Liikkeiden tunnistus ohjelma - LIIKKUVAXI

Turun AMK:n terveysteknologian laboratorion tilaama ohjelma.

Halutaan tutkia ihmisen liikkumisaktivisuutta eri tilanteissa.

Tämä toteutetaan käyttämällä RuuviTag®-sensoreita, jotka lähettävät Bluetooth-teknologialla dataa Raspberry PI Model 3 –serverille (testilaboratorion ympäristössä IP: 10.100.0.138), jossa laitevalmistajan RuuviCollector-ohjelma vastaanottaa sensorien signaaleja ja tallentaa tietoja InfluxDB-tietokantaan.

Lisäksi serverille on asennettu Grafana-ohjelma, jolla voidaan graafisesti esittää InfluxDB:hen tallennettuja tuloksia.

Asennusohje löytyy linkistä:

<https://blog.ruuvi.com/rpi-gateway-6e4a5b676510>

Projektissa kehitettiin React-ohjelma, jolle annettin nimeksi LIIKKUVAXI.

Ohjelma lukee dataa InfluxDB-kannasta, analysoi sitä ja esittää web-sovelluksessa.

Ohjelma LIIKKUVAXI on toteutettu client server -arkkitehtuurilla.

Ohjelmaruudulla on tavallaan kaksi osiota:

Ensimmäinen osio kertoo ohjelman tuoreita käynnistyshetken mittauksia.

Toisella osiolla voidaan tarkastella tietyn ajanjakson tilannetta menneisyydessä syöttämällä ruudun alaosan kenttiin haluttu ajanjakso päivämäärän ja kellonajan tarkkuudella.

Asennus ohje:

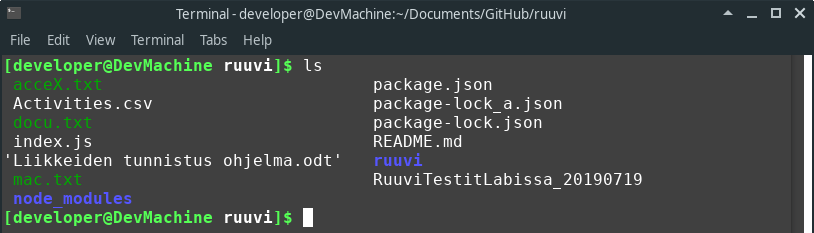
Ohjelman kaikki koodi ja dokumentaatio löytyy osoitteesta:

<https://github.com/NL-0/ruuvi>

$ git clone <https://github.com/NL-0/ruuvi.git>

$ cd ruuvi

$ ls

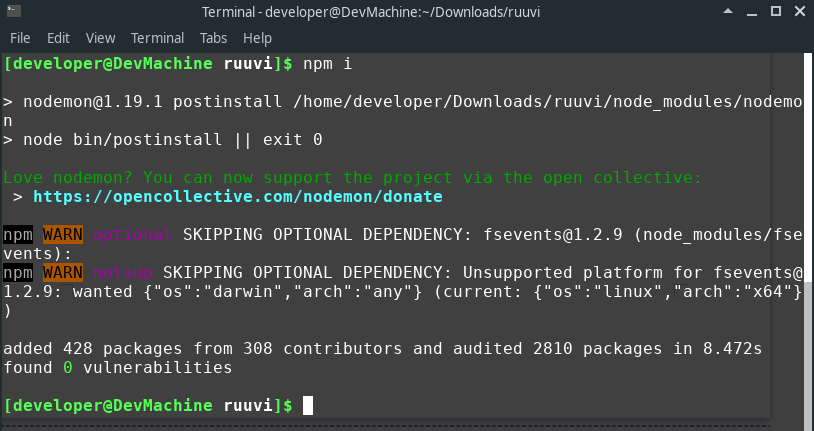


Ohjelma on jaettu 2 eri ohjelmaan joista molemmat pitää asentaa

$ npm i -komennolla

ensin ruuvi kansiossa

$ npm i



jonka jälkeen sama *ruuvi/*ruuvi kansiossa

$ npm i



Ensimmäisessä ruuvi kansiossa on nodejs server ja ruuvi/ruuvi on itse react app



Server sekä React App käynnistetään tästä ensimmäisestä kansiosta komennolla

$ npm run app

joka käynnistää molemmat ohjelmat.

**Ohjelman näyttö:**

Ohjelmaruudulla on tavallaan kaksi osiota:

Ensimmäinen osio kertoo ohjelman tuoreita käynnistyshetken mittauksia.

Toisella osiolla (alkaen kohdasta ’Liikkunut aikana’) voidaan tarkastella tietyn ajanjakson tilannetta menneisyydessä syöttämällä kenttiin haluttu ajanjakso päivämäärän ja kellonajan tarkkuudella.

**Ensimmäinen osio:**

Ohjelmalla näytetään kolmen eri RuuviTag-sensorin liikkeitä, kukin sensori omassa sarakkeessaan.

Esitysaika alkaa siitä hetkestä, kun ohjelma käynnistetään eli tiedot ovat ’live-tilanteesta’ 4 sekunnin viiveellä

**Rivien tietosisällöt:**

RuuviTag-sensorin mac-osoite

**Lämpötila:**

Celcius asteina

**Liike:**

x-akselilla havaittu liike

y-akselilla havaittu liike

z-akselilla havaittu liike

**Muutos %:**

kunkin akselin muutosprosentti edelliseen arvoon nähden

värilaatikko; punainen = ei havaittu liikettä, vihreä = liikettä havaittu

**Muutos total:**

havaittujen liikkeiden lukumäärä mittausaikana,

havaittujen muutosten prosentuaalinen ero,

värilaatikko; punainen = ei ole havaittu / vihreä = on havaittu, liikkeitä kulumassa olevan tunnin aikana (nollaus tasatunnein)

**Liikkeitä yhteensä:**

havaittujen liikkeiden lukumäärä kulumassa olevan tunnin aikana (nollaus tasatunnein)

**Signal:**

signaalin voimakkuus

**Toinen osio:**

Tässä osassa ohjelmaa voidaan tarkastella historiatapahtumia halutulta aikajaksolta.

**’Liikkunut aikana’** –kohtaan syötetään tarkasteluun haluttavan ajanjakson alku- ja loppuajankohta ’Aika1’- ja ’Aika2’ –kenttiin.

Ajat syötettävä muodossa **vvvv-kk-ppThh:mm:ss**, jossa

vvvv = vuosiluku

-=vakio

kk=kuukausi

-=vakio

pp=päivä

T=vakio

hh=tunnit

:=vakio

mm=minuutit

:=vakio

ss=sekunnit

Painamalla ’NÄYTÄ’ –painiketta saadaan esille sensorikohtaiset liikegraafit ylempään kaavioon ja kaikkien sensorien keskiarvograafi alempaan kaavioon.



ARDUINO TIME -painikkeen toiminta kerrotaan tämän dokumentin sivulla 27.

CSV DATA -painikkeen toiminta puolestaan sivulla 33.

**TESTAUKSIA:**

**Mittausdata 11.07.2019 11:52**

**n. 1 min, tasaiset liikkeet, kolme anturia samassa taskussa**

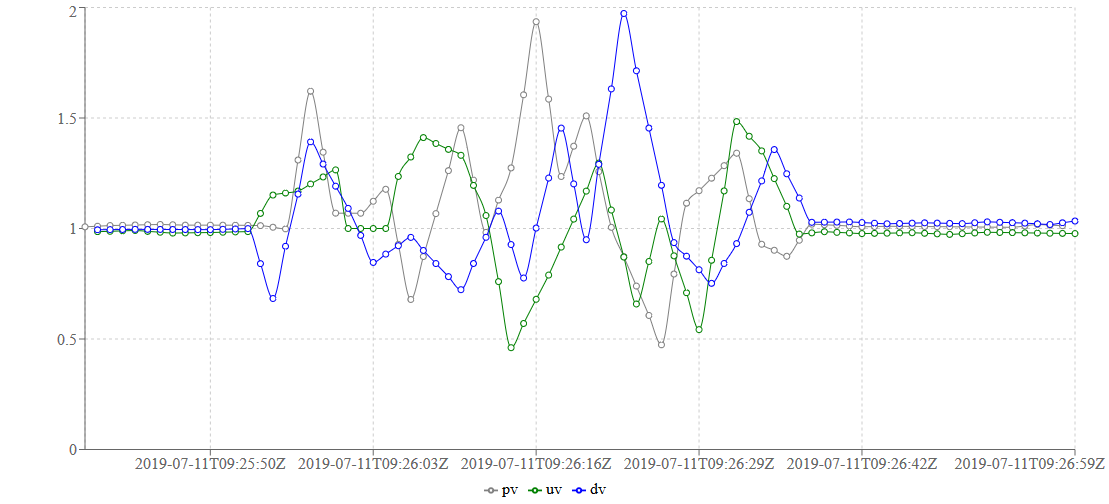
11.07.2019 12:25:40 - 12:27:00, liikkeet normaalinopeudella, kolme anturia "samassa taskussa"

- kaksi kierrosta

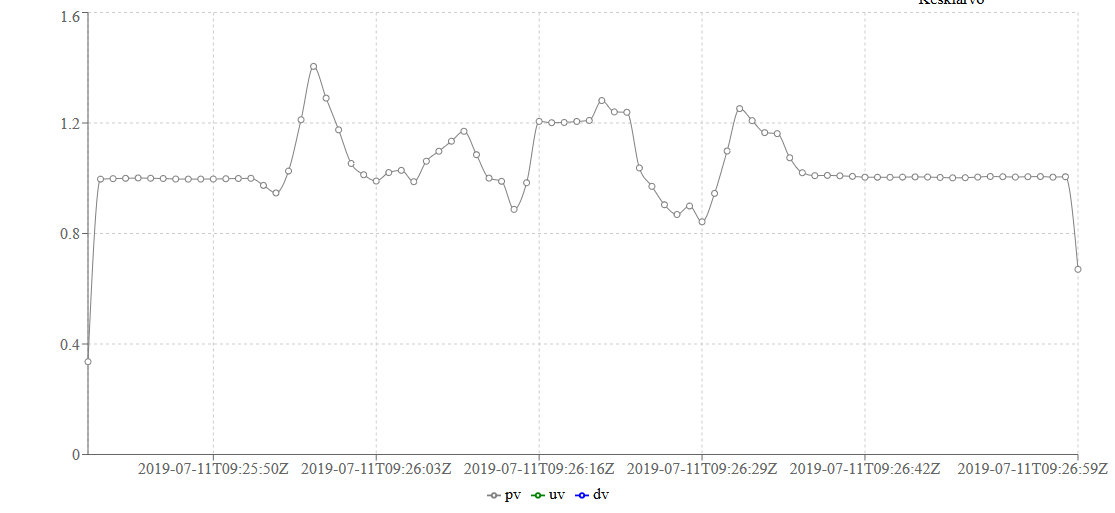
**HUOM ! Kaavioissa kellonaika on eri kuin otsikoissa mainituissa testiajoissa, koska Raspberryn kello oli UTC-ajassa eli 3 tuntia ’meidän’ kesäaikaamme (EEST) jäljessä.**

**HUOM 2 ! RuuviTag:ien firmware-versio alkuperäinen.**

Kolmen eri anturin yksittäiset liikehavainnot



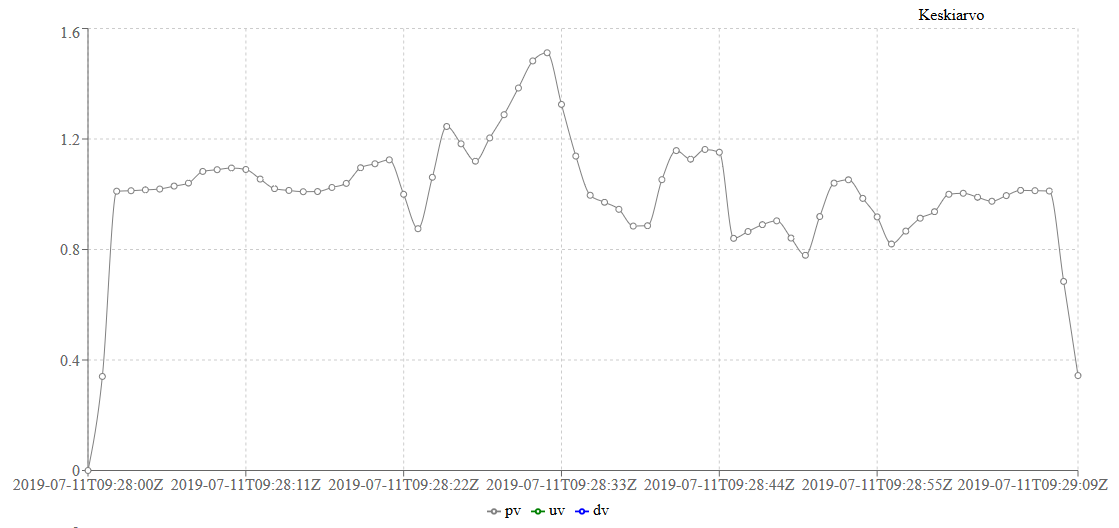
Kolmen anturin arvoista laskettu keskiarvo



11.07.2019 12:28 - 12:29, liikkeet normaalinopeudella, 8 mallinen, kolme anturia "samassa taskussa"

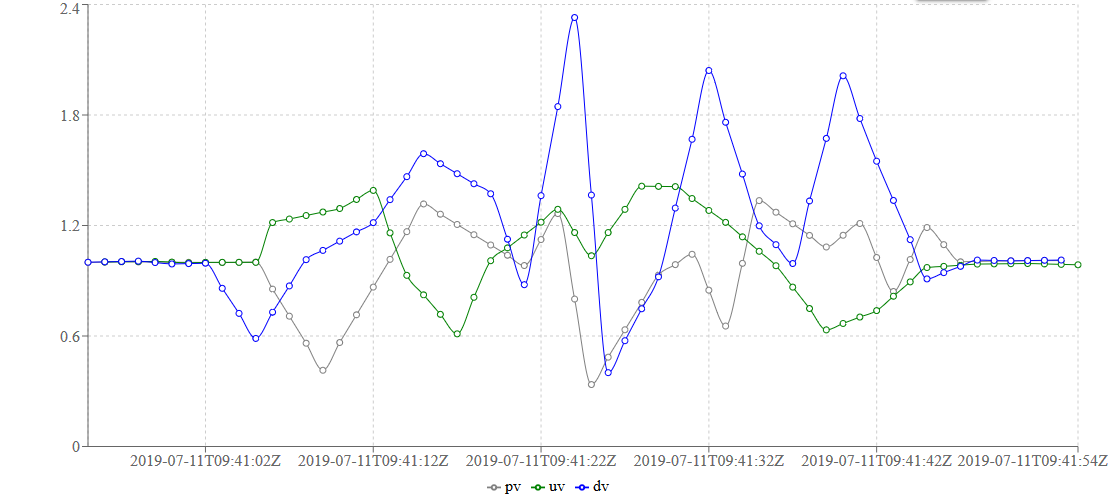
- kaksi kierrosta

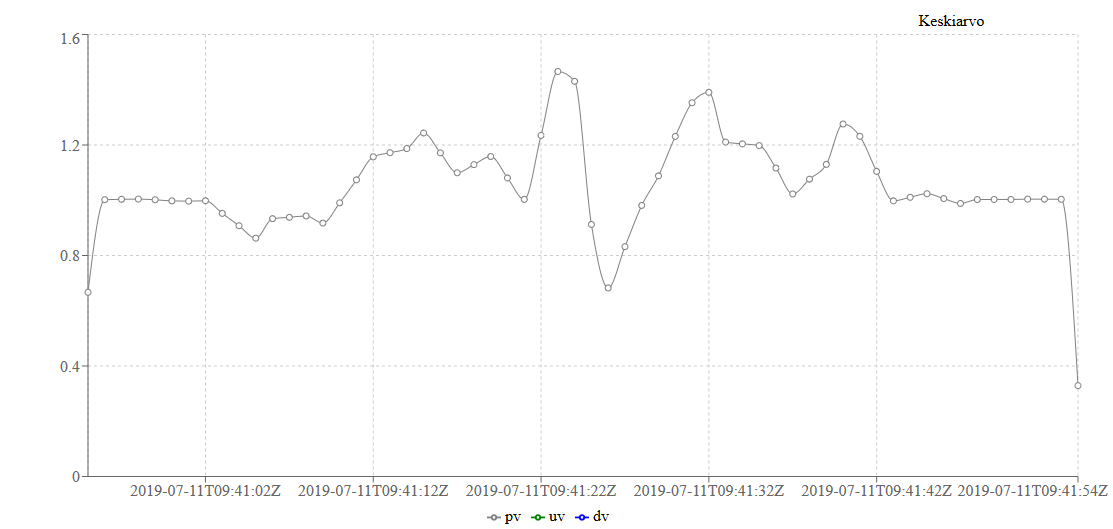




11.07.2019 12:41:02 - 12:41:48, liikkeet normaalinopeudella, 8 mallinen, kolme anturia "samassa taskussa"

- kaksi kierrosta

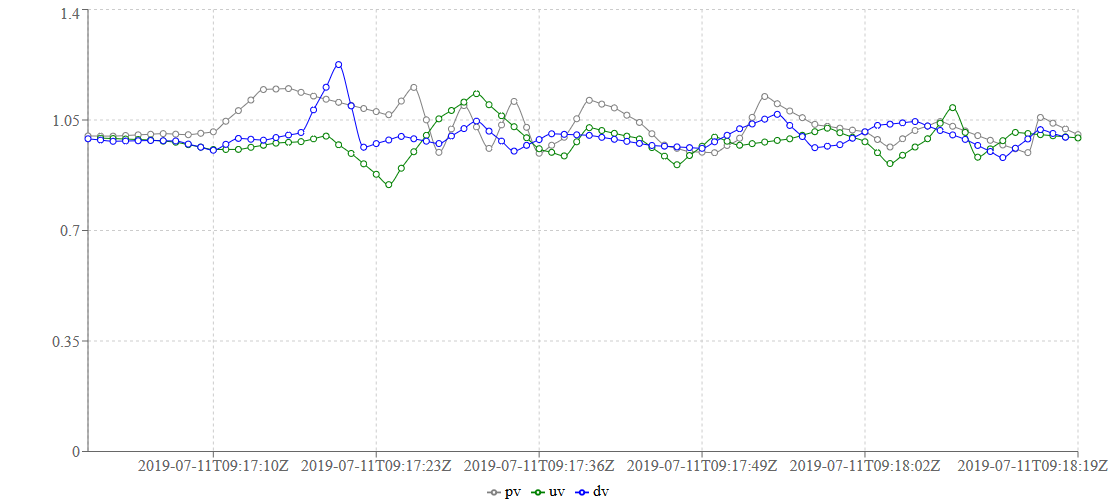


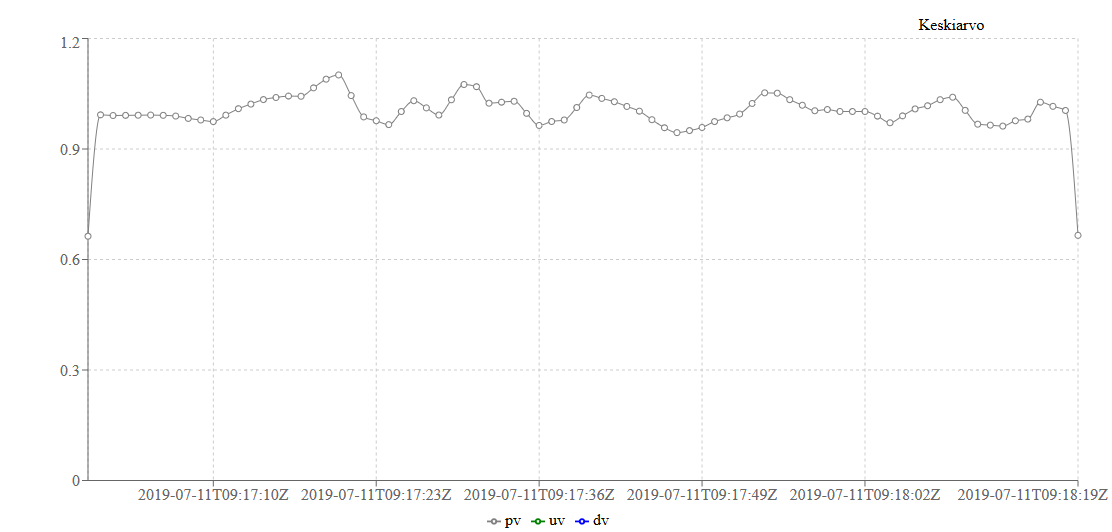


|  |
| --- |
| 11.07.2019 11:56 - 11:57, tasaiset hitaat liikkeet, kolme anturia "samassa taskussa" |
|  |
|  |

|  |
| --- |
| 11.07.2019 12:14 - 12:16, tasaiset hitaat liikkeet, kolme anturia "samassa taskussa" |
|  |
|  |

11.07.2019 12:17 - 12:18, tasaiset hitaat liikkeet, kolme anturia "samassa taskussa"





**15.7.2019**

**Testauksessa todettua:**

Clientin ja serverin välinen HTTP-yhteyskäytäntö kätevä testausvaiheessa, mutta tietokannan kasvaessa se hidastuu käyttökelvottomaksi. Suositellaan käytettäväksi InfluxDB Client for Node.js

<https://github.com/node-influx/node-influx>

**17.7.2019**

**Otettiin testattavaksi uudempi ja tehokkaampi Raspberry Pi 3 Model B+ -serveri.**

Aluksi serveri (testilaboratorion ympäristössä IP:10.100.0.111) päivitettiin Linux-versioon Raspbian Buster.

Sen jälkeen asennettiin RuuviCollector ja InfluxDB.

Tämän jälkeen todettiin, että sensoridata alkoi päivittyä InfluxDB-tietokantaan.

Samassa yhteydessä päivitettiin myös Ruuvi-sensorien firmware uusimpaan netistä löytyvään beta-versioon 2.4.2. Tähän löytyy ohje nettiosoitteesta:

<https://lab.ruuvi.com/dfu/>

**18.7.2019**

Määritettin sensoreista yksi lähettämään dataa ’vanhalle’ Raspberry Pi 3 –serverille ja kaksi uudelle

Raspberry Pi 3+ -serverille.

Tämä toteutettiin kopioimalla ensin tiedosto ruuvi-collector.properties.example hakemistoon, jossa ruuvi-collector-\*.jar on, ja muuttamala nimeksi ruuvi-collector.properties sekä muuttamalla sisällöstä seuraavat kentät:

filter.mode=whitelist

filter.macs=kirjoita mac-osoitteet tähän (esim. EF9F74296486)

measurementUpdateLimit=2000

Ohje/esimerkki löytyy tästä github-osoitteesta:

<https://github.com/Scrin/RuuviCollector/blob/master/ruuvi-collector.properties.example>

**19.7.2019**

Suoritettiin uudet liikkumistestit (hidas kävely, normaali kävely, ’rollaattori’) laboratoriossa.

Sensorit (3 kpl) asetettiin rinnakkain kovakantisen kirjan päälle.

Kävelytesteissä testikävelijä piti kirjaa käsin rinnan korkeudella edessään.

’Rollaattoritestissä’ kirja asetettiin pyörillä varustetun toimistotuolin istuimelle ja testaaja työnsi tuolia hitaahkolla nopeudella.

Seuraavissa kaaviosarjoissa esitetään ensin viivagraafina kolmen eri Ruuvi-sensorin liikkeet.

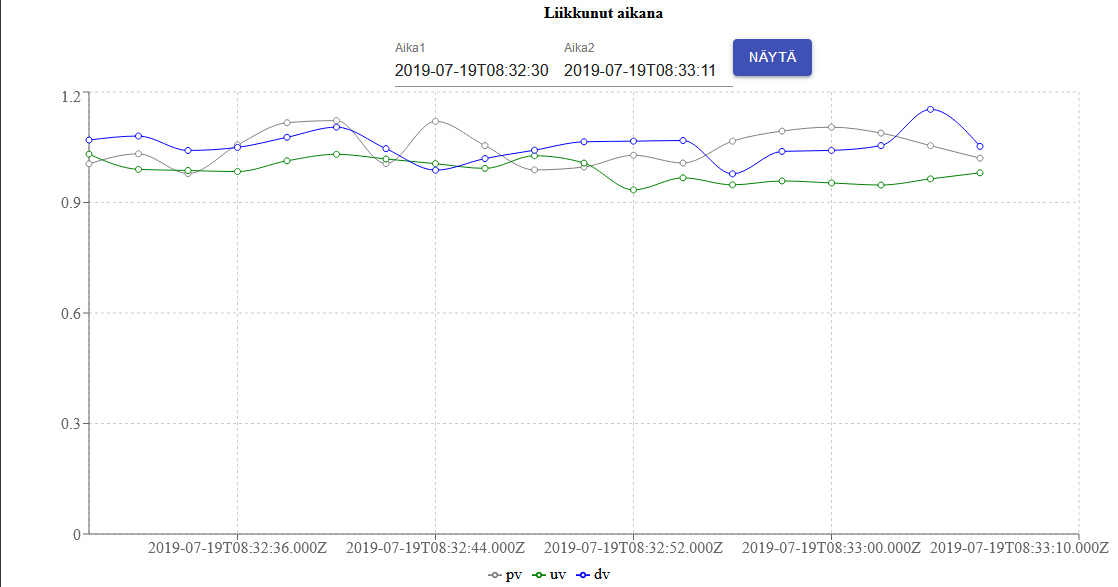
Kunkin sensorin liikeradassa on laskettu yhteen x-, y- ja z-akselien (vaaka-, pysty- ja syvyyssuuntaiset) liikkeet.

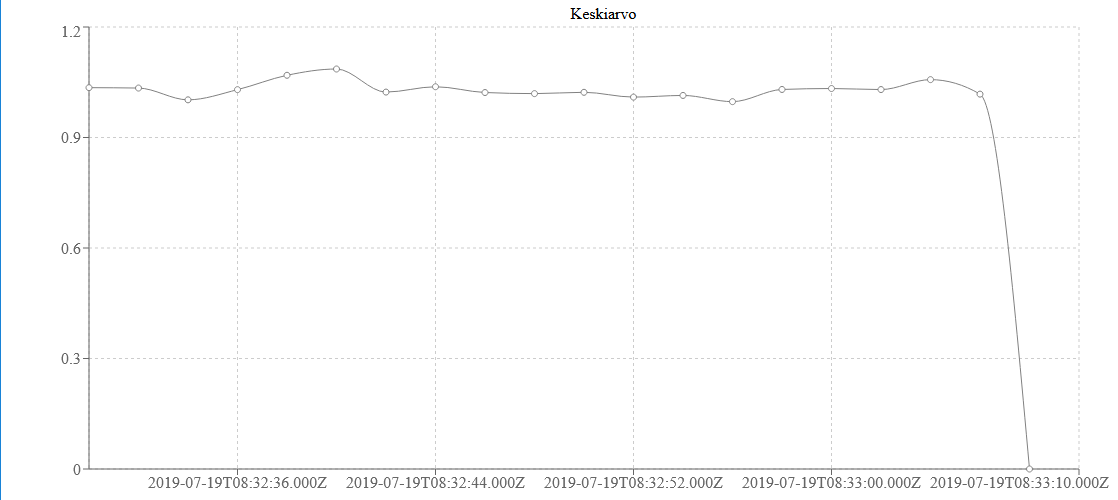
Toisena graafina näytetään edellä mainittujen sensorien arvoista lasketut keskiarvot.

**Hidas kävely:**

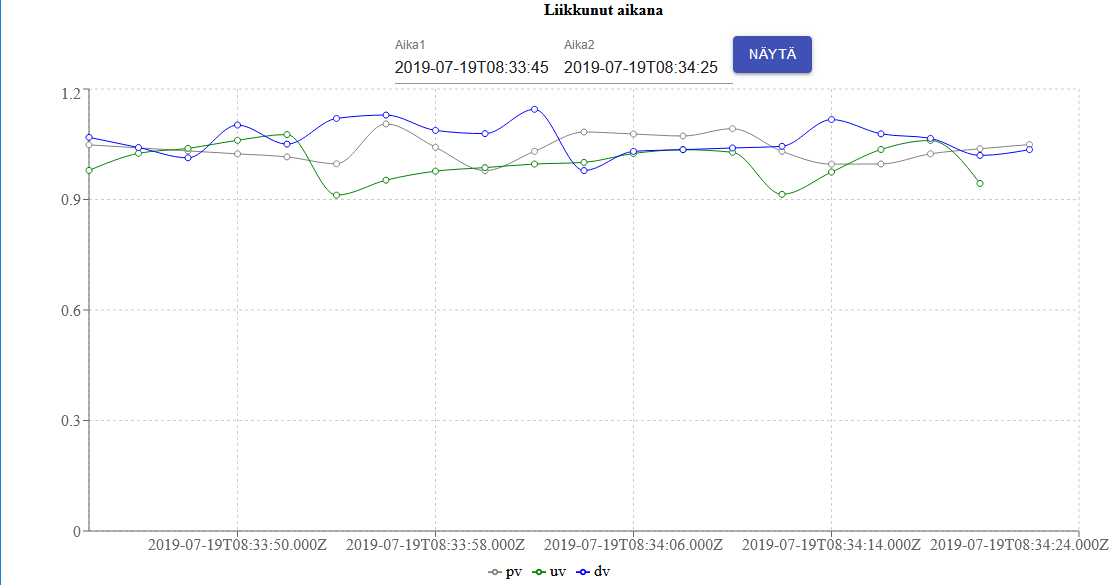
Yksi ympyrän muotoinen kierros laboratoriotilassa.

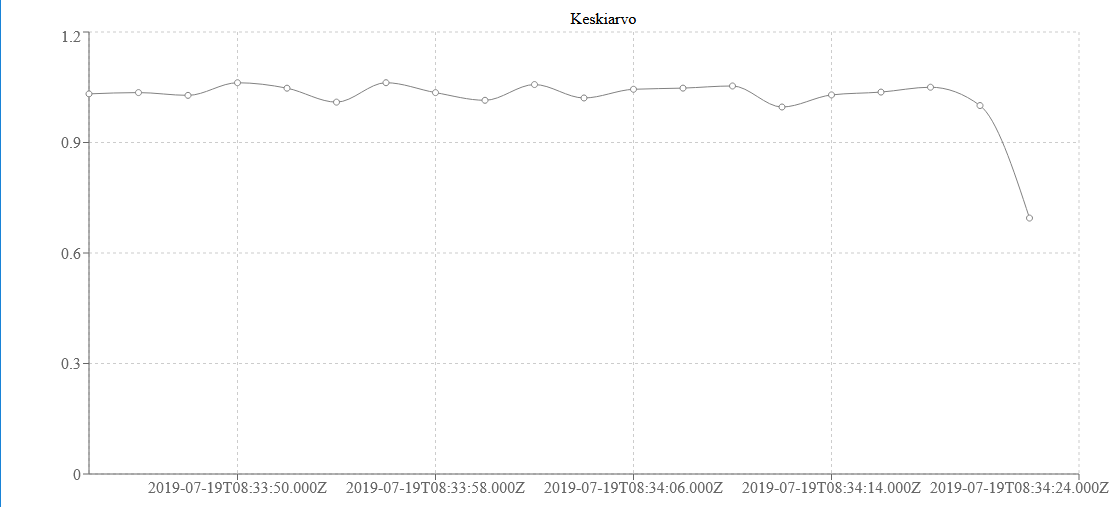
Hidas kävely, testi 1:



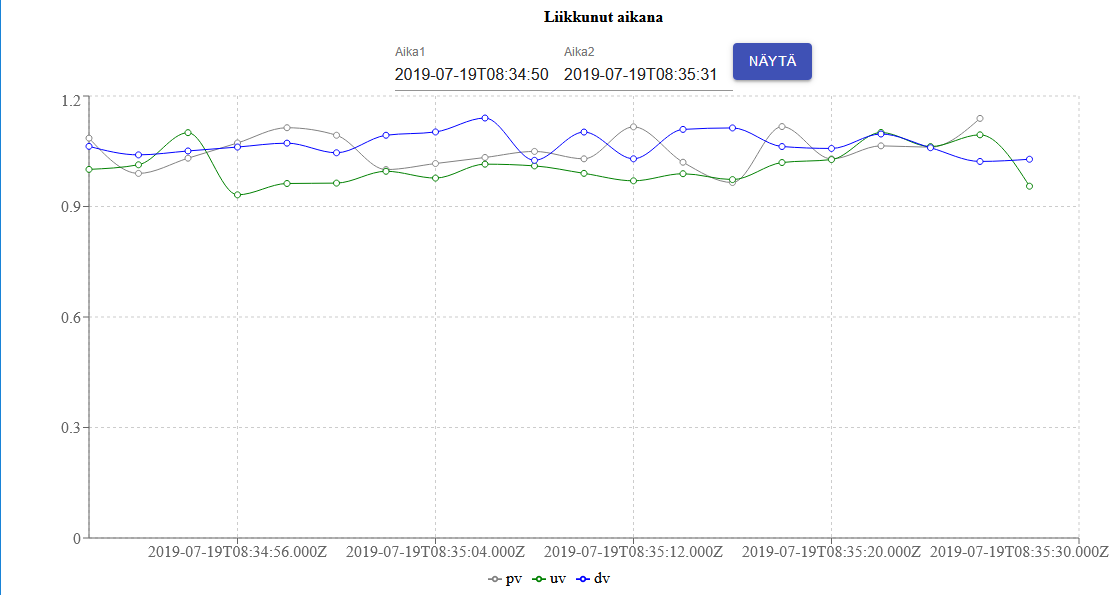


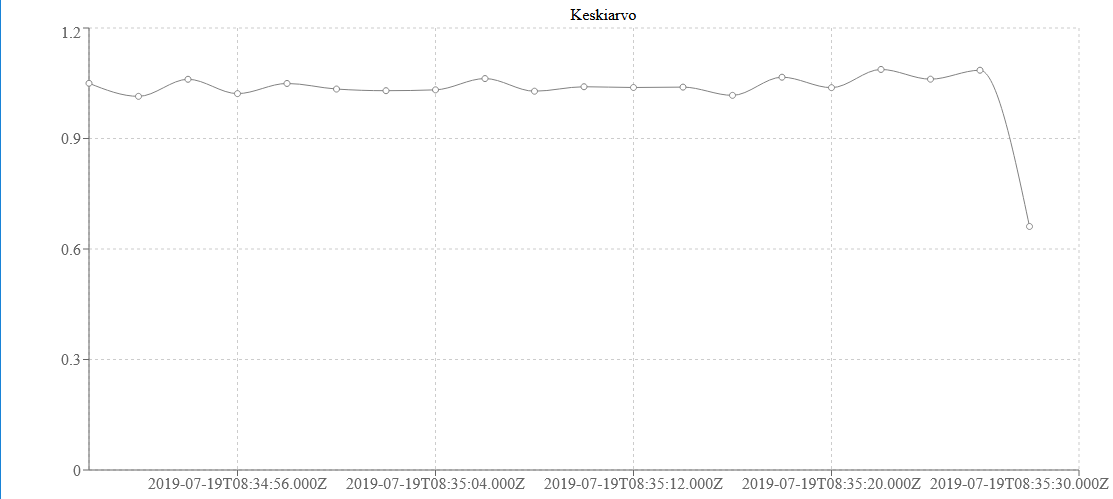
Hidas kävely, testi 2:





Hidas kävely, testi 3:

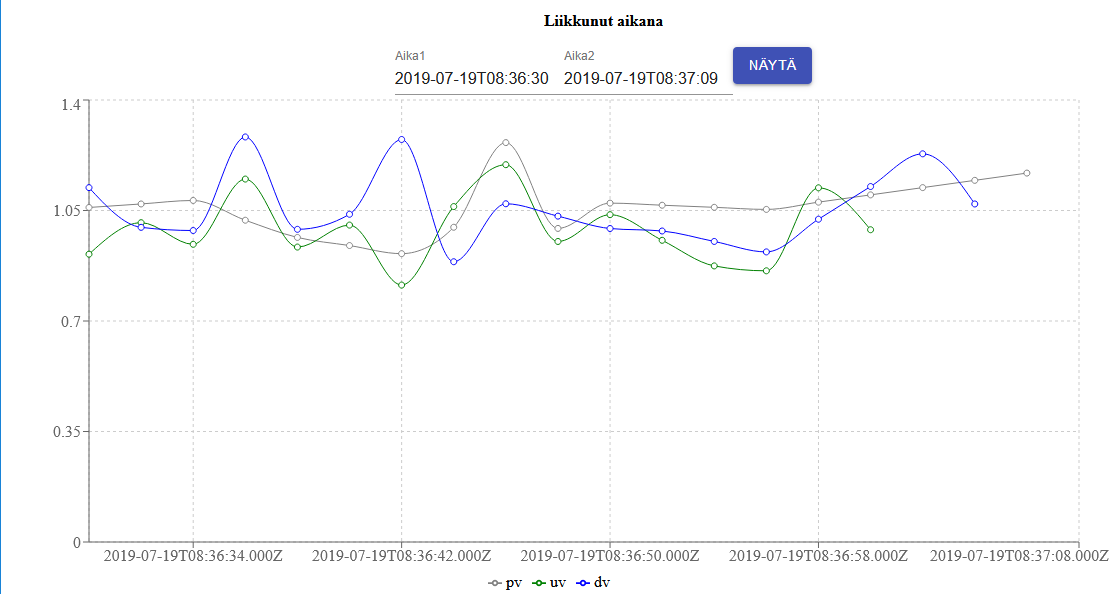


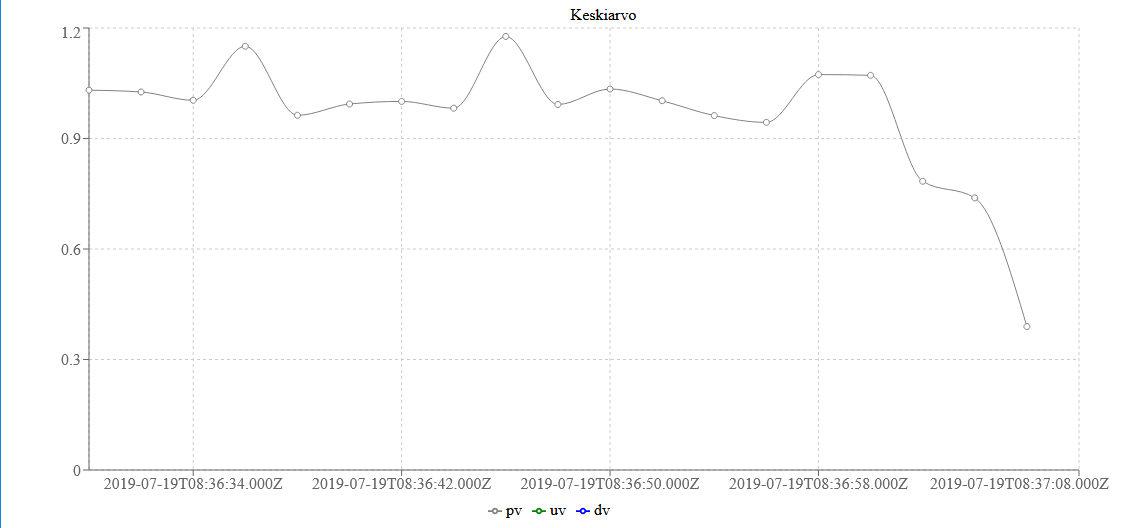


Normaali kävely:

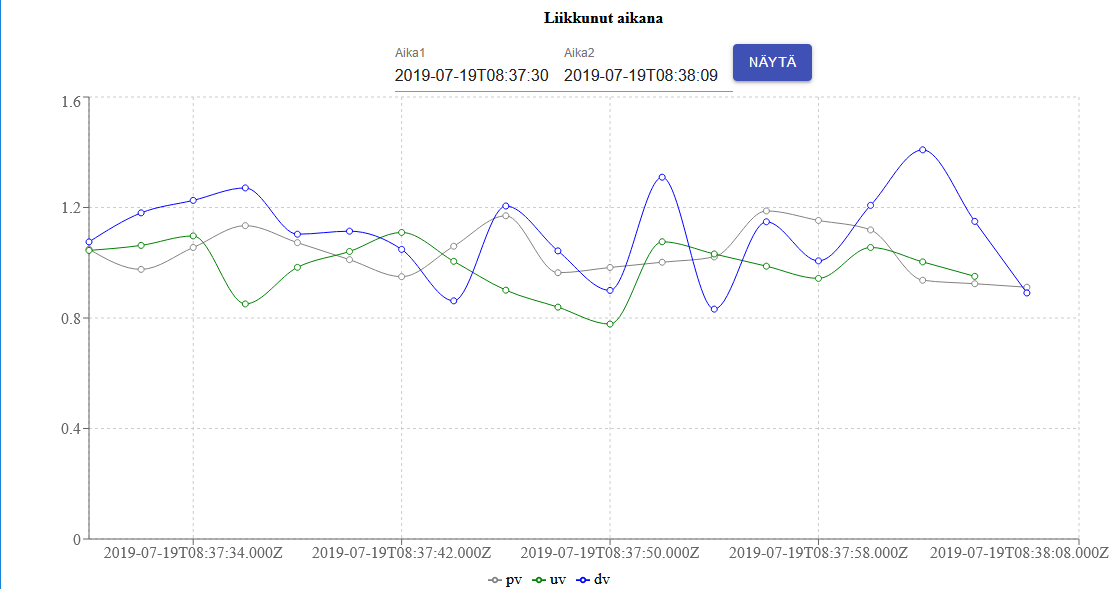
Kaksi kahdeksikon muotoista kierrosta laboratoriotilassa.

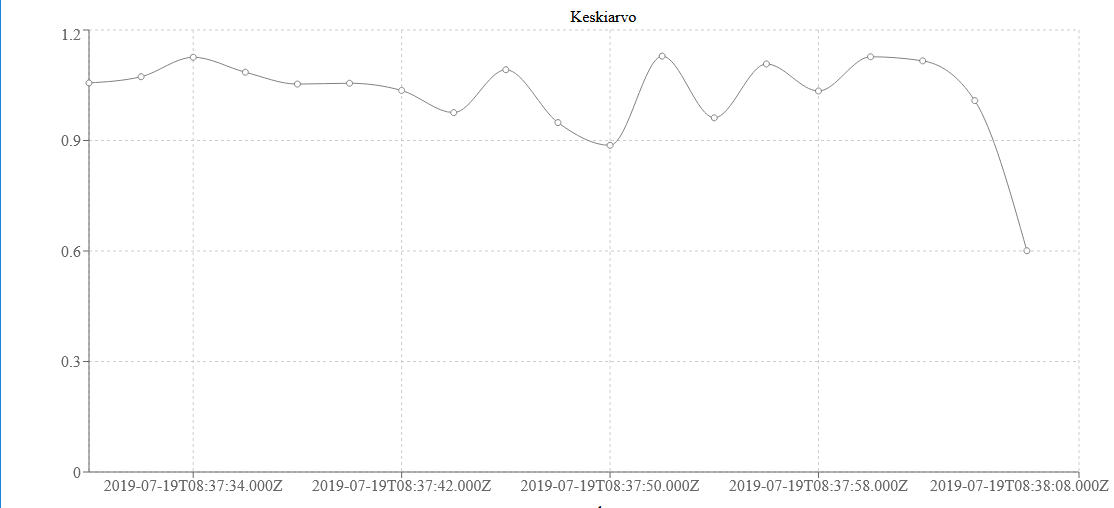
Normaali kävely, testi 1:



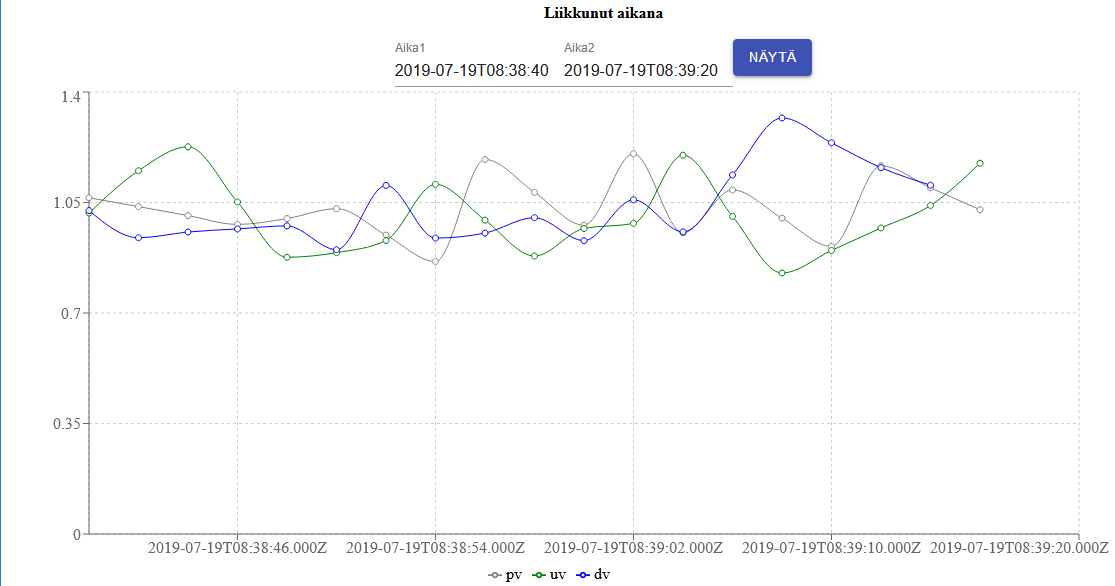


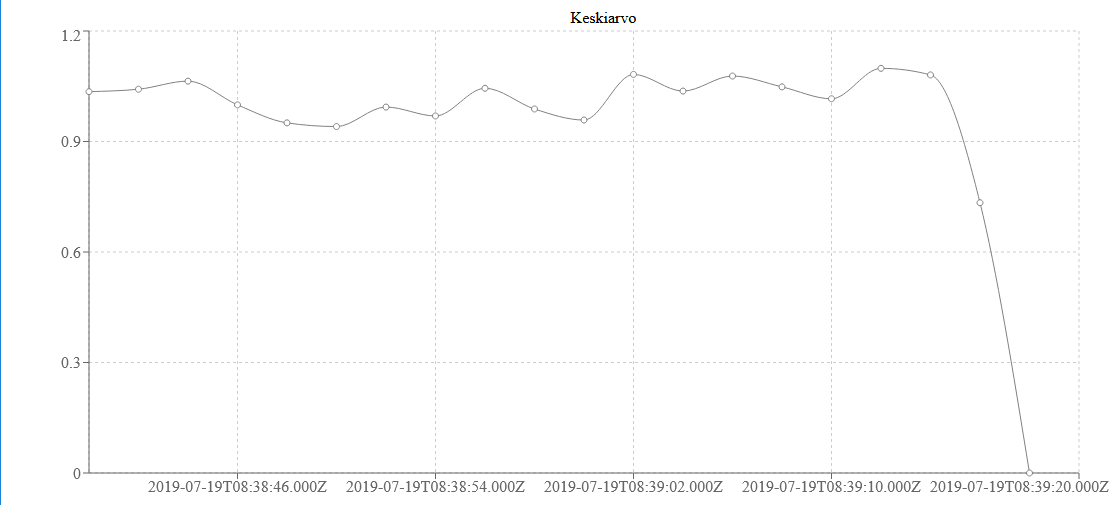
Normaali kävely, testi 2:





Normaali kävely, testi 3:

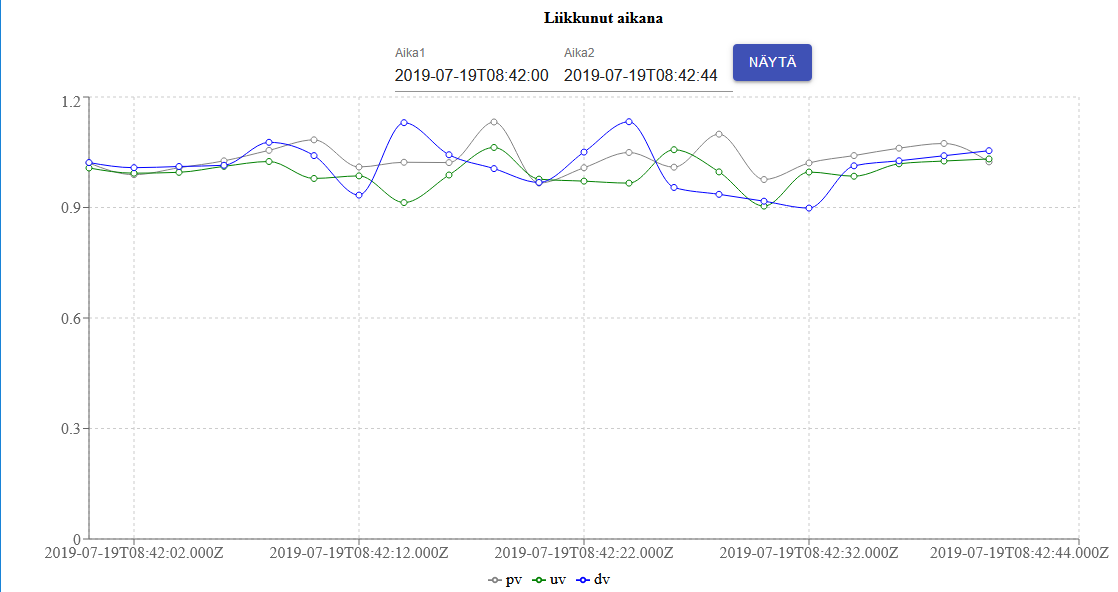


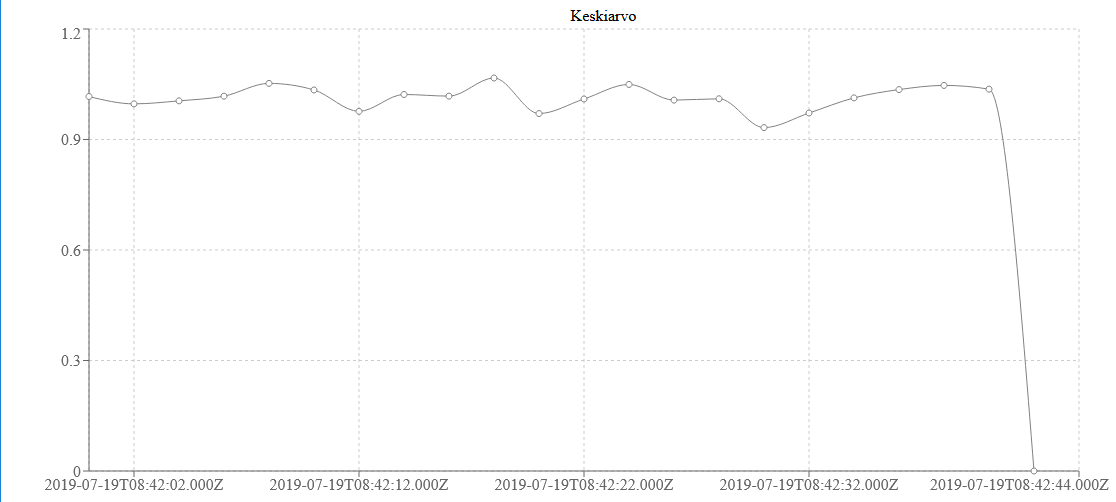


**’Rollaattorikävely’:**

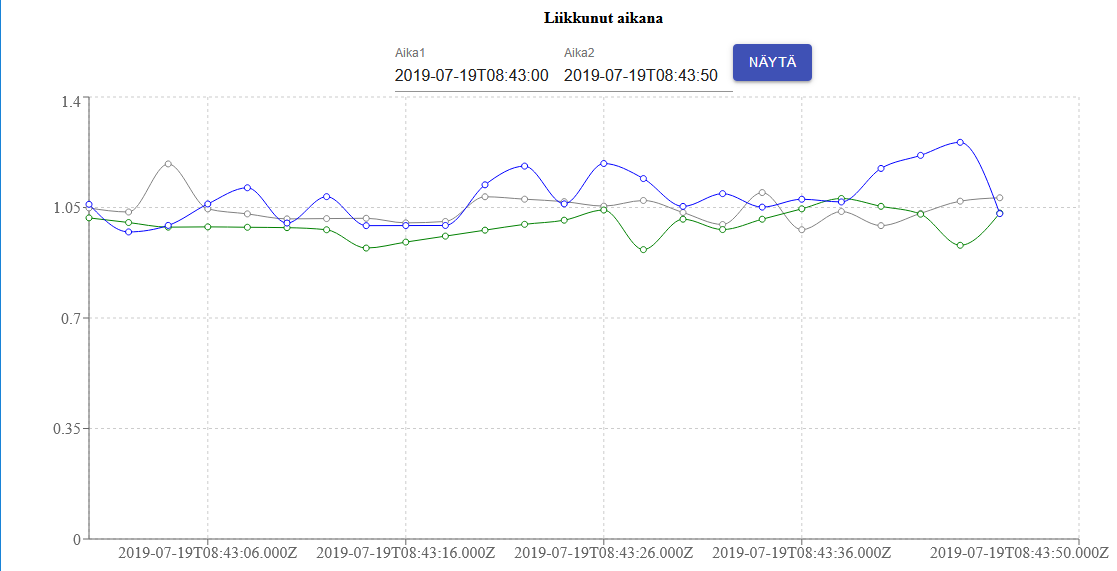
Yksi hitaahko kävelykierros laboratoriotilassa, jossa ’paluumatkalla’ loivia kaarroksia kuvaten käyttäjän huojuvaisuutta.

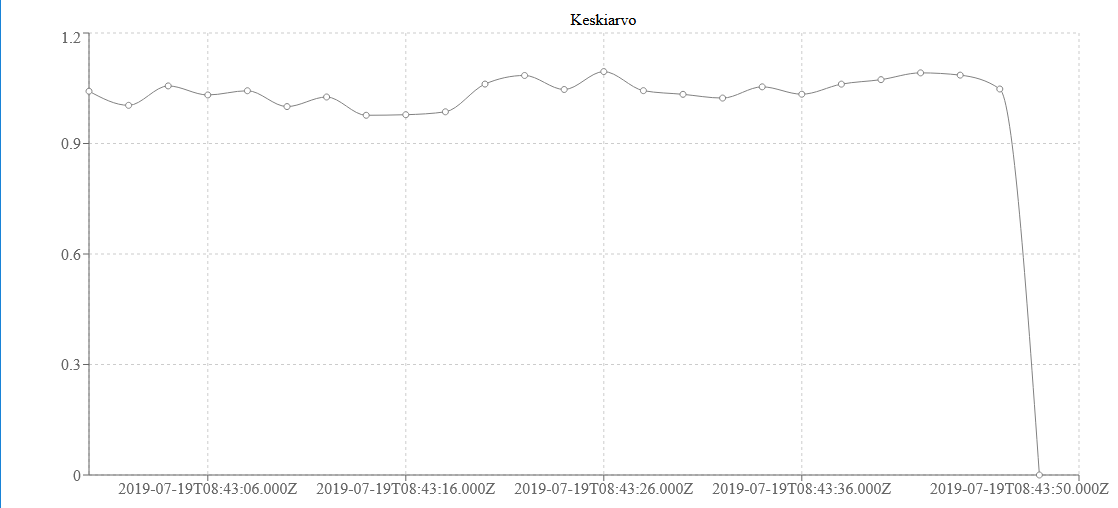
’Rollaattorikävely’, testi 1:



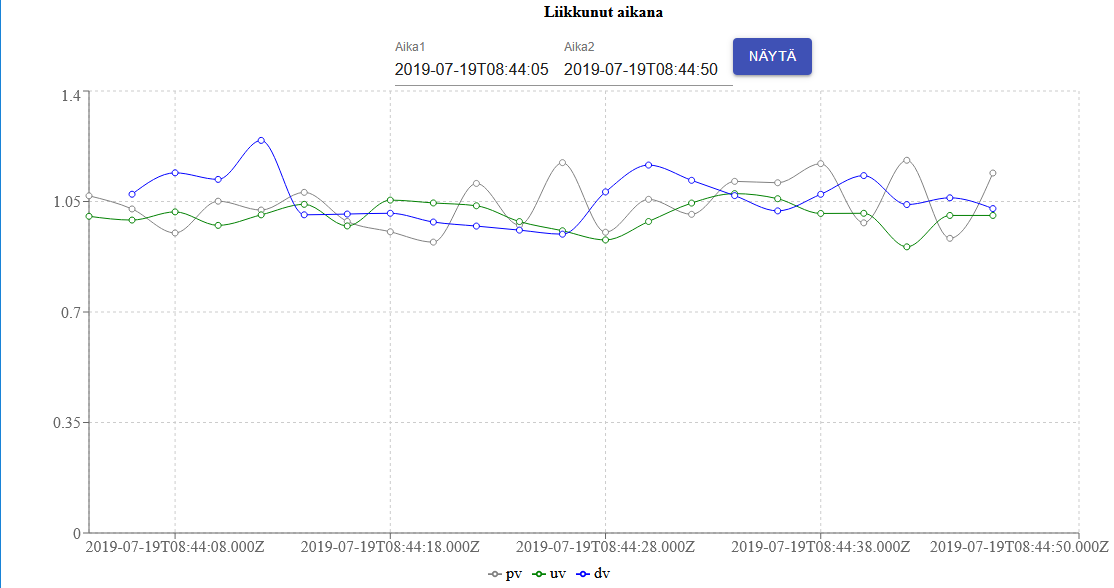


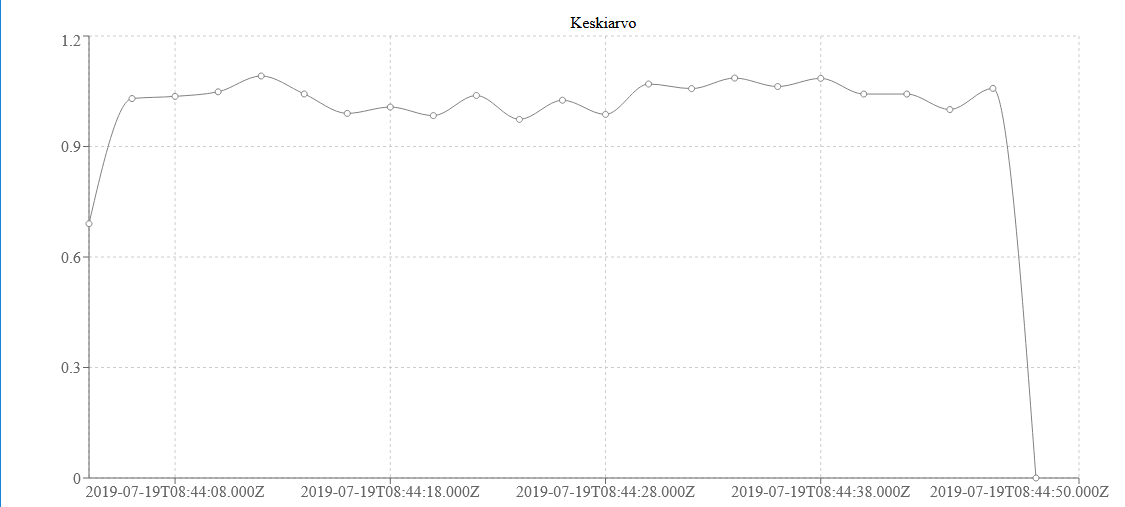
’Rollaattorikävely’, testi 2:





’Rollaattorikävely’, testi 3:





**22.7.2019**

Ohjelman hitauden takia testimielessä päätettiin asentaa sovellus luokan yhdelle pc:lle (testilaboratorion ympäristössä IP:10.100.0.119).

Pc:hen liitettiin aluksi usb-bluetooth adapteri.

Tämän jälkeen asennettiin RuuviCollector ja Influx.

Ohjelman todettiin toimivan pc:llä huomattavasti nopeammin kuin Raspberry Pi:llä.

Tietokantahaut kestivät nyt vain muutaman millisekunnin, kun ne Raspberry Pi:llä kestivät n, 100-20000 ms.

Esimerkki Ohjelman saamasta

datasta

<http://localhost:5000/all>

Kaikkien RuuviTag ohjelman

tarvitsemat tiedot

mean = temperature

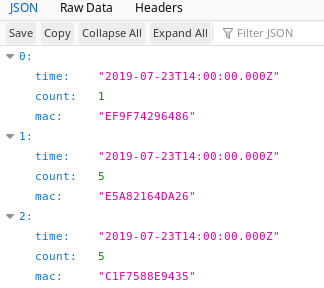
mean\_1 = rssi

mean\_2 = AccX

mean\_3 = AccY

mean\_4 = AccZ

mean\_5 = AccTotal

<http://localhost:5000/mdata>

Ohjelman tallentamat liikkumiset

viimeisen tunnin ajalta

<http://10.100.0.119:5000/time2?q=2019-07-23T10:40:00Z&q2=2019-07-23T10:41:00Z&mac=EF9F74296486>

Käyttäjän asettaman aikavälin

data per mac osoite

mean = totalAcceleration

**ARDUINO OSUUS**

Käytetty Arduino mikrokontrolleri ja pari sensoria LiikkuVaxi projektin toisena tietolähteenä:

- "oven avaamisen" simulointi ja rekisteröinti

- henkilön etäisyyden (0-100cm) mittaaminen (esim. lähestyminen ikkunaan) ja rekisteröinti

LiikkuVaxi projektin Arduino -osuuden arkkitehtuuri

1. Arduino kit - kerää data sensoreista ja lähettää sarjaportin kautta Node.ja serverille

2. Node.js - Arduino-NodeJS-MySQL-Serial-Data-Logger serveri, kuuntelee sarjaporttia, vastaanottaa dataa ja laitaa sitä tietokantaan

3. MySql - kantamoottori, säilyttää rekisteröity dataa

4. LiikuVaxi sovellus - ottaa yhteys MySql tietokantaan ja esittää dataa graafissa

Arkitektuuria vastaavat resurssit:

1.

https://www.arduino.cc

- pääsivu Arduino alustalle

https://www.arduino.cc/en/Main/Software

- Arduino IDE lataus (tarvitaan tässä projektissa)

http://wiki.epalsite.com/index.php?title=Starter\_Kit\_for\_Arduino

2.

Arduino-NodeJS-MySQL-Serial-Data-Logger: sends values over Serial and insert into a database

https://github.com/tttapa/Projects/tree/master/Arduino/NodeJS/Database/Arduino-NodeJS-MySQL-Serial-Data-Logger

- tässä lähdekoodit on hyvä kuvaus koskien kaikkia kolmea palokkaa, eli MySql kannan luontiskriptit, Node.js serverin koodi

ja Arduino projektin ohjelmakoodi (Arduino-NodeJS-MySQL-Serial-Data-Logger.ino - tiedosto)

3.

MySql kantamoottorin asennus:

XAMPP Apache + MariaDB (MySql) + PHP + Perl

Löytyy paikasta:

https://www.apachefriends.org/index.html

Arduino Uno alusta, kytketty DC Servomoottori, HC-SR501 PIR Motion Sensor Module

ja HC-SR04 Ultrasonic Distance Measurement module.

Arduino kytketty USB/Sarjaportin kautta Win koneeseen, jossa pyörii Node.js serveri,

joka kuuntelee sarjaporttia, kerää dataa ja lähettää sitä MySql kantaan.

Arduino Starter Kit, Beginner Level

Servomoottorilla "oven avaamisen" simulointi.

Pohjana voi käyttää Arduino Projects Book projekti 12. Knock Lock, sivu 124

Projektissa on käytetty sensoreita ja muita kompponentteja Professional UNO R3 Starter Kit for Arduino versiosta,

jota todettu mukavempi käyttää.

Lähellä kokoonpanosta oleva kitti:

http://www.epalsite.com/store/kits/starter-kit-for-arduino-infiduino-uno-r3.html

Starter Kit for Arduino/Infiduino Uno R3

Piezo anturi vaihdettu HC-SR501 PIR Motion Sensor Moduliin, kun se tunnistaa liikkeen,

antaa käskyn oven avaamiselle - tapahtuman tyyppi ja aikaleima tallennetaan kantaan.

Toisena toiminnallisuutena/tietolähteenä kokeilun mielessä

on lisätty HC-SR04 Ultrasonic Distance Measurement module,

tunnistaa etäisyyden ja tallentaa se kantaan (rekisteröidään 1-100cm väli).

Projektien kuvaukset ja esimerkkikoodit:

http://wiki.epalsite.com/index.php?title=Starter\_Kit\_for\_Arduino

Tässä projektissa yhdistetty osat kolmesta esimerkkiprojektista:

1. Lesson 25 PWM Servo Control

2. Lesson 32 PIR Motion Sensor playing

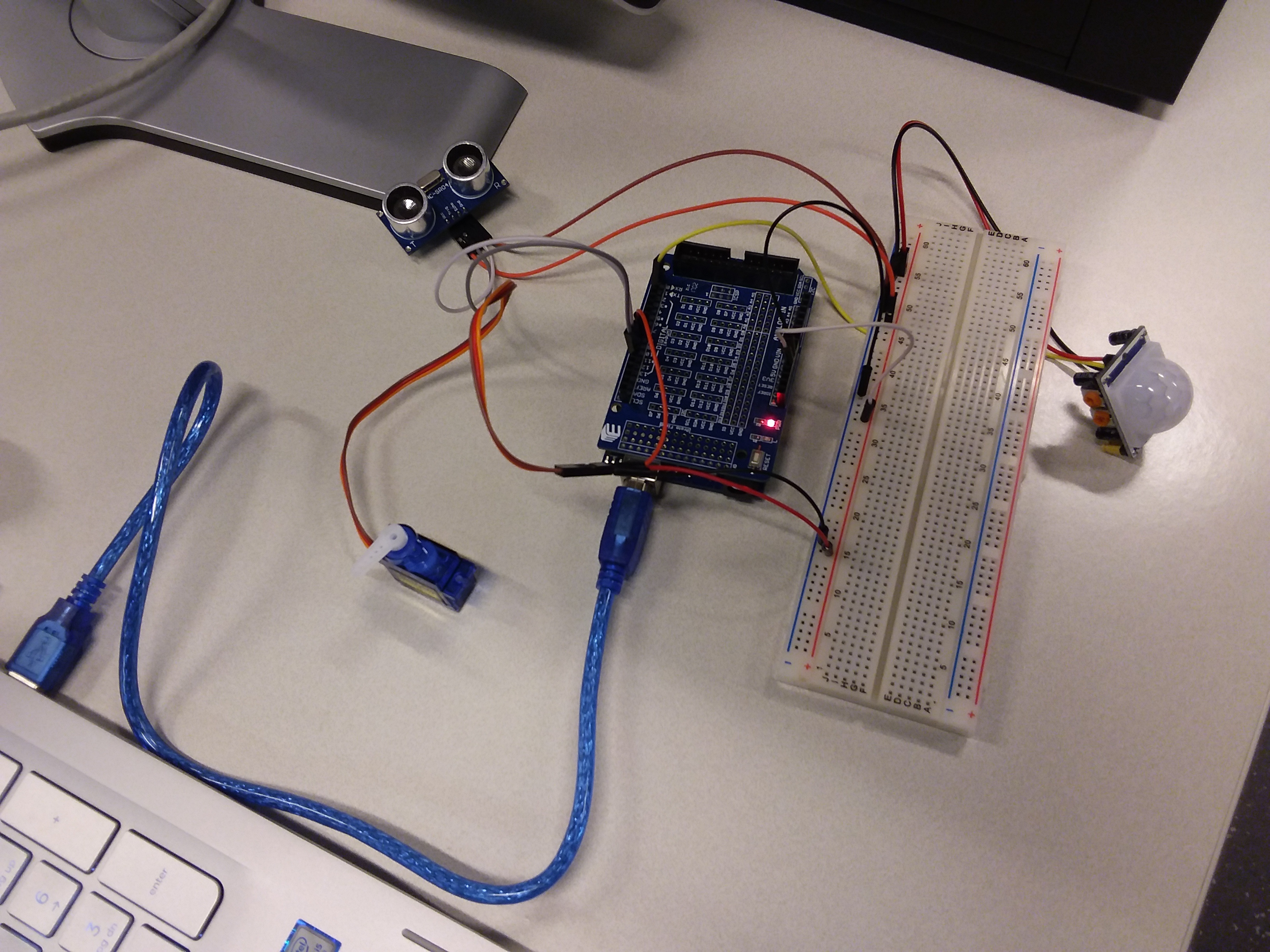
3. Lesson 33 Ultrasonic Distance Measurement

ks. linkki ja seuraa ohjeita:

http://wiki.epalsite.com/index.php?title=Starter\_Kit\_for\_Arduino#Lesson\_25\_PWM\_Servo\_Control

http://wiki.epalsite.com/index.php?title=Starter\_Kit\_for\_Arduino#Lesson\_32\_PIR\_Motion\_Sensor\_playing

Arduino projektin lähikuva



Arduino skatchin koodi (nopean prototyyppikokeilu tyylinen):

/\*Note:This code is used for Arduino 1.0.1 or later\*/

/\*

PIR Controlled LED

Turns on an LED when human motion is detected.

\*/

// import the library

#include <Servo.h>

#include <NewPing.h>

#define TRIGGER\_PIN 12 // Arduino pin tied to trigger pin on the ultrasonic sensor.

#define ECHO\_PIN 11 // Arduino pin tied to echo pin on the ultrasonic sensor.

#define MAX\_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.

// create an instance of the Servo library

Servo myServo;

void setup() {

// initialize the digital pin as an output.

Serial.begin(115200);

pinMode(2, INPUT\_PULLUP);

pinMode(13, OUTPUT);

// attach the servo to pin 9

myServo.attach(9);

// move the servo to the unlocked position

myServo.write(0);

}

void loop() {

delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).

if ((uS / US\_ROUNDTRIP\_CM > 0) and (uS / US\_ROUNDTRIP\_CM < 101))

{

Serial.println(uS / US\_ROUNDTRIP\_CM); // Convert ping time to distance in cm and print result (0 = outside set distance range)

}

int sensorValue = digitalRead(2);

if(sensorValue)

{

digitalWrite(13, HIGH); // set the LED on

myServo.write(90);

Serial.println("Ovi auki");

sensorValue=0;

// move the servo to the locked position

delay(5000);

// move the servo to the locked position

myServo.write(0);

Serial.println("Ovi kiinni");

}

else {

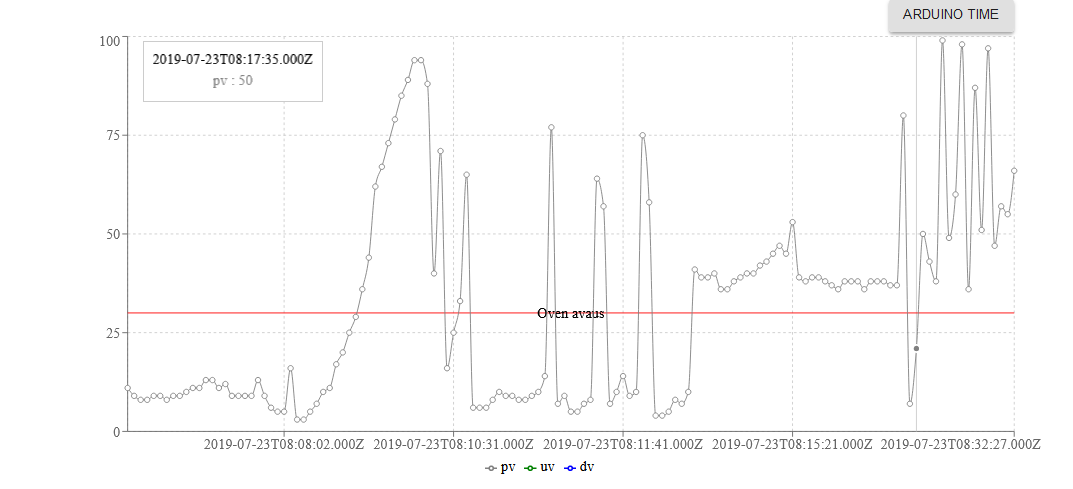
digitalWrite(13, LOW); // set the LED off

}

}

**ARDUINO TIME**

- Arduino sensoreilla kärätyn daytan näyttäminen LiikkuVaxi ohjelman puolella



**CSV DATA**

Otettu myyntipakkauksesta testattavaksi Garmin Vívosmart 4 –aktiivisuusranneke.

Suoritettu rannekkeen ensiasennus ja yhdistäminen mobileappiin (Android) sekä luotu Garmin Connect -tili myyntipakkauksessa mukana seuranneen Aloitusoppaan mukaisesti.

Syötetty manuaalisesti pari harjoitusta sovellukseen Suoritukset/Kaikki suoritukset –kohdassa. Myöhemmin tehty yksi todellinen kävelyharjoitus, jonka tulos tallentui samaan paikkaan.

Edelleen sovelluksen samasta kohdasta (Suoritukset/Kaikki suoritukset) ajettu edellä kerätyt suoritukset toiminnolla ’Vie CSV-tiedostoon’ pc:n kovalevylle .csv-muotoisen tiedostoon.

Tämä tiedosto siirretty lokaalitiedostoksi LIIKKUVAXI-ohjelman käyttöön, jonne se saatiin noudettua JSON-muotoisena.

Seuraavassa kuvassa esillä tiedot, jotka näin saadaan ulos Garmin Connect –sovelluksesta.

