Tero Jarva, Tapani Tyhtilä

ANTURIDATAN KERUU

Datan keruu Raspberry Pi:llä ja siirto SFTP palvelimelle

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Sähkö- ja automaatiotekniikan koulutusohjelma
Helmikuu 2020

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Tammikuu 2020	**Tekijä/tekijät** Tero Jarva, Tapani Tyhtilä
Koulutusohjelma Sähkö- ja automaatiotekniikka		
Työn nimi ANTURIDATAN KERUU		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 21 + 20
Työelämäohjaaja Antti Ryhänen		

Opinnäytetyön tavoitteena oli luoda järjestelmä, joka kerää antureilta saatavia arvoja ja tallentaa ne tekstimuodossa ulkoiselle palvelimelle. Järjestelmä koostui Raspberry Pi -laitteesta ja siihen asennettavasta A/D-muuntimesta. Ohjelmoinnissa käytettiin Python-ohjelmointikieltä.

Antureilta saapuva analoginen tieto muutettiin Raspberryn ymmärtämäksi digitaalitiedoksi ABElectronicsin valmistamalla ADC Pi laajennuskortilla. Muunnettu tieto siirrettiin i2c-väylää pitkin Raspberrylle, jossa ohjelma jakoi tulokset haluttuihin paikkoihin lopullisessa tiedostossa.

Antureiden tarkkailu alkaa välittömästi Raspberryn käynnistyttyä, eikä vaadi erillisiä komentoja toimiakseen. Tallennuksen asetuksia on mahdollista muuttaa joko Raspberrystä itsestään tai etänä verkon yli.

Lopputuloksena oli dataa jatkuvasti tallentava järjestelmä, joka lähettää arvoista muodostetun koosteen SFTP-palvelimelle vapaasti määriteltävin väliajoin. Säätömahdollisuudet saatiin toteutettua myös laitteeseen kytkettävien anturien määrälle ja näytteenoton tarkkuudelle. Ohjelma nimeää kaikki luomansa tiedostot yksilöllisesti sarjanumeron ja aikaleiman mukaan. Useat laitteet voivat siis tallentaa tuloksensa samalle palvelimelle aiheuttamatta nimistä johtuvia ongelmia.

Asiasanat
A/D-muunnos, SFTP, SSH, Python, Raspberry Pi

Centria
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Centria University of Applied Sciences	**Date** January 2020	**Author** Tero Jarva, Tapani Tyhtilä
Degree programme Electric engineering		
Name of thesis SENSOR DATA ACQUISITION		
Instructor Hannu Puomio		**Pages** 21 + 20
Supervisor Antti Ryhänen		

Goal of the thesis was to create a system which collects data from sensors and saves them in text format into an external server. The system consisted of a RaspberryPi single-board computer and an attached A/D transformer. Programming was done using Python language.

The analog signal sent by the sensors was transformed into digital format understood by the Raspberry using ADC Pi expansion board manufactured by ABElectronics. The transformed data was sent through I2C bus to Raspberry, where a program arranged the data in a desired fashion.

Data acquisition begins immediately after Raspberry starts up and requires no external commands to run. Settings for the program can be altered either directly from Raspberry or over the internet.

The result was a system capable of continuous data collection which sends compilation of the results to an SFTP server on adjustable intervals. It is also possible to adjust the number of sensors being used and the bitrate of their AD conversion. The program designates names for the files based on unique serial numbers and time stamps. Thus it is possible to save results of multiple systems on a single server without any filename related issues.

Key words
A/D-conversion, SFTP, SSH, Python, Raspberry Pi

KÄSITTEIDEN MÄÄRITTELY

A/D	Analogia-digitaalimuunnos.
APT	Linux-pakettien hallintaa helpottava työkalu.
ARDUINO	Ohjelmoitava elektroniikka-alusta.
BITRATE	Bitratella eli bittisyydellä tarkoitetaan opinnäytetyössä näytteenoton tarkkuutta.
COMPILER	Kääntää ihmisen kirjoittaman koodin tietokoneelle ymmärrettävään muotoon.
CRONTAB	Linux käyttöjärjestelmän osa, jolla voidaan ajoittaa tehtäviä.
GPIO	Mikrokontrollerissa oleva pinni, joka voidaan ohjelmoida joko tuloksi tai lähdöksi.
HDMI	Digitaalisten näyttölaitteiden liitäntästandardi.
IoT	Esineiden Internet.
IP	Internet protokolla.
PYTHON	Ohjelmointikieli, jonka koodi ajetaan kääntämättä sitä ensin.
PuTTy	Pääte-emulaattori ja asiakasohjelma etähallintaan.
SBC	Yhdelle piirilevylle rakennettu tietokone.
SFTP	SSH-suojauksen yli toimiva tiedonsiirtoprotokolla.
SMTP	Sähköpostiviestien välittämiseen tarkoitettu protokolla.
SSH	Salatun tietoliikenteen protokolla.
SSH2	SSH:n kehittyneempi ja suojatumpi versio.
SKRIPT	Skriptillä tarkoitetaan ajon aikana tulkattavaa ohjelmaa.
SSH	Secure Shell – protokolla, jonka avulla salataan tietoliikenneyhteyksiä
TERMINAL	Ohjelma, joka ottaa vastaan komentoja käyttöjärjestelmän suoritettavaksi.

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

1 JOHDANTO 1

1 DATAN NÄYTTEISTÄMINEN 2

2 TIETOKONEEN VALINTA 4
2.1 Projektiin vaadittavat lisälaitteet 4

3 UUDEN LAITTEEN KÄYTTÖÖNOTTO 5
3.1 Asentaminen 5
3.2 Yhteyden muodostaminen ilman näyttöä 5
3.3 Alkuasetusten suorittaminen 6
3.4 Skriptien siirtäminen ja kirjastojen asentaminen 7

4 DATAN LÄHETTÄMINEN 8
4.1 Datan lähettäminen SMTP -protokollan kautta 8
4.2 Datan lähettäminen SFTP -protokollan kautta 8

5 KÄYTTÖLIITTYMÄ 10

6 NÄYTTEENOTTO 11
6.1 Pääskripti 11
6.2 Mittaus 12

7 ETÄHALLINTA 16
7.1 Dataplicityn asentaminen ja käyttäminen 16

8 ONGELMAT, RATKAISUT JA KEHITTÄMINEN 17
8.1 Kirjastojen asentamisen ongelmat 17
8.2 Lähettämisen ongelmat 18
8.3 Datakatkokset 18
8.4 Data kirjoittaminen RAM-muistiin 19
8.5 Datan siivoaminen 20
8.6 Mittausnopeuden kasvatus 20
8.7 SMTP-protokollan hyödyntäminen 20

LÄHTEET 20
LIITTEET

1 JOHDANTO

Tässä opinnäytetyössä suunniteltiin järjestelmä, joka kerää anturidataa ja lähettää kerätyn datan tietyn väliajoin automaattisesti SFTP-palvelimelle. Järjestelmä kehitettiin sellaiseksi, että se on asentajaa ajatellen mahdollisimman helposti ja nopeasti käyttöönotettavissa. Projekti toteutettiin geoteknisiin mittalaitteisiin erikoistuneelle ja monitorointipalveluita tarjoavalle FinMeas Oy:lle.

Järjestelmän alustaksi valittiin yhden piirilevyn tietokone Raspberry Pi, joka tuli valituksi sen edullisuuden, toimintavarmuuden ja kattavan kirjasto- ja tukiverkoston vuoksi. Samaan järjestelmäkokonaisuuteen sisältyvät myös A/D-muunnin, jota tarvitaan analogisen anturisignaalin muuttamiseksi digitaaliseen muotoon, erillinen virtalähde virransyöttöön antureille sekä modeemi internet-yhteyden muodostamiseen datan lähettämistä ja etähallintaa varten. Internet-yhteys on jatkuva, jotta laitteen hallinta ja asetuksien muuttaminen etänä käy kätevästi. Etähallinnassa käytetään maksullista Unix-pohjaisille laitteille tarkoitettua Dataplicity-palvelua, jota voidaan käyttää joko suoraan internet-selaimen, mobiilisovelluksen tai erikseen tietokoneelle ladattavan ohjelman kautta.

Raspberryssä ohjelmointikielenä käytettiin Pythonia sen yleisyyden, helppokäyttöisyyden sekä sen vuoksi, että Pythonille on saatavilla paljon kirjastoja, joita tällaisissa projekteissa voi hyödyntää. Pythonilla ohjelmoitiin useita skriptejä, joilla jokaisella on jokin tehtävä, kuten datan kerääminen ja kirjoittaminen tekstitiedostoon sekä datan pakkaaminen ja lähettäminen. Pythonilla ohjelmoitiin myös etähallintaa varten käyttöliittymä, jonka avulla voidaan muuttaa asetuksia, kuten A/D-muuntimelta tulevan datan tarkkuutta ja mittaustaajuutta. Useimmat skripteistä olisi voitu kirjoittaa samaan tiedostoon, mutta selkeyden vuoksi teimme jokaiselle prosessille oman skriptin, jotka linkitimme tarvittaessa toisiinsa.

Raspberry Pi:n käyttöönottamisen kannalta tärkeimmät lähteet ja vinkit löytyivät raspberrypi.org -verkkosivustolta, josta löytyy kattavasti ohjeita erilaisiin projekteihin. Koodaamisen kannalta tärkeimmän lähteet olivat stackoverflow.com ja python.org.

1 DATAN NÄYTTEISTÄMINEN

Dataa käsitellessä ja erityisesti näytteenotosta puhuessa on hyvä tuntea joitain termejä ja ymmärtää niiden takana oleva teoria. Näytteenottotaajuus on termi, joka kuvastaa otettujen näytteiden määrää yhden aikayksikön, yleensä sekunnin, aikana. ADC Pi:n näytteenottotaajuus riippuu sen käyttämästä bittisyydestä ja on sidottu piirissä A/D muunnoksen suorittavien MCP3424-sirujen toimintaan. Vaihtoehtoja on neljä ja niiden näytteenottotaajuudet on testattu mittaamalla.

TAULUKKO 1. Näytteenottotaajuudet eri bittisyyksillä ja mittausasetuksilla mukaillen ABElectronicsin omia mittaustuloksia (ABElectronics 2019)

Bittisyys	Näytettä sekunnissa			
	Datalehti	Kanavat 1 ja 2		Kanavat 1 ja 5
		Jatkuva mittaus	Kertamittaus	Jatkuva mittaus
12	240	252.104	163.472	504.217
14	60	63.736	57.055	127.429
16	15	15.974	15.546	31.943
18	3.75	3.996	3.963	7.991

Taulukosta 1 ilmenevät mittausnopeudet eri asetuksilla. Jatkuva mittaus tarkoittaa, ettei mittaussirua päästetä lepotilaan mittausten välillä. Kertamittaus katkaisee mittausten välillä siruille menevän jännitteen ja on näin hiukan energiatehokkaampi. Koska mittauksista haluttiin mahdollisimman nopeita ja järjestelmän on tarkoitus olla jatkuvassa käytössä, mittausmuotona käytettiin työssä jatkuvaa moodia.

Mittausnopeuden suuri kasvu tietyillä kanavilla selittyy sillä, että ADC Pi-piirissä on käytetty kahta MCP3424-sirua, joissa on kussakin neljä analogista sisääntuloa. Mittausten välillä siru saa komennon sisääntulon vaihtamiselle, johon kuluu joitain millisekunteja. Kun käytössä on molemmilta siruilta vain yksi kanava, ei tätä vaihtoa tarvitse suorittaa ja molemmat sirut voivat toimia keskeytyksettä. Laitetta suunnitellessa oli yksi kriteereistä kuitenkin vähintään neljän kanavan yhtäaikainen mittaaminen vähintään 14 bitin tarkkuudella, joten nopeuden maksimiksi määrittyi tässä projektissa 63.736 näytettä sekunnissa.

Näytteen bittisyydestä puhuessa tarkoitetaan sen ilmaisun tarkkuutta. Bittisyys kuvastaa numeroväliä jolle näyte asettuu. Kun väli on suurempi mahdollisuuksien määrä minimin ja maksimin välillä kasvaa

ja tulos voidaan ilmaista tarkemmin. ADC Pi ilmaisee lukemat myös negatiivisina arvoina, tämä kuitenkin vähentää tulosten ääriarvoja yhden bitin verran. Ääriarvot voidaan määritellä seuraavalla kaavalla:

$$n=\pm 2^{b-1} \quad (1)$$

Missä n on ääriarvo ja b on käytetty bittisyys.

Kaavaan 1 sijoittamalla saadaan vaaditulla 14 bitin tarkkuudella tulokset väliltä ±8192. Näytteenottonopeus riittää myös 16-bittisellä tarkkuudella, jolla arvot saadaan väliltä ±32768.

2 TIETOKONEEN VALINTA

Projektin alkuvaiheessa tutkittiin, voisiko datan mittaamisen ja lähettämisen toteuttaa Arduino MKR 1000:lla. Suunnitelmat kariutuivat kuitenkin nopeasti Arduinon liian matalaan 12-bitin näytteenottotarkkuuteen vaatimuksen ollessa vähintään 14-bittiä (Arduino 2019). Tämän lisäksi Arduino ei kykene moniajoon, jolloin datan kerääminen olisi keskeytynyt lähettämisen ajaksi. Lopulta päädyimme ennestään opintojen kautta tuttuun Raspberry Pi-tietokoneeseen. Raspberry Pi on pääsääntöisesti harraste- ja oppimiskäyttöön tarkoitettu tietokone, jonka tehokkuus ja monipuolisuus ovat kasvaneet sen kehityskaaren myötä. Uusinta mallia Raspberry Pi 4 voidaankin verrata vanhoihin läppäreihin sen tehon perusteella. Raspberryn yleisyys on kasvanut ajan mittaan erilaisissa IoT-projekteissa, niin teollisissa kuin kotikäytöissäkin. Syynä tähän voidaan pitää ihmisten kasvanutta kiinnostuneisuutta ohjelmointia kohtaan sekä netistä löytyviä kattavia verkostoja, joista löytyy ohjeita eri projekteihin.

2.1 Projektiin vaadittavat lisälaitteet

Raspberry Pi ei itsessään ole täysin valmis paketti datan vastaanottamiseen ja käsittelyyn vaan toimiakseen se vaatii erillisiä lisälaitteita. Anturilta tuleva signaali on analoginen, joten se tulee muuttaa digitaaliseen muotoon. Raspberrystä kuitenkin puuttuu sisäinen A/D-muunnin, jolloin se tarvitsee erillisen ”hatun” hoitamaan signaalin muuttaminen digitaaliseksi. A/D-muuntimeen kytkettävät anturit vaativat myös osansa kuluttamalla virtaa sen verran, että Raspberryn oma virransyöttö ei kykene antamaan sitä tarpeeksi, jolloin tarvitaan erillinen virtalähde. Lisäksi tarvitaan vielä modeemi datan lähettämistä ja etäyhteyttä varten. Modeemi voi olla joko ethernet-liitännällä tai wifillä Raspberryyn yhteydessä, riippuen tilanteesta.

3 UUDEN LAITTEEN KÄYTTÖÖNOTTO

Uuden laitteen käyttöönottaminen aloitetaan käyttöjärjestelmän asentamisella micro SD-kortille, joka tullaan liittämään Raspberryyn. On mahdollista myös ostaa micro SD-kortteja, joissa on valmiiksi asennettu käyttöjärjestelmä, mutta tässä opinnäytetyössä asentaminen toteutetaan itse. Käyttöjärjestelmän asentamisella micro SD-kortille on useita ilmaisuja, kuten flashaaminen ja polttaminen. Selvyyden vuoksi käytetään tässä tapauksessa asentamista. Asennusvaiheen jälkeen kortti kytketään Raspberryyn ja voidaan aloittaa alkuasetusten tekeminen. Alkuasetuksilla tarkoitetaan tässä yhteydessä RapBerryn saattamista toimintakuntoon, tarvittavien kirjastojen asentamista sekä projektiin luotujen skriptien siirtämistä Raspberryyn ja niiden käyttöönottamista.

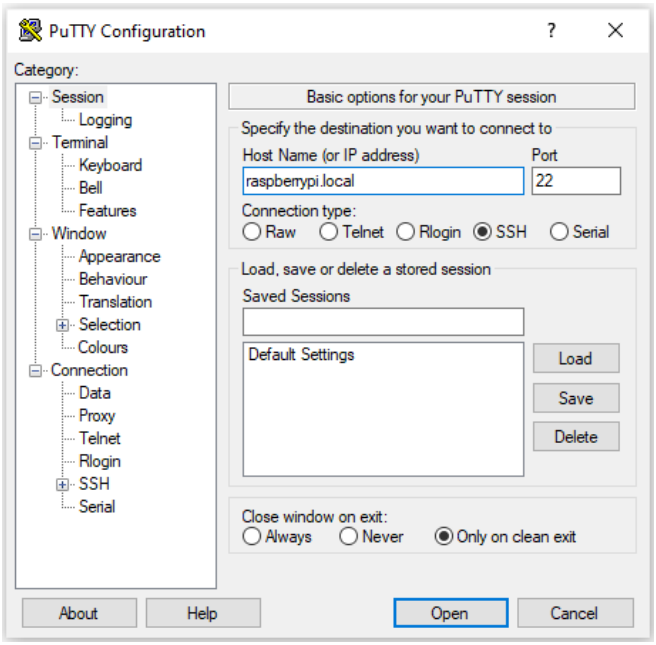
3.1 Asentaminen

Micro SD-kortin tulee olla alustettuna Fat-tiedostojärjestelmäksi ennen, kuin asennus voidaan aloittaa. Jos kortin koko on yli 32Gb eli suurempi, kuin Fat-tiedostojärjestelmä sallii, on kyseessä SDXC-kortti, joka tulee formatoida exFat-muotoon. Kun kortti on formatoitu oikeaan tiedostojärjestelmämuotoon, voidaan käyttöjärjestelmän asentaminen suorittaa erillisellä tietokoneelle ladattavalla ohjelmalla, kuten balenaEtcherillä. Asentamiseen tarvitaan myös itse käyttöjärjestelmä eli raspbian.iso-tiedosto, joka on ladattavissa ilmaiseksi Raspberryn kotisivuilta. Asentaminen itsessään käy helposti eikä zip-tiedostoa, jossa käyttöjärjestelmä on pakattuna, tarvitse purkaa, koska balenaEtcher tekee sen samalla, kuin asentaa käyttöjärjestelmää kortille (Raspberrypi).

3.2 Yhteyden muodostaminen ilman näyttöä

Alkuasetusten tekeminen ilman näyttöä on mahdollista, mutta se lisää huomattavasti työvaiheita. Tässä projektissa oli alun perin tarkoitus, että asetusten tekeminen tehdään näytöttömästi, mutta lopulta päädyttiin ratkaisuun hankkia työpaikalle tulevia asennuksia varten näyttö, näppäimistö ja hiiri. Näytöttömässä menetelmässä apuna käytetään PuTTy -nimistä ohjelmaa, jonka toiminta perustuu yhteyden muodostamiseen SSH-suojan yli. Juuri asennetussa Raspbian-käyttöjärjestelmässä, kuitenkin SSH-yhteys on oletusasetuksena pois päältä. Tämän vuoksi on luotu nerokas menetelmä yhteyden sallimiseen. Ennen juuri asennetun micro SD-kortin ottamista irti tietokoneesta, luodaan kortille tiedosto, jolla ei ole tiedos-

tomuotoa. Helpoiten tämä käy luomalla tekstitiedosto, josta poistetaan tiedostopääte. Nimeksi tiedostolle annetaan ”ssh”. Kun kortti liitetään Raspberryyn ja kytketään virrat, muuttaa tuo juuri luotu tiedosto SSH-yhteyden sallituksi. Raspberryssä WLAN-asetukset eivät ole valmiina kohdillaan, joten tarvitaan Ethernet kaapeli. Kaapeli voidaan kytkeä, joko suoraan Raspberrystä tietokoneeseen tai modeemiin, johon myös tietokone on yhdistetty kaapelilla tai langattomasti.

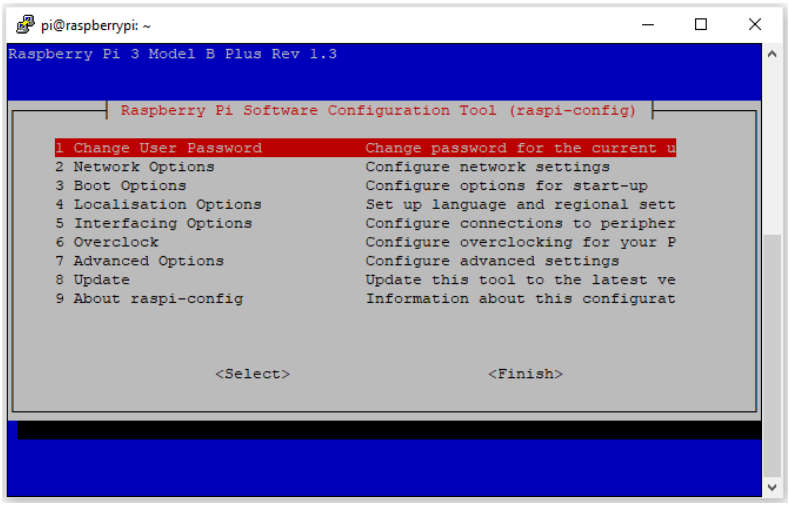


KUVA 1. Yhdistäminen Raspberryn ja tietokoneen välille Ethernet kaapelilla.

Kun laitteet ovat yhdistettynä toisiinsa, voidaan yhteys muodostaa kirjoittamalla Host Name -kohtaan ”raspberrypi.local”. Muutoin joudutaan scannaamaan Raspberryn IP-osoite esimerkiksi Advanced IP Scanner-ohjelmalla. Lopuksi katsotaan, että portti on 22 ja yhteystyyppi on asetettu kohtaan SSH, jolloin voidaan muodostaa yhteys painamalla Open.

3.3 Alkuasetusten suorittaminen

Kun yhteys on luotu joko PuTTyn avulla tai suoraan näyttöön, aloitetaan alkuasetusten suorittaminen. Asetusvalikko saadaan avattua kirjoittamalla terminaaliin komento: ”sudo raspi-config”.



KUVA 2. Asetukset-valikko

Valikon tultua näkyviin suoritetaan tiedostojärjestelmän laajennus kohdasta Advanced Options. Tämä toimenpide mahdollistaa kaiken micro SD-kortilla olevan vapaan tilan hyödyntämisen omaan käyttöön. Seuraavaksi sallitaan Interfacing Options-valikosta I2C, jota tarvitaan A/D-muunninta varten ja määritetään internetyhteys Network Options- kohdasta jos käytetään WLAN-yhteyttä. Lopuksi päivitetään pakettilista eli tieto pakettilähteiden sisällöstä komennolla: ”sudo apt-get update” ja päivitetään järjestelmään asennetut paketit komennolla: ”sudo apt-get upgrade”. Komennon osa ”apt” viittaa pakettienhallintatyökaluun, joka huolehtii pakettien riippuvuussuhteista ja niiden päivittämisestä (Linux).

3.4 Skriptien siirtäminen ja kirjastojen asentaminen

Näytöttömässä asentamisessa skriptien siirtäminen tapahtuu PuTTy:n ja muistitikun avulla, jonka muistissa tarvittavat skriptit tulee olla. Muistitikku kytketään Raspberryyn ja käynnistetään terminaali PuTTy:n kautta. Terminaaliin annetaan komento: ”cp -a /media/pi/muistitikku/skriptikansio/. /home/pi”, joka kopioi skriptit ja liittää ne Raspberryn muistiin. Näytön avustuksella skriptien kopiointi on huomattavasti helpompaa: laitetaan muistitikku sisään ja kopioidaan tiedostot, kuten normaalisti tietokoneella. Lopuksi aloitetaan kirjastojen asentaminen kirjoittamalla terminaaliin komento: ”python install.py”. Tässä kohtaa Raspberryn tulee olla yhteydessä internettiin. Asennusskripti install.py luotiin helpottamaan ja nopeuttamaan laitteen käyttöönottoa. Mittausprosessiin ja lähettämiseen vaaditaan kaiken kaikkiaan kahdeksan kirjastoa, joiden asentaminen ulkoistettiin asennusskriptille. Skriptiin sisällytettiin myös järjestelmän pakettien päivittäminen, data_log-kansion luominen, watchdog-asetusten laittaminen kohdilleen sekä Raspberryn laitekohtaisen serial-tunnuksen hakeminen ja tallentaminen config.py-asetustiedostoon. Serial-tunnus on siitä tärkeä, että lähetettävät datapaketit nimetään niiden mukaan.

4 DATAN LÄHETTÄMINEN

Projektin kannalta oleellisin asia on saada mitattu anturidata pakattuna palvelimelle, josta se voidaan siirtää edelleen jatkokäsiteltäväksi. Projektin alkuvaiheessa annettiin tutkittavaksi kaksi eri menetelmää datan lähettämiseen ja vastaanottamiseen, joista tulisi valita parempi. Ensimmäinen vaihtoehto oli datan lähettäminen sähköpostiin ja toinen vaihtoehto oli datan lähettäminen SFTP -palvelimelle. Ensimmäisessä vaihtoehdossa lähettäminen tapahtuisi käyttämällä SMTP -protokollaa, mutta heti alussa ongelmaksi havaittiin, että jokainen laite tarvitsisi oman sähköpostiosoitteen tai samaa osoitetta käytettäessä lähetysaikaa tulisi porrastaa, niin ettei lähettämisessä tulisi päällekkäisyyttä. Jälkimmäisessä vaihtoehdossa tätä ongelmaa ei ilmennyt, vaikka lähetysvaiheessa ohjelma käyttääkin kirjautuessaan SFTP -palvelimelle samaa tunnusta useiden eri laitteiden kohdalla.

4.1 Datan lähettäminen SMTP -protokollan kautta

Sähköpostiviestien lähettäminen itsessään käy helposti Pythonille suunnitellun smtplib -kirjaston avulla, jolla viestejä voidaan lähettää tekstin muodossa. Jotta data voidaan lähettää pakattuna, tarvitaan myös email -kirjastoa. Lisäksi lähettämiseen tarvitaan SMTP -palvelimen osoite ja portti sekä erillinen lähettäjänä toimiva sähköpostiosoite. Tämä osoite on suositeltavaa luoda erikseen pelkästään tähän tarkoitukseen, sillä sähköpostitilin pääsyoikeuksia tulee muuttaa asetuksista niin, että kolmannella osapuolella on mahdollisuus sen käyttöön. Tämä muutos heikentää sähköpostitilin turvallisuusasetuksia.

E-mail provider	SMTP server address	SMTP port
Yahoo	smtp.mail.yahoo.com	465
AOL	smtp.aol.com	587
Outlook	smtp.live.com	587
Gmail	smtp.gmail.com	587 (TLS/STARTTLS), 465 (SSL)

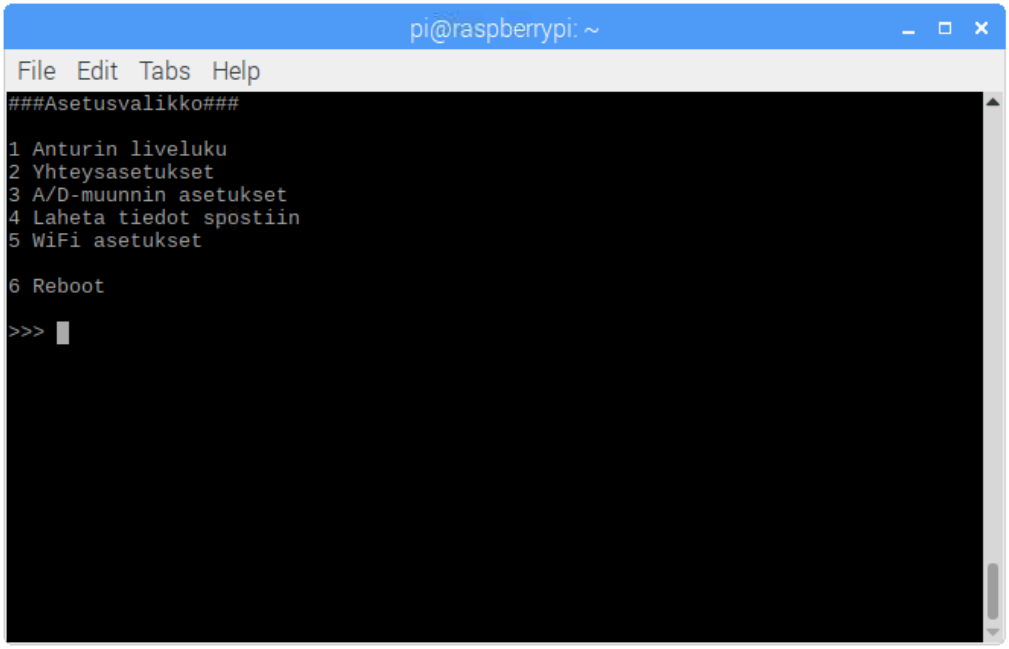
KUVA 3. Lista suosituimmista SMTP -servereistä ja niiden porteista (Ionos 2018).

4.2 Datan lähettäminen SFTP -protokollan kautta

Paremmaksi tavaksi lähettää mittausdataa, osoittautui maksullinen SFTP -palvelin, jonka saa käyttöönsä rekisteröitymällä jollekin palveluntarjoajan sivustolle. Esimerkkejä näistä ovat muun muassa Google Cloud ja Amazonin omistama AWS. Kun palveluun luodaan käyttäjä, saadaan jokin tietty määrä tallennustilaa käyttöön sekä IP -osoite, jota tarvitaan datan lähettämisessä. Pythonille löytyy paramiko -kirjasto, jolla luodaan SSH2 -protokollan avulla suojattuja yhteyksiä eri laitteiden välille. Tämän kirjaston avulla pystyy lähettämään dataa SFTP -palvelimelle, kun tiedetään palvelimen IP -osoite ja portti sekä käyttäjätunnus ja salasana. SFTP -palvelimen etuna on myös se, että dataa on helppo poimia sieltä jatkokäsittelyä varten.

5 KÄYTTÖLIITTYMÄ

Mittaus- ja lähetysprosessit ovat täysin automatisoituja, mutta niiden toiminnan kannalta tulee config.py -asetustiedostossa olla oikeat parametrit. Näitä parametreja ovat muun muassa antureiden lukumäärä, mittaustarkkuus eli bitrate sekä SFTP-palvelimen yhteysasetukset. Riippuen asennuskohteesta, oletus-asetukset eivät usein satu kohdilleen vaan ne tulee vaihtaa. Asetuksien vaihtamista varten luotiin Suomenkielinen käyttöliittymä, joka suoritetaan terminaalissa komennolla ”python Menu.py”. Käyttöliittymää voidaan käyttää paikallisesti muodostamalla Raspberry:n ja tietokoneen välille yhteys PuTTy:llä tai näytön välityksellä, mutta käyttöliittymän pääsääntöinen tarkoitus on etähallinta Dataplicity -palvelun kautta.



KUVA 4. Käyttöliittymän päävalikko.

6 NÄYTTEENOTTO

Näytteenoton haasteina oli tehdä siitä jatkuvaa, tarpeeksi nopeaa ja tarkkaa. Mittausnopeuden maksimin määritteli ADC Pi-kortin näytteenottotaajuus, joka oli selvästi hitaampi kuin esimerkiksi Raspberryn kellotaajuus. Tämän vuoksi ohjelmoitaessa ei tarvinnut huolehtia koodin monimutkaisuudesta tai mittausasetusten hausta skriptin ulkopuolisesta tiedostosta. Vähimmäistarkkuudeksi oli projektin suunnitteluvaiheessa määritelty 14 bittiä, johon ADC Pi kykeni halutulla kymmenen näytettä sekunnissa -nopeudella. Ainoa koodissa ratkaistava ongelma oli jatkuvan mittauksen toteutus. Varsinainen näytteenotto toteutettiin kahdella erillisellä skriptillä (LIITTEET 8 ja 10).

6.1 Pääskripti

Toimintavarmuuden lisäämiseksi tehtiin pääskripti, jonka crontab ajaa välittömästi Raspberryn käynnistyttyä. Tämä koodi tarkistaa onko ethernet-porttiin kytkettynä RJ-45 kaapeli ja ajaa sen jälkeen päälle käyttöliittymän, tai kaapelin ollessa irti mittaussilmukan. Pääskripti on mahdollista käynnistää paikallisesti terminaalikomennolla “python run.py”. Mikäli mittauksen aikana tapahtuu odottamaton virhe, mittaussilmukan ajo keskeytyy mutta pääohjelma käynnistää sen välittömästi uudestaan. Näin suuri osa mittauksen aikana tulevista virheistä korjautuu automaattisesti, eikä keskeytynyttä mittausta tarvitse käsin käynnistää uudestaan.

```
13  def info():
14      sana = (getlinkstate())
15      if sana=="u":
16          print ("Ethernet-cable connected")
17          time.sleep(2)
18          os.system("python Menu.py")
19      else:
20          print ("Ethernet-cable disconnected")
21          time.sleep(2)
22          while True: #Start mreasurement loop. In case of error a new file is created.
23              print ("Measuring...\n")
24              time.sleep(2)
25              os.system("python loop.py")
26              print ("Something unexpected occurred. \nSetting up a new file.\n")
27              time.sleep(2)
```

KUVA 5. Run.py:n päätoiminto.

6.2 Mittaus

Jotta mittausten tulokset olisivat halutunlaisia riippumatta käytettyjen anturien määrästä ja jotta samaa koodia olisi mahdollista käyttää useissa eri laitteissa, asetukset mittauksille asetettiin laitekohtaiseen tiedostoon. Tämä tiedosto nimettiin config.py:ksi. Ennen kuin koodi antaa yhtään mittauskäskyä, se hakee config.py:stä joitain olennaisia tietoja. Näitä ovat bitrate, mittausaika, joka tallennetaan yhteen tiedostoon, sensorien määrä ja portit, joihin ne on kytketty, Raspberryn sarjanumero, SFTP-palvelimen asetukset, käyttäjänimi ja salasana.

```
1   settings = dict(
2       Bitrate = 16,
3       minutes = 15,
4       sensors = [0,1,1,0,0,0,0,0,0],
5       serial = '00000000609a1234',
6       ip = 'device.osoite.com',
7       port = 22,
8       passWord = 'salasana4321',
9       userName = 'device',
10      EMAIL_ADDRESS = 'esimerkki@gmail.com',
11      PASSWORD = 'salasana1234'
12  )
```

KUVA 6. Config-tiedostoon tallennetut asetukset.

Kun mittausskripti on hakenut käytetyt asetukset config.py:stä, se määrittelee joitain käyttämiään hakutoimintoja. Datatiedoston alkuun sijoitetaan tieto laitteen omasta sen hetkisestä IP-osoitteesta sekä laitteen sisäisestä lämpötilasta.

```
32  def meas_temp(): #Pi internal temp for data file
33      temp = os.popen ("vcgencmd measure_temp").readline()
34      return (temp.replace("temp:",""))
35
36  def get_ip(): #IP for data file
37      s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
38      try:
39          s.connect(('10.255.255.255', 1))
40          IP = s.getsockname()[0]
41      except:
42          IP = '127.0.0.1'
43      finally:
44          s.close()
45      return IP
```

KUVA 7. Lämpötilan ja IP-osoitteen tarkastus.

Tämän jälkeen määritellään varsinainen mittaustoiminto. Koodi olisi voitu toteuttaa yksinkertaisella for-loopilla, mutta huomattavasti pidempi if−else-rakenne mahdollisti anturien kytkemisen ADC Pi:lle

missä tahansa järjestyksessä. For-rakenne olisi pakottanut käyttämään aina ensimmäisiä vapaana olevia sisääntuloja ja täyttämään ne numerojärjestyksessä.

```
47    def measur():
48        adc = ADCPi(0x68, 0x69, Bitrate)
49        adc.set_conversion_mode(1)
50        mea1=str()
51        mea2=str()
52        mea3=str()
53        mea4=str()
54        mea5=str()
55        mea6=str()
56        mea7=str()
57        mea8=str()
58        if sn[1]==1:
59            mea1=(';%01i' % adc.read_raw(1))
60        else:
61            mea1=(';N')
62        if sn[2]==1:
63            mea2=(';%01i' % adc.read_raw(2))
```

KUVA 8. Mittaustoiminnon määrittely.

Jokainen mittauskanava määriteltiin string-muuttujaksi, joka riippuen siitä onko kanava config.py:ssä määritelty käytettäväksi saa joko numeroarvon mittaustuloksen muodossa, tai kirjaimen N merkitsemään, ettei kanavalla ole anturia. Lopussa jokainen string-muuttuja yhdistetään yhdeksi string-ketjuksi, joka on measur()-toiminnon palautettava arvo.

```
90        measnn = str()
91        measnn = ("%s%s%s%s%s%s%s%s" % (mea1, mea2, mea3, mea4, mea5, mea6, mea7, mea8))
92        return measnn
```

KUVA 9. Tulosten yhdistäminen yhdeksi string-muuttujaksi.

Seuraavaksi koodiin määriteltiin datatiedoston avaaminen, sen asetukset ja aikaleiman käyttö. Ensimmäiseksi tiedostoa varten määritellään sen vastaanottamien näytteiden määrä. Tämä luku on yksinkertaisen laskutoimituksen tulos: Näytteenottotaajuus jaetaan sensorien määrällä ja jakomäärä kerrotaan tämän jälkeen 60:llä ja halutulla minuuttimäärällä joka datatiedoston tulee sisältää. Datatiedoston nimi määräytyy sen luomisajankohdan ja Raspberry Pi:n sarjanumeron perusteella. Alkuun sijoitetaan laitteen sen hetkinen lämpötila ja IP-osoite, kukin omalle rivilleen. Tämän jälkeen aletaan tulostaa riveittäin ensin aikaleima ja sen perään lukemat käytössä olevilta antureilta.

```
94   def meas():
95       Fs = int((SPS/sensors)*60*minutes) #Number of samples stored in a file
96       if Fs==0:
97           Fs=1
98       filename = "%s_%s.txt" % (serial, (time.strftime('%Y_%m_%d_%H_%M'))) #naming
         current file
99       f1=open("./data_log/%s" % (filename), 'w+') #opens file for writing
100      measn = str()
101      f1.write(("IP address: %s\n") % (get_ip())) #Print internal temp and IP address to
         data file
102      f1.write(("Pi internal %s") % (meas_temp()))
103      i = 0 #loop reset
104      while i<Fs:
105          t = datetime.now()
106          f1.write("%s" % t) #Print timestamp
107          measn = (measur())
108          f1.write(measn)
109          f1.write("\n")
110          i += (1)
111          print i
112          print measn
```

KUVA 10. Datatiedoston luonti ja tulosten tallentaminen.

ADC Pi:n kirjastoon kuuluivat komennot sen porttien lukemiseen. Data oli mahdollista saada joko jännitetietona tai numeroarvona bitrate:n ääriarvojen väliltä. Tilaajan toiveesta päädyimme käyttämään numeroarvoja, jotka ovat tulevaisuudessa tarkoitus skaalata keskenään vertailukelpoisiksi eli 18-bittisiä arvoja vastaaviksi.

Valmistuttuaan tiedosto siirretään asennuksen yhteydessä luotuun data_log-kansioon. Kansiossa tiedostot ovat välivarastossa odottamassa lähetykselle määriteltyä kellonaikaa. Tällöin kansioon kertyneet tiedostot pakataan yhdeksi arkistoksi ja lähetetään SFTP-palvelimelle. Pakkaaminen suoritetaan, jotta lähetettävän tiedoston käsittely olisi helpompaa. Lähettäminen on yksinkertaisempaa, kun käsiteltävänä on vain yksi zip-muotoinen tiedosto useiden tekstitiedostojen sijaan. Lisäksi pakattu tiedosto on kooltaan pienempi ja säästää hiukan palvelimen rajallista tilaa. Pakkaaminen toteutettiin käyttämällä valmista shutil-pythonkirjastoa.

```
1   import os
2   import time
3   import config
4
5   minutes = config.settings.get('minutes')
6
7   path = "/home/pi/data_log/"
8   def flushdir(dir):
9       now = time.time()
10      for f in os.listdir(dir):
11          fullpath = os.path.join(dir, f)
12          if os.stat(fullpath).st_mtime < (now - ((minutes*60)+1)): #age of files in
            seconds
13              if os.path.isfile(fullpath):
14                  os.remove(fullpath)
15              elif os.path.isdir(fullpath):
16                  flushdir(fullpath)
17
18  flushdir(path)
```

KUVA 11. Vanhojen tiedostojen poisto Sysflush-skriptillä

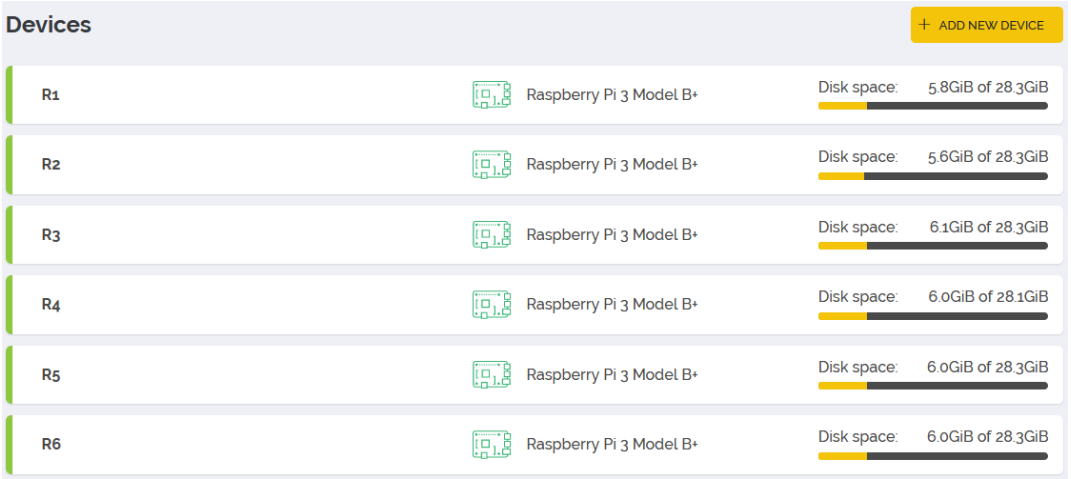
Mikäli lähetys onnistuu, kansiossa olevat vanhat tiedostot poistetaan erillisellä sysflush-skriptillä. Sysflush yksinkertaisesti vertailee kansiossa olevien tiedostojen ikää ennalta asetettuun raja-arvoon ja tämän jälkeen poistaa kaikki tämän iän ylittävät tiedostot. Tämän tarkoitus on estää tallennustilan hiljattainen täyttyminen, sekä varmistaa ettei samoja tietoja lähetetä palvelimelle useita kertoja.

7 ETÄHALLINTA

Kun laite on aloittanut datan keräämisen ja lähettämisen, saattaa jossain vaiheessa tulevaisuudessa tulla vastaan tilanne, että halutaan esimerkiksi saada tarkempia mittausarvoja, jolloin tulee voida muuttaa asetuksia. Sen sijaan, että lähetettäisiin työntekijä vaihtamaan asetuksia paikalliskäytöllä toiseen päähän Suomea, on järkevämpää käyttää etähallintaan tarkoitettua Dataplicity -palvelua. Dataplicity on Linux -pohjaisille laitteille tarkoitettu maksullinen etähallintasovellus, joka asennetaan Raspberryy:n terminaalin kautta. Dataplicityä voidaan käyttää internet-selaimen kautta, erikseen tietokoneelle ladattavalla ohjelmalla tai kännykkäsovelluksella. Dataplicityn toimintaa kuvataan liitteessä 14.

7.1 Dataplicityn asentaminen ja käyttäminen

Dataplicityn käyttäminen edellyttää palveluun rekisteröitymistä. Rekisteröitymisen jälkeen käyttäjä saa Dataplicityn kirjaston asennuslinkin, joka tulee kopioida ja suorittaa Raspberry:n terminaalissa. Asennuslinkki on käyttäjäkohtainen ja sen tarkoitus on varmistaa, että laite liitetään oikealle käyttäjälle. Tämän vuoksi kyseistä asennuslinkkiä ei sisällytetä install.py -asennusskriptiin. Asennuksen jälkeen Dataplicityn sivuille ilmestyy Devices -valikkoon uusi laite, jota klikkaamalla aukeaa terminaali. Terminaalin saatua yhteys Raspberryyn, kirjaudutaan sisään käyttäjätunnuksella ja salasanalla, jotka ovat oletuksena ”su pi” ja ”raspberry”. Käyttäjätunnuksen edessä oleva ”su” on lyhenne sanoista ”super user”. Sisäänkirjautumisen jälkeen annetaan komento: ”python /home/pi/Menu.py”, jolloin käyttöliittymä käynnistyy ja asetuksia voidaan muuttaa piittaamatta siitä, kuinka kaukana itse laite on.



KUVA 12. Devices -valikko, jossa kuusi Raspberry Pi -laitetta

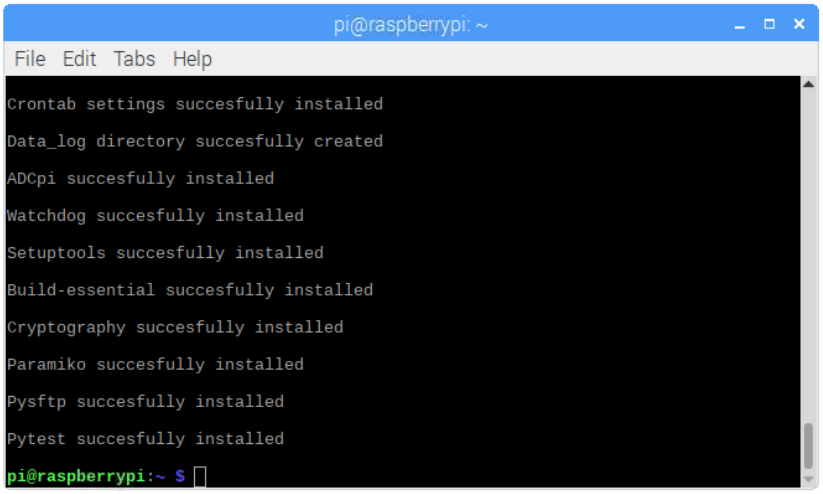
8 ONGELMAT, RATKAISUT JA KEHITTÄMINEN

Suunnitteluvaiheessa ilmoitettiin, että laite tulisi saattaa käyttökuntoon syksyyn 2019 mennessä. Tavoite saavutettiin, mutta laitteessa oli edelleen joitain ongelmia ja kehittämistarpeita muun muassa kirjastojen asentamisessa ilmenevät ongelmat, datan kasautuminen, datakatkokset, datan kirjoittaminen RAM-muistiin SD-kortin säästämiseksi ja turhan datan siivoaminen, joka säästää muistia ja nopeuttaa lähettämistä.

8.1 Kirjastojen asentamisen ongelmat

Jokainen kirjasto asentuessaan luo kansion, jossa kirjaston omat tiedostot sijaitsevat. Asennusskripti install.py tarkistaa asennusohjelman lopussa jokaisen kirjaston kohdalta, onko niiden kansiot olemassa ja tulostaa tämän perusteella lopputuloksen. Tämä menetelmä kuitenkin havaittiin puutteelliseksi, sillä kirjaston kansion sijainti saattaa muuttua eri päivityksissä, kun taas install.py-skriptissä sijainti on asetettu vakioksi. Tämä johtaa siihen, että ohjelma tulostaa kirjaston asentamisen epäonnistuneen, vaikka asentaminen olisi onnistunut. Vastaan on tullut myös ongelma, jossa kirjaston asennuslinkki on muuttunut ajan myötä, jolloin asennusskriptissä oleva linkki on toimimaton. Tästä seuraa, että kirjasto ei asennu ja alkaa oikean asennuslinkin selvittäminen. Asentumattoman kirjaston asentaminen tulee lopulta suorittaa erikseen terminaalin kautta.

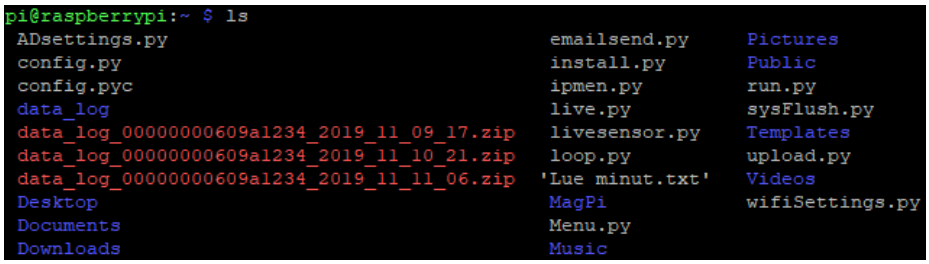
Asentamisen yhteydessä ilmenevä ongelma voidaan kuitenkin ratkaista yksinkertaisesti luomalla kopio toimivan järjestelmän SD-kortista esimerkiksi Win32DiskImager-ohjelmalla ja asentamalla tätä kopiota toisiin kortteihin. Install.py-skriptiä tarvitaan tällöin vain Raspberryn sarjanumeron hakemiseen ja sen lisäämiseen config.py-asetustiedostoon, sillä se on ainut eroavaisuus laitteiden välillä. Asentamalla kopiota toisiin kortteihin säästyy paljon aikaa ja vaivaa.



KUVA 13. Install.py-ohjelma tulostaa lopuksi, onnistuiko asentaminen.

8.2 Lähettämisen ongelmat

Crontab on ohjelmoitu suorittamaan lähettämiseen tarkoitettu upload.py-skripti, jokaisella tasatunnilla. Tällöin data_log-kansioon kerätty tekstitiedoston muodossa oleva data pakataan zip-tiedostoon ja siirretään sijaintiin ”/home/pi” lähetettäväksi. Joskus lähetys kuitenkin epäonnistuu jostakin tuntematto-masta syystä ja muistiin alkaa kerääntyä vanhoja zip-tiedostoja.



KUVA 14. Muistiin kerääntynyt vanha data zip-tiedostoina.

Ongelma voidaan ratkaista lisäämällä upload.py-skriptiin toiminto, joka tarkistaa onnistuneen lähetyk-sen jälkeen, onko tiedostosijainnissa zip- tiedostopäätteisiä tiedostoja ja jos on, niin tiedosto poistetaan. Muutoin ohjelma suljetaan. Näin datapaketit eivät pääse täyttämään muistia.

8.3 Datakatkokset

Kun koodia alettiin kehityksen alkuvaiheessa testaamaan, tulokset tallentuivat kuten oli odotettu ja uusi datatiedosto syntyi haluttuun sijaintiin kuten pitikin. Joidenkin päivien testikäytön jälkeen huomioitiin kuitenkin, että tiedostojen välissä oli joidenkin kymmenien millisekuntien katkos mittauksissa. Suurilla bittisyyksillä tämä oli jäänyt huomaamatta mutta pienemmillä bittisyyksillä mittausnopeuden kasvaessa ongelma alkoi korostua. Syyksi paljastui lopulta käyttöjärjestelmän hitaus. Uuden tekstitiedoston avaamiseen kului aika, joka näkyi hetken taukona skriptin odottaessa uutta tiedostoa kirjoittamista varten.

Ongelma ratkaistiin sijoittamalla mittauslooppiin bufferi, joka tarkkailee tiedoston lopun lähestymistä. Bufferille haettiin kokeellisesti arvot, joilla taukoa ei enää havaittu. Esimerkiksi 12-bitin tarkkuudella sen arvo on 60. Tämä tarkoittaa, että kun mittausta on jäljellä enää 60 riviä, skriptissä ajetaan ylimääräinen toiminto. Käyttöjärjestelmälle annetaan tässä vaiheessa käsky avata ja nimetä seuraava tiedosto, mutta tulosten tallentaminen jatkuu edelleen aiempaan. Kun alkuperäisen tiedoston maali saavutetaan, on seuraava jo auki odottamassa lämpötilan ja IP-osoitteen kirjoittamista.

```
113             if  i==Fs-BFF: #Open a separate file in advance to get rid of gap between
                measurements
114                 filename = "%s_%s.txt" % (serial, (time.strftime('%Y_%m_%d_%H_%M')))
115                 f2=open("./data_log/%s" % (filename), 'w+')
116                 i += (1)
117                 print i
118                 print ("new file done")
119                 if Bitrate == (18):
120                     f1.close()
121                     f1 = f2
122                     i += (1)
123                     print ("switched file")
124             if i==Fs-1:
125                 f1.close() #close file for storage
126                 f1 = f2 #switch to preopened file
127                 i += (1)
128                 print("switched file")
```

KUVA 15. Tiedoston vaihto ja bufferin toiminta.

8.4 Data kirjoittaminen RAM-muistiin

Loop.py-skriptin jauhaessa dataa, on SD-kortti jatkuvassa käytössä, jolloin sen käyttöikä lyhenee huomattavasti. Tämä ongelma on mahdollista ratkaista asentamalla laitteeseen Log2Ram-ohjelma, joka luo RAM-muistiin väliaikaisen paikan tiedostoille, jotka sijaitsevat tiedostosijainnissa ”/var/log”. Oletusasetuksissa on asetettuna, että cron siirtää joka tunti tiedostosijaintiin kertyneen datan SD-kortille, jolloin säästytään huomattavalta määrältä kirjoituskertoja. Loop.py-skriptiin voisi tehdä muutoksen, jossa dataa kerätään väliaikaisesti RAM-muistissa olevaan tekstitiedostoon ja tiedoston tullessa config.py-asetusten mukaiseksi, se siirretään SD-kortille (Log2Ram 2019; Styger).

8.5 Datan siivoaminen

Tällä hetkellä dataa kerätään talteen sillä taajuudella, jolla A/D-muunnin sitä tarjoaa. Palvelimelta dataa keräävä algoritmi, joka sijoittaa datan graafeihin, suodattaa datasta liian lähellä toisiaan olevat arvot. Tämä suodatus olisi kuitenkin järkevämpää tehdä jo datan kirjoitusvaiheessa, jolloin säästetään muistia ja nopeutetaan lähettämistä. Loop.py-skriptiin voisi luoda toiminnon joka ensimmäisellä kerralla tallentaa muistipaikkaan arvot, jotka se saa A/D-muuntimelta. Tämän jälkeen uusia arvoja verrataan muistipaikan arvoihin ja jos näiden itseisarvo ylittää sallitun rajan, korvataan muistipaikat uusilla arvoilla ja tallennetaan ne tekstitiedostoon. Skriptiin voisi myös lisätä aikamuuttujan, että jos aikaisempi ehto ei täyty, tallennetaan näyte tietyn ajan välein, jolloin ei pääse syntymään pitkiä datattomia välejä.

8.6 Mittausnopeuden kasvatus

Mittausnopeus on kasvatettu käytössä olleiden laitteiden rajojen ja asiakkaan latuvaatimuksien rajoissa niin suureksi kuin se on mahdollista. Tulevaisuudessa laitteistoa on kuitenkin mahdollista päivittää vaihtamalla AD-muuntimeksi ADC Pi-piirin kehittyneempi versio, joka julkistettiin pian ensimmäisten laitteiden testikäytön alkamisen jälkeen. Koska laite on mittausnopeutta lukuun ottamatta käytännössä identtinen, on vanha koodi todennäköisesti hyödynnettävissä hyvin pienillä muutoksilla. Lisäksi ainoat päivitystä vaativat skriptit ovat päämittaus ja paikalliskäytön reaaliaikaisen tarkkailun skripti, koska muu koodi on lähinnä tiedostonhallintaa ja asetusten muuttamista varten.

8.7 SMTP-protokollan hyödyntäminen

Opinnäytetyössä tutkittiin SMTP-protokollan roolia datan lähettämiseen, mutta SFTP-protokollan todettiin olevan parempi, jolloin on herännyt ajatus, että voisiko SMTP-protokollaa käyttää johonkin muuhun tarkoitukseen. Tällainen tarkoitus voisi olla hetkellisten mittausarvojen lähettäminen tai virhelokitietojen lähettäminen jollekin yrityksen henkilölle.

LÄHTEET

Abelectronics. ADC Sample Rate Comparison. Saatavissa: https://www.abelectronics.co.uk/kb/article/1047/adc-sample-rate-comparison. Viitattu 7.7.2019

Arduino. Arduino MKR1000 WIFI. Saatavissa: https://store.arduino.cc/arduino-mkr1000-wifi. Viitattu: 12.8.2019

Dataplicity. How it works. Saatavissa: https://docs.dataplicity.com/docs/how-it-works. Viitattu: 15.8.2019

E. Styger. Log2Ram. Saatavissa: https://mcuoneclipse.com/2019/04/01/log2ram-extending-sd-card-lifetime-for-raspberry-pi-lorawan-gateway/. Viitattu: 26.1.2020

Ionos. What is SMTP? Definition and basics. Saatavissa: https://www.ionos.com/digitalguide/e-mail/technical-matters/smtp/. Viitattu: 22.8.2019

Linux. APT. Saatavissa: https://www.linux.fi/wiki/APT. Viitattu: 19.7.2019

Raspberrypi. Installing images. Saatavissa: https://www.raspberrypi.org/documentation/installation/installing-images/README.md. Viitattu: 16.7.2019

```
import config
import os, sys
import time

Bitrate = config.settings.get("Bitrate")
minutes = config.settings.get("minutes")
sensors = config.settings.get("sensors")
srs = str(sensors).strip('[]')

def sum_digits(digit):
    return sum(int(x) for x in digit if x.isdigit())
Sens = (sum_digits(srs))

def submenu():
    os.system("clear")
    print("###A/D-muunnin asetukset###\n\n1 Tarkkuus: "+str(Bitrate)+" bit\n2
    Datatiedosto: "+str(minutes)+" min\n3 Anturit: "+str(Sens)+" kpl\n\n4 Palaa
    alkuvalikkoon\n")
    settings = input(">>> ")
    if settings == 1:
        measbitrate()
    elif settings == 2:
        measminutes()
    elif settings == 3:
        meassensors()
    elif settings == 4:
        os.system("python Menu.py")
    else:
        print"Arvon tulee olla 1 - 4"
        submenu()


def replace_line(file_name, line_num, text):
    lines = open(file_name, 'r').readlines()
    lines[line_num] = text
    out = open(file_name, 'w')
    out.writelines(lines)
    out.close()


def bitratereplace(newBitrate):
    alku = "    Bitrate = "
    enter = ",\n"
    kaikki = (alku+str(newBitrate)+enter)
    replace_line('config.py', 1, kaikki)
    print('Uusi arvo on nyt: ' + str(newBitrate))
    time.sleep(1)
    os.system('python ADsettings.py')
    return newBitrate


def minutesreplace(newMinutes):
    alku = "    minutes = "
    enter = ",\n"
    kaikki = (alku+str(newMinutes)+enter)
    replace_line('config.py', 2, kaikki)
    print('Uusi arvo on nyt: ' + str(newMinutes))
    time.sleep(1)
    os.system('python ADsettings.py')
    return newMinutes


def sensorsreplace(newS, newSensors):
    alku = "    sensors = "
    enter = ",\n"
    kaikki = (alku+str(newS)+enter)
    replace_line('config.py', 3, kaikki)
```

```
66      print('Uusi arvo on nyt: ' + str(newSensors))
67      time.sleep(1)
68      os.system('python ADsettings.py')
69      return newS,newSensors
70
71
72  def measbitrate():
73      os.system("clear")
74      newBitrate = input("Mittaustarkkuus: ")
75      if newBitrate == 12:
76          bitratereplace(12)
77      elif newBitrate == 14:
78          bitratereplace(14)
79      elif newBitrate == 16:
80          bitratereplace(16)
81      elif newBitrate == 18:
82          bitratereplace(18)
83      else:
84          print"Arvon tulee olla 12, 14, 16 tai 18!"
85          time.sleep(2)
86          measbitrate()
87
88
89  def measminutes():
90      os.system("clear")
91      newMinutes = input("Mittausaika: ")
92      if newMinutes <= 60 and newMinutes >= 1:
93          minutesreplace(newMinutes)
94      else:
95          print"Arvon tulee olla valilta 1 - 60"
96          time.sleep(2)
97          measminutes()
98
99
100 def meassensors():
101     os.system("clear")
102     newSensors = input("Anturien maara: ")
103     if newSensors == 1:
104         sensorsreplace("[0,1,0,0,0,0,0,0,0,0]", 1)
105     elif newSensors == 2:
106         sensorsreplace("[0,1,1,0,0,0,0,0,0,0]", 2)
107     elif newSensors == 3:
108         sensorsreplace("[0,1,1,1,0,0,0,0,0,0]", 3)
109     elif newSensors == 4:
110         sensorsreplace("[0,1,1,1,1,0,0,0,0,0]", 4)
111     elif newSensors == 5:
112         sensorsreplace("[0,1,1,1,1,1,0,0,0,0]", 5)
113     elif newSensors == 6:
114         sensorsreplace("[0,1,1,1,1,1,1,0,0,0]", 6)
115     elif newSensors == 7:
116         sensorsreplace("[0,1,1,1,1,1,1,1,0,0]", 7)
117     elif newSensors == 8:
118         sensorsreplace("[0,1,1,1,1,1,1,1,1,1]", 8)
119     else:
120         print"Arvon tulee olla valilta 1 - 8"
121         time.sleep(2)
122         meassensors()
123
124
125 submenu()
126
```

```
settings = dict(
    Bitrate = 16,
    minutes = 15,
    sensors = [0,1,1,0,0,0,0,0,0],
    serial = '00000000609a1234',
    ip = 'device.osoite.com',
    port = 22,
    passWord = 'salasana4321',
    userName = 'device',
    EMAIL_ADDRESS = 'esimerkki@gmail.com',
    PASSWORD = 'salasana1234'
)

```

```
import time
import os
from datetime import datetime
from ADCPi import ADCPi
from subprocess import call
from time import sleep
import config
import smtplib

Bitrate = "Bitrate: "+str(config.settings.get("Bitrate"))
minutes = "Data-tiedoston pituus: "+str(config.settings.get("minutes"))+" min"
sensors = "Anturit: "+str(config.settings.get("sensors"))
serial = "Raspin serial: "+config.settings.get("serial")
ip = "SFTP-palvelimen osoite: "+config.settings.get("ip")
port = "Portti: "+str(config.settings.get("port"))
userName = "Kayttajatunnus: "+config.settings.get("userName")

EMAIL_ADDRESS = config.settings.get("EMAIL_ADDRESS")
PASSWORD = config.settings.get("PASSWORD")

def clear():
    _ = call('clear' if os.name == 'posix' else 'cls')

def replace_line(file_name, line_num, text):
    lines = open(file_name, 'r').readlines()
    lines[line_num] = text
    out = open(file_name, 'w')
    out.writelines(lines)
    out.close()

def send_email(subject, msg):
    print("###Laheta tiedot spostiin###\n\n1 Anna osoite johon haluat lahettaa
    tiedot\n\n Tai \n\n2 Palaa takaisin alkuvalikkoon")
    ipt = raw_input(">>> ")
    if ipt == str("1"):
        try:
            receiver = raw_input("Vastaanottajan osoite: ")
            server = smtplib.SMTP('smtp.gmail.com:587')
            server.ehlo()
            server.starttls()
            server.login(EMAIL_ADDRESS, PASSWORD)
            message = 'Subject: {}\n\n{}'.format(subject, msg)
            server.sendmail(EMAIL_ADDRESS, receiver, message)
            print("Luodaan viestiä, odota hetki...")
            time.sleep(10)
            server.quit()
            print("Onnistui!")
            time.sleep(2)
            os.system('python Menu.py')

        except:
            print("Ei onnistunut :(")
            time.sleep(1)
            os.system('python emailsend.py')

    elif ipt == str("2"):
            os.system('python Menu.py')


subject = "Tiedot"
msg = "%s\n%s\n%s\n%s\n%s\n%s\n%s\n" % (Bitrate, minutes, sensors, serial, ip, port,
userName)

send_email(subject, msg)
```

```
import os
import time
import config
import sys

try:
    import pathlib
except:
    os.system("sudo pip install pathlib")

from pathlib import Path


def getserial(): #fetch unique Raspberry Pi serial number for file naming
  cpuserial = "0000000000000000"
  try:
    f = open('/proc/cpuinfo','r')
    for line in f:
      if line[0:6]=='Serial':
        cpuserial = line[10:26]
    f.close()
  except:
    cpuserial = "ERROR000000000"
  return cpuserial


def replace_line(file_name, line_num, text):
    lines = open(file_name, 'r').readlines()
    lines[line_num] = text
    out = open(file_name, 'w')
    out.writelines(lines)
    out.close()

ser = repr(getserial())
alku = "    serial = "
enter = ",\n"
kaikki = (alku+ser+enter)

replace_line('config.py', 4, kaikki)


os.system("sudo apt-get update")
pathdir = Path("/home/pi/data_log")
ADCpi = Path("/usr/local/lib/python2.7/dist-packages/ADCPi")
watchdog = Path("/sys/devices/platform/soc/3f100000.watchdog/misc/watchdog")
setuptools = Path("/usr/local/lib/python2.7/dist-packages/setuptools")
buildessential = Path("/usr/share/doc/libssl-dev")
cryptography = Path("/usr/local/lib/python2.7/dist-packages/cryptography")
paramiko = Path("/usr/local/lib/python2.7/dist-packages/paramiko")
pysftp = Path("/usr/local/lib/python2.7/dist-packages/pysftp")
pytest = Path("/usr/local/lib/python2.7/dist-packages/pytest_shutil-1.7.0-py2.7.egg")


def makedatadir():

    try:
        pth = ("/home/pi/data_log")
        os.mkdir(pth, 0755);

    except:
        pass

makedatadir()


def INSTALLATION():

```

```
68      try:
69          os.system("sudo apt-get update")
70      except:
71          pass
72
73      try:
74          os.system("sudo python2.7 -m pip install
            git+https://github.com/abelectronicsuk/ABElectronics_Python_Libraries.git")
75      except:
76          pass
77
78      try:
79          os.system("sudo apt-get install watchdog")
80          os.system("sudo update-rc.d watchdog defaults")
81      except:
82          pass
83
84      try:
85          os.system("sudo python -m pip install --upgrade pip setuptools wheel")
86      except:
87          pass
88
89      try:
90          os.system("sudo apt-get install build-essential libssl-dev libffi-dev
            python-dev")
91      except:
92          pass
93
94      try:
95          os.system("sudo pip install cryptography")
96      except:
97          pass
98
99      try:
100         os.system("sudo pip install paramiko")
101     except:
102         pass
103
104     try:
105         os.system("sudo pip install pysftp")
106     except:
107         pass
108
109     try:
110         os.system("sudo easy_install pytest-shutil")
111     except:
112         pass
113
114 INSTALLATION()
115
116
117 def setcrontab():
118
119     HM = '"'
120     HP = "'"
121
122     try:
123         global crn
124         os.system("echo  "+HP+HP+" | crontab -")
125
126         run = ("echo "+HM+"$(crontab -l ; echo  "+HP+"@reboot python
            run.py"+HP+")"+HM+" | crontab -")
127         os.system(run)
128
129         upload = ("echo "+HM+"$(crontab -l ; echo  "+HP+"31 * * * * python
            upload.py"+HP+")"+HM+" | crontab -")
130         os.system(upload)
```

```
131             crn = 1
132
133
134         except:
135             crn = 2
136             pass
137
138     setcrontab()
139
140
141     def CHECK():
142
143         if crn == 1:
144             print ("\nCrontab settings succesfully installed\n")
145         elif crn == 2:
146             print("\nInstalling crontab settings failed\n")
147         else:
148             pass
149
150         if pathdir.is_dir() == True:
151             print ("Data_log directory succesfully created\n")
152         else:
153             print ("Making Data_log directory failed\n")
154
155         if ADCpi.is_dir() == True:
156             print ("ADCpi succesfully installed\n")
157         else:
158             print ("ADCpi installation failed\n")
159
160         if watchdog.is_dir() == True:
161             print ("Watchdog succesfully installed\n")
162         else:
163             print ("Watchdog installation failed\n")
164
165         if setuptools.is_dir() == True:
166             print ("Setuptools succesfully installed\n")
167         else:
168             print ("Setuptools installation failed\n")
169
170         if buildessential.is_dir() == True:
171             print ("Build-essential succesfully installed\n")
172         else:
173             print ("Build-essential installation failed\n")
174
175         if cryptography.is_dir() == True:
176             print ("Cryptography succesfully installed\n")
177         else:
178             print ("Cryptography installation failed\n")
179
180         if paramiko.is_dir() == True:
181             print ("Paramiko succesfully installed\n")
182         else:
183             print ("Paramiko installation failed\n")
184
185         if pysftp.is_dir() == True:
186             print ("Pysftp succesfully installed\n")
187         else:
188             print ("Pysftp installation failed\n")
189
190         if pytest.is_dir() == True:
191             print ("Pytest succesfully installed\n")
192         else:
193             print ("Pytest installation failed\n")
194
195     CHECK()
```

```
import time
import os
from datetime import datetime
from ADCPi import ADCPi
from subprocess import call
from time import sleep
import config

ip = config.settings.get('ip')
port = config.settings.get('port')
userName = config.settings.get('userName')
passWord = config.settings.get('passWord')

def clear():
    _ = call('clear' if os.name == 'posix' else 'cls')

def replace_line(file_name, line_num, text):
    lines = open(file_name, 'r').readlines()
    lines[line_num] = text
    out = open(file_name, 'w')
    out.writelines(lines)
    out.close()

def ipmen():
    clear()
    print('###Yhteysasetukset###\n\n1 Domain: '+ ip +'\n2 Port: '+str(port)+'\n3
    Username: '+ userName +'\n4 Password: '+ passWord)
    print('\nMita asetusta haluat muuttaa?\n\nTai\n\n5 Palaa alkuvalikkoon')
    ipt = raw_input("\n>>> ")

    if ipt == (str("1")):
        newAddress = raw_input("\nAnna uusi osoite: ")
        ipp = repr(newAddress)
        alku = "    ip = "
        enter = ",\n"
        kaikki = (alku+ipp+enter)
        replace_line('config.py', 5, kaikki)
        print('Uusi osoite on nyt: ' + newAddress)
        time.sleep(2)
        os.system('python ipmen.py')

    elif ipt == (str("2")):
        newPort = raw_input("\nAnna uusi portti: ")
        alku = "    port = "
        enter = ",\n"
        kaikki = (alku+newPort+enter)
        replace_line('config.py', 6, kaikki)
        print('Uusi portti on nyt: ' + newPort)
        time.sleep(2)
        os.system('python ipmen.py')

    elif ipt == (str("3")):
        newUser = raw_input("\nAnna uusi kayttaja: ")
        NUser = repr(newUser)
        alku = "    userName = "
        enter = ",\n"
        kaikki = (alku+NUser+enter)
        replace_line('config.py', 8, kaikki)
        print('Uusi kayttaja on nyt: ' + newUser)
        time.sleep(2)
        os.system('python ipmen.py')

    elif ipt == (str("4")):
        newPsWd = raw_input("\nAnna uusi salasana: ")
        NPsWd = repr(newPsWd)
        alku = "    passWord = "
        enter = ",\n"
```

```
67            kaikki = (alku+NPsWd+enter)
68            replace_line('config.py', 7, kaikki)
69            print('Uusi salasana on nyt: ' + newPsWd)
70            time.sleep(2)
71            os.system('python ipmen.py')
72
73        elif ipt == (str("5")):
74            os.system('python Menu.py')
75
76        else:
77            print ("\nVirhe: %s\n") % (str(ipt))
78            ipmen()
79
80    ipmen()
```

```
1   import time
2   import os
3   from datetime import datetime
4   from ADCPi import ADCPi
5   from subprocess import call
6   from time import sleep
7   import config
8
9   Bitrate = config.settings.get('Bitrate')
10  minutes = config.settings.get('minutes')
11  sensors = config.settings.get('sensors')
12
13  def clear():
14      _ = call('clear' if os.name == 'posix' else 'cls')
15
16  def live():
17      clear()
18      print("###Anturin liveluku###\n\n1 Aloita luku\n\nHuom. Paina Ctrl + C,
        keskeyttaaksesi livelukemisen\n\n2 Palaa alkuvalikkoon\n")
19      ipt = raw_input("\n>>>")
20
21      if ipt == (str("1")):
22          clear()#Cleans all stuff above
23          print("Valitsit 1")
24          os.system('python livesensor.py')
25          live()
26
27      elif ipt == (str("2")):
28          os.system('python Menu.py')
29
30      else:
31          print ("\nInvalid input: %s\n") % (str(ipt))
32          live()
33
34
35  live()
```

```
from __future__ import absolute_import, division, print_function, \
                                                    unicode_literals
import time
import os
import config

try:
    from ADCPi import ADCPi
except ImportError:
    print("Failed to import ADCPi from python system path")
    print("Importing from parent folder instead")
    try:
        import sys
        sys.path.append('..')
        from ADCPi import ADCPi
    except ImportError:
        raise ImportError(
            "Failed to import library from parent folder")


def main():
    '''
    Main program function
    '''
    Bitrate = config.settings.get('Bitrate')

    adc = ADCPi(0x68, 0x69, Bitrate)

    while True:

        # clear the console
        os.system('clear')

        # read from adc channels and print to screen
        print("Channel 1: %01i" % adc.read_raw(1))
        print("Channel 2: %01i" % adc.read_raw(2))
        print("Channel 3: %01i" % adc.read_raw(3))
        print("Channel 4: %01i" % adc.read_raw(4))
        print("Channel 5: %01i" % adc.read_raw(5))
        print("Channel 6: %01i" % adc.read_raw(6))
        print("Channel 7: %01i" % adc.read_raw(7))
        print("Channel 8: %01i" % adc.read_raw(8))

        # wait 0.2 seconds before reading the pins again
        time.sleep(0.2)

if __name__ == "__main__":
    main()
```

```
from datetime import datetime
from ADCPi import ADCPi
import time
import os
import socket
import config

### SETUP ###
Bitrate = config.settings.get('Bitrate')
minutes = config.settings.get('minutes')
sn = config.settings.get('sensors')
serial = config.settings.get('serial')
ip = config.settings.get('ip')
port = config.settings.get('port')
userName = config.settings.get('userName')
passWord = config.settings.get('passWord')
sensors = sum(sn)
if Bitrate==(12):
    SPS = int(180)
    BFF = int(60)
elif Bitrate==(14):
    SPS = int(60)
    BFF = int(30)
elif Bitrate==(16):
    SPS = int(14)
    BFF = int (7)
elif Bitrate==(18):
    SPS = int(4)
    BFF = int(1)
### SETUP ###

def meas_temp(): #Pi internal temp for data file
    temp = os.popen ("vcgencmd measure_temp").readline()
    return (temp.replace("temp:",""))

def get_ip(): #IP for data file
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        s.connect(('10.255.255.255', 1))
        IP = s.getsockname()[0]
    except:
        IP = '127.0.0.1'
    finally:
        s.close()
    return IP

def measur():
    adc = ADCPi(0x68, 0x69, Bitrate)
    adc.set_conversion_mode(1)
    mea1=str()
    mea2=str()
    mea3=str()
    mea4=str()
    mea5=str()
    mea6=str()
    mea7=str()
    mea8=str()
    if sn[1]==1:
        mea1=(';%01i' % adc.read_raw(1))
    else:
        mea1=(';N')
    if sn[2]==1:
        mea2=(';%01i' % adc.read_raw(2))
    else:
        mea2=(';N')
    if sn[3]==1:
        mea3=(';%01i' % adc.read_raw(3))
```

```
68          else:
69              mea3=(';N')
70          if sn[4]==1:
71              mea4=(';%01i' % adc.read_raw(4))
72          else:
73              mea4=(';N')
74          if sn[5]==1:
75              mea5=(';%01i' % adc.read_raw(5))
76          else:
77              mea5=(';N')
78          if sn[6]==1:
79              mea6=(';%01i' % adc.read_raw(6))
80          else:
81              mea6=(';N')
82          if sn[7]==1:
83              mea7=(';%01i' % adc.read_raw(7))
84          else:
85              mea7=(';N')
86          if sn[8]==1:
87              mea8=(';%01i' % adc.read_raw(8))
88          else:
89              mea8=(';N')
90          measnn = str()
91          measnn = ("%s%s%s%s%s%s%s%s" % (mea1, mea2, mea3, mea4, mea5, mea6, mea7, mea8))
92          return measnn
93
94      def meas():
95          Fs = int((SPS/sensors)*60*minutes) #Number of samples stored in a file
96          if Fs==0:
97              Fs=1
98          filename = "%s_%s.txt" % (serial, (time.strftime('%Y_%m_%d_%H_%M'))) #naming
            current file
99          f1=open("./data_log/%s" % (filename), 'w+') #opens file for writing
100         measn = str()
101         f1.write(("IP address: %s\n") % (get_ip())) #Print internal temp and IP address to
            data file
102         f1.write(("Pi internal %s") % (meas_temp()))
103         i = 0 #loop reset
104         while i<Fs:
105             t = datetime.now()
106             f1.write("%s" % t) #Print timestamp
107             measn = (measur())
108             f1.write(measn)
109             f1.write("\n")
110             i += (1)
111             print i
112             print measn
113             if  i==Fs-BFF: #Open a separate file in advance to get rid of gap between
                measurements
114                 filename = "%s_%s.txt" % (serial, (time.strftime('%Y_%m_%d_%H_%M')))
115                 f2=open("./data_log/%s" % (filename), 'w+')
116                 i += (1)
117                 print i
118                 print ("new file done")
119                 if Bitrate == (18):
120                     f1.close()
121                     f1 = f2
122                     i += (1)
123                     print ("switched file")
124             if i==Fs-1:
125                 f1.close() #close file for storage
126                 f1 = f2 #switch to preopened file
127                 i += (1)
128                 print("switched file")
129     while True:
130         meas()
131
```

```
1   import time
2   import os
3   from datetime import datetime
4   from ADCPi import ADCPi
5   from subprocess import call
6   from time import sleep
7   import config
8
9
10  def clear():
11      _ = call('clear' if os.name == 'posix' else 'cls')
12
13
14  def menu():
15      clear()
16      print("###Asetusvalikko###\n\n1 Anturin liveluku\n2 Yhteysasetukset\n3 A/D-muunnin
        asetukset\n4 Laheta tiedot spostiin\n5 WiFi asetukset\n\n6 Reboot")
17      ipt = raw_input("\n>>> ")
18      if ipt == (str("1")):
19          os.system('python live.py')
20
21      elif ipt == (str("2")):
22          os.system('python ipmen.py')
23
24      elif ipt == (str("3")):
25          os.system('python ADsettings.py')
26
27      elif ipt == (str("4")):
28          os.system('python emailsend.py')
29
30      elif ipt == (str("5")):
31          os.system('python wifiSettings.py')
32
33      elif ipt == (str("6")):
34          os.system('sudo reboot')
35
36      else:
37          print ("\nInvalid input: %s\n") % (str(ipt))
38          menu()
39
40  menu()
```

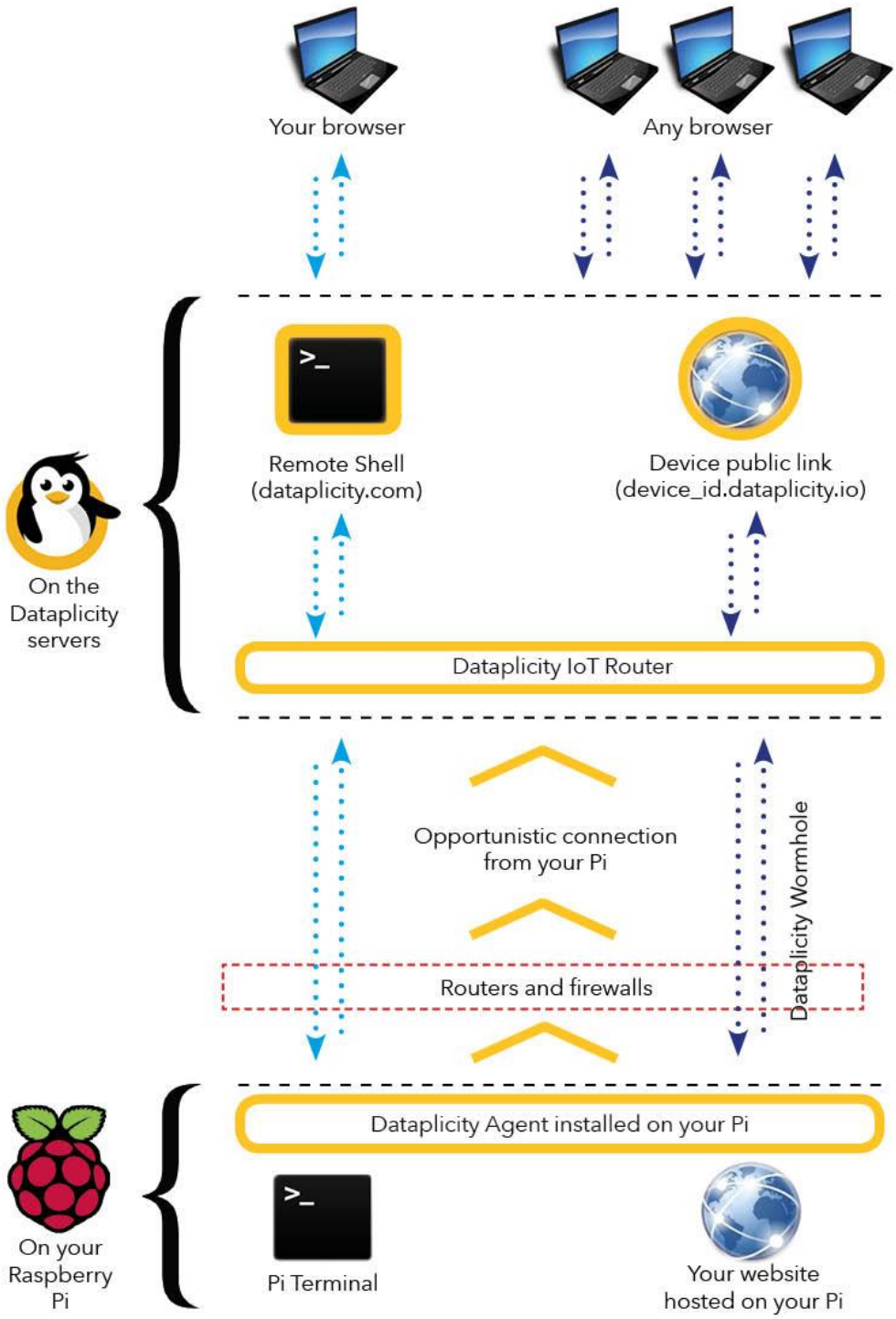
```
import os
import time

def getlinkstate():
    Linkstate = "0000000000000000"
    f = open('/sys/class/net/eth0/operstate','r')
    for line in f:
        Linkstate = line[0:26]
    f.close()
    return Linkstate[0]


def info():
    sana = (getlinkstate())
    if sana=="u":
        print ("Ethernet-cable connected")
        time.sleep(2)
        os.system("python Menu.py")
    else:
        print ("Ethernet-cable disconnected")
        time.sleep(2)
        while True: #Start mreasurement loop. In case of error a new file is created.
            print ("Measuring...\n")
            time.sleep(2)
            os.system("python loop.py")
            print ("Something unexpected occurred. \nSetting up a new file.\n")
            time.sleep(2)

info()
```

```
1   import os
2   import time
3   import config
4
5   minutes = config.settings.get('minutes')
6
7   path = "/home/pi/data_log/"
8   def flushdir(dir):
9       now = time.time()
10      for f in os.listdir(dir):
11          fullpath = os.path.join(dir, f)
12          if os.stat(fullpath).st_mtime < (now - ((minutes*60)+1)): #age of files in
            seconds
13              if os.path.isfile(fullpath):
14                  os.remove(fullpath)
15              elif os.path.isdir(fullpath):
16                  flushdir(fullpath)
17
18  flushdir(path)
```

```
import paramiko
import shutil
import time
import os
import config

Bitrate = config.settings.get('Bitrate')
minutes = config.settings.get('minutes')
sensors = config.settings.get('sensors')
serial = config.settings.get('serial')
ip = config.settings.get('ip')
port = config.settings.get('port')
userName = config.settings.get('userName')
passWord = config.settings.get('passWord')

try:

    os.system('sudo ifconfig wlan0 up')
    os.system('sudo ifconfig eth0 up')

    name = str("data_log")
    filename = "%s_%s_%s" % (name, serial, (time.strftime('%Y_%m_%d_%H')))
    path = str("/home/pi/%s") % (filename)

    shutil.make_archive(path,'zip','/home/pi/data_log')

    fLoad = str("%s.zip") % (path)
    fFile = str("/data/%s.zip") % (filename)

    paramiko.util.log_to_file('/tmp/paramiko.log')

    host = ip
    port = port
    transport = paramiko.Transport((host, port))

    password = passWord
    username = userName

    transport.connect(username = username, password = password)
    sftp = paramiko.SFTPClient.from_transport(transport)
    sftp.put(fLoad, fFile)

    sftp.close()
    transport.close()
    os.remove(fLoad)

    os.system('python sysFlush.py')
except:
    print "error"

```

```
1   import os, sys
2   import time
3
4   wifisettings = ("/etc/wpa_supplicant/wpa_supplicant.conf")
5
6
7   def replace_line(file_name, line_num, text):
8       lines = open(file_name, 'r').readlines()
9       lines[line_num] = text
10      out = open(file_name, 'w')
11      out.writelines(lines)
12      out.close()
13
14
15  def showconnections():
16      f = open(wifisettings,'r')
17      line_num = 0
18      search_phrase = "ssid"
19      for line in f.readlines():
20          line_num += 1
21          if line.find(search_phrase) >= 0:
22              SSID = str(line)
23              opr = (SSID.replace("ssid=", "").replace("\n", "").replace("\t", ""))
24              print opr
25
26
27  def submenu():
28      os.system("clear")
29      print("###WiFi asetukset###\n\n1 Lisaa yhteys\n2 Poista yhteys\n3 Palaa takaisin
        alkuvalikkoon\n\nYhteydet:\n")
30      showconnections()
31      setting = input("\n>>> ")
32      if setting == 1:
33          addnetwork()
34      elif setting == 2:
35          deletenetwork()
36      elif setting == 3:
37          os.system("python Menu.py")
38      else:
39          print"Arvon tulee olla valilta 1 - 4"
40          time.sleep(2)
41          os.system("python wifiSettings.py")
42
43
44  def addnetwork():
45      HM = '"'
46      os.system("clear")
47      os.system("sudo chmod 777 /etc/wpa_supplicant/wpa_supplicant.conf")
48      ssid = raw_input("Anna uusi SSID: ")
49      psword = raw_input("Anna salasana: ")
50      with open(wifisettings, "a") as f1:
51          f1.write("\n\nnetwork={\n\tssid="+HM+ssid+HM+"\n\tpsk="+HM+psword+HM+"\n}")
52      submenu()
53
54  def deletenetwork():
55      os.system("sudo chmod 777 /etc/wpa_supplicant/wpa_supplicant.conf")
56      f = open(wifisettings ,'r')
57      line_num = 0
58      inp = raw_input("Kirjoita yhteyden nimi joka poistetaan: ")
59      search_phrase = inp
60      for line in f.readlines():
61          line_num += 1
62          if line.find(search_phrase) >= 0:
63              first_line = int(line_num) + (-3)
64              replace_line(wifisettings, first_line, "")
65              replace_line(wifisettings, first_line, "")
66              replace_line(wifisettings, first_line, "")
```

```
67                replace_line(wifisettings, first_line, "")
68                replace_line(wifisettings, first_line, "")
69                print"Poistaminen onnistui"
70                time.sleep(2)
71                submenu()
72
73    submenu()
```



(Dataplicity 2019.)