TAMPEREEN
AMMATTIKORKEAKOULU

Talotekniikan IoT

Tiedonkerääminen ja välittäminen

Ville Turunen

Opinnäytetyö
Huhtikuu 2017
Tietotekniikan koulutusohjelma
Tiedonsiirtotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Tietoliikennetekniikka ja tietoverkot

Turunen Ville
Talotekniikan IoT
Tiedonkerääminen ja välittäminen

Opinnäytetyö 29 sivua, joista liitteitä 1 sivua
Huhtikuu 2017

Opinnäytetyön tarkoituksena oli rakentaa laite, jolla voidaan välittää mitattuja signaaleita välittäjänä toimivalle Raspberry Pi-tietokoneelle.

Työssä suunnitellaan ja rakennetaan laitteisto, jonka pohjana on Arduino mikrokontrolleri. Sekä kehitetään Arduinolle ohjelma, joka vastaa serveriltä tuleviin pyyntöihin lähettämällä mitatun arvon MQTT-viestinä.

Asiasanat: arduino, mqtt

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ITC Engineering
Telecommunications and Networks

VILLE TURUNEN:
Building services engineering IoT
Collection and transmission of data

Bachelor's thesis 29 pages, appendices 1 pages
April 2017

The purpose of this thesis was to build a device capable of sending measured signals to a Raspberry Pi MQTT-server.

This thesis consists of two parts. Designing and building a circuit board that can be connected to an Arduino Mega microcontroller with all the needed parts and connections. Writing a program for the microcontroller that responds to query's send by the Raspberry Pi server by sending the measured value as an MQTT-message.

Key words: arduino, mqtt

SISÄLLYS

LYHENTEET JA TERMIT

IoT	Intenet of Things, Esineiden internet.
AD-muunnin	Analogi digitaali muunnin.
I^2C	Inter-Integrated Circuit, Pakettikytkentäinen sarjaliikenne väylä.
SDA	Serial Data Line, I^2C-väylän datasignaali linja.
SCL	Serial Clock Line, I^2C-väylän kellosignaali linja.
SoC	System on Chip, järjestelmäpiiri jossa yhdistyvät useiden eri-piirien toiminnot.
IP-osoite	Internet Protocol, TCP/IP-mallin internet-kerroksen proto-kolla.
DHCP	Dynamic Host Configuration Protocol, Verkkoprotokolla, joka jakaa IP-osoitteita verkkoon kytkeytyneille laitteille.
MQTT	Message Queue Telemetry Transport, Kevyt TCP/IP proto-kollan päällä toimiva viestintä protokolla.

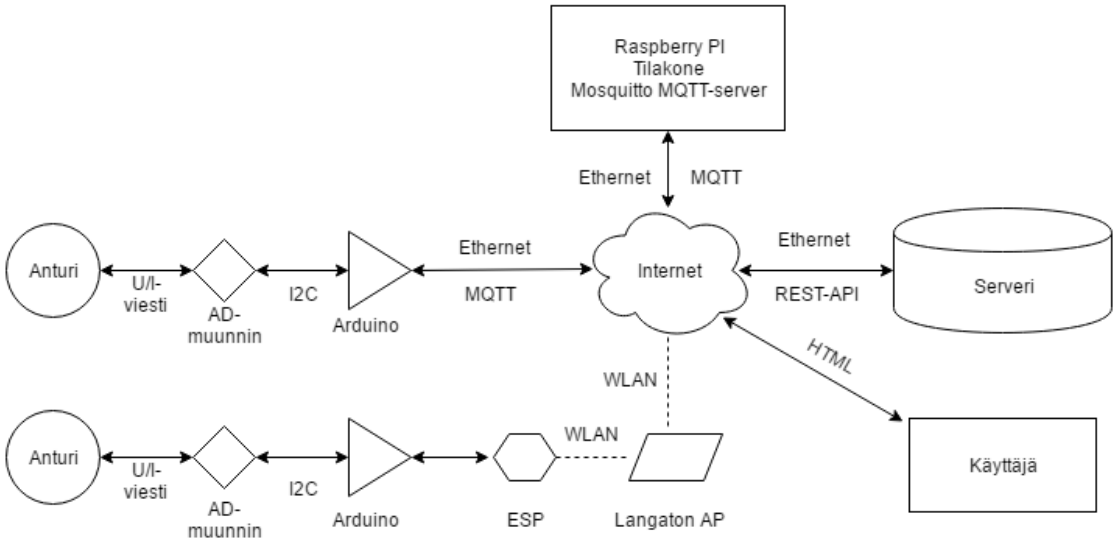
1 JOHDANTO

Talotekniikan IoT-projektin tarkoituksena oli kehittää järjestelmä, jonka avulla voidaan verrata mitattuja arvoja simulointiohjelmalla saatuihin arvoihin. Kehitettävään järjestelmän tuli olla helposti käytettävissä sekä laajennettavissa eri käyttökohteisiin ja mittauksiin. Mittaustulokset siirretään ja tallennetaan pilvipalveluun, josta ne ovat luettavissa reaaliajassa verkkosivun kautta.

Opinnäytetyön tarkoituksena oli rakentaa prototyyppilaite, joka lukee sensorilta tulevaa 0…10V jänniteviestiä ja välittää mittaustuloksen MQTT-serverinä toimivalle Raspberry Pi-tietokoneelle, Rasberry Pi-tietokoneessa pyörivän tilakoneen niitä pyytäessä. Työssä esitellään suunnitellun laitteen elektroniikka ja ohjelmisto toteutus.

2 JÄRJESTELMÄ JA LAITTEET

Järjestelmä päätettiin toteuttaa käyttäen Raspberry Pi yhden piirilevyn tietokonetta, sekä Arduino mikrokontrolleria. Kuvassa 1 on esitetty järjestelmän lohkokaavio.



Kuva 1 Järjestelmän lohkokaavio

Arduinon ja Raspberry Pi:n välinen kommunikointi päätettiin toteuttaa MQTT-protokollalla ethernetin ylitse käyttämällä Raspberry Pi-tietokoneessa pyörivää Mosquitto MQTT-broker ohjelmaa. Anturilta saatava analoginen jänniteviesti, muutetaan digitaaliseen muotoon AD-muuntimen avulla. AD-muunnin on kiinni Arduinossa, ja niiden välinen kommunikointi hoidetaan I^2C-väylän avulla.

2.1 Väylät ja protokollat

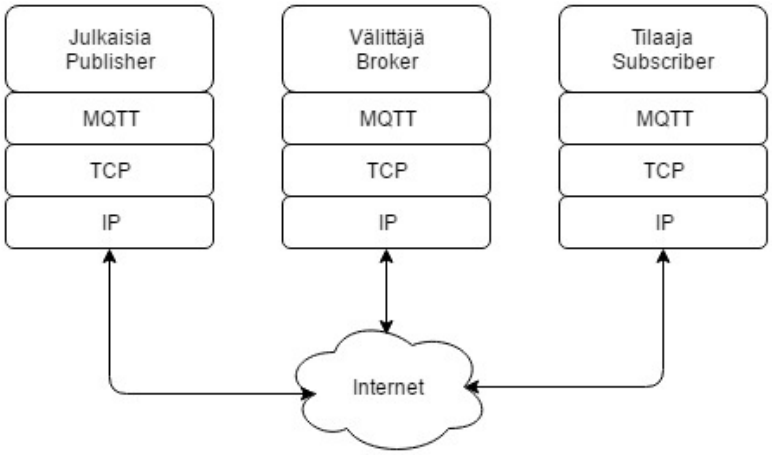
Kommunikointi Raspberry Pi-tietokoneen ja Arduinon välillä hoidetaan käyttämällä MQTT-protokollaa ethernetin ylitse. Raspberry Pi sekä Arduino tulee olla verkossa, jotta yhteys saadaan muodostettua.

2.1.1 I^2C

I^2C (Inter-Integrated Circuit) on Phillips Semiconductorin keksimä multi-master, multi-slave, pakettikytkentäinen sarjaliikenne väylä. Väylää tyypillisesti käytetään mikropiirien ja mikrokontrollereiden väliseen liikenteeseen. I^2C-väylässä käytetään kahta kaksisuuntaista linjaa, Sarja data linja (SDA, Serial Data Line) sekä sarja kello linja (SCL, Serial Clock Line). Työssä I^2C-väylää käytetään AD-muuntimen ja Arduino mikrokontrollerin välisessä kommunikoinnissa, Arduinon wire.h kirjaston avulla.

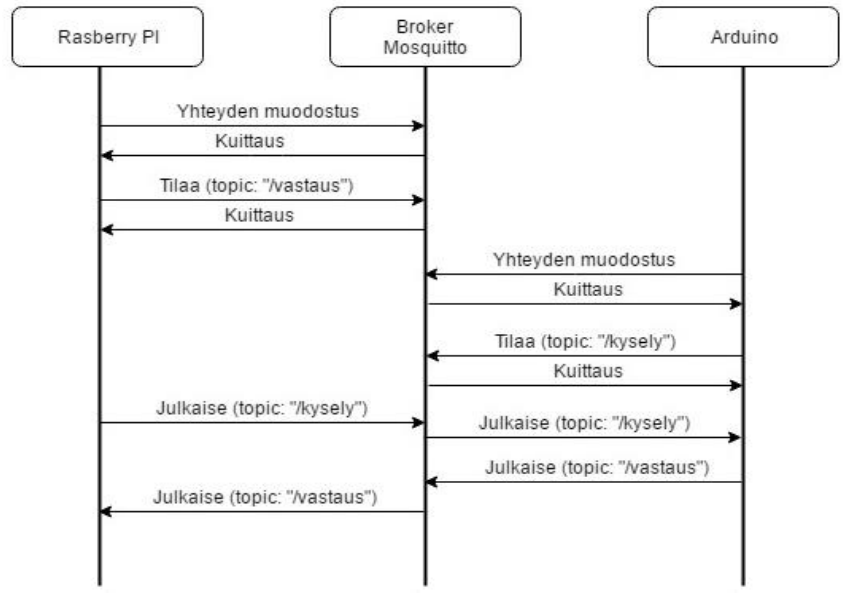
2.1.2 MQTT

MQTT (Message Queue Telemetry Transport) on ISO-standardin (ISO/IEC 20922:2016) mukainen julkaisu-tilaus perusteinen viestintä protokolla, jota toimii TCP/IP-protokollan päällä. Toimiakseen MQTT tarvitsee laitteen, joka toimii viestiverkon välittäjänä (broker). Välittäjä jakaa julkaisijan (publisher) lähettämät viestit tilaajille (subscriber), tilaajien tilaamien aihepiirien (topic) avulla. Kuvassa 2 on esitetty MQTT verkon rakenne



Kuva 2 MQTT viestiverkon rakenne

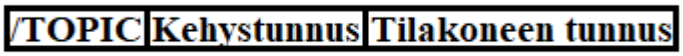
Työssä sekä Arduino, että Raspberry PI toimivat sekä julkaisijana, että tilaajana. Välittäjänä toimii Raspberry PI:ssä pyörivä Mosquitto MQTT-broker ohjelma. Käynnistyessään Arduino ottaa yhteyden Serverinä toimivaan Raspberry PI tietokoneeseen ja tilaa (subscribe) ”*/kysely*” MQTT-topicin. Halutessaan sensori dataa, Raspberry PI lähettää viestin ”*/kysely*”-topic:iin, jota Arduino seuraa. Viestin saapuessa, Arduino tulkitsee viestin ja lähettää vastauksen ”*/vastaus*”-topic:iin, jota vastaavasti Raspberry PI seuraa. Periaate Rasberry PI:n ja Arduinon välisestä MQTT-keskustelusta on esitetty kuvassa 3.



Kuva 3 MQTT-keskustelu

2.1.3 Työssä käytetyn MQTT-viestin rakenne

Työssä käytetyn MQTT-viestin perusrakenne koostuu topic:sta johon viesti lähetetään, kehystunnuksesta ja tilakoneen tunnistenumerosta. Kuvassa numero 4 on esitetty käytetyn MQTT-viestin perusrakenne.

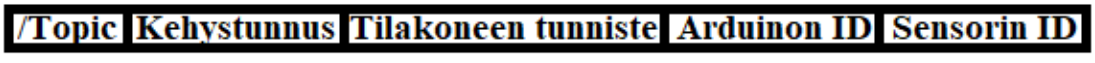


Kuva 4 MQTT-viestin perusrakenne

Käytettyjä aihepiirejä (topic) on kaksi. Topic ”*/kysely*” ja ”*/vastaus*”. Tilakoneen pyytäessään dataa Arduinolta, lähettää Raspberry Pi MQTT-viestin ”*/kysely*” topic:iin ja Arduino lähettää vastauksensa ”*/vastaus*”-topic:iin. MQTT-viestissä topic-kenttää seuraa kehystunnus-kenttä, joka sisältää tiedon halutusta toimenpiteestä. Tällä hetkellä käytössä on vain yksi kehystunnus *01*, joka tarkoittaa pyyntöä lukea dataa sensorilta. Tämä kenttä mahdollistaa tulevaisuudessa eri ominaisuuksien kuten laitteiden ohjaamisen implementoimisen järjestelmään viestin rakennetta muuttamatta. Kehystunnus-kenttää seuraa kenttä, joka pitää sisällään Raspberry Pi:ssä pyörivän tilakoneen tunniste tiedon. Tilakoneen tehtävänä on hoitaa data-kyselyt järjestelmään kytketyiltä sensoreilta. Tilakoneen tunnus-kentän avulla Rasberry Pi osaa yhdistää lähetetyn kyselyn ja saapuneen vastauksen toisiinsa ja merkata kyselyn onnistuneeksi.

2.1.4 Serverin lähettämä MQTT-viesti

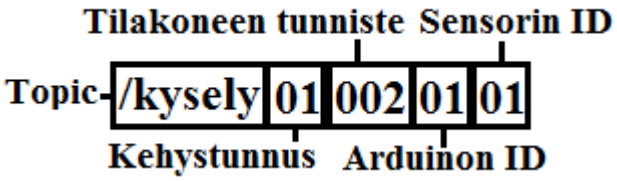
Serverin lähettämä viesti koostuu topic:sta, johon viesti lähetetään sekä yhdeksästä ASCII-merkistä. Kaksi ensimmäistä merkkiä ovat viestin kehystunnus, jolla kerrotaan Arduinolle mitä halutaan tehdä. Kolme seuraavaa merkkiä viestissä ovat Raspberry Pi:ssä pyörivän tilakoneen tietoja, joiden avulla Rasberry Pi osaa yhdistää lähetetyn pyynnön saapuneeseen vastaukseen. Viestin kaksi seuraavaa merkkiä sisältävät Arduinon ID:n jolle lähetetty viesti on tarkoitettu. Näin verkossa voi olla useita Arduino laitteita, mutta vain yksi vastaa viestiin. Viestin kaksi viimeistä merkkiä kertovat sensorin ID:n, jolta Rasberry Pi haluaa saada lukeman. Viestin rakenne on esitetty kuvassa 5.



Kuva 5 Serverin lähettämän MQTT-viestin rakenne

Kuvassa 6 on esitetty topic:iin ”*/kysely*” lähetetty viesti. Kehystunnus *01* tarkoittaa pyyntöä lukea dataa sensorilta. Tilakoneen tunniste on *002*. Tunnisteen avulla Rasberry osaa yhdistää lähetetyn pyynnön ja saapuneen vastauksen. Arduinon ID on *01*, tämän ID:n

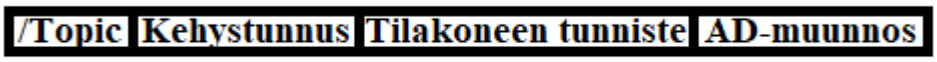
omaava Arduino vastaa tähän viestiin. Sensorin ID on *01* tarkoittaa, että tilakone haluaa tietoja sensorilta, joka on kytketty AD-muuntimen ensimmäiseen sisäänmenoon.



Kuva 6 Serverin lähettämä MQTT-viesti

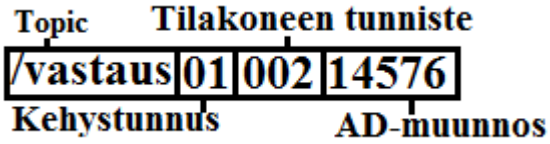
2.1.5 Arduinon lähettämä MQTT-viesti

Arduinon lähettämä vastausviesti koostuu topic:sta, johon viesti lähetetään sekä 6…10 ASCII-merkistä. Viestin pituus riippuu saadusta AD-muunnoksesta. Viestin alussa on topic, johon viesti on lähetetty. Tätä seuraa kaksi merkkiä, jotka ovat viestin kehystunnus. Kolme seuraavaa merkkiä ovat tilakoneelta saatuja tietoja, joilla viesti yksilöidään. Loput merkit ovat AD-muuntimelta saatu mitatun jännitteen AD-muunnos väliltä 0…26667, joka vastaa jännitettä väliltä 0…5V. Viestin rakenne on esitetty kuvassa 7.



Kuva 7 Arduinon lähettämän MQTT-viestin rakenne

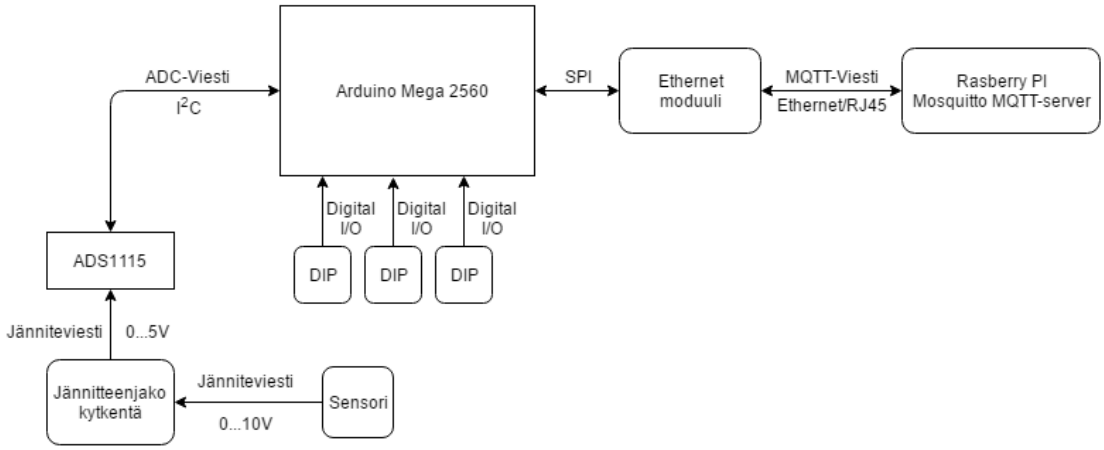
Kuvassa 8 on esitetty topic:iin ”*/vastaus*” lähetetty viesti. Kehystunnus *01* tarkoittaa vastausta datapyyntöön AD-muuntimesta luetulla raakadatalla. Tilakoneen tunniste on 002. Tunnisteen avulla Rasberry osaa yhdistää lähetetyn pyynnön ja saapuneen vastauksen. AD-muunnos kentässä oleva arvo *14576* vastaa luenta hetkellä AD-muuntimen mittaamaa jännitettä.



Kuva 8 Arduinon lähettämän MQTT-viestin rakenne

2.2 Laitteet

Tässä kappaleessa on kerrottu työssä käytetyistä laitteista ja niiden tekniikasta sekä laitteiden implementoinnista järjestelmään. Kuvassa 9 on esitetty rakennetun järjestelmän lohkokaavio sekä rajapinnat eri laitteiden välillä.



Kuva 9 Arduino ja siihen kytketyt laiteet

2.2.1 Rasbperry Pi

Raspberry Pi on yhden levyn tietokone, jonka on kehittänyt Raspberry Pi Foundation. Tietokone perustuu Broadcomin BCM2835-järjestelmäpiiriin (SoC, System-on-Chip), jossa käytetään ARM-prosessorin, muistin sekä integroidun grafiikkapiirin. Tietokone kytketään kiinni monitoriin tai televisioon, syöttölaitteena (input device) voidaan käyttää USB-näppäimistöä ja hiirtä. Järjestelmän massamuistina toimii SD-muistikortti (Secure Digital) ja verkkoyhteydestä vastaa 10/100 ethernet portti (RJ45).

Raspberry Pi:ssä pyörii Mosquitto broker ohjelma, joka toimii käytetyn MQTT-viestiverkon välittäjänä. Mosquiton lisäksi Rasberry Pi:ssä pyörii tilakone ohjelma, joka hoitaa datakyselyt kaikilta järjestelmään kytketyiltä sensoreilta.

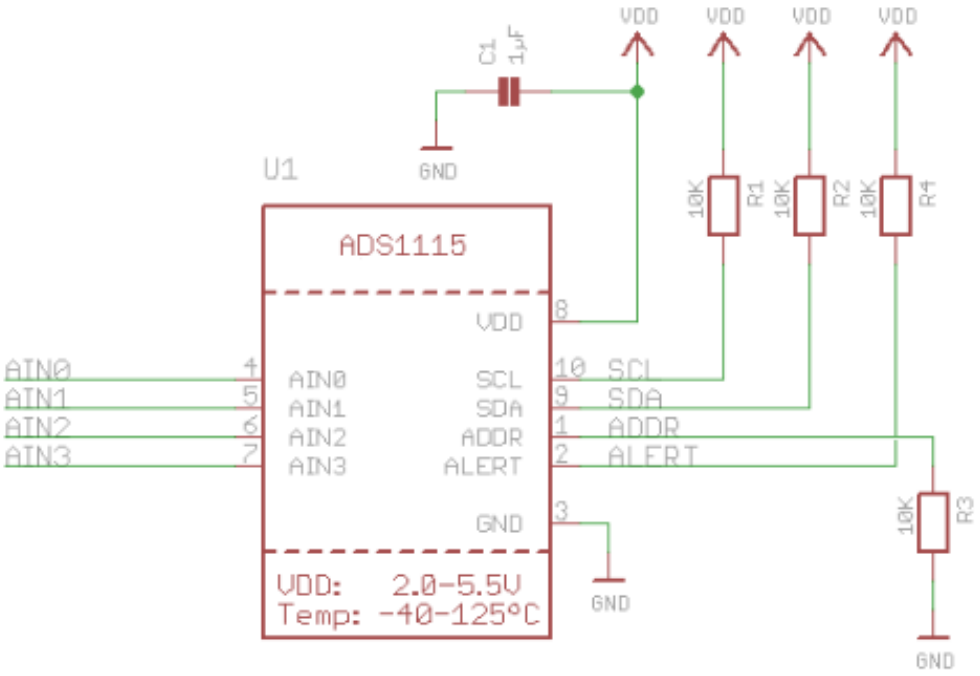
2.2.2 Arduino

Arduino on avoimeen laitteistoon perustava mikrokontrolleri alusta ja ohjelmointi ympäristö. Arduinon laitteisto perustuu 8-bittiseen Atmel AVR mikrokontrolleriin, johon on mahdollista kytkeä erilaisia komponentteja. Laitteistolle kirjoitetaan ohjelmia C++-kieleen perustuvan Arduino-ohjelmointikielen avulla. Ohjelmointi laitteelle tapahtuu USB-portin kautta käyttäen Arduino IDE-kehitysympäristöä. Arduinon ohjelma koostuu kahdesta pääfunktiosta. *Setup()*-funktio, jonka aika alustetaan laitteen asetukset ja *loop()*-funktio, jota Arduino pyörittää virran sammuttamiseen asti.

2.2.3 AD-muunnin

AD-muunnin eli analogia-digitaalimuunnin on laite, joka muuttaa jatkuvan analogisen signaalin kuten jännitteen digitaalisiksi lukuarvoiksi, joita voidaan käsitellä tietokoneen avulla. Työssä AD-muunninta tarvitaan sensorilta saadun jänniteviestin muuttamiseksi digitaaliseen muotoon, joka voidaan lähettää Arduinon avulla ethernetin ylitse palvelimena toimivalle Raspberry Pi-tietokoneelle.

Käytettäväksi AD-muuntimeksi valittiin Adafruitin valmistama AD-muunnin, jossa käytetään Texas Instrumentsin valmistamaa ADS1115 piiriä, joka kykenee 16-bitin erottelu tarkkuuteen. Näin AD-muuntimen resoluutioksi saadaan 2^{16}=65536 eri arvoa. Käytettävässä AD-muuntimessa on myös ohjelmoitavissa oleva vahvistin (PGA, Programmable Gain Amplifier), jolla AD-muunnin voidaan asettaa toimimaan halutulla jännitevälillä. AD-muuntimessa on 4 sisäänmenoa (A0-A3) ja kommunikointi Arduinon kanssa tapahtuu I^2C väylän avulla käyttäen SCL ja SDA pinnejä. AD-muuntimen käyttöjännite saadaan Arduinon +5V nastasta. Käytettyjä AD-muuntimia on mahdollista ketjuttaa 4 peräkkäin, AD-muuntimessa olevan ADDR nastan avulla. Ketjutuksen avulla saadaan sisäänmenojen määrää kasvatettua 16 sisäänmenoon. Kuvassa 10 on esitetty käytetyn AD-muuntimen kytkentäkaavio.



Kuva 10 ADS1115 kytkentäkaavio

2.2.4 Anturit

Järjestelmä suunniteltiin tukemaan mitä tahansa anturia, josta saadaan ulostulo 0…10V jännite viestinä. Valmistetussa prototyypissä anturina käytetään Vaisalan valmistamaa GMW83RP-sensoria, jossa on 3 erillistä anturia. Anturit mittaavat ilmanlämpötilaa

(0…50C), -kosteutta (0-100%) ja ilman CO_2-pitoisuutta (0…2000ppm). Sensorissa on itsessään näyttö, josta voidaan lukea mittaustulokset. Lisäksi anturissa on kolme jännitelähtöä, joista saadaan kunkin anturin mittaustulos jänniteviestillä ulos.

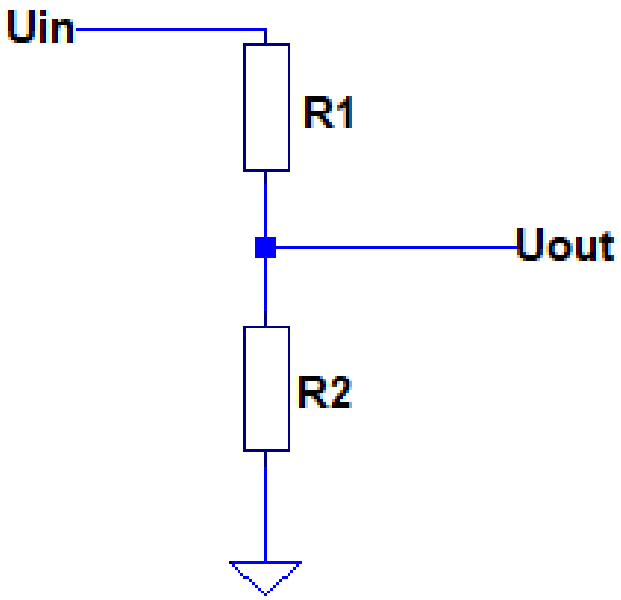
2.2.5 Kytkentä

Sensoreilta tuleva mittasignaali muunnetaan AD-muuntimella digitaaliseen muotoon, jonka Arduino välittää ethernetin ylitse MQTT-palvelimelle. AD-muuntimessa otettiin käyttöön oletus vahvistus, jolloin AD-muuntimen mittausväli asettui ±6,144 volttiin. Koska anturilta saatava analoginen signaali on väliltä 0…10V, tarvitsee signaalille suorittaa jännitteenjako.

Ohmin lain mukaisesti voidaan vastuksen läpi kulkeva virta esittää jännitteen ja resistanssin osamääränä kaavan 1 mukaisesti.

$$U = R * I \rightarrow I = \frac{U}{R} \tag{1}$$

Rakennetaan kuvan 11 mukainen jännitteenjako kytkentä kahdesta vastuksesta.



Kuva 11 Jännitteenjako kytkentä

Ohmin laista tiedetään, että kummankin vastuksen läpikulkeva virta on sama, voidaan virta *I* esittää kaavan 2 avulla.

$$U = R * I \rightarrow I = \frac{U}{R_1 + R_2} \tag{2}$$

Näin voidaan ratkaista vastusten yli olevat jännitteet kaavan 3 avulla.

$$U_{R1} = R_1 \frac{U}{R_1 + R_2} \text{ ja } U_{R2} = R_2 \frac{U}{R_1 + R_2} \quad (3)$$

Joka voidaan ilmoittaa kaavan 4 muodossa.

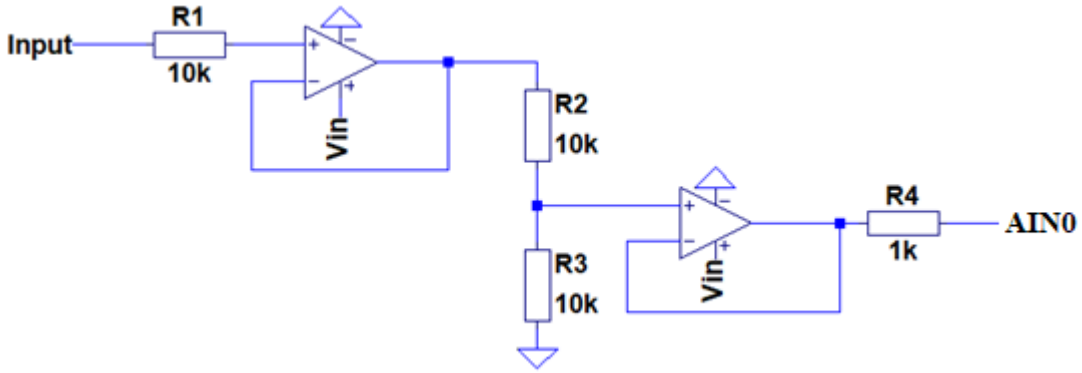
$$U_{R1} = U \frac{R_1}{R_1 + R_2} \text{ ja } U_{R2} = U \frac{R_2}{R_1 + R_2} \quad (4)$$

Kaavan 4 avulla voidaan laskea jännite vastusten yli.

Sensorilta saatavan maksimi jännitteen ollessa 10V ja AD-muuntimelle menevän jännitteen maksimin ollessa 6,144V. Päädyttiin jakamaan sensorilta tuleva jännite puoleen. Käyttämällä kahta saman resistanssin omaavaa vastusta voidaan jakaa sensorilta tuleva jännite puoleen kaavan 5 mukaisesti.

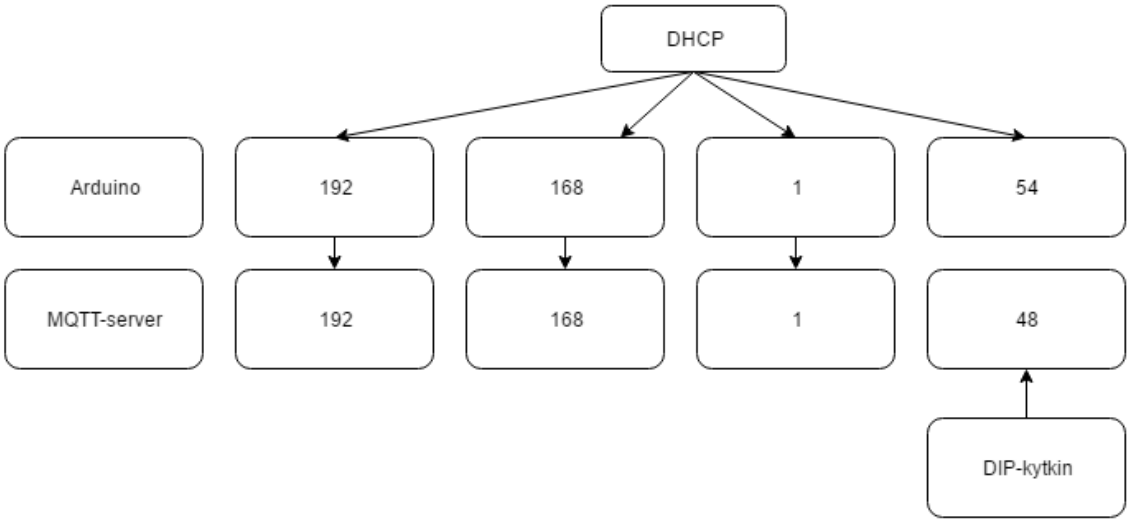
$$U_{R1} = U \frac{R_1}{R_1 + R_2} = 10\text{V} \frac{10\text{k}\Omega}{10\text{k}\Omega + 10\text{k}\Omega} = 10\text{V} \frac{1}{2} = 5\text{V} \quad (5)$$

Jännitteenjaon lisäksi tarvitaan kytkentään kaksi operaatiovahvistinta negatiivisella takaisinkytkennällä. Negatiivisen takaisinkytkennän ansiosta virtaa ei kulje tuloihin, jolloin jännitteen jako ei kuormita AD-muunninta tai Anturia. Operaatiovahvistimeksi valittiin LMC662 operaatiovahvistin. Suunniteltu jännitteenjako kytkentä on esitetty kuvassa 12.



Kuva 12 Suunniteltu jännitteenjako kytkentä

Kuvassa 14 on esitetty rakennetun kytkennän kytkentäkaavio. Kytkennässä on 3 kappaletta 4 bittisiä DIP-kytkimiä. DIP-kytkimellä 1 asetetaan Arduinolle yksilöllinen ID-tunniste väliltä 0…15. Tunnisteen avulla voi samassa verkossa olla useita Arduinoja, joilla kaikilla on yksilöllinen ID. Vertaamalla MQTT-viestistä saatua Arduino ID kentän sisältöä, DIP-kytkimellä asetettuun tunnisteeseen, tiedetään mitä verkossa olevaa Arduino lähetetty viesti koskee.

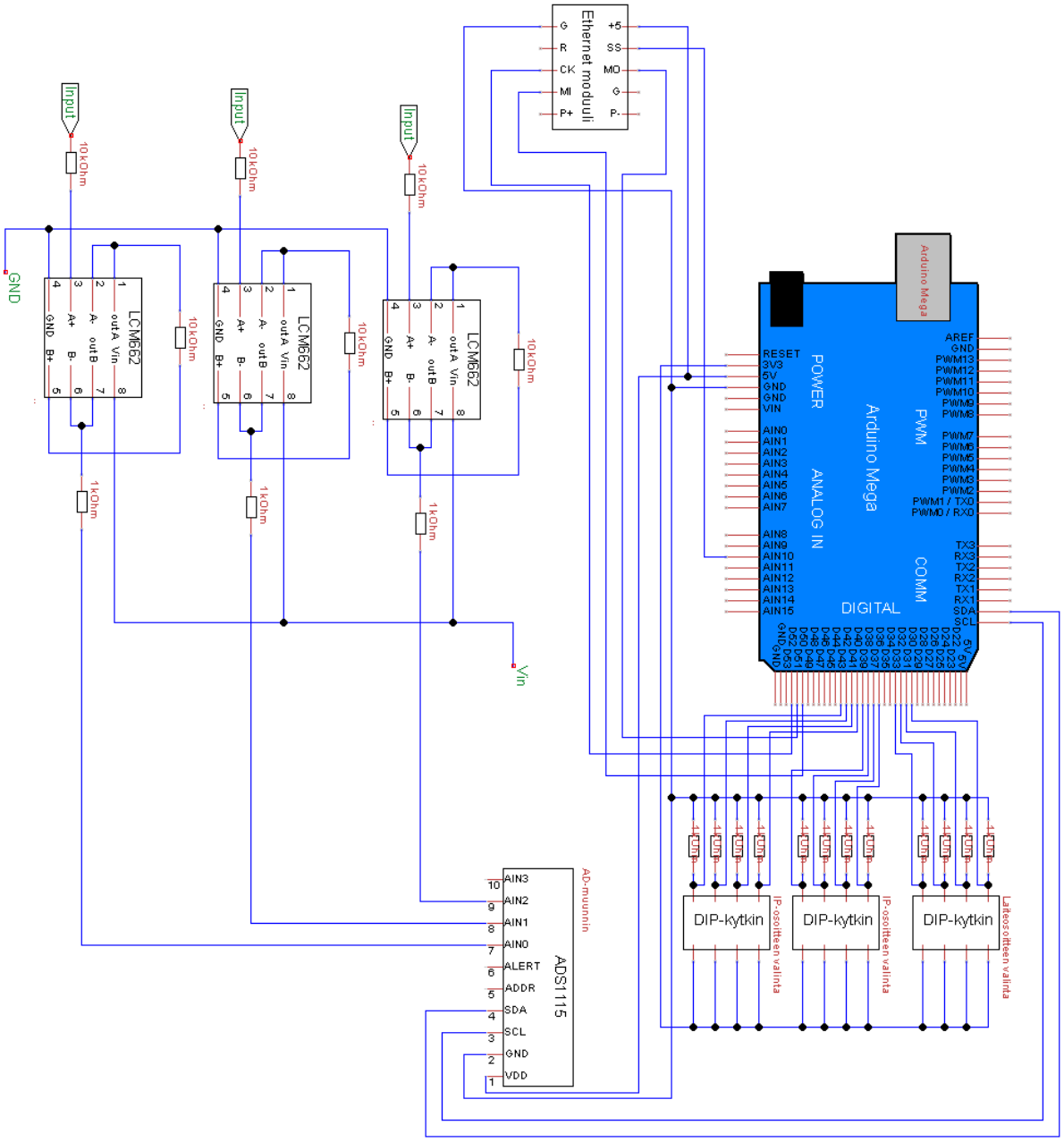


Kuva 13 IP-osoitteen asetus

IP-osoite koostuu neljästä tavusta (tavu = 8 bittiä). Kuvan 13 mukaisesti Arduino saa itselleen IP-osoitteen DHCP-palvelimelta. Osoitteen saamisen jälkeen Arduino asettaa MQTT-serverin IP-osoitteeksi oman IP-osoitteensa. Tämän jälkeen Arduino muuttaa serverin IP-osoitteen viimeisen tavun luvuksi, joka on asetettu DIP-kytkimillä 2 ja 3 (4Bit+4Bit). Arduino tulee olla samassa aliverkossa MQTT-serverin kanssa, jotta IP-osoite voidaan asettaa oikein. Tällä vältytään Arduinon koodin muokkaamiselta laitteen käyttöönoton yhteydessä, eikä verkkoon tarvitse asettaa erillistä kiinteää IP-osoitetta MQTT-serverille.

AD-muunnin kytketään Arduinoon I^2C-väylän avulla käyttäen SCL- ja SDA-linjoja, näiden lisäksi AD-muunnin tarvitsee +5V käyttöjännitteen ja maadoituksen toimiakseen.

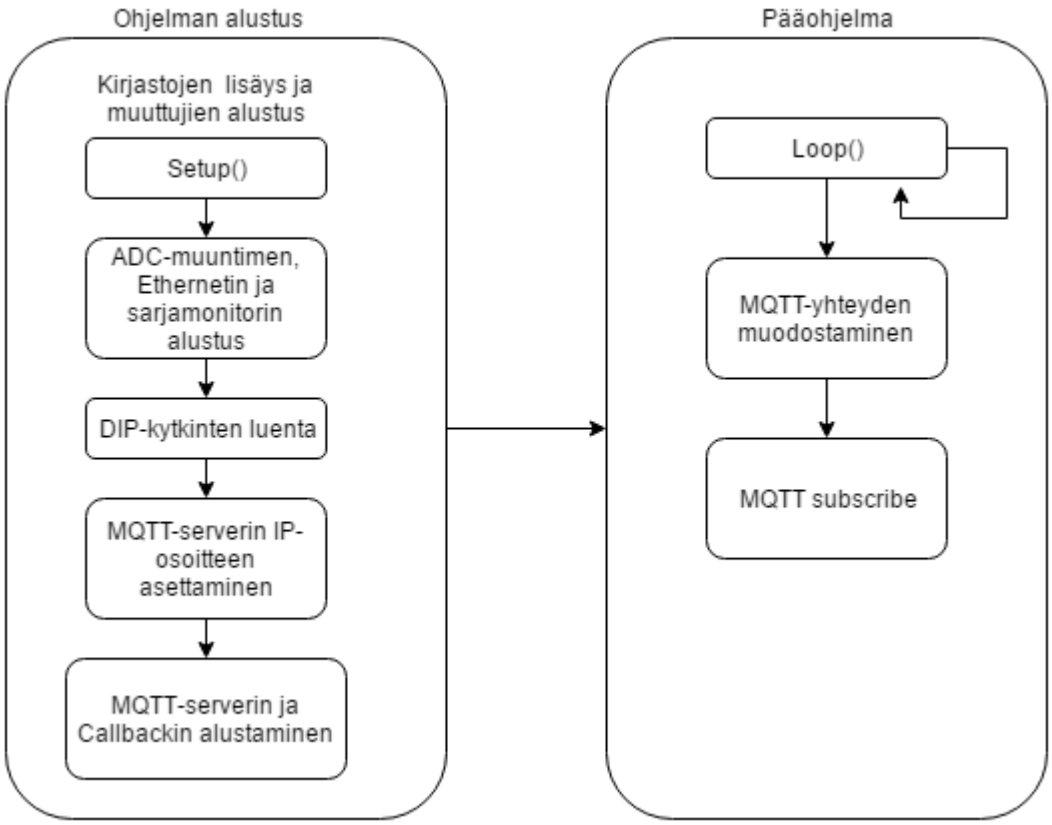
Ethernet yhteys muodostetaan ethernet moduulin avulla. Moduulin SS (slave select) kytketään kiinni Arduinon nastaan numero 10. Moduulin CK (Clock) kytketään kiinni Arduinon nastaan numero 52. Moduulin MO (Master out) kytketään kiinni Arduinon nastaan numero 51. Moduulin MI (Master in) kytketään kiinni Arduinon nastaan numero 50. Näiden lisäksi moduuli tarvitsee +5V käyttöjännitteen ja maadoituksen toimiakseen.



Kuva 14 Kytkentäkaavio

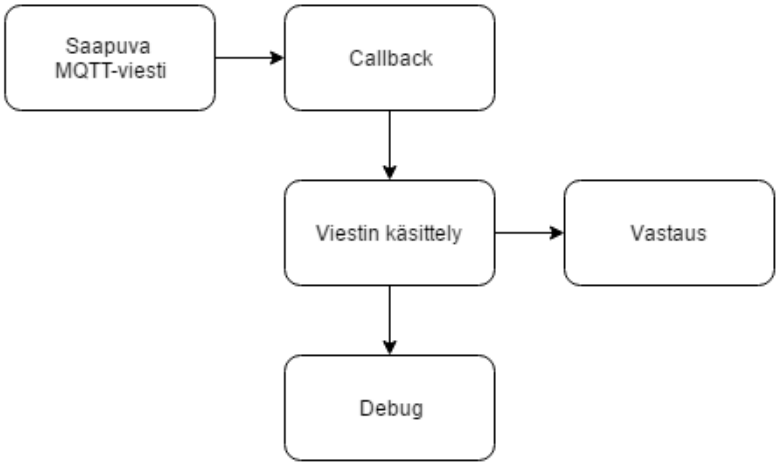
3 ARDUINON OHJELMA

Arduinossa pyörivä ohjelma kirjoitettiin Arduinon omalla IDE-kehitysympäristöllä käyttämällä C++ ohjelmointikieltä. Ohjelman tarkoituksena on muodostaa yhteys Raspberry Pi:ssä pyörivään Mosquitto MQTT broker ohjelmaan. Tilata ja kuunnella MQTT topic:ia ”*/kysely*”. Kuvassa 15 on esitetty ohjelman lohkokaavio.



Kuva 15 Ohjelman lohkokaavio

Ohjelman käynnistyessä alustetaan kaikki tarvittavat muuttujat sekä sisällytetään tarvittavat kirjastot. Tämän jälkeen suoritetaan *setup()*-funktio, jonka sisällä alustetaan AD-muunnin, ethernet sekä sarjamonitori yhteys. Luetaan DIP-kytkinten arvot. Asetetaan MQTT-serverille IP-osoite. Alustetaan MQTT-serveri yhteys sekä *callback*-funktio. Tämän jälkeen käynnistyy ohjelman *main loop*, joka pyörii, kunnes laitteesta katkaistaan virta. *Main loop:ssa* muodostetaan yhteys MQTT-serveriin ja tilataan määritelty MQTT topic.



Kuva 16 Callback-funktion lohkokaavio

Kuvassa 16 on esitetty *callback()*-funktion lohkokaavio. Viestin saapuessa seurattuun topic:iin aktivoituu *callback()*-funktio. *Callback()*-funktio hoitaa saapuneen MQTT-viestin käsittelyn. Käsitellystä viestistä tulkitaan, täytyykö viestiin vastata. Jos viestiin tarvitsee vastata, lähetetään MQTT-viesti topic:iin ”*/vastaus”*. Vastauksen sisältönä on sensorilta luettu data.

3.1 Kirjastot ja muuttujat

Ohjelman toimintaa varten tarvitaan ulkoisia kirjastoja. Kuvassa 17 on esitetty ohjelmaan sisällytetyt kirjastot.

```
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>
```

Kuva 17 Ohjelmassa käytetyt kirjastot

SPI.h (Serial Peripheral Interface) on synkroninen sarjadata protokolla, jota mikrokontrollerit käyttävät kommunikoinnissa lisälaitteiden kanssa. *Ethernet.h* kirjasto sisältää toiminnot ethernet yhteyden luomiseksi. *SPI.h* ja *Ethernet.h* kirjastot vaaditaan, jotta ethernet yhteys saadaan muodostettua Arduinoon kytketyn ethernet moduulin avulla.

PubSubclient.h on kirjasto, jonka avulla mahdollistaa MQTT liikenteen Arduinon ja MQTT-serverin välillä. *Wire.h* kirjasto mahdollistaa I^2C-protokollan käytön Arduinolla. *Adafruit_ADS1015.h* kirjasto tarvitaan AD-muuntimen käyttämisen Arduinolla.

Integer-tyyppisiä globaaleja muuttujia tarvittiin 6 kappaletta. Kuvassa 18 on esitetty käytetyt kokonaislukumuuttujat.

```
int ardIDint;   //Arduinon ID
int sensorIDint;  //Sensorin ID
int laite = 0;  //DIP-kytkimellä asetettu ID
int ADarvo = 0; //AD-muuntimen lukema
int kehystunnus;  //Kehystunnus
int osoite = 0;   //Serverin IP-osoitteen viimeineen 3 viimeisetä numeroa
```

Kuva 18 Käytetyt kokonaislukumuuttujat

Muuttuja *ardIDint* on kokonaisluku, joka kertoo miltä Arduinolta serveri haluaa saada tietoja. Muuttuja *sensorIDint* on kokonaisluku, joka kertoo miltä sensorilta serveri haluaa tietoja. Muuttuja *laite* on kokonaisluku, joka määräytyy DIP-kytkimen asennosta. DIP-kytkimellä määritetään Arduinolle uniikki ID. *ADarvo* muuttuja on kokonaisluku, johon on luettu AD-muutimelta saatu arvo väliltä 0…26667, joka vastaa jännitettä väliltä 0…5V. Muuttuja *kehystunnus* on kokonaisluku, joka saadaan saapuneesta MQTT-viestistä. Muuttuja *osoite* on kokonaisluku, joka on kahdella DIP-kytkimellä asetettu 8-bittinen luku, joka kertoo Arduinolle serverin IP-osoitteen loppuosan.

Byte tyyppisiä muuttujia on kolme kappaletta. Muuttujat on esitetty kuvassa 19

```
byte DIPid;
byte DIPip;
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xFE }; //Arduinon MAC-osoite
```

Kuva 19 Käytetyt byte-muuttujat

DIPid muuttumaan luetaan laitteen id:n määräävän DIP-kytkimen arvo. *DIPip* muuttujan luetaan serverin IP:n loppuosan määräävien DIP-kytkinten arvot. *mac* muuttujalle annetaan Arduinon mac-osoite.

Näiden muuttujien lisäksi koodissa määritellään IP-osoitteeksi muuttuja *server*, johon on tallennettu serverin IP-osoite. Ethernet clientiksi muuttuja *ethClient* sekä MQTT kirjaston PubSubclientin nimeksi *client*. Koodin alussa myös määritetään, että käytettävä AD-muunnin on 16-bittinen versio ja nimetään se *ads*:ksi. Nämä on esitetty kuvassa 20.

```
IPAddress server;
EthernetClient ethClient;
PubSubClient client(ethClient);
Adafruit_ADS1115 ads; //16bittinen adc
```

Kuva 20 Muut käytetyt muuttujat

3.2 Setup

Setup() kohdan alussa alustetaan AD-muunnin, ethernet yhteys ja sarjamonitori yhteys. Luetaan DIP-kytkinten asennot. Asetetaan MQTT-serverin IP-osoite sekä alustetaan MQTT-serveri yhteys ja *callback*-funktio.

3.2.1 Alustukset

Kuvasta 20 nähdään, kuinka käytettävä AD-muunnin nimettiin *ads*:ksi. Komennolla ads.begin() alustetaan AD-muunnin käyttövalmiiksi. *Ethernet.begin()* funktio saa parametrikseen *mac* muuttujan johon on asetettu Arduinon MAC-osoite. *Serial.begin()* alustaa sarjamonitori yhteyden ja saa parametrikseen käytettävän baudinopeuden, joka on tässä tauksessa 57600. Sarjamonitoria käytetään ohjelman debug toiminnossa. Alustukset on esitetty kuvassa 21.

```
void setup()
{
  //ADC:n alustus
  ads.begin();
  //Ethernetin alustus
  Ethernet.begin(mac);
  //Serialin alustus
  Serial.begin(57600);
```

Kuva 21 Alustus

3.2.2 DIP-kytkinten luenta

Alustusten jälkeen luetaan käyttäjän asettamat arvot DIP-kytkimistä. Luettujen arvojen avulla asetetaan laitteelle ID sekä MQTT-serverin IP-osoitteen viimeinen tavu kuvan 13 mukaisesti.

```
for(int i = 0; i < 4; i++)
   {
     DIP = DIP | (digitalRead(i+30) << i); // tehdään bitin siirto, johon on luettuna dippikytkimen asento.
   }
 laite = DIP;
 DIP =0;
 //Serverin alustus
 for(int i = 0; i < 8; i++)
   {
     Dstate = Dstate | (digitalRead(i+36) << i); // tehdään bitin siirto, johon on luettuna dippikytkimen asento.
   }
 osoite = Dstate;
```

Kuva 22 DIP-kytkinten luenta

Kuvassa 22 esitetyllä *digitalRead()*-funktiolla luetaan Arduinon digitaalinastan tila. Tilan arvoksi saadaan 0 tai 1, riippuen DIP-kytkimen asennosta. Parametriksi *digitalRead()*-funktio saa luettavan digitaalinastan numeron. Laitteen ID:n määräävä DIP-kytkin on kytketty Arduinon digitaalinastoihin 30,31,32 ja 33 kuvan 14 mukaisesti. Luettujen nastojen arvot asetetaan muuttujan *DIPid* arvoksi. For-silmukan jälkeen on muuttujassa *DIPid* 4-bittinen luku, joka asetetaan muuttujan *laite* arvoksi, tämä arvo on sama kuin Arduinon ID. Sama operaatio tehdään muuttujalle *DIPip*, jonka avulla asetetaan serverin IP-osoite.

3.2.3 MQTT-serverin IP-osoitteen asettaminen

DIP-kytkinten luennan jälkeen asetetaan MQTT-serverille IP-osoite. Asetetaan muuttujan *sever* arvoksi Arduinon lokaali IP-osoite. Muuttuja *server* on taulukkomuuttuja, joka koostuu neljästä alkiosta (0…3). Kuvan 14 mukaisesti saa muuttujan 3 (0…2) ensimmäistä alkiota arvonsa Arduinon IP-osoitteesta. Viimeinen alkio (3) asetetaan DIP-kytkimillä luetun muuttujan *osoite* avulla kuvan 23 mukaisesti.

```
IPAddress server = Ethernet.localIP();
server[3] = osoite;
Serial.print("Asetettu serverin ip soite on ");
Serial.println(server);
Serial.print("Lokaali ip soite on ");
Serial.println(Ethernet.localIP());
```

Kuva 23 IP-osoitteen asettaminen ja tulostus

IP-osoitteen asettamisen jälkeen, tulostaa Arduino sarjamonitoriin serverin IP-osoitteen ja Arduinon lokaalin IP-osoitteen ongelmanratkaisua varten.

3.2.4 MQTT-serverin ja callback:n alustus

MQTT-serverin IP-osoitteen asettamisen jälkeen alustetaan käytetty MQTT-serveri yhteys sekä *callback*-funktio kuvan 24 mukaisesti.

```
//serverin alustus
client.setServer(server, 1883);
//callbackin alustus
client.setCallback(callback);
```

Kuva 24 Serveri yhteyden ja callbackin alustus

Funktio *client.setServer()* saa parametrikseen muuttujan *server*, jossa on serverin IP-osoite sekä MQTT:n käyttämän TCP/IP portin numeron 1883. Funktio *client.setCallback()* saa parametrikseen *callback*:n, joka on pointteri *callback* funktioon, jota kutsutaan MQTT-viestin saapuessa seurattuun topic:iin.

3.3 Main loop

Ohjelman main loop on esitetty kuvassa 25.

```
//Ohjelma
void loop()
{

  delay(10);
  if (!client.connected())
  {
    reconnect();
  }

  client.loop();
}
```

Kuva 25 Main loop

Ohjelman main loop, pitää sisällään yhteyden muodostamisen, sekä *client.loop()*-funktion, jonka avulla kuunnellaan tilattua MQTT-topic:ia.

3.3.1 Yhteyden muodostaminen ja MQTT-subscribe

Kun yhteyttä ei ole muodostettu tai se katkeaa kesken ohjelman, aktivoituu *reconnect()*-funktio. *Reconnect()*-funktio on esitetty kuvassa 26.

```
void reconnect() {
  //Kun ei ole yhteyttä
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Yritys yhdistää
    //Clientin nimi
    if (client.connect(laite)) {
      Serial.println("connected");
      // Kun saatu yhteys, lähetetään ilmoitus
      //TOPIC/VIESTI
      char viesti[100];
      sprintf(viesti, "Mega %0d online", laite);
      client.publish("vastaus", viesti);
      //Tilaus tarvittaviin topikkeihin
      client.subscribe("kysely");

      //Jos yhteys ei onnistu
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Delay uudelle yritykselle
      delay(5000);
    }
  }
}
```

Kuva 26 Yhteyden muodostaminen

Yhteyden muodostus käynnistyy, kun yhteys MQTT-serveriin ei ole aktiivinen. Arduinon sarjamonitoriin tulostuu viesti, joka kertoo yrityksestä muodostaa MQTT yhteys. Jos yhteyden muodostus onnistuu, tulostetaan sarjamonitoriin viesti yhteyden muodostumisesta. Viestin tulostamisen lisäksi lähetetään MQTT-viesti topic:iin ”*/vastaus*” funktiolla *client.publish()*. Viestin sisältönä ”Mega %0d online”, jossa %0d vastaa *laite* muuttujaa, jonka arvona on DIP-kytkimillä luettu Arduinon ID tunniste. MQTT-viestin lähettämisen jälkeen, Arduino tilaa topic:in ”*/kysely”* funktiolla *client.subscribe(”kysely”)*. Jos yhteyden muodostus ei onnistu, tulostaa Arduino sarjamonitoriin *client.state()*-funktion tilan, jonka avulla saadaan selville miksi yhteyden muodostus ei onnistunut. Kuvassa 27 on esitetty ei tilat, joita *client.state()*-funktio voi saada.

- -4 : MQTT_CONNECTION_TIMEOUT - the server didn't respond within the keepalive time
- -3 : MQTT_CONNECTION_LOST - the network connection was broken
- -2 : MQTT_CONNECT_FAILED - the network connection failed
- -1 : MQTT_DISCONNECTED - the client is disconnected cleanly
- 0 : MQTT_CONNECTED - the cient is connected
- 1 : MQTT_CONNECT_BAD_PROTOCOL - the server doesn't support the requested version of MQTT
- 2 : MQTT_CONNECT_BAD_CLIENT_ID - the server rejected the client identifier
- 3 : MQTT_CONNECT_UNAVAILABLE - the server was unable to accept the connection
- 4 : MQTT_CONNECT_BAD_CREDENTIALS - the username/password were rejected
- 5 : MQTT_CONNECT_UNAUTHORIZED - the client was not authorized to connect

Kuva 27 client.state-funktion tilat

3.4 Callback

Viestin saapuessa Arduinon seuraamaan MQTT-topic:iin aktivoituu kuvassa 28 esitetty *callback()*-funktio.

```
void callback(char* topic, byte* payload, unsigned int length)
```

Kuva 28 Callback-funktio

Callback-funktio saa parametreikseen *topic*:n johon MQTT-viesti saapui, *payload*:in joka on saapuneen viestin sisältö sekä *length*:n joka on saapuneen *payload* viestin pituus.

3.4.1 Viestin käsittely

MQTT-viestin saapumisen jälkeen tulkitaan saapunut viesti kuvassa 5 esitetyn viestin rakenteen avulla. Kuvassa 29 on esitetty muuttujien *ardIDint*, *sensorIDint* ja *kehystunnus* asettaminen saapuneen viestin avulla. Sekä char-taulukon *tempp* luominen.

```
ardIDint = ((payload[5]-48)*10)+(payload[6]-48);
sensorIDint = ((payload[7]-48)*10)+(payload[8]-48);
kehystunnus = ((payload[0]-48)*10)+(payload[1]-48);
char tempp[5];
for(int i =0; i<5; i++)
  {
    tempp[i]=payload[i]-48;
  }
```

Kuva 29 Viestin tulkinta

Asetetaan muuttujan *ardIDint* arvoksi *payload* matriisissa paikoissa 5 ja 6 sijaitsevat merkit. Merkit muunnetaan ACSII merkeiksi vähentämällä arvosta 48. Kertomalla merkki paikassa numero 5 kymmenellä ja lisäämällä siihen merkki paikassa numero 6, saadaan muodostettua kokonaisluku väliltä 00...99, joka vastaa Arduinolle asetettua ID-numeroa. Muuttujille *sensorIDint* ja *kehystunnus* tehdään sama käsittely, jonka avulla ne muutetaan kokonaisluvuiksi.

Seuraavaksi luodaan char-tauluko *tempp*, johon tallennetaan *payload*:n 5 ensimmäistä merkkiä ACSII numeroina. Kuvassa 5 esitetystä MQTT-viestin rakenteesta tiedetään, että

payloadin 5 ensimmäistä merkkiä pitivät sisällään saapuneen viestin kehystunnuksen sekä tilakoneen tunnisteen, jotka halutaan lähettää muokkaamattomina takaisin vastausviestissä.

3.4.2 Vastaus

Jos saapuneen viestin kehystunnus oli 01 ja viestistä tulkittu *ardIDint* muuttuja vastaa DIP-kytkimestä luettua *laite* muuttujaa, luetaan tiedot pyydetyltä sensorilta ja lähetetään vastausviesti kuvan 30 mukaisesti.

```
if (kehystunnus == 01)
{
  if (laite == ardIDint)
  {
    //ADC kutsu
    ADarvo = ads.readADC_SingleEnded(sensorIDint - 1);
    char viesti[100];
    sprintf(viesti, "%d%d%d%d%d%d", tempp[0], tempp[1], tempp[2], tempp[3], tempp[4], ADarvo );
    client.publish("vastaus", viesti);
  }
}
```

Kuva 30 Vastaus data pyyntöön

Muuttujan *ADarvo* arvo saadaan AD-muuntimelta funktion *ads.readADC_SingleEnded()* avulla. Funktio saa parametrikseen kokonaisluvun, joka vastaa AD-muuntimen sisäänmenoa. Tämä parametri on sama kuin serverin lähettämästä viestistä tulkittu muuttuja *sensorIDint*, josta täytyy vähentää luku 1, koska ensimmäinen sisäänmeno on 0. Näin serverin pyytäessä tietoa anturilta numero 1 luetaan AD-muuntimen sisäänmeno 0.
Näiden muuttujien lisäksi viestiin lisätään AD-muuntimelta luettu muuttuja *ADarvo*. Näiden parametrien avulla funktio *sprintf()* rakentaa char muuttujaan *viesti*, kuvan 7 mukaisen MQTT-viestin. Rakennettu viesti julkaistaan funktion *client.publish()* avulla, joka saa parametrikseen topic:n johon viesti lähetetään sekä *sprintf()*-funktiolla rakennetun viestin ”*viesti”*.

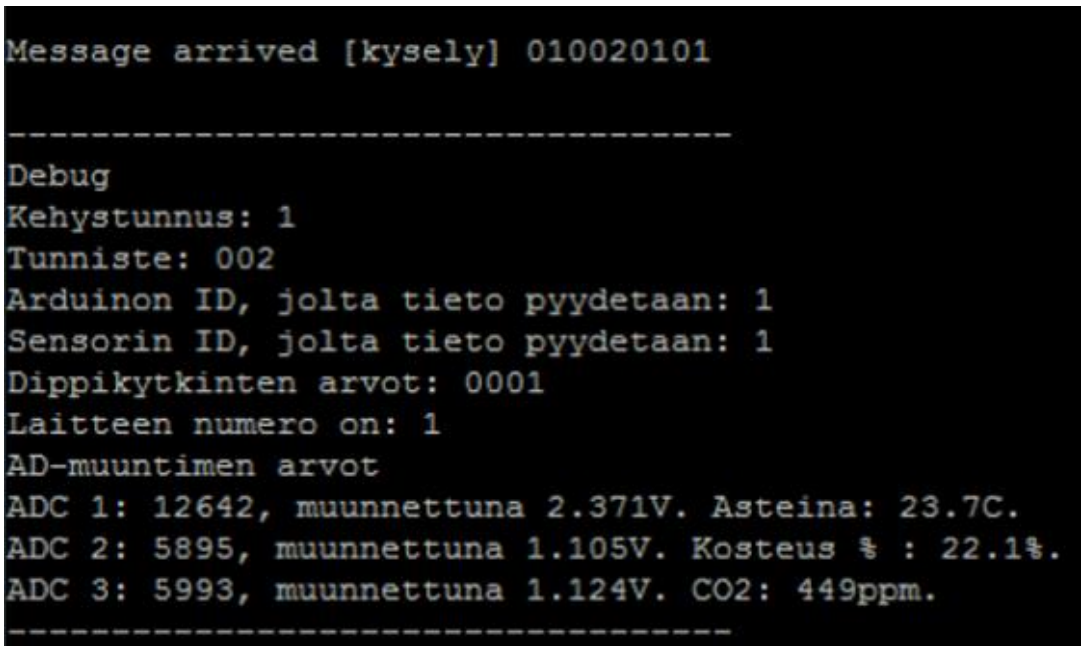
3.4.3 Debug

Ohjelman debug osiota ongelman ratkaisussa, sekä ohjelman toiminnallisuuden esittelyssä. Viestin saavuttua seurattuun MQTT-topic:iin tulostaa Arduino sarjamonitoriin viestin tiedot, riippumatta siitä tarvitseeko viestiin vastata vai ei. Näin voidaan ongelma ratkoa mahdollisia yhteys tai laite ongelmia. Kuvassa 31 on esitetty debug osion käsittely Arduinon koodissa.

```
Serial.println("");
Serial.println("-----------------------------------");
Serial.println("Debug");
Serial.print("Kehystunnus: ");
Serial.println(kehystunnus);
Serial.print("Tunniste: ");
for (int i = 2; i < 5; i++)
{
  Serial.print((char)payload[i]);
}
Serial.println("");
Serial.print("Arduinon ID, jolta tieto pyydetaan: ");
Serial.print(ardIDint);
Serial.println("");
Serial.print("Sensorin ID, jolta tieto pyydetaan: ");
Serial.print(sensorIDint);
Serial.println("");
Serial.print("Dippikytkinten arvot: ");
for (int i = 33; i > 29; i--)
{
  Serial.print(digitalRead(i)); //tulostetaan seriaalin dippien asennot
}
Serial.println("");
Serial.print("Laitteen numero on: ");
Serial.print(laite);
Serial.println("");
Serial.println("AD-muuntimen arvot");
Serial.print("ADC 1: ");
Serial.print(ads.readADC_SingleEnded(0));
Serial.print(", muunnettuna ");
Serial.print(ads.readADC_SingleEnded(0)*kerroin, 3);
Serial.print("V. ");
Serial.print("Asteina: ");
Serial.print(ads.readADC_SingleEnded(0)*kerroin * 10, 1);
Serial.println("C.");
Serial.print("ADC 2: ");
Serial.print(ads.readADC_SingleEnded(1));
Serial.print(", muunnettuna ");
Serial.print(ads.readADC_SingleEnded(1)*kerroin, 3);
Serial.print("V. ");
Serial.print("Kosteus % : ");
Serial.print(((ads.readADC_SingleEnded(1)*kerroin) / 5) * 100, 1);
Serial.println("%.");
Serial.print("ADC 3: ");
Serial.print(ads.readADC_SingleEnded(2));
Serial.print(", muunnettuna ");
Serial.print(ads.readADC_SingleEnded(2)*kerroin, 3);
Serial.print("V. ");
Serial.print("CO2: ");
Serial.print(((ads.readADC_SingleEnded(2)*kerroin) / 5) * 2000, 0);
Serial.println("ppm.");
Serial.println("-----------------------------------");
Serial.println("");
```

Kuva 31 Debug osio

Debug osiossa tulostetaan sarjamonitoriin saapuneesta MQTT-viestistä tulkitut tiedot, laitteen tunnisteen määräävien DIP-kytkinten asennot sekä AD-muuntimen mittaamat lukemat AD-muunnoksena, jännitteenä kuin mittasuureiksi muutettuina yksikköinäkin. Kuvassa 32 on esitetty sarjamonitori näkymä viestin saapumisen jälkeen.



Kuva 32 Sarjamonitoriin tulostettu debug näkymä

4 YHTEENVETO

Työn alkuperäisen idean mukaan saatiin suunniteltua ja rakennettua toimiva tiedonkeräysjärjestelmä, johon voidaan kytkeä kiinni mikä tahansa anturi josta saadaan mittaustulos ulos 0…10V jänniteviestinä. Prototyyppilaitteessa käytettiin 16-bittistä AD-muunninta, jonka koko mittausresoluutiota ei voitu ottaa käyttöön. Täysi 16-bitin alue olisi käytettävissä jännite välillä ±6,144V mitattavan alueen ollessa 0…5V. Mitattu AD-muunnos välitetään eteenpäin MQTT-viestinä, serverinä toimivalle Raspberry Pi-tietokoneelle. Serverin sitä pyytäessä. Laitteen käyttöönottamiseksi ei käyttäjän tarvitse tehdä muuta kuin siirtää valmis ohjelma mikrokontrolleriin ja asettaa DIP-kytkimellä serverin IP-osoitteen viimeinen tavu ja laitteelle yksilöllinen laite ID. Näin ei loppukäyttäjän tarvitse muokata mikrokontrollerissa pyörivää koodia.

Jatkokehitysideoita syntyi työn edetessä. Arduino Mega mikrokontrollerin korvaaminen huomattavasti pienemmällä Arduino Nano mikrokontrollerilla, jolloin laite voitaisiin integroida suoraan mittalaitteen koteloon. Langallinen ethernet yhteys voitaisiin korvata langattomalla Wlan yhteydellä, jolloin laite tarvitsisi vain virtajohdon. Kytkennän muokkaaminen sellaiseksi, että laitteeseen voitaisiin kytkeä kiinni mitä tahansa väylää käyttävä sensori, eikä pelkästään 0…10V jännitesignaalia lähettävää.

LÄHTEET

Evans, B. 2007 Arduino Programming Notebook. Yhdysvallat: Creative Commons

Getting Started with Arduino and Genuino products. Luettu 30.1 https://www.arduino.cc/en/Guide/HomePage

GMW80 Series Carbon Dioxide, Humidy and Temperature Transmitters for DCV, Datasheet, Luettu 20.2 http://www.vaisala.com/Vaisala%20Documents/Brochures%20and%20Datasheets/CEN-G-GMW80-Series-Datasheet-B211435EN.pdf

ADS111x Ultra-Small, Low-Power, I 2C-Compatible, 860-SPS, 16-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator, Datasheet. Luettu 20.12 http://www.ti.com/lit/ds/symlink/ads1115.pdf

Arduino Client for MQTT, Luettu 15.1 http://pubsubclient.knolleary.net/api.html

LIITTEET

Liite 1. Kytkentäkaavio