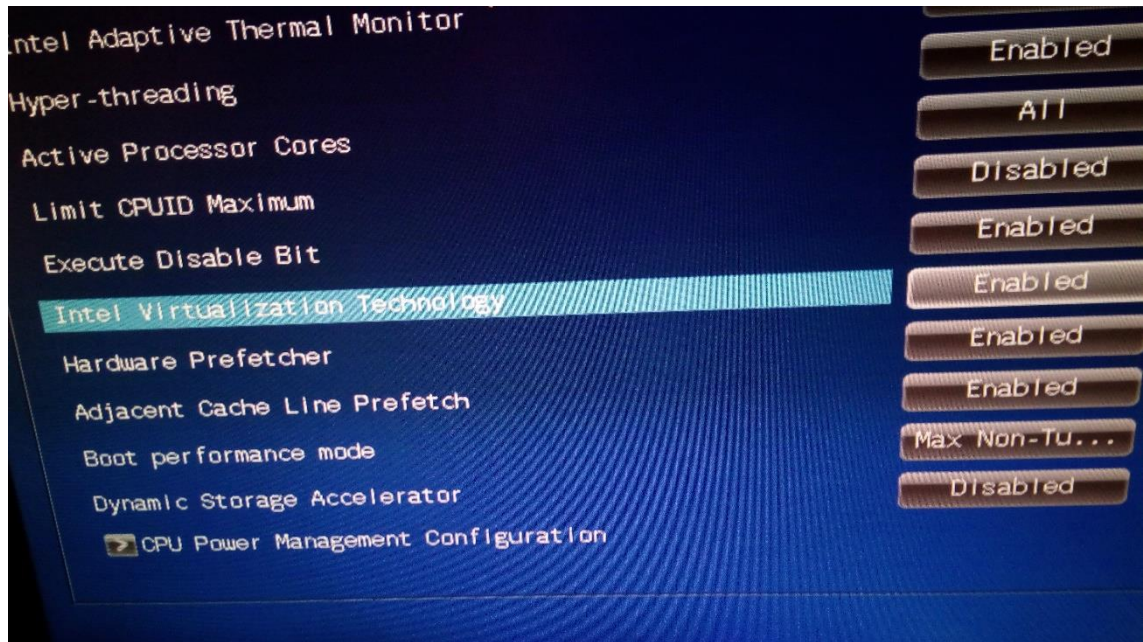
//Kysyy gh-tunnuksia, syötä käyttis ja passu kysyttäessä

Ja niinhän siinä kävi että tehdyt muutokset päivittyivät repositorioon. Vertailutoiminto web-näkymässä korostaa mukavasti commitissa tehdyt muutokset. Toisinaan saatetaan saada commit-vaiheessa virheilmoitus, ettei tiedostoja ole muutettu vaikka näin olisikin. Tällöin saattaa auttaa komento "git add .".



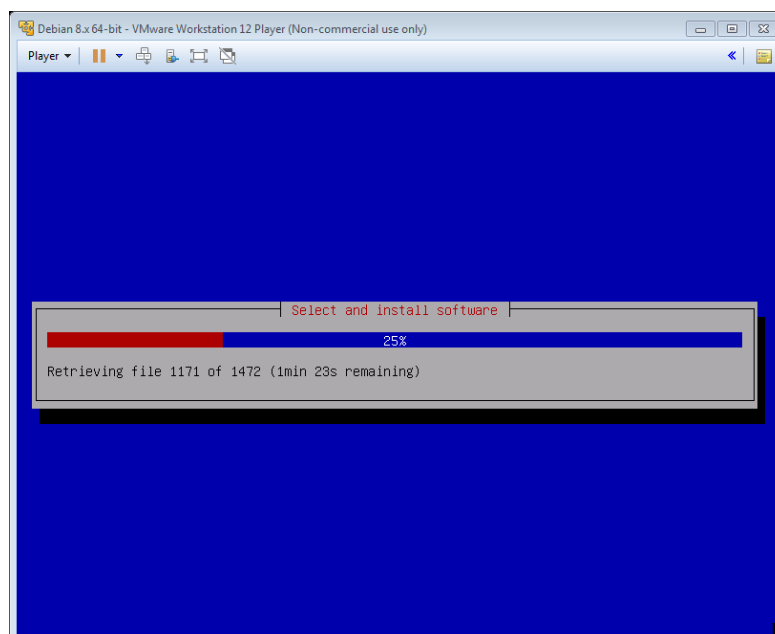
***Kuva 10. Terminalista tehty commit***

LAMP:n testaamista varten oli välttämätöntä saada 64-bittinen virtualisointiympäristö toimimaan, sillä kyseinen sovellus oli saatavilla vain 64-bittisenä. Suunnattiin siis tutkailemaan isäntäkoneen BIOS-asetuksia, mistä oikea täppä löytyikin lyhyen haeskelun jälkeen.



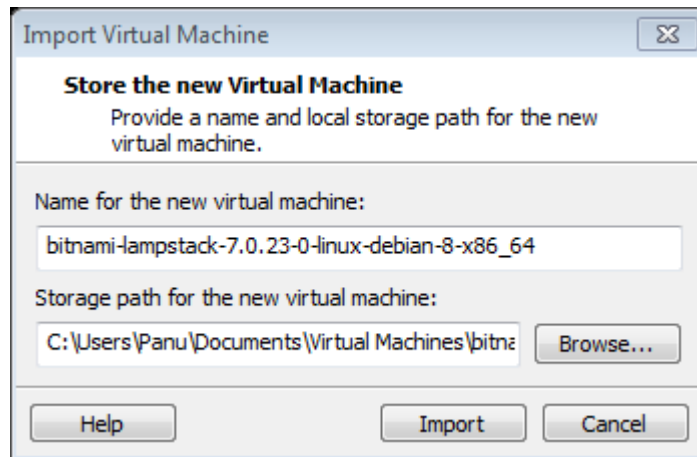
***Kuva 11. 64-bittisen virtualisoinnin enableointi isäntäkoneen BIOSista***

LAMP ei kuitenkaan ottanut toimiakseen 64-bittiselläkään Mintillä, joten seuraavaksi kokeiltiin alustaa uusi virtuaalikone debian 9.1.0 käyttöjärjestelmällä. Debianin asennus oli huomattavasti pidempi prosessi kuin Mintin ja valintoja piti tehdä yllinkyllin. Pakkaus oli myös ilmeisesti huomattavasti Mintiä laajempi sillä virtuaalikoneessa suoritettu asennusohjelma haaskeli ohjelmistokirjastoja verkon yli hyvän tovin. Samalla kun asennettiin uusi käyttöjärjestelmä, käyttökokemuksen parantamiseksi virtuaalikoneeseen lisättiin myös hieman muistia sekä kahden suoritinytimen käyttömahdollisuus.

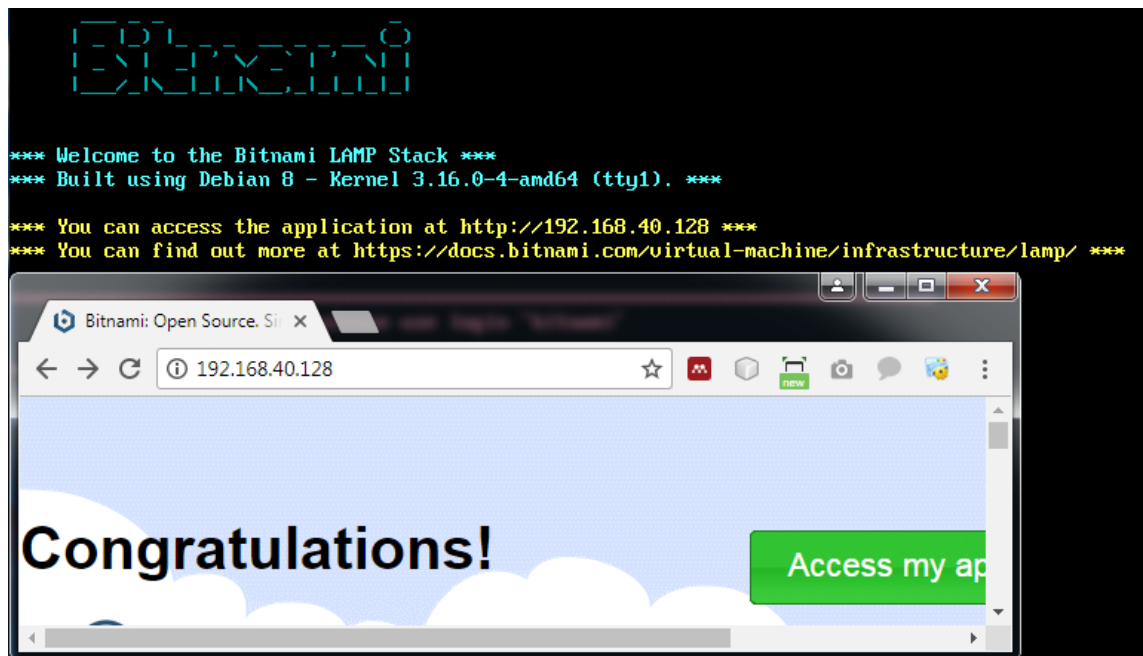


***Kuva 12. Debian asentuu virtuaalikoneeseen***

LAMPia ei vieläkään saatu toimimaan vaivalla asennetussa debian-ympäristössä. Tässä vaiheessa luettiin ohjeet ja havaittiin, että LAMP:n sivuilta ladattu Ova-tiedostoon on ilmeisesti paketoitu kokonainen valmis virtuaalikone, joka sisältää mm. käyttöjärjestelmän sekä apache-palvelimen käyttämät sovellukset esiasennettuna. Ova-virtuaalikone voidaan ottaa käyttöön vmWaren Import Virtual Machine-toiminnolla.



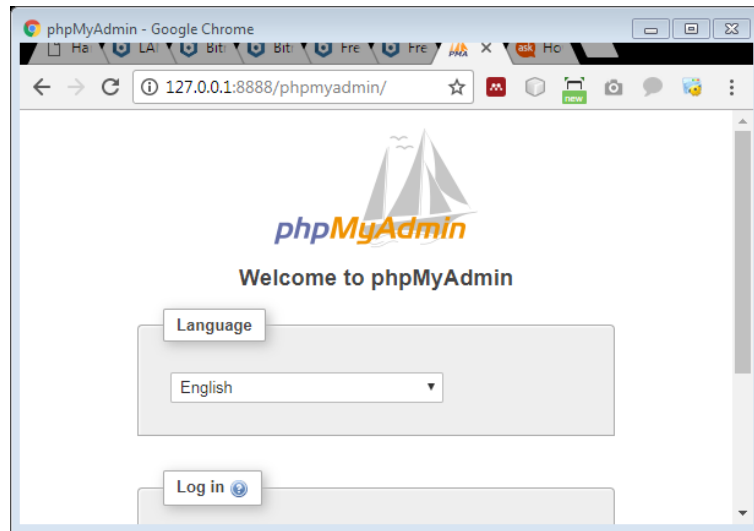
Oletusasetuksista poiketen virtuaalikoneen verkkokokoonpano säädettiin tilaan 'Host only', koska virtuaaliserverin toimintaa haluttiin tässä tapauksessa testata nimenomaan isäntäkoneella. Näiden säätöjen jälkeen serveri saatiinkin pystyyn ja yhteyskin näytti toimivan.



*Kuva 13. LAMP-serveri virtuaalikoneessa sekä isäntäjärjestelmän selainikkuna, jolla onnistuneesti otetty yhteys serveriin*

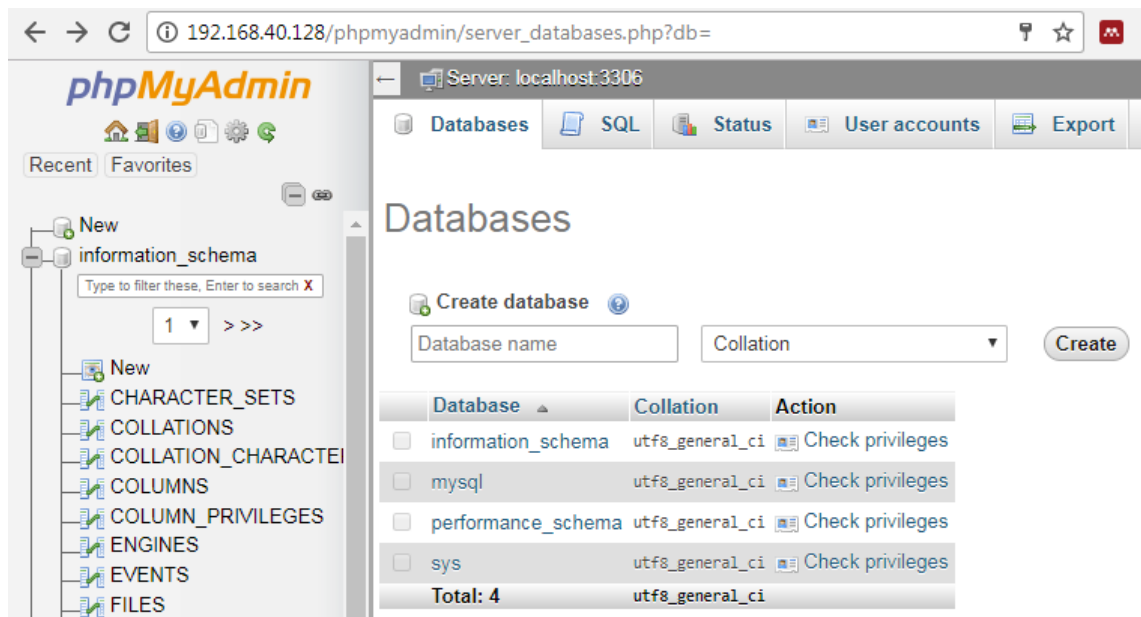
Seuraavaksi kokeiltiin phpMyAdminin toimintaa. Oletuksena phpmyadmin hyväksyy vain lokaalit yhteydenotot, joten ensin piti luoda Puttylla isäntäkoneesta virtuaaliserveriin

SSH-tunnelointi noudattaen lähteissä [4] ja [5] annettuja ohjeita. Samalla tuli harjoiteltua debianin tiedostohierarkiassa liikkumista sekä tekstitiedostojen editointia komentorivin kautta ja ainakin komennot `ls`, `edit` sekä `view` tulivat tutuiksi.



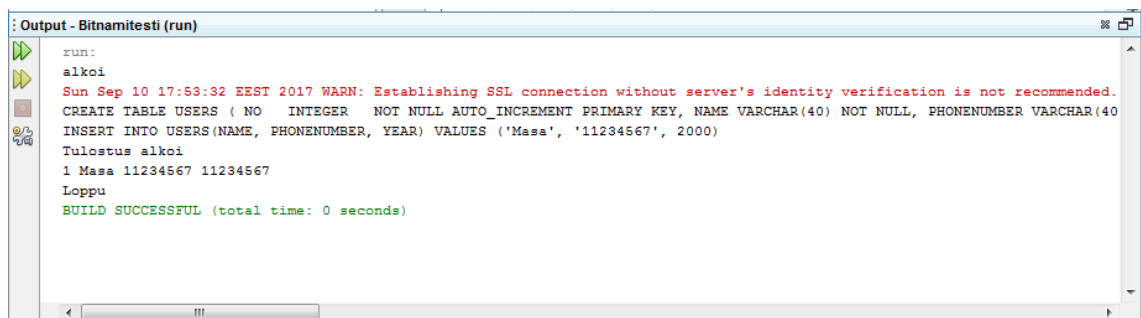
***Kuva 14. phpmyadmin kirjautuminen SSH-tunneloinilla***

Tässä vaiheessa luettiin jälleen ohjeet, ja käytiin muokkaamassa tiedostoa `/opt/bitnami/apps/phpmyadmin/conf/httpd-app.conf` siten, että yhteys phpmyadminiin saadaan myös ilman ssh-tunnelointia. Tiedostosta vaihdettiin rivi `Require Local` muotoon `Require all granted` [6]. Kun palvelin oli käynnistetty uudestaan komennolla `sudo ./ctlscript.sh restart`, phpmyadminiin pääsi käsiksi myös ilman ssh-yhteyttä polusta `192.168.40.128/phpmyadmin`. Pitkähkön pääraapimisen jälkeen arvattiin myös oikeat oletustunnukset (`root/bitnami`) ja päästiin vihdoin tutkimaan phpmyadminin näkymiä, missä oli joitakin järjestelmän ilmeisesti automaattisesti generoimia tietokantoja valmiina.



**Kuva 15. Tietokantojen hallintanäkymä phpmyadminissa**

Luotiin seuraavaksi phpmyadminin toiminnoilla uusi tietokanta 'javatesti' sekä liitettiin tähän käyttäjä 'testaaja'. Suoritettiin portinavauskomento `sudo afq allow 3306` sekä kommentoitiin tiedostosta `/opt/bitnami/mysql/my.cnf` rivi 'bind-address' jotta mahdollistetaan tietokannan käyttö etänä. Suoritettiin valmiina saatu esimerkkikoodi [7] tietokannan käsittelyn demonstroimiseksi. Kaikesta päätellen koodi sai yhteyden kantaan ja taulujen muokkaus onnistui kuten pitikin.



**Kuva 16. Testikoodin suoritus netbeansissa...**

Server: localhost:3306 » Database: javatesti » Table: users

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[More](#)

✓ Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

`SELECT * FROM `users``

☐ Profiling
 [\[Edit inline\]](#)
[\[Edit\]](#)
[\[Explain SQL\]](#)
[\[Create PHP code\]](#)
[\[Refresh\]](#)

☐ Show all | Number of rows: 25 | Filter rows:  Search this table

+ Options

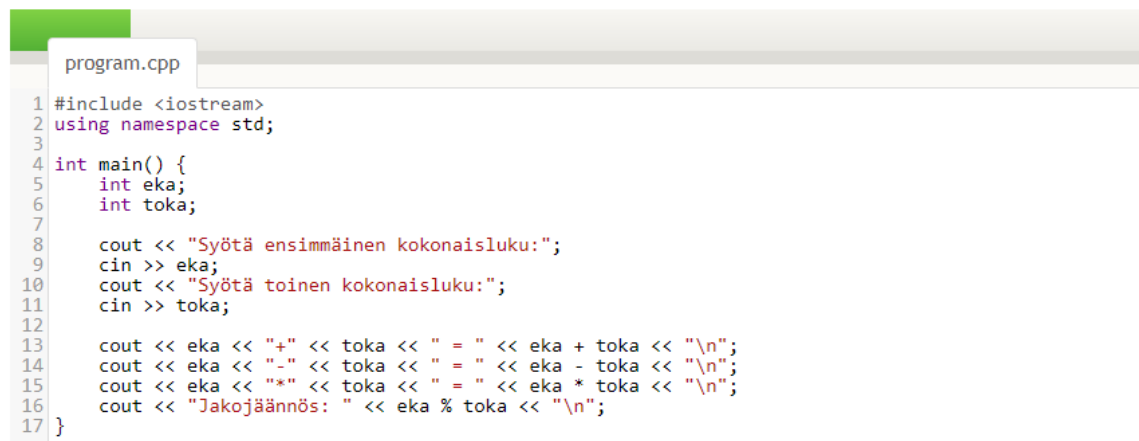
	NO	NAME	PHONENUMBER	YEAR
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	Masa	11234567	2000

*Kuva 17. ...ja tulosten tarkastus phpmyadminin kautta*

### 3. OHJELMOINTI

11.9.2017 – 12.9.2017

Osiossa tehtiin C++ sekä Python-ohjelmointi –sekä visailuharjoituksia viope-alustalla. Tehtävissä ratkaistavat ongelmat olivat melko helppoja sekä aiemmilta ohjelmointikursseilta tuttuja, joten pääpaino harjoituksella oli kyseisten kielten erityispiirteiden opettelussa. Pythonista oli kirjoittajalla jonkin verran aiempaa kokemusta, C++:sta ei käytännössä lainkaan.



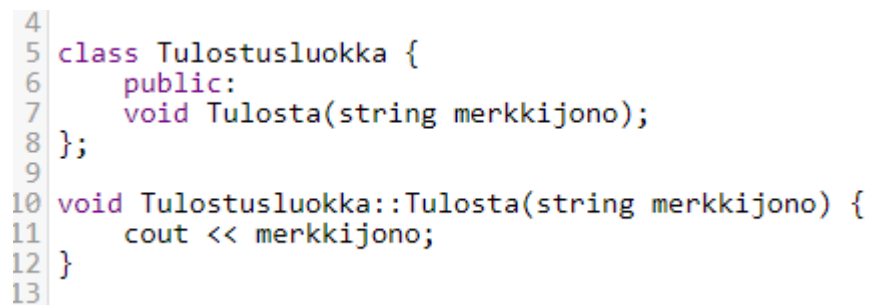
```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int eka;
6     int toka;
7
8     cout << "Syötä ensimmäinen kokonaisluku:";
9     cin >> eka;
10    cout << "Syötä toinen kokonaisluku:";
11    cin >> toka;
12
13    cout << eka << "+" << toka << " = " << eka + toka << "\n";
14    cout << eka << "-" << toka << " = " << eka - toka << "\n";
15    cout << eka << "*" << toka << " = " << eka * toka << "\n";
16    cout << "Jakojäännös: " << eka % toka << "\n";
17 }

```

**Kuva 18.** Koodausta viope-ympäristössä, jossa automaattitestit tarkistavat ratkaisun oikeellisuuden.

C++:n osalta kenties suurimpana oppimiskokemuksena oli Javaan nähden huomattavan erilainen luokkien käsittely. Ainakin kyseisen viope-kurssin esimerkeissä vallitsevana käytäntönä oli, että luokan esittelyä seuraavien aaltosulkeiden sisään kirjoitetaan attribuuteista ja metodeista vain abstraktit esittelyt, ja metodien toteutukset annetaan muualla koodissa syntaksilla `LuokanNimi::MetodinNimi(parametrit){}`. Ja mikäli toteutus on annettu, on muistettava antaa myös esittely tai muuten ohjelma ei toimi odotetusti.



```

4
5 class Tulostusluokka {
6     public:
7     void Tulosta(string merkkijono);
8 };
9
10 void Tulostusluokka::Tulosta(string merkkijono) {
11     cout << merkkijono;
12 }
13

```

**Kuva 19.** Luokkien käyttö C++:ssa

Periyttäessä luokkia Javasta tutun extends-avainsanan korvaa kaksoispiste ja public –avainsana:

```

45
46 class Auto : public Ajoneuvo
47 {
48     //Auto perii nyt Ajoneuvon piirteet
49 };
50

```

**Kuva 20. Luokkien periyttämissyntaksi**

C++:n peruskurssista suoritettiin yhteensä 61/74 harjoitusta, joista saatiin kerättyä kasaan 56 pistettä (joihinkin quiz-kysymyksiin ei pystynyt viopen bugaamisen takia vastaamaan!).

Rank	Name	Exercises completed	★ Score
1.	****, ****	74	★ 69
2.	****, ****	59	★ 57
3.	Aho, Panu	61	★ 56

**Kuva 21. Yhteenveto suoritetuista harjoituksista, C++ kurssi**

Harjoituksia jatkettiin C++ oliokurssin parissa, kiinnittäen nyt enemmän huomiota myös Viopessa tarjoiltuun kirjalliseen materiaaliin. Aiemmin tehty luokkien määrittelyn ja toteutuksen erottamiseen liittyvä huomio osoittautui paikkansa pitäväksi. Luokan attribuutit (jäsenmuuttuja) sekä metodit (jäsenfunktiot) esitetään tyypillisesti omassa määrittely (otsikko) tiedostossaan sekä jäsenfunktioiden toteutukset omassaan. Jäsenmuuttujien nimeämisessä on mitä ilmeisimmin vakiintuneena käytäntönä päättää nimet alaviivaan. Jäsenmuuttujille suositeltu näkyvyysmääre private noudattelee Javan konventiota. [8]



<pre>#include "testi.cc"  int main() {     Testi testi;     testi.TulostaJasen();     return 0; }</pre> <p>main.cpp: - Testi-luokkaa käyttävä ohjelma</p>	<pre>#ifndef TESTI_HH #define TESTI_HH  class Testi { public:     void TulostaJasen();     Testi(); private:     int jasen_; };  #endif</pre> <p>testi.hh: - Testi-luokan määrittelytiedosto</p>
---	--

```
#include<iostream>
using namespace std;
#include "testi.hh"

Testi::Testi() : jasen_(100)
{
}

void Testi::TulostaJasen()
{
    cout << "Jasen arvo: " << jasen_ << endl;
}
```

testi.cc:  
- Testi-luokan toteutustiedosto

**Kuva 22. Luokkien määrittelyparadigma C++:ssa [8]**

Java-maailmasta siirtyvälle osoittimien ja viitemuuttujien käyttö on uutta. Viitemuuttujat sisältävät viittauksen muuttujan muistiosoitteeseen, ja näin aliohjelmat voivat suoraan käsitellä muuttujia siten, että muutokset välittyvät myös aliohjelman kutsujalle. Opiskeltiin myös dynaamista muistinhallintaa (muistin varaaminen new-avainsanalla ja vapauttaminen deletellä). Kun osoitinmuuttujia ei tarvita, ne kannattaa alustaa NULL-arvoilla virheviittausten välttämiseksi. Näiltäkin kommervenkeiltä Java-koodaaja on välttynyt automaattisen roskienkeruun ansiosta.

```
int main()
{
    int x = 100;
    int* o_x = &x;
    funktio(o_x);
    cout << x;
    return 0;
}

void funktio(int *o_x)
{
    *o_x *= *o_x;
}
```

**Kuva 23. Tulostaa "10000". Viitemuuttujan avulla funktion tekemät muutokset välittyvät funktion kutsujalle ilman erillistä return-lausetta [8]**

Opittiin myös C++:n eri oliotyypeistä. Siteerataan tässä viopen loistavaa tiivistelmää aiheesta:

*"Automaattinen olio*

*Olio on ohjelmalohkon paikallinen olio (vrt. paikalliset muuttujat). Kun olion muodostanut ohjelmalohko päättyy, olio hävitetään ja sille varattu keskusmuistialue vapautuu*

*Dynaaminen olio*

*Oliolle varataan ajonaikaisesti luokan määrittelyn mukainen tila muistista new-operaattorilla. Ohjelmassa muodostuneiden olioiden määrä voi vaihdella eri ajokerroilla. Oliolle varattu muistitila vapautetaan delete-operaattorilla, jonka vuoksi olio voi "säilyä hengissä", vaikka ohjelmalohko, jossa se muodostettiin päättyisikin*

*Staatinen olio*

*Olio on joko globaali, tai static-määreellä paikalliseksi määritetty. Oliolle varataan luokan määrittelyn mukainen tila pääohjelman käynnistyessä. Oliolle varattu muistitila vapautuu pääohjelman päättyessä." [8]*

Harjoiteltiin parametrillinen ja parametriton konstruktori sekä tuhoaja:

```
//Parametriton konstruktori
Piste::Piste() : x_(1), y_(1) {}

//Parametrillinen konstruktori
Piste::Piste(int x, int y) : x_(x), y_(y) {}

//Tuhoaja
Piste::~~Piste() {
    cout << "Tuhoaja suoritettu ja olio tuhottu.";
}
```

Viopen C++ -kurssilla ei tullut vastaan Javaan verrattavia kehittyneitä Collection-tyyppejä säiliöluokkia. Todennäköisesti niitä kuitenkin löytyy, jos asiaa lähtisi kaivelemaan. Oliosäilöinä käytettiin nyt taulukoita, joiden koko pitää olla etukäteen päätetty. Esim. luotaessa luokka Kenneli, voitaisiin päättää Koirien maksimimääräksi 50. Taulukko alustetaan Kenneliä luotaessa esim. NULL-arvoilla. Lisättäessä tai poistettaessa Koiria täytyy pitää huoli uuden Koiran sijoittamisesta oikeaan indeksiin, ylläpitää Koirien lukumäärää erillisessä apumuuttujassa jne. Melko hankalaa, mutta kenties tämä on hinta joka joudutaan maksamaan laiteläheisyydestä ja suorituskyvystä.

```
23: class Kenneli
24: {
    //määrittelyjä...
25: private:
26:     Koirat* koirat_[50];
27:     int koirien_lkm_;
28: };
29:
```

C++ oliokurssin harjoituksista suoritettiin tässä vaiheessa noin puolet, ennen kuin jatkettiin eteenpäin.



**Kuva 24. Yhteenveto tehdyistä harjoituksista, C++ oliokurssi**

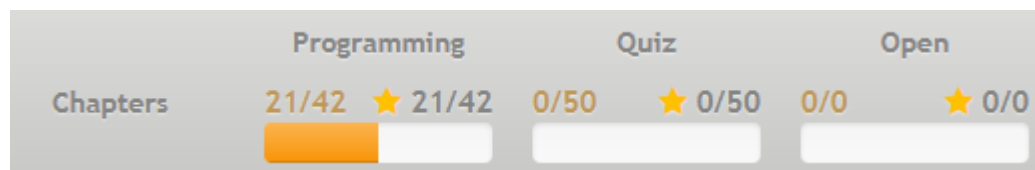
Jatkettiin seuraavaksi Python-harjoitusten merkeissä. Olennaisimmat havainnot:

- Python3 tulostussyntaksi erilainen 2.x nähden, print() käyttäytyy nyt funktiokutsun tavoin ja voidaan parametroida hauskoilla tavoilla
- raw\_input on nyt pelkkä input
- Merkkijonojen käsittely on helppoa ja hauskaa:
  - Substringejä saadaan miono[alku:loppu+1] syntaksilla, indeksointi lähtien nolasta
  - Koko merkkijonon kääntäminen miono[::-1]
  - Jne...
- Muuttujilla ei edelleenkään vahvaa tyypitystä
- Ohjausrakenteet tehdään edelleen sientämällä (välilyönti, ei tab), aaltosulkeita ei käytetä
- True ja False avainsanat isolla
- Main metodin kutsutapa ohjelman aloittamiseksi on melko eksoottinen:
 

```

7 if __name__ == "__main__":
8     main()
      
```
- Oletusparametri funktiolle määrittelyllä def function(param = "default")
- Listat erittäin tehokas tapa säilöä ja käsitellä tietoa

Python3-ohjelmointiharjoituksista suoritettiin 21/42 eli 50 %. Harjoituksia voidaan jatkaa vielä myöhemmin, jos aikaa jää ja/tai tähän koetaan tarvetta. Visailuharjoituksiin ei nyt kulutettu aikaa sillä asiat tuntuivat olevan melko hyvin hallinnassa.



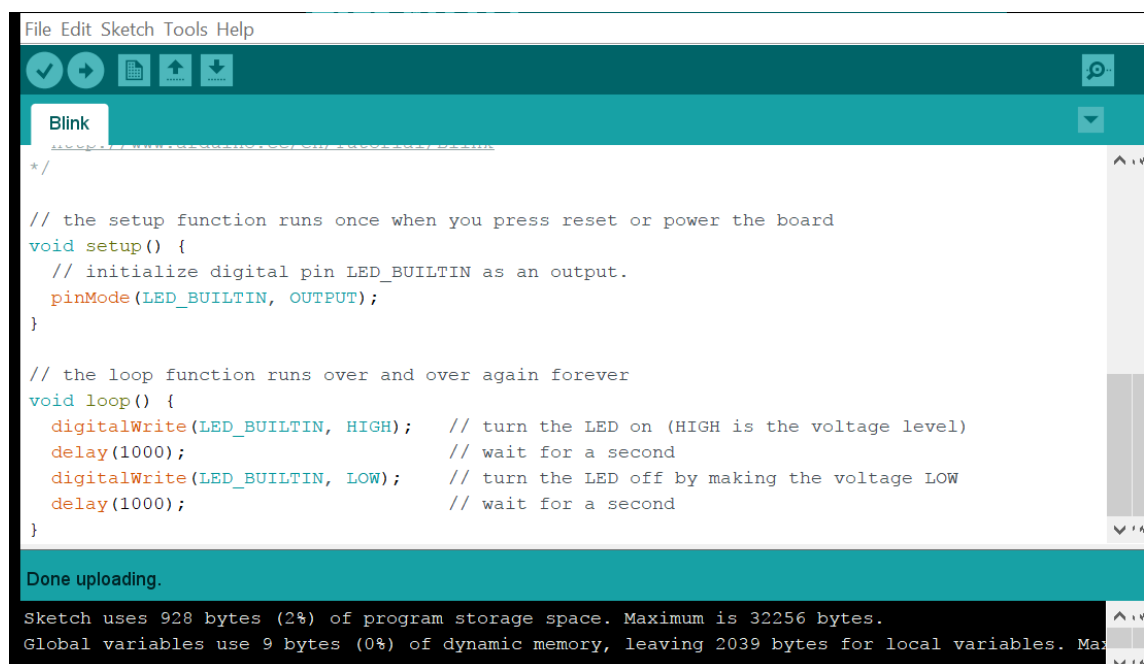
**Kuva 25. Yhteenveto tehdyistä Python-harjoituksista viopessa**

## 4. ARDUINO

20.9.2017 – 21.9.2017

Arduino on avoimeen lähdekoodiin perustuva mikrokontrolleripohjainen alusta erilaisille elektroniikkaprojekteille. Arduinolla voidaan lukea input-tietoja kuten sensoridataa, napinpainalluksia tai sosiaalisen median viestejä. Laitteeseen ohjelmoitava logiikka käsittelee syötteet sekä tuottaa ohjelmoijan haluamat output-vasteet kuten moottorin käynnistäminen, LED-valon aktivointi tai verkkojulkaisun tekeminen. Logiikka ohjelmoidaan Arduinon omalla ohjelmointikielellä. [9]

Kurssilla käytössä oli Arduino UNO –laitteet sekä kokoelma erilaisia lisälaitteita sekä komponentteja, joilla harjoitustyöt suoritettiin. Logiikkaohjelmointia varten asennettiin Arduinon IDE-ympäristö windows natiivi-alustaiselle työasemalle, sillä virtualisointiympäristössä sen käyttö havaittiin haasteelliseksi. Kielen syntaksi muistuttaa c++:aa. Valmiit ohjelmistot ladataan Arduinoon USB-kaapelin välityksellä, josta laite ottaa myös virtansa. Laitetta on mahdollista käyttää myös sopivan verkkomuuntajan avulla.



```
File Edit Sketch Tools Help

Blink

*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

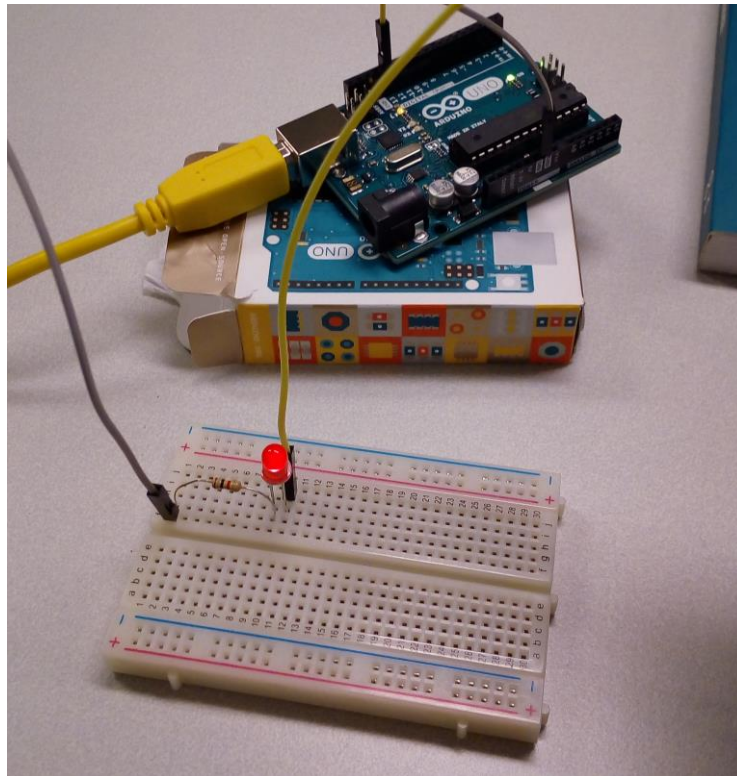
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}

Done uploading.

Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Max
```

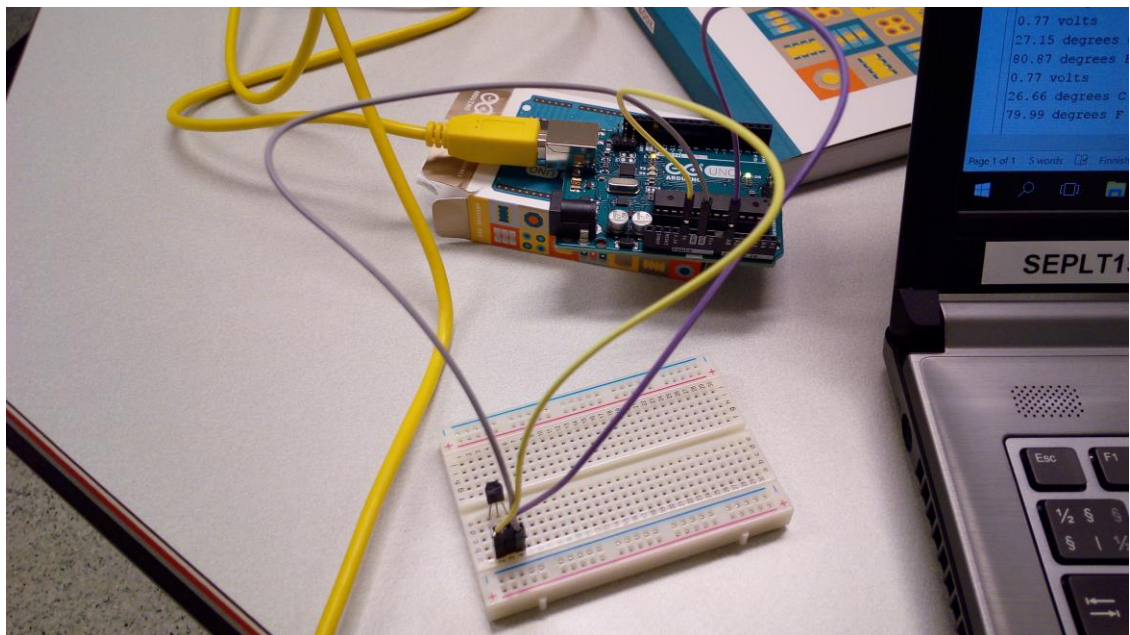
**Kuva 26. Harjoituksen 1. koodi Arduino IDE:ssä**

Harjoituksen 1 aiheena oli ledinvilkutus. Arduinoon kytketty LED syttyy ja sammuu koodissa millisekunteina annettavan viiveen mukaisesti. Kytkentä oli varsin yksinkertainen ja saatiin toimimaan melko vaivattomasti alla (Kuva 27)



***Kuva 27. LED syttyy ja sammuu Arduinin ohjaamana 1000 ms välein, vaikka sitä onkin kuvassa haastavaa demonstroida***

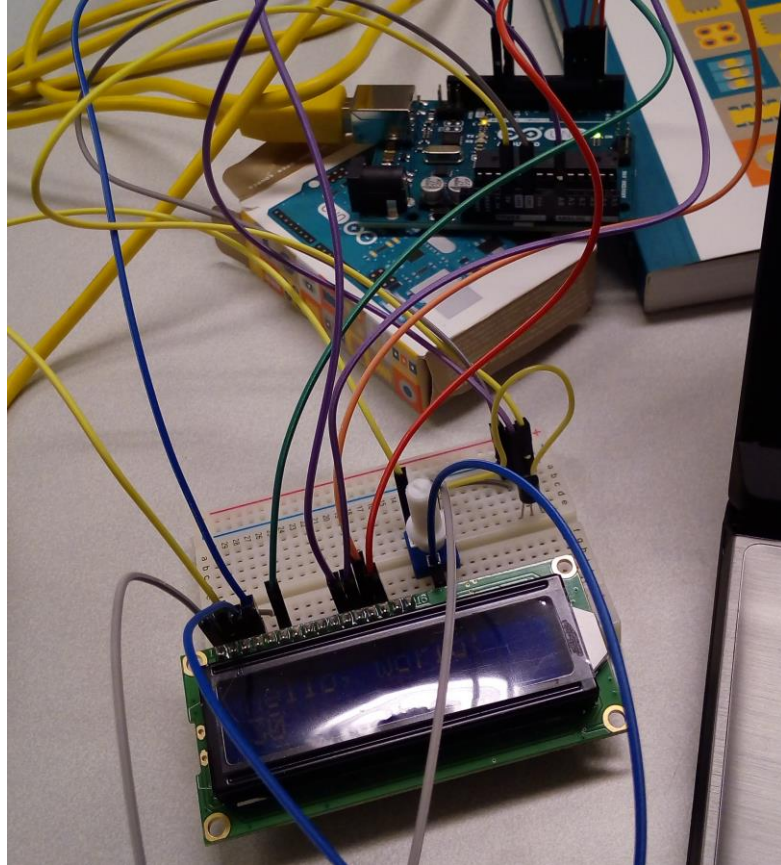
Seuraavaksi harjoiteltiin lämpötilan mittausta TMP36-sensorilla sekä tiedon tulostamista Arduinin terminaliin. Kuvassa näkyy kytkentä sekä kuvan oikeassa ylä laidassa pieni pätkä tulostusvirtaa. Anturi reagoi herkästi esim. sormella kosketukseen.



***Kuva 28. Lämpötilan mittausta Arduinolla***

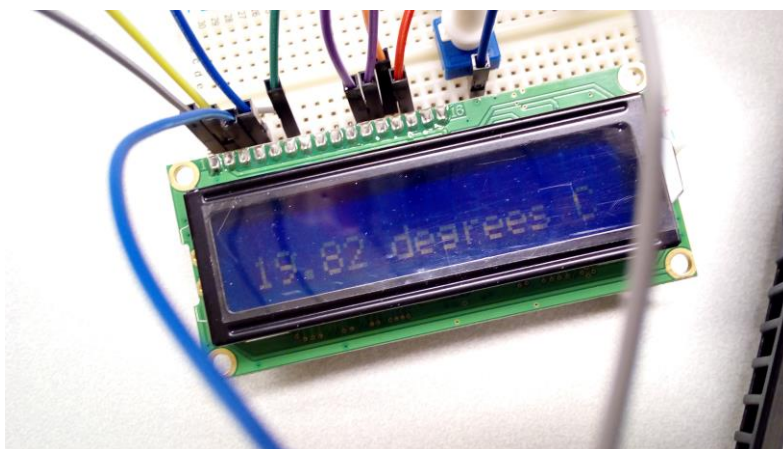


Seuraava tehtävä oli edellisen jatkojalostamista eli lämpötila piti saada nyt tulostumaan terminal-ikkunan sijaan erilliseen LCD-näyttöön. Kuvassa 29 näkyy LCD-näytön kytkentä. Valkoinen pitkänomainen komponentti on säätövastuksen nuppi, jolla näytön kontrasti saatiin säädettyä sopivaksi.



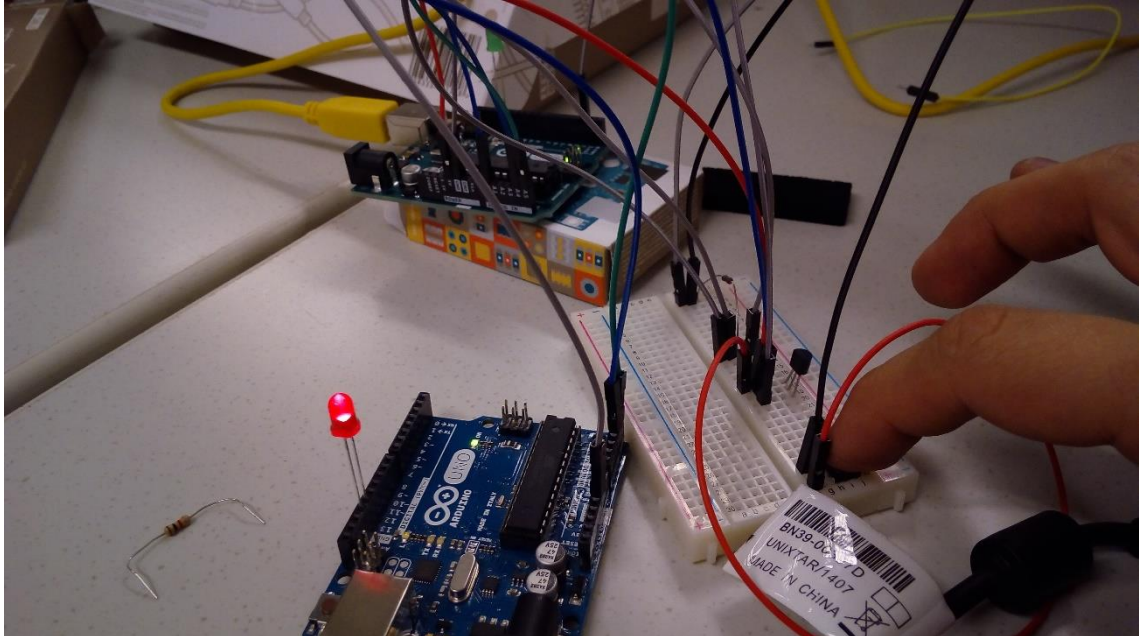
*Kuva 29. LCD-näyttö kytkettynä Arduinoon.*

Muutaman koodirivimuutoksen jälkeen myös haluttu tieto saatiin tulostumaan näytölle (Kuva 30).



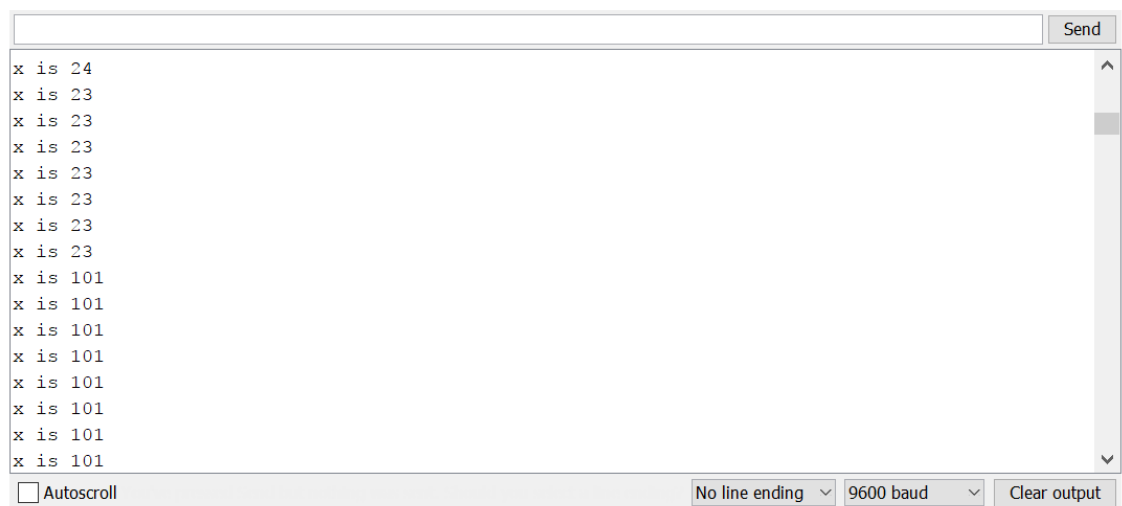
*Kuva 30. Lämpötila-anturin lukema LCD-näytöllä.*

Seuraavan harjoituksen aiheena oli kahden Arduinin muodostama systeemi. Ideana oli, että Master-laite kerää sensoritietoa (lämpötila) sekä rekisteröi napinpainalluksia ja välittää tiedot Slave-laitteelle jossa tiedot luetaan, käsitellään sekä tuotetaan käyttäjälle ohjelmoinnin mukaiset vasteet. Kuvan 31 tilanteessa Master-laitteeseen kytkettyä painonappia pidetään pohjassa, jolloin slaven punainen ledi syttyy.



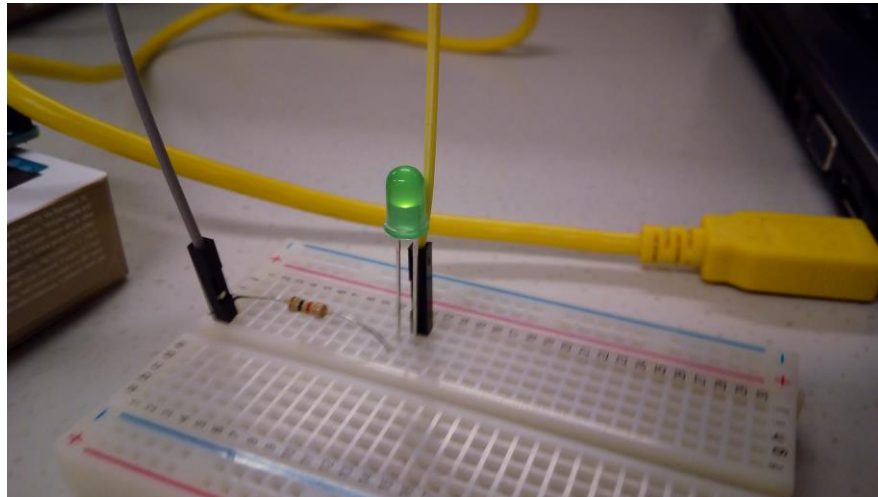
***Kuva 31. Master-Slave kokoonpanon testausta***

Samalla slaven terminaliin tuleva tulostevirta muuttuu. Kun nappia ei paineta, tulostetaan sinne vallitseva lämpötila celsiusasteina ja kun nappia painetaan, tulostetaan arvo 101.



***Kuva 32. Master ohjaa slaven tulostusvirtaa***

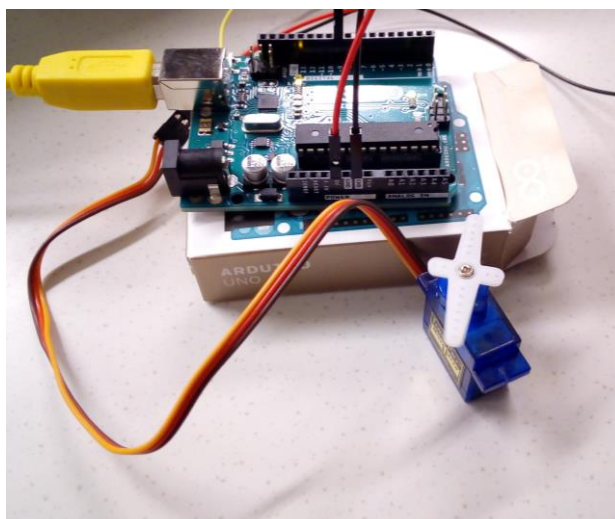
Viimeisenä muttei vähäisimpänä, tehtiin ohjelmointiharjoitus jossa Arduino IDE:ssä luotiin luokkatiedostot Morse.h sekä Morse.cpp. KytKentä oli tässä sama kuin harjoituksessa 1 jossa testattiin ledinvilkutusta. Morse-luokka tarjoaa metodit dash (pitkä ledinväläytys) ja dot (lyhyt ledinväläytys). Näitä ketjuttamalla voidaan saada aikaan morse-sanomia. Testissä toteutettiin vaaditut luokat ja tämän jälkeen kytkennässä oleva LED vilkutteli iloisesti SOS-hätäsanomaa (Kuva 32).



*Kuva 33. Vihreä LED morsettaa, vaikei se kuvassa valitettavasti näy.*

28.9.2017

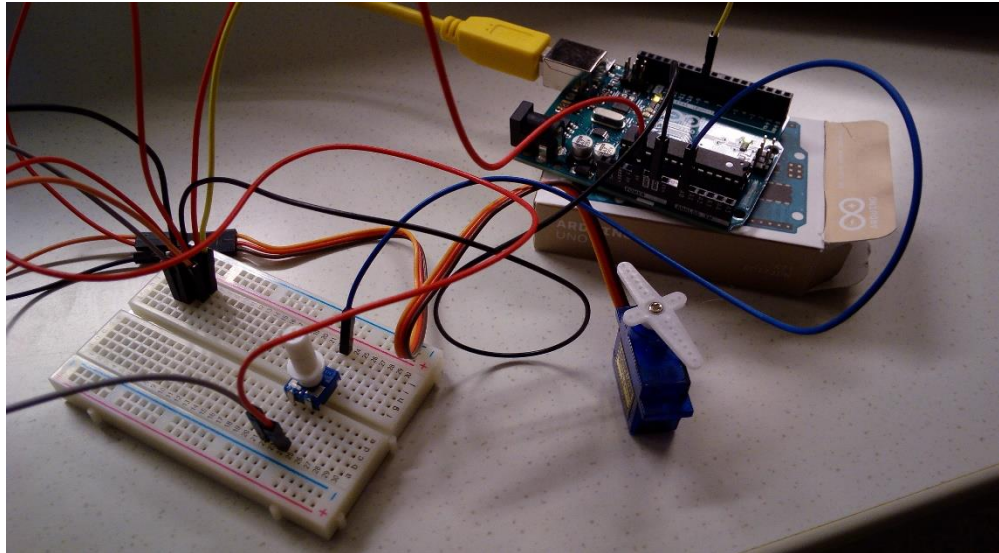
Jatkettin vielä työskentelyä muutaman Arduino-lisäharjoituksen parissa. Ensimmäinen näistä oli servomoottorin (Tower Pro SG90) ohjaaminen. Ensin tehtiin minimi-kytkentä, jossa esiohjelmoitu logiikka liikutti moottoria yhdellä asteella 15 millisekunnin välein, saaden aikaan illuusion jatkuvasta liikkeestä vaikka todellisuudessa moottori etenee diskreetein ”naksahduksin”. Jos tarkkaan katsoo kuvaa 34, on mahdollista nähdä tämä liike ”viisarin” kärjen hienoisena blurrautumisena.



*Kuva 34*

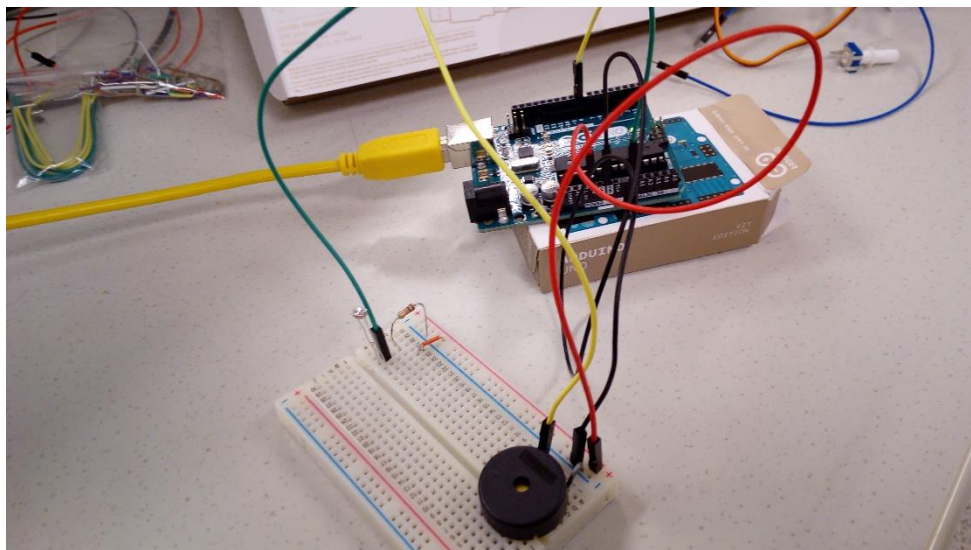


Toisena vaiheena oli kytkennän jatkojalostaminen siten, että servomootorin asento olisi käyttäjän säädettävissä. Haluttu moottorin asento tulkitaan nyt ennalta määritetyn logiikan sijaan säätövastuksen analogisesta input-tiedosta. Tämäkin kytkentä saatiin melko pienellä vaivalla toimimaan ja kytkentä on esitetty kuvassa 35.



**Kuva 35**

Bonustehtävänä toteutettiin vielä fotoresistoriohjattu Theremin-soitin, johon saatiin ohjeet Arduino Tutorials-sivustolta [11]. Theremin on Neuvostoliitossa 1920-luvulla kehitetty elektroninen instrumentti, jota soitetaan koskettamatta itse soitinta lainkaan [10]. Arduino-pohjaisessa toteutuksessa (kuva 36) ideana on muuttaa piezoelementistä tulevan äänen taajuutta valoon reagoivan vastuksen aistiman valoisuuden määrän mukaan, jota puolestaan voidaan säädellä liikuttelemalla kättä tms. ruumiinosaa fotoresistorin yllä.



**Kuva 36. Valoisuusohjattu Theremin. Valoon reagoiva vastus vasemmalla vihreän johdon vieressä, musta pyöreä piezoelementti alareunassa.**

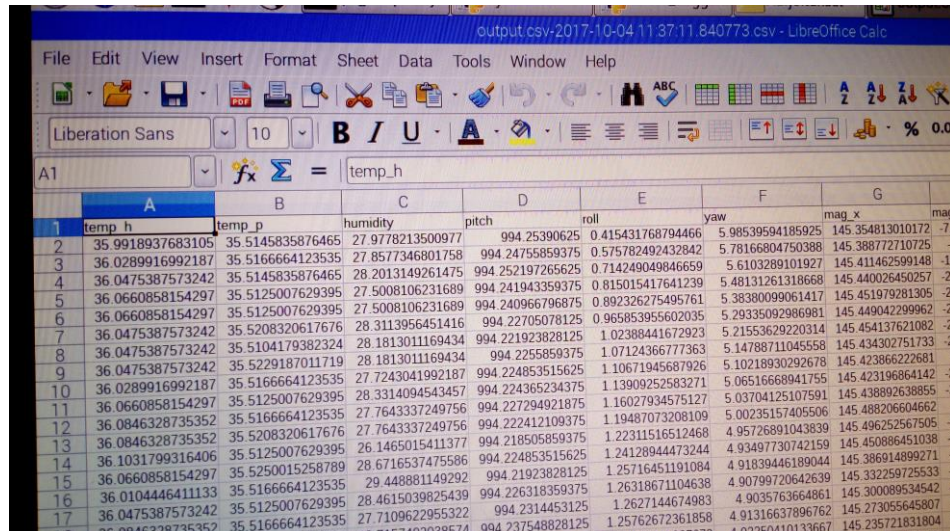
## 5. RASPBERRY PI

*4.10.2017, 11.10.2017*

Osiassa tutustuttiin Raspberry PI:hin ja vastaaviin ”luottokorttikoneisiin”. Näitä ARM-proessoriarkkitehtuuriin perustuvia minitietokoneita löytyy nykyään useammalta valmistajalta sekä eri konfiguraatioilla, kuten mm. kurssilla käsitellystä Mikrobotin artikkelista ilmenee [12]. Siinä missä esim. Arduino on pohjimmiltaan AD/DA-muunnosalusta sekä yksinkertainen mikrokontrolleri, ovat luottokorttikoneet täysiverisiä tietokoneita. Niihin pystyy (ja on myös pakko, jos laitteella haluaa ylipäättään mitään tehdä) asentamaan graafisen käyttöjärjestelmän ja periaatteessa käyttää vaikka web-selailuun tai toimisto-ohjelmien pyörittämiseen. Virrankulutus on pientä ja sen vuoksi erilaiset sulautetut järjestelmät ovatkin erityisen hyviä sovellusalueita.

Käyttöjärjestelmänä on tyypillisesti jokin linux-distributio, joten kurssilla aiemmin opitut taidot tulevat tarpeeseen. Esim. Raspberryn tapauksessa käytössä on laitteelle räätälöity Raspbian. Kyseinen käyttöjärjestelmä sisältää perustoiminnallisuuksien lisäksi hyvinkin mielenkiintoisia lisäherkkuja, kuten normaalisti maksullisen symbolisen Mathematica-ohjelmiston esiasennettuna.

Kurssilla testattava laite oli Raspberry PI 3. Heti alkuvaiheessa oli haasteita saada koneita boottaamaan, mutta pienen tutkiskelun jälkeen homma onnistui ja päästiin harjoitusten pariin. Ensimmäinen harjoitus oli oikeastaan perusasetusten säätämistä (lokalisaatio, käytettävät io-rajapinnat yms). Tämän jälkeen lähdettiin heti kokeilemaan ”sense hat”-sensorilisäpalikkaa joka sisältää matkapuhelimistakin tuttujen asento- ja kiihtyvyysanturien lisäksi myös mahdollisuuden ilmanpaineen, kosteuden ja lämpötilan mittaukseen. Yksinkertainen data logging-sovellus toteutettiin Python 3 -kielellä, joka tallentaa antureiden tietoja csv-formaattiin.



	A	B	C	D	E	F	G
	temp_h	temp_p	humidity	pitch	roll	yaw	mag_x
1							
2	35.9918937683105	35.5145835876465	27.9778213500977	994.25390625	0.415431768794466	5.98539594185925	145.354813010172
3	36.0289916992187	35.5166664123535	27.8577346801758	994.24755859375	0.575782492432842	5.78166804750388	145.388772710725
4	36.0475387573242	35.5145835876465	28.2013149261475	994.252197265625	0.714249049846659	5.6103289101927	145.411462599148
5	36.0660858154297	35.5125007629395	27.5008106231689	994.241943359375	0.815015417641239	5.48131261318668	145.440026450257
6	36.0660858154297	35.5125007629395	27.5008106231689	994.240966796875	0.892326275495761	5.38300990961417	145.451979281305
7	36.0475387573242	35.5208320617676	28.3113956451416	994.22705078125	0.965853955602035	5.29335092986981	145.449042299962
8	36.0475387573242	35.5104179382324	28.1813011169434	994.221923828125	1.02388441672923	5.21553629220314	145.454137621082
9	36.0475387573242	35.5229187011719	28.1813011169434	994.2255859375	1.07124366777363	5.147887711045558	145.434302751733
10	36.0289916992187	35.5166664123535	27.7243041992187	994.224853515625	1.10671945687926	5.10218930292678	145.423196864142
11	36.0660858154297	35.5125007629395	28.3314094543457	994.224365234375	1.13909252583271	5.06516668941755	145.438892638855
12	36.0846328735352	35.5166664123535	27.7643337249756	994.227294921875	1.16027934575127	5.03704125107591	145.482065046662
13	36.0846328735352	35.5125007629395	27.7643337249756	994.222412109375	1.19487073208109	5.00235157405506	145.486252567505
14	36.1031799316406	35.5208320617676	26.1465015411377	994.218505859375	1.22311516512468	4.93497730742159	145.450886451038
15	36.0660858154297	35.5125007629395	28.6716537475586	994.224853515625	1.24128944473244	4.91839446189044	145.386914899271
16	36.0104446411133	35.5250015258789	29.448881149292	994.21923828125	1.25716451191084	4.90799720642639	145.332259725533
17	36.0475387573242	35.5166664123535	28.4615039825439	994.226318359375	1.26318671104638	4.9035763664861	145.300089534542
18	36.0475387573242	35.5125007629395	27.7109622955322	994.2314453125	1.2627144674983	4.91316637896762	145.273055645807
19	36.0846328735352	35.5166664123535	28.7157402038574	994.237548828125	1.25762672361858	4.92250410133667	145.235721831804

*Kuva 37*

Toisessa isommassa harjoituskokonaisuudessa tutustuttiin Raspin omaan kameramoduuliin. Kuvassa näkyy moduuli paikalleen asennettuna.



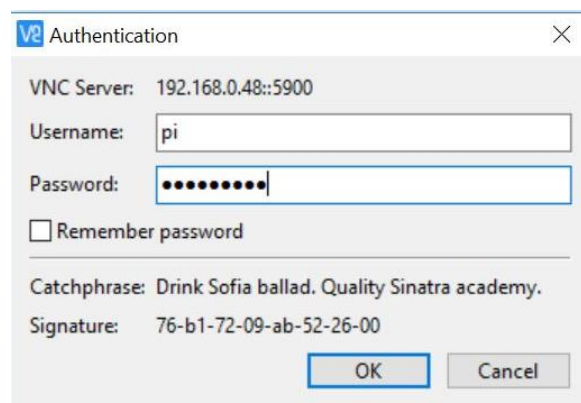
*Kuva 38*

Peruskäyttö eli kuvien ottaminen käy suoraan konsolista komennolla raspistill. Pyyntöä voidaan parametroida eri tavoin. Mm. ”itselaukaisuviiveen” säätö on mahdollista sekä kuvan manipulointi eri tavoin, kuten kääntö pysty tai vaakasuunnassa sekä rotaatio onnistuvat helposti, mikä on näppärää mikäli moduuli sattuu olemaan asemoitu väärin päin kuvattavaan kohteeseen nähden. Videokuvaaminenkin onnistuu 1920x1080 resoluutiolla, oletusasetuksilla 30 fps taajuudella mutta laite antaa mahdollisuuden jopa 60 hz asti. [13] Kuvanlaatua voisi luonnehtia vähintäänkin kohtuullisesti – toki moduuli on tähtäilylle herkkä siinä missä mikä tahansa muukin kamera, kuten esim. kuvasta 39 havaitaan.



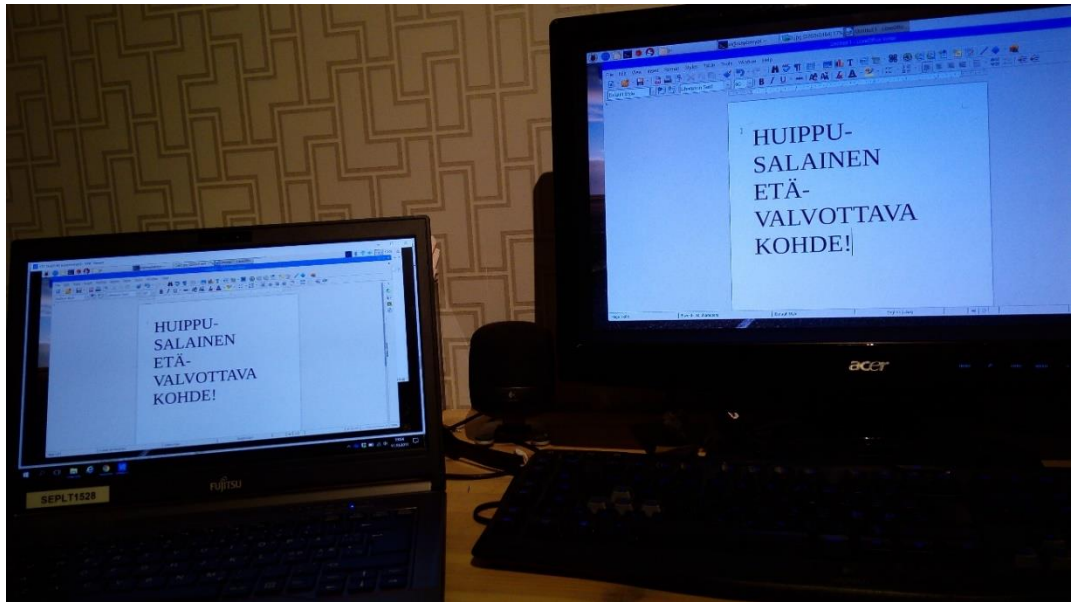
***Kuva 39. Firstpic.jpg - ensimmäinen kuva raspin kameralla***

Etätyöpöydän virittely oli seuraava tehtävä. Raspberryä voidaan käyttää etäyhteydellä virtuaalityöpöydän kautta omalta läppäriltä, jolloin itse Raspberry voi fyysisesti sijaita missä hyvänsä, eikä tarvitse välttämättä edes näyttöä tai muita oheislaitteita kunhan verkkoyhteys on saatavilla. Tätä varten asennettiin Raspberryyn VNC remote connect-ohjelmisto sekä yhteyttä ottavaan koneeseen VNC Viewer (kuva 40) noudattaen raspberryn sivuilla olevaa dokumentaatiota [lähde]. Etäkäyttö onnistu tällä hetkellä oman langattoman kotiverkon kautta, ks. kuvat 42 ja 43. VNC on ilmeisesti mahdollista saada toimimaan myös pilvipalvelun yli, mutta tätä toiminnallisuutta ei vielä tässä vaiheessa päästy testaamaan.

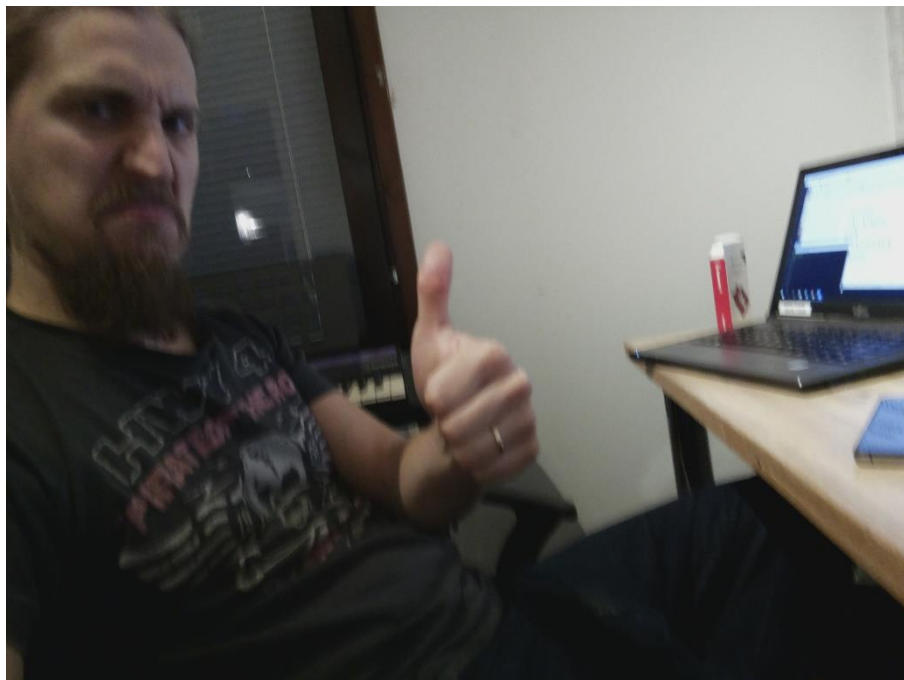


***Kuva 40. Yhteydenotto VNC Viewerillä Raspberryyn oli helppoa. IP-osoite saadaan tarvittaessa Raspin konsolista sanomalla "ifconfig", tai kotiverkossa myös tutkimalla reitittimen ARP-taulua***



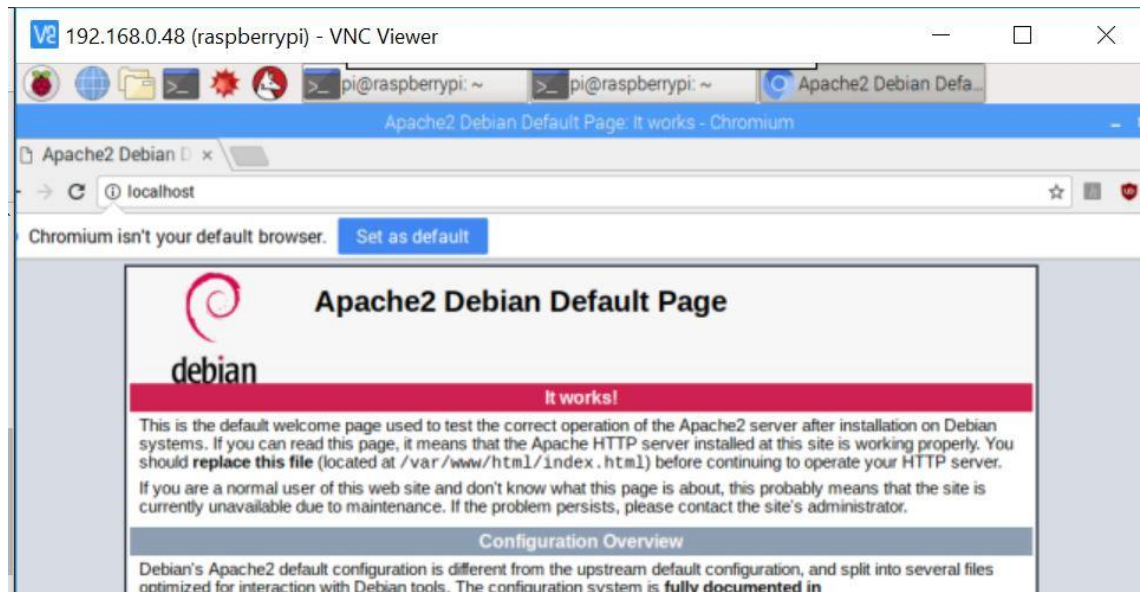


*Kuva 41. Raspberryn työpöytään otettu yhteys läppäriltä WLAN-verkon yli.*



*Kuva 42. Etätyöpöydän kautta napattu kuva*

Vuorossa oli seuraavaksi harjoitus, jossa yhdisteltiin http-palvelimen pystyttämistä, bash-skriptausta, sekä tehtävien ajastusta crontabilla. Tavoitteena oli valvontakamera-tyyppinen ratkaisu joka ottaa määritellyin väliajoin kuvan ja lataa sen web-palvelimelle. Työ aloitettiin asentamalla apache2-palvelin ohjeiden mukaisesti [15] ja testaamalla sen toiminta (kuva).











*Kuva 43*

Seuraavaksi luotiin bash-skripti ottamaan kuvia ja tallentamaan ne päivämäärällä ja kellonajalla nimettynä. Tässä käytettiin hieman tehtävänannosta poikkeavaa syntaksia. fswebcam-sovellusta ei syystä tai toisesta saatu toimimaan, joten varsinainen kuvan ottaminen toteutettiin edellisestä harjoituksestakin tutulla raspistill-komennolla. Webcam-sh skriptin sisällöksi tuli siis:

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
sudo raspistill -t 1 -o /var/www/html/webcam/$DATE.jpg
```

Sudo-komento vaadittiin, jotta saatiin riittävät kirjoitusoikeudet hakemistoon. Kun kyseinen skripti ajastettiin crontabiin (komento crontab -e) suoritettavaksi yhden minuutin välein, palvelimelle alkoikin ilmestyä kuvia tasaiseen tahtiin (kuva 44)

## Index of /webcam

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
 <a href="#">Parent Directory</a>		-	
 <a href="#">2017-10-11_1943.jpg</a>	2017-10-11 19:43	5.1M	
 <a href="#">2017-10-11_1954.jpg</a>	2017-10-11 19:54	5.2M	
 <a href="#">2017-10-11_1955.jpg</a>	2017-10-11 19:55	5.3M	
 <a href="#">2017-10-11_1956.jpg</a>	2017-10-11 19:56	5.2M	
 <a href="#">2017-10-11_1957.jpg</a>	2017-10-11 19:57	5.2M	
 <a href="#">2017-10-11_1958.jpg</a>	2017-10-11 19:58	5.2M	
 <a href="#">moi.jpg</a>	2017-10-11 19:52	5.1M	
Apache/2.4.25 (Raspbian) Server at 192.168.0.48 Port 80			

**Kuva 44.** *www-palvelimella muutamia manuaalisesti otettuja testikuvia sekä automaattisesti ajastettuja kuvia*

## 6. IOT

### 6.1 IOT:n turvallisuusnäkökohdat (Sforzin & Conti 2016)

*Osiossa perehdyttiin Sforzin ja Contin artikkeliin [34], josta esitetään tässä lyhyt referaatti.*

IoT-infrastrukturi tulee tulevaisuudessa parantamaan ihmisten elämänlaatua, mutta asettaa myös haasteita turvallisuuden ja yksityisyyden näkökulmasta. IoT-laitteet ovat usein huonosti suojattuja monessakin mielessä. Laitteiden suorituskyky on rajallinen, joten vahvoja salausalgoritmeja ei voida käyttää. Verkkoyhteydet on usein toteutettu langattomasti, joka tekee laitteesta helpon kohteen salakuuntelulle tai tietojen sabotoinnille. Pahimmillaan koko laite on täysin vartioimattomassa tilassa, jolloin pahantahtoinen henkilö voi päästä siihen fyysisesti käsiksi. Mm. nämä tekijät tekevät IoT-laitteista mielenkiintoisen kohteen kyberhyökkääjille.

Ongelmaan kirjoittajat esittävät omaa RPIDS-nimellä kulkevaa ratkaisuaan. Kyse on ”Plug & Protect”-ratkaisusta, jossa on paketoitu IDS (Intrusion Detection System) eli tunkeutumisen havainnointi järjestelmä kannettavalle laitteelle, tässä tapauksessa siis Raspberry Pi:hin. Plug & Protect –nimitys tulee siitä, että laite voidaan helppokäyttöisesti liittää mihin hyvänsä IoT-verkkoon, jonka jälkeen se alkaa tarkkailla liikennettä ja havaitessaan epäilyttävää toimintaa raportoi tästä verkon ylläpitäjälle. Lisäksi laitteet voivat muodostaa keskenään klustereita ja tätä kautta voidaan saada tietoja myös lokaalin verkon ulkopuolella meneillään olevista hyökkäyksistä, mikä parantaa edelleen kokonaisturvallisuutta.

Varsinaisia IDS-järjestelmiä on olemassa useita erilaisia. Artikkelin tapauksessa on ollut käytössä avoimeen lähdekoodiin perustuva Snort, joka kykenee verkkoliikenteen reaaliaikaiseen analysointiin sekä lokitietojen kirjaamiseen. Snort on hyvin ylläpidetty ja kykenee suojaamaan uusimmiltakin turvallisuushkilta. Huonona puolena voitaneen pitää sitä, että Snort ei tue säikeistämistä eli vain yhden suoritinytimen käyttö on mahdollista per Snort-instanssi.

Artikkelissa on arvioitu Raspberry PI:n suorituskykyä Snortin alustana. Tarkemmin käytössä on ollut Raspin muutaman vuoden vanha Model 2 B. Koeasetelmana oli Rasperryn sekä Ubuntu-alustaisen PC-tietokoneen muodostama Internetistä ja muista laitteista eristetty lähiverkko. Verkkoon luotiin tcpreplay-työkalulla simuloitua liikennettä, josta osa oli normaalia liikennettä ja osalla pyrittiin mallintamaan erilaisia kyberhyökkäyksiä. Liikenteen nopeutta, pakettikokoa sekä tunnistuksen parametreja varioitiin tutkimuksen aikana jotta saataisiin monipuolinen kuva konfiguraation suoriutumisesta.



Artikkelin johtopäätöksissä todetaan, että suorituskky pysyi useita päiviä kestävien testien ajan hyvällä tasolla. Oletettavasti Raspberry Pi toimii hyvin IDS-järjestelmän alustana, kun tarkkailtavan verkon koko on suhteellisen pieni ja kuormitus alhainen. Testeissä noin 70 Mbit / s tasoon asti laite toimi moitteetta, kunnes pakettien ”droppausta” alkoi ilmaantua. Raspberrystä löytyvä 100 Mbit / s Fast Ethernet-rajapinta rajoittaa nopeutta, mitä artikkelin tekijät harmittelevat, koska nykyään olisi saatavilla jo nopeampiakin ethernet-tekniikoita.

Arviona tekstistä voisi todeta, että artikkelissa on esitetty mielenkiintoinen ratkaisu IoT-maailman kyberuhkien torjumiseen sekä demonstroitu sen toimivuutta, mutta toistaiseksi vain simulaatiodatalla ja rajallisessa verkossa. Kokeet todellisessa käyttökohteessa olisivatkin selkeästi seuraava askel. On täysin ajateltavissa, että laitetta olisi hyvinkin mahdollista jatkokehittää aina kaupalliseksi tuotteeksi asti. Tätä tukee pitkälti ”Plug & Protect”-filosofia, joka tekee laitteesta helposti markkinoitavan hyvinkin erilaisiin IoT-käyttökohteisiin ja eri kohderyhmille.

## 6.2 Case: Autojen kyberturvallisuus (Checkoway ym. 2011)

Checkoway ym. ovat analysoineet kokeellisesti modernien autojen ulkoisia hyökkäyspintoja [35]. Nykyaikaisissa autoissa on tyypillisesti kymmeniä erillisiä mikroprosessoripohjaisia laitteita, jotka käytetään auton eri alijärjestelmien ohjaamiseen. Näihin lukeutuvat ajamisen kannalta olennaiset ydinjärjestelmät kuten moottorinohjaus tai lukkiutumaton jarrujärjestelmä, kuten myös ”mukavuuslaitteet” kuten mediatoistinjärjestelmä tai bluetooth-ominaisuudet. Laitteet vaihtavat tietoja keskenään yhteisen CAN-väylän avulla, joten saadessaan yhden alijärjestelmän haltuunsa, on mahdollista että hyökkääjä pystyy ajamaan mielivaltaista koodia missä tahansa auton tietojärjestelmän osassa. Teoriassa olisi siis mahdollista esimerkiksi tilanne, että työntämällä saastunut CD-levy auton soittimeen voitaisiin kytkeä jarrut pois toiminnasta.

Artikkelissa demonstroitiin useita esimerkkitapauksia, joissa tutkijat simuloivat erilaisia hyökkäystilanteita. Hyökkäystyypit jaettiin kategorioihin epäsuora fyysinen, lyhyen kantaman langaton yhteys tai pitkän kantaman langaton yhteys. Epäsuorasta fyysisestä on esimerkkinä edellä mainittu CD-hyökkäys. Social Engineering-lähestymistapaa käyttämällä voidaan hyökkäyksen uhri suostutella soittamaan saastunut CD-levy auton soittimessa. Näin tehdessään pahaa-aavistamaton kuljettaja ei havaitse tulleen huijatuksi, vaan olettaa ainoastaan CD:n olevan epäkunnossa. Saastuminen on kuitenkin tapahtunut ja autossa saattaakin piillä troijalainen odottamassa aktivoitumista sopivalla hetkellä.

Autot on nykyään laajasti varusteltu bluetooth-ominaisuuksilla, joiden kautta voidaan mm. hyödyntää älypuhelimien musiikkisoitinta auton äänentoistojärjestelmässä tai

käyttää hands-free ominaisuuksia puheluiden soittamiseen. Tutkijat havaitsivat kuitenkin, että nuuskimalla auton bluetoothin MAC-osoite on mahdollista muodostaa laitepari myös hyökkäystarkoituksessa ja ujuttaa autoon tätä kautta haitallista koodia. Laiteparin muodostaminen onnistuttiin tekemään takaportin kautta siten, että auton normaalien käyttöliittymien kautta paritettua laitetta ei ollut mahdollista nähdä eikä paritusta pystynyt poistamaan.

Viimeisenä ja kenties uhkaavimpana hyökkäyspintana artikkelissa arvioidaan mahdollisuutta ottaa autoja hallintaan internetin välityksellä. Yhä useammat autot ovat nykyään varustettu GSM/3G-moduulilla, jotka välittävät telematiikkatietoja autojen valmistajalle. Tutkijat testasivat, että moduulin gsm-numero voidaan helposti urkkia ja tämän jälkeen siihen soittaa tavallisella pöytäpuhelimella. Kun yhteys on luotu, toistetaan luuriin erityisesti tarkoitusta varten ”tohtoroitu” signaali. Tämän ”modeemiäänän” vastaanottavassa päässä oleva ohjelmistomodeemi sitten muuntaa binääreiksi, joka sisältää hyökkääjän haluaman haittakoodin.

Huomionarvoisena seikkana artikkelissa, on että tutkijat testasivat hyökkäyksiä vain yhteen spesifiin automalliin ja yksilöön, joka heillä oli jatkuvasti saatavilla laboratoriossaan. Näin heidän oli mahdollista rauhassa tutkia autojen alijärjestelmiä ja kehittää niihin sopivia hyökkäyksiä. Tämä vaati heiltä huomattavaa reverse engineering-osaamista kuten disassembler-ohjelmistojen käyttötaitoja, jolla yritettiin selvittää laitteiden firmwarejen toimintaa purkamalla konekielistä koodia. Tehtävää vaikeutti edelleen se, että eri laitteet eivät suinkaan olleet toteutettu yhdellä ja samalla mikroprosessorilla, vaan tulkittavia käskykantoja oli useita.

Hyökkäykset eivät onnistu ilman spesifiä tietotaitoa ja työkaluja, joten ne jäävät normaalin autovarkaan ja satunnaisen ”sunnuntaihakkerin” ulottumattomiin. On kuitenkin mahdollista visioida, kuten artikkelissakin todetaan, tilanteita jossa jokin rikollisryhmittymä tarjoaisi hakkerointia palveluna toisille maksua vastaan – hieman samaan tapaan kuin bottiverkkoja pystyy nykyään vuokraamaan palvelunestohyökkäysten tms. toteuttamiseen. Varkauksien lisäksi, arkaluonteisten keskustelujen salakuunteleminen (ja niistä hyötyminen taloudellisesti) hands-free järjestelmän kautta saattaisi muodostaa toisen potentiaalisen motiivin hyökkäyksille.

Artikkelin voisi kiteyttää siten, että siinä on demonstroitu vakavienkin hyökkäysten olevan mahdollisuuksia ja autojen kyberturvallisuudessa olevan erittäin paljon parantamisen varaa. Artikkelin herätti ilmestyessään laajaa julkista keskustelua ja tiettävästi mm. autoalan järjestöt kuten SAE ovat reagoineet ja erilaisia toimikuntia tilanteen parantamiseksi on perustettu [36]. Autoala on perinteisesti hyvin hidasliikkeinen ja kankea, joten nähtäväksi jää näemmekö vielä laajamittaista autoihin kohdistuvaa hyökkäystoimintaa ennen kuin haavoittuvuudet paikataan.

## 7. RAKENNUSPROJEKTI

2.11.2017

Kurssiin kuuluvan isomman rakennusprojektin aiheeksi valittiin yksinkertainen ajoneuvodatan loggaussovellus. Pääasiallisena perusteluna valinnalle on kirjoittajan osalta työelämän puolelta tuleva paine perehtyä kyseiseen sovellusalueeseen, joten aihealueena tämä tarjoaa mahdollisuuden lyödä useampi kärpänen yhdellä iskulla. Mutta toki myös siviilielämän helpottaminen monin tavoin on yhtäläillä taustalla, kuten toivon mukaan projektin aikana tullaan huomaamaan.

Nykyaikaisissa henkilöautoissa (sekä yhtäläillä raskaassa kalustossa) liikkuu monenlaista sensori- yms. dataa. Paitsi että auton eri alijärjestelmät kuten moottorinohjain, abs-jarrut, luistonesto, vakionopeudensäädin jne. ovat tästä datasta täysin riippuvaisia, voi niistä olla oikein käytettynä myös hyötyä auton käyttäjälle. Esimerkiksi polttoaineen kulutuksen, ajonopeuden sekä kaasun/vaihteiden käyttötavan keskinäistä yhteyttä voitaisiin tilastoida datasta ja tehdä näin analytiikkaa kuljettajan ajotavan vaikutuksesta polttoaineen kulutukseen. Uudemmissa autoissa, joissa renkaiden ilmanpaine on erikseen anturoitu, voidaan ohjelmoida logiikka antamaan hälytys kun ilmanpaine laskee alle määritellyn kynnyksrajan - näin renkaiden vierintävastus saadaan pidettyä aisoissa, ja polttoainetta säästyy. Jos ja kun järjestelmä on kytketty verkkoon, käyttömahdollisuudet laajenevat jälleen. Ohjelmallisesti voidaan esimerkiksi tarkkailla polttoaineen tasoa, sekä saavutettaessa kynnyksarvo etsiä automaattisesti kuljettajalle halvin tankkauspaikka nykyisen reitin varrelta (ainakin teoriassa, olettaen että saatavilla on sopiva rajapinta josta tiedot saadaan haalittua).

Ajoneuvojen data on tätä nykyä CAN-väylädataa (Controller Area Network) ja siihen päästään käsiksi esimerkiksi OBD-diagnostiikkaliitännän kautta. Yleisin OBD:n käyttötarkoitus liittyy epäilemättä erilaisten vikakoodien lukemiseen ja nollaamiseen, mutta todellisuudessa liitännä mahdollistaa kaiken CAN-viestiliikenteen lukemisen reaaliajassa. Tämä johtuu siitä, että CAN on osoitteeton protokolla eli kaikki sanomat välitetään verkon kaikille solmuille. Jokaisen solmun tehtävänä on sanomatunnusten perusteella päätellä, tekeekö se yksittäisellä tiedolla jotain vaiko ei. [16]

Rakennusprojekti päätettiin toteuttaa Raspberry PI 3 –alustalle helpon ohjelmoitavuuden ja hyvien liitäntöjen ansiosta. OBD-väylästä data saadaan ELM 327-lukijalla, joka kytketään bluetoothin yli Raspberryyn. Laitteeseen ohjelmoidaan logiikka CAN-viestien dekodaukseen sekä tallentamiseen ihmisluettavassa muodossa. Muihin vaatimuksiin kuuluu mm. virranhallinta, sillä järjestelmän on osattava ajaa itsensä hallitusti alas kun virransyöttö menetetään esim. auton sytytysvirran sammuttamisen seurauksena.

Toistaiseksi on vielä ratkaisematta, tarvitaanko esim. akkua turvaamaan välivaiheen virransaanti.

Projektin toteutus tulee tapahtumaan vaiheittain siten, että aloitetaan perustoiminnallisuudesta ja edetään asteittain kohti vaativampia ominaisuuksia, niin kauan kuin aika ja muut resurssit antavat myöden. Alustava vaihejako on seuraavanlainen:

1. Peruskonfiguraatio: Virransyöttö 12VDC → 5VDC usb muuntajalta, bluetooth yhteys OBD lukijaan (konfigurointi ja testaus), CAN-sanomien lukeminen ja dekodaus sekä tallennus muistikortille.
2. ”Älykäs” virranhallinta – laitteen tulisi sammuttaa itsensä hallitusti kun virransyöttö menetetään – toteutustapa vielä avoin
3. Yhteydet ulkomaailmaan. Toteutetaan esim. älypuhelin-hotspotin kautta internet-yhteys ja pystytetään http-palvelin (vast) joka mahdollistaa (raaka)datan lukemisen etänä.
4. Siirrytään käyttämään nettiyhteyden muodostamisessa dedikoitua GPRS-moduulia (ks. esim. [17]) , jolloin laitteelle saadaan oma SIM-kortti eikä olla riippuvaisia matkapuhelimen jakamasta wifi-yhteydestä.
5. Tietojen automaattinen synkronointi/varmuuskopiointi pilvipalveluun tms.
6. Jatkojalostetaan nettiin päin tarjottavaa näkymää: analytiikka (polttoaineenkulutus, ajotapa), ulkoasu, HTML, CSS...

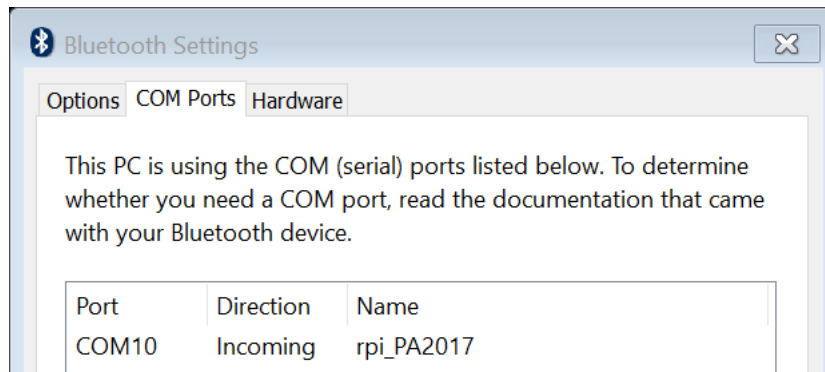
Projektille budjettikatoksi on määritelty 100 euroa. Tarvittava bluetooth-OBD lukija on olemassa jo valmiina, kuten myös tarvittavat sähkö/USB-johdot sekä auton tupakansytytinmuuntaja. Raspberry PI 3 – yksikkö tilattiin CDON.comista. Käyttökohdetta ajatellen kotelo kokoonpanolle on välttämätön hankinta, kuten tietysti myös 16GB microSD-muistikortti käyttöjärjestelmän ja ohjelmalogiikan tallennusta varten. Toistaiseksi hankinnoista on kustannuksia syntynyt 59,70 €. Mikäli myöhemmin päätetään hankkia GPRS-moduuli, syntyy tästä lisäkustannuksia, jotka voivat pistää budjetin ahtaalle.

#### 4. – 5.11.2017

Projektin alkuvaiheessa syntyi ajatus, että olisi kätevää kyetä simuloimaan ELM327-bluetooth rajapinnan tarjoamaa viestiliikennettä. Tällöin ei olla sidoksissa fyysisesti autoon, vaan sovelluskehitystä voidaan tehdä mukavasti sisätiloissa, mikä helpottaa etenkin alkuvaiheen kokeiluja. Tarkoitusta varten löytyikin Windows-alustalle tehty obdSIM [18]. Kyseinen näppärä pikkuohjelma kykenee emuloimaan useammankin eri CAN-protokollan mukaista viestiliikennettä ja välittämään sitä bluetooth-sarjayhteyden yli vastaanottavalle laitteelle, tässä tapauksessa siis Raspberryille. Vastaanottavassa päässä tietojen käsittelyyn valittiin Python OBD-kirjasto [19], joka on vaivatonta ottaa käyttöön Raspbianin natiivin Python-tuen ansiosta.

Emuloinnin toteutus pääpiirteittäin Windows 10-ympäristössä:

1. Asennetaan ObdSIM ja asetetaan Windows 'Test Modeen'
2. Bluetooth-asetuksista luodaan virtuaalinen COM-portti, joka välittää bluetoothin yli tiedot raspberryyn (kuva)



3. Käynnistetään simulaatio komentoriviltä ohjaten liikenne oikeaan porttiin

```
obdsim.exe -w COM10
```

4. Ellei ole jo, luodaan laitepari PC:n ja rasperryn välille bluetoothctl työkalulla tai graafisen UI:n kautta:

```
sudo apt-get install blueman
```

5. Asetetaan Raspberry kuuntelemaan sarjaporttiliikennettä bluetoothin kautta

```
sudo rfcomm bind /dev/rfcomm0 XX:XX:XX:XX:XX:XX
```

Missä XX... on lähettävän laitteen Bluetooth-osoite.

6. Asennetaan Python OBD ja tarvittaessa tätä ennen PIP

```
#Asenna pip: https://pip.pypa.io/en/latest/installing/
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
sudo python get-pip.py
```

```
#Asenna python-obd
#http://python-obd.readthedocs.io/en/latest/
pip install obd
```

7. Testataan yhteys Python OBD:llä

```
import obd

#Parametreina asetettu portti ja baud rate
#Portin paikalle None → automaattinen haku
connection = obd.OBD('/dev/rfcomm0', 9600)

cmd = obd.commands.SPEED # select an OBD command (sensor)
```

```
response = connection.query(cmd) # send the command, and parse the re-
sponse

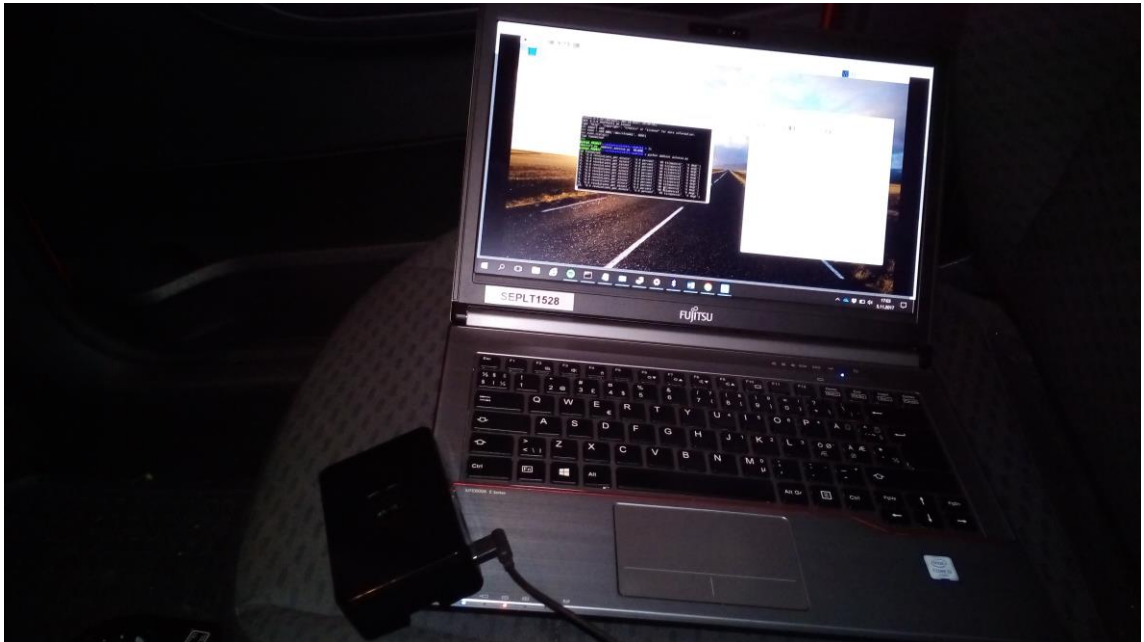
print(response.value) # returns unit-bearing values thanks to Pint
print(response.value.to("mph")) # user-friendly unit conversions
```

Jos kaikki meni hyvin niin yhteys on nyt luotu ja tietoja pystyy lukemaan erittäin mukavalla ja helppokäyttöisellä Python-syntaksilla, josta löytyy lisätietoa Python OBD kirjaston dokumentaatiosta [19]. Koska linux ja bluetooth on yleisesti ottaen hieman kenkkumainen yhdistelmä, kaikki ei kuitenkaan välttämättä mene hyvin. Tällöin vianselvittelyyn kannattaa varautua käyttämään 0 – 8 tuntia aikaa. Lisätietoa ja keskustelua aihepiiristä esim. lähteistä [20][21][22].

6.11.2017

Testattiin bluetooth-yhteyden toimivuus ja datalogging-perusominaisuudet autossa. ELM327-tökkeli OBD-väylään, raspberry ottaa virran tupakansytyttimestä ja ohjaus läppäriltä VNC-pilviyhteyden yli. Matkapuhelin-hotspot vaaditaan yhteyksien jakamista varten, koska dedikoitua gsm-moduulia ei tässä vaiheessa vielä ole Raspissa. Tiedot tallentuvat csv-formaattiin onnistuneesti, mutta valitettavan moni PythonOBD:n funktiokutsu tuottaa pelkkiä Null-arvoja ECU:ltä. Mm. polttoaineen kulutustietoa ei vielä saada onnistuneesti.





11.11.2017

Siirryttiin käyttämään MySQL-alustaa tietojen tallennukseen csv-tiedostojen sijaan. Asennettiin Raspiin LAMP-konfiguraatio (Apache, MySQL, PHP) jonka päälle voidaan toivottavasti tulevaisuudessa toteuttaa myös sovellukseen kuuluva web-reaaliaikanäkymä. Tällä hetkellä (simulaatio)datan tarkastelu onnistuu kotiverkossa phpMyAdminin kautta.

192.168.0.11:3000/phpmyadmin/sql.php?db=obdlogger&token=79b758610d960492...

Server: localhost:3306 » Database: obdlogger » Table: Entry

Browse Structure SQL Search Insert Export More

Show all Number of rows: 25 Filter rows: Search this table Sort by key:

+ Options

	EntryID	Timestamp	DeviceID	RPM	Calc_load	Sp
<input type="checkbox"/> Edit Copy Delete	1	2017-11-11 19:02:28		2003	0	
<input type="checkbox"/> Edit Copy Delete	2	2017-11-11 19:02:36		560	0	
<input type="checkbox"/> Edit Copy Delete	3	2017-11-11 19:02:46		1448	0	
<input type="checkbox"/> Edit Copy Delete	4	2017-11-11 19:02:57		3500	0	
<input type="checkbox"/> Edit Copy Delete	6	2017-11-11 19:54:31	1	3308	24.7058823529	
<input type="checkbox"/> Edit Copy Delete	7	2017-11-11 19:54:36	1	3891.75	42.7450980392	
<input type="checkbox"/> Edit Copy Delete	8	2017-11-11 19:54:42	1	4377	60.7843137255	

Teoriassa olisi mahdollista toteuttaa porttiohjaus, jotta web-palvelimen tarjoamat rajapinnat olisivat käytettävissä myös oman verkon ulkopuolelta. Tätä testattiin kaapelilaajakaistayhteydellä varustetussa kotiverkossa onnistuneesti, mutta kun samaa

yrityttiin puhelimesta jaetulla hotspot-yhteydellä ei sitä saatu toimimaan. Liikkuvasta autosta yhteyksien toteuttaminen järkevästi on siis edelleen hieman kysymysmerkki.

Ohje yms-matskua vastaisuuden varalle:

Apache/mysql käyttöönottoon linuxissa [23].

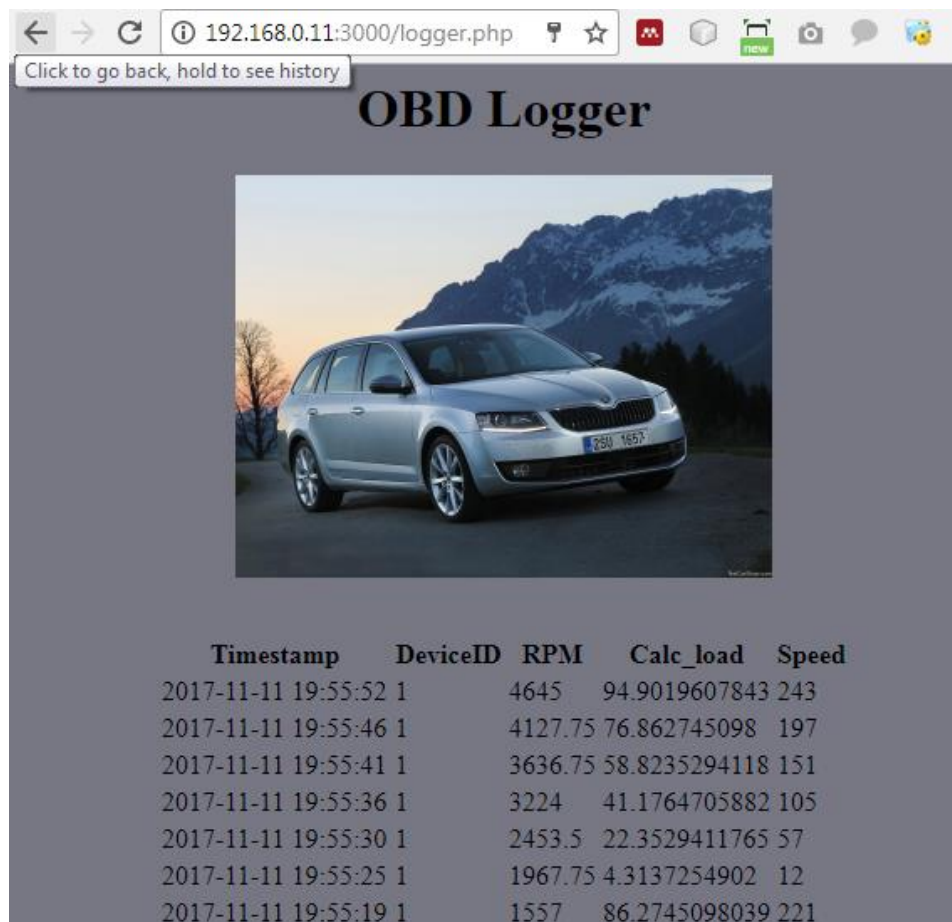
Python rajapinta MySQL käyttöön [24].

12.11.2017

Ensimmäinen viritelmä fronttinäkymästä kyhätty sisältäen kirjautumistoiminnon sekä tietokannan tietojen näyttämisen. Tällä hetkellä näytetään 15 uusinta tietoa kannasta kyselyllä:

”SELECT \* FROM Entry ORDER BY Timestamp DESC LIMIT 15”

Käytettyjen teknologioiden avainsanoja mm. html, php sekä erityisesti tietokantayhteyden luomiseen käytetty mysqli-kirjasto [25][26]. Ulkoasussa ja toiminnallisuudessa riittää vielä viilattavaa.



Timestamp	DeviceID	RPM	Calc_load	Speed
2017-11-11 19:55:52	1	4645	94.9019607843	243
2017-11-11 19:55:46	1	4127.75	76.862745098	197
2017-11-11 19:55:41	1	3636.75	58.8235294118	151
2017-11-11 19:55:36	1	3224	41.1764705882	105
2017-11-11 19:55:30	1	2453.5	22.3529411765	57
2017-11-11 19:55:25	1	1967.75	4.3137254902	12
2017-11-11 19:55:19	1	1557	86.2745098039	221

GPS-moduuli laitettu tilaukseen hintaan 21,80 sis. toimituskulut [27].



15.11.2017

Paranneltu web-näkymää, staattisen kuvan tilalla nyt google maps-upote joka näyttää autosta viimeisen tunnetun sijainnin. Tietokantaskeema ja tiedonkeruupään koodi muutettu vastaamaan tätä (lisätty Entry-tauluun uudet sarakkeet gps-koordinaateille). Sijaintitiedot ovat toistaiseksi simuloituja, mutta myöhemmin on tarkoitus lisätä tietojen lukeminen oikeasta GPS/GLONASS vastaanottimesta. Lisätty myös alkeellinen kirjautumisenhallinta php-sessioilla.

Loggauspäähän tehty myös hiukan muutoksia. Skripti kykenee nyt sammuttamaan raspberrystä virrat kun halutut ehdot täyttyvät – esim. yhteys bluetooth-lukijaan menetetään tai jokin kanava (esim. moottorin kierrosluku) tuottaa tarpeeksi pitkään pelkkiä 0-arvoja. Kehitysvaiheessa sammutus kannattaa pitää kuitenkin pois päältä.

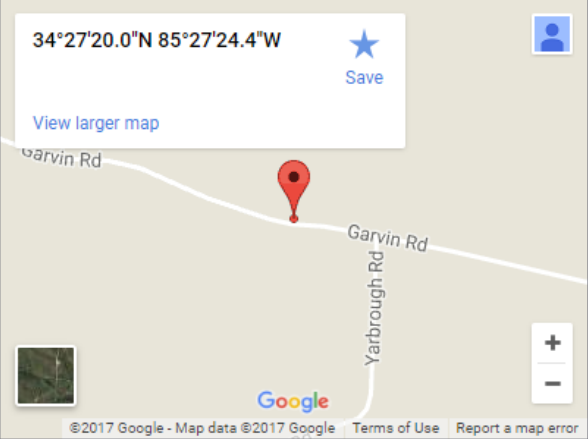
212.149.236.220:3000/logger.php

## OBD Logger

34°27'20.0"N 85°27'24.4"W

[Save](#)

[View larger map](#)



©2017 Google - Map data ©2017 Google Terms of Use Report a map error

Timestamp	DeviceID	GPS_Lat	GPS_Long	RPM	Calc_load	Speed
2017-11-15 11:56:50		34.455555	-85.456789			
2017-11-15 11:32:51	1	60.452513	22.222222	0	0	0
2017-11-15 11:32:47	1	60.452513	22.222222	0	0	0
2017-11-11 19:55:52	1	0	0	4645	94.9019607843	243
2017-11-11 19:55:46	1	0	0	4127.75	76.862745098	197
2017-11-11 19:55:41	1	0	0	3636.75	58.8235294118	151
2017-11-11 19:55:36	1	0	0	3224	41.1764705882	105
2017-11-11 19:55:30	1	0	0	2453.5	22.3529411765	57
2017-11-11 19:55:25	1	0	0	1967.75	4.3137254902	12
2017-11-11 19:55:19	1	0	0	1557	86.2745098039	221
2017-11-11 19:55:14	1	0	0	1142.5	68.6274509804	175
2017-11-11 19:55:09	1	0	0	682.75	50.5882352941	130
2017-11-11 19:55:03	1	0	0	6263.5	32.5490196078	84
2017-11-11 19:54:58	1	0	0	5828.25	14.9019607843	39
2017-11-11 19:54:52	1	0	0	5336.5	96.862745098	248

[Log out](#)

Käytännön kehitystyö VNC-etätyöpöydällä on etenkin pilviyhteyden kautta tökkivää ja etenkin skriptien kirjoittelu voi viiveen takia olla melkoista tuskaa. Onneksi graafista käyttöliittymää linuxissa tarvitaan harvoin, joten on olemassa parempi ratkaisu. SSH-yhteys ei ole oletuksena päällä raspberryssä, ja se saadaan käyttöön laitteen asetuksista komennolla `sudo raspi-config`. Lisäksi tarvitaan muutamia muita ennakkovalmisteluja ja säätöjä, joihin tutustuminen kannattaa aloittaa tiedostosta `/etc/ssh/sshd_config`. Autentikointimenetelmäksi lienee ainakin kehitysvaiheessa luontevin vaihtoehto perinteinen käyttäjätunnus/salasanakirjautuminen. Tunnukset ovat samat, kuin fyysiselle laitteelle suoraan kirjautuessa. SSH ei oletuksena myöskään huoli tyhjää salasanaa, joten mikäli kirjautuminen ei onnistu, kannattaa ensimmäisenä varmistaa että käyttäjätunnus on olemassa ja että salasana on asetettu. Myös `/var/log/auth.log` saattaa olla kätevä työkalu kirjautumisongelmien selvittelyyn.

Kun SSH on asennettu ja kofiguroitu onnistuneesti, komentojen ajaminen Raspberryssä etänä onnistuu windows-maailmassa esim. Puttylla [29] tai vaikka android-älypuhelimella JuiceSSH-clientilla [30]. Enää ei tarvitse siis välttämättä kantaa edes läppäriä mukana kenttätesteissä, vaikka toki laajoja skriptejä puhelimella työstäessä alkaa aika nopeasti kaivata kunnollista näppäimistöä ja näyttöä.

*16.11.2017*

Päästiin testaamaan juuri postissa saapunutta GPS-vastaanotinta, joten ei muuta kuin USB-tökkeli kiinni raspberryyn.



Aivan ensiksi kannattaa komennolla `lsusb` testata että laite tunnistetaan onnistuneesti. Tämän jälkeen asennettavia juttuja ovat ainakin `gpsd` [31] sekä pythoniin `gps3`-kirjasto [32], joka käyttää ensiksi mainittua. GPS-vastaanottimen toiminnan voi testata komennolla `cgps -s`, jonka pitäisi alkaa pukkaamaan sijaintitietoja ruudulle.



tankkaamaan sekä poikettiinpa vielä testimielessä munkkikahveille paikalliseen sotilaskotiin. Koko tämän ajan loggaus jatkui suunnitellusti siitä huolimatta, että moottori oli välillä sammuksissa. Sen sijaan klo 16.28, kun oltiin jo Yläneellä, tiedonkeruu keskeytyi jälleen täysin yllättäen.

Keskeytymisten tarkkaa syytä ei ole vielä päästy tutkimaan, mutta vahva epäily kohdistuu nimenomaan gps-signaalin luotettavuuteen ja siihen, että koodi ei osaa vielä käsitellä näitä tilanteita asianmukaisesti.

*17.11.2017*

Otettiin merkittävä askel kohti kerrosarkkitehtuuria, Apache/MySQL-palvelin pyörii nyt Raspberryn sijasta kiinteällä laajakaistayhteydellä varustetussa kotikoneessa. Tämä mahdollistaa tietokantakyselyiden tekemisen sekä web-näkymän tarkastelun myös silloin kun raspberryllä ei ole verkkoyhteyttä tai virransyötön kanssa on ongelmia (erittäin yleinen ongelma mobiileissa IoT-laitteissa). Tarkalleen palvelin on nyt toteutettu Windows 7-pöytäkoneella pyörivästä bitnamin LAMP-virtuaalikoneesta, jonka verkkoyhteys konfiguroitiin sillattuun moodiin siten että serveriin pääsee internetistä käsiksi. Tämä vaatii myös porttiohjauksen toteuttamisen reitittimen asetuksista. Tämän jälkeen tietojen katselu onnistuu mistä tahansa, kunhan reitittimen julkinen IP sekä avoin portti on tiedossa. Pitkäaikaiseen käyttöön kyseistä viritystä tuskin kuitenkaan voidaan suositella mm. turvallisuusnäkökohtien vuoksi.

Polttoaineen kulutustiedon saaminen suoraan OBD:ltä on osoittautunut haastavaksi. Python-OBd:ssä on hetkellistä kulutusta varten kyllä oma kysely `obd.commands.FUEL_RATE`, jonka pitäisi teoriassa antaa polttoaineen massavirta yksikössä g/s. Kumpikaan projektin aikien testatuista autoista ei kuitenkaan ole tukenut tätä komentoa suoraan. Tässä vaiheessa projekti on kestänyt jo niin kauan, että autokin on ehtinyt vaihtua välissä vm. -11 Fabiasta vuoden 2014 farmari-Octaviaan. Uudesta autosta saadaan sentään ulos polttoainetankin taso prosentteina sekä ajetut kilometrit, joista kulutuksen periaatteessa voisi laskea, kun tunnetaan polttonestesäiliön kokonaistilavuus. Laskenta tosin vaatisi vielä huomattavaa kehitystyötä sillä tankkaukset pitäisi jotenkin kyetä ottamaan huomioon ja tällöinkin tarkkuuden kanssa voisi olla vähän niin ja näin. Kyseiset tiedot on nyt joka tapauksessa lisätty mukaan tiedonkeruuseen.

Tarkemmin ajatellen tietokanta olisi kyllä verkkoyhteysongelmien varalta syytä kahdentaa siten, että luetut tiedot rekisteröityvät myös etäserverin lisäksi Raspiin paikallisesti. Tietojen synkroonissa pitäminen olisi kuitenkin kokonaan oma harjoituksensa, mihin tuskin ehditään lähitulevaisuudessa paneutua.

18.11.2017

Projektin kehitystyö keskeytetään toistaiseksi. Tähän mennessä on saatu aikaan toimiva prototyyppi IoT-sovelluksesta, jollaisia tiettävästi on jo myös kaupallisessa käytössä. Kaikki perustoiminnallisuudet kuten tietojen lukeminen OBD:stä, tallennus tietokantaan sekä tarkastelu web-käyttöliittymästä ovat toteutettuna sekä testattu todellisissa käyttöolosuhteissa. Poikkeustilanteita lukuunottamatta, järjestelmä on toiminut jopa hämmästyttävän hyvin. Selkeitä kehityskohteita onkin juuri vahvemman poikkeuskäsittelyn sekä virhetilanteesta toipumisen toteuttaminen. Tällä hetkellä loggausskripti joko kaatuu tai ajautuu ikuisen silmukkaan, mikäli esim. yhteys tietokantaan tai GPS-signaali menetetään. Nämä eivät luonnollisestikaan ole IoT-laitteelle toivottuja ominaisuuksia.

Laitteen tulisi myös optimitalanteessa toimia mahdollisimman autonomisesti siten, että tietoja tallennetaan silloin kun sille on edellytykset olemassa ja kun se on ylipäättään järkevää. Tällä hetkellä loggauksen aloitus tapahtuu manuaalisesti, kuten myös lopetus, ellei suoritus sitten keskeydy virheeseen. Optimitalanteessa laite menisi jonkinlaiseen ”virransäästötilaan”, kun moottori on ollut sammuksissa ajan x, ja käynnistyisi jälleen uudestaan kun se havaitsee että väylässä alkaa liikkua mielenkiintoista dataa. Tällöin laite voitaisiin luultavasti huoletta jättää autoon pidemmäksikin aikaa tietoja tallentamaan.

Kolmas toistaiseksi ratkaisematon haaste on rasperryn mobiili verkkoyhteys, joka toistaiseksi on toteutettu matkapuhelimesta hotspottaamalla, mikä on mahdollistanut myös etätyöpöytä- sekä SSH-käytön langattoman lähiverkon yli. Optimitalanteessa raspissa tulisi olla kuitenkin joko oma integroitu gsm-moduuli tai USB-väylään liitettävä 3/4g-”mökkula”, jolla yhteys muodostetaan. SSH:n käyttömahdollisuus julkisen IP-osoitteen yli olisi ehdoton plussa, joten ratkaisun tulisi optimaalisesti olla sellainen joka tukee tätä. Markkinoilla on olemassa erityisesti ajoneuvokäyttöön tarkoitettuja SIM-korttipaikalla varustettuja reitittimiä, jotka voisivat ajaa asian, ks. esim [33]. Tällaisen hankkiminen harjoitustyötä varten olisi kuitenkin hiukan yliampuvaa.

Projektin lähdekoodit löytyvät repositoriosta <https://github.com/donkkis/sulautetut2017s/tree/master/obdPIPA> .

## LIITE 1

Viitteellinen tuntikirjanpito tähän astisista tehtävistä:

Pvm	Tunnit	Osio(t)	Tehtävät
9.9.2017	4	Käyttöjärjestelmät	Virtuaalikoneet, Linux, Python, C++
10.9.2017	8	Käyttöjärjestelmät, Ohjelmointi	GitHub perusteet + linux integraatio, Virtualisointi rautatasolla, Linux Terminal, BitNami LAMP, SSH, mySQL/phpMyAdmin
11.9.2017	4	Ohjelmointi	C++ perus
12.9.2017	6	Ohjelmointi	C++ oliot, Python 3
20.9.2017	4	Arduino	Harjoitukset ja niiden dokumentointi
21.9.2017	3	Arduino	Harjoitukset ja niiden dokumentointi
28.9.2017	2	Arduino	Harjoitukset ja niiden dokumentointi
4.10.2017	4	Raspberry PI	Yleistä luottokorttikoneista, perusasetukset, sense hat dataloggaus
11.10.2017	4	Raspberry PI	Kameramoduulin käyttö, vnc-etätyöpöytä, www-palvelin, bash-skriptaus/crontab linuxissa
12.10.2017	1	Raspberry PI	Dokumentointi
26.10.2017	6	Rakennusprojekti	Suunnittelu
2.11.2017	1	IoT	Kirjallisuustutkimuksen aineistoon alustava tutustuminen ja aiheen valinta.
4.11.2017	8	Rakennusprojekti	Kehitystyö
6.11.2017	6	Rakennusprojekti	Kehitystyö
11.11.2017	6	Rakennusprojekti	Kehitystyö
12.11.2017	4	Rakennusprojekti	Kehitystyö

15.11.2017	8	Rakennusprojekti	Kehitystyö
16.11.2017	8	Rakennusprojekti	Kehitystyö
17.11.2017	8	Rakennusprojekti	Kehitystyö
18.11.2017	2	Rakennusprojekti	Dokumentointi
23.11.2017	2	IoT	Kirjallisuustutkimus
29.11.2017	2	IoT	Kirjallisuustutkimus
30.11.2017	2	IoT	Kirjallisuustutkimus



## LÄHTEET

- [1] NotepadQQ-ohjelmiston kotisivut. Viitattu 10.9.2017  
<http://notepadqq.altervista.org/wp/download/>
- [2] GitHub Guides Hello World. Viitattu 10.9.2017 <https://guides.github.com/activities/hello-world/>
- [3] GitHub-tili. Viitattu 10.9.2017 <https://github.com/donkkis/>
- [4] Bitnami PhpMyAdmin. Bitnami Docs. Viitattu 10.9.2017 [https://docs.bitnami.com/virtual-machine/components/phpsudmyadmin/?utm\\_source=bitnami&utm\\_medium=virtualmachine&utm\\_campaign=Virtual%2BMachine](https://docs.bitnami.com/virtual-machine/components/phpsudmyadmin/?utm_source=bitnami&utm_medium=virtualmachine&utm_campaign=Virtual%2BMachine)
- [5] Frequently asked questions for virtual machines. Bitnami Docs. Viitattu 10.9.2017 <https://docs.bitnami.com/virtual-machine/faq/#connecting-with-an-ssh-client-on-windows>
- [6] Verkkokeskustelu Bitnami communityssä. Viitattu 10.9.2017  
<https://community.bitnami.com/t/accessing-phpmyadmin-remotely-i-get-this-security-error-for-security-reasons-this-url-is-only-accessible-using-localhost-127-0-0-1-as-the-hostname/6718>
- [7] Embedded With LAMP. Käyttäjän miksa007 projekti GitHub-palvelussa. Viitattu 10.9.2017 [https://github.com/miksa007/Embedded\\_with\\_LAMP](https://github.com/miksa007/Embedded_with_LAMP)
- [8] Viope. Olio-ohjelmoinnin perusteet C++ -kielellä. Kurssimateriaali.
- [9] <https://www.arduino.cc/en/Guide/Introduction>
- [10] <https://en.wikipedia.org/wiki/Theremin>
- [11] <https://www.arduino.cc/en/Tutorial/tonePitchFollower>
- [12] Luottokortin kokoiset monitaiturit. Mikrobitti Elokuu 2016.
- [13] [https://www.raspberrypi.org/magpi-issues/Essentials\\_Camera\\_v1.pdf](https://www.raspberrypi.org/magpi-issues/Essentials_Camera_v1.pdf)
- [14] <https://www.raspberrypi.org/documentation/remote-access/vnc/>
- [15] <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>
- [16] <https://fi.wikipedia.org/wiki/CAN-v%C3%A4yl%C3%A4>
- [17] [http://ihmevekotin.fi/product/577\\_raspberry-pi-gsm-hat](http://ihmevekotin.fi/product/577_raspberry-pi-gsm-hat)

- [18] <https://icculus.org/obdgpslogger/obdsim.html>
- [19] <http://python-obd.readthedocs.io/en/latest/>
- [20] <https://www.raspberrypi.org/forums/viewtopic.php?t=138145>
- [21] <https://www.raspberrypi.org/forums/viewtopic.php?t=78838>
- [22] <https://www.raspberrypi.org/forums/viewtopic.php?p=521067#p521067>
- [23] <https://pchelp.ricmedia.com/setup-lamp-server-raspberry-pi-3-complete-diy-guide/3/>
- [24] <https://dev.mysql.com/doc/connector-python/en/>
- [25] <http://php.net/manual/en/book.mysqli.php>
- [26] [https://www.w3schools.com/php/php\\_mysql\\_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp)
- [27] [http://ihmevekotin.fi/product/831\\_gpsglonass-vastaanotin-usb-porttiin-u-blox7](http://ihmevekotin.fi/product/831_gpsglonass-vastaanotin-usb-porttiin-u-blox7)
- [28] <https://www.raspberrypi.org/documentation/remote-access/ssh/>
- [29] <http://www.putty.org>
- [30] <https://play.google.com/store/apps/details?id=com.sonelli.juicessh&hl=fi>
- [31] <http://www.catb.org/gpsd/installation.html>
- [32] <https://pypi.python.org/pypi/gps3/>
- [33] <https://teltonika.lt/product/rut950/>
- [34] Sforzin, A. & Conti, M. 2016. RPiDS: Raspberry Pi IDS A Fruitful Intrusion Detection System for IoT
- [35] Checkoway, S., McCoy D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces.
- [36] <https://www.youtube.com/watch?v=bHfOziIwXic>