Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Tietojärjestelmien kehitys

Skoutti Sami

RaspberryPi:n digitaaliset liitännät

Opinnäytetyö 2015

Tiivistelmä

Skoutti Sami
RaspberryPi:n digitaaliset liitännät, 37 sivua, 3 liitettä
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikka
Tietojärjestelmien kehitys
Opinnäytetyö 2015
Ohjaajat: Huhtanen Mikko, lehtori, Saimaan Ammattikorkeakoulu

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa pieni mittaus- ja ohjausjärjestelmä käyttäen hyväksi RaspberryPi-laitetta. Tehtävänä oli myös toteuttaa siitä etäohjattava internetin yli.

Teoriaosuudessa on käsitelty mittaus- ja ohjausjärjestelmän rakentamisprosessia ja sen käyttöä. Asiakirja pitää sisällään tietoja käyttöjärjestelmästä, itse Raspberry Pi-laitteesta ja siinä käytettävistä lisälaitteista sekä tekniikoista.

Työn tuloksena syntyneillä työkaluilla saatiin aikaan toimiva kokonaisuus, jossa Raspberry PI toimii mittaus- ja ohjausjärjestelmänä ja Raspberry PI GPIO-liitäntöjen ohjaus onnistuu BerryIO:n hallintasivun avulla internetin kautta.

Asiasanat: Raspberry, BerryIO, Linux, GPIO, DS18B20, AM2302

Abstract

Skoutti Sami
Digital interfaces of RaspberryPI, 37 pages, 3 Appendices
Saimaa University of Applied Sciences
Technology Lappeenranta
Information Technology
Software Engineering
Bachelor´s Thesis 2015
Instructor: Huhtanen Mikko, lecturer, Saimaa University of Applied Sciences

The purpose of the study was to create a small measuring and control system using Raspberry Pi. Making it a remote-controlled system over the internet was also researched as part of the work.

The process of creating and using the measurement and control system is described in this document. The document contains information about the operating system, Raspberry device itself and last accessories used in it, as well as technologies.

The tools created during this study comply with the demands of the commissioner. Instructions to help the commissioner in using these tools were also produced.

Keywords: Raspberry, BerryIO, Linux, GPIO, DS18B20, AM2302, Cron

Sisällys

Käytetyt lyhenteet

HTTP	Hypertransfer Protocol on tiedonsiirtoprotokolla, jota selaimet ja palvelimet käyttävät.
USB	Universal Serial Bus on tietoliikenneportti oheislaitteiden liittämiseksi tietokoneeseen.
IP	TCP/IP internet-kerroksen Internet Protokolla
ARM	Advanced RISC Machine, 32-bittinen mikroprosessoriarkkitehtuuri.
SDHC	Secure Digital High Capacity, muistikorttityyppi
FAT32	Microsoftin kehittämä tiedostojärjestelmä
GPIO	GPIO (General Purpose I/O) on yleiskäyttöinen portti mikrokontrollereissa ja mikroprosessoreissa.
Kernel	Käyttöjärjestelmän ydin joka määrittelee käyttöjärjestelmän ominaisuudet, sekä rakenteen.
CPU	Central Processing Unit joka suorittaa ohjelman sisältämiä konekielisiä käskyjä.
CLI	Komentorivi joka toimii ihmisen ja tietokoneen välisenä kommunikointisiltana.
Firmware	Laiteohjelmisto joka huolehtii perustoiminnoista.
IoT	Internet of Things, tarkemmin esineiden internet.
PowerShell	Microsofting kehittämä komentotulkki Windows käyttöjärjestelmälle.
CSI	Camera Serial Interface, joka on kameroille tarkoitettu tiedonliikenneväylä
ISP	Internet Service Provider, internet palveluiden tarjoaja.

1 Johdanto

Tämän opinnäytetyön tarkoituksena on selvittää, miten luottokortin kokoista Raspberry PI-tietokonetta voidaan käyttää internetin yli mittaus- ja ohjausjärjestelmänä.

Laite on Raspberry PI -säätiön kehittämä tietokone jonka perimmäisenä tarkoituksena on kannustaa nuoria opiskelemaan tietotekniikkaa ja ohjelmointia kouluissa.

Raspberry PI:n GPIO-porttien ja vähäisen tehonkulutuksen ansiosta sitä voidaan käyttää elektroniikkaprojekteissa. GPIO-portteja käytetään muun muassa lukemaan lämpötila-antureista tietoja, jotka siirrettään tietokantaan. GPIO-portteja hyväksi käyttämällä voidaan ohjata moottoreita, servoja, venttiileitä sekä releitä.

Tässä opinnäytetyössä on ajateltu ohjattavaksi kasvihuonetta. Kasvihuone voi sijaita paikalla jossa ei ole saatavilla verkkovirtaa tai kiinteää internetyhteyttä, mutta kuitenkin mobiiliyhteyden läheisyydessä jolloin kasvihuoneen hallinta tapahtuu 3g/4g-verkon kautta.

Asiakas huolehtii itse veden, sekä sähköisistä kytkennöistä, jotta vältytään mahdollisista onnettomuuksista ja suurista taloudellisista menetyksistä kiinteille rakennuksille.

2 Linux

Tässä luvussa käydään läpi Linux-käyttöjärjestelmän perustietoja, asennus, peruskonfiguraatio sekä lisenssitiedot.

2.1 Historia

Linux on avoimeen lähdekoodiin perustuva vuonna 1990 Linus Torvaldsin aloittama Unixin kaltainen käyttöjärjestelmä. Linux kernel versio 1.0 julkaistiin vuonna 1994. Linux eli tarkemmin Kernel on käyttöjärjestelmän ydin, jonka tehtävän on hallinnoida kaikkea kommunikointi alemman laitetason kanssa. Nykyisin Linuxia kehittää useat vapaaehtoiset yksityiset kehittäjät, sekä suuria määriä eri yrityksiä. Linus Torvalds kuitenkin johtaa Kernelin kehitystyötä ja kaikki viralliseen vanilla-kerneliin menevät muutokset joutuvat hänen hyväksyttäviksi. Kernelistä on myös saatavilla useita eri haaraumia joita hallinnoivat jakelupakettien ylläpitäjät. (cs.cmu.edu, 2015)

Linux on avoimeen lähdekoodiin perustuva käyttöjärjestelmä, joka käyttää GNU GPL v2–lisenssiä. Kyseinen lisenssi antaa mahdollisuuden kenen tahansa muokata sitä omien mieltymysten mukaan tiedettyjen edellytyksin. (Kernel.org, 2015)

Linuxia käytetään monenlaisissa laitteissa, matkapuhelimissa, kelloissa, pelikoneissa, parkkimittareissa, autoissa, laskimissa, robotiikassa. Linux-käyttöjärjestelmä on suurelta osin kirjoitettu C- ja c++-ohjelmointikielellä ja myös Python-ohjelmointikielellä. (labor-liber.org, 2004)

2.2 Mikä on Linux

Puhekielessä yleensä käytetään Linuxia kuvaamaan koko käyttöjärjestelmää, todellisuudessa niin ei kuitenkaan ole, vaan viitataan suoraan Linux kerneliin. Tarkemmin termillä GNU/Linux viitataan koko käyttöjärjestelmään joka pitää sisällään kaikkia muita apuohjelmia.

2.3 Lisenssit

Linux kernel julkaistaan General Public Licenseä (GPL) eli vapaiden ohjelmistojen julkaisemiseen tarkoitetulla lisenssillä. GPL lisenssi on hyvin avoin ja mahdollistaa kenen tahansa käyttää, muuttaa, kopioida ja jakaa edelleen lähdekoodia ja ohjelmistoa. GPL lisenssi ei estä koodin kaupallista käyttöä, minkä takia se toimii hyvänä pohjana koko avoimen lähdekoodin yhteisölle. Ohjelmiston käyttäjä saa myydä tuotteen kopioita tai muunnelmia, kunhan noudattaa lisenssin asettamia ehtoja:

- Lisenssiä ei saa poistaa tai muokata.
- Ohjelman tulee sisältää tietoja lisenssistä.
- Jatkokehitetyt teokset tulee lisensoida samalla lisenssillä
- Lähdekoodin tulee olla saatavilla kaikille, myös kaupallisista ohjelmista.
- Lähdekoodin muokkauksista tulee tehdä merkintä (sofokus.com, 2015)

(NOTE! This copyright does not cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does not fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.Also note that the only valid version of the GPL as far as the kernel is concerned is this particular version of the license (ie v2, not v2.2 or v3.x or whatever), unless explicitly otherwise stated.) (Torvalds, 2010)

2.4 Jakelupaketit ja version valinta

GNU/Linuxista on saatavilla useita satoja eri distroja, tarkemmin jakelupaketteja. Jotkin näistä jakelupaketeista on suunniteltu vain yhtä tarkoitusta varten, esimerkiksi mediakeskus, palomuuri tai palvelin, mutta osa on suunniteltu moneen eri käyttöön.

Versiota valittaessa tulee määrittää omat tarpeensa ja varmistaa, ettei tarjolla ole jo valmista jakelupakettia kyseiseen tarkoitukseen. Jakelupakettia valittaessa tulee myös varmistaa, että jakelupaketti soveltuu käyttöön. Esimerkiksi palvelinta varten jakelupakettia valittaessa tulisi miettiä tarvitaanko graafista käyttöliittymää ollenkaan. Sen pois jättämisellä säästetään jonkin verran resursseja, muistia ja CPU-aikaa ja tietoturvan kannalta saadaan suljettua ylimääräisiä prosesseja pois päältä, mikä mahdollistaa pienemmän hyökkäyspinnan.

2.4.1 Raspbian

Raspbian on avoimeen lähdekoodin käyttöjärjestelmä, joka perustuu Debian jakelupakettiin, joka on optimoitu toimimaan Raspberry Pi-laitteistolla. Raspbian on epävirallinen Debian julkaisuun perustuva haarauma, koska virallinen Debian armhf julkaisu on yhteensopiva vain uudempien ARM-arkkitehtuuriin perustuvien prosessorien kanssa. (raspbian.org, 2015)

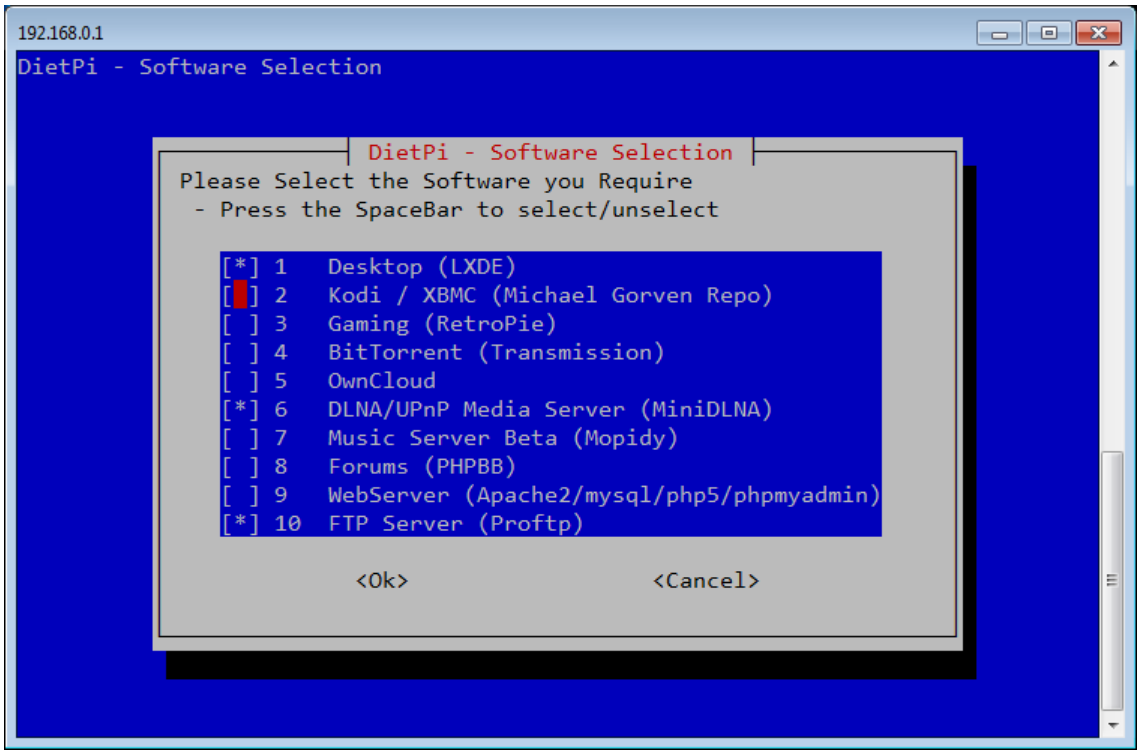
Raspbianin kääntäjä on säädetty tuottamaan koodia jossa on rautapohjainen liukuluku päällä (Hardware Floating Point) ABI (Application Binary Interface) ja sen suorittamisen. (raspbian.org, 2015)

Muilta osin Raspbian on Debianin kaltainen käyttöjärjestelmä jossa tulee mukana graafinen käyttöliittymä tai CLI (command-line interface), sekä mukana tulee yli 35,000 esikäännettyä pakettia. Suuri määrä esikäännettyjä paketteja mahdollistaa uusien ohjelmien nopean asennuksen, koska niitä ei tarvitse erikseen kääntää hitaammalla Raspberry Pi-laitteistolla. (raspbian.org, 2015)

2.4.2 DietPI

DietPI on Raspbianiin perustuva jakelupaketti, joka on suunniteltu palvelinkäyttöä silmällä pitäen. DietPI:n erikoisuutena on se, että siitä on pyritty poistamaan kaikki ylimäärämäiset ja tarpeettomat ohjelmat sekä niiden riippuvuudet.

Käyttöjärjestelmän muistikortille asennuksen jälkeen voidaan suorittaa komento ”dietpi-software” jonka avulla tarvittavien lisäohjelmien asennus tapahtuu helposti CLI:n kautta. Kuva 1.



Kuva 1 - DietPi:n Ohjelmiston valinta

Mikäli mitään ylimääräisiä ohjelmia ei asenneta, käytössä on toimiva, mutta hyvin pelkistetty Linux. DietPi:n ominaisuuksia:

- Suorituskykyisempi Raspbian käyttöjärjestelmä
- Käyttöjärjestelmä mahtuu 360mb levytilalle
- Mahtuu hyvin ~1GB muistikortille.
- Käytössä olevia prosesseja on vähän (13)
- Muistin käyttö on noin 20MB käynnistyksen jälkeen.

2.4.3 Hankkiminen

Helpoin tapa hankkia Raspbian tai DietPI on ladata ne netistä omalta sivulta:

- Raspbian (http://www.raspbian.org/)
- DietPI (http://fuzon.co.uk/phpbb/viewtopic.php?f=8&t=6)

RaspberryPi:lle on myös saatavilla useita kymmeniä eri Linux-pohjaisia käyttöjärjestelmiä. Kattavan listan niistä ja niiden ominaisuuksista ylläpidetään osoitteessa: http://elinux.org/RPi_Distributions

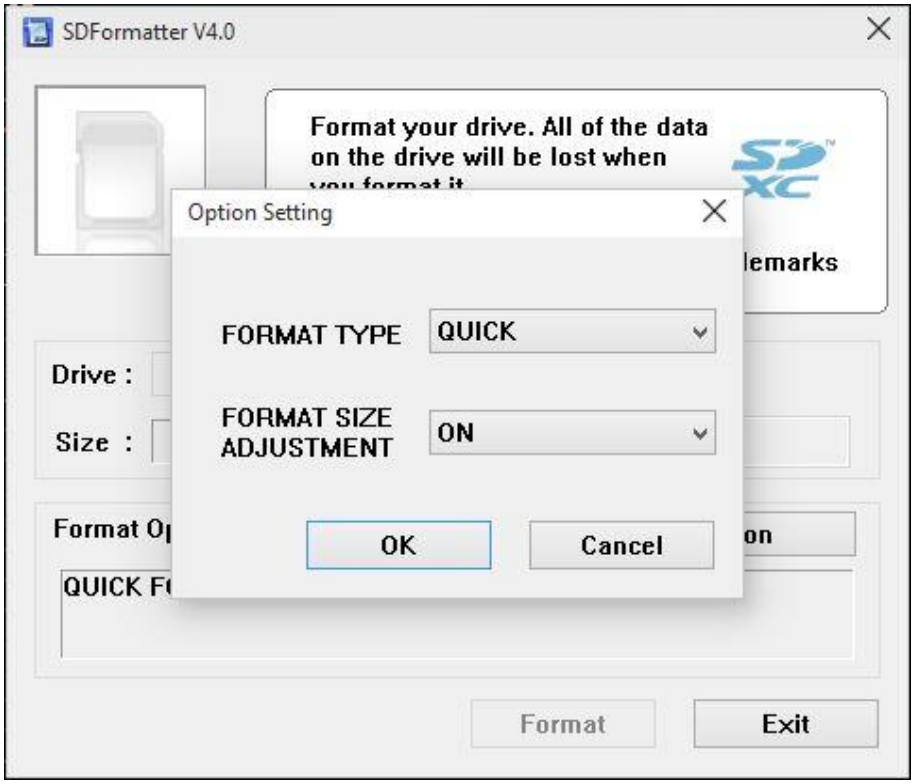
2.5 Asentaminen

Tässä kohdassa käydään läpi Linux-käyttöjärjestelmän asentamiseen liittyviä asioita sekä asentamisprosessi Raspberry Pi-laitteelle.

2.5.1 Huomioitavaa ennen asentamista

Ennen asennusta muistikortti täytyy alustaa FAT-tiedostojärjestelmään. Kyseisen toimenpiteen voi suorittaa myös Windows-käyttöjärjestelmän omalla levyjen alustustoiminnolla. Kuitenkin on suositeltavaa suorittaa alustus sille tarkoitetulla ohjelmalla, jotta varmistutaan siitä, että koko SDHC-kortti alustetaan jättämättä sinne tyhjiä sektoreita.

SDFormatterin voi ladata osoitteesta https://www.sdcard.org. Ohjelmaa suoritettaessa täytyy asetuksista muistaa vaihtaa 'format size adjustment' päälle. Kuva 2.

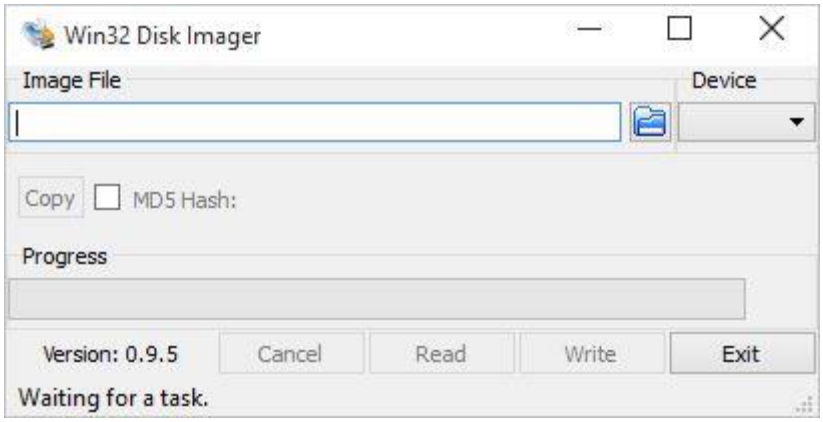


Kuva 2 - SDFormatter, muistikortin alustustyökalu

2.5.2 Asennuksen vaiheet

Käyttöjärjestelmän asennus muistikortille tapahtuu hiukan eri tavoin kuin perinteisen Linux-käyttöjärjestelmän asennus. Koska Raspberry Pi on säädetty käynnistymään ensisijaisesti ja ainoastaan SDHC-muistikortilta, ei voida käynnistää käyttöjärjestelmän asennusta USB-tallenteelta.

Käyttöjärjestelmän asennus tapahtuu purkamalla valmis asennus levykuvasta suoraan SDHC-muistikortille, josta se käynnistetään. Kuva 3.

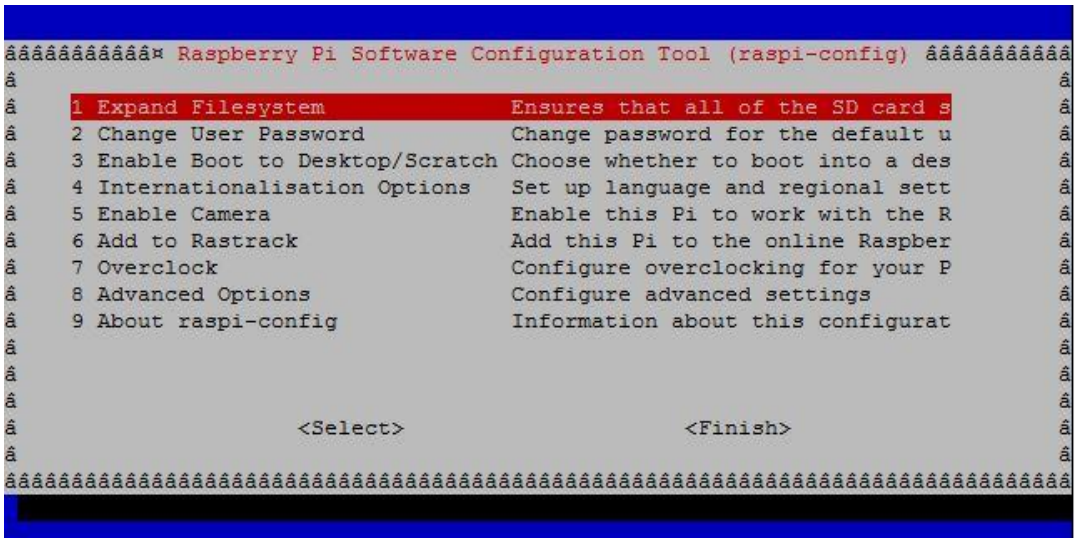


Kuva 3 - Win32formatter, käyttöjärjestelmän kopiointityökalu

Samalla win32formatter-ohjelmalla voidaan myös ottaa varmuuskopio SDHC-kortin sisällöstä. Ohjelma luo SDHC-kortin koko sisällöstä levykuvan tietokoneelle, josta se on helppo kopioida muille SDHC-korteille ja sijoittaa toiselle Raspberry Pi-laitteelle. Mikäli käytössä on useampia kymmeniä Raspberry Pi-laitteita, niin on valmiin levykuvan kopioiminen nopeampaa ja yksinkertaista.

Täytyy kuitenkin muistaa, että levykuvia kopioidessa kaikki tiedot siirtyvät, mukaan lukien salasanat, tietokannat sekä muut mahdollisesti arkaluontoiset tiedot. Mikäli ulkopuolinen käyttäjä pääsee käsiksi levykuviin, kaikki tiedot ovat luettavissa selkokielisenä.

Asennuksen jälkeen Raspbian- ja DietPI-käyttöjärjestelmässä voidaan tehdä jälkitoimenpiteet. Raspbianissa komennolla ”raspi-update” voidaan päivittää firmware ja ”raspi-config” (Kuva 4.) komennolla voidaan muokata asennusta, kuten esimerkiksi vaihtamalla salasanaa, ottamalla CSI-kameramoduuli käyttöön.



Kuva 4 - Raspi-config, RaspberryPi:n asetusten hallintatyökalu

3 Laitteet, tekniikat, menetelmät

3.1 RaspberryPI

Raspberry Pi, lähemmin Raspi, on yhden piirilevyn kokoinen tietokone, joka toimitetaan yleensä sellaisenaan ilman koteloa. Raspberry Pi tarvitsee myös muita oheislaitteita, jotta se olisi käyttövalmis. Sen on kehittänyt brittiläinen Raspberry Pi-säätiö. Raspberry Pi:n GPIO (General Purpose Input/Output) ansiosta sitä voidaan käyttää myös elektroniikkaan painottuvissa projekteissa.

3.1.1 Historia

RaspberryPi-tietokoneen kehityksestä vastaa englantilainen Raspberry Pi-säätiö. Säätiön perustettiin vuonna 2009 ja sen tarkoituksena on ollut tietotekniikan opetuksen kehittäminen kouluissa, koska lasten arvosanat ja tietotekniikan osaaminen oli laskemassa. (raspberrypi.org, 2015)

Säätiön perustajiin kuuluu Cambridgen yliopiston tietotekniikkalaboratorion jäsenet (Jack Lang, Rob Mullins, Alan Mycroft ja Eben Upton) sekä David Braben, joka aikaisemmin oli tullut tunnetuksi tietokonepelien maailmasta. Pete Lomas puolestaan osallistui Raspberry Pi:n tekniseen suunnitteluun ja valmistukseen. (raspberrypi.org, 2015)

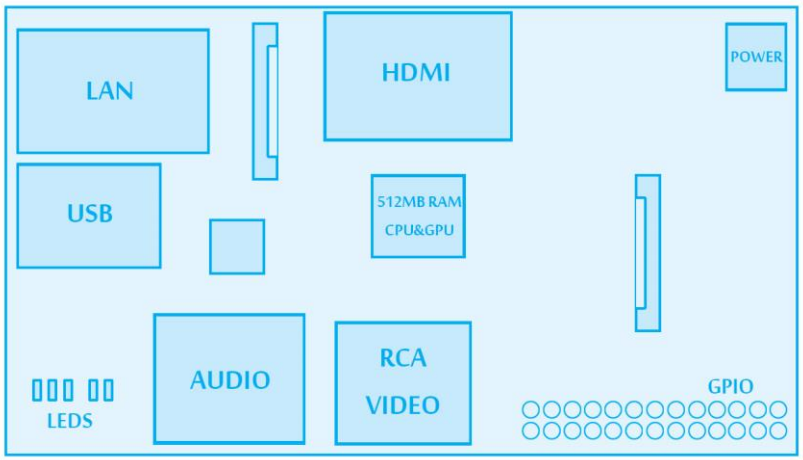
Ensimmäinen RaspberryPi julkaistiin 29.2.2012, RS Components ja Farnellin toimesta. Ensimmäinen 10,000 kappaleen erä myytiin loppuun lähes välittömästi. Vuonna 2015 säätiö ilmoitti yli 5 miljoonaa myytyä kappaletta, mikä teki Raspberrystä Britannian myydyimmän tietokoneen. (wired.co.uk, 2015)

3.1.2 Tekniset tiedot

Seuraavasta kuvasta 5 on tekniset tiedot ja kuvassa 6 on piirilevyn asettelu.

	Model A/+	Model B/+
Myyntihinta	25 USD	35 USD
Prosessori	700 Mhz BCM2835 ARM11	
Näytönohjain	Broadcom VC IV, Opengl ES 2.0, 1080p h.264	
Muisti	256MB (jaettu)	512 (jaettu)
USB 2.0	1	2
Videoulostulo	HDMI v1.4, RCA	
Ääniulostulo	HDMI, 3,5mm	
Massamuisti	SD / MMC / SDIO-muistikortti	
Verkkosovitin	Ei	10/100 RJ45 (Jaettu USB)
Muut liittimet	26-pinninen GPIO (40-pinninen A+/B+), CSI/DSI, UART	
Tehonkulutus	500mA (noin 2,5W)	700mA (noin 3 W)
Virtalähde	5v MicroUSB (2A)	
Koko:	85,60 x 53,98 x 17 mm	
Paino	45g	55g
Käyttöjärjestelmä	Linux, BSD, Solaris, Risc OS	

Kuva 5 - RaspberryPi:n tekniset tiedot



Kuva 6 - RaspberryPi, mallin B kaavakuva

3.1.3 Virrankulutus

Raspberry Pi-laitteissa on suositeltua käyttää 5v 2A virtalähdettä vaikka teknisten tietojen mukaan laitteen virran kulutus olisi vain 5v 0.7A luokkaa, koska liitettävät lisälaitteet lisäävät virrankulutusta. Raspberryssä on suositeltavaa käyttää aktiivista USB-hubia omalla virtalähteellä, mikäli laitteeseen aiotaan kytkeä useita lisälaitteita. Seuraavasta kuvasta 7 nähdään Raspi.TV-sivuston tekemän virrankulutustestien tulokset eri käytössä. (RasPi.TV, 2015). Kyseiset mittaustulokset on saatu aikaan niin, ettei Raspberry Pi:hin ole kytketty muita oheislaitteita.

	A+	A	B+	B	Pi 2 B
	/mA	/mA	/mA	/mA	/mA
Idling	100	140	200	360	230
Loading LXDE	130	190	230	400	310
Watch 1080p Video	140	200	240	420	290
Shoot 1080p Video	230	320	330	480	350

Kuva 7 - Raspberry Pi-mallien virrankulutus eri käytötarkoituksissa

Raspberryyn voidaan kytkeä virta suoraan liittämällä 5v GPIO:n P1-02 liitäntään. Kyseinen ratkaisu on hyvin käytännöllinen siinä vaiheessa, kun laitetta käytetään sellaisessa projektissa, jossa perinteinen microUSB-liitäntä ei ole saatavilla. Yksi tämän tapainen käyttötarkoitus on akkuvirran ja aurinkopaneelin avulla toimiva järjestelmä.

3.1.4 SDHC-kortin nopeus ja kestävyys

Raspberryssä tulisi suosituksien mukaan käyttää minimissään Class 6 tasoista SDHC-muistikorttia, jotta käyttöjärjestelmä tuntuisi vielä käyttäjälle reaktiiviselta.

SDHC-kortteja käyttäessä tulee myös muistaa, että kyseessä on erilainen tallennemedia kuin perinteinen kovalevy. SDHC-korteissa on vain rajattu määrä kirjoituskertoja ja jokainen kirjoituskerta kuluttaa soluja. Solujen kulumisen vuoksi tulisi välttää kirjaavia tiedostojärjestelmiä (journaling).

SDHC-muistikortin elinikää parantavia optimointeja on tarjolla runsaasti, mutta tässä opinnäytetyössä niihin ei paneuduta sen tarkemmin. Raspberryn foorumille

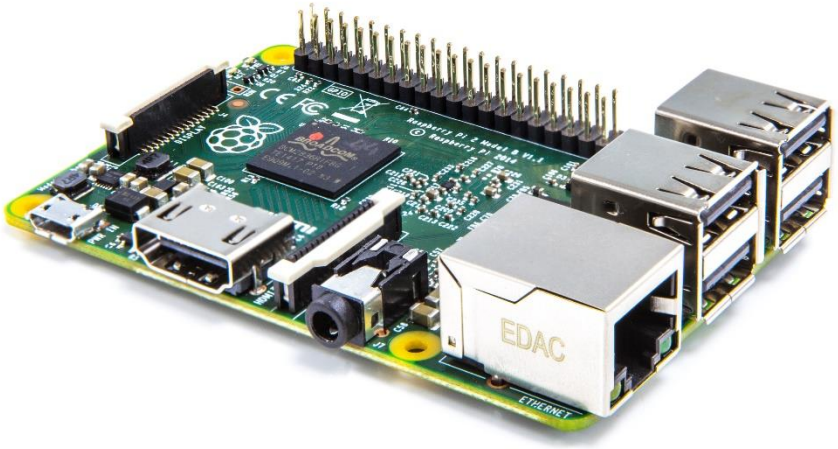
on koottu lista toimenpiteistä, joilla voidaan parantaa muistikortin elinikää sekä nopeuttaa käyttöjärjestelmän toimintaa. (Obarthelemy, 2015)

3.2 RaspBerryPI2

Koska uudempi RaspberryPi2-malli on julkaistu tulevat vanhemmat mallit poistuvat myynnistä ja koska Raspberry2 malli on hinnoiteltu samalla tavalla edellisien mallien kanssa, ei vanhempia malleja edes kannata hankkia vaikka sellaisen jostain saisi käsiinsä. Seuraavasta kuvasta voidaan nähdä uuden Raspberry Pi2:sen tekniset tiedot ja kuvasta 9 uuden piirilevyn asettelu.

	Model A
Myyntihinta	25 USD
Prosessori	900Mhz BCM2836 4-ytiminen ARM Cortex-A7
Näytönohjain	Broadcom VC IV, Opengl ES 2.0, 1080p h.264
Muisti	1024MB (jaettu)
USB 2.0	4 (2 Slave, 2 Host)
Videoulostulo	HDMI v1.4, RCA (jaettu 3,5mm)
Ääniulostulo	HDMI, 3,5mm (jaettu RCA)
Massamuisti	SD / MMC / SDIO-muistikortti
Verkkosovitin	10/100 RJ45 (Jaettu USB)
Muut liittimet	40-pinninen GPIO, CSI/DSI, UART
Tehonkulutus	700mA (noin 2,5W)
Virtalähde	5v MicroUSB (2A)
Koko:	85,60 x 53,98 x 17 mm
Paino	55g
Käyttöjärjestelmä	Linux, BSD, Solaris, Risc OS, Windows IoT

Kuva 8 - RaspberryPi2:n tekniset tiedot



Kuva 9 - Raspberry Pi2 (Raspberry.org, 2015)

3.2.1 Windows10 (for IoT)

Raspberrylle tuli myös saataville Microsoftin kehittämä Windows10, joka on suunnattu IoT-laitteille (Internet of Things). Microsoft pyrkii pääsemään mukaan teollisen internetin kehitykseen ja näin ollen tarjoaa Windows10-käyttöjärjestelmää ilmaiseksi Raspberrylle, sekä muille sen kaltaisille laitteille.

Käyttöjärjestelmän asentaminen muistikortille vaatii vähintään versionumerolla 10061 varustetun Windows10-isäntäkoneen.

Raspberry Pi:lle tarkoitettu Windows 10 on hyvin pelkistetty versio, joka ei pidä sisällään Windows-käyttöjärjestelmistä tuttua käyttöliittymää ainakaan kehityksen tässä vaiheessa, vaan se on tarkoitettu käytettäväksi PowerShell-työkalun tai HTTP-käyttöliittymän avulla. Uudessa Visual Studio 2015 on huomioitu IoT-laitteet, joille voidaan luoda ohjelmia etänä.

3.3 GPIO-Liitännät

GPIO (General Purpose I/O) on yleiskäyttöinen portti mikrokontrollereissa ja mikroprosessoreissa. General Purpose I/O voidaan asettaa joko signaalia lähettäväksi tai vastaanottajaksi. (Wikipedia, 2015)

3.3.1 GPIO-liitäntöjen käyttö

Seuraavasta kaaviosta voidaan nähdä, että osa Raspberryn GPIO-liitännöistä ovat monikäyttöisiä. (Broadcom, 2012)

- GPIO 3, 5 = I2C
- GPIO 8, 10 = UART
- GPIO 11 = UART-RTS
- GPIO 12 = PWM
- GPIO 19, 21, 23 = SPI
- GPIO 24, 26 = SPI

GPIO-pinnien numerointi vaihtelee myös sitä käyttävien rajapintojen kanssa. Seuraavassa kuvassa 8 numerointia on selvennetty tarkemmin. (Matt, 2012) (Henderson, 2015)

Kuva 10 - GPIO-pinnien numerointi eri rajapinnoilla

3.3.2 Komentorivi (Shell)

GPIO-pinnien asetuksia voidaan vaihtaa yksinkertaisimmillaan komentoriviltä ilman ylimääräistä ohjelmointikieltä tai kirjastoja. Komennot voidaan suorittaa joko sellaisenaan tai niitä varten voidaan luoda komentojonotiedosto, joka suorittaa sarjan toimintoja tarvittavalla tavalla. Seuraavassa kuvassa 9. käydään läpi, kuinka GPIO-porttien hallinnointi tapahtuu suoraan komentorivistä.

```
#!/bin/sh
# Asetetaan fyysinen GPIO pinni 7 input asentoon
echo "7" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio7/direction

#Asetetaan fyysinen GPIO pinni 7 output asentoon
echo "7" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio7/direction

# Määritellään fyysinen GPIO pinni 7 päälle.
echo "1" > /sys/class/gpio/gpio7/value

# Luetaan tämän hetkinen asento pinnistä 7
cat /sys/class/gpio/gpio7/value

# Poistetaan fyysisestä pinnistä 7 kaikki asetukset
echo "7" > /sys/class/gpio/unexport
```

Kuva 11 - Komentoriviesimerkki

3.3.3 Python

Python-ohjelmointikieltä käyttäessä täytyy käyttää RPi.GPIO-moduulia, joka tulee Raspbian-käyttöjärjestelmän mukana. Seuraavassa kuvassa 10. käydään läpi yksinkertainen esimerkki, kuinka Python-ohjelmointikielellä voidaan hallinnoida GPIO-pinnien tilaa.

```
import RPi.GPIO as GPIO

# Määritellään kumpaa numerointia käytetään.
GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)

# Määritellään input/output
GPIO.setup(7, GPIO.IN)
GPIO.setup(7, GPIO.OUT)

# Luetaan arvo pinnistä 7
input_value = GPIO.input(7)

# Määritellään pinniin 7 arvo 1/HIGH/True
GPIO.output(7, GPIO.HIGH)
```

Kuva 12 - Python-esimerkki

3.3.4 WiringPI

WiringPI on monipuolinen kirjasto GPIO-pinnien hallintaan, sitä voidaan käyttää suoraan komentoriviltä tai ohjelmointikielen yhteydessä kuten esimerkiksi C-ohjelman kanssa.

WiringPI täytyy ladata GIT-versiohallinnasta ja asentaa manuaalisesti. Ennen asennusta täytyy asentaa Git-asiakasohjelma komennolla:

sudo apt-get install git-core

Tämän jälkeen WiringPI tulee ladata komennolla:

git clone git://git.drogon.net/wiringPi

Tämän jälkeen WiringPI tulee kääntää binäärimuotoon ja asentaa komennoilla:

cd wiringPI
./build

Tämän jälkeen voidaan testata, onnistuiko asennus komennolla:

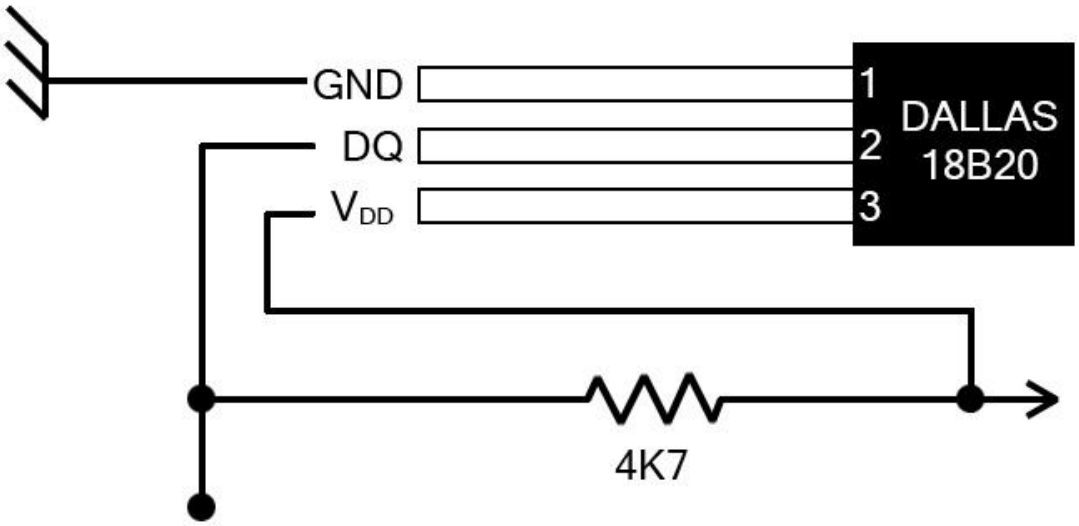
gpio -v
gpio readall

3.4 Digitaaliset anturit

Tässä luvussa kerrotaan käytettyjen antureiden ominaisuuksista ja toiminnasta.

3.4.1 Lämpötila-anturi Dallas DS18B20

Lämpötila-anturiksi valittiin Dallas DS18B20, joka soveltuu hyvin 1wire-tekniikan kanssa käytettäväksi anturiksi. Kuvasta 11. voidaan havaita, kuinka lämpötila-anturi tulee kytkeä. (Maxim integrated, 2015)



Kuva 13 - Dallas DS18B20:n kytkentäkaavio

1wire-tekniikan avulla anturin voi kytkeä saman johdon perään muiden samanmallisten antureiden kanssa. Tämä mahdollistaa helpon tavan ottaa mittaustietoja eri kohteista.

Raspberryssä on oletuksena säädetty antureiden maksimi määräksi 10 kpl, mutta kyseisen rajoituksen voi muuttaa joko komentoriviltä komennolla:

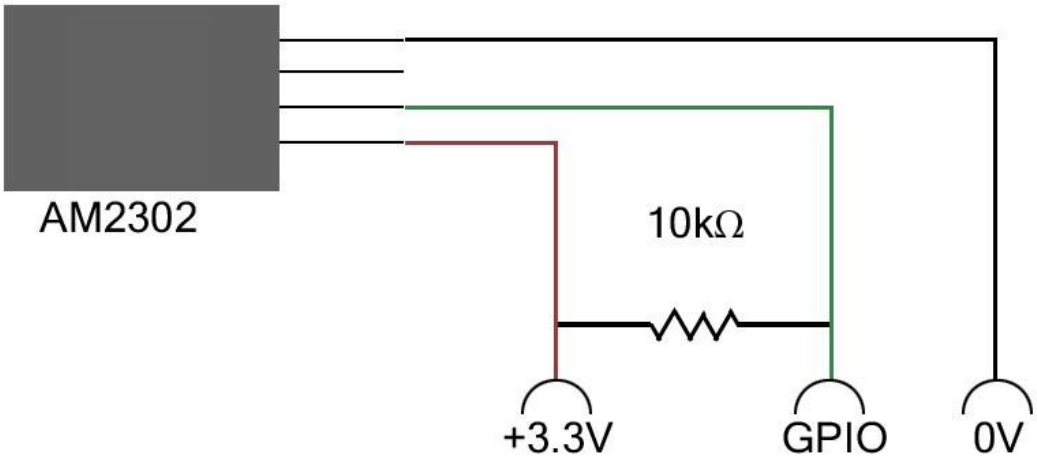
modprobe wire max_slave_count=15

Komento on voimassa vain hetkellisesti. Jos taas tarvitaan muuttaa tieto pysyvästi, se tapahtuu muokkaamalla /etc/modules tiedostoa ja lisäämällä sinne rivi

wire max_slave_count=15

3.4.2 Kosteusanturi AM2302

AM2302 on yksinkertainen kosteusanturi, joka mittaa kosteuden sekä lämpötilan. Anturin haittapuolena on se, että niitä ei voida kytkeä 1wire-antureiden tapaan samaan johtoon, sekä anturin hitaus. Anturilta voi saada uutta tietoa vain muutaman sekunnin välein. Kuvasta 12. voidaan nähdä AM2302-kosteusanturin kytkentäkaavio. AM2302- tai DHT22-anturissa saattaa olla neljä pinniä, mutta yhtä niistä ei tarvitse kytkeä mihinkään. (Adafruit, 2015)



Kuva 14 - AM2302-kosteusanturin kytkentäkaavio

3.5 BerryIO

BerryIO on Daniel Bullin RaspberryPi:lle kehittämä HTTP-pohjainen hallintapaneeli, jolla voidaan ohjata SPI-, GPIO-pinnien tilaa ja CSI-kameramoduulin toimintaa. Kuvassa 13. voidaan nähdä BerryIO:n HTTP-käyttöliittymä yleisesti.

Kuva 15 - BerryIO käyttöliittymä (BerryIO, 2012)

BerryIO on vapaa ohjelmisto, joka on julkaistu GNU GPL v3 lisenssillä. Lisenssi mahdollistaa sen edelleen jakamisen ja muuttamisen.

3.5.1 Ominaisuudet

Lista BerryIO:n tarjoamista hallinta ominaisuuksista:

- GPIO-porttien hallinnointi.
- SPI ADC/DAC hallinnointi.
- Mahdollisuus ottaa kuvia ja ohjata kameran asetuksia.
- LCD (mallien HDD44780 ja KS0066U) näyttöjen hallinnointi.
- Järjestelmän tilan seuranta (CPU, MEM, HDD, TEMP, NET)
- Sähköposti-ilmoitukset IP-osoitteen muutoksesta

- Automaattinen päivitys
- API jatkokehitystä varten.

3.5.2 Asennus

BerryIO:n asennus tapahtuu muiden Linux-ohjelmien tapaan, suorittamalla seuraavat komennot konsolissa. Ensin kannattaa varmistaa, että käyttöjärjestelmä on ajan tasalla komennolla:

sudo apt-get update
sudo apt-get upgrade

Seuraavilla komennoilla ladataan BerryIO:n asentamisen suorittavan komentosarjan, jonka jälkeen se suoritetaan:

wget -N https://raw.github.com/NeonHorizon/berryio/master/scripts/berryio_install.sh
chmod a+x berryio_install.sh
sudo ./berryio_install.sh

Mikäli halutaan ottaa käyttöön sähköposti-ilmoitukset, lähtevän sähköpostipalvelimen asetukset tulee määrittää tiedostoon

sudo nano /etc/msmtprc

Seuraavalla komennolla voidaan tarkistaa, onnistuiko asennus.

berryio help

3.5.3 Käyttö

Käyttö tapahtuu internetselaimen kautta osoitteessa http:<ip>:80. Käyttöliittymä on suojattu käyttäjätunnuksen ja salasanakyselyn taakse, jotta ulkopuoliset käyttäjät eivät pääsisi käsiksi laitteeseen. Käyttäjätunnuksena ja salasanana ovat samat, joita käytetään terminaaliin kirjautuessa.

3.6 Kamera

Raspberryssä voidaan käyttää kahdenlaista kameraa. Toinen on perinteinen USB-mallinen kamera ja toinen on Raspberry-säätiön itse kehittämä CSI-kamera.

3.6.1 CSI-kameramoduuli

Säätiö on kehittänyt CSI-kameramoduulin Raspberryä varten. Kameramoduuli pitää sisällään 5 Megapixelin CMOS-kameran. Moduulilla pystyy ottamaan sekä valokuvia resoluutiolla 2592x1944, että ottamaan videokuvaa 1080p@30 tarkkuudella. Kuvassa 14. voidaan nähdä CSI-kameramoduuli, joka tulee ilman muovista suojakotelointia.

Kuva 16 - CSI-kameramoduuli (itc.ua, 2015)

Valokuvan ottaminen tapahtuu yksinkertaisimmillaan komennolla:

raspistill -vf -hf -o /home/pi/camera/kuva.jpg

Vastaavasti videokuvan ottaminen tapahtuu komennolla:

raspivid -o video.h264 -w 1280 -h 720

Tarjolla on paljon enemmän komentoja, jotka saadaan näkyviin suorittamalla ohjelma ilman ylimääräisiä parametreja:

raspivid

3.6.2 USB-kamera

Raspberryssä oleviin USB-portteihin voidaan myös kytkeä perinteinen USB-kamera ja sitä voidaan käyttää lähes samalla tavalla projekteissa. Ainoana poikkeuksena videokuvan pakkausta ei voida tehdä tehokkaasti.

CSI-kamerasta poiketen joudutaan käyttämään erillistä fswebcam-ohjelmaa joka täytyy asentaa erikseen komennolla:

sudo apt-get install fswebcam

Tämän jälkeen kuvia voidaan ottaa helposti komennolla:

fswebcam /home/pi/camera/kuva.jpg

4 Ratkaisut ja toteutus

4.1 Lämpötila-anturi

Käytettäessä Dallas DS18B20-lämpötila-anturia ja 1wire-tekniikkaa tulee ottaa käyttöön asianmukaiset kernel-moduulit lisäämällä /etc/modules-tiedostoon seuraavat rivit:

wire
w1-gpio
w1-therm

Tämän jälkeen kernel hoitaa digitaalisten tietojen keräämisen antureilta ja muuntaa ne luettavaan muotoon, jotka voidaan lukea tiedostosta /sys/bus/w1/devices/*HEX_ID*/w1_slave. Jokaisella anturilla on oma yksilöllinen heksadesimaali tunnus. (WebShedOrg, 2015)

Käytössä oli useampia lämpötila-antureita, joista piti saada kerättyä lämpötilatietoja pitkältä aikaväliltä. Tehtävä toteutettiin ottamalla käyttöön MariaDB/MySQL tietokanta, sekä luomalla python-ohjelma, joka iteroi jokaisen anturin läpi ja syöttää antureista saadut tiedot tietokantaan.

Ensimmäiseksi ohjelman tiedostosta ”1wire.py” täytyy käydä muokkaamassa tietokannan tiedot vastaamaan käytössä olevaa tietokantaa. Seuraavaksi ohjelma tulee ajaa seuraavalla komennolla.

python 1wire.py –config

Tämä komento kerää lämpötila-antureiden heksadesimaalitunnukset ja lisää ne ”config”-tiedostoon. Tämän jälkeen config-tiedostosta täytyy muokata tietokannan kenttä osio ”Column” ja nimetä se sensorin nimellä esimerkiksi ”Column: kasvihuone” ja tiedoston lopussa heksadesimaalitunnuksen perään tulee nimetä, mihin kenttään kyseinen anturi kuuluu.

Seuraavaksi suoritetaan komento joka luo tietokannan ja siihen kuuluvat taulut.

python 1wire.py –ccreate

Tämän jälkeen ohjelma suoritetaan ilman parametreja.

4.2 Kosteusanturi

Testattavana olleen kosteusanturin AM2302 käyttöönotossa tarvittiin Adafruit-kirjastoa, joka voidaan ladata käytettäväksi git-versiohallintaohjelmalla seuraavalla komennolla:

git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git

Tämän toimenpiteen jälkeen käytössä on useita Adafruit-kirjastoja, mutta tarvitsemme käyttöön vain ../Adafruit-Raspberry-Pi-Python-Code/Adafruit_DHT_Driver/Adafruit_DHT, joka kopioidaan kansioon /usr/bin, jonka jälkeen se on globaalisti käytettävissä.

Tämän jälkeen AM2302-anturia voidaan testata kirjoittamalla komentoriville seuraava komento

sudo Adafruit_DHT 2302 [gpio pinnin numero]

Kosteusanturin tietojen talletuksessa käytettiin myös Python-ohjelmointikieltä, jolla luetaan saatu tieto anturilta ja se lisätään tietokantaan.

Ensimmäiseksi tulee muokata ”sensors_DHT.py”-tiedostosta tietokantaan liittyvät rivit vastaamaan käytössä olevaa tietokantaa, sekä myös anturin malli, joka voi olla DHT2302-, DHT22-, DHT11- ja GPIO-pinnin numero.

Tämän jälkeen suoritetaan seuraava komento, joka luo tietokantaan tarvittavat kentät.

python sensors_DHT.py –dbcreate

Tämän jälkeen tietokantaan lisätään tiedot kosteusanturilta komennolla:

python sensors_DHT.py –dbwrite

Tietokannasta voidaan lukea tietoja komennolla:

python sensors_DHT.py –dbread

4.3 Kamera

Vaatimuksena oli saada selvitettyä kuinka RaspberryPi:llä voidaan ottaa kuvia, jotta etäohjattavasta kohteesta voitaisiin myös päästä näkemään, mitä siellä tapahtuu. Käytössä oli vain tavallinen USB-kamera, johon luotiin skripta, joka ottaa kuvan ja nimeää sen vuosi-kuukausi-päivämäärä-tunti-minuutti.jpg

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
fswebcam –r 1280x720 $HOME/kuva/$DATE.jpg
```

4.4 IP-osoite

Koska yhtenä vaatimuksena oli, että kyseinen ohjaus- ja mittausjärjestelmä voisi toimia itsenäisesti syrjäisessä paikassa aurinkopaneelin voimin, täytyi selvittää, kuinka laitteeseen voidaan saada yhteys IP-osoitteen muutoksen jälkeen. IP-osoite saattaa vaihtua monestakin eri syystä kuten esimerkiksi sähkökatkos, uudelleen käynnistäminen, päivitykset tai ISP:n huoltotyöt. Täytyi selvittää keino, jolla IP-osoite lähetetään automaattisesti käyttäjälle muutoksen tapahtuessa.

Ratkaisu oli osittain jo valmiina saatavilla GitHub-alustalta. Javier Rengel Jiménez alias ”Rephus” on kehittänyt python-ohjelman Jenkins-integraatioalustalle. Ohjelma on saatavilla kyseisestä osoitteesta https://github.com/coconauts/IP-change

Kyseiseen ohjelmaan lisäämällä uuden funktion jonka avulla Raspberry Pi pystyy lähettämään käynnistämisen yhteydessä muuttuneen IP-osoitteen haluttuun sähköpostiin.

Kyseiseen skriptani ”ip_change.py” tulee muokata oikeat sähköpostitiedot sekä smtp-palvelimen tiedot.

4.5 Releohjaus

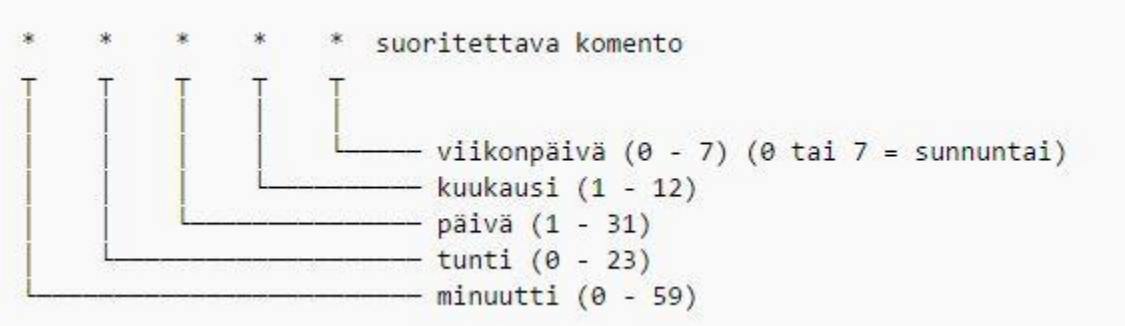
Releohjaus päätettiin toteuttaa suoraan käyttämällä BerryIO-käyttöliittymää, ja sähköiset kytkennät jätettiin asiakkaan hoidettavaksi turvallisuussyistä.

4.6 Ohjelmien ajastaminen

Ajastusohjelmana Unix sekä Linux käyttöjärjestelmässä käytetään Cron-ohjelmaa. Ajastimia muokataan crontab-ohjelmalla komennolla:

crontab –e

crond-daemon tarkistaa minuutin välein kun aika- ja päivämääritykset täsmäävät nykyhetkeen ja suorittavat komennon, skriptan tai muun ohjelman. Seuraavassa kuvassa 17, voidaan nähdä crontab-käyttöliittymä:



Kuva 17 - Crontab ajan esitysmuoto (Wikipedia, 2015)

1wire.py python-ohjelma ajastetaan suoritettavaksi kerran minuutissa lisäämällä seuraava rivi:

**** * * * * /usr/bin/python $HOME/1wire.py***

dht2302.py python-ohjelma ajastetaan suoritettavaksi kerran minuutissa lisäämällä

**** * * * * /usr/bin/python $HOME/sensors_DHT.py***

ip_change.py python-ohjelma voidaan määrittää suoritettavaksi vain sen jälkeen kun RaspberryPi:n käyttöjärjestelmä on käynnistetty uudelleen. IP-osoitteet useimmiten vaihtuvat vasta silloin kun verkkoon kytkeydytään uudestaan. Kyseisen ohjelman voi myös määrittää ajettavaksi tunnin välein. IP-osoite lähetetään sähköpostitse vain siinä tapauksessa jos se on vaihtunut.

@reboot /usr/bin/python $HOME/ip_change.py

5 Yhteenveto ja pohdinta

Opinnäytetyössä tehtävänä oli suunnitella pieni etäohjattava mittaus- ja ohjausjärjestelmä käyttäen hyväksi Raspberry Pi-laitetta.

Opinnäytetyölle ei ollut vaadittu tiukkaa aikataulua, mutta tavoitteena oli saada se valmiiksi keväällä 2015. Opinnäytetyön etenemistä hiukan haittasi ohjaajan vaihtuminen kesken työn, jolloin dokumentaatio alkoi hakemaan uutta suuntaa ja näin ollen siitä tuli hiukan irtonainen. Loppujen lopuksi aikataulussa kuitenkin on pysytty olosuhteisiin nähden hyvin. Kaikki projektin tavoitteet on saavutettu, joten sitä voidaan sanoa onnistuneeksi. Raportin kirjoittamiseen tulisi panostaa enemmän ja siihen tulisi myös varata paremmin aikaa.

Tietenkään projektin alussa ei ollut kaikkea vaadittavia ohjelmistoa ja työkalua tarkalleen tiedossa ja niitä alettiin tekemään omien kokeilujen pohjalta itse. Lopussa tultiin siihen tulokseen, että saatavilla olisi ollut valmiita ratkaisuja ohjelmista sekä suunnitelmista. Vaikka kyseiset ratkaisut eivät olleet suoraan suunniteltu Raspberry Pi-alustalle, ne ovat hyvin yleismaallisia ratkaisuja, jotka toimivat alustariippumattomasti tiedetyin rajauksin.

Opinnäytetyön aikana tuli huomattua, miten erilainen lähtötaso allekirjoittaneella ja muilla voi olla. Hyvin monessa tilanteessa pidin Linux-käyttöjärjestelmään liittyviä tehtäviä ja toimintoja itsestäänselvyyksinä ja niiden dokumentointi jäi vähemmälle, vaikka juuri näihin pitäisi panostaa ja mitään ei saisi ottaa itsestäänselvyytenä.

Lopputuloksena tulin siihen päätelmään, että mikäli tämän kaltaista mittausjärjestelmää tulisi tehtyä uudestaan, valitsisin Raspberry Pi:n sijaan toisen laitteen, joka on suunniteltu tarkoitusta varten, eikä jokapaikanhöylä periaatteella. Yksi varteenotettava vaihtoehto olisi MicroPython-alusta tai WiPy.

Kuvaluettelo

Lähdeluettelo

Adafruit, 2015. *Adafruit.* [Online]
Available at: https://www.adafruit.com/products/393
[Accessed 11 Huhtikuu 2015].

BerryIO, 2012. *Github.* [Online]
Available at: https://github.com/NeonHorizon/berryio
[Accessed 28 Toukokuu 2015].

Broadcom, C., 2012. In: *BCM2835 ARM Peripherals.* s.l.:s.n., pp. 102 - 104.

cs.cmu.edu, 2015. *cs.cmu.edu.* [Online]
Available at: https://www.cs.cmu.edu/~awb/linux.history.html
[Accessed 24 Toukokuu 2015].

Henderson, G., 2015. *Wiring Pi.* [Online]
Available at: http://wiringpi.com/pins/
[Accessed 15 Maaliskuu 2015].

itc.ua, 2015. *itc.ua.* [Online]
Available at: http://itc.ua/news/anonsirovan-modul-kameryi-dlya-raspberry-pi-po-tsene-25/
[Accessed 28 Toukokuu 2015].

Kernel.org, 2015. *kernel.org.* [Online]
Available at: https://www.kernel.org/doc/pending/gplv2-howto.html
[Accessed 25 Toukokuu 2015].

labor-liber.org, 2004. *labor-liber.org.* [Online]
Available at: http://labor-liber.org/en/gnu-linux/development/programming_languages
[Accessed 24 Toukokuu 2015].

Matt, 2012. *Raspberry Pi Spy.* [Online]
Available at: http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-

gpio-header-and-pins/
[Accessed 25 Maaliskuu 2015].

Maxim integrated, 2015. *DS18B20 Programmable Resolution 1-wire Digital Thermometer.* [Online]
Available at: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
[Accessed 15 Huhtikuuta 2015].

Obarthelemy, 2015. *Raspberrypi.org.* [Online]
Available at: https://www.raspberrypi.org/forums/viewtopic.php?f=9&t=850
[Accessed 28 Helmikuu 2015].

Raspberry.org, 2015. *Raspberry.org.* [Online]
Available at: https://www.raspberrypi.org/raspberry-pi-2-on-sale/
[Accessed 4 Kesäkuu 2015].

raspberrypi.org, 2015. *raspberrypi.org.* [Online]
Available at: https://www.raspberrypi.org/about/
[Accessed 25 Toukokuu 2015].

raspbian.org, 2015. *www.raspbian.org/.* [Online]
Available at: https://www.raspbian.org/RaspbianFAQ
[Accessed 23 Toukokuu 2015].

RasPi.TV, 2015. *RasPi.TV.* [Online]
Available at: http://raspi.tv/2015/raspberry-pi2-power-and-performance-measurement
[Accessed 22 Helmikuu 2015].

sofokus.com, 2015. *www.sofokus.com.* [Online]
Available at: https://www.sofokus.com/blogi/avoin-lahdekoodi/
[Accessed 23 Toukokuu 2015].

Torvalds, L., 2010. [Interview] (18 kesäkuu 2010).

WebShedOrg, 2015. *Web Shed Org.* [Online]
Available at: http://webshed.org/wiki/RaspberryPI_DS1820
[Accessed 17 Tammikuu 2015].

Wikipedia, 2015. *Wikipedia.* [Online]
Available at: http://en.wikipedia.org/wiki/General-purpose_input/output
[Accessed 23 Toukokuu 2015].

Wikipedia, 2015. *Wikipedia.* [Online]
Available at: http://fi.wikipedia.org/wiki/Cron
[Accessed 28 Toukokuu 2015].

wired.co.uk, 2015. *wired.co.uk.* [Online]
Available at: http://www.wired.co.uk/news/archive/2015-02/18/raspberry-pi-5-million
[Accessed 25 Toukokuu 2015].

Liitteet

Liitteet

- Liite 1 1wire.py
- Liite 2 dht2302.py
- Liite 3 ip_change.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
#   1wire Dallas DS18B20
#

from warnings import filterwarnings
import MySQLdb as Database
import MySQLdb
import sys
import datetime
import time
import argparse
import base64


# ----------------------------------------------------------------------

config_file = 'config'

db_fields = []              # tietokanta kenttä
db_dict = {}                    # sanakirja
sensor_list = []                # sensorilista
sensor_locations = []           # sensoreiden sijainti
sensor_dict = {}                # sanakirja/pointer -> sensori :
sijainti
sensordata = []
data_temp = []

databaseHost = '192.168.1.1'
databaseUsername = 'user'
databasePassword = 'password'
databaseName = 'rpi'
databaseTable = '1wire'
database_fields_all = ''
db = ''

database_all = [databaseTable, databaseName, databaseHost,
databaseUsername, databasePassword]
read_sens = 1


# ----------------------------------------------------------------------

parser = argparse.ArgumentParser()
parser.add_argument(
    '-cconfig',
    action='store_const',
    dest='cconfig',
    const='value-to-store',
    )
parser.add_argument(
    '-ccreate',
    action='store_const',
    dest='ccreate',
    const='value-to-store',
    )
args = parser.parse_args()
```

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

def read_config(init_level):

    global databaseHost
    global databaseName
    global databaseUsername
    global databasePassword
    global databaseTable
    global database_all
    global database_fields_all

    try:
        config = open(config_file, 'r')
        settings = []
        for line in config:
            if line[0] != '#':
                settings.append(line)
        config.close

        for x in range(0, len(settings)):
            line = settings[x]
            item1 = line.split()
            if len(item1) > 1:
                if item1[0] == 'Column:':
                    db_fields.append(item1[1])
                if len(item1) > 2:
                    if item1[0] == 'ID':
                        sensor_list.append(item1[1])
                        sensor_locations.append(item1[2])
                        sensor_dict[item1[1]] = item1[2]

        for x in range(0, len(db_fields)):
            if x == 0:
                database_fields_all = str(db_fields[x])
            else:
                database_fields_all = database_fields_all + ', ' +
str(db_fields[x])
            db_dict[db_fields[x]] = x

        database_all = [databaseTable, databaseName, databaseHost,
databaseUsername, databasePassword]

    except IOError:
        print 'Error: no ' + config_file + ' found'
    return ()


# --------------------------------------------------------------------
#
def add_sensors():

    config = open(config_file, 'a')

    sensors = 0
    file = open('/sys/devices/w1_bus_master1/w1_master_slaves')

    w1_slaves = file.readlines()
    file.close()
```

```
  # Toistetaan kaikille 1wire sensoreille

    for line in w1_slaves:
        w1_slave = line.split('\n')[0]
        config.write('#ID ' + w1_slave + '\n')

    file.close()

    sys.exit(0)
    return ()


# --------------------------------------------------------------------

def read_sensor(sensor_slave):

    sensor_device = '/sys/bus/w1/devices/' + str(sensor_slave) \
        + '/w1_slave'
    try:
        file = open(sensor_device)
        filecontent = file.read()
        file.close()
        stringvalue = filecontent.split('\n')[1].split(' ')[9]
        if stringvalue[0].find('YES') > 0:
            temperature = error_temp
        else:
            temperature = float(stringvalue[2:]) / 1000
    except IOError:

        print 'ERROR: no sensors found'

        sys.exit(1)

    return temperature


# --------------------------------------------------------------------

def read_sensors(read_level):
    sensors = 0
    sensor_slaves = '/sys/devices/w1_bus_master1/w1_master_slaves'

    try:
        file = open(sensor_slaves)

        w1_slaves = file.readlines()
        file.close()

  # Toistetaan jokaiselle löytyneelle 1wire sensorille.

        for line in w1_slaves:
            w1_slave = line.split('\n')[0]
            time.sleep(0.2)
            temperature = read_sensor(w1_slave)

            sensors = sensors + 1

            if read_level:
                sensordata.append((w1_slave, temperature))
        return sensors
```

```
    except IOError:

        print 'read_sensors - Cannot find file: ', sensor_slaves
    return 0


# ----------------------------------------------------------------------

def create_database(database_all, database_fields_all):

      filterwarnings('ignore', category=Database.Warning)
      databaseName = database_all[1]
      databaseTable = database_all[0]
      databaseHost = database_all[2]
      databaseUsername = database_all[3]
      databasePassword = database_all[4]

      db_connect(database_all)
      string = 'CREATE DATABASE IF NOT EXISTS ' + databaseName
      cur = db.cursor()
      try:
            cur.execute(string)
            db.commit()
      except:
            print 'ERROR x001'

      db_connect(database_all)
      string = 'DROP TABLE IF EXISTS ' + databaseTable
      cur = db.cursor()
      try:
            cur.execute(string)
            db.commit()
      except:
            print 'ERROR x002'

      string = 'CREATE TABLE ' + databaseTable + ' ('
      string = string + 'id INT NOT NULL AUTO_INCREMENT PRIMARY KEY'
      string = string + ', dattim TIMESTAMP'
      dbf = database_fields_all.split()
      for x in range(0, len(dbf)):
            x = dbf[x]
            x = x.split(',')
            string = string + ', ' + x[0] + " FLOAT DEFAULT '0'"
      string = string + ' )'
      cur = db.cursor()
      try:
            cur.execute(string)
            db.commit()
      except:
            print 'ERROR x003'
      return ()


# ----------------------------------------------------------------------

def write_database(read_items,sensordata,sensor_dict,database_all):
      data_temp = []
      comb1 = []
      comb2 = []
      Val_String = ''
```

```
      databaseTable = database_all[0]
      databaseName = database_all[1]
      databaseHost = database_all[2]
      databaseUsername = database_all[3]
      databasePassword = database_all[4]
      db_connect(database_all)

      for x in range(0, read_items):
            dataset = sensordata[x]
            sen_loc = str(sensor_dict.get(dataset[0]))
            if sen_loc == 'None':
                  print 'ERROR: Sensor invalid ', dataset[0]
            elif x == 0:
                  Sen_String = sen_loc
                  Val_String = '%s'
                  data_temp.append(dataset[1])
                  comb1.append(dataset[1])
            else:
                  Sen_String = Sen_String + ', ' \
                        + str(sensor_dict.get(dataset[0]))
                  Val_String = Val_String + ', %s'
                  data_temp[0] = (data_temp[0], dataset[1])
                  comb1.append(dataset[1])

      comb2.append(comb1)
      string = 'INSERT INTO ' + databaseTable + ' (' + Sen_String \
            + ') VALUES (' + Val_String + ')'
      cur = db.cursor()
      cur.executemany(string, comb2)
      db.commit()
      return 0


# --------------------------------------------------------------------

def db_connect(database_all):

      global db
      db_table = database_all[0]
      databaseName = database_all[1]
      databaseHost = database_all[2]
      databaseUsername = database_all[3]
      databasePassword = database_all[4]

      try:
            print 'Saved'
            db = MySQLdb.connect(host=databaseHost, user=databaseUsername,
                                       passwd=databasePassword,
db=databaseName)

      except:
            print 'ERROR: cannot connect to database'
            sys.exit(2)

      return 0

# --------------------------------------------------------------------

if args.cconfig:
```

```
    add_sensors()

return_val = read_config(0)
if return_val:
    print 'Error: cannot find config file ' + config_file
    sys.exit(0)

if args.ccreate:
    create_database(database_all, database_fields_all)
    read_sens = 0

if read_sens:
    read_items = read_sensors(1)
    write_database(read_items, sensordata, sensor_dict, database_all)

sys.exit(0)


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

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
#   AM2302 kosteusanturi
#


import subprocess
import re
import os
import sys
import time
import MySQLdb as database
import argparse
import base64
from warnings import filterwarnings

databaseHost = '192.168.1.1'
databasePort = '3306'
databaseUsername = 'user'
databasePassword = 'password'
databaseName = 'rpi'
databaseTable = 'DHT2302'

sensorModel = '2302'
sensorPin = '22'

status = 'false'
#createDatabase = 0
#readSensors = 1
verbose = 1
newDatabase = 0

# --------------------------------------------------------------------------

parser = argparse.ArgumentParser()
parser.add_argument(
    '-dbwrite',
    action='store_const',
    dest='dbwrite',
    const='value-to-store',
    help='Writes data from the sensors to database.'
    )
parser.add_argument(
    '-dbcreate',
    action='store_const',
    dest='dbcreate',
    const='value-to-store',
    help='Creates new database.'
    )
parser.add_argument(
    '-dbread',
    action='store_const',
    dest='dbread',
    const='value-to-store',
    help='Read values from database.'
    )
args = parser.parse_args()

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

```
def create_database():

    try:
        dbConnection = database.connect(databaseHost, databaseUsername,
databasePassword)
        with dbConnection:
            cursor = dbConnection.cursor()
            if newDatabase == 1:
                try:
                    cursor.execute("DROP DATABASE IF EXISTS %s" %
databaseName)
                    cursor.execute("CREATE DATABASE %s" % databaseName)
                except database.Error, e:
                    print "Error %d: %s" % (e.args[0],e.args[1])

            cursor.execute("USE %s" % databaseName)
            sql = """CREATE TABLE IF NOT EXISTS %s (
                    id int(255) NOT NULL AUTO_INCREMENT,
                    datetime timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
                    temperature double NOT NULL,
                    humidity varchar(20) NOT NULL,
                    PRIMARY KEY (id)
                    ) ENGINE=InnoDB  DEFAULT CHARSET=utf8
AUTO_INCREMENT=1;""" % databaseTable

            cursor.execute(sql)
            if verbose == 1:
                print "Database created"
    except database.Error, e:
        print "Error %d: %s" % (e.args[0],e.args[1])

# ------------------------------------------------------------------------

def writeDatabase(temperature, humidity):

    try:
        dbConnection = database.connect(databaseHost, databaseUsername,
                databasePassword, databaseName)
        with dbConnection:
            cursor = dbConnection.cursor()
            cursor.execute("""INSERT INTO `{0}` (temperature, humidity)
VALUES (%s,%s)""".format(databaseTable), (temperature, humidity))

            if verbose == 1:
                print 'Saved'
                return 'true'
            else:
                return 'true'
    except database.Error, e:
        print "Error %d: %s" % (e.args[0],e.args[1])

# ------------------------------------------------------------------------

def read_sensor():

    # Luetaan kunnes saadaan validia tietoa joka ei ole Null.
    notNull = 'false'
    while notNull == 'false':
```

```
      # Ajetan Adafruit_DHT ohjelma aliprosessina joka lukee sensorin
tiedot.

        _adafruit_pipe = subprocess.check_output(['Adafruit_DHT',
                sensorModel, sensorPin])
        if verbose == 1:
            print _adafruit_pipe

            # Haetaan lämpötilaa Adafruit_DHT tulosteesta

        string = re.search("Temp =\s+([0-9.]+)", _adafruit_pipe)
        if not string:
            time.sleep(5)
            continue
        temperature_value = float(string.group(1))

          # Haetaan kosteustilaa Adafruit_DHT tulosteesta

        string = re.search("Hum =\s+([0-9.]+)", _adafruit_pipe)
        if not string:
            time.sleep(5)
            continue
        humidity_value = float(string.group(1))

        return writeDatabase(temperature_value, humidity_value)

# --------------------------------------------------------------------

def read_database():

    try:
        dbConnection = database.connect(databaseHost, databaseUsername,
databasePassword, databaseName)
        with dbConnection:
            cursor = dbConnection.cursor(database.cursors.DictCursor)
            cursor.execute("SELECT * FROM %s" % databaseTable)

            rows = cursor.fetchall()
            for row in rows:
                print row["id"], " |\t", row["datetime"], "|\t",
row["temperature"], "|\t", row["humidity"], "\t|"

    except database.Error, e:
        print "Error %d: %s" % (e.args[0],e.args[1])


if args.dbwrite:
    while status != 'true':
        status = read_sensor()

if args.dbcreate:
    create_database()

if args.dbread:
    read_database()

sys.exit(0)

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

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
#   1wire Dallas DS18B20 lämpötila-anturi
#
#   Original source: https://github.com/coconauts/IP-
change/blob/master/ip_change.py
#

import requests
import time
import sys
import os.path
import smtplib
import base64
from email.mime.text import MIMEText


#Global variables
ipFile="/root/rpi/ip_change/ip.log"
timeout = 10

class Service:
  url=""
  def request(self): return requests.get(self.url, timeout = timeout)

class wtfismyip(Service):
  name="wtfismyip"
  url="http://wtfismyip.com/text"
  def ip(self): return self.request().text.strip()

class Freegeoip(Service):
  name="freegeoip"
  url="https://freegeoip.net/json/"
  def ip(self): return self.request().json()["ip"]

class Telize(Service):
  name="telize"
  url="http://www.telize.com/ip"
  def ip(self): return self.request().text.strip()

class IpApi(Service):
  name="ip-api"
  url="http://ip-api.com/json"
  def ip(self): return self.request().json()["query"]

class Ifconfig(Service):
  name="ifconfig.me"
  url="http://ifconfig.me/all.json"
  def ip(self): return self.request().json()["ip_addr"]

# --------------------------------------------------------------------

def send_mail(current_ip, request_ip):

  username = 'username'
  password = 'password'

  recipients = ["email@address.to"]
  sender = "email@address.from"
```

```
  subject = "[RaspberryPI] New IP"
  body = """
  Hey, my IP address has changed:
  Old IP: {}  New IP: {}""".format(current_ip, request_ip)

  msg = MIMEText(body)
  msg['Subject'] = subject
  msg['From'] = sender
  msg['To'] = ", ".join(recipients)

  server = smtplib.SMTP('smtp.gmail.com:587')
  server.starttls()
  server.login(username,password)
  for toad in recipients:
    server.sendmail(sender, toad, msg.as_string())
  server.quit()

# --------------------------------------------------------------------

def request_ip():
  #List of services
  services = [ Ifconfig(), Freegeoip(), Telize(), IpApi(), wtfismyip()]
  for i in range(len(services)):

    service = services[i]
    try:
      start = time.time()
      print "* Requesting current ip with '{}'".format(service.name)
      ip = service.ip()
      print "* Request took {} seconds ".format(int(time.time() - start))
      return ip
    except Exception as error:
      print "* Exception when requesting ip using '{}': {}
".format(service.name, error )

  error = "Non available services, add more services or increase the
timeout (services = {},
  timeout = {}) ".format(len(services), timeout)
  raise RuntimeError(error)

def current_ip():
  return open(ipFile,"r").readlines()[0]

def save_ip(ip):
  f = open(ipFile,'w')
  f.write(str(ip))

#Main
if os.path.isfile(ipFile) : #File exists
  request_ip = request_ip()
  current_ip = current_ip()

  if request_ip != current_ip:
      save_ip(request_ip)
      print "IP has changed from {} to {}".format(current_ip, request_ip)
      send_mail(current_ip, request_ip)
      sys.exit(1)
  else :
    print "* IP is still the same: {}".format(current_ip)
```

```
else:
  request_ip = request_ip()
  save_ip(request_ip)
  print "* This is the first time to run the ip_change script, I will
create a file in {} to store your current address: {} ".format(ipFile,
request_ip)
```