



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
LÄHTEET	31

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.

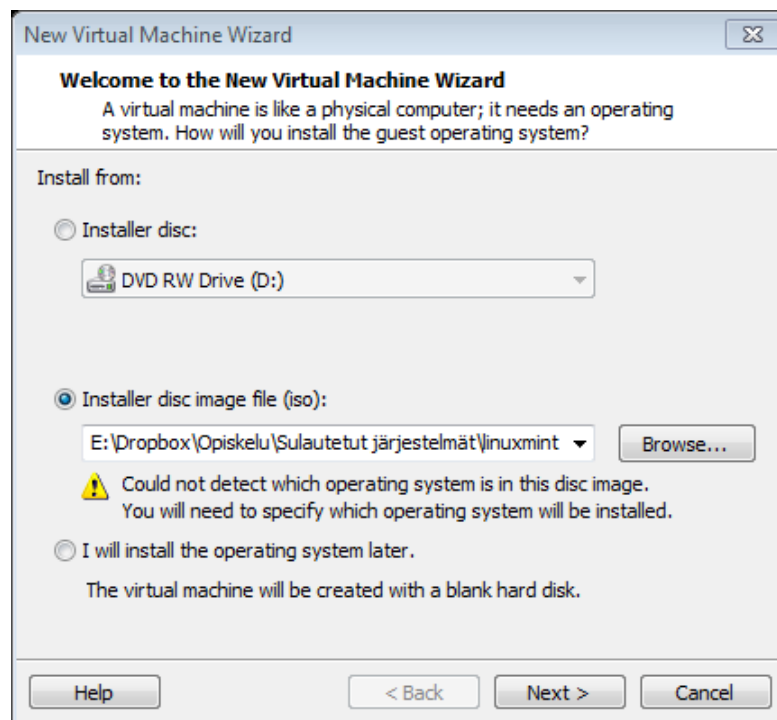
Oppimispäiväkirja on nimensä mukaisesti muodoltaan päiväkirjamainen, tarkoittaen sitä, että kerran kirjoitettuja merkintöjä ei jälkikäteen ole juurikaan palattu editoimaan. Tekstin koherenttius kärsinee tästä hieman, mutta etuna saavutetaan paras mahdollinen kuvaus nimenomaan oppimisprosessista, mihin 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

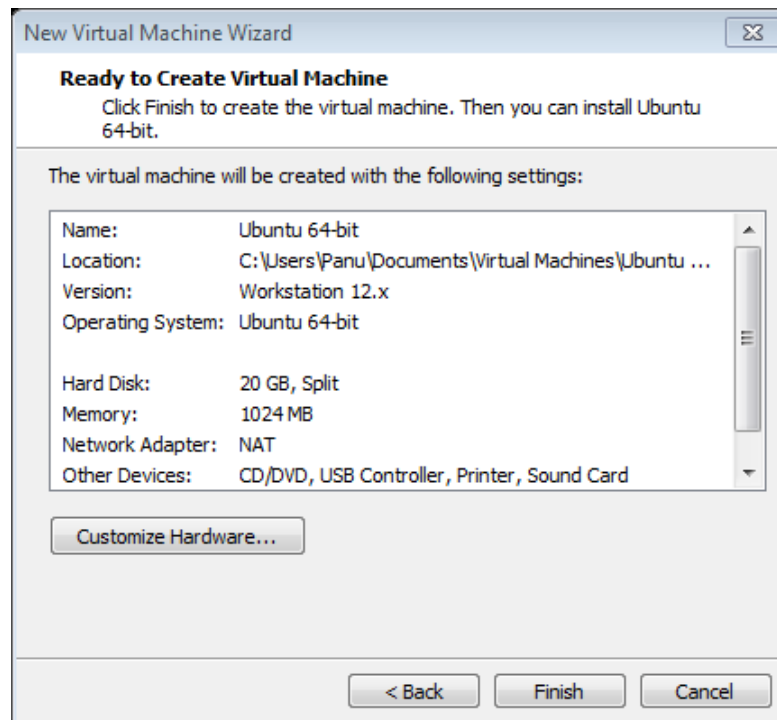
9.9.2017 – 10.9.2017

Tutustuttiin virtuaalikoneiden mahdollisuuksiin sekä Linux-käyttöjärjestelmän asennukseen sekä peruskäyttöön vMWare-virtualisointiympäristössä. Linux-distroksi valittiin Mint, jonka iso-levykuva ladattiin 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.



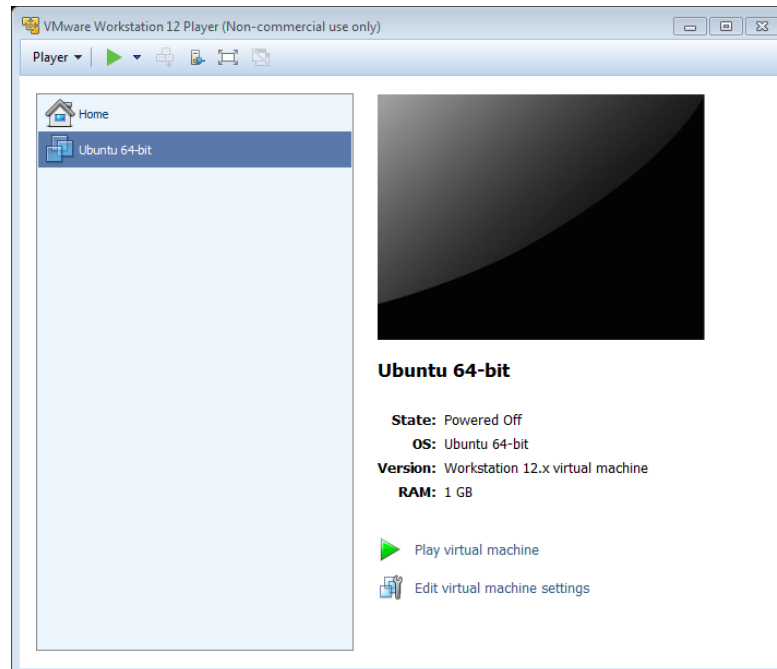
Kuva 1. Virtuaalikoneen luontityökalu

Seuraavana piti valita asennettavan Linuxin tyyppi. Listalla ei ollut valittavana Mintiä, mutta Mintin dokumentaatiosta löytyi tieto että se perustuu Ubuntun versioon 16.04. Niinpä tässä kohdassa valittiin Linux-distroksi 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

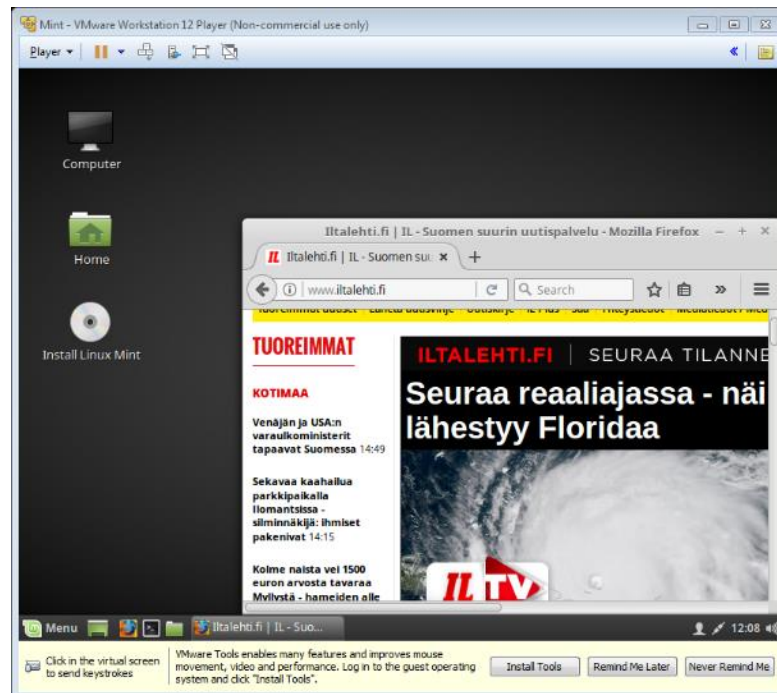
Näin oli saatu virtuaalikone luotua ja oli aika pistää kone käyntiin Play virtual machine-painikkeesta.



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 asennuk-

sen 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.

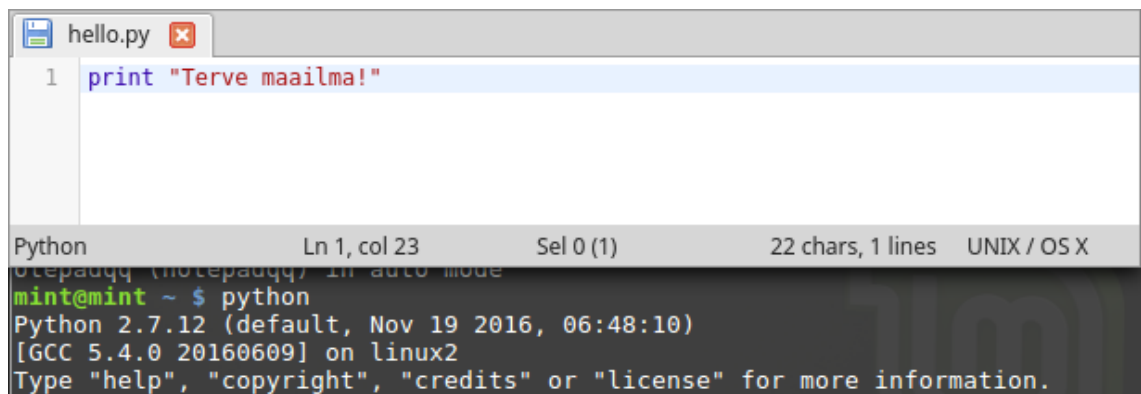
```

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]

```

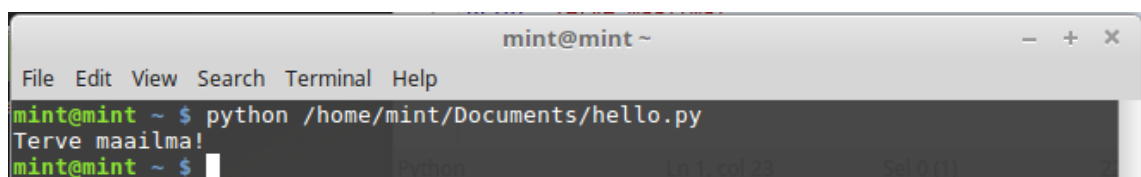
Kuva 5. Tekstieditorin asennusta terminaalin kautta

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 kohdassa tarvitsisi huolehtia.



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

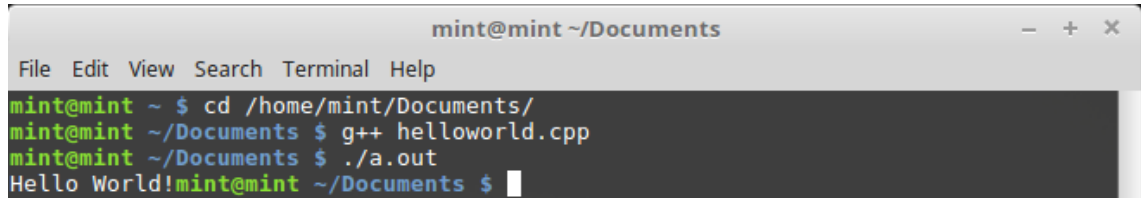
Koodi ajettiin komentoriviltä windows-ympäristöstäkin tutun oloisella python-syntaksilla:



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.



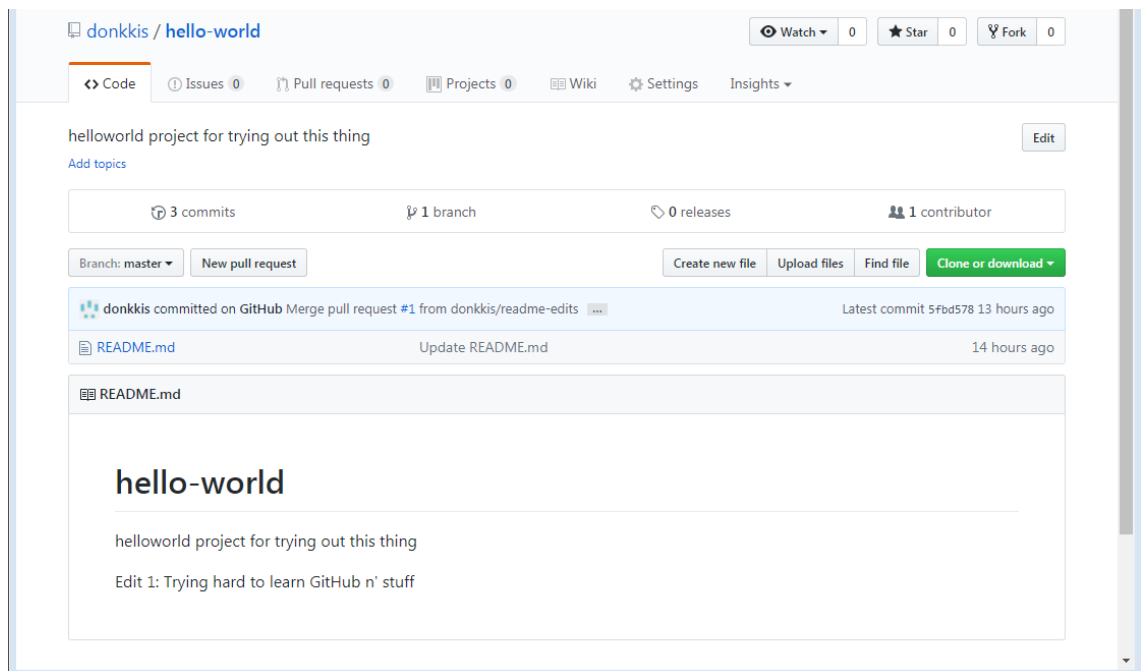
```

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
```

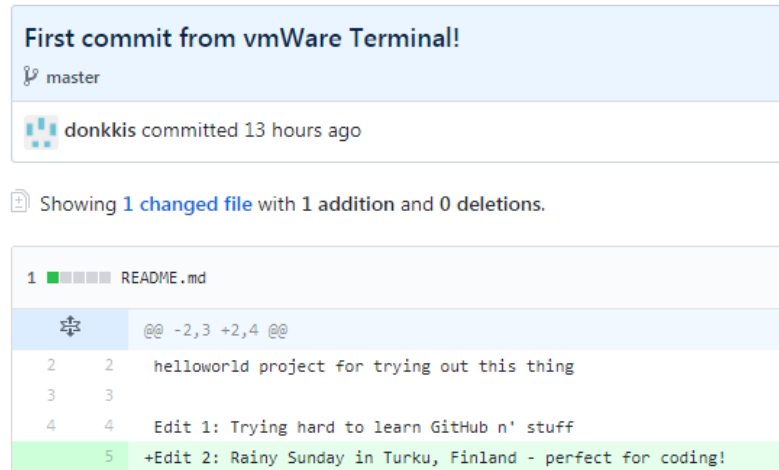
```
//Käytiin tekemässä muutoksia repon, esim. README-tiedoston sisältöön...
```



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

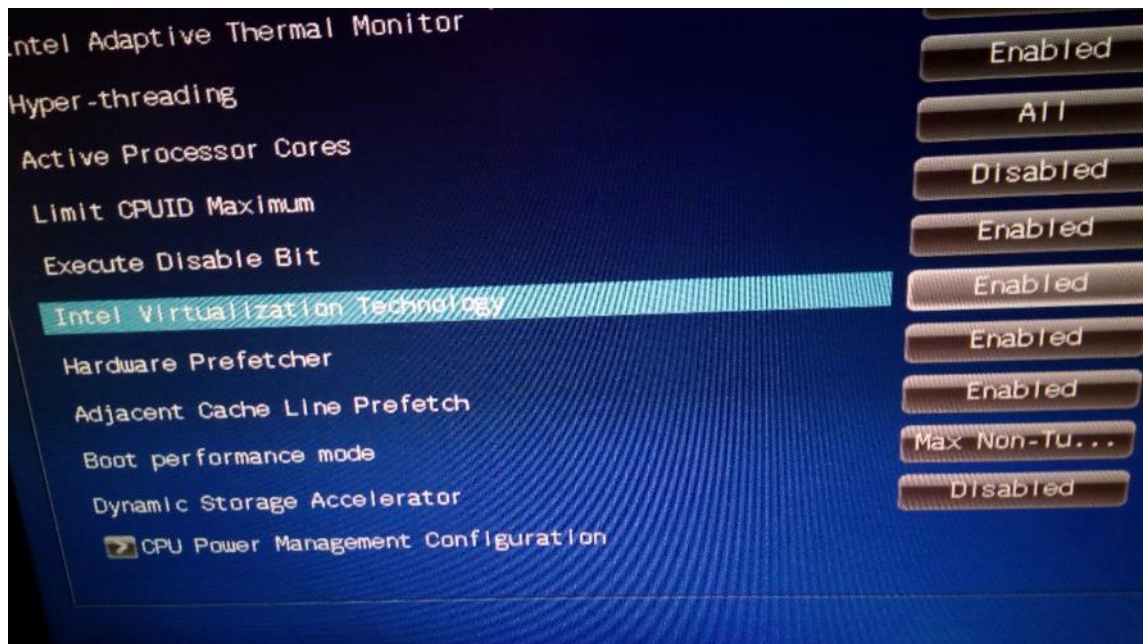
//Kysyy gh-tunnuksia, syötä käyttis ja passu kysyttäessä

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 .".



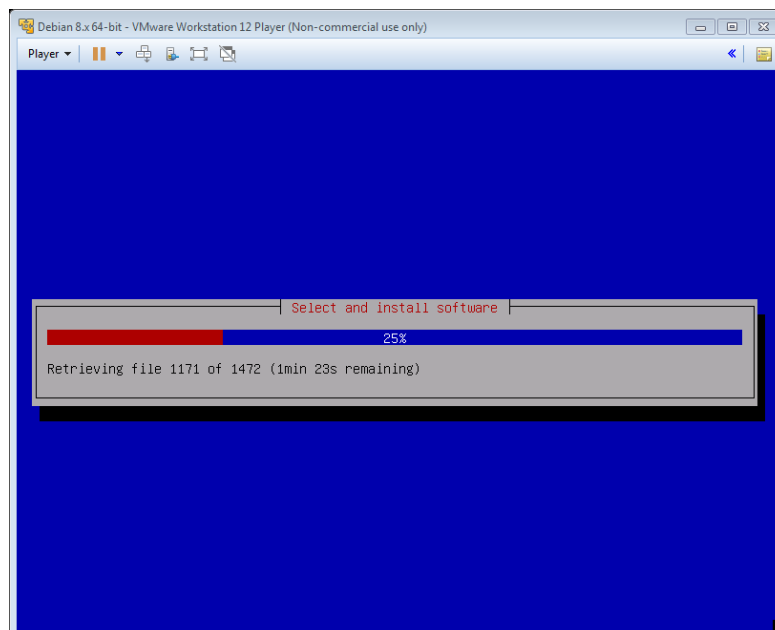
Kuva 10. Terminalista tehty commit

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.



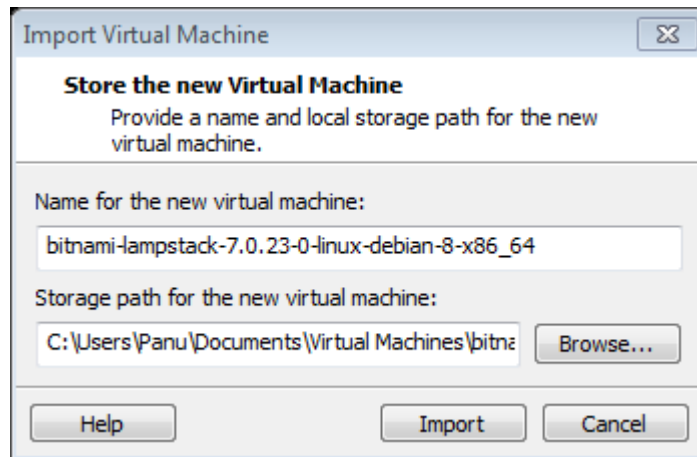
Kuva 11. 64-bittisen virtualisoinnin enableointi isäntäkoneen BIOSista

LAMP ei kuitenkaan ottanut toimiakseen 64-bittiselläkään Mintillä, joten seuraavaksi kehitettiin 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 haeskeli ohjelmistokirjastoja verkon yli hyvän tovin. Samalla kun asennettiin uusi käyttöjärjestelmä, käyttökokemuksen parantamiseksi virtuaalikoneeseen lisättiin myös hieman muistia sekä kahden suoritinajan käyttömahdollisuus.

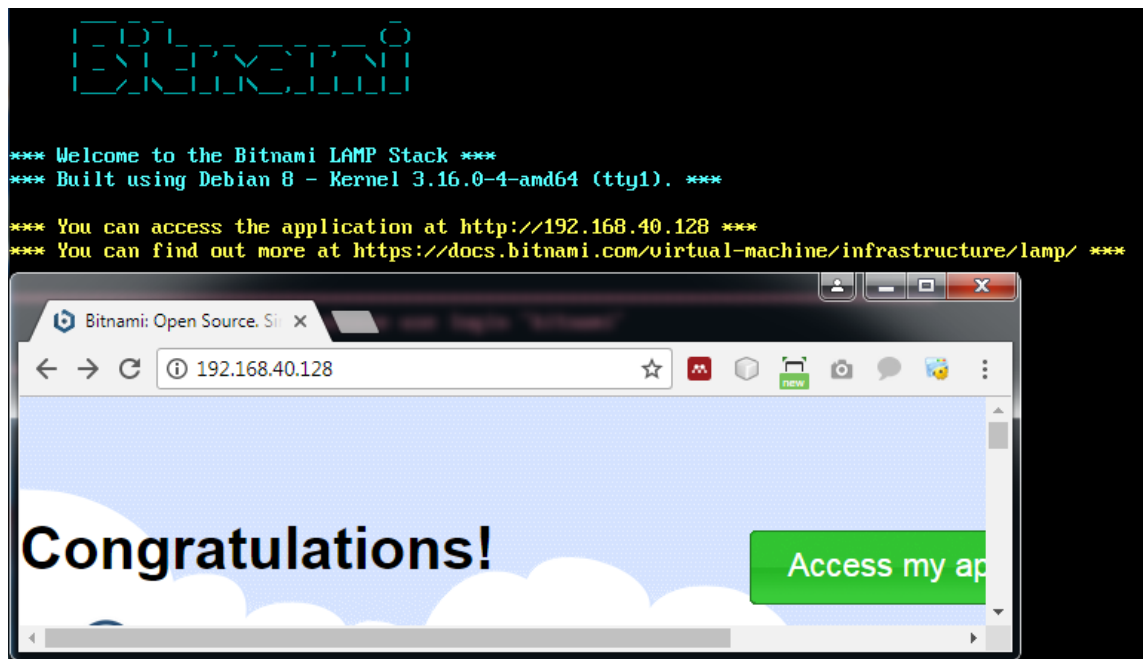


Kuva 12. Debian asentuu virtuaalikoneeseen

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.



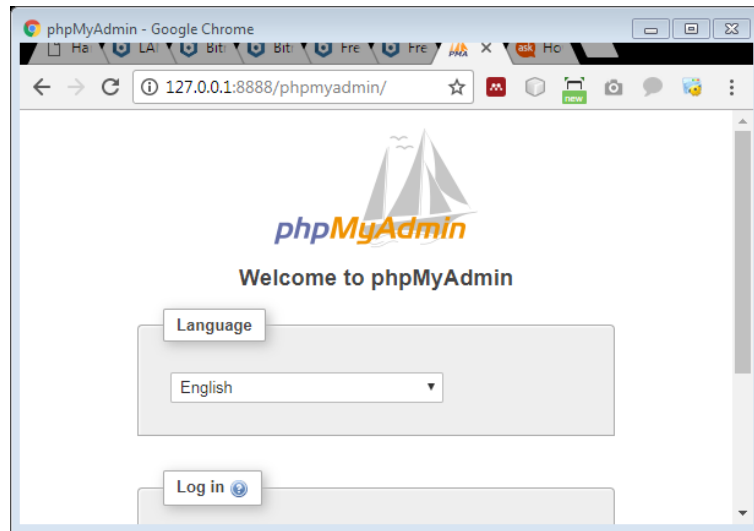
Oletusasetuksista poiketen virtuaalikoneen verkkokokoonpano säädettiin tilaan 'Host only', koska virtuaaliserverin toimintaa haluttiin tässä tapauksessa testata nimenomaan isäntäkoneella. Näiden säätöjen jälkeen serveri saatiinkin pystyyn ja yhteyskin näytti toimivan.



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

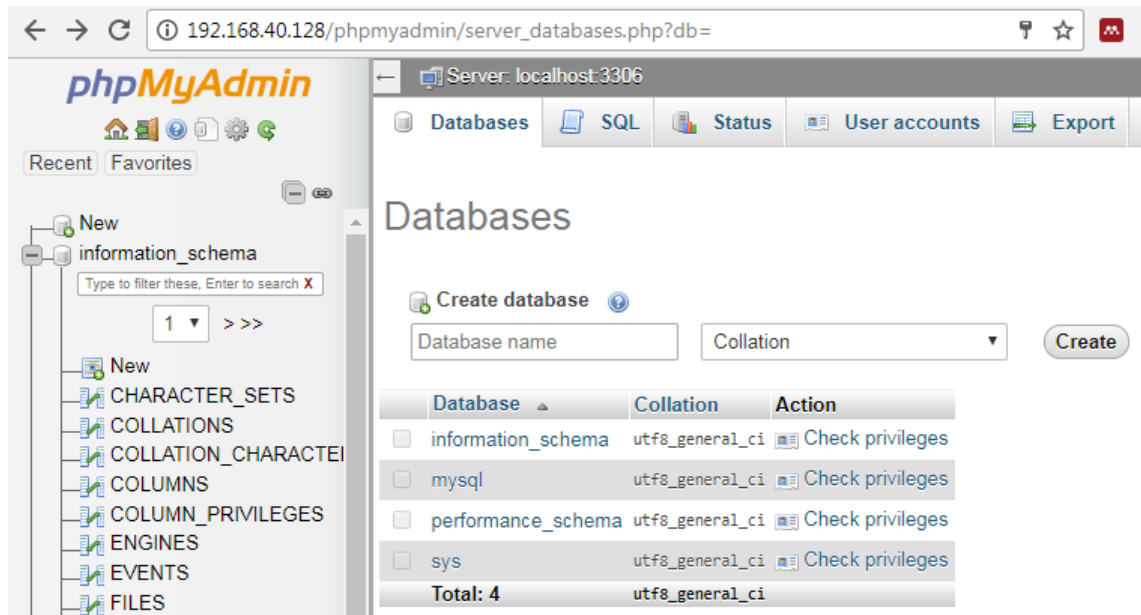
Seuraavaksi kokeiltiin phpMyAdminin toimintaa. Oletuksena phpmyadmin hyväksyy vain lokaalit yhteydenotot, joten ensin piti luoda Puttylla isäntäkoneesta virtuaaliserveriin

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.



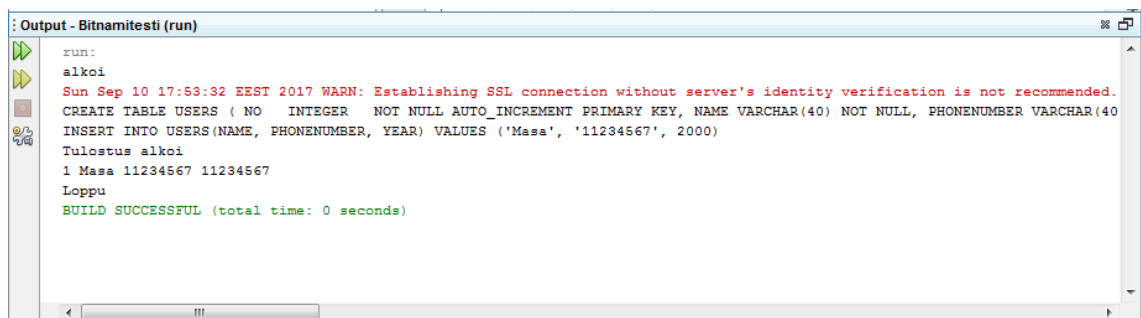
Kuva 14. phpmyadmin kirjautuminen SSH-tunneloinilla

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 päännraapimisen 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...

The screenshot shows the phpMyAdmin interface for a MySQL server at localhost:3306, database 'javatesti', and table 'users'. The top navigation bar includes 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and 'More'. A green status bar indicates 'Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)'. Below this, the SQL query 'SELECT * FROM `users`' is displayed. A toolbar offers options like 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. Further down, there are controls for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box. A '+ Options' link is visible. The table structure is shown with columns: NO, NAME, PHONENUMBER, and YEAR. A single row is displayed with values: 1, Masa, 11234567, and 2000. Action buttons for 'Edit', 'Copy', and 'Delete' are provided for each row.

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

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

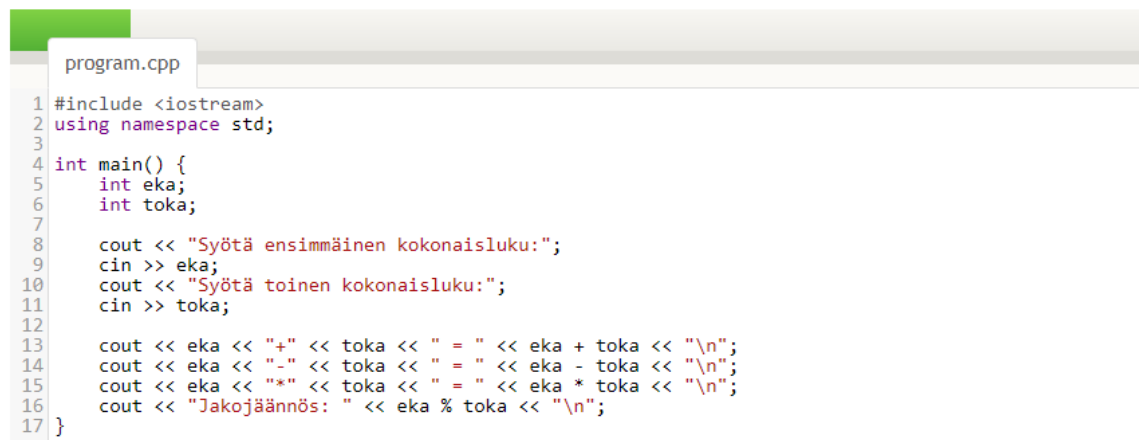
	NO	NAME	PHONENUMBER	YEAR
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	Masa	11234567	2000

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.



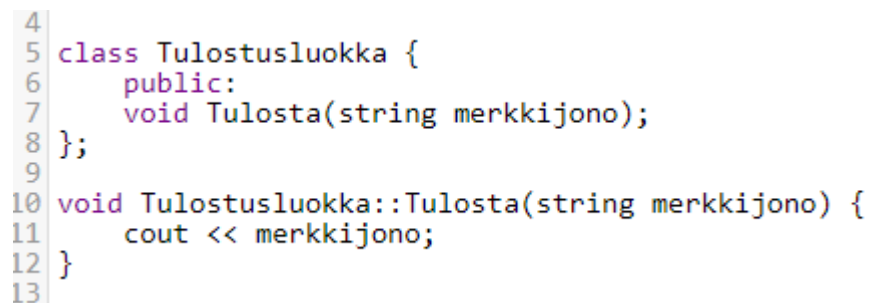
```

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 attribuu-teista ja metodeista vain abstraktit esittelyt, ja metodien toteutukset annetaan muualla koodissa syntaksilla `LuokanNimi::MetodinNimi(parametrit){}`. Ja mikäli toteutus on an-nettu, 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 tarjoihtuun 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

Staattinen 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...
30: private:
31:     Koira* koirat_[50];
32:     int koirien_lkm_;
33: };
```

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



Kuva 24. Yhteenveto tehdyistä harjoituksista, C++ oliokurssi

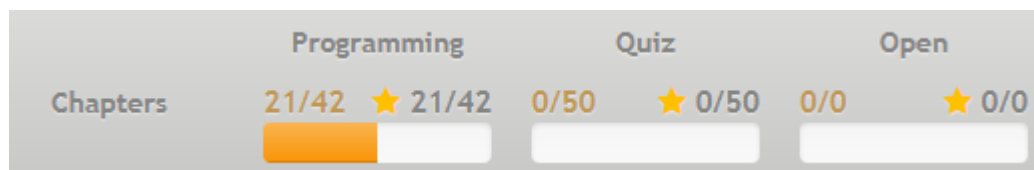
Jatkettiin seuraavaksi Python-harjoitusten merkeissä. Olennaisimmat havainnot:

- 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 tyyppitystä
- 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.



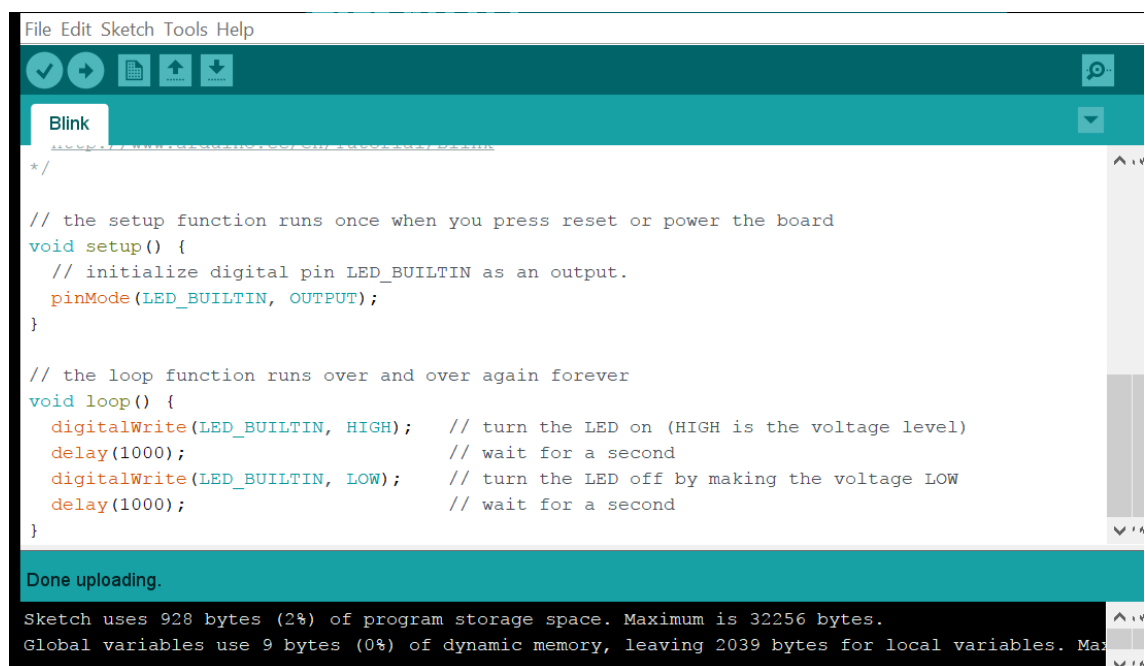
Kuva 25. Yhteenveto tehdyistä Python-harjoituksista viopessa

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.



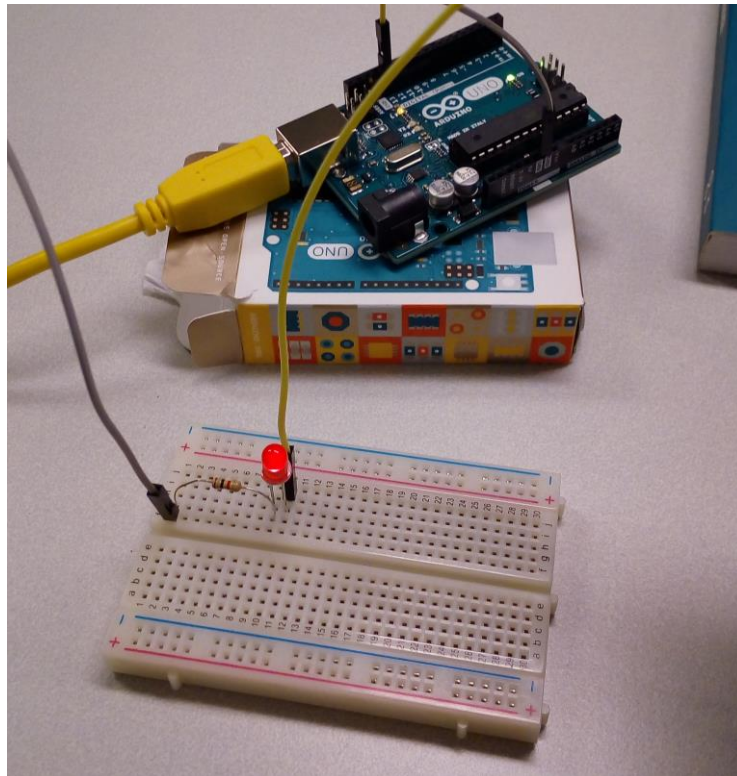
```
File Edit Sketch Tools Help
Blink
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// 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
```

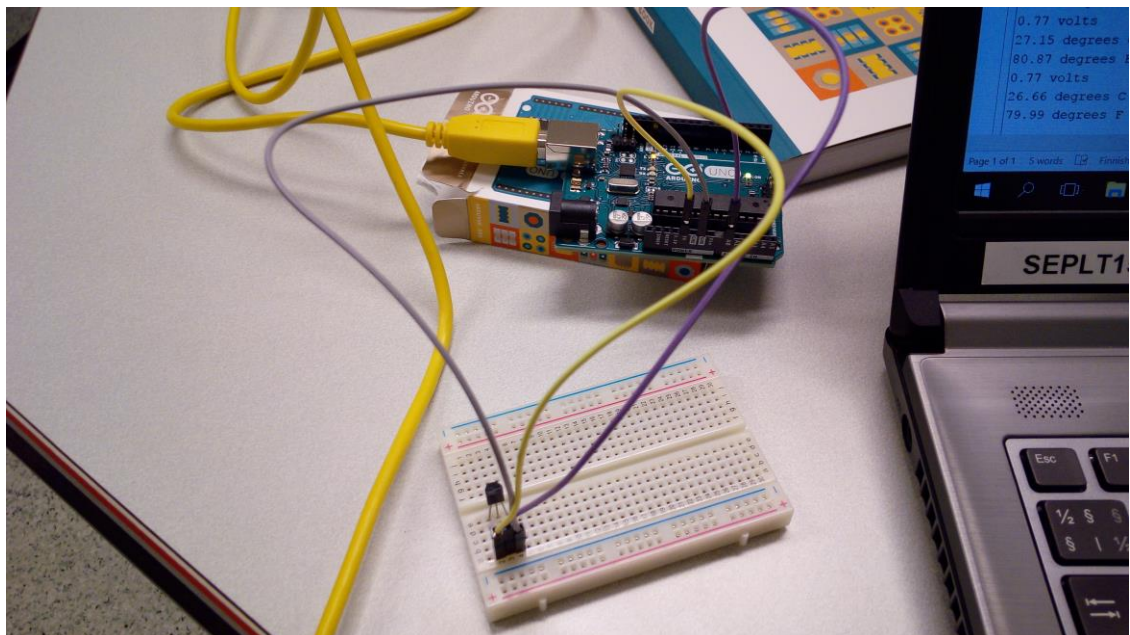
Kuva 26. Harjoituksen 1. koodi Arduino IDE:ssä

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



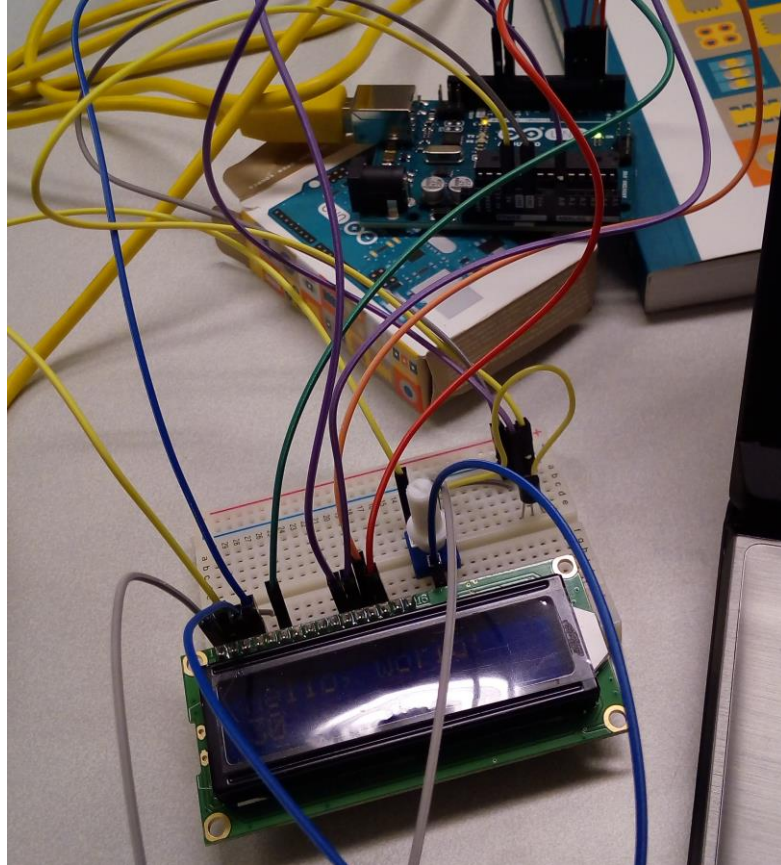
Kuva 27. LED syttyy ja sammuu Arduinon ohjaamana 1000 ms välein, vaikka sitä onkin kuvassa haastavaa demonstroida

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



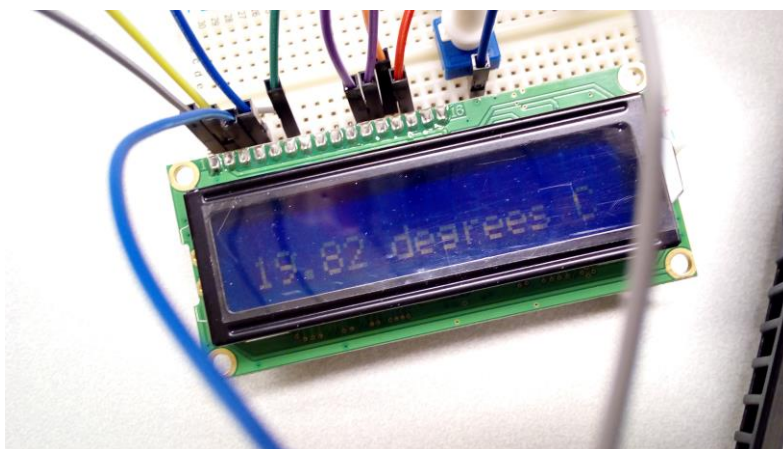
Kuva 28. Lämpötilan mittausta Arduinolla

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.



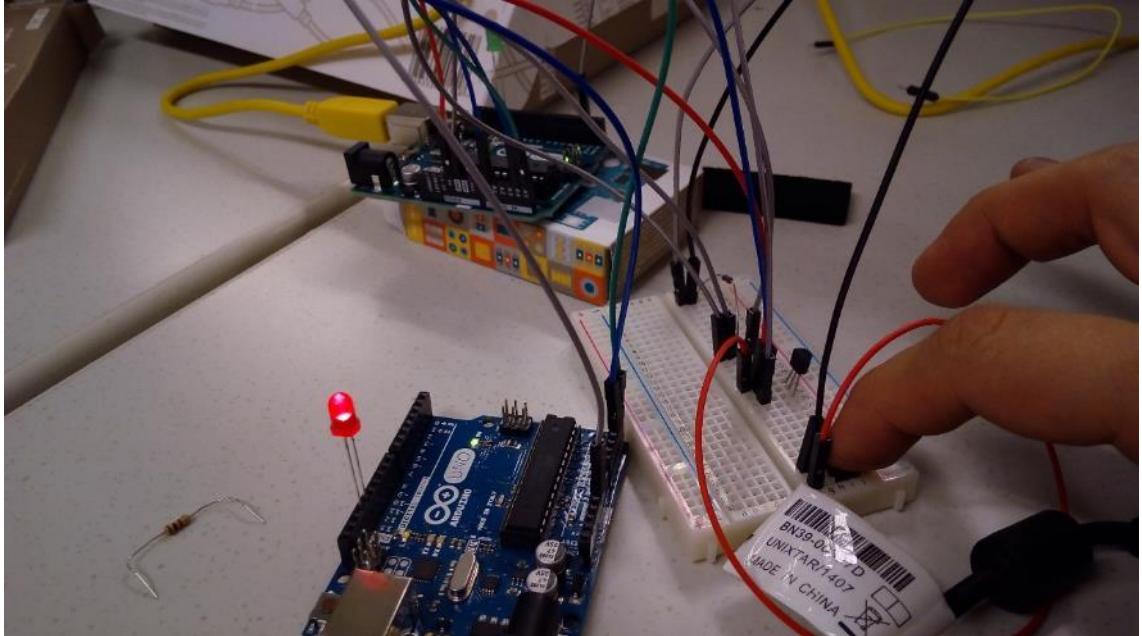
Kuva 29. LCD-näyttö kytkettynä Arduinoon.

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



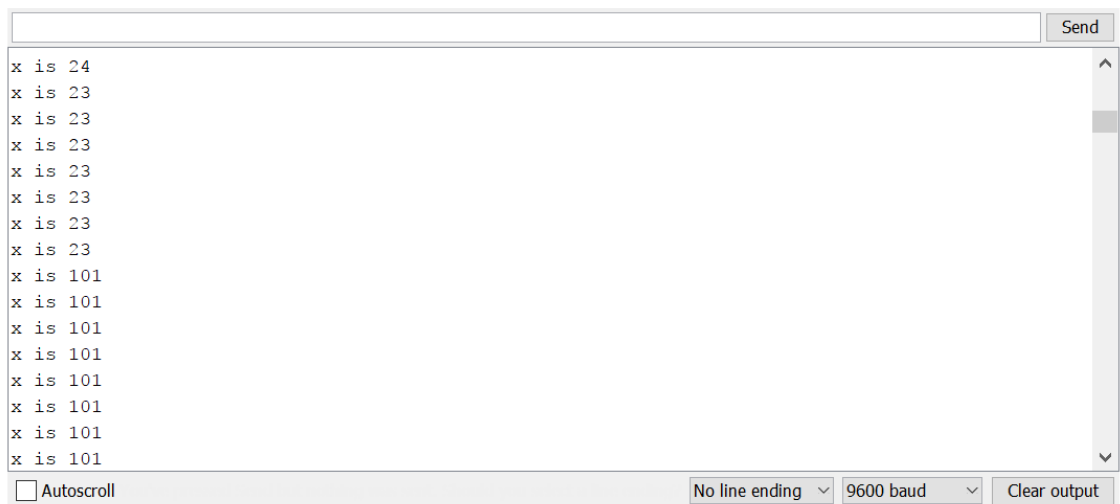
Kuva 30. Lämpötila-anturin lukema LCD-näytöllä.

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.



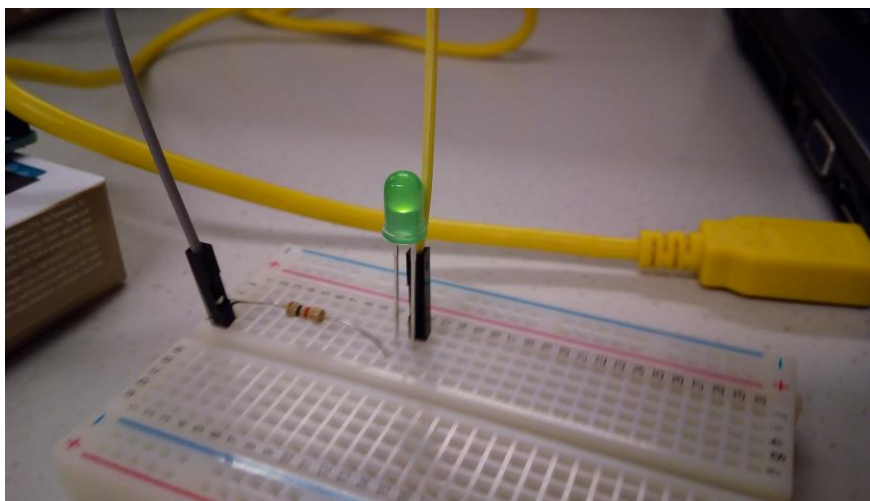
Kuva 31. Master-Slave kokoonpanon testausta

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

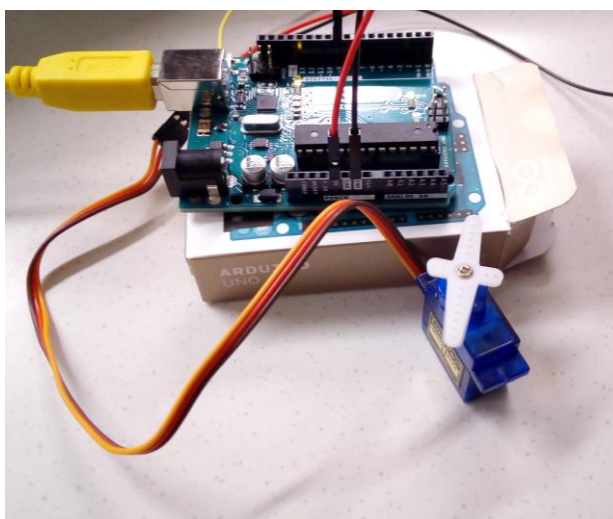
Viimeisenä muttei vähäisimpänä, tehtiin ohjelmointiharjoitus jossa Arduino IDE:ssä luotiin luokkatiedostot Morse.h sekä Morse.cpp. KytKentä oli tässä sama kuin harjoituksessa 1 jossa testattiin ledinvilkutusta. Morse-luokka tarjoaa metodit dash (pitkä ledinväläytys) ja dot (lyhyt ledinväläytys). Näitä 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, vaikkei se kuvassa valitettavasti näy.

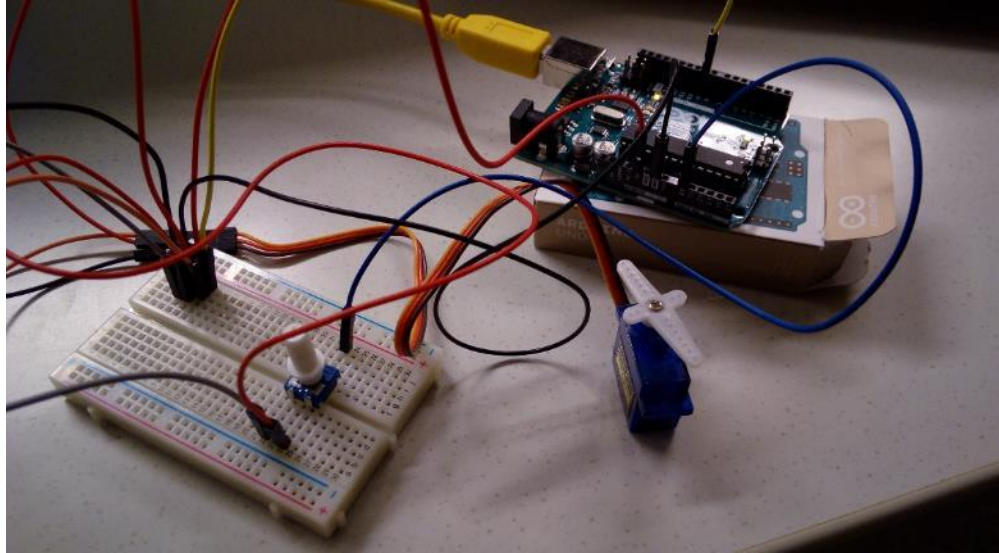
28.9.2017

Jatkettin vielä työskentelyä muutaman Arduino-lisäharjoituksen parissa. Ensimmäinen näistä oli servomoottorin (Tower Pro SG90) ohjaaminen. Ensin tehtiin minimikytKentä, jossa esiohjelmoitu logiikka liikkutti 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.



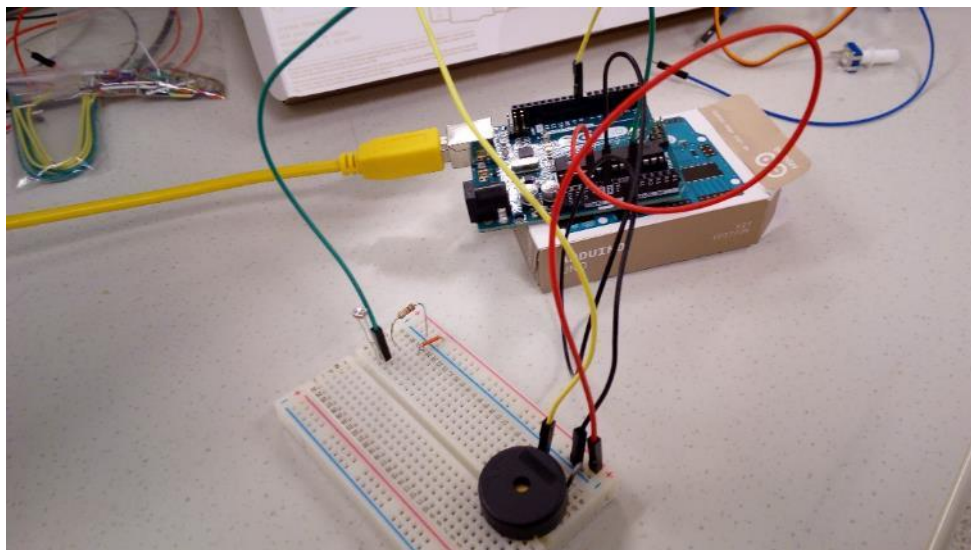
Kuva 34

Toisena vaiheena oli kytkennän jatkojalostaminen siten, että servomoottorin 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.

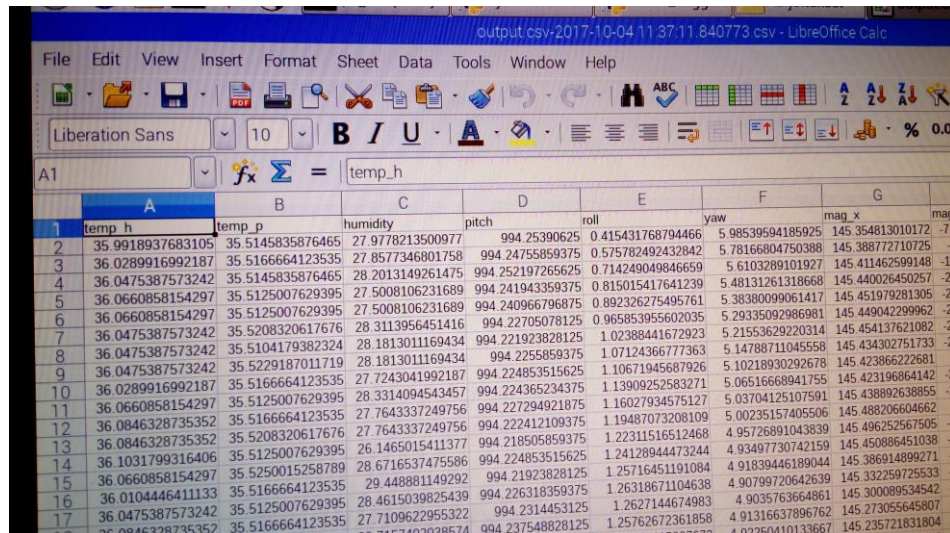
5. RASPBERRY PI

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 Mikrobitin 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 tyyppillisesti jokin linux-distributio, joten kurssilla aiemmin opitut taidot tulevat tarpeeseen. Esim. Raspberryn tapauksessa käytössä on laitteelle räätälöity Raspbian, joka sisältää perustoiminnallisuuksien lisäksi hyvinkin mielenkiintoisia lisäherkkuja, kuten normaalisti maksullisen symbolisen Mathematica-ohjelmiston esi-asennettuna.

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	H
	temp_h	temp_p	humidity	pitch	roll	yaw	mag_x	mag_y
1	35.9918937683105	35.5145835876465	27.9778213500977	994.25390625	0.415431768794466	5.98539594185925	145.354813010172	-7.1
2	36.0289916992187	35.5166664123535	27.8577346801758	994.24755859375	0.575782492432842	5.78166804750388	145.388772710725	-1
3	36.0475387573242	35.5145835876465	28.2013149261475	994.252197265625	0.714249049646659	5.6103289101927	145.411462599148	-19
4	36.0660858154297	35.5125007629395	27.5008106231689	994.241943359375	0.815015417641239	5.48131261318668	145.440026450257	-20
5	36.0660858154297	35.5125007629395	27.5008106231689	994.240966796875	0.892326275495761	5.38380099061417	145.451979281305	-21
6	36.0660858154297	35.5208320617676	28.3113956451416	994.22705078125	0.965853955602035	5.29335092986981	145.449042299962	-21
7	36.0475387573242	35.5104179382324	28.1813011169434	994.221923828125	1.02388441672923	5.21553629220314	145.454137621082	-21
8	36.0475387573242	35.5229187011719	28.1813011169434	994.2255859375	1.07124366777363	5.14788711045558	145.434302751733	-21
9	36.0289916992187	35.5166664123535	27.72430411992187	994.224853515625	1.10671945687926	5.10218930292678	145.423196864142	-21
10	36.0289916992187	35.5166664123535	27.72430411992187	994.224365234375	1.13909252583271	5.06516668941755	145.438992638855	-21
11	36.0660858154297	35.5125007629395	28.3314094543457	994.227294921875	1.16027934575127	5.03704125107591	145.423196864142	-21
12	36.0846328735352	35.5166664123535	27.7643337249756	994.222412109375	1.19487073208109	5.00235157405506	145.488206046662	-21
13	36.0846328735352	35.5208320617676	27.7643337249756	994.218505859375	1.22311516512468	4.95726891043839	145.496252567505	-21
14	36.1031799316406	35.5125007629395	26.1465015411377	994.224853515625	1.24128944473244	4.93497730742159	145.450886451038	-21
15	36.0660858154297	35.5250015258789	28.6716537475586	994.21923828125	1.25716451191084	4.91839446189044	145.386914899271	-21
16	36.0104446411133	35.5166664123535	29.448881149292	994.226318359375	1.26318671104638	4.90799720642639	145.332259725533	-21
17	36.0475387573242	35.5125007629395	28.4615039825439	994.226318359375	1.2627144674983	4.9035763664861	145.300089534542	-21
18	36.0475387573242	35.5166664123535	27.7109622955322	994.2314453125	1.25762672361858	4.91316637896762	145.273055645807	-21
19	36.0846328735352	35.5166664123535	28.7157402038574	994.237548828125	1.28004455007673	4.92250410133667	145.235721831804	-21

Kuva 37

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



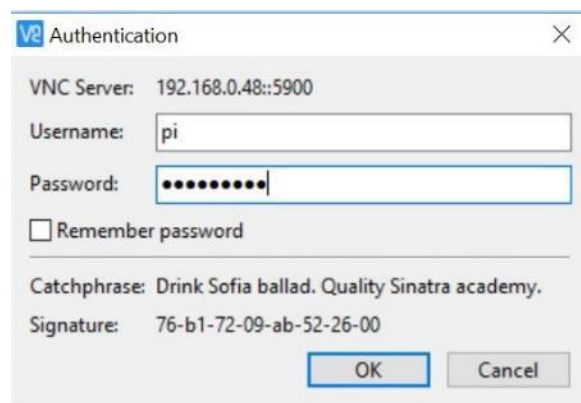
Kuva 38

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.

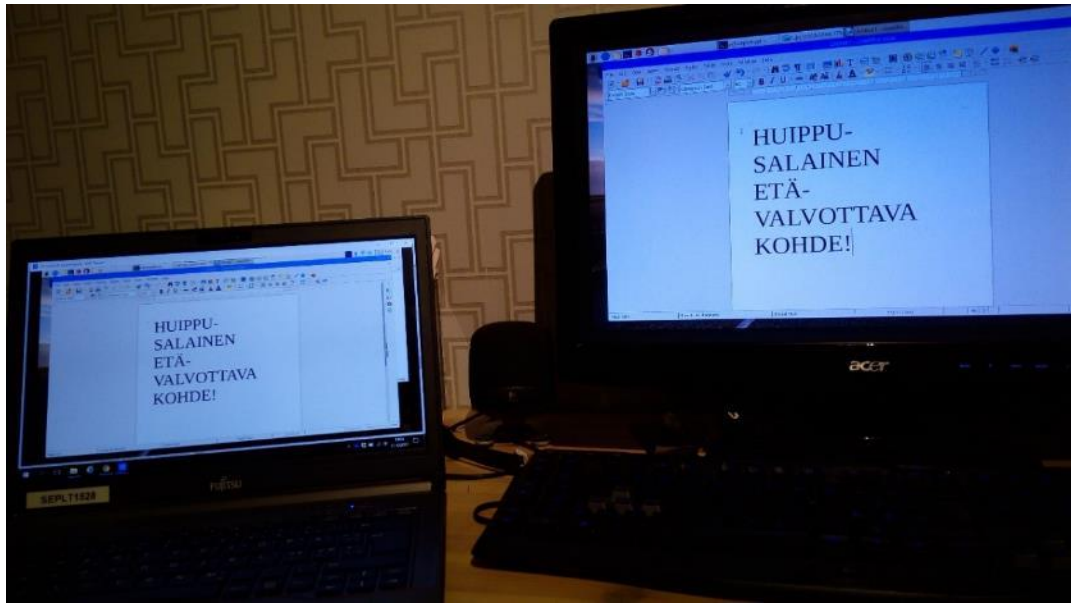


Kuva 39. Firstpic.jpg - ensimmäinen kuva raspin kameralla

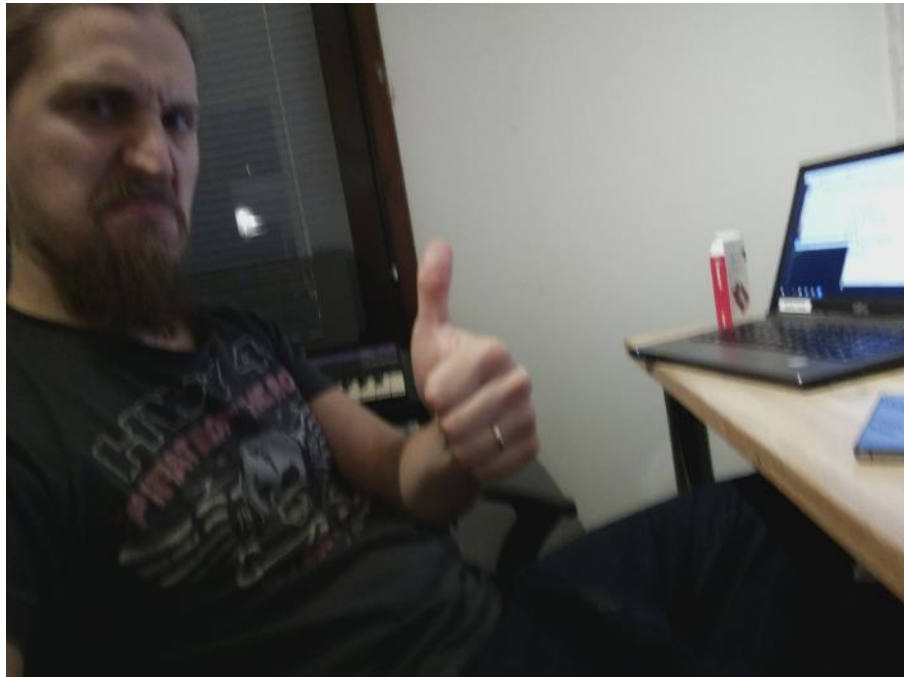
Etätyöpöydän virittely oli seuraava tehtävä. Raspberryä voidaan käyttää etäyhteydellä virtuaalityöpöydän kautta omalta läppäriltä, jolloin itse Raspberry voi fyysisesti sijaita missä hyvänsä, eikä tarvitse välttämättä edes näyttöä tai muita oheislaitteita kunhan verkko-yhteys on saatavilla. Tätä varten asennettiin Raspberryyhin 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 Raspberryyhin oli helppoa. IP-osoite saadaan tarvittaessa Raspin konsolista sanomalla "ifconfig", tai kotiverkossa myös tutkimalla reitittimen ARP-taulua

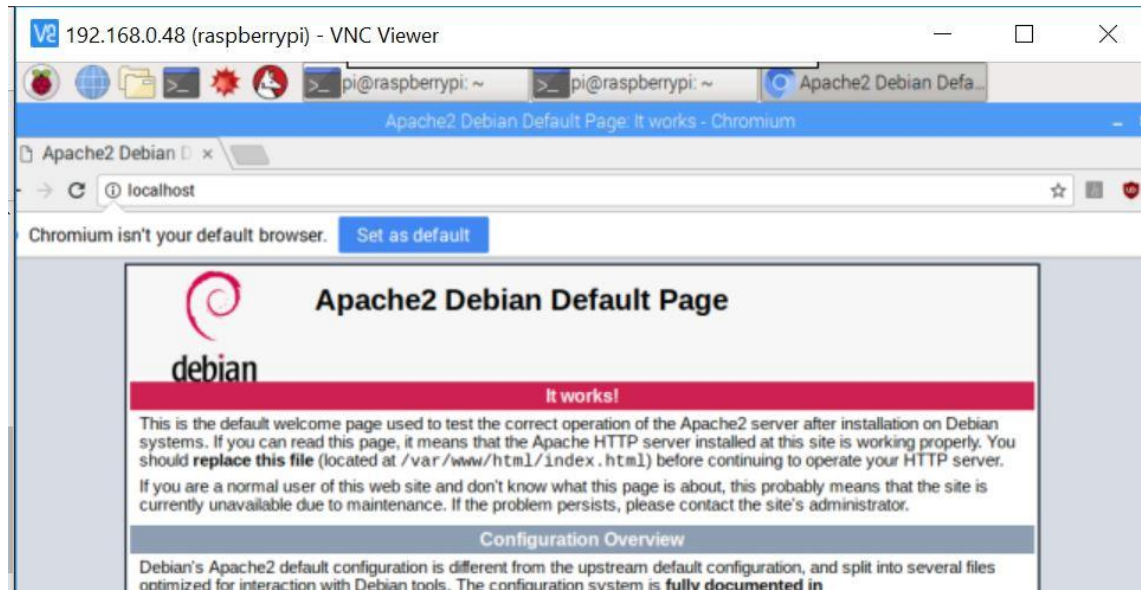


Kuva 41. Raspberryn työpöytään otettu yhteys läppärltä WLAN-verkon yli.



Kuva 42. Etätyöpöydän kautta napattu kuva

Vuorossa oli seuraavaksi harjoitus, jossa yhdisteltiin http-palvelimen pystyttämistä, bash-skriptiasta, 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

Name	Last modified	Size	Description
 Parent Directory		-	
 2017-10-11_1943.jpg	2017-10-11 19:43	5.1M	
 2017-10-11_1954.jpg	2017-10-11 19:54	5.2M	
 2017-10-11_1955.jpg	2017-10-11 19:55	5.3M	
 2017-10-11_1956.jpg	2017-10-11 19:56	5.2M	
 2017-10-11_1957.jpg	2017-10-11 19:57	5.2M	
 2017-10-11_1958.jpg	2017-10-11 19:58	5.2M	
 moi.jpg	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*

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

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/phpmyadmin/?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
- [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
- [14] <https://www.raspberrypi.org/documentation/remote-access/vnc/>
- [15] <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>