Mansour Ahmadi

Liikeanturin mittausjärjestelmän toteuttaminen

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikka ja liikenne
Tietotekniikka
Aika

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikka ja liikenne	Koulutusohjelma Tietotekniikka
Tekijä(t) Mansour Ahmadi	
Työn nimi Liikeanturin mittausjärjestelmän toteuttaminen	
Vaihtoehtoiset ammattiopinnot	Ohjaaja(t) Tuomo Rantala
	Toimeksiantaja Markku Karppinen
Aika 30.4.2010	Sivumäärä ja liitteet 43+ 12

Insinöörityö tehtiin Kajaanin ammattikorkeakoululle. Työn tarkoituksena oli tutkia ja testata ADIS16360-liikeanturia ja toteuttaa 3-akselinen mittausjärjestelmä.

ADIS16360-liikeanturi on 6-akselinen inertia-anturi. Liikeanturi sisältää 3-aksellisen gyroskoopin, joka pystyy mittaamaan kulmanopeuksia arvoon ±75°/s, ±150°/s tai ±300°/s ja 3-aksellisen kiihtyvyysanturin, joka mittaa kiihtyvyyksiä arvoon ±18 g asti, missä yksi g merkitsee maan vetovoiman kiihtyvyyttä. Komponentissa on myös lämpötila-anturi. Kiihtyvyys- ja kulmanopeusanturin resoluutio on 14 bittiä, mutta lämpötila-anturin ja ADC-muuntimen resoluutio on 12 bittiä.

Liikeanturin käyttöjännitealue on 4,75 V…5,25 V, käyttölämpötila-alue -40…+105 °C. ADIS16360 kuluttaa 49 mA normaalitilassa, 24 mA virransäästötilassa ja 500 µA lepotilassa.

Anturi sisältää ohjelmoitavat itsetestaus-, virranhallinta-, ja hälytinyksiköt. Kaikki komennot ja tiedot kommunikoivat SPI-väylän kautta. Komponentilla on 24 pinniä ja sen mitat ovat 23 mm × 23 mm × 23 mm (pituus × leveys × korkeus).

Työn tärkeä osa oli liikeanturin ohjelmointi. Liikeanturille tehtiin kohina-, mittausalue- ja toistettavuustestit. Kulmanopeusanturin toistettavuustestissä käytettiin apuna Kajaanin ammattikorkeakoulussa olevaa robottilaitetta.

ADIS16360-liikeanturissa olevassa kiihtyvyysanturin luotettavuus on erinomainen kulmanopeusanturiin verrattuna. Myös sen kohina on pieni.

Tuloksista ja ADIS16360-anturin spesifikaation perusteella voitiin päätellä, että anturilta voitiin lukea kulmanopeus- ja kiihtyvyysarvot kohtalaisen tarkasti.

Kieli	Suomi
Asiasanat	ADIS16360, liikeanturi, SPI-väylä
Säilytyspaikka	☒ Verkkokirjasto Theseus ☒ Kajaanin ammattikorkeakoulun kirjasto

THESIS
ABSTRACT

School School of Engineering	Degree Programme Information Technology
Author(s) Mansour Ahmadi	
Title The Motion Sensor Measurement System	
Optional Professional Studies	Instructor(s) Tuomo Rantala
	Commissioned by Markku Karppinen
Date 30.4.2010	Total Number of Pages and Appendices 43 + 12

This Bachelor's thesis was commissioned by the Kajaani University of Applied Sciences. The goal of this thesis was to design and implement the ADIS16360 sensor measurement system.

The ADIS16360 is complete six-degrees-of-freedom inertial sensing system. Each contains: a 3-axis gyroscope with ±75°/s, ±150°/s, and ±300°/s range settings; a 3-axis accelerometer with ±18 g range; and a temperature sensor. They provide 14-bit digital data proportional to the angular rate about–and the acceleration along–the X-, Y-, and Z-axes–and 12-bit digital data proportional to the on-chip temperature, power-supply voltage, and voltage at an auxiliary analog input. An auxiliary 12-bit DAC provides an analog output with 2.5V full-scale range. The devices are fully calibrated for sensitivity, bias, axial alignment, and linear acceleration at 25 °C.

Functionally complete, they include programmable self-test, power management, and alarms. All data and commands are communicated via an SPI-compatible serial interface. Operating on a single 4.75 V to 5.25 V supply, the ADIS16360 consume 49 mA in normal mode, 24mA in low-power mode and 500μA in sleep mode. Available in 24-lead, 23 mm × 23 mm × 23 mm compact modules.

Language of Thesis	Finnish
Keywords	ADIS16360, motion sensor, tri-axis gyroscope sensor, tri-axis acceleration sensor, SPI-bus
Deposited at	☒ Electronic library Theseus ☒ Library of Kajaani University of Applied Sciences

ALKUSANAT

Tämä insinöörityö on tehty Kajaanin ammattikorkeakoulun toimeksiannosta. Työn valvojana toimi Markku Karppinen ja ohjaajana Tuomo Rantala Kajaanin ammattikorkeakoulusta. Haluan osoittaa heille kiitokset ohjauksesta työn edetessä. Kiitän myös Eero Soinista kielellisestä ohjauksesta ja haluan kiittää Jukka Heinoa.

SISÄLLYS

1 JOHDANTO

Tämän insinöörityön tarkoituksena oli ADIS16360-liikeanturin mittausjärjestelmän toteuttaminen siten, että voitaisiin mitata sekä kolmiulotteinen kiihtyvyys että kolmiulotteinen kulmanopeus.

Kiihtyvyys- ja kulmanopeusanturit ovat kehittyneet valtavasti 1970-luvulta. Nyt ne ovat pieniä, halpoja ja helppokäyttöisiä komponentteja, kuten muukin elektroniikka. Puolijohdeteknologian vallankumous muutti myös anturiteknologian. Syntyi uusi johdannaisteknologia, mikromekaniikka, jota myöhemmin alettiin kutsua lyhenteellä MEMS, Micro Electro Mechanical Systems.

Mems-anturien käyttö yleistyy nopeasti arkipäiväisissäkin kohteissa. Nykyään antureita löytyy esimerkiksi peliohjaimista, PDA-laitteista (Personal Digital Assistant), digikameroista, matkapuhelimista, autojen turvatyynyistä, lääketieteellisistä laitteista, navigointijärjestelmistä ja digitaalikompasseista. MEMS-liikeanturin mukaan on integroitu myös SPI-väylä, joten anturi on liitettävissä suoraan elektroniseen ohjausyksikköön.

ADIS16360-liikeanturi on täydellinen inertiajärjestelmä, jossa on yhdistetty samaan koteloon sekä 3-akselinen MEMS-kiihtyvyysanturi että 3-akselinen gyroskooppi eli kulmanopeusanturi (kuva 1). Inertia-termi viittaa kappaleen pyrkimykseen jatkaa kulkuansa, jos kappaleeseen ei kohdistu ulkoisia voimia. Kiihtyvyysanturit perustuvatkin tähän periaatteeseen, kuten myös osa gyroista. Kolmen akselin kulmanopeuden ja kolmen akselin kiihtyvyysantureiden avulla voi helposti mitata ja kontrolloida liikkeet kolmiulotteisessa tilassa. ADIS16360-inertia-anturi mittaa liikkeen kaikki kuusi vapausastetta.

Kuva 1. 6-akselinen ADIS16360-liikeanturi [1]

ADIS16360-IMU (inertial measurement unit) on Analog Devicesin suunnittelema älykäs anturituoteperhe, joka on 6-akselinen inertia-anturi (six-degrees-of-freedom, 6 DOF). ADIS16360 tarjoaa korkean suorituskyvyn, yksinkertaisuuden, nopeamman vasteajan ja vähentää virrankulutusta muihin IMU-laitteisiin verrattuna.

ADIS16360 tekee monimutkaisen liikkeentunnistuksen huomattavasti helpommaksi. Uusi IMU vähentää virrankulutusta vähintään 20 prosenttia, pienentää käynnistysaikaa noin 10-osaan ja bias-stabiilisuutta jopa 50 prosenttia, alentaa kohinaa, jotka vaikuttavat esimerkiksi navigoinnin tarkkuuteen.

Kiihtyvyysanturia voidaan käyttää moniin eri tarkoituksiin, ei pelkästään kiihtyvyyden mittaamiseen. Kiihtyvyysanturilla voidaan mitata mm. nopeutta, matkaa ja asentoa.

ADIS16360:aa voidaan käyttää sellaisten liikkeiden ohjaukseen, jotka vaativat korkean suorituskyvyn. Merkittävimmät käyttökohteet ovat ajoneuvoissa, jotka ovat riippuvaisia GPS-järjestelmästä, “high-end”-laitteissa, kameroissa tai tehtaan robotin käsivarressa.

Laite sisältää useita ainutlaatuisia ominaisuuksia, jotka edelleen vähentävät suunnitteluun kuluvaa aikaa, kuten automaattista anturin referenssipisteen uudelleenjärjestelyä, skaalausta ja itsetestausta.

ADIS16360:n ominaisuuksia ovat mm. nopeampi tiedonsiirto, kiihtyvyysanturin laajempi dynamiikka-alue (18 g) ja laajempi lämpötila-alue, -40…+105 °C.

2 INERTIA-ANTURIT

2.1 Kiihtyvyys

Kiihtyvyys on fysikaalinen suure, joka kuvaa kappaleen nopeuden muutosta tietyssä ajassa. Sen yksikkö on SI-järjestelmässä m/s². Nopeus voi muuttua, jos sen suuruus tai suunta muuttuu. Kiihtyvyydellä on aina suunta ja suuruus, se on siis vektorisuure. Jos nopeuden suunta ja etumerkki ovat samat kuin kiihtyvyyden, kappaleen vauhti kasvaa. Vastaavasti, jos nopeuden ja kiihtyvyyden suunta on sama, mutta etumerkki on eri, kappaleen vauhti hidastuu. Kappaleen kiihtyvyys määritellään matemaattisesti nopeuden ensimmäisenä ja matkan toisena derivaattana ajan suhteen.

$$a = \frac{dv}{dt} = \frac{d^2 x}{d t^2}, \tag{1}$$

jossa a = kiihtyvyys, v = nopeus, x = matka ja t = aika.

Jos nopeuden muutos samanpituisissa ajanpätkissä on yhtä suuri, kiihtyvyys on tasaista. Tasaisessa liikkeessä (v = vakio) kiihtyvyys on nolla ja tasaisesti kiihtyvässä liikkeessä kiihtyvyys on vakio. Putoamiskiihtyvyydellä maan pinnalla on oma tunnuksensa, joka on g (n. 9,81 m/s²).

2.2 Newtonin toinen laki

Newtonin toisen lain mukaan kappaleeseen vaikuttava voima aiheuttaa kappaleeseen kiihtyvyyden, jonka suunta on voiman suunta ja suuruus on suoraan verrannollinen voimaan ja kääntäen verrannollinen kappaleen massaan.

$$a = F/m \tag{2}$$

tai

$$F = ma, \tag{3}$$

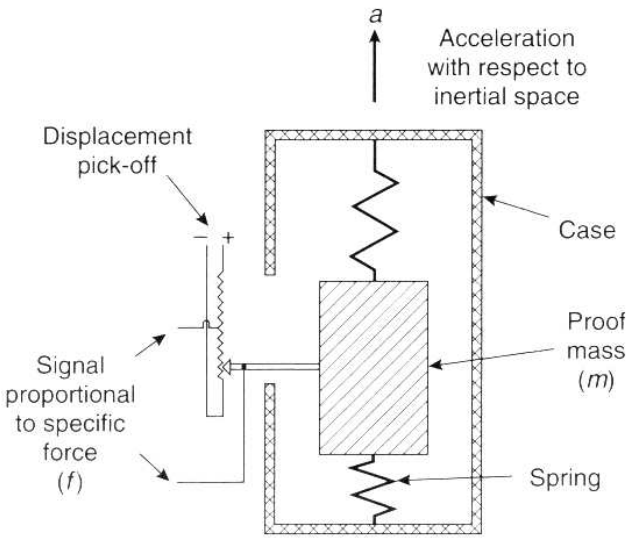
missä F = voima, m = massa ja [F] = kgm/s² = N (Newton).

2.3 Inertia-anturin toimintaperiaate

Newtonin toiseen lakiin perustuvia antureita nimitetään inertia-antureiksi. Newtonin toiseen lakiin perustuvilla antureilla on yksi yhteinen ominaisuus, joka selittää niiden suosion liikeantureina: liikkeen mittaus ei vaadi vertailupistettä. Ne toimivat kaikissa ympäristöissä ilman mekaanista, optista tai muuta linkkiä ympäristöön.

Kiihtyvyysanturin toimintaperiaate on sangen yksinkertainen. Yleisesti anturissa on massa, johon liikkeestä kohdistuva voima mitataan. Sen toiminta perustuu Newtonin toiseen lakiin (3). Anturi täytyy kalibroida maan vetovoiman suhteen, muuten se näyttää 9,81 m/s^2 kiihtyvyyttä ylöspäin. Anturi mittaa voimien vaikutusta kolmen eri akselin suhteen. Akselit ovat X, Y ja Z. Antureita voidaan valmistaa käyttötarkoituksen mukaan mittaamaan joko yhden, kahden tai kolmen akselin suuntaan.

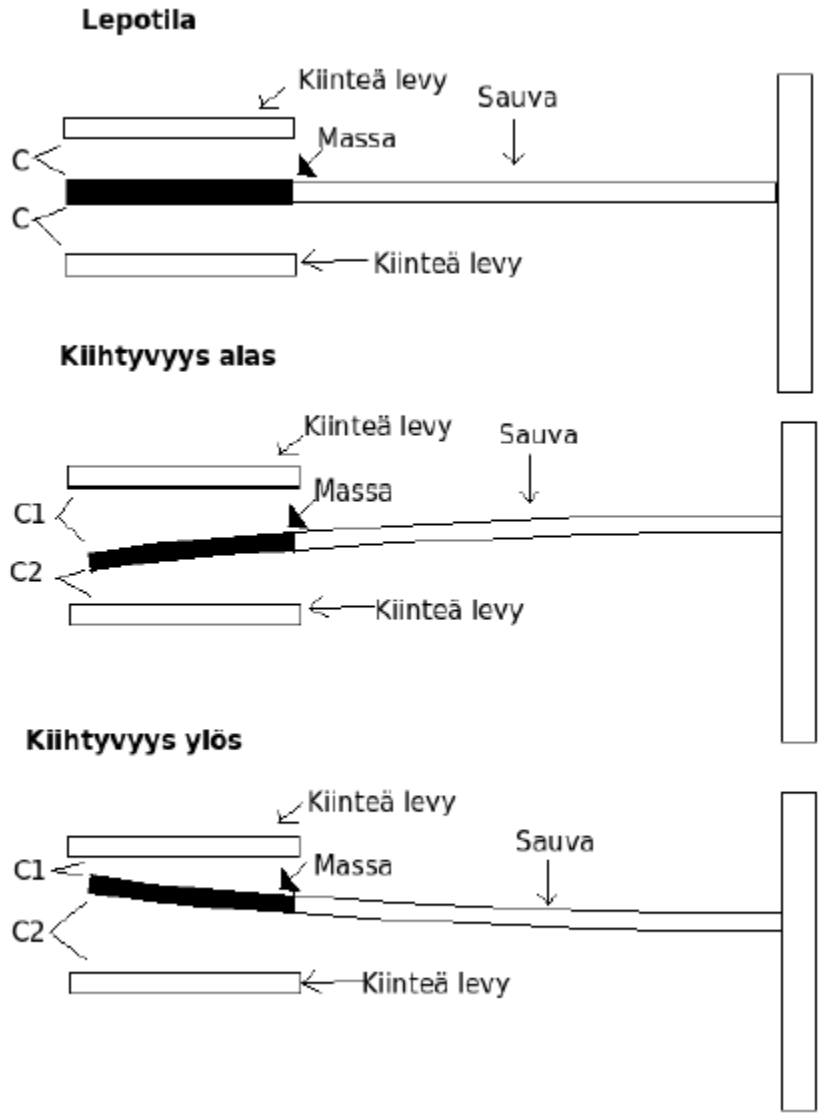
Kiihtyvyysanturin rakentamiseen tarvitaan massa, jousi, kotelo ja jonkinlainen indikaattori massan paikasta kotelon suhteen (kuva 2).



Kuva 2. Kiihtyvyysanturin toimintaperiaate [2]

Kiihtyvyysantureita voidaan valmistaa toimimaan usealla eri tekniikalla. Hyvin yleisesti käytetään kapasitiivisia kiihtyvyysantureita niiden luotettavuuden ja edullisuuden vuoksi. Kapasitiivinen anturi voidaan tehdä siten, että kahden elektrodilevyn väliin laitetaan piisauva, jonka päähän on kiinnitetty lisämassa voimistamaan kiihtyvyyden aiheuttamaa sauvan taipumaa. Kiihtyvyyden muutos saa siis sauvan taipumaan lähemmäksi toista elektrodilevyä.

Tästä aiheutuva kapasitanssin muutos voidaan mitata ja muuttaa jännitetiedoksi. Kuvassa 3 on havainnollistettu kapasitiivisen kiihtyvyysanturin toimintaperiaatetta.

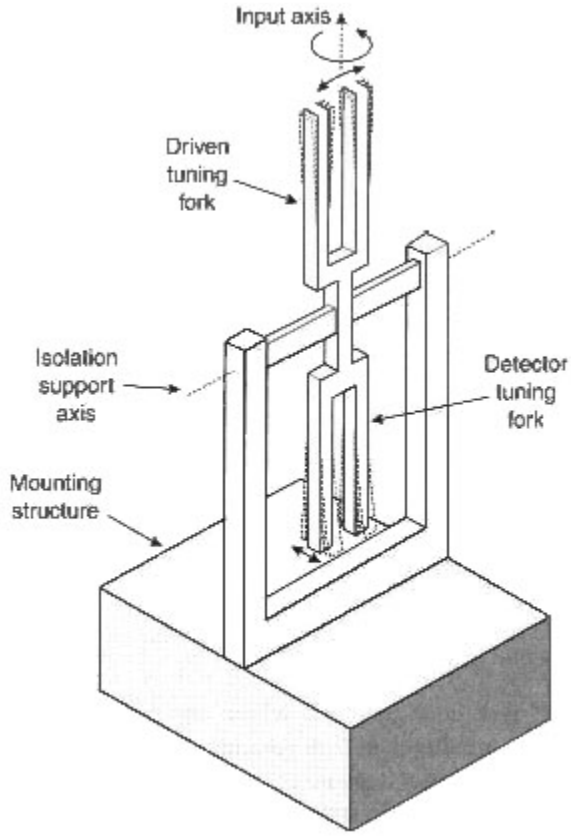


Kuva 3. Kapasitiivisen kiihtyvyysanturin toimintaperiaate [3]

Kulmanopeuden mittaamiseen on useita eri vaihtoehtoja. Mekaaniset gyrot perustuvat myös inertiaan, tällä kertaa liikkuvan anturielementin pyrkimyksiin vastustaa kiertoja. Pyörimisliikkeen anturi, gyro, perustuu Newtonin toiseen lakiin. Kun laki kirjoitetaan pyörivälle liikkeelle, tulee mukaan lisää voimatermejä. Erästä niistä kutsutaan Coriolis-voimaksi. Coriolis-voima esiintyy, kun massalla on alkunopeus ja sen lisäksi se saatetaan pyörivään liikkeeseen. Massa reagoi merkillisesti: pyörimisliikettä vastustaa voima, joka on kohtisuorassa pyörimisakseliin sekä alkuperäiseen nopeusvektoriin. Herkässä gyrossa tarvitaan suuri alkunopeus, joka perinteisesti saadaan pyörivällä hyrrällä. Mikromekaniikalla

ei voi toteuttaa hyviä laakereita; hyrrä on siis mahdoton. Sen sijaan alkuliikkeenä käytetään edestakaista värähtelyä. Mitattava pyörimisliike kytkee värähtelyliikettä alkuperäisestä suunnasta kohtisuoraan suuntaan.

Kuva 4 esittää "äänirautagyron" (tuning fork gyro) toimintaperiaatteen. Ylempi haarukka laitetaan resonoimaan sähkövirran avulla. Jos anturi pyörii sisäänmenoakselin ympäri, Coriolis-voima aiheuttaa edestakaisen liikkeen, joka on kohtisuorassa pakotetun resonoinnin suuntaan ja sisäänmenoakselin suuntaan nähden (nuolet alemmassa haarukassa). Tätä liikettä mitataan (yleensä kapasitiivisesti), ja tuloksena on kulmanopeudella moduloitu signaali, josta saadaan kulmanopeus selville. Mikromekaaniset (MEMS) gyrot toimivat tällä periaatteella.



Kuva 4. Esimerkki Coriolis-voimaan perustuvasta gyrosta [2]

3 SPI-VÄYLÄ

3.1 Yleistä

SPI (Serial peripheral interface) on alun perin Motorolan kehittämä synkroninen, kaksisuuntainen sarjasiirtoprotokolla, jota käytetään lyhyillä välimatkoilla, siis lähinnä piirikortin sisäiseen tiedonsiirtoon. Se on nykyään hyvin laajalti käytetty mm. AVR-mikro-ohjaimissa ja niiden oheispiireissä (ADC- ja DAC-muuntimet, muistipiirit).

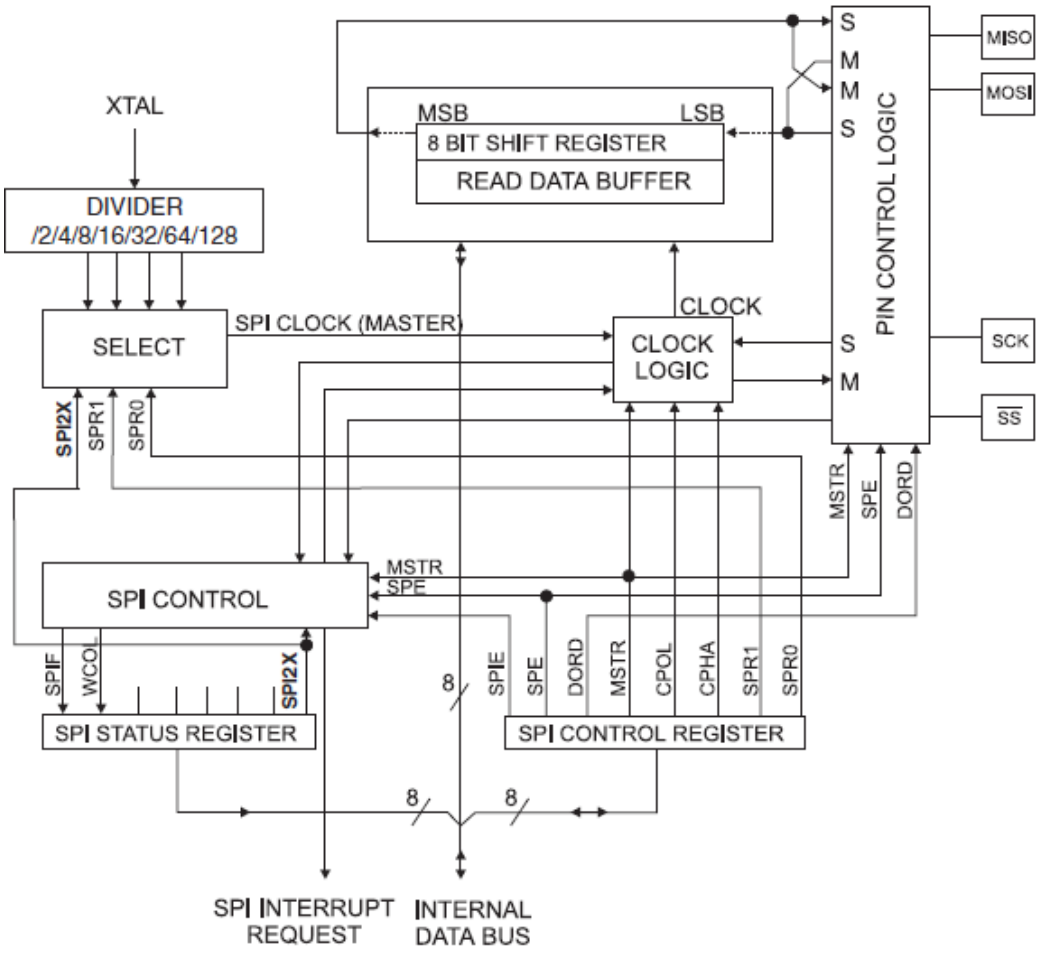
SPI on nopea ja yksinkertainen tapa siirtää tietoa mikro-ohjaimen ja oheispiirien välillä. Siinä ei käytetä osoitteita eikä kuittausta - se on vain helppo ja nopea tapa siirtää tietoa sarjamuotoisesti. Tiedonsiirto tapahtuu isäntä-orja-systeemillä ja on isännän hallitsemaa. Sekä lähettävässä että vastaanottavassa piirissä on 8-bittinen siirtorekisteri. AVR voidaan konfiguroida molempiin moodeihin (Master/Slave), mutta esim. jotkut anturit vain orja-moodiin.

Datan siirto tapahtuu full duplex -periaatteella, jossa dataa siirtyy molempiin suuntiin samanaikaisesti, eli kun isäntä siirtää orjalle dataa MOSI-linjalla, samalla orja siirtää isännälle dataa MISO-linjalla. Isäntä-laite määrittelee tiedonsiirtoparametrit. Data voidaan siirtää LSB- tai MSB-bitti ensimmäisenä. SPI määrittää vain dataliikennelinjat ja kellon. SPI-väylässä ei ole minkäänlaista rautapohjaista datavuonohjausta tai kuittausta, joten isäntä ei välttämättä saa koskaan tietää, vastaanottiko kukaan hänen lähettämäänsä viestiä.

Kun tavun siirto on valmis, ja jos keskeytysjärjestely on konfiguroitu, vastaanotettu merkki voidaan lukea talteen koneen muistiin.

3.2 ATmega128-prosessorin SPI-väylän toimintaperiaate

SPI-väylä koostuu neljästä linjasta: MOSI (Master Out, Slave In)-, MISO (Master In, Slave Out)-, SCK (Serial Clock)- ja alhaalla aktiivisesta SS (Slave Select)-linjoista. Viimeisestä käytetään myös nimitystä CS (Chip Select). Kuvassa 5 on esitetty ATmega128-prosessorin SPI-väylän lohkokaavio.



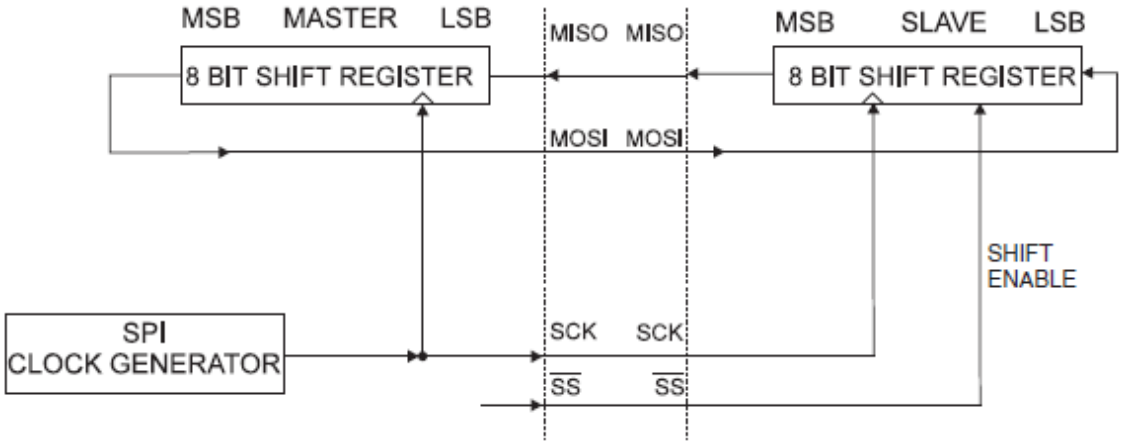
Kuva 5. ATmega128-prosessorin SPI-väylän toiminnan lohkokaavio [6]

SPI-tiedonsiirto koostuu kahdesta siirtorekisteristä ja isännän generoimasta kellosignaalista (kuva 6). Siirtorekisterit ovat usein kahdeksan bitin mittaisia, ja toinen niistä sijaitsee isännässä ja toinen orjassa. Ennen tiedonsiirron aloittamista isäntä laittaa siirtorekisteriin tavun, jonka haluaa lähettää orjalle. Tiedonsiirto alkaa, kun isäntä pakottaa valitun piirin SS-linjan alas. Tietoa siirretään isännältä orja-laitteelle MOSI-linjalla ja orja-laitteelta isännälle MISO-linjalla. Jokaisen kellojakson aikana yksi bitti siirtyy isännältä orjalle ja orjalta isännälle. Tieto siirtyy aina molempiin suuntiin, vaikka toisella laitteella ei olisikaan mitään hyödyllistä tietoa lähetettävänä. Rekisterin sisältö siirretään usein eniten merkitsevä bitti edellä, ja samalla kellojaksolla rekisteriin siirretään uusi vähiten merkitsevä bitti. Kun koko rekisterien sisältö on vaihdettu keskenään, isäntä nostaa SS-linjan takaisin ylös.

Ohjelmoijan on huolehdittava SS-linjan tilan muutoksista. Kun oheiskomponentti on valittu, tiedonsiirto aloitetaan kirjoittamalla data SPI:n datarekisteriin. Tällöin kellosignaali käynnistyy ja tieto siirretään orja-laitteen SPI-siirtorekisteriin eli kahdeksan kellopulssin

aikana datat on vaihdettu samanaikaisesti. Kun tavu on siirretty, kellosignaali loppuu ja haluttaessa tästä voidaan saada myös keskeytystieto. Riippuen konfiguroinnista tiedonsiirtoa voidaan nyt jatkaa tai se voidaan keskeyttää nostamalla SS-linja ylös. Vastaanotettu tavu tulee lukea ja siirtää muistiin talteen ennen kuin uutta siirretään.

Kun siirto on valmis, SPSR-rekisterissä oleva SPIF-bitti asetetaan 1-tilaan. Se aiheuttaa keskeytyksen, jos SPIE-bitti on asetettu 1-tilaan. Moodi hoidetaan MSTR-bitillä, joka on SPCR-rekisterissä.



Kuva 6. Liikennöinti isännän ja orja-laitteen välillä [6]

3.3 ATmega128-prosessorin SPI-rekisterit

SPI-tiedonsiirtoa varten tarvitaan kolme rekisteriä: SPI Status Register, SPI Control Register ja SPI Data Register (kuvat 7−9). Data kirjoitetaan ja luetaan data-rekisteristä, tiedonsiirto käynnistetään Control-rekisteristä.

Bitti	7	6	5	4	3	2	1	0
NIMI	SPIF	WCOL	-	-	-	-	-	SP12X

Bitti	Nimi	Merkitys	Toiminto
7	SPIF	SPI Interrupt Flag	0: 1: serial transfer complete
6	WCOL	Write COLlision Flag	0: 1: if the SPDR is written during a data transfer
0	SP12X	Double SPI Speed Bit	0: MSB first, 1: SPI speed doubled

Kuva 7. SPI-väylän status-rekisteri ja sen bittien merkitykset [8]

Bitti	7	6	5	4	3	2	1	0
Nimi	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

Bitti	Nimi	Merkitys	Toiminto
7	SPIE	SPI Interrupt Enable when SPIF bit in SPSR register is set -an interrupt occures.	0: disable, 1: enable
6	SPE	SPI Enable	0: disable, 1: enable
5	DORD	Data Order, bittien järjestys siirrossa	0: MSB first, 1: LSB first
4	MSTR	Master/Slave Select If SS input is low level while MSTR is set, MSTR will be cleared and SPIF flag in SPSR will be set (interrupt occures). After this user has to re-enable MSTR to continue in master mode;	0: Slave SPI mode, 1: Master SPI mode
3	CPOL	Clock Polarity	
2	CPHA	Clock Phase	
1	SPR1	Clock rate selection (refer to table in datasheet);	
0	SPR0	Clock rate selection (refer to table in datasheet),	

Kuva 8. SPI-väylän control-rekisteri ja sen bittien merkitykset [8]

SPI-väylällä on neljä erilaista tiedonsiirtomoodia, jotka ovat kellon polariteetin ja vaiheen yhdistelmiä (taulukko 1). CPOL (Clock Polarity) määrittää kellon polariteetin. Kun CPOL=0, ollaan normaalissa tilassa eli kellosignaali alkaa 0-tilasta, ja kun CPOL=1, kellosignaali invertoidaan (eli laskeva reuna = nouseva reuna kellon vaihetta ajatellessa). CPHA (Clock Phase) eli kellon vaihe määrittää, luetaanko tieto väylästä kellopulssin laskevalla vai nousevalla reunalla, ja samoin sen, kirjoitetaanko väylään nousevalla vai laskevalla reunalla. Kun CPHA=0, luetaan nousevalla reunalla ja kirjoitetaan laskevalla reunalla.

Taulukko 1. SPI-moodit

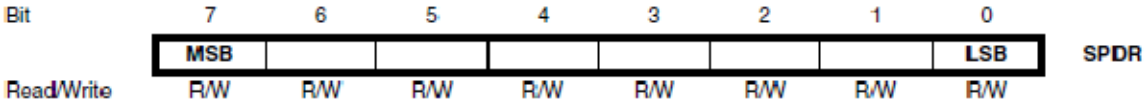
SPI-moodi	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

SPI-väylän nopeus on mahdollista valita jakajien avulla. Väylän kellotaajuus muodostetaan järjestelmäkellosta, joka jaetaan valitulla jakajalla, esim. 8 MHz:n järjestelmäkello ja 16:n jakaja muodostaa 500 kHz:n väylän. Jakajia on väliltä 4−128. Jotta väylä voisi toimia, on matalan ja korkean tilan kestettävä vähintään 2 kellojaksoa (taulukko 2).

Taulukko 2. SPI-väylän nopeus valitaan jakajien avulla. [6]

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

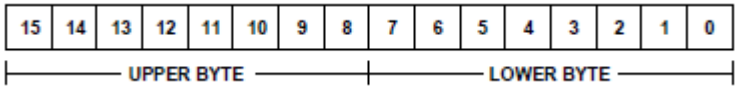
Useissa laitteissa tiedonlähetysjärjestys (DORD, data order) on myös valittavissa. Tällä asetuksella määritetään, aloitetaanko tiedonsiirto vähiten vai eniten merkitsevästä bitistä. Kun DORD=0, eniten merkitsevä bitti lähetetään ensin.



Kuva 9. SPI-väylän data-rekisteri [6]

3.4 ADIS16360-liikeanturin ja ATmega128-prosessorin SPI-väylän kommunikointi

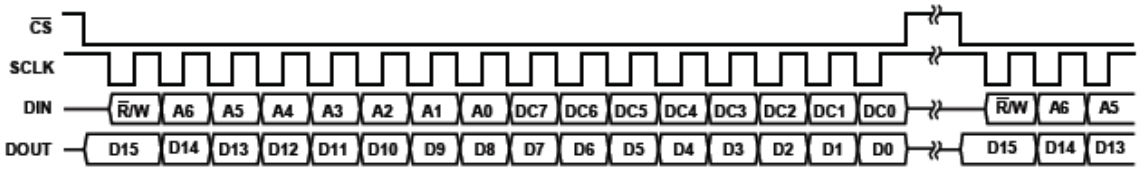
ADIS16360 on rakenteeltaan yksinkertainen ja se voidaan integroida teollisuuden järjestelmiin. Se vaatii vain 5,0 V:n jännitteensyötön ja neljä johdinta eli SPI-väylän. Kaikki lähdöt ja ohjelmoitava toiminnot hoidetaan yksinkertaisella rekisterin rakenteella. Kukin rekisteri on 16-bittinen (kuva 10). Kussakin rekisterissä on oltava ylempi tavu (bitti 8…bitti 15) ja alempi tavu (bitti 0…bitti 7), joilla kullakin on oma 7-bittinen osoite, mutta jompikumpi voidaan käyttää koko 16 bitin datan lähettämiseen.



Kuva 10. ADIS16360-liikeanturin rekisterit ovat 16-bittisiä. [5]

ADIS16360-liikeanturin SPI-väylä sisältää neljä signaalia: piirinvalinta (CS), sarjakello (SCLK), tulosignaali (DIN), ja lähtösignaali (DOUT). CS-linja mahdollistaa ADIS16360 SPI-väylän. Kun tämä signaali on high-tilassa, DOUT-linja on "high-impedance"-tilassa (kytkeytyy irti väylästä) ja signaaleilla DIN ja SCLK ei ole vaikutusta toimintaan. Yksi täydellinen datakehys on 16 kellopulssia. Koska SPI-väylä toimii full duplex -periaatteella, se vastaanottaa (DIN) ja lähettää (DOUT) 16 bittiä samanaikaisesti.

Tässä sovelluksessa kytkettiin SPI-dataväylän linjat eli SS-, MOSI-, MISO- ja SCLK- sekä VCC- ja GND-linjat STK500-kehitysalustasta anturin piirilevylle. Myös anturin RESET-linja kytkettiin +5 V:iin. Anturin maksiminopeus SPI-väylällä on 2 MHz. 8 MHz:n ulkoista kidettä käyttäen kellotaajuudeksi saatiin 500 kHz, joka oli mittauksiin sopiva (jakaja = 16). Malli SPI-väylän kommunikoinnista on esitetty kuvassa 11.



Kuva 11. SPI-väylän kommunikointi [5]

Vaikka ADIS16360 tuottaa tietoa itsenäisesti, se toimii SPI-väylän orjalaitteena, joka kommunikoi järjestelmän prosessorin kanssa. SCLK-linjaan tulee SPI-linjassa aina 16 kellopulssia, jotka prosessori generoi linjalle. CS on piirinvalintalinja. Se tulee olla 0-tilassa aina, kun anturin kanssa kommunikoidaan. 16 kellopulssin jälkeen CS on aina asetettava 1-tilaan.

3.4.1 Liikeanturin rekisterin lukeminen

Yksittäisen rekisterin lukeminen vaatii kaksi 16-bittistä sekvenssiä. Ensimmäinen 16-bittinen sekvenssi sisältää lukukomennon bitti (R/W = 0) ja seitsemän bittiä kohderekisterin osoitteen (A6−A0), viimeiset kahdeksan bittiä ovat "don't care" -bittejä (DC7−DC0), kun kyseessä on lukeminen. Toinen 16-bittinen sekvenssi siirtää rekisterin sisällön (D15-D0) DOUT-linjassa. Esimerkiksi jos DIN = 0x0A00, XACCL_OUT sisältö siirretään ulos DOUT-linjassa seuraavassa 16-bittisessä jaksossa. Tässä esimerkissä 0x0A on rekisterin osoite ja 00 on DC7−DC0:n arvot.

SPI-väylä on kaksisuuntainen, mikä tarkoittaa, että isäntäprosessori voi lukea outputdataa DOUT-linjasta ja lähettää seuraavan kohdeosoitteen DIN-linjan kautta, kun käyttää samaa SCLK-pulssia.

3.4.2 Liikeanturin rekisteriin kirjoittaminen (laitekonfigurointi)

Tässä tapauksessa ensimmäinen bitti DIN-ketjussa on 1 (R/W = 1), sen jälkeen 7-bittinen osoite (A6−A0) ja 8-bittinen data (DC7−DC0).

Koska jokainen kirjoituskomento kattaa yhden tavun datan, tarvitaan kaksi datakehystä, kun halutaan kirjoittaa koko 16-bittinen rekisterin tila.

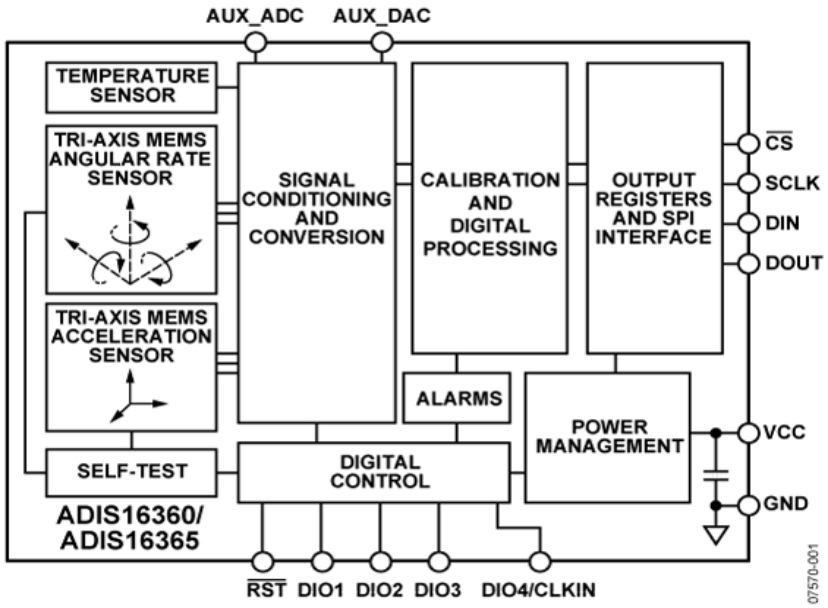
Esimerkiksi, jos halutaan kirjoittaa data 0x1F = 0000 1111 XACCL_OFF rekisterin ylempään tavuun, jonka osoite on 0x21 = 0010 0001, on kyseessä kirjoitustapahtuma, joten R/W = 1. Nyt DIN-linjaan on kirjoitettava 1010 0001 0000 1111 eli DIN = 0xA11F.

4 ADIS16360-LIIKEANTURIN TOIMINTA

4.1 ADIS16360-IMU:n (Inertial Measurement Unit) teknisiä ominaisuuksia

ADIS16360-liikeanturi on Analog Devicesin suunnittelema liikeanturi. Se on kolmen ulottuvuuden kiihtyvyys- ja kulmanopeusanturi, joka tarjoaa suuren tarkkuuden ja luotettavuuden pienessä ja edullisessa paketissa. Komponentissa on myös lämpötila-anturi.

Analog Devices on valmistanut myös ADIS16365-liikeanturin, jonka kiihtyvyys- ja kulmanopeusanturin ominaisuudet ovat hyvin samanlaiset, mutta lämpötilaominaisuuksia on parannettu. Kuvassa 12 voidaan nähdä anturin ADIS16360/ADIS16365 toiminnallinen lohkokaavio.



Kuva 12. ADIS16360/ADIS16365:n toiminnallinen lohkokaavio [5]

Kolmen akselin kulmanopeuden ja kolmen akselin kiihtyvyysantureiden avulla voi helposti mitata ja kontrolloida liikkeet kolmiulotteisessa tilassa. Komponentti mittaa kiihtyvyyksiä arvoon ±18 g asti, missä yksi "g" merkitsee maan vetovoiman kiihtyvyyttä. ADIS16360 pystyy mittaamaan myös kulmanopeuksia arvoon ±75°/s, ±150°/s tai ±300°/s saakka.

ADIS163-liikeanturin 3 dB:n kaistanleveys on 330 Hz. Kulmanopeusanturin lähdön kohina on 0,8°/s rms (rms = tehollisarvo) ja kohinatiheys on 0,044°/s/$\sqrt{Hz}$ rms. Nämä arvot kiihtyvyysanturille ovat 9 mg ja 0,5 mg/$\sqrt{Hz}$ rms.

Kiihtyvyys- ja kulmanopeusanturin resoluutio on 14 bittiä, mutta lämpötila-anturin ja ADC-muuntimen resoluutio on 12 bittiä.

Liikeanturin syöttöjännitealue on 4,75…5,25 V, käyttölämpötila-alue -40…+105 °C ja sen kiihtyvyysanturin iskunkestävyys on 2000 g. ADIS16360 kuluttaa 49 mA normaalitilassa, 24 mA virransäästötilassa ja 500 μA lepotilassa.

Anturi sisältää ohjelmoitavat itsetestaus-, virranhallinta- ja hälytinyksiköt. Kaikki komennot ja tiedot kommunikoivat SPI-väylän kautta.

Komponentilla on 24 pinniä, ja sen mitat ovat 23 mm × 23 mm × 23 mm (pituus × leveys × korkeus).

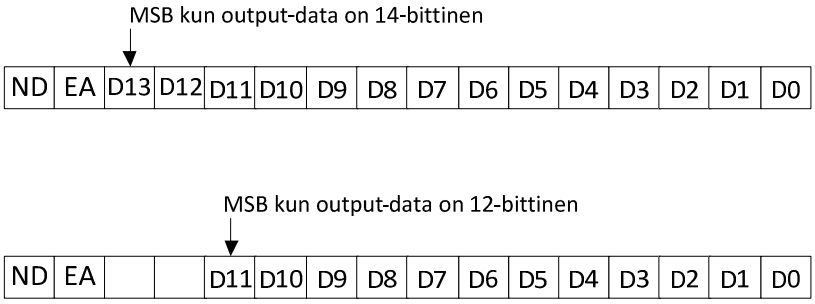
ADIS16360 on itsenäinen anturijärjestelmä, joka käynnistyy ja tuottaa mittaustiedot tehdasasetuksessa, kun on olemassa sopiva jännite. Sen oletusnäytteenottotaajuus on 819,2 SPS (Samples Per Second), ja jokaisen näytteen syklin jälkeen anturitietoa ladataan output-rekistereihin.

4.2 Output-datarekisterit

ADIS16360-liikeanturin kaikki rekisterit ovat 16-bittisiä. Jokaisella on ylempi tavu (bitti 8…bitti 15) ja alempi tavu (bitti 0…bitti 7). Myös jokaisella tavulla on oma 7-bittinen osoite. Ylempi tavun osoite voidaan saada kaavasta 4.

Ylempi tavun osoite = alempi tavun osoite + 1 (4)

Output-datarekisterien bittikartta ja muodot on esitetty kuvassa 13 ja taulukossa 3. Taulukossa mainitut osoitteet ovat alemman tavun osoite ja kulmanopeusanturin skaalaus on asetettu ±300°/s.



Kuva 13. Output-datarekisterin bittimäärittely

Kun output-datarekisterit päivitetään uusilla tiedoilla, ND-bitti menee 1-tilaan, jos rekisteri sisältää lukematonta tietoa. Sen jälkeen, kun tiedot on luettu, se palaa 0-tilaan. EA-bitti on 1, kun virhelippu/hälytyslippu DIAG_STAT-rekisterissä on 1. EA-bittiä käytetään osoittamaan järjestelmän virhettä tai hälytystä, joka voi johtua useasta asiasta, kuten jännitelähteestä, joka on määritellyn toiminta-alueen ulkopuolella.

Taulukko 3. Output-datarekisterien muodot

Rekisterin nimi	Osoite	Bitti	Skaala	Rekisterin kuvaus
SUPPLY_OUT	0x02	12	2,418 mV	jännitelähteen mittaus
XGYRO_OUT	0x04	14	0,05°/s	X-akselin kulmanopeusanturin lähtö
YGYRO_OUT	0x06	14	0,05°/s	Y-akselin kulmanopeusanturin lähtö
ZGYRO_OUT	0x08	14	0,05°/s	Z-akselin kulmanopeusanturin lähtö
XACCL_OUT	0x0A	14	3,333 mg	X-akselin kiihtyvyysanturin lähtö
YACCL_OUT	0x0C	14	3,333 mg	Y-akselin kiihtyvyysanturin lähtö
ZACCL_OUT	0x0E	14	3,333 mg	Z-akselin kiihtyvyysanturin lähtö
XTEMP_OUT	0x10	12	0,136 °C	X-akselin kulmanopeusanturin lämpötilan mittaus
YTEMP_OUT	0x12	12	0,136 °C	Y-akselin kulmanopeusanturin lämpötilan mittaus
ZTEMP_OUT	0x14	12	0,136 °C	Z-akselin kulmanopeusanturin lämpötilan mittaus
AUX_ADC	0x16	12	805,8 µV	ADC-muuntimen lähtö

Outputdata on joko 12 bittiä tai 14 bittiä pitkä. Kaikki 12-bittinen lähtö datalle, bitit D12 ja D13 ovat "don't care" (ei ole merkitystä). Taulukosta 3 nähdään, että kiihtyvyysanturien ja kulmanopeusanturien databittejä on 14, mutta lämpötila-anturin ja ADC-muuntimen databittejä on 12.

Kaikki inertia-anturien lähdöt ovat 14 bittiä pitkiä ja ovat kahden komplementtimuodossa, mikä tarkoittaa, että 0x0000 on yhtä suuri 0 LSB, 0x0001 on yhtä suuri +1 LSB, ja 0x3FFF

on yhtä suuri −1 LSB (taulukko 4). Seuraavassa on esimerkki siitä, miten voidaan laskea X-akselin kulmanopeusanturin lähdön mittauksen:

XGYRO_OUT = 0x3B4A

Bitti D13 on 1, eli luku on negatiivinen:

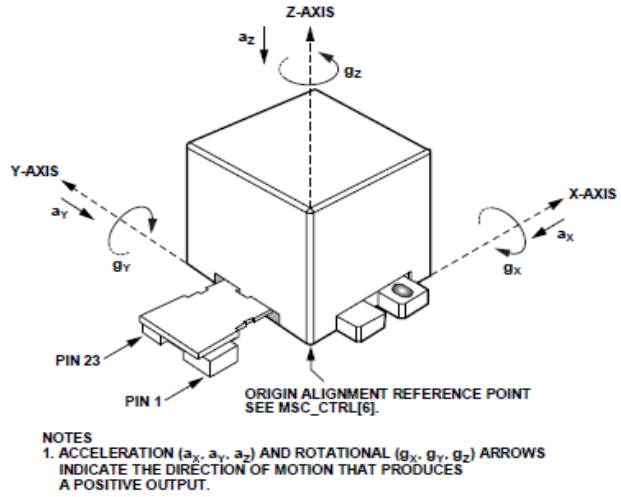
0x000 − 0x33B4A = −0x04B6 = −(4 × 256 + 11 × 16 + 6) = −1206 LSB

Kulmanopeus = 0,05°/s × (−1206) = −60,3°/s

Siis kun XGYRO_OUT-ulostulo on 0x3B4A, vastaa 60,3°/s myötäpäivään kierto z-akselin suunnassa (ks. kuva 14), kun katsotaan anturin yläosaan.

Taulukko 4. Kulmanopeuden laskeminen eri lukujärjestelmissä

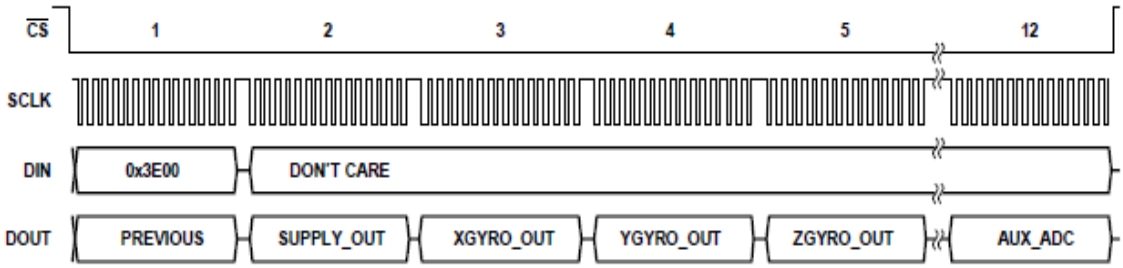
Kulmanopeus	Desimaali	Heksadesimaali	Binaari
+300°/s	+6000 LSB	0x1770	XX01 0111 0111 0000
+0,1°/s	+2 LSB	0x0002	XX00 0000 0000 0010
+0,05°/s	+1 LSB	0x0001	XX00 0000 0000 0001
0°/s	0 LSB	0x0000	XX00 0000 0000 0000
−0,05°/s	−1 LSB	0x3FFF	XX11 1111 1111 1111
−0,1°/s	−2 LSB	0x3FFE	XX11 1111 1111 1110
−300°/s	−6000 LSB	0x2890	XX10 1000 1001 0000



Kuva 14. Aksiaaliset suunnat [5]

4.3 Purskemoodi-tiedonkeruu (Burst mode)

Purskemoodi-tiedonkeruu tarjoaa menetelmän, jonka avulla voidaan hyödyntää enemmän prosessin suorituskyvystä ADIS16360-anturin tiedonkeruussa (kuva 15).



Kuva 15. Purske-moodi tiedonkeruu

13 peräkkäisessä datajaksossa (jokainen on erotettu yhden SCLK-jakson kautta) siirretään kaikki kaksitoista outputrekisteriä ulos DOUT-lähdössä. Tämä sarja alkaa, kun DIN on 0011 1110 0000 0000 (DIN = 0x3E00). Sen jälkeen kunkin outputrekisterin sisältö on DOUT:ssa, alkaen SUPPLY_OUT-rekisteristä ja päättyy AUX_ADC-rekisteriin.

4.4 Anturin kalibrointi

Sovelluksiin, jotka vaativat anturin kalibrointia, ADIS16360-liikeanturi tarjoaa bias offset -rekistereitä kaikille kuudelle kulmanopeus- ja kiihtyvyysanturille (taulukko 5).

Taulukko 5. Bias offset -rekisterit

Rekisteri	Osoitteet	Yhteiset parametrit
XGYRO_OFF	0x1B, 0x1A	Oletusarvo = 0x0000
YGYRO_OFF	0x1D, 0x1C	Skaala = 0,0125°/s per LSB
ZGYRO_OFF	0x1F, 0x1E	read/write
XACCL_OFF	0x21, 0x20	Oletusarvo = 0x0000
YACCL_OFF	0x23, 0x22	Skaala = 3,333 mg/LSB
ZACCL_OFF	0x25, 0x24	read/write

ADIS16360-anturin kalibroimiseksi on olemassa neljä vaihtoehtoa: manuaalinen bias kalibrointi, gyroskoopin automaattinen bias nollaus, tehdasasetusten palautus ja lineaarinen kiihtyvyyden biaskorvaus.

Taulukoissa 6 ja 7 on esitetty bias offset -rekisterien bittimäärittely. Näiden bittien avulla voidaan säätää jokainen anturi manuaalisesti. Esimerkiksi jos XGYRO_OFF = 0x1FF6 (DIN = 0x9B1F, 0x9AF6), XGYRO_OUT:n offset siirtyy −10 LSB tai −0,125 °/s. Tämän kalibroinnin toteutumiseksi kirjoitetaan 0x1F osoitteeseen 0x1B (1001 1011 0001 1111 = 0x9B1F) ja 0xF6 osoitteeseen 0x1A (1001 1010 1111 0110 = 0x9AF6). Ensimmäinen bitti on 1, koska kyseessä on kirjoitustapahtuma.

Taulukko 6. Kulmanopeusanturin bias offset -rekisterien bittimäärittely

XGYRO_OFF, YGYRO_OFF, ZGYRO_OFF	
Bitit	Kuvaus
[15:13]	Ei käytetä
[12:0]	Databitit

Taulukko 7. Kiihtyvyysanturin bias offset -rekisterien bittimäärittely

XACCL_OFF, YACCL_OFF, ZACCL_OFF	
Bitit	Kuvaus
[15:12]	Ei käytetä
[11:0]	Databitit

Command-rekisterin avulla voidaan kalibroida kaikki kolme kulmanopeusanturia automaattisesti. Kun kirjoitetaan 0x01 osoitteeseen 0x3E eli GLOB_CMD [0] = 1 (DIN = 0xBE01), toteuttaa se automaattisen biasnollaus-kalibrointitoiminnon. Tämä toiminto mittaa kaikki kolme gyroskoopin outputrekisteriä ja lataa sitten jokaiseen rekisteriin vasta-arvon. Sen jälkeen kaikki antureiden tiedot asetetaan nolliksi, ja flash-muisti päivittyy automaattisesti 50 ms:n aikana.

Tehdasasetusten palautusta varten on kirjoitettava 0x02 osoitteeseen 0x3E eli GLOB_CMD [1] = 1 (DIN = 0xBE02). Resetointi kestää 50 ms, ja resetoinnin jälkeen kaikki laitteen hallinnasta tehdyt asetukset palautuvat oletusarvoihin.

4.5 Toiminnan hallinta

4.5.1 Globaaliset komennot

ADIS16360-liikeanturi tarjoaa globaalisia komentoja yleisiä toimintoja varten, kuten kalibrointia, flash-päivitystä ja ohjelmiston resetointia. Kaikilla globaalisella komennoilla on yksi ohjausbitti command-rekisterissä (taulukko 8). Jokainen toiminta aloitetaan, kun kirjoitetaan sen ohjausbitti ykköseksi ja ohjausbitti palaa nollaksi täydentämisen jälkeen.

Taulukon 8 mukaan tehdasasetusten palautusta varten GLOB_CMD-rekisterin toinen bitti asetetaan 1-tilaan eli data 0x02 kirjoitetaan GLOB_CMD-rekisterin osoitteeseen 0x3E, mutta on huomattava, että kyseessä on kirjoitustapahtuma, joka vaatii R/W = 1 (1011 1110 = 0xBE) ja DIN = 0xBE02.

Ohjelmiston resetointia varten GLOB_CMD[7] on asettava 1-tilaan eli data on 0x08. Nyt DIN-linjaan on kirjoitettava 0xBE80. Sen jälkeen anturin toiminta pysähtyy ja anturi käynnistetään uudelleen.

GLOB_CMD-rekisterin lukeminen (DIN = 0x3E00) alkaa purskemoodissa. GLOB_CMD rekisteri on 16-bittinen, mutta vain alempi tavu on käytettävässä.

Taulukko 8. GLOB_CMD-rekisterin bittimäärittely

Bitit	Kuvaus (Oletusarvo=0x0000)
[15:8]	Ei käytetä
[7]	Ohjelmiston resetointikomento
[6:5]	Ei käytetä
[4]	Tarkka automaattinen nollauskomento
[3]	Flash-päivityskomento
[2]	DAC data latch
[1]	Tehdasasetusten palautuskomento
[0]	Automaattinen nollauskomento

4.5.2 Sisäinen näytteenottotaajuus

Sisäinen näytteenottotaajuus määrittää, kuinka usein tiedot päivitetään. SMPL_PRD-rekisteri valvoo ADIS16360-anturin sisäistä näytteenottotaajuutta (taulukko 9). Näytteenoton aikajakso voidaan laskea seuraavasta kaavasta:

$$T_S = T_B \times (N_S + 1), \quad (5)$$

missä T_S = sisäinen näytteenoton aikajakso, T_B = aikakanta (time base) ja N_S = kerroin.

SMPL_PRD-rekisterin alemman tavun osoite on 0x36 ja ylemmän tavun osoite on 0x37. Rekisterin pienin oletusarvo on 0x01, joka vastaa maksiminäytteenottotaajuutta 819,2 näytettä sekunnissa (SPS). Rekisterin sisältö on haihtumaton (nonvolatile), eli sen tiedot eivät katoa, vaikka virta katkaistaan.

Taulukko 9. SMPL_PRD-rekisterin bittimäärittely

Bitit	Kuvaus (Oletusarvo=0x0000)
[15:8]	Ei käytetä
[7]	aikakanta; 0 = 0,61035 ms, 1 = 18,921 ms
[6:0]	kerroin

Esimerkiksi lasketaan näytteenottoaikajakso ja näytteenottotaajuus ADIS16360-liikeanturille, kun SMPL_PRD[7:0] = 0x0A:

SMPL_PRD[7:0] = 0x0A = 0000 1010

B7 = 0 → T_B = 0,61035 ms

B6…B0 = 000 1010 → N_S = 10

$T_S = T_B \times (N_S + 1)$ = 0,61035 ms × (10 + 1) = 6,71385 ms

$$f_S = \frac{1}{T_S} = \frac{1}{6{,}71385\ \text{ms}} = 149\ \text{SPS}$$

Näytteenottotaajuudella on suora vaikutus SPI-väylän nopeuteen. Kun SMPL_PRD-rekisterin arvo on pienempi tai yhtä suuri kuin 0x09 (normaalitila), SPI-väylän sarjakello

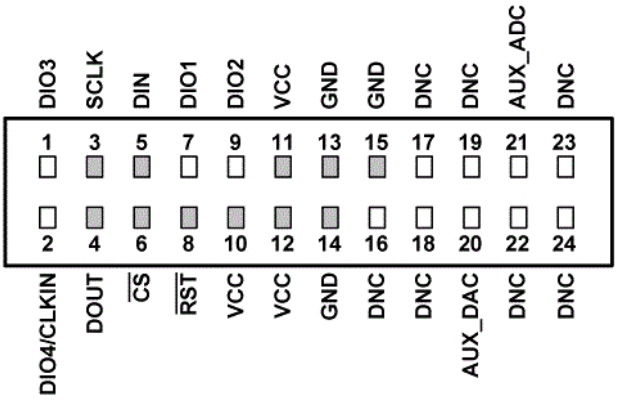
(SCLK) voi toimia enintään 2,0 MHz:n taajuudella. Jos rekisterin arvo on suurempi tai yhtä suuri kuin 0x0A (virransäästötila), SPI-väylän sarjakello voi ajaa enintään 300 kHz:n taajuudella.

Näytteenottotaajuuden asettaminen arvoon 819,2 SPS säilyttää anturin kaistanleveyden ja tarjoaa optimaalisen suorituskyvyn. Järjestelmille, jotka ovat hitaampia kuin näytenopeus, pidetään sisäinen näytteenottotaajuus arvossa 819,2 SPS ja käytetään ohjelmoitavaa suodatinta (SENS_AVG), joka pienentää kaistanleveyttä ja auttaa estämään aliasing-ilmiötä eli taajuuden laskostumista.

5 KYTKENNÄN SUUNNITTELU

5.1 ADIS16360-liikeanturin pinnit ja niiden toiminta

Anturilla on 24 pinniä (kuva 16), joiden toiminnan kuvaus on esitetty taulukossa 10. SPI-väylän kommunikointia varten käytetään SCLK-, DOUT-, DIN- ja CS-pinnejä. Lisäksi kolme VCC-pinniä ja RST-pinni kytketään +5 V:n käyttöjännitteeseen ja kolme GND-pinniä kytketään maahan. Kuvassa 16 ja taulukossa 10 SPI-väylän kommunikointia varten tarvittavat pinnit on esitetty harmaana.



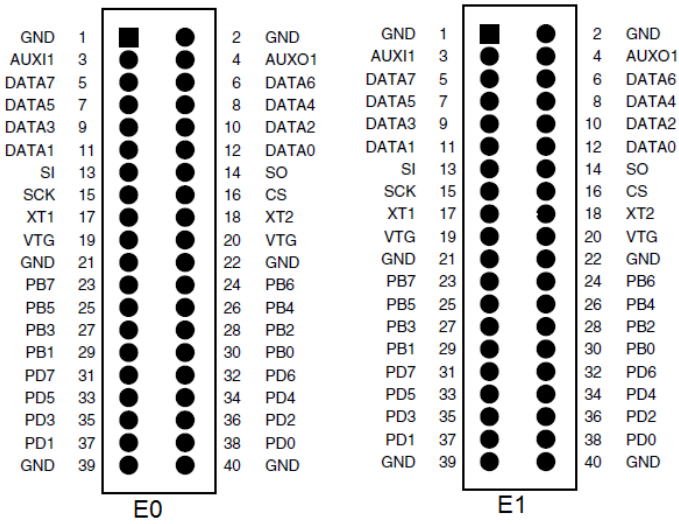
Kuva 16. ADIS16360-liikeanturin pinnit

Taulukko 10. ADIS16360-liikeanturin pinnien toiminnan kuvaus

Pinnin numero	Nimi	Kuvaus
1	DIO3	Konfiguroitava digitaalinen input/output
2	DIO4/CLKIN	Konfiguroitava digitaalinen input/output tai synkroninen kellon input
3	SCLK	SPI-väylän sarjakello
4	DOUT	SPI-väylän data output (sarjakellon laskevassa reunassa)
5	DIN	SPI-väylän data input (sarjakellon nousevassa reunassa)
6	CS	SPI-väylän piirinvalinta
7, 9	DIO1, DIO2	Konfiguroitava digitaalinen input/output
8	RST	Reset
10, 11, 12	VCC	Käyttöjännite
13, 14, 15	GND	Maa
16, 17, 18, 19, 22, 23, 24	DNC	Ei kytketä
20	AUX_DAC	DAC-muuntimen output
21	AUX_ADC	ADC-muuntimen input

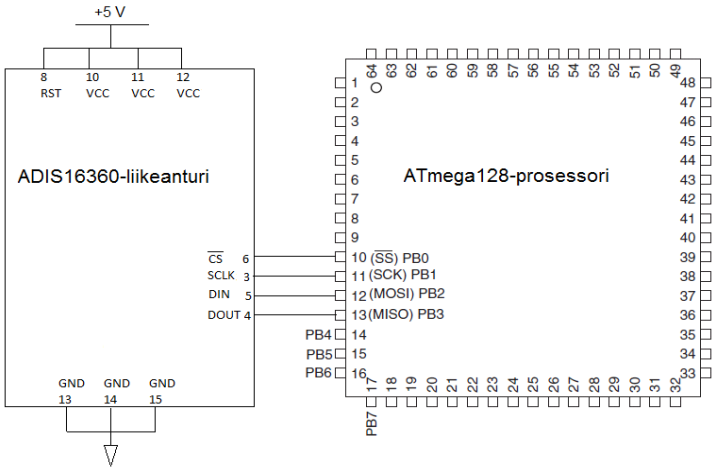
5.2 STK500-alustan kytkentä

STK500 on kehityspaketti Atmelin mikro-ohjaimille. Paketin avulla voidaan nopeasti ottaa käyttöön Atmelin AVR-mikro-ohjaimet. STK500-alustassa on kaksi laajennusliitäntää, joihin kaikki AVR I/O-portit, ohjelmointisignaalit ja ohjaussignaalit on reititetty. Laajennusliitännät on esitetty kuvassa 17.



Kuva 17. STK500-alustan laajennusliitännät [11]

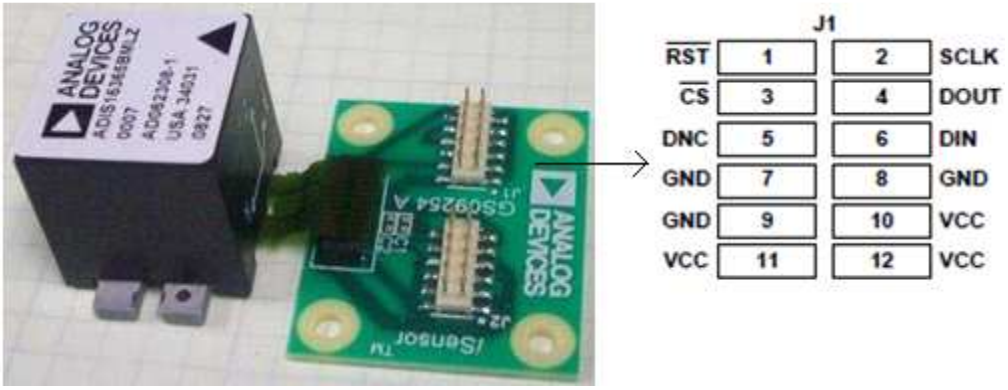
STK500-kehitysalustaan kytkettiin ATmega128L-prosessori. STK500 toimii 9−12 V DC-tason käyttöjännitteillä, mutta varsinaisissa testeissä käytettiin 9 V:n paristoa. ATmega128-prosessoria voidaan käyttää vaihtoehtoisesti SPI-väylän portissa B. Periaatteessa ADIS16360-anturi voidaan liittää suoraan ATmega128-prosessoriin (kuva 18).



Kuva 18. ADIS16360-anturin ja ATmega128-prosessorin SPI-väyläkommunikointi

5.3 Kytkennän rakentaminen

ADIS16360/PCBZ:n paketti sisältää yhden ADIS16360BMLZ-liikeanturin ja yhden liitäntäkortin (kuva 19). Liitäntäkortti yksinkertaistaa prosessia, jossa liikeanturi integroidaan prosessorin järjestelmään.

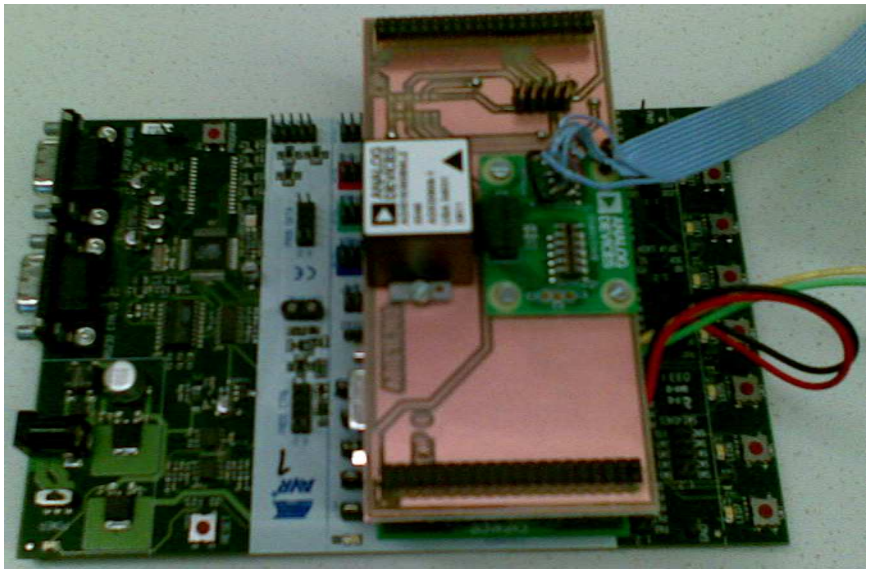


Kuva 19. ADIS16360/PCBZ ja J1-liittimen pinnijärjestely [10]

Anturille tehtiin piirilevy, jonka avulla kytkettiin ADI16360-anturin liitäntäkortti STK500-alustan laajennusliitäntöihin (kuva 20). Anturin linjat CS, SCLK, DIN, DOUT, GND, VCC ja RST kytketään taulukon 11 mukaan ATmega128-prosessorin SS-, SCLK-, MOSI-, MISO-, GND- ja VCC-linjoihin (portti B). On huomattava, että anturin RESET-linja (RST) kytketään +5 V:n käyttöjännitteeseen.

Taulukko 11. Anturin linjat CS, SCLK, DIN, DOUT, GND, VCC ja RST kytketään ATmega128-prosessorin SS-, SCLK-, MOSI-, MISO-, GND- ja VCC-linjoihin.

ADIS16360-liikeanturi		J1-liitin	STK500/laajennusliitännät		ATmega128	
Pinnin numero	Nimi	Numero	laajennusliitäntä	Numero	Pinni	Nimi
6	CS	3	E1	30	PB0	SS
3	SCLK	2	E1	29	PB1	SCLK
5	DIN	6	E1	28	PB2	MOSI
4	DOUT	4	E1	27	PB3	MISO
			E1	26	PB4	
			E1	25	PB5	
			E1	24	PB6	
			E1	23	PB7	
13, 14, 15	GND	7, 8, 9	E0	1	GND	
8; 10, 11, 12	RST/VCC	10, 11, 12	E0	21	VCC	



Kuva 20. Anturin piirilevy, joka on kytketty STK500-alustaan.

6 ANTURIN OHJELMOINTI

Ohjelman kirjoittamiseen ja kääntämiseen käytettiin ilmaiseksi saatavilla olevaa AVR Studio 4:ää ja AVR-GCC C-kääntäjää. Ohjelma kirjoitettiin C-kielellä. SPI-väylän nopeus on mahdollista valita jakajien avulla. Väylän kellotaajuus muodostetaan järjestelmäkellosta, joka jaetaan valitulla jakajalla. Tiedonsiirtoon käytettiin SPI-väylää 16:n jaolla, jolloin 8 MHz:n järjestelmäkellolla väylän nopeus oli 500 kHz.

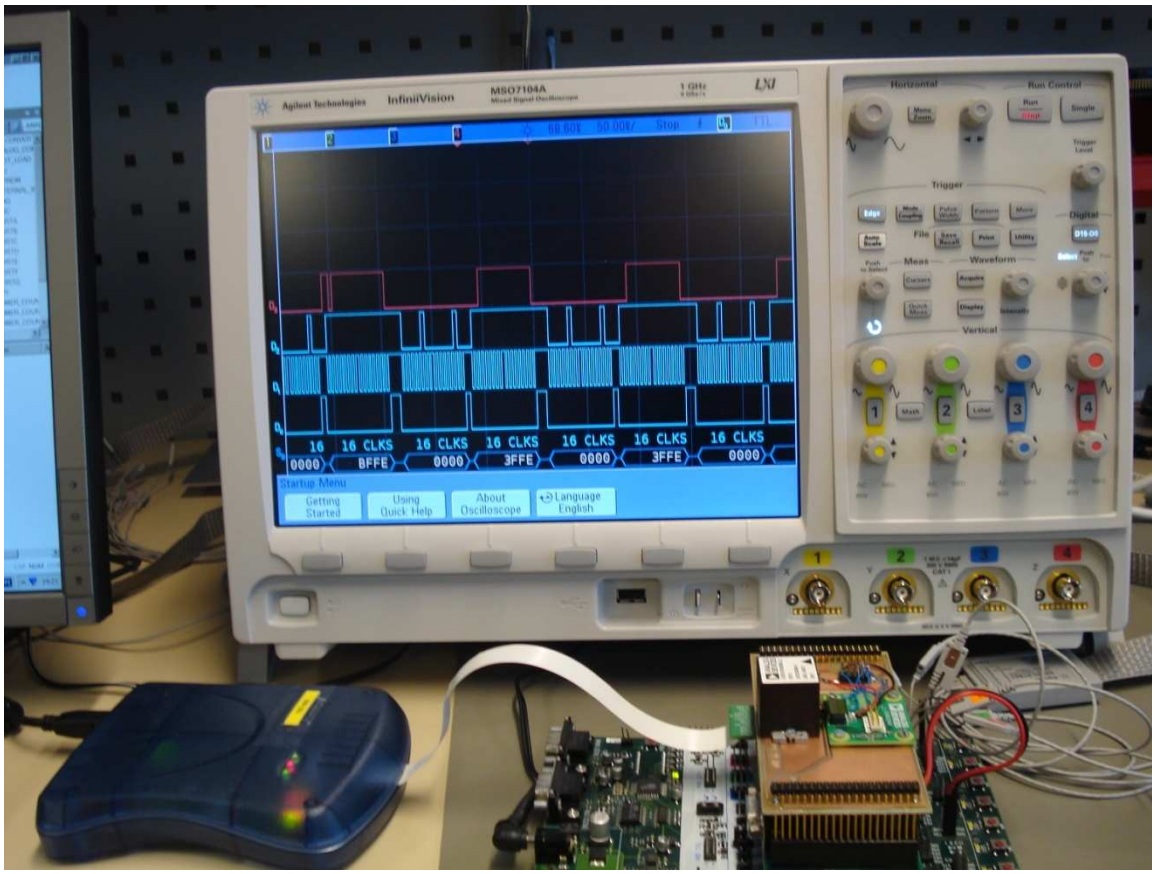
Koodin ajaminen PC:ltä STK500:n ATmega128-prosessoriin tapahtui JTAG ICE -emulaattorilla, jolla voi ajaa koodia prosessorilla rivi kerrallaan. Testissä tulokset laitettiin RS232-kaapelilla PC:n sarjaporttiin. RS232-kaapelin toinen pää kiinnitettiin STK500-alustan Spare-liittimeen. Anturin antamat arvot ajettiin sarjaportin kautta Windowsin HyperTerminal-ohjelmaan.

Ohjelma koostuu kolmesta otsikkotiedostosta ja kahdesta lähdekoodista. Otsikkotiedostossa "defines.h" määritetään prosessorin sisäinen taajuus ja baudinopeus. Tiedostot "serial.h" ja "serial.c" kuuluvat sarjaporttiin [12]. Otsikkotiedostossa "ADIS.h" määritetään ADIS16360-liikeanturin rekisterien osoitteet, globaaliset- ja testauskomennot. Varsinainen tiedosto on "ADIS.c", joka sisältää pääohjelman ja SPI-väylän kommunikointiin kuluvat aliohjelmat. Liitteessä 1 on esitetty koko ohjelma.

Ohjelman ongelma on se, että normaalitilanteessa SPI-väylän sekä lähettävässä että vastaanottavassa piirissä on 8-bittinen siirtorekisteri, mutta tässä tilanteessa ATmega128-prosessorin siirtorekisteri on 8-bittinen, kun taas ADIS16360-anturin rekisterit ovat 16-bittisiä. Toinen ongelma on se, että anturin databittejä on 12 (käyttöjännite, lämpötila, ADC-muunnin) tai 14 (kiihtyvyys, kulmanopeus). Lisäksi saadut arvot voivat olla positiivisia tai negatiivisia, eli niidenkin pitää erottua toisistaan. Ongelmien ratkaiseminen tapahtuu seuraavasti:

Ensin vedetään piirinvalintalinja alas ja lähetetään kohderekisterin osoite ensimmäiseen SPI_Transmitin avulla. Kuitenkin ADIS tarvitsee 16 kellojaksoa komennon suorittamiseen, jotta lähetetään toinen merkityksetön tavu (0x00) ja vedetään piirinvalintalinja ylös. Sen jälkeen ADIS16360-anturi voi laittaa 16-bittisen kohderekisterin sisällön outputpuskuriin. Kuvassa 21 on esitetty, miten ohjelma toimii. On huomattava, että toisella lähettävällä tavulla ei ole merkitystä.

Sen jälkeen vedetään piirinvalintalinja alas ja lähetetään kaksi merkityksetöntä tavua ja samanaikaisesti luetaan kohderekisterin sisältö SPI_WriteReadin avulla. Ensimmäinen SPI_WriteRead lukee ylemmän tavun ja toinen SPI_WriteRead lukee alemman tavun. Lopuksi vedetään piirinvalintalinja ylös.



Kuva 21. Ohjelman toiminta

Nyt kaksi tavua ovat alla olevan mukaisia, jossa D:t edustavat todellisia tietoja:

Ylempi tavu = 00000000DDDDDDDD

Alempi tavu = 00000000DDDDDDDD

"msb_byte<<8" siirtää ylemmän tavun tietoja matemaattisesti oikeaan paikkaan, eli se siirtää tietoja 8 bittiä vasemmalle:

Ylempi tavu = DDDDDDDD00000000

Alempi tavu = 00000000DDDDDDDD

Nyt voidaan yhdistää kaksi tavua OR-operaatiolla, jotta saadaan koko 16-bittinen arvo (temp = temp | lsb_byte):

temp = DDDDDDDDDDDDDDDD

Kiihtyvyys- ja kulmanopeusanturille kaksi ylempää bittiä ovat uusi datalippu (ND) ja virhelippu/hälytyslippu (EA). Kun output datarekisterit päivitetään uusilla tiedoilla, ND-bitti (new data) menee 1-tilaan, jos rekisteri sisältää lukematonta tietoa. Sen jälkeen kun tiedot on luettu, se palaa 0-tilaan. EA-bitti (error alarm) on 1, kun virhelippu/hälytyslippu DIAG_STAT-rekisterissä on 1. Ne eivät muodosta rekisterin matemaattista arvoa, joten ne on nollattava. Tämä voidaan tehdä 16-bittisellä AND-operaatiolla (temp = (temp & 0b0011111111111111)):

DDDDDDDDDDDDDDDD & 0011111111111111 = 00DDDDDDDDDDDDDD

Lopuksi voidaan tarkistaa, onko saatu arvo negatiivinen vai ei. Jos MSB-bitti (D13) on yksi, luku on negatiivinen:

MSB = temp & 0x2000

Jos luku on negatiivinen (if (MSB == 0x2000)), käännetään kaikki bitit päinvastaisiksi(1 → 0 ja 0 → 1) XOR-operaation avulla ja lisätään luku 1:

temp = temp ^ 0x3FFF

temp = temp+1

temp = temp * −1.

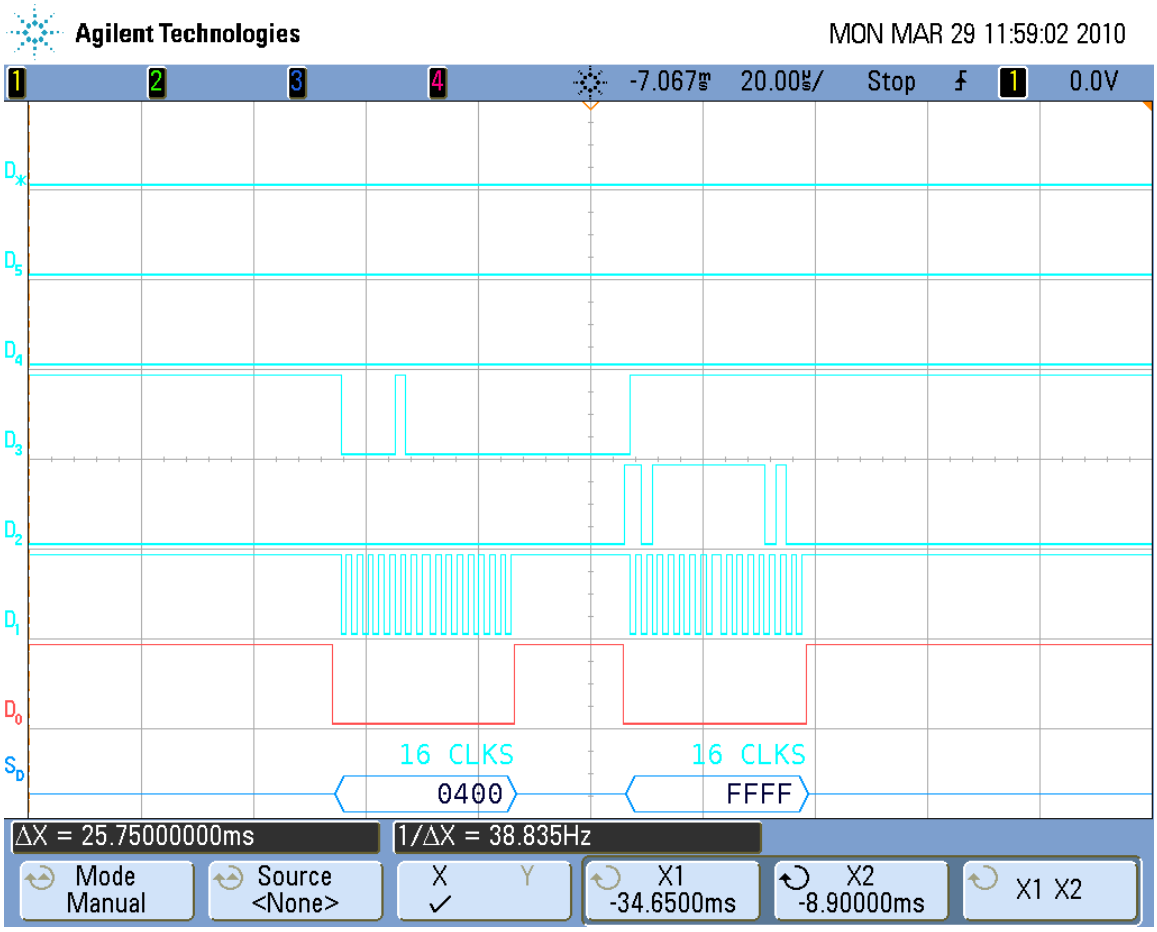
7 TESTAUS JA TULOSTEN ANALYSOINTI

7.1 Kohinatesti

Kaikilla inertia-antureilla on kohinaa. Yleensä kohina on suoraan verrannollinen kaistanleveyteen. Käyttäjän on laskettava kohina kaistanleveyden perusteella. Anturin kohinan RMS-arvo (tehollisarvo) voidaan laskea alla olevasta kaavasta:

$$\text{Kohinan RMS - arvo} = \text{Kohinatiheys} \times \sqrt{\text{Kaistanleveys} \times \frac{\pi}{2}} \tag{6}$$

ADIS16360-anturin spesifikaation perusteella 25 Hz:n kaistanleveyttä käytettäessä kulmanopeusanturin kohinatiheys on 0,044°/s/$\sqrt{Hz}$ rms. Kiihtyvyysanturin kohinatiheys on 0,5 mg/$\sqrt{Hz}$ rms (g = 9,81 m/s^2).



Kuva 22. Kaistanleveyden määrittäminen logiikka-analysaattorin avulla

Kohinatestin tarkoituksena oli kuitenkin mitata kohinaa, jolloin pääasia ei ollut anturin asento, vaan anturin liikkumattomuus. Testissä tulokset välitettiin RS232-kaapelilla PC:n sarjaporttiin. RS232-kaapelin toinen pää kiinnitettiin STK500-alustaan Spare-liittimelle.

ADIS16360-anturille suoritettiin testi, jossa anturi oli pöydällä lepotilassa, niin että anturiin kohdistui vain painovoima. Maan vetovoima kohdistui näin mahdollisimman suurella prosentilla z-akseliin. Anturilta luettiin 5 ms:n välein x-, y- ja z-akseleiden arvot 600 kertaa. Liikeanturin minimi-, maksimi- ja keskiarvot 600 näytteestä on laskettu taulukkoon 12.

Käytetty kaistanleveys testeissä oli noin 40 Hz (kuva 22). Kaavasta 6 voidaan laskea kohinan tehollisarvo sekä kulmanopeus- että kiihtyvyysanturille:

$$\text{Kulmanopeusanturin kohina} = 0{,}044°/\text{s}/\sqrt{\text{Hz}}\ \text{rms} \times \sqrt{40\ \text{Hz} \times \frac{\pi}{2}} = 0{,}349°/\text{s rms}$$

$$\text{Kiihtyvyysanturin kohina} = 0{,}5\ \text{mg}/\sqrt{\text{Hz}}\ \text{rms} \times \sqrt{40\ \text{Hz} \times \frac{\pi}{2}} = 3{,}96\ \text{mg}/\sqrt{\text{Hz}}\ \text{rms}$$

On huomattava, että kulmanopeusanturin kohinatiheys riippuu käytettävästä kaistanleveydestä, mutta sitä tässä ei otettu huomioon.

Taulukko 12. Liikeanturin minimi-, maksimi- ja keskiarvot 600:sta näytteestä liikkumattomassa tilassa

	Kulmanopeusanturi [°/s]			Kiihtyvyysanturi [g]		
	x-akseli	y-akseli	z-akseli	x-akseli	y-akseli	z-akseli
Maksimi	1,400	1,300	0,300	0,003	0,033	−0,996
Keskiarvo	0,289	−0,008	−0,651	−0,007	0,022	−1,003
Minimi	−0,950	−1,200	−1,800	−0,017	0,013	−1,012

Anturin teoreettiseen arvoon vaikuttaa suuri määrä tekijöitä, joiden aiheuttamia virheitä emme kyenneet havaitsemaan, kuten tarkka painovoima, anturin tarkka asento maahan nähden sekä muut voimat, jotka anturiin vaikuttavat. Koska ulkoiset häiriötekijät virheenarvioinnissa olisivat olleet pieniä, oli viisaampaa verrata anturilta luettuja yksittäisiä arvoja anturin kaikkien arvojen keskiarvoon kuin teoreettiseen oikeaan arvoon.

Suurimmista arvoista voitiin vähentää akselin keskiarvo ja keskiarvosta vähentää minimi, jolloin saatiin jokaiselle akselille suurin poikkeama keskiarvosta. Ne on esitetty taulukossa 13 harmaana.

Taulukko 13. Liikeanturin suurimmat poikkeamat keskiarvosta liikkumattomassa tilassa

	Kulmanopeusanturi [°/s]			Kiihtyvyysanturi [g]		
	x-akseli	y-akseli	z-akseli	x-akseli	y-akseli	z-akseli
Maksimi − Keskiarvo	1,112	1,308	0,951	0,010	0,011	0,007
Keskiarvo − Minimi	1,239	1,192	1,149	0,010	0,009	0,009

Tälle kulmanopeusanturille voitiin laskea x-akselin arvon poikkeaman olevan maksimissaan 1,239°/s, y-akselin 1,308°/s ja z-akselin 1,149°/s. Laskettu kohinan tehollisarvo oli 0,349°/s rms. Myös kiihtyvyysanturille voitiin laskea x-akselin arvon poikkeaman olevan maksimissaan 10 mg, y-akselin 11 mg sekä z-akselin 9 mg. Laskettu kohinan tehollisarvo oli 3,96 mg/$\sqrt{Hz}$ rms. Näistä tuloksista ja ADIS16360-anturin spesifikaation perusteella voitiin päätellä, että anturilta voitiin lukea kulmanopeus- ja kiihtyvyysarvot kohtalaisen tarkasti.

7.2 Ympäristöolosuhteiden vaikutus

Antureiden toiminta vaihtelee ympäristöolosuhteista riippuen. Myös ympäristöolosuhteet vaikuttavat mittauksen luotettavuuteen. Anturin toiminta heikkenee, kun se sijoitetaan vaikeisiin ympäristöolosuhteisiin. Korkea lämpötila, kova paine, voimakkaat magneetti- ja sähkökentät, kovat tärinät sekä vaihteleva lämpötila saattavat häiritä anturin toimintaa.

Taulukko 14. Ympäristöolosuhteiden vaikutus

	Kulmanopeusanturi [°/s rms]			Kiihtyvyysanturi [mg/$\sqrt{Hz}$ rms]		
	x-akseli	y-akseli	z-akseli	x-akseli	y-akseli	z-akseli
Mitattu kohina	1,239	1,308	1,149	10	11	9
Teoreettinen kohina	0,349	0,349	0,349	3,96	3,96	3,96
Mitattu − Teoreettinen	0,89	0,959	0,8	6,04	7,04	5,04

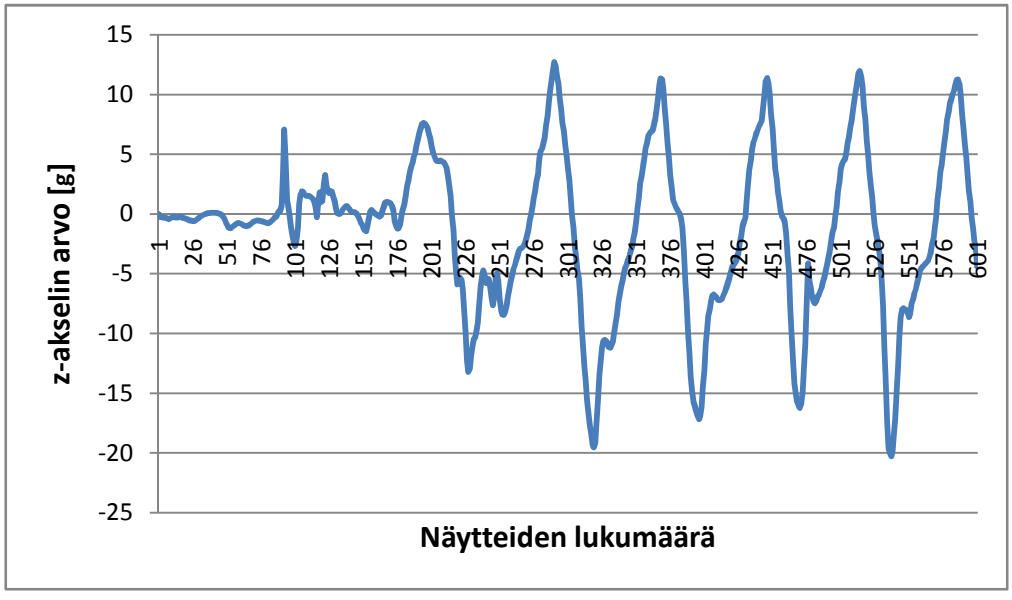
Ympäristöolosuhteiden vaikutus saadaan, kun mitatusta kohinasta (ks. taulukko 13) vähennetään teoreettinen kohina (taulukko 14). Nämä tulokset sisältävät myös virheitä, jotka aiheutuvat testausmenetelmästä, testauslaitteista ja ohjelmasta.

ADIS16360-anturin mittausjärjestelmän suunnitteluvaiheessa on siis mahdollisimman tarkkaan huomioitava anturin sijoituspaikassa vallitsevien ympäristöolosuhteiden vaikutukset. Näin minimoidaan virheet mittaustuloksissa ja saadaan mittausjärjestelmästä mahdollisimman luotettava.

7.3 Mittausaluetesti

Tässä testissä pyrittiin testaamaan, onko anturi soveltuva liikeanturiksi mittausalueensa osalta. ADIS16360-anturilla voidaan mitata kiihtyvyyksiä ±18 g:hen asti. Myös anturilla voidaan mitata kulmanopeuksia ±300°/s saakka.

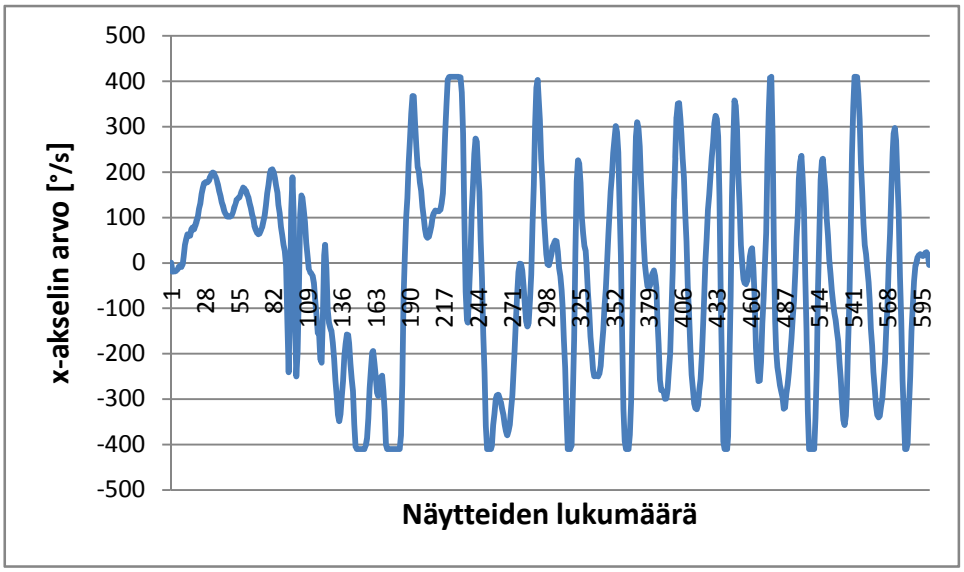
Testit suoritettiin kokeellisesti ottamalla anturi käteen ja heiluttamalla sitä rajusti siten, että liikesuunta muuttui koko ajan. Kun liikesuunta vaihtelee, kiihtyvyys ja kulmanopeus muuttuvat nopeasti.



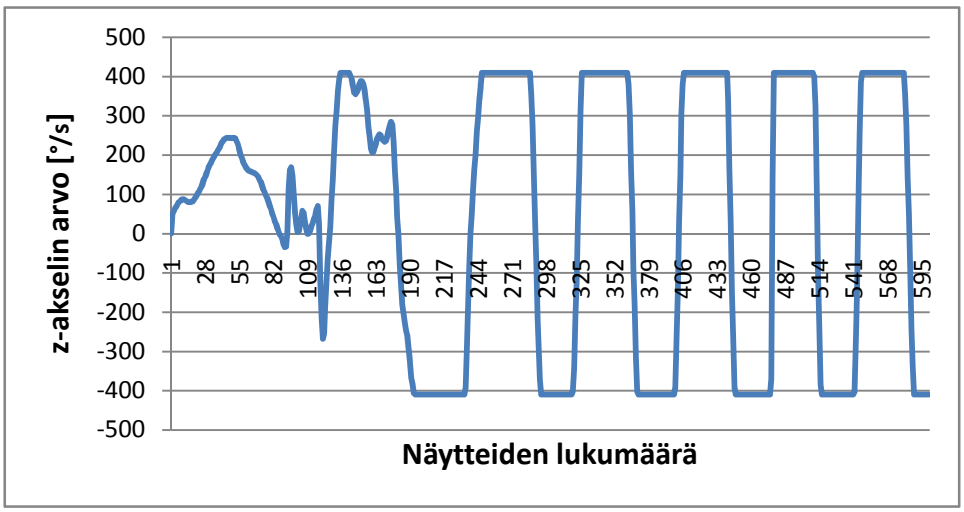
Kuva 23. Kiihtyvyysanturin z-akselin mittausaluetesti

Kuvassa 23 on kiihtyvyysanturin z-akselin suuntaisen kiihtyvyyden graafinen esitys. Kuvasta ilmenee, että raju heilautus menee noin −2 g yli mittausalueen, kun anturin liike on maan vetovoiman suuntaan.

Kuvissa 24 ja 25 on kulmanopeusanturin x- ja z-akselin suuntaisten kiihtyvyyksien graafiset esitykset. Tuloksista ilmenee, että mittausalue on reilusti ylittynyt. Kuvioista ilmenee myös, että kun kulmanopeudet ovat lähellä ±400°/s, anturin datan voidaan havaita olevan epävakaata. Kuvasta 25 nähdään, että kun kulmanopeus on yli ±400°/s, tulee kiihtyvyyteen leikkaantuminen.



Kuva 24. Kulmanopeusanturin x-akselin mittausaluetesti

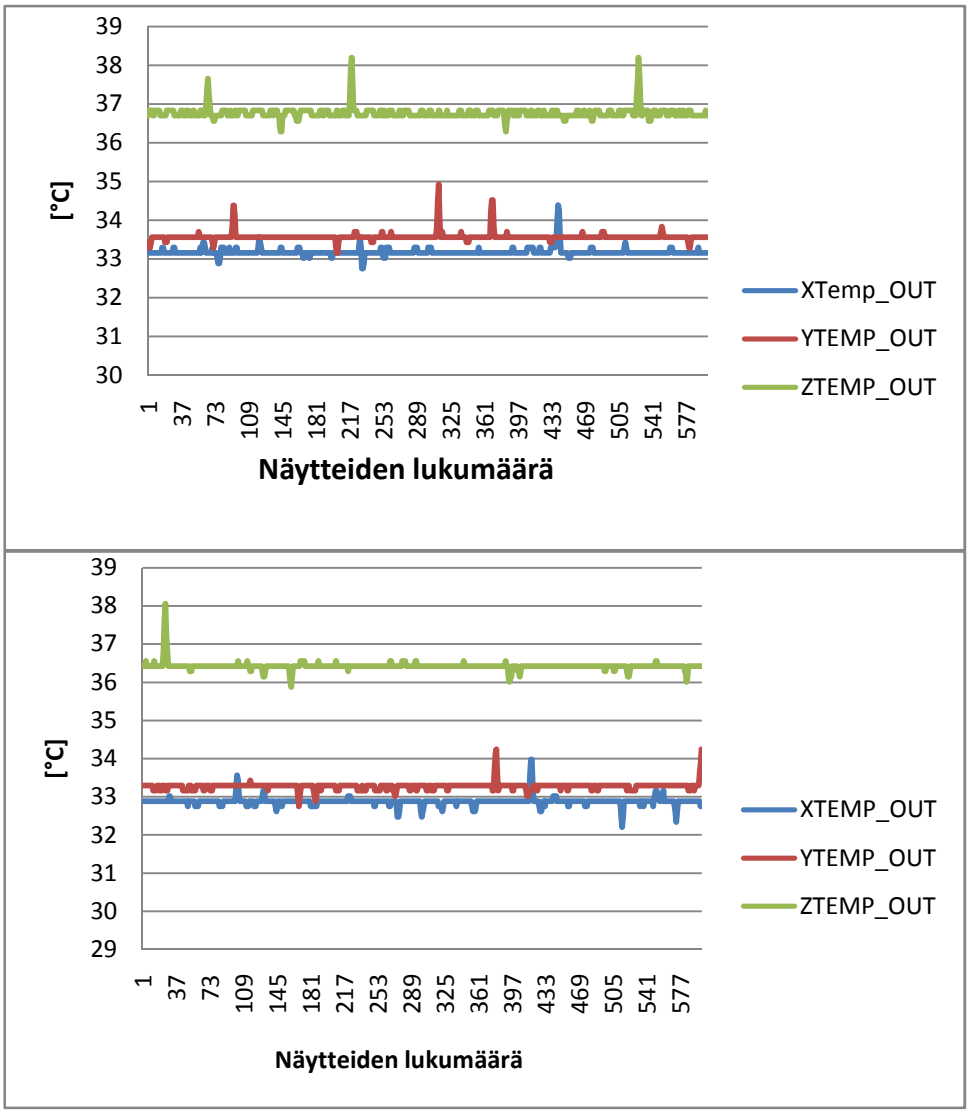


Kuva 25. Kulmanopeusanturin z-akselin mittausaluetesti

7.4 Toistettavuus

7.4.1 Lämpötila-anturin toistettavuustesti

ADIS16360-liikkeanturin komponentissa olevan lämpötila-anturin akseleiden arvot eivät ole ympäristön lämpötila-arvoja, vaan ne ovat laitteen sisäinen lämpötila. Testissä otettiin kaksi kertaa 600 näytettä lämpötila-anturin akseleiden arvoja peräkkäisesti (kuva 26). Taulukosta 15 nähdään, että kohinan osuus voi olla maksimissa 1,6 °C, mutta jokaiselle akselille keskiarvojen erot näytteistä on vain 0,3 °C, eli tulokset ovat toisiaan lähellä ja lämpötila-anturin luotettavuus on hyvä.



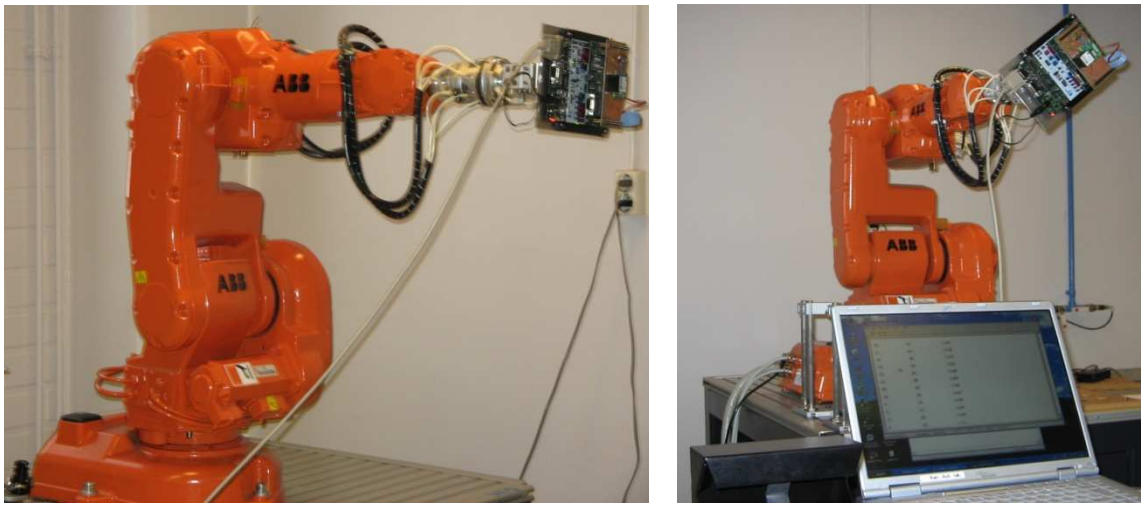
Kuva 26. Lämpötila-anturin akseleiden toistettavuustesti

Taulukko 15. Lämpötila-anturin toistettavuustestin tulokset

Akseli	x-akseli [°C]	y-akseli [°C]	z-akseli [°C]
Maksimi 1	34,4	34,9	38,2
Keskiarvo 1	33,2	33,6	36,8
Kohina 1 = Maksimi 1 − Keskiarvo 1	1,2	1,3	1,4
Maksimi 2	34,0	34,2	38,1
Keskiarvo 2	32,9	33,3	36,4
Kohina 2 = Maksimi 2 − Keskiarvo 2	1,1	1,0	1,6
Keskiarvojen ero	0,3	0,3	0,3

7.4.2 Kulmanopeusanturin toistettavuustesti

Anturille suoritettiin testi, jossa ADIS16360-liikeanturi kytkettiin Kajaanin ammattikorkeakoulussa sijaitsevaan robottilaitteeseen (kuva 27). Robotilla voitiin ohjelmoida tarkasti, paljonko anturi liikkuu. Testin tavoitteena oli testata ja analysoida anturin toistettavuutta. Toistettavuus ilmaisee anturin kykyä pitää tarkkuutensa koko ajan.



Kuva 27. Liikeanturi robottikourassa

Testissä tulokset välitettiin RS232-kaapelilla PC:n sarjaporttiin. Tiedot kerättiin talteen käyttäen HyperTerminal-ohjelmaa, jossa käytettiin asetukset 9600, N, 8, 1.

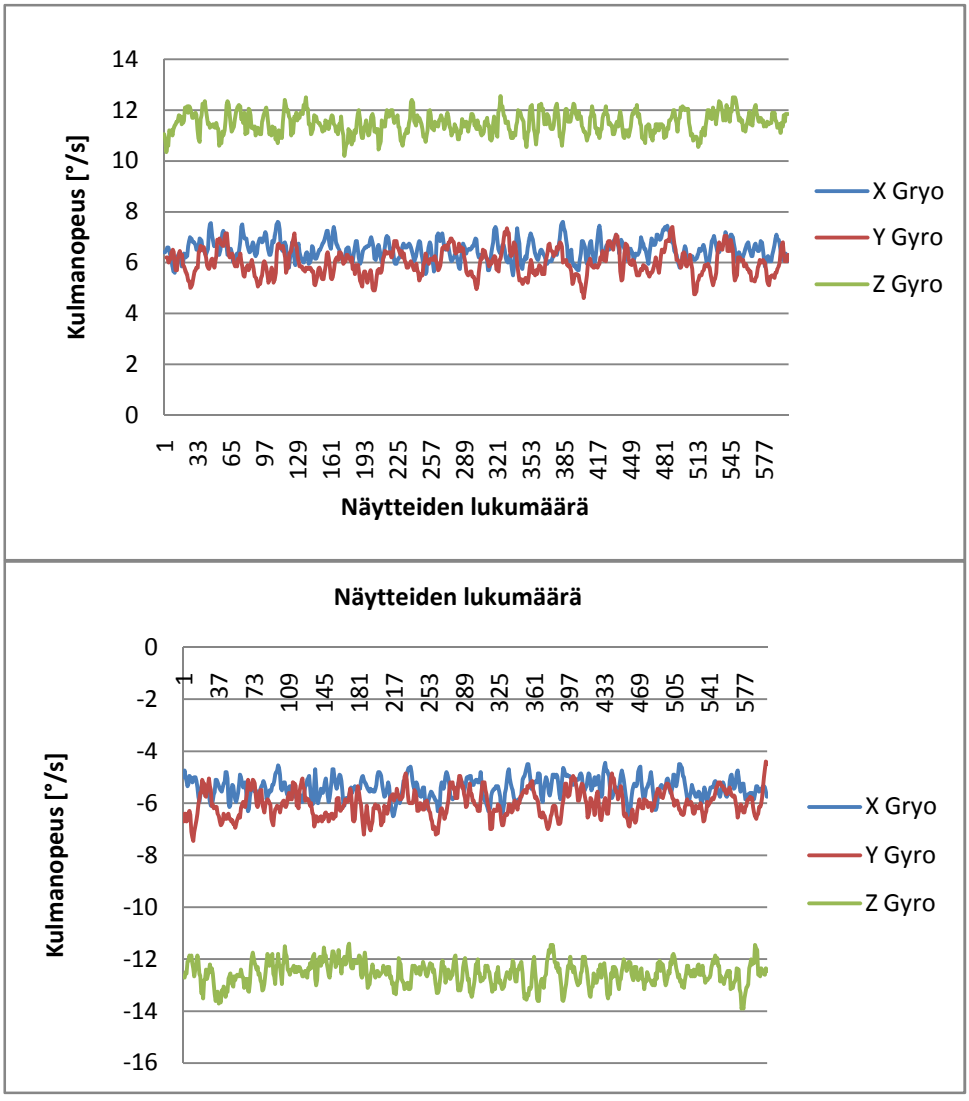
Käytetty kaistanleveys testeissä oli 38,835 Hz (ks. kuva 22), ja sen perusteella yhden kerran ohjelman suorittaminen kestää 25,75 ms ja 600 näytteenoton kestoaika on 15,45 s (600 × 25,75 ms).

Tasaisessa pyörimisliikkeessä kulmanopeus (ω) on vakio. Jos hetkellä t = 0 kulma-asema φ = 0, niin hetkellä t kulma-asema on:

$$\varphi = \omega t, \tag{7}$$

missä kulmanopeuden yksikkö on rad/s (1 radiaani = 57,295 astetta).

Kaavasta 7 laskettiin kulmanopeus kun aikaväli oli 15,45 s ja kulma-aseman muutos oli sekä 90° että 180°, jotka saatettiin tulokset noin 6°/s ja 12°/s.



Kuva 28. Kulmanopeusanturin akseleiden toistettavuustesti

Ensin toistettavuustesti suoritettiin kulmanopeusanturin x-akselille, jotta ADIS16360-liikeanturi käännettiin robotin avulla 90° pystytasossa kulmanopeudella 6°/s.

Myös kulmanopeusanturin y-akselille tehtiin testi, jossa liikeanturi käännettiin robotilla 90° vaakatasossa kulmanopeudella 6°/s.

Kulmanopeusanturin z-akselin suoritettiin toistettavuustesti, jossa liikeanturi käännettiin robotin avulla 180° kulmanopeudella 12°/s. Tulokset on esitetty taulukossa 16 ja kuvassa 28.

Taulukko 16. Kulmanopeusanturin toistettavuustestin tulokset

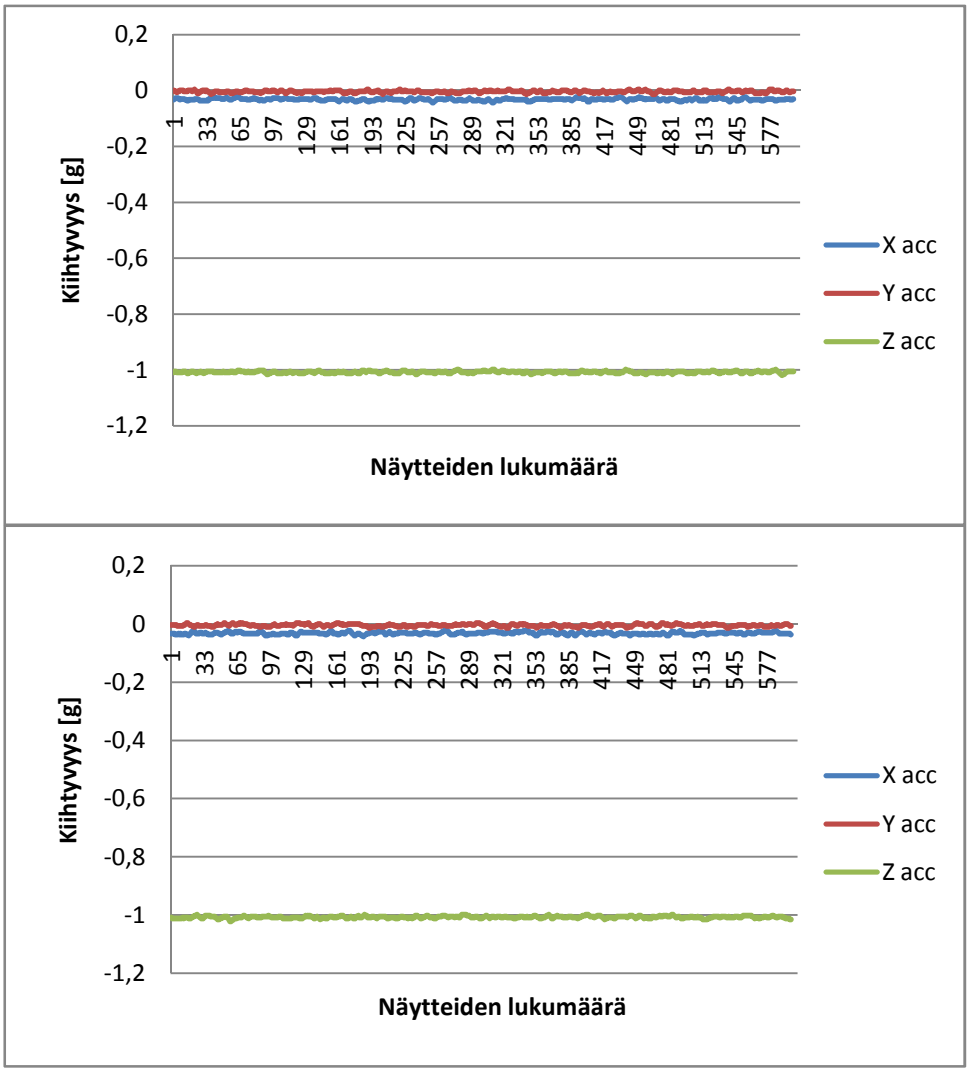
Akseli	x-akseli [°/s]	y-akseli [°/s]	z-akseli [°/s]
Keskiarvo 1	6,5	6,0	11,5
Suurin poikkeama 1	1,1	1,4	1,3
Keskiarvo 2	5,4	6,0	12,6
Suurin poikkeama 2	1,1	1,6	1,3
Keskiarvojen ero	1,1	0	1,1

Kuvasta 28 voidaan havaita, että käyrät eivät ole suoria viivoja. Kohina on niin suuri, että yksittäisessä mittauksessa se tekee tuloksen tietyissä rajoissa epävarmaksi. Jokaisesta kuudestasadasta näytteenotosta kohinasta aiheutuma suhteellinen virhe on noin 16 %, mutta kohinan merkitys pienenee, kun kulmanopeus kasvaa.

Taulukosta 16 nähdään, että kulmanopeusanturin toistettavuustestissä y-akselin keskiarvot ovat samoja, mutta x- ja z-akselin keskiarvojen ero on 1,1°/s. Suurilla kulmanopeuksilla tulosten eron merkitys pienenee.

7.4.3 Kiihtyvyysanturin toistettavuustesti

ADIS16360-anturille suoritettiin toistettavuustesti, jossa anturi oli pöydällä lepotilassa, niin että anturiin kohdistui vain painovoima. Maan vetovoima kohdistui näin mahdollisimman suurella prosentilla z-akseliin. Anturilta luettiin 5 ms:n välein x-, y- ja z-akseleiden arvot 600 kertaa. Sama testi suoritettiin kaksi kertaa. Tulokset on esitetty taulukossa 17 ja kuvassa 29.



Kuva 29. Kiihtyvyysanturin akseleiden suuntaisten kiihtyvyyksien toistettavuustesti

Taulukosta 16 nähdään, että kiihtyvyysanturin toistettavuustestissä keskiarvojen ero kaikille kolmelle akselille on nolla, eli tulokset ovat täysin samat. ADIS16360-liikeanturissa olevassa kiihtyvyysanturin luotettavuus on erinomainen kulmanopeusanturiin verrattuna. Myös sen kohina on pieni.

Taulukko 17. Kiihtyvyysanturin toistettavuustestin tulokset

Akseli	x-akseli [g]	y-akseli [g]	z-akseli [g]
Keskiarvo 1	0,032	0,004	1,008
Suurin poikkeama 1	0,043	0,013	0,019
Keskiarvo 2	0,032	0,004	1,008
Suurin poikkeama 2	0,043	0,013	0.014
Keskiarvojen ero	0	0	0

8 YHTEENVETO

Tämän insinöörityön tarkoituksena oli ADIS16360-liikeanturin mittausjärjestelmän toteuttaminen siten, että voitaisiin mitata sekä kolmiulotteinen kiihtyvyys että kolmiulotteinen kulmanopeus. Sen lisäksi liikeanturilla voidaan mitata käyttöjännite ja lämpötila. Anturissa on myös AD-muunnin.

Tämä työ vaati paljon itsenäistä opiskelua ja tekemistä. Työn tekeminen alkoi tutustumisella ADIS16360-liikeanturin rekistereihin ja sen toiminnallisuuteen sekä tiedonkeräykseen.

Kun ADIS16360-liikeanturin toiminnallisuus oli opittu tarpeeksi hyvin, voitiin siirtyä seuraavaan tehtävään, joka oli kytkennän suunnittelu ja rakentaminen. Työn tärkeä osa oli liikeanturin ohjelmointi. Samalla tuli tutustuttua työkaluihin, joita käytetään liikeanturin testauksessa.

Liikeanturille tehtiin kohina-, mittausalue- ja toistettavuustestit. Kulmanopeusanturin toistettavuustestissä käytettiin apuna Kajaanin ammattikorkeakoulussa olevaa robottilaitetta.

Itse insinöörityö-dokumentin tekeminen oli vaikeaa ja ongelmallista. Kuitenkin vaikein asia työssä oli omaksua suuri määrä tietoa nopeasti. Lähes kaikki materiaalit ovat englanninkielisiä, joten tämän dokumentin tekeminen suomeksi vaati myös todella paljon työtä.

Tämän dokumentin tekeminen oli työläs ja aikaa vievää. Kieliongelmien vuoksi dokumenttiin tehtiin useita kertoja isoja muutoksia.

LÄHTEET

1 ADIS1636x: High Precision Tri-Axis Inertial Sensor

http://dkc1.digikey.com/tw/zh/ph/ADI%5CADIS1636.html?WT.z_Tab_Cat=Featured%20Products d. 7.3.2010

2 Kiihtyvyys- ja kulmanopeusanturit

http://math.tut.fi/courses/MAT-45800/paikannuksen_matematiikka_ja_menetelmat_2008.pdf d. 8.3.2010

3 Kiihtyvyysanturit

https://publications.theseus.fi/bitstream/handle/10024/4336/Hissin%20ajokayran%20raja-arvojen%20maarittaminen.pdf?sequence=1 d. 8.3.2010

4 Newtonin toinen laki – inertia-anturit

http://www.vuorimiesyhdistys.fi/materia/pdf/Materia_2007-4.pdf d. 9.3.2010

5 ADIS16360/ADIS16365 datasheet versio B

http://www.analog.com/static/imported-files/data_sheets/ADIS16360_16365.pdf

d. 9.3.2010

6 ATmega128 datasheet

http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf d. 10.3.2010

7 SPI (Serial peripheral interface)

http://users.jyu.fi/~jupeihal/Kontrollerit3.pdf d. 10.3.2010

8 AVR_rauta. SPI-ohjelmointia

koti.mbnet.fi/aleksiho/Kirja_29092008/OSA_2/6.5.AVR_rauta.SPI-ohjelmointia.pdf d. 10.3.2010

9 SPI

https://publications.theseus.fi/bitstream/handle/10024/5173/Salo_Sampsa.pdf?sequence=1 d. 11.3.2010

10 Demo/Evaluation Tips for the ADIS1636x

http://www.analog.com/static/imported-files/eval_boards/Demo_EvalTips_ADIS1636x.pdf d. 18.3.2010

11 AVR STK500 User Guide

http://www.atmel.com/dyn/resources/prod_documents/doc1925.pdf d. 19.3.2010

12 avr-libc

AVR Studio → Help → avr-libc Reference Manual → Example Projects → Using the standard IO facilities → The source code d. 21.3.210

13 AT90Can128 and Adis 16355

http://www.elektroda.pl/rtvforum/topic1419309.html d. 23.3.2010

14 ADIS1636x: High Precision Tri-Axis Inertial Sensor

http://dkc1.digikey.com/us/en/ph/ADI%5CADIS1636.html?WT.z_Tab_Cat=Featured%20Products d. 17.4.2010

15 Tervonen, Ari-Antti. Peliohjaimen kiihtyvyysanturi. Insinöörityö. Kajaanin ammattikorkeakoulu, Tekniikka ja liikenne, tietotekniikka. Kajaani, 2009.

LIITTEET

ADIS16360-LIIKEANTURIN OHJELMOINTI

```
// ADIS.c

#include <avr/io.h>      //Declare this for register definitions of the MCU
#include <string.h>
#include <stdio.h>
#include <inttypes.h>

#include <ctype.h>
#include <stdint.h>
#include <stdlib.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>


#include "ADIS.h"
#include "serial.h"

//------------------------------------------------

#define F_CPU       8000000     // 8MHz processor
#include <util/delay.h>

#ifndef BV
        #define BV(bit)             (1<<(bit))
#endif

#ifndef cbi
        #define cbi(reg,bit)        reg &= ~(BV(bit))
#endif

#ifndef sbi
        #define sbi(reg,bit)        reg |= (BV(bit))
#endif

#define CS_DOWN     cbi(PORTB,0)
#define CS_UP       sbi(PORTB,0)

////////////////////////////////////////////////

void SPI_MasterInit(void)
{
                            // setup SPI I/O pins
        sbi(PORTB, 1);  // set SCK high
```

```
        sbi(DDRB, 1);           // set SCK as output
        cbi(DDRB, 3);           // set MISO as input
        sbi(DDRB, 2);           // set MOSI as output
        sbi(DDRB, 0);           // SS must be output for Master mode to wor

          /* Enable SPI, Master, set clock rate fck/16= 1Mb/s */
         SPCR = (1<<SPE)|(1<<MSTR) |(1<<SPR0) | (1<<CPHA) | (1<<CPOL) | (1<<SPIE);

}

//-----------------------------------------------

// writes 8-Bit Data to MOSI
void SPI_Transmit(char cData)
{
        /* Start transmission */
        SPDR = cData;
        /* Wait for transmission complete */
        while(!(SPSR & (1<<SPIF)));
}

//-----------------------------------------------

// returs 8-Bit data  from MISO
unsigned char SPI_Receive(void)
{
        /* Wait for reception complete */
        while(!(SPSR & (1<<SPIF)));
        /* Return data register */
        return SPDR;
}

//-----------------------------------------------

// sends  8-Bit OutData to MOSI
// returs 8-Bit data  from MISO
unsigned char SPI_WriteRead(char cData)
{
        /* Start transmission */
        SPDR = cData;
        /* Wait for transmission complete */
        while(!(SPSR & (1<<SPIF)));
        /* Return data register */
        return SPDR;
}

//-----------------------------------------------

unsigned  int   read_register(unsigned char adr)
{
  unsigned  int     temp=0;
  int MSB=0;
```

```
  unsigned char   msb_byte=0, lsb_byte=0;


  adr = adr & 0b01111111;          // R/W=0 ja osoite 0n 7-bittinen;

  // writes 16-Bit Data to Address of ADIS-device
  // (0x00 is don't care)
  CS_DOWN;   // Set Chip select low
  _delay_us(0.1);
  SPI_Transmit(adr);       //Send register address
  SPI_Transmit(0x00);    //Send dummy byte (ignored by the ADIS)
  CS_UP;                   // Set Chip select high
  _delay_us(20);


  // sends  16-Bit OutData to MOSI (0xFF are don't care)
  // returs 16-Bit data  from MISO (msb_byte, lsb_byte)
  CS_DOWN;               // Set Chip select low
  // Read the MSByte
  msb_byte = SPI_WriteRead(0xFF);       // msb_byte  = 0000 0000 DDDD DDDD
  // Read the LSByte
  lsb_byte = SPI_WriteRead(0xFF);       // lsb_byte    = 0000 0000 DDDD DDDD
  CS_UP;                                // Set Chip select high

  // Bitshift the MSByte 8 places left
  temp = (unsigned int )(msb_byte<<8);   // msb_byte = DDDD DDDD 0000 0000

  // then add them together
  temp = temp | lsb_byte;                 // temp       = DDDD DDDD DDDD DDDD

  // Strip off the ND and EA flags
  temp = (temp & 0b0011111111111111); // temp       = 00DD DDDD DDDD DDDD
  // Jos kyseessä on lämpötila vielä pitää poistaa bitit D13 ja D12,
  // eli                                    temp       = 0000 DDDD DDDD DDDD

   MSB=temp & 0x2000;                     // katsotaan onko MSB 1

   if (MSB==0x2000)
   {
          temp = temp ^ 0x3FFF;           // jos msb=1, invertoidaan, lisätään 1
          temp = temp+1;
          temp = temp*-1;
    }

          return temp;

          MSB=0;

}

//-------------------------------------------------------
```

```
FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, uart_getchar, _FDEV_SETUP_RW);
int main()
{
	unsigned int XGYRO,YGYRO,ZGYRO;

	char p[17];

	SPI_MasterInit();		// Initialise SPI

	uart_init();			// Initialise UART

	stdout = stdin = &uart_str;

	DDRA=0x00;			//switches

	while(1)
	{
		while((PINA != 0xFE));

		for(int i=0;i<600;i++)
		{
			XGYRO = read_register(XGYRO_OUT);
			_delay_us(1);
			itoa(XGYRO, p, 10);
			printf(p);
			printf(", \t\t");


			YGYRO = read_register(YGYRO_OUT);
			_delay_us(1);
			itoa(YGYRO, p, 10);
			printf(p);
			printf(", \t\t");


			ZGYRO = read_register(ZGYRO_OUT);
			_delay_us(1);
			itoa(ZGYRO, p, 10);
			printf(p);
			printf("\n\n");

			_delay_ms(10);

		}


		while((PINA != 0xFE));

	return 0;
}
```

```
//serial.c

#include "defines.h"

#include <stdint.h>
#include <stdio.h>

#include <avr/io.h>

#include "serial.h"


//-------------------------------------------------
/*
 * Send character c down the UART Tx, wait until tx holding register
 * is empty.
 */
 int uart_putchar(char c, FILE *stream)
{

  if (c == '\a')
    {
      fputs("*ring*\n", stderr);
      return 0;
    }

  if (c == '\n')
    uart_putchar('\r', stream);
  loop_until_bit_is_set(UCSR0A, UDRE0);
  UDR0 = c;

  return 0;
}

//-------------------------------------------------

/*
 * Receive a character from the UART Rx.
 */
int uart_getchar(FILE *stream)
{
  uint8_t c;
  char *cp, *cp2;
  static char b[RX_BUFSIZE];
  static char *rxp;

  if (rxp == 0)
    for (cp = b;;)
      {
            loop_until_bit_is_set(UCSR0A, RXC0);
            if (UCSR0A & _BV(FE))
              return _FDEV_EOF;
```

```
if (UCSR0A & _BV(DOR))
  return _FDEV_ERR;
c = UDR0;
/* behaviour similar to Unix stty ICRNL */
if (c == '\r')
  c = '\n';
if (c == '\n')
  {
    *cp = c;
    uart_putchar(c, stream);
    rxp = b;
    break;
  }
else if (c == '\t')
  c = ' ';

if ((c >= (uint8_t)' ' && c <= (uint8_t)'\x7e') ||
    c >= (uint8_t)'\xa0')
  {
    if (cp == b + RX_BUFSIZE - 1)
      uart_putchar('\a', stream);
    else
      {
            *cp++ = c;
            uart_putchar(c, stream);
      }
    continue;
  }

switch (c)
  {
  case 'c' & 0x1f:
    return -1;

  case '\b':
  case '\x7f':
    if (cp > b)
      {
            uart_putchar('\b', stream);
            uart_putchar(' ', stream);
            uart_putchar('\b', stream);
            cp--;
      }
    break;

  case 'r' & 0x1f:
    uart_putchar('\r', stream);
    for (cp2 = b; cp2 < cp; cp2++)
      uart_putchar(*cp2, stream);
    break;

  case 'u' & 0x1f:
```

```
            while (cp > b)
              {
                    uart_putchar('\b', stream);
                    uart_putchar(' ', stream);
                    uart_putchar('\b', stream);
                    cp--;
              }
            break;

           case 'w' & 0x1f:
            while (cp > b && cp[-1] != ' ')
              {
                    uart_putchar('\b', stream);
                    uart_putchar(' ', stream);
                    uart_putchar('\b', stream);
                    cp--;
              }
            break;
           }
    }

 c = *rxp++;
 if (c == '\n')
   rxp = 0;

 return c;
}

//-------------------------------------------------
/*
* Initialize the UART to 9600 Bd, tx/rx, 8 N 1.
*/
void uart_init()  // 1 stopbit,nopeus=9600, ei ylivuotoa, ei parityä
{
        UCSR0A=0x00;
        UCSR0B=0x18;
        UCSR0C=0x06;
        UBRR0H=0x00;
        UBRR0L=0x33;
}
```

```
//defines.h

/* CPU frequency */
#define F_CPU 8000000UL

/* UART baud rate */
#define UART_BAUD  9600
```

```
//serial.h


/*
 The internal line buffer is RX_BUFSIZE (80) characters long, which
 includes the terminating \n (but no terminating \0).
*/
#define RX_BUFSIZE 80

int         uart_getchar(FILE *stream);

int         uart_putchar(char c, FILE *stream);

void uart_init();
```

```
//ADIS.h
// Register and command definitions for ADIS16360

#ifndef ADIS_H_
#define ADIS_H_

//-------------------------------

// CONTROL REGISTERS

//        Name            Address         Function

#define FLASH_CNT   0x00            // Flash memory write count

#define SUPPLY_OUT  0x02            // Power supply measurement

#define XGYRO_OUT   0x04            // X-axis gyroscope output
#define YGYRO_OUT   0x06            // Y-axis gyroscope output
#define ZGYRO_OUT   0x08            // Z-axis gyroscope output

#define XACCL_OUT   0x0A            // X-axis accelerometer output
#define YACCL_OUT   0x0C            // Y-axis accelerometer output
#define ZACCL_OUT   0x0E            // Z-axis accelerometer output

#define XTEMP_OUT   0x10            // X-axis gyroscope temperature measurement
#define YTEMP_OUT   0x12            // Y-axis gyroscope temperature measurement
#define ZTEMP_OUT   0x14            // Z-axis gyroscope temperature measurement

#define AUX_ADC     0x16            // Auxiliary ADC output

#define XGYRO_OFF   0x1A            // X-axis gyroscope bias offset factor
#define YGYRO_OFF   0x1C            // Y-axis gyroscope bias offset factor
#define ZGYRO_OFF   0x1E            // Z-axis gyroscope bias offset factor

#define XACCL_OFF   ox20            // X-axis acceleration bias offset factor
#define YACCL_OFF   ox22            // Y-axis acceleration bias offset factor
#define ZACCL_OFF   ox24            // Z-axis acceleration bias offset factor

#define ALM_MAG1    0x26            // Alarm 1 amplitude threshold
#define ALM_MAG2    0x28            // Alarm 2 amplitude threshold

#define ALM_SMPL1   0x2A            // Alarm 1 sample size
#define ALM_SMPL2   0x2C            // Alarm 2 sample size

#define ALM_CTRL    0x2E            // Alarm control

#define AUX_DAC     0X30            // Auxiliary DAC data

#define GPIO_CTRL   0x32            // Auxiliary digital input/output control

#define MSC_CTRL    0x34            // Micellaneous control
```

```
#define SMPL_PRD     0x36            // Internal sample period (rate) control

#define AVG_CNT      0x38            // Dynamic range and digital filter control

#define SLP_CNT      0x3A            // Sleep mode control

#define DIAG_STAT    0x3C            // System status

#define GLOB_CMD     0x3E            // System command

//--------------------------------

// GLOB_CMD BIT DESCRIPTIONS

#define GLOB_CMD_RST                 0x0080  // Software reset command,
                                             // 1000 000 --> GLOB_CMD[7]=1 (DIN=0xBE80)

// Bits [6:5] not used

#define GLOB_CMD_STATCLR             0x0010  // Precision autonull command (reset all bits to 0),
                                             // 0001 0000 --> GLOB_CMD[4]=1

#define GLOB_CMD_FLSHUPD             0x0008  // Flash update command (backs up all registers),
                                             // 000 1000 --> GLOB_CMD[3]=1

#define GLOB_CMD_DACLATCH            0x0004  // Auxiliary DAC data latch
                                             // 000 0100 --> GLOB_CMD[2]=1

#define GLOB_CMD_FACTCALB            0x0002  // Factory calibration restore command,
                                             // 000 0010 --> GLOB_CMD[1]=1

#define GLOB_CMD_AUTONULL            0x0001  // Autonull command
                                             // 0000 0001 --> GLOB_CMD[0]=1

//--------------------------------

// MSC_CTRL BIT DESCRIPTIONS

#define MSC_CTRL_SELFTEST            0x0800
                    // Memory test (clears upon completion) at power on:1=enable, 0=disable
                                                          // 0000 1000 0000 0000

#define MSC_CTRL_SFTESTEN            0x0400
                    // Internal self-test enable (clears upon completion): 1 = enabled, 0 = disabled
                                                          // 0000 0100 0000 0000

#define MSC_CTRL_MNSELFTEST   0x0200
                    // Manual self-test, negative stimulus: 1 = enabled, 0 = disabled
                                                          // 0000 0010 0000 0000

#define MSC_CTRL_MPSFTESTEN   0x0100
// Manual self-test, positive stimulus:1=enable,0=disable
```

```
                                                                    // 0000 0001 0000 0000

#define MSC_LABCCTRL_SFTESTEN 0x0080
// Linear acceleration bias compensation for gyroscopes: 1 = enabled, 0 = disabled
                                                                    // 0000 0000 1000 0000

#define MSC_LAOACTRL_SFTESTEN 0x0040
// Linear accelerometer origin alignment: 1 = enabled, 0 = disabled
                                                                    // 0000 0000 0100 0000

#define MSC_CTRL_DATRDYEN 0x0004 // Data-ready enable: 1=enable, 0=disable
                                                                    // 0000 0000 0000 0100

#define MSC_CTRL_DATRDYPO 0x0002 // Data-ready polarity: 1=active high, 0=active low
                                                                    // 0000 0000 0000 0010

#define MSC_CTRL_DATRDYIO 0x0001 // Data-ready line select: 1=DIO2, 0=DIO1
                                                                    // 0000 0000 0000 0001

//--------------------------------

#endif /*ADIS_H_*/
```