LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

JARRUPENKIN ELEKTRONIIKKA

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Tietokone-elektroniikka
Opinnäytetyö
Kevät 2012
Joni Niiranen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

NIIRANEN, JONI: Jarrupenkin elektroniikka

Tietokone-elektroniikan opinnäytetyö, 29 sivua, 9 liitesivua

Kevät 2012

TIIVISTELMÄ

Opinnäytetyönä suunniteltiin ja rakennettiin elektroninen kortti, joka ottaa vastaan tietoa jarrupenkin antureilta ja prosessoinnin jälkeen syöttää antureilta saatujen tietojen mukaiset luvut tietokoneelle. Tietokone näyttää ruudulla saadut tiedot käyttäjälle: moottorin kierrosluku, lämpötilat, teho, vääntö, pakokaasun lambda ja sytytyksen ennakko. Työ tehtiin yksityiselle Petteri Valtosen suunnittelemalle ja rakentamalle jarrupenkille.

Suunnitteluosuus alkaa anturien tarkastelusta. Siinä tarkastellaan lähemmin käytettyjä antureita ja niiden tuottamia signaaleja, joita kortin tulee osata lukea. Antureja on kaiken kaikkiaan kahdeksan, ja niiltä tuleva tieto vaihtelee kahdenkymmenen millivoltin ja seitsämänkymmenen voltin välillä. Suunnitteluosuuden toisessa kohdassa käydään läpi käytetyt komponentit ja niiden toiminta hieman tarkemmin. Kerrotaan myös, miksi valittuihin komponentteihin on päädytty. Viimeinen suunnitteluosuus käsittelee kuinka kortin suunniteltiin ja mitä ohjelmaa käytettiin kortin suunnittelussa.

Ohjelmointiosuudessa tarkastellaan ensiksi kortin firmwaren luontia ja siihen käytettyjä ohjelmia. Osuudessa käydään myös läpi eri anturien vaatimat ohjelmaosuudet. Mukaan kuuluu itseään tasaisin väliajoin toistavia ohjelmaosuuksia ja keskeytyksestä aktivoituvia ohjelmia. Viimeisenä ohjelmointiosuudessa käydään läpi tietokoneen puolelle tullut ohjelma, joka näyttää tietokoneen ruudulla kortin vastaanottamia tietoja jarrupenkin antureilta. Siinä myös käydään hieman läpi asiaoita, joita olisi jatkossa tarkoitus lisätä ohjelmaan.

Testausosuudessa käydään läpi testauksessa käytetyt menetelmät. Viimeisenä on yhteenveto, jossa analysoidaan projektia yleisesti. Käydään läpi asiat, jotka tulisi tehdä toisin, jos projekti pitäisi tehdä uudestaan.

Avainsanat: suunnittelu, piirikaavio, piirilevyt, ohjelmointi, anturit, mikro-ohjaimet

Lahti University of Applied Sciences
Degree Programme in Information Technology

NIIRANEN, JONI: Electronic design for a dynamometer

Bachelor's Thesis in Computer Electronics, 29 pages, 9 appendices

Spring 2012

ABSTRACT

This Bachelor's Thesis revolves around designing and building an electronic card that takes information from sensors on a dynamometer and after processing the data feeds it to a computer which then shows the results on a computer screen: revolutions per minute, temperatures, power, torque, lambda and ignition advance. The thesis was made for Petteri Valtonen's home made dynamometer.

The designing part of the thesis starts by presenting sensors. Signals that the card needs to be abble to read and the used sensors are inspected in detail. There are eight sensors and information that they produce varies between twenty millivolts and seventy volts. The second part of designing part deals with the components that were used, describing in detail how they work. It also tells why these components were selected for the project. The last part covers designing the card and the program used to design the card.

The programming part first describes the creation of the firmware on the card and the programs used for doing it. The first part also goes through the software that was needed to get information in from the sensors. The part consists of programs that repeat themselves at regular intervals and programs that become active when there is an interrupt. The last part of presents the program that is on the computer that shows the data from the dynamometer's sensors that is collected by the card. It also discusses possible future additions to the program.

The second last part presents the testing methods used to test the card. The final part is a synopsis where the project is analyzed generally. It also mentions what would be done differently if the project were restarted.

Key words: designing, circuit diagram, circuit, programming, sensors, microcontrollers

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on luoda kortti, joka ottaa vastaan tietoa Petteri Valtosen luoman jarrupenkin antureilta. Jarrupenkki on Petteri Valtosen siviiliprojekti, joka on ollut valmiina jo reilun kymmenen vuotta. Laitteessa on seitsämän anturia, joiden antama tieto vaihtelee kahdenkymmenen millivoltin ja seitsämänkymmenen voltin välillä. Prosessorin valinnan jälkeen työn ensimmäinen vaihe on suunnitella jokaiselle anturille oma vastaanottoyksikkö, joka muuntaa tiedon tulemaan välillä nolla volttia ja viisi volttia. Toisena vaiheena on luoda antureille tarvittavat jännitteet, joita ne tarvitsevat toimiakseen. Kolmen anturin tiedot on saatava talteen niiden tapahtumishetkellä, joten nämä tapahtumat ovat tärkeysjärjestyksessä muiden anturien edellä. Petterin pyynnöstä työssä käytetään Burr-Brown Corporationin ina125-vahvistinta. Muilta osin on vapaat kädet tehdä, mitä itse haluaa.

Kortin ohjelmoinnissa pitää tehdä antureiden tietoja keräävä ohjelmallinen silmukka ja kolmen anturin kohdalla tiedon keräyksen tulee tapahtua antureiden ilmoittamalla ajalla, joten nämä tapahtumat pitää hoitaa heti niiden ilmaannuttua. Koodi siis sisältää keskeytysten hallinnan. Korttia ohjelmoitaessa on jo valmiiksi mietittävä, millä tapaa tieto lähetetään tietokoneelle.

Tietokoneen ja kortin väliseksi yhteydeksi valittu USB toimii hyvin tiedon siirtäjänä kortin ja tietokoneen välillä. Sen kautta on helppo myös ohjelmoida mikro-ohjain ja tarvittaessa ohjelmoida uudelleen. Valittu mikro-ohjain lupaa sata tuhatta poisto- ja uudelleenkirjoituskierrosta. Tietokoneen puolen ohjelman on tarkoitus vain näyttää ja tallentaa testausdataa, jota suunnittelemani piiri antaa.

2 SUUNNITTELU

Suunnittelu alkoi kahden dokumentin lukemisesta, joissa oli kerrottu, mitä työssä olisi tarkoitus tehdä ja millaista tietoa anturit antavat. Dokumentit kertovat myös, mikä on antureiden tarkoitus varsinaisessa laitteessa.

Varsinainen laite on jarrupenkki, jolla mitataan auton moottorin arvoja: teho, lämpöjä, sytytyksen ajat ja kierrosnopeus. Projektin tavoitteena on tehdä kortti, joka ottaa antureilta tiedon vastaan ja käsittelee tiedon niin, että tietokone voi näyttää esimerkiksi kierrosluvun suoraan numeroina, eikä sen tarvitse enää laskea sen arvoa kierrokseen kuluvasta ajasta.

Aluksi mietin antureilta tulevaa tietoa ja sitä kuinka saan sen muutettua mikro-ohjaimen ymmärtämään muotoon. Mikro-ohjain ymmärtää vain nollan ja viiden voltin välillä olevaa tietoa, jonka se ADC-pinneillään ottaa vastaan. ADC-pinnit käytännössä katsovat kuinka paljon sisääntuleva jännite on ja antavat sille kahdeksan bittisen numeerisen arvon. Nolla volttia on arvoltaan nolla ja viisi volttia on arvoltaan 255, joten antureilta tuleva tieto täytyy muuttaa välille nolla ja viisi volttia, jotta ADC-pinnit osaavat antaa mahdollisimman tarkkoja tietoja. Yli viiden voltin sisään tulevia jännitteitä täytyy pienentää viiteen volttiin ja alle viiden voltin jännitteet kasvattaa viiteen volttiin lineaarisesti.

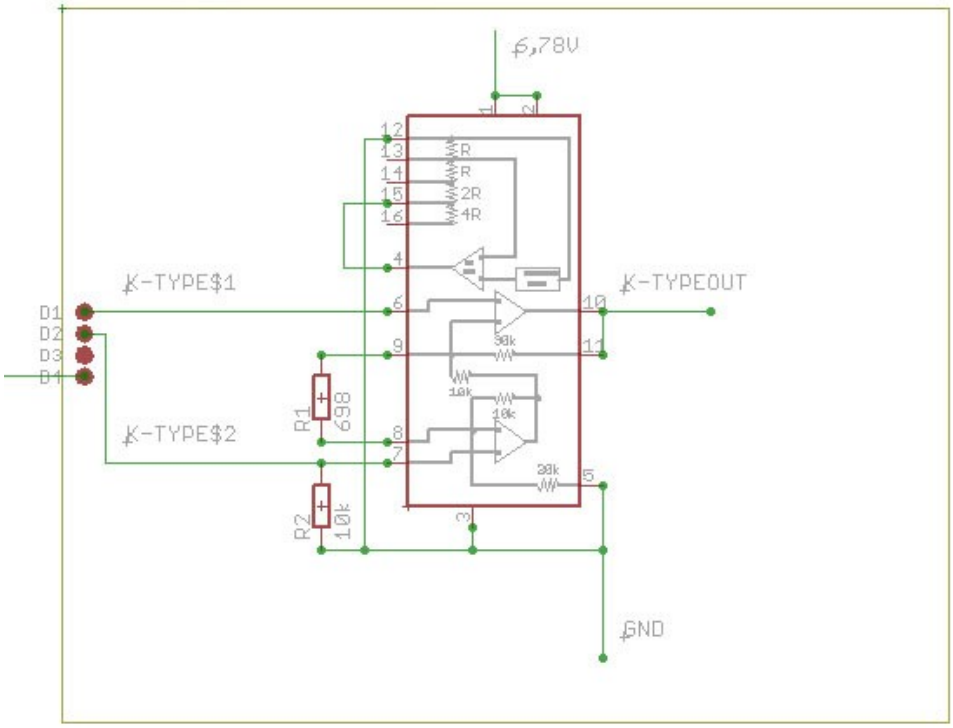
2.1 Anturit

2.1.1 Lämpötila-anturit

K-tyypin termopari

”Termopareja valmistetaan monista metallipareista, joista toinen pidetään tiedetyssä, eli ns. vertailulämpötilassa ja toista tutkittavassa lämpötilassa. Parin lämpötilaeron noustessa niiden välinen jännite kasvaa. Mitattava lämpötila saadaan määritettyä jännite-eron avulla, kun tiedetään kyseisen metalliparin herkkyys eli jännitteen riippuvuus lämpötilasta. Metallipareissa käytetyt materiaalit vaikuttavat syntyvään jännite-eroon.” K-tyypin termoparin muodostaa chromel (nikkelin ja kromin seos) ja alumel (nikkelin ja alumiinin seos). (Termopari 2012.)

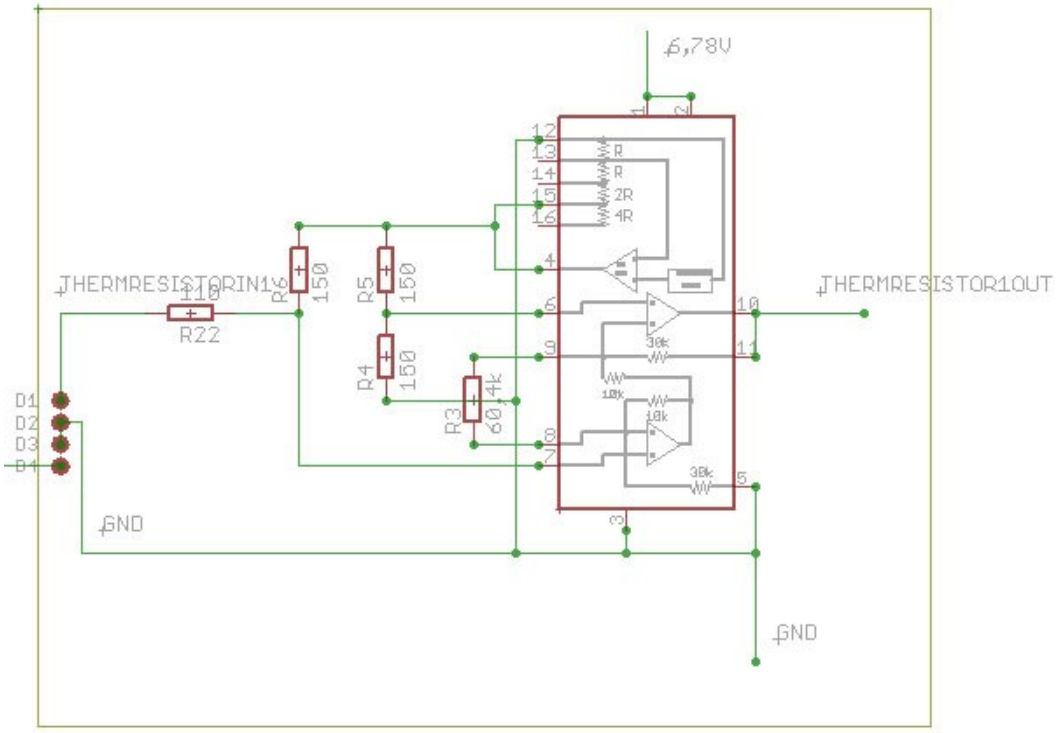
Jarrupenkki sisältää yhden K-tyypin termoparin, jolla mitataan pakokaasun lämpötilaa. Maksimilämpötila, jota se pystyy mittaamaan, on 1300 celciusta. Se syöttää datan kahta linjaa pitkin, mutta syötetyt jännitteet ovat varsin heikkoja. Molemmat tulot ovat yhtä suuria, mutta vastakkaista merkkiä. Noin kaksikymmentä millivolttia on maksimi, mitä irtoaa termoparista. Sitä pitää vahvistaa huomattavasti, että viiden voltin ADC-pinni pystyy määrittämään sille arvoja. Kuviossa 1 on termoanturin kytkentäkaavio.



KUVIO 1. K-tyypin termoanturin kytkentäkaavio

Lämpötilaresistanssi

Jarrupenkissä on kaksi lämmön mukaan resistanssiansa muuttavaa vastusta. Niillä mitataan öljyjen lämpötiloja, mutta ne eivät pysty mittaamaan yli 350 celciuksen lämpötiloja. Resistanssi on viidenkymmenen ohmin luokkaa, kun lämpötila on nolla celciusta, ja maksimilämpötilassa se on 375 ohmin luokkaa. Käytin Wheatstonen siltaa saadakseni kaksi eri jännitettä. Toisen puolen jännite pysyy vakiona ja lämpötilaresistanssin puoleinen jännite kasvaa. Resistanssien koot valitsin niin, että jännite-ero ei kasvaisi liian suureksi, että ina125 pystyy vielä vahvistamaan jännitettä ilman, että maksimi- ja minimilämpötilat jäisivät pois. Apuresistanssilla säädettiin jännite-ero nollaan, kun lämpötila on nolla celciusta, jotta vahvistettu jännite-ero olisi nolla volttia, kun lämpötila on nolla celciusta ja kasvaisi tasaisesti viiteen volttiin, kun lämpötila on 350 celciusta. Kuviossa 2 sisältää lämpötilaresistanssin kytkentäkaavio.

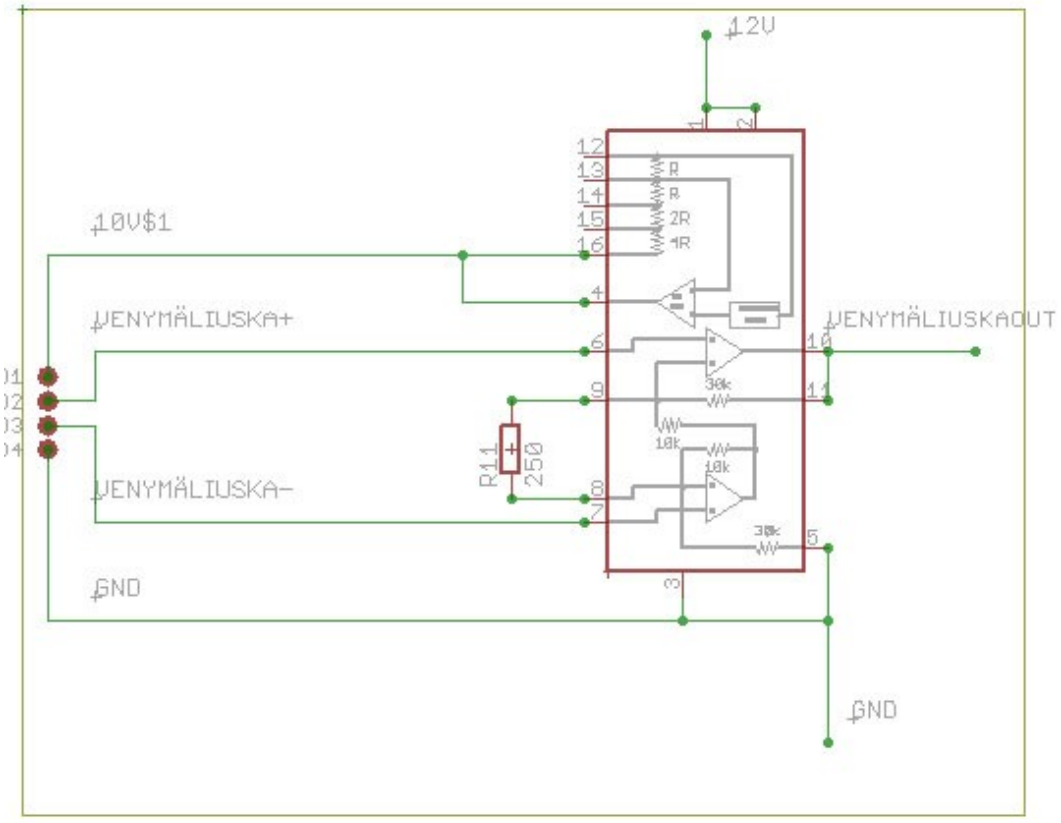


KUVIO 2. Lämpötilaresistanssin kytkentäkaavio

2.1.2 Venymäliuska

”Venymäliuska-anturi on materiaalin venymisen mittaamiseen tarkoitettu anturi. Sen toiminta perustuu siihen, että anturissa olevan metallijohteen muoto muuttuu venytyksen takia ja siksi myös johteen resistanssi muuttuu.” (Venymäliuska-anturi 2012.)

Venymäliuskalla mitataan moottorin tuottamaa voimaa. Se tarvitsee kymmenen voltin jännitteen toimiakseen, jonka luomiseen käytetään kahdentoista voltin jännitettä ja ina125:n tarkkuusreferenssiä, josta saa tarkan kymmenen voltin jännitteen. Ulostulona venymäliuska ei anna kovinkaan suuria arvoja. Jännite vaihtelee nollasta kahteenkymmeneen millivolttiin. Siitä syystä kytkin ina125 vahvistuksen siten, että se antaa 250-kertaisen vahvistuksen, jotta kahdenkymmenen millivoltin sisääntulo aiheuttaa viiden voltin ulostulon. Kuviossa 3 näyttää venymäliuskan kytkentäkaavion.



KUVIO 3. Venymäliuskan kytkentäkaavio

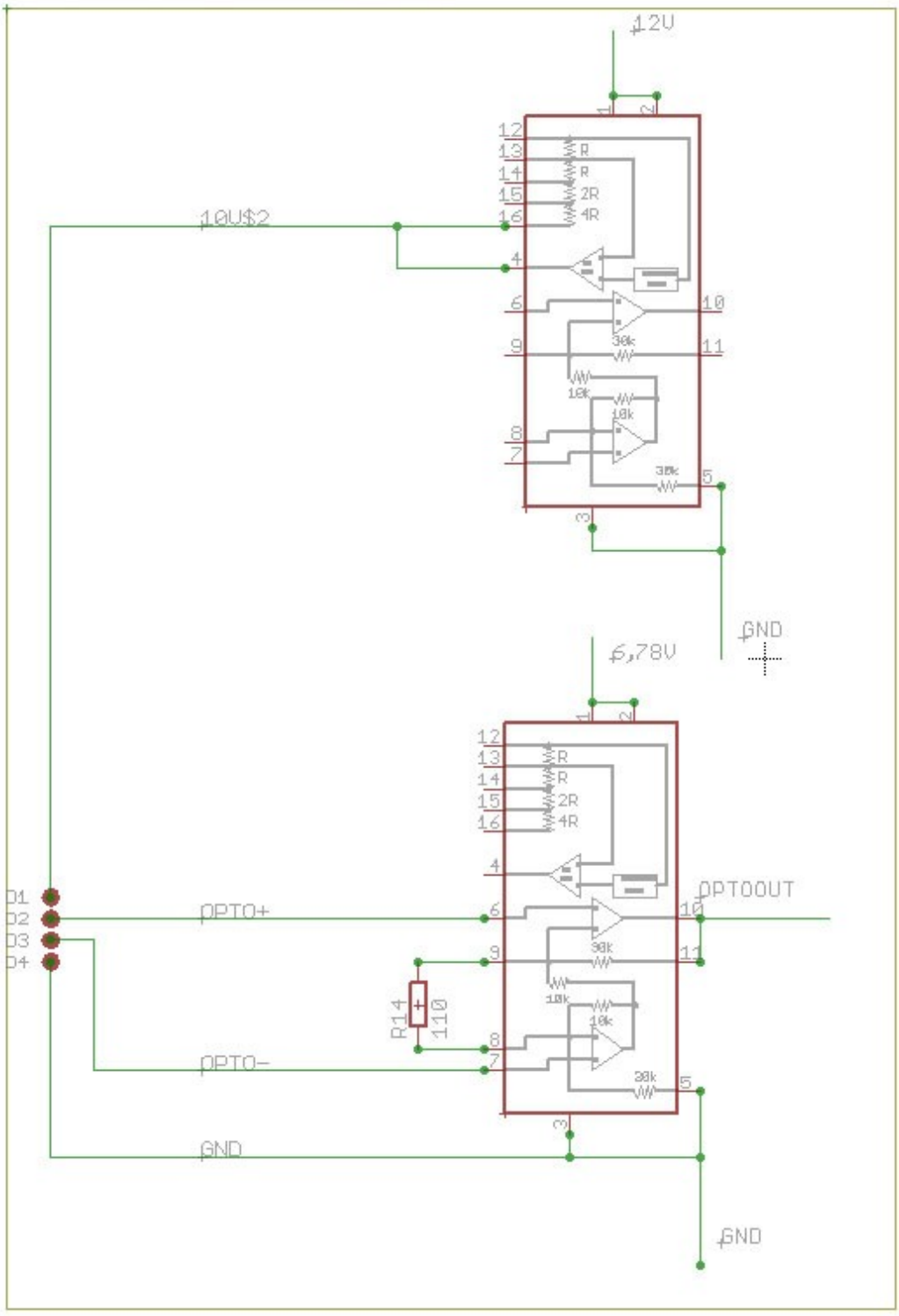
2.1.3 Lambda-anturi

”Lambda-anturi on happitunnistin, joka on sijoitettu pakoputken alkuun. Se on yksi moottorinohjausjärjestelmän keskeinen osa. Se mittaa pakokaasun jäännöshapen määrää ja välittää siitä tiedon moottorinohjausjärjestelmälle, joka osaa muuttaa ilman ja bensiinin seoksen moottorille sopivaksi. Lambda-anturi on yleistynyt katalysaattorin mukana, ja se on tärkeä esimerkiksi typen oksidien määrän vähentämisessä. Lambda-anturi parantaa myös moottorin hyötysuhdetta.” (Lambda-anturi 2012.)

Lambda-anturi oli alkuperäisesti kaikkein vaikein tapaus. Se tuotti epälineaarista käyrää nollan ja 1,5 voltin väliltä, mutta onneksi se vaihtui lineaariseen anturiin välillä nolla ja viisi volttia. Näin ollen siitä tulikin helpoin mahdollinen anturi. Ei tarvinnut muuta tehdä kuin kytkeä anturi suoraan ADC-pinniin.

2.1.4 Optinen anturi

Optisella anturilla mitataan kierrosnopeutta ja se säädetään siten, että siitä näkee, milloin ykkössylinteri on nollakohdassaan. Tätä tietoa tarvitaan laskettaessa sytytyksen ennakko. Optinen anturi tarvitsee myös kymmenen voltin jännitteen toimiakseen, jonka ina125 antaa, ja toista ina125 käytän vastaanottamaan optisen anturin tuottamat jännitteet, jotka tulevat vastakkaismerkkisinä. Aina kun anturi huomaa kohdan, sen ulostulot vaihtavat tilaa. Toinen ulostuloista on normaalisti kytketty maahan ja toinen on normaalisti kytketty käyttöjännitteeseen (Contrast sensor with white emission 2011). Kuviossa 4 on optisen anturin kytkentäkaavio.

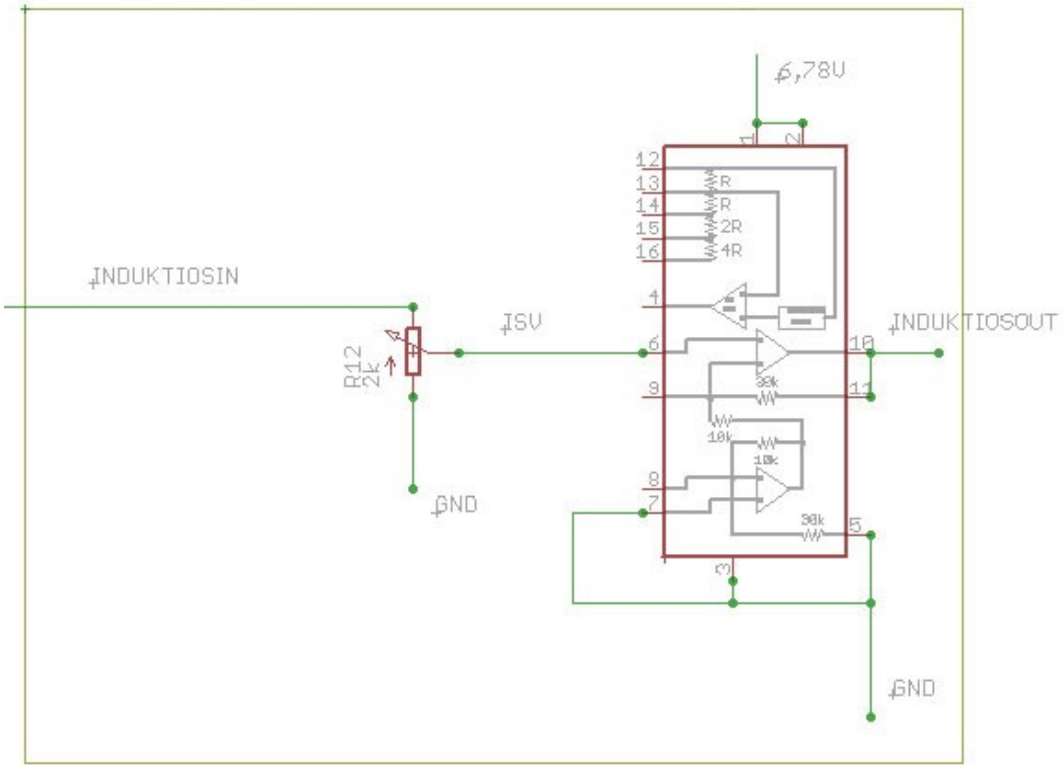


KUVIO 4. Optisen anturin kytkentäkaavio

2.1.5 Induktiiviset anturit

Induktiivinen silmukka

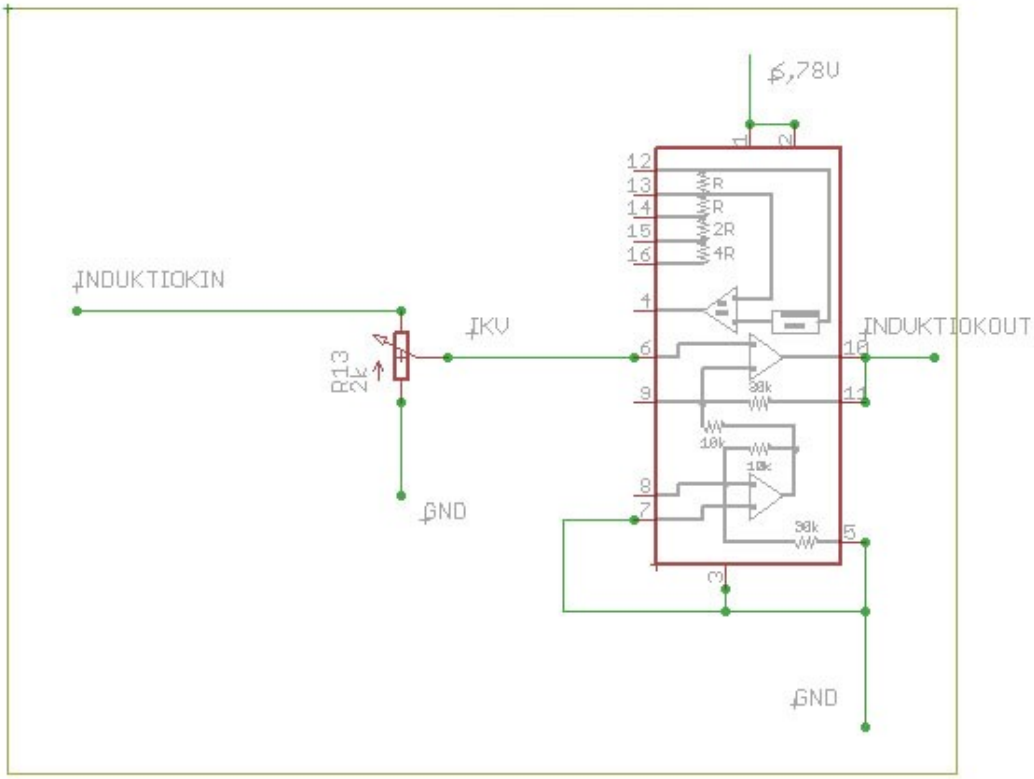
Induktiivinen silmukka on otettu ennakkolampusta, jolla saadaan puolanjohdon läpi otettua kipinätieto. Jännite on noin seitsämänkymmenen voltin pulssi sekä mukana kolme kappaletta häiriöpulsseja muista tulpista. Häiriöt ovat noin kymmenestä kahteenkymmeneen volttia. Tätä varten tarvitaan potentiometri, jolla saadaan laskettua seitsämänkymmenen voltin jännite viiden voltin lähimaastoon. Sillä tiputuksella häiriöimpulssit ovat alle kaksi volttia eivätkä enää häiritse. Ina125 ottaa noin viiden voltin signaalin ja tekee siitä maksimissaan viiden voltin signaalin ja syöttää sen mikro-ohjaimen keskeytyspinniin. Saatua tietoa tarvitaan sytytyksen ennakon laskemiseen. Kuviossa 5 on induktiivisen silmukan kytkentäkaavio.



KUVIO 5. Induktiivisen silmukan kytkentäkaavio

Induktiivinen kytkin

Induktiivinen kytkin tarkkailee, milloin venymäliuskan tieto täytyy ottaa talteen. Se sai käytännössä samat kytkennät kuin induktiivinen silmukka, vaikka sen tuottama jännite onkin vain kaksikymmentäviisi volttia, mutta samat säännöt pätevät tällekin. Potentiometrillä jännite säädetään noin viiteen volttiin ja ina125 tekee siitä maksimissaan viiden voltin signaalin ja syöttää sen mikro-ohjaimen keskeytyspinniin, minkä jälkeen mikro-ohjain ottaa venymäliuskan tiedon sisään. Kuviossa 6 on induktiivisen kytkimen kytkentäkaavio.



KUVIO 6. Induktiivisen kytkimen kytkentäkaavio

2.2 Komponentit

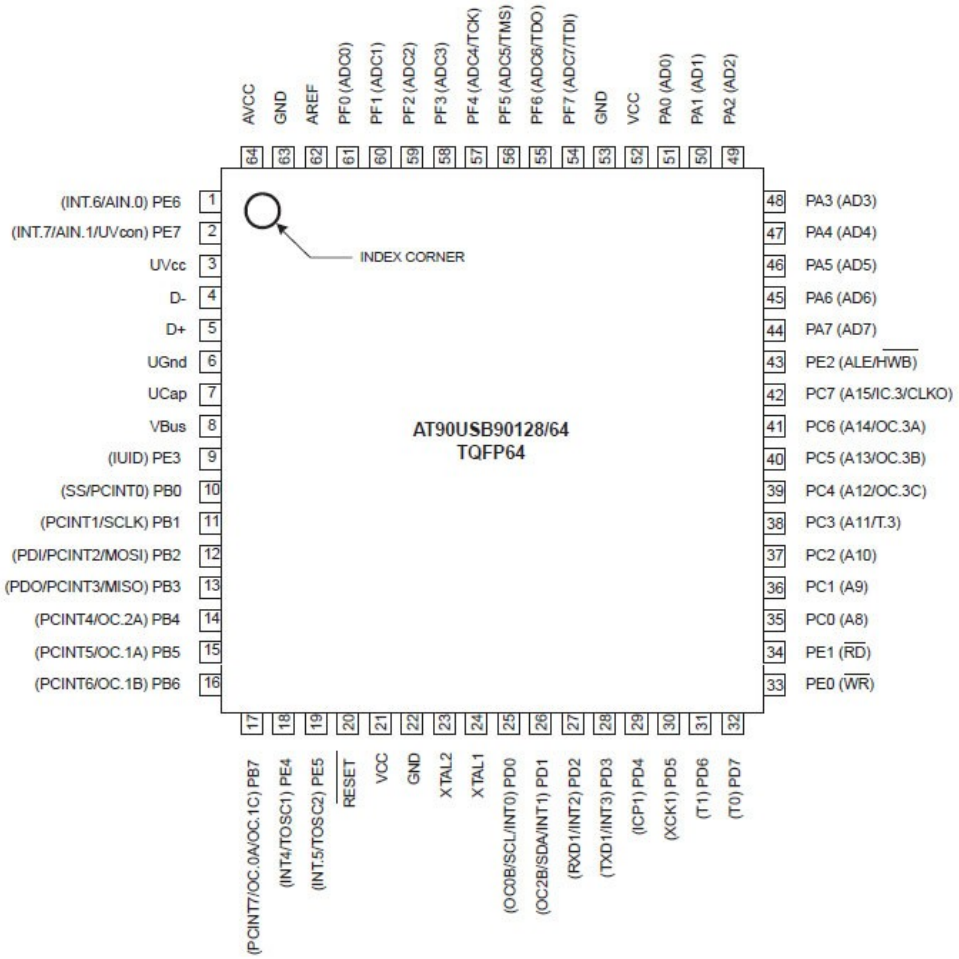
Pyrin käyttämään pintaliitoskomponentteja, mutta muutama osa on läpijuotettavaa mallia. Sisään- ja ulostuloterminaalit esimerkiksi ovat läpijuotettavat, mutta kaikki kondensaattorit, nappulat, vastukset yhtä lukuunottamatta ja tietenkin mikro-ohjain ovat pintaliitoskomponentteja.

Muutaman komponentin tilauksessa tekemäni virheen takia jouduin hieman soveltamaan, että sain ne juotettua paikoilleen. Esimerkiksi potentiometrejä tilatessani en muistanut tarkistaa, ovatko ne oikeanlaiset kortilla oleville paikoilleen, ja tilatessani vastuksia virheellisesti tilasin fyysiseltä kooltaan liian pieniä vastuksia, jotka kuitenkin sain kolvattua paikalleen.

2.2.1 Mikro-ohjain

”Mikrokontrolleri eli mikro-ohjain on mikropiiri eli IC-piiri, jossa on mikroprosessori ja joitain muisti- ja liityntälohkoja. Mikrokontrollereita käytetään sulautetuissa järjestelmissä, eli melkeinpä kaikissa taskulamppua monimutkaisemmissa elektroniikkalaitteissa. Esimerkiksi television, pesukoneen, mikroaaltouunin ja digitaalisen lämpömittarin ohjaustoiminnot ovat usein mikrokontrollerilla toteutettuja.”(Mikrokontrolleri 2012.)

Aluksi valitsin Microchip Technology Inc:n PIC24FJ64GB004-mikro-ohjaimen, mutta myöhemmin huomasin, että se toimi ainoastaan nollan ja 3,5 voltin alueella, ja se ei riittänyt minulle, koska olin jo suunnitellut kaiken toimimaan nollan ja 5 voltin alueella. Olisin voinut tietysti laskea jännitteet 3,5 volttiin, mutta ei ollut tarvittavia tietoja, että olisin saanut mikro-ohjaimen toimimaan ja ohjelmoitua usb-linjan kautta, joten vaihdoin ennalta tuntemaani ATMEL - AT90USB1287-AU mikro-ohjaimeen, jossa on hieman enemmän pinnejä (kuvio 7). Olen aikaisemminkin käyttänyt tätä ja tiesin, että sen pystyy ohjelmoimaan suoraan usb:n kautta ja siinä on toiminta-alue juuri sopiva viiden voltin sisääntuloille. (AVR Microcontroller with 64/128K Bytes of ISP Flash and USB Controller 2012.)



KUVIO 7. AT90USB1287:n pinnit (AVR Microcontroller with 64/128K Bytes of ISP Flash and USB Controller 2012)

2.2.2 48 MHz:n kideoskillaattori

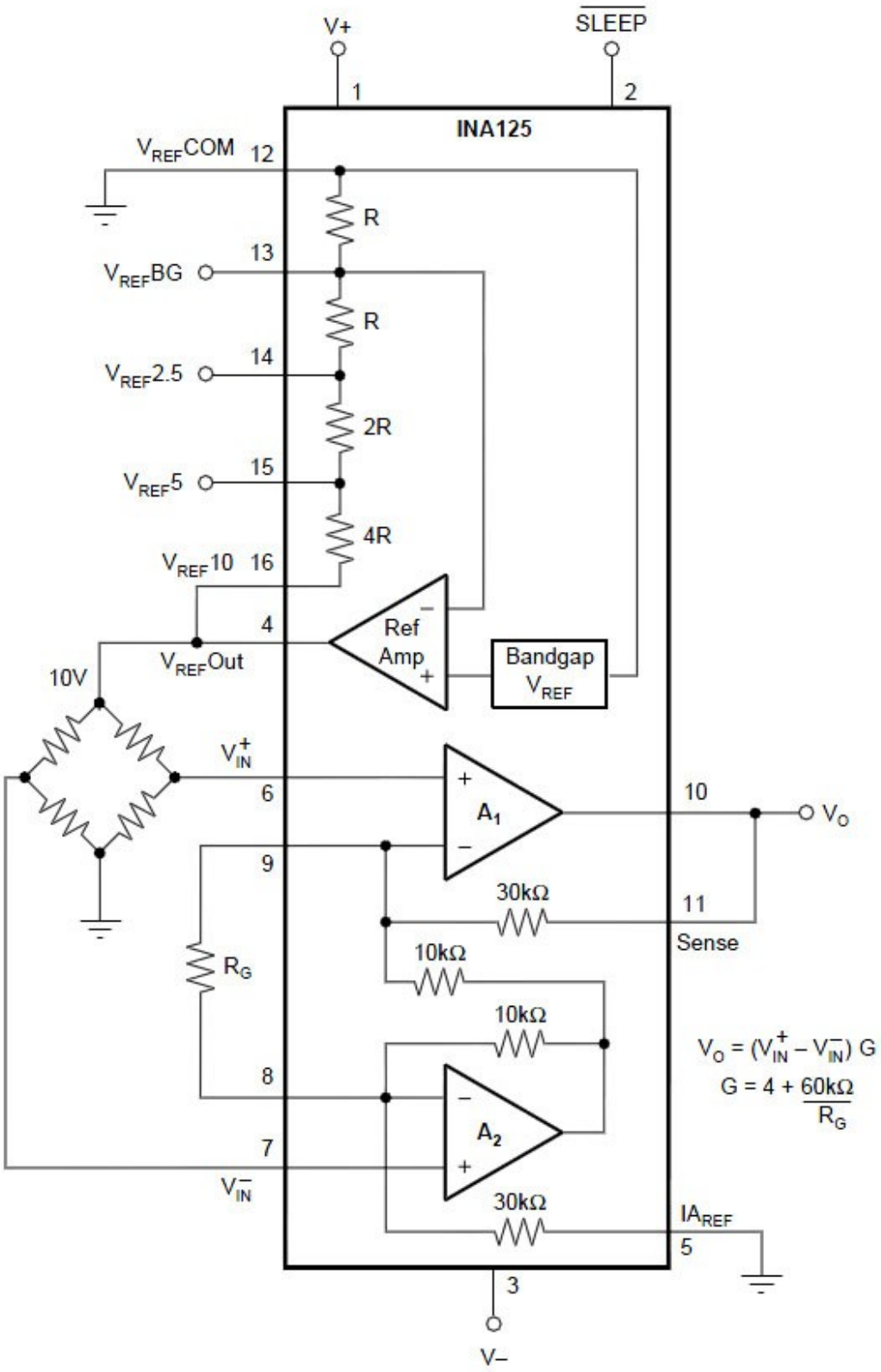
”Digitaalielektroniikassa käytetään usein kideoskillaattoreita, joiden taajuus pysyy oikeana tyypillisesti muutaman miljoonasosan tarkkuudella. Kideoskillaattorin keskeinen komponentti on pietsosähköinen kide, joka on yleensä kvartsia. Kytkennässä kide värähtelee mekaanisesti jollain ominaisvärähtelytaajuudellaan ja toimii lähes suuri-induktanssisen kelan tavoin. Suuren induktanssin ja pienen sisäisen resistanssin vuoksi kideoskillaattorin hyvyysluku on suuri ja siten toimintataajuus erittäin vakaa. Vakautta voidaan edelleen parantaa pitämällä kiteen lämpötilaa vakiona.”. (Oskillaattori 2012.)

Mikro-ohjain vaatii 48 Mhz:n kideoskillaattorin USB:n Full-Speed-toimintoa varten. Valitsin suoraan mikro-ohjaimen datalehden mukaisen kiteen.

2.2.3 Ina125

Ina125 on instrumentointivahvistin (kuvio 8). ”Instrumentointivahvistimet ovat säätötekniikassa ja teollisuuden prosessinohjauksessa käytettäviä vahvistimia, joita käytetään mittaus- ja ohjaussignaalien vahvistamiseen, esimerkiksi anturisignaalien vahvistamiseen. Instrumentointivahvistimilta vaaditaan usein hyviä tasavirtavahvistusominaisuuksia ja vähäistä kohinaa. Lisäksi niissä tulo- ja lähtöpuolet on joskus galvaanisesti erotettu toisistaan joko optoisolaattorin tai muuntajakytkennän avulla.” (Vahvistin 2012.)

Ina125 käsittelee melkein jokaista signaalia jossain välissä. Se pystyy luomaan referenssijännitteet viidelle voltille ja kymmenelle voltille. Se kestää suuriakin jännitteitä ja ehkäisee suurten jännitteiden pääsyn mikro-ohjaimelle. Ina125-pinnit on suojattu neljäänkymmeneen volttiin asti, joten antureista vain yksi pystyy synnyttämään jännitteen, joka rikkoo ina125:n. (Instrumentation amplifier 2011.)

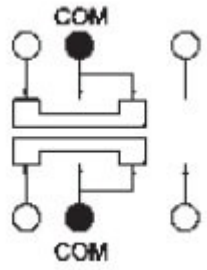


KUVIO 8. Kaavio ina125 sisäisestä toiminnasta (Instrumentation amplifier 2011)

2.2.4 Nappulat

On/Off-kytkin

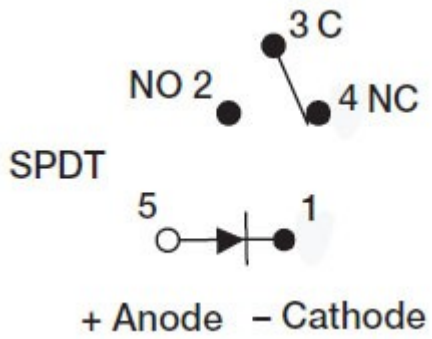
Kytkimessä on kahdeksan pinniä, joista yhden pään pari ei ole käytössä. Keskimmäiseen pinnipariin on johdettu viiden voltin jännite usb:n puolelta. Toinen puoli kytkintä ohjaa viiden voltin jännitteen kaikkialle muualle ja toinen puoli johtaa viisi volttia joko punaiselle tai vihreälle ledille. Punaisen ledin palaessa viiden voltin jännite ei pääse kytkintä pidemmälle ja vihreän palaessa viiden voltin jännite pääsee kytkimen läpi. Kuvio 9 näyttää, millainen kytkentäkaavio kytkimellä on.



KUVIO 9. Kytkimen kytkentäkaavio (Microminiature Slide Switches 2011)

Reset-nappulat

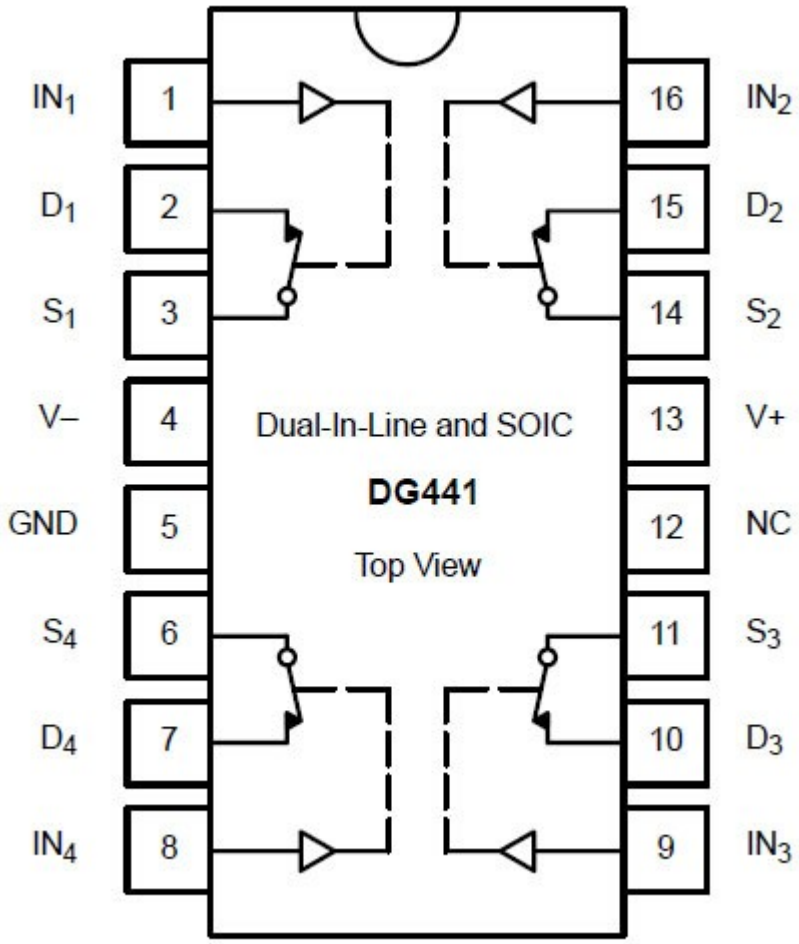
Toinen on kytketty ohjaimen reset-pinniin ja toinen HWD-pinniin, jotka molemmat täytyy aktivoida, jos haluaa ohjelmoida mikro-ohjainta usb:n kautta. Tarvittaessa pelkkää reset-nappulaa painamalla saa resetoitua mikro-ohjaimen ohjelman alkutilaan. Nappuloissa on viisi pinniä, joista kaksi on varattu ledeille ja yksi pinneistä vaihtaa yhdyspistettään kahden muun pinnin kanssa (kuvio 10). Molemmissa nappuloissa on ledit sisällä ja ledit palavat koko ajan, kun virta on kytketty korttiin.



KUVIO 10. Reset-nappulan pinnien toimintakuva (Right Angle Illuminated Latching Pushbutton 2012)

2.2.5 Analoginen kytkin

Analoginen kytkin (kuvio 11) päästää kahdentoista voltin jännitteen sisään vain, kun siihen tulee on/off-kytkimeltä viiden voltin jännite, joten jos on/off-kytkin on asennossa off, ei kahdentoista voltin jännitekään pääse kortille. (Quad SPST CMOS Analog Switches 2012.)

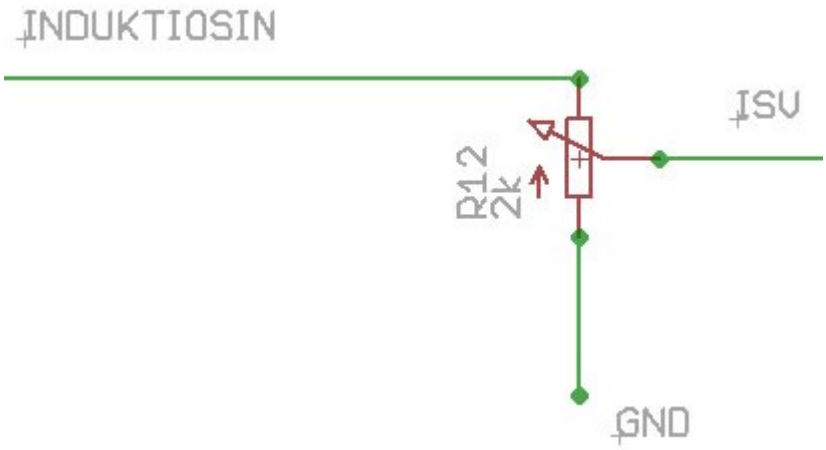


KUVIO 11. Analogiakytkin (Quad SPST CMOS Analago Switches 2012)

2.2.6 Potentiometrit

”Potentiometri (puhek. ”potikka”) on elektroniikassa kolminapainen säätövastus, jota voidaan käyttää myös jännitteenjakajana. Potentiometrin keskeinen ominaisuus on portaaton säädettävyys.” (Potentiometri 2012.)

Kaksi potentiometriä kontrolloi sisään tulevia suuria jännitteitä. Molemmilla on samanlainen peruskytkentä sisääntuloon, maahan ja ina125:lle. Säätämällä potentiometriä säädetään, kuinka paljon sisääntuleva jännite pienenee. Kuvio 12 esittää induktiosilmukan sisääntulon pienennystä potentiometrillä.



KUVIO 12. Potentiometrien kytkentäkaavio

2.2.7 Led

”LED (engl. Light-Emitting Diode) eli valodiodi, loistediodi tai ledi on puolijohdekomponentti, joka säteilee valoa, kun sen läpi johdetaan sähkövirta. Lediä kutsutaan arkikielessä myös valodiodiksi, vaikka historiallisesti valodiodi eli fotodiodi tarkoittaa valolle herkkää diodia. Ledien valmistusmateriaali (monesti gallium-yhdisteitä) määrää komponentin lähettämän valon värin, jota voidaan edelleen muokata ledin pintaan lisätyillä kalvoilla ja pinnoitteilla. Kaukosäätimissä käytetään yleensä ledejä, jotka säteilevät infrapunasäteilyä. Yksittäisen ledin emittoiman valon spektri on yleensä varsin kapea eli säteily on

lähes monokromaattista, mutta useita ledejä voidaan pakata samaan koteloon yhdistelmävärien saamiseksi.” (LED 2012.)

Valittu ledi on kaksivärinen, ja se syttyy vihreäksi, kun on/off nappula on ”On”-asennossa, ja palaa punaisena, kun se on ”Off”-asennossa. Ledin tilaamisessa ilmeni ongelmia, koska lediä oli vain Amerikassa ja vastaavanlaisella kytkennällä ei ollut yhtään lediä Euroopan puolella. Jouduin kytkemään lähimmän vastineledin langalla kiinni maihin.

2.2.8 Terminaalit

12 voltin terminaali

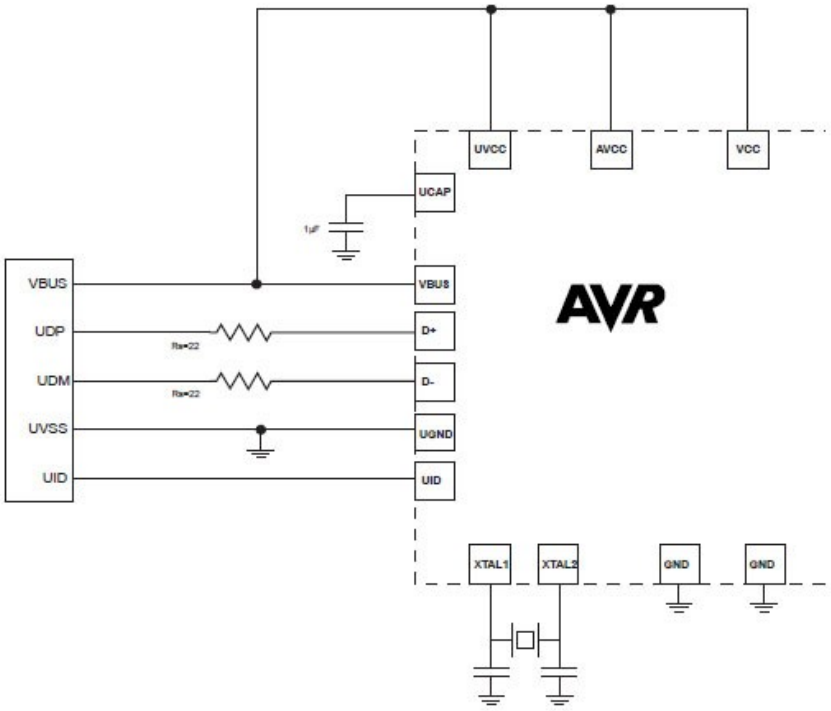
Kahdentoista voltin sisääntuloksi valitsin 6,1 millimetrin aukolla ja 2,1 millimetrin pinnillä varustetun pintaliitoskomponentin, jossa on 2 tappia, jotka tarvitsevat reiät kortille. Tietysti terminaaliin sopiva kahdentoista voltin adapteri oli myös tarpeellinen. Kahdentoista voltin jännitteestä tuotetaan 6,78 voltin jännite, jota ina125:set tarvitsevat toimiakseen viiden voltin jännitteellä.

Anturien sisääntulot

Antureille otin neljäpinniset läpijuotettavat portit, jotka kiristyvät johtoihin ruuvilla. Tämä helpottaa johdon vaihtamista toiseen ja pitää myös tarvittaessa tiukasti kiinni. Sisääntuloja ei ole kuin viisi, vaikka antureita onkin kahdeksan. Osa antureista ei tarvitse kuin yhden tai kaksi pinniä portilta, joten laitoin useampaan porttiin enemmän kuin yhden anturin.

Mini USB Type B

Mini USB on liitin, jonka kautta mikro-ohjain ohjelmoidaan ja jonka kautta kortin keräämät tiedot välitetään tietokoneelle. Kuvio 13 näyttää käyttämäni liitännän.



KUVIO 13. Suositeltu USB liitäntä mikro-ohjaimelle (AVR Microcontroller with 64/128K Bytes of ISP Flash and USB Controller 2012)

2.3 Kortti

Käytin kortin suunnittelussa EAGLE:n (Easily Applicable Graphical Layout Editor) ohjelmaa. Eri komponenttien datalehtien opiskelun jälkeen loin Eaglella oman kirjaston, jonne loin kaikki puuttuvat komponentit, kuten mikro-ohjaimen ja ina125:set. Liitteenä on kortin kytkentäkaavio.

Aloitin suunnitelun tekemällä korttiin kaikki sisääntulot antureilta ja suunnittelemalla niille vastaanoton siten, että kaikki olisivat jännitealueella nollasta viiteen volttiin. Tein jokaiselle anturille oman yksikön, jotta saisin hallittua niitä vähän helpommin.

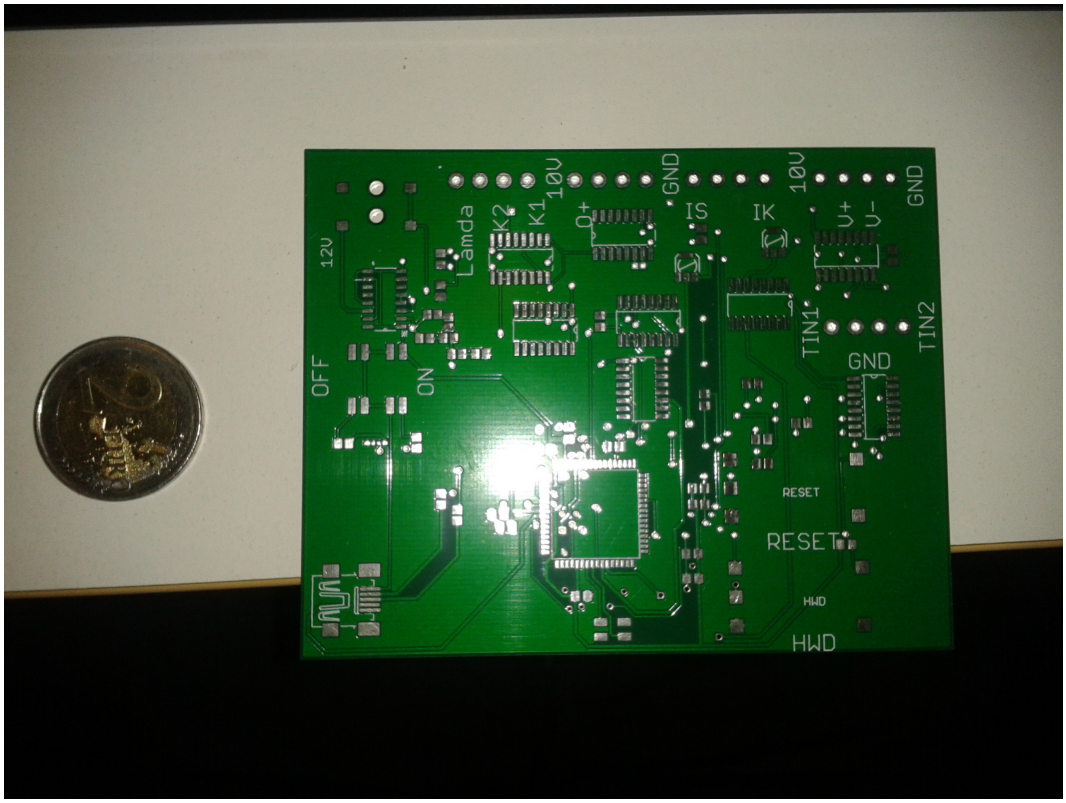
Pari antureista vaatii ulostulona tarkan kymmenen voltin jännitteen, jonka olin päättänyt luoda kahdentoista voltin jännitteestä. Käytin hyödykseni ina125:sia, joilla saa luotua tarkan kymmenen voltin jännitteen, kunhan positiivinen sisääntulo on 1,25 volttia suurempi kuin käytetty referenssijännite eli tässä tapauksessa 11,25 volttia.

Myös kahdentoista voltin jännitelähde vaatii mielestäni hallintaa, joten sille tulee analoginen portti, jonka avulla kortti ehkäisee kahdentoista voltin sisäänpääsyn piirille ilman, että se on kytkettynä USB-porttiin. Kahdentoista voltin jännitelähteestä loin myös 6,75 voltin jännitteen, jota ina125:set tarvitsevat toimiakseen viidellä voltilla.

Mikro-ohjaimen kytkennät tein datalehden suositusten mukaan. USB-portin ja mikro-ohjaimen väliin laitoin pari vastusta ja viiden voltin jännite suoraan mikrokontrollerin käyttöjännitteeksi virtakytkimen läpi. Mikro-ohjain tarvitsi myös kaksi nappia, joita käyttämällä saa mikro-ohjaimen ajettua ohjelmointitilaan. Nappuloille käytin suositeltuja kytkentöjä datalehdestä.

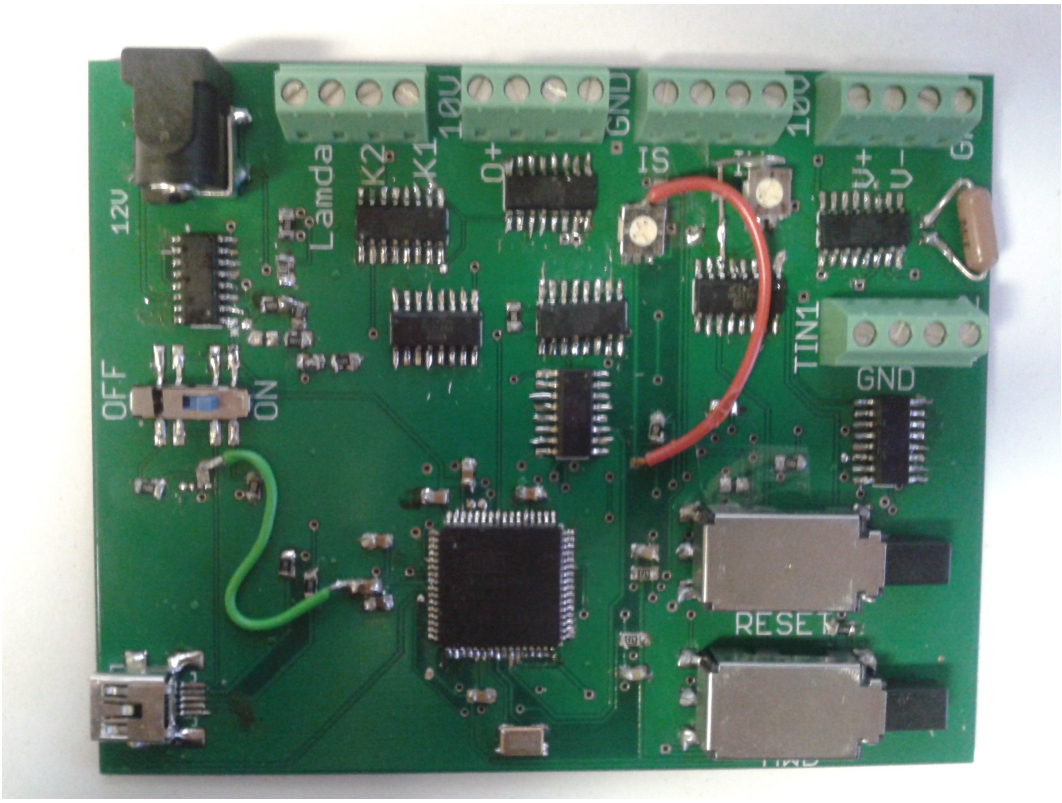
3 KASAUS

Tilasin kortin (kuvio 14) suoraan BatchPCB.comista, josta sain sen valmiiksi rei'itettynä ja kirjoituksien kanssa. Ei olisi edes ollut mahdollista tehdä levyä itse ja näin sai paljon huolitellumman lopputuloksen. Positiivinen yllätys oli, kun läpivientien rei'iät olivat itseasiassa jo valmiiksi läpivientejä. Oletukseni oli, että korttiin olisi vain porattu reiät valmiiksi ja joutuisin tekemään läpiviennit itse. Asia valkeni minulle, kun olin rupeamassa tekemään läpivientejä, mutta jostain syystä kuitenkin kokeilin ensin, kulkeeko virta yläpinnalta alapinnalle, ja kulkihan se.



KUVIO 14. Piirilevy ilman komponentteja

Osien juottaminen levylle ei ollut monimutkainen tehtävä, mutta aikaa vievä. Edellisestä kolvaus kerrastani oli kulunut aikaa, joten alkuun kolvausjälki oli aika ruman näköistä ja tilausvaiheessa sattuneen virheen takia tilasin vahingossa fyysisesti liian pieniä osia. Liian pienet osat sai kyllä kolvattua paikoilleen, mutta lopputulos näyttää huonolta (kuvio 15). Onneksi kriittiset osat, kuten mikro-ohjain ja nappulat, olivat oikean kokoisia.



KUVIO 15. Valmis piirilevy

4 OHJELMOINTI

4.1 Kortti

Kortin ohjelman kirjoitin AVR studio 5:llä. En tehnyt tällä ohjelmalla muuta kuin kirjoitin C-koodia. Kun C-koodi oli valmiina testaukseen, käytin winavr-ohjelmaa komentokehotteessa: yksi komento koodin kasaukselle ja toinen koodin kortille asennukselle. (WinAVR 2012.)

Kortin ohjelmassa on kaksi eri anturien tietoja talteen ottavaa tapaa. Ensimmäinen on itseään toistava ohjelma, joka tarkastelee vuorotellen antureita, joilta saatava tieto ei ole aikakriittistä, ja toinen on keskeytysohjelmat, jotka käynnistyvät vain kun anturi sitä vaatii.

Koodia en tehnyt kokonaan yksin, koska internetistä löytyy USB-laitteille valmiita koodinpätkiä. Käytin LUFA-koodipaketin General HID-laitetta. Aluksi minulla meni yli viikko ennen kuin sain kortin lähettämään dataa tietokoneelle. Datan lähetys tietokoneelle päin oli yhdestä rivistä koko LUFA-tietokannassa. (LUFA 2012.)

Aloitin HID-näppäimistökoodilla, jota muokkasin ensiksi työntämään USB läpi ennalta määrättyjä näppäimistön painalluksia muistioon. Saatuani sen toimimaan siirryin käyttämään Generic HID -koodia, jonka laitoin lähettämään tietokoneelle tietoa paketeissa. Näitä tietoja varten tarvitsin tietokoneelle ohjelman, joka osasi vastaanottaa niitä.

Varsinainen lopullinen koodi ei ole vielä valmis, mutta valmistunee, kunhan saan ina125:set korttiin kiinni ja pääsen testaamaan sitä. Kortin ohjelmoinnin pitäisi olla suhteellisen helppoa, käytännössä ohjelmasta puuttuu vain kalibrointi.

4.1.1 Ensimmäinen tapa ottaa tietoa antureilta

Tämä ohjelman pätkä toistaa itseään, kunnes tapahtuu keskeytys. Tähän ohjelman pätkään on kytketty lambda-anturi, molemmat lämpötilavastukset ja K-tyypin termistori. Kaikille on varattu oma ADC-pinni, jolta ne ottavat tiedon vastaan. Kaikki tieto pinnille tulee välillä nolla ja viisi volttia, joten ohjelman ei tarvitse kuin muuttaa sisään tullut luku oikeaksi luvuksi. Esimerkiksi lämpötilaa varten on taulukko, josta ohjelma saa saadun luvun mukaan lukua vastaavan lämpötilan celsiusasteina. Todennäköisesti tekemäni taulukkoa joutuu muuttamaan, kunhan pääsen testaamaan korttia, jotta saadaan lämpötilat näkymään oikein.

4.1.2 Toinen tapa ottaa tietoa antureilta

Keskeytyksiä aiheuttavia antureita on kolme: optinen anturi, induktiivinen kytkin ja induktiivinen silmukka. Jokaiselle niistä on oma keskeytyspinni, ja jokainen niistä laukaisee oman keskeytysohjelman.

Optinen anturi ja induktiivisen silmukan ohjelma

Optinen anturi ja induktiivinen silmukka laukaisevat keskeytysohjelmat, joita käytetään laskemaan sytytyksen ennakko. Induktiivisen silmukan ohjelma nollaa ja käynnistää laskurin. Optisen anturin ohjelma pysäyttää laskurin ja vähentää laskurin ajasta vakiomittaisen viiveen, joka haittaisi sytytyksen ennakkoa laskettaessa.

Induktiivisen kytkimen ohjelma

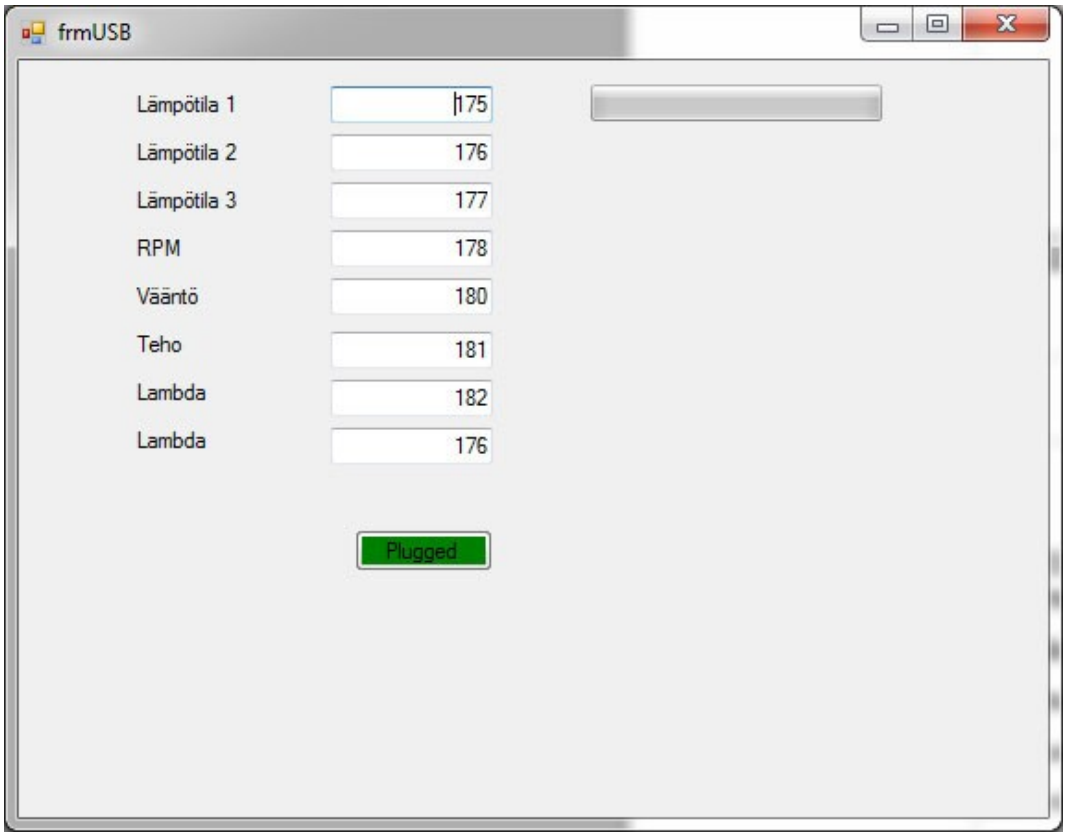
Induktiivinen kytkin laukaisee ohjelman, joka muuttaa venymäliuskan syöttämän analogisen jännitteen digitaaliseksi numeroksi ADC-pinniltä ja laittaa numeron talteen tietokoneelle lähetystä varten. Jos induktiiviselta kytkimeltä ei tule signaalia, niin venymäliuskalta tullut numero pysyy samana ja kortti lähettää samaa numeroa tietokoneelle uudestaan ja uudestaan.

4.2 Tietokoneen ohjelma

Tietokoneelle tulevan ohjelman ei tarvitse kuin näyttää kortin antamia lukuja ja tallentaa niitä. Teho, kolme lämpötilaa, kierrosluku, sytytyksen ennakko ovat kortilta tulevia lukuja. Testausta varten ohjelma ei tallenna mitään tietoja. Se vai näyttää ennalta määrättyjä lukuja. Loin ohjelman Microsoft Visual Basic 2010:llä, jolla sain tehtyä selkeän näköisen ohjelman. Liitteenä oleva tietokoneella olevan ohjelman koodi sivulla kaksi näkyy kuinka tietokoneen ohjelma ottaa vastaan tietoa kortilta ja näyttää sen käyttäjälle.

Ohjelman pohjana käytin USB-hid templatea, jolla sain luotua ohjelmalle datan keräyksen kortilta melkein automaattisesti. Ohjelmoinnin aloitusvaiheessa minulla oli monta muutakin vaihtoehtoa, mutta tämä oli niistä kaikista yksinkertaisin ja varsin riittävä tähän projektiin. (USB HID Template for Visual Basic 2012.)

Ohjelmassa ei tällä hetkellä ole tietojen tallennusta ja ulkoasua tulen muuttamaan Petteri Valtosen toiveiden mukaisesti. Ohjelmassa pitäisi olla esimerkiksi kierroslukumittari, mutta en ole löytänyt ilmaisia mittareita, joita voisin käyttää ohjelmassa. Saatan ehkä ohjelmoida sellaisen itse myöhemmin, mutta tällä hetkellä riittää, että ohjelma näyttää kaiken datan numeroina. (Kuvio 16.)



KUVIO 16. Käytetty testiohjelma

5 TESTAUS

Valitettavasti en ole ollut aikaa testata korttia paikan päällä, mutta testiohjelmistolla sain lähetettyä ennalta määrättyjä tietoja kortilta tietokoneelle ja ne tulivat läpi odotetusti. Testiohjelma kierrättää dataa välillä nolla ja 255 ja lähettää sen tietokoneelle. Tietokoneen puolella testiohjelma ottaa datan vastaan ja näyttää sen tekstiruuduissa. Ohjelma myös näyttää, milloin kortti on kytkettynä kiinni USB-porttiin. Näytössä on vihreä ”Plugged”-ruutu, kun laite on USB-portissa, ja punainen ”Unplugged”-ruutu, kun se ei ole kiinni. Kortti antoi aluksi informaatiota tietokoneelle niin nopeasti, että paljaalla silmällä ei ehtinyt nähdä numeroita ollenkaan. Laitoin kuitenkin viiveen datan vaihtuvuudelle kortilla ja sain dataa ulos paljon paremmin. Liitteenä olevassa kortin koodissa sivulla kolme näkyy kuinka tein sain aikaiseksi viiveen ja kuinka testaus data muuttuu.

Tarkoituksena on lähteä testaamaan Petteri Valtosen luo korttia, kunhan saan ina125:set. Ohjelmaan tietenkin tulee silloin datan keräysohjelmat, mutta datan lähetysosuus säilyy samanlaisena.

6 YHTEENVETO

Koko projekti eteni mielestäni varsin hyvin. Pieniä ongelmia lukuunottamatta kaikki sujui ilman isompia vastoinkäymisiä. Projektin osalta on montakin asiaa, jotka tekisin eri tavalla, jos pitäisi projekti tehdä uudemman kerran.

Mikro-ohjaimen valinnassa tein virheen alkuun valitessani Microchip Technology Inc:n ohjaimen PIC24FJ64GB004, joka datalehtiä ensimmäisen kerran lukiessani näytti sopivalta mikro-ohjaimelta projektiin. Myöhemmin tarkistaessani datalehteä huomasin, että mikro-ohjain ei kävisi ilman melkoisia muutoksia antureiden vastaanottoon. Onneksi olin vielä suunnitteluvaiheessa, joten sain vaihdettua mikro-ohjaimen. ATMEL AT90USB1287-AU –mikro-ohjain toimii tavoitteiden mukaisesti, mutta se on hieman ylimitoitettu projektille. Tämän tilalle vaihtaisin hieman pienemmän mikro-ohjaimen.

Potentiometrien liitosalueet ovat väärät, mikä ei ole kovin suuri ongelma, mutta jos pitäisi tehdä kortti uudestaan, vaihtaisin näille uudet liitosalueet tai tilaisin liitosalueisiin sopivat potentiometrit. Tekisin myös kortille pari LED-valoa näyttämään, missä tilassa mikro-ohjain on, koska nyt ei voi millään tietää, onko mikro-ohjain jäänyt johonkin tilaan vai onko se muuten vain kiireinen. Todennäköisesti käyttäisin mikro-ohjaimella vahtikoiratoimintoa, joka sytyttäisi ledin palamaan, mikäli vahtikoira-ajastinta ei nollattaisi tietyin väliajoin.

Eniten uutta tietoa sain USB-ohjelmoinnista, niin mikrokontrollerin kuin tietokoneen ohjelmoinninkin puolelta. Myöhemmin aikomukseni on lisätä tietokoneen ohjelmaan ainakin kierroslukumittari.

LÄHTEET

AVR Microcontroller with 64/128K Bytes of ISP Flash and USB Controller.2012. [viitattu 5.2.2012]. Saatavissa: http://www.atmel.com/dyn/resources/prod_documents/7593S.pdf.

BatchPCB.com.2011. [viitattu 5.2.2011]. Saatavissa: http://batchpcb.com.

Contrast sensor with white emission.2011. [viitattu 5.2.2011]. Saatavissa: http://www.getec.cl/pdf/sensores/fotoelectricos/S50/S50b_w_e.pdf.

Instrumentation amplifier.2011. [viitattu 5.2.2011]. Saatavissa: https://www1.elfa.se/data1/wwwroot/assets/datasheets/07338114.pdf.

Lambda-anturi.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Lambda-anturi.

LED.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/LED.

LUFA.2012. [viitattu 5.2.2012]. Saatavissa: http://www.fourwalledcubicle.com/LUFA.php.

Microminiature Slide Switches.2011. [viitattu 5.2.2011]. Saatavissa: http://www.farnell.com/datasheets/523992.pdf.

Mikrokontrolleri.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Mikrokontrolleri.

Oskillaattori.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Oskillaattori.

Potentiometri.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Potentiometri.

Quad SPST CMOS Analago Switches.2011. [viitattu 5.2.2011]. Saatavissa: http://www.farnell.com/datasheets/7343.pdf.

Right Angle Illuminated Latching Pushbutton.2012. [viitattu 14.2.2012]. Saatavissa: http://www.farnell.com/datasheets/85772.pdf.

Termopari.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Termopari.

USB HID Template for Visual Basic.2012. [viitattu 5.2.2012]. Saatavissa: http://helmpcb.com/software/usb-hid-template-for-visual-basic-2005.

Vahvistin.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Vahvistin.

Venymäliuska-anturi.Wikipedia.2012. [viitattu 14.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Venym%C3%A4liuska-anturi.

WinAVR.2012. [viitattu 5.2.2012]. Saatavissa: winavr.sourceforge.net/.

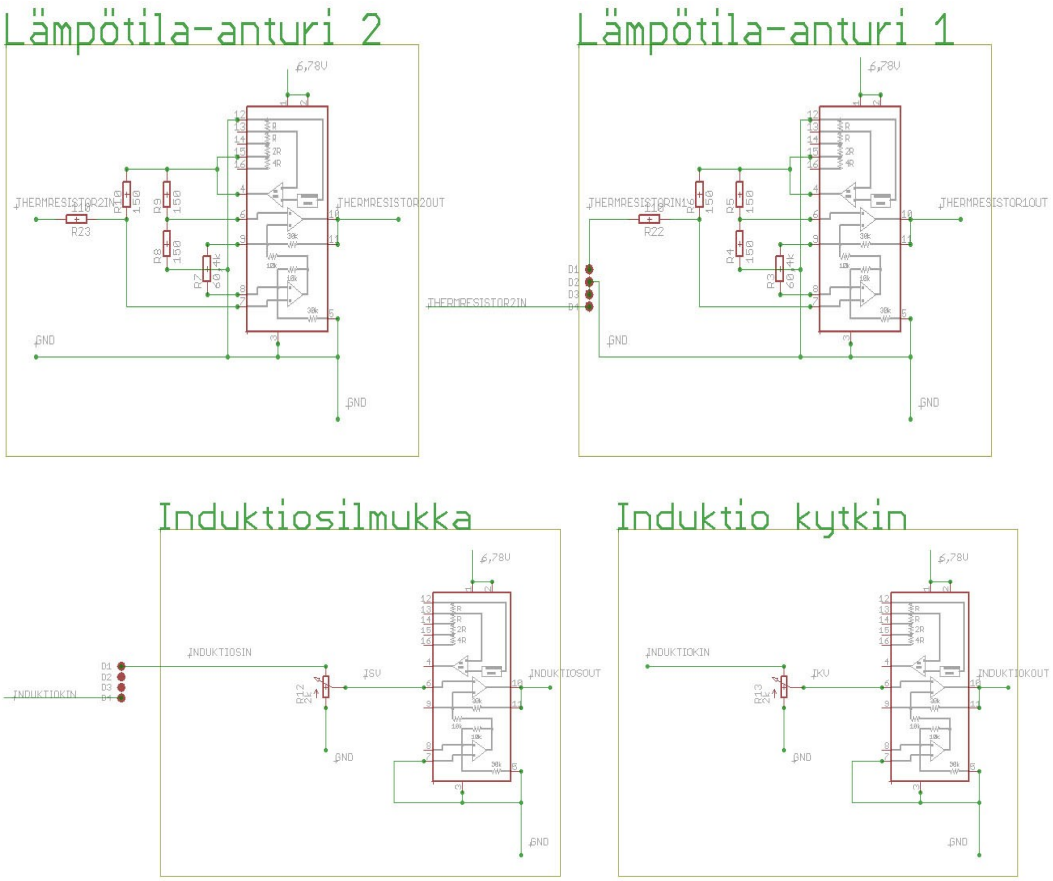
LIITTEET

Liite 1: Kytkentäkaavio

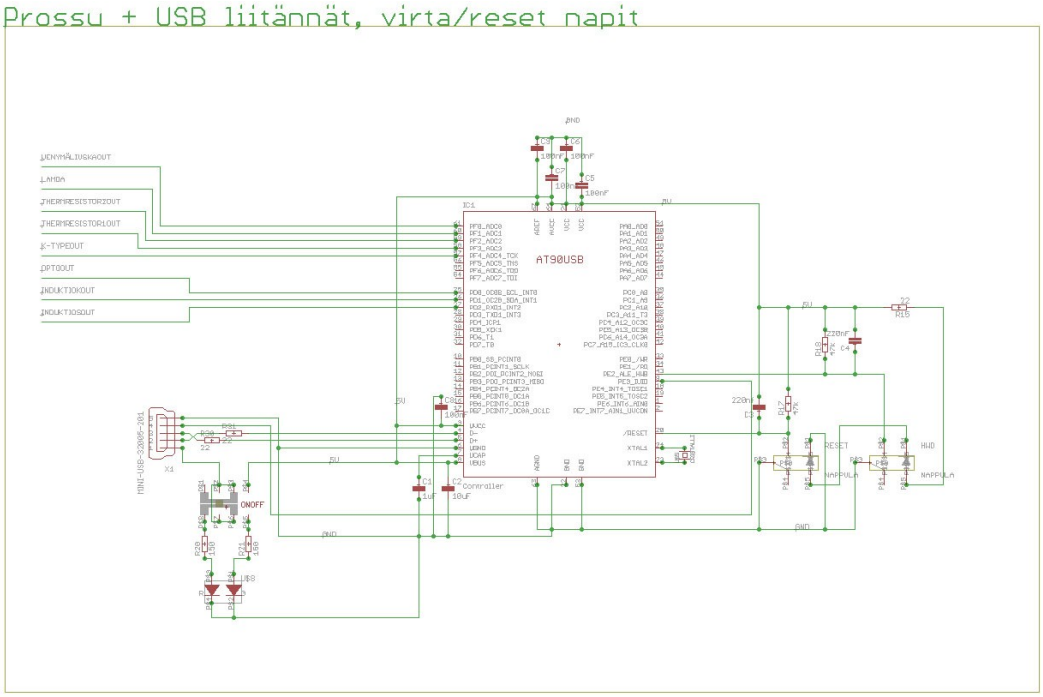
Liite 2: Testauskoodi kortille

Liite 3: Testauskoodi tietokoneelle

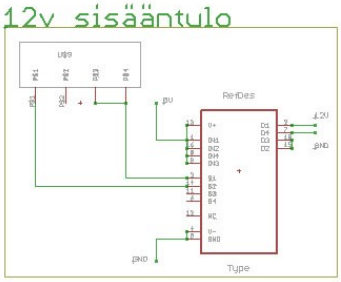
Kytkentäkaavio



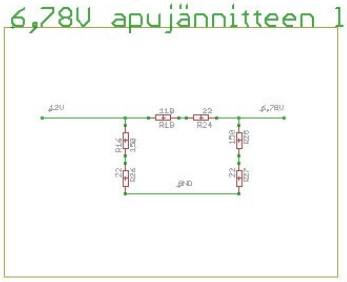
Prossu + USB liitännät, virta/reset napit



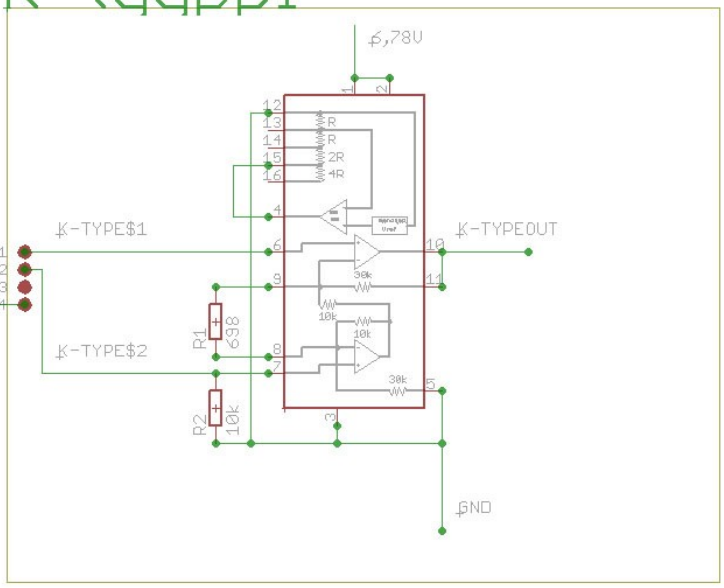
12v sisääntulo



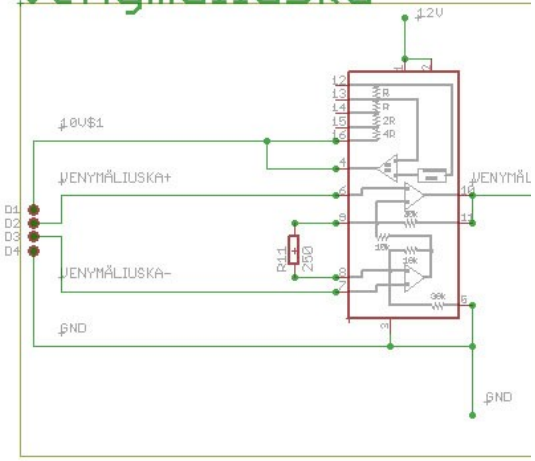
6,78V apujännitteen luonti ina125sille



K-tyyppi

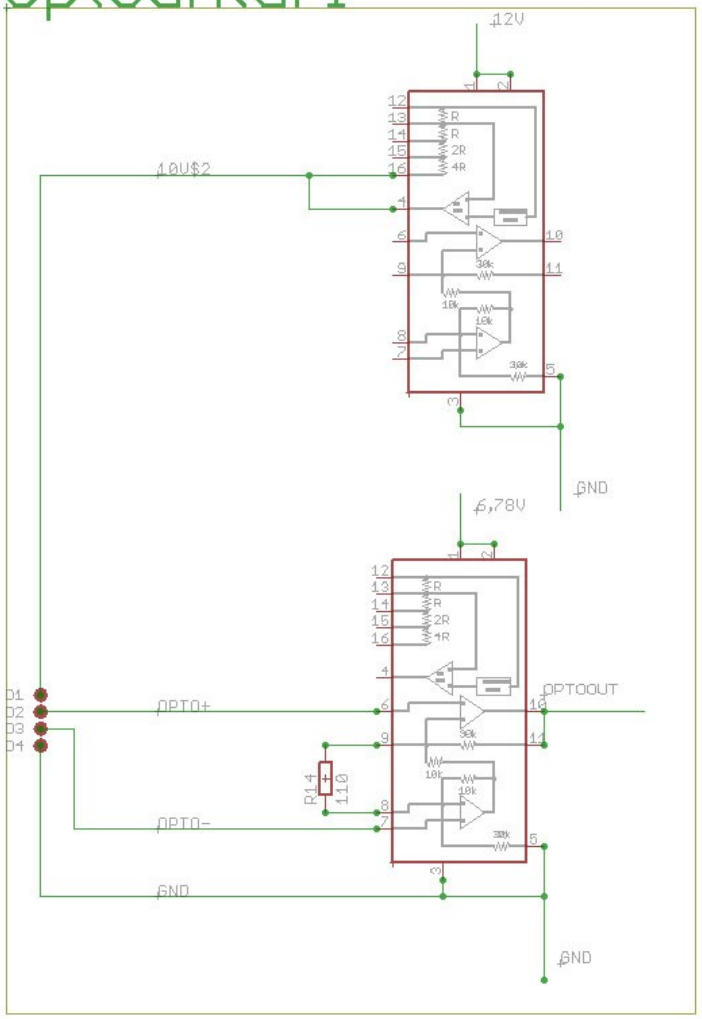


Venymäliuska



Lamda

Optoanturi



Testauskoodi kortille

```
/*
             LUFA Library
     Copyright (C) Dean Camera, 2012.

  dean [at] fourwalledcubicle [dot] com
           www.lufa-lib.org
*/

/*
  Copyright 2012  Dean Camera (dean [at] fourwalledcubicle [dot] com)

  Permission to use, copy, modify, distribute, and sell this
  software and its documentation for any purpose is hereby granted
  without fee, provided that the above copyright notice appear in
  all copies and that both that the copyright notice and this
  permission notice and warranty disclaimer appear in supporting
  documentation, and that the name of the author not be used in
  advertising or publicity pertaining to distribution of the
  software without specific, written prior permission.

  The author disclaim all warranties with regard to this
  software, including all implied warranties of merchantability
  and fitness.  In no event shall the author be liable for any
  special, indirect or consequential damages or any damages
  whatsoever resulting from loss of use, data or profits, whether
  in an action of contract, negligence or other tortious action,
  arising out of or in connection with the use or performance of
  this software.
*/

/** \file
 *
 *  Main source file for the GenericHID demo. This file contains the main
tasks of
 *  the demo and is responsible for the initial application hardware
configuration.
 */

#include "GenericHID.h"

/** Buffer to hold the previously generated HID report, for comparison
purposes inside the HID class driver. */
static uint8_t PrevHIDReportBuffer[GENERIC_REPORT_SIZE];

/** LUFA HID Class driver interface configuration and state information.
This structure is
 *  passed to all HID Class driver functions, so that multiple instances of
the same class
 *  within a device can be differentiated from one another.
 */
USB_ClassInfo_HID_Device_t Generic_HID_Interface =
            {
                        .Config =
                                    {
                                                .InterfaceNumber
= 0,

                                                .ReportINEndpointNumber
= GENERIC_IN_EPNUM,
                                                .ReportINEndpointSize
= GENERIC_EPSIZE,
                                                .ReportINEndpointDoubleBan
k    = false,
```

```
                                                .PrevReportINBuffer
= PrevHIDReportBuffer,
                                                .PrevReportINBufferSize
= sizeof(PrevHIDReportBuffer),
                                        },
            };


/** Main program entry point. This routine contains the overall program
flow, including initial
 *  setup of all components and the main program loop.
 */
            int i,viive;
            int tieto[7] = {0,1,2,3,4,5,6,7};

int main(void)
{
            SetupHardware();
            LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);
            sei();



            for (;;)
            {

                        HID_Device_USBTask(&Generic_HID_Interface);
                        USB_USBTask();
            }
}

/** Configures the board hardware and chip peripherals for the demo's
functionality. */
void SetupHardware(void)
{
            /* Disable watchdog if enabled by bootloader/fuses */
            MCUSR &= ~(1 << WDRF);
            wdt_disable();

            /* Disable clock division */
            clock_prescale_set(clock_div_1);

            /* Hardware Initialization */
            LEDs_Init();
            USB_Init();
}

/** Event handler for the library USB Connection event. */
void EVENT_USB_Device_Connect(void)
{
            LEDs_SetAllLEDs(LEDMASK_USB_ENUMERATING);
}

/** Event handler for the library USB Disconnection event. */
void EVENT_USB_Device_Disconnect(void)
{
            LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);
}

/** Event handler for the library USB Configuration Changed event. */
void EVENT_USB_Device_ConfigurationChanged(void)
{
            bool ConfigSuccess = true;
```

```
            ConfigSuccess &=
HID_Device_ConfigureEndpoints(&Generic_HID_Interface);

            USB_Device_EnableSOFEvents();

            LEDs_SetAllLEDs(ConfigSuccess ? LEDMASK_USB_READY :
LEDMASK_USB_ERROR);
}

/** Event handler for the library USB Control Request reception event. */
void EVENT_USB_Device_ControlRequest(void)
{
            HID_Device_ProcessControlRequest(&Generic_HID_Interface);
}

/** Event handler for the USB device Start Of Frame event. */
void EVENT_USB_Device_StartOfFrame(void)
{
            HID_Device_MillisecondElapsed(&Generic_HID_Interface);
}

/** HID class driver callback function for the creation of HID reports to
the host.
 *
 *  \param[in]     HIDInterfaceInfo  Pointer to the HID class interface
configuration structure being referenced
 *  \param[in,out] ReportID    Report ID requested by the host if non-zero,
otherwise callback should set to the generated report ID
 *  \param[in]     ReportType  Type of the report to create, either
HID_REPORT_ITEM_In or HID_REPORT_ITEM_Feature
 *  \param[out]    ReportData  Pointer to a buffer where the created report
should be stored
 *  \param[out]    ReportSize  Number of bytes written in the report (or
zero if no report is to be sent)
 *
 *  \return Boolean true to force the sending of the report, false to let
the library determine if it needs to be sent
 */
bool CALLBACK_HID_Device_CreateHIDReport(USB_ClassInfo_HID_Device_t* const
HIDInterfaceInfo,
                                         uint8_t* const ReportID,
                                         const uint8_t ReportType,
                                         void* ReportData,
                                         uint16_t* const ReportSize)
{
            uint8_t* Data = (uint8_t*)ReportData;
            uint8_t  CurrLEDMask = LEDs_GetLEDs();

            if (viive>100) {
            Data[0] = tieto[0]++;
            Data[1] = tieto[1]++;
            Data[2] = tieto[2]++;
            Data[3] = tieto[3]++;
            Data[4] = tieto[4]++;
            Data[5] = tieto[5]++;
            Data[6] = tieto[6]++;
            Data[7] = tieto[7]++;
            viive = 0;
            }
            else {
            Data[0] = tieto[0];
            Data[1] = tieto[1];
            Data[2] = tieto[2];
            Data[3] = tieto[3];
            Data[4] = tieto[4];
```

```
            Data[5] = tieto[5];
            Data[6] = tieto[6];
            Data[7] = tieto[7];
            viive = viive++;
            }

            *ReportSize = GENERIC_REPORT_SIZE;
            return true;
}

/** HID class driver callback function for the processing of HID reports
from the host.
 *
 *  \param[in] HIDInterfaceInfo  Pointer to the HID class interface
configuration structure being referenced
 *  \param[in] ReportID    Report ID of the received report from the host
 *  \param[in] ReportType  The type of report that the host has sent,
either HID_REPORT_ITEM_Out or HID_REPORT_ITEM_Feature
 *  \param[in] ReportData  Pointer to a buffer where the received report
has been stored
 *  \param[in] ReportSize  Size in bytes of the received HID report
 */
void CALLBACK_HID_Device_ProcessHIDReport(USB_ClassInfo_HID_Device_t* const
HIDInterfaceInfo,
                                          const uint8_t ReportID,
                                          const uint8_t ReportType,
                                          const void* ReportData,
                                          const uint16_t ReportSize)
{
            uint8_t* Data = (uint8_t*)ReportData;
            uint8_t  NewLEDMask = LEDS_NO_LEDS;

            if (Data[0])
              NewLEDMask |= LEDS_LED1;

            if (Data[1])
              NewLEDMask |= LEDS_LED2;

            if (Data[2])
              NewLEDMask |= LEDS_LED3;

            if (Data[3])
              NewLEDMask |= LEDS_LED4;

            LEDs_SetAllLEDs(NewLEDMask);

}
```

Testauskoodi tietokoneelle

```
Public Class frmUSB
    ' vendor and product IDs
    Private Const VendorID As Integer = &H3EB    'Replace with your
device's
    Private Const ProductID As Integer = &H204F      'product and vendor
IDs

    ' read and write buffers
    Private Const BufferInSize As Integer = 8 'Size of the data buffer
coming IN to the PC
    Private Const BufferOutSize As Integer = 8    'Size of the data buffer
going OUT from the PC
    Dim BufferIn(BufferInSize) As Byte          'Received data will be
stored here - the first byte in the array is unused
    Dim BufferOut(BufferOutSize) As Byte    'Transmitted data is stored
here - the first item in the array must be 0

    ' ****************************************************************
    ' when the form loads, connect to the HID controller - pass
    ' the form window handle so that you can receive notification
    ' events...
    '****************************************************************
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' do not remove!
        ConnectToHID(Me)
    End Sub

    '****************************************************************
    ' disconnect from the HID controller...
    '****************************************************************
    Private Sub Form1_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        DisconnectFromHID()
        T1.Text = "Unplugged"
    End Sub

    '****************************************************************
    ' a HID device has been plugged in...
    '****************************************************************
    Public Sub OnPlugged(ByVal pHandle As Integer)
        If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
= ProductID Then
            ' ** YOUR CODE HERE **
            Button1.Text = "Plugged"
            Button1.BackColor = Color.Green
        End If
    End Sub

    '****************************************************************
    ' a HID device has been unplugged...
    '****************************************************************
    Public Sub OnUnplugged(ByVal pHandle As Integer)
        If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
= ProductID Then
            hidSetReadNotify(hidGetHandle(VendorID, ProductID), False)
            ' ** YOUR CODE HERE **
            Button1.Text = "Unplugged"
            Button1.BackColor = Color.Red
        End If
    End Sub
```

```
    '*****************************************************************
    ' controller changed notification - called
    ' after ALL HID devices are plugged or unplugged
    '*****************************************************************
    Public Sub OnChanged()
        ' get the handle of the device we are interested in, then set
        ' its read notify flag to true - this ensures you get a read
        ' notification message when there is some data to read...
        Dim pHandle As Integer
        pHandle = hidGetHandle(VendorID, ProductID)
        hidSetReadNotify(hidGetHandle(VendorID, ProductID), True)
    End Sub

    '*****************************************************************
    ' on read event...
    '*****************************************************************
    Public Sub OnRead(ByVal pHandle As Integer)
        ' read the data (don't forget, pass the whole array)...
        If hidRead(pHandle, BufferIn(0)) Then
            ' ** YOUR CODE HERE **
            ' first byte is the report ID, e.g. BufferIn(0)
            ' the other bytes are the data from the microcontroller...
            T1.Text = BufferIn(1).ToString
            T2.Text = BufferIn(2).ToString
            T3.Text = BufferIn(3).ToString
            RPM.Text = BufferIn(4).ToString
            Vaanto.Text = BufferIn(5).ToString
            Teho.Text = BufferIn(6).ToString
            Lambda.Text = BufferIn(7).ToString
            Viive.Text = BufferIn(8).ToString
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        T3.Text = hidRead(BufferIn(2), 123)

        RPM.Text = BufferIn(2)
    End Sub
End Class
```