VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mietola Jouni

LANGATON AKUSTON VALVONTA

TEKNIIKKA JA LIIKENNE
2013

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Jouni Mietola
Opinnäytetyön nimi	Langaton akuston valvonta
Vuosi	2013
Kieli	suomi
Sivumäärä	38
Ohjaaja	Jukka Matila

Vaspec Oy:n toimeksiannosta työssä on tutkittu langattoman teknologian hyödyntämistä sähköaseman apusähköjärjestelmissä. Apusähköjärjestelmissä on kaksi osaa, vaihtosähköjärjestelmä, sekä tasasähköjärjestelmä. Työssä keskitytään tasasähköjärjestelmän toimintavarmuuden lisäämiseen, lisäämällä valvontaa.

Työssäni toimii langattomana teknologiana ANT+. ANT-piirien tukena toimii Texas Instrumentsin mikrokontrollerit.

ANT sopii tasasähköjärjestelmän vara-akuston valvomiseen. Työssä käytetty jaettukanava riittää helposti useammankin noodin liittämiseen monen akun tapauksessa. Työssä rakennettiin yksi isäntänoodi, sekä kaksi orjanoodia, jotka toimivat moitteettomasti.

Avainsanat	ANT, sähköasema, langaton, apusähköjärjestelmä (4–5 avainsanaa)

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Jouni Mietola
Title	Wireless Battery Monitoring
Year	2011
Language	Finnish
Pages	38
Name of Supervisor	Jukka Matila

The thesis is made by commission for Vaspec Oy. It is made to study the wireless monitoring possibilities in electricity station. Backup system consist two parts. Alternating current backup and direct current backup system. In this thesis the focus is on direct current backup system.

The wireless technology is ANT+. There is microcontroller with the ANT chip.

ANT can be used with electricity station's backup system. The Shared Channel has been used in this thesis and the data is broadcast data. One master node is made by using the Antware software and two slave nodes with special interface connection.

Keywords ANT, wireless, Electricity station

LYHENNELUETTELO

GPRS	General Packet Radio Service Pakettikytkentäinen tiedonsiirtopalvelu
USB	Universal Serial Bus Sarjaväyläarkkitehtuuri
MSP	Mixed Signal Microcontroller Mikrokontrolleri
A/D	Analog to Digital Analogia digitaali
RISC	Reduced Instruction Set Computer Suoritinarkkitehtuuri
SPI	Serial Peripheral Interface Tiedonvälitysstandardi
LED	Light Emitting Diode Hohtodiodi
RF	Radio Frequency Radiotaajuus
FET	Field Effect Transistor Transistori
SCLK	Serial Clock Väyläkello
MCU	Microcontroller Mikrokontrolleri

SISÄLLYS

Tiivistelmä

ABSTRACT

1 JOHDANTO 7

2 TASASÄHKÖJÄRJESTELMÄ 8

2.1 Ongelman Kuvaus 8

2.2 Työn tavoite ja rajaus 8

3 LAITTEISTO 10

3.1 ANT 10

3.2 MSP430 10

3.2.1 MSP430 kehitysalusta 11

4 ANT-PROTOKOLLA 12

4.1 Kanava tyypit 12

4.2 Tietotyypit 13

4.2.1 Broadcast Data 13

4.2.2 Purske(Burst) 17

4.3 Itsenäiset kanavat 18

4.3.1 Yksittäisen ANT-kanavan salaus 18

4.3.2 Jaettukanava 19

4.3.3 Jatkuvaskannaus 22

4.4 Laitteiden paritus 23

5 PROTOTYYPIN KASAAMINEN 24

5.1 Käyttöönotto ja konsepti 24

5.2 Työn eteneminen 25

5.2.1 Kokoonpano 25

5.2.2 Synkroninen viestintä bitti virta kontrollilla 32

5.3 Jatkokehitys 34

6 PÄÄSOVELLUS 35

6.1 MSP430-isäntäkontrollerin sovellus 35

6.1.1 A/D-muunnos 37

6.2 Antware 38

7 TESTAUS 40
8 YHTEENVETO 41
LÄHTEET 42

1 JOHDANTO

Työn tilaajana on toiminut Vaspec Oy. Vaspec Oy on vuonna 2011 perustettu kasvuyritys, jonka toimiala on sähkötekninen konsultointi ja suunnittelu sekä valmistus. Työ on tärkeä tulevaisuuden älykästä sähköverkkoa ajatellen. Sovelluksella tai sen variaatiolla saattaa olla käyttöä tässä etenevässä teknologiassa. Yleisesti ajatellen työ on tärkeää nimenomaan sähköverkkojen kehityksen kannalta. Tulevaisuudessa sähköverkko on älykkäämpi ja entistäkin automatisoidumpi kuin tänä päivänä. Pääperiaatteena tällä työllä on valottaa langattoman tiedonsiirron mahdollisuuksia älyverkon kehityksessä. Tässä nimenomaisessa tapauksessa keskitytään sähköaseman vara-akuston kunnonvalvonnan kehittämiseen. Tavoitteena on saada aikaan mahdollisimman helposti kohteeseen kytkettävä kevyt sensori. Varsinainen lopputuote jää tämän työn ulkopuolelle.

2 TASASÄHKÖJÄRJESTELMÄ

2.1 Ongelman Kuvaus

Sähköaseman apusähköjärjestelmään kuuluu kaksi osiota, vaihtosähköjärjestelmä, sekä tasasähköjärjestelmä. Tasasähköjärjestelmään kuuluu sähköaseman ohjaus-,suojaus- ja kaukokäyttöjärjestelmät. Mikäli sähköasema jäisi sähköttä siitä tulisi hengenvaarallinen ja riski suurille tuhoilla kasvaa. Tasasähkökeskuksiin kytketyt akustot ovat yleensä 110V tai 220V. Akustot kytketään sarjaan ja ne koostuvat yleensä lyijyakuista. /1/ Työssä keskitytään ainoastaan tasasähköjärjestelmään. Tavoitteena on langattoman valvonnan avulla saada luotettavampi apusähköjärjestelmä. Ongelma on saada valvottua akkuja vaivattomasti. Uusimmat, ja varsinkin suljetut geeliakut vaativat myös valvontaa ja huoltoa, joka osaltaan lisää tämän työn merkitystä. Käytännössä kaappiin sijoitettavien akustojen kanssa langaton akkukohtainen valvonta ei ole mahdollista suljetun metallirakenteen vuoksi. Ongelma on sähköaseman lattialle sijoitettujen lyjyakustojen jännitteiden valvonta. Ongelman ratkaisemiseksi tarvitaan helposti asennettava valvontayksikkö, joka valvoo akuston jännitettä, ja purkaa akkua, jolloin jännite asettuu muiden akkujen jännitteiden tasolle.

Työssä tarkoituksena on tutkia langattoman tiedonsiirron mahdollisuuksia sähköaseman apusähköjärjestelmän käyttövarmuuden lisäämiseksi.

2.2 Työn tavoite ja rajaus

Tavoitteena on luoda pohja järjestelmälle jolla valvotaan akkukohtaisesti jännitettä, ja tieto lähetetään langattomasti eteenpäin käsiteltäväksi. Työ esittelee ANT-verkon mahdollisuuksia tähän tarkoitukseen. Mikäli jännitetaso poikkeaa muiden akkujen napajännitteestä aloitetaan akun purkaminen. Tällöin akun rinnalle kytketään fet-transistorilla purkuvastus. Kun jännite on muiden akkujen tasolla fet-transistori kytketään pois päältä. Mikäli akuston jännitetaso putoaa liian alas, välitetään viesti isäntänoodille, joka vie tiedon järjestelmän keskusyksikölle. Työssä keskitytään antureiden väliseen tiedonsiirtoon ja sovellustason toimintoihin, kuten mainitut jännitetason seuranta ja ohjaaminen automaattisesti.

Se miten hälytys akuston jännitetason laskiessa liian alas lähtee keskusyksiköltä eteenpäin jää tämän työn ulkopuolelle. Yleensä vastaavanlaisissa sovelluksissa käytetään gprs-modeemia viestin saattamiseksi muihin järjestelmiin. Tämä työ ei ota kantaa järjestelmä integraatioon. Mahdollinen akuston kunnonvalvonta muilta osin kuin jännitteen passiivisen seurannan osalta jää myös työn ulkopuolelle. Työn lopputuloksena on karkea prototyyppi kyseisen langattoman protokollan soveltuvuudesta nykyisiin ja tulevaisuuden älyverkko sovelluksiin. Mahdollisen jatkokehityksen kannalta myös tietoturva tulee ottaa erityisen huomion kohteeksi. Työssä ei esitetä eikä testata mitään erityisiä tietoturvaan liittyviä seikkoja. Tietoturvasta vastaa ANT-verkon oma matalantason tietoturva kerros. Prototyypissä on akuston jännitteen sijaan käytössä MSP430 prosessorin sisäisen lämpötila-anturin analogia-signaali.

3 LAITTEISTO

3.1 ANT

ANT/ANT+ ANT Wireless, on Dynastream Innovations Inc divisioonan kehittämä langatonprotokolla. Dynastream Innovations Inc valmistaa lähinnä urheilu ja terveyssovelluksiin tarkoitettuja langattomia piirejä. Vuonna 2012 ST Ericsson käytti ANT+:aa omissa projekteissaan. Samana vuonna myös Android USB alkoi tukemaan ANT+:aa. ANT-piirejä ei voi ohjelmoida suoraan. On käytettävä ulkoista mikropiiriä, vaikkakin joitain toimintoja voidaan toteuttaa Senscore scriptin avulla.

ANT-piirit on helposti muokattavissa käyttämään langattoman verkon kanavia ulkoisen mikrokontrollerin avulla. Senscore teknologia vie ANT-teknologiaa askeleen pidemmälle, senscore sriptillä voidaan ottaa ANT-senscorea tukevasta piiristä käyttöön analogiatuloja sisääntuloja, ja/tai digitaali sisääntuloja ja ulostuloja, eliminoiden näin tarpeen käyttää ulkoista mikrokontrolleria. Työssä ei käytetä SensCore skriptiä, vaan kaikki tapahtuu aina yhteistyössä ulkoisen mikrokontrollerin kanssa. Poikkeuksena toki isäntänoodi, joka käyttää tietokoneen prosessoria ja ohjelmistoja(Antware).

ANT-kommunikaatio tapahtuu 2,4Ghz taajuudella. Yhdellä itsenäisellä kanavalla tai mahdollisesti voidaan käyttää useampia eri kanavia ja taajuusasetuksia.

3.2 MSP430

Kuva1. MSP430-piiri /6/

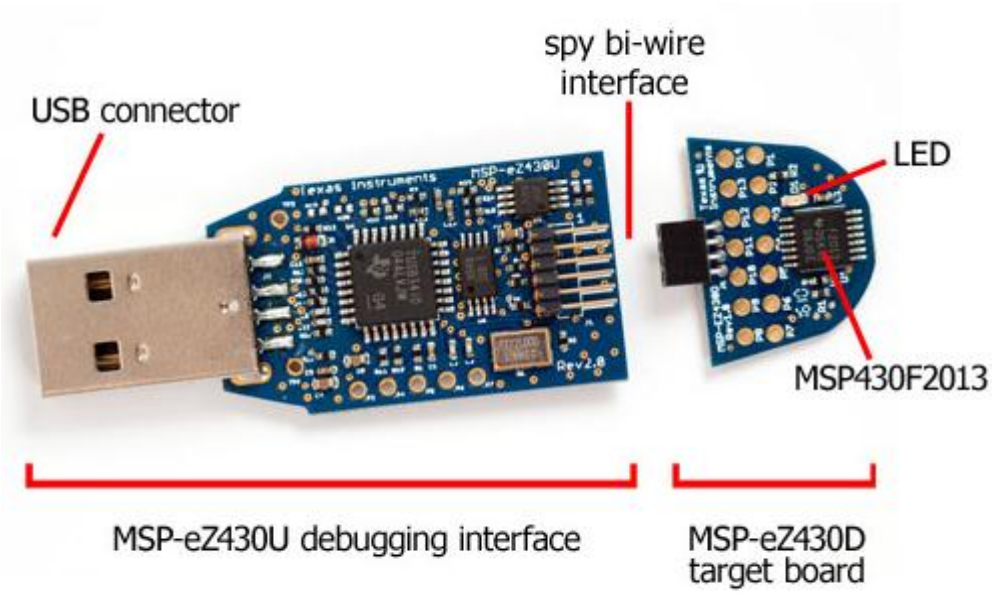
Ulkoiseksi mikrokontrolleriksi valittiin käytettäväksi MSP430F20x3 piiri. Kääntäjäksi ja kehitysympäristöksi valittiin IAR Embedded workbench.

Texas Instruments MSP430 matalatehoinen mikrokontrolleri. 16-bittinen RISC prosessori, 16-bit rekisterillä.

Digitaalisesti kontrolloitu oskillaattori (DCO) mahdollistaa herätyksen matala-teho tilasta aktiiviseen tilaan alle mikrosekunnissa. Sisäänrakennettu 16-bittinen ajastin.

Sisäänrakennettu valmius käyttää synkronisoituja protokollia (SPI or I2C) ja a 10-bittinen A/D-muunnin, tai a 16-bittinen sigma-delta A/D-muunnin.

3.2.1 MSP430 kehitysalusta



Kuva 2. MSP430 kehitysalusta /6/

4 ANT-PROTOKOLLA

ANT ™ on matala tehoinen langaton protokolla 2.4Ghz:in kaistalla. Suunniteltu erityisesti erittäin matala tehoisten järjestelmien käyttöön ja se tukee ’peer-to-peer’, ’star’, ’tree’ and ’practical mesh’ topologioita. Työssä käytetään star topologiaa, koska anturit eivät ainoastaan kerää tietoa ja lähetä sitä isäntänoodille, vaan joutuvat myös vastaanottamaan käskyjä tältä. ANT soveltuu erittäin hyvin halpojen matalateho prosessorien käyttöön ja auttaa täten pitämään tuotantokustannukset alhaisina. Oikein ohjelmoituna ANT-protokollalla tehty itsenäinen anturi voi kestää vuosia yhdellä patterilla. Työssä keskitytään sovellustasolle, koska ANT-protokolla sisältää alemmat tasot.

Kommunikaatio laitteiden välillä toimii eritavoin. Kanava tyyppistä, kanavan asetuksista ja data tyypistä, myös mihin suuntaan dataa lähetetään. Liikenne voi olla joko yksi tai kaksisuuntaista. Liikenne tapahtuu ’full-duplex’ periaatteella, mikäli liikenne on kumpaankin suuntaan. Suurin osa sovelluksista käyttää synkronista, itsenäistä ja kaksisuuntaista kanavaa. Kun isäntänoodi aukaisee synkronisen kanavan, se avaa ensin etsintäikkunan tarkistaakseen, että sen lähetykset ei häiritse toista laitetta. Sitten se lähettää viestit halutulla kanavaperiodilla. Kun kanava on kerran avattu, isäntälaite lähettää aina viestin, joka aikaikkunassa. Käytettäessä kaksisuuntaisia kanavia isäntälaite pitää radiovastaanoton päällä lyhyen aikaa kun se on lähettänyt viestin. Tästä syystä orja lähettää dataa isännälle takaisin heti kun orjanoodi on saanut viestin.

4.1 Kanava tyypit

Kanavalla on merkittävä rooli. Voidaan luoda itsenäisiäkanavia tai käyttää jaettuakanavaa. Työssä päädyttiin käyttämään jaettuakanavaa. Käytössä on jaettukanava asetus. Jaettu kaksisuuntainen kanava-asetus laajentaa vakio kaksisuuntaista kanavatyyppiä. Tämä sopii hyvin, koska tällöin yksittäinen ANT-noodi vastaanottaa, ja mahdollisesti käsittelee dataa monelta eri noodilta. Tässä tapauksessa akkukohtaisista antureista(noodeista). Moni noodi jakaa yksittäisen itsenäisen kanavan kommunikoidakseen isäntänoodin kanssa.

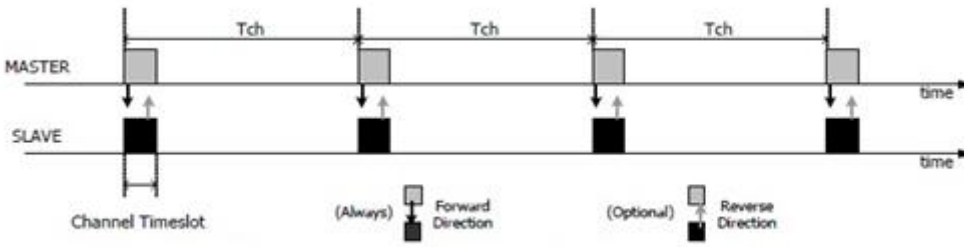
Isäntänoodi lähettää orjanoodille ’broadcast dataa’, jonka orjat vastaanottavat, ja lähettävät takaisin 'broadcast data' muotoista dataa isännälle. Mikäli viestiliikenne ei ole hyväksytty tyyppistä, niin laitteet eivät tiedä onko viesti mennyt perille. Tietosisältö on kahdeksan tavua/paketti.

4.2 Tietotyypit

ANT tukee neljää tietotyyppiä broadcast, acknowledged(hyväksytty), burst(purske) ja advanced data burst data. Tieto lähetetään kahdeksan tavun paketeissa RF-kanavan yli. Tietotyyppi ei ole kanava-asettelu parametri ja kaksisuuntainen ANT-kanava ei ole rajoitettu yhteen tietotyyppiin. Kaikkia neljää tietotyyppiä voidaan lähettää eteenpäin tai eteenpäin/takaisin kanavan aikaikkunassa. Ainoa rajoitus on yksisuuntainen kanava, joka voi lähettää ainoastaan broadcast muotoista dataa eteenpäin.

4.2.1 Broadcast Data

Broadcast data on yleisin tietotyyppi ja se on vakioasetuksena. Broadcast data lähetetään isäntä kanavalta orjalle jokaisen kanava aikaikkunan aikaan. Broadcast data lähetetään orjalta isännälle vain toiseen suuntaan mikäli orjan mikrokontrolleri sitä erikseen pyytää. Vakioasetuksena dataa ei lähetetä ilman pyyntöä. Työssä tilanne on juuri kuvaillun kaltainen, eli mikrokontrolleri pyytä ANT:ia lähettämään broadcast muotoista dataa isäntänoodille.

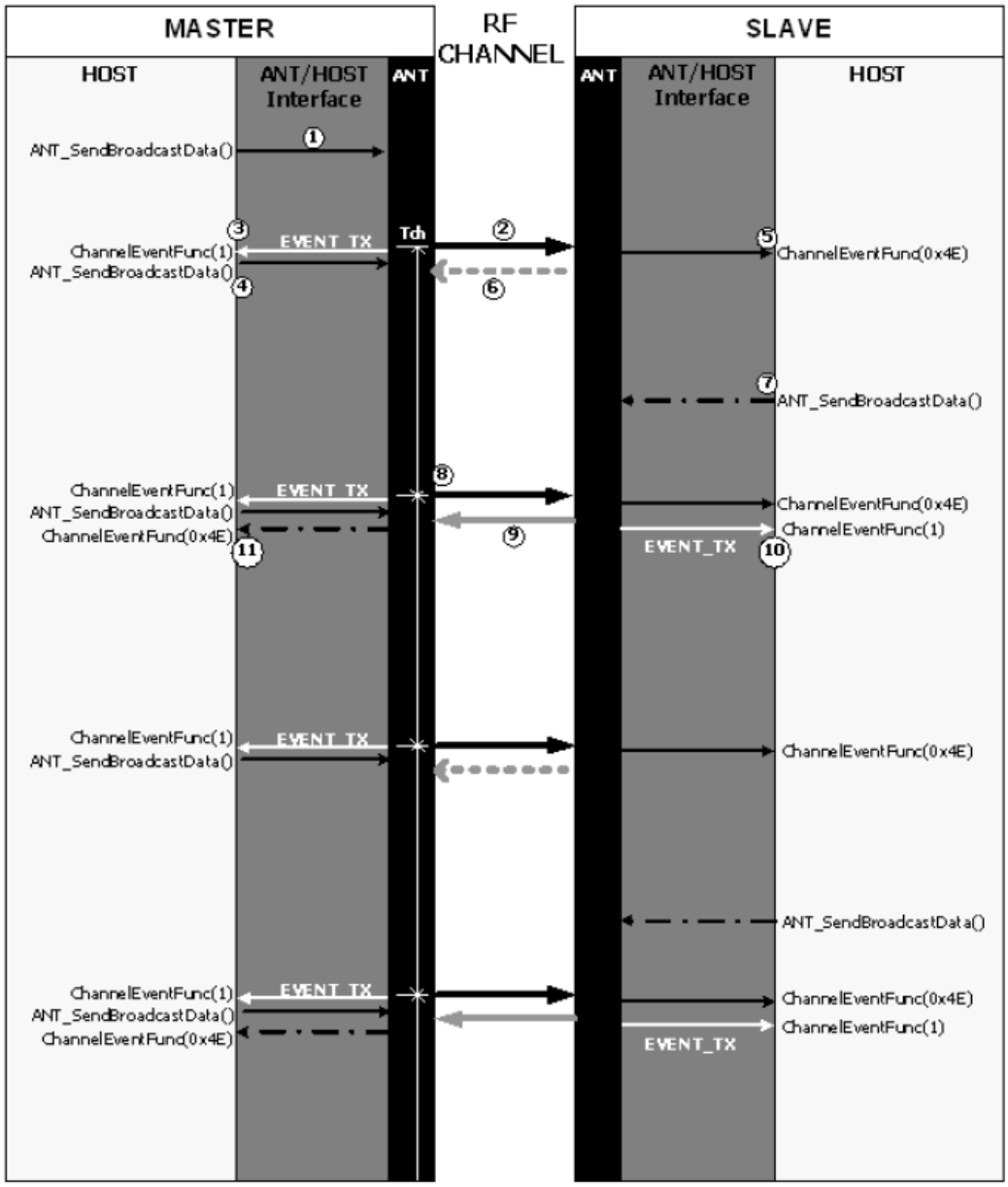


Kuva 3. ANT-viestintä isännältä orjalle /2/

Isäntälaite lähettää dataa jatkuvasti eteenpäin, joka aikaikkunassa. Mikäli uutta dataa ei tule, lähetetään vanha data uudelleen. Vastakkaiseen suuntaan kulkevia viestejä ei "pollata" jokaisella kanava periodilla. Broadcast viestit siis lähetetään

vastakkaiseen suuntaan vain kerran. Broadcast dataa ei koskaan hyväksytä(acknowledged), joten viestien lähteenä toimiva noodi ei tiedä mikäli dataa katoaa.

Broadcast data kuluttaa vähiten RF-kaistaa ja tehoa. Tätä tulee käyttää silloin kun satunnaiset data häviöt eivät haittaa. Tässä työssä tätä on mahdollista käyttää, koska akustojen jännitteet muuttuvat varsin hitaaseen tahtiin suhteessa viestinnän nopeuteen. Tarkoitus on tutkia lepojännitettä, kun akut ovat puskulatauksessa.



Kuva4. Broadcast muotoisen viestinnän sekvenssikaavio /2/

Isäntäkanava lähettää vakiona broadcast viestin jokaisella kanavaperiodilla. Isäntäkontrolleri käyttää ANT_SendBroadcastData viestiä lähettääkseen dataa ANT:ille(1). Data laitetaan puskuriin lähetettäväksi RF-kanavalla seuraavassa aikaikkunassa. Seuraavan aikaikkunan alkaessa ANT lähettää viestin RF-kanavaa (2) pitkin ja kutsuu EVENT_TX Channel tapahtuman funktiota(3). Tämä EVENT_TX viesti viestii isäntäkontrollerille, että ANT on valmis puskuroimaan uutta dataa. Isäntä voi lähettää lisää dataa uudella ANT_SendBroadvastData() komennolla(4).

Kun orja vastaanottaa lähetetyn datan, se ilmoittaa ja lähettää datan isäntäkontrollerille ChannelEvenFunc (0x4E) viestinä (5). Orjanoodilla on optio lähettää dataa kumpaankin suuntaan(6). Sekvenssi kuvassa orjanoodilla ei ollut lähettää dataa, pisteinen viiva tarkoittaa viestintää kumpaankin suuntaan, mutta varsinaista dataa ei lähetetty.

Seuraavalla kanava periodilla (8) prosessi toistetaan. ANT lähettää dataa puskuriin radiotaajuus kanavan yli, isäntänoodin isäntäkontrolleri vastaan ottaa EVENT_TX:n ja orjanoodin isäntäkontrolleri vastaan ottaa ChannelEventFunc:in(x04E). /2/

4.2.2 **Hyväksytty**(acknowledged)

Kahdensuuntaisen yhteyden aikana, niin eteenpäin/takaisinpäin suuntautuvassa liikenteessä, laite voi valita lähetettäväksi hyväksyttyä dataa seuraavassa aikaikkunassa. Noodi, joka vastaan ottaa hyväksytyn datapaketin, vastaa hyväksytty viestillä takaisin viestin lähettäneelle laitteelle. viestin lähettäneen isäntäkontrolleri saa paketin lähetyksen onnistumisesta/epäonnistumisesta. Ei ole olemassa automaattista uudelleenlähetystä epäonnistuneille datapaketeille. Isäntälaitteen sovellus voi lähettää jokaisen datapaketin hyväksyttynä datana tai voi sekoittaa broadcast ja hyväksyttyä dataa riippuen sovelluksen tarkoituksesta.

-Hyväksytyn data paketit käyttävät enemmän RF kaistaa ja käyttää enemmän tehoa.
- Hyväksytty data on ideaali tilanteessa, missä kummankin noodin tulee tietää toistensa tila.

Jos tietotyyppiä ei ole erikseen määritelty hyväksytyksi tai hyväksytty viesti on lähetetty, mutta uutta dataa ei tarjota ennen seuraavaa lähetys aikaikkunaa, niin viesti lähetetään broadcast tietotyyppinä seuraavan kanava aikaikkunan aikaan.

4.2.2 **Purske(**Burst**)**

Purskedatalähetys on mekanismi suurille tietomäärille. Purskelähetys koostuu nopeista sarjoista jatkuvasti hyväksytty data viestejä. Nopeus jolla paketit liikkuvat kanavan yli on paljon nopeampi kuin kanavaperiodi. Maksimi nopeus on 20kbps. Purskedata paketit on synkronisoitu toisiinsa nähden, eikä normaalin kanavaperiodin mukaan.

Samalla tavalla kuin hyväksytty viesteissä, lähettävä isäntäkontrolleria informoidaan purskelähetyksen onnistumisesta tai epäonnistumisesta. Tiedonsiirron onnistuminen ilmoitetaan koko purskeelle, ei pelkästään paketille. Toisinkuin hyväksytyissä viesteissä, mikä tahansa kadonnut datapaketti lähetetään automaattisesti uudelleen. Mikäli lähetys ei onnistu viiden uudelleen yrityksen jälkeen, ANT keskeyttää purskelähetyksen ja ilmoittaa isäntä mikrokontrollerille epäonnistuneesta viestistä.

Kun yksittäinen purskepaketti lähetetään, se käyttäytyy identtisesti hyväksytty viestin kanssa, eikä tällöin ole myöskään uudelleenlähetys yrityksiä yksittäisen purskepaketin kanssa. Pursketransaktion kestolle ei ole määritelty rajoituksia. Kuitenkin pursketransaktiot ottavat käyttöön kaikki avonaiset kanavat kummankin osapuolen(noodin) kanssa.

Mikäli on järjestelmässä on muita kanavia auki, tulee niitä palvella sopivassa syklissä. Vaikka ANT-protkolla kestää kevyitä häiriöitä, niin yletön kanavan kuormittaminen voi johtaa datan tai synkronoinnin katoamiseen.

4.3 Itsenäiset kanavat

Itsenäisellä kanavalla on vain yksi isäntä ja yksi orja. Jokaisella kanavalla on vain yksi isäntä ja yksi orja. Broadcast verkko muodostetaan käyttäen itsenäisiä kanavia vaikka data liikkuu vain monelta orjalta isännälle. Tällaisella verkolla on uniikki isäntä joka ei tarkoituksellisesti kommunikoi monen orjan kanssa samalla kanavalla. Data broadcast verkossa lähetetään dataa lähinnä eteenpäin, ei vastakkaiseen suuntaan. Tämä vähentää useamman orjan yhtäaikaisen datan lähettämisen yksittäiselle isännälle. Tämä poikkeaa jaetusta kanavasta, jossa on yksi isäntä ja monia orjia. Tähän liittyy osoitejärjestelmä, joka mahdollistaa datan liikkumisen kumpaankin suuntaan.

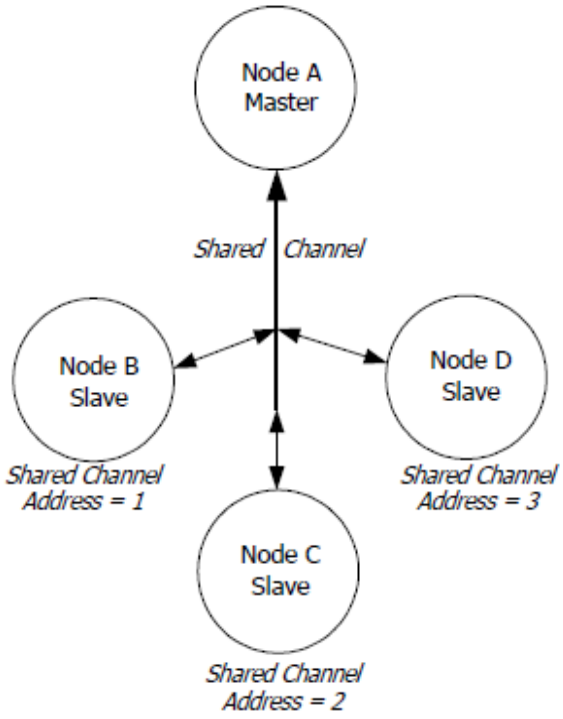
Vaikka itsenäiset kanavat tarjoaa yksinkertaisen implementaation, noodi kykenee ylläpitämään rajoitetun määrän samanaikaisia itsenäisiä kanavia, joka rajoittaa järjestelmän laskenta kykyä.

4.3.1 Yksittäisen ANT-kanavan salaus

Yksittäinen kanava salaus voidaan aktivoida itsenäistä kanavaa tukeville laitteille. Sitä ei voi käyttää jaetulla kanavalla. Salatut kanavat mahdollistavat ja helpottavat tapauksia, jossa tarvitaan salattu langatonyhteys. Jossain ANT-laitteissa on yksittäistä kanavaa varten salausominaisuus, joka voidaan ottaa käyttöön. Näissä laitteissa salaus tapahtuu protokolla tasolla vähentäen sovellustason kuormaa.

Mikä tahansa määrä orjanoodeja voidaan parittaa salatulle isäntä kanavalle. Kerran paritetut laitteet, jotka tukevat yksittäisen kanavan salausta voivat keskustella isännän kanssa. Onnistunut keskustelu mahdollistaa orjan purkaa salatut viestit, jotka se vastaanottaa isännälle. Kaikki orjalle tämän jälkeen isännälle lähettämät viestit lähetetään salattuna. Jokainen kuunteleva laite joka ei tue salausta tai epäonnistuu keskustelussa, ei voi purkaa dataa.

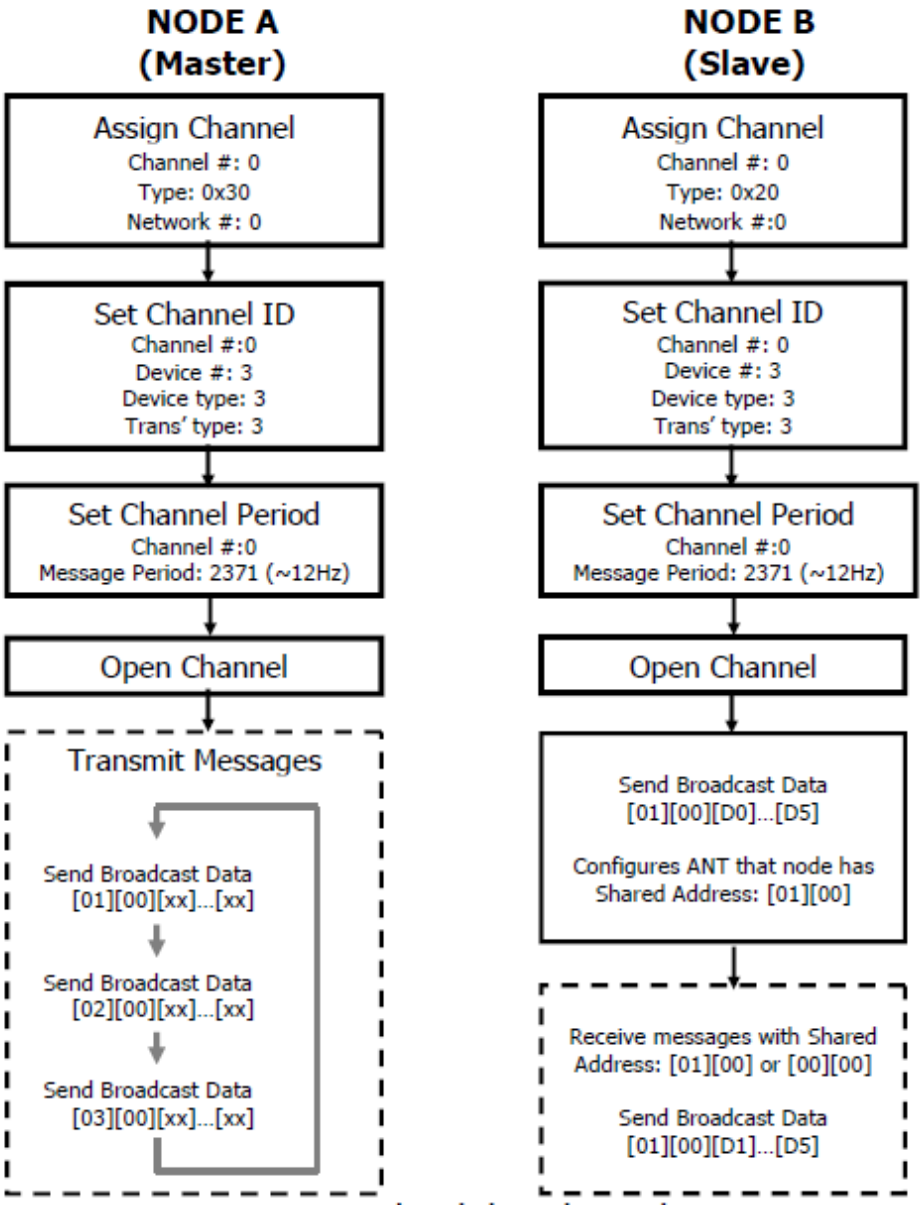
4.3.2 Jaettukanava



Kuva5. Jaetun kanavan topologia /2/

Jaettua kanava-asetusta voidaan käyttää kun noodin täytyy vastaanottaa ja mahdollisesti prosessoida dataa monelta noodilta. Tällä asetuksella useampi noodi jakaa itsenäisen kanavan kommunikoidakseen isäntänoodin kanssa. Jaetun kanavan osoite voi olla yksi tai kaksi tavua pitkä. Käytännössä tämä tarkoittaa verkkoon liitettävien laitteiden maksimimäärää. Kaksi tavua käyttämällä voidaan lisätä yli 65000 noodia ANT-verkkoon. Tämä vaikuttaa käytännössä siihen, että dataa voidaan lähettää vähemmän. Koska Jaettukanavan osoite käyttää tietosisällöstä 1-2 tavua.

Verkon tyyppin, radiotaajuuden, laitenumeron, lähetystyypin ja laitetyypin asettaa isäntänoodi. Kaikki orjanoodit jotka tahtovat käyttää tätä jaettua kanavaa, täytyy käyttää näitä samoja parametrejä. Kanavaperiodi jokaiselle lähettävälle noodille on 4Hz itsenäisellä kanavalla. Jaetullakanavalla kanavaperiodi on asetettava 12Hz:iin, jotta isäntänoodi kerkeää palvelemaan kaikkia orjanoodeja. Orjat voivat kommunikoida takaisin isännälle vain silloin kun niitä palvellaan. Esimerkiksi isäntä käy ensimmäisellä kanavaperiodilla läpi noodi a:n ja seuraavalla kanava periodilla noodi b:n. Isäntä palvelee jokaista noodia kanavaperiodi jaettuna noodien määrällä. Eli isäntä palvelee 6Hz tahtiin.



Kuva6. Laitteiden kanava-asetukset /2/

Hyvä käytäntö on varmistaa, että isäntäkanava on auki ennen orjanoodeja. Kun kanava on auki isäntä jakaa dataa joka kanavaperiodilla. Isännän mikrokontrollerin sovelluksessa on tärkeää huolehtia siitä, että jaettu osoitekenttä vaihtuu jokaisen lähetetyn viestin aikana. Jaetun osoitekentän pitäisi vaihtua 6Hz välein.

Kun kanava on auki orjanoodilla, isäntäkontrollerin täytyy lähettää yksittäinen broadcast viesti ANT-piirille, ensimmäisen tai toisen tavun täytyy sisältää kyseisen noodin jaetunkanavan osoite. Tämä asettelee ANT:in kuuntelemaan

viestejä jotka on tarkoitettu kyseiselle orjalle. ANT informoi isäntäkontrolleria aina kun se vastaanottaa viestin isännältä, jossa on kyseisen orjanoodin jaetun kanavan osoite.

Isäntä puolella ANT ilmoittaa aina mikrokontrollerille(tai tässä tapauksessa tietokoneen prosessorille), kun viesti on vastaanotettu käänteisestä suunnasta orjalta tietyllä jaetunkanavan osoitteella. Tällaisessa verkossa jokainen orjanoodi lähettää viestin takaisin isäntänoodille kun kyseisen orjan jaetun kanavan osoite on mukana viestissä.

4.3.3 Jatkuvaskannaus

Jatkuvaskannaus tila on yksi metodi jota ANT-noodit voivat käyttää, kun niiden täytyy vastaanottaa ja prosessoida dataa monelta noodilta. Toisinkuin tapauksessa, jossa yksi isäntä kontrolloi monta orjia(kuten jaetussa kanavassa), jatkuvan skannauksen tila vastaanottaa täysiaikaisesti, mahdollistaen vastaanoton useilta lähettäviltä isänniltä milloin tahansa. Kuten jaetussa kanavalla, kaikki laitteet käyttävät samaa radiotaajuutta.

ANT-radio keskusnoodissa on aina "työllistetty" jatkuvan skannauksen tilassa, eli muita kanavia ei voi avata tälle noodille. Kun radiotaajuus on aktiivinen, kyseinen noodi vie suuren määrän tehoa.

Jokaisella lähettävällä noodilla tulee olla uniikki jaettu osoite. Vastaanottava noodi on asetettu kaksisuuntaiseksi vastaanottavaksi kanavaksi joka on avattu "Open Rx Scan Mode":lle. Koska noodi vastaanottaa täysiaikaisesti, niin kanava periodia ei tarvitse asettaa. Vaikka keskusnoodi vastaanottaa täysiaikaisesti, niin se silti voi lähettää viestejä takaisin isäntänoodille. Tätä varten isännän pitää ensin lähettää vastaanottavalla noodille, joka voi sitten lähettää mahdollisesti dataa takaisin tälle tietylle isännälle vastakkaiseen suuntaan. "Continuous scanning mode":ssa on pienempi latenssi kuin jaetulla kanavalla. /2/

4.4 Laitteiden paritus

Kahden laitteen paritus(isäntä orjalle) vaatii suhteen luomisen kahden noodin välille, jotka haluavat kommunikoida keskenään. Suhde voi olla kestävä tai väliaikainen.

Paritusta varten orjalaitteen täytyy hankkia isäntälaitteen uniikki kanavatunnus. Mikäli on kyse pysyvästä parittamisesta, niin orjanoodin täytyy tallentaa isännäntunnus pysyvään muistiin. Tätä tunnusta käytetään kanavan avaamiseen tällä tunnuksella kaikissa seuraavissa kommunikaatio sessioissa. Ei pysyvässä suhteessa paritus kestää kanavan aukiolon ajan. Aikakatkaisun jälkeen paritus katoaa, eli se on väliaikaista.

Mikäli isäntä käyttää vain broadcast viestintää tai jaettua kanava ominaisuutta, niin useammat orjat voivat keskustella saman isännän kanssa.

Kun isäntä laitteen kanava avataan se aloittaa broadcast viestinnän. Sen uniikki kanava tunnus on mukana joka viestissä. Kun orja laitteen kanava on avattu, alkaa se heti etsimään isäntiä, joka on isännän yhteensopiva kanavatunnus. Kanava tunnuksen antaa orjan isäntäkontrolleri. Mikäli orjalla ei ole tiedossa tietyn isännän kanavatunnusta, niin paritus mekanismi ei on saatavilla. Orja voi etsiä isäntää käyttämällä 'wildcard' tunnusta kaikissa kanavan tunnus kentissä. Tämän jälkeen orja etsii isäntää tietämiensä kriteereiden perusteella. Esimerkiksi orja voi tietää millaisen laitetyyppiin se haluaa yhdistää, mutta ei varsinaista laitteen numeroa tai lähetys tapaa. Orjan mikrokontrollerin sovellus asettaa sen jälkeen kanava tunnuksen tiedetyn laitetyypin mukaan, ja asettaa 'wildcard':n jäljelle jääviin kenttiin. Kanavan auetessa orja etsii mitä tahansa isäntiä tuolla tietyllä laitetyypillä. Onnistuessaan isännän tunnus tallennetaan ja käytetään samaan tapaan kuin aiemmin kuvailluissa tapauksissa, kaikissa tulevaisuuden yhteyksissä.

5 PROTOTYYPIN KASAAMINEN

5.1 Käyttöönotto ja konsepti

Piirien toiminta ja asetukset testattiin ensin Antwaren avulla, ANT-piirit liitettyinä ANT USB-alustoille. Toisessa vaiheessa käytettiin yhtä rakennettua noodia ja Antware orjia. Broadcast viesti lähetettiin isännälle Antware noodista. Kummatkin lähettivät viestejä isännälle. Ajan säästämiseksi on käytetty Antware lähdekoodia ja se on muokattu vastaamaan akuston valvonnan pääsovellusta. Tarkoitus on vastaanottaa kaikilta akuilta jännitteet ja mikäli jokin jännite poikkeaa liikaa isäntä lähettää kyseiselle orjanoodille viestin. Tämän jälkeen orjanoodi kytkee transistorin päälle, jolloin akun käyttöjännite pääsee mitoitetulle vastukselle ja tästä syntyvä virta purkaa akkua. Orja lähettää jatkuvasti jännitetietoja isännälle, kun akustojen jännitteet ovat samalla tasolla orja kytkee transistorin pois päältä saatuaan viestin isäntänoodilta. Työssä käytettiin analogista lämpötila jännitettä akuston analogisen jännitteen sijaan.

Antwaren kaupallista hyödyntämistä varten selvitettävä lisenssin laajuus.

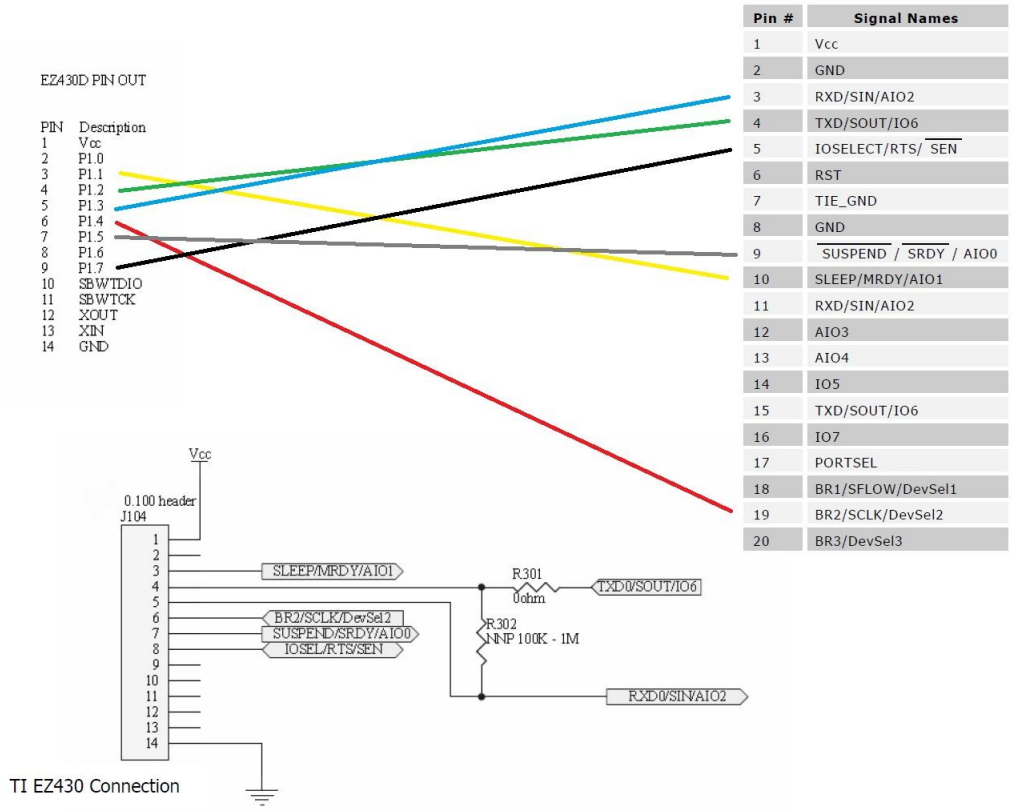
5.2 Työn eteneminen

5.2.1 Kokoonpano

Kuva7. ANT-rakennussarja /4/

Työn toteuttaminen tapahtuu osittain rajallisin resurssein, sillä jo yksi ANT-rakennussarja maksaa noin 600€. Kuten aiemmin on todettu, tämä työ keskittyy ANT-verkon ominaisuuksien testaamiseen, ja akuston valvonnan soveltuvuuden testaamiseen. TI Eval Kit sisältää 4 CC2571 moduulia, 2 EEPROM Boardia, 2 Battery Boardia, 2 USB-alusta, 2 CR2032 patteria. ANT:in virallinen sivusto tarjoaa kehittäjille varsin kattavan tarjonnan erilaisia lähdekoodeja ja esimerkkejä. Esimerkit ja tekniset dokumentaatiot ovat myös erittäin laadukkaita ja laajoja. www.thisant.com sivustolta löytyy myös kehittäjäyhteisö, joka ainakin jossain määrin on aktiivinen. Forum on hyvä paikka etsiä tietoa, sekä kysyä kysymyksiä. MSP430 liittyvät erityiset ongelmat ratkeaa kuitenkin paremmin Texas Instrumentsin sivustolla, myös siihen liittyvä dokumentaatio löytyy sieltä. Kehitystyökaluna käytetty IAR Embedded on Texas Instrumentsin ylläpitämä lisensöitävä kehitystyökalu. /4/

Yksi ANT-noodi tehtiin liittämällä MSP430 ja ANT-piiri yhteen interfacella. Ajan puutteen vuoksi ja vaativan piirilevytekniikan puutteiden vuoksi jouduttiin rajapinta tekemään johdottamalla, kuten kuvassa. MSP430 on irrotettavissa development boardista, käyttöjännitteen voi tällöin ottaa patterialustalta.

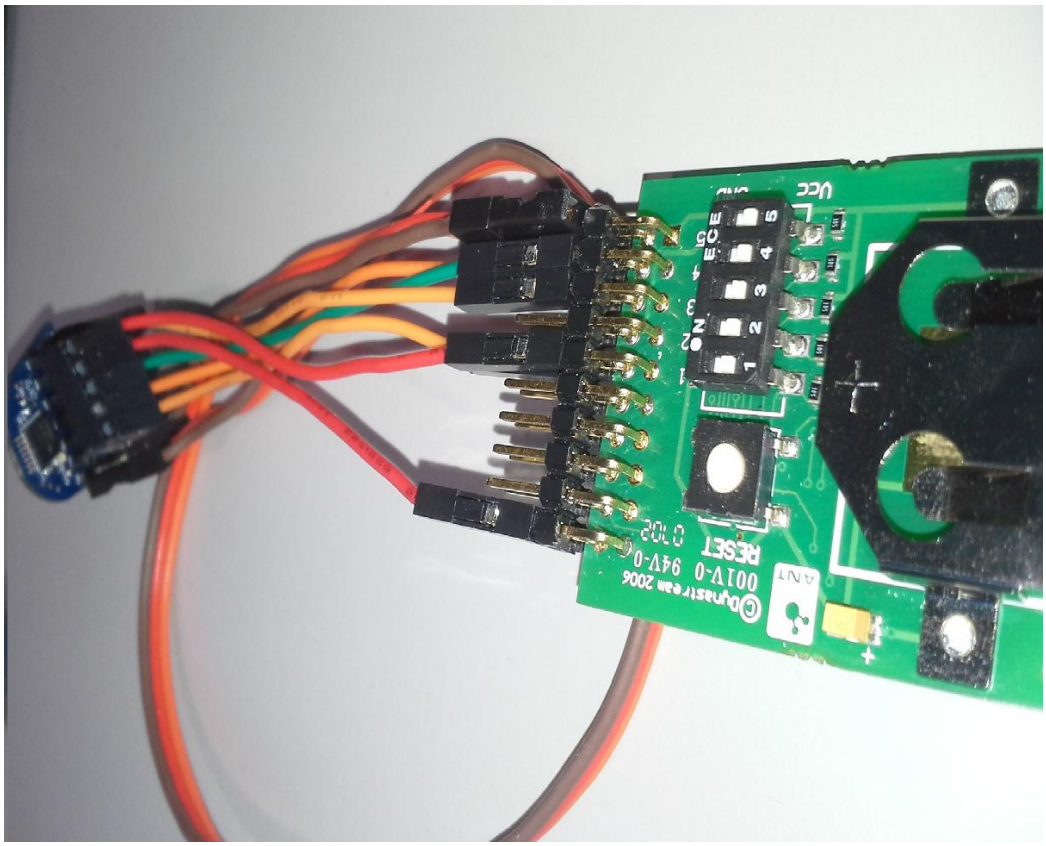


Pin #	Signal Names
1	Vcc
2	GND
3	RXD/SIN/AIO2
4	TXD/SOUT/IO6
5	IOSELECT/RTS/ $\overline{\text{SEN}}$
6	RST
7	TIE_GND
8	GND
9	$\overline{\text{SUSPEND}}$ / $\overline{\text{SRDY}}$ / AIO0
10	SLEEP/MRDY/AIO1
11	RXD/SIN/AIO2
12	AIO3
13	AIO4
14	IO5
15	TXD/SOUT/IO6
16	IO7
17	PORTSEL
18	BR1/SFLOW/DevSel1
19	BR2/SCLK/DevSel2
20	BR3/DevSel3

Kuva8. Rajapinta kytkentä /5/

Pohjana käytettiin DynaStreamin toimittavaa ohjelmaa MSP430 piirille. Ongelmia aiheutti aluksi ajastin, joka tuntui jäävän jumiin. Tällöin on syytä tarkistaa patterialustan patterin napajännite, rajapinnan johdotukset, ja mahdollisesti rikkoutunut ANT-piiri.

Yksi noodi rakennettiin ANT-USB + CC2571 alustalle. Tätä käytettiin isäntänoodina. Antware2 toimii tässä tapauksessa isäntänoodin pohjana. Isäntänoodin tehtävä on myös käsitellä dataa ja tarvittaessa lähettää tietylle orjanoodille käsky suorittaa akun purku, eli fet-transistoria käyttäen ohjata akuston jännite valikoidulle vastukselle. Kun jännite on laskenut muiden akuston muiden akkujen kanssa samalle tasolle isäntä lähettää viestin ja akun purkaminen lopetetaan, taas isäntänoodin antaman viestin perusteella. Orjanoodin MSP430 käsittelee viestin ja mikäli se sisältää on/off viestin toimii sen mukaan. /5/



Kuva9. Orjanoodi

Toinen noodi rakennettiin samaan tapaan kuin ensimmäisenkin. Näitä kutsutaan noodi b:ksi ja noodi c:ksi. Isäntänoodina toimii ANT:in usb-alusta CC2571 piirillä. Isäntänoodin pääsovelluksen pohjana toimii Antware, jota muokkaamalla saatiin aikaan haluttuja ominaisuuksia. Pääasiassa käsitellä orjanoodilta tulevaa jännitedataa ja tarvittaessa lähettää käskyjä, myös mahdolliset hälytykset. Olisi ehkä mielekästä tutkia kevyempiäkin ohjelmistoratkaisuja, sillä pääsovellus saattaa toimia alaerotinasemalla gprs-yhteyttä käyttäen. Välttämättä kuitenkaan ei ole mahdollista käyttää pc:n tehoista laitteistoa. Käytetyn pc-sovelluksen avulla voidaan kuitenkin demonstroida pääpiirteittäin akuston valvonnan toimintaa.

Varsinaisessa sovelluksessa noodit tulevat saamaan käyttöjännitteensä suoraa akulta, jännite mitoitetaan yksinkertaisella jännitteenjako kytkennällä mikrokontrollerille sopivaksi, noin 1,8-3,0V. Transistori toimii kytkimenä, joka on tavallinen FET-transistori. Kun hila saa positiivisen jännitteen se alkaa vetämään, jolloin jännite pääsee kulkemaan transitorin läpi ja siitä taas vartavasten mitoitetulle virtavastukselle. Piirilevyn suunnittelu karkeaa layout kuvausta lukuunottamatta jää tämän työn ulkopuolelle. Tässä keskitytään tutkimaan ANT-piirien ja isäntäkontrollerin ominaisuuksia. Lopputuloksena karkeahko prototyyppia varsinaisesta mahdollisesta lopputuotteesta. Työssä siis rakennettiin kaksi noodia, joissa kummassakin on ANT-piiri, sekä isäntäkontrolleri. Nämä on yhdistetty aiemmin kuvatulla rajapinnalla. Jaettua kanavaa käyttämällä onnistuttiin lähettämään dataa kummaltakin noodilta isäntänoodille, joka on ANT-piiri kytkettynä usb:llä tietokoneeseen. Tämän isäntänoodin toimintaa ohjaa Antware ohjelmisto, joka muokattiin vastaamaan akuston valvonnan sovellusta, vaikkakin yksinkertaisempi sovellus tulee varmasti olemaan tarpeellinen, niin riittää tämä kuitenkin osoittamaan ANT-piirin toiminnallisuuden kyseisessä sovellutuksessa(langaton akuston valvonta).

Antware sovellusta muokattiin siten, että otan orjanoodilta tulevat datat talteen ja vertailen tätä dataa keskenään. Viestin mukana lähetettiin myös akuston tunnus. Antware lähetys lokista voi lukea, että tapahtumien arvo on EVENT_TX_03, joka tarkoittaa lähetyksen onnistumista orjille, kun orjanoodit vastaanottavat tämän tapahtuma-arvon, niin orjanoodin ohjelma käsittelee mukana tulleen arvon. Tällöin tarkistetaan arvon mukana tullut akuston tunnus, sekä komento(päälle/pois). Tämä ei noudata mitään virallista viitekehystä, vaan työssä päätettiin lähettää tietyn akuston tunniste ja sillä toimivan orjanoodin tunniste erikseen broadcast viestin mukana, kuten myös mahdollisen komennon. Koska akkuja on yleensä vain 12, riittää tavu helposti. Itse päälle/pois komennon voin luonnollisesti esittää yhdellä bitillä. Joten käytän tavun ensimmäiset neljä bittiä akuston tunnuksen esittämiseen ja viimeisen bitin komennon esittämiseen. Orjanoodi ohjelma tarkistaa kuuluuko tunnus kyseiselle orjanoodille, jonka jälkeen se ajaa lähetetyn komennon. Työssä käytettiin karrikoitua esitystä, eli kun orjanoodin täytyisi aktivoida transistori asettamalla ulostulo ylätilaan, niin näkyy se näkyy ledin aktivoitumisena. Varsinaisessa lopputuotteessa aktivoituisi myös transistori, jonka jälkeen akun jännite pääsisi kulkemaan transistorin yli virtavastukselle, ja akun jännite alkaisi laskea. Näin tapahtuu kunnes isäntänoodin ohjelmisto toteaa kyseisen akun olevan samalla tasolla muiden akkujen kanssa.

Noodit lähettävät jatkuvasti broadcast muotoista dataa isännälle ja kanavaperiodia valittaessa tämä tulee ottaa huomioon. Työssä kanava meni aluksi tukkoon, mutta kun nostin 'message periodin' 12Mhz:iin, ongelma poistui. Muuten asetukset on kuten aiemmin mainittu. Isännälle on aseteltu Channel ID:ksi 49, Device Type 1 ja, Transmission Type 5, Message perioid 12Mhz. Kanava on asetettu myös jaettu tilaan. Kanava avataan Antware ohjelman käskystä. Orjanoodit on aseteltu muuten samoin, mutta kummankin isäntäkontrollerin koodissa on kutakin orjanoodia kohden oma uniikki Jaettukanava osoite. Tuo osoite on tiedossa vain kyseisellä noodilla ja isäntänoodilla. Tämän osoitteen avulla isäntä kykenee tunnistamaan jaettua kanavaa käyttävät orjanoodit. Isäntäsovellus laskee käytössä olevien orjanoodien määrän käymällä läpi kaikki jaetun kanavan osoitteet. Periaatteena on käyttää tunnusta yhdestä eteenpäin, tunnus vastaa samalla verkossa olevien laitteiden määrää.

Orjakontrollerien mikrokontrollerit lukevat myös koko ajan isänniltä tulevat tapahtumat ja niiden broadcast muotoisen datan. Mikäli viesti sisältää tietyn arvon, niin akuston ulostulo kanavan tila muutetaan jännitteiseksi(ylätilaan). Työssä demonstroitiin sitä ledin avulla, koska varsinaista akustoa ei ole käytettävissä, tätä työtä varten ei tehdä erikseen piirilevyjä, vaan käytetään kahta prototyyppiä, sekä isäntänoodia, Antware ohjelmistoa, sekä isäntäkontrollereille ladattua sovellusta.

Antware käsittelee jatkuvasti orjanoodien lähettämää dataa. Mikäli jonkin yksittäisen akun jännite poikkeaa määritellystä arvosta tietyn prosentuaalisen määrän verran, niin isäntä lähettää broadcast viestinä kaikille akuille datan. Orja jonka tunnus löytyy tietosisällöstä vaihtaa ulostulon ylätilaan. Kun jännite alkaa laskea ja on määritellyllä tasolla, niin taas isäntä lähettää broadcastin kaikille orjanoodeille jaetun kanavan välityksellä. Tietosisältö pitää sisällään tavun, jossa on akun(orjanoodi) tunnus, sekä komento ajaa ulostulo alas. Tämän jälkeen ulostulon tila menee nollaksi, jonka jälkeen transistori lakkaa vetämästä ja akun purkaus keskeytyy, jännitteen ollessa nyt muiden akkujen tasolla. Työssä demonstroitiin tätä antamalla lämpötilasensorin jännitteen laskea, antamalla sen jäähtyä.

Tietosisältö sisältää siis noodin tunnisteen, jota kutsun akuston tunnukseksi, sekä päälle/pois komennon. Koska akusto koostuu 12 akusta, riittää helposti tavu tunnus sekä komennon lähettämiseen. Työssä asetetaan jokin pysyvä vakioarvo, josta jännitedata ei saa poiketa. Mikäli jonkun akun jännite kuitenkin suuresti poikkeaa tästä vakioarvosta, niin isäntä lähettää 'päälle' komennon.

Työssä siis demonstroitiin akuston jännitteen seuraamista ja siihen reagoimista lämpötila-anturin jännitedataa seuraamalla. Kun tietyn noodin lämpötila nousee liiaksi, niin noodin ulostulo aktivoituu(led syttyy) tällöin jäähdytän noodia, jolloin jännite laskee, kuten käy kun akulta kulkee virta vastuksen läpi. Ulostulon ollessa aktiivinen ledi palaa.

Orjanoodit käyttävät omien uniikkien kanavien sijaan jaettua kanavaa. Tämä vie enemmän virtaa, ja jaetun osoitekentän vuoksi laskee liikutettavan datan määrää pakettia kohden. Orjanoodilla täytyy olla oma jaettu osoite. Tämän tietää vain yksittäinen orja ja isäntä noodi. Kanava periodi asetus tulee olla 4Hz sijaan 12Hz. Parhaimmillaan jaettua kanavaa voi käyttää 65000-noodia, mikäli jaettua osoitetta varten on kaksi tavua käytössä.

5.2.2 Synkroninen viestintä bittivirta kontrollilla

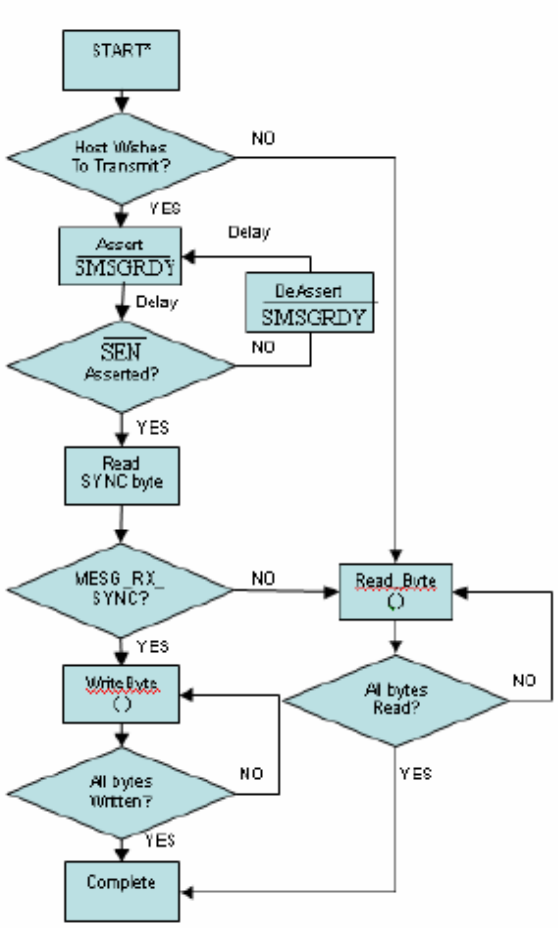
Työssä käytettiin synkronista bittivirta kontrolloitua viestintää.

Jos isäntä mikrokontrollerissa ei ole SPI-yksikköä , ANT:iä voi silti kontrolloida käyttäen bittivirta kontrollia. Tätä metodia käytettäessä

sarjalinjat on implementoitu ohjelman avulla, joka kontrolloi IO-linjoja. Ainoa ero on se rautaversioon se, että joka tavun jälkeisen pulssituksen sijaan, SRDY pulssitetaan viestin jokaiselle bitille. Data bitit vaihtavat tilaa SCLK:n laskevasta reunasta ja luetaan SCLK:n nousevan reunan aikaan.

Host->Ant transaktio on hyvin samankaltainen prosessi kuin Ant->Host transaktio. Suurin ero on, että isäntäkontrolleri asettaa SMSGRDY:n aktiiviseksi infromoidakseen ANT:ia, että haluaa lähettää viestin. ANT vastaa aktivoimalla SEN linjan, jonka jälkeen odottaa isäntäkontrollerin asettavan SRDY:n aktiiviseksi. Kun tämä on tehty ANT lähettää SYNC tavun.

Isäntäkontrolleri tekee kaiken bittiprosessoinnin SCLK:n signaalin nousevalla reunalla. Poikkeuksena kun lähetetään tavu isäntäkontrollerilta ANT:lle. Ensimmäinen data bitti täytyy aktivoida ensimmäisen nousevan kellon aikana. Viimeinen nouseva reuna tavu transaktiossa on tapahtuma, joka ajaa tavu prosessia eteenpäin.



Kuva10. Tilakone Synkronisesta viestinnästä /5/

Mikäli ohjelman kulku pysähtyy SCLK:n kohtaan, on syytä tarkistaa kanavien asetukset. On hyvä testata ne esimerkiksi ANT-usb tikkuihin kiinnitettynä käyttäen vakioasetuksia, jotta voidaan poissulkea mahdollinen rautavika. Patterialustojen patterit on myös syytä tarkistaa.

Mikäli ajastin on ongelma on varmistettava ohjelmistoympäristön optimointi asetukset, sillä kääntäjän optimoija saattaa aiheuttaa ongelmia tietyissä IAR-embedded versioissa. USSHORT ustime muuttuja on syytä olla volatile. /5/

5.3 Jatkokehitys

Jatkon kannalta on mietittävä miten jaetut osoitteet jaetaan orjanoodeille, sekä tunnukset. Mikäli arvot ohjelmoidaan aina erikseen isäntämikrokontrollerin muistiin, niin anturimäärän kasvaessa tehtävä muuttuumahdottomaksi. Periaatteessa tietosisällön välityksellä kulkevat orja kohtaiset tunnukset voisi periaatteessa antaa isännän tehtäväksi ohjelman ajon aikana.

Orjanoodi yksikön tulee olla helposti liitettävissä akun napoihin ja kotelon tulee olla hermeettinen, jotta saadaan pidettyä kosteus ulkopuolella.

6 PÄÄSOVELLUS

6.1 MSP430-isäntäkontrollerin sovellus

Isäntäkontrollerin ohjelma sisältää kuusi lähdetiedostoa ja näiden headertiedostot. Lisäksi Config.h tiedosto, jossa määritelty vakiot.

Ohjelman osat:

-Main.c

-BitSyncSerial.c

-System.c

-Timer.c

-ANTInterface.c

-Atod.c

Ohjelmaa lähdetään suorittamaan luonnollisesti Main.c tiedostosta. Aluksi määritellään ANT-verkon asetuksien vakiomuuttujat, jonka jälkeen staattiset ANT-asettelumuuttujat, lähinnä kanava asetuksia. Tämän jälkeen esitellään paikalliset funktiot kuten Main_ProcessANTEvent ja määritellään muut staattiset data muuttujat. Tämän jälkeen alkaa varsinainen main funktio. Siinä alustetaan kontrollerin kellotaajuudet, ajastin, A/D-muunnin, ANT-rajapinta ja kytketään pois watchdog. Sovelluksissa jossa patterin käyttöikää pyritään maksimoimaan on käytettävä matalaenergia herätys toiminnallisuuksia. Nämä alustetaan ohjelman tässä vaiheessa. Kun kaikki on initalisoitu aiheutetaan 500ms viive(ajastin toiminnot on määritelty timer.c tiedostossa). Tämän jälkeen alkaa varsinainen kommunikaatio ANT-piirin kanssa, ensimmäisenä tehdään kanava-asetukset. Kun tämä on tehty ohjelma siirtyy ikuiseen silmukkaan, jossa käsitellään tulevevat viestit. Tässä silmukassa kutsutaan myös funktiota Main_ProcessANTEvent, jota

muokkaamalla ja laajentamalla saadaan eri toiminnallisuuksia aikaan. Mikäli halutaan myös lähettää dataa on määriteltävä funktion ANTInterface_MesgProcess(pucTxBuffer) parametriksi osoitin lähetettävän viestin puskuriin. Mikäli on tarkoitus vain vastaanottaa dataa, se jätetään null arvoiseksi.

BitSyncSerial_Transaction funktiossa BitSyncSerial.c tiedostossa tarkistetaan onko viestiä lähetettävänä(mikäli pucTxMsgBuffer ei ole null). Varmistetaan, että MSGRDY on asetettu aktiiviseksi. Tämän jälkeen ajetaan sekä vastaanotto, että lähetys funktiot. ReadyByte() käsittelee tulevan tavun ja WriteByte() lähetettävän tavun. Kumpikin käsitellään teoriassa samaan aikaan. Funktio palauttaa lopuksi bRxMessage muuttujan arvon, joka sisältää tiedon onko viestiä vastaanotettu vai ei. Sekä Read, että WriteByte funktiossa putsataan keskeytys tilalippu, sekä käsitellään vastaanotettava tai lähetettävä tavu. Vastaanotettaessa SYNC_SRDY_PULSE() funktio lähettää pulssin ANT-piirille yhteistä rajapintaa käyttäen SRDY linjaa pitkin, jonka jälkeen odotetaan SCLK linjalta tämän nousevaa reunaa, luetaan bitti. Kun tavu on käsitelty ReadyByte() funktio palauttaa tavun. Lähettäminen toimii samoin kuin vastaanotto funktio.

```
UCHAR *pucEventBuffer = ANTInterface_GetPendingEvent();

if(pucEventBuffer[10]==0xFF&&pucEventBuffer[9]==SHARED_ADDRESS_CHECK)
{
LED_OUT |= LED_BIT;
}
else if(pucEventBuffer[10]==0x00&&pucEventBuffer[9]==SHARED_ADDRESS_CHECK)
{
LED_OUT    &= ~LED_BIT;
}
```

Kuva11. Datan käsittelyä orja:n päässä

Tässä koodissa isäntäkontrolleri käsittelee ANT-viestin. Tarkistetaan onko tietosisältöistä mukana kyseisen kontrollerin id, sekä komento(päälle/pois). ANTInterface_GetPendingEvent funktio noutaa kyseisellä ajanhetkellä olevan tapahtuman, joka asetetaan *pucEventBufferiin. Tämän jälkeen käydään läpi pucEventBuffer tietosisällöstä komennot, sekä orjan id:n. Mikäli ehdot

täyttyvät(komento 0xff ja SHARED_ADDRESS_CHECK pitää sisällään kyseisen orjanoodin id:n, ledi alkaa palaa.

6.1.1 A/D-muunnos

Atod.c sisältää A/D-muuntimen initialisoinnin. Käytössä oleva kanava on A6+/- ,jossa on sisäisen lämpötilasensorin jännite. Koodissa kanava asetetaan SD16INCTL0 muuttujan avulla, arvoksi tulee SD16INCH_6. Tästä tiedostosta löytyy myös funktio, joka hakee A/D-muunnoksen tuloksen. Kolmas funktio käsittelee datan ja vie tuloksen pucTxMesg pointterin osoittamaan puskuriiin, jolloin se viedään rajapintaa pitkin ANT-piirille lähetettäväksi eteenpäin.

Main_ProcessANTEvent funktio käsittelee isäntäilta tulevat tapahtumat ja broadcast muotoisen datan. Se kutsuu ANTInterface_GetPendingEvent() funktiota, joka palauttaa vastaanotetun datapuskuriin pointterin. Tässä vaiheessa ohjelmat tarkistaa, onko akuston id kyseisen noodin, sekä komennon. Mikäli id täsmää, ohjelma ajaa isännän lähettämän viestin komennon(Ajaa output kanavan ylös). Tämä tarkoittaa prototyypissäni käytännössä sitä, että isäntäkontrollerin led syttyy, koska ulostulokanavan pinnistä tulee jännitteinen. Jännite pysyy, niin kauan kunnes kyseisen akun jännite on tasoittunut muiden akkujen jännitteiden kanssa samalle tasolle. Komento tulee isäntänoodilta, joka käsittelee jatkuvasti kaikilta akuilta erikseen tulevia jännitteitä.

Samassa funktiossa haetaan myös A/D-muuntimelta akun jännite ja lähetetään se isäntäkontrollerille. Tietosisältö pitää sisällään akun id:n, sekä akun jännitteen. Työssä käytettiin ympäristön lämpötilaa demonstroimaan akun jännitettä. Periaate on kuitenkin sama, kuin mikä tahansa A/D-muuntimelle tuleva jännite käytettäessä.

```
void AtoD_SetTempMesg(UCHAR *pucTxMesg, UCHAR ucChannel)
{
   //Temperature conversion for this particular AtoD and built in temperature sensor
   //Temperature (1/100 Degrees C) = (((AtoD Value/65536)*600)/1.32 - 273) * 100;

   ULONG ulAtoDVal = AtoD_GetSample();                    // Get A/D Sample (Temperature Sensor Reading)

   ulAtoDVal = ((ulAtoDVal * 45455) >> 16) - 27300;       // The equation above rearranged

   *pucTxMesg++ = MESG_DATA_SIZE;                         // set the data message size
   *pucTxMesg++ = MESG_BROADCAST_DATA_ID;                 // set the message ID
   *pucTxMesg++ = ucChannel;                              // set the channel number
   *pucTxMesg++ = 0;                                      // reserved
   *pucTxMesg++ = 0;                                      // reserved
   *pucTxMesg++ = SHARED_TYPE_A_TO_D;                     // define as a standard temperature sensor message
   *pucTxMesg++ = TEMPERATURE;                            // AtoD Subtype
   *pucTxMesg++ = 0;                                      // reserved
   *pucTxMesg++ = SHARED_ADDRESS;                         // Battery ID
   *pucTxMesg++ = (UCHAR)ulAtoDVal;                       // temperature (USHORT)lsb
   *pucTxMesg   = (UCHAR)(ulAtoDVal >> 8);                // temperature (USHORT)msb
}
```

Kuva13. A/D-muuntimen viestit orjan päässä

Tässä funktiossa asetetaan pucTxMesg osoittimen avulla tarvittavat arvot puskuriin.

6.2 Antware

AntChannelPanel.xaml.cs luokassa voidaan asettaa esimerkiksi kaikki kanavan vakioarvo asetukset.

MessagingPanel.xaml.cs koodissa pollataan läpi isännälle saapuvat broadcast viestit, sekä myös valitaan lähetettävä data sleiveille. GetPayload metodi palauttaa viestin jota juuri käsitellään. Se palauttaa 8-tavuisen taulukkoon. Payload[7] sisältää varsinaisen jännitedatan, joka luetaan, ja tarkistetaan poikkeaako se keskiarvosta.

Työssä käytettiin Antware pc-sovellusta. Varsinaista lopputuotetta varten voi olla syytä kehittää kevyempi versio sulautettuun ympäristöön, esim c-kielellä. Antware hoitaa kaiken datan-käsittelyn liittyen isäntänoodin toimintaan.

Kun orjanoodit lähettävät dataa, niin otan tietosisällön talteen getPayload-metodin avulla, joka löytyy valmiiksi lähdekoodista. Mikäli jonkun akun jännite poikkeaa merkittävästi asetetusta arvosta, niin lähetetään broadcast dataa, jossa tietosisällössä akun tunnus, sekä komento(päälle). Broadcast data asetetaan broadcast puskuri byte[] taulukkoon.

```
try
{
    byte[] payload = newResponse.getDataPayload();

    ulong sharedChannelId = BitConverter.ToUInt16(payload, 4);
    ulong ulAtoDVal = BitConverter.ToUInt16(payload, 5);
    ulong ulAtoDVal2 = BitConverter.ToUInt16(payload, 6);
    ulong DefaultValueOfBattery=3000;

    Byte channelToSend = 0;
    ulong ulAtoDValError = 0;

    sharedChannelId = sharedChannelId / 256;
    channelToSend = (byte)sharedChannelId;

                if (ulAtoDVal2 >= DefaultValueOfBattery)
                {
                    broadcastBuffer = new byte[] { 0, 0, 0, 0, 0, 0, channelToSend, 255 };
                    stringToPrint.Append("Discharging");
                }
                else
                {
                    broadcastBuffer = new byte[] { 0, 0, 0, 0, 0, 0, channelToSend, 00 };
                }


}
catch (ANT_Exception e)
{
   // stringToPrint.Append(e);
}
```

Kuva14. Datan käsittelyä isännän päässä

Tietosisältö data haetaan broadcast puskurista, jossa se on tavuittain. Seitsemäs tavu pitää sisällään lämpötila-anturin datan. Viidennessä tavussa on mukana akuston tunnus, eli jaetun kanavan osoite. Data käsitellään, sitä verrataan arvoon, jota tasoa jännite ei saa ylittää. Mikäli näin käy asetetaan kyseinen akusto purkutilaan, muutoin asetetaan purkutila pois päältä.

7 TESTAUS

Työ testattiin avaamalla jaettukanava isäntänoodin, sekä orjanoodien välille. Antware konsolista seurattiin liikenteen sujuvuutta. Broadcast viestit liikkuivat kumpaankin suuntaan ja kumpikin orja oli aktiivinen. Myös toiminnallisuus testattiin lämmittämällä lämpötilasensoria. Jännitteen noustessa ledi syttyi, ja taas sensorin jäähtyessä sammui. Viestiliikenne siis toimi ja oikea noodi reagoi, riippuen kumman A/D-muuntimen sisääntulo jännite nousi yli asetetun arvon.

8 YHTEENVETO

Vaikka työ ei pidä sisällään varsinaista markkinoille valmista tuotetta, on tämä kuitenkin hyvä pohja lähteä jalostamaan työtäni pidemmälle ja varsinaiseksi lopputuotteeksi. Olennaista on ottaa huomioon tiedonsiirtoon vaadittavan rajapinnan viemä osuus isäntämikrokontrollerin porteista. Työssäni käytettiin laboratoriosta jo löytyviä mikrokontrollereita, jotka olivat erittäin hyviä ANT-verkon ominaisuuksien selvittämiseen ja tutkimiseen. Texas Instruments tarjoaa lisäksi hyvät työkalut ja laajat dokumentit. Mikäli MSP tuoteperheestä löytyy mikrokontrolleri, joka tukee sekä rajapinta toteutusta(ANT-piirin, sekä isäntäkontrollerin välillä), ja tämän lisäksi jää vapaita portteja ulkoiselle analogiselle jännitteelle, niin työssäni käytettyjä ohjelma koodeja voidaan käyttää miltei sellaisenaan. Suurin muutos on vaihtaa MSP-kontrollerin analogia digitaali kanava-asetusta.

Jatkokehitys tulee pitämään varmasti sisällään piirilevyn, sekä koteloinnin. Lisäksi anturin tulee olla helposti kiinnitettävissä akun napoihin.

LÄHTEET

/1/ Siivonen, Kalle Tampereen Ammattikorkeakoulu.Sähkötekniikka.Sähkövoimatekniikka..Sähköaseman apusähköjärjestelmät. https://publications.theseus.fi/bitstream/handle/10024/9834/Siivonen.Kalle.pdf?sequence=2

/2/ http://www.thisisant.com/resources/ant-message-protocol-and-usage/

/3/ http://www.thisisant.com/resources/interfacing-with-ant-general-purpose-chipsets-and-modules/

/4/ http://www.thisisant.com/resources/ti-evaluation-kit-user-manual/

/5/ http://www.thisisant.com/resources/ant-reference-design-user-manual/

/6/ http://www.ti.com/lit/ug/slau176d/slau176d.pdf